第3章

存储器映像、中断源与硬件最小系统

本章导读

本章主要介绍以 ARM Cortex-M0+为核心的 MSPM0 系列 MCU，给出该 MCU 的存储映像、中断源与硬件最小系统，并由此构建一种通用嵌入式计算机（AHL-MSPM0L1306），作为本书硬件实践平台。MCU 的外围电路简单清晰，它以 MCU 为核心，辅以最基本电子线路，构成了 MCU 硬件最小系统，使得 MCU 的内部程序可以运行起来。我们在此基础上进行嵌入式系统的软硬件学习。

3.1 MSPM0 系列 MCU 概述

本节简要概述 MSPM0 系列的 MCU 命名规则、存储映像以及中断源。其中，MCU 命名规则帮助使用者获得芯片信息；MSPM0 存储映像为 M0+内核之外的模块，用类似存储器编址的方式，统一分配地址；中断源主要包括 MSPM0 中断源的定义及分类。

3.1.1 MSPM0 系列 MCU 命名规则

MSPM0 系列 MCU 是德州仪器（TI）公司于 2023 年开始陆续推出的基于 M0+内核的超低功耗微控制器，与所有 ARM 工具和软件兼容。其内部硬件模块主要包括通用输入输出接口（GPIO）、串行通信接口（UART）、串行外设接口（SPI）等。

MCU 型号名称中主要包括处理器系列、MCU 平台、产品系列、器件子系列、Flash 大小、温度范围以及封装类型等信息。MSPM0 系列芯片的命名格式为“**MSP M**0 **X AAA Y** S BBB R”，加粗的字段是基本部分，印刷在芯片表面；后面不加粗的字段为产品系列名称的后缀，购买时分别表示温度范围、封装类型和配送形式。MSPM0 系列芯片命名字段说明如表 3-1 所示。

例如，本书所使用的芯片型号为 **MSPM0L1306**SRHBR。从基本字段可知，该芯片为 TI 的混合信号处理器 MSP，基于 ARM Cortex M0+内核的微控制器家族，频率 32MHz，130 子系列（12 位 ADC），Flash 大小为 64KB；从后缀字段可知，芯片温度范围是－40～125℃，封装形式为 32 引脚超薄无引线四方扁平封装（Very-thin Quad Flat No-lead，VQFN），配送形式为大卷带包装。

表 3-1 MSPM0 系列芯片命名字段说明

表现位置	字 段	说 明	取 值
芯片表面	MSP	处理器系列	MSP=混合信号处理器
	M0	MCU 平台	M0 表示 TI 的基于 ARM Cortex M0+内核的微控制器家族
	X	产品系列	L=32MHz 频率
	AAA	器件子系列	130=ADC、2x OPA、COMP 134=ADC、2x OPA(10pA 输入偏置电流)、COMP
	Y	Flash 存储器	4=16KB;5=32KB;6=64KB
购买选型	S	温度范围	T=-40～105℃;S=-40～125℃
	BBB	封装类型	DGS 代表 VSSOP,DYY 代表 SOT,RGE 代表 VQFN-24,RHB 代表 VQFN-32
	R	配送形式	T=小卷带;R=大卷带;无标记=管装或托盘

3.1.2 MSPM0 存储器映像

ARM Cortex-M0+处理器直接寻址空间为 4GB,地址范围是 0x0000_0000～0xFFFF_FFFF。存储器映像是指把 4GB 空间当作存储器,分成若干区间,都可安排一些实际的物理资源。MCU 生产厂家会规定好各地址服务的资源,用户只能用不能改。

MSPM0L130 为 M0+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 的存储映射空间内,片内 Flash、静态存储器 SRAM、系统配置寄存器以及其他外设,如通用型输入/输出(GPIO),被分配给独立的地址,以便内核进行访问。表 3-2 给出了 MSPM0L 系列存储器映像的地址范围及对应内容。

表 3-2 MSPM0L 存储器映射表

32 位地址范围	对 应 内 容	说 明
0x0000_0000～0x1FFF_FFFF	代码	Flash 存储与 ROM
0x2000_0000～0x3FFF_FFFF	静态随机存储器 M	SRAM
0x4000_0000～0x5FFF_FFFF	外设	映射到各个外设
0x6000_0000～0x7FFF_FFFF	子系统	本地 CPU 子系统内存映射寄存器
0xE000_0000～0xE00F_FFFF	系统 PPB	ARM 私有外设总线使用

关于存储空间的使用,主要关注片内 Flash 区和片内 RAM 区存储映像。因为中断向量、程序代码、常数放在片内 Flash 中,所以在源程序编译后的链接阶段使用的链接文件中,需要含有目标芯片 Flash 的地址范围以及用途等信息,才能顺利生成机器码。在产生的链接文件中还需要包含 RAM 的地址范围及用途等信息,用于生成机器码,准确定位全局变量、静态变量的地址及堆栈指针。

