一、嵌入式系统开发的基础知识

1. 嵌入式系统的特点、分类、发展与应用，熟悉嵌入式系统的逻辑组成。

（1）特点：

专用性

隐蔽性　:嵌入式系统是被包装在内部

资源受限:要求小型化、轻型化、低功耗及低成本，因此软硬件资源受到限制

高可靠性:任何误动作都可能会产生致命的后果

软件固化:软件都固化在只读存储器 ROM 中，用户不能随意更改其程序功能

实时性

（2）逻辑组成

硬件：

1)处理器（运算器、控制器、存储器）

目前所有的处理器都是微处理器 中央处理器（CPU）和协助处理器（数字信号处理器 DSP、图像处理器、通信处理器）

2)存储器（随机存储器 RAM 和只读存储器 ROM）

RAM 分为动态 DRAM 和静态 SRAM 两种。DRAM 电路简单、集成度高、功耗小、成本低，但速度稍慢慢；SRAM 电路较复杂、集成度低、功耗较大、成本高，但工作速度很快，适合用作指令和数据的高速缓冲存储器 RAM 当关机或断电时，其中的信息都会消失，属于易失性存储器

ROM 属于不易失性存储器。分为电可擦可编程只读存储器（存放固件）和闪速存储器（Flash ROM 简称内存）。内存的工作原理：在低
压下，存储的信息可读但不可写，这类似于 ROM；在较高的电压下，
所存储的信息可以更改和删除，这有类似于 RAM。

3)I/O 设备与 I/O 接口

4)数据总线

软件

（3）分类

按嵌入式系统的软硬件技术复杂程度进行分类：

1)低端系统　采用 4 位或 8 位单片机，在工控领域和白色家电领域占主导地位，如计算器、遥控器、充电器、空调、传真机、BP 机等。

2)中端系统　采用 8 位/16 位/32 位单片机，主要用于普通手机、摄像机、录像机、电子游戏机等。

3)高端系统采用 32 位/64 位单片机，主要用于智能手机、调制解调器、掌上计算机、路由器、数码相机等。

（4）发展

20 世纪 60 年代初，第一个工人的现代嵌入式系统（阿波罗导航计算机）；

20 世纪 60 年代中期，嵌入式计算机批量生产；

20 世纪 70 年代，微处理器出现；

20 世纪 80 年代中期，外围电路的元器件被集成到处理器芯片中，昂贵的模拟电路元件能被数字电路替代；

20 世纪 90 年代中期 SOC 出现，集成电路进入超深亚微米乃至纳米加工时代

2. 嵌入式系统的组成与微电子技术（集成电路、EDA、SoC、IP 核等技术的作用和发展）

2.1 集成电路 IC

集成电路的制造大约需要几百道工序，工艺复杂。集成电路是在硅衬底上制作而成的。硅衬底是将单晶硅锭经切割、研磨和抛光后制成的像镜面一样光滑的圆形薄片，它的厚度不足 1mm，其直径可以是 6、8、12 英寸甚至更大这种硅片称为硅抛光片，用于集成电路的制造。

制造集成电路的工艺技术称为硅平面工艺，包括氧化、光刻、掺杂等多项工序。把这些工序反复交叉使用，最终在硅片上制成包含多层电路及电子元件的集成电路。

集成电路的特点：体积小、重量轻、可靠性高。其工作速度主要取决于逻辑门电路的晶体管的尺寸。尺寸越小，工作频率就越高，门电路的开关速度就越快。

2.2 EDA（电子设计自动化）

2.3 SoC 芯片（片上系统）

既包含数字电路，也可以包含模拟电路，还可以包含数模混合电路和射频电路。SoC 芯片可以是一个 CPU，单核 SoC，也可以由多个 CPU 和/或 DSP，即多核 SoC。

开发流程：

（1）总体设计　可以采用系统设计语言 System C（或称 IEEE 1666，它是 C++的扩充）或 System Vetilog 语言对 SoC 芯片的软硬件作统一的描述，按照系统需求说明书确定 SoC 的性能能参数，并据此进行系统全局的设计。

（2）逻辑设计　将总体设计的结果用 RTL（寄存器传输级描述语言）语言进行描述（源文件）后，在使用逻辑综合将源文件进行综合生成，生成最简的布尔表达式核心好的连接关系（以类型为 EDF 的 EDA 工业标准文件表示）。

（**3**）综合和仿真

（**4**）芯片制造　借助 EDA 中的布局布线工具。

2.4 IP 核

IC 设计文件：逻辑门级，包括各种基本的门电路；寄存器传输级，如寄存器、译码器、数据转换器；行为级，如 CPU、DSP、存储器、总线与接口电路等。

核库中的设计文件均属于知识产权 IP 保护的范畴，所以称为“知识产权核”或“IP 核”。

IP 核是开发 SoC 的重要保证。按 IC 设计文件的类型，IP 通常分为：软核、固核、硬核。IP 核的复用可以减少研发成本，缩短研发时间，是实现 SoC 的快速设计，尽早投放市场的有效途径。

目前主要的 CPU 内核有 ARM、MIPS、PowerPC、Coldfile、x86、8051 等。ARM 内核占所有 32 位嵌入式 RISC 处理器的 90%以上。

3. 嵌入式系统与数字媒体（文本、图像和音频/视频等数字媒体的表示与处理）

（**1**）文本

含义：在计算机中的文字信息，最常用的一种数字媒体。

字符集及其编码

1）西方字符的编码

ASCII 字符集和 ASCII 编码，基本的 ASCII 字符集共 128 个字符，每个字符使用 7 个二进位制进行编码。

2）汉字的编码

汉子国家编码标准有 GB2312 和 GB18030。每个汉字用 2 个字节表示。GB2312 只有 6763 个汉字，经常不够用。GB18030 字符集与 UCS/Unicode 字符集基本兼容，采用不等长的编码方法，单字节编码表示 ASCII 字符，与 ASCII 码兼容；双字节表示汉字，与 GB2312 保持向下兼容（即 GB2312 中有的 GB18030 字符集都有）

3）UCS/Unicode 编码

文本类型

1）简单文本

只能顺序阅读。

2）丰富格式文本

有插图、对文字颜色等定义，调整页面，文本布局，插入声音视频等。

3）超文本

通过超链接实现跳转、导航、回溯等操作

（2）图像

图像获取过程的核心是模拟信号的数字化，处理步骤为：

1）扫描　将画面网格化，每个网格为一个取样点；

2）分色　将每个取样点的颜色分解成三原色；

3）取样　测量每个取样点的每个分量（基色）亮度值；

4）量化　把模拟量使用数字量来表示，A/D 转换；

数字图像的主要参数：图像大小(水平分辨率*竖直分辨率)、位平面数目、像素深度、颜色模型；

一幅图像的数据量计算公式：

图像数据量=图像大小*像素深度/8

（3）音频/视频

音频/视频信息的数字化，处理步骤为：

1）取样

2）量化

3）编码

数字音频的主要参数：取样频率、量化位数、声道数目、使用的压缩编码方法、比特率（每秒钟的数据量）

压缩前　波形声音的码率（比特率）= 取样频率 * 量化为数 * 声道数（单位 b/s）

压缩后　码率 = 压缩前码率 / 压缩倍数（压缩比）

4．嵌入式系统与网络通信技术（数字通信与计算机网络，TCP/IP 协议，互联网接入技术等）

（**1**）数字通信

（**2**）计算机网络

（**3**）音频/ **TCP/IP** 协议

网络类别	最大网络数	第一个可用的网络号	最后一个可用的网络号	每个网络中的最大主机数	保留 IP 地址	私有 IP 地址
A	126 (2^7-2)	1	126	16777214 ($2^{24}-2$)	127. X. X. X	10. 0. 0. 0~ 10. 255. 255. 255
B	16384 (2^{14})	128. 0	191. 255	65534 ($2^{16}-2$)	169. 254. X. X	172. 16. 0. 0~ 172. 31. 255. 255
C	2097152 (2^{21})	192. 0. 0	223. 255. 255	254 (2^8-2)	无	192. 168. 0. 0~ 192. 168. 255. 255

