第3章

存储器映像、中断源与硬件最小系统

本章导读：本章首先概述以 ARM Cortex-M4 为核心的 STM32L4 系列 MCU，然后给出该 MCU 的存储器映像、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-STM32L431），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 MCU 硬件最小系统，使得 MCU 的内部程序可以运行起来。本章在此基础上进行嵌入式系统的软硬件学习。


视频讲解

3.1 STM32L4 系列 MCU 概述

本节简要概述 STM32L4 系列的 MCU 命名规则、存储器映像以及中断源，其中 MCU 命名规则帮助使用者获得芯片信息；STM32L4 存储器映像是把 M4 内核之外的模块，用类似存储器编址的方式，统一分配地址，关于存储空间的使用，主要记住片内 Flash 区和片内 RAM 区的存储器映像；中断源主要包括 STM32L4 中断源的定义及中断源的分类。

3.1.1 STM32L4 系列 MCU 命名规则

STM32L4 系列 MCU 是意法半导体（ST）公司于 2016 年开始陆续推出基于 M4 内核带 FPU 处理器的超低功耗微控制器，工作频率为 80MHz，与所有 ARM 工具和软件兼容，内部硬件模块主要包括 GPIO、UART、Flash、RAM、SysTick、Timer、PWM、RTC、Incapture、12 位 A/D、SPI、I2C 与 TSC。该系列包含不同的产品线：STM32L4x1（基本型系列），STM32L4x2～6 为不同 USB 体系及 LCD 等模块的扩展型 MCU，满足不同应用的选型需要。

认识一个 MCU，从了解型号含义开始，一般来说，主要包括**芯片家族、产品类型、具体特性、引脚数目、Flash 大小、封装类型以及温度范围等**。

STM32 系列芯片的命名格式为 STM32 X AAA Y B T C，各字段说明如表 3-1 所示，本书所使用的芯片型号为 STM32L431RCT6。对照命名格式，可以从型号获得以下信息：属于 32 位的 MCU，超低功耗型，高性能微控制器，引脚数为 64，Flash 大小为 256KB，封装

形式为64引脚LQFP封装；工作范围为－40℃～＋85℃。

表3-1 STM32系列芯片命令字段说明

字段	说明	取　值
STM32	芯片家族	表示32位MCU
X	产品类型	F表示基础型；L表示超低功耗型；W表示无线系统芯片
AAA	具体特性	取决于产品系列。0xx表示入门级MCU；1xx表示主流MCU；2xx表示高性能MCU；4xx表示高性能微控制器，具有DSP和FPU指令；7xx表示配备ARM Cortex-M7内核的超高性能MCU
Y	引脚数目	T表示36；C表示48；R表示64；V表示100；Z表示144；B表示208；N表示216
B	Flash大小	8表示64KB；C表示256KB；E表示512KB；I表示2048KB
T	封装类型	T表示LQFP封装；H表示BGA封装；I表示UFBGA封装
C	温度范围	6/A表示－40℃～＋85℃；7/B表示－40℃～＋105℃；3/C表示－40℃～＋125℃；D表示－40℃～＋150℃

3.1.2 STM32L4存储器映像

ARM Cortex-M处理器直接寻址空间为4GB，地址范围是0x0000_0000～0xFFFF_FFFF。**所谓存储器映像，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**。哪些地址服务什么资源是MCU生产厂家规定好的，用户只能用而不能改。

STM32L4把M4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部设备，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给出了本书使用的STM32L4系列存储器映像的常用部分内容。

表3-2 STM32L4存储器映射表

32位地址范围	对应内容	说　明
0x0000_0000～0x0003_FFFF	Flash、系统存储器或SRAM	取决于BOOT配置
0x0004_0000～0x07FF_FFFF	保留	—
0x0800_0000～0x0803_FFFF	Flash存储器	256KB
…	…	…
0x2000_0000～0x2000_BFFF	SRAM1[注]	48KB
0x2000_C000～0x2000_FFFF	SRAM2	16KB
0x2001_0000～0x3FFF_FFFF	保留	
0x4000_0000～0x5FFF_FFFF	系统总线和外围总线	GPIO(0x4800_0000～0x4800_1FFF)
…	…	…
0xE000_0000～0xFFFF_FFFF	带FPU的M4内部外部设备	—

注：SRAM区分为两部分是因为SRAM1可以被映射到位带区，参见本书10.3节。

关于存储空间的使用，主要记住片内Flash区和片内RAM区的存储器映像。因为中断向量、程序代码、常数放在片内Flash中，在源程序编译后的链接阶段需要使用的链接文

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，以便生成机器码来准确定位全局变量、静态变量的地址及堆栈指针。

1. 片内 Flash 区的存储器映像

STM32L4 片内 Flash 大小为 256KB，与其他芯片不同，Flash 区的起始地址并不是从 0x0000_0000 开始，而是从 0x0800_0000 开始，其地址范围是 0x0800_0000～0x0803_FFFF。Flash 区中扇区大小为 2KB，扇区总共有 128 个。