1. MSPM0L1306 片内 Flash 区存储映像

片内 Flash 存储器用来中断向量、程序代码、常数等。MSPM0L1306 片内 Flash 大小为 64KB,其地址范围是 0x0000_0000～0x0000_FFFF,分为 64 扇区,每扇区 1KB。

2. MSPM0L1306 片内 RAM 区存储映像

片内 RAM 一般用来存储全局变量、静态变量、临时变量(堆栈空间)等。MSPM0L1306 片内 RAM 为静态随机存储器(SRAM),大小为 4KB,地址范围是 0x2000_0000～0x2000_0FFF。该芯片堆栈空间的使用是从高地址向低地址方向进行的,因此将堆栈的栈顶(Stack Top)设置成 RAM 地址的最大值。这样,全局变量及静态变量从 RAM 的低地址向高地址方向使用,堆栈从 RAM 的高地址向低地址方向使用,可以减少重叠错误。

3.1.3　MSPM0 中断源

中断是计算机发展中一项重要的技术,它的出现在很大程度上解放了处理器,提高了处理器的执行效率。中断指 MCU 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(又称中断服务例程)。

引起 MCU 中断的事件称为中断源,一个 MCU 具有哪些中断源是在芯片设计阶段确定的。MSPM0L 的中断源分为两类,一类是内核中断,另一类是非内核中断,如表 3-3 所示。内核中断主要是异常中断,当出现错误的时候,这些中断会复位芯片或做出其他处理。非内核中断是 MCU 各个模块引起的中断,MCU 执行完中断服务例程后,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类中断。表 3-3 中,中断向量号是从 0 开始编号的,包含内核中断和非内核中断,与中断向量表一一对应。中断请求(**I**nterrupt **R**e**Q**uest,IRQ)号是非内核中断从 0 开始的编号,因此对内核中断来说为负值,编程时直接使用统一的按照中断向量号排序的中断向量表。

表 3-3　MSPM0L1306 中断向量表

中断类型	IRQ 号	中断向量号	优先级号	中　断　源	默认引用名
内核中断		0		_estack	
		1	－3	重启	Reset
	－14	2	－2	NMI	NMI Interrupt
	－13	3	－1	硬性故障	HardFault Interrupt
		4～10		保留	
	－5	11	可选	SVC	SV Call Interrupt
		12～13		保留	
	－2	14	可选	PendSV	Pend SV Interrupt
	－1	15	可选	SysTick	SysTick Interrupt
非内核中断	0	16	可选	INT_GROUP0	Combined Peripheral GROUP0
	1	17	可选	INT_GROUP1	Combined Peripheral GROUP1
	2	18	可选	TIMG1	Timer TIMG1 Interrupt

续表

中断类型	IRQ 号	中断向量号	优先级号	中 断 源	默认引用名
非内核中断	3	19	可选	保留	
	4	20	可选	ADC0	ADC0 Interrupt
	5～8	21～24	可选	保留	
	9	25	可选	SPI0	SPI0 Interrupt
	10～12	26～28	可选	保留	
	13	29	可选	UART1	UART1 Interrupt
	14	30	可选	保留	
	15	31	可选	UART0	UART0 Interrupt
	16	32	可选	TIMG0	Timer TIMG0 Interrupt
	17	33	可选	保留	
	18	34	可选	TIMG2	Timer TIMG2 Interrupt
	19	35	可选		
	20	36	可选	TIMG4	Timer TIMG4 Interrupt
	21～23	37～39	可选	保留	
	24	40	可选	I2C0	I2C0 Interrupt
	25	41	可选	I2C1	I2C1 Interrupt
	26～30	42～46	可选	保留	
	31	47	可选	DMA	DMA Interrupt

3.2 MSPM0L 的引脚图与硬件最小系统

要使一个 MCU 芯片可以运行程序，必须为它做好服务工作，找出哪些引脚需要由用户提供服务，如电源与地、晶振、程序写入引脚、复位引脚等。

3.2.1 MSPM0L 的引脚图

本书以 32 引脚 VQFN 封装的 MSPM0L1306 芯片为例阐述 MCU 的应用编程，其引脚图如图 3-1 所示。芯片引脚分为两大部分，一是需要用户为它服务的引脚，也称为硬件最小系统引脚；二是它为用户服务的引脚，也称为 I/O 端口资源类引脚。芯片的引脚功能请参阅电子资源“\01-Document\MSPM0L1306 芯片资料”文件夹中的数据手册第 6 章。