（**4**）互联网接入技术

二、嵌入式处理器

1．嵌入式处理器的结构、特点与分类（不同类型的典型嵌入式处理器及其特点，嵌入式处理器分类等）

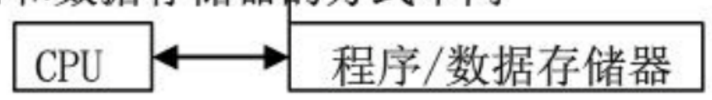
（1）不同内核嵌入式微控制器性能比较

性能 \ 内核	51 内核	其他 8 位内核	16 位内核	其他 32 位内核	ARM Cortex-M 内核
处理速度	差	差	一般	好	好
低能耗	好	好	好	差	好
代码密度	差	差	一般	差	好
内存>64KB	差	差	差	好	好
向量中断	好	好	好	一般	好
低中段延时	好	好	好	差	好
低成本	好	好	好	差	好
多供资源	好	差	差	差	好
编译器选择	好	一般	一般	一般	好
软件可移植性	好	一般	一般	一般	好

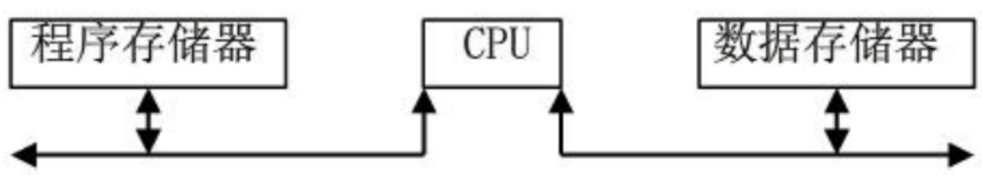
（2）冯－诺依曼结构和哈佛结构的区别

两者连接 CPU 程序存储器和数据存储器的方式不同

冯－诺依曼结构



哈佛结构



程序总线　　数据总线

（3）分类

按指令集分为：复杂指令集结构 CISC 和精简指令集结构 RISC

按存储机制分为：冯－诺依曼结构和哈佛结构

按字长分为：8 位、16 位、32 位、64 位结构

按不同内核系列可以分为：51、AVR、PIC、MSP430、PowerPC、Coldfile、ARM

（4）不同典型内核简介

内核系列	推出公司	内核结构	简单描述
51	Intel	CISC 哈佛结构	8 位字长，常用于简单的检测与控制应用领域，最早被称为单片机。其价格低，应用资料齐全，开发工具便宜，开发周期短，成本低，因此被广泛应用到各个行业。随着 1T 改进型 51 内核的推出，加上许多器件厂家增加了自己的特色组件，51 系列还在使用。
AVR	Atmel	RISC 哈佛结构	8 位、16 位和 32 位三类字长的微控制器内核，以适应不同应用层次的要求。主要特点是高性能、高速度、低功耗。
PIC	Microchip	RISC 哈佛结构	8 位、16 位和 32 位三类字长的微控制器内核，以适应不同应用层次的要求。主要用于工业控制，主要优势是针对性强，特别是抗干扰能力强。
MSP430	TI	RISC 冯-结构	16 位字长的微控制器区内核，广泛应用于手持设备嵌入式应用系统中，突出特点就是以超低功耗著称全球。
MIPS	MIPS	RISC 哈佛结构	高性能高档次 32 位和 64 位处理器内核。主要特点是适应于高速、大数据吞吐量应用场合
PowerPC	Apple， IBM,Motorola	RISC 哈佛结构	高性能高档次含有 32 位子集的 64 位处理器内核。具有优异的性能、较低的能耗以及较低的散热量。
MC68K	Motorola	RISC 哈佛结构	32 位字长的处理器内核，具有超标量的超级指令流水线，性能优异明显，主要用于与高端嵌入式应用领域。
Coldfile	Frescale	RISC 哈佛结构	32 位字长的高性能处理器内核，性能优越，集成度高，可用于工业应用领域、消费电子领域、医疗电子领域、测试与测量领域等。
ARM	ARM	RISC 多数为哈佛结构	32 位字长的高性能处理器内核，目前嵌入式处理器的领跑者

2．ARM 处理器内核的体系结构（工作状态，工作模式，寄存器组织，异常，数据类型与存储格式等）

（1）工作状态（3 种）

一是 ARM 状态，二是 Thumb 指令状态及 Thumb-2 状态，三是调试状态。

ARM 处理器复位后开始执行代码时总是只处于 ARM 状态，如果需要，可通过下面的方法

切换到 Thumb 状态或 Thumb-2 状态

ARM 状态切换到 Thumb 指令状态：通过 BX 指令，将操作数寄存器的最低位设置为 1 即可。如果 R0[0]=1，则执行 BX R0 指令将进入 Thumb 状态

状 Thumb 态切换到 ARM 状态：通过 BX 指令，将操作数寄存器的最低位设置为 0 即可。如果 R0[0]=0，则执行 BX R0 指令将进入 ARM 状态。

（**2**）工作模式（**7** 种）

工作模式	功能说明	可访问的寄存器	CPSR[M4:M0]
用户模 User	程序正常执行工作模式	PC,R14-R0,CPSR	10000
快速中断模式 FIQ	处理高速中断，用于高速数据传输或通道处理	PC,R14_fiq-R8_fiq, R7-R0,CPSR,SPSR_fiq	10001
外部中断模式 IRQ	用于普通中断处理	PC,R14_irq-R13_irq, R12-R0,CPSR,SPSR_irq	10010
管理模式 SVC	操作系统的保护模式，处理软中断 SWI	PC,R14_svc-R13_svc, R12-R0,CPSR,SPSR_svc	10011
中止模式 ABT	处理存储器故障，实现虚拟存储器和存储器保护	PC,R14_abt-R13_abt, R12-R0,CPSR,SPSR_abt	10111
未定义指令模式 UND	处理为定义的指令陷阱，用于支持硬件协处理器仿真	PC,R14_und-R13_und, R12-R0,CPSR,SPSR_und	11011
系统模 SYS	运行特权及的操作系统任务	PC, R14-R0,CPSR	11111

（**3**）寄存器组织

ARM 处理器共有 37 种寄存器，包括 31 个通用寄存器（含 PC）和 6 个状态寄存器。无论何种模式，R15 均作为 PC 使用；CPSR 为当前程序状态寄存器；R7-R0 为公用的通用寄存器。所有通用寄存器均为 32 位结构。

寄存器类别	寄存器在汇编中的名称	各模式实际访问的寄存器					
		用户 系统	管理	中止	未定义	中断	快中断
通用寄存器 程序计数器	R0（a1）	R0					
	R1（a2）	R1					
	R2（a3）	R2					
	R3（a4）	R3					
	R4（v1）	R4					
	R5（v2）	R5					
	R6（v3）	R6					
	R7（v4）	R7					
	R8（v5）	R8					R8_fiq
	R9（SB,v6）	R9					R9_fiq
	R10(SL,v7)	R10					R10_fiq
	R11(FP,v8)	R11					R11_fiq
	R12（IP）	R12					R12_fiq
	R13（SP）	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14（LR）	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15（PC）	R15					
状态寄存器	CPSR	CPSR					
	SPSR	无	SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

R13:SP(stack point)堆栈指针；　　R14:LR(Link Register)程序链接寄存器

R15:PC(Program counter)程序计数器；　　SPSR：备份程序状态寄存器

CPSR 程序状态寄存器的格式：

31	30	29	28	27	26……8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	状态保留	I	F	T	M4	M3	M2	M1	M0

条件码标志含义如下：

N 为符号标志位，N=1 为负数，N=0 为正数。

Z 为全 0 标志位，运算结果为 0，则 Z=1，否则 Z=0；

C 为进借位标志，有进/借位时 C=1，否则 C=0.

V 为溢出标志，加减法运算结果溢出时 V=1，否则 V=0.

Q 为增强的 DSP 运算指令溢出标志，溢出时 Q=1，否则 Q=0.

