电子科技大学
University of Electronic Science and Technology of China

第一章 嵌入式系统概要

主讲人：漆强

电子科技大学 ytqiqiang@163.com

本章内容

- 嵌入式系统的概念及特点
- 嵌入式系统硬件
- 嵌入式系统软件
- 嵌入式系统的编程模式
- 微控制器的程序开发方式

教学目标

- 掌握嵌入式系统的概念和组成
- 了解嵌入式系统的编程模式及程序开发方式
- 了解硬件抽象层设计思想

1.1 嵌入式系统的概念及特点

基本概念

国外定义

Devices used to control,monitor,or assist the operation of equipment,machinery or plants.

用于控制、监视或者辅助操作机器或设备的装置。

国内定义

嵌入到对象体系中，以应用为中心，以计算机技术为基础，软硬件可裁剪，适应应用系统对功能、可靠性、成本、体积、功耗等严格要求的专用计算机系统。

基本特点

嵌入性

嵌入到对象系统，满足对象系统的要求，如物理环境、电气环境、成本

专用性

量身定做，专用开发环境，专用开发工具

计算机系统

以计算机技术基础，光、机、电、算、软一体化，多学科融合

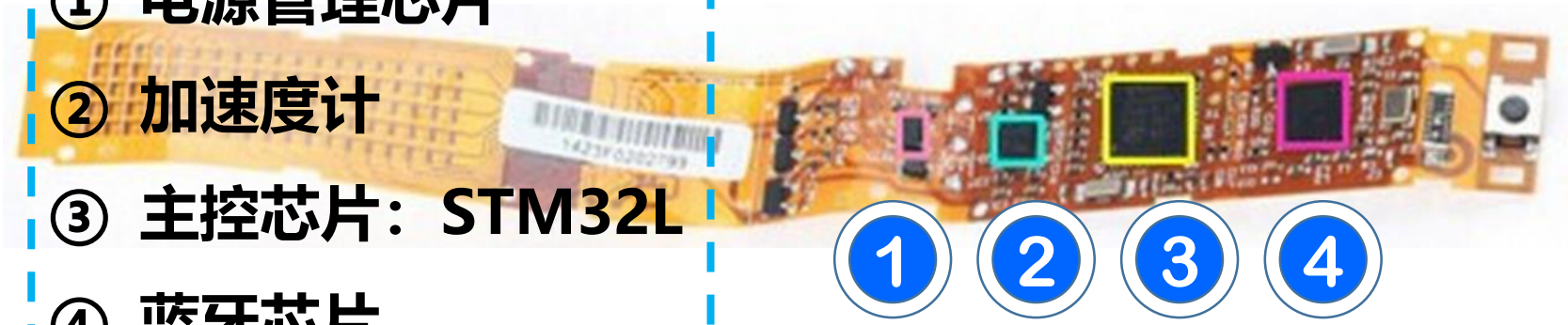
咕咚手环

① 电源管理芯片

② 加速度计

③ 主控芯片：STM32L

④ 蓝牙芯片



应用演变

传统应用： MCU 作为主控

监控

电动助力车

扩展坞

仪器仪表

汽车音响

家电

2007 — 2013

新应用: MCU + RF + Sense + Algorithm

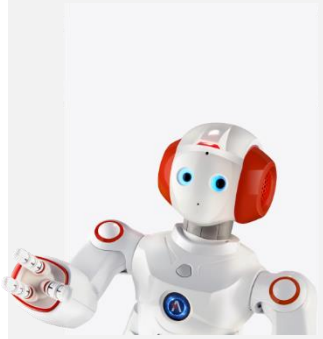
穿戴式

飞行器

AI

电动代步车

机器人

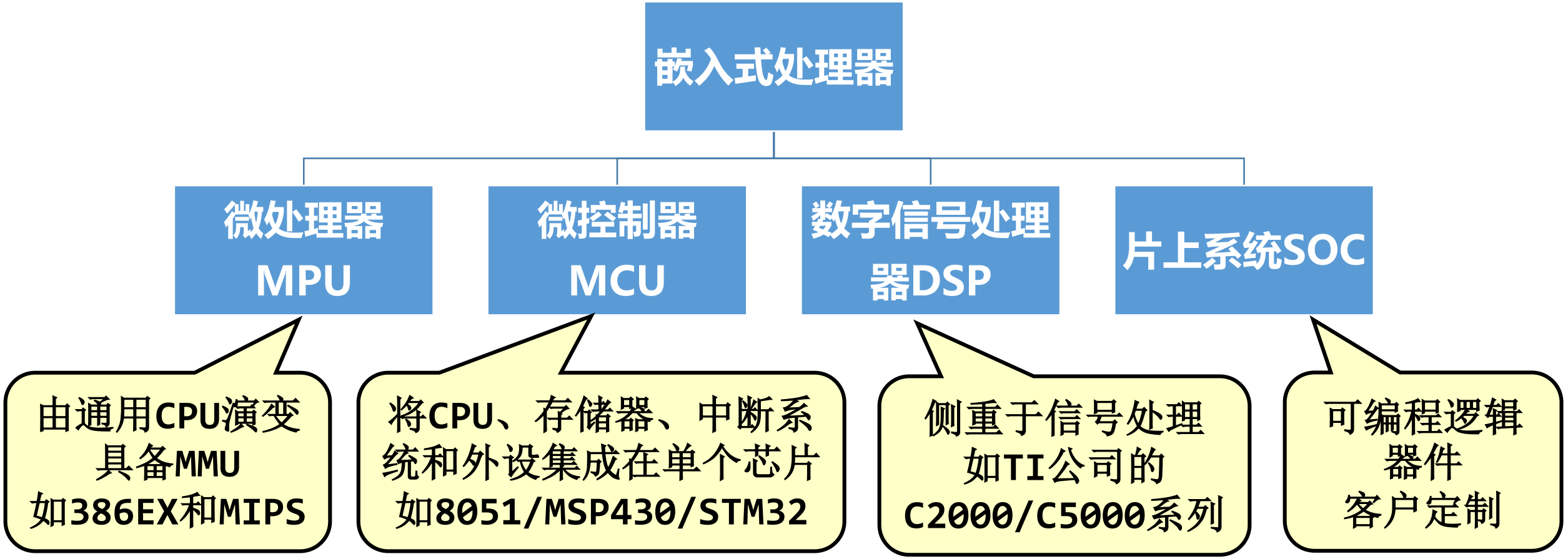


IoT

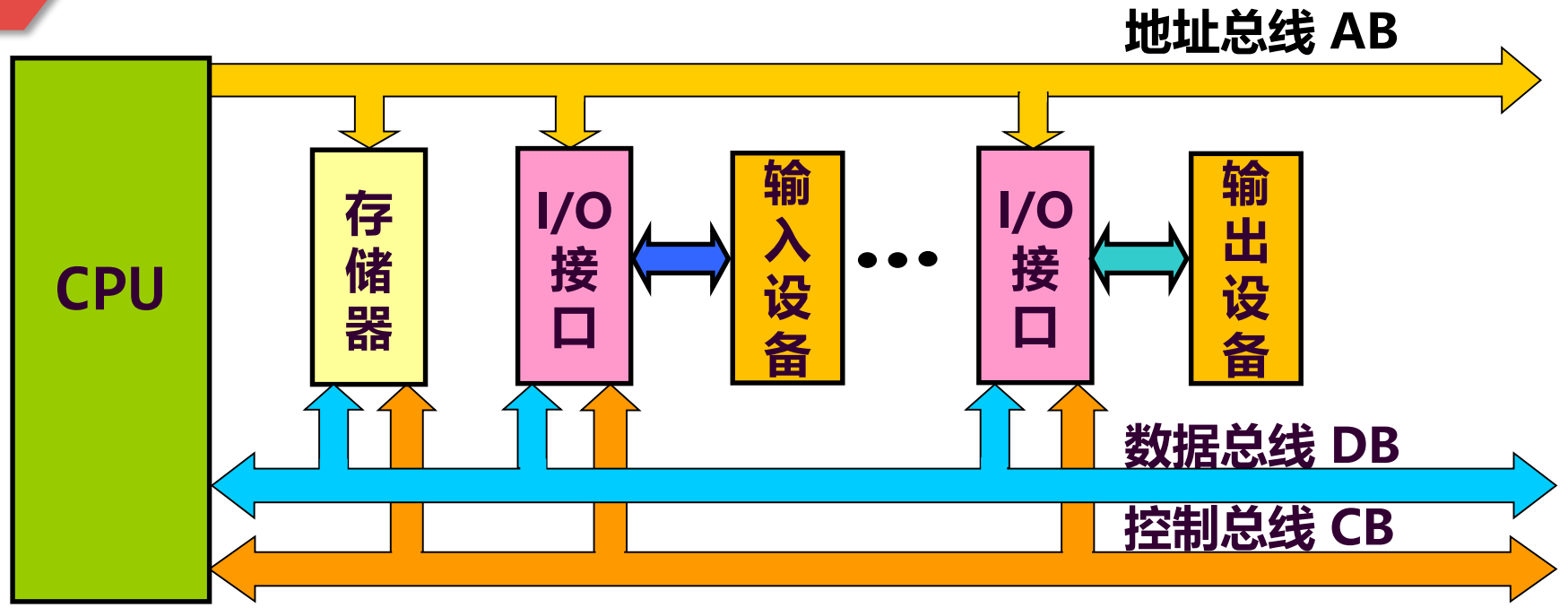
2014 — 2019

1.2 嵌入式系统硬件

嵌入式处理器


嵌入式处理器
微处理器
MPU
微控制器
MCU
数字信号处理器DSP
片上系统SOC
由通用CPU演变
具备MMU
如386EX和MIPS
将CPU、存储器、中断系统和外设集成在单个芯片
如8051/MSP430/STM32
侧重于信号处理
如TI公司的
C2000/C5000系列
可编程逻辑器件
客户定制