1. 硬件最小系统引脚

硬件最小系统引脚是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，表 3-4 中给出了 MSPM0L1306 的硬件最小系统引脚。MSPM0L1306 芯片电源类引脚在 VQFN32 封装中有 3 个。

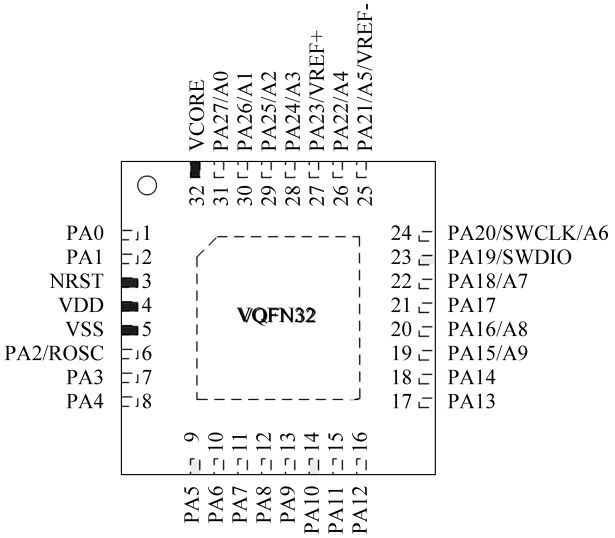

图 3-1　32 引脚 VQFN 封装 MSPM0L1306

表 3-4　MSPM0L1306 硬件最小系统引脚表

分　类	引　脚　名	引脚号	功 能 描 述
电源输入	VDD	4	电源，典型值：3.3V
	VSS	5	地，典型值：0V
	VCORE	32	稳压内核电源输出(一般悬空)
复位	NRST	3	低电平有效复位信号，须上拉至 VCC，否则无法启动
SWD 接口	SWDIO/PTA19	23	SWD 数据信号线
	SWDCLK/PTA20	24	SWD 时钟信号线
引脚个数统计			硬件最小系统引脚为 6 个

2. 对外提供服务引脚

对外提供服务的引脚也可称为 I/O 端口资源类引脚，具体信息见表 3-5。这些引脚一般具有多种复用功能。MSPM0L(32 引脚 VQFN 封装)有 28 个 I/O 引脚，记为 PA 口，为了与其他芯片统一，可把它标识为 PTA 口。这些引脚一般均具有多个复用功能，在复位后会立即被配置为高阻状态，且为通用输入引脚，有内部上拉功能。

表 3-5　MSPM0L 对外提供 I/O 端口资源类引脚表

端　口　号	引　脚　数	引　脚　名	硬件最小系统复用引脚
A	28	PTA[0-27]	PTA19、PTA20
合计	28		
说明	本书中所涉及的 GPIO 端口(如 PTA 引脚)与图 3-1 中的 PA 引脚同义，均可作为 Port A 的缩写		

【思考】 把 MCU 的引脚分为硬件最小系统引脚与对外提供服务引脚，对嵌入式系统的硬件设计有何益处？

3.2.2 MSPM0L 硬件最小系统原理图

MCU 的硬件最小系统指包括电源、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电路。图 3-2 给出了 MSPM0L 硬件最小系统原理图。使用一个芯片，必须完全理解其硬件最小系统。随着 Flash 存储器制造技术的发展，大部分芯片提供了在板或在线系统(On System)的写入程序功能，即，把空白芯片焊接到电路板上后，再通过写入器把程序下载到芯片中，因此硬件最小系统包含写入器的接口电路。