控制位含义如下：

I 为中断禁止控制位，I=1 禁止 IRQ 中断，I=0，允许中断。

F 为禁止快速中断 FIQ 的控制位，F=1 禁止 FIQ 中断，F=0 允许。

T 为 ARM 和 Thumb 指令切换，T=1 时执行 Thumb 指令，否则执行 ARM 指令。

M4-M0 为模式选择位，决定工作模式

（**4**）存储格式

大端模式：32 位数据字的高字节存储在低地址，而数据字的低字节则存放在高地址中。

小端模式：32 位数据字的高字节存储在高地址，而数据字的低字节则存放在低地址中。系统复位时，自动默认为小端模式。

例如：一个 32 位数据字 0x12345678，存放在起始地址为 0x30001000，则大端模式下 0x30001000 单元存放 0x12,0x30001001 单元存放 0x34，0x30001002 单元存放 0x56，0x30001003 单元存放 0x78；而小端模式下 0x30001000 单元存放 0x78,0x30001001 单元存放 0x56，0x30001002 单元存放 0x34，0x30001003 单元存放 0x12。

（5）数据类型

8 位、16 位、32 位三种数据类型

（6）ARM 处理器中 **MMU** 和 **MPU**

MMU 存储器管理单元（memory management unit）功能：

1)虚拟地址到物理地址映射

2）存储器访问权限受限

3）虚拟存储空间的缓冲特性设置

MPU 存储器保护单元（memory protect unit）

（7）异常（**7** 种）

异常类型	优先级	工作模式	异常向量地址
复位 RESET	1	管理模式	0x00000000
未定义的指令 UND	6	未定义指令中止模式	0x00000004
软件中断 SWI	6	管理模式	0x00000008
指令预取中止 PABT	5	中止模式	0x0000000C
数据访问中止 DABT	2	中止模式	0x00000008
外部中断请求 IRQ	4	外部中断模式	0x00000010
快速中断请求 FIQ	3	快速中断模式	0x0000001C

3．典型 ARM 处理器内核（ARM9，Cortex-A，Cortex-M，Cortex-R 等的技术特点与应用领域）

（1）Cortex-A 系列是面向高端嵌入式应用的处理器核：具有 MMU、Cache、最快频率、最高性能、合理功耗。

（2）Cortex-R 系列是面向实时控制的处理器：具有 MPU、Cache、实时响应、合理性能、较低功耗。

（3）Cortex-M 系列是面向低端微控制器的处理器，没有 MMU 但有 MPU，极高性价比、最低成本，极低功耗。

系列	相应内核	主要性能特点
ARM7		冯-诺依曼结构，3 级流水线，无 MMU
经典 ARM9	ARM920T/ARM922T	哈佛结构，5 及流水线，单 32 位 AMBA 接口
ARM9E	ARM926EJ-S/ARM946E-S/ARM966E-S/	哈佛结构，5 及流水线，支持 DSP 指令，软核（soft IP）

	ARM968E-S/ARM996HS	
ARM10	ARM1020E/ARM1022E/ARM1026EJ-S	哈佛结构，6 及流水线，分支预测，支持 DSP 指令，高性能浮点操作，双 64 位总线接口，内部 64 位数据通路。
ARM11	ARM11MPCore/ARM1136J(F)-S	哈佛结构，8 级流水线，分支预测和返回栈，支持 DSP 指令、SIMD/Thumb-2 核心技术
	ARM1156T2(F)-S/ARM1176JZ(F)-S	哈佛结构，9 级流水线，分支预测和返回栈，支持 DSP 指令、SIMD/Thumb-2 核心技术
嵌入 Cortex-M	Cortex-M0, Cortex-M0+	冯-诺依曼结构，3 级流水线，支持 Thumb 指令集并包含 Thumb-2、嵌套向量中断，M0+内部有 MPU，而 M0 没有。
	Cortex-M1	冯-诺依曼结构，3 级流水线，支持 FPGA 设计，Thumb 指令集并包含 Thumb-2
	Cortex-M3	哈佛结构， 3 级流水线，Thumb-2、嵌套向量中断，分支指令预测，内置 MPU
	Cortex-M4	哈佛结构， 3 级流水线，Thumb-2、嵌套向量中断，分支指令预测，内置 MPU，高效信号处理，SIMD 指令，饱和运算，FPU
Cortex-R	Cortex-R4/R4F/ Cortex-R5/Cortex-R7	哈佛结构，8 级流水线，实时应用，支持 ARM、Thumb 和 Thumb-2 指令集，F 标示内置 FPU，DSP 扩展，分支预测，超标量执行，内置 MPU
应用 Cortex-A	Cortex-A5/ Cortex-A5MPcore	
	Cortex-A7/ Cortex-A7MPcore	
	Cortex-A8/ Cortex-A8MPcore	
	Cortex-A9/ Cortex-A9MPcore	哈佛结构，MPcore 为多核，超标量结构，13 级流水线，动态分支指令预测，有分支目标缓冲器 BTB、MMU、FPU、L1、L2，支持 ARM、 Thumb 和 Thumb/EE 指令集,SIMD/Jazelle RCT 技术。
	Cortex-A15/ Cortex-A15MPcore	哈佛结构，可乱序执行指令流水线

4. ARM 处理器指令系统及汇编语言程序设计（指令格式，寻址方式，指令集，伪指令，语句格式与程序结构，ARM 汇编语言与 C 的混合编程等）

4.1 指令格式

指令一般格式

<opcode>{<cond>}{S} <Rd>,<Rn>{,<op2>} 其中<>不可省

指令格式说明：

项目	*含义*	备注
〈opcode〉	指令的操作码	即助记符，如 MOV、ADD、B 等

{cond}	条件域，满足条件才执行指令	可不加条件即可省略条件，如 EQ、NE 等
{S}	指令执行时是否需要更新 CPSR	可省略
Rd	目的寄存器	Rd 可为任意通用寄存器
Rn	第一个源操作数	Rd 可为任意通用寄存器，可以与 Rd 相同
Op2	第二个源操作数	可为#imm8m、寄存器 Rm 及任意移位寄存器

关于#imm8m 的说明：#表示立即数，其后可以是十进制或十六进制数

对于 ARM 指令集，#imm8m 表示一个由 8 位立即数经循环右移任意偶数位次形成的 32 位操作数。

对于 Thumb 指令集，#imm8m 表示一个由 8 位立即数经左移任意位次形成的 32 位操作数。

指令的条件码

条件码	助记符	标　　志	含　　义
0000	EQ	Z 置位	相等
0001	NE(Not Equal)	Z 清零	不相等
0010	CS/HS(Carry Set/High or Same)	C 置位	无符号数大于或等于
0011	CC/LO(Carry Clear/LOwer)	C 清零	无符号数小于
0100	MI(MInus)	N 置位	负数
0101	PL(PLus)	N 清零	正数或零
0110	VS(oVerflow set)	V 置位	溢出
0111	VC(oVerflow clear)	V 清零	未溢出
1000	HI(HIgh)	C 置位 Z 清零	无符号数大于
1001	LS(Lower or Same)	C 清零 Z 置位	带符号数小于或等于
1010	GE(Greater or Equal)	N 等于 V	带符号数大于或等于
1011	LT(Less Than)	N 不等于 V	带符号数小于
1100	GT(Greater Than)	Z 清零且 N 等于 V	带符号数大于
1101	LE(Less or Equal)	Z 置位或 N 不等于 V	带符号数小于或等于
1110	AL	忽略	无条件执行

4.2 寻址方式

1）立即寻址（立即数寻址）

例如：MOV R0，#0x1212121212

ADC R0，R0，#100　　;R0　　R0+100+C

2）寄存器寻址(执行效率较高)

例如：ADD R0，R1，R2　　;　R0←R1+R2

3）寄存器间接寻址

寄存器间接转址就是以寄存器中的值作为操作数地址，而操作数本身存放在存储器中。用间接寻址的寄存器必须用[ ]括起来。

例如:LDR R5，[R4]　　;R5←[R4],间接寻址的寄存器是 R4

STR R1，[R2]　　;[R2]←R1,间接寻址的寄存器是 R2

4）基址加变址寻址

常见的几种形式：

```
LDR R0，[R1，#4]
STR R1，[R2，#8]
LDR R0，[R1，#4]!     (！表示指令在完成数据传输后更新基址存储器)
LDR R0，[R1]，#4
LDR R0, [R1, R2]
STR R0, [R1, R2]
```

5）相对寻址

相对寻址以程序计数器 PC 的当前值为基地址，指令中的地址标号作为偏移量，将两者相加后得到操作数的有效地址。

下列程序中跳转指令 BL 利用相对寻址方式：

```
        BL Subroutine_A          ;跳转子程序 Subroutine_A 处执行
        …
Subroutine_A:
        …
        MOV PC ,LR               ;从子程序返回
```

6）堆栈寻址

7）块拷贝寻址（多寄存器寻址）

块拷贝寻址是多寄存器传送指令 LDM/STM 的寻址方法

LDMIA R0，{R1，R2，R5，R9};	STMIA R0，{R1-R7}
R1←[R0];	[R0]←R1;
R2←[R0+4]];	[R0+4]←R2;
R5←[R0+8];	[R0+8]←R3;
R9←[R0+12];	[R0+12]←R4;
	[R0+16]←R5;
	[R0+20]←R6;
	[R0+24]←R7;

