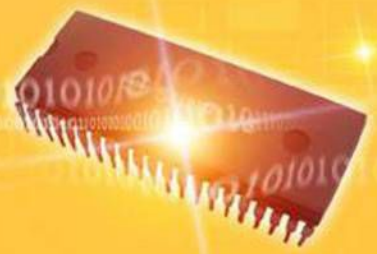
嵌入式系统工程师



嵌入式系统概述

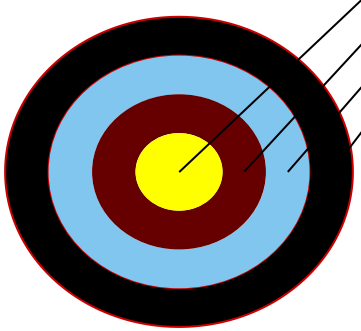
大纲

- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210试验仪介绍
- s5pv210存储器组成结构
- 裸机开发调试

嵌入式系统的组成

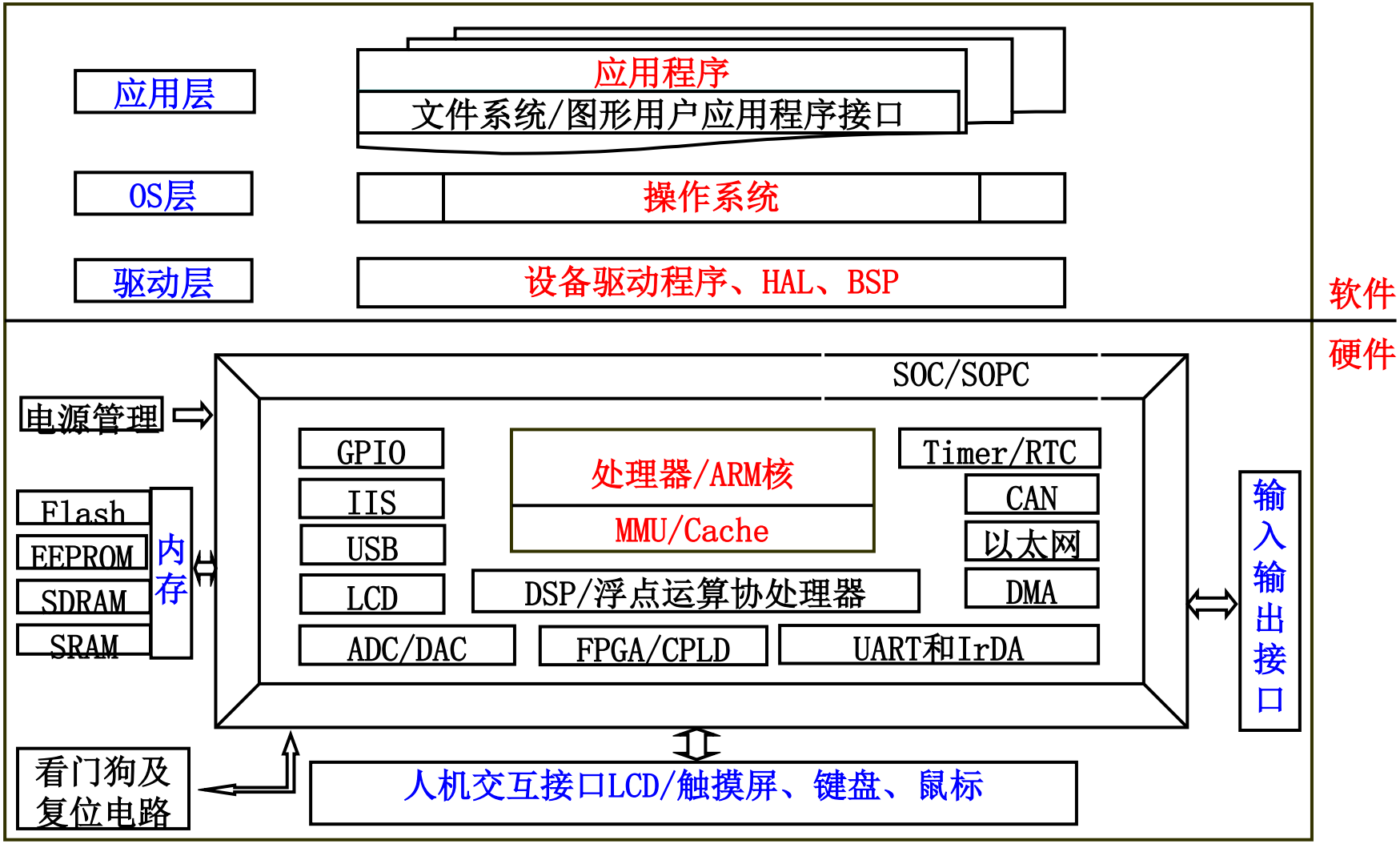
- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210试验仪介绍
- s5pv210存储器组成结构
- 裸机开发调试

