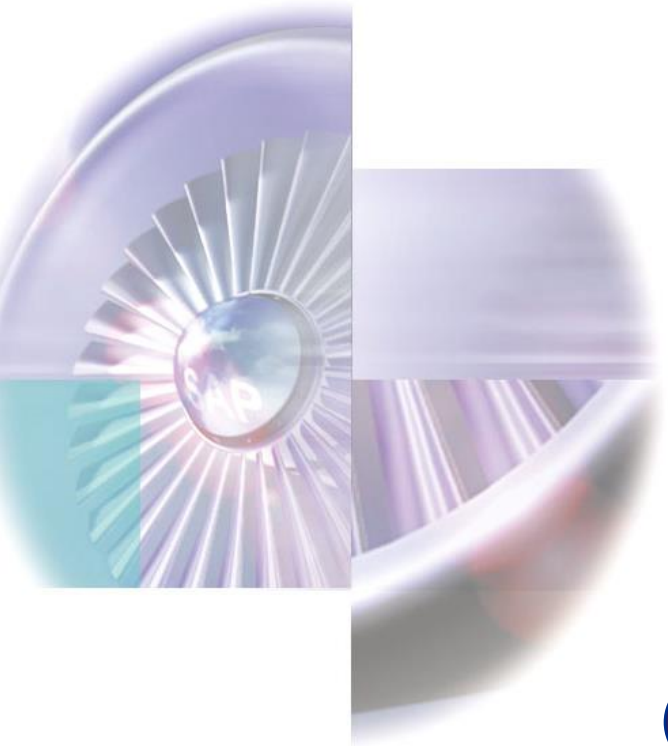
暨南大學
JINAN UNIVERSITY

第5章 S3C2410处理器概述与嵌入式系统架构

杨光华

物联网与物流工程研究院 / 电气信息学院
办公室：行政楼 631
电邮：ghyang@jnu.edu.cn 电话：8505687

第5章 嵌入式系统架构



1. S3C2410处理器
2. 嵌入式系统硬件结构
3. 嵌入式系统软件结构

S3C2410处理器

1. 功能模块与总线结构
2. 设备控制器
3. 设备控制器设备侧接口
4. 设备控制器寄存器

S3C2410简介

- S3C2410是Samsung公司推出的16/32位RISC处理器，主要面向手持设备以及高性价比、低功耗的应用。CPU内核采用ARM920T处理器

S3C2410A提供一组完整的系统外围设备：

✓1.8V/2.0V内核供电，3.3V存储器供电，3.3V外部I/O供电；

✓具有16KB的ICache和16KB的DCache以及MMU；

✓外部存储器控制器；

✓LCD控制器提供1通道LCD专用DMA

✓4通道DMA并有外部请求引脚；

✓3通道UART和2通道SPI；

✓1通道多主机IIC总线和1通道IIS (Inter-IC Sound)总线控制器；

✓SD主接口版本1.0和MMC卡协议2.11兼容版

✓2个USB主设备接口，1个USB从设备接口

✓4通道PWM定时器和1通道内部定时器

✓看门狗定时器

✓117位通用I/O口和24通道外部中断源

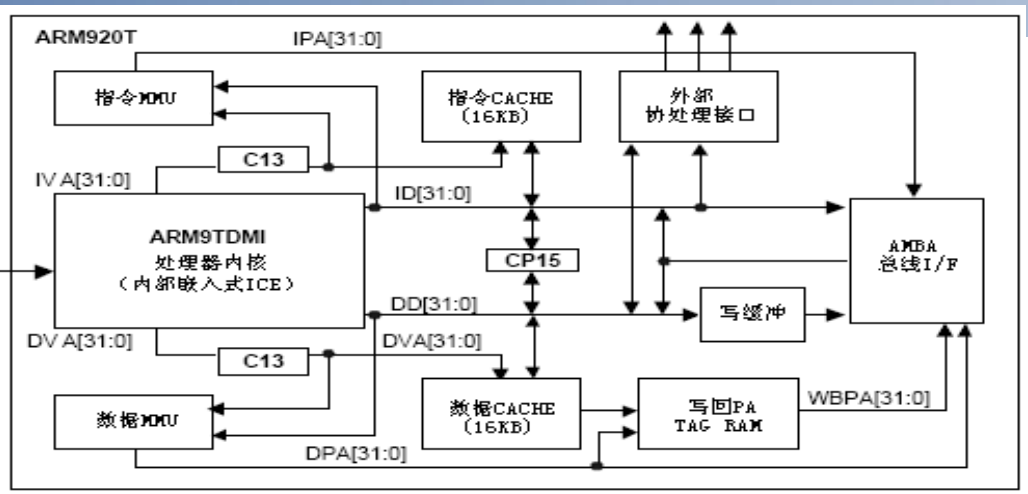
✓电源控制模式包括：正常、慢速、空闲和掉电四种模式

✓ 8通道10位ADC和触摸屏接口

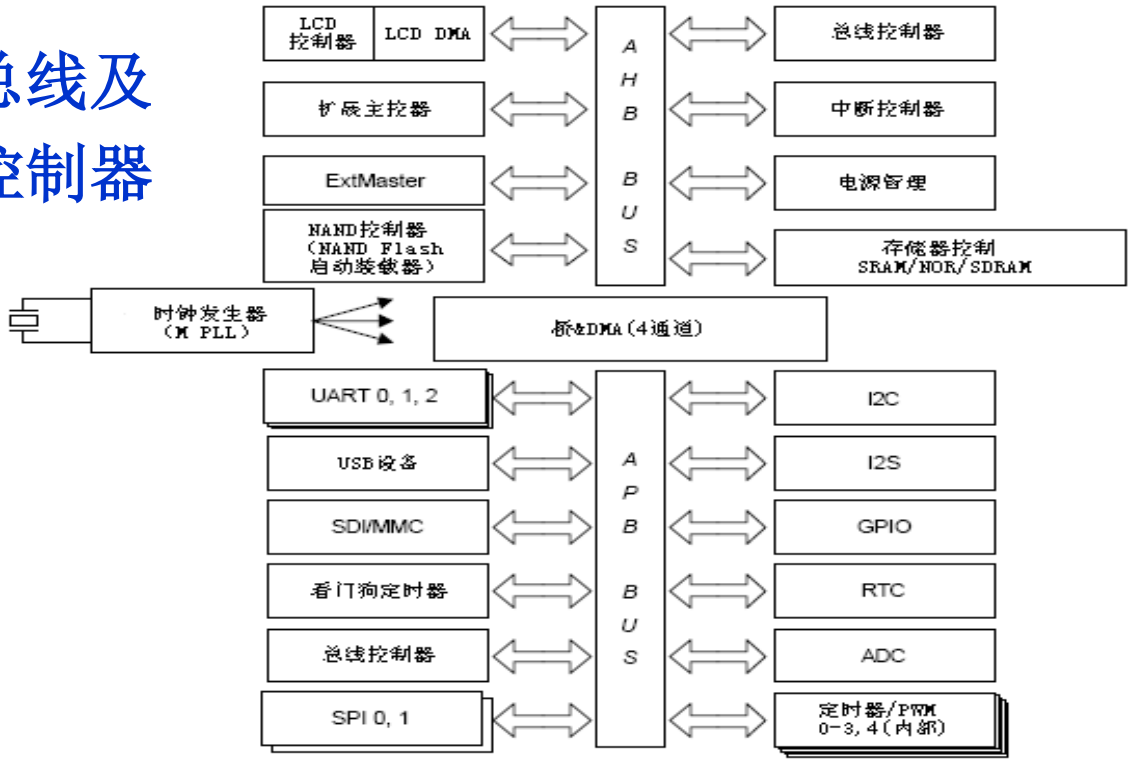
✓具有日历功能的RTC

✓使用PLL的片上时钟发生器

1. 功能模块与总线结构



片上两条总线及
众多设备控制器



S3C2410处理器

1. 功能模块与总线结构
2. 设备控制器
3. 设备控制器设备侧接口
4. 设备控制器寄存器

2. 设备控制器

- 设备控制器是CPU核与外部设备间的桥接器，在CPU侧与设备侧各有一个接口

  - CPU侧接口用于挂接总线，与CPU核相连，包括数据、地址与控制三组总线
  - 设备侧接口用于挂接外部设备，通常包括有数据、状态与控制三类信号线

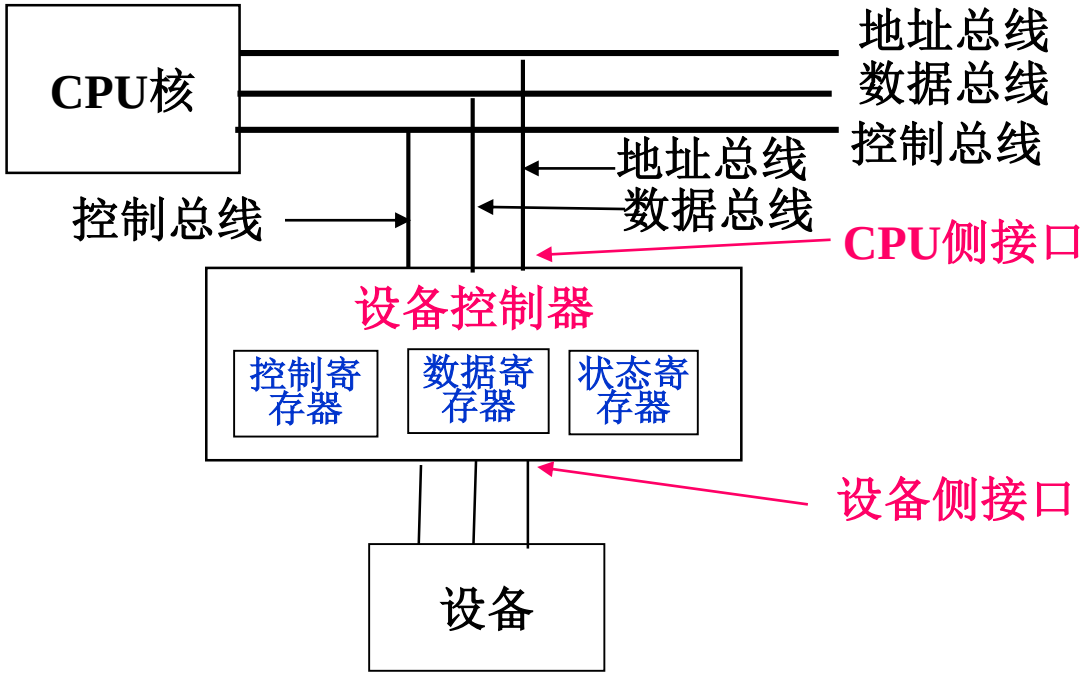


图 6.2 设备控制器结构框图

2. 设备控制器

- 设备控制器还应包括数据、控制与状态三类寄存器

  - 数据寄存器用于存储CPU核发往设备的数据或设备向CPU核提供的数据
  - 控制寄存器用于接收CPU核发往设备的命令，并对命令进行译码
  - 状态寄存器用于存储设备的状态，供CPU核读取

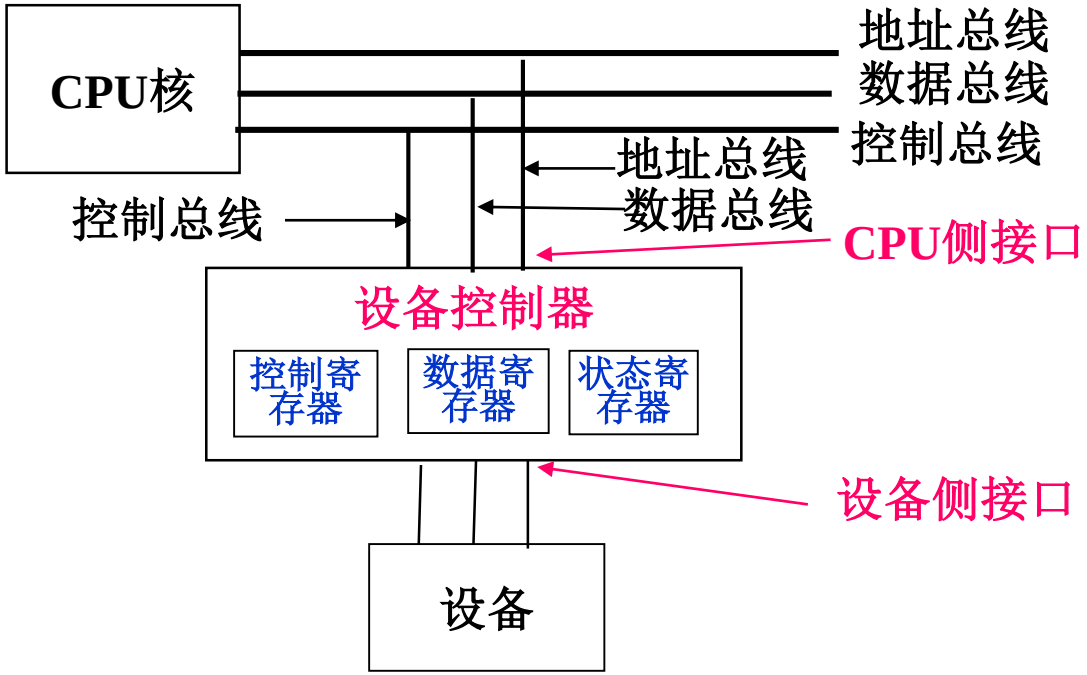


图 6.2 设备控制器结构框图

S3C2410处理器

1. 功能模块与总线结构
2. 设备控制器
3. 设备控制器设备侧接口
4. 设备控制器寄存器

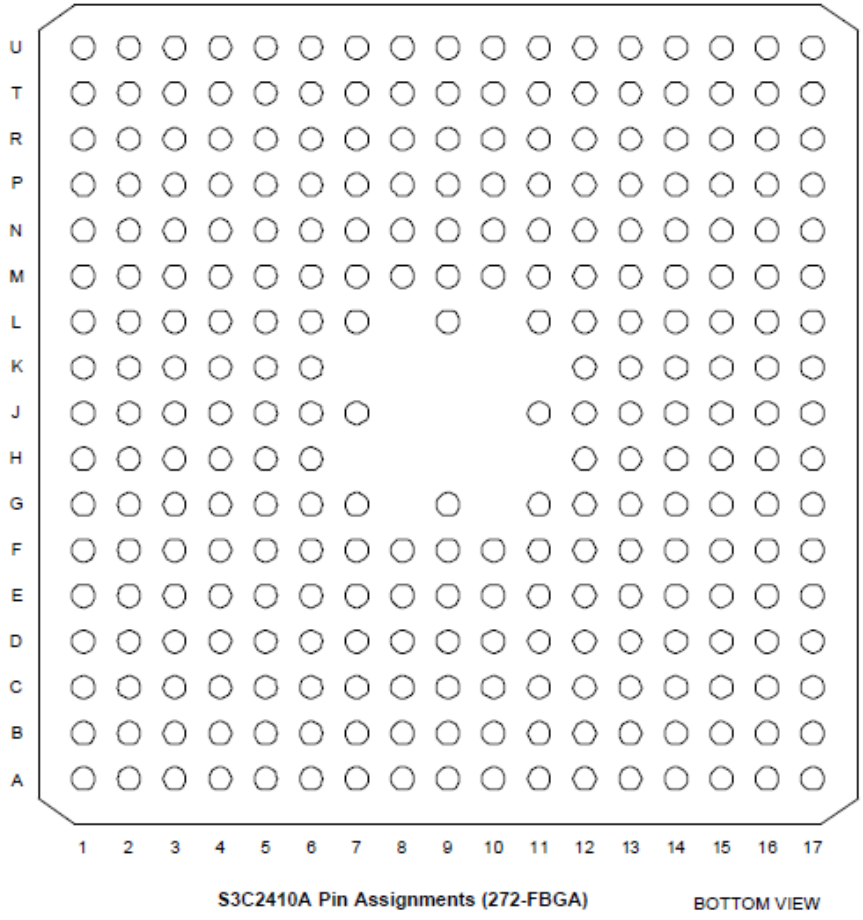
3. 设备控制器设备侧接口

272-FBGA封装

主要引脚信号：

- addr0---addr26
- Data0---data31
- GPA0---GPA22
- GPB10、GPC15
- GPD15、GPE15
