嵌入式实践与基础手札（1）

## 1.1.1、嵌入式系统的定义

嵌入式系统是一种计算机硬件和软件的组合，也许还有机械装置，用于实现一个特定功能。在某些特定情况下，嵌入式系统是一个大系统或产品的一部分

## 1.1.2嵌入式系统的由来及发展简史

通俗地说，计算机是因科学家需要一个高速的计算工具而产生的。为了区分两种计算机类型，通常把满足海量高速数值计算的计算机称为通用计算机系统，而把嵌入到实际应用系统中， 实现嵌入式应用的计算机称为嵌入式计算机系统，简称嵌入式系统。可以说，是因为通信、测控与数据传输等领域对计算机技术的需求催生了嵌入式系统的产生。

## 1.1.3 嵌入式系统的分类

从嵌入式系统的学习角度来看，因应用于不同领域的嵌入式系统，其知识要素与学习方法有所不同，所以可以按应用范围简单地把嵌入式系统分为电子系统智能化（微控制器类）和计算机应用延伸（应用处理器）这两大类。一般来说，微控制器与应用处理器的主要区别在于可靠性、数据处理盘、工作频率等方面，相对应用处理器来说，微控制器的可靠性要求更高、数据处理批较小、工作频率较低。

1.电子系统智能化类

电子系统智能化类的嵌入式系统，主要用于工业控制、现代衣业、家用电器、汽车电子、测控系统、数据采集等，这类应用所使用的嵌入式处理器一般被称为微控制器(Microcontroller Unit, MCU) 。这类嵌入式系统产品， 从形态上看，更类似千早期的电子系统，但内部计算程序起核心控制作用。

从学习与开发角度，电子系统智能化类的嵌入式应用，需要终端产品开发者面向应用对象设计硬件、软件，注重软件、硬件协同开发。因此，开发者必须掌握底层硬件接口、底层驱动及软硬件密切结合的开发调试技能。电子系统智能化类的嵌入式系统，即微控制器，是嵌入式系统的软硬件基础， 是学习嵌入式系统的入门环节，且为重要的一环。

2.计算机应用延伸类

计算机应用延伸类的嵌入式系统，主要用千平板电脑、智能手机、电视机顶盒、企业网络设备等，这类应用所使用的嵌入式处理器一般被称为应用处理器(Application Processor) 。这类嵌入式系统产品，从形态上看，更接近通用计算机系统。开发方式上，也类似于通用计算机的软件开发方式。从学习与开发角度，计算机应用延伸类的嵌入式应用，终端产品开发者大多购买厂家制作好的硬件实体在嵌入式操作系统下进行软件开发，或者还需要掌握少量的对外接口方式。因此，从知识结构角度，学习这类嵌入式系统，对硬件的要求相对较少。计算机应用延伸类的嵌入式系统， 即应用处理器， 也是嵌入式系统学习中重要的一环。但是，从学习规律角度看，若是要全面学习掌握嵌入式系统，应该先学习掌握微控制器，然后在此基础上，进一步学习掌握应用处理器编程，而不要倒过来学习。

## 1.1.4 嵌入式系统的特点

1. 嵌入式系统属于计算机系统，但不单独以通用计算机的面目出现

嵌入式系统的本名叫嵌入式计算机系统(Embedded Computer System), 它不仅具有通

用计算机的主要特点，又具有自身特点。嵌入式系统也必须要有软件才能运行，但其隐含在种类众多的具体产品中。同时，通用计算机种类屈指可数，而嵌入式系统不仅芯片种类繁多，而且由于应用对象大小各异，嵌入式系统作为控制核心，已经融入到各个行业的产品之中。

2. 嵌入式系统开发需要专用工具和特殊方法嵌入式系统不像通用计算机那样有了计算机系统就可以进行应用软件的开发。一般情况下，微控制器或应用处理器芯片本身不具备开发功能，必须要有一套与相应芯片配套的开发工具和开发环境。这些工具和环境一般基于通用计算机上的软硬件设备以及逻辑分析仪、示波器等。开发过程中往往有工具机（ 一般为PC 或笔记本）和目标机（实际产品所使用的芯片）之分，工具机用于程序的开发，目标机作为程序的执行机，开发时需要交替结合进行。编辑、编译、链接生成机器码在工具机完成，通过写入调试器将机器码下载到目标机中，进行运行与调试。

3. 使用MCU 设计嵌入式系统，数据与程序空间采用不同存储介质在通用计算机系统中， 程序存储在硬盘上。实际运行时，通过操作系统将要运行的程序从硬盘调入内存(RAM), 运行中的程序、常数、变最均在RAM 中。而在以MCU 为核心的嵌入式系统中， 一般情况下，其程序被固化到非易失性存储器中气变量及堆栈使用RAM存储器。

4. 开发嵌入式系统涉及软件、硬件及应用领域的知识嵌入式系统与硬件紧密相关，嵌入式系统的开发需要硬件、软件协同设计、协同测试。同时，由于嵌入式系统专用性很强，通常是用在特定应用领域，如嵌入在手机、冰箱、空调、各种机械设备、智能仪器仪表中起核心控制作用，功能专用，因此，进行嵌入式系统的开发，还需要对领域知识有一定的理解。

# 1.2 嵌入式系统的学习困惑，知识体系和学习建议

## 1.2.1 嵌入式系统的学习困惑

## 1.2.2 嵌入式系统的知识体系

从由浅入深、由简到繁的学习规律来说，嵌入式学习的入门应该选择微控制器，而不是应用处理器，应通过对微控制器基本原理与应用的学习，逐步掌握嵌入式系统的软件与硬件基础，然后在此基础上进行嵌入式系统其他方面知识的学习。

概括地说，学习以MCU 为核心的嵌入式系统，需要以下软件硬件基础知识与实践训

练，即以MCU 为核心嵌入式系统的基本知识体系如下觅

(1) 掌握硬件最小系统与软件最小系统框架。硬件最小系统是包括电源、晶振、复位、写入调试器接口等可使内部程序得以运行的、规范的、可复用的核心构件系统气软件最小系统框架是一个能够点亮一个发光二极管的，甚至带有串口调试构件的，包含工程规范完整要素的可移植与可复用的工程模板