嵌入式系统的组成



嵌入式微处理器
外围硬件设备
嵌入式操作系统
用户应用程序

嵌入式系统的组成



s5pv210系统主要资源

- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210试验仪介绍
- s5pv210寻址空间介绍
- 裸机开发调试

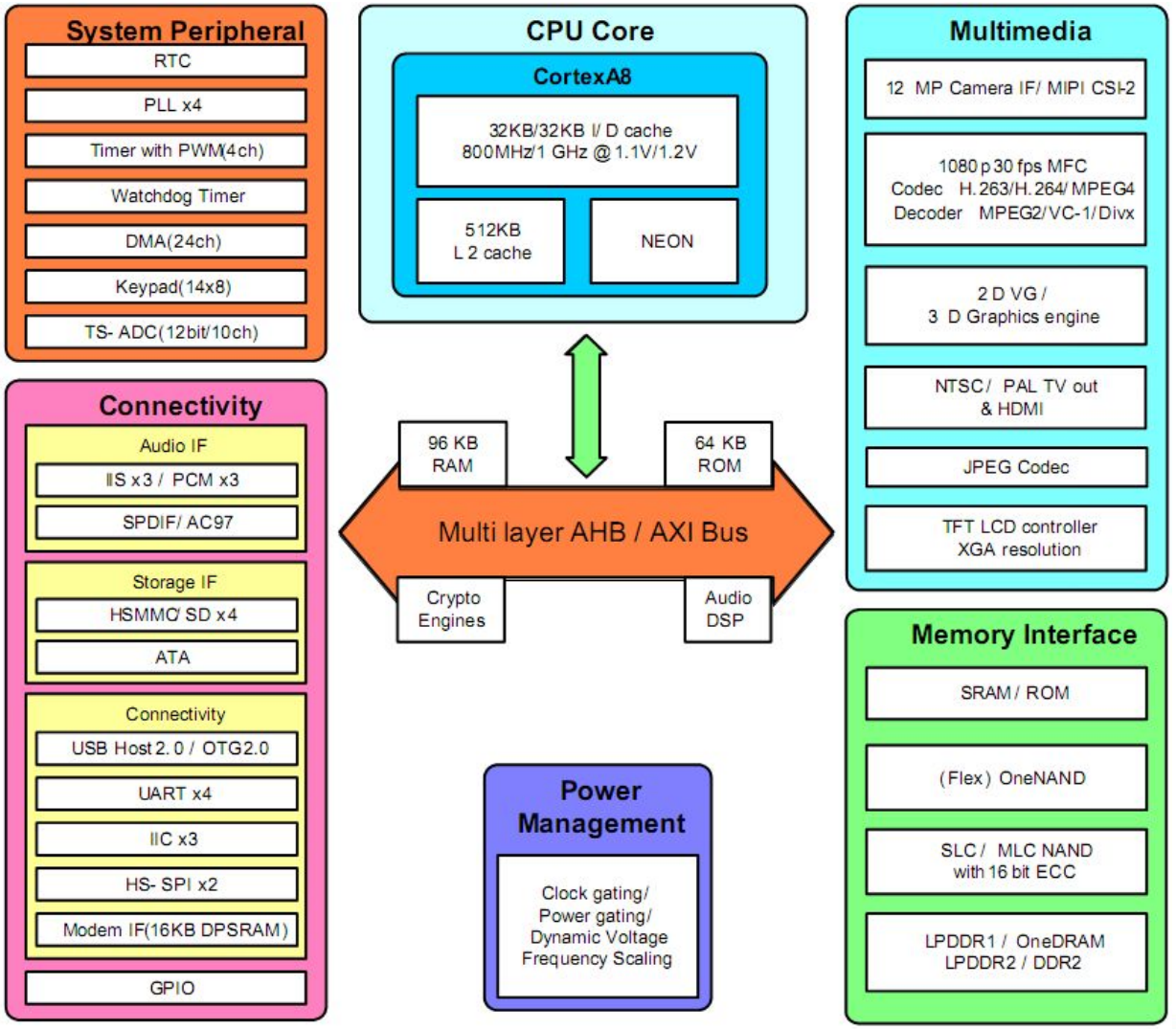
s5pv210系统主要资源

- s5pv210是三星公司推出的32位RISC微处理器，其CPU采用的是ARM Cortex-A8内核，基于ARMv7架构

- 丰富的片内资源，为手持设备和其它移动领域应用，提供了低价格、低功耗、高性能的微处理器解决方案。

- s5pv210结构框图如下：

s5pv210七大资源模块

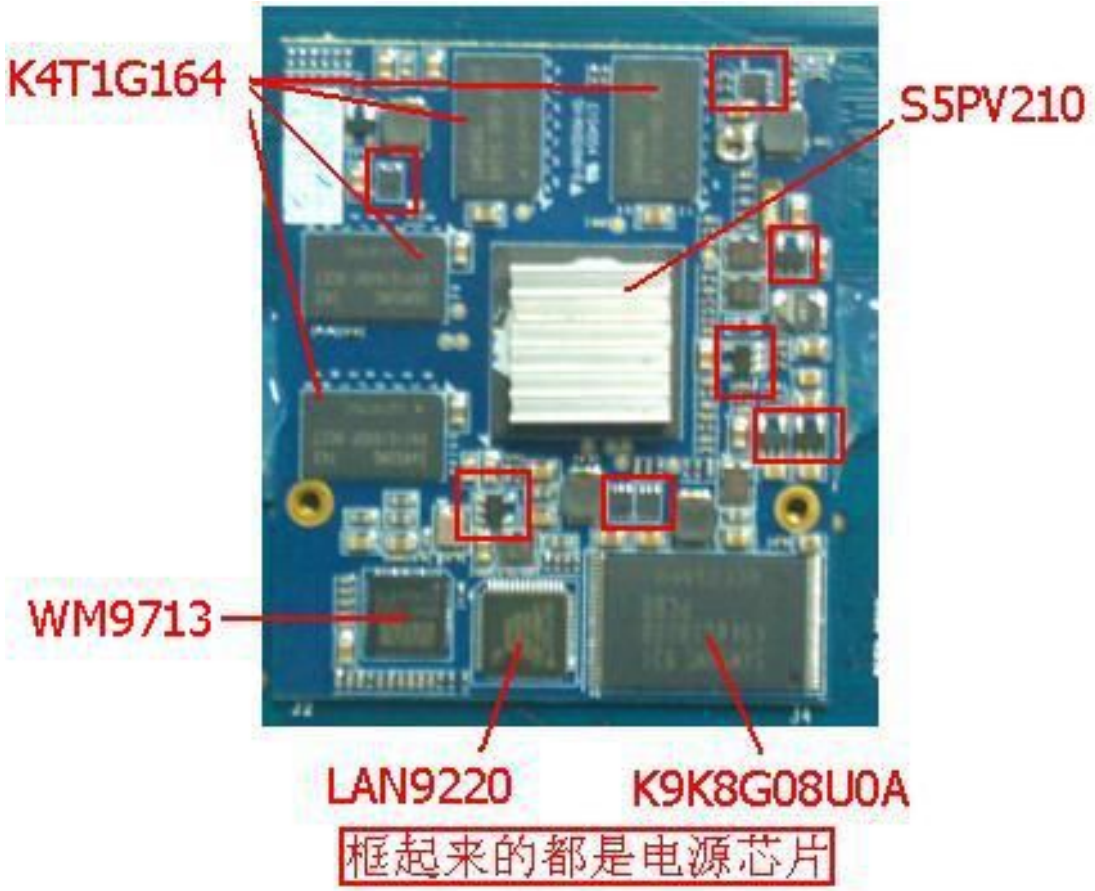


s5pv210实验仪介绍

- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210试验仪介绍
- s5pv210存储器组成结构
- 裸机开发调试