- GPF7、GPG15
- GPH10、EINT23
- nGCS0—nGCS7
- AIN7、IIC、SPI
- OM0---OM3

大部分都是复用的



S3C2410A Pin Assignments (272-FBGA) BOTTOM VIEW

3. 设备控制器设备侧接口

Bus Controller		
OM [1:0]	I	OM [1:0] sets S3C2410A in the TEST mode, which is used only at fabrication. Also, it determines the bus width of nGCS0. The pull-up/down resistor determines the logic level during the RESET cycle. 00:Nand-boot 01:16-bit 10:32-bit 11:Test mode
ADDR [26:0]	O	ADDR [26:0] (Address Bus) outputs the memory address of the corresponding bank.
DATA [31:0]	IO	DATA [31:0] (Data Bus) inputs data during memory read and outputs data during memory write. The bus width is programmable among 8/16/32-bit.
nGCS [7:0]	O	nGCS [7:0] (General Chip Select) are activated when the address of a memory is within the address region of each bank. The number of access cycles and the bank size can be programmed.
nWE	O	nWE (Write Enable) indicates that the current bus cycle is a write cycle.
nOE	O	nOE (Output Enable) indicates that the current bus cycle is a read cycle.
nXBREQ	I	nXBREQ (Bus Hold Request) allows another bus master to request control of the local bus. BACK active indicates that bus control has been granted.
nXBACK	O	nXBACK (Bus Hold Acknowledge) indicates that the S3C2410A has surrendered control of the local bus to another bus master.
nWAIT	I	nWAIT requests to prolong a current bus cycle. As long as nWAIT is L, the current bus cycle cannot be completed. If nWAIT signal isn't used in your system, nWAIT signal must be tied on pull-up resistor.

