



第5章 STC单片机汇编语言 编程模型

何宾

2015.02

本章主要内容

- 汇编语言程序结构
- 汇编代码中段的分配
- 汇编语言符号及规则
- 汇编语言操作数描述
- 汇编语言控制描述
- Keil μ Vision集成开发环境介绍
- Keil μ Vision5汇编语言设计流程
- 单片机端口控制汇编语言程序设计
- 单片机中断汇编语言程序设计

汇编语言程序结构

实际上，所谓的汇编语言程序就是按照一定的规则组合在一起的机器语言助记符和汇编器助记符命令。这些按一定规则组合在一起的汇编语言助记符机器指令，能通过软件开发环境的处理，转换成可以在STC 8051 CPU上按照设计要求运行的机器代码。

汇编语言程序结构

NAME main	; 声明模块
my_seg SEGMENT CODE	; 声明代码段my_seg
RSEG my_seg	; 切换到代码段my_seg
TABLE: DB 3,2,5,0xFF	; 声明四个常数
myprog SEGMENT CODE	; 声明代码段myprog
RSEG myprog	; 切换到代码段myprog
LJMP main	; 在程序存储器地址0x0000的位置跳转
USING 0	; 使用第0组寄存器
ORG 100H	; 定位到代码段100H的位置

汇编语言程序结构

```
main:  MOV  DPTR, #TABLE      ; 将TABLE表的地址送
                                   ; 给DPTR寄存器
      MOV  A,#3              ; 将立即数3送到累加器A中
      MOVC A,@A+DPTR         ; 将(A)+(DPTR)所指向的程序
                                   ; Flash的内容送给累加器A
      MOV  P1,0              ; 给P1端口清零
      MOV  P1,A              ; 将累加器A的内容送到P1端口
      END
```

注：进入本书配套资料\STC_example\例子5-1\目录下，打开并参考该设计

汇编代码中段的分配

在一个由汇编语言所构建的程序代码中，包括：

- 绝大部分代码都是机器语言助记符。这些程序代码中的机器语言助记符经过软件处理后变成机器指令（机器码），然后保存在程序存储器中。通常地，将保存程序代码的区域称为代码段。
- 根据程序设计的复杂度，需要提供代码中所需数据所在的位置。这些需要操作的数可能保存在不同的存储空间中。因此，就需要在程序中明确的说明这些数据所存放的位置。

□ 在STC 15系列单片机中，提供了片内基本RAM、片内扩展RAM等存储单元。通常地，将保存数据的区域统称为数据段。

注：更具体的划分，会根据不同的CPU的结构有所不同。

汇编代码中段的分配

- 在运行程序的过程中，可能还需要对8051 CPU内功能部件的状态进行保存和恢复操作。前面已经说明，这是由堆栈机制实现的。在这种情况下，就需要明确指出保存这些运行状态的存储空间大小和位置。

注：在任何一个程序中，必须要有代码段，而其它段的存在与否，由具体的程序模型决定。

汇编代码中段的分配

--CODE段

CODE段，也称为代码段，它是用来保存程序中汇编助记符描述的机器指令部分。

- **CODE放在STC单片机中的程序Flash存储空间。**
- **CODE段可以由MOVC指令，并且通过DPTR寄存器进行访问。**

汇编代码中段的分配

--CODE段

代码清单 定义和访问CODE段的代码

```
my_seg SEGMENT CODE ; 定义为CODE段
        RSEG  my_seg
TABLE:   DB    1,2,4,8,0x10 ; 定义常数表
myprog SEGMENT CODE ; 定义CODE段
        RSEG  myprog
        MOV   DPTR, #TABLE ; 加载TABLE的地址
        MOV   A, #3 ; 加载偏移量
        MOVC  A, @A+DPTR ; 通过MOVC指令访问
END
```

汇编代码中段的分配

--BIT段

在8051汇编语言中,BIT段可以用来保存比特位 , 可以通过指令系统中的位操作指令来访问BIT段。

注：可以通过位操作指令访问特殊功能寄存器SFR。

■ 可位寻址的地址只能是可以被8整除的地址。

- 典型地 , 包括 : 80H , 88H , 90H , 98H , 0A0H , , 0A8H , 0B0H , 0B8H , 0C0H , 0C8H , 0D0H , 0D8H , 0E0H , 0E8H , 0F0H和0F8H地址空间。

汇编代码中段的分配

--BIT段

代码清单 定义和访问BIT段的代码

```
mybits SEGMENT BIT ; 定义BIT段
        RSEG  mybits
FLAG:    DBIT  1 ; 保留1位空间
P1       DATA  90H ; 8051 SFR 端口1
GREEN_LED BIT  P1.2 ; 在端口P1的第2引脚P1.2定义
                    符号GREEN_LED
myprog SEGMENT CODE ; 定义为代码段
        RSEG  myprog
        LJMP  main ;无条件跳转到main
        ORG  100H ;定位到100H的位置
```

汇编代码中段的分配

--BIT段

```
main:   SETB   GREEN_LED           ; P1.2 = 1
        JB     FLAG, is_on         ; 到DATA的直接访问
SETB    FLAG
        CLR    ACC.5               ; 复位ACC的第5位
        :
is_on:   CLR    FLAG
        CLR    GREEN_LED           ; P1.2 = 0
END
```

注：进入本书配套资料\STC_example\例子5-2\目录下，打开并参考该设计

汇编代码中段的分配

--IDATA段

在8051汇编语言中，可以定义IDATA段。

- 在IDATA段可以定义少量的变量，这些变量将最终保存在STC单片机的片内RAM的高地址和低地址区域中。

注：IDATA的低128个字节和DATA段重叠。

- 可以通过寄存器R0或者R1，间接寻址保存在IDATA段中的变量。

汇编代码中段的分配

--IDATA段

代码清单 定义和访问IDATA段的代码

```
myvars SEGMENT IDATA          ; 定义IDATA段
    RSEG  myvars
BUFFER:  DS    100            ; 保留100个字节
myprog SEGMENT CODE           ; 定义CODE段
    RSEG  myprog
    LJMP  main                ; 无条件跳转到main
    ORG   100H                ; 定位到100H的位置
main:    MOV   R0,#BUFFER      ; 将BUFFER的地址加载到R0寄存器
        MOV   A,@R0           ; 读缓冲区的内容到寄存器A
        INC   R0              ; R0内的地址递增
        MOV   @R0,A           ; 将A内容写到BUFFER+1存储空间
END
```

汇编代码中段的分配

--DATA段

在8051汇编语言中，定义了DATA段，该段指向STC单片机内部数据RAM的低128个字节。

- 通过直接和间接寻址方式，程序代码可以访问在DATA段中的变量。

汇编代码中段的分配

--DATA段

代码清单 定义和访问DATA段的代码

myvar SEGMENT DATA

; 定义DATA段

RSEG myvar

VALUE: DS 1

; 在DATA空间保存一个字节

IO_PORT2 DATA 0A0H

; 特殊功能寄存器

VALUE2 DATA 20H

; 存储器的绝对地址

myprog SEGMENT CODE

; 定义CODE段

RSEG myprog

LJMP main

; 无条件跳转到main

ORG 100H

; 定位到100H的位置

汇编代码中段的分配

--DATA段

```
main:  MOV    A,IO_PORT2           ; 直接访问DATA
        ADD    A,VALUE
        MOV    VALUE2,A
        MOV    R1,#VALUE          ; 加载VALUE 的值到R1
        ADD    A,@R1              ; 间接访问VALUE
END
```

注：进入本书配套资料\STC_example\例子5-4\目录下，打开并参考该设计

汇编代码中段的分配

--XDATA段

在8051汇编语言中，定义了XDATA段，XDATA段指向扩展RAM区域。

- 通过寄存器DPTR和MOVX指令，程序代码就可以访问XDATA段。
- 对于一个单页的XDATA存储空间来说，也可以通过寄存器R0和R1访问。

汇编代码中段的分配

--XDATA段

代码清单 定义和访问XDATA段的代码

<code>my_seg SEGMENT XDATA</code>	<code>; 定义XDATA段</code>
<code> RSEG my_seg</code>	
<code>XBUFFER: DS 2</code>	<code>; 保留2个字节存储空间</code>
<code>myprog SEGMENT CODE</code>	<code>; 定义CODE段</code>
<code> RSEG myprog</code>	
<code> LJMP main</code>	<code>; 无条件跳转到main</code>
<code> ORG 100H</code>	<code>; 定位到100H的位置</code>

汇编代码中段的分配

--XDATA段

```
main:  MOV    DPTR,#XBUFFER    ; XBUFFER的地址送到DPTR寄存器
      CLR    A                ; 累加器A清0
      MOVX   @DPTR,A          ; 将累加器A的内容送给DPTR指向的
                                ; XBUFFER区域
      INC    DPTR              ; 寄存器DPTR的内容加1
      CLR    A                ; 累加器A清0
      MOVX   @DPTR,A          ; 累加器A的内容送给DPTR指向的
                                ; XBUFFER区域
      END
```

注：进入本书配套资料\STC_example\例子5-5\目录下，打开并参考该设计

汇编语言符号及规则

--符号的命名规则

在AX51汇编器中，符号最多可以由31个字符组成。符号中的字符可以包括：

- A~Z之间的大写字母；
- a~z之间的小写字母；
- 0~9之间的数字；
- 空格字符
- 问号字符.

注：数字不可以作为符号的开头；

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

汇编语言符号及规则

--符号的作用

在汇编语言中，符号的作用包括：

- 使用EQU或者SET控制描述，将一个数值或者寄存器名赋给一个指定的符号名，例如：

NUMBER_FIVE	EQU	5
TRUE_FLAG	SET	1
FALSE_FLAG	SET	0

汇编语言符号及规则

--符号的作用

■ 在汇编程序中，符号可以用来表示一个标号。其中：

- 标号用于在程序或者数据空间内定义一个位置（地址）。
- 标号是该一行的第一个字符域。
- 标号后面必须跟着 ‘:’ 符号。一行只能定义一个标号，例如：

```
LABEL1:    DJNZ  R0, LABEL1
```

■ 在汇编程序中，符号可以用于表示一个变量的位置，例如：

```
SERIAL_BUFFER  DATA  99h
```

汇编语言操作数描述

--数字

数字以十六进制数、十进制数、八进制数和二进制数的形式指定。如果没有指定数字的形式，默认为十进制数。
对于：

- 十六进制数：后缀H，h，有效数字在0~9，A~F或a~f之间

- 例如：0FH，0FFH；

注：(1)当其第一个数字在A~F之间时，必须加前缀‘0’。

(2)十六进制数也可使用C语言的表示方法，例如：0x12AB。

汇编语言操作数描述

--数字

■ 十进制数：后缀D，d（可无后缀），有效数字在0~9之间。

□ 例如：1234，20d；

■ 八进制数：后缀O，o，有效数字在0~7之间。

□ 例如：25o，65O；

■ 二进制数：后缀B，b，有效数字0和1。

□ 例如：111b，10100011B；

注：可以在数字之间插入符号‘\$’，用于增加数字的可读性，比如：
1\$2\$3\$4等效于1234。

汇编语言操作数描述

--字符

在表达式中可以使用ASCII字符来生成数字值。表达式可以由单引号包含的两个ASCII字符组成。

注：字符个数不能超过两个，否则在对汇编程序处理的过程中会报错。

- 在汇编语言的任何地方都可以使用字符，它可以用来作为立即数。

- 例如： 'A' 表示0041h， 'a' 表示0061h。

汇编语言操作数描述

--字符串

字符串与汇编器描述符DB一起使用，用来定义在AX51汇编程序中的消息。

■ 字符串用一对单引号' '包含。

□ 比如：KEYMSG: DB 'Press any key to continue.'

该声明将在KEYMSG指向的缓冲区内生成下面的十六进制数，即：50h、72h、65h、73h、73h、20h、...、6Eh、75h、65h、2Eh。

汇编语言操作数描述

--位置计数器

在AX51汇编器中，为每个段保留了一个位置计数器。在这个计数器中，包含了指令或者数据的偏移地址。

- 默认地，将位置计数器初始化为0。
- 可以用ORG描述修改位置计数器的初值。
 - 在表达式中，使用' \$ ' 符号，用于得到位置计数器当前的值，例如：
使用位置计数器确定一个字符串的长度。比如：

```
msg:    DB    'This is a message', 0
```

```
msg_len: EQU   $-msg
```

汇编语言操作数描述

--操作符

操作符及优先级

优先级	操作符
1	()
2	(1) NOT、HIGH、LOW 、 (2) BYTE0、BYTE1、BYTE2、BYTE3、 (3) WORD0、WORD2、MBYTE
3	一元+、一元-
4	*, /、MOD
5	+, -
6	SHL、SHR
7	AND、OR、XOR
8	EQ、=、NE、<>、LT、<、LTE、<=、GT、>、GTE、>=

汇编语言操作数描述

--操作符

注：

(1)1级具有最高的优先级，8级优先级最低。

(2)SHL表示左移运算、SHR表示右移运算。

(3)BYTE x 根据 x 所指定操作数的位置，返回相应的字节。例如：BYTE0返回最低的字节（与LOW等效）；BYTE1返回紧挨BYTE0的字节（与HIGH等效），如下表所示。

(4)WORD x 根据 x 指定的操作数的位置，返回相应的字。例如：WORD1返回最低的字（16位）；WORD2返回最高的两个字节（16位）。

汇编语言操作数描述

--操作符

(5) MBYTE_x 操作符返回用于C51实时库的存储器类型信息。所得到的值是存储器类型字节。这些存储器类型字节用在C51实时库中用于访问带有"far"存储器类型定义的变量。

位操作数的分配

32位操作数			
MSB			LSB
BYTE3	BYTE2	BYTE1	BYTE0
WORD2		WORD1	
		HIGH	LOW

汇编语言操作数描述

--表达式

表达式是操作数和操作符的组合，该表达式由汇编器计算。

- 没有操作符的操作数是最简单的表达式。
- 表达式能用在操作数所要求的地方。

代码清单 表达式用法代码清单

EXTRN CODE (CLAB)	; CODE空间的入口
EXTRN DATA (DVAR)	; DATA空间的变量
MSK EQU 0F0H	; 定义符号来替换0xF0值
VALUE EQU MSK - 1	; 其它常数符号值
FOO SEGMENT CODE	
RSEG FOO	

汇编语言操作数描述

--表达式

LJMP ENTRY ;

ORG 100H

ENTRY: MOV A,#40H ; 用常数加载累加器

MOV R5,#VALUE ; 加载一个常数表示的符号值

MOV R3,#(0x20 AND MASK) ; 一个计算例子

MOV R7,#LOW (VALUE + 20H)

MOV R6,#1 OR (MSK SHL 4)

MOV R0,DVAR+20 ; DVAR地址加上20 , 加载R0寄存器

MOV R1,#LOW (CLAB+10) ; 加载CLAB地址加10的低部分到寄存器R1

MOV R5,80H ; 加载地址80H的内容到R5寄存器

SETB 20H.2 ; 设置20H.2

END

汇编语言控制描述

--地址控制

地址控制描述用于控制程序计数器PC的指向和寄存器组的选择，地址控制描述包括：

- **EVEN**：迫使位置计数器指向下一个偶数地址。例如：

```
MYDATA: SEGMENT DATA WORD
```

```
        RSEG  MYDATA
```

```
var1:   DSB    1
```

```
        EVEN
```

```
var2:   DSW    1
```

汇编语言控制描述

--地址控制

- **ORG** : 设置位置计数器指向一个指定的偏移量或地址。例如 :

ORG 100h

- **USING** : 说明使用哪个寄存器组。例如 :

USING 3 ; 选择第3组寄存器

PUSH R2 ; 将第3组中的R2寄存器入栈

汇编语言控制描述

--条件汇编

根据符号条件的真假，条件汇编控制模块的运行。条件汇编描述包括：

- IF：条件为真，汇编模块。
- ELSE：如果前面的IF条件为假，则汇编模块。
- ELSEIF：如果前面的IF和ELSEIF条件为假，则汇编模块。
- ENDIF：结束IF模块

汇编语言控制描述

--条件汇编

下面给出条件编译的例子：

```
IF (SWITCH = 1)
```

```
...
```

```
ELSEIF (SWITCH = 2)
```

```
...
```

```
ELSE
```

```
...
```

```
ENDIF
```

汇编语言控制描述

--存储器初始化

存储器初始化描述包括：

- DB：该描述符用于说明所分配空间的类型是字节。
 - 例如：TAB: DB 2, 3, 5, 7, 11, 13, 17, 19, ','
- DD：该描述符用于说明分配空间的类型是双字，即4个字节。
 - 例如：VALS: DD 12345678h, 98765432h
- DW：该描述符用于说明分配空间的类型是字，即2个字节。
 - 例如：HERE: DW 0

汇编语言控制描述

--分配存储器空间

分配存储器空间描述符，用于在存储器内为变量预先分配存储空间。分配存储器空间描述包括：

■ DBIT：该描述符用于说明为变量所分配存储空间类型为比特

□ 例如：A_FLAG: DBIT 1 ;保留的存储空间为1位

■ DS (DSB)：该描述符用于说明为变量所分配的存储空间类型为字节

□ 例如：TIME: DS 8 ;保留的存储空间为8个字节

汇编语言控制描述

--分配存储器空间

- DSD : 该描述符用于说明为变量所分配的存储空间类型为双字, 即4个字节
 - 例如 : COUNT: DSD 9 ;保留的存储空间为36个字节
- (4) DSW : 该描述符用于说明为变量所分配的存储空间类型为字, 即2个字节
 - 例如 : COUNT: DSW 9 ;保留的存储空间为18个字节

