



内容

- 1、ICCAVR集成环境
 - 1.1 ICCAVR编译器简介
 - 1.2 ICCAVR中文件类型及扩展名
 - 1.3 头文件路径设置
- 2、新建工程
- 3、设置硬件及其初始化
- 4、定时器中断设置
- 5、串口调试数据获取
- 6、访问AVR的底层硬件
- 7、位操作



1.1 ICCAVR编译器简介

- **ICCAVR**是一种使用**ANSI C**语言来开发微控制器（**MCU**）程序的一个工具。
- 特点：
 - 是一个综合了编辑器和工程管理器的集成工作环境（**IDE**），是纯**32位**的程序。支持长文件名。
 - 源程序全部被组织到工程中，文件的编辑和工程的构筑也在**IDE**环境中完成。该工程管理器还能直接产生**INTER HEX**格式文件的烧写文件和符合**AVR Studio**的调试文件（**COFF**格式）。

导航、制导与控制

3/38



1.2 ICCAVR中文件类型及扩展名

(1) 输入文件类型

- .c C源程序文件
- .s 汇编源文件
- .h C头文件
- .prj 工程文件
- .a 库文件（可由几个库封装在一起，也可创建或修改自定义的库）。

(2) 输出文件类型

- .s 对应C源文件，在编译时产生同名的汇编输出文件。
- .o 由汇编产生的同名目标文件。
- .hex为**INTEL HEX**格式文件，包含程序的全部可执行代码。
- .eep为**INTEL HEX**格式文件，包含**EEPROM**的初始化数据。

导航、制导与控制

4/38



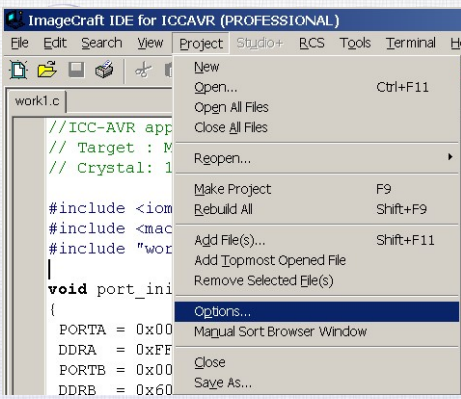
1.2 ICCAVR中文件类型及扩展名

(2) 输出文件类型

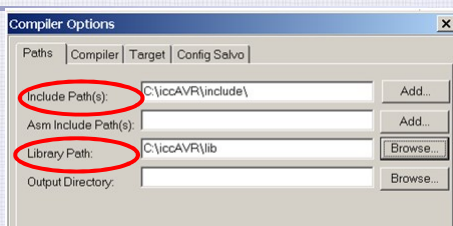
- .cof为COFF格式输出文件，用于在ATMEL的AVR Studio环境下进行程序调试。
- .lis 列表文件，列举源文件的全部语句对应的汇编代码，但变量和代码没完成绝对定位。
- .lst 列表文件，列举了含启动文件一起编译生成的全部汇编代码，是整个工程绝对定位后的完整列表文件。
- .mp 为内存映像文件，包含程序中有关符号及其所占内存大小的信息。
- .cmd 为NoICE2.xx调试命令文件。
- .noi 为NoICE3.xx调试命令文件。
- .dbg 为ImageCraft调试命令文件。

导航、制导与控制

5/38



1.3 头文件路径设置






(1) 低版本路径:

"C:\Program Files\icc\include"

"C:\Program Files\icc\lib"



(2) 高版本路径:

"C:\iccv7avr\include"

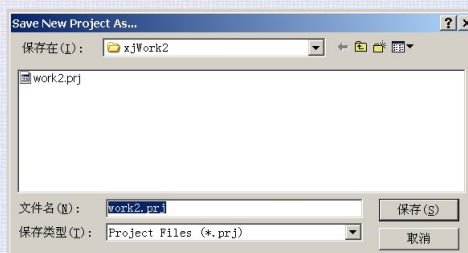
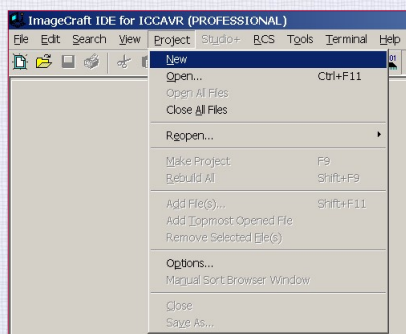
"C:\iccv7avr\lib"