(2) 掌握常用基本输出的概念、知识要素、构件使用方法及构件设计方法。如通用I/0

CG PIO) 、模数转换AD 、数模转换DA 、定时器模块等。

(3) 掌握若干嵌入式通信的概念、知识要素、构件使用方法及构件设计方法。如串行通信接口UART 、串行外设接口SPI , 集成电路互连总线IZC 、CAN 、USB 、嵌入式以太网、无线射频通信等。

(4) 掌握常用应用模块的构件设计方法及使用方法及数据处理方法。如显示模块

(LED 、LCD 、触摸屏等）、控制模块（控制各种设备，包括PWM 等控制技术）等。数据处理如图形、图像、语音、视频等处理或识别等。

(5) 掌握一门实时操作系统RTOS 的基本用法与基本原理。

(6) 掌握嵌入式软硬件的基本调试方法。如断点调试、打桩调试、printf 调试方法等。

在嵌入式调试过程中，特别要注意确保在正确硬件环境下调试未知软件，在正确软件环境下调试未知硬件。

## 1.2.3 基础阶段的学习建议

(1) 遵循“先易后难，由浅入深＂的原则，打好软硬件基础。跟随本书，充分利用本书提

供的软硬件资源及辅助视频材料，逐步实验与实践矶充分理解硬件基本原理、掌握功能模

块的知识要素、掌握底层驱动构件的使用方法、掌握一两个底层驱动构件的设计过程与方

法；熟练掌握在底层驱动构件基础上，利用C 语言编程实践。理解学习嵌入式系统， 必须勤于实践。

( 2) 充分理解知识要素、掌握底层驱动构件的使用方法。本书对诸如GPIO 、UART 、定时器、PWM 、AD 、DA 、Flash 在线编程、USB 等模块，首先阐述其通用知识要素，随后给出其底层驱动构件的基本内容。期望读者在充分理解通用知识要素基础上，学会底层驱动构件使用方法。

(3) 基本掌握底层驱动构件的设计方法。

(4)掌握单步跟踪调试、打桩调试、printf 输出调试等调试手段。

(5) 日积月累，勤学好问，充分利用本书及相关资源。

# 1.3 微控制器和应用处理器简介

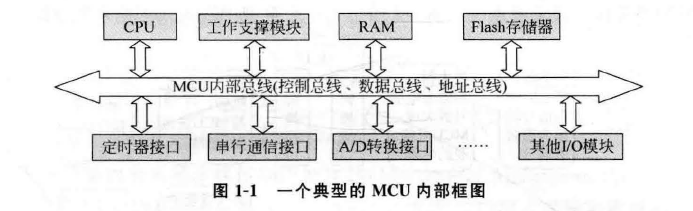
## 1.3.1 微控制器简介

1、微控制器的基本含义

MCU 是单片微观计算机（单片机）的简称，早期的英文名是Sing l e-chi p

Microcomputer, 后来大多数称之为微控制器(Mi crocon troll er) 或嵌入式计算机(EmbeddedComp uter) 。

MCU的基本含义是：在一块芯片内集成了中央处理单元( Central Processing Unit, CPU存储器(RAM/ROM 等）、定时器／计数器及多种输入输出(1/0) 接口的比较完整的数字处理系统。



MCU 是在计算机制造技术发展到一定阶段的背景下出现的，它使计算机技术从科学计算领域进入到智能化控制领域。从此，计算机技术在两个重要领域一通用计算机领域和嵌人式(Emb edded) 计算机领域都获得了极其重要的发展，为计算机的应用开辟了更广阔的空间。

2. 嵌入式系统与MCU 的关系

大部分嵌入式系统以MCU 为核心进行设计。MCU 从体系结构到指令系统都是按照嵌入式系统的应用特点专门设计的，它能很好地满足应用系统的嵌入、面向测控对象、现场可靠运行等方面的要求。因此以MCU 为核心的系统是应用最广的嵌入式系统

## 1.3.2 以MCU 为核心的嵌入式测控产品的基本组成

一个以M<::U 为核心，比较复杂的嵌入式产品或实际嵌入式应用系统，包含模拟量的输入、模拟量的输出、开关量的输入、开关量的输出及数据通信的部分

1. MCU 工作支撑电路

MCU 工作支撑电路也就是MCU 硬件最小系统，它保障MCU 能正常运行，如电源电

路、晶振电路及必要的滤波电路等， 甚至可包含程序写入器接口电路。

2. 模拟信号检入电路

实际模拟信号一般来自相应的传感器

3. 开关量信号输入电路

实际开关信号一般也来自相应的开关类传感器，如光电开关、电磁开关、干簧管（磁开

关）、声控开关、红外开关等， 一些儿童电子玩具中就有一些类似的开关。手动开关也可作为开关信号送到MCU 中。对MCU 来说，开关信号就是只有"O" 和"1" 两种可能值的数字

信号。

4. 其他输入信号或通信电路

其他输入信号通过某些通信方式与MCU 沟通。常用的通信方式有异步串行(UART)通信、串行外设接口(SPD 通信、并行通信、USB 通信、网络通信等。

5. 捈出执行机构电路

在执行机构中，有开关量执行机构，也有模拟量执行机构。开关量执行机构只有“开““关”两种状态。模拟量执行机构需要连续变化的模拟最控制。MCU 一般不能直接控制这些执行机构，需要通过相应的隔离和驱动电路实现。还有一些执行机构，既不是通常开关量控制，也不是通常DA 转换量控制，而是“脉冲“最控制，如控制调频电动机， MCU 则通过软件对其控制。

## 1.3.3 应用处理器简介

应用处理器的全名是多媒体应用处理器(Multimedia Application Processor, MAP) 。它是在低功耗CPU 的基础上扩展音视频功能和专用接口的超大规模集成电路。与MCU相比， MAP 的最主要特点是：工作频率高；硬件设计更为复杂；软件开发需要选用一个嵌入式操作系统；计算功能更强；抗干扰性能较弱；较少直接应用千控制目标对象；此外，一般情况下， MAP 芯片价格也高千MCU 。

# 1.4 嵌入式系统常用术语

1、封装

集成电路的封装( Package) 是指用塑料、金属或陶瓷材料等把集成电路封在其中。封装可以保护芯片， 并使芯片与外部世界连接。常用的封装形式可分为通孔封装和贴片封装两大类。