4.3 ARM 处理器支持的伪指令

伪指令为指示性语句，指示性语句是不可执行的语句，为数据分配内存空间或相应

寄存器，不产生任何机器代码。

ADR：用于相对偏移地址加载到通用寄存器中

LDR：用于 32 位常数的加载或者地址的加载

1）当用 MOV 或 NMV 无法加载符合要求的 32 位立即数时，可用 LDR 伪指令加载任意 32 位操作数到寄存器

2）当需要程序的相对偏移地址或外部地址加载到寄存器时可使用 LDR

LDR R1,=0x1234

NOP:产生无操作代码，用于延时

4.4 ARM 汇编器支持的伪指令：

DCB：分配一片连续的字节存储单元，并用伪指令中指定的表达式初始化（DCB 可用=替代）

DCW：分配一片连续的半字节存储单元，并用伪指令中指定的表达式初始化

DCD：分配一片连续的字存储单元，并用伪指令中指定的表达式初始化

AREA：用于定义一个代码段或者数据段

CODE16：伪指令通知编译器其后的指令序列为 16 位的 Thumb 指令

CODE32：伪指令通知编译器其后的指令序列为 32 位的 ARM 指令

ENTRY：指定汇编程序的入口点，完整的汇编文件中至少有一个或多个，但一个源文最多有一个

EQU：用于为程序中的常量、标号等定义一个等效的字符名称（EQU 可以用*替代）

IMPORT：通知编译器要使用的标号在其他的源文件中定义，但要在当前文件中引用，无论当前源文件是否引用，标号均会被加入当前源文件的符号表中

EXTERN：通知编译器要使用的标号在其他的源文件中定义，但要在当前文件中引用；如果当前源文件未引用，标号不会被加入当前源文件的符号表中。

4.5 ARM 指令集

表A-2 ARM 存储访问指令表

助记符	说明	操作	条件码位置
LDR Rd, addressing	加载字数据	Rd←[addressing], 寻址索引	LDR{cond}
LDRB Rd, addressing	加载无符字节数据	Rd←[addressing], 寻址索引	LDR{cond}B
LDRT Rd，addressing	以用户模式加载字数据	Rd←[addressing], 寻址索引	LDR{cond}T
LDRBT Rd，addressing	以用户模式加载无符号字数据	Rd←[addressing], 寻址索引	LDR{cond}BT
LDRH Rd，addressing	加载无符半字数据	Rd←[addressing], 寻址索引	LDR{cond}H
LDRSB Rd，addressing	加载有符字节数据	Rd←[addressing], 寻址索引	LDR{cond}SB
LDRSH Rd，addressing	加载有符半字数据	Rd←[addressing], 寻址索引	LDR{cond}SH
STR Rd，addressing	存储字数据	[addressing]←Rd, 寻址索引	STR{cond}
STRB Rd，addressing	存储字节数据	[addressing]←Rd, 寻址索引	STR{cond}B
STRT Rd，addressing	以用户模式存储字数据	[addressing]←Rd, 寻址索引	STR{cond}T
SRTBT Rd，addressing	以用户模式存储字节数据	[addressing]←Rd, 寻址索引	STR{cond}BT
STRH Rd，addressing	存储半字数据	[addressing]←Rd, 寻址索引	STR{cond}H
LDM{mode} Rn{!},reglist	批量(寄存器)加载	reglist←[Rn…], Rn回存等	LDM{cond}{more}
STM{mode} Rn{!},rtglist	批量(寄存器)存储	[Rn…]←reglist, Rn 回存等	STM{cond}{more}
SWP Rd,Rm,Rn	寄存器和存储器字数据交换	Rd←[Rd], [Rn]←[Rm] (Rn≠Rd或Rm)	SWP{cond}
SWPB Rd,Rm,Rn	寄存器和存储器字节数据交换	Rd←[Rd], [Rn]←[Rm] (Rn≠Rd或Rm)	SWP{cond}B

表A-4 ARM 数据处理指令

助记符号	说明	操作	条件码位置
MOV Rd, operand2	数据转送	Rd←operand2	MOV {cond}{S}
MVN Rd, operand2	数据非转送	Rd←(operand2)	MVN {cond}{S}
ADD Rd, Rn operand2	加法运算指令	Rd←Rn+operand2	ADD {cond}{S}
SUB Rd, Rn operand2	减法运算指令	Rd←Rn-operand2	SUB {cond}{S}
RSB Rd, Rn operand2	逆向减法指令	Rd←operand2-Rn	RSB {cond}{S}
ADC Rd, Rn operand2	带进位加法	Rd←Rn+operand2+carry	ADC {cond}{S}
SBC Rd, Rn operand2	带进位减法指令	Rd←Rn-operand2-(NOT)Carry	SBC {cond}{S}
RSC Rd, Rn operand2	带进位逆向减法指令	Rd←operand2-Rn-(NOT)Carry	RSC {cond}{S}
AND Rd, Rn operand2	逻辑与操作指令	Rd←Rn&operand2	AND {cond}{S}
ORR Rd, Rn operand2	逻辑或操作指令	Rd←Rn|operand2	ORR {cond}{S}
EOR Rd, Rn operand2	逻辑异或操作指令	Rd←Rn^ operand2	EOR {cond}{S}
BIC Rd, Rn operand2	位清除指令	Rd←Rn&(~operand2)	BIC {cond}{S}
CMP Rn, operand2	比较指令	标志N、Z、C、V←Rn-operand2	CMP {cond}
CMN Rn, operand2	负数比较指令	标志N、Z、C、V←Rn+ operand2	CMN {cond}
TST Rn, operand2	位测试指令	标志N、Z、C、V←Rn& operand2	TST {cond}
Rn, operand2	相等测试指令	标志 N、Z、C、V←Rn^ operand2	TEQ {cond}

表A-6 ARM跳转指令

助记符	说明	操作	条件码位置
B label	跳转指令	Pc←label	B{cond}
BL label	带链接的跳转指令	LR←PC-4， PC←label	BL{cond}
BX Rm	带状态切换的跳转指令	PC←label，切换处理状态	BX{cond}

4.6 Thumb 指令集

Thumb 指令集没有协处理器指令，信号量指令以及访问 CPSR 或 SPSR 的指令，没有乘加指令及 64 位乘法指令等，且指令的第二操作数受到限制;除了跳转指令 B 有条件执行功能外，其它指令均为无条件执行;大多数 Thumb 数据处理指令采用 2 地址格式。

在编写 Thumb 指令时，先要使用伪指令 CODE16 声明，而且在 ARM 指令中要使用 BX 指令跳转到 Thumb 指令，以切换处理器状态。编写 ARM 指令时，则可使用伪指令 CODE32 声明。

表A-9 Thumb 存储器访问指令

助记符	说明	操作	影响标志
LDR Rd,[Rn,#immed_5×4]	加载字数据	Rd←[Rm,#immed_5×4],Rd,Rn 为R0～R7	无
LDRH Rd,[Rn,#immed_5×2]	加载无符半字数据	Rd←[Rm,#immed_5×2],Rd,Rn 为R0～R7	无
LDRB Rd,[Rn,#immed_5×1]	加载无符字节数据	Rd←[Rm,#immed_5×1],Rd,Rn 为R0～R7	无
STR Rd,[Rn,#immed_5×4]	存储字数据	Rn,#immed_5×4Rd←Rd,Rn 为R0～R7	无
STRH Rd,[Rn,#immed_5×2]	存储无符半辽数据	Rn,#immed_5×2]Rd←Rd,Rn 为R0～R7	无
STRB Rd,[Rn#immed_5×1]	存储无符字节数据	Rn,#immed_5×1]Rd←Rd,Rn 为R0～R7	无
LDR Rd,[Rn,Rm]	加载字数据	Rd←[Rn,Rm],Rd,Rn,Rm 为R0～R7	无
LDRH Rd,[Rn,Rm]	加载无符半字数据	Rd←[Rn,Rm],Rd,Rn,Rm 为R0～R7	无
LDRB Rd,[Rn,Rm]	加载无符字节数据	Rd←[Rn,Rm],Rd,Rn,Rm 为R0～R7	无
LDRSH Rd[Rn,Rm]	加载有符半字数据	Rd←[Rn,Rm],Rd,Rn,Rm 为R0～R7	无
LDRSB Rd[Rn,Rm]	加载有符字节数据	Rd←[Rn,Rm],Rd,Rn,Rm 为R0～R7	无
STR Rd,[Rn,Rm]	存储字数据	[Rn,Rm]←Rd,Rd,Rn,Rm 为R0～R7	无
STRH Rd,[Rn,Rm]	存储无符半字数据	[Rn,Rm]←Rd,Rd,Rn,Rm 为R0～R7	无
STRB Rd,[Rn,Rm]	存储无符字节数据	[Rn,Rm]←Rd,Rd,Rn,Rm 为R0～R7	无
LDR Rd,[PC,#immed_8×4]	基于PC加载字数据	Rd←{PC,#immed_8×4]Rd 为R0～R7	无
LDR Rd,label 基于PC	加载字数据	Rd←[label],Rd 为R0～R7	无
LDR Rd,[SP,#immed_8×4]	基于SP加载字数据	Rd←{SP,#immed_8×4]Rd 为R0～R7	无
STR Rd,[SP,#immed_8×4]	基于SP存储字数据	{SP,#immed_8×4]←Rd,Rd 为R0～R7	无
LDMIA Rn{!}reglist	批量(寄存器)加载	regist←[Rn…],Rn 回存等(R0～R7)	无
STMIA Rn{!}reglist	批量(寄存器)加载	[Rn…]←reglist,Rn 回存等(R0～R7)	无
PUSH {reglist[,LR]}	寄存器入栈指令	[SP…]←reglist[,LR],SP回存等(R0～R7,LR)	无
POP {reglist[,PC]}	寄存器入栈指令	reglist[,PC]←[SP…],SP回存等(R0～R7,PC)	无