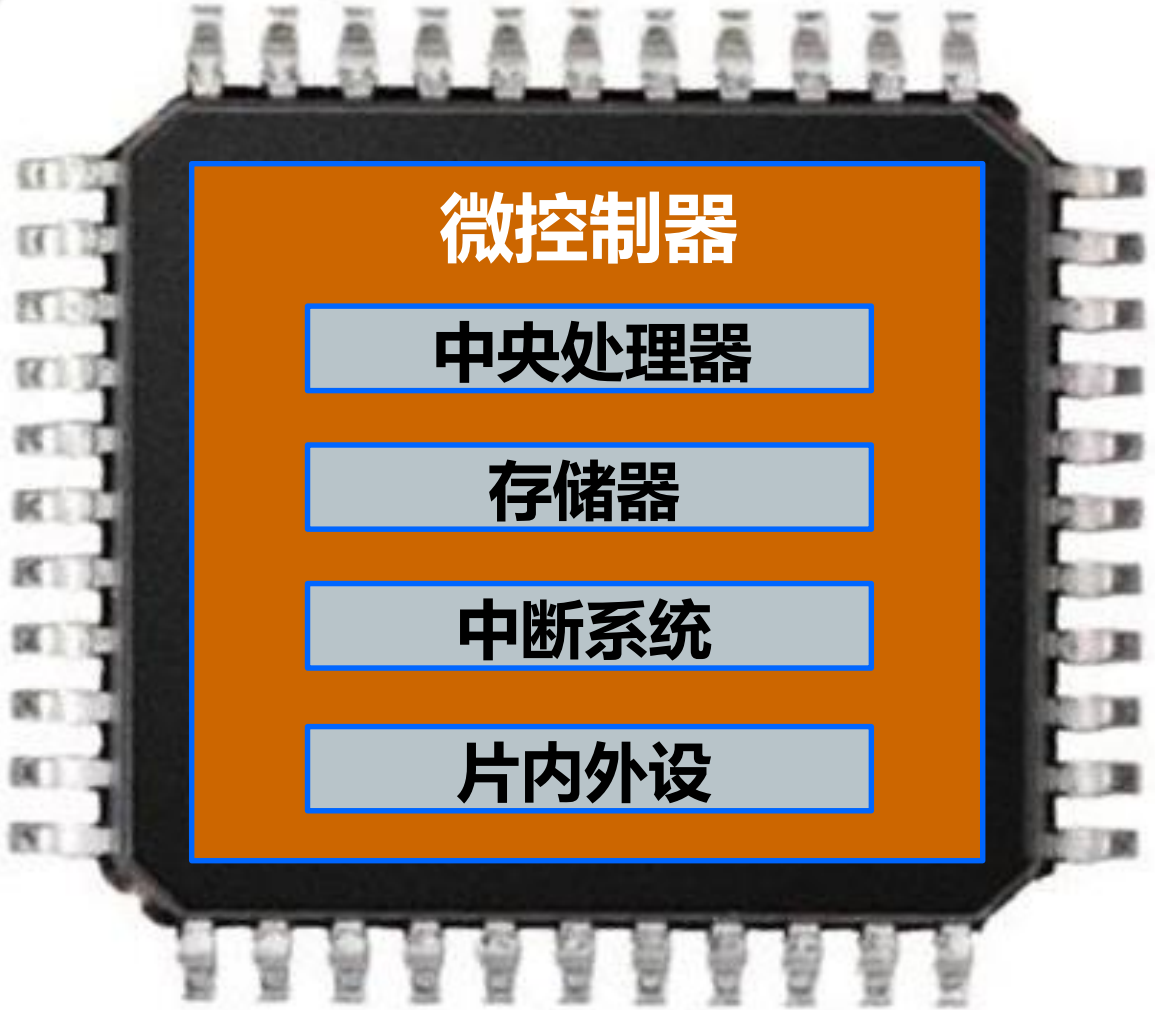
计算机硬件



构成部件

- 微机的硬件由CPU、存储器、输入/输出设备构成
- 输入/输出设备通过输入/输出接口与系统相连
- 各部件通过总线连接

微控制器



微控制器

↕

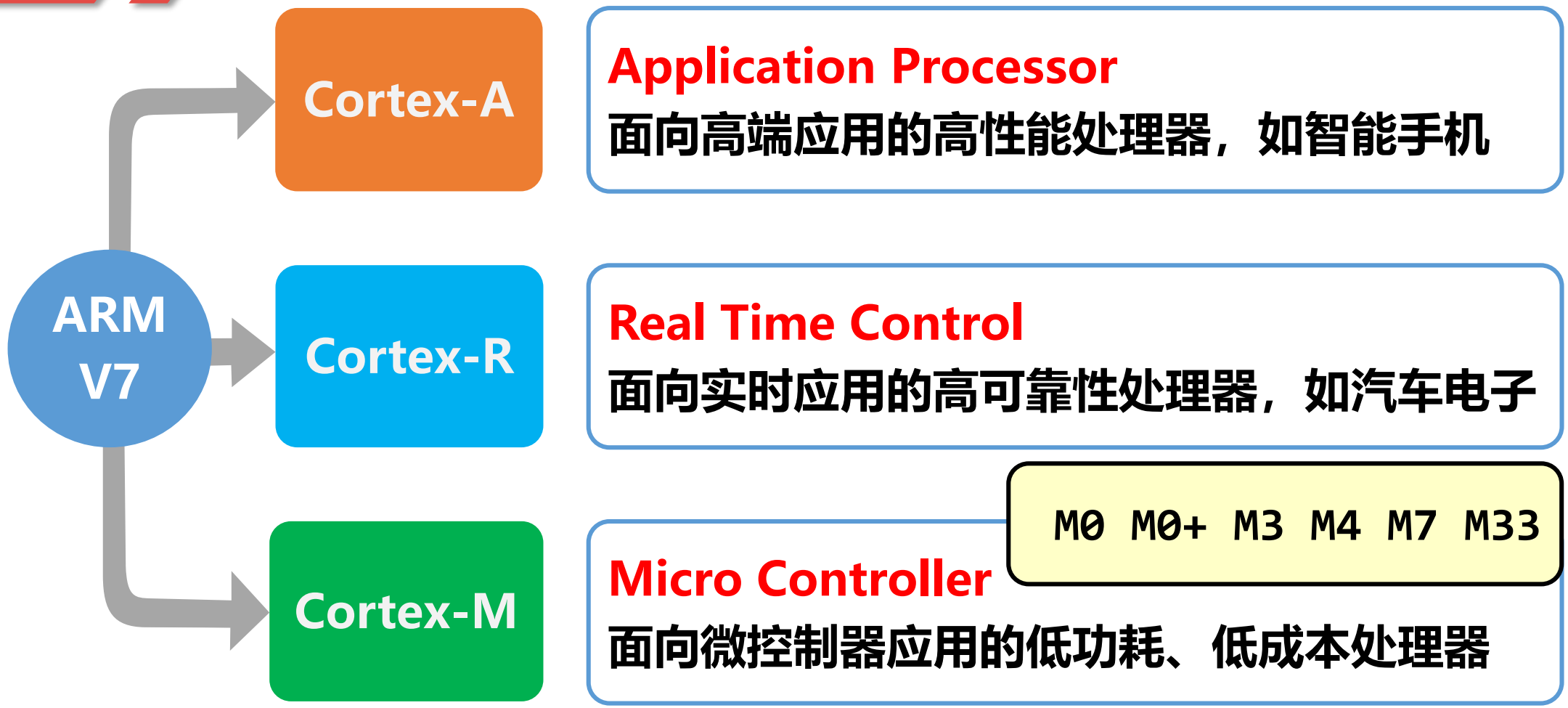
计算机主机

ARM处理器

- ARM是Advanced RISC Machines的缩写
- ARM公司的特点是只设计芯片，而不生产芯片
- 它将技术授权给世界上许多著名的半导体厂商

ARM → 将技术授权给其它芯片厂商 → ST / NXP / HUAWEI → 形成各具特色的ARM内核芯片

内核架构


ARM
V7
Cortex-A
Application Processor
面向高端应用的高性能处理器，如智能手机
Cortex-R
Real Time Control
面向实时应用的高可靠性处理器，如汽车电子
Cortex-M
M0 M0+ M3 M4 M7 M33
Micro Controller
面向微控制器应用的低功耗、低成本处理器

