

TE2440 用户手册 V3.0



保定飞凌嵌入式技术有限公司

网站: <http://www.witech.com.cn>

论坛: <http://www.witech.com.cn/bbs>

TE2440是由飞凌嵌入式技术有限公司设计生产的一款基于ARM9的嵌入式开发平台，它基于三星公司的ARM处理器S3C2440A，内部带有全性能的MMU(内存处理单元)，适用于设计移动手持设备类产品。TE2440全六层板，性能稳定可靠，具有高性能、低功耗、接口丰富和体积小等优良特性。目前已成功移植Linux, WINCE等操作系统到TE2440开发板。

在使用开发板时，请注意以下事项：

1. 用户在拿到开发板后，请至<http://www.witech.com.cn>网站“**客户服务**”页面注册，并用Email或者QQ通知我们您的姓名，购买时间，注册名称，开发板的编号，我们会及时为您开通会员权限，便于您及时下载更新的资料！
2. 第一次使用TE2440开发板时，**请务必先阅读用户手册**，按照手册上所述进行相关操作，谨防随意破坏系统程序！
3. 每次使用TE2440开发板前，请先将手接触开发板周围金属接口或者其它地方放电，**避免直接用手触摸芯片造成芯片烧坏！**
4. **需要对开发板进行物理操作时，请关闭电源，除USB以及网络接口（如果与局域网相接请使用普通网线，开发板带网线为计算机直连网线）外，其它接口均不支持热插拔，开发板工作时，请不要带电插拔。**
5. 本开发板硬件保修时间为三个月（非人为或不可抗力原因除外），技术支持时间三个月（论坛技术支持及“客户服务”下载时间不在此限）。

最后，欢迎您使用TE2440开发板，并提出宝贵意见！

编者：飞凌嵌入式技术有限公司

地址：河北保定市七一西路165号

邮编：071051

电话：0312-3113161

传真：0312-3102667

QQ：93644331 360189317

E-mail：bdht@witech.com.cn

网址：<http://www.witech.com.cn>

论坛：<http://www.witech.com.cn/bbs>

目 录

一、 第一章 TE2440 开发板硬件介绍.....	5
1.1 开发套件包含的内容.....	5
1.2 用户光盘内容说明.....	5
1.3 TE2440 开发板外观.....	6
1.4 TE2440 开发板硬件资源.....	6
1.5 硬件资源分配.....	8
1.5.1 地址空间分配以及片选信号定义.....	8
1.5.2 开发板接口说明.....	9
1.5.3 按键说明.....	10
1.5.4 LED 指示灯说明.....	10
1.5.5 跳线分配表.....	10
1.6 TE2440 开发板主要硬件说明.....	10
1.6.1 系统存储器.....	10
1.6.2 JTAG 及复位逻辑.....	11
1.6.3 LCD/触摸屏接口引脚定义.....	13
1.6.4 IDE （也作为总线接口） 接口引脚定义.....	15
1.6.5 GPIO 扩展口引脚定义.....	17
1.6.6 SD 卡接口.....	18
1.6.7 IIS 音频输入输出接口.....	19
1.6.8 摄像头接口.....	20
1.6.9 串口电路.....	20
1.6.10 USB 接口.....	22
1.6.11 功能按键及用户 LED 指示灯.....	22
1.6.12 红外收发电路.....	24
1.7 TE2440 支持的操作系统及其驱动.....	24
1.7.1 Linux 操作系统.....	24
1.7.2 WINCE 操作系统.....	25
二、 第二章 TE2440 开发板基本使用.....	26
2.1 TE2440 外部硬件连接.....	26
2.2 WINDOWS 下驱动的安装.....	26
2.2.1 安装 USB 驱动.....	26
2.2.2 安装并口驱动程序.....	29
2.3 调试终端使用.....	33
2.3.1 DWN 软件的使用.....	33
2.3.2 超级终端的使用.....	34
2.4 BOOTLOADER 使用全攻略.....	37
2.4.1 bootloader 简介.....	37
2.4.2 功能菜单说明.....	38
2.4.3 选择菜单说明.....	39
2.4.4 参数设置说明.....	40
2.4.5 通过 USB 下载程序并写入 FLASH.....	41

2.4.6 用 sjf2440.exe 烧写 bootloader 程序.....	43
2.5 ADS 下的 LED 试验.....	44
2.5.1 ADS 安装.....	44
2.5.2 使用 ADS 创建工程.....	44
2.5.3 编译和链接工程.....	50
2.5.4 H-JTAG 的安装使用.....	59
2.5.5 用 AXD 进行代码调试.....	62

一. 第一章 TE2440 开发板硬件介绍

TE2440 开发板为全六层板，开发板的布局和走线经过专业人士精心设计，工作非常可靠，可稳定运行在 400MHz。接口丰富，功能强大，适用于各种手持设备，消费电子和工业控制设备的开发。

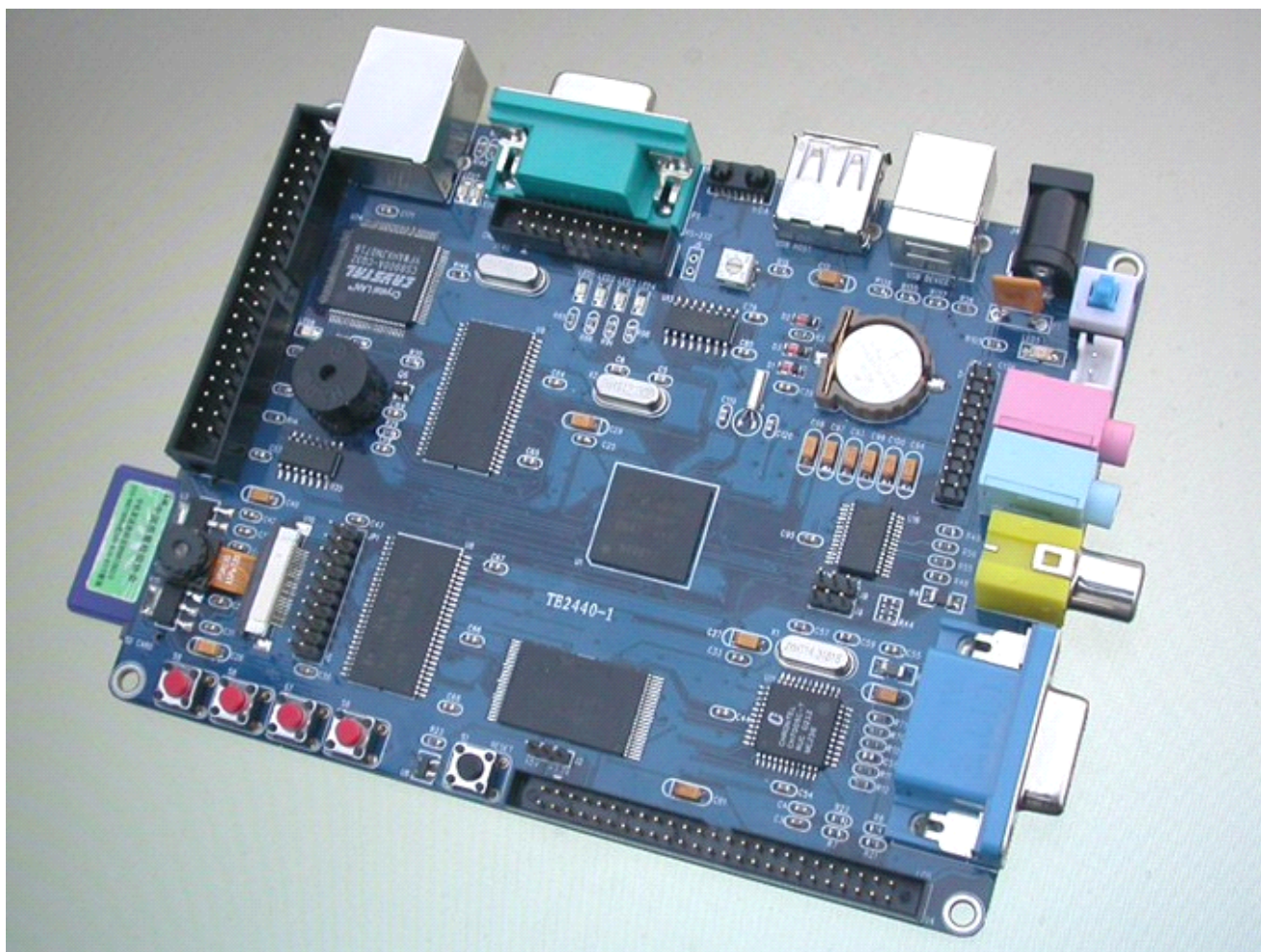
1.1 开发套件包含的内容

- 1) 一块已测试好的 TE2440 开发板
- 2) TE2440 用户光盘
- 3) 3.5" TFT Sumsang LCD 一块，带触摸屏(选配)
- 4) 一个 JTAG 板(带 20 芯排线)
- 5) 一条串口线(一头针一头孔，串口线)
- 6) 一条网线(交叉网线)
- 7) USB 线一条
- 8) 触摸笔一支(随 3.5" TFT Sumsang LCD 赠送)
- 9) 一个 5V 直流电源
- 10) 一个包装盒

1.2 用户光盘内容说明

1. TE2440 开发板用户手册(PDF)
2. 芯片 DATASHEET: 包含开发板上芯片，以及摄像头 ov9650 和 LCD 的 DATASHEET.
3. TE2440 原理图
4. windows 下的驱动: 包括并口和 USB
5. 实用工具: ADS 安装程序
 - H-JTAG 调试软件
 - Windows 下烧写 bootloader 的软件 SJF2440.EXE
 - DNW 及另一款串口调试软件
 - 以及其他一些常用工具
7. bootloader, wince, linux 参考用户光盘部分目录说明

1.3 TE2440 开发板外观



1.4 TE2440 开发板硬件资源

中央处理器

- CPU: 三星 S3C2440A, 主频 400MHz;

外部存储器

- 内存: 64M 字节;
- NAND Flash: 64M 字节 (K9F1208, 用户可自己更换为 16M、32M 或 128M 的 NandFlash)

串口

- 一个五线异步串行口, 波特率最高 115200bps
- 两个三线串口扩展引出。可同时使用三串口

网络接口

- 一个 10M 网口，采用 CS8900Q3，带联接和传输指示灯；

USB 接口

- 一个 USB1.1 HOST 接口；
- 一个 USB1.1 Device 接口；

音频接口

- 采用 IIS 接口芯片 UDA1341，一路立体声音频输出接口可接耳机或音箱；

存储接口

- 一个 SD 卡接口；
- 一个 IDE 接口可直接挂接硬盘，也可以做为总线扩展口；

LCD 和触摸屏接口

- 板上集成了 4 线电阻式触摸屏接口的相关电路；
- 一个 50 芯 LCD 接口引出了 LCD 控制器的全部信号；
- 标准配置为 256K 色 320x240/3.5 英寸 TFT 液晶屏，带触摸屏；
- 支持黑白、4 级灰度、16 级灰度、256 色、4096 色 STN 液晶屏；
- 板上引出一个 3.3V/5V 电源输出接口，可为更多款液晶提供电压支持；

摄像头接口

- 板上带有一个 2mm 间距的 20P 插座做为扩展，用户可使用此扩展口连接各种摄像头

时钟源

- 内部实时时钟（带有后备锂电池）；

复位电路

- 一个复位按键；并采用专用复位芯片进行复位，稳定可靠

调试及下载接口

- 一个 20 芯 Multi-ICE 标准 JTAG 接口，支持 SDT2.51，ADS1.2 等调试；

电源接口

- 5V 电源供电，带电源开关和指示灯；

其他

- 四个用户按键；
- 一个 PWM 控制蜂鸣器

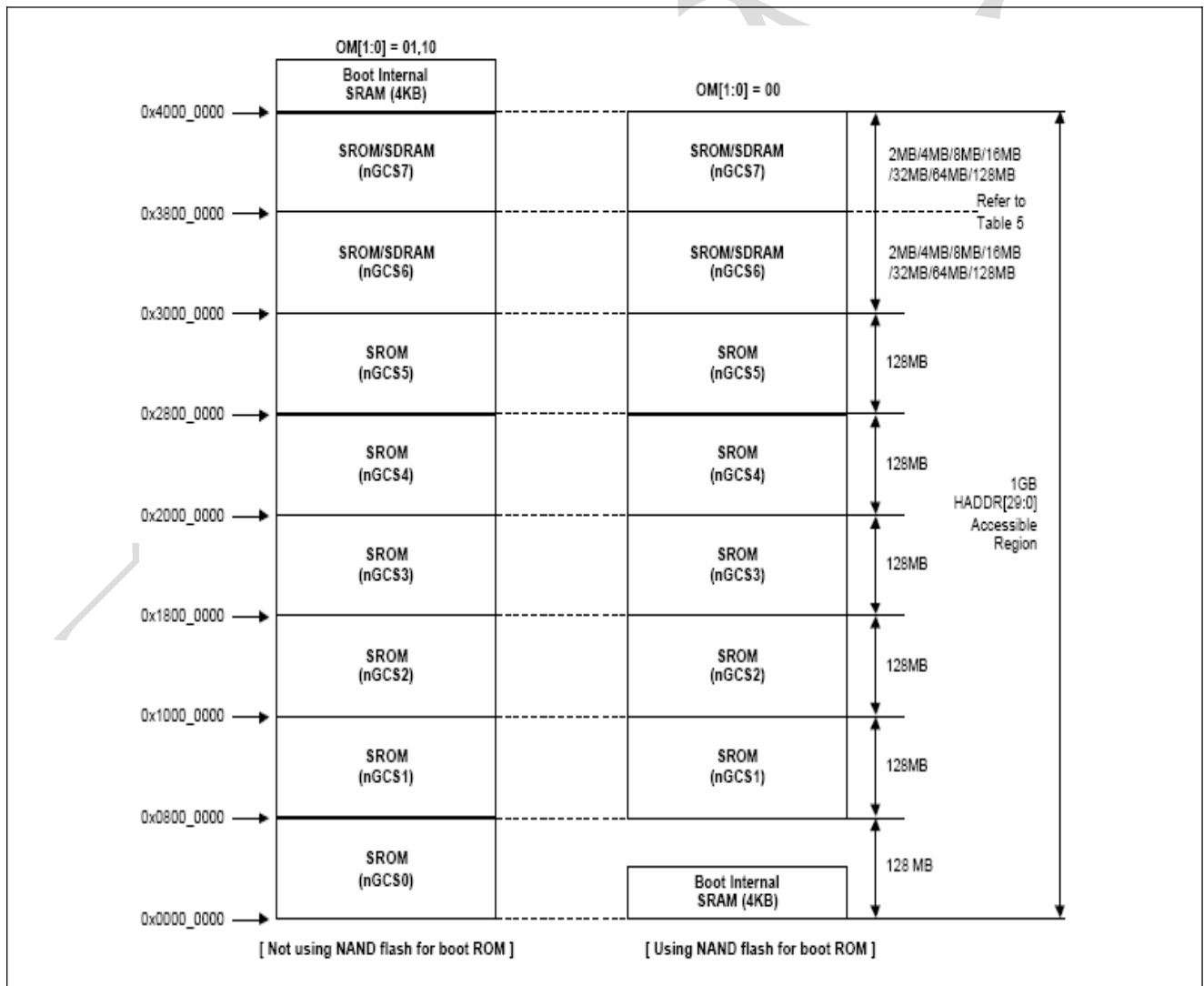
- 一个可调电阻接到 ADC 引脚上用来验证模数转换
- 一个红外收发头
- 扩展接口，引出 SPI/GPIO 及 I2C 等

1.5 硬件资源分配

1.5.1 地址空间分配以及片选信号定义

S3C2440 支持两种启动模式：一种是从 Nand FLASH 启动；一种是从 Nor FLASH 启动。TE2440 支持从 Nand FLASH 启动。

在这种启动模式下，各片选的存储空间分配是不同的，这两种启动模式的存储分配图如下：



a)图左是 nGCS0 片选的 Nor Flash 启动模式下的存储分配图；

b)图右是 NAND FLASH 启动模式下的存储分配图；说明：SFR Area 为特殊寄存器地址空间。

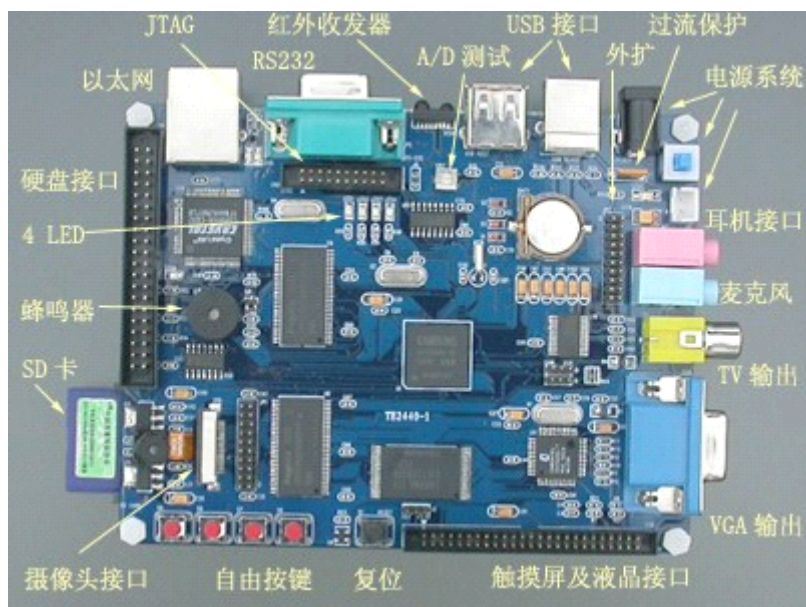
下面是器件地址空间分配和其片选定义 在进行器件地址说明之前，有一个点需要注意，nGCS0 片选的空间在不同的启动模式下，映射的器件是不一样的。由上图可以知道：

在 NAND FLASH 启动模式下，内部的 4K Bytes BootSRam 被映射到 nGCS0 片选 的空间。

在 Nor Flash 启动模式（非 NAND FLASH 启动模式）下，与 nGCS0 相连的外部存储器 Nor Flash 就被映射到 nGCS0 片选的空间

SDRAM 地址空间：0x30000000~0x34000000

1.5.2 开发板接口说明



名 称	说 明
J1	音频输出接口（PHONE）
J3	音频输入接口（MIC）
U4	LCD/触摸屏接口
JP1	摄像头模块接口
CN2	20 针扩展口，标准的 JTAG 接口
J4	电源插孔
JP2	GPI0 扩展口

IDE	硬盘光驱接口
-----	--------

1.5.3 按键说明

名 称	说 明
S1 (RESET)	复位按键(黑色)
S6, S7, S8, S9	4 个用户按键(红色)

1.5.4 LED 指示灯说明

名 称	说 明
LED1, LED2, LED3, LED4	I/O 口指示灯
LED5	5V 电源指示灯
LED6, LED7	网络指示灯
LED8	硬盘指示灯

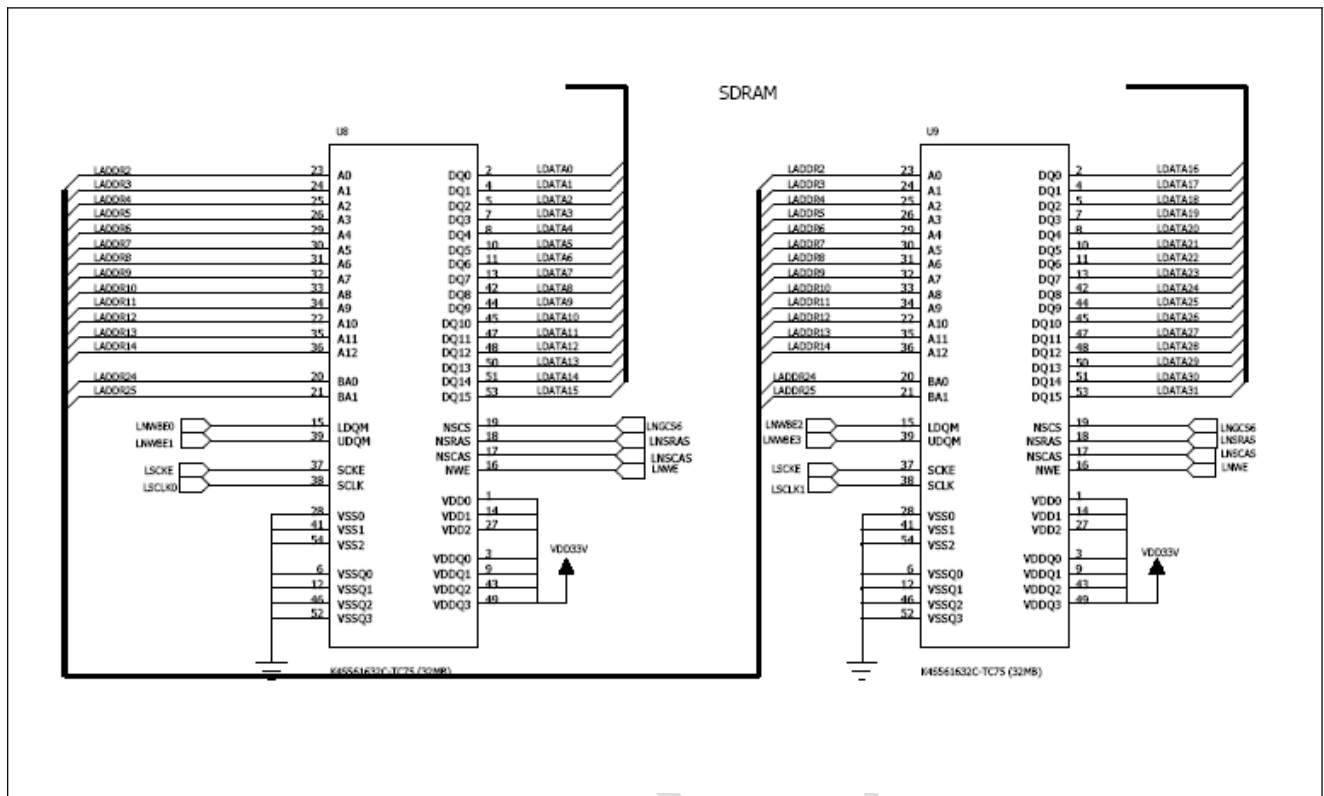
1.5.5 跳线分配表

名 称	说 明
J2	跳线帽接在 1, 2 脚, LCD 工作在 5V 跳线帽接在 2, 3 脚, LCD 工作在 3.3V (默认)
J6	红外 (接上跳线帽后红外工作)

1.6 TE2440 开发板主要硬件说明

1.6.1 系统存储器

TE2440 开发板上存储器包括 SDRAM 和 FLASH, 系统配置了两片 32M 的三星 SDRAM, 一片 nand flash (64M), 开发板采用两片半字 (half-word) SDRAM 器件共同组成一个 32 位数据宽度的 SDRAM 系统, 提高了其与 CPU 的通信效率。更好的发挥 S3C2440A 芯片的潜能。



1.6.2 JTAG 及复位逻辑

1.JTAG 介绍

JTAG (Joint Test Action Group:联合测试行动小组) 是 1985 年制定的检测 PCB 和 IC 芯片的一个标准, 1990 年被修改后成为 IEEE 的一个标准, 即 IEEE1149.1-1990。通过这个标准, 可对具有 JTAG 接口的芯片的硬件电路进行边界扫描和故障检测。具有 JTAG 接口的芯片, 相关 JTAG 引脚的定义为: **TCK** 为测试时钟输入; **TDI** 为测试数据输入, 数据通过 TDI 引脚输入 JTAG 接口; **TDO** 为测试数据输出, 数据通过 TDO 引脚从 JTAG 接口输出; **TMS** 为测试模式选择, TMS 用来设置 JTAG 接口处于某种特定的测试模式; **TRST** 为测试复位, 输入引脚, 低电平有效。 **GND** 为地。

JTAG 最初是用来对芯片进行测试的, 基本原理是在器件内部定义一个 TAP (Test Access Port;测试访问口) 通过专用的 JTAG 测试工具对内部节点进行测试。JTAG 测试允许多个器件通过 JTAG 接口串联在一起, 形成一个 JTAG 链, 能实现对各个器件分别测试。现在, JTAG 接口还常用于实现 ISP (In-System programmable;在线编程), 对 FLASH 等器件进行编程。