3. 设备控制器设备侧接口

SDRAM/SRAM		
nSRAS	O	SDRAM Row Address Strobe
nSCAS	O	SDRAM Column Address Strobe
nSCS [1:0]	O	SDRAM Chip Select
DQM [3:0]	O	SDRAM Data Mask
SCLK [1:0]	O	SDRAM Clock
SCKE	O	SDRAM Clock Enable
nBE [3:0]	O	Upper Byte/Lower Byte Enable (In case of 16-bit SRAM)
nWBE [3:0]	O	Write Byte Enable

NAND Flash		
CLE	O	Command Latch Enable
ALE	O	Address Latch Enable
nFCE	O	NAND Flash Chip Enable
nFRE	O	NAND Flash Read Enable
nFWE	O	NAND Flash Write Enable
NCON	I	NAND Flash Configuration. If NAND Flash Controller isn't used, it has to be tied on pull-up resistor.
R/nB	I	NAND Flash Ready/Busy. If NAND Flash Controller isn't used, it has to be tied on pull-up resistor.

3. 设备控制器设备侧接口

LCD Control Unit		
VD [23:0]	O	**STN/TFT/SEC TFT:** LCD Data Bus
LCD_PWREN	O	**STN/TFT/SEC TFT:** LCD panel power enable control signal
VCLK	O	**STN/TFT:** LCD clock signal
VFRAME	O	**STN:** LCD Frame signal
VLINE	O	**STN:** LCD line signal
VM	O	**STN:** VM alternates the polarity of the row and column voltage
VSYNC	O	**TFT:** Vertical synchronous signal
HSYNC	O	**TFT:** Horizontal synchronous signal
VDEN	O	**TFT:** Data enable signal
LEND	O	**TFT:** Line End signal
STV	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
CPV	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
LCD_HCLK	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
TP	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
STH	O	**SEC TFT:** SEC (Samsung Electronics Company) TFT LCD panel control signal
LCDVF [2:0]	O	**SEC TFT:** Timing control signal for specific TFT LCD (OE/REV/REVB)
Interrupt Control Unit		
EINT [23:0]	I	External Interrupt request
DMA		
nXDREQ [1:0]	I	External DMA request
nXDACK [1:0]	O	External DMA acknowledge

3. 设备控制器设备侧接口

UART		
RxD [2:0]	I	UART receives data input
TxD [2:0]	O	UART transmits data output
nCTS [1:0]	I	UART clear to send input signal
nRTS [1:0]	O	UART request to send output signal
UEXTCLK	I	UART clock signal
ADC		
AIN [7:0]	AI	ADC input [7:0]. If it isn't used pin, it has to be in Ground.
Vref	AI	ADC Vref
IIC-Bus		
IICSDA	IO	IIC-bus data
IICSCL	IO	IIC-bus clock
IIS-Bus		
I2SLRCK	IO	IIS-bus channel select clock
I2SSDO	O	IIS-bus serial data output
I2SSDI	I	IIS-bus serial data input
I2SSCLK	IO	IIS-bus serial clock
CDCLK	O	CODEC system clock
Touch Screen		
nXPON	O	Plus X-axis on-off control signal
XMON	O	Minus X-axis on-off control signal
nYPON	O	Plus Y-axis on-off control signal
YMON	O	Minus Y-axis on-off control signal

3. 设备控制器设备侧接口

USB Device		
PDN0	IO	DATA (–) for USB peripheral. (470Kohm pull-down)
PDP0	IO	DATA (+) for USB peripheral. (1.5Kohm pull-up)
SPI		
SPIMISO [1:0]	IO	SPIMISO is the master data input line, when SPI is configured as a master. When SPI is configured as a slave, these pins reverse its role.
SPIMOSI [1:0]	IO	SPIMOSI is the master data output line, when SPI is configured as a master. When SPI is configured as a slave, these pins reverse its role.
SPICLK [1:0]	IO	SPI clock
nSS [1:0]	I	SPI chip select (only for slave mode)

SD		
SDDAT [3:0]	IO	SD receive/transmit data
SDCMD	IO	SD receive response/ transmit command
SDCLK	O	SD clock
General Port		
GPn [116:0]	IO	General input/output ports (some ports are output only)
TIMMER/PWM		
TOUT [3:0]	O	Timer Output [3:0]
TCLK [1:0]	I	External timer clock input

3. 设备控制器设备侧接口

JTAG TEST LOGIC		
nTRST	I	nTRST (TAP Controller Reset) resets the TAP controller at start. If debugger is used, A 10K pull-up resistor has to be connected. If debugger (black ICE) is not used, nTRST pin must be issued by a low active pulse (Typically connected to nRESET).
TMS	I	TMS (TAP Controller Mode Select) controls the sequence of the TAP controller's states. A 10K pull-up resistor has to be connected to TMS pin.
TCK	I	TCK (TAP Controller Clock) provides the clock input for the JTAG logic. A 10K pull-up resistor must be connected to TCK pin.
TDI	I	TDI (TAP Controller Data Input) is the serial input for test instructions and data. A 10K pull-up resistor must be connected to TDI pin.
TDO	O	TDO (TAP Controller Data Output) is the serial output for test instructions and data.
Reset, Clock & Power		
nRESET	ST	nRESET suspends any operation in progress and places S3C2410A into a known reset state. For a reset, nRESET must be held to L level for at least 4 FCLK after the processor power has been stabilized.
nRSTOUT	O	For external device reset control (nRSTOUT = nRESET & nWDTRST & SW_RESET)
PWREN	O	2.0V core power on-off control signal
nBATT_FLT	I	Probe for battery state (Does not wake up at power-off mode in case of low battery state). If it isn't used, it has to be High (3.3V).
OM [3:2]	I	OM [3:2] determines how the clock is made. OM [3:2] = 00b, Crystal is used for MPLL CLK source and UPLL CLK source. OM [3:2] = 01b, Crystal is used for MPLL CLK source and EXTCLK is used for UPLL CLK source. OM [3:2] = 10b, EXTCLK is used for MPLL CLK source and Crystal is used for UPLL CLK source. OM [3:2] = 11b, EXTCLK is used for MPLL CLK source and UPLL CLK source.

3. 设备控制器设备侧接口

Signal	I/O	Description
Reset, Clock & Power (Continued)		
EXTCLK	I	External clock source. When OM [3:2] = 11b, EXTCLK is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 10b, EXTCLK is used for MPLL CLK source only. When OM [3:2] = 01b, EXTCLK is used for UPLL CLK source only. If it isn't used, it has to be High (3.3V).
XTIpll	AI	Crystal Input for internal osc circuit. When OM [3:2] = 00b, XTIpll is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 01b, XTIpll is used for MPLL CLK source only. When OM [3:2] = 10b, XTIpll is used for UPLL CLK source only. If it isn't used, XTIpll has to be High (3.3V).
XTOpll	AO	Crystal Output for internal osc circuit. When OM [3:2] = 00b, XTIpll is used for MPLL CLK source and UPLL CLK source. When OM [3:2] = 01b, XTIpll is used for MPLL CLK source only. When OM [3:2] = 10b, XTIpll is used for UPLL CLK source only. If it isn't used, it has to be a floating pin.
MPLLCAP	AI	Loop filter capacitor for main clock.
UPLLCAP	AI	Loop filter capacitor for USB clock.
XTIrtc	AI	32.768 kHz crystal input for RTC. If it isn't used, it has to be in High (RTCVDD = 1.8V).
XTOrtc	AO	32.768 kHz crystal output for RTC. If it isn't used, it has to be Float.
CLKOUT [1:0]	O	Clock output signal. The CLKSEL of MISCCR register configures the clock output mode among the MPLL CLK, UPLL CLK, FCLK, HCLK and PCLK.

