

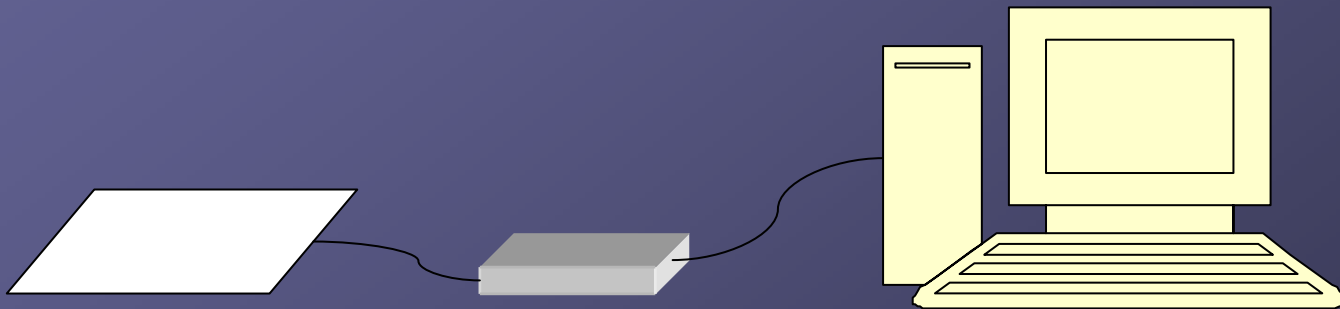
ARM的整个开发流程

- 系统连接关系
- ads介绍
- BOOT的编写过程
- ARM的实际应用
 - arm在电力继电保护中的应用
 - arm在军用雷达伺服系统设计中的应用
 - arm在仪器仪表中的应用
- 介绍一款芯片及其应用

ARM的整个开发流程

- ARM 公司的实时调试工具Multi-ice ,
- 集成开发环境ADS1.2
- 底层驱动程序Multi-ice sever

一，硬件连接



Multi-ice sever

- 启动Multi-ice的应用程序
- 从Windows的程序菜单中启动Multi-ice的服务程序，会显示出下面的图形界面来。

站内搜索

- Programs
- Protel 99 SE
- UltraEdit-32
- Windows Update
- WinZip
- 腾讯Explorer
- 腾讯QQ
- 打开 Office 文档
- 新建 Office 文档
- 程序(P)
- 文档(D)
- 设置(S)
- 搜索(C)
- 帮助(H)
- 运行(R)...
- 关机(U)...

- ARM Developer Suite v1.2
- ARM Multi-ICE v2.2
- Atmel AVR Tools
- PonyProg
- Tornado 2.2
- 附件
- 金山词霸 2002
- 用友财务及企管软件UFERP-M8.11
- SUPERPRO III For Win9x ME 2K NT or XP
- GNU C Compiler for ARM
- Bloodshed Dev-C++
- IAR Systems

带您遨游

奇妙数字世界

Multi-ICE Server

位置: D:\ARM

- 全数字电磁流量计
- 中文显示
- 红外遥控操作更方便

更多



傅立叶电子科技始终坚持跟踪电子业界最新尖端技术. 成功为军方客户开发了系列星上高速数字信号处理系统. 在数字技术领域积累了丰富的成功经验.

技术天地

定期刊登新技术资料, 欢迎各位大侠慷慨投稿

进入

文章精选

大容量Flash型
AT91系列ARM核
微控制器作者:
北理工徐...

32位处理器, 片内带2M巨大flash, 能加密, 片内ram大io口32根; 3个16位定时器; 2个串口, 可以支持跑操作系统开发时间缩短一半, 价格8.5美金