2. 印刷电路板

印刷电路板(Printed C汀cu it Board, PCB) 是组装电子元件用的基板，是在通用基材上按

预定设计形成点间连接及印制元件的印制板，是电路原理图的实物化。PCB 的主要功能是

提供集成电路等各种电子元器件固定、装配的机械支撑；实现集成电路等各种电子元器件

之间的布线和电气连接（信号传输）或电绝缘；为自动装配提供阻焊图形，为元器件插装、检查、维修提供识别字符和图形等。

3. 动态可读写随机存储器与静态可读写随机存储器

动态可读写随机存储器(Dynamic Random Access Memory, DRAM), 由一个MOS 管组成一个二进制存储位。MOS 管的放电导致表示"1" 的电压会慢慢降低。一般每隔一段时间就要控制刷新信息，给其充电。DRAM 价格低，但控制烦琐，接口复杂。

4.只读存储器

只读存储器(Read Only Memory,ROM), 数据可以读出，但不可以修改，所以称为只读存储器。通常存储一些固定不变的信息，如常数、数据、换码表、程序等。它具有断电后数据不丢失的特点。ROM 有固定ROM 、可编程ROM( 即PROM) 和可擦除ROM( 即EPROM)三种。

5.闪速存储器

闪速存储器(Fl as h M emory) 简称闪存，是一种新型快速的E2 PROM 。由于工艺和结构上的改进，闪存比普通的E2 PROM 的擦除速度更快，集成度更高。闪存相对千传统的E2 PROM 来说，其最大的优点是系统内编程，也就是说不需要另外的器件来修改内容

6.模拟量和开关量

模拟最是指时间连续、数值也连续的物理量，如温度、压力、流量、速度、声音等。在工程技术上，为了便千分析，常用传感器、变换器将模拟蜇转换为电流、电压或电阻等电学量。开关量是指一种二值信号，用两个电平（高电平和低电平）分别来表示两个逻辑值（逻辑1 和逻辑0) 。

## 1.4.2 与通信相关的术语

1. 并行通信

并行通信是指数据的各位同时在多根并行数据线上进行传输的通信方式，数据的各位同时由源到达目的地。适合近距离、高速通信。常用的有4 位、8 位、16 位、32 位等同时传输

2. 串行通信

串行通信是指数据在单线（电平高低表征信号）或双线（差分信号）上，按时间先后一位一位地传送，其优点是节省传输线，但相对于并行通信来说，速度较慢。

3. 串行外设接口

串行外设接口(Seri a l Peripheral Interface, SPI) 也是一种串行通信方式，主要用于MCU 扩展外围芯片使用。这些芯片可以是具有SPI 接口的AD 转换、时钟芯片等。

4. 集成电路互连总线

集成电路互连总线(Inter-Integrated C订cuit, I 2C) 是一种由PHILIPS 公司开发的两线式串行总线，有的书籍也记为IIC 或12 C, 主要用于用户电路板内MCU 与其外围电路的连接。

5. 通用串行总线

通用串行总线(Universal Serial Bus, USB) 是MCU 与外界进行数据通信的一种新的方式，其速度快，抗干扰能力强，在嵌入式系统中得到了广泛的应用。USB 不仅成为通用计算机上最重要的通信接口，也是手机、家电等嵌入式产品的重要通信接口。

6. 控制器局域网

控制器局域网(Controller Area Network ,CAN) 是一种全数字、全开放的现场总线控制网络，目前在汽车电子中应用最广。

7. 背景调试模式

背景调试模式(Background Debug Mode, BDM) 是Freescale 半导体公司提出的一种调试接口，主要用于嵌入式MCU 的程序下载与程序调试。

8. 边界扫描测试协议

边界扫描测试协议(Joint Test Action Group, JTAG) 是由国际联合测试行动组开发，对芯片进行测试的一种方式，可将其用于对MCU 的程序进行载入与调试。JTAG 能获取芯片寄存器等内容，或者测试遵守IEEE 规范的器件之间引脚连接情况。

9. 串行线调试技术

串行线调试(Seria l Wire Debug, SWD) 技术使用2 针调试端口， 是JTAG 的低针数和高性能替代产品，通常用于小封装微控制器的程序写入与调试。SWD 适用千所有ARM 处理器，兼容JTAG 。

## 1.4.3 与功能模块相关的术语

1. 通用捡入／ 给出

通用输入／输出(General Purpose I/ 0,GPIO), 即基本的输入／输出，有时也称并行I/0 。作为通用输入引脚时， MCU 内部程序可以读取该引脚，知道该引脚是"l" ( 高电平）或" O "（低电平），即开关橄输入。作为通用输出引脚时， MCU 内部程序向该引脚输出"1" (高电平）或"O"( 低电平），即开关量输出。

2. 模数转换与数模转换

模数转换(ADC) 的功能是将电压信号（模拟量） 转换为对应的数字量。实际应用中，这个电压信号可能由温度、湿度、压力等实际物理量经过传感器和相应的变换电路转化而来。经过AD 转换， MCU 就可以处理这些物理最。而与之相反，数模转换CDAC) 的功能则是将数字最转换为电压信号（模拟拯）。

3. 脉冲宽度调制器脉冲宽度调制器(Pulse Width Modulator,PWM), 是一个DA 转换器，可以产生一个高电平和低电平之间重复交替的输出信号，这个信号就是PWM 信号。

4. 看门狗

看门狗( Watch Dog), 是一种为了防止程序跑飞而设计的自动定时器。当程序跑飞时，由千无法正常执行清除看门狗定时器，看门狗定时器会自动溢出，使系统程序复位。

5. 液晶显示

液晶显示(Liquid Crystal Dispaly, LCD), 是电子信息产品的一种显示器件，可分为字段型、点阵字符型、点阵图形型三类。

6. 发光二极管

发光二极管(Light Emitting Diode,LED), 是一种将电流顺向通到半导体PN 结处而发光的器件。常用于家电指示灯、汽车灯和交通警示灯。

7. 键盘

键盘是嵌入式系统中最常见的输入设备。

1.5.1、C语言的运算符与数据类型

见P20

第二章 ARM Cortex-M0+处理器

# 2.1 ARM Cortex M0+处理器简介

Cortex-MO 十处理器不仅延续了易用性、C 语言编程模型等优势，而且能够兼容巳有的Cortex- MO 处理器的工具。作为Cortex-M 处理器系列中的一员， Cortex-MO 十处理器同样可获得ARM Cortex-M 整个系统的全面支持，其良好的软件兼容性，使其能够方便地被移植到更高性能的Cortex-M3 或Cortex-M4 等系列处理器。