3. 设备控制器设备侧接口

Signal	I/O	Description
Power		
VDDalive	P	S3C2410A reset block and port status register V_{DD} (1.8V / 2.0V). It should be always supplied whether in normal mode or in power-off mode.
VDDi/VDDiarm	P	S3C2410A core logic V_{DD} (1.8V / 2.0V) for CPU.
VSSi/VSSiarm	P	S3C2410A core logic V_{SS}
VDDi_MPLL	P	S3C2410A MPLL analog and digital V_{DD} (1.8V / 2.0V).
VSSi_MPLL	P	S3C2410A MPLL analog and digital V_{SS}.
VDDOP	P	S3C2410A I/O port V_{DD} (3.3V)
VDDMOP	P	S3C2410A Memory I/O V_{DD} 3.3V: SCLK up to 133MHz
VSSMOP	P	S3C2410A Memory I/O V_{SS}
VSSOP	P	S3C2410A I/O port V_{SS}
RTCVDD	P	RTC V_{DD} (1.8 V, Not support 2.0 and 3.3V) (This pin must be connected to power properly if RTC isn't used)
VDDi_UPLL	P	S3C2410A UPLL analog and digital V_{DD} (1.8V / 2.0V)
VSSi_UPLL	P	S3C2410A UPLL analog and digital V_{SS}
VDDA_ADC	P	S3C2410A ADC V_{DD} (3.3V)
VSSA_ADC	P	S3C2410A ADC V_{SS}

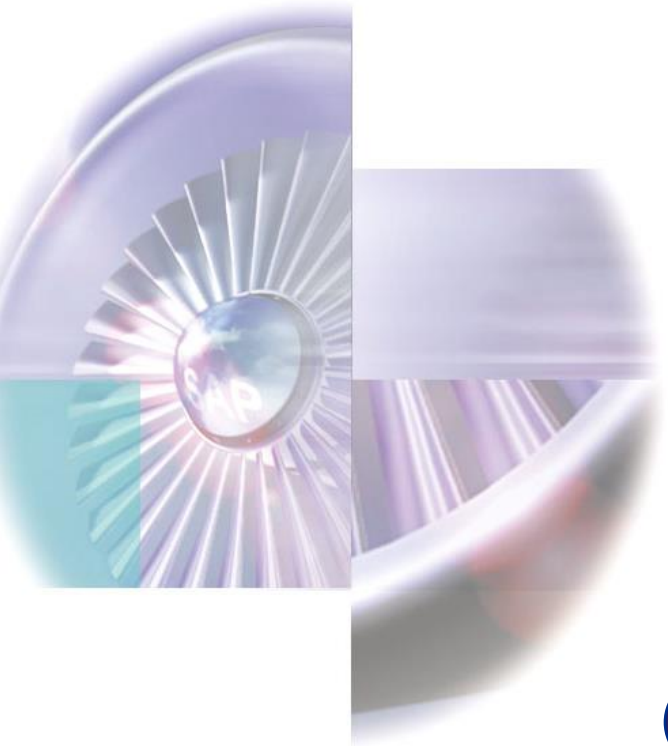
S3C2410处理器

1. 功能模块与总线结构
2. 设备控制器
3. 设备控制器设备侧接口
4. 设备控制器寄存器

4. 设备控制器寄存器

- 控制器的寄存器有控制寄存器、状态寄存器与数据寄存器三类
  - 控制寄存器用于接受CPU核的命令，对命令进行译码，产生设备控制信号
  - 状态寄存器用于存储设备的状态，供CPU核查询
  - 数据寄存器用于存储CPU核写往设备的数据或设备向CPU核提供的数据
- S3C2410片上集成的设备控制器的寄存器定义及功能等详细细节请参考下一章内容

第5章 嵌入式系统架构



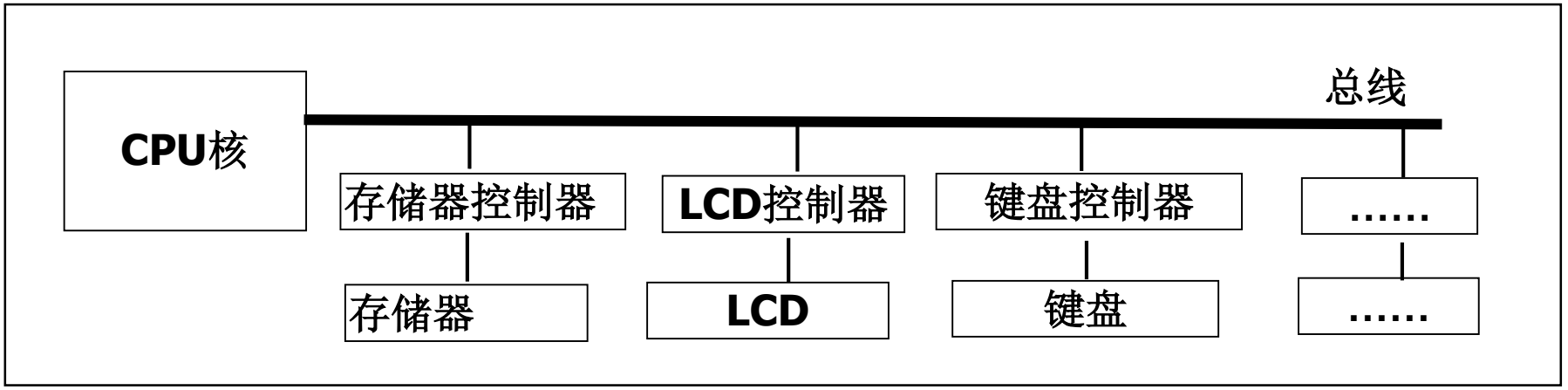
1. S3C2410处理器
2. 嵌入式系统硬件结构
3. 嵌入式系统软件结构

嵌入式系统硬件结构

1. 单总线结构
2. 多总线结构

1. 单总线结构

- 嵌入式系统硬件由微处理器、总线、控制器与设备组成。最简单的结构是单总线结构，存储器、LCD与键盘等设备都通过其控制器挂接到一组总线上与CPU核进行通信。
- 单总线结构过于简单，通常用于揭示嵌入式系统硬件结构原理，实际嵌入式系统的硬件结构远比单总线复杂，是一种多总线结构



2. 多总线结构

- 包含一组以上总线的嵌入式系统硬件结构称为多组总线结构，简称多总线结构。实际的嵌入式系统硬件结构往往为多总线结构
- 多总线的具体结构形式主要取决于所用微处理器的总线结构与外设情况
- 为便于阐明嵌入式系统硬件结构，同时也不失一般性，我们选择三星公司的**S3C2410**为嵌入式微处理器