汇编语言控制描述

--过程声明

过程声明用于说明过程的开始和结束。过程声明描述主要包括：

- PROC：该描述符用于定义过程的开始。
- ENDP：该描述符用于定义过程的结束。
- LABEL：该描述符为符号名分配一个地址。标号后面可以跟一个“：”，或者不用。标号继承了当前活动代码的属性，因此不能在程序段之外使用。格式如下：

标号名： LABEL [类型]

汇编语言控制描述

--过程声明

过程声明的格式如下：

过程名字 PROC [类型]

 ;汇编助记符

 ;汇编助记符

 ;

过程名字 ENDP

汇编语言控制描述

--过程声明

其中：

类型说明用于规定所定义过程的类型，如下表所示。

过程的类型

类型	说明
无	默认为NEAR
NEAR	定义为一个NEAR类型的过程，采用LCALL或者ACALL指令调用
FAR	定义一个FAR类型过程，采用ECALL指令调用

汇编语言控制描述

--程序链接

程序链接主要用于控制模块之间参数的传递。控制描述包括：

■ EXTERN (EXTERN)

该控制描述符用于定义一个外部的符号。其格式为：

EXTERN 类：类型（符号1,符号2,...,符号N）

其中：

- 类：表示符号所在的存储器段的类型；
- 类型：BYTE（字节变量），DWORD（双字变量）、FAR（远标号）、NEAR（近标号）和WORD（字变量）。

汇编语言控制描述

--程序链接

例如：`EXTERN CODE: FAR (main)`

`EXTERN DATA: BYTE (counter)`

- **NAME**：该控制描述符用于指定当前模块的名字。
- **PUBLIC**：该控制描述符用于定义符号，用于说明其它模块会使用这些符号。

例如：`PUBLIC myvar,yourvar,othervar`

注：每个符号应该在当前的程序模块内定义过。寄存器和段符号不能指定为**PUBLIC**。

汇编语言控制描述

--段控制

段控制主要为段分配绝对地址或者可重定位描述。段控制描述包括：

■ BSEG：该控制符用于定义一个绝对BIT段，例如：

BSEG AT 10 ; 地址=0x20+10位 = 0x2A

DEC_FLAG: DBIT 1 ; DEC_FLAG为比特位类型

INC_FLAG: DBIT 1 ; INC_FLAG为比特位类型

汇编语言控制描述

--段控制

■ CSEG : 该控制符用于定义一个绝对CODE段, 例如 :

CSEG AT 0003h ; CODE段开始的绝对地址为0x3

VECT_0: LJMP ISR_0 ; 跳转到中断向量的位置

CSEG AT 0x100 ; 绝对地址0x100

CRight: DB "(C) MyCompany" ; 固定位置的字符串

CSEG AT 1000H ; 绝对地址0x1000

Parity_TAB: ; Parity_TAB的名字

DB 00H ; 初始化Parity_TAB开始的缓冲区

DB 01H

DB 01H

DB 00H

汇编语言控制描述

--段控制

- DSEG : 该控制符用于定义一个绝对DATA段 , 例如 :

DSEG AT 0x40 ; DATA段开始的绝对地址为40H

TMP_A: DS 2 ; TMP_A变量

TEM_B: DS 4 ; TMP_B变量

- ISEG : 该控制符用于定义一个绝对IDATA段 , 例如 :

ISEG AT 0xC0 ; IDATA段开始的绝对地址为0C0H

TMP_IA: DS 2 ; TMP_IA变量

TEM_IB: DS 4 ; TMP_IB变量

汇编语言控制描述

--段控制

- RSEG（段名字）：该控制符用于定义一个可重定位段，例如：

MYPROG SEGMENT CODE ; 定义一个段

RSEG MYPROG ; 选择段

- XSEG：该控制符用于定义一个绝对的XDATA段，例如：

XSEG AT 1000H ; XDATA段的绝对开始地址为0x1000

OEM_NAME: DS 25 ; OEM_NAME变量

PRD_NAME: DS 25 ; PRD_NAME变量

汇编语言控制描述

--杂项

杂项控制描述包含：

- ERROR：产生错误消息；
- END：表示汇编模块的结束；

Keil μ Vision集成开发环境介绍

--软件功能介绍

μ Vision5提供了功能强大的编辑器，并且提供了管理工程的能力。 μ Vision5集成了用于开发嵌入式应用的所有工具包括：C/C++编译器、宏汇编器、链接器/定位器和HEX文件生成器。 μ Vision5集成开发环境提供以下功能：

- 全功能的源代码编辑器。
- 用于配置开发工具的元件库。
- 用于创建和维护工程的工程管理器。

Keil μ Vision集成开发环境介绍

--软件功能介绍

- 提供用于对嵌入式设计文件进行处理的汇编器、编译器和链接器。
- 提供用于所有开发环境设置的对话框。
- 真正集成了带有高速CPU和外设仿真器的源码级和汇编器级调试器工具。
- 提供用于对目标硬件进行软件调试的高级图形调试（ Graphic Debug Interface , GDI ）接口，以及Keil ULINK调试适配器。

Keil μ Vision集成开发环境介绍

--软件功能介绍

- 提供了Flash编程工具，用于将应用程序下载到Flash存储器。

注：STC单片机使用STC提供的stc-isp软件下载程序

- 提供了下面的链接：手册、在线帮助、器件手册和用户指南。

μ Vision5集成开发环境和调试器是整个Keil开发工具链的中心，它们提供了大量的特性以帮助程序开发人员快速完成嵌入式应用的开发。 μ Vision5提供了：

- 建立模式（Build Mode），用于创建应用程序；
- 以及调试模式（Debug Mode），用于调试应用程序。

Keil μ Vision集成开发环境介绍

--软件下载

下载 μ Vision5集成开发环境的步骤主要包括：

- 在IE浏览器中，输入<http://www.keil.com>，登陆keil官网。
- 在打开的Keil官网左侧的Software Downloads下找到并点击Product Downloads。



如需开始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

Keil μ Vision集成开发环境介绍

--软件下载

- 在打开的页面中，出现Download Products页面。在该页面中，点击C51，如下图所示。

Download Products

Select a product from the list below to download the latest version.



MDK-ARM v5

Version 5.13 (December 2014)

Development environment for Cortex and ARM devices.



MDK-ARM v4

Version 4.74 (April 2014)

Development environment for Cortex and ARM devices.



C51

Version 9.53 (August 2014)

Development tools for all 8051 devices.



C166

Version 7.54 (August 2014)

Development tools for C166, XC166, & XC2000 MCUs.



C251

Version 5.57 (August 2014)

Development tools for all 80251 devices.

Keil μ Vision集成开发环境介绍

--软件下载

- 打开C51界面，该界面提供了列表，需要填写相关信息，如右图所示。

Home / Product Downloads

C51

Development tools for Classic and Extended 8051 Microcontrollers
Version 9.53
Complete the following form to download the Keil software development tools.

Enter Your Contact Information Below

First Name:	he
Last Name:	bin
E-mail:	hebin@mail.buct.edu.cn
Company:	Buct
Address:	Beijing
City:	beijing
State/Province:	Select Your State or Provi
Zip/Postal Code:	100029

注：凡是标识黑体的项，都需要提供信息，不必是真实的信息。但是E-mail信息必须是真实的。

Keil μ Vision集成开发环境介绍

--软件下载

- 当填写所要求的必要的信息后，点击该页面下方的Submit按钮。
- 出现新的界面。在该界面下，点击C51V953.EXE图标

[Home](#) / [Product Downloads](#)

C51

Development tools for Classic and Extended 8051 Microcontrollers
Version 9.53

The Keil C51 Evaluation Kit allows you to create programs for all 8051 derivatives.

- Review the [hardware requirements](#) before installing this software.
- Note the [limitations of the evaluation tools](#).

MD5: 459fa4dd8dbc5a38597ae903d8fca052

To install the C51 Software...

- Right-click on **C51V953.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

C51V953.EXE (68,242K)
Monday, August 11, 2014

Estimated File Download Time:
< 5.0 Hours: 56Kb Modem
< 2.2 Hours: 128Kb ISDN
< 11 Minutes: T1/Broadband

Keil μVision集成开发环境介绍

--软件下载

- 出现提示信息，点击保存按钮，出现浮动菜单，如右图所示。在浮动菜单内，选择另存为，将下载的安装包保存到读者指定的路径下。



保存安装包提示信息

至此，成功的下载μVision5安装包文件。

Keil μ Vision集成开发环境介绍

--软件的安装

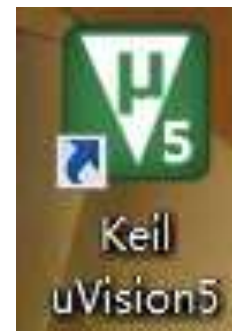
安装 μ Vision5集成开发环境的步骤主要包括：

- **在保存安装包的路径下，双击安装包图标，开始安装软件的过程。**
- **按照安装过程中的提示信息，完成软件的安装。**

Keil μ Vision集成开发环境介绍

--软件的安装

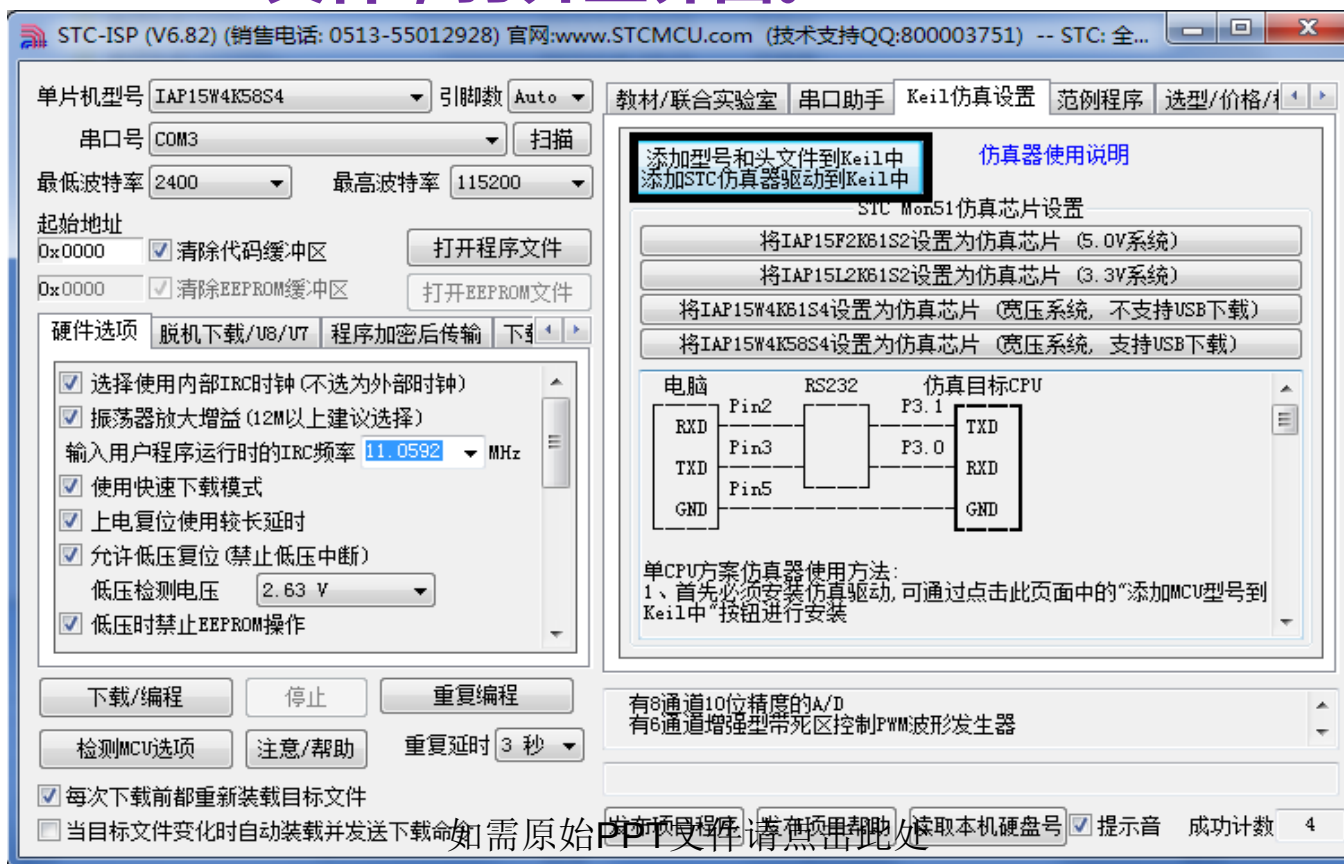
- 当安装成功后，可以看到在Windows 7操作系统的开始菜单下，出现图标，如左图所示；或者在Windows 7操作系统桌面上出现图标，如右图所示。



Keil μ Vision集成开发环境介绍

--软件的安装

■ 在本书提供的资料中，找到并双击STC公司提供的stc-isp-15xx-v6.82.exe文件，打开主界面。



Keil μ Vision集成开发环境介绍

--软件的安装

- 在该界面内的右侧窗口中，点击“Keil仿真设置”标签。在该标签页下，点击“添加型号和头文件到Keil中，添加STC仿真器驱动到Keil中”按钮。
- 出现浏览文件夹对话框界面。在该界面中，出现“请选择Keil的安装目录（例如：C:\Keil）（目录下必须有C51目录和Uvx目录存在）”提示信息，如右图所示。



Keil μ Vision集成开发环境介绍

--软件的安装

- 在该界面中，将路径定位到c:\keil_v5路径下。

注：读者根据自己安装 μ Vision5的路径选择所指向的路径。

- 点击OK按钮。
- 出现添加STC-MCU器件成功的消息对话框界面。

Keil μ Vision集成开发环境介绍

--软件的启动

启动 μ Vision5集成开发环境的步骤主要包括：

- 在Windows 7操作系统中，选择并点击Keil μ Vision5图标
- 出现Keil μ Vision5启动界面，如下图所示。



Keil μ Vision集成开发环境介绍

--建立新的设计工程

建立新设计工程的步骤主要包括：

- 打开 μ Vision5集成开发环境。
- 在 μ Vision5集成开发环境主界面主菜单下，选择Project->New μ Vision Project...。
- 出现Create New Project对话框界面。在文件名右侧的文本框中输入top。

注：表示该工程的名字是top.uvproj。

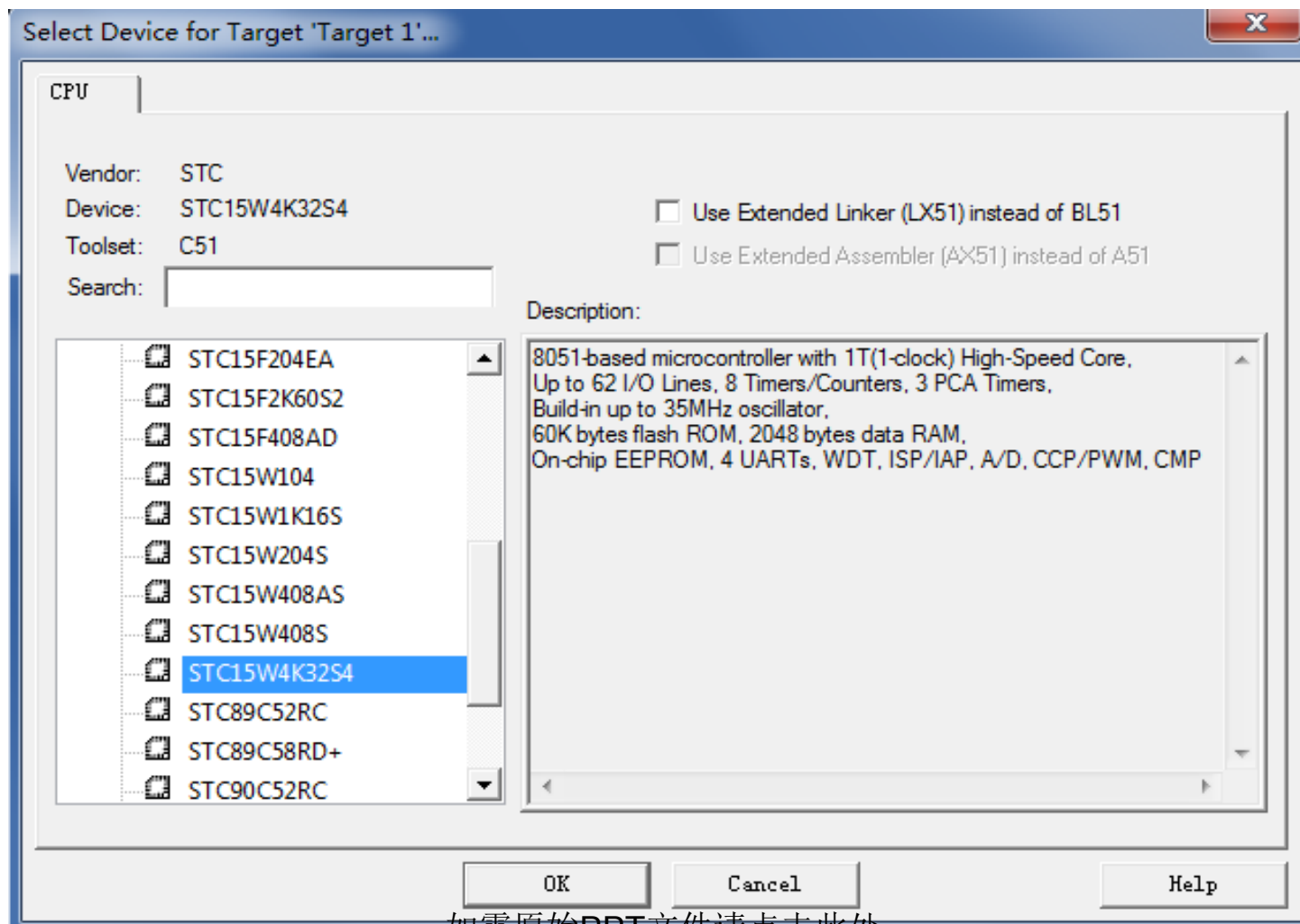
Keil μ Vision集成开发环境介绍

--建立新的设计工程

- 单击OK按钮。
- 出现Select a CPU Data Base File对话框界面。在该界面中的下拉框中，选择STC MCU Database选项。
- 单击OK按钮。
- 出现Select Device for Target' Target 1' ...对话框界面。在该界面中左侧的窗口中，找到并展开STC前面的'+'。在展开项中，找到并选择STC15W4K32S4，如下图所示。