GO

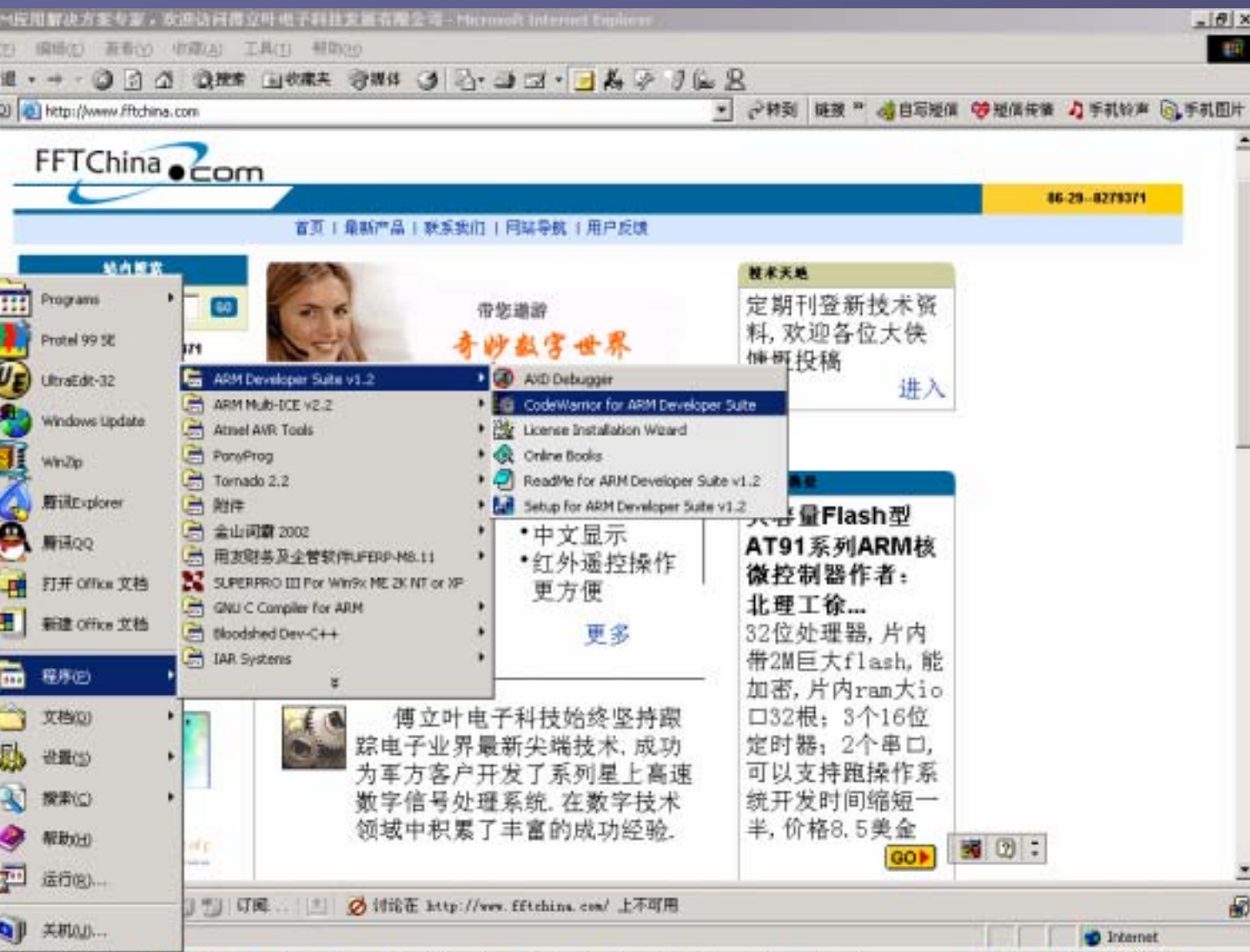
启动以后会出现以下画面：

注意菜单中服务程序已经自动监测到目标板上的arm芯片是arm7tdmi。

这一步表明仿真机、目标板、pc机都已经连接成功，可以进入调试状态了。

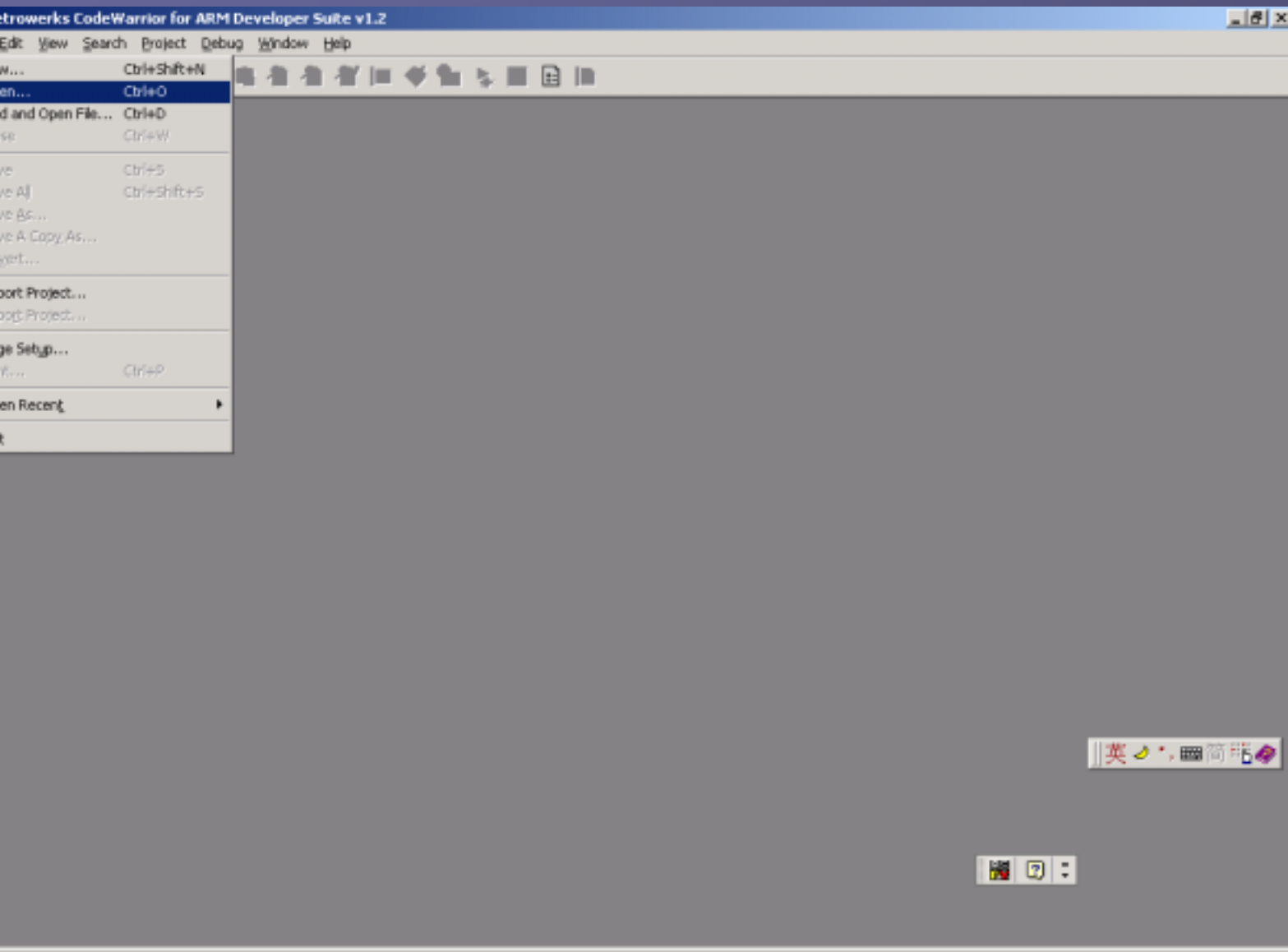


打开ads1.2的编辑编译环境

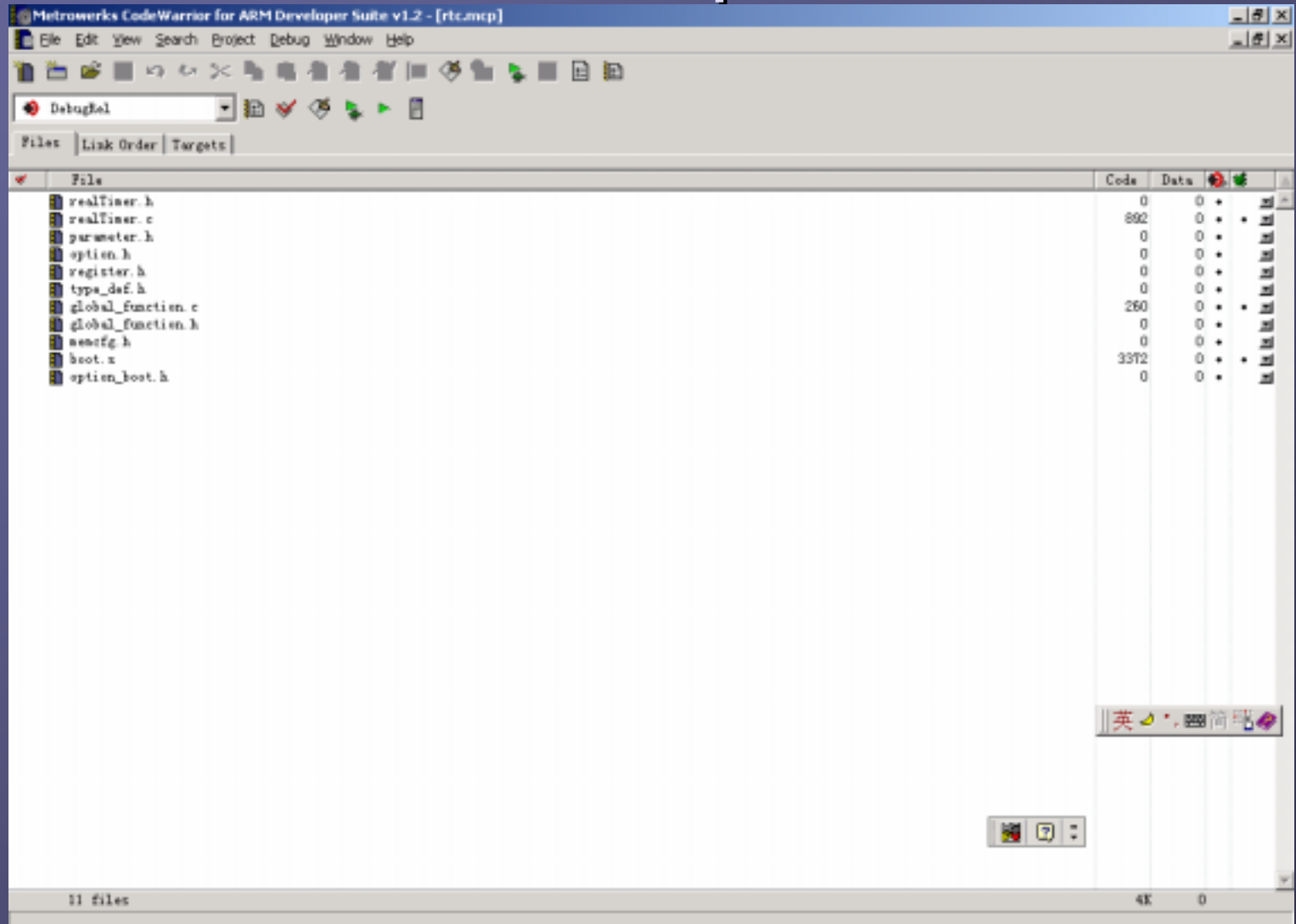


注意看ads包括两部分：
codewarrior
for arm、axo
调试器。

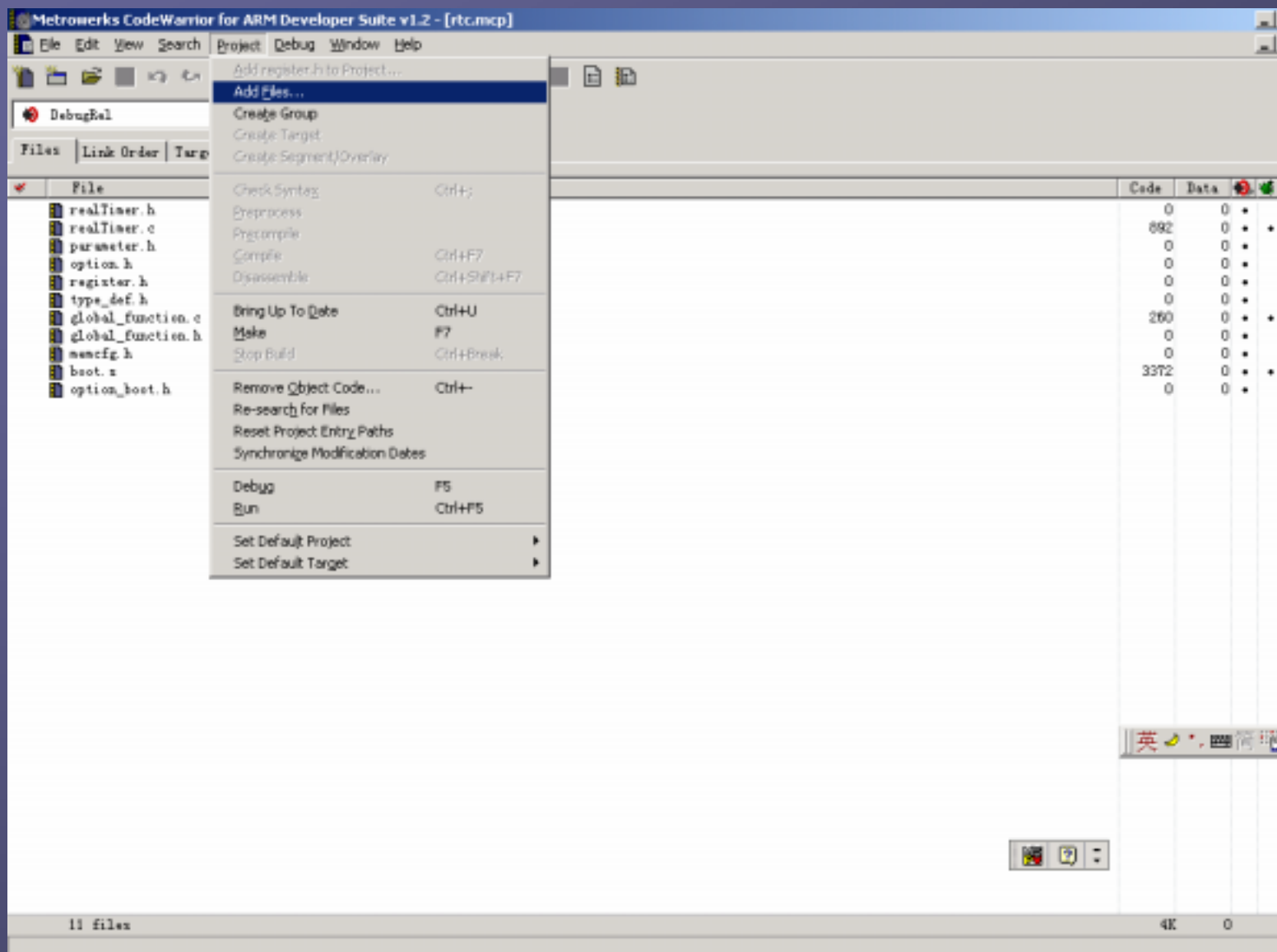
打开codewarrior后的画面见下



在文件中打开一个arm工程文件 rtc.mcp

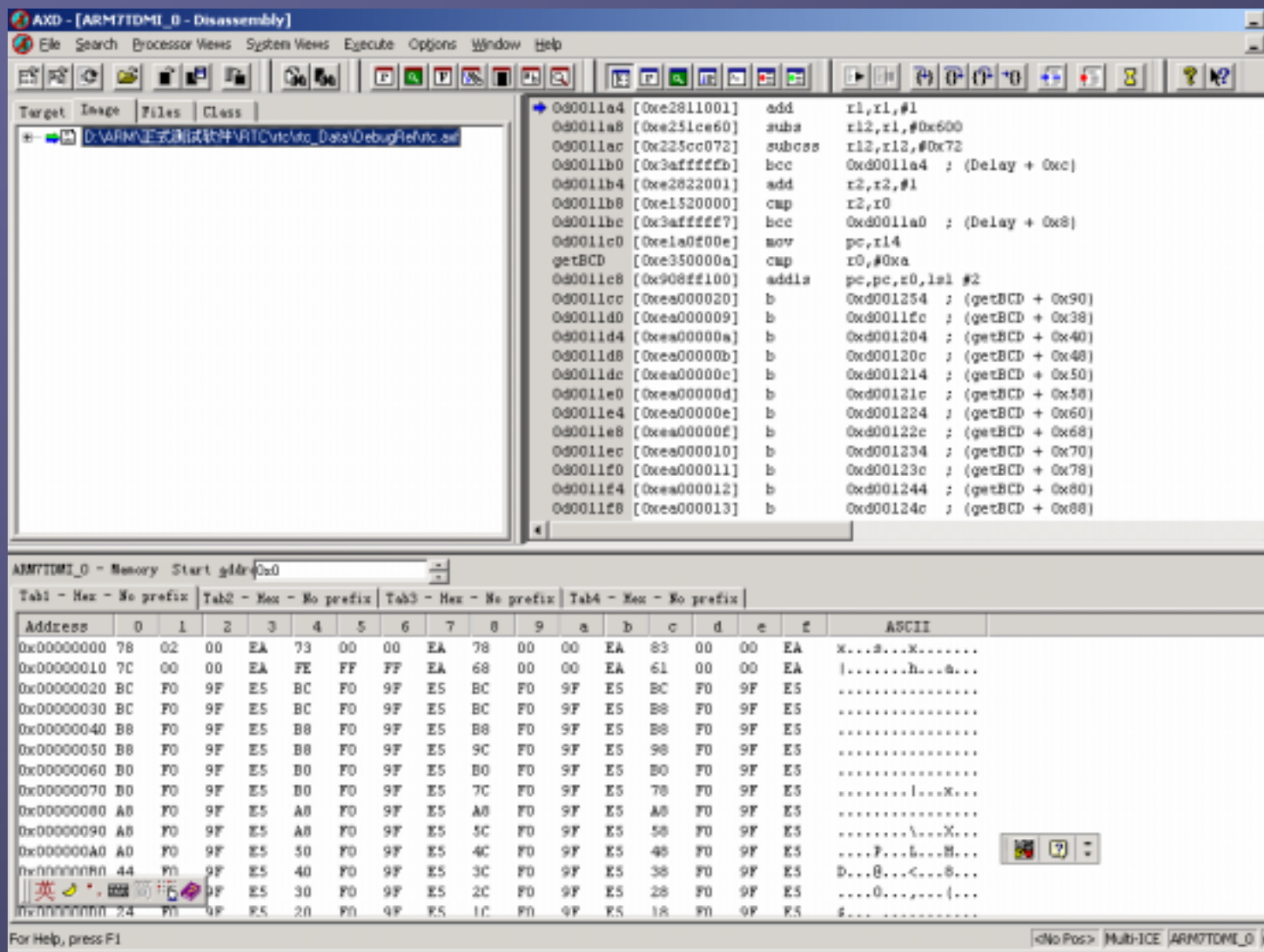


详细看工程文件下的文件群，共分为3类文件
文件可以通过project下的add files添加新文件到工程文件中来。



在project菜单下的make可以编译工程文件，形成可调试程序和目标文件。在编译成功之后，在project菜单下执行debug,自动将程序下载到调试器axd中。进入axd后的画面见下图：

在调试器下就可以调试我们的程序了。



二、BOOT的编写过程

- 什么是boot程序？ Boot程序实际就是采用arm汇编语言编写的针对具体板卡的初始化程序。
- 嵌入式系统在启动或复位的后，需要对系统硬件和软件运行环境进行初始化，这些工作由启动程序完成，通常启动程序都是用汇编语言书写的。写好启动程序是设计好嵌入式程序的关键，系统启动程序所执行的操作跟具体的目标系统和开发系统相关，一般流程如下：

● 1, 设置入口指针