## 2.1.1 ARM Cortex-MO 十处理器内部结构概要

1.M0+内核

32 位ARM Cortex-MO 十处理器是以一个“处理器子系统”的形式出现的，其CPU 内核本身与N VIC 和一系列调试块都亲密耦合。Cortex-MO 十处理器基于2 级流水线冯· 诺依曼架构CD , 内核版本为ARMv6-M, 支持16 位Thumb 指令集，同时采用Thumb2 技术。MO + 内核的性能是接近8 位或16 位竞争产品CoreMark/mA® 的二倍。

2.嵌套中断向量控制器

3.总线网络

4.调试组件

MO 十处理器实现了一个完全基于硬件的调试解决方案。该调试方案通过2 针脚串行线协议(SWD) 访问处理器、内存和外设

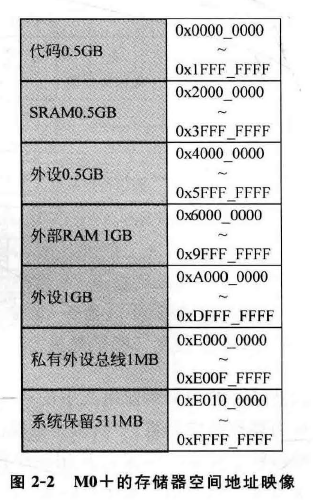
5.总线接口

CMO 十处理器提供一个基千被称之为高级微控制器总线体系结构的总线规范AMBA技术的单一32 位系统接口，可以高速整体访问所有系统外设和内存。

6.其他模块

## 2.1.2 ARM Cortex M0+处理器映像

CMO+ 处理器直接寻址空间为4GB, 地址范围是： OxOOOO\_OOOO ~ OxFFFF FFFF 。这里所说的存储器映像，其含义是指，把这4GB 空间当作存储器来看待，分成若干区间，都可安排一些什么实际的物理资源



## 2. 1. 3 ARM Cortex-MO 十处理器的寄存器

1. 通用寄存器RO~Rl2RO ~ Rl 2 是最具“通用目的＂的32 位通用寄存器，用千数据操作，复位后初始值为随机值。

2. 堆栈指针Rl3

Rl3 是堆栈指针。在CMO 十处理器内核中共有两个堆栈指针， 主堆栈指针MSP 和进程堆栈指针PSP, 若用户用到其中一个， 另一个必须用特殊指令来访问(MRS 、MSR 指令），因此任一时刻只能使用其中的一个

3. 连接寄存器R 14

当调用一个子程序时，由Rl4 存储返回地址

4. 程序计数寄存器R1 5

R15 是程序计数器(Program Counter, PC), 指向当前的程序地址。如果修改它的值，就能改变程序的执行流程（很多高级技巧隐藏其中） 。在汇编代码中也可以使用名字" PC"来访问它。

5. 特殊功能寄存器

MO + 内核包括一组特殊功能寄存器，包括程序状态字寄存器组(xPSR) 、中断屏蔽寄存器(PRIMASK) 和控制寄存器(CONTROL) 。

1) 程序状态字寄存器

程序状态字寄存器在内部分为以下几个子寄存器： APSR 、IPSR 、EPSR 。用户可以使用MRS 和MSR 指令访问寄存器。三个子寄存器既可以单独访问，也可以两个或三个组合到一起访问。使用三合一方式访问时， 把该寄存器称为xPSR

(1) 应用程序状态寄存器(Application Program Status Register, APSR) : 显示算术运算单元ALU 状态位的一些信息。

(2) 中断程序状态寄存器(Int errupt Program Status Register, IPSR) : 每次异常完成之后，处理器会实时更新IPSR 内的异常号。只

(3) 执行程序状态寄存器(Execution Program Status Register, EPSR): T 标志位指示当前运行的是否是Thumb 指令，该位是不能被软件读取的。运行复位向量对应的代码时置1 。如果该位为o, 会发生硬件异常，进入硬件中断服务例程。

2) 中断屏蔽寄存器

中断屏蔽寄存器PRIMASK 的D31~01 位保留，只有DO 位（记为PM) 有意义。当该位被置位时，除不可屏蔽中断和硬件错误之外的所有中断都被屏蔽

3) 控制寄存器

内核中的控制寄存器CONTROL 的D31 ~ 02 位保留， Dl 、DO 位含义如下。

Dl(SPSEL) -—堆栈指针选择位

DO(nPRIV) 一一－如果权限扩展，在线程模式下定义执行特权：

# 2.2 ARM Cortex-M0 +处理器指令系统

CPU 的功能是从外部设备获得数据，通过加工、处理，再把处理结果送到CP U 的外部世界。设计一个CPU, 首先需要设计一套可以执行特定功能的操作命令，这种操作命令称为指令。CPU 所能执行的各种指令的集合，称为该CPU 的指令系统

## 2.2.1 ARM Cortex-MO 十指令简表与寻址方式

1. ARM Cortex- MO 十指令简表



2. CMO 十的寻址方式

指令是对数据的操作， 通常把指令中所要操作的数据称为操作数， CMO 十处理器所需的操作数可能来自寄存器、指令代码、存储单元。而确定指令中所需操作数的各种方法称为寻址方式( Addressing Mode )

1）立即数寻址

在立即数寻址方式中，操作数直接通过指令给出，数据包含指令编码中，随着指令一起被编译成机器码存储于程序空间中。

2) 寄存器寻址

在寄存器寻址中，操作数来自千寄存器。

3) 偏移寻址及寄存器间接寻址

在偏移寻址中，操作数来自于存储单元，指令中通过寄存器及偏移量给出存储单元的地址。偏移址不超过4KB( 指令编码中偏移蜇为12 位）。偏移量为0 的偏移寻址也称为寄存器间接寻址。

4) 直接寻址

在直接寻址方式中，操作数来自于存储单元，指令中直接给出存储单元地址。

## 2.2.2 数据传送类指令

数据传送类指令的功能有两种情况， 一是取存储器地址空间中的数传送到寄存器中，二是将寄存器中的数传送到另一寄存器或存储器地址空间中。MO 十数据传送类基本指令有16 条。

1.取数指令

存储器中内容加载到寄存器中的指令见表2-4, 其中， LDR 、LDRH 、LDRB 指令分别表示加载来自存储器单元的一个字、半字和单字节（不足部分以0 填充）。LDRSH 和LDRSB指令指加载存储单元的半字、字节有符号数扩展成32 位到指定寄存器Rt 。