2. 片内 RAM 区的存储器映像

STM32L4 片内 RAM 为静态随机存储器（SRAM），大小为 64KB，分成 SRAM1 和 SRAM2，地址范围分别为 0x2000_0000～0x2000_BFFF（48KB）和 0x2000_C000～0x2000_FFFF（16KB），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。大部分编程把它们连续在一起使用，即地址范围是 0x2000_0000～0x2000_FFFF，共 64KB。由于 SRAM1 具有可以被映射到位带区功能，在某些高级编程中用于快速位操作（参见本书 12.6 节）。

STM32L4 芯片堆栈空间的使用方向是向小地址方向进行的，因此将堆栈的栈顶（stack top）设置为 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 的最高地址向低地址方向使用，从而减少重叠错误。

3. 其他存储器映像

与其他芯片不同的是，STM32L4 芯片在 Flash 区前，驻留了 BootLoader 程序，地址范围为 0x0000_0000～0x07FF_FFFF。用户可以根据 BOOT0、BOOT1 引脚的配置，设置程序复位后的启动模式。在 STM32L4 芯片中，BOOT0 为引脚 PTH3，无 BOOT1 时，可用 Flash 选项寄存器（FLASH_OPTR）中的第 23 位（详细内容参考本书第 8 章）与 BOOT0 引脚搭配使用，用于选择 Flash 主存储器、SRAM1 或系统存储器的启动方式。其他存储器映像，如外部设备区存储器映像（如 GPIO 等）、系统保留段存储器映像等，只需了解即可，实际使用时，由芯片头文件给出宏定义。

3.1.3 STM32L4 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 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。

这些引起 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。STM32L4 芯片的中断源分为两类，一类是内核中断，另一类是非内核中断，如表 3-3 所示，这种表共中断编程时备查。内核中断主要是异常中断，也就是说，当出现错误时，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。IRQ 号是非内核中断从 0 开始的编号，因此对内核中断来说，IRQ 号是负值，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即可。

表 3-3 STM32L4 中断向量表

中断类型	IRQ 号	中断向量号	优先级	中 断 源	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
	−12	4	0	内存管理故障	MemManage Interrupt
	−11	5	1	总线故障	Bus Fault Interrupt
	−10	6	2	用法错误	Usage Fault Interrupt
		7～10		保留	
	−5	11	3	SVCall	SV Call Interrupt
	−4	12	4	调试	Debug Interrupt
		13		保留	
	−2	14	5	PendSV	Pend SV Interrupt
	−1	15	6	Systick	SysTick Interrupt
非内核中断	0	16	7	看门狗(Watch Dog)	WWDG
	1	17	8	PVD_PVM	CS Interrupt
	2	18	9	RTC_TAMP_STAM	RTC_TAMP_STAM Interrupt
	3	19	10	RTC_WKUP	RTC_WKUP Interrupt
	4	20	11	Flash	Flash Interrupt
	5	21	12	RCC	RCC Interrupt
	6～10	22～26	13～17	EXTI	EXTIn Interrupt(n 为 1～5)
	11～17	27～33	18～24	DMA1	DMA1 channel n Interrupt
	18	34	25	ADC	ADC Interrupt
	19～22	35～38	26～29	CAN	CANn Interrupt
	23	39	30	EXTI9_5	EXTI9_5 Interrupt
	24～27	40～43	31～34	TIM1	TIM1 Interrupt
	28	44	35	TIM2	TIM2 Interrupt
	29～30	45～46		保留	
	31～34	47～50	38～41	I2C	I2C Interrupt
	35～36	51～52	42～43	SPI	SPI Interrupt
	37～39	53～55	44～46	USART	USARTn Interrupt(n 为 1～3)
	40	56	47	EXTI15_10	EXTI15_10 Interrupt
	41	57	48	RTC_ALArm	RTC_ALArm Interrupt
		58～64		保留	
	49	65	56	SDMMC1	SDMMC1 Interrupt
		66		保留	
	51	67	58	SPI3	SPI3 Interrupt
		68～69		保留	
	54	70	61	TIM6	TIM6 Interrupt
	55	71	62	TIM7	TIM7 Interrupt
	56～60	72～76	63～67	DMA2	DMA2 channel n Interrupt

续表

中断类型	IRQ号	中断向量号	优先级	中 断 源	引 用 名
非内核中断		77～79		保留	
	64	80	71	COMP	COMP Interrupt
	65～66	81～82	72～73	LPTIM	LPTIM Interrupt
		83		保留	
	68	84	75	DMA2_CH6	DMA2 channel 6 Interrupt
	69	85	76	DMA2_CH7	DMA2 channel 7 Interrupt
	70	86	77	LPUART	LPUART Interrupt
	71	87	78	QUADSPI	QUADSPI Interrupt
	72	88	79	I2C3_EV	I2C3_EV Interrupt
	73	89	80	I2C3_ER	I2C3_ER Interrupt
	74	90	81	SAI	SAI Interrupt
		91		保留	
	76	92	83	SWPMI1_IRQn	SWPMI1 Interrupt
	77	93	84	TSC_IRQn	TSC Interrupt
		94～95		保留	
	80	96	87	RNG_IRQn	RNG Interrupt
	81	97	88	FPU_IRQn	Floating Interrupt
	82	98	89	CRS_IRQn	CRS Interrupt