s5pv210实验仪介绍

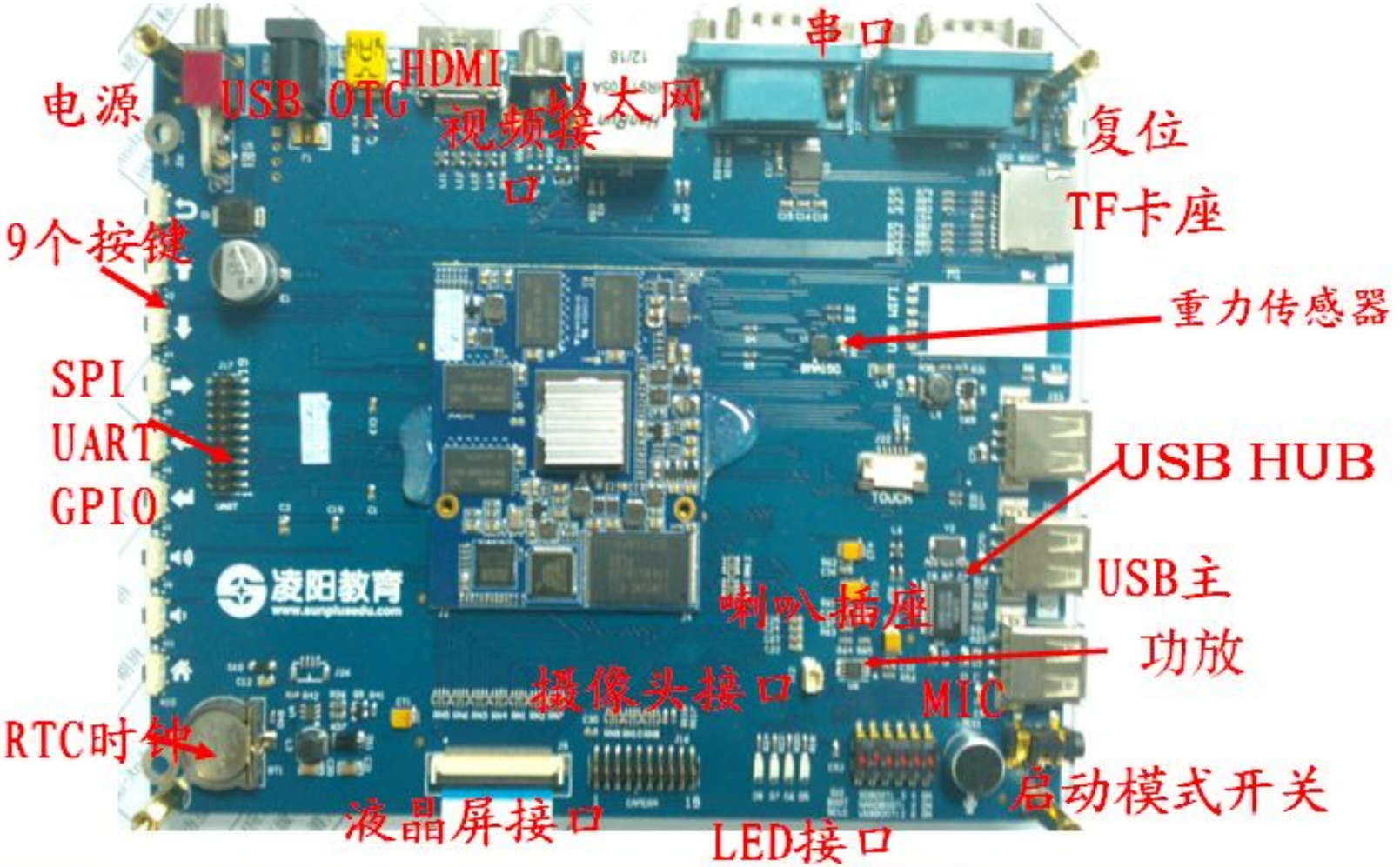
➢ s5pv210核心板资源介绍：



s5pv210实验仪介绍

- s5pv210核心板资源介绍：
  - 处理器，CPU采用samsung s5pv210
  - 内存采用DDR2 RAM(K4T1G164)
  - Nand flash，采用samsung K9K8G08U0B,SLC结构，大小为1Gx8bit
  - 网卡，选用支持10/100Mbps、小型、电压可变的以太网芯片LAN9220
  - 音频编解码芯片，采用AC97接口的WM9713
  - 电源芯片，为核心板各模块提供所需的各种电压转换