Keil μ Vision集成开发环境介绍

--建立新的设计工程



Keil μ Vision集成开发环境介绍

--建立新的设计工程

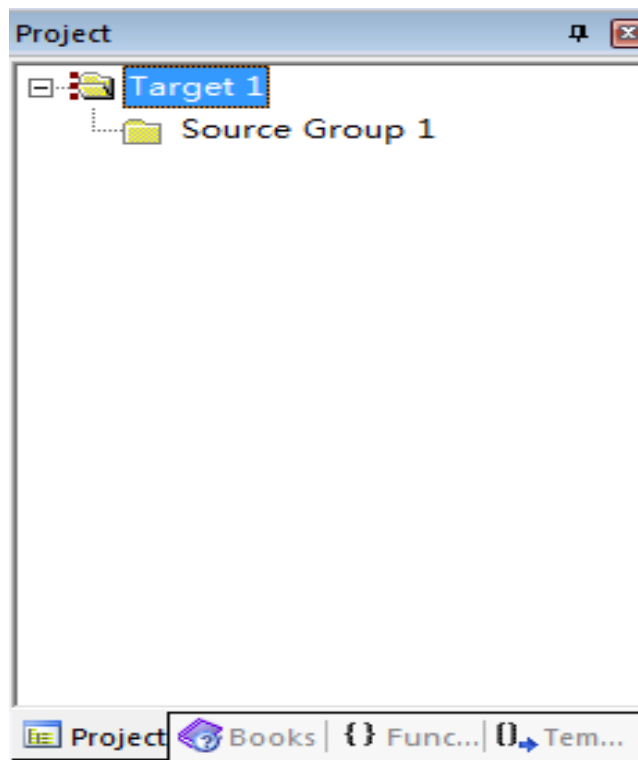
- 单击OK按钮。
- 出现Copy' STARTUP.A51' to Project Folder and Add File to Project?对话框界面。该界面提示是不是在当前设计工程中添加STARTUP.A51文件。

注：在汇编语言程序设计中，不需要添加该文件。在C语言程序设计中，也不需要添加该文件。

- 单击“否(N)”按钮。
- 在主界面左侧窗口中，选择Project标签。在该标签窗口下，给出了工程信息，如右图所示。

Keil μ Vision集成开发环境介绍

--建立新的设计工程



其中，顶层文件夹名字为Target1。在该文件夹下，存在一个Source Group 1子目录。

Keil μ Vision集成开发环境介绍

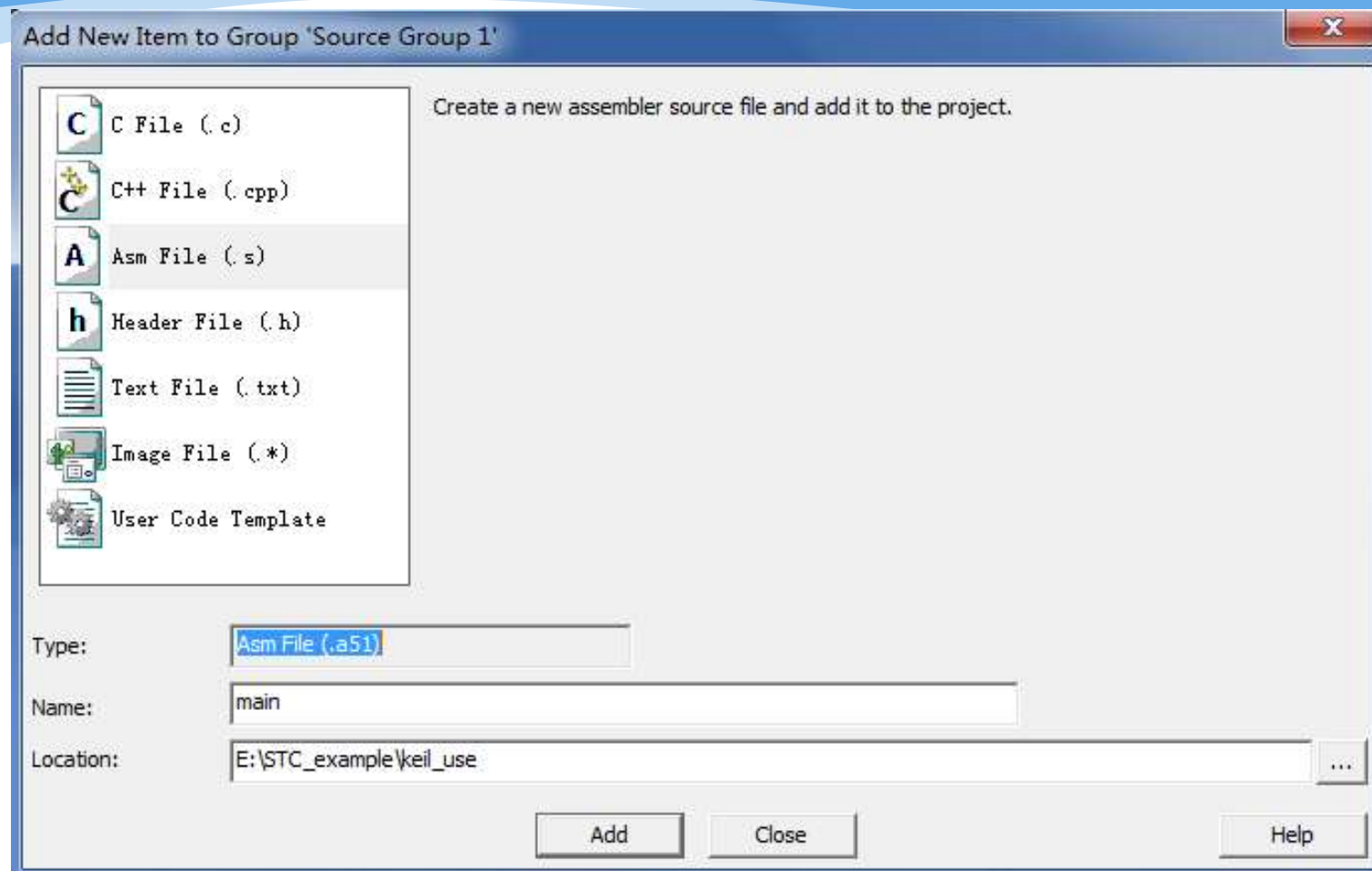
--添加新的汇编语言文件

添加汇编语言文件的步骤主要包括：

- 在Project窗口界面下，选择Source Group 1，单击右键，出现浮动菜单。在浮动菜单内，选择Add New Item to Group 'Source Group 1' 选项。
- 出现Add New Item to Group' Source Group 1' 对话框界面，如后图所示，按下面设置参数：
 - 在该界面左侧窗口中，选中Asm File(.s)。
 - 在Name右侧的文本框中输入main。

Keil μ Vision集成开发环境介绍

--添加新的汇编语言文件



注：该汇编语言的文件名字为main.a51。

■ **点击Add按钮。**

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

Keil μ Vision集成开发环境介绍

--添加新的汇编语言文件

- 在右图所示的Project窗口中，在Source Group 1子目录下添加名字为main.a51的汇编语言文件。
- 在右侧窗口中，自动打开了main.a51文件。
- 输入代码，如后图所示。
- 保存设计代码。

```

1  NAME main
2  seg_1 SEGMENT CODE
3          RSEG      seg_1
4  TABLE:  DB      3,2,5,0x0EE
5  seg_2 SEGMENT IDATA
6          RSEG      seg_2
7  BUFFER:  DS      19
8  seg_3 SEGMENT XDATA
9          RSEG      seg_3
10 XBUFFER: Ds      100
11
12 myprog SEGMENT CODE
13         RSEG      myprog
14         USING 0
15         LJMP      main
16         org      0x100
17 main:    MOV      DPTR, #TABLE      ;access code segment
18         MOV      A, #3
19         MOVC     A, @A+DPTR
20         MOV      P1, #0
21         MOV      P1, A
22
23         MOV      R0, #BUFFER      ;access IDATA segment
24         MOV      @R0, A
25         INC      A
26         INC      R0
27         MOV      @R0, A
28
29         MOV      DPTR, #XBUFFER    ;access XDATA segment
30         INC      A
31         MOVX     @DPTR, A
32         INC      DPTR
33         CLR      A
34         MOVX     @DPTR, A
35
36 END
37

```

Keil μ Vision5汇编语言设计流程

--建立设计

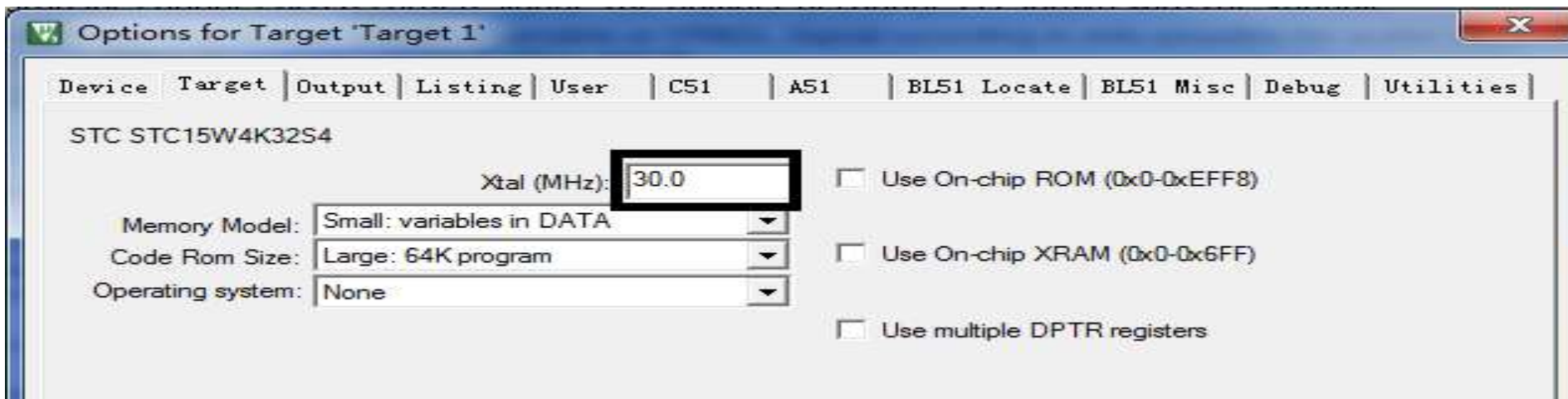
建立（Build）参数进行设置，并实现对设计的建立过程的步骤主要包括：

- 在如右图所示的窗口中，选中Target 1文件夹，并单击右键，出现浮动菜单。在浮动菜单内，选中Options for Target 'Target 1' ...选项。

Keil μ Vision5汇编语言设计流程

--建立设计

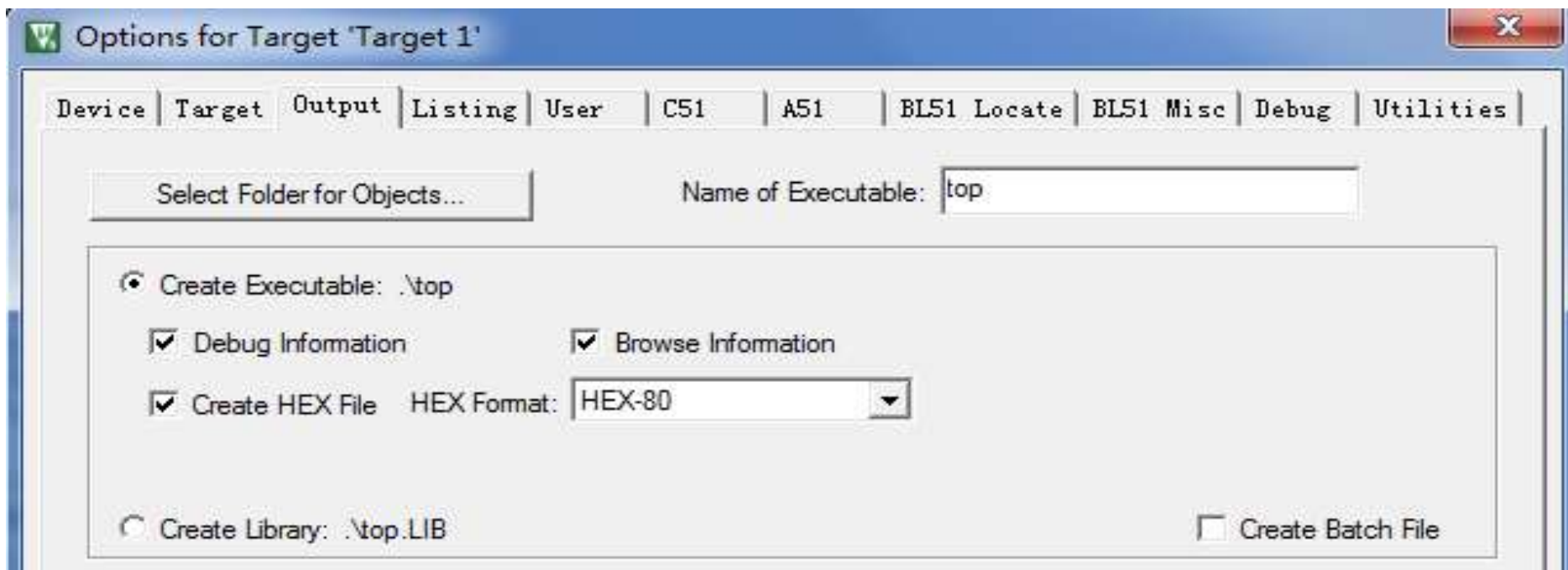
- 出现Options for Target 'Target 1' 对话框界面。在该界面中，点击Target标签。在该标签界面中，按下面设置参数：
 - 在Xtal (MHz) 右侧文本框中，输入30.0，如下图所示。
 - 其余按默认设置。



Keil μ Vision5汇编语言设计流程

--建立设计

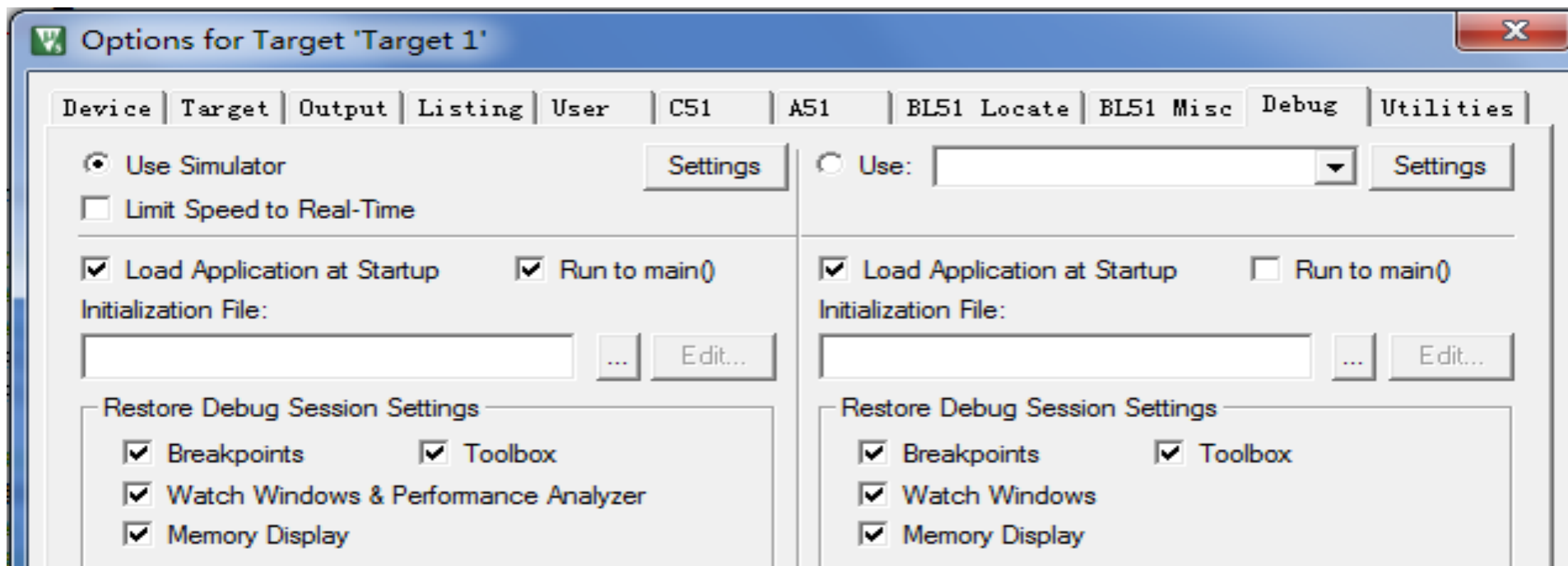
- 在该对话框界面下，再次选中Output标签。在该标签界面下，选中Create HEX File前面的复选框，如下图所示。