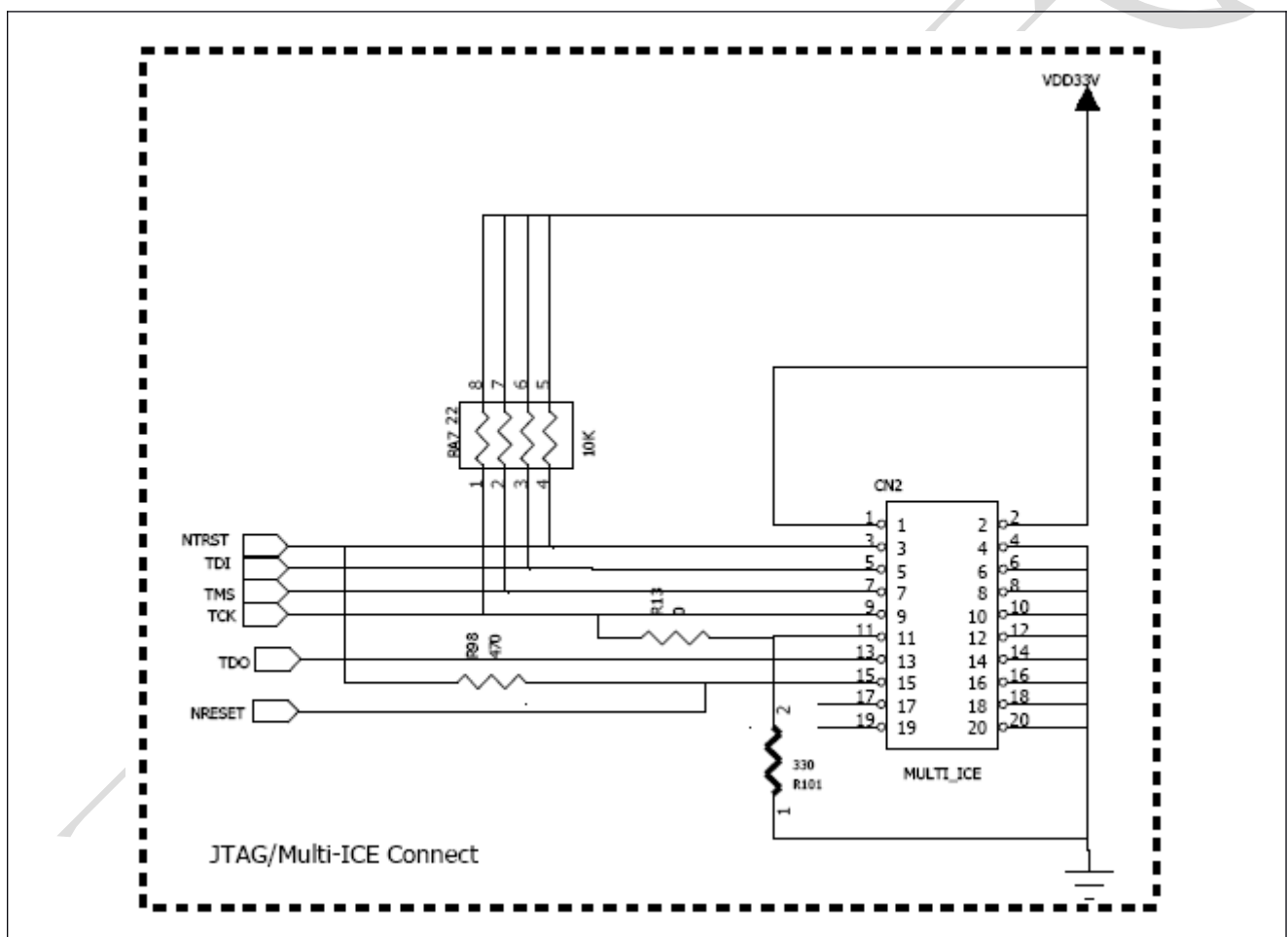
JTAG 编程方式是在线编程, 传统生产流程中先对芯片进行预编程现再装到板上因此而改

变，简化的流程为先固定器件到电路板上，再用 JTAG 编程，从而大大加快工程进度。JTAG 接口可对 PSD 芯片内部的所有部件进行编程

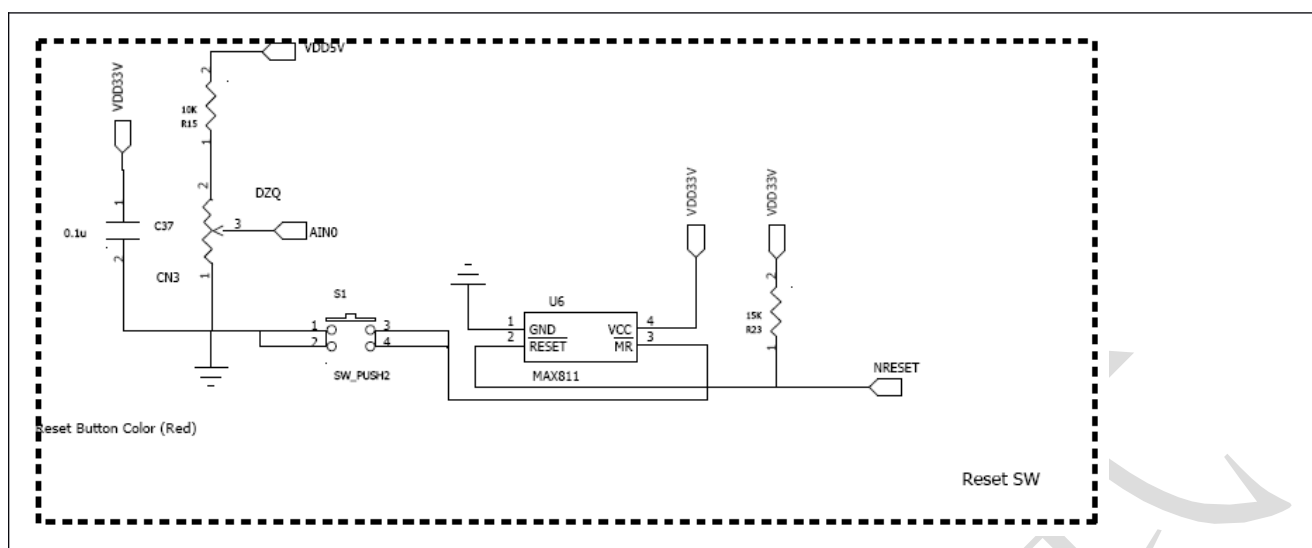
2. 开发板上的 JTAG

开发板的JTAG接口支持flash下载和仿真功能, 电路图如下。系统的复位信号nRESET必须保持低电平4个时钟周期才能保证可靠复位，一旦外部复位信号被CPU确认，它将使CPU内部复位信号变低电平，并且保持128个时钟周期。

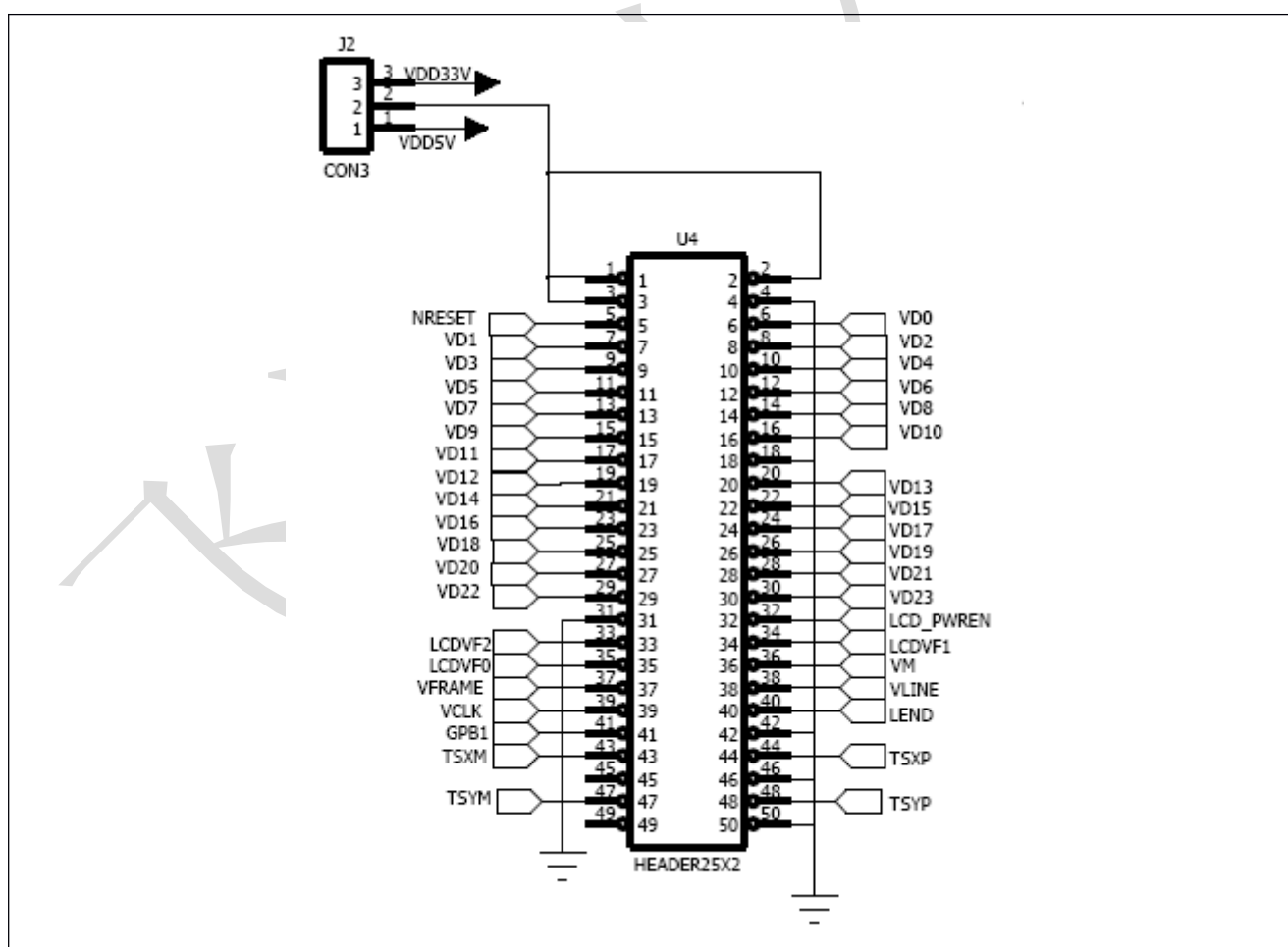
JTAG 电路：20 线标准接口，具备调试功能



复位电路:采用专业复位芯片复位，稳定可靠

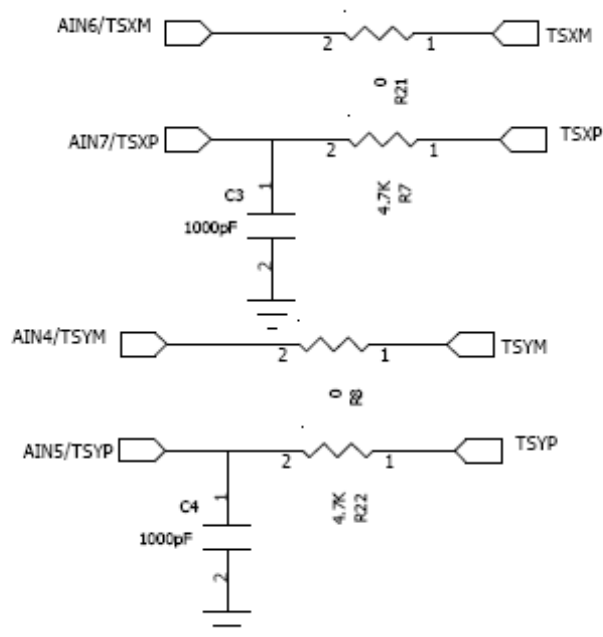


1.6.3 LCD/触摸屏接口引脚定义

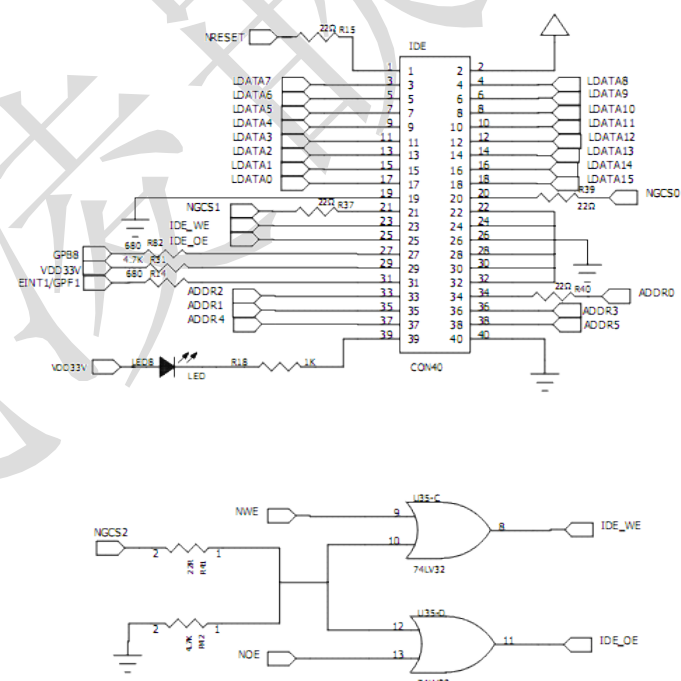


引脚	功能	引脚	功能
1	VCC	2	VCC
3	VCC	4	GND
5	NRESET	6	VD0
7	VD1	8	VD2
9	VD3	10	VD4
11	VD5	12	VD6
13	VD7	14	VD8
15	VD9	16	VD10
17	VD11	18	GND
19	VD12	20	VD13
21	VD14	22	VD15
23	VD16	24	VD17
25	VD18	26	VD19
27	VD20	28	VD21
29	VD22	30	VD23
31	GND	32	LCD_POWE
33	LCDVF2	34	LCDVF1
35	LCDVF0	36	VM
37	VFRAME	38	VLINE
39	VCLK	40	LEND
41	GPB1	42	GND
43	TSXM	44	TSXP
45	NC	46	GND
47	TSYM	48	TSYP
49	NC	50	GND

LCD控制信号连接匹配电阻，可以实现更远距离传送数据。其中，TSXP，TSXM 用于连接标准四线电阻式触摸屏的 X 组引脚，TSYP，TSYM 用于连接 Y 组引脚。

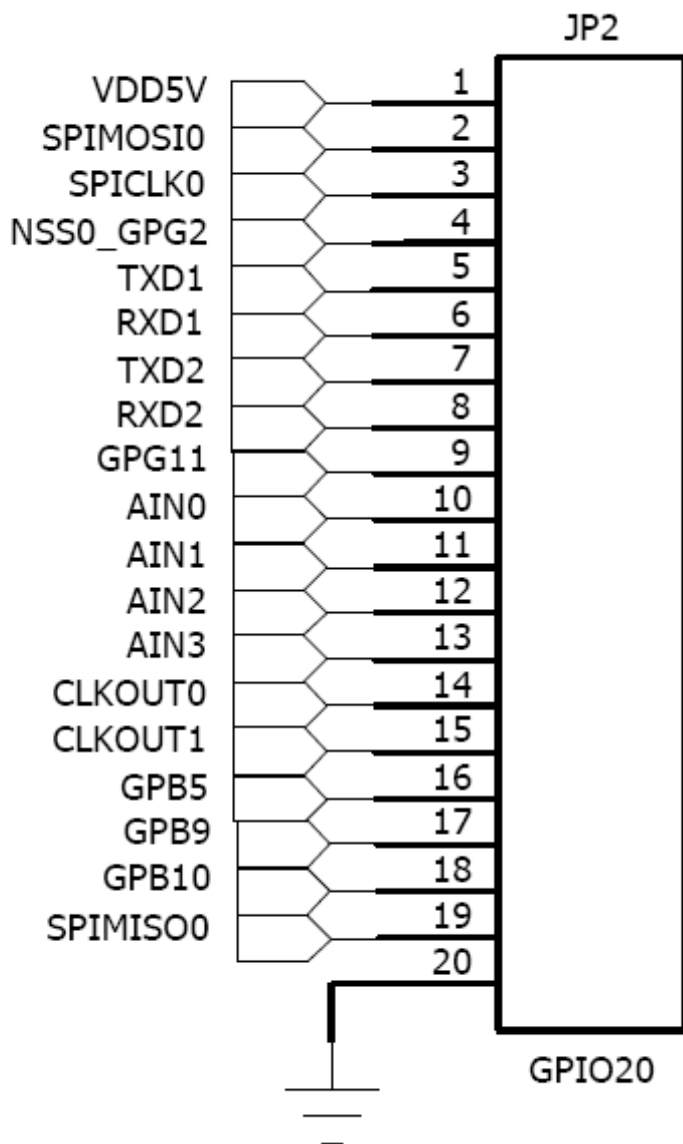


1.6.4 IDE （也作为总线接口） 接口引脚定义



引脚	功能	引脚	功能
1	NRESET	2	GND
3	LDATA7	4	LDATA8
5	LDATA6	6	LDATA9
7	LDATA5	8	LDATA10
9	LDATA4	10	LDATA11
11	LDATA3	12	LDATA12
13	LDATA2	14	LDATA13
15	LDATA1	16	LDATA14
17	LDATA0	18	LDATA15
19	GND	20	可接NGCS0默认NC
21	可接NGCS0默认NC	22	GND
23	IDE_WE	24	GND
25	IDE_OE	26	GND
27	GPB8	28	GND
29	VDD5V	30	GND
31	EINT1	32	GND
33	ADDR2	34	可接ADDR0默认NC
35	ADDR1	36	ADDR3
37	ADDR4	38	ADDR5
39	VDD	40	GND

1.6.5 GPIO 扩展口引脚定义

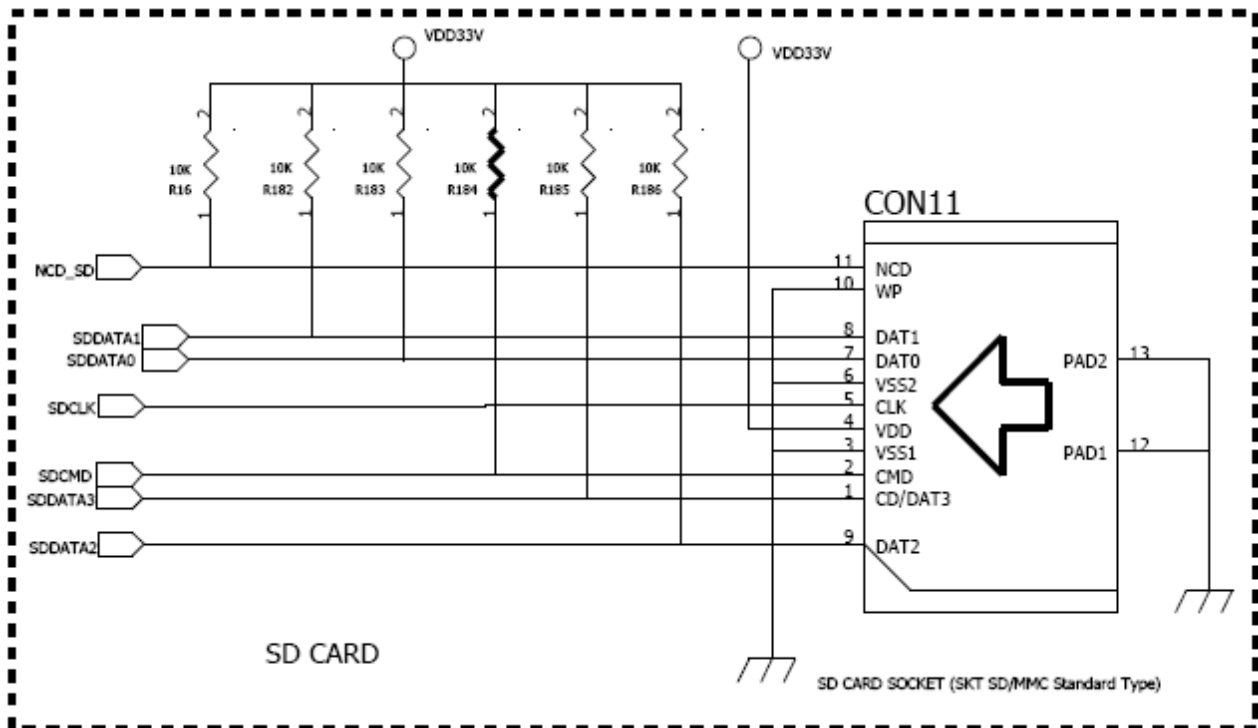


引脚	功能	引脚	功能
1	GPE11	2	GPE12
3	GPE13	4	GPG2/EINT10
5	GPG3/EINT11	6	GPG5/EINT13
7	GPG6/EINT14	8	GPG7/EINT15
9	GPG11/EINT19	10	GPG12/EINT20
11	GPG13/EINT21	12	GPG14/EINT22
13	GPG15/EINT23	14	CLKOUT0
15	CLKOUT1	16	GPB5

17	GPB9	18	GPB10
19	NRSTOUT/GPA21	20	GND

1.6.6 SD 卡接口

SD 卡电路图如下



支持2GB大容量SD卡插入（支持热插拔）。

关于 SD 卡：

SD 卡（Secure Digital Memory Card）中文翻译为安全数码卡，是一种基于半导体快闪记忆器的新一代记忆设备，它被广泛地于便携式装置上使用，例如数码相机、个人数码助理(PDA)和多媒体播放器等。

SD 卡由日本松下、东芝及美国 SanDisk 公司于 1999 年 8 月共同开发研制。大小犹如一张邮票的 SD 记忆卡，重量只有 2 克，但却拥有高记忆容量、快速数据传输率、极大的移动灵活性以及很好的安全性。

SD 卡在 24mm×32mm×2.1mm 的体积内结合了 SanDisk 快闪记忆卡控制与 MLC（Multilevel Cell）技术和 Toshiba（东芝）0.16u 及 0.13u 的 NAND 技术，通过 9 针的接口界面与专门的驱动器相连接，不需要额外的电源来保持其上记忆的信息。而且它是一体化固体介质，没有任何移动部分，所以不用担心机械运动的损坏。

SD 卡的技术建是基于 MultiMedia 卡（MMC）格式上发展而来，大小和 MMC 差不多，尺寸为 32mm x

24mm x 2.1mm。长宽和 MMC 一样，只是比 MMC 厚了 0.7mm，以容纳更大容量的存储单元。SD 卡与 MMC 卡保持着向上兼容，也就是说，MMC 可以被新的 SD 设备存取，兼容性则取决于应用软件，但 SD 卡却不可以被 MMC 设备存取。（SD 卡外型采用了与 MMC 厚度一样的导轨式设计，以使 SD 设备可以适合 MMC）。

SD 接口除了保留 MMC 的 7 针外，还在两边加多了 2 针，作为数据线。采用了 NAND 型 Flash Memory，基本上和 SmartMedia 的一样，平均数据传输率能达到 2MB/s。

SD 卡共支持三种传输模式：SPI 模式（独立序列输入和序列输出），1 位 SD 模式（独立指令和数据通道，独有的传输格式），4 位 SD 模式（使用额外的针脚以及某些重新设置的针脚。支持四位宽的并行传输）

目前市场上 SD 卡的品牌很多，诸如：SANDISK，Kingmax，松下和 Kingston。

- SANDISK 产的 SD 卡，是市面上最常见的，分为高速和低速 SD 卡。

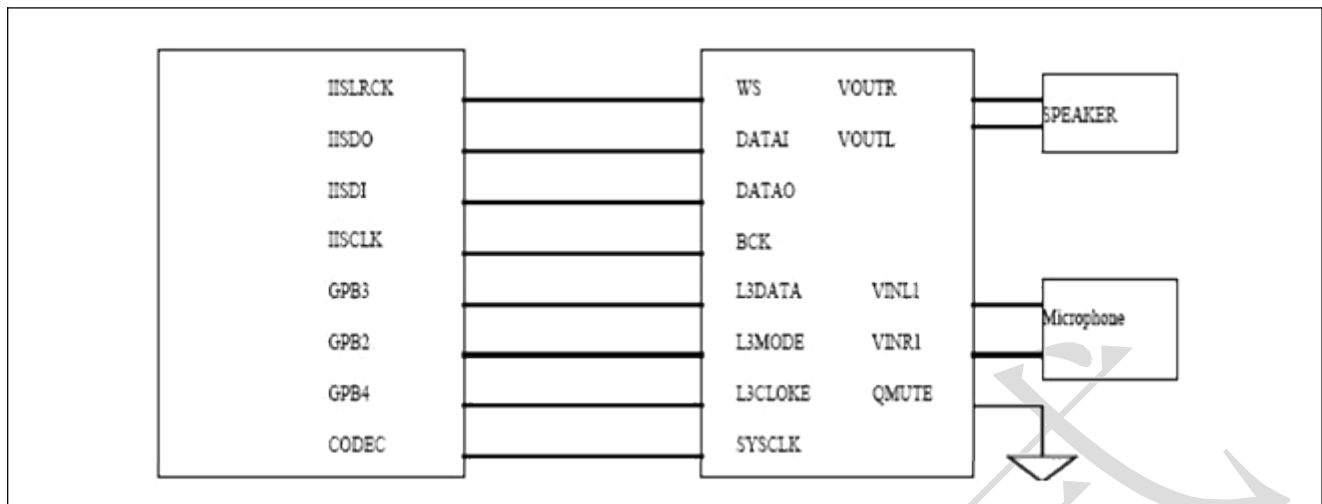
- Kingmax 的 SD 卡，采用了独特的一体化封装技术（PIP），使得造假者很难仿制，Kingmax SD 卡最高传输速率 10MB/秒，具有防水、防震性能，防压的三防设计，它可以满足野外拍摄各种要求。