6/38



2、新建工程

- 1) 打开ICCAVR平台
- 2) 建立本项目所在的目录xjWork2，并将当前目录设置到该目录下，指定项目名称work2



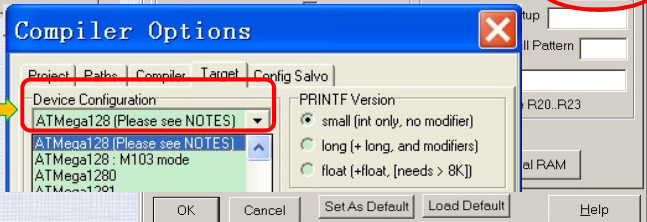
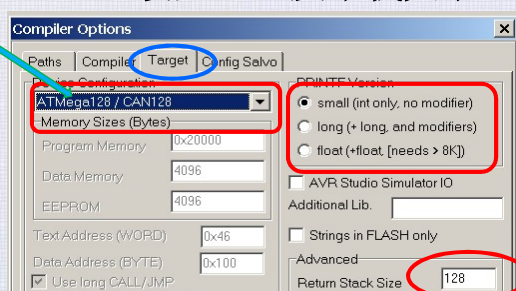
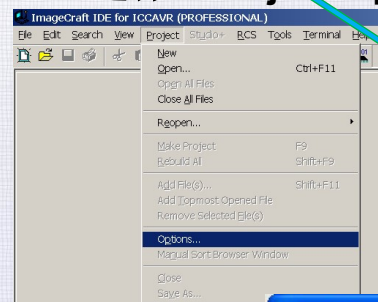
导航、制导与控制

7/38



3、设置硬件及其初始化

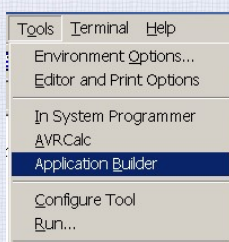
- 0) 选择“Project/Option”，设置CPU及堆栈大小





应用构筑向导

- 是用于创建外围设备初始化代码的一个图形界面。
- 使用编译选项中指定的目标**MCU**来产生相应的选项和代码。
- 显示目标**MCU**的每一个外围设备子系统，设置**MCU**所具有的中断、内存、定时器、**I/O**端口、**UART**、**SPI**和模拟量比较器等外围设备，并产生相应的代码。




导航、制导与控制

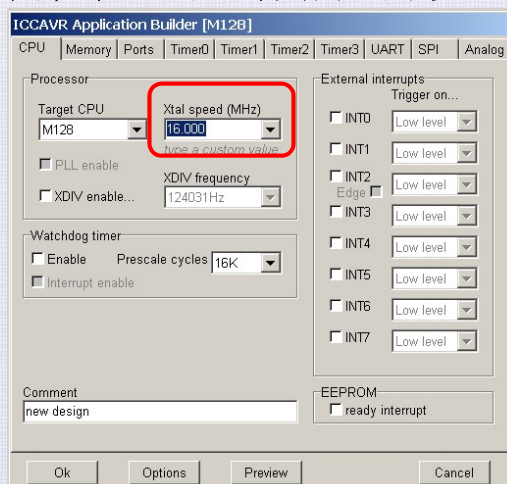
9/38



进行其他硬件接口设置

点击 ，弹出以下菜单，可进行相应的设置

- 1) **CPU**菜单
CPU种类
晶振: 16.0MHz
是否使用INT以及设置触发方式。



8



2) Memory菜单

设定是否:

- 使用SRAM
- 插入等待周期

The screenshot shows the 'Memory' tab in the 'ICCAVR Application Builder [M128]' software. The 'External memory' section has 'Enable external memory' checked and 'Wait states' set to 'None'. The 'Absolute address definitions' table is empty, with the first row showing 'Address: 0x0000', 'Name: ', 'Type: signed char', and 'Count: 1'. Buttons for 'Add', 'Modify', and 'Remove' are present. At the bottom are 'Ok', 'Options', 'Preview', and 'Cancel' buttons.