Keil μ Vision5汇编语言设计流程

--建立设计

- 在该对话框界面下，再次选中Debug标签。在该标签界面下，选中Use Simulator前面的单选框，如下图所示。



Keil μ Vision5汇编语言设计流程

--建立设计

- 点击OK按钮，退出目标选项对话框界面。
- 在主界面主菜单下，选择Project->Build target。开始对设计进行建立的过程。

注：该过程对汇编文件，进行汇编和链接，最后生成可执行二进制文件和HEX文件。

Keil μ Vision5汇编语言设计流程

--分析.m51文件

分析.m51文件的步骤包括：

- 在当前设计工程的目录中，找到并用写字板打开top.m51文件。
在该文件中，LINK MAP OF MODULE：标题下给出了该设计中，不同段在存储器中的空间分配情况，如下图所示。

Keil μ Vision5汇编语言设计流程

--分析.m51文件

LINK MAP OF MODULE: top (MAIN)

TYPE	BASE	LENGTH	RELOCATION	SEGMENT NAME

*****		D A T A	M E M O R Y	*****
REG	0000H	0008H	ABSOLUTE	"REG BANK 0"
IDATA	0008H	0013H	UNIT	SEG_2
*****		X D A T A	M E M O R Y	*****
XDATA	0000H	0064H	UNIT	SEG_3
*****		C O D E	M E M O R Y	*****
CODE	0000H	0119H	UNIT	MYPROG
CODE	0119H	0004H	UNIT	SEG_1

Keil μ Vision5汇编语言设计流程

--分析.m51文件

- 继续浏览该文件，在该文件SYMBOL TABLE OF MODULE标题下，给出了该程序代码中，所有变量和端口在存储器中的空间分配，如下图所示。
- 关闭该文件。

Keil μ Vision5汇编语言设计流程

--分析.m51文件

SYMBOL TABLE OF MODULE: top (MAIN)

VALUE	TYPE	NAME
-----	-----	-----
	MODULE	MAIN
C:0119H	SEGMENT	SEG_1
I:0008H	SEGMENT	SEG_2
X:0000H	SEGMENT	SEG_3
C:0000H	SEGMENT	MYPROG
I:0008H	SYMBOL	BUFFER
C:0100H	SYMBOL	MAIN
D:0090H	SYMBOL	P1
C:0119H	SYMBOL	TABLE
X:0000H	SYMBOL	XBUFFER
C:0000H	LINE#	15
C:0100H	LINE#	17
C:0103H	LINE#	18
C:0105H	LINE#	19
C:0106H	LINE#	20
C:0109H	LINE#	21
C:010BH	LINE#	23
C:010DH	LINE#	24
C:010EH	LINE#	25
C:010FH	LINE#	26
C:0110H	LINE#	27
C:0111H	LINE#	29
C:0114H	LINE#	30
C:0115H	LINE#	31
C:0116H	LINE#	32
C:0117H	LINE#	33

Keil μ Vision5汇编语言设计流程

--分析.lst文件

分析.lst文件的步骤包括：

- 在当前设计工程的目录中，找到并用写字板打开main.lst文件。

在该文件中，LINK MAP OF MODULE：标题下给出了该设计中，不同段在存储器中的空间分配情况，如下图所示。

注：(1)在该文件中，给出了每条汇编语言助记符机器指令在Flash程序存储器中的相对位置。

(2)在该文件中，给出了每条汇编语言助记符的机器指令的机器码。

- 关闭该文件。

程序代码在 STC 单片机程序 Flash 的位置分配

汇编语言助记符对应的机器指令

LOC	OBJ	LINE	SOURCE
		1	NAME main
		2	seg_1 SEGMENT CODE
		3	RSEG seg_1
0000	030205EE	4	TABLE: DB 3, 2, 5, 0x0EE
		5	seg_2 SEGMENT IDATA
		6	RSEG seg_2
0000		7	BUFFER: DS 19
		8	seg_3 SEGMENT XDATA
		9	RSEG seg_3
0000		10	XBUFFER: Ds 100
		11	
		12	myprog SEGMENT CODE
		13	RSEG myprog
		14	USING 0
0000	020000	F 15	LJMP main
0100		16	org 0x100
0100	900000	F 17	main: MOV DPTR, #TABLE ;access code segment
0103	7403	18	MOV A, #3
0105	93	19	MOVC A, @A+DPTR
0106	759000	20	MOV P1, #0
0109	F590	21	MOV P1, A
		22	
010B	7800	F 23	MOV R0, #BUFFER ;access IDATA segment
010D	F6	24	MOV @R0, A
010E	04	25	INC A
010F	08	26	INC R0
0110	F6	27	MOV @R0, A
		28	
0111	900000	F 29	MOV DPTR, #XBUFFER ;access XDATA segment
0114	04	30	INC A
0115	F0	31	MOVX @DPTR, A

Keil μ Vision5汇编语言设计流程

--分析HEX文件

分析HEX文件的步骤包括：

- 在当前设计工程的目录中，找到并用写字板打开top.hex文件，如下图所示。

```
:04011900030205EEEA
```

```
:03000000020100FA
```

```
:10010000900119740393759000F5907808F604082F
```

```
:09011000F690000004F0A3E4F0F5
```

```
:00000001FF
```

Keil μ Vision5汇编语言设计流程

--分析HEX文件

许多Flash编程器都要求输入文件具有Intel HEX格式，一个Intel HEX文件的一行称为一个记录，每个记录都由16进制字符构成，两个字符表示一个字节的值。Intel HEX文件通常由若干记录组成，每个记录具有如下的格式：

: II aaaa tt dd...dd cc

其中：

- “ : ” 表示记录起始的标志。Intel HEX文件的每一行都是以 “ : ” 开头。

Keil μ Vision5汇编语言设计流程

--分析HEX文件

- “ll” 表示记录的长度。用来标识该记录的数据字节数。
- “aaaa” 装入地址。他是该记录中第一个数据字节的16位地址值，用来表示该记录在EPROM存储器中的绝对地址。
- “tt” 记录类型。00表示数据记录，01表示文件结束。
- “dddd” 记录的实际字节数据值。每一个记录都由ll个字节的数据值。
- “cc” 校验和。将它的值与记录中所有字节（包括记录长度字节）内容相加，其结果应该为0，如果为其它数值为表明该记录有错。

■ 关闭该文件。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

本节将对程序进行软件仿真。软件仿真是指，在Keil μ Vision5集成开发环境中运行设计程序代码。

- **这个运行过程不需要真实STC单片机硬件平台。当程序设计者在没有实际的STC单片机开发平台时，可以借助于集成开发环境提供的各种调试工具，初步判断一下所设计的软件代码是否有缺陷，这样就能及时发现程序设计中的问题。**
- **程序软件仿真也称为脱机仿真，也就是脱离基于STC单片机的具体硬件平台的仿真。**

Keil μ Vision5汇编语言设计流程

--程序软件仿真

程序软件仿真的步骤主要包括：

- 在Keil μ Vision主界面主菜单下，选择Debug->Start/Stop Debug Session，进入调试器模式。
- 出现调试器界面，如图所示。在该调试器左边出现Registers标签窗口。在该界面的上方出现Disassembly窗口，该窗口时程序代码的反汇编程序。在该窗口下方是汇编语言源程序界面。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

The screenshot displays the Keil μ Vision5 IDE interface. On the left, the 'Registers' window shows the status of various registers. The 'Sys' section includes PC (Program Counter) at address C:0x0000. The main window shows the 'Disassembly' of the assembly code. The code starts with a jump to 'main' at address 0x100. The disassembly window also shows the source code for 'main.a51', which includes data segments (TABLE, BUFFER, XBUFFER) and a code segment (myprog) containing the 'main' function. The 'main' function starts with a jump to 'main' and then moves the address of 'TABLE' into register A.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0000
auxr1	0x00
dp1r	0x0000
states	0
sec	0.00000000
psw	0x00

```
15:          LJMP    main
16:          org     0x100
->C:0x0000  020100  LJMP    MAIN(C:0100)
C:0x0003   00      NOP
C:0x0004   00      NOP
C:0x0005   00      NOP
C:0x0006   00      NOP
C:0x0007   00      NOP
C:0x0008   00      NOP
C:0x0009   00      NOP

main.a51
4  TABLE:  DB      3,2,5,0x0EE
5  seg_2 SEGMENT IDATA
6          RSEG    seg_2
7  BUFFER:  DS      19
8  seg_3 SEGMENT XDATA
9          RSEG    seg_3
10 XBUFFER:  Ds     100
11
12 myprog SEGMENT CODE
13          RSEG    myprog
14          USING  0
15          LJMP    main
16          org     0x100
17 main:     MOV     DPTR, #TABLE      ;access code segment
18          MOV     A, #3
19          MOVC    A, @A+DPTR
20          MOV     D1, #0
```

如需原始PPT文件请点击此处

<http://www.gpnewtech.com/ppt>

Keil μ Vision5汇编语言设计流程

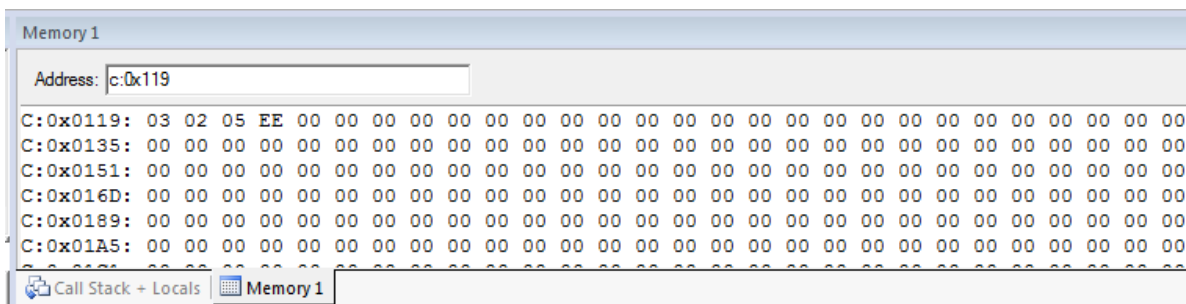
--程序软件仿真

- 在当前调试界面工具栏内，选择并单击按钮 ，对程序代码进行单步运行，然后观察寄存器界面内的寄存器内容的变化情况。然后，再单击按钮 ，再次观察寄存器内容的变化，一直运行程序直到单步运行到END为止结束。
- 在当前调试模式主界面主菜单下，选择Debug->Reset CPU，准备重新运行程序。
- 在当前调试模式主界面主菜单下，选择View->Memory Windows->Memory1；或者在当前调试模式主界面工具栏内单击按钮 ，出现浮动菜单。在浮动菜单内，选择Memory1。

Keil μVision5汇编语言设计流程

--程序软件仿真

- 在当前调试模式主界面右下角出现Memory1界面，如下图所示。
在Address:右侧输入c:0x0119。



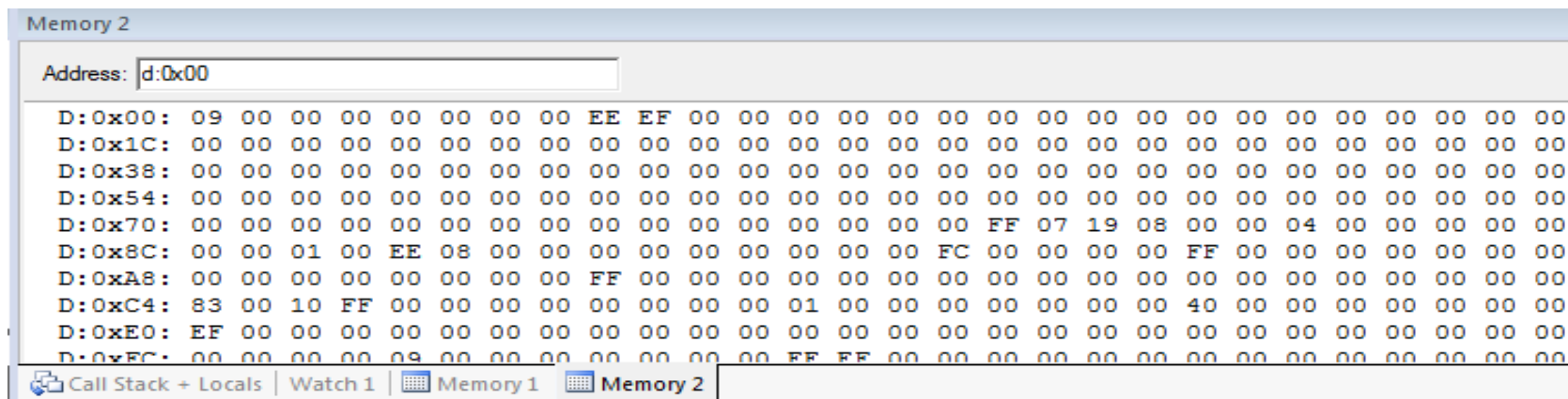
- 在当前调试界面工具栏内，单击按钮 ，对程序代码连续运行单步调试，一直到运行完第21行程序  如下图所示，然后观察图1内的存储器内容的变化情况。

```
17  main:      MOV     DPTR, #TABLE      ;access code segment
18             MOV     A, #3
19             MOVC    A, @A+DPTR
20             MOV     P1, #0
21             MOV     A, #1
22
```

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 当前调试模式主界面主菜单下，选择View->Memory Windows->Memory2。或者在当前调试模式主界面工具栏内单击按钮 ，出现浮动菜单。在浮动菜单内，选择Memory2。
- 在当前调试模式主界面右下角出现Memory2界面，如图所示。在Address: 右侧输入d:0x00。



Keil μVision5汇编语言设计流程

--程序软件仿真

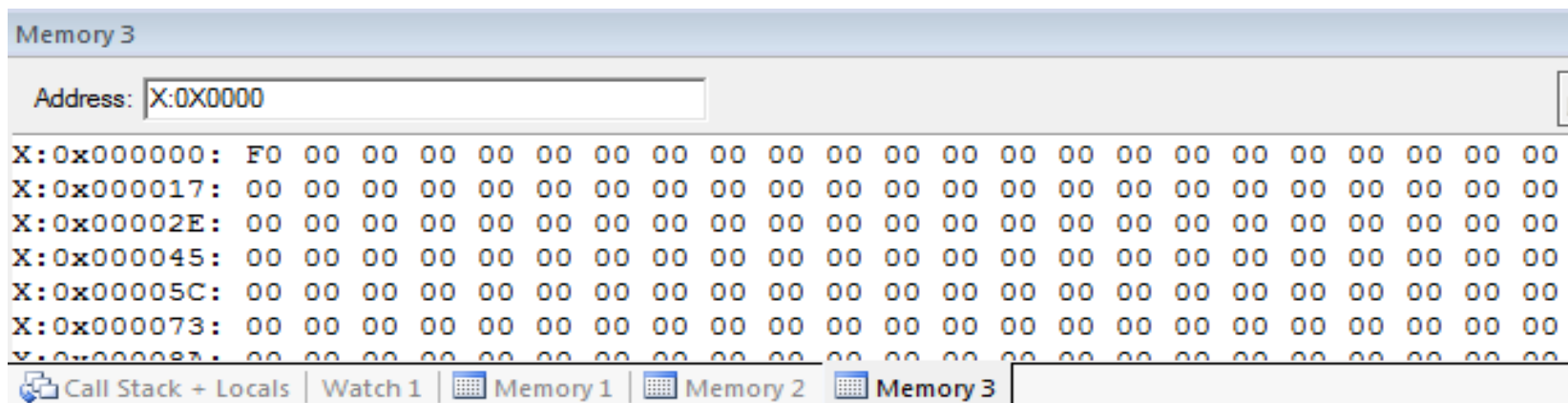
- 在当前调试界面工具栏内，单击按钮 ，对程序代码连续运行单步调试，一直到运行完第28行程序，如下图所示，然后观察上图内的存储器内容的变化情况。

23	MOV	R0, #BUFFER
24	MOV	@R0, A
25	INC	A
26	INC	R0
27	MOV	@R0, A
28		

Keil μ Vision5汇编语言设计流程

--程序软件仿真

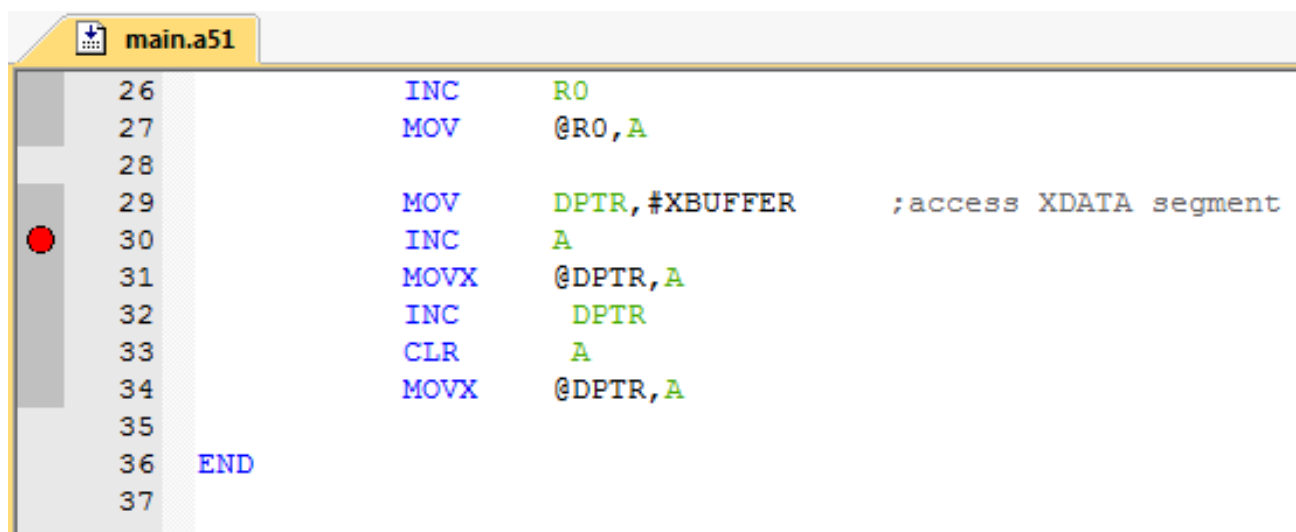
- 当前调试模式主界面主菜单下，选择View->Memory Windows->Memory3。或者在当前调试模式主界面工具栏内单击按钮 ，出现浮动菜单。在浮动菜单内，选择Memory3。
- 在当前调试模式主界面右下角出现Memory3界面，如图所示。在Address:右侧输入x:0x0000。



Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试界面工具栏内，单击按钮 ，对程序代码连续运行单步调试，一直到运行完第34行程序，如图所示，然后观察上图内的存储器内容的变化情况。



```
main.a51
26      INC      R0
27      MOV      @R0,A
28
29      MOV      DPTR,#XBUFFER      ;access XDATA segment
30      INC      A
31      MOVX     @DPTR,A
32      INC      DPTR
33      CLR      A
34      MOVX     @DPTR,A
35
36      END
37
```

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试界面主菜单，选择Debug->Reset CPU，准备重新运行程序。
- 在当前调试界面主菜单下，选择View->Trace->Enable Trace Recording前面的复选框。然后，再次选择View->Trace->Instruction Trace；或者在当前调试主界面工具栏内，单击按钮 ，出现浮动菜单。在浮动菜单内，选择Enable Trace Recording前面的复选框，然后，再次单击按钮 ，出现浮动菜单。在浮动菜单内，选择Instruction Trace选项。
- 出现指令Instruction Trace（跟踪调试）窗口界面。
- 再次连续单步运行程序，一直到程序结束为止。在每运行一步单步调试时，在跟踪调试窗口界面中，可以看到所执行的指令，以及该指令在程序存储器内所分配的地址，以及该指令的机器码，如后图所示。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

Instruction Trace			
Filter: Execution-All			
Nr.	Address	Opcode	Instruction
1	C:0x0000	020100	LJMP MAIN (C:0100)
2	C:0x0000		15: LJMP main
3	C:0x0100	900119	MOV DPTR, #0x0119
4	C:0x0100		17: main: MOV DPTR, #TABLE ; access code segment
5	C:0x0103	7403	MOV A, #0x03
6	C:0x0103		18: MOV A, #3
7	C:0x0105	93	MOVC A, @A+DPTR
8	C:0x0105		19: MOVC A, @A+DPTR
9	C:0x0106	759000	MOV P1 (0x90), #0x00
10	C:0x0106		20: MOV P1, #0
11	C:0x0109	F590	MOV P1 (0x90), A
12	C:0x0109		21: MOV P1, A
13	C:0x010B	7808	MOV RO, #0x08
14	C:0x010B		23: MOV RO, #BUFFER ; access IDATA segment
15	C:0x010D	F6	MOV @RO, A
16	C:0x010D		24: MOV @RO, A
17	C:0x010E	04	INC A
18	C:0x010E		25: INC A
19	C:0x010F	08	INC RO
20	C:0x010F		26: INC RO
21	C:0x0110	F6	MOV @RO, A
22	C:0x0110		27: MOV @RO, A
23	C:0x0111	900000	MOV DPTR, #0x0000
24	C:0x0111		29: MOV DPTR, #XBUFFER ; access XDATA segment
25	C:0x0114	04	INC A
26	C:0x0114		30: INC A
27	C:0x0115	F0	MOVX @DPTR, A
28	C:0x0115		31: MOVX @DPTR, A
29	C:0x0116	A3	INC DPTR
30	C:0x0116		32: INC DPTR
31	C:0x0117	E4	CLR A
32	C:0x0117		33: CLR A
33	C:0x0118	F0	MOVX @DPTR, A
34	C:0x0118		34: MOVX @DPTR, A
35	C:0x0119	03	RR A
36	C:0x011A	0205EE	LJMP C:05EE



Disassembly



Logic Analyzer



Instruction Trace

如需原始PPT文件请点击此处

<http://www.gpnewtech.com/ppt>

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试模式主界面主菜单下，选择View->Symbols Window；或者在当前调试模式主界面工具栏内，单击按钮 。
- 在调试界面右侧，出现Symbols窗口界面，如下图所示。在该界面下，给出SFR寄存器的地址，以及程序代码中所用到变量所分配的段以及地址信息。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

Symbols		
Module / Name	Location	Type
+ Virtual Registers		
+ Special Function R...		
- top		Application
MAIN		Module
BUFFER	I:0x08	uchar
P1	D:0x90	uchar
XBUFFER	X:0x000000	uchar

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试界面主菜单，选择Debug->Reset CPU，准备重新运行程序。
- 在当前调试界面主菜单下，选择Peripherals->I/O-Ports->Port 1。
- 弹出Parallel Port 1端口界面，如图所示。在该界面中，给出了端口1各个引脚当前的状态。
- 再次单步运行程序，一直运行到程序代码的第23行为止。



Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试界面主菜单，选择Debug->Reset CPU，准备重新运行程序。
- 在当前调试主界面主菜单下，选择View->Analysis Windows->Code Coverage选项；或者在当前调试主界面工具栏内，单击按钮 ，出现浮动菜单。在浮动菜单内，选择Code Coverage选项。
- 在调试主界面内，出现Code Coverage窗口界面，如图所示。
- 在Code Coverage窗口界面中Module:右侧的下拉框中，选择MAIN。
- 运行单步调试，可以看到代码覆盖率随程序的单步执行而不断的增加。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在当前调试界面主菜单下，选择Debug->Start/Stop Debug Session，退出调试模式主界面。

注：该步操作的目的是为了清除前面所做操作，对存储空间内容的影响。因此，下面的步骤会调用软件逻辑分析工具，调试信号的逻辑状态变化。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

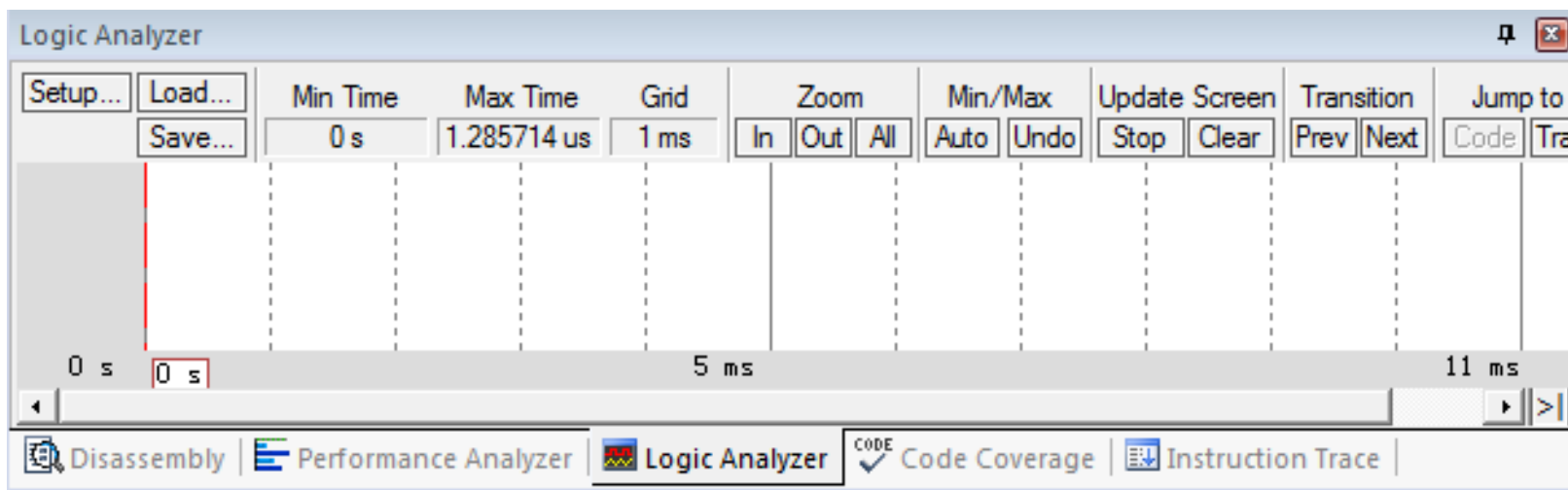
对该段代码使用逻辑分析仪进行分析的步骤主要包括：

- 在Keil μ Vision主界面主菜单下，选择Debug->Start/Stop Debug Session，进入调试模式主界面。
- 在当前调试模式主界面主菜单下，选择View->Analysis Windows->Logic Analysis；或者在当前调试主界面工具栏中，单击按钮 ，出现浮动菜单。在浮动菜单内，选择Logic Analyzer。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

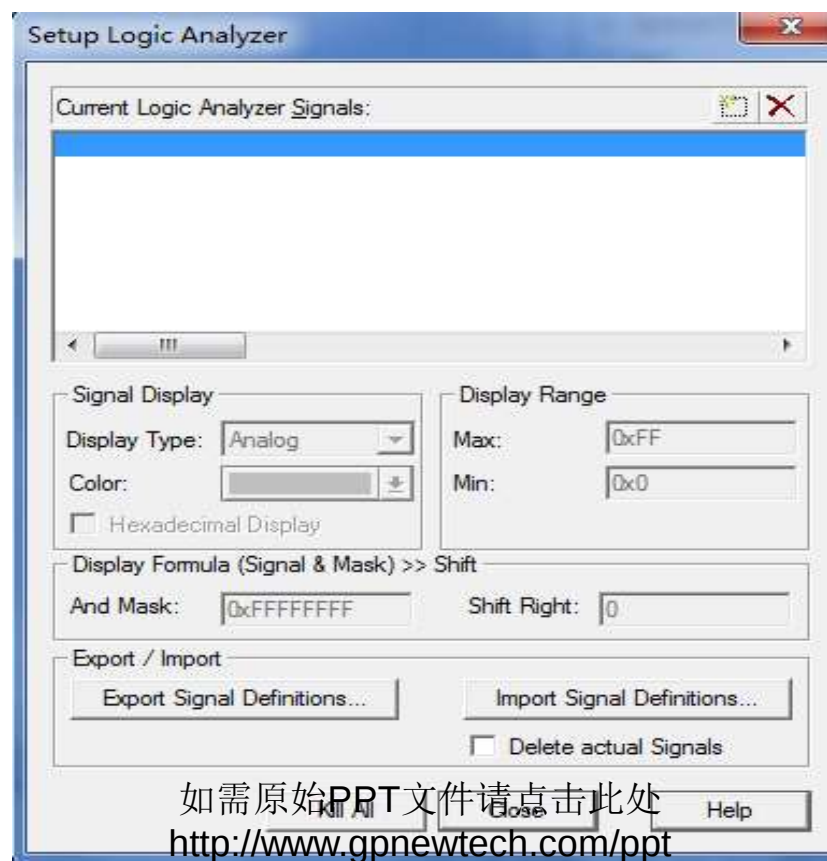
- 出现Logic Analyzser (逻辑分析仪) 窗口界面 , 如下图所示。



Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在该窗口界面中，单击Setup...按钮。
- 出现Setup Logic Analyzer界面，如下图所示。在该界面内，单击  按钮。



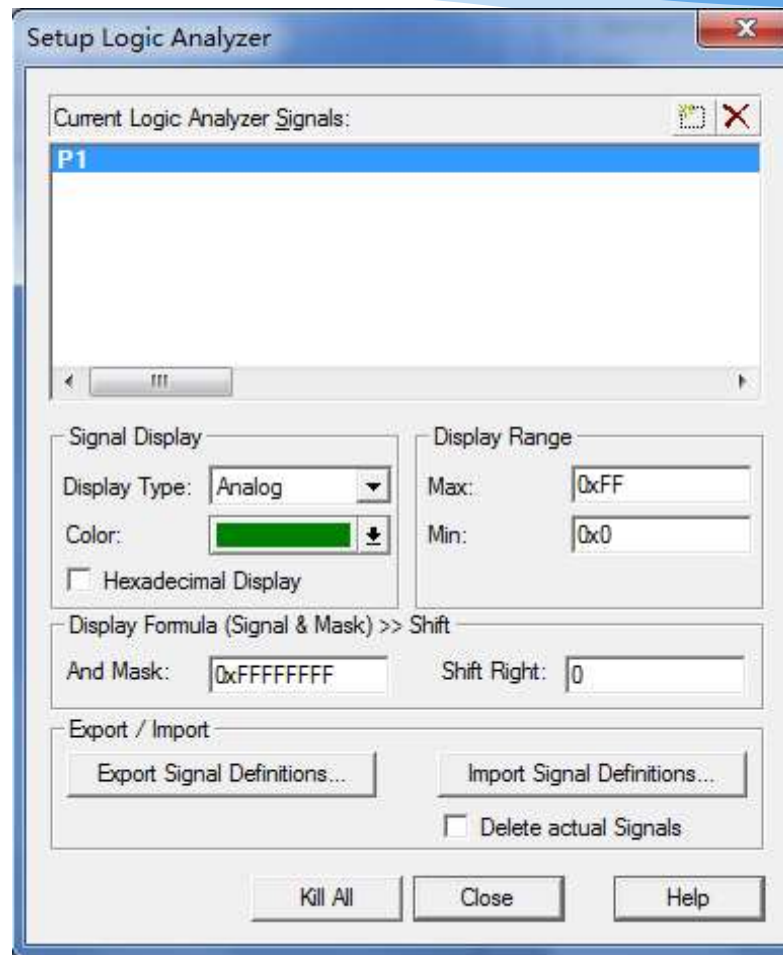
Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在Current Logic Analyzer Signals : 窗口下新添加了一个空白行。在该空白行中输入 'P1' 。然后 , 选中该行。
- 在该窗口下面的And Mask : 右侧的文本框中输入0xFFFFFFFF。其余参数保持不变 , 如下图所示。

Keil μ Vision5汇编语言设计流程

--程序软件仿真



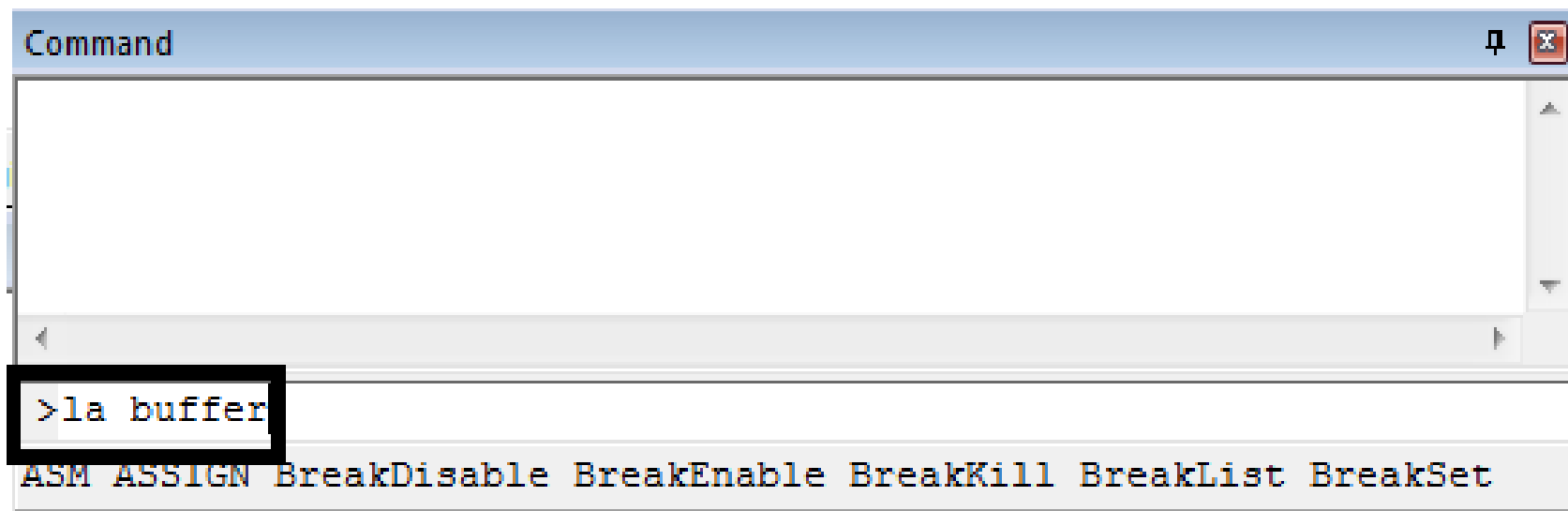
■ 单击Close按钮，退出配置逻辑分析仪选项界面。

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

Keil μ Vision5汇编语言设计流程

--程序软件仿真

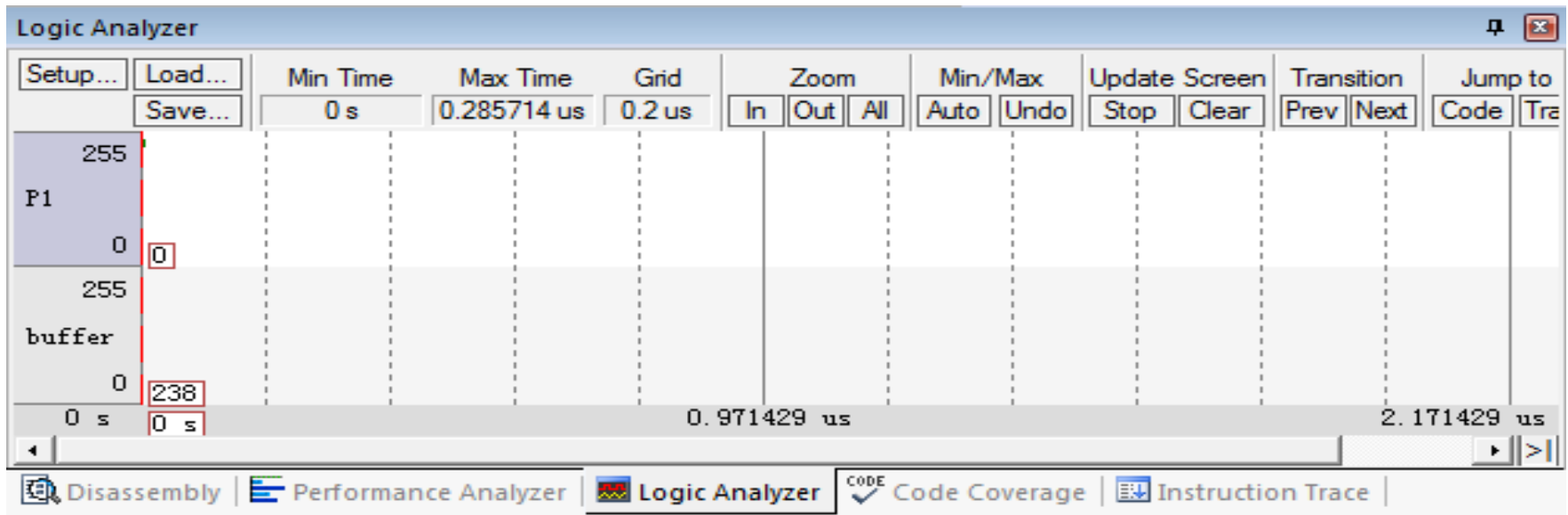
- 在当前调试模式主界面下面的Command窗口内的命令行中，输入la buffer命令，如下图所示。



Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 按回车键，可以看到在在Logic Analyzer窗口界面内，新添加了P1和buffer两个逻辑信号，如图所示。

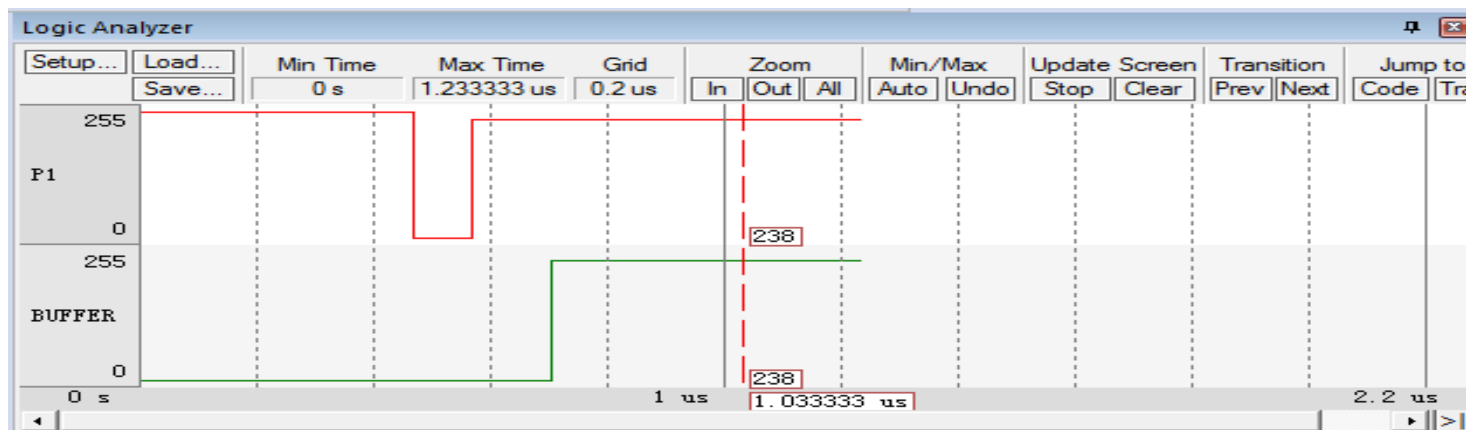


- 单步运行程序，一直运行到程序代码的第29行。

Keil μ Vision5汇编语言设计流程

--程序软件仿真

- 在Logic Analyzer窗口界面中，连续单击按钮多次 ，用于放大窗口内的信号。
- 当调整到观察范围内时，看到信号的变化过程，如图所示。



- 在当前调试主界面主菜单下，选择Debug->Start/Stop Debug Session，退出调试模式主界面。

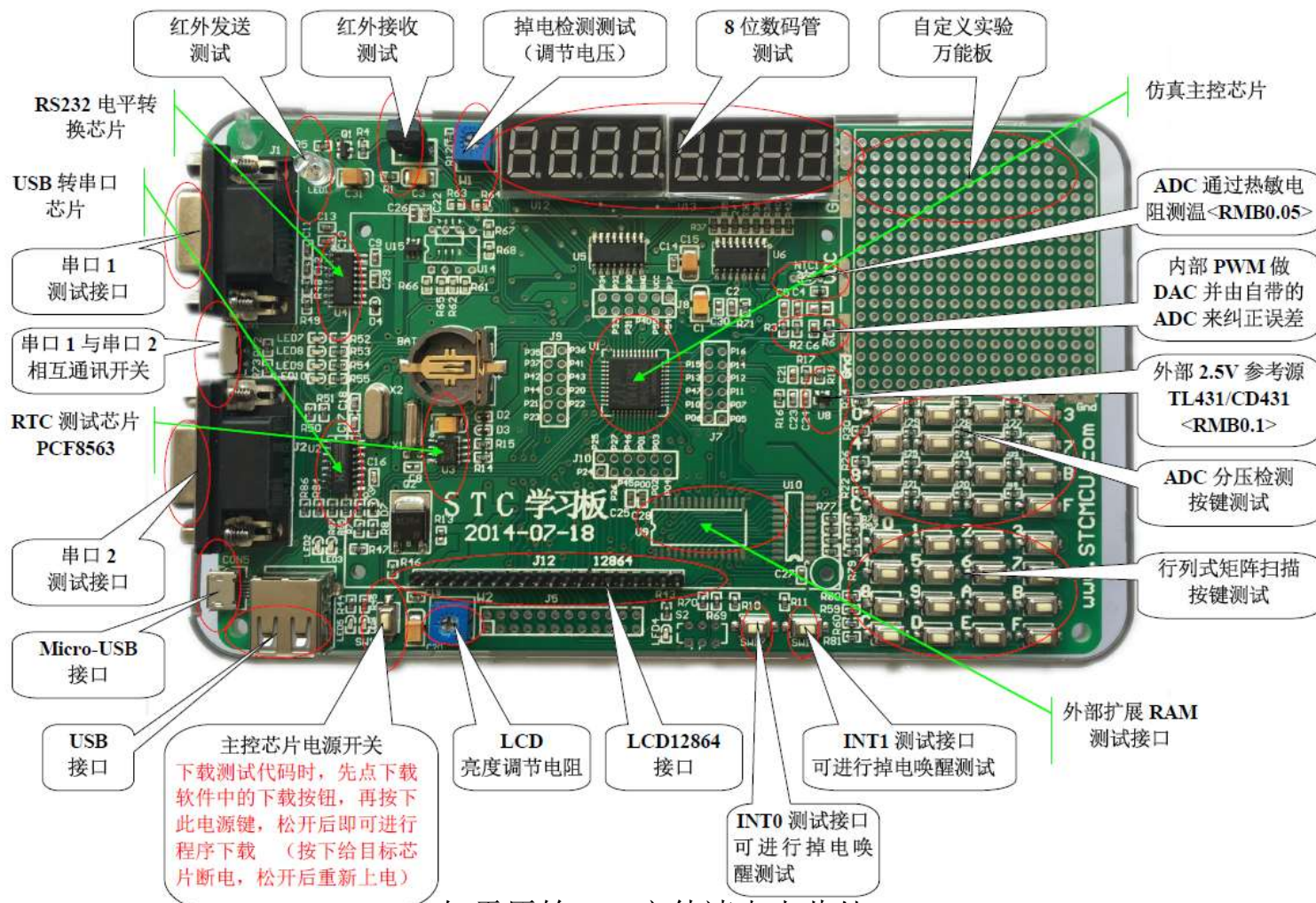
Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

本节将通过STC提供的STC学习板，对设计进行验证，如下图所示。该开发平台完全由USB供电。

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明



如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

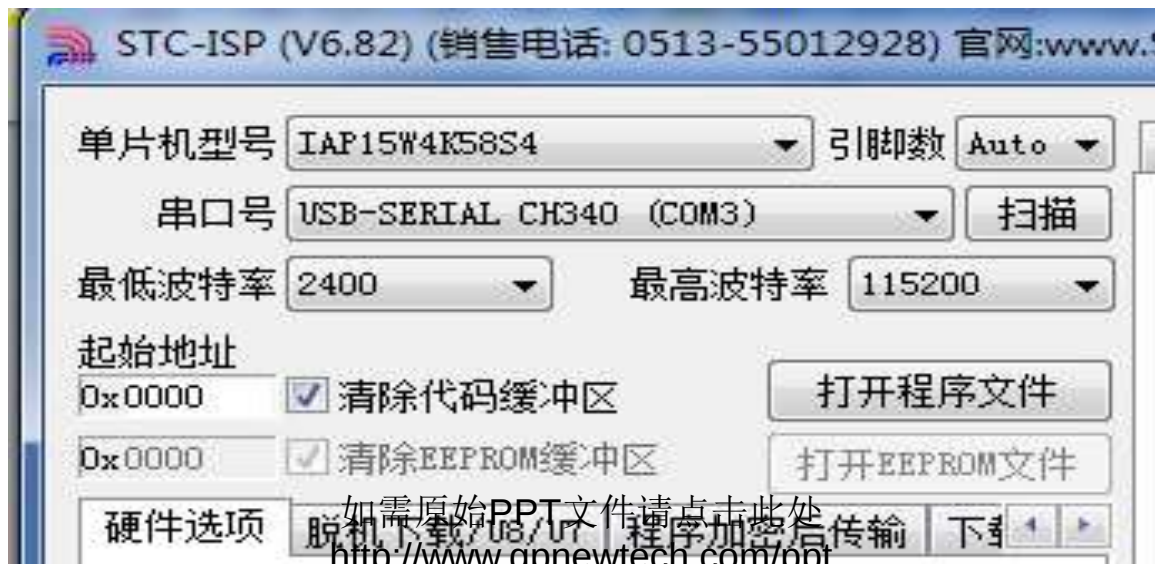
在进行硬件仿真前，需要通过下面的步骤配置硬件仿真所需要的硬件和软件环境，配置步骤主要包括：

- 打开STC学习板，在该开发板左侧找到标识为CON5 mini USB接口。将STC提供USB数据电缆的两端，分别和开发板上标识为CON5 的USB插座和PC机/笔记本电脑上的USB插座进行连接。
- 在本书所提供资料的文件夹下，找到USB电缆驱动程序子目录。打开该子目录后，进入USB to UART Driver子目录。在该子目录下，进入到CH340-CH341子目录。在该子目录下找到安装程序ch341ser，安装USB-UART的串口驱动程序。

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

■ 打开本书所提供资料下的STC-ISP软件，如图所示。在该界面，名字为串口号右侧的下拉框中，选择USB-SERIAL CH340（COM3）选项。按照图中所示，设置最低波特率和最高波特率参数。在本书中，将最低波特率设置为2400，最高波特率设置为115200。



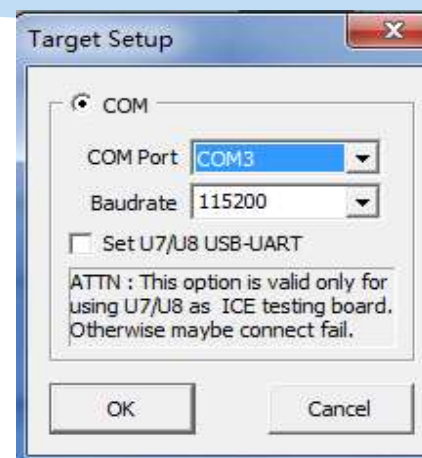
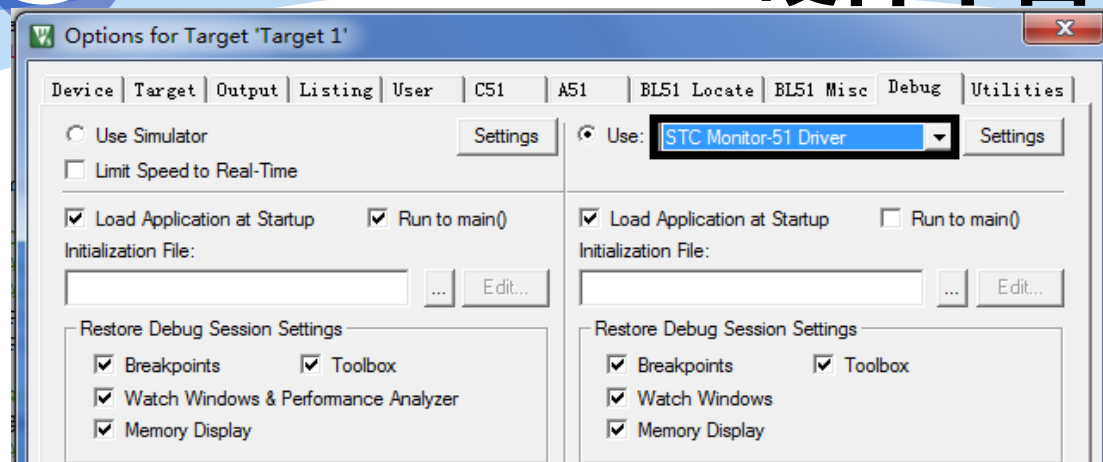
Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

- 在Keil μ Vision5集成开发环境左侧的Project窗口中，选中Target 1并单击右键，出现浮动菜单，选择Options for Target 'Target 1' 选项。
- 出现Options for Target 'Target 1' 对话框界面，如图所示。在该界面中，单击Debug标签。在该标签界面右侧窗口中，按如下设置参数：

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明



□ 选择Use前面的复选框。

□ 在右侧下拉框中，选择STC Monitor-51 Driver。

□ 单击Settings按钮。

□ 出现Target Setup对话框界面，如图2所示。在该界面中，将COM Port设置为COM3；将Baudrate设置为115200。

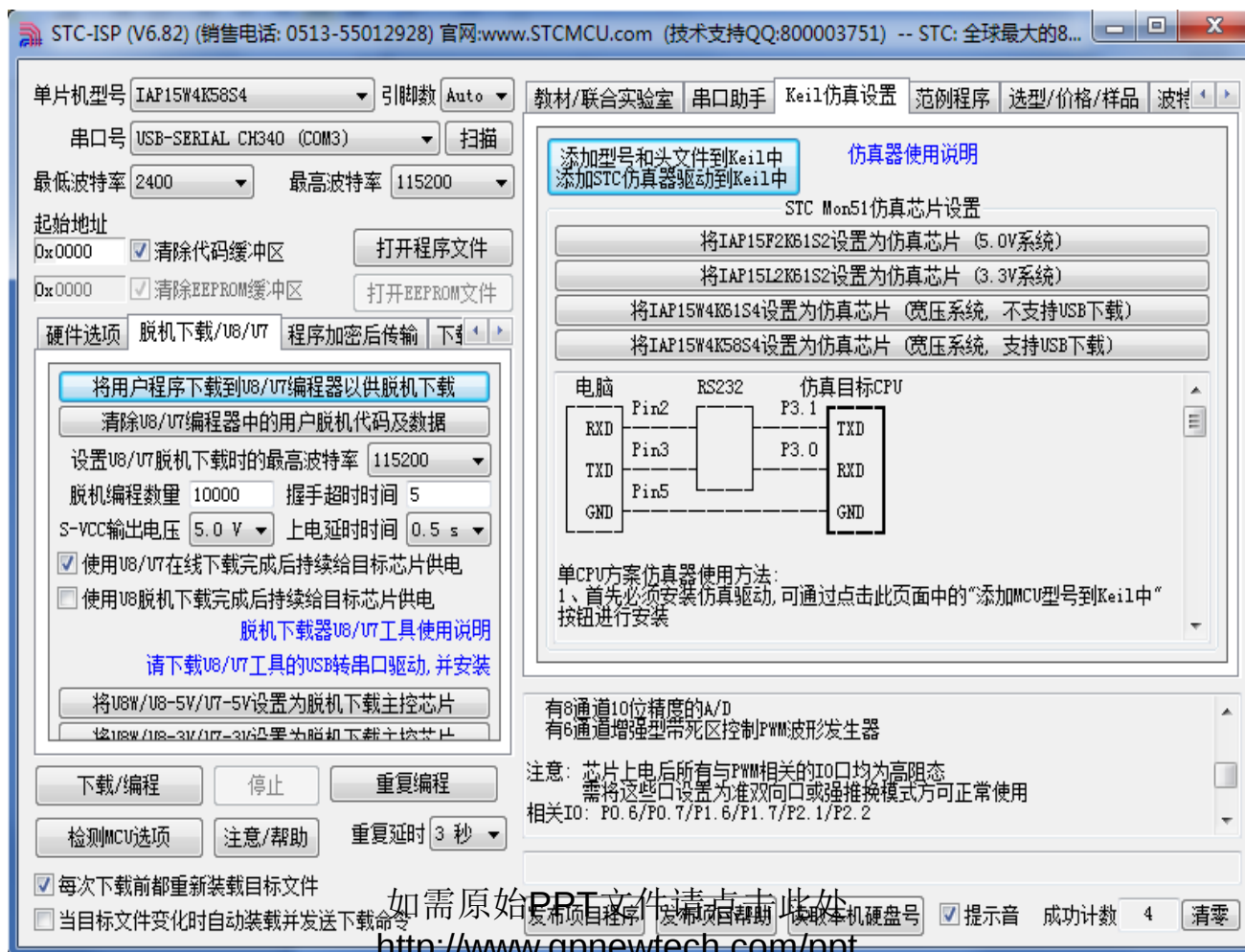
□ 单击OK按钮。退出Target Setup对话框界面。

■ 在上图所示的界面最下方，单击OK按钮，退出配置界面。

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

■ 再次进入STC-ISP程序界面，如图所示。



Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

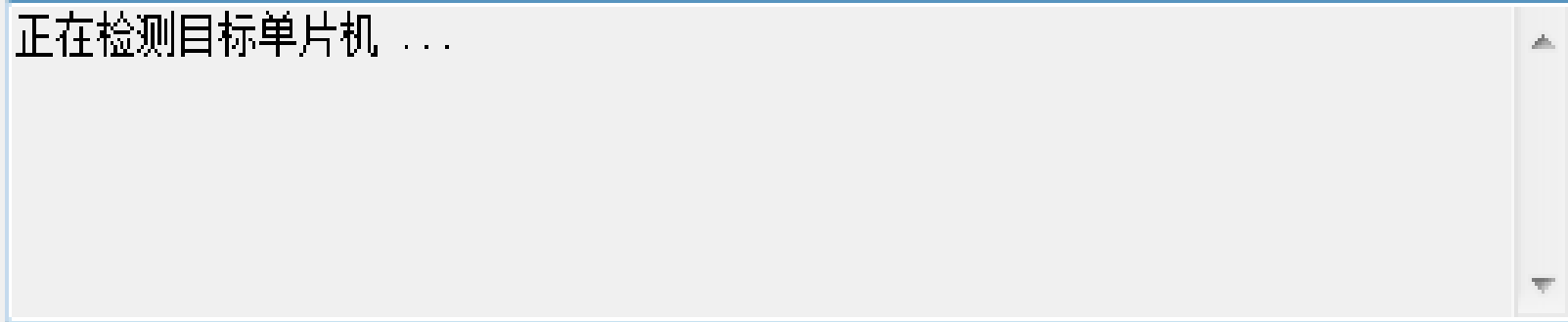
在该界面中按如下设置参数：

- 单击“打开程序文件”按钮。
- 出现“打开程序代码文件”对话框界面。在该界面中，定位到当前工程路径下，并打开top.HEX文件。
- 单击确定按钮。
- 在左侧窗口中，选择脱机下载/U8/U7标签。在该标签栏界面下，选中“使用U8/U7在线下载完成后持续给目标芯片供电”前面的复选框。

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

- 在右侧窗口中，选择Keil仿真设置标签。在该标签栏界面下，单击“将IAP15W4K58S4设置为仿真芯片（宽压系统，支持USB下载）”按钮。
- 在STC-ISP软件右下方的窗口界面内，出现正在检测目标单片机....信息，如图所示。



正在检测目标单片机 ...