- 松下 SD 卡，作为 SD 标准的缔造者，其技术可以说是市面上最好的 SD 卡之一了。不过需要注意的是松下 SD 多数没有保修，购买时一定要问清楚质保期限这个重要问题。

- Kingston SD 卡，在众多的闪存类产品中，是体积最小的一种，提供了长达 5 年的质保时间。

1.6.7 IIS 音频输入输出接口

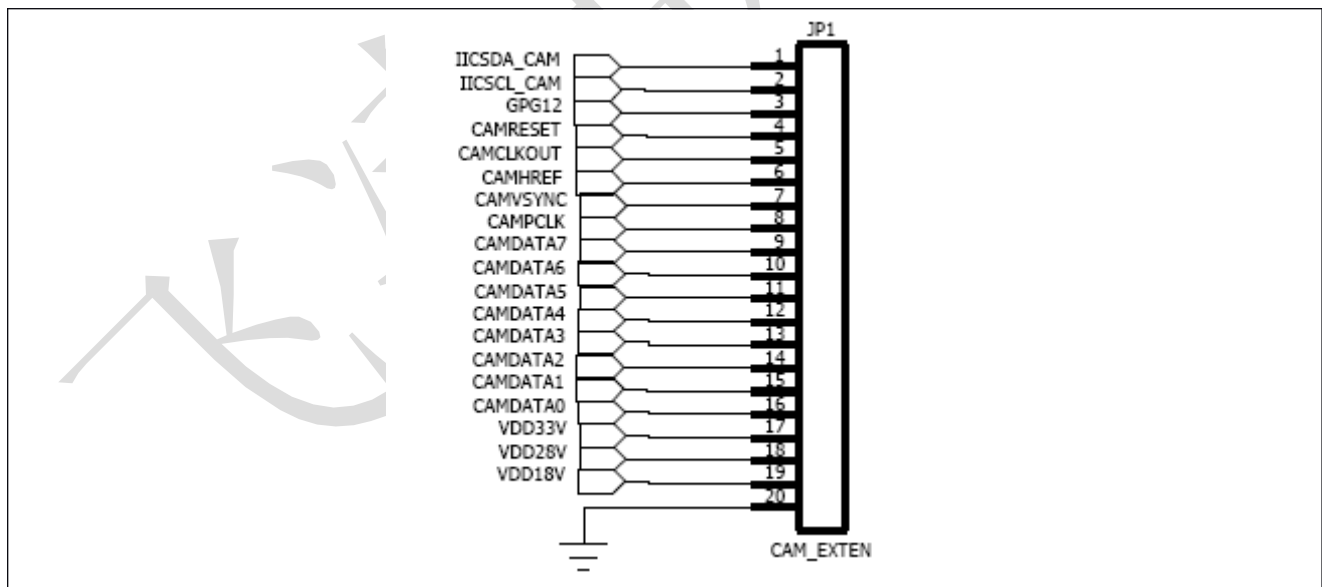
IIS(Inter-IC Sound bus) 又称 I2S 即音频数据接口，它是 SONY、PHILIPS 等电子巨头共同推出的串行数字音频总线协议。如图 3-8 IIS 接口电路所示，本系统把 IIS 接口与 PHILIPS 的 UDA1341TS 音频数字信号编译码器相连接，得到 MICROPHONE 音频输入通道和 SPEADER 音频输出通道。UDA1341TS 可把立体声模拟信号转化为数字信号，同样也能把数字信号转换成模拟信号，并可用 PGA（可编程增益控制），AGC（自动增益控制）对模拟信号进行处理；对于数字信号，该芯片提供了 DSP（数字音频处理）功能。在实际中，UDA1341TS 可广泛应用于 MD、CD、notebook、PC 和数码摄影机等。S3C2440X01 的 IIS 口可与 UDA1341TS 的 BCK、WS、DATAI、SYSCLK 相连。对于 UDA1341TS 的 L3 总线，它是该芯片工作于微控制器输入模式时使用的，它包括 L3DATA、L3MODE、L3CLOCK 共三根接线，它们分别表示为微处理器接口数据线、微处理器接口模式线，微处理器接口时钟线。通过这个接口，微处理器能够对 UDA1341TS 中的数字音频处理参数和系统控制参数进行配置



1.6.8 摄像头接口:

板上带有一个 2mm 间距的 20P 插座做为扩展，用户可使用此扩展口连接其它种摄像头。若为 USB 摄像头则可直接插在 USB 接口上。

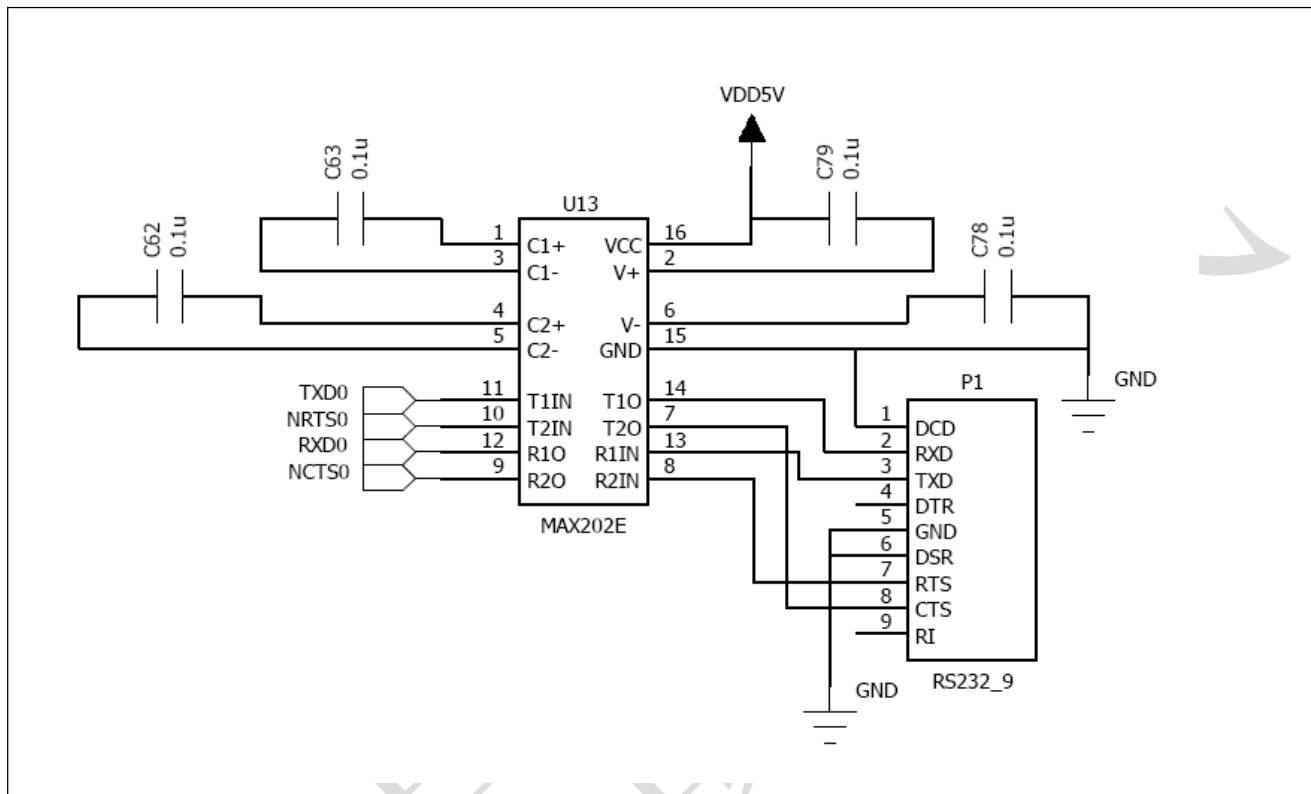
20P 插座电路如下:



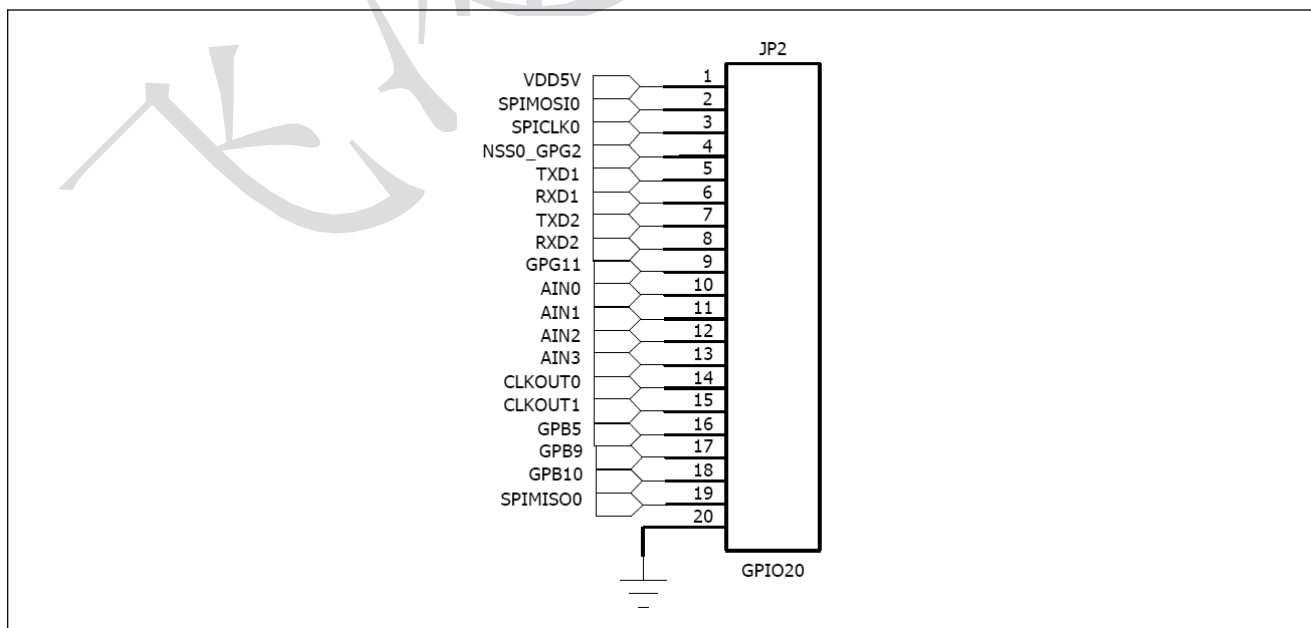
1.6.9 串口电路

TE2440 开发板支持三个串口, 串口 0, 串口 1, 串口 2

串口 0 的电路如下:



串口 1 和串口 2 由 JP2 扩展引出, 接口如下:

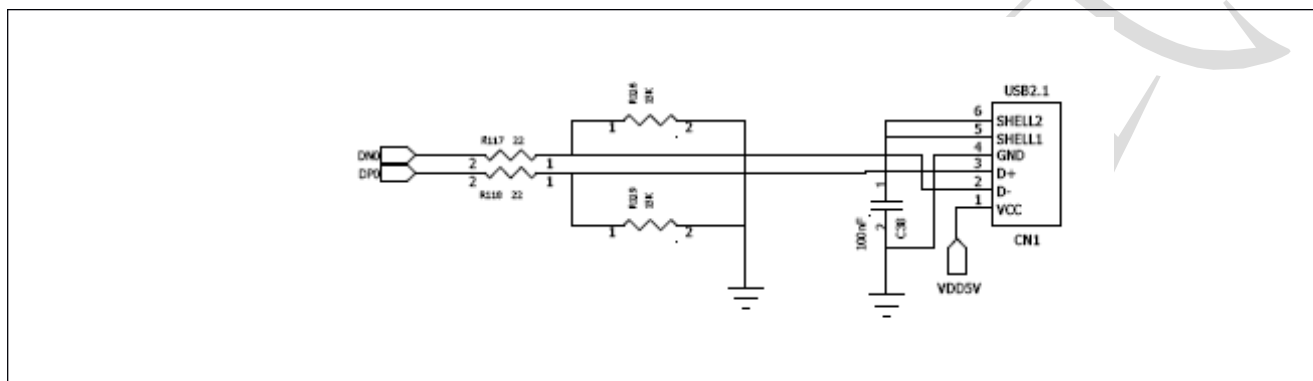


1.6.10 USB 接口

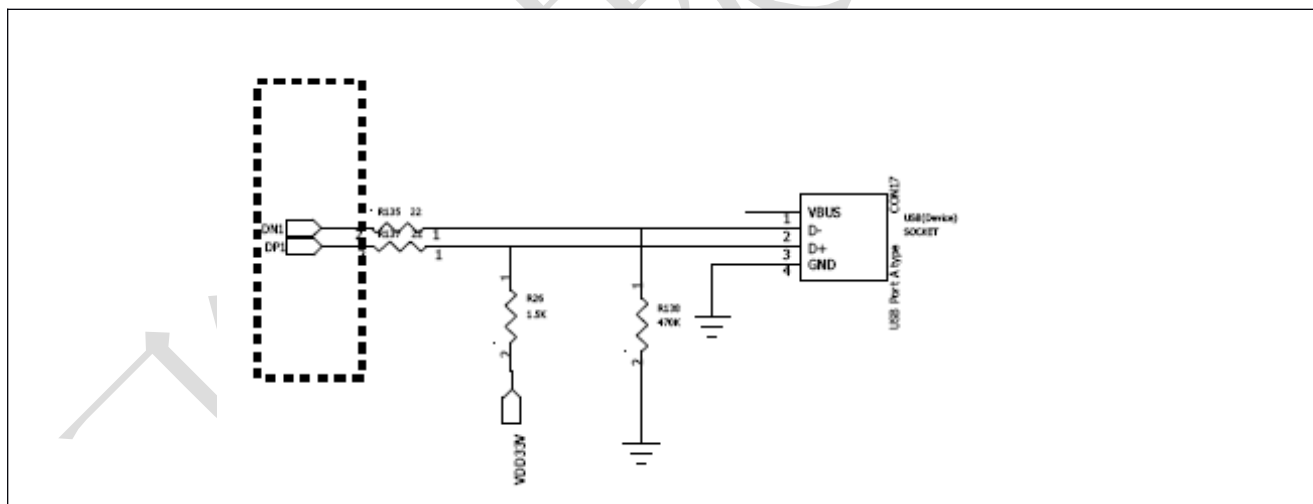
USB 是英文 Universal Serial BUS 的缩写,中文含义是“通用串行总线”。它不是一种新的总线标准,而是应用在 PC 领域的接口技术。USB 是在 1994 年底由英特尔、康柏、IBM、Microsoft 等多家公司联合提出的。

什么是 USB HOST 和 USB DEVICE 呢,举个例子:U 盘和电脑,U 盘是 USB DEVICE 而电脑就是 USB HOST。

USB HOST



USB DEVICE



1.6.11 功能按键及用户 LED 指示灯

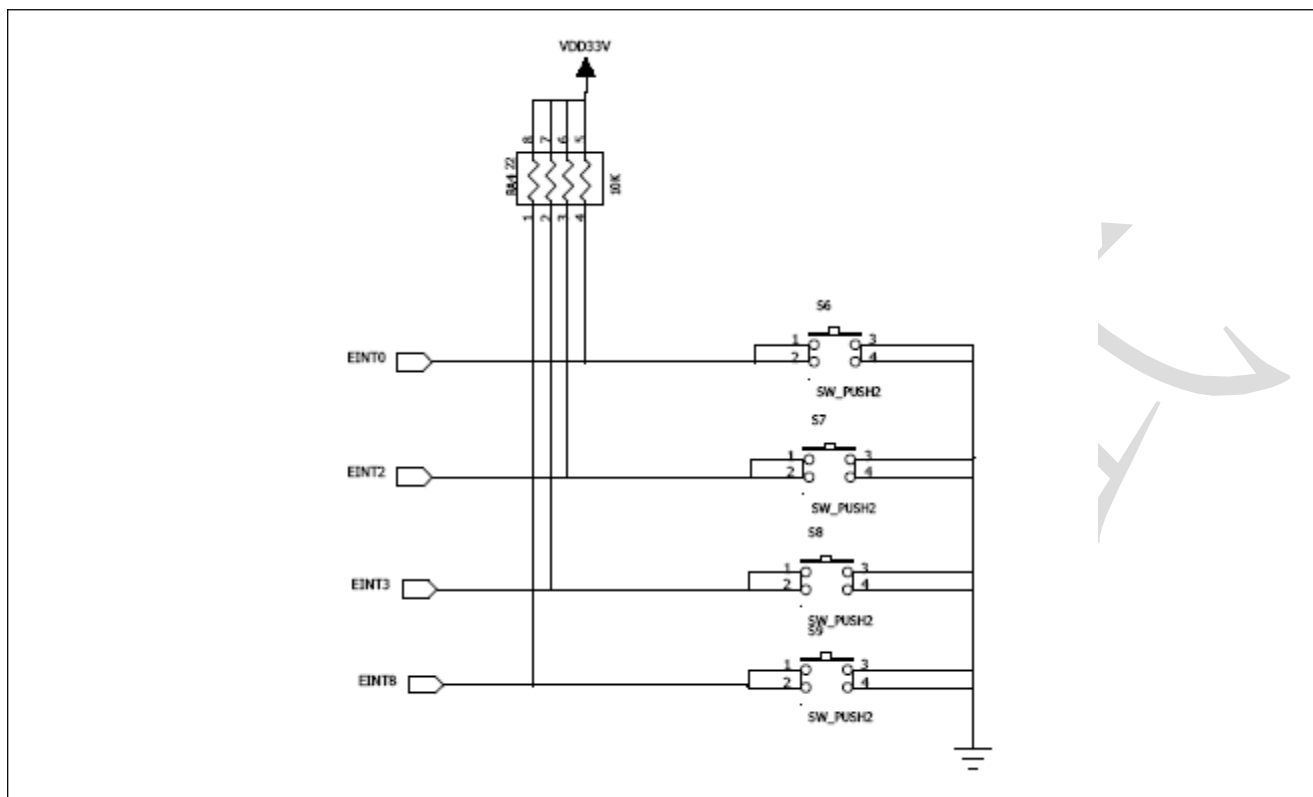
开发板提供了四个用户按键,可供用户设计程序时使用,同时还提供由四个I/O口扩展的LED指示灯,他们占用CPU资源如下所示:

按键:

S6、S7、S8、S9 四个按键用到以下四个I/O口

EINT0/GPF0 EINT2/GPF2 EINT3/GPF3 EINT8/GPG0

按键 电路如下：



LED:

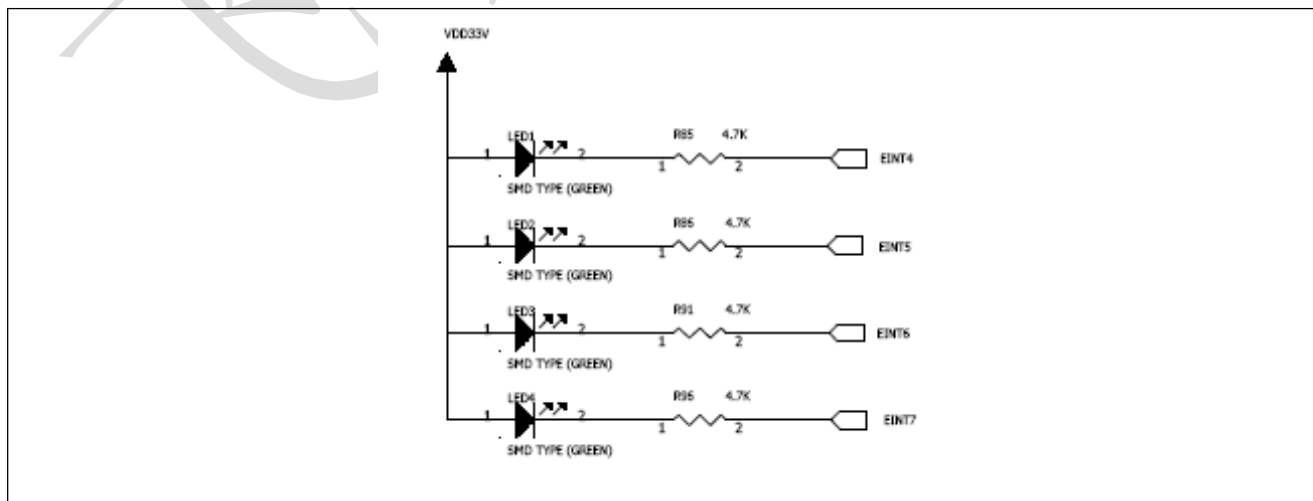
LED1 EINT4/GPF4

LED2 EINT5/GPF5

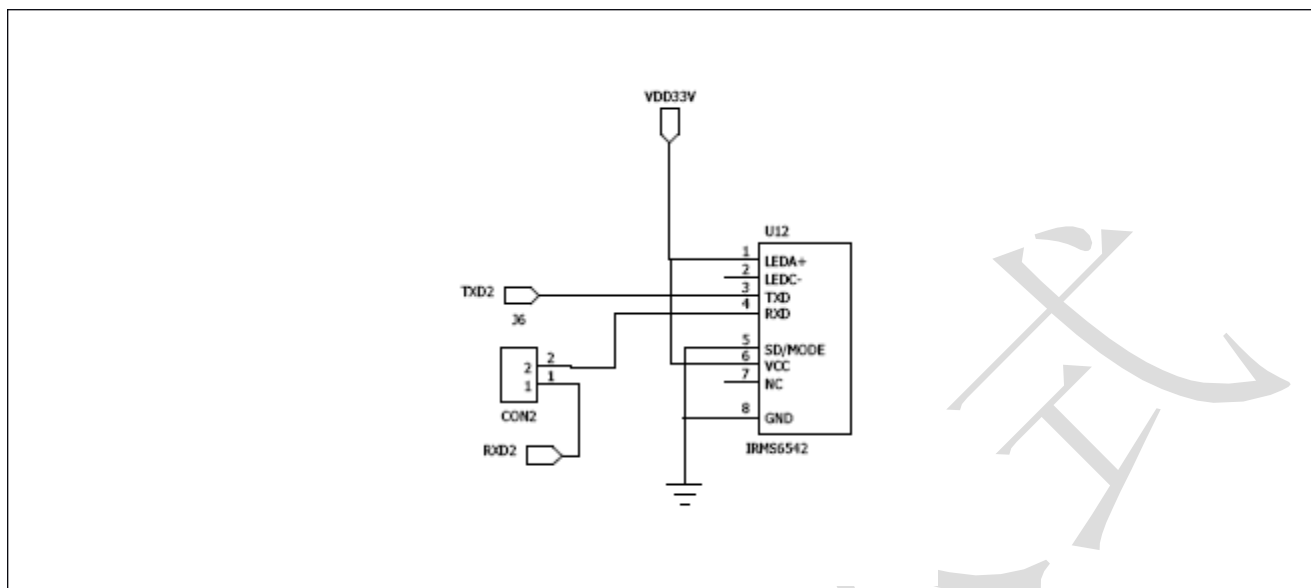
LED3 EINT6/GPF6

LED4 EINT7/GPF7

LED指示灯电路如下：



1.6.12 红外收发电路



1.7 TE2440 支持的操作系统及其驱动

TE2440 开发板支持 WINCE 和 Linux 嵌入操作系统。

1.7.1 Linux 操作系统

—— 内核: Linux2. 6.12

—— 文件系统: cramfs/ramfs/etx2/fat32/nfs/yaffs/jeffs2

—— 驱动支持: 系统中断和系统时钟驱动

串口驱动(Serial device driver)

块设备驱动(包括 IDE 硬盘, SD 卡)

Nand Flash 驱动

网卡驱动

实时时钟驱动

USB Host 驱动

液晶(LCD)驱动, 触摸屏驱动

—— 常用命令: cat, chmod, discard, echo, flashfsd, flashwrite, free, genhtml, hostname, init,

kill, loader, ls, mkdir, mount, ps, reboot, rm, smanaged, sysconf, yes, insmod, lsmod, rmmod

—— 图形界面：支持 MIZI PDA demo, QT opia, microwindows, minigui 等。其中中文 QTopia, 提供源代码。

—— 常用 USB 设备：优盘，移动硬盘，USB 鼠标，键盘

—— 常用服务： web server;ftp;telnet

—— 常用应用程序：MP3 播放器等

—— 网络协议及网络应用程序：

TCP/IP：完整的 TCP/IP 协议

1.7.2 WINCE 操作系统

—— 内核： WINCE 5. 0

—— 驱动支持：系统中断和系统时钟驱动

串口驱动 (Serial device driver)

块设备驱动 (SD 卡)

Nand Flash 驱动

以太网卡驱动 (CS8000A)