导航、制导与控制

11/38



3) Ports菜单(端口)

O: 0,1

I: 空, ↑

The screenshot shows the 'Ports' tab in the 'ICCAVR Application Builder [M128]' software. It displays configuration for seven ports (A through G). Each port has a table for 'Direction' and 'Value' bits (0-7). For Port A, bit 0 is '0' and bit 1 is '1'. For Port I, the value is '空' (empty) with an upward arrow. A note at the bottom says 'right click on a "value" bit to define signal name'. Buttons for 'Ok', 'Options', 'Preview', and 'Cancel' are at the bottom.

导航、制导与控制

12/38



Ports设置以及得到的初始化程序

```

void port_init(void)
{
    PORTA = 0x00;
    DDRA = 0xFF;
    PORTB = 0x00;
    DDRB = 0x60;
    PORTC = 0xF0;
    DDRC = 0xF0;
    PORTD = 0x0C;
    DDRD = 0x08;
    PORTE = 0xC3;
    DDRE = 0xF2;
    PORTF = 0x00;
    DDRF = 0x00;
    PORTG = 0x00;
    DDRG = 0x00;
}

```

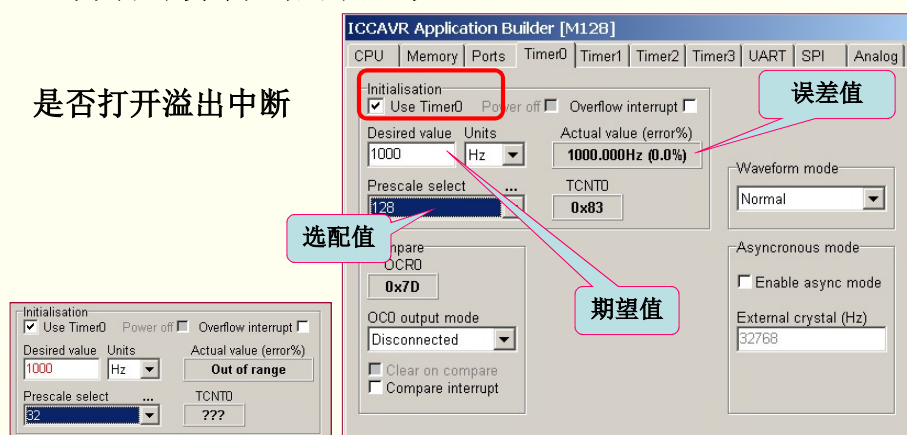
Ports连线以及设置 ■ 输出=1，输入=0

A端口:	PORTA = 0x00=0000,0000; DDRA = 0xFF=1111,1111;	LED段驱动
B端口:	PORTB = 0x00=0000,0000; DDRB = 0x60=0110,0000;	PB4: 排开1,W1 PB7: 排开2,W2
C端口:	PORTC = 0xF0=1111,0000; DDRC = 0xF0=1111,0000;	PC4~PC7: LED位选
D端口:	PORTD = 0x0C=0000,1100; DDRD = 0x08=0000,1000;	232串口1等
E端口:	PORTE = 0xC3=1100,0011; DDRE = 0xF2=1111,0010;	232串口0:收PE0发PE1
F端口:	PORTF = 0x00; DDRF = 0x00;	PF0~PF3: ADC0~ADC3
G端口:	PORTG = 0x00; DDRG = 0x00;	PG3: 排开3,W3 PG4: 排开4,W4