外围电路

温度/湿度传感器
光电类传感器
压力传感器

1 环境感知

2 接口

A/D接口
同步/异步串口
USB接口

SRAM
DRAM
NAND/Nor Flash

3 存储

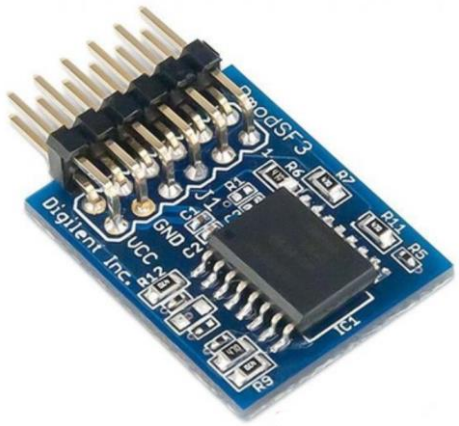
4 人机交互

指示灯/数码管
液晶显示屏
键盘

实物图片


8:8:8


GND VCC DO D1 RST DC CS
08:08
2013.07.01
菜单
QQ

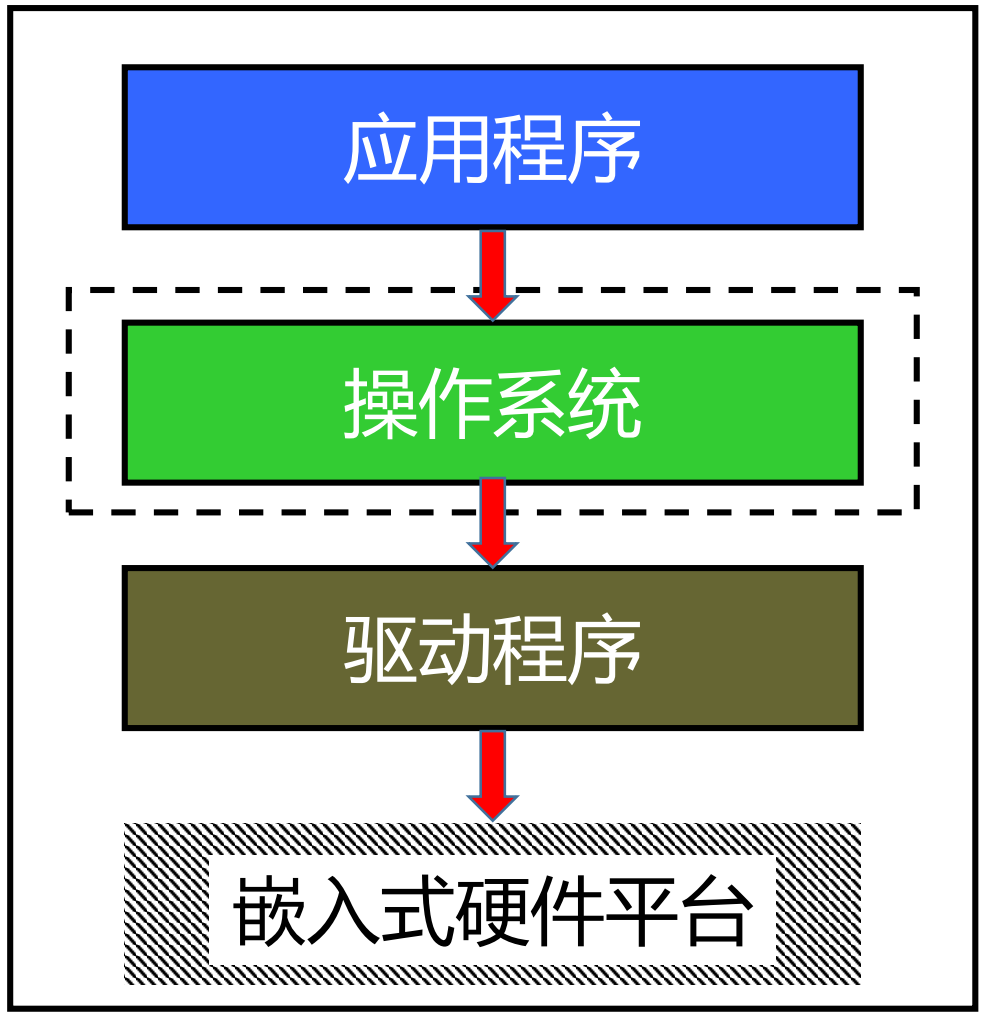

PmodSF3
Digilent Inc.