表A-11 Thumb跳转指令

助记符	说明	操作	条件码位置
B label	跳转指令	PC←label	B{cond}
BL label	带链接的跳转指令	LR←PC←4,PC←label	无
BX Rm	带状态切换的跳转指令	PC←label	无

表格 1 Thumb 数据处理指令

助记符	说明	操作	影响标志
MOV Rd, #expr	数据转送	Rd←expr，Rd 为 R0～R7	影响 N、Z
MOV Rd, Rm	数据转送	Rd←Rm，Rd、Rm 均可为 R0～R15，	Rd 和 Rm 均为 R0～R7 时影响 N、Z，清零 C、V
MVN Rd, Rm	数据取反传送指令	Rd←(～Rm)，Rd、Rm 均为 R0～R7	影响 N、Z
NEG Rd, Rm	数据取负指令	Rd←(-Rm)，Rd、Rm 均为 R0～R7	影响 N、Z、C、V
ADD Rd。Rn, Rm	加法运算指令	Rd←Rn+Rm，Rd、Rn、Rm 均为 R0～R7	影响 N、Z、C、V
ADD Rd。Rn, #expr3	加法运算指令	Rd←Rn+expr#，Rd、Rn 均为 R0～R7	影响 N、Z、C、V
ADD Rd, #expr8	加法运算指令	Rd←Rd+expr8，Rd 为 R0～R7	影响 N、Z、C、V
ADD Rd, Rm	加法运算指令	Rd←Rd+Rm，Rd、Rm均可为R0～R15 Rd 和 Rm 均为 R0～R7	影响 N、Z、C、V
ADD Rd, Rp#expr SP/PC	加法运算指令	Rd←SP+expr 或 PC+expr，Rd 为 R0～R7	无
ADD SP, #expr SP	加法运算指令	SP←SP+expr	无
SUB Rd, Rn, Rm	减法运算指令	Rd←Rn-Rm, Rd、Rn、Rm 均为 R0～R7	影响 N、Z、C、V
SUB Rd, Rn#expr3	减法运算指令	Rd←Rn-expr3，Rd、Rn 均为 R0～R7	影响 N、Z、C、V
SUB Rd, #expr8	减法运算指令	RD←Rd-expr8，Rd 为 R0～R7	影响 N、Z、C、V
SUB SP, #expr SP	减法运算指令	SP←SP-expr	无
ADC Rd, Rm	带进位加法指令	Rd←Rd+Rm+Carry，Rd、Rm 为 R0～R7	影响 N、Z、C、V
SBC Rd, Rm	带位减法指令	Rd←Rd-Rm-(NOT)Carry，Rd、Rm 为 R0～R7	影响 N、Z、C、V
MUL Rd, Rm	乘法运算指令	Rd←Rd*Rm，Rd、Rm 为 R0～R7	影响 N、Z
AND Rd, Rm	逻辑与操作指令	Rd←Rd&Rm，Rd、Rm 为 R0～R7	影响 N、Z
ORR Rd, Rm	逻辑或操作指令	Rd←Rd|Rm，Rd、Rm 为 R0～R7	影响 N、Z
EOR Rd, Rm	逻辑异或操作指令	Rd←Rd^Rm，Rd、Rm 为 R0～R7	影响 N、Z
BIC Rd, Rm	位清除指令	Rd←Rd&(～Rm)，Rd、Rm 为 R0～R7	影响 N、Z
ASR Rd, Rs	算术右移指令	Rd←Rd 算术右移 Rs 位，Rd、Rs 为 R0～R7	影响 N、Z、C
ASR Rd, Rm#expr	算术右移指令	Rd←Rm 算术右移 expr 位，Rd、Rm 为 R0～R7	影响 N、Z、C
LSL Rd, Rs	逻辑左移指令	Rd←Rd<<Rs，Rd、Rs 为 R0～R7	影响 N、Z、C
LSL Rd, Rm, #expr	逻辑左移指令	Rd←Rm<<expr，Rd、Rm 为 R0～R7	影响 N、Z、C
LSR Rd, Rs	逻辑右移指令	Rd←Rd>>Rs，Rd、Rs 为 R0～R7	影响 N、Z、C
LSR Rd, Rm, #expr	逻辑右移指令	Rd←Rm>>mexpr，Rd、Rm 为 R0～R7	影响 N、Z、C
ROR Rd, Rs	循环右移指令	Rd←Rm 循环右移 Rs 位，Rd、Rs 为 R0～R7	影响 N、Z、C
CMP Rn, Rm	比较指令	状态标←Rn-Rm，Rn、Rm 为 R0～R15	影响 N、Z、C、V
CMP Rn, #expr	比较指令	状态标←Rn-expr，Rn 为 R0～R7	影响 N、Z、C、V
CMN Rn, Rm	负数比较指令	状态标←Rn+Rm，Rn、Rm 为 R0～R7	影响 N、Z、C、V
TST Rn, Rm	位测试指令	状态标←Rn&Rm，Rn、Rm 为 R0～R7	影响 N、Z、C、V

4.7 Thumb-2 指令集

三、嵌入式系统硬件组成

1．嵌入式硬件组成与嵌入式处理芯片（组成，特点，类型，**ARM** 的 **AMBA** 总线，嵌入式处理芯片的选型）

（**1**）基于 **ARM** 内核的典型嵌入式应用系统硬件组成

典型嵌入式系统硬件由嵌入式最小硬件系统（电源电路、时钟（晶振）电路、复位电路、JTAG 测试接口）、前向通道（输入接口）、后向通道（输出接口）、人机交互通道（键盘，触摸屏以及 LED 或 LCD 显示输出接口）以及相互互联通信通道（CAN 通信接口、以太网通信接口、USB 通信接口）等组成。

电源电路为整个嵌入式系统提供能量，是整个系统工作的基础，具有极其重要的位置。一般来说 ，如果电源电路处理得好，整个系统的故障往往能显著减少。选择设计电源电路是主要考虑以下因素：输出电压电流、输入电压电流（交流还是直流）、安全因素、体积限制、功耗限制、成本限制。

常用的电源模块是交流变直流（AC-DC）模块、直流变直流模块（DC-DC）、低压稳压器（LDO）。稳压器包括普通稳压器和低压差稳压器 LDO。78XX 系列属于普通稳压器，LM2576/2596 为开关稳压芯片，CAT6219/AS2815/1117/2908 等属于低压稳压器。稳压器的最大特点就是低噪声、低成本、纹波小、精度高、电路简单。