s5pv210实验仪介绍



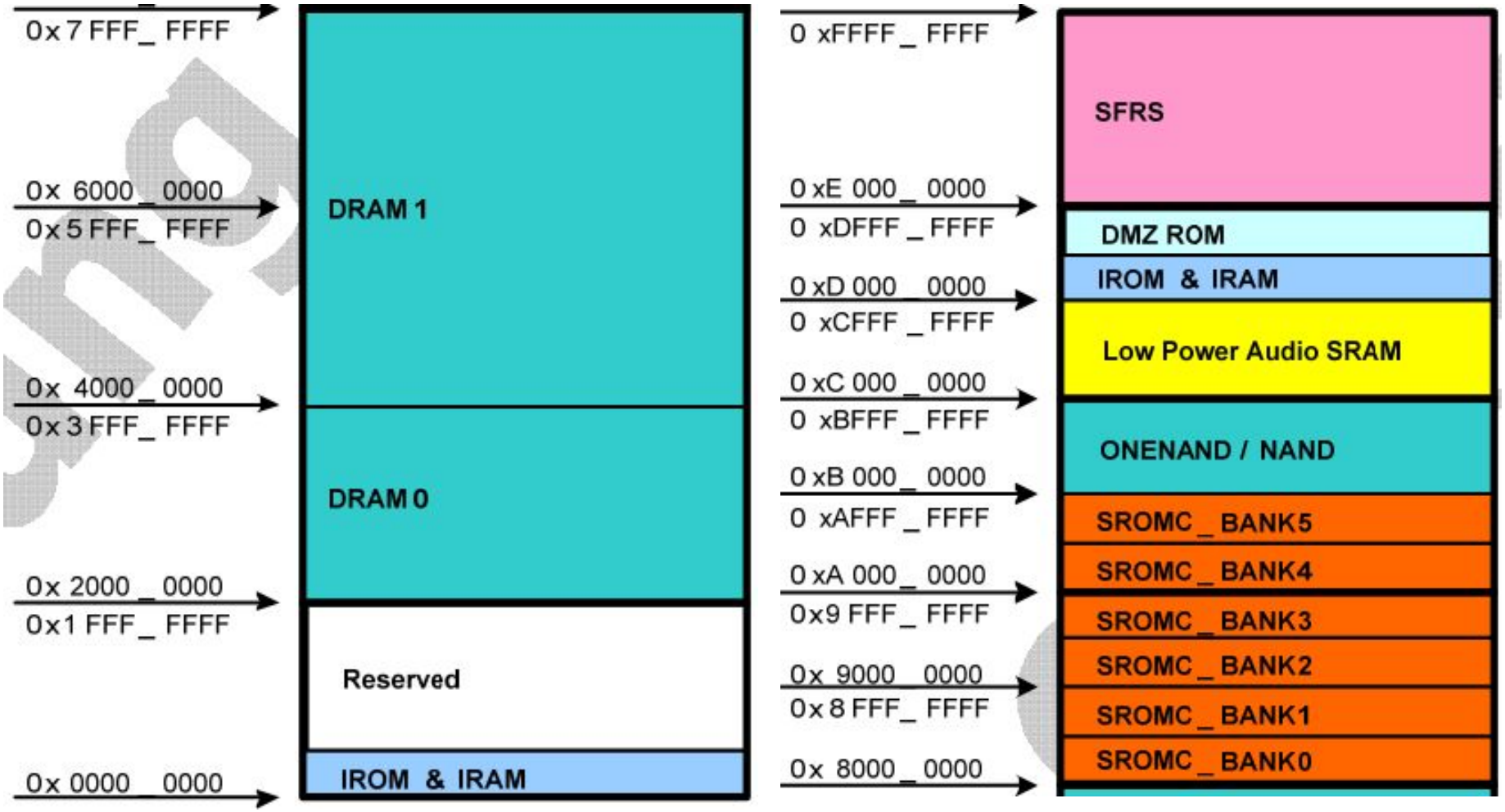
s5pv210寻址空间介绍

- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210试验仪介绍
- s5pv210寻址空间介绍
- 裸机开发调试