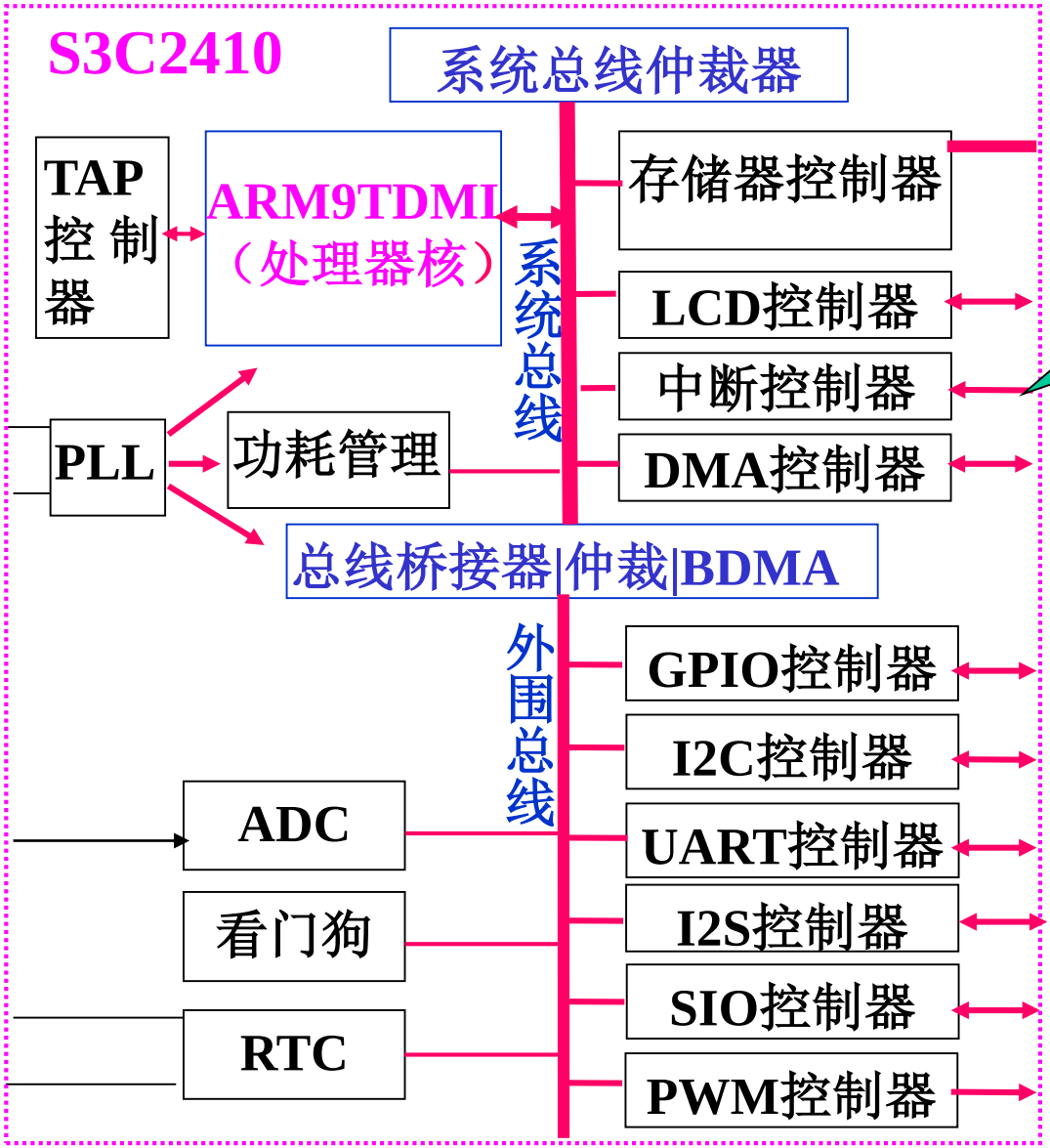
2. 多总线结构

并假设我们的嵌入式系统具有下列外设

- 1个复位开关
- 1个RS232串口
- 2MB Flash
- 8MB SDRAM
- 4Kbit EEPROM
- LCD及TSP触摸屏
- 4×4小键盘
- USB接口
- 10M 以太网接口
- MICROPHONE输入口
- IIS音频信号输出口，可接双声道SPEAKER
- 两个LED
- 1个20针JTAG接口
- 1个8段数码管
- 电源由DC5V提供或由USB接口提供

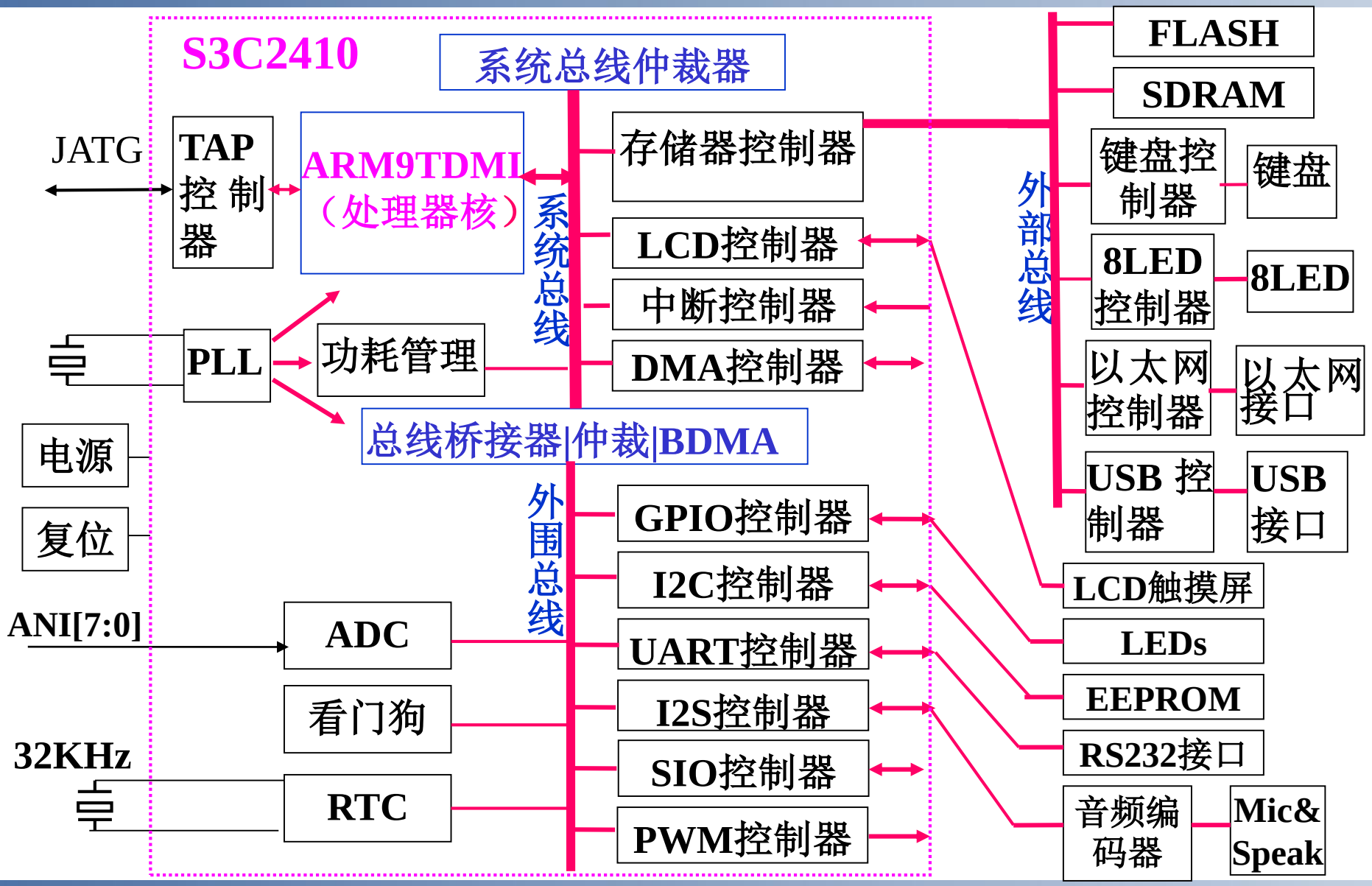
这样一个嵌入式系统硬件应采取怎样的结构呢?