启动程序首先必须定义入口指针, 而且整个应用程序只有一个入口指针。

AREA boot, CODE, READONLY ENTRY

在编译的时候, 编译器需要知道整个程序的入口在哪儿, 所以在编译前要设置好相关的编译选项, 如程序入口所在的目标文件 (开发板提供的环境中是boot.o), 文件中具体的模块区域 (这里就是boot)。

● 2, 设置中断向量

ARM7 要求中断向量表必须设置在从0 地址开始, 连续 8×4 字节的空间, 分别是复位、未定义指令错误、软件中断、预取指令错误、数据存取错误、IRQ、FIQ和一个保留的中断向量。如果用户要使用CPU 支持的硬件中断方式, 还需要按要求在硬件中断向量表的地址上进行正确设置。对于未使用的中断, 使其指向一个只含返回指令的哑函数, 可以防止错误中断引起系统的混乱。

3，初始化堆栈

系统堆栈初始化取决于用户使用了哪些处理器模式，以及系统需要处理哪些错误类型。对于将要用到的每一种模式，都应该先定义好堆栈指针。

上面程序中用到的Stack 指针都是在数据区定义的地址标号，代表对应堆栈的起始地址，不同处理器模式下的sp 寄存器在物理上是不同的，所以上面的程序一共初始化了5 个不同的堆栈寄存器。

开发板的堆栈位于外部SDRAM 的底部，中断向量表前面。每个堆栈分配了256bytes 的空间。

```
^ (_ISR_STARTADDRESS-0x500) ; _ISR_STARTADDRESS =  
    0xdff_ff00
```

```
UserStack      # 256
```

```
SVCStack       # 256
```

```
UndefStack     # 256
```

```
AbortStack     # 256
```

```
IRQStack       # 256
```

```