4) 定时器/计数器Timer0菜单

- 选择定时器0，设定周期的期望值、选配值、自动计算得到的误差值。

是否打开溢出中断



5) 定时器/计数器Timer1菜单

控制寄存器A、B

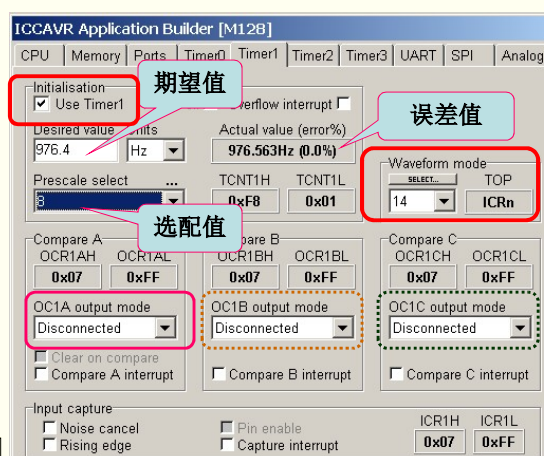
设定比较寄存器A、B以及输入捕获功能

- 波形模式：0~15
- 输出模式：

Disconnected	00
Toggled	01
Cleared	10
Set	11

OCnx: (非PWM模式)
未连接/取反/清零/置位

7	6	5	4	3	2	4	3	1	0
A	B	C	波形mode						



Timer1设置以及得到的初始化程序

```
//TIMER1 initialize - prescale:8
// WGM: 14) PWM fast, TOP=ICRn
// desired value: 976.4Hz
// actual value: 976.563Hz (0.0%)
void timer1_init(void)
{
    TCCR1B = 0x00; //stop
    TCNT1H = 0xF8; //setup
    TCNT1L = 0x01;
    OCR1AH = 0x07;
    OCR1AL = 0xFF;
    OCR1BH = 0x07;
    OCR1BL = 0xFF;
    OCR1CH = 0x07;
    OCR1CL = 0xFF;
    ICR1H = 0x07;
    ICR1L = 0xFF;
    TCCR1A = 0xA2;
    TCCR1B = 0x1A; //start Timer
}

//TIMER3 initialize - prescale:8
// WGM: 14) PWM fast, TOP=ICRn
// desired value: 976.4Hz
```

6) 定时器/计数器Timer2菜单

- 选择定时器2及其相关的期望值、选配值、自动计算得到的误差值。

选配值

期望值

误差值

7) 定时器/计数器Timer3菜单

控制寄存器A、B

- 波形模式: 0~15
- 输出模式:

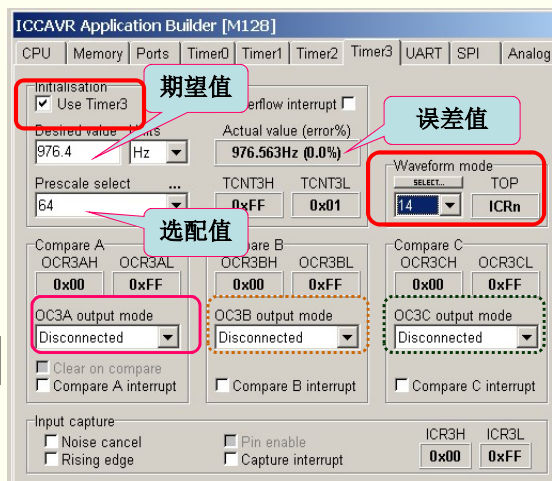
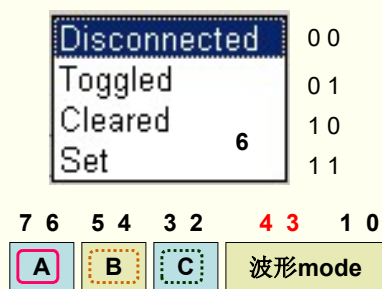


Table 61. 波形产生模式的位描述

模式	WGMn3	WGMn2	WGMn1	WGMn0	T/C工作模式	TOP
0	0	0	0	0	普通模式	0xFFFF
1	0	0	0	1	8位相位修正PWM	0x00FF
2	0	0	1	0	9位相位修正PWM	0x01FF
3	0	0	1	1	10位相位修正PWM	0x03FF
4	0	1	0	0	CTC	OCRnA
5	0	1	0	1	8位快速PWM	0x00FF
6	0	1	1	0	9位快速PWM	0x01FF
7	0	1	1	1	10位快速PWM	0x03FF
8	1	0	0	0	相频修正PWM	ICRn
9	1	0	0	1	相频修正PWM	OCRnA
10	1	0	1	0	相位修正PWM	ICRn
11	1	0	1	1	相位修正PWM	OCRnA
12	1	1	0	0	CTC	ICRn
13	1	1	0	1	保留	-----
14	1	1	1	0	快速PWM	ICRn
15	1	1	1	1	快速PWM	OCRnA