2. 存数指令

寄存器中内容存储至存储器中的指令见表2-5 。STR 、STRH 和STRB 指令存储Rt 寄存器中的字、低半字或低字节至存储器单元。存储器单元地址由Rn 与Rm 之和决定。Rt 、Rn 和Rm 必须为RO ~ R7 之一。

3. 寄存器间数据传送指令

4. 堆栈操作指令

堆栈操作指令见表2-7 。PUSH 指令将寄存器值存于堆栈中，最低编号寄存器使用最低存储地址空间， 最高编号寄存器使用最高存储地址空间； POP 指令将值从堆栈中弹回寄存器， 最低编号寄存器使用最低存储地址空间， 最高编号寄存器使用最高存储地址空间

## 2.2.3 数据操作类指令

1.算术运算类指令

2.逻辑运算类指令

3.移位类指令

4.位测试指令

5.数据序转之令

6.扩展类指令

2.2.4 跳转控制类指令

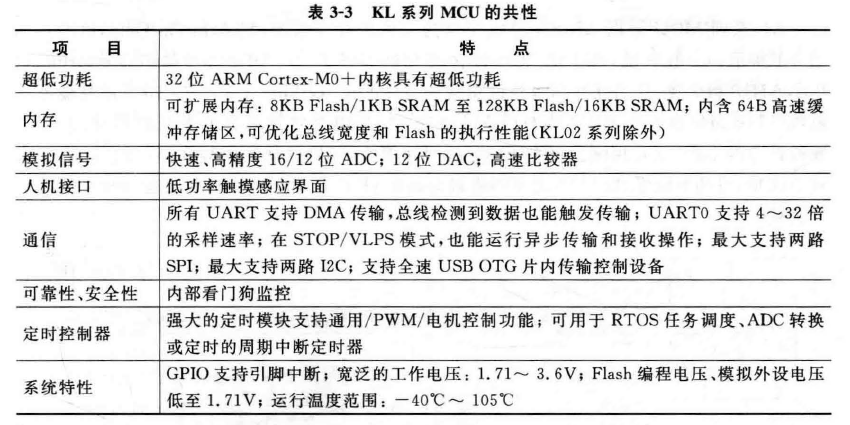
2.2.5 其他类指令

第三章 存储映像、中断源与硬件最小系统

# 3.2 KL系列MCU简介与体系结构概述

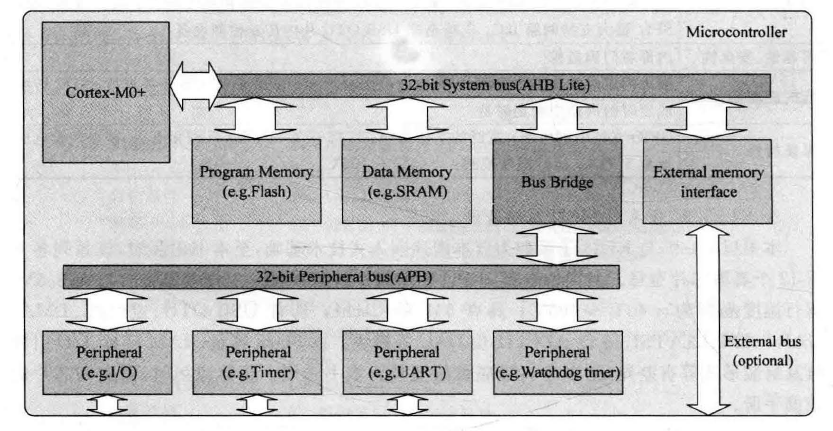
KL 系列MCU 千2012 年6 月提供样片， 2013 年正式上市。该系列MCU 是业内首款基于ARM Cortex-MO+ 内核的MCU, 具有超低功耗、应用设计方便、扩展性好、系列品种齐全等特点。目标市场是传统8 位MCU 应用领域的32 位升级换代

KL 系列MCU 在内核、低功耗、存储器、模拟信号、人机接口、安全性、定时器及系统特性等方面具有一些共同特点，简要总结



## 3. 2. 2 KL 系列MCU 体系结构概述

KL 系列MCU 是以AMBA 总线规范为架构的片上系统(System On Chip, SOC) , 如图3-1 所示。一般来说， AMBA 架构包含高性能系统总线( Advanced High PerformanceBus,AHB) 和低速、低功耗的高级外设总线( Adva nced PeriPheral Bus, APB) 。高性能系统总线AHB 是负责连接ARM 内核、DMA 控制器、片内存储器或其他需要高带宽的模块。而外设总线APB 则是用来连接系统的外围慢速模块，其协议规则相对系统总线AHB 来说较为简单，它与系统总线AHB 之间则通过总线桥( Bus Bridge ) 相连， 期望能减少系统总线的负载。



1.AMBA总线规范

ARM 公司定义了AMBA (Advanced Microcontroller Bus Arch i tecture) 总线规范，它

是一组针对基千ARM 内核、片内系统之间通信而设计的标准协议。在AMBA 总线规范

中，定义三种总线，分别是：1）高性能总线 2）高级系统总线 3）高级外设总线

2.总线桥

外设桥的作用是把交叉开关(CrossbarSwitch) 接口协议，转换成私有外设总线协议CIPS/ APB) 。本书中MCU 外设桥以外设槽(Slot) 形式，最多可连接128 个外设，给每个外设分配4KB 的寄存器映像空间。外设桥为每个外设槽提供了独立时钟，以便更好地支持低速外设。

3.交叉开关

交叉开关(Crossbar Switch) 将总线主机与总线从机相连，该结构允许多达4 路主机同时访问不同总线从机。所谓总线主机是指其可在总线上产生与控制所有时序