FIQStack       # 0
```

4，初始化存储器系统

有些芯片可通过寄存器编程初始化存储器系统，而对于较复杂系统通常集成有MMU 来管理内存空间。Arm7tdmi通过存储器控制模块的配置寄存器来初始化存储器系统。

```
ldr    r0,=SMRDATA
```

```
ldmia  r0,{r1-r13}
```

```
ldr    r0,=BWSCON ; memory controller
```

```
stmia  r0,{r1-r13}
```

注意S3C44B0X 要求存储器模块的配置必须全部13 个寄存器连续写入（许多CPU 都有类似要求），所以一定要使用上面的多字节操作指令来完成。BWSCON 是寄存器组的开始地址，SMRDATA 是存放参数的数据区起始地址，13 个32 位的参数必须要连续放置，如下所示：

SMRDATA DATA

DCD 0x22222220

DCD

((B0_Tacs<<13)+(B0_Tcos<<11)+(B0_Tacc<<8)+(B0_Tcoh<<6)+(B0_Tah<<4)+(B0_Tacp<<2)+(B0_PMC))

DCD

((B1_Tacs<<13)+(B1_Tcos<<11)+(B1_Tacc<<8)+(B1_Tcoh<<6)+(B1_Tah<<4)+(B1_Tacp<<2)+(B1_PMC))

DCD

((B2_Tacs<<13)+(B2_Tcos<<11)+(B2_Tacc<<8)+(B2_Tcoh<<6)+(B2_Tah<<4)+(B2_Tacp<<2)+(B2_PMC))

DCD

((B3_Tacs<<13)+(B3_Tcos<<11)+(B3_Tacc<<8)+(B3_Tcoh<<6)+(B3_Tah<<4)+(B3_Tacp<<2)+(B3_PMC))