Timer3设置以及得到的初始化程序

```
//TIMER3 initialize - prescale:8
// WGM: 14) PWM fast, TOP=ICRn
// desired value: 976.4Hz
// actual value: 976.563Hz (0.0%
void timer3_init(void)
{
    TCCR3B = 0x00; //stop
    TCNT3H = 0xF8; //setup
    TCNT3L = 0x01;
    OCR3AH = 0x07;
    OCR3AL = 0xFF;
    OCR3BH = 0x07;
    OCR3BL = 0xFF;
    OCR3CH = 0x07;
    OCR3CL = 0xFF;
    ICR3H = 0x07;
    ICR3L = 0xFF;
    TCCR3A = 0x2A;
    TCCR3B = 0x1A; //start Timer
}

//UART0 initialize
// desired baud rate: 9600
```

导航、制导与控制

21/38



8) UART串口菜单

- 选用以及设置串口模式、通信波特率、数据位、RX和TX是否中断等。

```
//UART0 initialize
// desired baud rate: 9600
// actual: baud rate:9615 (0.2%)
// char size: 8 bit
// parity: Disabled
void uart0_init(void)
{
    UCSRB0 = 0x00; //disable while setting baud rate
    UCSRA0 = 0x00;
    UCSRC0 = 0x06;
    UBRR0L = 0x67; //set baud rate lo
    UBRR0H = 0x00; //set baud rate hi
    UCSRB0 = 0x98;
}

#pragma interrupt_handler uart0_rx_isr:iv_USART0
void uart0_rx_isr(void)
{
    //uart has received a character in UDR
}

//UART1 initialize
// desired baud rate:9600
// actual: baud rate:9615 (0.2%)
// char size: 8 bit
// parity: Disabled
void uart1_init(void)
{
    UCSRB1 = 0x00; //disable while setting baud rate
    UCSRA1 = 0x00;
    UCSRC1 = 0x06;
    UBRR1L = 0x67; //set baud rate lo
    UBRR1H = 0x00; //set baud rate hi
    UCSRB1 = 0x18;
}
```

导航、制导与控制

22/38



9) SPI同步串行口菜单

- 设定是否使用**SPI**以及**SPI**的模式等参数

ICCAVR Application Builder [M128]

CPU | Memory | Ports | Timer0 | Timer1 | Timer2 | Timer3 | UART | SPI | Analog

SPI

☒ Use SPI Power off ☐

☐ Enable

☐ Master mode

☐ Data LSB first

☐ SCK phase high

☐ SCK idle high

Clock rate: 4000000Hz

4

☐ Double speed

☐ Serial Transfer done interrupt

TWI

☐ Use TWI Power off ☐

☐ Enable

☐ Generate acknowledge

Slave address

Address mask

Bit rate Prescale

Rate: 988888Hz

☐ General call recognition

☐ TWI interrupt

导航、制导与控制

23/38



10) Analog模拟比较器及A/D转换菜单

ICCAVR Application Builder [M128]