（**2**）基于 **ARM** 内核的典型嵌入式芯片的硬件组成

1）存储器及控制器

片内程序存储器通常是用 Flash ROM,一般配有几 KB 到几 MB 不等。片内数据存储器通常使用 SRAM，一般几 KB 到几百 KB。

2）中断控制器

一般采用向量中断（VIC）或嵌套向量中断（NVIC）。Cortex-M 支持嵌套的向量中断。

3）DMA 控制器（直接存储器访问控制器）

使用 DMA 控制器，可将数据块从外设传输至内存、从内存传输至外设或从内存传输至内存。

4）电源管理与时钟控制器

电源管理具有正常模式（NORMAL MODE），慢速模式（SLOW MODE），空闲/休眠模式（IDLE MODE）和掉电模式（POWER_OFF MODE）

正常模式：电源管理为内核及 ARM 芯片内部所有硬件组件提供时钟源，所有设备处于开启状态。

慢速模式：不使用 PLL 时钟(MPLL 关闭不使用)，仅使用外部晶振或外部时钟直接提供给其他组件使用，不通过锁相环电路（低速不需要）。

休眠模式：电源管理模块仅断开 ARM 内核时钟 FCLK，让 CPU 处于休眠状态，仍为外围硬件提供时钟。

掉电模式：断开内部电源，除非唤醒逻辑有效，否则内核不产生功耗。

任何模式下复位操作均会自动进入正常工作模式。在掉电模式和休眠模式下，只要任何一个外部中断 EINT[0:23]或者 RTC 时钟中断发生，均将返回到正常模式。

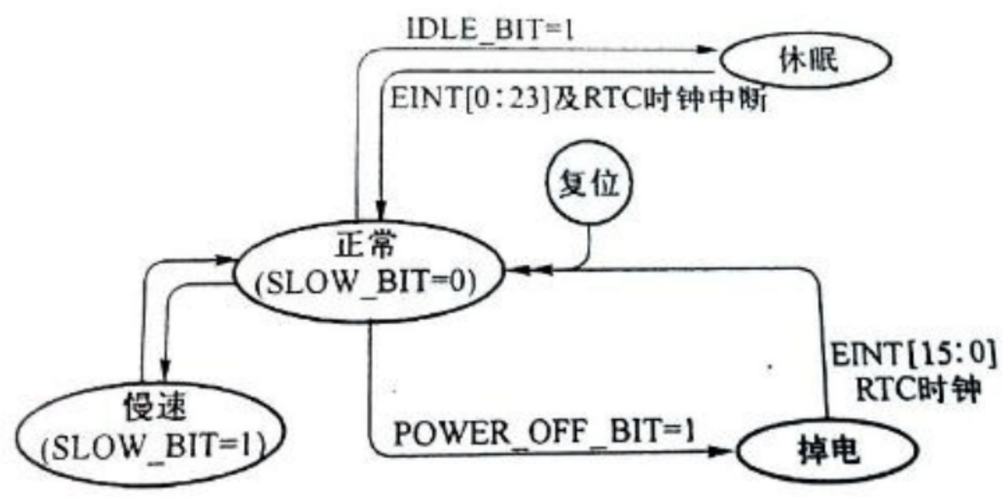


图 3-38 工作模式及其转换（电源）

5）GPIO 接端口（General Purpose Input Output 通用输入/输出端口）

作为输入时具有缓冲功能，而作为输出是具有锁存功能，GPIO 也可以作为双向 I/O 使用。在 ARM 处理芯片中，GPIO 引脚通常是多功能的，以减少引脚数，减少功耗。

6）定时计数组件

主要包括看门狗定时器（WDT） 监视着程序的运行状态

开门狗计时器的计数脉冲周期为：

t_watchdog = 1/(PCLK(预分频值+1)/分频系数)

表格 2 看门狗寄存器 WTCON 功能

位	位名称	描 述
[15:8]	Prescaler Value	预分频值：0~2^8-1
[7:6]	Reversed	保留
[5]	Watchdog Timer Enable/Disable	看门狗定时器的允许位 0=禁止看门狗定时器 1=允许看门狗定时器
[4:3]	Clock Select	确定时钟除数因子 00=1/16，01=1/32 10=1/64，11=1/128
[2]	Interrupt Enable/Disable	看门狗中断允许位 0=禁止中断产生 1=允许中断产生
[1]	Reversed	保留，在正常状态下为 0
[0]	Reset Enable/Disable	看门狗输出复位信号的允许位 0=不允许，1=允许

Timer 通用定时器 用于一般的定时

RTC 可直接提供年月日时分秒，使应用系统具有独立的日期和时间

脉冲宽度调制解调器（PWM） 用于脉冲宽度的调制，比如电机控制、用于变频调整等。

7）模拟通道组件

8）互联通信组件

（**3**）**ARM** 的 **AMBA** 总线

（**4**）常用 **ARM** 嵌入式处理芯片

1）NXP 的典型 ARM 芯片

2）TI 的典型 ARM 芯片

3）Samsung 的典型 ARM 芯片

4）Atmel 的典型 ARM 芯片

5）ST 的典型 ARM 芯片

6）Freescale 的典型 ARM 芯片

7）Nuvoton 的典型 ARM 芯片

8）Intel 的典型 ARM 芯片

9）其他 ARM 芯片厂家

（**5**）嵌入式处理芯片的选型

1）性价比原则 性能高，价格低

2）参数选择原则

ARM 内核（指令流水线、支持 Thumb/Thumb-2 指令集、最高时钟频率的限制、最低功耗要求以及低成本要求）

系统时钟频率(频率越高,处理速度越快;通常ARM芯片的速度主要取决于ARM内核）

芯片内部存储器的容量

片内外围电路（GPIO 外部引脚条数、定时计数器、LCD 液晶显示控制器、多核处理器、ADC、通信接口）

2．嵌入式系统的存储器（层次结构，分类，性能指标；片内存储器，片外存储器，外部存储设备等）

3．**I/O** 接口、**I/O** 设备以及外部通信接口（**GPIO**、**I²C**、**SPI**、**UART**、**USB**、**HDMI** 等；键盘、**LED**、**LCD**、触摸屏、传感器等；**RS-232/RS-485**、**CAN**、以太网和常用无线通信接口）

3.1 GPIO(通用输入输出接口**)**

在嵌入式处理器内部，输入具备缓冲功能，输出具有锁存功能。GPIO 一般有三态：0 态、1 态、高阻状态。

3.2 集成电路互连总线接口 **IIC**

集成电路互连总线用于连接嵌入式处理器及外围器件，采用串行半双工传输的总线标准。

IIC 总线具有的接口线少，控制方式简单，器件封装紧凑，通信速率较高（100kb/s,400kb/s,高速模式可达 3.4Mb/s）等优点。

IIC 总线的操作时序

IIC 总线只有两条信号线，一条是数据线 SDA，另一条是时钟线 SCL，所有的操作均通

过这两条信号线完成。数据线 SDA 上的数据必须在时钟的高电平周期保持稳定，它的高/低电平状态只有在 SCL 时钟信号线是低电平时才能改变。

1）启动和停止条件

总线上的所有器件都不使用总线时，SCL 线和 SDA 线各自的上拉电阻把电平拉高，使它们均处于高电平。主控制器启动总线操作的条件是当 SCL 线保持高电平时 SDA 线有高电平转为低电平，此时主控制器在 SCL 产生时钟信号，SDA 线开始传输数据。若 SCL 线为高电平时 SDA 由低转为高，则总线工作停止，恢复空闲状态

2）数据传送格式

数据传送高位在前，低位在后，每次传送的字节数没有限制。

3）应答（**ACK**）信号传送

4）读/写操作

5）总线仲裁

IIC 属于多主总线，允许总线上有一个或者多个主控器件和若干从器件同时进行操作。总线仲裁遵循“低电平优先”的原则，即谁先发送低电平谁就会掌握对总线的控制权；主控器通过检测 SDA 上自身发送的电平来判断是否发生总线仲裁。（线与结构）

6）异常中断条件

没有从器件对主器件发送的地址进行确认，SDA 时钟保持高电平。

3.3 串行外设接口 SPI

8 个时钟周期可以完成一个字节的发送。

数据传输高位在前，低位在后。

多主多从系统主要考虑 SSEL 电平的接法（大部分场合一主多从结构）。

3.4 串行异步通信接口 UART（RS485，RS422，RS232）

全双工串行通信，有 4 个 UART。由发送器、接收器、控制单元和波特率发生器等构成。

发送器负责字符发送，可以采用 FIFO 模式和普通模式；

接收器负责字符接收，可以使 FIFO 模式或者普通模式。

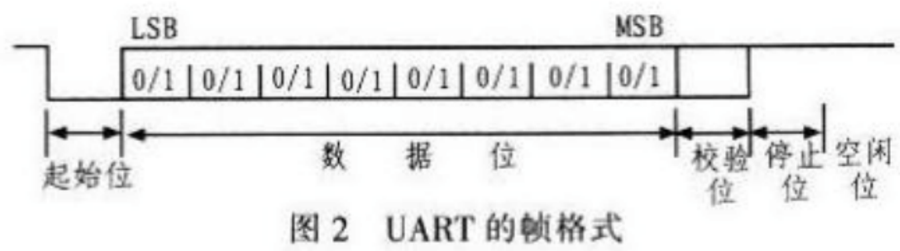


图 2 UART 的帧格式

字符总以起始位开始，以结束位为结束，数据总以低位在前，高位在后次序传输。接收采用中断模式，发送采用查询方式，因为发送由程序直接控制，接收时对方的信息是随机的。

UART 的波特率由除数寄存器决定，也取决于外部时钟，关系式如下：

UBRDIVn = INT(UCLK/(波特率 x 16)) – 1

UCLK 可以使 PCLK，还可以是 EXTCLK，由相应寄存器决定。

3.5 通用串行总线 USB

串行半双工差分通信方式，支持设备的即插即拔和热插拔功能，采用主从方式。目前许多处理器集成了 USB 控制器，包括 USB 主机、USB 设备以及 USB OTG.