DCD

((B4_Tacs<<13)+(B4_Tcos<<11)+(B4_Tacc<<8)+(B4_Tcoh<<6)+(B4_Tah<<4)+(B4_Tacp<<2)+(B4_PMC))

DCD

((B5_Tacs<<13)+(B5_Tcos<<11)+(B5_Tacc<<8)+(B5_Tcoh<<6)+(B5_Tah<<4)+(B5_Tacp<<2)+(B5_PMC))

DCD ((B6_MT<<15)+(B6_Tred<<2)+(B6_SCAN))

DCD ((B7_MT<<15)+(B7_Tred<<2)+(B7_SCAN))

DCD ((REFEN<<23)+(TREFMD<<22)+(Trp<<20)+(Tre<<18)+(Tchr<<16)+REFCNT)

DCD 0x17

DCD 0x20

5，如有必要改变处理器模式、状态

如果前面已经进行了堆栈的初始化，那么堆栈初始化时的最后一个模式就是现在的处理器运行模式，用户如果有需要改编处理器模式和其它如中断使能这些状态位的，可以设置CPSR 来实现。

6, 初始化必要的I/O 状态

某些严格的I/O 和用户认为需要在调用主程序前完成的状态控制, 可以在启动程序里面完成初始化, 特别是一些输出设备, 上电后往往呈现一种随机态, 需要及时加以控制。

7, 初始化C 语言所需的存储器空间

为正确运行应用程序，在初始化期间应将系统需要读写的数据和变量从ROM拷贝到RAM 里；一些要求快速响应的程序，如中断处理程序，也需要在RAM 中运行；如果使用FLASH，对FLASH 的擦除和写入操作也一定要在RAM 里运行。

处理程序变量的时候分两种，一种是有初始值的，需要把初始值一起拷贝；一种是没有初始值的，则统一放在一个初始化为0 的区域里面。

8, 呼叫C 程序

在进入主程序之前，需要确定主程序代码的编译模式是ARM 还是THUMB，由此决定相应的跳转指令。如果使用ARM 模式，假定用户主程序入口函数是main（）那么就写成下面这样：

```
BL main ; jump to C program
```

注意main 作为不在启动程序中定义的使用变量，要在前面引用（import）进来。

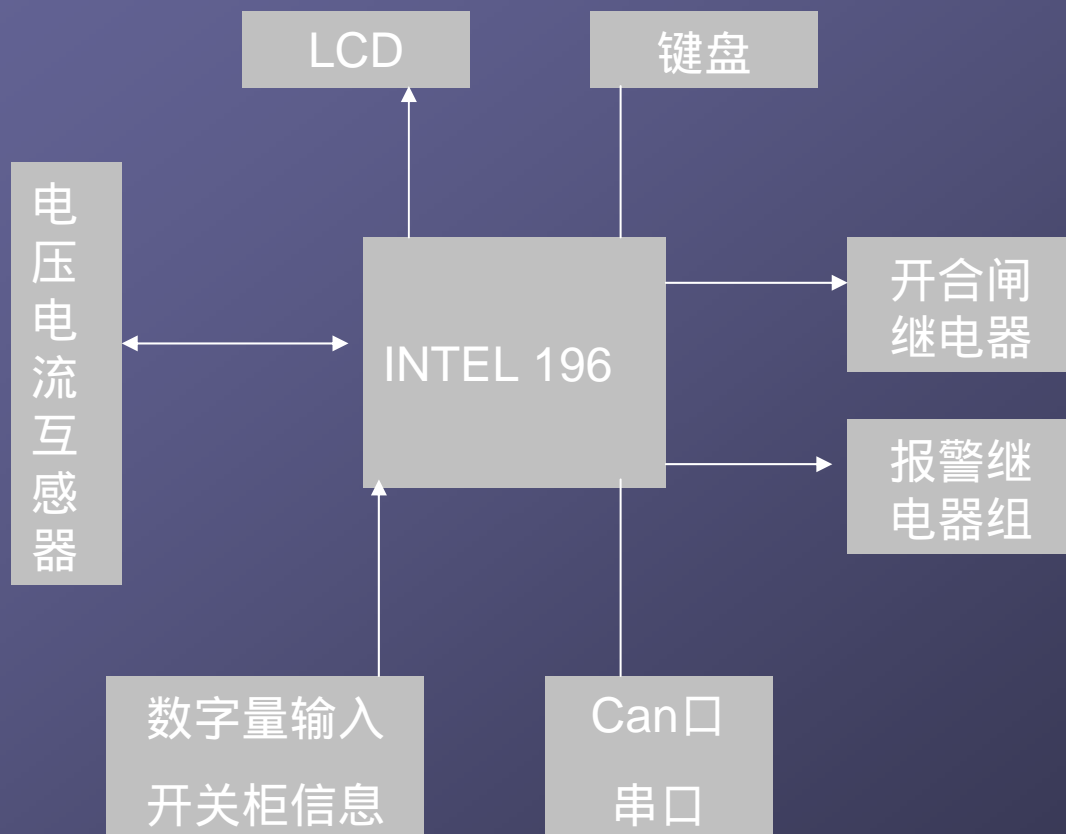
& 上面各步操作的操作并非固定不变。关于ARM 汇编的语法请参考相关的资料。

ARM的实际应用

- arm在电力继电保护中的应用
- arm在军用雷达伺服系统设计中的应用
- arm在仪器仪表中的应用

arm在电力继电保护中的应用

● 常见继电保护产品框图 (35千伏)