HR911105A
W5500模块

电子科技大学
University of Electronic Science and Technology of China

1.3 嵌入式系统软件

传统软件框架

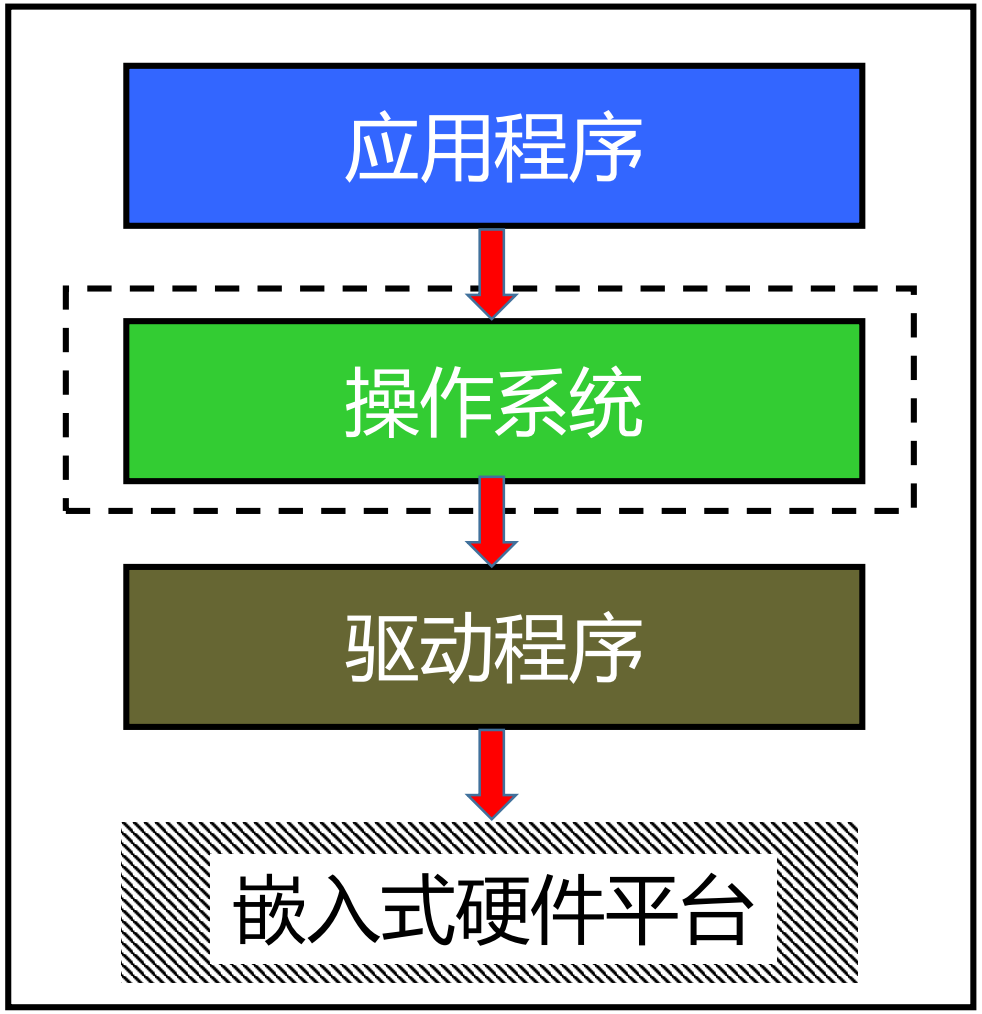


完成具体业务逻辑
根据需求编写

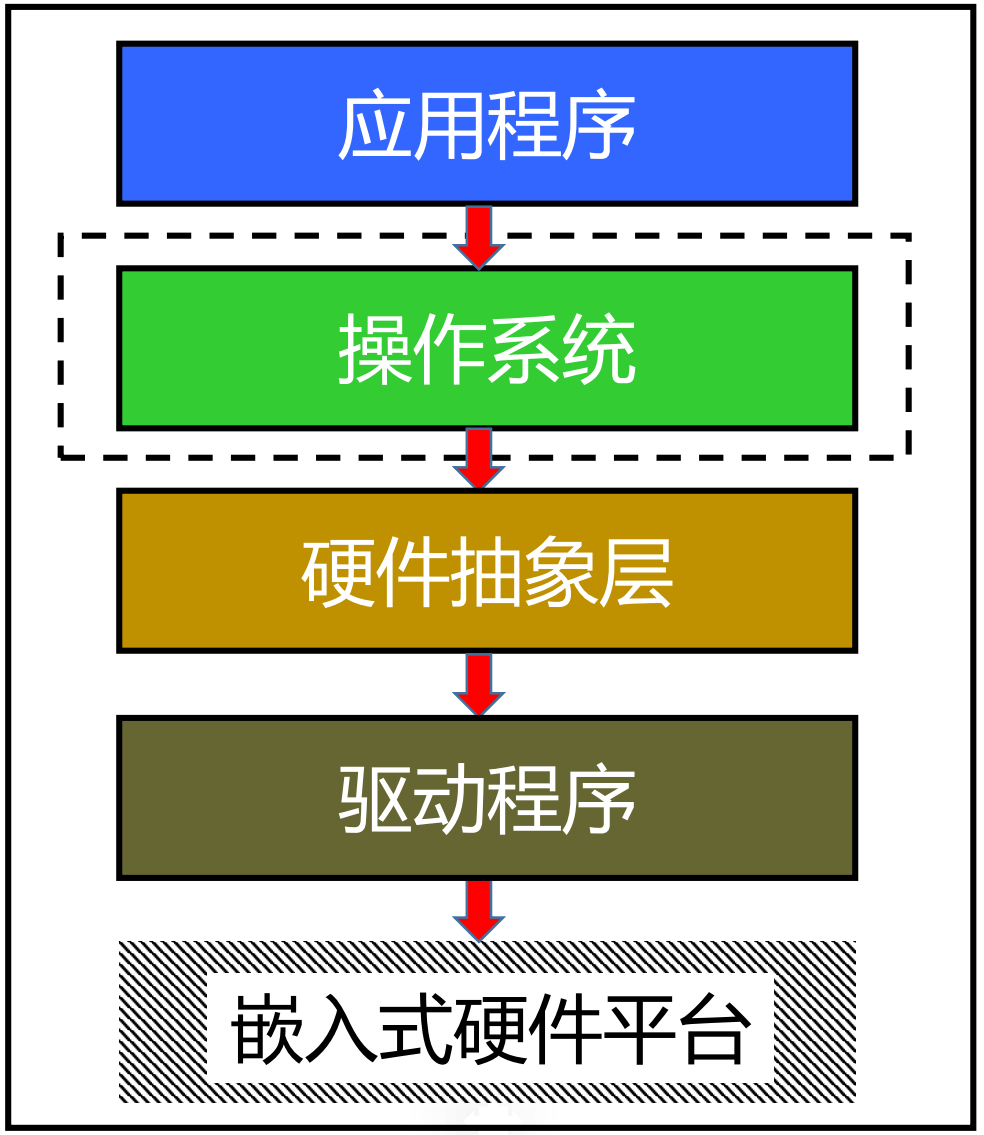
管理系统所有资源
提供接口函数

控制硬件模块
片内外设和外围电路
寄存器或库函数

演进软件框架



加入HAL层



目的和意义

减少嵌入式软件移植的工作量和难度，提高嵌入式软件的通用性和复用性。

引入中间层，屏蔽了底层硬件的多样性，应用程序和操作系统不再面对具体的硬件环境，而是面对由这个中间层所代表的、逻辑上的硬件环境。

应用程序和操作系统调用硬件抽象层提供的接口函数。只要接口函数能够在下层硬件平台上实现，那么操作系统和应用程序的代码就可以无缝移植。

典型

Arduino　Mbed　MicroPython