Keil μ Vision5汇编语言设计流程

--STC硬件平台说明

- 在STC学习板左下方，找到一个标识为SW19的白色按键，按一下该按键。在上图所示的界面下，出现编程的相关信息，如图所示。



- 等待编程结束。

Keil μ Vision5汇编语言设计流程

--程序硬件仿真

本节将进行程序的硬件仿真。硬件仿真就是在STC单片机上真正的运行程序，然后对程序代码的执行情况进行分析。运行程序硬件仿真的步骤主要包括：

- 在Keil μ Vision主界面主菜单下，选择Debug->Start/Stop Debug Session，进入调试器模式。

Keil μ Vision5汇编语言设计流程

--程序硬件仿真

- 再次运行单步调试，先单步运行完第20行代码。即：

MOV P1,#0

注：该指令是使得P1端口置零，其中就包括让P1.6和P1.7端口置零。因此，看到LED6和LED7灯亮。

```
15          LJMP      main
16          | org      0x100
17  main:    MOV       DPTR, #TABLE
18          MOV       A, #3
19          MOVC      A, @A+DPTR
20          MOV       P1, #0
21          MOV       P1, A
22
```

Keil μ Vision5汇编语言设计流程

--程序硬件仿真

- 再次运行单步调试，先单步运行完第21行代码。即：

MOV P1 , A

注：该指令将累加器A的内容送给P1端口。由于此时累加器A的内容是0xEE，也就是P1.7和P1.6置1。因此，看到LED6和LED7灯灭。

- 在执行指令的时候，可以按照前面的方法，观察寄存器的变化情况、存储器内容和端口1的变化情况。
- 在调试主界面下，选择Debug->Reset CPU选项，准备重新执行程序。

Keil μ Vision5汇编语言设计流程

--程序硬件仿真

- 在程序代码行号前单击左键，分别在第20行、第24行和第30行添加断点，如图所示。

```
12 myprog SEGMENT CODE
13     RSEG     myprog
14     USING 0
15     LJMP     main
16     org     0x100
17 main:     MOV     DPTR, #TABLE      ;access code segment
18           MOV     A, #3
19           MOVC    A, @A+DPTR
20           MOV     P1, #0
21           MOV     P1, A
22
23           MOV     R0, #BUFFER      ;access IDATA segment
24           MOV     @R0, A
25           INC     A
26           INC     R0
27           MOV     @R0, A
28
29           MOV     DPTR, #XBUFFER   ;access XDATA segment
30           INC     A
31           MOVB    @DPTR, A
32           INC     DPTR
33           CLR     A
34           MOVB    @DPTR, A
35
36     END
```

Keil μ Vision5汇编语言设计流程

--程序硬件仿真

- 在当前调试模式主界面主菜单下，选择Debug->Run；或者直接按键F5，运行断点调试功能。
- 在当前调试模式主界面主菜单下，选择Debug->Start/Stop Debug Session，退出调试器模式。
- 在Keil μ Vision主界面主菜单下，选择Project->Close Project，退出当前工程。

单片机端口控制汇编语言程序设计

本节将设计更为复杂的端口控制汇编语言程序，帮助同学进一步的理解“软件”控制“硬件”逻辑行为的方法。

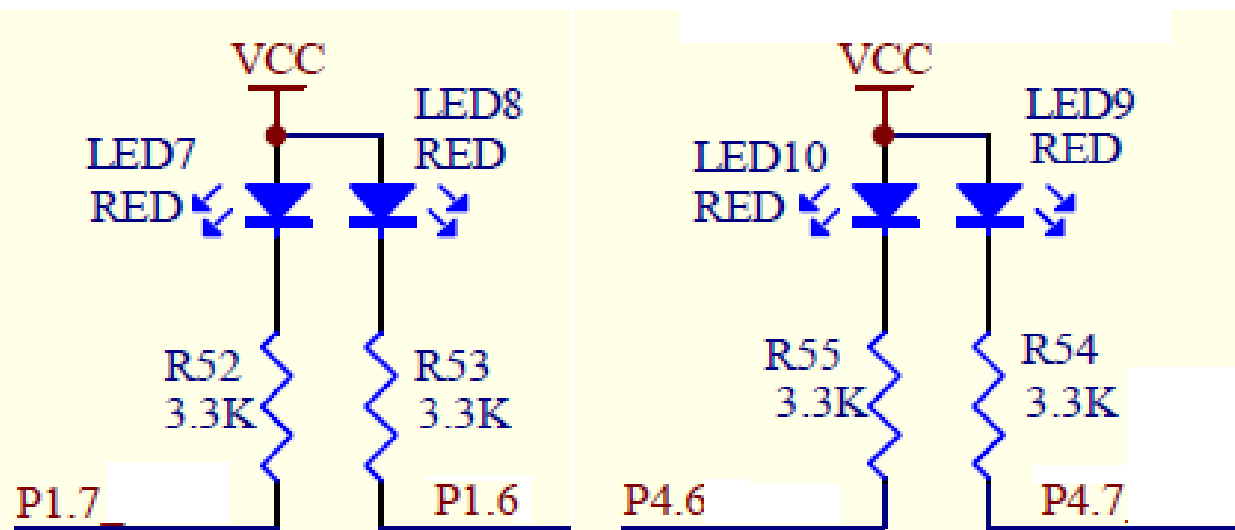
■ 设计目标：在该设计中，将设计一个在0~3之间计数（4进制）的计数器。

注：读者进入本书提供资料的\STC-example\例子5-7目录下，打开并参考该设计。

单片机端口控制汇编语言程序设计

--设计原理

在该设计中，使用了STC提供的学习板。在该学习板上提供了四个LED灯、名字分别用LED7、LED8、LED9和LED10表示。



单片机端口控制汇编语言程序设计

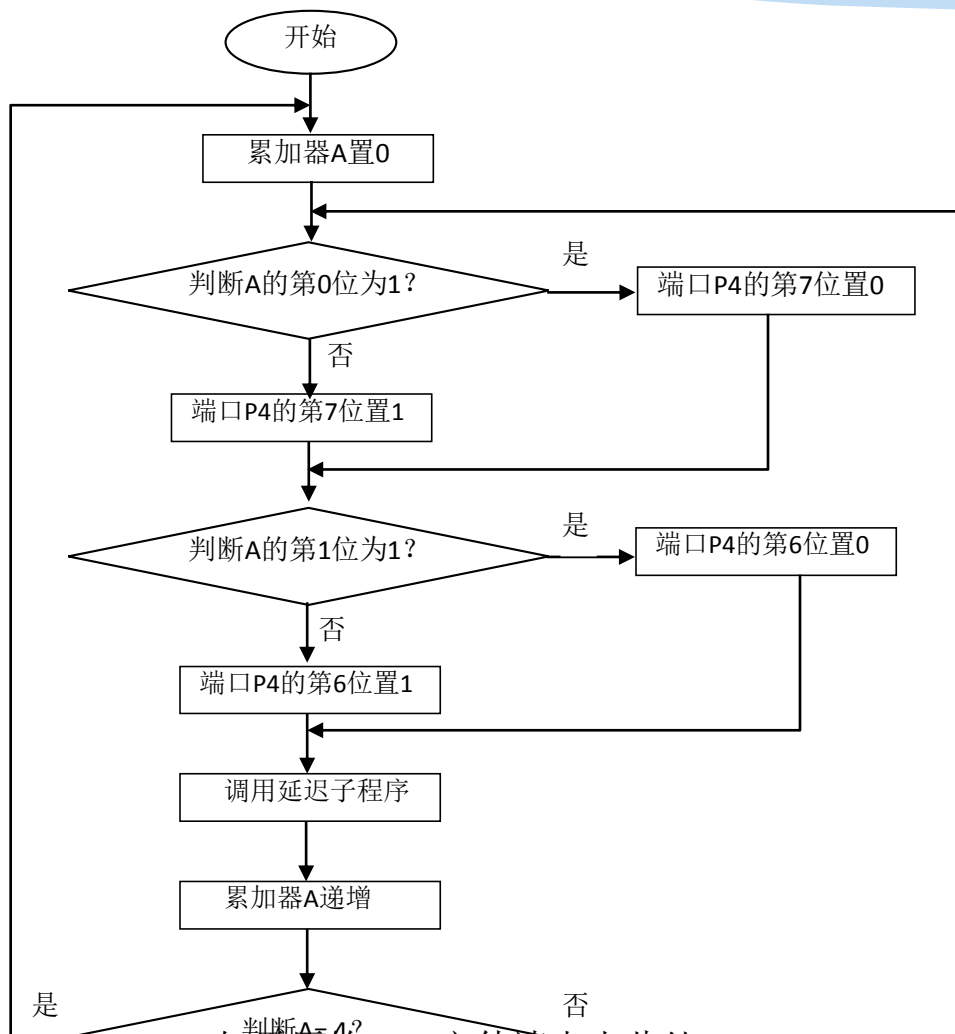
--设计原理

- 这四个LED灯的阳极共同接到了VCC电源（+5V供电），另一端通过限流电阻R52、R53、R54、R55与STC单片机IAP15W4K58S4的 P1.7、P1.6、P4.7和P4.6引脚连接。当：
 - 单片机对应的引脚置位为低时，所连接的LED亮；
 - 单片机对应的引脚置位为高时，所连接的LED灭。

注：在该设计中，只用到了与单片机P4.6和P4.7引脚连接的LED9和LED10两个LED灯。

单片机端口控制汇编语言程序设计

--设计原理



单片机端口控制汇编语言程序设计

--建立新的工程

本节将建立新的设计工程，建立新设计工程的步骤主要包括：

- 打开 μ Vision5集成开发环境。
- 在 μ Vision5集成开发环境主界面主菜单下，选择Project->New μ Vision Project...。
- 出现Create New Project对话框界面。在文件名右侧的文本框中输入top。
- 单击OK按钮。

单片机端口控制汇编语言程序设计

--建立新的工程

- 出现Select a CPU Data Base File对话框界面。在该界面中的下拉框中，选择STC MCU Database选项。
- 单击OK按钮。
- 出现Select Device for Target' Target 1' ...对话框界面。在该界面中左侧的窗口中，找到并展开STC前面的'+'。在展开项中，找到并选择STC15W4K32S4。
- 单击OK按钮。
- 在主界面左侧窗口中，选择Project标签。在该标签窗口下，给出了工程信息。

单片机端口控制汇编语言程序设计

--建立新的工程

- 出现Copy 'STARTUP.A51' to Project Folder and Add File to Project?对话框界面。该界面提示是不是在当前设计工程中添加STARTUP.A51文件。
- 单击“否(N)”按钮。

单片机端口控制汇编语言程序设计

--添加汇编语言程序

本节将为当前工程添加新的汇编语言文件。添加汇编语言文件的步骤主要包括：

- Project窗口界面下，选择Source Group 1，单击右键，出现浮动菜单。在浮动菜单内，选择Add New Item to Group 'Source Group 1' 选项。
- 出现Add New Item to Group 'Source Group 1' 对话框，按下面设置参数：
 - 在该界面左侧窗口中，选中Asm File(.s)。
 - 在Name右侧的文本框中输入main。

单片机端口控制汇编语言程序设计

--添加汇编语言程序

- 点击Add按钮。
- 在Project窗口中，在Source Group 1子目录下添加了名字为main.a51的汇编语言文件。
- 在右侧窗口中，自动打开了main.a51文件。
- 输入下面的代码。

单片机端口控制汇编语言程序设计

--添加汇编语言程序

P4 DATA 0C0H ;定义P4端口的地址

```
my_prog SEGMENT CODE  
      RSEG my_prog  
      LJMP main  
      ORG 0x100
```

单片机端口控制汇编语言程序设计

--添加汇编语言程序

main: USING 0

Loop1: MOV A,#0 ;累加器A初始化为0

Loop2: JB ACC.0, SETP41 ;如果累加器A的第0位为1，则跳转

 SETB P4.7 ;否则，则置位P4.7（P4端口的第7位）

 JMP CON ; 跳转判断下一个条件

SETP41: CLR P4.7 ;清零P4.7（P4端口的第7位）

CON: JB ACC.1,SETP42 ;如果累加器A的第1位为1，则跳转

 SETB P4.6 ;否则，则设置P4.6（P4端口的第6位）

 JMP CON1 ;跳转判断下一个条件

单片机端口控制汇编语言程序设计

--添加汇编语言程序

SETP42: CLR	P4.6	;清零P4.6 (P4端口的第6位)
CON1: MOV	R0,#20	;寄存器R0初始化为20
dly:	ACALL delay	;调用延迟子程序
	DEC R0	;R0递减
	CJNE R0,#0,dly	;当R0≠0时，一直调用延迟子程序
	INC A	;否则累加器A递增
	CJNE A,#4,Loop2	;如果累加器A的值到4，则跳转到Loop2
	JMP Loop1	;无条件跳转到Loop1

单片机端口控制汇编语言程序设计

--添加汇编语言程序

;**delay子程序为二重循环语句**

Delay : MOVR3, #0FFH

;R3寄存器初始化为0xFF

delay_1: MOVR4, #0FFH

;R4寄存器初始化为0xFF

delay_2: DECR4

;R4寄存器的值递减

CJNER4,#0,delay_2

;当R4≠0时，一直循环递减

DECR3

;否则R3递减

CJNE R3,#0,delay_1

;如果R3≠0，则跳转到delay1

RET

;从子程序返回

END

■ **保存设计代码。**

如需原始PPT文件请点击此处
<http://www.gpnewtech.com/ppt>

单片机端口控制汇编语言程序设计

--建立设计

本节将对设计建立（Build）参数进行设置，并实现对设计的建立过程，其步骤主要包括：

- 在Project窗口中，选中Target 1文件夹，并单击右键，出现浮动菜单。在浮动菜单内，选中Options for Target 'Target 1' ...选项。
- 出现Options for Target 'Target 1' 对话框界面。在该界面中，点击Target标签。在该标签界面中，按下面设置参数：
 - 在Xtal（MHz）右侧文本框中，输入30.0。
 - 其余按默认设置。

单片机端口控制汇编语言程序设计

--建立设计

- 在该对话框界面下，再次选中Output标签。在该标签界面下，选中Create HEX File前面的复选框。
- 单击OK按钮，退出目标选项对话框界面。
- 在主界面主菜单下，选择Project->Build target。开始对设计进行建立的过程。

单片机端口控制汇编语言程序设计

--下载设计

在进行硬件仿真前，需要通过下面的步骤配置硬件仿真所需要的硬件和软件环境，配置步骤主要包括：

- **打开STC学习板，在该开发板左侧找到标识为CON5的mini USB接口。将STC提供的USB数据电缆的两端，与开发板上标识为CON5的USB插座和PC机/笔记本电脑上的USB插座进行连接。**
- **打开STC-ISP软件。在该界面上面，标题为串口号右侧下拉框中，选择USB-SERIAL CH340（COM3）选项。设置最低波特率和最高波特率参数，将最低波特率设置为2400，最高波特率设置为115200。**