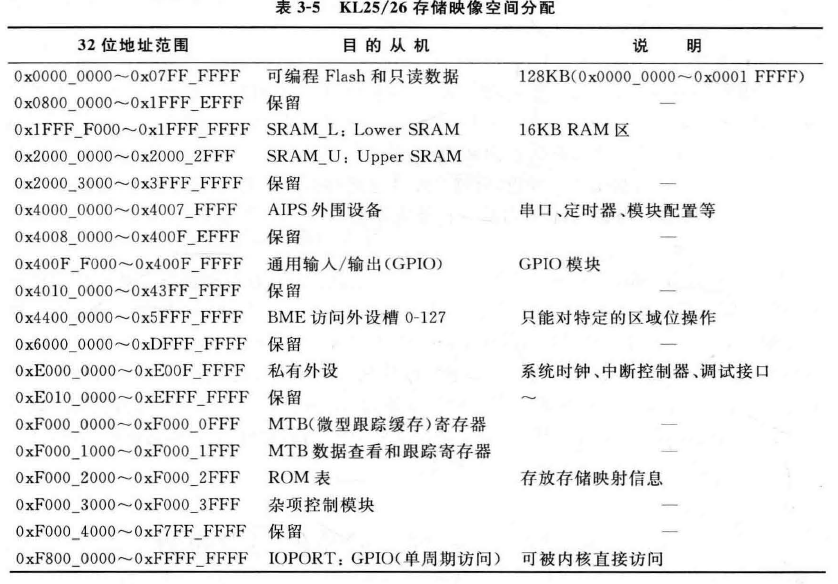
# 3.3 KL25/26 系列存储映像与中断源

## 3.3.1 KL25/26系列存储映像

所谓存储映像(Memory Mapping) 在这里可以直观地理解为， MO + 寻址的4GB 地址空

间(OxOOOO\_OOOO ~ OxFFFF\_FFFF严被如何使用，都对应了哪些实际的物理介质。为区别于CPU 内部寄存器， GPIO 寄存器也被称为＂映像寄存器"(MappingRegister), 相对应的地址被称为＂映像地址" (Mapping Address) 。整个可直接寻址的空间被称为＂映像地址空间"(Mapping Address Space) 。

KL25 /2 6 把MO + 内核之外的模块，用类似存储器编址的方式，统一分配地址。在4GB的映像地址空间内，分布着片内F lash 、SRAM 、系统配置寄存器以及其他外设等，以便CPU通过直接地址进行访问



1. 片内Fl as h 区存储映像

KL25/26 片内Flash 大小为128KB, 地址范围是： OxOOOO\_OOOO ~ OxOOOI\_ FFFF, 一般被用来存放中断向量、程序代码、常数等，其中前1928 为中断向量表。

2. 片内RAM 区存储映像

KL25/26 片内RAM 为静态随机存储器SRAM, 大小为16KB, 地址范围是： OxlFFF\_FOOO~Ox2000\_2FFF, 一般被用来存储全局变量、静态变量、临时变量（堆栈空间）等。

3. 其他存储映像

其他存储映像，如外设区存储映像（外设桥、GPIO 、位操作引擎等）、私有外设总线存储映像系统保留段存储映像等

## 3. 3. 2 KL25/26 中断源

中断是计算机发展中一个重要的技术，它的出现很大程度上解放了处理器，提高了处理器的执行效率。所谓中断，是指MCU 在正常运行程序时，由于MCU 内核异常或者MCU各模块发出请求事件，引起MCU 停止正在运行的程序，而转去处理异常或执行处理外部事件的程序（又称中断服务程序）。

KL25 / 26 的中断源分为两类，如表3-6 所示，

一类是内核中断，另一类是非内核中断。内核中断主要是异常中断， 也就是说，当出现错误的时候，这些中断会复位芯片或是做出其他处理。非内核中断是指MCU 各个模块被中断源引起的中断， MCU 执行完中断服务程序后，又回到刚才正在执行的程序，从停止的位置继续执行后续的指令。非内核中断又称可屏蔽中断，这类中断可以通过编程控制，开启或关闭该中断。



## 3. 4. 1 硬件最小系统引脚

KL2 5 / 26 硬件最小系统引脚是我们需要为芯片提供服务的引脚，包括电源类引脚、复位引脚、晶振引脚等

# 3.4 KL25/26的引脚功能

每个引脚都可能有多个复用功能，有的引脚有两个复用功能，有的有4 个复用功能，实际嵌入式产品的硬件系统设计时必须注意只能使用其中的一个功能。进行硬件最小系统设计时， 一般以引脚的第一功能作为引脚名进行原理图设计，若实际使用的是其另一功能，可以用括号加以标注，这样设计的硬件最小系统就比较通用。下面从需求与供给的角度把MCU 的引脚分为“硬件最小系统引脚”与"I/0 端口资源类引脚”两大类。

## 3.4.2 对外提供服务的引脚

除了需要我们为芯片服务的引脚（最小硬件系统引脚）之外，芯片的其他引脚为我们提供服务， 也可称之为I / 0 端口资源类引脚



# 3. 5 KL25/26 硬件最小系统原理图

MCU 的硬件最小系统是指包括电源、晶振复位、写入调试器接口等可使内部程序得以运行的、规范的、可复用的核心构件系统

## 3. 5. 1 电源及其滤波电路

电路中需要大量的电源类引脚用来提供足够的电流容量同时保持芯片电流平衡，所有的电源引脚必须外接适当的滤波电容抑制高频噪声。

## 3.5.2 复位电路及复位功能

复位，意味着MCU 一切重新开始。复位引脚为RESET 。若复位引脚有效（低电平） ，则会引起MCU 复位。复位电路原理如下： 正常工作时，复位引脚RESET 通过一个lOk.O.的电阻接到电源正极， 所以应为高电平。若按下复位按钮，则RESET 脚接地为低电平， 导致芯片复位。若是系统重新上电，芯片内部电路会使RESET 脚拉低，使芯片复位

从引起MCU 复位的内部与外部因素来区分，复位可分为外部复位和内部复位两种。

从复位时芯片是否处千上电状态来区分， 复位可分为冷复位和热复位。芯片从无电状态到上电状态的复位属于冷复位，芯片处于带电状态时的复位叫热复位。冷复位后， MCU内部 RAM 的内容是随机的。而热复位后， MCU 内部RAM 的内容会保持复位前的内容，即热复位并不会引起RAM 中内容的丢失。

从CPU 响应快慢来区分， 复位还可分为异步复位与同步复位。异步复位源的复位请求一般表示一种紧要的事件，因此复位控制逻辑不等到当前总线周期结束， 复位立即有效。异步复位源有上电、低电压复位等。同步复位的处理方法与异步复位不同： 当一个同步复位源给出复位请求时， 复位控制器并不使之立即起作用，而是等到当前总线周期结束之后，这是为了保护数据的完整性。