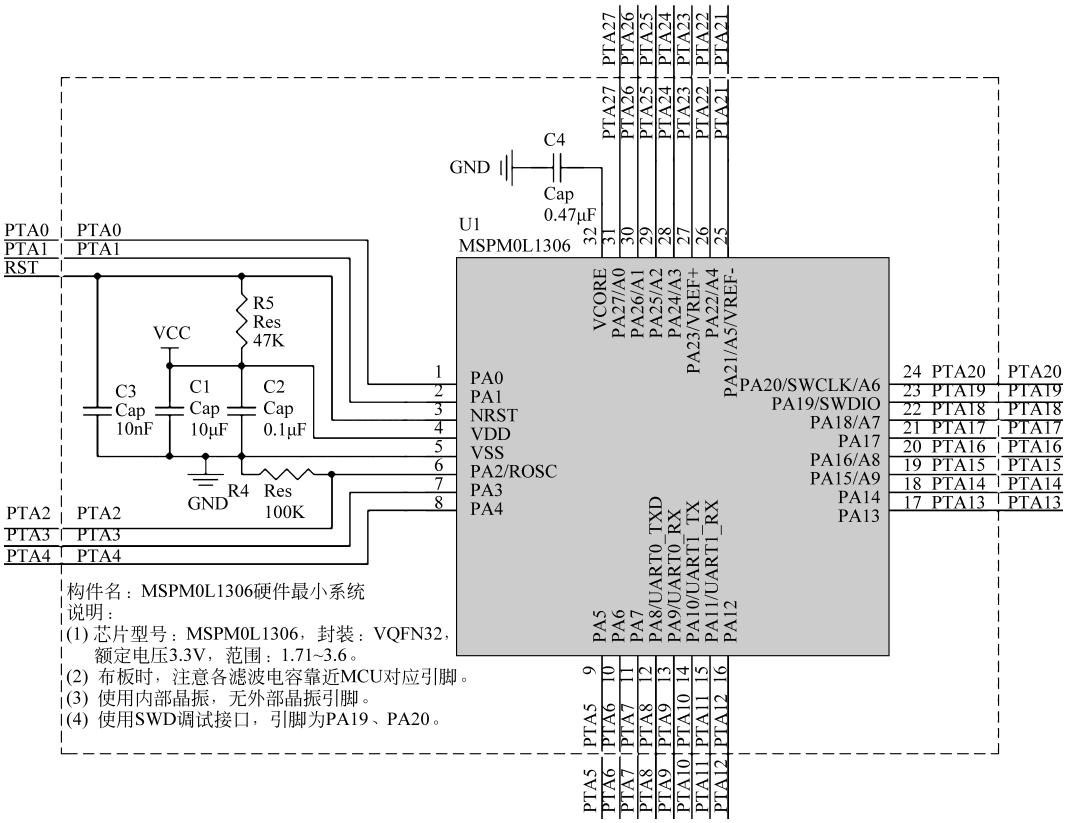

图 3-2 MSPM0L1306 硬件最小系统原理图

1. 电源及其滤波电路

为使进入 MCU 内部的电源稳定，电源引出脚必须外接适当的滤波电容，以抑制电源波动。电源滤波电路可改善系统的电磁兼容性，降低电源波动对系统的影响，增强电路工作的稳定性。实际布板时滤波电容尽量靠近芯片引脚，才能起到滤波效果。

【思考】 实际布板时，电源与地之间的滤波电容为什么要靠近芯片引脚？简要说明电容容量大小与滤波频率的关系。

2. 复位引脚

复位引脚为 RESET，若复位引脚有效(低电平)，则会引起 MCU 复位，可从不同角度对复位进行分类。根据复位时芯片是否处于上电状态来区分，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(如按

下复位按键的复位)。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。

3. 晶振电路

计算机工作需要一个时间基准,这个时间基准由晶振电路提供。MSPM0L 芯片可使用内部晶振为 MCU 提供工作时钟,不需要再另接外部晶振。MSPM0L1306 芯片内部时钟源可以通过编程产生最高 32MHz 的时钟频率,供系统总线及各个内部模块使用。

4. SWD 接口引出脚

在芯片内部没有程序的情况下,需要用写入器将程序写入芯片,串行线调试接口 SWD 是一种写入方式的接口。MSPM0L1306 芯片的 SWD 基于 CoreSight 架构①,该架构在限制输出引脚和其他可用资源的情况下,提供了最大的灵活性。通过 SWD 接口可以实现程序下载和调试功能。SWD 接口只需两根线——数据输入/输出线(DIO)和时钟线(CLK),实际应用时还包含电源与地。MSPM0L1306 芯片中,DIO 为引脚 PTA19,CLK 为引脚 PTA20。

在本书中,SWD 写入器用于写入 BIOS,随后在 BIOS 支持下,利用一根 Type-C 线连接 PC 的 USB 接口,即可把 PC 作为工具机进行嵌入式系统的学习与应用开发,这就是通用嵌入式计算机的优点。因此,本书内所夹带的 AHL-MSPM0L1306 硬件系统不包含 SWD 写入器,而是通过 Type-C 线与 PC 连接,实现用户程序的写入。

3.3 由 MCU 构建通用嵌入式计算机

嵌入式计算机是一台微型计算机。目前嵌入式系统的开发模式大多数是从“零”做起的,即,硬件从 MCU(或 MPU)芯片做起,软件从自启动开始,因而增加了嵌入式系统的学习与开发难度,存在软硬件开发颗粒度低、可移植性弱等问题。MCU 性能的不断提高与软件工程概念的普及给解决这些问题提供了契机。若能像通用计算机那样,把做计算机与用计算机的工作相对分开,则可以提高软件可移植性,降低嵌入式系统开发门槛,对嵌入式人工智能、物联网、智能制造等嵌入式应用领域形成有力推动。