Mbed

Mbed是面向ARM处理器的原型开发平台

软件库

硬件抽象层，屏蔽了不同MCU厂商提供的微控制器之间的差异

硬件设计参考

提供硬件参考设计，统一程序下载接口，单步调试接口和串行调试接口

在线开发环境

基于浏览器的微控制器集成开发环境，包括代码编写和程序编译等功能

板级支持包

板级支持包BSP是硬件抽象层HAL的一种实现形式

应用层

操作系统

板级支持包

寄存器或固件库

调用BSP提供的
接口函数 API

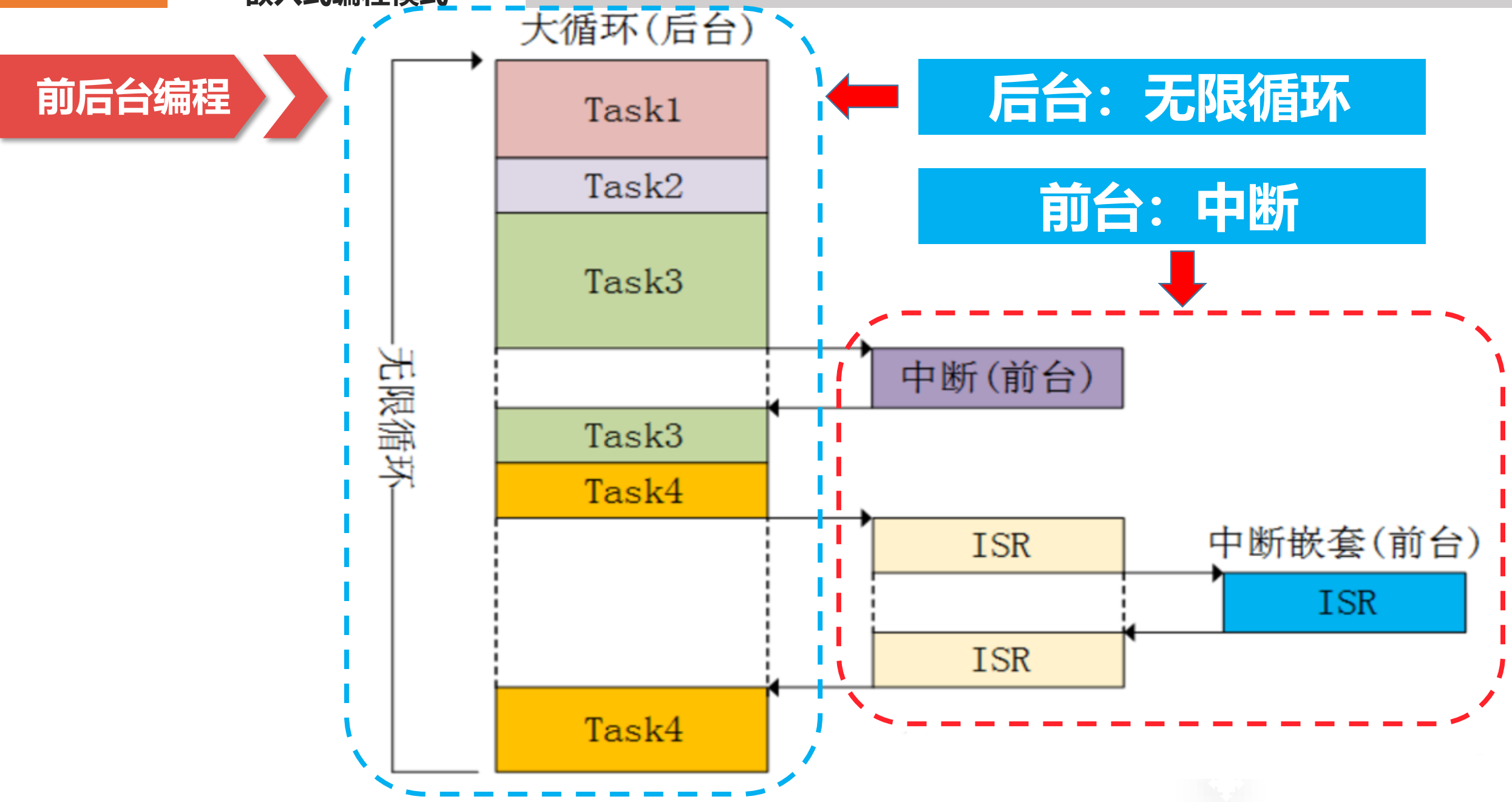
调用BSP提供的
接口函数API

BSP_LED_Init
BSP_LED_On
BSP_LED_Off

硬件平台
8051/MSP430/STM32

1.4 嵌入式系统编程模式

前后台编程


大循环(后台)
Task1
Task2
Task3
Task3
Task4
Task4
无限循环
后台：无限循环
前台：中断
中断(前台)
ISR
中断嵌套(前台)
ISR
ISR