实时时钟驱动

USB Host 驱动

USB Slave 驱动

LEDS 灯驱动

液晶 (LCD) 驱动，触摸屏驱动

—— 网络协议：标准 IPV4 协议。包括比如 TCP/IP, UDP, SMTP 等

—— 浏览器：IE

—— 媒体播放：MediaPlayer

—— 显示：可在 LCD 上输出完美图形界面

—— 其他功能：标准 WINCE 功能。可以增删

—— BSP 源代码：提供。

二. 第二章 TE2440 开发板基本使用

2.1 TE2440 外部硬件连接

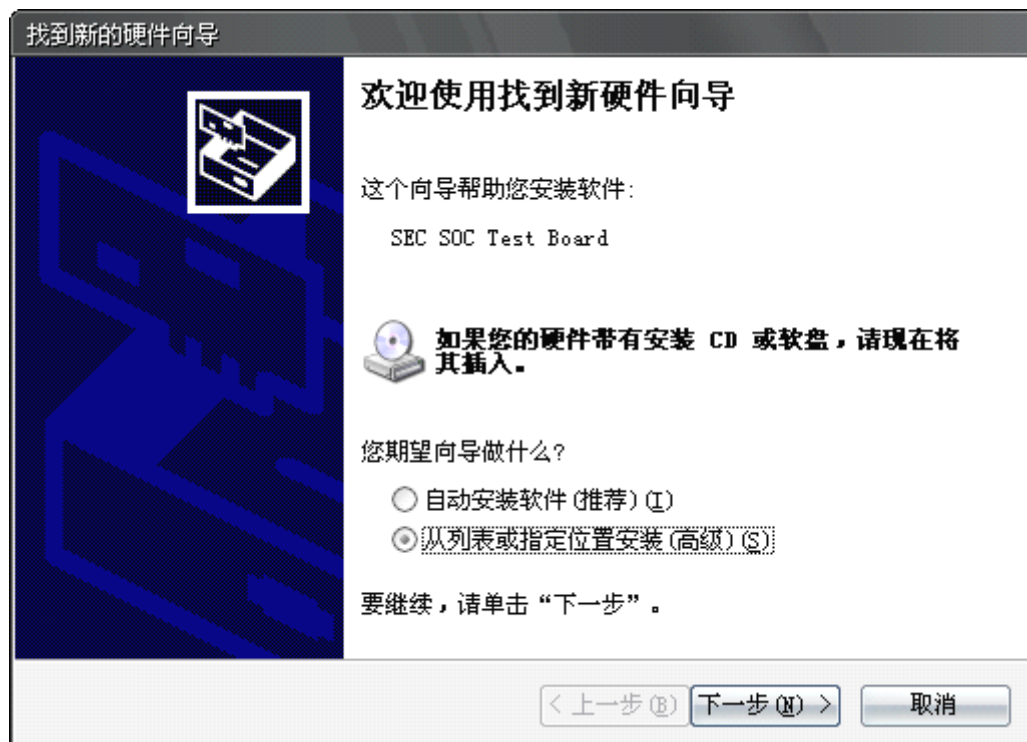
1. 把我们提供的 5 V 电源适配器连接到开发板 5V 输入插孔上
——板子为 5V 供电
2. 用串口连接线（开发套件中提供）将开发板上的串口与 PC 机串口相连
——当使用 DNW、minicom、超级终端与板子通信时，这些软件上的信息是通过串口打印出来的
——也可以通过串口来烧写 flash，只是速度慢些
3. 用 USB 连接线将开发板与 PC 机相连（开发板接 USB DEVICE 口）
——在烧写 flash 的时候，可以通过 usb 下载，速度要比串口快得多
4. 用交叉网线（一头为 A 型接法一头为 B 型接法）将 ETHERNET 与 PC 机相连
——在挂载网络文件系统 NFS 及练习网络编程时使用
5. 如果有液晶屏连接上液晶屏 (LCD) 和 J2 跳线（根据自己的液晶屏，选择 3.3v 或 5v, 我们提供的 3.5 寸液晶屏为 3.3V）
6. 音箱连接到音频输出接口(蓝色)

2.2 windows 下驱动的安装

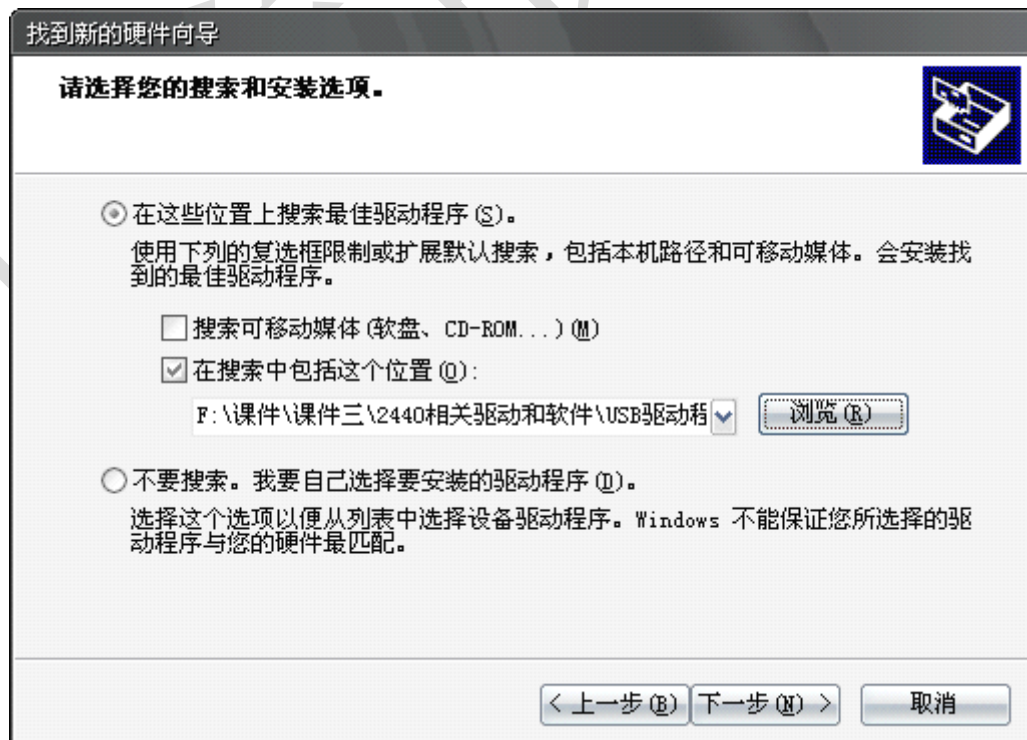
2.2.1 安装 USB 驱动

先用 USB 线将 PC 机和开发板的 USB DEVICE 口相连，打开 DNW(或超级终端)，打开电源后串口会有输出信息，按键盘（PC 端）任意键进入 BOOTLOADER 菜单。这时 PC 端系统会提示找到新硬件，按以下步骤安装好 USB 驱动：

- 1、出现以下提示，选择“从列表或指定位置安装……”

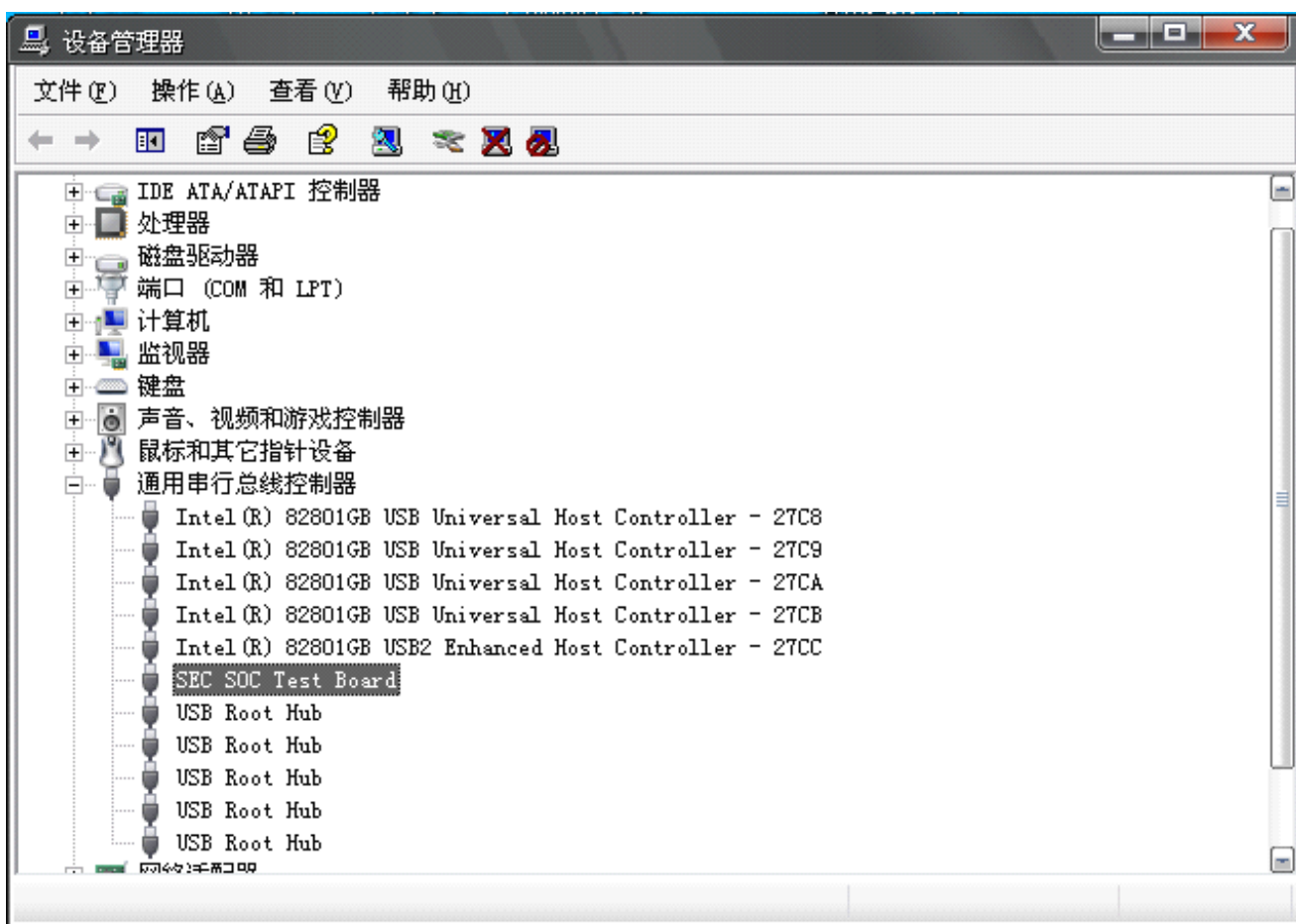


2、点击下一步，选择 USB 驱动的路径，如在 F:\课件\课件三\2440 相关驱动和软件\USB 驱动程序，点击下一步。



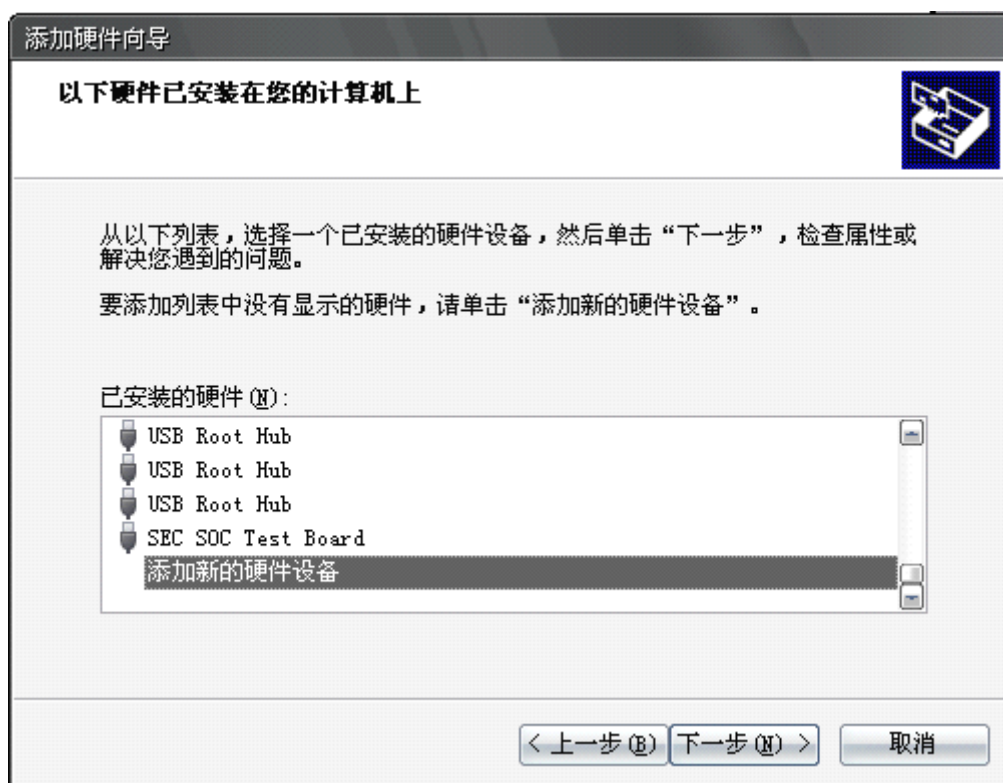


3、安装好 USB 驱动后可以看到在设备管理器中的“通用串行总线控制器”下多了“SEC SOC Test Board ”一项：

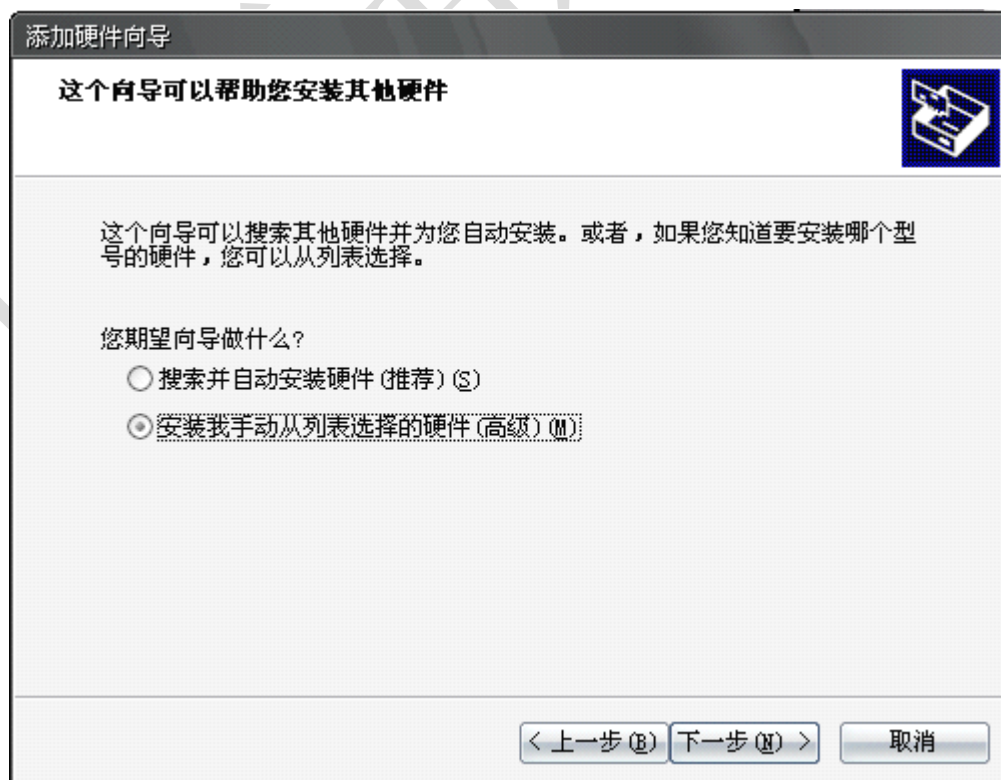


2.2.2 安装并口驱动程序

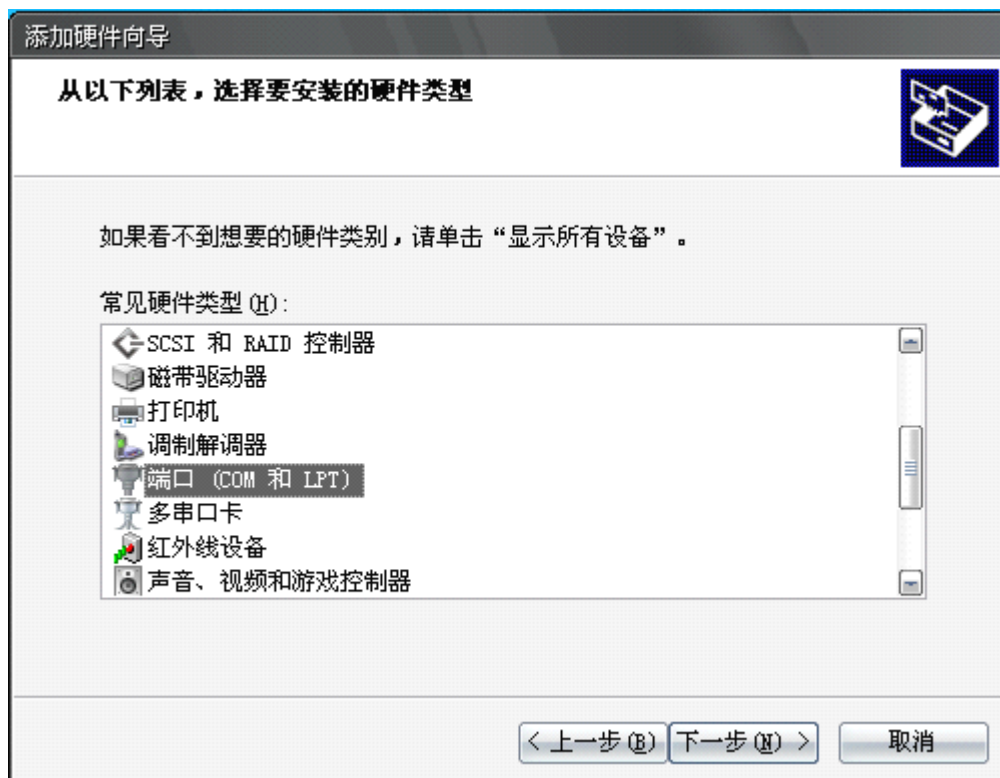
首先将光盘中“TE2440\windows下驱动\GIVEIO”文件夹中的“GIVEIO.SYS”拷贝到 WINDOWS的系统驱动目录下（笔者的机器为C:\WINNT\system32\drivers”）。在WINDOWS的“控制面板”中选择“添加硬件”。



选择“添加新的硬件设备”，点击“下一步”，



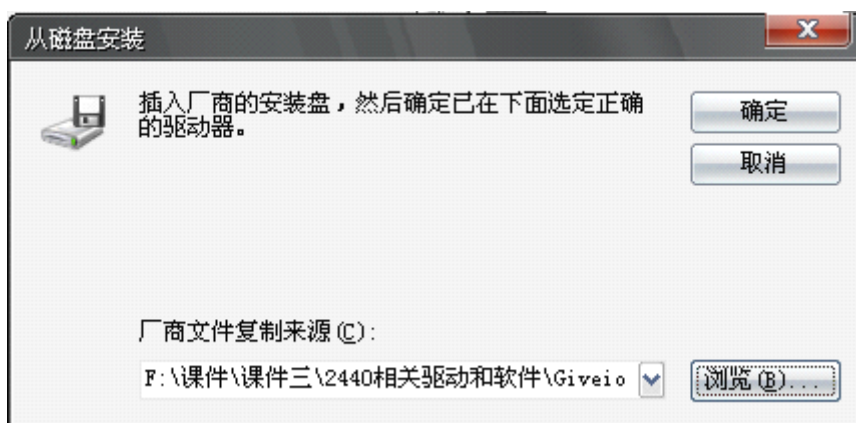
选择“安装我手动从列表选择的硬件”，点击“下一步”



选择“端口”，点击“下一步”



选择“从磁盘安装”

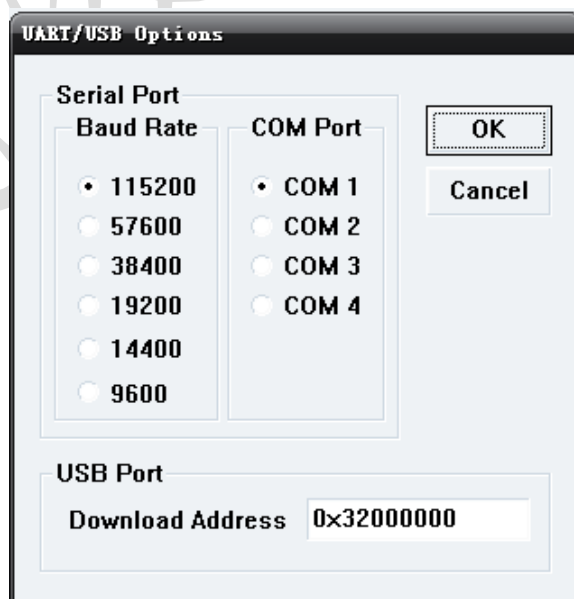


在“厂商文件复制来源”中选择光盘中 GIVEIO.inf 所在的目录，点击“确定”接下来的步骤一直选择“下一步，”最后 GIVEIO 将成功安装到系统中。

2.3 调试终端使用

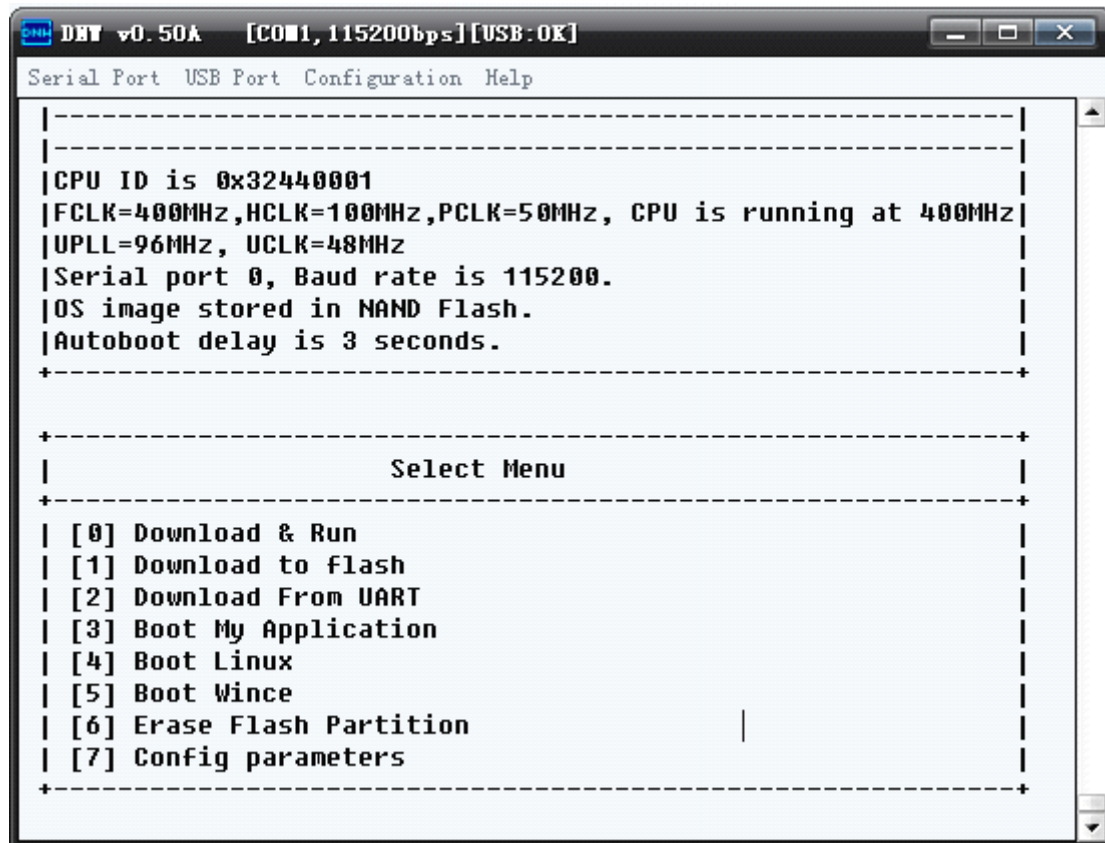
2.3.1 DWN 软件的使用

DWN 软件是三星公司为 S3C2440 芯片配置的一款专用软件，可以在 windows 下通过 USB 烧写 Flash。在 PC 机上打开 DNW(v0.50A)，在 configuration 中设置为：COM1，波特率为 115200，Download Address 设置为 0x32000000(此参数为下载到内存的地址：0x30000000~0x34000000，bootloader 用了 0x30300000 之前的内存地址，注意不要与之冲突)。具体配置如图所示：