3.3.1 嵌入式终端开发方式存在的问题与解决办法

1. 嵌入式终端 UE 开发方式存在的问题

微控制器 MCU 是嵌入式终端 UE 的核心,承担着传感器采样、滤波处理、融合计算、通信、控制执行机构等功能。MCU 生产厂家往往配备一本厚厚的参考手册,少则几百页,多则可达近千页。许多厂家也提供软件开发包(Software Development Kit,SDK)。但是,MCU 的应用开发人员通常花费太多的精力在底层驱动上,终端 UE 的开发方式存在软硬件设计颗粒度低、可移植性弱等问题。

(1) 硬件设计颗粒度低。以窄带物联网(Narrow Band Internet of Things,NB-IoT)终端(Ultimate-Equipment,UE)为例说明硬件设计颗粒度问题。通常,在 NB-IoT 终端 UE 的

① CoreSight 是 ARM 定义的一个开放体系结构,使 SOC 设计人员能够将其他 IP 内核的调试和跟踪功能添加到 CoreSight 基础结构中。

硬件设计中，选好一款 MCU，一款通信模组，一款 eSIM 卡，然后根据终端 UE 的功能，就开始了 MCU 最小系统设计、通信适配电路设计、eSIM 卡接口设计及其他应用功能设计。这里有许多共性问题可以提取。

(2) 寄存器级编程，软件编程颗粒度低，门槛较高。MCU 参考手册属于寄存器级编程指南，是终端工程师的基本参考资料。例如，要完成一次串行通信，需要涉及波特率寄存器、控制寄存器、状态寄存器、数据寄存器等。一般情况下，工程师会封装寄存器芯片的驱动。即使利用厂家给出的 SDK 编写自己的应用程序，也需要一番周折。此外，工程师面向个性化产品制作，导致产品不具备社会属性，常常被弱化了可移植性。又比如，对 NB-IoT 通信模组，厂家提供的是 AT 指令，要想打通整个通信流程，需要花费一番工夫。

(3) 可移植性弱，更换芯片困难，影响产品升级。一些终端厂家的某一产品使用一种 MCU 芯片多年，有的芯片甚至已经停产，且价格较贵，但由于早期开发可移植性较弱，更换芯片需要较多的研发投入，因此即使新的芯片性价比高，也较难更换。对于 NB-IoT 通信模组，如何做到更换其型号，而原来的软件不变，是值得深入分析思考的。

2. 解决终端开发方式颗粒度低与可移植性弱的基本方法

为解决嵌入式终端 UE 开发方式存在的问题，可提高硬件设计和软件编程的颗粒度，提高可移植性，从而大幅度降低嵌入式系统应用开发的难度。

(1) 提高硬件设计的颗粒度。若能将 MCU 及其硬件最小系统、通信模组及其适配电路、eSIM 卡及其接口电路做成一个整体，则可提高 UE 的硬件开发颗粒度。硬件设计也应该从元件级过渡到以硬件构件为主，辅以少量接口级、保护级元件，以提高硬件设计的颗粒度。

(2) 提高软件编程颗粒度。针对大多数以 MCU 为核心的终端系统，可以通过从面向知识要素角度设计底层驱动构件，把编程颗粒度从寄存器级提高到以知识要素为核心的构件级。以 GPIO 为例阐述这个问题。共性知识要素包括引脚复用成 GPIO 功能、初始化引脚方向；若定义成输出，设置引脚电平；若定义成输入，获得引脚电平，等等。寄存器级编程涉及引脚复用寄存器、数据方向寄存器、数据输出寄存器、引脚状态寄存器等。寄存器级编程因芯片不同，其地址、寄存器名字、功能有所不同。可以面向共性知识要素编程，把寄存器级编程不同之处封装在内部，把编程颗粒度提高到知识要素级。

(3) 提高软硬件可移植性。特定厂家提供 SDK 时，也要注意可移植性。但是由于厂家之间的竞争关系，SDK 的社会属性被弱化。因此，让芯片厂家工程师从共性知识要素角度封装底层硬件驱动，会有些勉为其难。把共性抽象出来，面向知识要素封装，把个性化的寄存器屏蔽在构件内部，才能使得应用层编程具有可移植性。在硬件方面，可遵循硬件构件的设计原则，提高硬件可移植性。

3.3.2 提出 GEC 概念的时机及 GEC 的定义与特点

1. 提出 GEC 概念的时机

要提高编程颗粒度、提高可移植性，可借鉴通用计算机(General Computer)的概念与做法。在一定条件下，可以做通用嵌入式计算机(General Embedded Computer，GEC)，把基本输入输出系统(Basic Input and Output System，BIOS)与用户程序分离开来，实现彻底的工作分工。BIOS 程序与 User 程序独立编译，独立下载。BIOS 服务于 User，类似于 PC 工