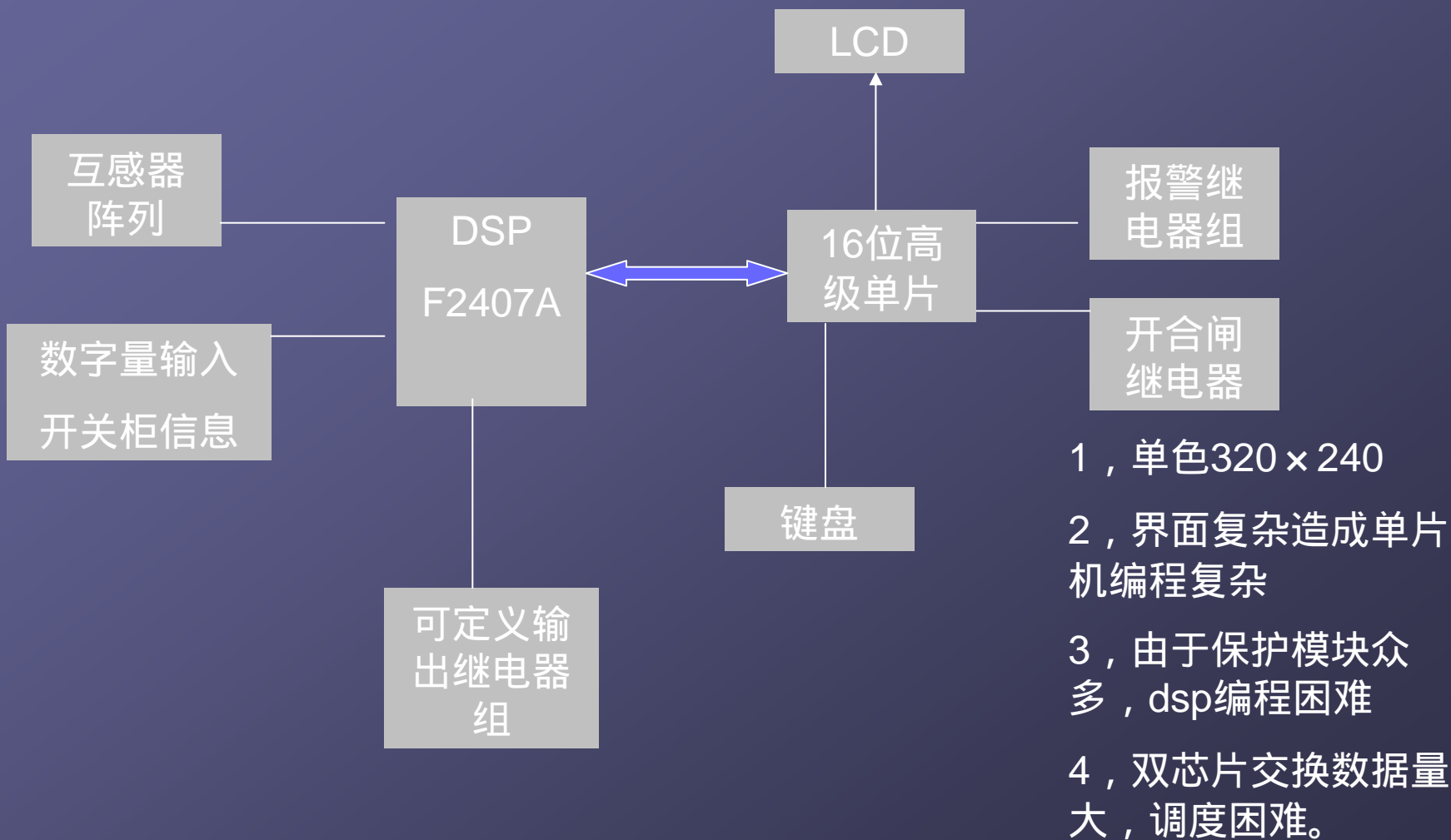
1, 人机对话内容少
128 × 64单色

2, 保护类型简单

3, 精度低

4, 功能有限

高级智能继电保护产品



采用arm920t设计

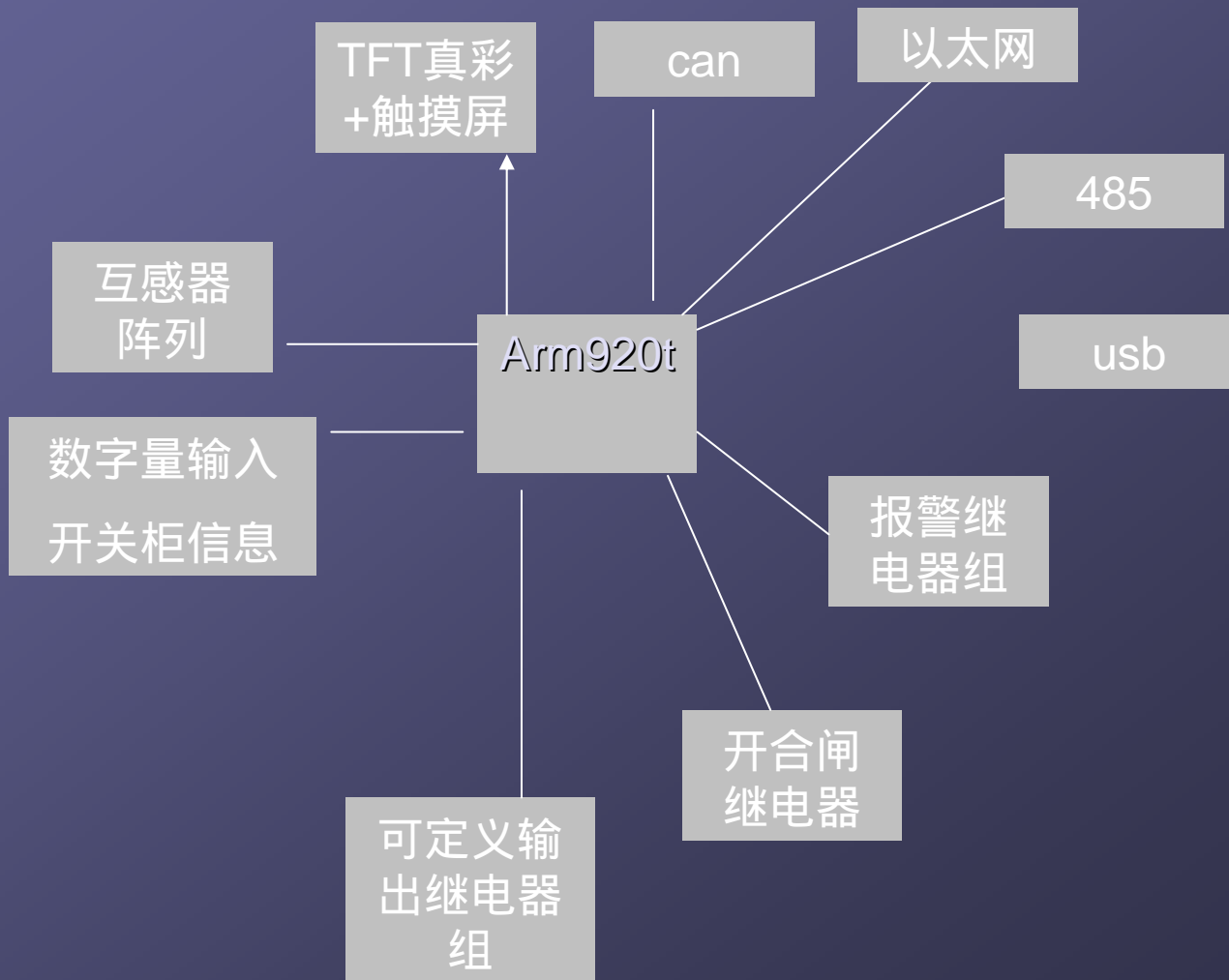
采用win ce嵌入式操作系统，编写界面更为友好，更适合windows风格，而设计难度大大降低，用vb就能够完成编程任务。