第四章 GPIO及程序框架

# 4.1 通用I/0 接口基本概念及连接方法

1. I / 0 接口的概念

I/0 接口，即输入／输出(Input/ Output) 接口， 是MCU 同外界进行交互的重要通道，MCU 与外部设备的数据交换通过1/ 0 接口来实现。I/ 0 接口是一个电子电路，其内由若干专用寄存器和相应的控制逻辑电路构成。

有时把1/ 0 引脚称为接口(Interface) , 而把用千对I / 0 引脚进行编程的寄存器称为端口(Port), 实际上它们是紧密相连的。因此，不必深究它们之间的区别。有些书中甚至直接称1/ 0 接口（端口）为I/ 0口。在嵌人式系统中，接口千变万化，种类繁多，有显而易见的人机交互接口，如操纵杆、键盘、显示器；也有无人介入的接口，如网络接口、机器设备接口等。

2. 通用I/ 0

所谓通用1/0, 也记为GPIO(General Purpose 1/0), 即基本的输入／输出，有时也称并行I/0, 或普通I/ 0, 它是1/ 0 的最基本形式。本书中使用正逻辑，电源(Vee) 代表高电平，对应数字信号"l" ; 地(GND) 代表低电平，对应数字信号"O" 。作为通用输入引脚， MCU 内部程序可以通过端口寄存器获取该引脚状态，以确定该引脚是"l" (高电平）或"O"( 低电平），即开关撮输入。作为通用输出引脚， MCU 内部程序通过端口寄存器控制该引脚状态，使得引脚输出"l"( 高电平）或"O"( 低电平），即开关筐输出。大多数通用I/ 0 引脚可以通过编程来设定其工作方式为输入或输出，称为双向通用I/ 0 。

3. 上拉下拉电阻与输入引脚的基本接法

芯片输入引脚的外部有三种不同的连接方式：带上拉电阻的连接、带下拉电阻的连接和“悬空“连接。通俗地说，若MCU 的某个引脚通过一个电阻接到电源(Vee ) 上，这个电阻被称为“上拉电阻＂；与之相对应，若MCU 的某个引脚通过一个电阻接到地(GND) 上，则相应的电阻被称为“下拉电阻＂ 。这种做法使得悬空的芯片引脚被上拉电阻或下拉电阻初始化为高电平或低电平。

4. 输出引脚的基本接法

作为通用输出引脚， MCU 内部程序向该引脚输出高电平或低电平来驱动器件工作，即开关量输出

# 4.2 端口控制模块与GPIO模块的编程结构

## 4.2.1 端口控制模块——决定引脚复用功能

KL2 5 的大部分引脚具有复用功能，可以通过端口控制模块(Port Control andInterrupts, PORT ) 提供的寄存器编程指定其为某一具体功能。

PORT 模块内含三类寄存器，分别是引脚控制寄存器(Pin Control Register入全局引脚控制寄存器(Global Pin Control Regi ster汃中断状态标志寄存器(Interrupt Status Flag Register) 。

1.寄存器映像地址分析

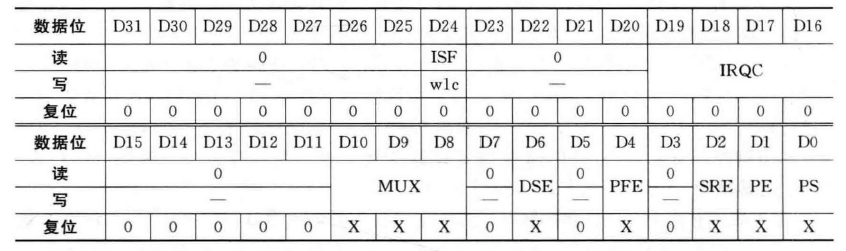
KL2 5 芯片有5 个端口A~ E 。每个端口有32 个引脚控制寄存器PORTx\_PCRn( 其中x=A~ E,n=0~31), 两个全局引脚控制寄存器(PORTx \_G PCLR 、PORTx\_GPCHR) 、一个中断状态标志寄存器(PORTx\_ ISFR) 。以下地址分析计算均为十六进制，为书写简化起见，在不致引起歧义的情况下，略去十六进制前缀"Ox" 不写。

每个端口有32 个引脚控制寄存器PORTx\_PCRn 。端口x 的基地址= 4004\_9000+ xXlOOO(x= A~ E, 对应0 ~ 4) 。端口x 的每个引脚控制寄存器PORTx\_PCRn 的地址为＝4004\_9000+xX lOOO+nX 4(x= A~E, 对应0~4, n =0~31) 。这样5 个端口，共5X32 =1 60 个引脚控制寄存器，每个引脚控制寄存器的地址很容易计算出来。例如， PORTA \_PCRl 的地址为： 4004\_9000 + 0 X 1000+ 1 X 4 = 4004\_ 9004 。

每个端口有两个全局引脚控制寄存器。全局引脚控制寄存器（低） PORTx\_GPCLR, 地址= 4004 \_ 9080 + x X 1000 (x= A ~ E, 对应0 ~ 4) ; 全局引脚控制寄存器（高） PORTx\_GPCHR, 地址=4004\_9084+ x X l OOO(x=A~E, 对应0~4) 。每个端口有一个中断状态标志寄存器。地址=4004\_90 AO+ xX 1000(x= A~ E, 对应0~4) 。

2.引脚控制寄存器

每个端口的每个引脚均有一个对应的引脚控制寄存器(PORTx\_ PCRn ), 可以配置引脚中断或DMA 传输请求，可以配置引脚为GPIO 功能或其他功能，可以配置是否启用上拉或下拉，可以配置选择输出引脚的驱动强度，可以配置选择输入引脚是否使用内部滤波等。



其中， "X"表示复位后状态不确定。下面给出有关功能说明，未说明的位或字段均为保留（只读，值为0) 。

D24CISF)—中断状态标志（只读） 。数字引脚模式下有效。ISF = O, 未检测到引脚中

断； ISF= 1 , 检测到引脚中断。向该位写1 , 可清除中断状态标志

D1 9 ~ D16CIRQC) 一— 中断配置情况（读／写） 。数字引脚模式下有效。IRQC = O O O O ,关闭引脚中断/DMA 请求； IRQC = OOOl ~ OOll ——分别对应上升沿、下降沿、沿跳变，触发DMA 请求； 0100一保留； 1000 ~ 1100-分别对应逻辑低电平（逻辑0 ) 、上升沿、下降沿、沿跳变、高电平（逻辑1)' 触发引脚中断。其他值一保留。特别注意：井不是所有KL25 的引脚均可配置为中断功能，只有A 口、D 口的引脚具有上述这种中断功能。