配置完毕之后，点击 Serial Port 下的 connect，在软件的上边即会显示[COM1, 115200bps]。

插好 USB，给板子上电，在 COM1 后边会跟着显示[USB:OK], 若没有显示可将 USB 重插。（前提是已安装 USB 驱动程序）。



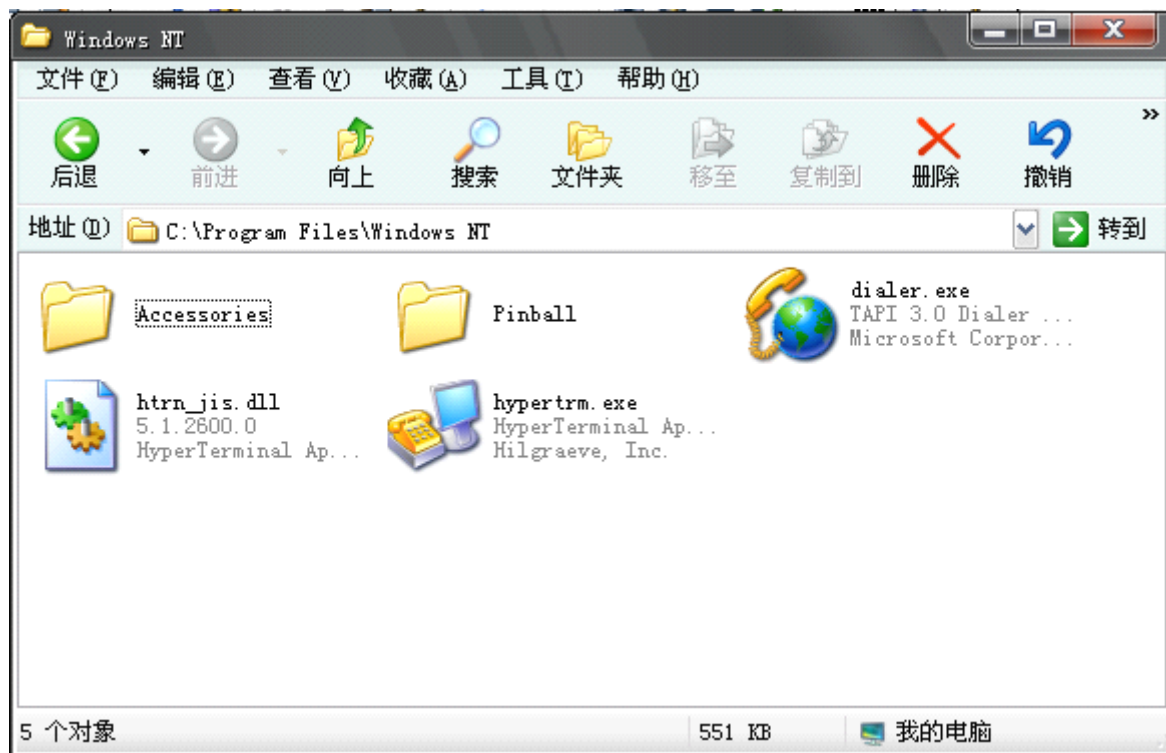
2.3.2 超级终端的使用

为了通过串口连接TE2440，必须使用一个模拟终端程序，几乎所有的类似软件都可以使用，其中Windows 自带的超级终端也是最常用的选择，当你安装Windows9x 时需要自定义选择安装该项，Windows2000 及更高版本则已经缺省安装；在Linux 操作系统下minicom 是一个非常好用的虚拟终端程序，大多数发行版安装时均已包含了该程序，使用时你只要设置一下即可，其他类似软件在这里就不介绍了。

在此首先着重介绍一下Windows 自带的超级终端程序并以Windows XP为例，或许其他Windows 版本的程序界面有所不同。

超级终端程序通常位于“开始->程序->附件->通讯->超级终端”，点击超级终端即可。也可按如图所

示目录来进行查找：



“Hypertrm.exe”图标所指即是超级终端程序，当该程序第一次运行时，会跳出一个对话框，询问你是否要安装一个Modem，此时你不需要安装Modem，因此点“否”按钮。

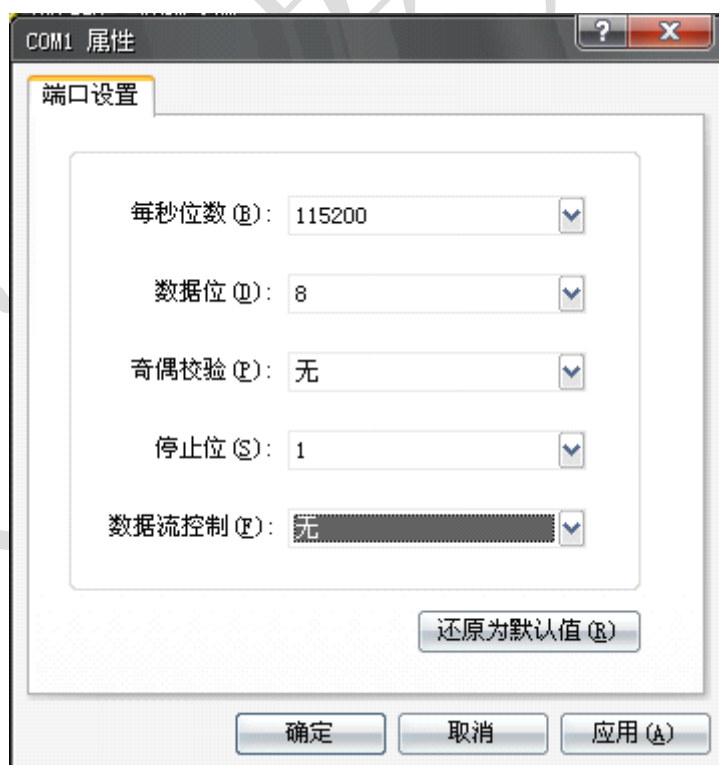
接下来，超级终端会要求你为新的连接取一个名字，如下图所示，这里我取了”OK2440”，Windows系统会禁止你取类似”COM1”这样的名字，因为这个名字被系统占用了。



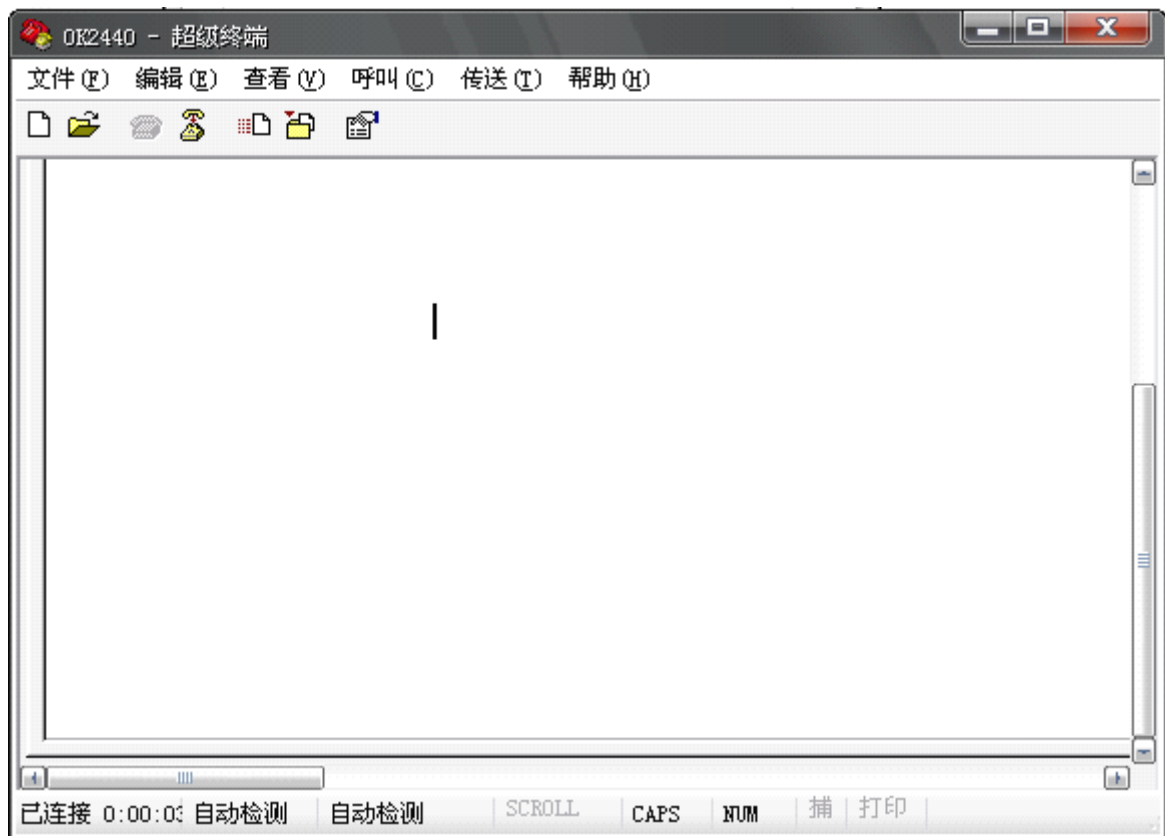
当命名完成以后，会有一个新的对话框，你需要选择连接TE2440的串口，这里选择了串口1，如图所示：



最后，最重要的一步是设置串口，注意必须选择无流控制，否则，或许你只能看到输出而不能输入，另外TE2440工作时的串口波特率是115200，如图所示。



当所有的连接参数都设置好以后，超级终端会显示如下所示主窗口，此时串口是空白的。



选择超级终端“文件”菜单下的“另存为...”，保存该连接设置，以便于以后再连接时就不必重新执行以上设置了。

当板子上电后，终端上就会显示 bootloader 的相关信息。

2.4 bootloader 使用全攻略

2.4.1 bootloader 简介

在专用的嵌入式板子运行 GNU/Linux 系统已经变得越来越流行。一个嵌入式 Linux 系统从软件的角度看通常可以分为四个层次：

- 1、 引导加载程序。包括固化在固件(firmware)中的 boot 代码(可选)，和 BootLoader 两大部分。
- 2、 Linux 内核。特定于嵌入式板子的定制内核以及内核的启动参数。
- 3、 文件系统。包括根文件系统和建立于 Flash 内存设备之上文件系统。通常用 ramdisk 来作为 rootfs。

4、 用户应用程序。特定于用户的应用程序。有时在用户应用程序和内核层之间可能还会包括一个嵌入式图形用户界面。

引导加载程序是系统加电后运行的第一段软件代码。PC 机中的引导加载程序由 BIOS(其本质就是一段固件程序)和位于硬盘 MBR 中的 OS BootLoader (比如, LILO 和 GRUB 等) 一起组成。BIOS 在完成硬件检测和资源分配后, 将硬盘 MBR 中的 BootLoader 读到系统的 RAM 中, 然后将控制权交给 OS BootLoader。BootLoader 的主要运行任务就是将内核映像从硬盘上读到 RAM 中, 然后跳转到内核的入口点去运行, 也即开始启动操作系统。

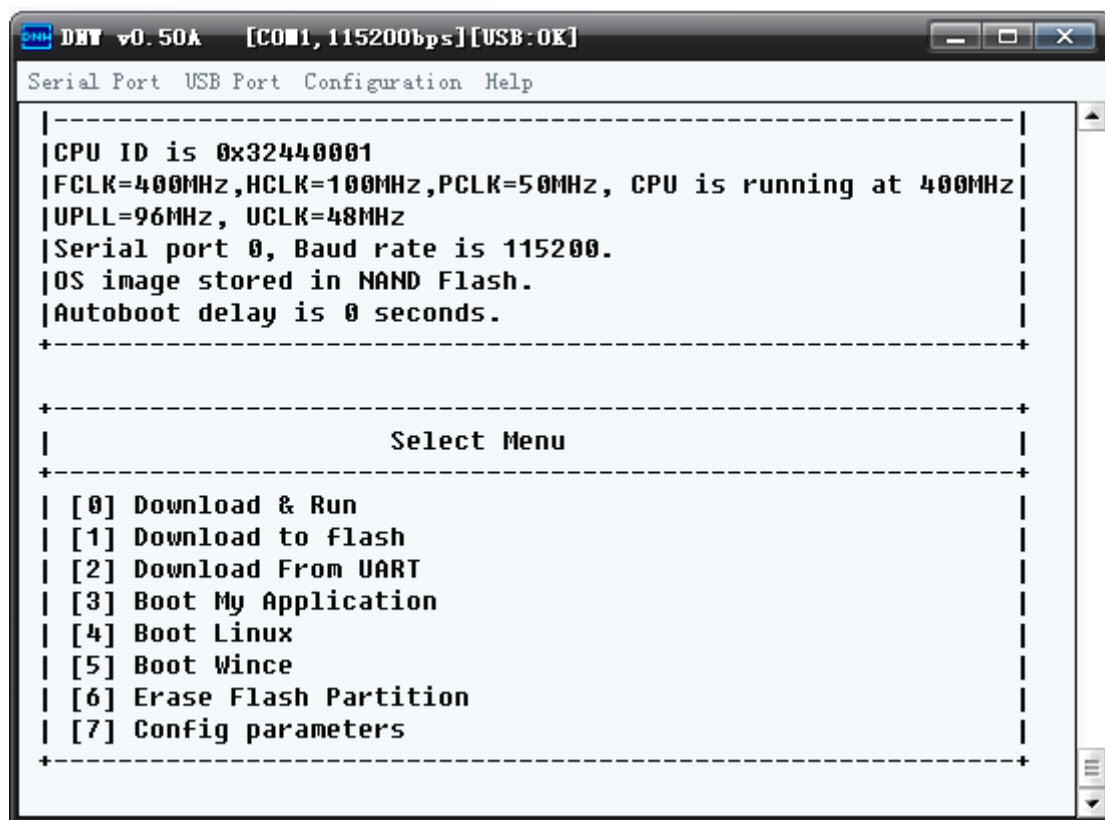
而在嵌入式系统中, 通常并没有像 BIOS 那样的固件程序(注, 有的嵌入式 CPU 也会内嵌一段短小的启动程序), 因此整个系统的加载启动任务就完全由 BootLoader 来完成。比如在一个基于 ARM920T core 的嵌入式系统中, 系统在上电或复位时通常都从地址 0x00000000 处开始执行, 而在这个地址处安排的通常就是系统的 BootLoader 程序。

简单地说, BootLoader 就是在操作系统内核运行之前运行的一段小程序, 功能类似 BIOS。通过这段小程序, 我们可以初始化硬件设备、建立内存空间的映射图, 从而将系统的软硬件环境带到一个合适的状态, 以便为最终调用操作系统内核准备好正确的环境。

注意由于 BootLoader 是严重地依赖于硬件而实现的, 特别是在嵌入式世界。因此, 在嵌入式世界里建立一个通用的 BootLoader 几乎是不可能的。

2.4.2 功能菜单说明

打开 DNW 并连接上串口, 板子上电后在 DNW 中敲入任意键进入 bootloader 菜单界面如下图:



TE2440 开发板所带的 bootloader 主要功能有通过串口或 USB 口下载文件 (配合 DNW 使用)、NAND FLASH 烧写、启动 FLASH 中的程序、设置启动参数。

注意用 USB 下载要先在 PC 端装好驱动程序，保证 USB 连接好，有时 PC 端出现发现无法识别的 USB 设备时，此时可按 ESC 键取消下载，等几秒钟再输入 0 或者 1 启动 USB 下载。

另外需要注意的是，在启用 USB 下载时，如不能下载需要将 USB 重新插拔！

2.4.3 选择菜单说明

功能[0]：通过USB DEVICE 下载程序，并运行

功能[1]：通过USB DEVICE 下载程序，下载完后会自动写入 NAND FLASH

功能[2]：通过串口下载程序，BIOS 内部固定了下载地址为 0x30400000

功能[3]：启动用户自己的应用程序，即分区 1 中的程序

功能[4]：启动 LINUX 操作系统

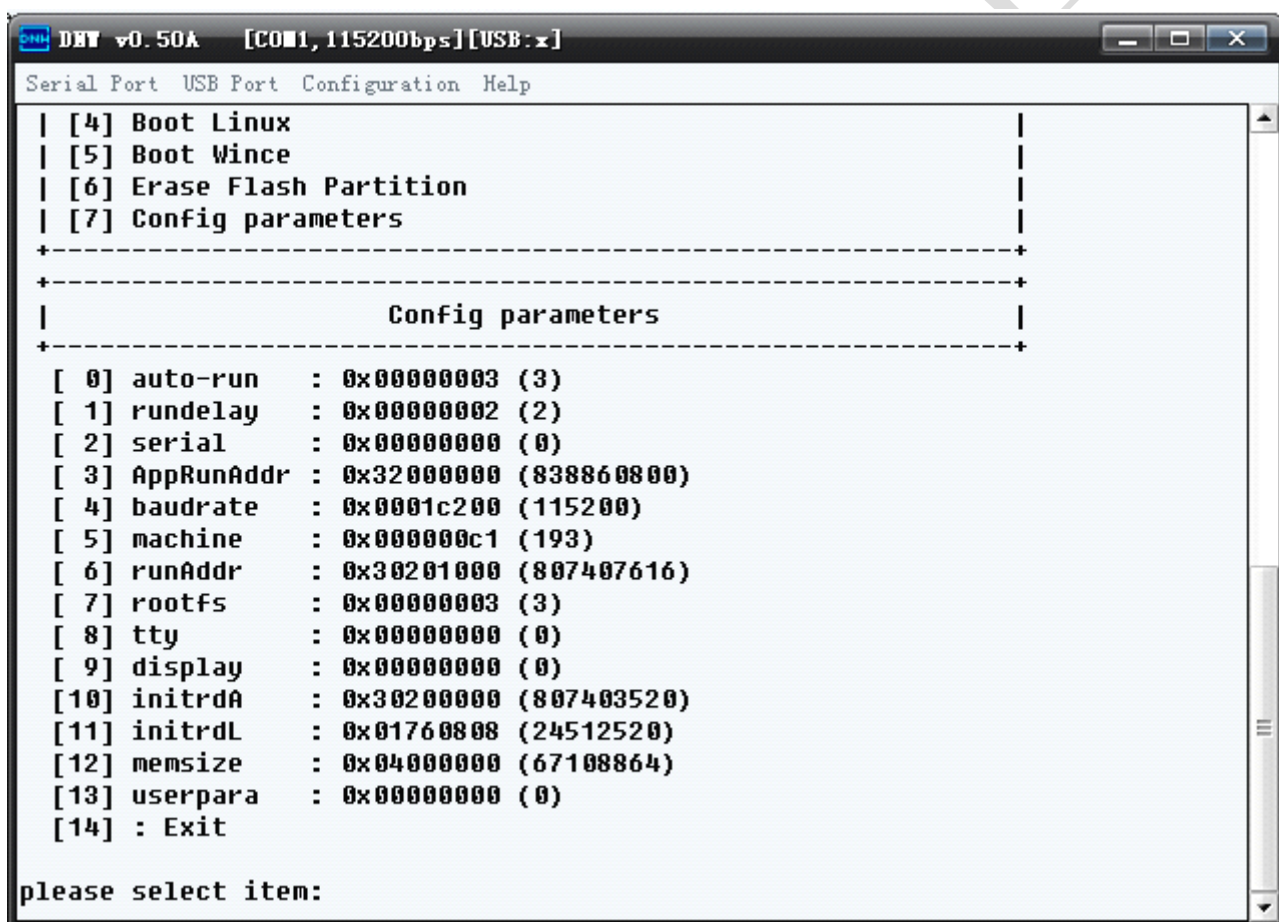
功能[5]：启动 WINCE 操作系统

功能[6]：擦除NAND FLASH 分区

功能[7]：配置一些系统参数和Linux 启动参数

2.4.4 参数设置说明

在功能[7]中，会出现 13 个配置，设置方法是在 please select item 后面输入选项编号，然后在 please enter value 后面输入要修改的值，回车。选择 14 或直接按回车可退出。若修改了参数，退出时出现提示是否需要保存，按“Y”后系统会保存参数退出，显示主菜单。



各参数说明如下：

参数[0]：自启动设置，1 为用户自己的程序（分区 1 中的程序），2 为 LINUX 操作系统，3 为 WINCE 系统；

参数[1]：rundelay，这个选项用来设置自启动延迟的时间。设置为 0，则不自启动，设置某个非 0 参数，则 BOOTLOADER 启动后，不按任意键延迟一段时间后，会按照参数[0]自启动程序。

参数[2]：调试串口选择，这个选项用来设置启动 BOOTLOADER 的控制台串口。默认设置为 0，设置 0 选择串口 0，设置 1 选择串口 1，设置其它值默认为 0。

参数[3]: 用户应用程序的运行地址, 若是用 ADS 开发的应用程序, 应与 R0 Base 地址一致。

参数[4]: 调试串口的波特率设置, 这个选项用来设置启动 Linux 的控制台串口的波特率。默认设置为 115200。

参数[5]: 传递给 Linux 的机器号 ID, 这个选项用来设置启动 Linux 的机器号 ID 的验证。

参数[6]: 操作系统运行的入口地址, 一般为 0x30201000。

参数[7]: rootfs, 设置 Linux 根文件系统所在的分区, 从 0 开始计数。TE2440 提供两种格式的文件系统: 分区 3 为 cramfs 格式的文件系统, 分区 4 为 yaffs 格式的文件系统。(注: 由于 flash 空间问题, cramfs 格式的文件系统带有图形界面, 而 yaffs 格式的文件系统只提供一个根文件系统)

参数[8]: LINUX 控制台串口选择, 0 为串口 0, 1 为串口 1。

参数[9]: LINUX 显示分辨率选择, 0 为 320×240 (用于 3.5 寸液晶屏), 1 为 640×480。

参数[10]: WINCE 需要引导的内核地址, 默认设置为 0x30200000。

参数[11]: WINCE 需要引导的内核长度, 自动设置为当前下载镜像长度, 可以手动修改。

参数[12]: 设置内存大小, 这里出厂硬件配置为 64M 的 SDRAM, 故而设置为 0x4000000。

参数[13]: 传给 LINUX 的参数, 扩展用, 可由用户自定义。

参数设置后, 选择 14 退出, 将会提示是否要保存, 输入 “N”, 不保存, 输入 “y”, 参数将保存到 FLASH 中。

2.4.5 通过 USB 下载程序并写入 FLASH

TE2440 开发板出厂时将 FLASH 分为 6 个分区, 并烧写了相关的程序:

0	:	offset 0x0	,	size 0x30000		[boot]
1	:	offset 0x30000	,	size 0x200000		[MyApp]
2	:	offset 0x230000	,	size 0x1d0000		[kernel]
3	:	offset 0x400000	,	size 0x1400000		[fs_cramfs]
4	:	offset 0x1800000	,	size 0x800000		[fs_yaffs]
5	:	offset 0x2000000	,	size 0x2000000		[wince]

0 为 bootloader 分区 (2440boot.bin)

1 分区留作用户存放自己的应用程序，出厂时烧写了硬件测试程序（2440test.bin）

2 为 LINUX 内核分区 (zImage)

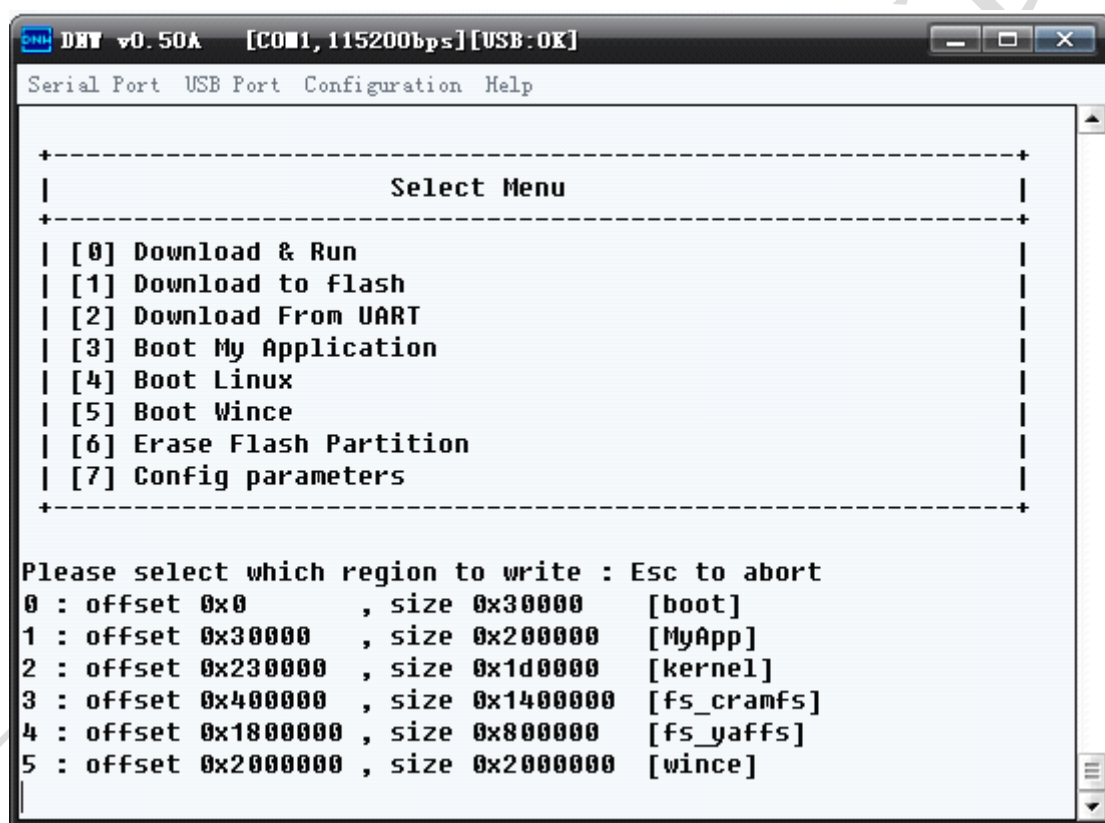
3 为 LINUX 的文件系统分区—cramfs 格式 (qte_mouse.cramfs)