USB2.0 速度可达 480Mb/s,USB3.0 可达 5Gb/s,即 640MB/s.

USB 通过 HUB 最多接 127 个设备.

USB 采用反转不归零制（NRZI）编码方式进行数据编码，电平保持时传送逻辑 1，翻转时传送 0.

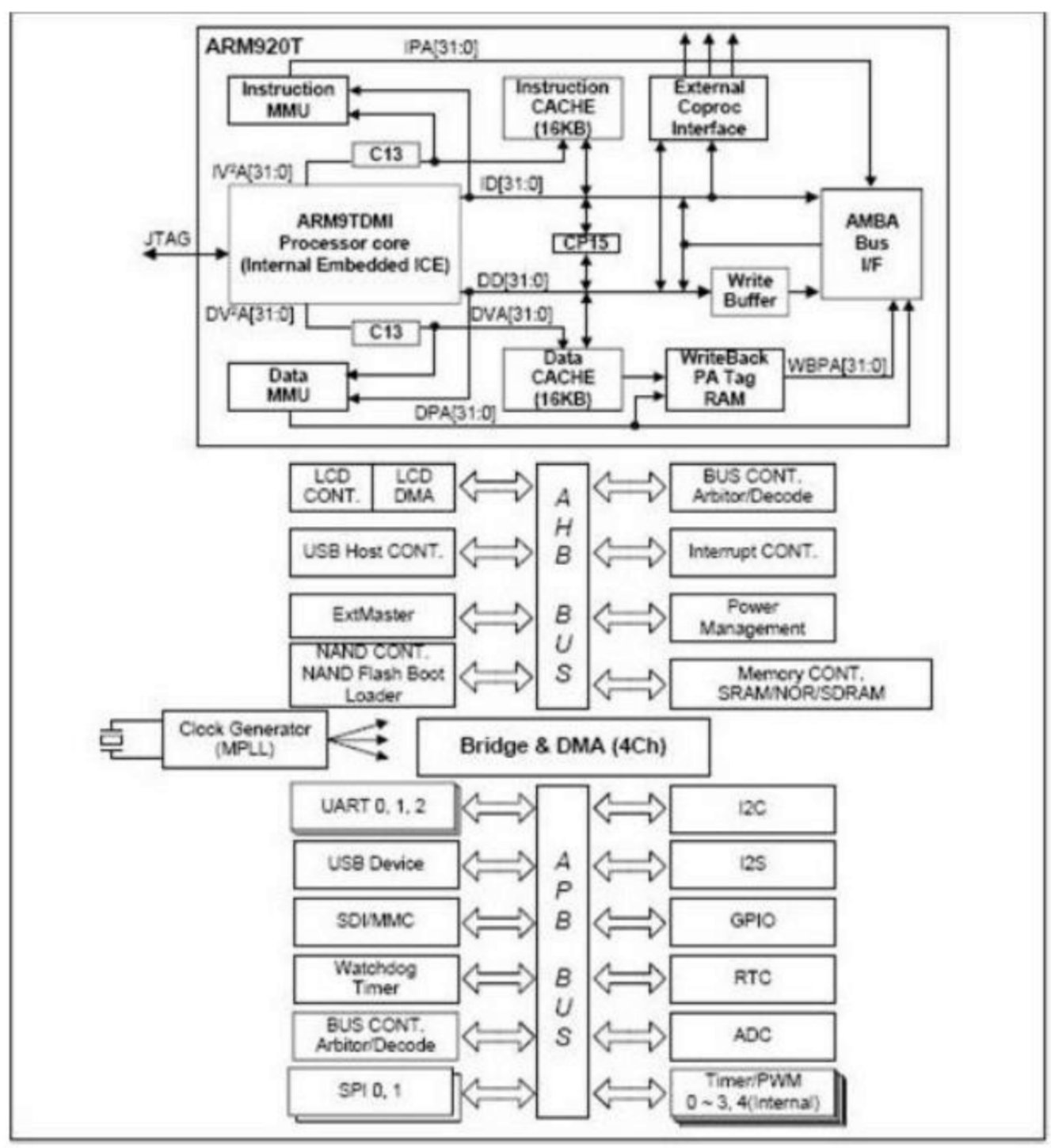
3.6 高清多媒体接口 **HDMI**

3.7 常用简单输入设备（键盘、触摸屏、传感器）

3.8 常用简单输出设备（**LED**、数码管、**LCD**、）

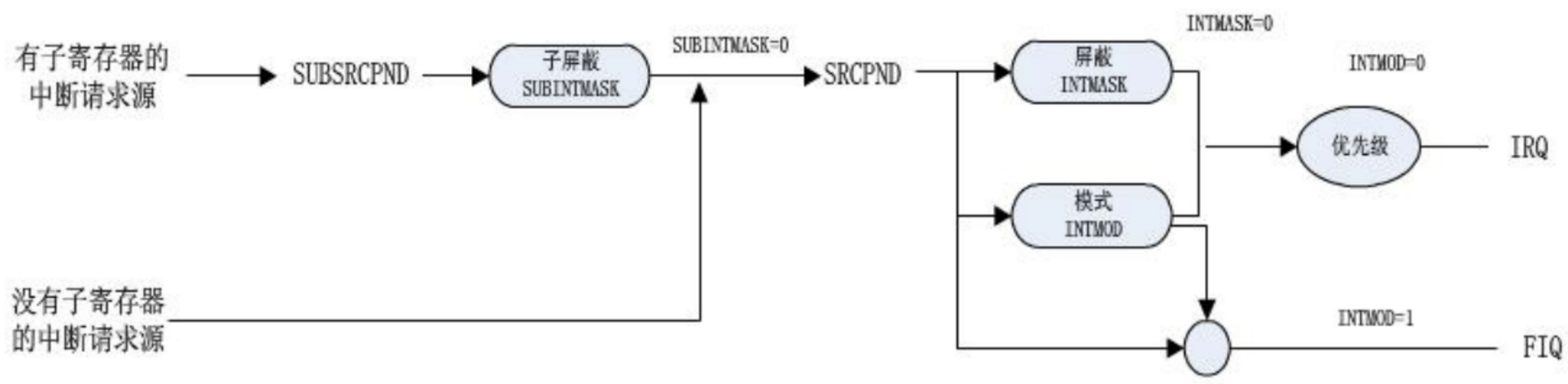
3.9 基于 **UART** 的 **RS—232/RS—485** **CAN** 总线接口 以太网通信接口常用无线通信接口（**GPS** 模块、**GPRS** 模块、**WiFi** 模块、蓝牙模块、射频无线收发模块）

4. 基于 **ARM** 内核的典型嵌入式处理芯片（**S3C2410/S3C2440** 芯片的内部结构，如片上总线、**DMA**、时钟控制、中断控制、**GPIO**、**UART**、**I²C**、**SPI**、**Timer**、**RTC**、**WDT** 及其他硬件组件）



s3c2410 内部组成框图

4.1 中断控制器及中断控制



当多个中断源请求服务时，SRCPND 相应位写入 1，通过优先级仲裁当前的最高优先权的中断源对应的 INTPND 位写入 1，如果中断没有被屏蔽，CPU 将处理该中断。