s5pv210寻址空间介绍

- s5pv210寻址空间采用统一编址方式进行管理
- 寻址空间映射图：

s5pv210寻址空间介绍



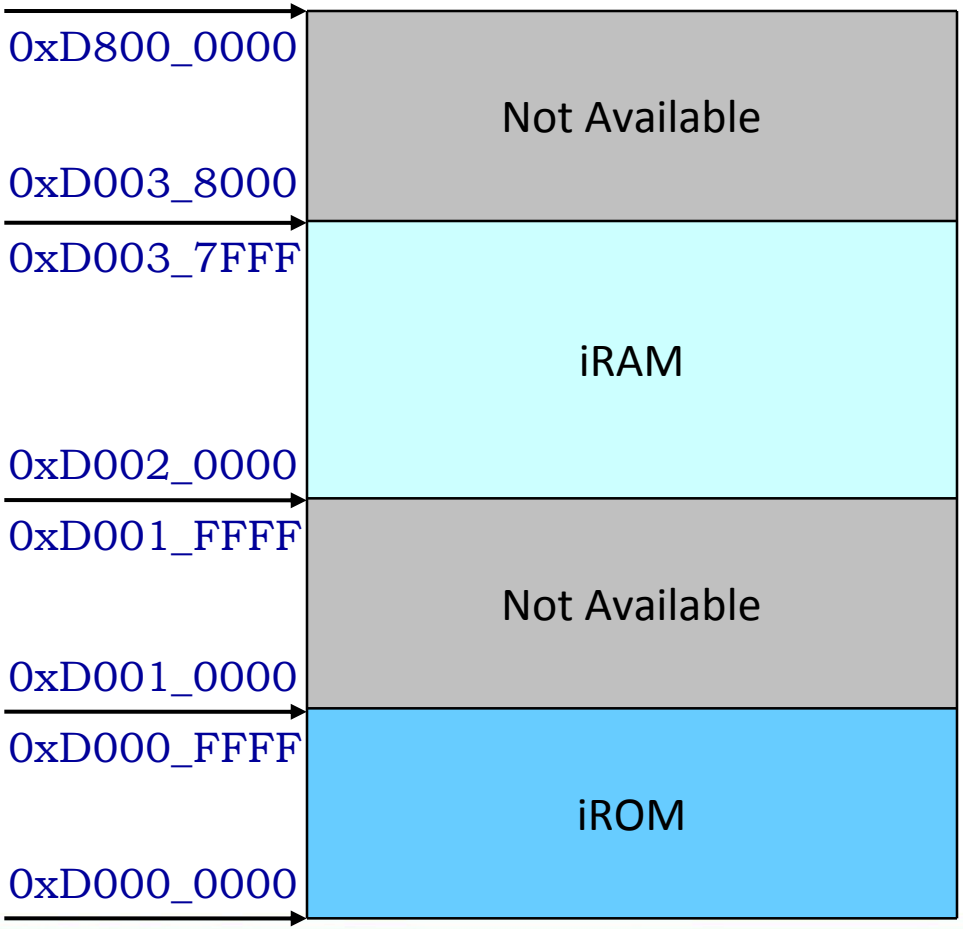
s5pv210寻址空间介绍

2.1.1 DEVICE SPECIFIC ADDRESS SPACE

Address		Size	Description	Note
0x0000_0000	0x1FFF_FFFF	512MB	Boot area	Mirrored region depending on the boot mode.
0x2000_0000	0x3FFF_FFFF	512MB	DRAM 0	
0x4000_0000	0x7FFF_FFFF	1024MB	DRAM 1	
0x8000_0000	0x87FF_FFFF	128MB	SROM Bank 0	
0x8800_0000	0x8FFF_FFFF	128MB	SROM Bank 1	
0x9000_0000	0x97FF_FFFF	128MB	SROM Bank 2	
0x9800_0000	0x9FFF_FFFF	128MB	SROM Bank 3	
0xA000_0000	0xA7FF_FFFF	128MB	SROM Bank 4	
0xA800_0000	0xAFFF_FFFF	128MB	SROM Bank 5	
0xB000_0000	0xBFFF_FFFF	256MB	OneNAND/NAND Controller and SFR	
0xC000_0000	0xCFFF_FFFF	256MB	MP3_SRAM output buffer	
0xD000_0000	0xD000_FFFF	64KB	IROM	
0xD001_0000	0xD001_FFFF	64KB	Reserved	
0xD002_0000	0xD003_7FFF	96KB	IRAM	
0xD800_0000	0xDFFF_FFFF	128MB	DMZ ROM	
0xE000_0000	0xFFFF_FFFF	512MB	SFR region	

s5pv210寻址空间介绍

- iROM和iRAM具体分布：



s5pv210寻址空间介绍

- s5pv210实验仪动态内存大小为512MByte，对应的地址空间落在0x30000000-0x4FFFFFFF，也就是DRAM0的后256MByte和DRAM1的前256MByte,一共512M的DDR2 RAM空间
- 0xE0000000以后的地址空间作为各种特殊功能寄存器地址
- 特殊功能寄存器被分组映射到内存中各地址空间，用户可以像操作内存一样操作各功能模块的配置寄存器，部分截图：

s5pv210寻址空间介绍

2.1.2 SPECIAL FUNCTION REGISTER MAP

Address		Description
0xE000_0000	0xE00F_FFFF	CHIPID
0xE010_0000	0xE01F_FFFF	SYSCON
0xE020_0000	0xE02F_FFFF	GPIO
0xE030_0000	0xE03F_FFFF	AXI_DMA
0xE040_0000	0xE04F_FFFF	AXI_PSYS
0xE050_0000	0xE05F_FFFF	AXI_PSFR
0xE060_0000	0xE06F_FFFF	TZPC2
0xE070_0000	0xE07F_FFFF	IEM_APC
0xE080_0000	0xE08F_FFFF	IEM_IEC
0xE090_0000	0xE09F_FFFF	PDMA0
0xE0A0_0000	0xE0AF_FFFF	PDMA1
0xE0D0_0000	0xE0DF_FFFF	CORESIGHT
0xE0E0_0000	0xE0EF_FFFF	SECKEY

s5pv210资源介绍

- 嵌入式系统的组成
- s5pv210系统主要资源
- s5pv210存储器组成结构
- s5pv210试验仪介绍
- 裸机开发调试

裸机开发调试

- 裸机工程简介：
  - 常用GNU工具介绍
  - 链接和链接脚本
  - 裸机程序编译过程简介

GNU常用工具的使用

- 常用GNU工具包括：
  - 预处理器　　cpp
  - C编译器　　gcc
  - C++编译器　　g++
  - 汇编器　　as
  - 链接器　　ld
  - 二进制工具集　objcopy、objdump、……