作模式。

GEC 概念的实质是把面向寄存器编程提高到面向知识要素编程的水平，提高了编程颗粒度。但是，这样做也会降低实时性。弥补实时性降低这一缺陷的方法是提高芯片的运行时钟频率。目前 MCU 的总线频率是早期 MCU 总线频率的几十倍，甚至几百倍，因此，更高的总线频率给提高编程颗粒度提供了物理支撑。

另一方面是软件构件技术的发展与认识的普及，也为提出 GEC 概念提供了机遇。嵌入式软件开发人员越来越认识到软件工程对嵌入式软件开发的重要支撑作用，也意识到掌握和应用软件工程的基本原理对嵌入式软件的设计、升级、芯片迭代与维护等方面，具有不可或缺的作用。因此，从“零”开始的编程将逐步分化为构件制作与构件使用两个不同层次，也为嵌入式人工智能提供先导基础。

2. GEC 定义及基本特点

一台具有特定功能的通用嵌入式计算机，其特点体现在硬件与软件两个方面。在硬件方面，把 MCU 硬件最小系统及面向具体应用的共性电路封装成一个整体，为用户提供 SOC 级芯片的可重用的硬件实体，并按照硬件构件要求进行原理图绘制、文档撰写及硬件测试用例设计。在软件方面，把嵌入式软件分为 BIOS 程序与 User 程序两部分。BIOS 程序先于 User 程序固化于 MCU 内的非易失存储器（如 Flash）中，启动时，BIOS 程序先运行，随后转向 User 程序。BIOS 提供工作时钟及面向知识要素的底层驱动构件，并为 User 程序提供函数原型级调用接口。

与 MCU 对比，GEC 具有硬件直接可测性、用户软件编程快捷性与可移植性三个基本特点。

（1）GEC 硬件的直接可测性。与一般 MCU 不同，GEC 通电后可直接运行内部 BIOS 程序。BIOS 驱动保留的小灯引脚，实现高低电平切换（在 GEC 上，可直接观察到小灯闪烁）。可利用 AHL-GEC-IDE 开发环境，使用串口连接 GEC，直接将 User 程序写入 GEC。User 程序中包含类似于 PC 程序调试的 printf 语句，通过串口向 PC 输出信息，实现了 GEC 硬件的直接可测性。

（2）GEC 用户软件的编程快捷性。GEC 内部驻留的 BIOS 的作用与 PC 上电过程中 BIOS 的作用类似：完成系统总线时钟初始化；提供一个系统定时器，提供时间设置与获取函数接口；BIOS 内驻留了嵌入式常用驱动，如 GPIO、UART、ADC、Flash、I2C、SPI、PWM 等，并提供了函数原型级调用接口。利用 User 程序的不同框架，用户软件不需要从“零”编起，而是在相应框架的基础上充分应用 BIOS 资源实现快捷编程。

（3）GEC 用户软件的可移植性。GEC 的 BIOS 软件由 GEC 提供者研发完成，随 GEC 芯片提供给用户，即软件被硬件化了，具有通用性。BIOS 驻留了大部分面向知识要素的驱动，提供了函数原型级调用接口。在此基础上编程，只要遵循软件工程的基本原则。GEC 用户软件则具有较高的可移植性。

3.3.3　由 MSPM0L1306 构成的 GEC

本书以 MSPM0L1306 为核心构建一种通用嵌入式计算机，命名为 AHL-MSPM0L1306。

1. AHL-MSPM0L1306 硬件系统基本组成

AHL-MSPM0L1306 内含 MSPM0L1306 芯片及其硬件最小系统、5V-3.3V 转换电路、三色灯、两路 TTL-USB 串口，并引出了 MSPM0L1306 芯片的所有 I/O 口，基本组成见表 3-6。

表 3-6 AHL-MSPM0L1306 的基本组成

序号	部 件	功 能 说 明
1	5V 转 3.3V 电路	实验时通过 Type-C 线连接 PC，将 5V 电压引入本板，在板上转为 3.3V 给 MCU 供电
2	MCU	MSPM0L1306 芯片
3	三色灯	红、绿、蓝
4	TTL-USB	两路 TTL 串口电平转 USB，与工具计算机通信，下载程序，用户串口
5	引出脚编号	1～40，把 MCU 的基本引脚全部再次引出，供应用开发者使用