4 为 LINUX 的文件系统分区—yaffs 格式 (root.yaffs)

5 为 WINCE 操作系统分区 (nk.bin)。

我们提供了 bootloader 的部分源代码，用户可根据需求进行修改。

烧写 FLASH 的具体操作如下，在 bootloader 菜单出现 select menu 时敲入 1 即进入下面的界面：



选择要烧写的分区，在键盘上敲入要烧写内容所在的分区号（如要烧写 rootfs 则输入 3），这时会出现下面两行提示：

Now download and write nand flash part [rootfs]

press [USB Port-->transmit] to choose the file

接下来点击 DNW 菜单的 USB Port-->transmit 选择要烧写的文件即可，如下图：



注意用 USB 下载要先在 PC 端装好驱动程序, 保证 USB 连接好, 有时 PC 端出现发现无法识别的 USB 设备时, 可重新插拔 USB 线或者重启开发板。

2.4.6 用 sjf2440.exe 烧写 bootloader 程序

若不小心将 bootloader 分区擦掉了或者下载了不正确的 bootloader 程序, 导致 bootloader 不能正常运行, 不能通过 USB 下载并烧写 FLASH 时, 可利用 sjf2440 工具重新烧写 bootloader, 过程如下:

首先将 TE2440 开发板的 JTAG 口与小板的 JTAG 口相连接到 PC 机的并口上 (在此之前请先在 PC 机上安装并口驱动)。然后将光盘中实用工具文件夹下的 sjf2440.exe 拷贝到 C:\下, 在 WINDOWS 的命令行模式下进入到 C:\盘符, 执行命令: `sjf2440 /f:2440boot.bin`

注意, 如果 bootloader 文件 和 sjf2440 不在同一个目录下, 请使用绝对路径名来指定 bootloader 文件, 例如 `sjf2440 /f:c:\temp\2440boot.bin`

将报告检测到 S3C2440, 接下来的选项中均选择 0, 可以看到烧写开始, 烧写过程中将持续输出 “Eppppppppppp...”

```
C:\WINDOWS\system32\cmd.exe - sjf2440 /f:2440boot.bin
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>f:

F:\>cd 2440

F:\2440>sjf2440 /f:2440boot.bin

+-----+
|      SEC JTAG FLASH(SJF) v 0.1      |
|      <S3C2440X & SMDK2440 B/D>      |
+-----+

Usage: SJF /f:<filename> /d=<delay>
> S3C2440X(ID=0x0032409d) is detected.

[ SJF Main Menu ]
  0:K9S1208 prog      1:28F128J3A prog      2:AM29LV800 Prog      3:SST39UF1601 Prog
  4:Memory Rd/Wr      5:Exit
Select the function to test:0

[ K9S1208 NAND Flash JTAG Programmer ]
K9S1208 is detected. ID=0xec76
  0:K9S1208 Program      1:K9S1208 Pr BlkPage      2:Exit
Select the function to test :0

[ SMC(K9S1208U0M) NAND Flash Writing Program ]

Source size:0h~c0afh

Available target block number: 0~4095
Input target block number:0
target start block number      =0
target size      <0x4000*n> =0x10000
STATUS:Eppppppppppppppppp_
```

大约 3 分钟后, 烧写完成, 选择 2, 退出程序。

2.5 ADS 下的 led 试验

2.5.1 ADS 安装

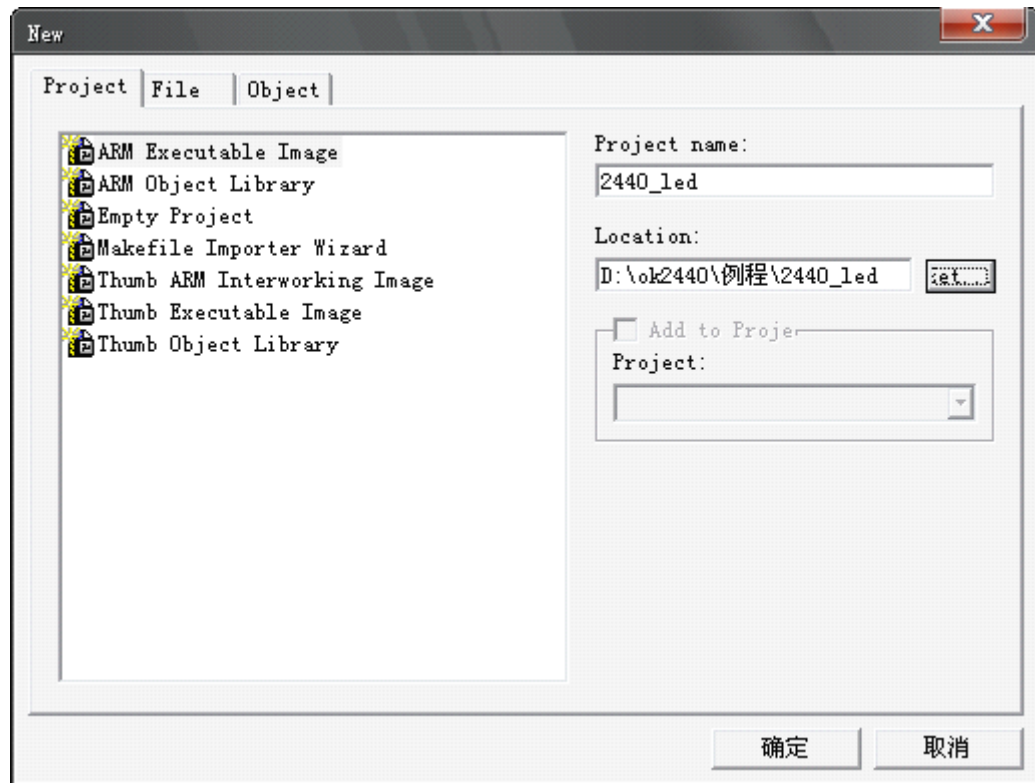
TE2440 的用户光盘中提供了安装程序，直接安装，安装完后将 license.dat 放入 arm/adsv1 2/licenses/下即可。

2.5.2 使用 ADS 创建工程

下面介绍如何利用 CodeWarrior 建立自己的工程，以及如何编译链接，生成包含调试信息的映像文件和可以直接烧写到 flash 中的 bin 格式二进制可执行文件。

首先打开 CodeWarrior, 点击 File 菜单下的 New 来创建新工程。Project 对话框中选择 ARM Executable

Image。在 Project name 中输入工程名，本例为：2440_led，点击“Location:”文本框的“Set...”按钮，选择要将工程保存的路径，然后点击确定即可建立一个新的工程。如下图所示：



在该提示框中我们见到有七种工程类型：

ARM Executable Image:用于由 ARM 指令代码生成一个 ELF 格式的可执行映像文件。

ARM Object Library:用于有 ARM 指令代码生成一个 armar 格式的目标文件库。

Empty Project: 创建一个不包含任何库和源文件的空工程。

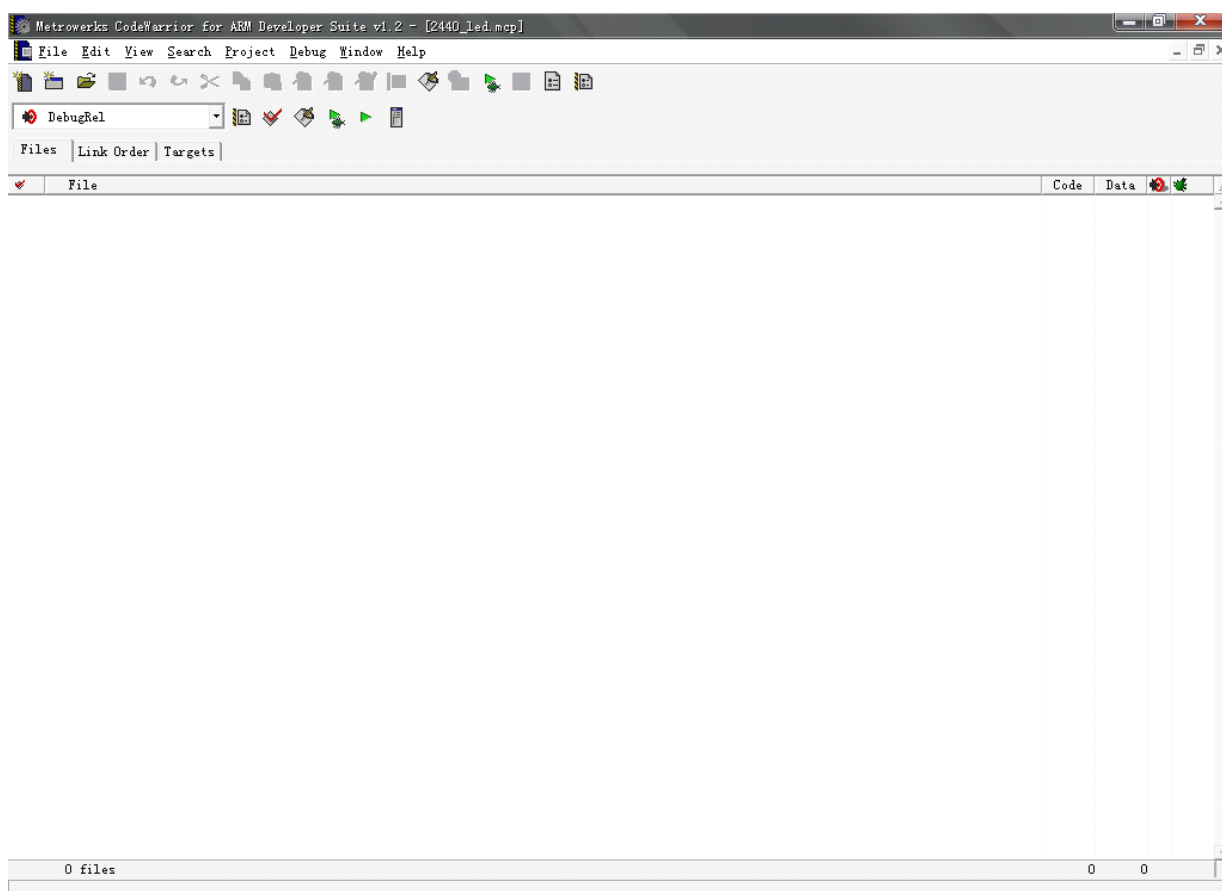
Makefile Importer Wizard:用于将 Visual C 的 nmake 或者 GNU make 文件转入到 CodeWarrior IDE 工程文件。

Thumb ARM Executable Image:用于由 ARM 指令和 Thumb 指令的混合代码生成一个可执行的 ELF 格式的映像文件。

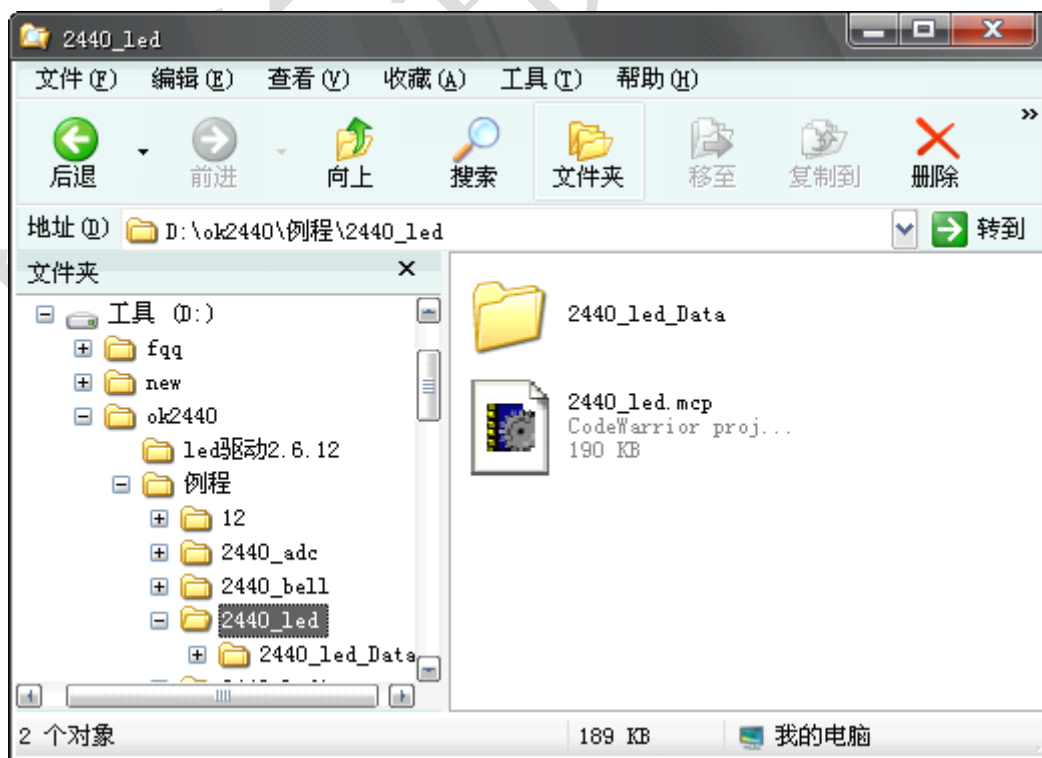
Thumb Executable Image:用于由 Thumb 指令代码生成一个 ELF 格式的可执行映像文件。

Thumb Object Library:用于由 Thumb 指令代码生成一个 armar 格式的目标文件库。

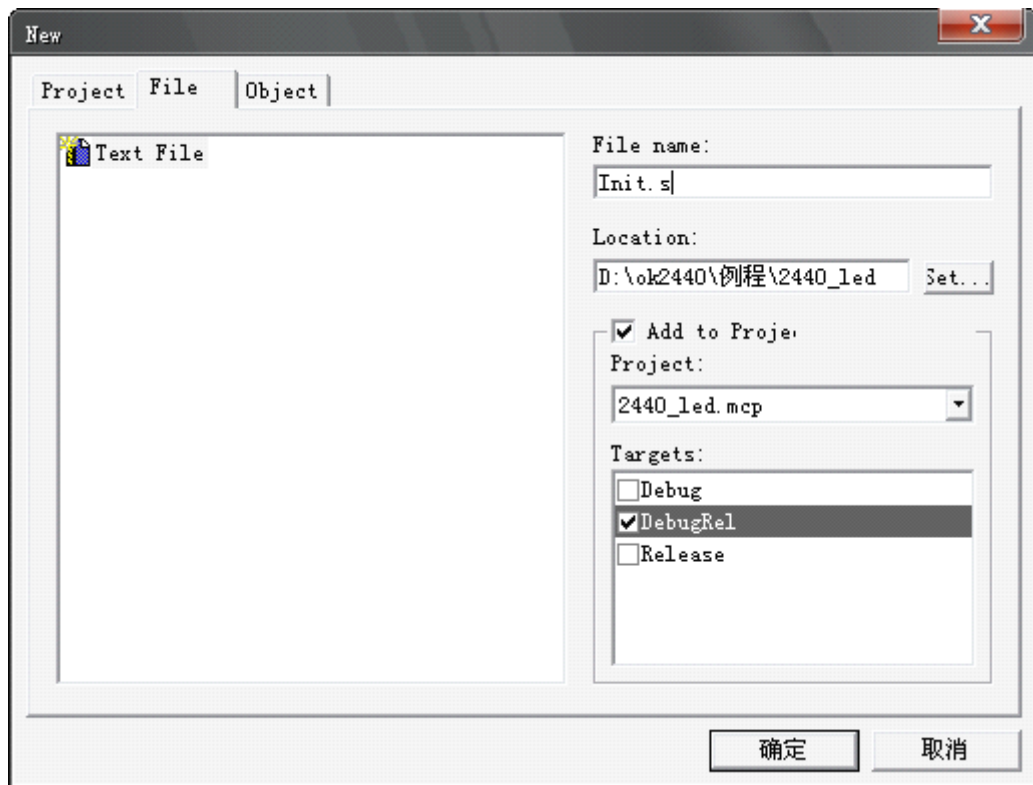
在这里我们选择第一种工程 ARM Executable Image，在 Project name 中输入工程文件名，我们取名为 2440_led，在 location 处点击 set……选择你的工程文件将保存于何处本例保存于 D:\ok2440\例程\2440_led，最后确定就可以了，这时候会出现一个 2440_led.mcp 窗口。



同时 D:\ok2440\例程\2440_led 该目录下会有两个文件



下面开始创建源文件，点击 File 菜单下的 New，选择标签页 File 在 File name 中输入要建立的文件名：Init.s（.s 文件为 arm 中的汇编文件），如此时选上了 Add to Project，以后就不要添加了。点击确定关闭窗口。



在打开的文件编辑框中输入下面的代码：

```
AREA |DATA|, CODE, READONLY  
  
ENTRY  
  
ldr r13, =0x1000  
  
IMPORT LedMain  
  
b LedMain  
  
END
```

这段代码只是简单的跳转，保存该文件。

用同样的办法建立文件：main.c 输入下面的代码：

```
int LedMain()  
{
```

```
#define GPFCON (*(volatile unsigned *)0x56000050)

#define GPFDAT (*(volatile unsigned *)0x56000054)

#define GPFUP (*(volatile unsigned *)0x56000058)

void Delay(unsigned int);

unsigned char ledtab[]={0xdf, 0xbf, 0x7f};

int i;

GPFUP&= 0xfffffffff;//IO 上拉使能

GPFCON&= 0xffff0000;//清零

GPFCON|= 0x5400;//输出有效

while(1)
{
    for(i=0;i<3;i++)
    {
        GPFDAT=ledtab[i];

        Delay(10);
    }
}

return(0);
}

/*延时函数*/

void Delay(unsigned int x)
{
    unsigned int i, j, k;

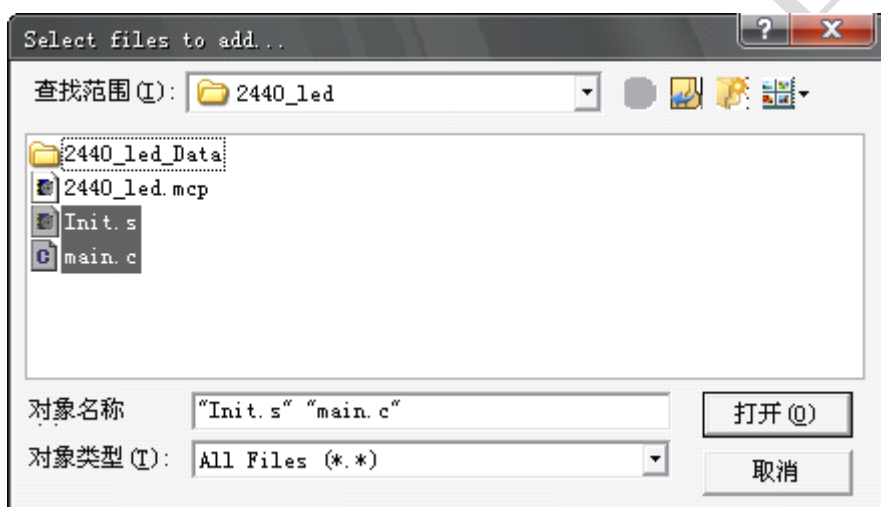
    for(i=0;i<=x;i++)

        for(j=0;j<0xff;j++)
```

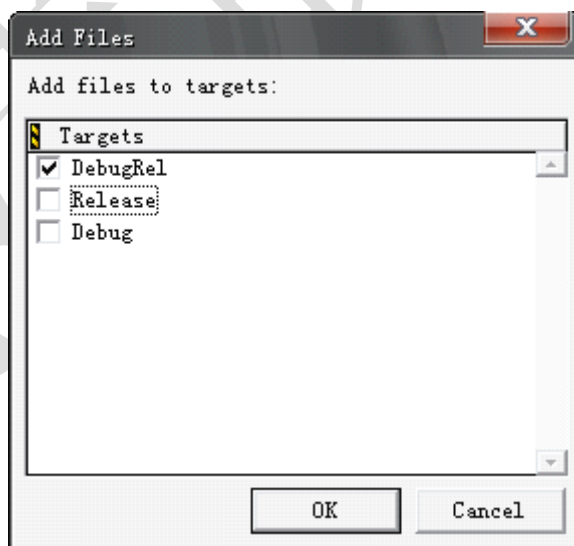
```
for(k=0;k<0xff;k++);  
  
}
```

这段代码首先将 I/O 模式设为输出模式（可参考 S3C2440 的芯片文档），然后在循环中给 I/O 赋值，实现 led 轮流被点亮的功能。

若前边未复选上 Add to Project，那么则需要把创建好这两个源文件添加到工程里面。点击 Project 菜单下的 Add Fils 选择要添加的源文件（init.s 和 main.c），



打开之后会弹出询问添加哪类目标中的串口，如图



此时有三个选项：

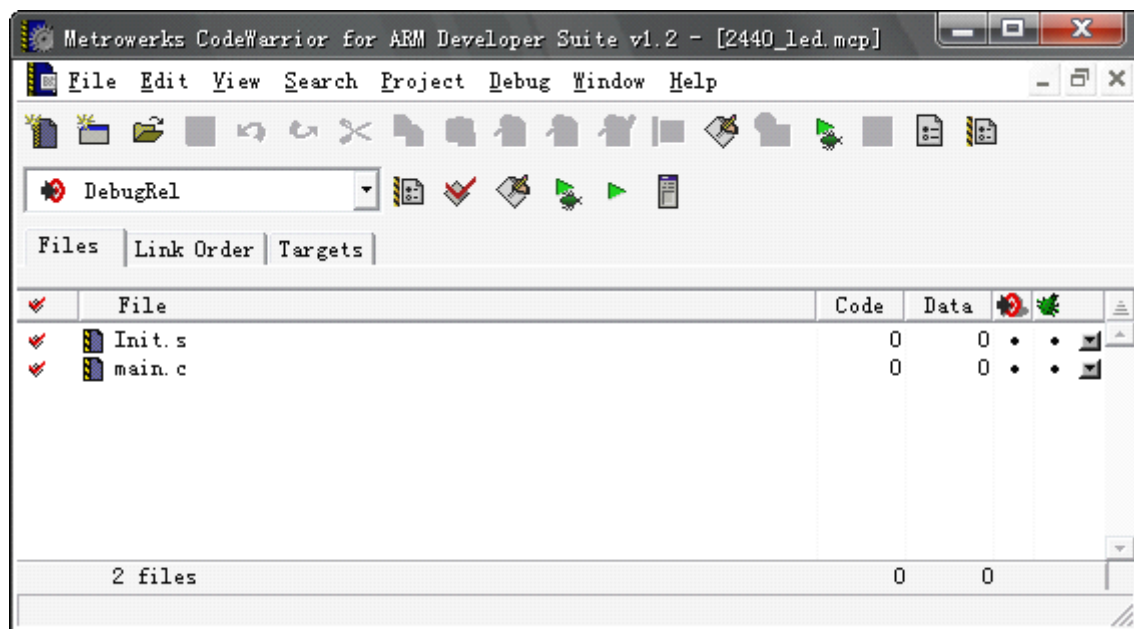
DebugRel:使用该目标选项，在生成目标的时候，会为每一个源文件生成调试信息。

Release: 使用该目标选项，在生成目标的时候，不生成任何调试信息。

Debug: 使用该目标选项，在生成目标的时候，会为每一个源文件生成完美的调试信息。

我们选择 DebugRel。也可在工程窗体上的 File 标签页中单击右键--->Add Fils。

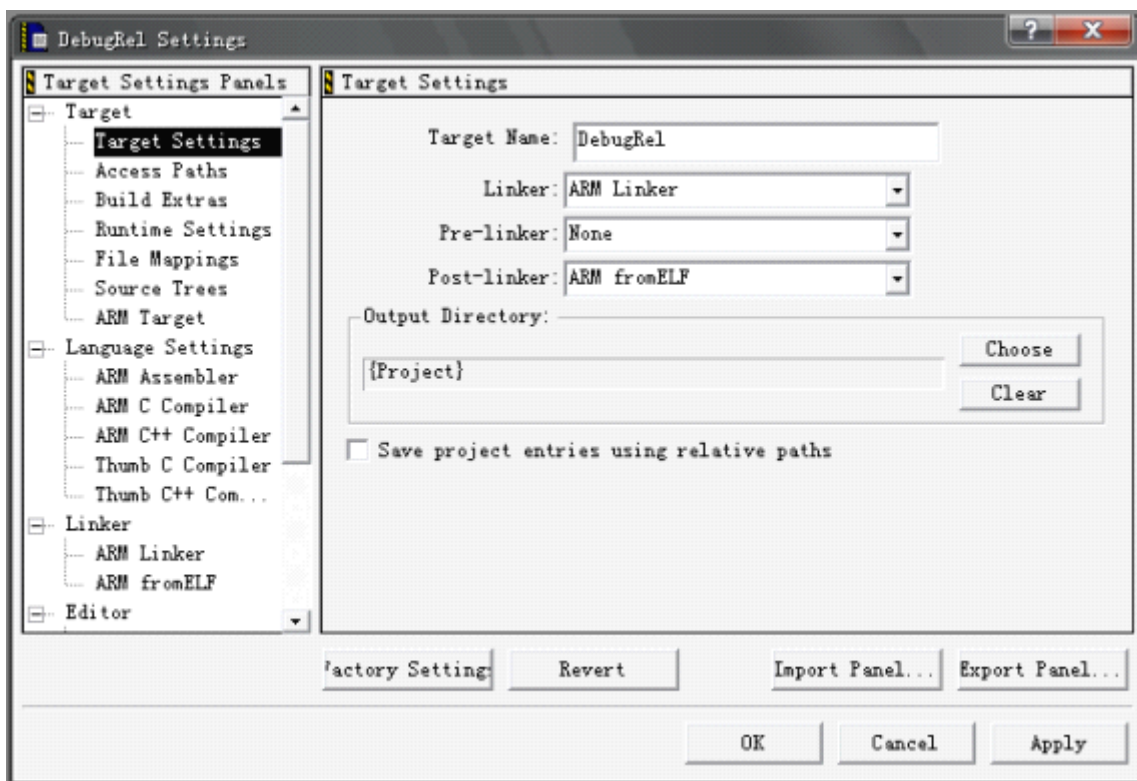
接下来要进行编译和链接。



2.5.3 编译和链接工程

在进行编译和链接前，首先讲述一下如何进行生成目标的配置。

点击Edit菜单，选择“DebugRel Settings...”（注意，这个选项会因用户选择的不同目标而有所不同），出现如下图所示的对话框。



这个对话框中的设置很多，在这里介绍一些最为常用的设置选项，读者若对其他未涉及到的选项感兴趣，可以查看相应的帮助文件。

1. target设置选项

Target Name文本框显示了当前的目标设置。

Linker选项供用户选择要使用的链接器。在这里默认选择的是ARM Linker，使用该链接器，将使用armlink链接编译器和汇编器生成的工程中的文件相应的目标文件。