编程实例

温度采集系统设计

1. 温度的采集，每隔1s；
2. 温度显示在数码管上；
3. 可以通过键盘设置温度上/下限；
4. 温度超过上/下限，声光报警

用户编程

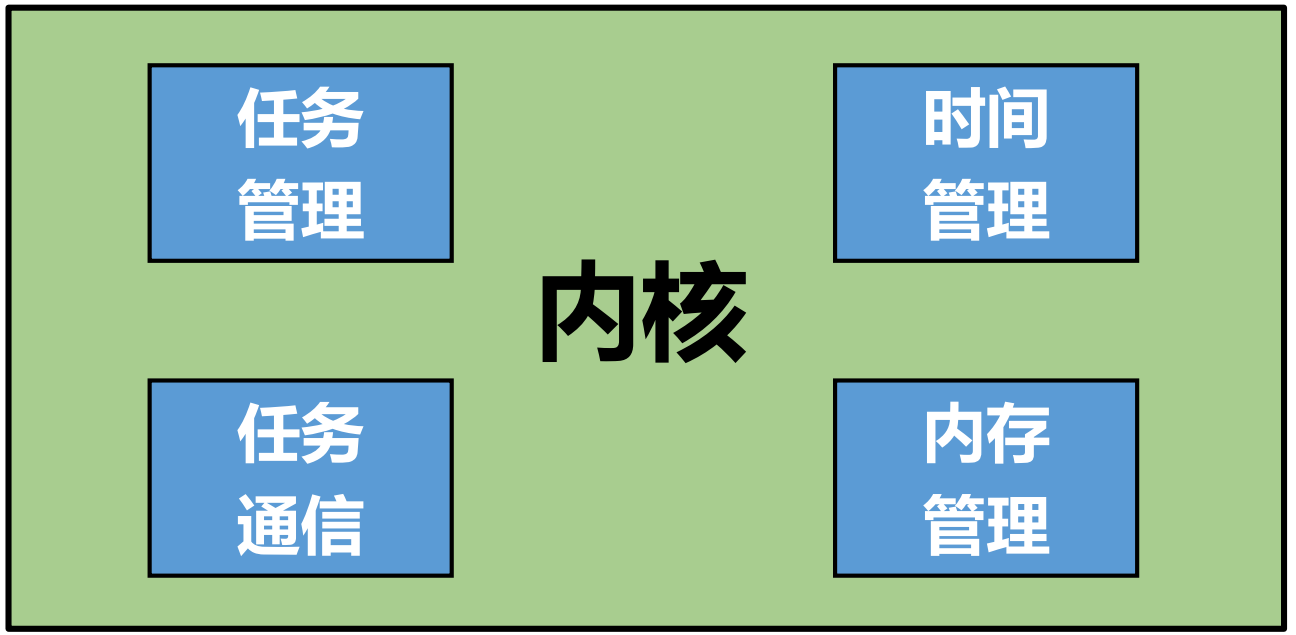
- 编写定时中断程序
- 定义标志变量
- 判断标志变量并清除

```
int main(void)
{
    System Initialize;
    while(1)
    {
        if( Flag1s     == 1 )
          GetTemperatrue();
        if( Flag20ms   == 1 )
          DispTemperatrue();
        if( Flag10ms   == 1 )
          ReadKey();
        if( AlarmFlag  == 1 )
          SendAlarm();
    }
}
```

由定时器中断产生标志

由温度采集程序产生标志

嵌入式操作系统



用户编程

- 任务划分
- 调用操作系统的接口函数
- 编写应用程序

任务定义

一个具有独立功能的无限循环的程序段的一次运行活动

采用嵌入式操作系统实现温度采集系统

温度采集任务

```
void Task_GetTemperatrue(void)
{
    while(1)
    {
        GetTemperatrue();
        SendEventFlag();
        OsDelay(1000);
     }
}
```

温度显示任务

```
void Task_DispTemperatrue(void)
{
    while(1)
    {
        DispTemperatrue();
        OsDelay(20);
     }
}
```

使用操作系统提供的接口函数，用户不用编写定时器中断程序和定义标志变量

读取按键任务

```
void Task_ReadKey(void)
{
    while(1)
    {
        ReadKey();
        OsDelay(10);
     }
}
```

温度报警任务

```
void Task_SendAlarm(void)
{
    while(1)
    {
        WaitEventFlag();
        SendAlarm();
    }
}
```

FreeRTOS

01 实时内核，提供完整功能

02 免费、开源

03 稳定性强，市场占有率高

04 开发文档完善，有大量实例可供参考

µC/OS-II

01 实时操作系统

02 收费、闭源

03 稳定性强

04 开发文档完善，有大量实例可供参考

RTX

01 实时内核

02 免费、开源

03 偏向于物联网应用

04 开发文档较少

RTX 封装为 CMSIS-RTOS

CMSIS-RTOS

ARM推出的嵌入式操作系统的统一接口，便于用户在不同操作系统之间的程序移植

RT-Thread

01 实时操作系统

02 免费、开源

03 偏向于物联网应用

04 国产系统，开发文档丰富，社区活跃

1.5 微控制器的程序开发方式

寄存器开发

以STM32微控制器为例

优点

- 从细节上更加清晰地了解和掌握STM32的架构、原理
- 程序代码简练、短小，执行效率高

缺点

- STM32的外设多，每个外设对应多个寄存器，需要用户了解每个寄存器的功能以及寄存器中每一个位的定义
- 程序后期维护、移植会相对困难

寄存器开发

开发要求

- 熟悉所使用外设的初始化设置流程、读写方法
- 熟悉所涉及寄存器的功能以及每个寄存器位的定义与作用

读写操作

- 利用赋值语句设置或读取相关寄存器的值

固件库开发

优点

- 对硬件的理解要求相对较低，会调用函数就会写程序，容易上手
- 程序代码容错性好，后期维护相对简单

缺点

- 程序冗余较多，代码量较大，运行速度相对会有所影响

固件库开发

开发要求

- 学习STM32的固件库官方手册和范例程序
- 掌握固件库接口函数的功能与调用方法

读写操作

- 通过函数调用来设置或读取相关寄存器的值

两种方式比较

设置引脚PA5输出高电平

寄存器开发

- GPIOA->BSRR |= 0x00000020;

PC13

固件库开发

- HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);

```
HAL_GPIO_TogglePin(GPIOC,GPIO_PIN13,GPIO_PING_SET)
```

本章结束