，无需编写网络驱动程序即可简单接入国际互联网。

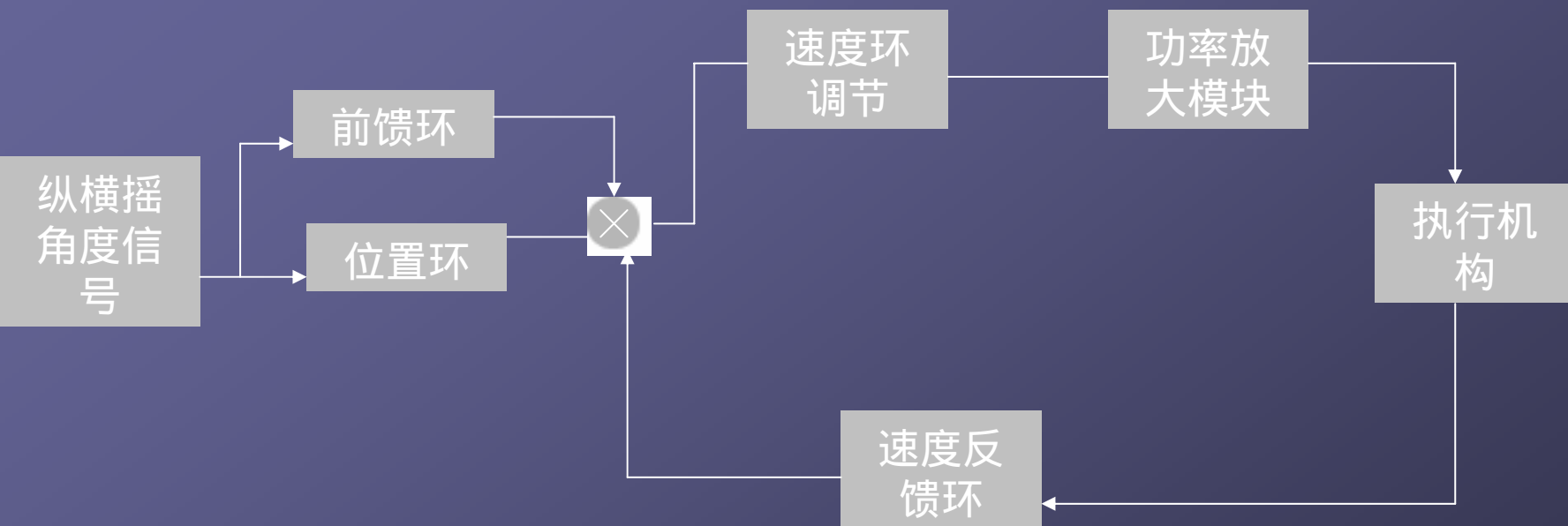
，通过互联网可以远程升级应用程序，检查设备运行情况。

，具有大容量数据区可以保存大容量的有效数据，可以完成长时间的录波。

，真彩显示和触摸屏



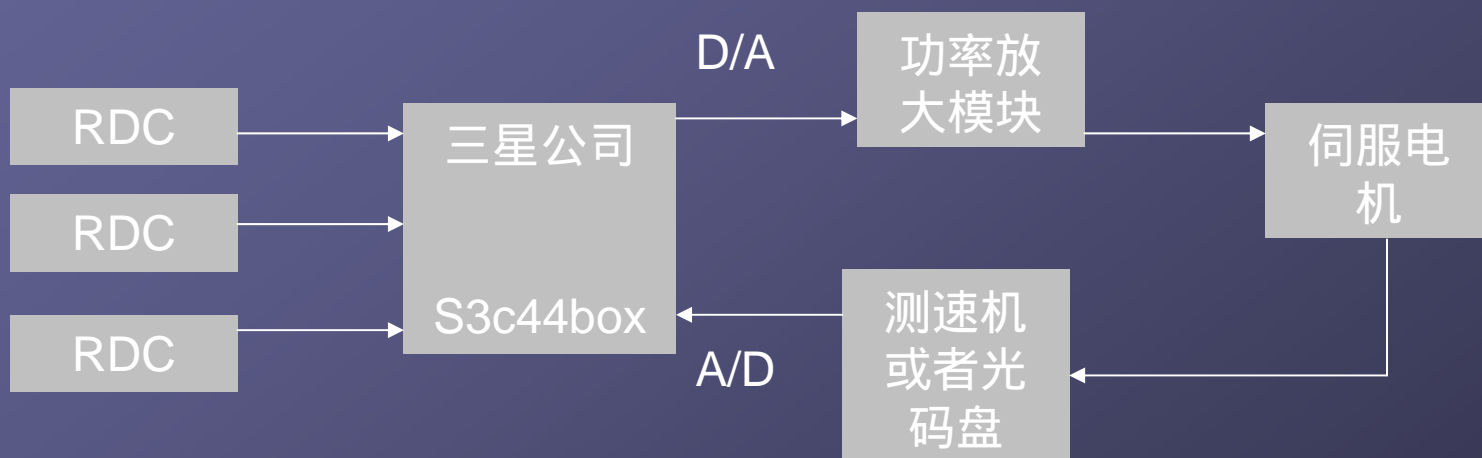
ARM在雷达伺服系统中的应用



传统硬件处理

- 1，前馈环和位置环采用80c196进行解算。
- 2，速度环采用dsp来完成。
- 3，功率模块采用IGBT来驱动伺服电机。

ARM在雷达伺服系统中的应用

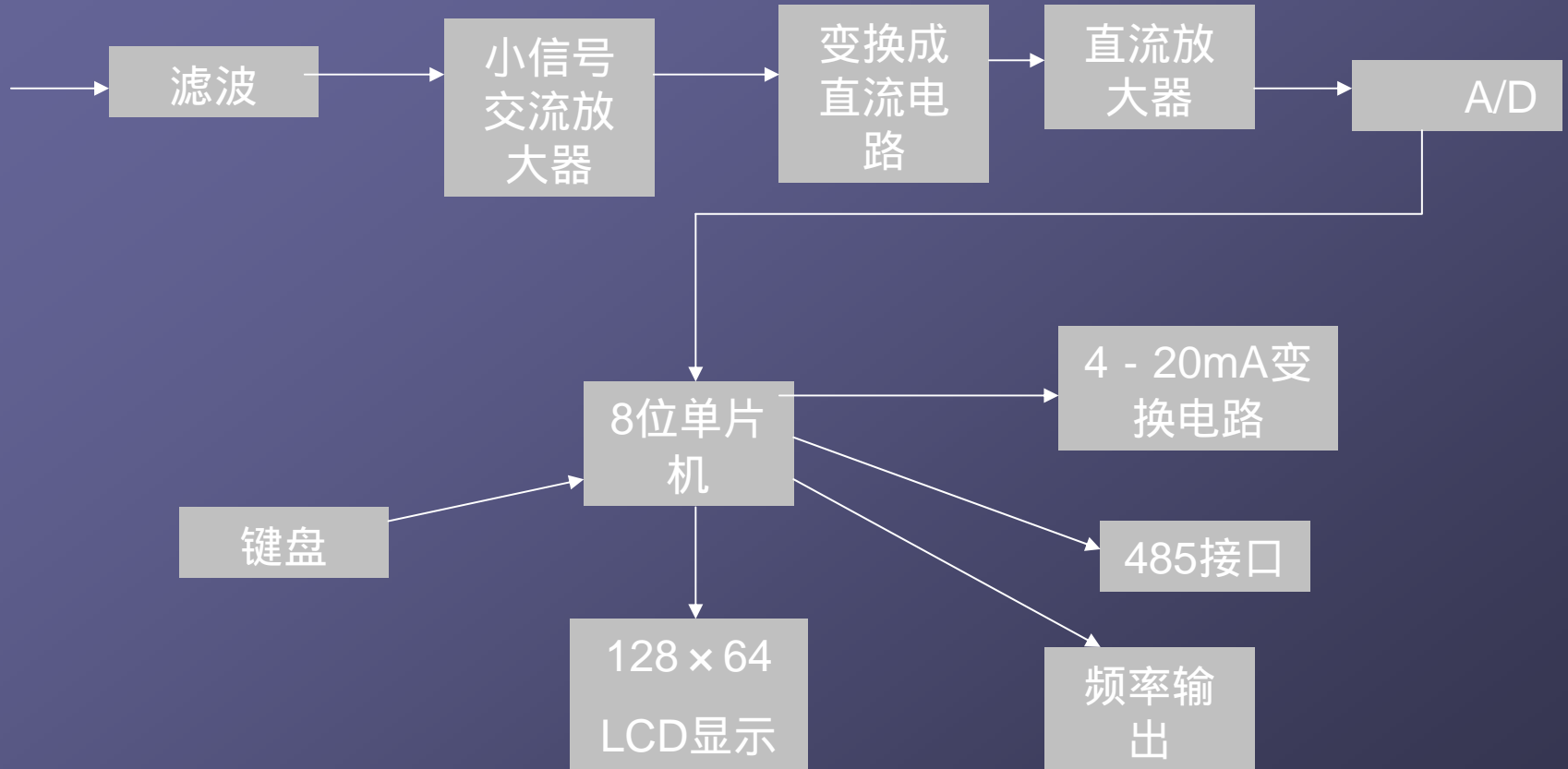


低功耗，高速，单片应用解决方案

由于ARM片内集成大量的I/O口，大量的定时器，大量的中断，多路A/D,可采用单芯片完成以上工作。这种方案集成度很高系统运行非常稳定。