(1) LED 三色灯。红(R)、绿(G)、蓝(B)三色灯电路原理图如图 3-3 所示。三色灯的型号为 XL-1615RGBC-RF，内含红、绿、蓝三个发光二极管。图中，每个二极管的负极外接 1kΩ 限流电阻后接入 MCU 引脚。只要 MCU 内的程序控制相应引脚输出低电平，对应的发光二极管就亮起来了，从而达到软件控制硬件的目的。

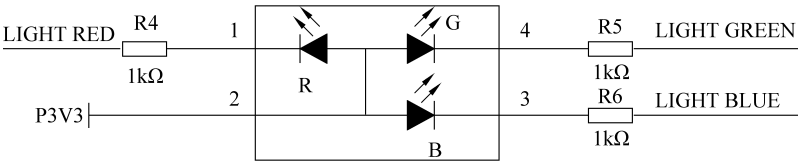

图 3-3 三色灯电路图

【思考】 上网查找三色灯 XL-1615RGBC-RF 的芯片手册，根据手册查看其内部发光二极管的额定电流是多少。为了延长三色灯的使用寿命，限流电阻应该适当增大，还是适当减小？限流电阻增大或减小带来的影响是什么？

(2) TTL-USB 串口。使用 Type-C 线将 GEC 与 PC 的 USB 连接起来，实质是串行通信连接，为了方便 PC 使用 USB 接口模拟串口。TTL-USB 串口提供了两路串口，一路用于下载用户与调试程序，一路供用户使用，第 6 章阐述其编程方法。

2. AHL-MSPM0L1306 的对外引脚

AHL-MSPM0L1306 具有 40 个引出脚，如图 3-4 所示，其中的 UART_Debug、UART_User 两个串口在板内已经通过 TTL-USB 芯片转为 DP、DN 两个引脚，并接入了 Type-C 接口，以便与 PC 直接相连，实现基于串口的程序下载。

MSPM0L1306 的许多引脚具有复用功能，如表 3-7 所示。在进行具体应用的硬件系统设计时可查阅此表。更详细的信息请查阅电子资源\01-Document\MSPM0L1306 芯片资料下的 MSPM0L1306 数据手册及参考手册。

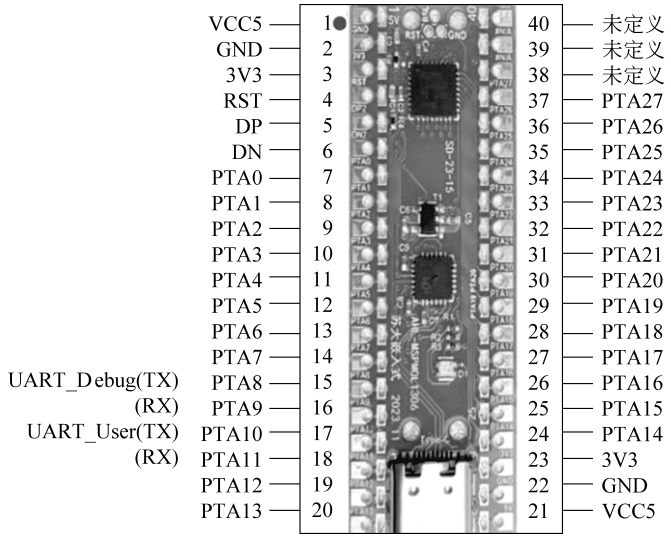

图 3-4　AHL-MSPM0L1306 的对外引脚图

表 3-7　AHL-MSPM0L1306 对外引脚的复用功能

编号	引脚名	复 用 功 能
1	VCC5	5V
2	GND	地
3	3V3	3.3V
4	RST	复位引脚，板内已经上拉到 VCC，外部将其接地后放开即复位
5	DP	（TTL-USB 串口的 DP）
6	DN	（TTL-USB 串口的 DN）
7	PTA0	UART1_TX/I2C0_SDA/TIMG1_C0/SPI0_CS1（默认 BSL[注] I2C_SDA）
8	PTA1	UART1_RX/I2C0_SCL/TIMG1_C1（默认 BSL I2C_SCL）
9	PTA2	TIMG1_C1/SPI0_CS0（已经下拉到地，不再使用）
10	PTA3	TIMG2_C0/SPI0_CS1/UART1_CTS/COMP0_OUT
11	PTA4	TIMG2_C1/SPI0_POCI/UART1_RTS
12	PTA5	TIMG0_C0/SPI0_PICO/FCC_IN
13	PTA6	TIMG0_C1/SPI0_SCK
14	PTA7	COMP0_OUT/CLK_OUT/TIMG1_C0
15	PTA8	UART0_TX/SPI0_CS0/UART1_RTS/TIMG2_C0
16	PTA9	UART0_RX/SPI0_PICO/UART1_CTS/TIMG2_C1/CLK_OUT
17	PTA10	UART1_TX/SPI0_PICI/I2C0_SDA/TIMG4_C0/CLK_OUT
18	PTA11	UART1_RX/SPI0_SCK/I2C0_SCL/TIMG4_C1/COMP0_OUT
19	PTA12	UART0_CTS/TIMG0_C0/FCC_IN