单片机端口控制汇编语言程序设计

--下载设计

- 进入STC-ISP程序界面。在该界面中按如下设置参数：
 - 单击“打开程序文件”按钮。
 - 出现“打开程序代码文件”对话框界面。在该界面中，定位到当前工程路径下，并打开top.HEX文件。
 - 单击确定按钮。
 - 在左侧窗口中，单击“下载/编程”按钮
 - 在STC-ISP软件由下方的窗口内，出现“正在检测目标单片机....”信息。
 - 在STC学习板左下方，找到一个标识为SW19的白色按键，按一下该按键。此时，STC-ISP软件右下角窗口中，显示编程信息。
 - 等待编程结束。

单片机中断汇编语言程序设计

本节将使用汇编语言设计中断服务程序。

- **内容包括：设计原理、建立新的工程、添加新的汇编语言文件、建立文件、下载文件。**
- **这些内容将帮助读者进一步的理解中断的机制和中断服务程序的含义。**

单片机中断汇编语言程序设计

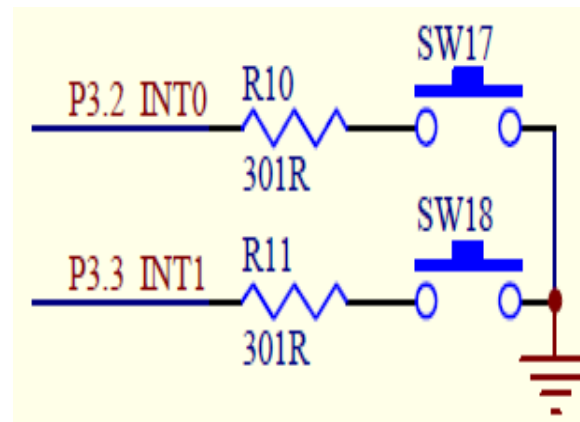
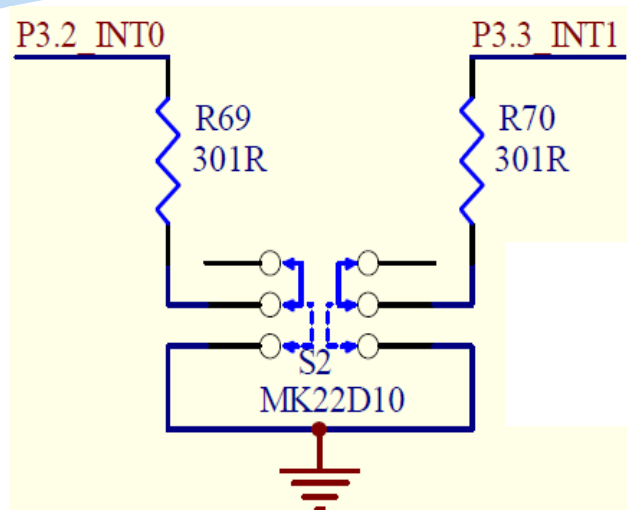
--设计原理

在该设计中，将设计一个在0~3之间计数（4进制）的计数器。

- 通过STC学习板上的P4.6和P4.7端口上的LED，显示计数的值。
- 和前面例子不一样的是，计数是通过外部中断INT0触发的，即：
 - 每次当INT0引脚下拉到地时，触发一次中断，计数器递增一次。该设计的硬件电路的触发由开关控制，如下图所示。

单片机中断汇编语言程序设计

--设计原理



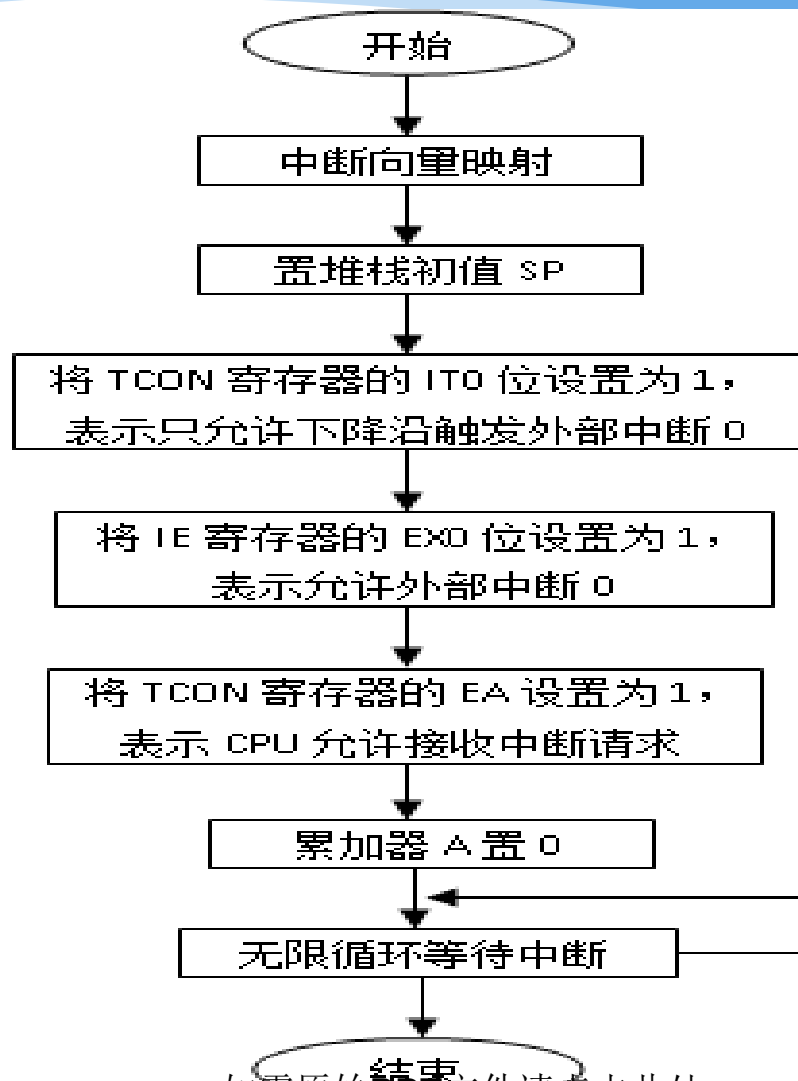
为了正确的下载程序，以及方便读者后续的实验。STC学习板上没有焊接开关。但是，提供了SW17和SW18两个按键。当：

- 按下SW17时，P3.2引脚接地，也就是产生一个INT0下降沿低脉冲信号。
- 按下SW18时，P3.3引脚接地，也就是产生一个INT1下降沿低脉冲信号。

注：在本节设计的例子中，只使用了INT0外部中断信号。

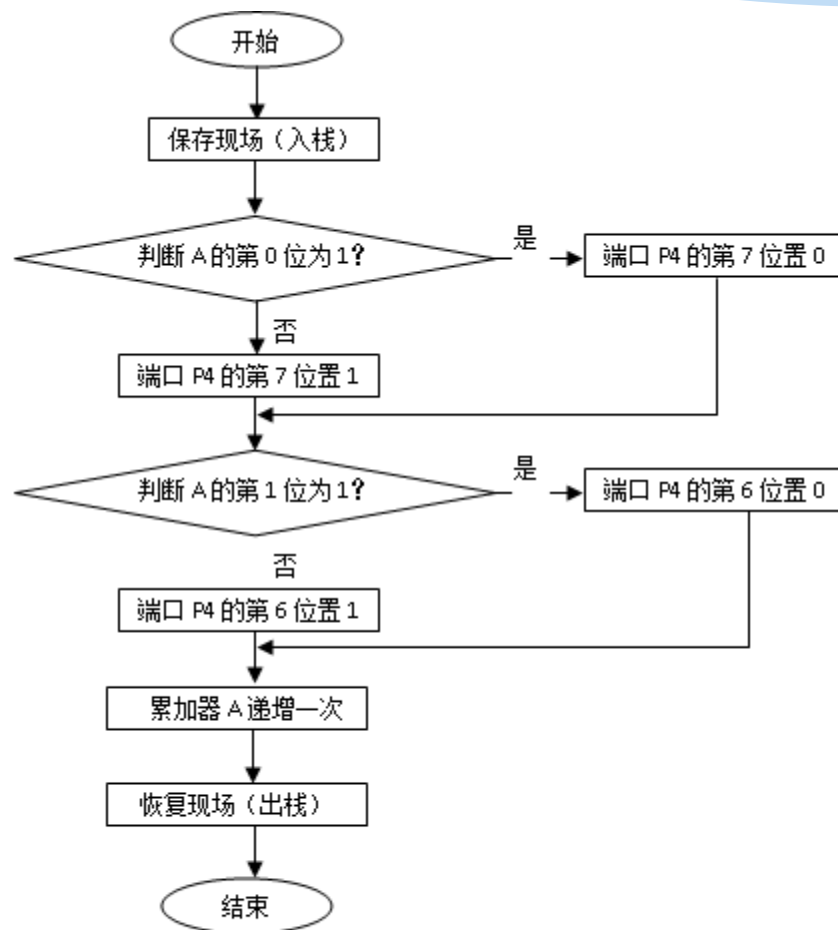
单片机中断汇编语言程序设计

--设计原理



单片机中断汇编语言程序设计

--设计原理



(b) 中断程序流程图

单片机中断汇编语言程序设计

--建立新的工程

本节将建立新的设计工程，建立新设计工程的步骤主要包括：

- 打开 μ Vision5集成开发环境。
- 在 μ Vision5集成开发环境主界面主菜单下，选择Project->New μ Vision Project...。
- 出现Create New Project对话框界面。在文件名右侧的文本框中输入top。
- 单击OK按钮。

单片机中断汇编语言程序设计

--建立新的工程

- 出现Select a CPU Data Base File对话框界面。在该界面中的下拉框中，选择STC MCU Database选项。
- 单击OK按钮。
- 出现Select Device for Target' Target 1' ...对话框界面。在该界面中左侧的窗口中，找到并展开STC前面的'+'。在展开项中，找到并选择STC15W4K32S4。
- 单击OK按钮。
- 在主界面左侧窗口中，选择Project标签。在该标签窗口下，给出了工程信息。

单片机中断汇编语言程序设计

--建立新的工程

- 出现Copy' STARTUP.A51' to Project Folder and Add File to Project?对话框界面。该界面提示是不是在当前设计工程中添加STARTUP.A51文件。
- 单击“否(N)”按钮。

单片机中断汇编语言程序设计

--添加汇编语言文件

本节将为当前工程添加新的汇编语言文件。添加汇编语言文件的步骤主要包括：

- Project窗口界面下，选择Source Group 1，单击右键，出现浮动菜单。在浮动菜单内，选择Add New Item to Group 'Source Group 1' 选项。
- 出现Add New Item to Group 'Source Group 1' 对话框，按下面设置参数：
 - 在该界面左侧窗口中，选中Asm File(.s)。
 - 在Name右侧的文本框中输入main。
- 点击Add按钮。

单片机中断汇编语言程序设计

--添加汇编语言文件

- 在Project窗口中，在Source Group 1子目录下添加了名字为main.a51的汇编语言文件。
- 在右侧窗口中，自动打开了main.a51文件。
- 输入下面的代码。

单片机中断汇编语言程序设计

--添加汇编语言文件

P4 DATA 0C0H ;P4端口的存储器地址

```
my_prog SEGMENT CODE
    RSEG      my_prog
    LJMP main
    ORG       0x0003
    LJMP      count                      ;中断映射
    ORG       0x100
```

单片机中断汇编语言程序设计

--添加汇编语言文件

main:

USING 0

MOV SP, #40H ;堆栈指针指向内部数据存储器的堆栈区

SETB IT0 ;设置外部中断0为低电平触发

SETB EX0 ;使能外部中断0

SETB EA ;使能CPU响应中断请求

MOV A, #0 ;累加器ACC初始化为0

loop: ljmp loop ;无限循环

单片机中断汇编语言程序设计

--添加汇编语言文件

```
;//=====
```

```
;// 中断服务程序: count
```

```
;//=====
```

```
count:
```

```
    PUSH    DPH                ;DPH入栈
```

```
    PUSH    02H                ;寄存器R2入栈
```

```
    JB      ACC.0, SETP41      ;如果累加器A的第0位为1，则跳转
```

```
    SETB    P4.7               ;否则，置端口P4.7为高，LED灯灭
```

```
    JMP     CON                ;无条件跳转
```

单片机中断汇编语言程序设计

--添加汇编语言文件

```
SETP41: CLR    P4.7           ;置端口P4.7为低，LED灯亮

CON:     JB     ACC.1,SETP42   ;判断如果累加器A的第1位为1，则跳转

        SETB    P4.6           ;否则，置端口P4.6为高，LED灯灭

        JMP     CON1           ;无条件跳转

SETP42:  CLR    P4.6           ;置端口P4.6为低，LED灯亮

CON1:    INC     A             ;累加器ACC递增

        POP     DPH            ;DPH出栈

        POP     02H            ;寄存器R2出栈

        RETI                  ;中断返回
```

END

单片机中断汇编语言程序设计

--分析.lst文件

在当前设计工程中，找到并打开main.lst文件

单片机中断汇编语言程序设计

--建立设计



本节将对设计建立（Build）参数进行设置，并实现对设计的建立过程，其步骤主要包括：

- 在Project窗口中，选中Target 1文件夹，并单击右键，出现浮动菜单。在浮动菜单内，选中Options for Target 'Target 1' ...选项。
- 出现Options for Target 'Target 1' 对话框界面。在该界面中，点击Target标签。在该标签界面中，按下面设置参数：
 - 在Xtal（MHz）右侧文本框中，输入30.0。
 - 其余按默认设置。

单片机中断汇编语言程序设计

--建立设计

- 在该对话框界面下，再次选中Output标签。在该标签界面下，选中Create HEX File前面的复选框。
- 在该界面中，单击Debug标签。在该标签界面右侧窗口中，按如下设置参数：
 - 选择Use前面的复选框。
 - 在右侧下拉框中，选择STC Monitor-51 Driver。
 - 单击Settings按钮。
 - 出现Target Setup对话框界面。在该界面中，将COM Port设置为COM3；将Baudrate设置为115200。
 - 单击OK按钮。退出Target Setup对话框界面。

单片机中断汇编语言程序设计

--建立设计

- 单击OK按钮，退出目标选项对话框界面。
- 在主界面主菜单下，选择Project->Build target。开始对设计进行建立的过程。

单片机中断汇编语言程序设计

--下载设计

在进行硬件仿真前，需要通过下面的步骤配置硬件仿真所需要的硬件和软件环境，配置步骤主要包括：

- 打开STC学习板，在该开发板左侧找到标识为CON5 的mini USB接口。将STC提供的USB数据电缆的两端，分别和开发板上标识为CON5的USB插座和PC机/笔记本电脑上的USB插座进行连接。
- 打开本书所提供资料下的STC-ISP软件。在该界面上方，标题为串口号右侧的下拉框中，选择USB-SERIAL CH340 (COM3) 选项。设置最低波特率和最高波特率参数。在本书中，将最低波特率设置为2400，最高波特率设置为115200。

单片机中断汇编语言程序设计

--下载设计

- 进入STC-ISP程序界面。在该界面中按如下设置参数：
 - 单击“打开程序文件”按钮。
 - 出现“打开程序代码文件”对话框界面。在该界面中，定位到当前工程路径下，并打开top.HEX文件。
 - 单击确定按钮。
 - 在左侧窗口中，单击“下载/编程”按钮
 - 在STC-ISP软件右下方的窗口内，出现“正在检测目标单片机....”信息。
 - 在STC学习板左下方，找到一个标识为SW19的白色按键，按一下该按键。此时，STC-ISP软件右下角窗口中，显示编程信息。
 - 等待编程结束。

单片机中断汇编语言程序设计

--硬件仿真

再次进入STC-ISP程序界面。在该界面中按如下设置参数：

- 单击“打开程序文件”按钮。
- 出现“打开程序代码文件”对话框界面。在该界面中，定位到当前工程路径下，并打开top.HEX文件。
- 单击确定按钮。
- 在左侧窗口中，选择脱机下载/U8/U7标签。在该标签栏界面下，选择“使用U8/U7在线下载完成后持续给目标芯片供电”前面的复选框。

单片机中断汇编语言程序设计

--硬件仿真

- 在右侧窗口中，选择Keil仿真器设置标签。在该标签栏界面下，点击“将IAP15W4K58S4设置为仿真芯片（宽压系统，支持USB下载）”按钮。
- 在STC-ISP软件右下方的窗口内，出现“正在检测目标单片机....”信息。
- 在STC学习板左下方，找到一个标识为SW19的白色按键，按一下该按键。在图STC-ISP程序界面内，出现编程的相关信息。

单片机中断汇编语言程序设计

--硬件仿真

- 在Keil μ Vision集成开发环境中，打开main.a51文件，并在该程序的第32行设置断点，如图所示。

```
31 count:
32     PUSH DPH
33     PUSH 02H
34     JB ACC.0, SETP41
35     SETB P4.7
36     JMP CON
37 SETP41: CLR P4.7
38 CON:   JB ACC.1, SETP42
39     SETB P4.6
40     JMP CON1
41 SETP42: CLR P4.6
42 CON1:  INC A
43     POP DPH
44     POP 02H
45     RETI
46 END
```

单片机中断汇编语言程序设计

--硬件仿真

- 在Keil μ Vision主界面主菜单下，选择Debug->Start/Stop Debug Session，进入调试器模式。
- 在当前调试模式主界面主菜单下，选择Debug->Run，或者直接按键F5，运行断点调试功能。
- 按下STC开发板的按键SW7一次。
- 可以看到程序进入中断服务程序，然后断点执行完中断服务程序，退出中断服务程序。

单片机中断汇编语言程序设计

--硬件仿真

- 连续触发中断若干次，进入中断服务程序。
- 在当前调试模式主界面主菜单下，选择Debug->Start/Stop Debug Session，退出调试器模式。
- 在Keil μ Vision主界面主菜单下，选择Project->Close Project，退出当前工程。