CPU | Memory | Ports | Timer0 | Timer1 | Timer2 | Timer3 | UART | SPI | Analog

Analog Comparator

☒ Use Analog comparator

☐ Disable

☐ Timer1 input capture enable

☐ Multiplexer enable

☐ Bandgap reference

Interrupt trigger level

Output toggle

☐ Compare interrupt

Analog to Digital Converter

☒ Use ADC Power off ☐

☐ ADC enable

☐ Free-run select

☐ Start conversion

☐ Auto trigger

☐ Bandgap reference

☐ Internal reference

☐ Left adjust

Reference Trigger source

AREF

Free runnir

Prescale ☐ High speed

2

Conversion time: 1uS

☐ Conversion complete interrupt

Interrupt trigger level

Output toggle

Output toggle

*Reserved

Falling output edge

Rising output edge

Reference

AREF

AVCC

*Reserved

Internal

Trigger source

Free runnir

Free running

Comparator

Ext Int0

T0 CompA

T0 Overflow


T1 CompB

T1 Overflow

T1 Capture

导航、制导与控制

24/38

 **Analog设置以及得到的初始化程序**
(应该是这个画面)

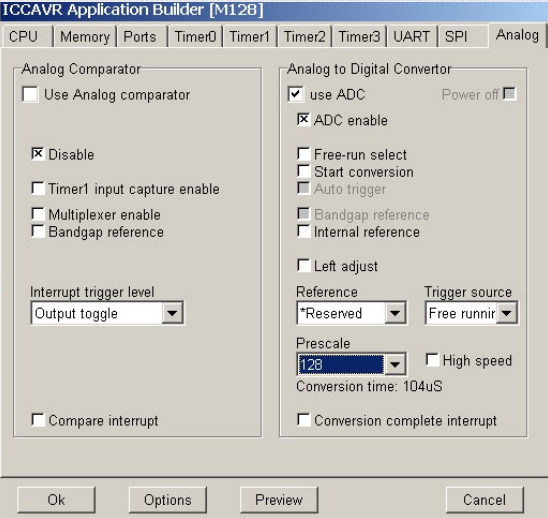
```

UCSR1A = 0x00;
UCSR1C = 0x06;
UBRR1L = 0x67; //set
UBRR1H = 0x00; //set
UCSR1B = 0x18;
}

//ADC initialize
// Conversion time: 104uS
void adc_init(void)
{
    ADCSRA = 0x00; //disa
    ADMUX = 0x40; //sele
    ACSR = 0x80;
    ADCSRA = 0x87;
}

//call this routine to in
void init_devices(void)
{
    //stop errant interru
    CLI(); //disable all
    XDIV = 0x00; //xtal


```



本实验板采用的参考电压方式

导航、制导与控制

25/38

 **Analog设置以及得到的初始化程序**
(但在计算机得下画面——需要手工修改)

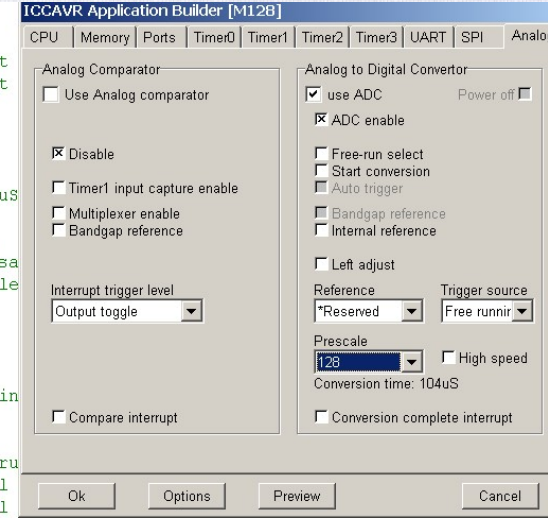
```

UCSR1A = 0x00;
UCSR1C = 0x06;
UBRR1L = 0x67; //set
UBRR1H = 0x00; //set
UCSR1B = 0x18;
}

//ADC initialize
// Conversion time: 104uS
void adc_init(void)
{
    ADCSRA = 0x00; //disa
    ADMUX = 0x00; //sele
    ACSR = 0x80;
    ADCSRA = 0x87;
}

//call this routine to in
void init_devices(void)
{
    //stop errant interru
    CLI(); //disable all
    XDIV = 0x00; //xtal

```



导航、制导与控制

26/38



生成初始化代码

- 按下“ok”后，就可以得到所需的硬件初始化程序段。
- 注意：
 1. 这些代码需要保存到一个指定的*.c文件中
 2. 需要将该文件添加到所建立的项目中。

```
//ICC-AVR application builder : 2008-1-27 12:56:58
```

```
// Target : M128
```

```
// Crystal: 16.000Mhz
```

```
#include <iom128v.h> //不检查源文件所在的文件目录，
```

```
#include <macros.h> //而直接按系统标准方式检索文件目录
```

```
#include "demo3.h"
```