系统特点：抗干扰能力强，可靠性高。

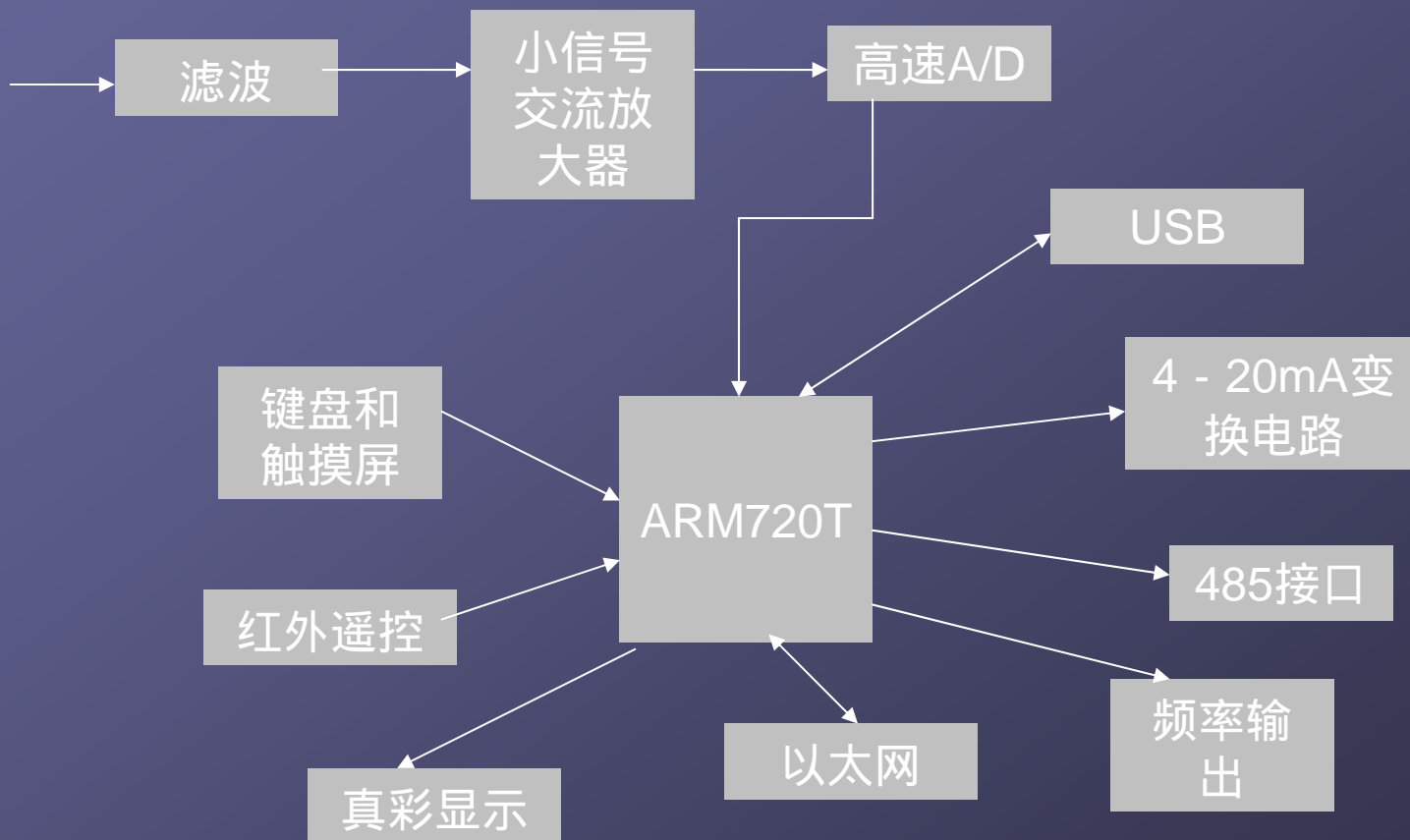
ARM在智能仪表的应用 - - 传统流量计的设计



新一代智能仪表的需求

- 1，保持低功耗
- 2，仪表的网络化，完成远程抄表，远程管理。
- 3，人机对话界面更为美观更为人性化。
- 4，现场功能要求更为丰富，打印，红外遥控，无纸记录仪。年曲线，月曲线显示等等。

ARM在智能仪表的应用



ARM在智能仪表的应用 - 优点

- 1, 低功耗, 内核1.8V, 外围3.3V。
- 2, 由于高速ARM提供足够运算能力, 传统的前端模拟电路大部分被数字化, 大大简化了整机电路设计。仪表调试工作简化, 仪表的一致性更好。
- 3, 采用win ce系统, 可以编程更为优秀的windows风格的人机界面。
- 4, 通过以太网化, 实现仪表网络化的管理。

谢谢！