这个设置中还有两个可选项：

None不是不用任何链接器，如果使用它，则工程中的所有文件都不会被编译器或汇编器处理。ARM Librarian表示将编译或汇编得到的目标文件转换为ARM库文件。对于本例，使用默认的链接器ARM Linker。

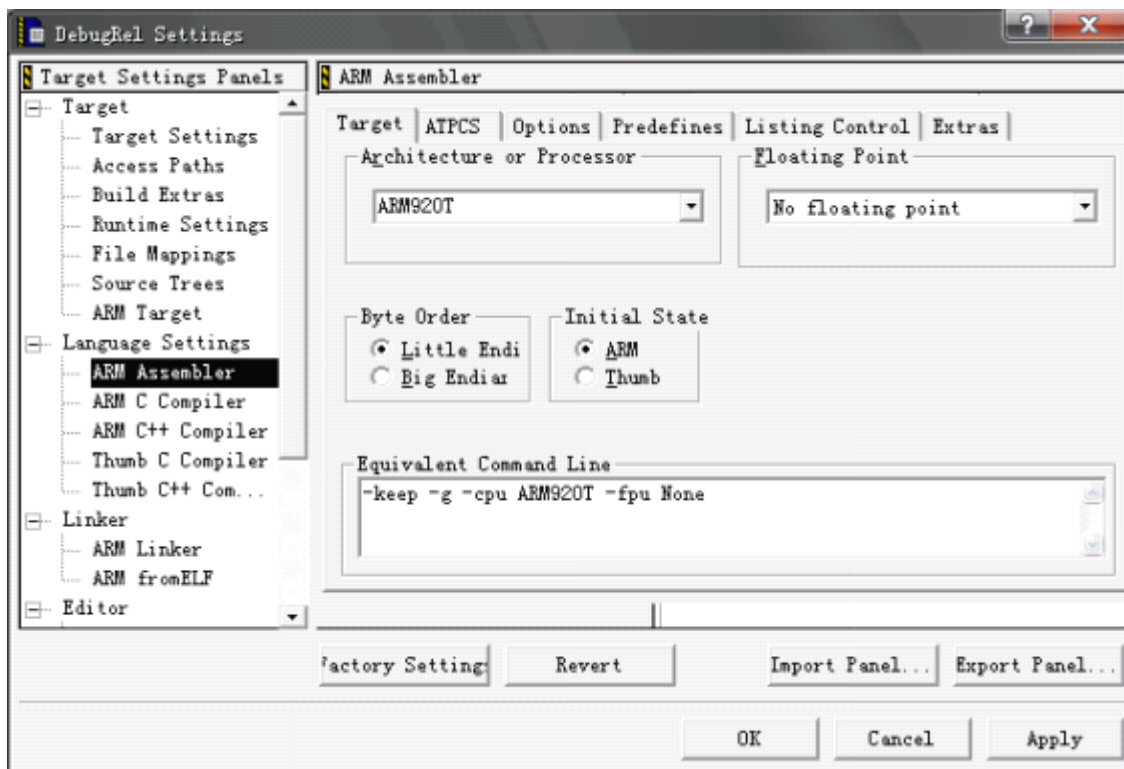
Pre-linker：目前CodeWarrior IDE不支持该选项。

Post-Linker：选择在链接完成后，还要对输出文件进行的操作。因为在本例中，希望生成一个可以烧写到Flash中去的二进制代码，所以在这里选择ARM fromELF，表示在链接生成映像文件后，再调用FromELF命令将含有调试信息的ELF格式的映像文件转换成其他格式的文件。

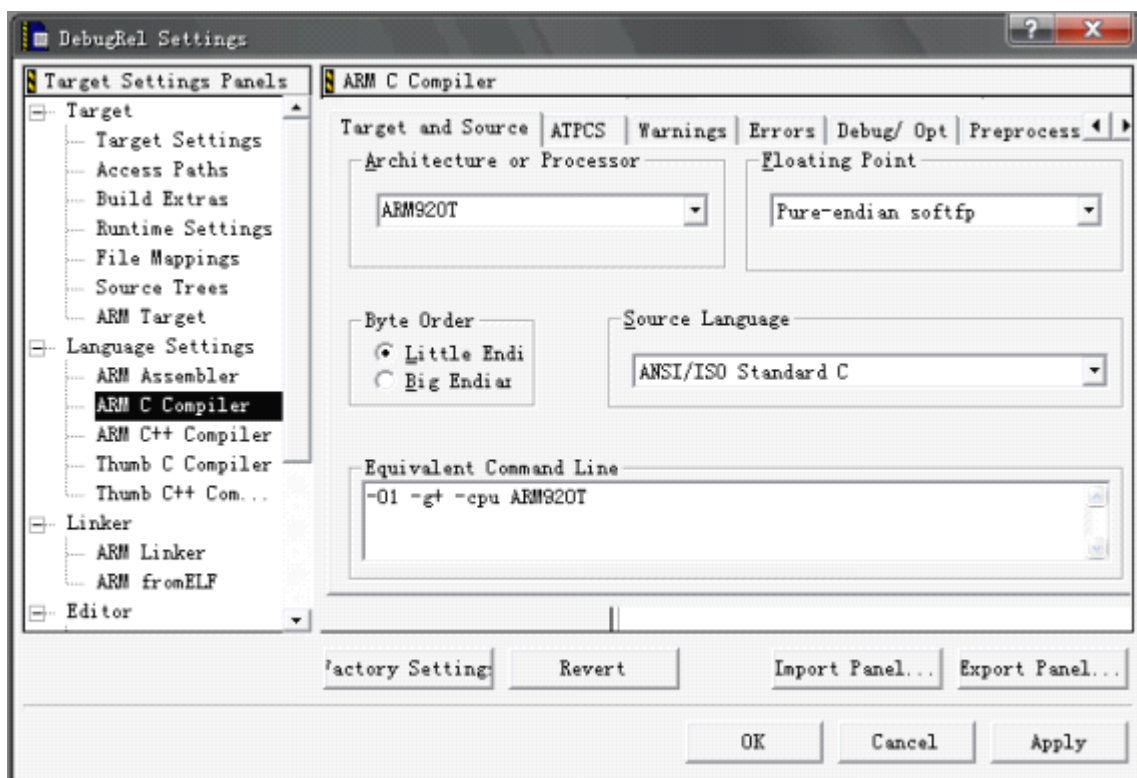
最终配置如上图。

2. Language Settings

因为本例中包含有汇编源代码，所以要用到汇编器。首先看ARM汇编器，默认的ARM体系结构是ARM7TDMI，将其改为ARM920T。字节顺序默认就是小端模式。其他设置，就用默认值即可。



本工程中还包含C语言代码，因此还有必要设置ARM C Compiler 选项，点击该选项会在右侧出现相应的设置选项，将默认的ARM体系结构ARM7TDMI改为ARM920T，字节顺序仍然是小端模式，其它设置采取默认值即可，如图：

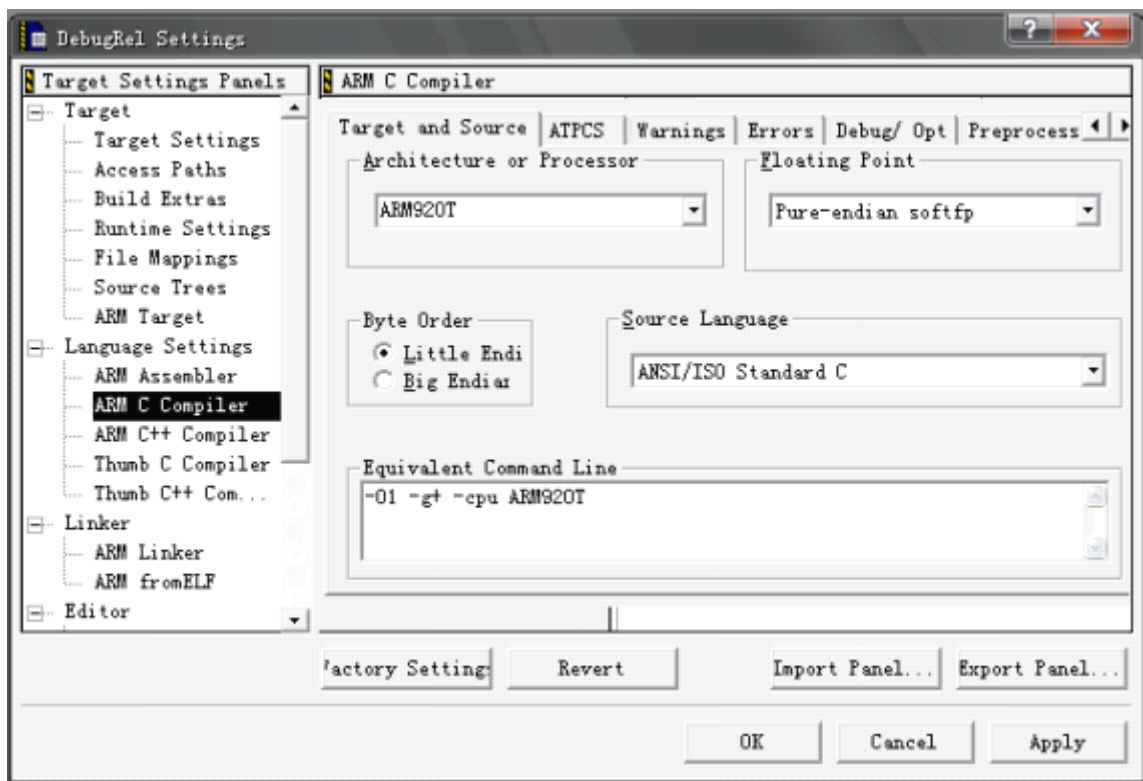


有的读者可能注意到了在设置框的下边有一个命令行，当对上边某个设置进行修改时，该行中的某个参数就会跟着发生相应的改变，实际上，这行文字显示的就是相应的编译或者连接选项，由于有了CodeWarrior，开发人员可以不再去记数量繁多的命令行选项，只要在界面中选择或撤销某个选项，软件就会自动生成相应的代码。不过该命令框也为习惯DOS下键入命令行的用户提供了极大方便。

3. Linker设置

鼠标选中ARM Linker，出现如下图所示对话框。这里详细介绍该对话框的主要的标签页选项，因为这些选项对最终生成的文件有着直接的影响。

在标签页Output中，Linktype中提供了三种链接方式。Partial方式表示链接器只进行部分链接，经过部分链接生成的目标文件，可以作为以后进一步链接时的输入文件。Simple方式是默认的链接方式，也是最为频繁使用的链接方式，它链接生成简单的ELF格式的目标文件，使用的是链接器选项中指定的地址映射方式。Scattered方式使得链接器要根据scatter格式文件中指定的地址映射，生成复杂的ELF格式的映像文件。这个选项一般情况下，使用不太多。因为我们所举的例子比较简单，选择Simple方式就可以了。在选中Simple方式后，就会出现Simple image。



R0 Base: 这个文本框设置包含有R0段的加载域和运行域为同一个地址。默认是0x8000。这里用户要根据自己硬件的实际SDRAM的地址空间来修改这个地址，保证在这里填写的地址，是程序运行时，SDRAM地址空间所能覆盖的地址。

RW Base: 这个文本框设置了包含RW和ZI输出段的运行域地址。如果选中split选项，链接器生成的映像文件将包含两个加载域和两个运行域，此时，在RW Base中所输入的地址为包含RW和ZI输出段的域设置了加载域和运行域地址

Ropi: 选中这个设置将告诉链接器使包含有R0输出段的运行域位置无关。使用这个选项，链接器将保证下面的操作：

- 检查各段之间的重定址是否有效；

- 确保任何由armlink自身生成的代码是只读位置无关的。

Rwpi: 选中该选项将会告诉链接器使包含RW和ZI输出段的运行域位置无关。如果这个选项没有被选中，域就标识为绝对。每一个可写的输入段必须是读写位置无关的。如果这个选项被选中，链接器将进行下面的操作，

- 检查可读/可写属性的运行域的输入段是否设置了位置无关属性；

- 检查在各段之间的重地址是否有效；

- 在Region\$\$Table和ZISection\$\$Table中添加基于静态存储器sb的选项。

该选项要求RW Base有值，如果没有给它指定数值的话，默认为0值。

Split Image: 选择这个选项把包含R0和RW的输出段的加载域分成2个加载域：一个是包含R0输出段的域，一个是包含RW输出段的域。

这个选项要求RW Base有值，如果没有给RW Base选项设置，则默认是-RW Base 0。

Relocatable: 选择这个选项保留了映像文件的重定址偏移量。这些偏移量为程序加载器提供了有用信息。

在Options选项中，需要读者引起注意的是Image entry point文本框。它指定映像文件的初始入口点地址值，当映像文件被加载程序加载时，加载程序会跳转到该地址处执行。如果需要，用户可以在这个文本框中输入下面格式的入口点：

入口点地址：这是一个数值，例如-entry 0x0

符号：该选项指定映像文件的入口点为该符号所代表的地址处，比如：

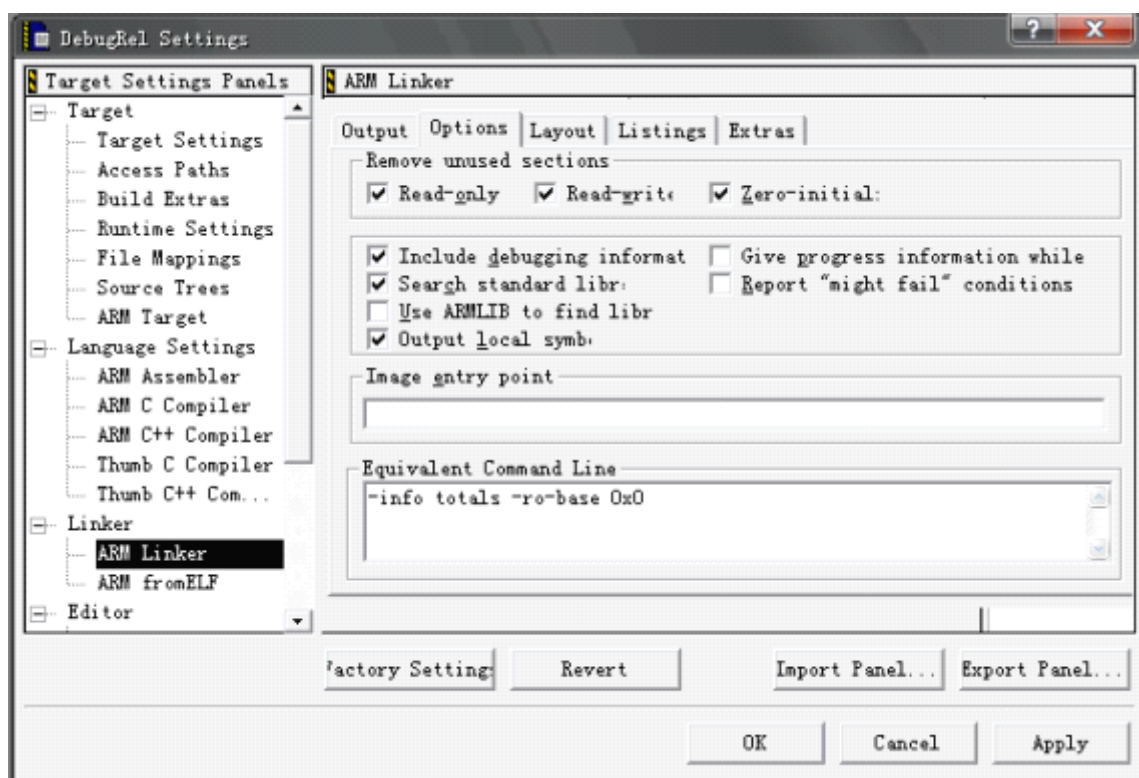
-entry int_handler

如果该符号有多处定义存在，armlink将产生出错信息。

offset+object(section): 该选项指定在某个目标文件的段的内部的某个偏移量处为映像文件的入口地址，例如：

-entry 8+startup(startupseg)

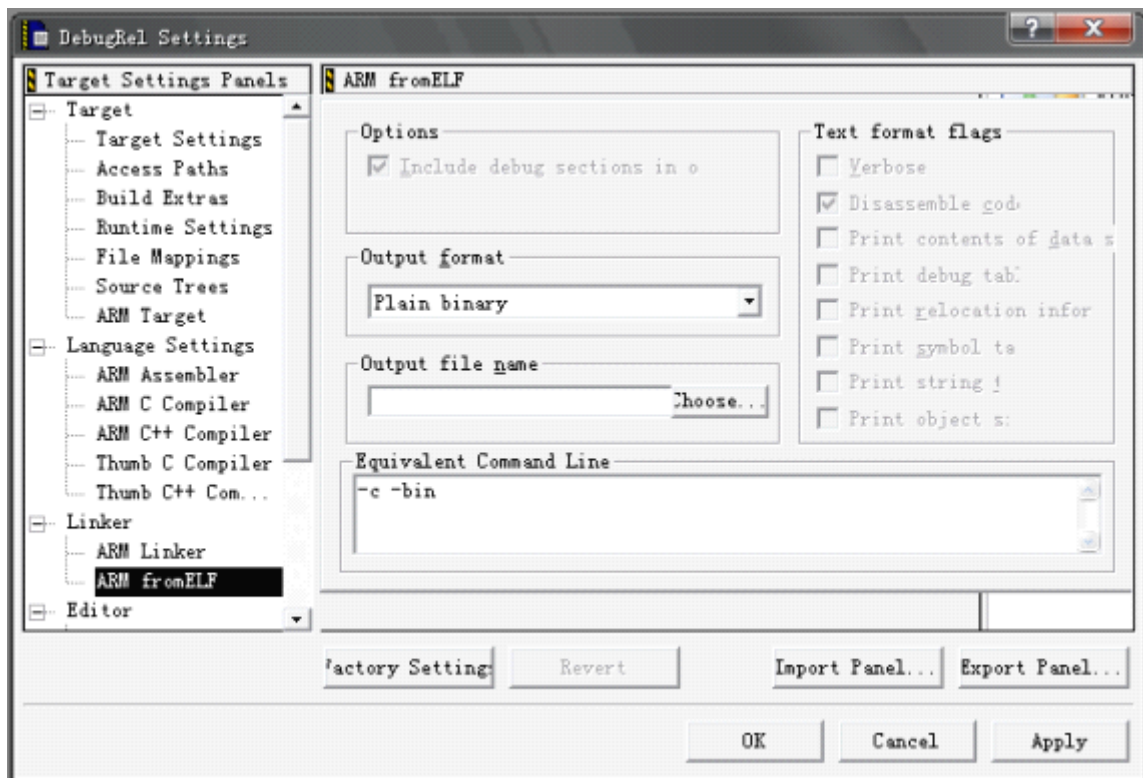
在此处指定的入口点用于设置ELF映像文件的入口地址。



需要引起注意的是，这里不可以用符号main作为入口点地址符号，否则会出现类似“Image dose not have an entry point(Not specified or not set due to multiple choice)”的错误信息。

关于ARM Linker的设置还有很多，对于想进一步深入了解的读者，可以查看帮助文件，都有很详细的介绍。

在Linker下还有一个ARM fromELF，如下图所示：



fromELF是一个实用工具，它实现将链接器，编译器或汇编器的输出代码进行格式转换的功能。例如，将ELF格式的可执行映像文件转换成可以烧写到ROM的二进制格式文件；对输出文件进行反汇编，从而提取出有关目标文件的大小，符号和字符串表以及重定址等信息。

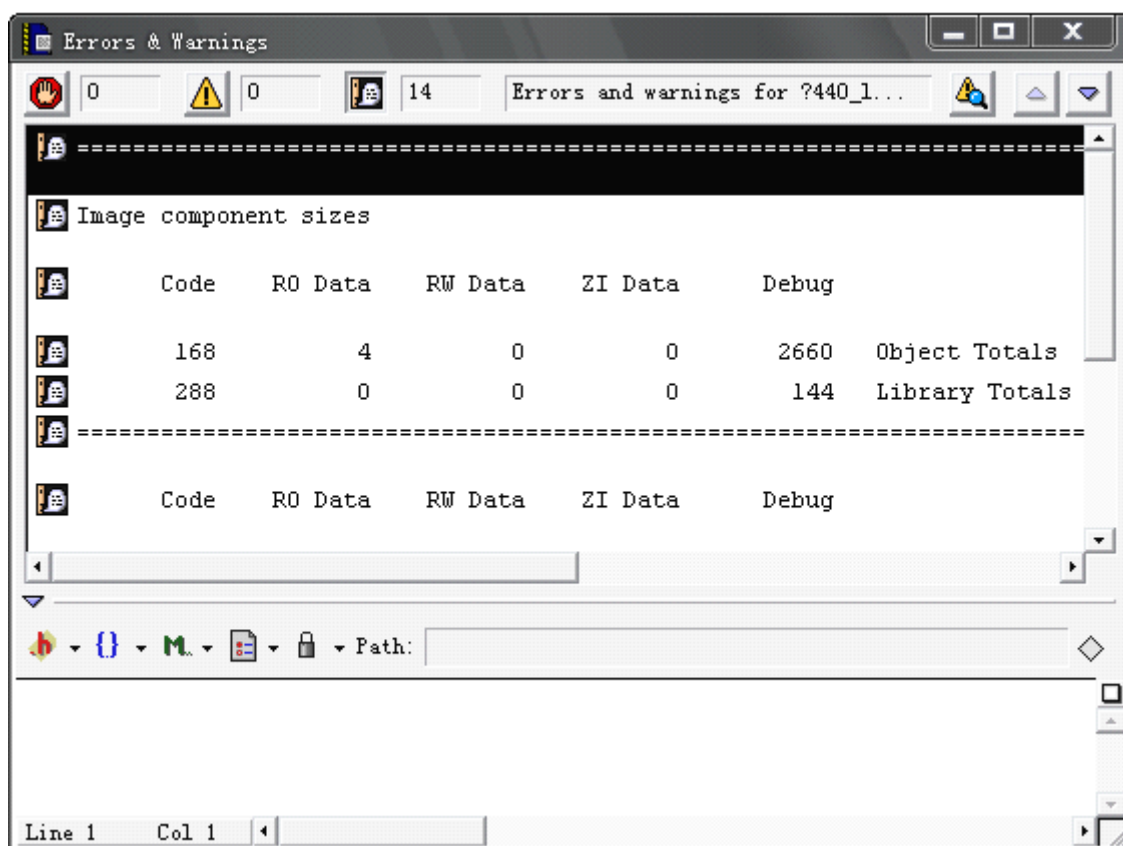
只有在Target设置中选择了Post-Linker，才可以使用该选项。

在Output format下拉框中，为用户提供了多种可以转换的目标格式，本例选择Plain binary，这是一个二进制格式的可执行文件，可以被烧些的目标板的Flash中。