27/38



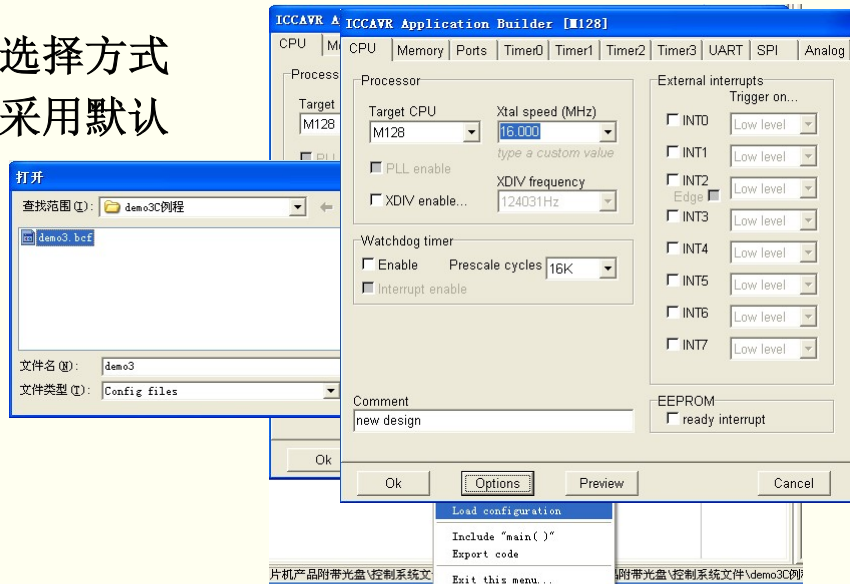
4、定时器中断设置

只生成并设置定时中断步骤：

- 4.1 选择CPU菜单(或采用原来默认设置)
- 4.2 定时器选择以及设置
- 4.3 中断服务子程序结构位置

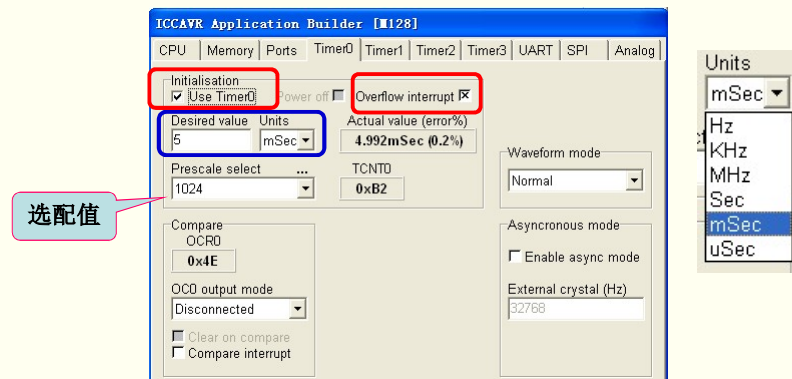
4.1 选择CPU菜单 (或采用原来默认设置)

- 选择方式
- 采用默认



4.2 定时器选择以及设置

- 由于**Time1**和**Time3**已经被用于产生**PWM**,所以可供我们使用的只有**Time0**和**Time2**.
- 假设需要采用**Time0**, 需要设置**5毫秒**的中断。





4.3 定时器设置及其产生的代码

```
//TIMER0 initialize - prescale:1024
// WGM: Normal
// desired value: 5mSec
// actual value: 4.992mSec (0.2%)
void timer0_init(void)
{
    TCCR0 = 0x00; //stop
    ASSR = 0x00; //set async mode
    TCNT0 = 0xB2; //set count
    OCRO = 0x4E;
    TCCR0 = 0x07; //start timer
}

#pragma interrupt_handler timer0_ovf_isr:17
void timer0_ovf_isr(void)
{
    TCNT0 = 0xB2; //reload counter value
}
```

ICCAVR Application Builder [128]

CPU | Memory | Ports | Timer0 | Timer1 | Timer2 | Time

Initialisation

☐ Use Timer0 Power off ☐ Overflow interrupt ☐

Desired value Units Actual value (error%)

5 mSec 4.992mSec (0.2%)