2. 多总线结构

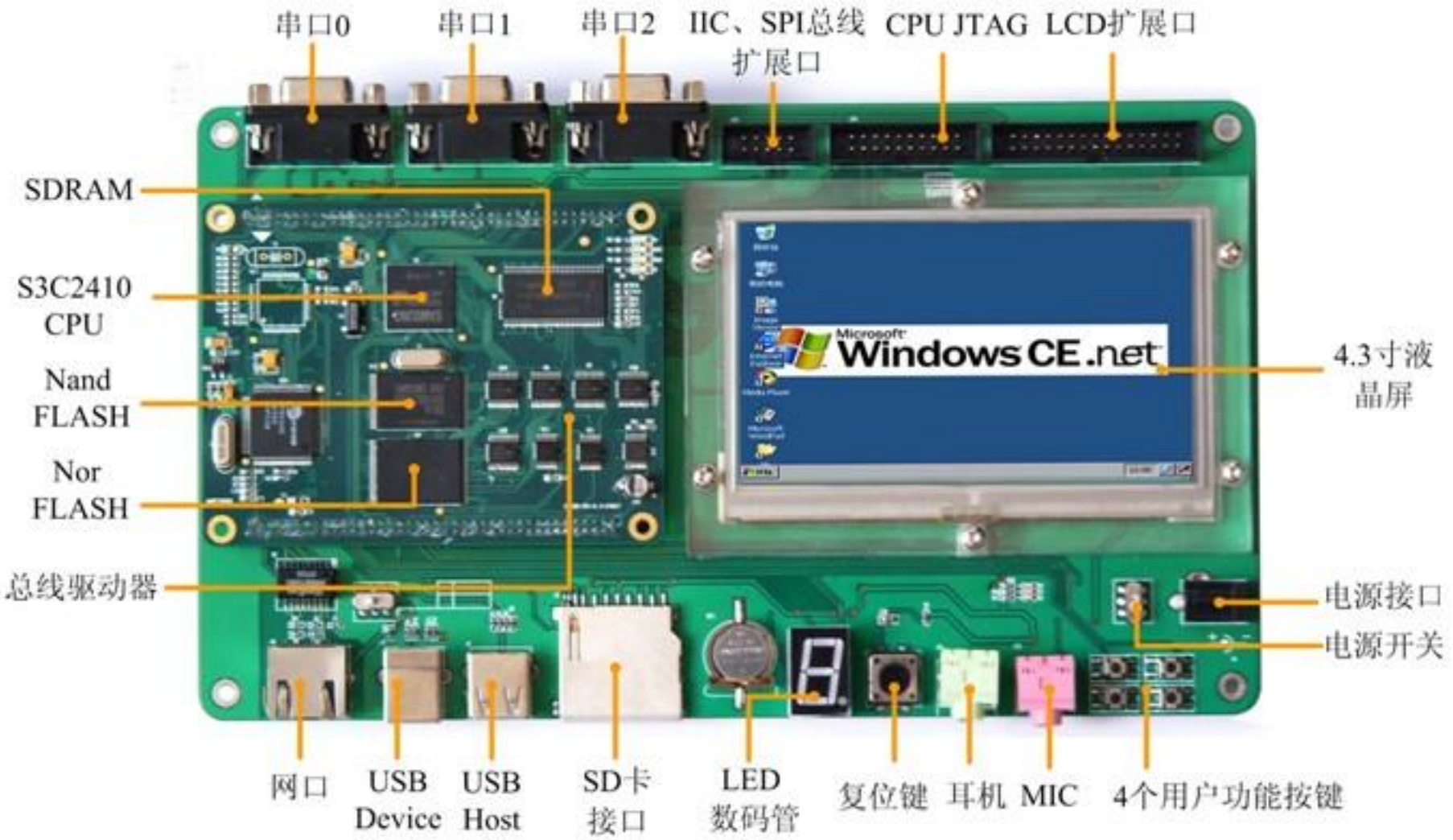


S3C2410 内部总线结构

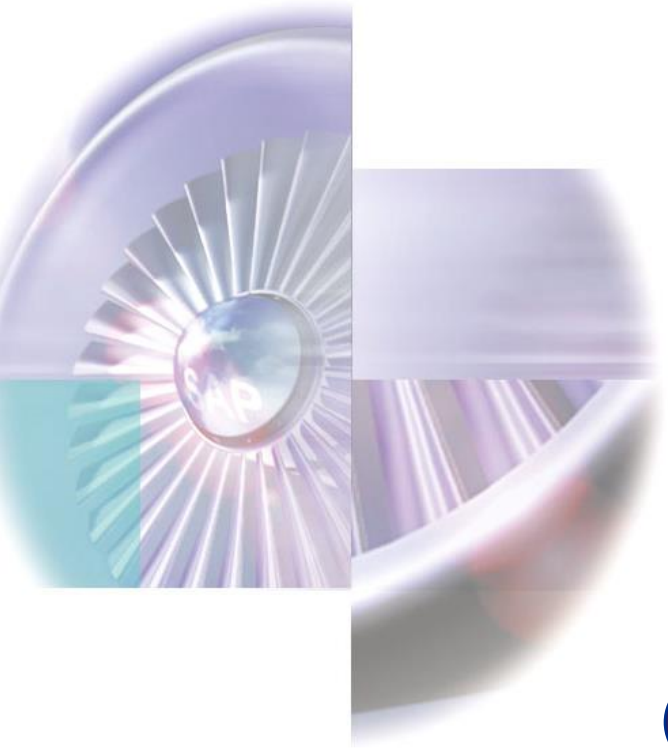
2. 多总线结构


S3C2410
系统总线仲裁器
JATG
TAP 控制器
ARM9TDMI（处理器核）
系统总线
存储器控制器
LCD控制器
中断控制器
DMA控制器
PLL
功耗管理
总线桥接器|仲裁|BDMA
电源
复位
外围总线
GPIO控制器
I2C控制器
UART控制器
I2S控制器
SIO控制器
PWM控制器
ANI[7:0]
ADC
看门狗
32KHz
RTC
外部总线
FLASH
SDRAM
键盘控制器
键盘
8LED控制器
8LED
以太网控制器
以太网接口
USB 控制器
USB接口
LCD触摸屏
LEDs
EEPROM
RS232接口
音频编码器
Mic&Speak

2. 多总线结构


串口0
串口1
串口2
IIC、SPI总线扩展口
CPU JTAG
LCD扩展口
SDRAM
S3C2410 CPU
Nand FLASH
Nor FLASH
总线驱动器
Microsoft Windows CE.net
4.3寸液晶屏
电源接口
电源开关
网口
USB Device
USB Host
SD卡接口
LED数码管
复位键
耳机
MIC
4个用户功能按键

第5章 嵌入式系统架构



1. S3C2410处理器
2. 嵌入式系统硬件结构
3. 嵌入式系统软件结构

嵌入式系统软件结构

- 嵌入式系统软件包括系统软件与应用软件两部分，系统软件主要是指操作系统，应用软件则是针对某一具体应用的软件
- 并非所有的嵌入式系统软件都包括有操作系统；因为需要处理的事务繁杂或是对实时性有严格要求，才需要操作系统
- 根据嵌入式系统中是否带有操作系统内核把其分为有核系统与无核系统
  - 从层次上看，无核系统是两层，有核系统则是三层
  - 从组成结构上看，无核系统软件采取单任务结构，有核系统则采取多任务结构