四、嵌入式系统软件

1. 嵌入式系统的软件组成与实时操作系统（嵌入式系统软件组成，嵌入式操作系统的发展，实时系统与实时操作系统，微内核与宏内核，嵌入式操作系统的仿真平台等）

1.1 实时系统

任务时限的两种类型：截止时间，任务执行预设时间

实时系统属性:实时约束性，预测性，可靠性

RTOS 衡量时间一致性的术语：抖动

RTOS 的 7 个特征：①具有异步 IO 和中断处理能力；②任务切换时间和中断延迟时间的确定；③优先级中断和调度；④抢占式调度；⑤内存锁定；⑥连续文件；⑦同步

RTOS 的三个实时指标：响应时间，吞吐量，生存时间

RTOS 响应时间的具体指标：中断延迟时间，任务切换时间（进程切换时间或线程切换时间）

1.2 操作系统

1.2.1 操作系统整体介绍

强实时操作系统：Vxworks、pSOS、uCOS-II

普通实时操作系统：Wince、uClinux、Symbian

单内核操作系统：UNIX，Linux，Wince，Mac OS，Android OS，DOS

①缺点:占内存空间大，缺乏可扩展性，维护困难，可靠性较低，排除故障需重新编译

②优点：应用程序生成效率高，内核切换的开销小，对外来事件反应较快，操作系统内核运行效率较高

微内核操作系统：Symbian，VxWorks，QNX，uC/OS-II，iOS

特点：1）内核小巧；2）接口一致；3）各个功能模块之间松散耦合；4）基于客户机/服务器的体系结构，消息机制；5）微内核扩充方便，但是功能切换开销大

1.2.2 操作系统介绍

(1)Android

以 Linux 为基础的半开放源代码操作系统，用于移动便携涉笔

(2) uC/OS -II/III

微内核，开放源码，不包含设备驱动程序和文件系统，优先级抢占调度方式，(**uC/OS -III** 是商业软件),支持 64 个任务数（8 个留给系统），有良好的裁剪性和可移植性，不支持时间片轮转调度(每个任务优先级不同)，每个任务拥有自有栈。

四层软件结构：应用软件层、API 层、uC/OS-II 内核、驱动设备层

(3)Linux

大型操作系统，开放源码。

Linux 主要由用户进程、OS 服务组件和 Linux 内核组成。

Linux 内核的主要作用有：进程管理与调度、内存管理、文件管理、网络接口和进程间通信，Linux 内核为用户进程提供了一个虚拟接口（系统调用），系统调用是 Linux 的用户进程和操作系统之间的重要接口。

经典的有 RTAI，uClinux，RTlinux。

(4)VxWorks

实时性最强可靠性最高的 RTOS，基于抢占式优先级调度，具有配套 Tornado 集成开发环境，大型商用操作系统。

移植 VxWorks 有四种操作类型：①宿主机移植；②体系结构移植；③目标板移植；④升级以前 Tornado 集成开发环境的 BSP 到最新版本。

(5)Unix

单内核，与 linux 源代码完全不同，指令函数参数十分相同

(6)QNX

加拿大 QNX 公司开发，微内核，支持进程调度、进程间通信、底层网络通信、中断处理。

(7)IOS

微内核，分为 4 个层次：核心操作系统、核心服务层、媒体层、触摸框架层

2．板级支持软件包（**BSP**）和引导加载程序 **Bootloader**（硬件抽象层 **HAL**，**BSP** 的功能和移植，**Bootloader** 的执行过程，**U-boot** 及其移植等）

硬件抽象层 **HAL**：指的是嵌入式软件中直接访问底层硬件的例程集合，它对底层硬件的初始化程序或者控制程序进行封装，向上提供了访问底层硬件的函数接口。

特点：①HAL 是嵌入式硬件电路板的基本软件；

②HAL 基本上是用标准 C 编写的；

③负责嵌入式操作系统的工程师只需要集中精力在 HAL 上做关键代码的编程工作。

板级支持包 BSP：BSP 是一些汇编程序和 C 语言代码相结合的操作系统底层软件，系统上层软件通过 BSP 访问硬件。BSP 支持的硬件分为 4 类：与中央执行单元架构相关的部分，与系统总线相关的部分，与内存相关的部分，外设部分。

HAL 和 BSP 异同：HAL 的 API 具有通用库的特点，适用于应用程序直接控制硬件设备，或者让其他操作系统与（不限定具体的操作系统）使用通用库函数的调用规则来使用 HAL 中的函数，间接访问硬件设备。BSP 则具有操作系统专用规格的特点。对用同一个电路板，HAL 一般只有一个，但对于不同的操作系统则拥有不同的 BSP。

嵌入式 Linux 的 Bootloader 执行步骤：

① 初始化处理器及外设的硬件资源配置；

② 建立内存空间的映射关系，使能 MMU；

③ 装载操作系统镜像到内存；

④ 对 Flash 存储器编程；

⑤ 运行操作系统；

⑥ 传递系统启动参数；

⑦ 命令行解析和输入输出控制。

3．嵌入式 **Linux** 操作系统（嵌入式 **Linux** 的发展和自由软件，嵌入式 **Linux** 内核的结构、系统调用接口，常见嵌入式 **Linux** 等）

4．嵌入式操作系统 **µC/OS-II**（基本特点、代码结构、任务管理与调度、任务通信、中断处理、移植等）

µC/OS-II 启动过程：①操作系统初始化 OSInit();②任务创建 OSTASKCreate();③任务调度开始 OSStart();

OSInit()函数先建立最初的任务就绪表，然后建立 4 个空白的数据链表，分别是空白任务控制块链表、空白事件控制块链表、空白标志链表和空白内存控制块链表

µC/OS-II 任务创建：OSTASKCreate 和 OSTASKCreateExt

µC/OS-II 任务调度**:** µC/OS-II 总是运行处于就绪态任务中优先级最高的一个，确定优先级是由调度器（schedule）完成的。任务级调度：OSSched()完成，中断级调度由 OSIntExit()完成。

µC/OS-II 任务切换：上下文切换或文境切换，OSCtwsw()完成，工作现场保存在自用栈中。

µC/OS-II 中断处理：中断主要使用汇编语言编写，中断任务的执行现场保存在自用栈中。允许中断嵌套，嵌套层数最多 255 层，OSIntEnter()和 OSIntExit()完成。

µC/OS-II 时钟管理：节拍率应该在 10~100Hz。时钟节拍率越高，系统的额外负荷就越重。用户必须在多任务系统启动后以后再激活时钟节拍信号发生器，也就是在调用 OSStart()之后。换言之，在调用 OSStart()之后的第一件事就是初始化定时器中断。

µC/OS-II 任务间通信

五、嵌入式系统的开发

1．嵌入式系统的开发过程和工具（开发步骤，交叉开发平台和工具，系统的调试工具等）

开发步骤：

①需求分析与规格说明；②系统设计(概要设计或者总体设计)；

③构件设计（详细设计）；④系统集成与测试

嵌入式系统开发特点：

①使用宿主机-目标机的开发架构； ②与底层硬件控制程序密切相关；

③软硬件资源受到限制； ④需要固化程序

嵌入式系统的调试：

①在线仿真器（ICE），主要针对低端的嵌入式系统；

②片上扫描技术（OCD）—JTAG，适用于高速嵌入式系统的开发；

③驻留监控软件调试；

④指令集模拟器（ISS），在宿主机上模拟应用程序在目标机上运行行为的一个软件，ARMulator 和 SkyEye

2．系统开发工具软件（**ADS**、**RVDS** 的特点与使用，**GCC** 的常用命令与参数）

ADS1.2：是 ARM 公司针对 ARM 系列 CPU 为核心的嵌入式操作系统常用的工具套件；

连接中的地址映射：对工程项目中的各段代码和数据进行存储空间的地址分配，以确定在目标机上实际存储空间中的地址值。

Simple 连接类型

Scattered 连接类型：需要提供一个 Scattered 格式的配置文件，该配置文件是一个文本文件，用于指定生成的输出文件中地址映射关系，其中包含了各域及各段的分组及定位信息，一个 Scattered 文件通常要描述一个下载时域和若干运行时域。

RVDS：是 ARM 公司推出的基于 ARM 系列 CPU 进行开发的工具套件，最好的 ARM 开发工具。相对于 ADS1.2，生成的代码紧凑，执行效率高。

GCC：Linux 操作系统环境下应用程序编译工具，支持 C、C++及汇编语言

gcc –o test test.c:生成名为 test 的输出文件

gcc –c –o test.o test.c：生成名为 test.o 的目标文件

gcc –S –o test.s test.c：生成名为 test.s 的汇编文件

gcc –E –o test.i test.c：源文件预处理，生成名为 test.i 的文件

gcc –g –o test test.c：生成名为 test 的输出文件并加入 GDB 调试信息

GDB：GUN 开发工具中的程序调试工具，在编译是应包含调试信息，即在编译命令中加入 -g 参数

3．以 **S3C2410/S3C2440** 为背景的应用系统开发（硬件接口及部件的综合使用；无操作系统环境下的系统开发；**μC/OS-II** 环境下的系统开发）

考试方式

考试时间：120 分钟，满分 100 分

包含：选择题（40 分）、填空题（40 分）、综合题（20 分）