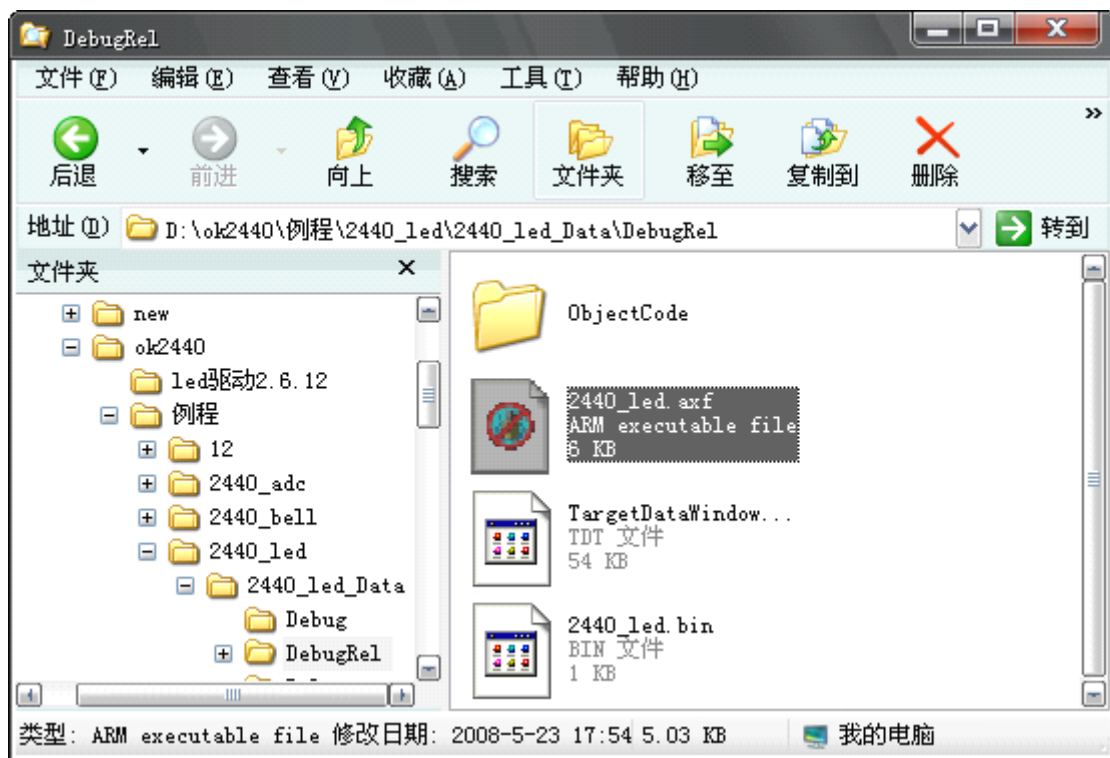
在Output file name文本域输入期望生成的输出文件存放的路径，或通过点击Choose...按钮从文件对话框中选择输出文件。如果在这个文本域不输入路径名，则生成的二进制文件存放在工程所在的目录下。

进行好这些相关的设置后，以后在对工程进行make的时候，CodeWarrior IDE 就会在链接完成后调用fromELF 来处理生成的映像文件。

对于本例的工程而言，到此，就完成了 make 之前的设置工作了。点击 CodeWarrior IDE 的菜单 Project 下的 make 菜单，对工程进行编译和链接。整个编译链接过程下图所示：



在工程2440_1ed所在的目录下，会生成一个名为：工程名_data目录，在本例中就是2440_1ed_Data目录，在这个目录下不同类别的目标对应不同的目录。在本例中由于我们使用的是DebugRe目标，所以生成的最终文件都应该在该目录下。进入到DebugRe1目录中去，会看到make后生成的映像文件和二进制文件，映像文件用于调试，二进制文件可以烧写到TE2440的Flash中运行。



2.5.4 H-JTAG 的安装使用

1. H-JTAG 简介

H-JTAG 是一个针对 ARM 处理器设计的免费 JTAG 调试代理软件,从功能的角度来说,类似于 MULTI-ICE 仿真器。H-JTAG 支持大多数主流的调试软件: SDT2.51, ADS1.2, REALVIEW 以及 IAR。使用 H-JTAG, 你可以轻松地通过 WIGGLER 或者 SDT-JTAG 或用户自定义的 JTAG 小板调试所有的 ARM7/ARM9 处理器。

1. 支持 RDI 1.5.0 以及 1.5.1;
2. 支持 ARM7 以及 ARM9;
3. 支持 THUMB 以及 ARM 指令;
4. 支持 LITTLE-ENDIAN 以及 BIG-ENDIAN;
5. 支持 SEMIHOSTING;
6. 支持 WIGGLER, SDT-JTAG 和用户自定义接口;
7. 支持 WINDOWS 9.X/NT/2000/XP;

H-FLASHER 是一个通用的 FLASH 烧写软件。通过 H-JTAG SERVER, H-FLASHER 可以用来烧写不同的 NOR FLASH 芯片和片内 FLASH。一方面, H-FLASHER 使用了 DCC 来实现快速 FLASH 烧写。

同时, H-FLASHER 还采用了自动校验, 自动擦除和自动恢复等技术, 使 H-FLASHER 变得简单易

用。

2. H-JTAG 安装



H-JTAG V0.4.0.EXE

该软件位于用户光盘/实用工具下，双击

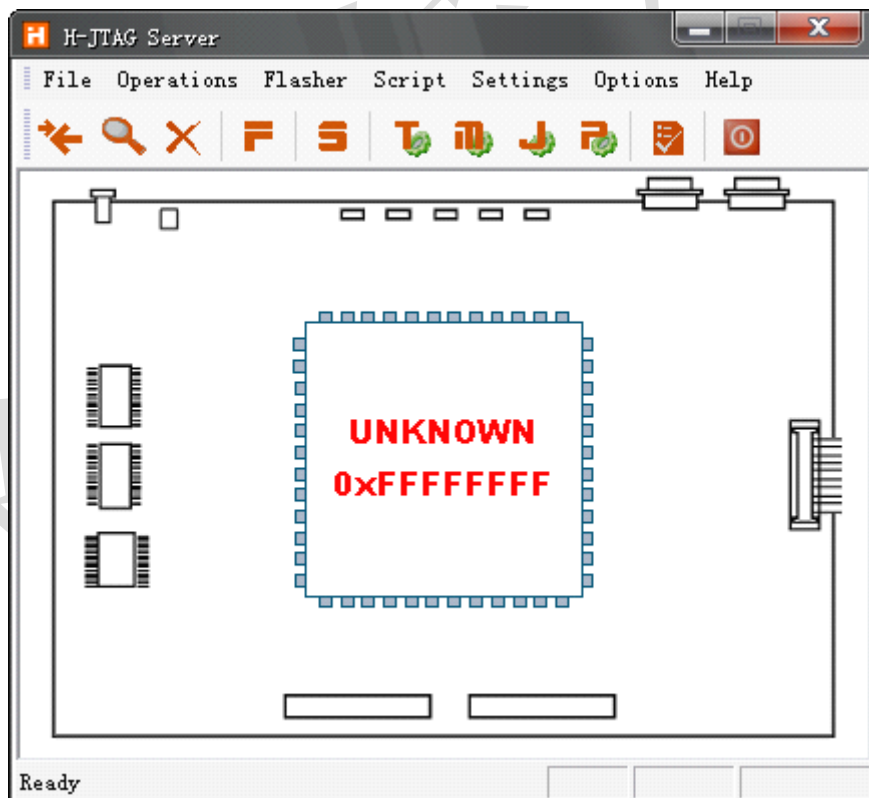
按其提示即可安装成功。

3. H-JTAG 使用

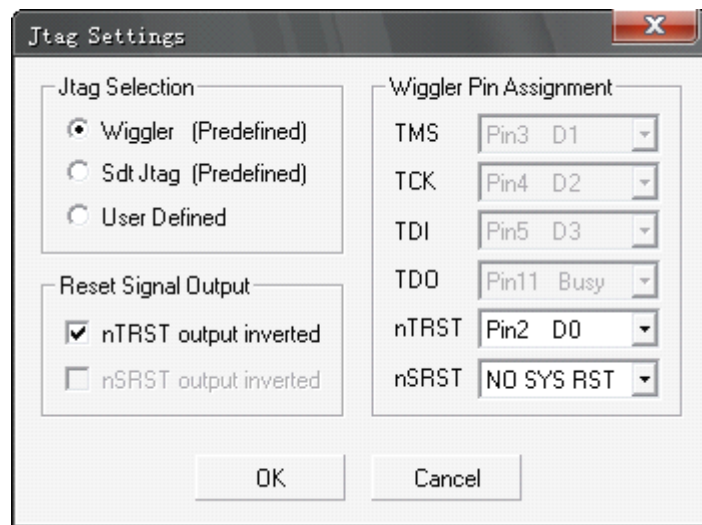
安装完毕会在桌面上生成 H-JTAG 和 H-Flasher 快捷方式，双击运行 H-JTAG，程序会自动检测是否连接了 JTAG 装置。如果未连接任何 JTAG 装置则会弹出如下提示窗：



点击确定进入主界面，由于没有连接任何目标器件，则主界面为：



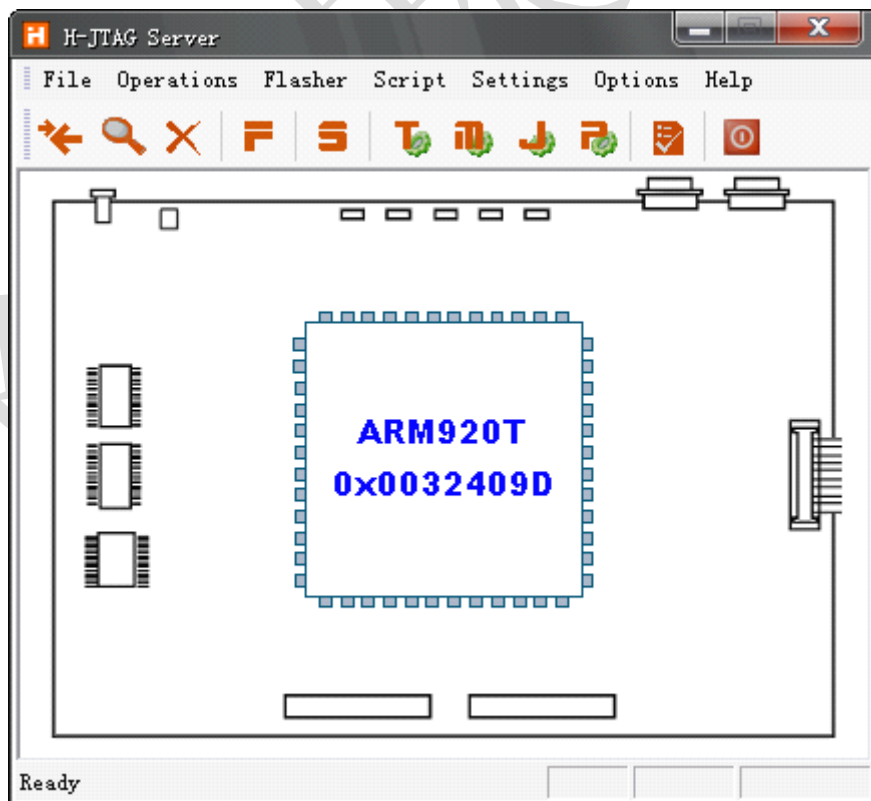
在 H-JTAG 主界面菜单里点击 Setting->Jtag Setting，做如下图所示的设置，点击 OK 返回主界面



注意：通常情况下，我们不需要更改配置，因为我们使用的是 H-JTAG 的默认配置，一旦您误改之后可按上图做正确修改。

使用随开发板附带的 JTAG 小板和开发板上的 JTAG 接口连好，并接上电源，点击 “Operation->Detect

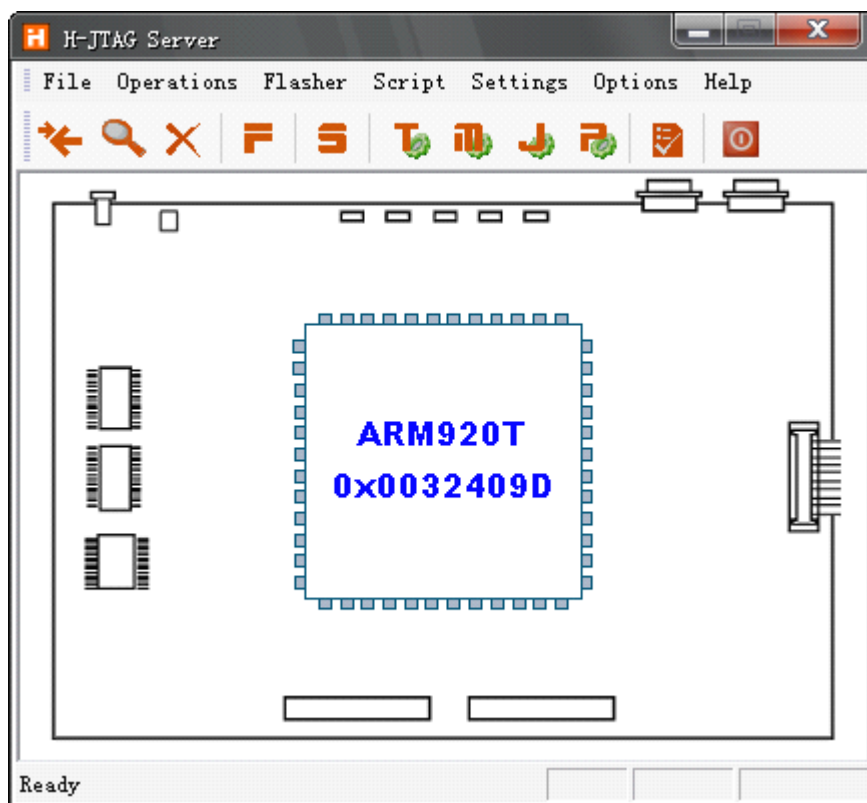
Tartget” 或点击相应的图标，这时就可以看到已经检测到目标器件了，如图：



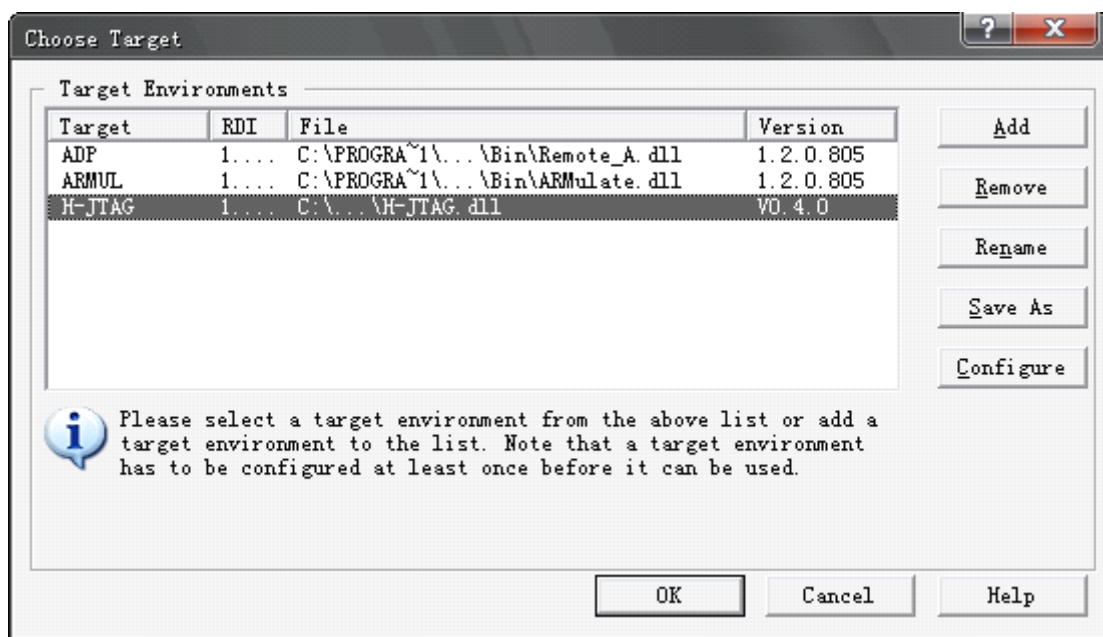
检测到目标器件以后，即可将该软件最小化，会在桌面的右下角有个红色小图标。

2.5.5 用 AXD 进行代码调试

AXD (ARM eXtended Debugger) 是 ADS 软件中独立于 CodeWarrior IDE 的图形软件。打开 AXD 软件前应先打开 H-JTAG，会自动检测到 ARM 内核：ARM920T。如下图所示：



打开 AXD 软件，点击 Options 菜单下的 Config target，弹出 Choose Target 对话框，点击 Add 按钮，在文件浏览器中找到 H-JTAG 安装目录下的 H-JTAG.dll 文件，点击打开。选中 H-JTAG，点击 OK，如下图所示：
(如果要进行仿真调试的话，在 Choose Target 对话框中选择 ARMUL)

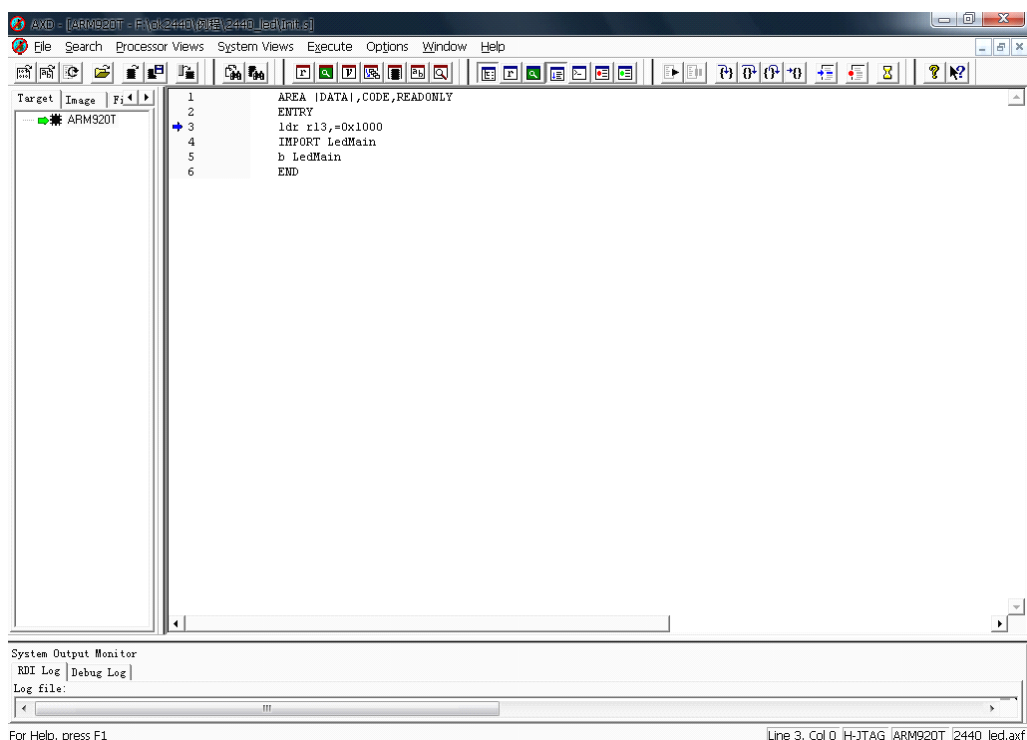


注意：TE2440开发板出厂时都烧写了操作系统，在上电后会自动启动（LINUX或者WINCE）。启动时在超级终端中按任意键停在命令行下，之后再打开AXD调试软件。

另外要使用AXD必须首先要生成包含有调试信息的程序，上一节中已经生成的2440_led.axf就是含有调试信息的可执行ELF格式的映像文件。

1. 在AXD中打开调试文件

在菜单File中选择“Load image...”选项，打开Load Image对话框，找到要装载的.axf映像文件，点击“打开”按钮，就把映像文件装载到目标内存中了。



在所打开的映像文件中会有一个蓝色的箭头指示当前执行的位置。

在菜单Execute中选择“Go”，将全速运行代码。要想进行单步的代码调试，在Execute菜单中选择“Step”选项，或用F10即可以单步执行代码，窗口中蓝色箭头会发生相应的移动。

有时候，用户可能希望程序在执行到某处时，查看一些所关心的变量值，此时可以通过断点设置达到此要求。将光标移动到要进行断点设置的代码处，在Execute菜单中，选择“Toggle Breakpoint”或按F9，就会在光标所在位置出现一个实心圆点，表明该处为断点。

还可以在AXD中查看寄存器值，变量值，某个内存单元的数值等等。

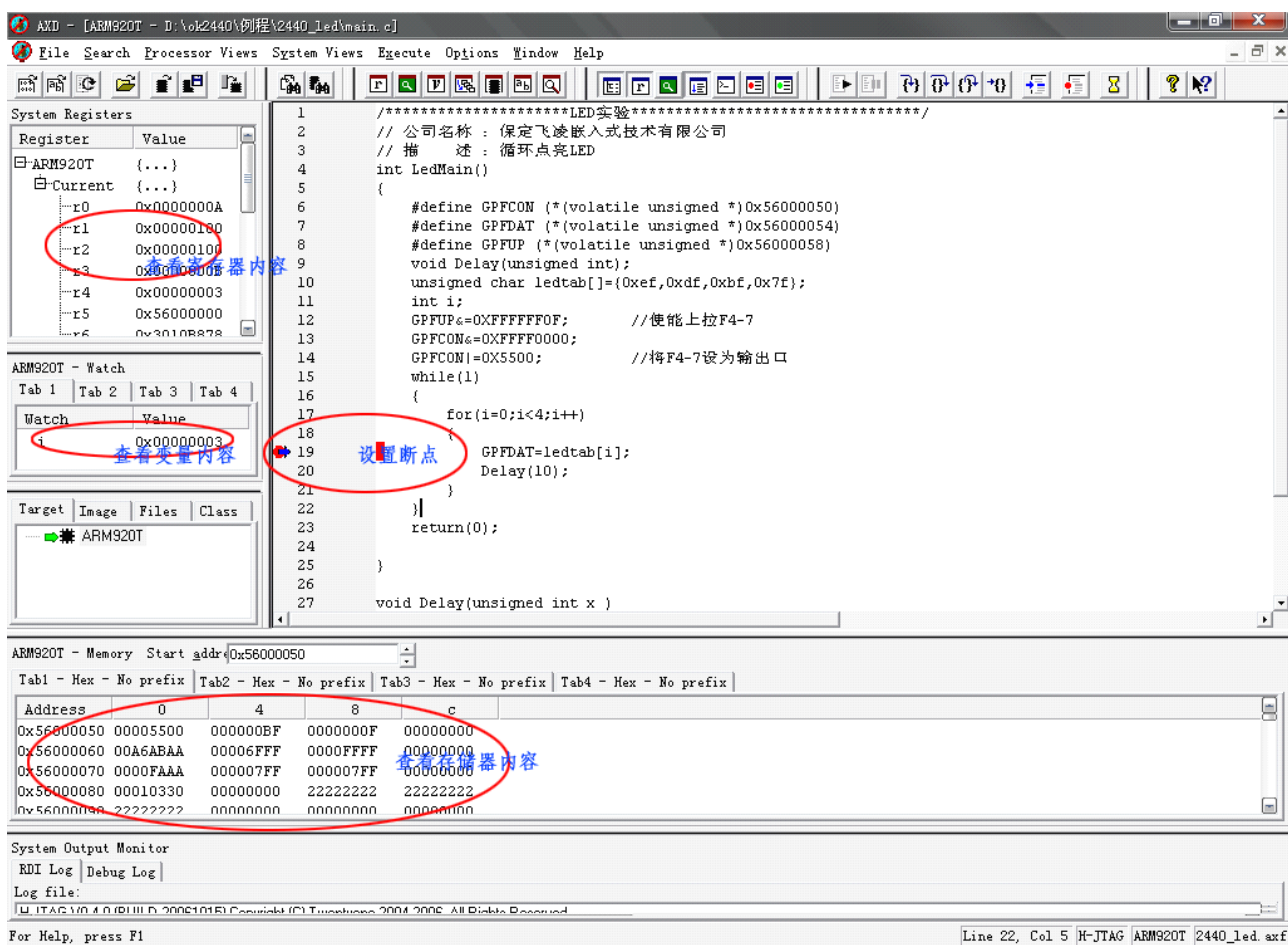
下面就结合本章中的例子，介绍在AXD中调试过程。

2. 查看存储器内容

从Processor Views菜单中选择“Memory”选项，在Memory Start address选择框中，用户可以根据要查看的存储器的地址输入起始地址，在下面的表格中会列出连续的64个地址。

3. 设置断点

可以在for循环体的“Delay(10);”语句处设置断点，将光标定位在该语句处，使用快捷键F9在此处设置断点，按F5键，程序将运行到断点处，如果读者想查看子函数Delay是如何运行的，可以在Execute菜单中选择“Step In”选项，或按下F8键，进入到子函数内部进行单步程序的调试。



4. 查看变量值

在Delay函数的内部,如果用户希望查看某个变量的值,比如查看变量i的值,可以在Processor Views菜单中选择“Watch”,然后用鼠标选中变量i,点击鼠标右键,在快捷菜单中选中“Add to watch”,这样变量i默认是添加到watch窗口的Tab1中。程序运行过程中,用户可以看到变量i的值在不断的变化。默认显示变量数值是以十六进制格式显示的,如果用户对这种显示格式不习惯的话,可以通过在watch窗口点击鼠标右键,在弹出的快捷菜单中选择“Format”选项,用户可以选择所查看的变量显示数据的格式。如果用户想从Delay函数中跳出到主函数中去,最简单的方法就是将光标定位到你想要跳转到的主函数处,在Execute菜单中选择“Run to Cursor”选项,则程序会从Delay函数中跳转到光标所在位置。