嵌入式系统软件结构

- 本节主要探讨嵌入式系统软件的单任务结构与多任务结构，为阐述方便，先虚构一个需求，设我们的嵌入式系统同时需要做如下4件事：

  1、LCD上滚动显示“暨南大学珠海校区”，显示位置每次不同

  2、8段数码管上按序显示0、1、…E、F，到F后又从0开始显示，如此重复

  3、LED不断闪烁，每1秒闪烁1次

  4、当按小键盘时，被按下的键出现在超级终端上

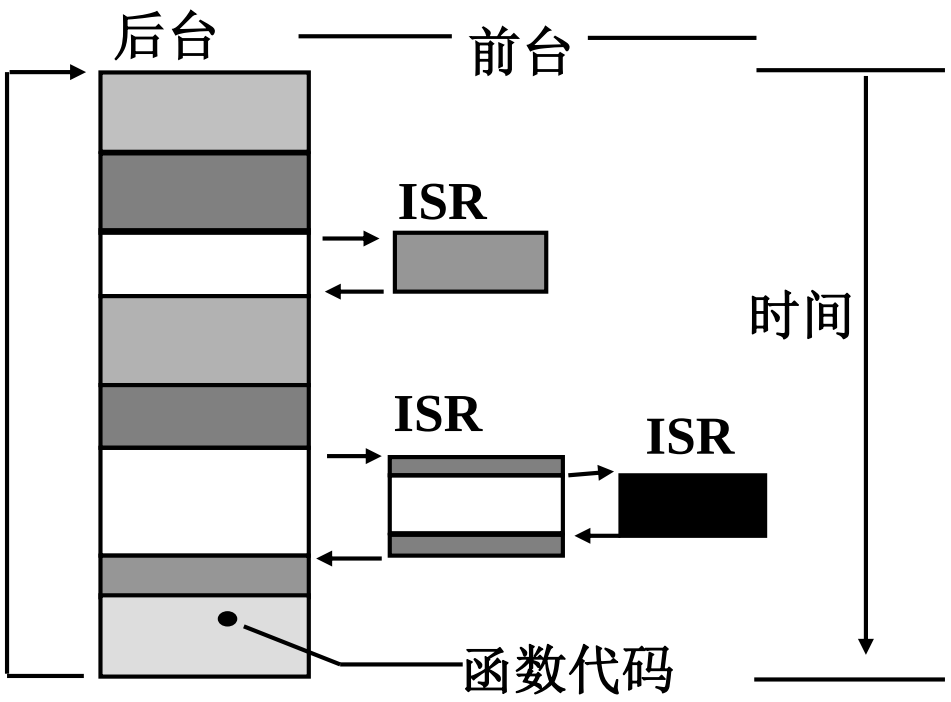
- 完成上述4件事的单任务结构与多任务结构软件是一个什么样的结构呢？

嵌入式系统软件结构

1. 单任务结构（无核结构）
2. 多任务结构（有核结构）

1. 单任务结构

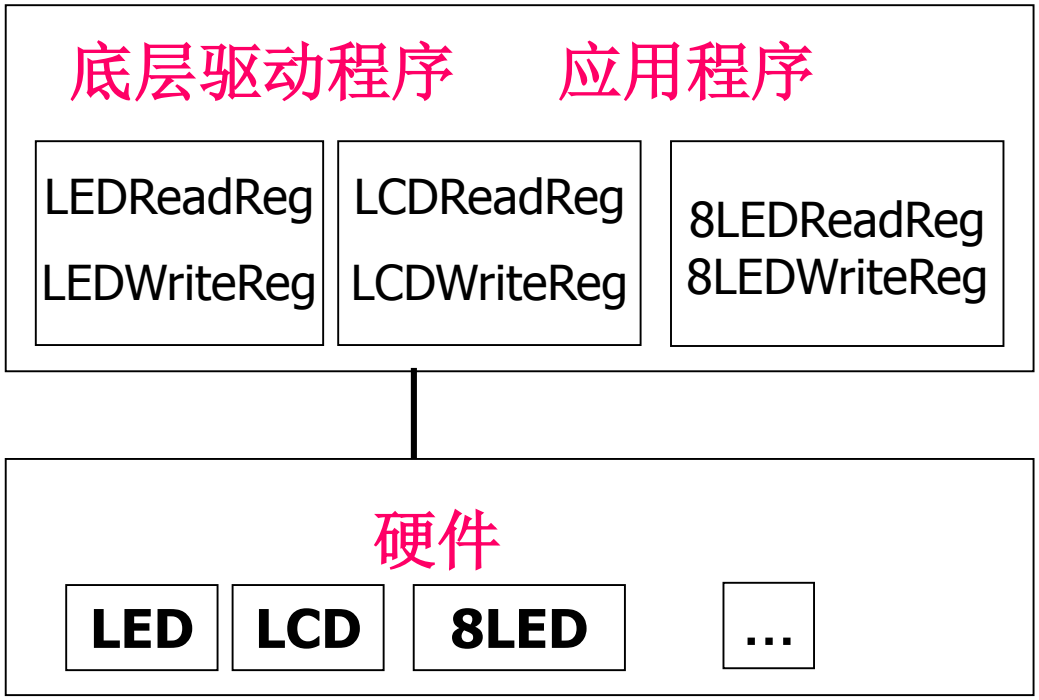
- 应用程序是一个无限循环，循环中调用一系列函数来完成其对应的操作，称后台操作，ISR完成的操作称前台操作
- 也称无核结构、前后台结构或超循环(Super-Loops)结构
- 任务级响应差
- 前面虚构的4件事如何实现？



1. 单任务结构

单任务结构实现一

这是什么结构？结构图？



应用程序中直接操作硬件寄存器，不另外设计驱动模块。弊端：

无可重用的驱动模块!!!

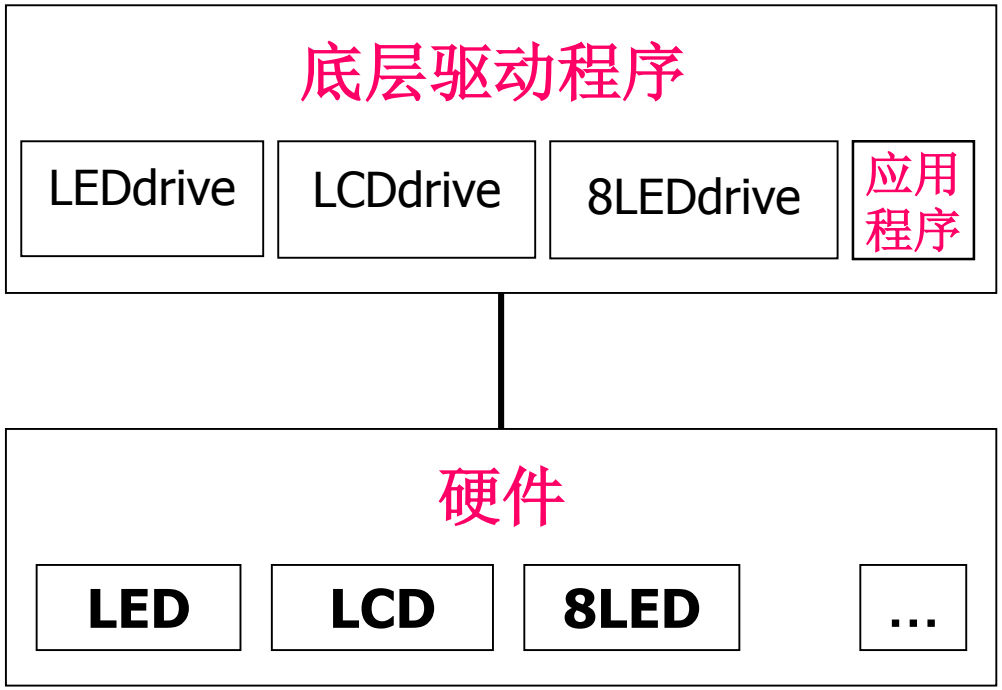
```
main()
{
   while(1)
   {
      if (LEDInt==1)
        {
          ……
         rPDATB=rPDATB&0x5ff;
          ….
          LEDInt=0;
        }
      if (KeyInt==1)
        {
  temp = *(keyboard_base+0xfd);
   ……
   if( (temp&0x1) == 0 )
   value = 3;
          ….
         KeyInt=0;
         }
         …….
     LED8ADDR = ~Seg[seg_num];
        …..
       LCDReadReg;
       LCDWriteReg;
        ……
}
```

在应用程序中直接读写寄存器

1. 单任务结构

单任务结构实现二

这是什么结构？结构图？



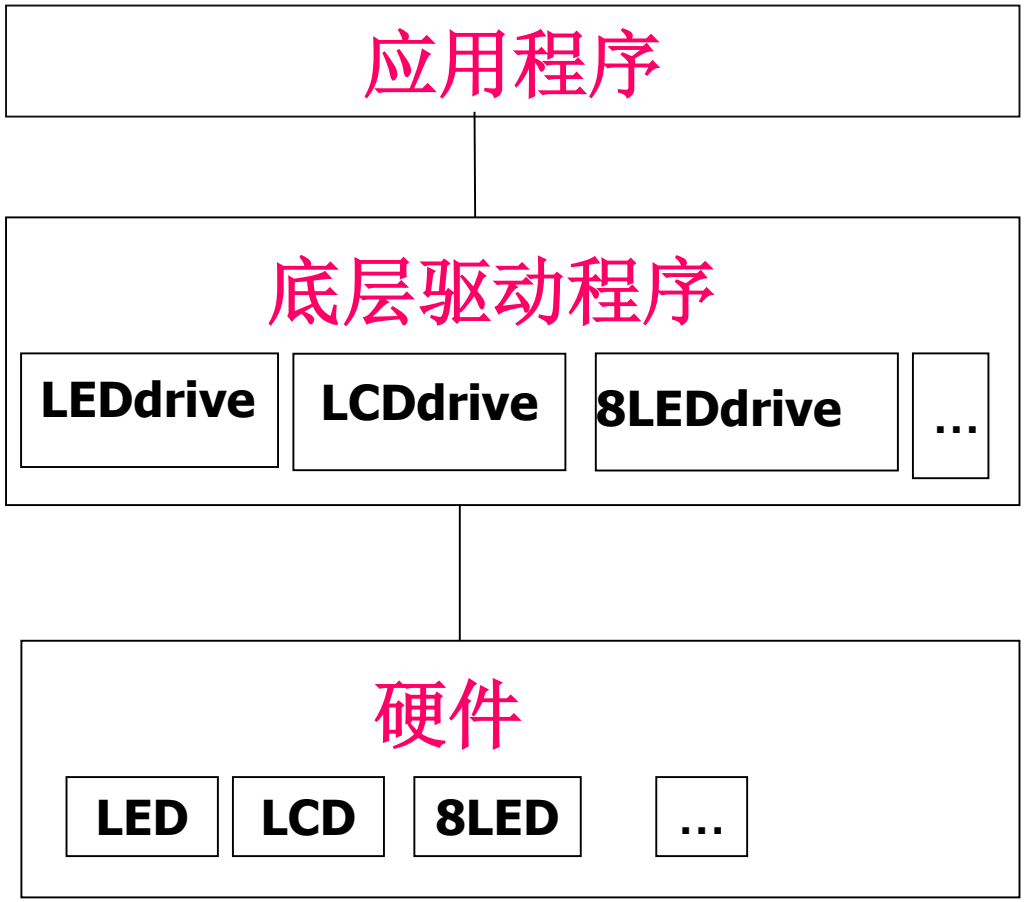
设备驱动和应用软件模块处于同一层。弊端：

不符合软件设计高内聚低耦合要求!!!

```
main()
{
    while(1)
   {
      if (LEDInt==1)
        {
          ……
          LEDdrive();//LED驱动
          ….
          LEDInt=0;
         }
      if (KeyInt==1)
        {
          ……
          Key_Read();//Key 驱动
          ….
          KeyInt=0;
        }
        …….
      8LEDdrive (); //8LED驱动
       …..
       LCDdrive ();//LCD驱动
        ……
   }
}
```