Prescale select ... TCNT0

1024 0xB2

Compare

OCRO

0x4E

OCO output mode

Disconnected

☐ Clear on compare

☐ Compare interrupt

中断服务子程序

导航、制导与控制

31/38



5、串口调试数据获取

■ 利用“串口调试助手V2.2”



串口调试助手 SComAssistant V2.2 For WIN9X/NT/2000

串口 COM1

波特率 9600

校验位 NONE

数据位 8

停止位 1

☐ 关闭串口

清空接收区 接收区

停止显示

☒ 自动清空

☐ 十六进制显示

保存显示数据 更改

C:\COMDATA

清空重植 发送的字符/数据 http://www.gjwtech.com

☐ 十六进制发送 手动发送

☐ 自动发送 (周期改变后重连)

自动发送周期: 1000 毫秒

选择发送文件 还没有选择文件 发送文件

MAIL WEB

帮助 GJW TECH 关闭程序

STATUS: COM1 OPENED, 9600, N, 8, 1 RX: 0 TX: 0 计数清零

32/38



6、访问AVR的底层硬件

- **AVR系列允许C语言程序对目标单片机的底层硬件进行访问。**
 - 在有些情况下，C语言不能很好地控制单片机的硬件，这时就要使用在线汇编和预处理宏，来访问这些硬件。
- 头文件io*.h，如io8518.h，iom128v.h等定义了指定AVR单片机的I/O寄存器的细节。
 - 这些文件是从ATMEL官方发布的文件经过修改而形成的，以匹配C语言编译器的语法要求。
 - 文件macros.h定义了很多有用的宏，例如宏UART_TRANSMIT_ON()，能使UART开始工作。
- ICCARV中的编译器效率很高，当访问由I/O寄存器映射的内存时，能产生单周期指令像in、out、sbis、sbi等。

```
#include <iom128v.h> //不检查源文件所在的文件目录
```

```
#include <macros.h> //而直接按系统标准方式检索文件目录
```

33/38



7、位操作

不支持bit和sbit数据类型，采用unsigned char来代替

- (1) a|b: 按位或。用于打开某些位，常用 |= 的形式
PORTA |= 0x80; // 打开位7（最高位）
- (2) a&b: 按位与。用于检查某些位是否置1
if (PORTA & 0x81) == 0; // 检查位7和位0
- (3) a^b: 按位异或。这个运算对一个位取反有用。
PORTA ^= 0x80; // 翻转位7
- (4) ~a: 按位取反。运算执行一个位取反。
 - 当用按位与运算关闭某些位，与这个运算组合使用时，尤其有用。
PORTA &= ~0x80; // 关闭位7
- (5) 右移, OCR1AH = pwmValue[0]>>8;
- (6) 左移, ch[2] = ADCH << 8;

注意：左右移时，移动位数在>>或<<的右边。

守规、制与控制

34/38



例子

- 如果置位I/O寄存器中的PA7位，可采用：
`PORTA |=BIT(PA7);`
- 如果需置位数据寄存器中的第7位（**char**类型变量**a**），则
`a |= (1<<7)` 或 `a =BIT (7)`
- 在程序中需要开全局中断，可以使用指令
`SEI();` 或 `_SEI();`

注意：

1. 当数0或1在<<的左边的时候，并不表示左右移动，而是表示对该符号右边的对应位进行清零或置位1
2. 当数0或1在>>或<<的左边，右边为数**a**时，表示对第**a**位数清零或置位1。

导航、制导与控制

35/38



例子

- 按位或
`PORTA |=BIT(PA7);` //置位端口**A**中的第7位
`PORTA |= 0x80;` //置位端口**A**中的第7位
`tmpB |=0x80;` //置位变量**tmpB**中的第7位
- 按位取反
`PORTA &= ~0x80;` //将端口**A**中的第7位清零
- 按位异或
`PORTA ^= ~0x80;` //将端口**A**中的第7位翻转
- 按位与
`While(PORTA & 0x40) PORTA &=~0x80;`
//若端口**A**的第6位为1,则关闭端口**A**中的第7位

导航、制导与控制

36/38



ICCAVR集成环境及 建立项目的过程 结束！

导航、制导与控制

37/38