- 下面介绍几个常用的工具：
  - ls /usr/local/arm/4.3.2/bin/

GNU常用工具的使用

- nm:符号显示器
  - 显示符号 $nm -n main_elf

```
@sunplusedu$
@sunplusedu$pwd
/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/usr/lib
@sunplusedu$arm-linux-nm -n crt1.o
         U __libc_csu_fini
         U __libc_csu_init
         U __libc_start_main
         U abort
         U main
00000000 R _IO_stdin_used
00000000 D __data_start
00000000 T _start
00000000 W data_start
@sunplusedu$
```

GNU常用工具的使用

- 各段含义见下页图：

段	描述
b/B	.bss(b静态/B非静态)未初始化变量
d/D	.data(d静态/D非静态)已初始化变量
r/R	.rodata(r静态/R非静态)只读数据段
t/T	.text(t静态/T非静态)函数
A	不可改变的绝对值
C	.o中未初始化非静态变量
N	调试用的符号
U	表示符号只有声明没有定义

GNU常用工具的使用

- objdump:信息查看器
  - 查看所有段信息 $objdump -h main_elf
  - 查看文件头信息 $objdump -f main_elf
  - 查看反汇编信息 $objdump -d main_elf
  - 查看内嵌反汇编 $objdump -S -d main_elf
- objcopy:段剪辑器
  - 去除elf格式信息