1. 单任务结构

单任务结构实现三




```
main()
{.....
    while(1)
   {
       if (LEDInt==1)
         {
函数1    ProcessLED();
         LEDInt=0;
         }
       if (KeyInt==1)
         {
函数2    ProcessKey();
         KeyInt=0;
         }
函数3Process8LED();
函数4 ProcessLCD ();
   }
}
```

1. 单任务结构

```
ProcessLED()
{
        …
        LEDdrive();  //LED驱动
        ….
}
```

(a) 让LED闪烁

```
ProcessKey()
{
        …
        ReadKey ()；  //Key驱动
        ….
}
```

(b) 读键

```
Process8LED()
{
        …
        8LEDdrive() ；//8LED驱动
        ….
}
```

(c) 8LED显示

```
ProcessLCD()
{
        ……
        LCDdrive()；  //LCD驱动
        ….
}
```

(d) LCD显示字符

事件处理函数代码结构

1. 单任务结构

```
void timer1_Int(void)
{
   保护现场指令
   …
   LEDInt=1;//设置中断标志
   …
   恢复现场、中断返回指令
}
```

(a) timer1中断处理程序

```
void Keyboard_Int(void)
{
   保护现场指令
   …
   KeyInt=1; //设置中断标志
   ….
   恢复现场、中断返回指令
}
```

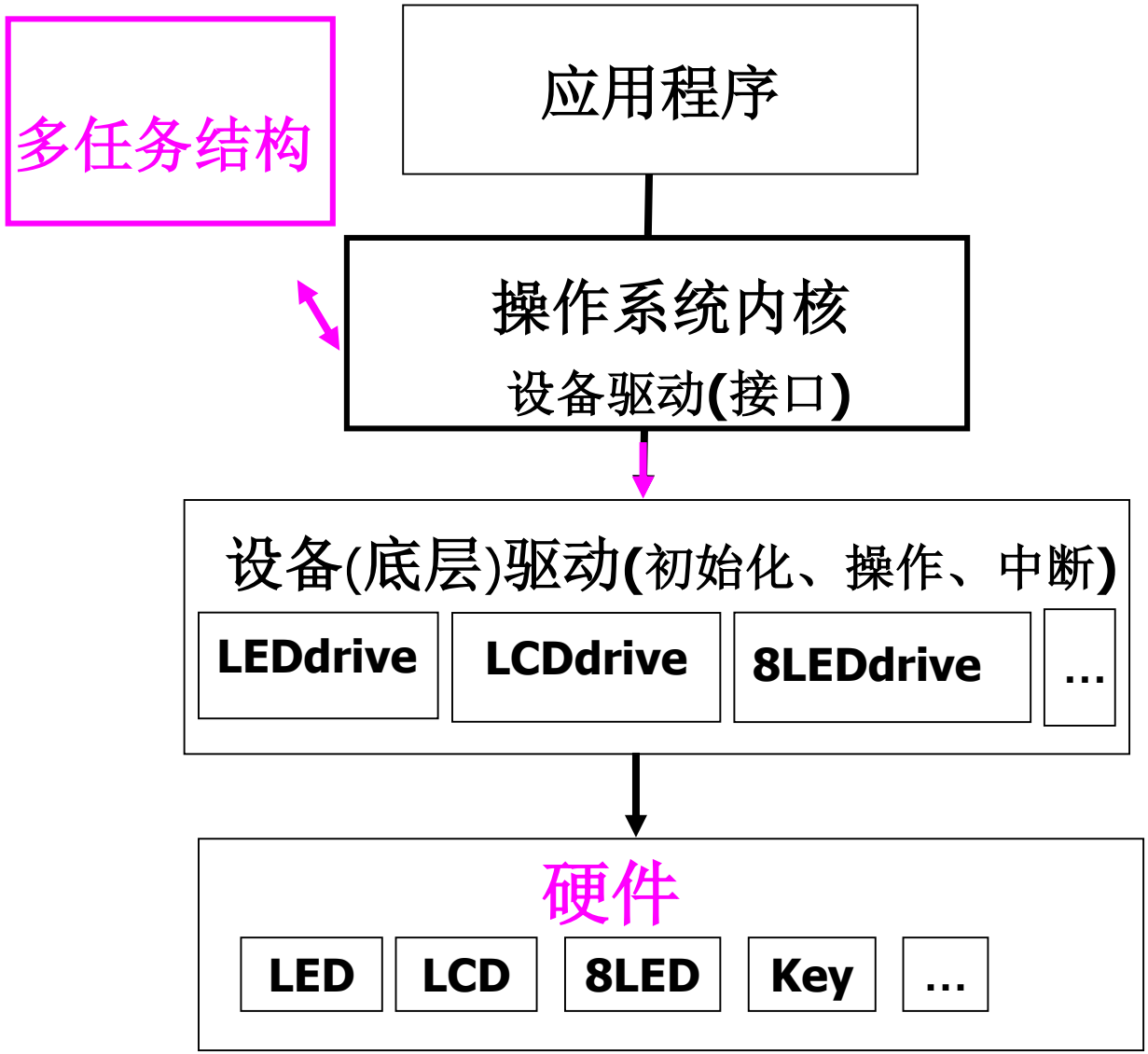
(b) 键盘中断处理程序

中断处理程序代码结构

嵌入式系统软件结构

1. 单任务结构（无核结构）
2. 多任务结构（有核结构）

2. 多任务结构


多任务结构
应用程序
操作系统内核
设备驱动(接口)
设备(底层)驱动(初始化、操作、中断)
LEDdrive
LCDdrive
8LEDdrive
…
硬件
LED
LCD
8LED
Key
…

2. 多任务结构

内核创建任务1：
ProcessLED()

```
ProcessLED()
{
…
LEDdrive();  //LED驱动
….
}
```

内核创建任务2：
ProcessKey()

```
ProcessKey()
{
…
Key_Read()；  //Key驱动
….
}
```

内核创建任务3：
Process8LED ()

```
Process8LED()
{
…
8LEDdrive()；  //8LED驱动
….
}
```

内核创建任务4：
ProcessLCD ()

```
ProcessLCD()
{
…
LCDdrive()；  //LCD驱动
….
}
```

2. 多任务结构

内核给每个任务分配一定的时间片，优先级

调度器按某种策略从就绪的任务中选择一个最值得运行的任务去使用CPU

当任务的时间片用完时必须交出CPU(即使要做的事还未完成）

微观上系统内只有一个任务在运行(使用CPU)，宏观上有多个任务在运行(处于不同态)

```
        if (LEDInt==1)          任务1就绪
          ….
任务1   ProcessLED();
        LEDInt=0;
        …..
        if (KeyInt==1)          任务2就绪
        ……
任务2   ProcessKey();
        KeyInt=0;
          ……
任务4   Process8LED ();         总是就绪
任务3   ProcessLCD ();          总是就绪
```

2. 多任务结构

```
void Keyboard_Int(void)
{
   …
 KeyInt=1;
  ….
}
```

```
ProcessKey()
{
…
Key_Read()；  //Key驱动
….
}
```

```
void timer1_Int(void)
{
 …
 LEDInt=1;
 …..
}
```

```
Process8LED()
{
…
8LEDdrive()；//8LED驱动
….
}
```

```
ProcessLED()
{
…
LEDdrive(); //LED驱动
….
}
```

```
ProcessLCD()
{
…
LCDdrive()；  //LCD驱动
….
}
```

Thank you