D6(DSE)一驱动能力使能位（读／写）。表明引脚被配置为数字输出时的驱动能力状况，数字引脚模式下有效。

D4(PFE)一无源滤波使能位（读／ 写） 。数字引脚模式下有效。PFE =O , 相应的引脚

禁止无源输入滤波； PFE= l, 相应的引脚启用无源输入滤波

DZ(SRE)一转换速率使能位（读／写）。数字引脚模式下有效。0一—引脚配置成快转换速率； 1一—引脚配置成慢转换速率。

DlCPE) -—上拉或下拉使能位（读／写）。数字引脚模式下有效。0——相应的引脚关闭内部上拉或下拉电阻； 1-一相应的引脚启用内部上拉或下拉电阻，引脚作为数字输入。

DO(PS) -—－上拉或下拉选择（ 读／ 写）。数字引脚模式下有效。PS =O , 如果PE = l, 引脚下拉电阻使能； PS= l, 如果PE = l, 引脚上拉电阻使能。KL2 5 内部上下拉电阻大小为20~50k0 。

3. 全局引脚控制寄存器

4. 中断状态标志寄存器

## 4. 2. 2 GPIO 模块一—对外引脚与内部寄存器

1. KL25 的GPIO 引脚

KL25 的大部分引脚具有多重复用功能，可以通过给出的寄存器编程来设定使用其中某一种功能。80 引脚封装的KL25 芯片的GPIO 引脚分为5 个端口，标记为A 、B 、C 、D 、E , 共含61 个引脚。端口作为GPIO 引脚时，逻辑1 对应高电平，逻辑0 对应低电平

2. GPIO寄存器

每个GPIO 口均有6 个寄存器， 5 个GPIO 口共有30 个寄存器。A 、B 、C 、D 、E 各口寄存器的基地址分别为400F\_FOOOh 、400F\_F040h 、400F\_F0080h 、400F\_FOCOh 、400F\_Fl OOh ,所以各口基地址相差40h 。各GPIO 口的6 个寄存器分别是数据输出寄存器、输出置1 寄存器、输出清0 寄存器、输出反转寄存器、数据输入寄存器、数据方向寄存器其中，输出寄存器的地址就是口的基地址，其他寄存器的地址依次加4 。



## 4.2.3 GPIO基本编程步骤和基本打通程序

1. GPIO 基本编程步骤

要使芯片某一引脚为GPIO 功能，并定义为输入／输出，随后进行应用，基本编程步骤如下。

(1) 通过端口控制模块(PORT) 的引脚控制寄存器PORTx\_ PCRn 的引脚复用控制字段(MUX) 设定其为GPI O 功能（即令MUX=ObOOl) 。

(2) 通过GPIO 模块相应口的“数据方向寄存器”来指定相应引脚为输入或输出功能。若指定位为o, 则为对应引脚输入；若指定位为I, 则为对应引脚输出。

(3) 若是输出引脚，则通过设置“数据输出寄存器”来指定相应引脚输出低电平或高电平，对应值为0 或1 。也可通过“输出置位寄存器”“输出清位寄存器”“输出取反寄存器”改变引脚状态，参见表4-1 中关于寄存器的说明。

(4) 若是输入引脚，则通过“数据输入寄 存器”获得引脚的状态。若指定位为o, 表示当前该引脚上为低电平； 若为1, 则为高电平。

# 实例演示:

1、改绿灯亮，且用一个输入控制灯的亮，且进行单步调试

步骤：

1）通过端口控制模块将绿灯的引脚LIGHT\_GREEN (PTB\_NUM|18) 的全局引脚寄存器的MUX字段赋值001——设定为GPIO引脚，并指定为输出功能

(1)计算给出PORTB18的引脚控制寄存器地址

//PORTB端口的引脚控制寄存器基地址为0x4004A000u（后缀u表示无符号数）

volatile uint\_32 \*portB\_ptr = (uint\_32\*) 0x4004A000u;

//PORTB9的引脚控制寄存器地址=基地址+偏移量

volatile uint\_32 \*portB\_PCR\_18 = portB\_ptr + 18;

//PORTB端口（作为GPIO功能）的基地址为0x400FF040u

volatile uint\_32 \*gpioB\_ptr = (uint\_32\*) 0x400FF040u;

//PORTB的数据方向寄存器地址=基地址+偏移量

volatile uint\_32 \*portB\_PDDR = gpioB\_ptr + 5;

//PORTB的输出寄存器地址=基地址+偏移量

volatile uint\_32 \*portB\_PDO = gpioB\_ptr + 0;

//PORTB的输出反转寄存器地址=基地址+偏移量

volatile uint\_32 \*portB\_PTOR = gpioB\_ptr + 3;

//(4)通过PORTB的输出寄存器(给蓝色小灯的寄存器）赋初值1，保证定义为输出时为暗

\*portB\_PDO |= (1 << 18);

//(5)通过PORTB的方向寄存器，定义PORT18引脚输出

\*portB\_PDDR |= (1 << 18);

2）制定A端口的A1为GPIO引脚并指定为输入功能，并计算相关寄存器的地址

volatile uint\_32 \*portA\_ptr = (uint\_32\*) 0x40049000u;

volatile uint\_32 \*portA\_PCR\_1 =portA\_ptr+1;//端口A引脚1的寄存器

\*portA\_PCR\_1 &= 0b11111111111111111111100011111111; //清MUX位段

\*portA\_PCR\_1 |= 0b00000000000000000000000100000000;

//PORTA的GPIO寄存器的基地址和输入寄存器的和方向寄存器的基地址

volatile uint\_32 \*gpioA\_ptr = (uint\_32\*) 0x400FF000u;

volatile uint\_32 \*portA\_PDIR =gpioA\_ptr+4 ;

volatile uint\_32 \*portA\_PDDR =gpioA\_ptr+5 ;

\*portA\_PDDR &= ~(1 << 1);

3）设定主程序当A1引脚输入为0时，灯亮，输入为1时，灯灭。

for (;;) {

if((\*portA\_PDIR) & (1<<1)){

\*portB\_PDO &= ~(1<<18);//输出寄存器的第18位控制小灯的输出——亮灭

}else{

\*portB\_PDO |= (1<<18);

}

单步调试:进入调试模式并且按F5/F6分别向前向后进行调试；