续表

编号	引脚名	复用功能
20	PTA13	UART0_RTS/TIMG0_C1/UART1_RX
21	VCC5	5V
22	GND	地
23	3V3	3.3V
24	PTA14	UART1_CTS/CLK_OUT/UART1_TX/TIMG1_C0
25	PTA15	本板默认 GPIO 接蓝色发光二极管,低电平点亮
26	PTA16	COMP0_OUT/I2C1_SDA/SPI0_POCI/TIMG0_C0/FCC_IN(A8 /OPA1_OUT)
27	PTA17	UART0_TX/I2C1_SCL/SPI0_SCK/TIMG4_C0/SPI0_CS1(OPA1_IN1-)
28	PTA18	UART0_RX/SPI0_PICO/I2C1_SDA/TIMG4_C1(BSL,A7/OPA1_IN0+/GPAMP_IN-)
29	PTA19	SWDIO/I2C1_SDA/SPI0_POCI
30	PTA20	SWCLK/I2C1_SCL/TIMG4_C0(A6/COMP0_IN1+)
31	PTA21	TIMG2_C0/UART0_CTS/UART0_TX(A5/VREF-)
32	PTA22	本板默认 GPIO 接绿色发光二极管,低电平点亮
33	PTA23	UART0_TX/SPI0_CS3/TIMG0_C0/UART_CTS/UART1_TX(VREF+/COMP0_IN1-)
34	PTA24	本板默认 GPIO 接红色发光二极管,低电平点亮
35	PTA25	TIMG4_C1/UART0_TX/SPI0_PICO(A2/OPA0_IN0+)
36	PTA26	TIMG1_C0/UART0_RX/SPI0_POCI(GPAMP_IN+/COMP0_IN0+)
37	PTA27	TIMG1_C1/SPI0_CS3(A0/COMP0_IN0-)
38	未定义	
39	未定义	
40	未定义	

[注] 引导加载程序(Bootstrap Loader,BSL)用于支持进行器件配置以及通过 UART 或 I2C 串行接口对器件存储器进行编程。通过 BSL 对器件存储器和配置的访问受 256 位用户定义的密码保护,如果需要,可以完全禁用器件配置中的 BSL。TI 默认会启用 BSL,以支持将 BSL 用于生产编程。

本章小结

1. 关于初识一个 MCU

从 MCU 型号标识中可获得芯片家族、产品类型、具体特性、引脚数目、Flash 大小、温度范围、封装类型等信息。介绍了内部 RAM 及 Flash 的大小、地址范围,以便设置链接文件,为程序编译及写入做好准备;讲解了中断源及中断向量号,为中断编程做准备。

2. 关于硬件最小系统

使用一个芯片,必须完全理解其硬件最小系统。本章以 32 引脚 VQFN 封装的 MSPM0L1306 芯片为例,介绍了芯片引脚的分类及功能。读者学习本章后,应掌握硬件最

小系统的构成部分，如电源、晶振、复位、写入调试器接口等。掌握电容滤波原理及布板时靠近对应引脚的基本要求。

3. 关于利用 MCU 构建通用嵌入式计算机

引入通用嵌入式计算机概念的目的不仅仅是降低硬件设计复杂度，更重要的是降低软件开发难度。硬件上，MCU 只要供电就可工作，关键是其内部有 BIOS。BIOS 中不仅可以驻留构件，还可以驻留实时操作系统，提供方便灵活的动态命令[①]等。在最小硬件系统基础上，辅以 Wi-Fi 通信、Cat1 通信，可以形成不同应用的 GEC 系列，为嵌入式人工智能与物联网的应用提供技术基础。

习题

1. 举例说明，对照命名格式，从所用 MCU 芯片的芯片型号标识中可以获得哪些信息。

2. 给出所学 MCU 芯片的 RAM 及 Flash 大小、地址范围。

3. 中断的定义是什么？什么是内核中断？什么是非内核中断？给出所学 MCU 芯片的中断个数。

4. 什么是芯片的硬件最小系统？它由哪几个部分组成？简要阐述各部分技术要点。

5. 谈谈你对通用嵌入式计算机的理解。

6. 若不用 MCU 芯片的引脚直接控制三色灯，给出 MCU 引脚通过三极管控制三色灯的电路。

① 动态命令用于扩展嵌入式终端的非预设功能，用于深度嵌入式开发，这里了解即可，不做深入阐述。