    $objcopy -O binary -S main_elf main.bin

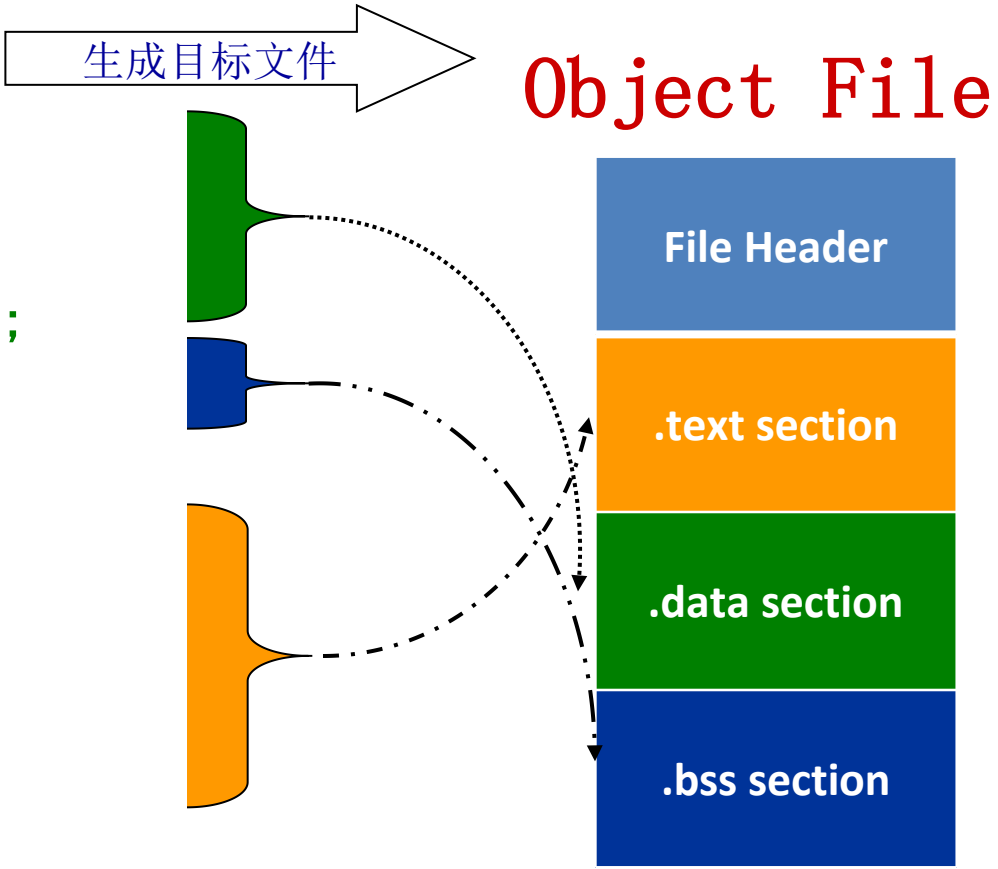
程序的编译链接过程

什么是链接呢？

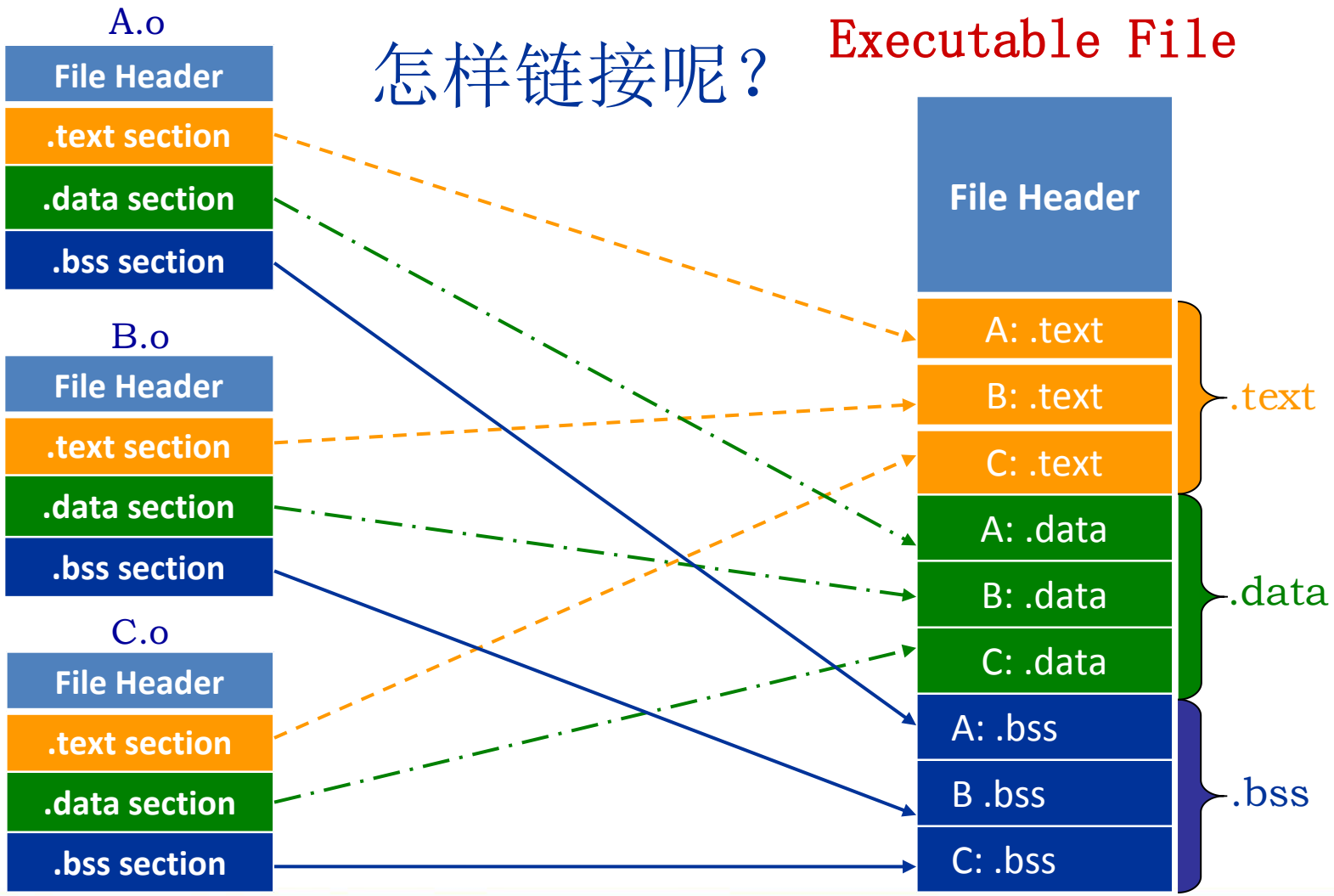
C code

```
int b = 0x12;
const int c = 0x34;
static int d = 0x56;
char *p = "hello world";
int a;

int main(void)
{
    b++;
    a = 0;
    return 0;
}
```



程序的编译链接过程



程序的编译链接过程

手动链接并生成可执行文件过程如下：

- 直接通过参数指定程序入口和段地址：

```
arm-linux-ld -Ttext=0x30000 -Tdata=0x40000
-e main -o app head.o main.o
```

- 通过链接脚本指定程序入口和段地址：

```
arm-linux-ld -Tapp.lds -o app head.o main.o
```

程序的编译链接过程

```
//连接文件app.lds为:
ENTRY(main)
SECTIONS
{//“*”号指所有目标，可以指定.o目标文件，多个用空格隔开
        . = 0x30000;          //“.”指的是当前位置
        .text:{*(.text)}
        . = 0x40000;
        .data:{*(.data)}
        .bss:{*(.bss)}
}
```

裸机程序编译过程简介

参考基础代码中的Makefile文件

```
app.bin :$(OBJS)
    $(LD) -v $(LFLAGS) -o app_elf $(OBJS) $(ARMLIBS)
    $(OBJCOPY) -O binary -S app_elf $@
    $(OBJDUMP) -D -m arm  app_elf > app.dis
    $(NM) -v -l app elf > app.map
```

凌阳教育官方微信：Sunplusedu

Tel: 400-705-9680 , BBS: www.51develop.net , QQ群：241275518