3.2 STM32L4芯片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，也就是找出哪些引脚需要用户提供服务，如电源与地、晶振、程序写入引脚、复位引脚等。

3.2.1 STM32L4芯片的引脚图

本书以64引脚LQFP封装的STM32L431RCT6芯片为例阐述ARM Cortex-M4架构的MCU的编程和应用，图3-1给出的是64引脚LQFP封装的STM32L431的引脚图，芯片的引脚功能请参阅电子资源..\Information文件夹中的数据手册第4章。

芯片引脚可以分为两大部分，一部分是需要用户为它服务的引脚，另一部分是它为用户服务的引脚。

1. 硬件最小系统引脚

硬件最小系统引脚是指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给出了STM32L431的硬件最小系统引脚表。STM32L431芯片电源类引脚在LQFP封装中有11个，芯片使用多组电源引脚为内部电压调节器、I/O引脚驱动、AD转换电路等电路供电，其中内部电压调节器为内核和振荡器等供电。为了提供稳定的电源，MCU内部包含多组电源电路，同时给出多处电源引脚，便于外接滤波电容。为了电源平衡，MCU提供了内部有共同接地点的多处电源引脚，供电路设计使用。

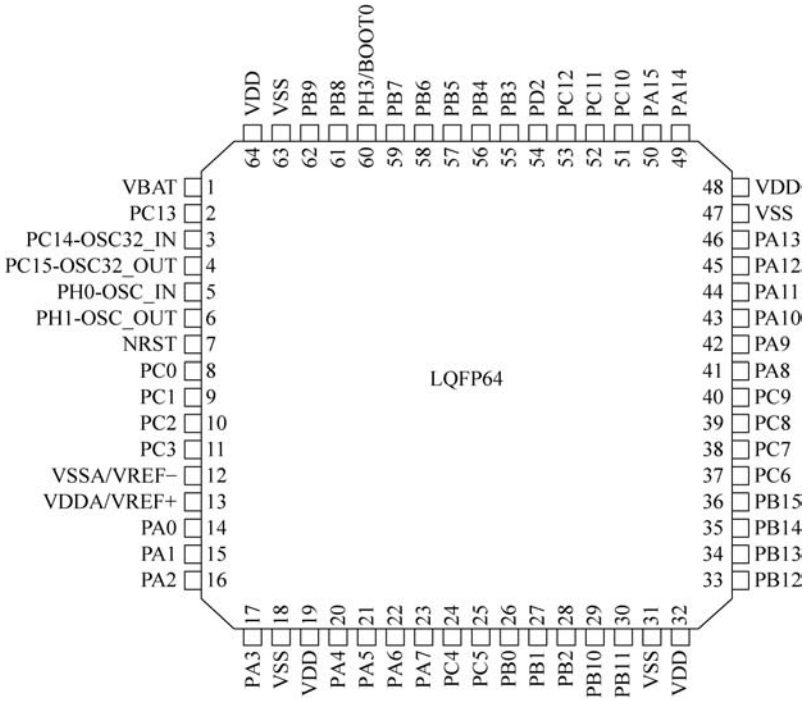

图 3-1　64 引脚 LQFP 封装 STM32L431

表 3-4　STM32L431 的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	19,32,48,64	电源,典型值：3.3V
	VSS	18,31,47,63	地,典型值：0V
	VSSA	12	AD 模块的电源接地,典型值：0V
	VDDA	13	AD 模块的输入电源,典型值：3.3V
	VBAT	1	内部 RTC 备用电源引脚
复位	NRST	7	双向引脚,有内部上拉电阻。作为输入,拉低可使芯片复位
晶振	PTC14、PTC15	3、4	低速无源晶振输入、输出引脚
	PTH0、PTH1	5、6	外部高速无源晶振输入、输出引脚
SWD 接口	SWD_IO/PTA13	46	SWD 数据信号线
	SWD_CLK/PTA14	49	SWD 时钟信号线
启动方式	BOOT0/PTH3	60	程序启动方式控制引脚,BOOT0＝0,从内部 Flash 中程序启动(本书使用)
引脚个数统计			硬件最小系统引脚为 19 个

2. 对外提供服务引脚

除了需要为芯片服务的引脚(硬件最小系统引脚)之外,芯片的其他引脚是向外提供服务的,也可称为 I/O 端口资源类引脚,如表 3-5 所示,这些引脚一般具有多种复用功能。

表 3-5 STM32L4 对外提供 I/O 端口资源类引脚表

端 口 号	引脚个数/个	引 脚 名	硬件最小系统复用引脚
A	16	PTA[0-15]	PTA13、PTA14
B	16	PTB[0-15]	
C	16	PTC[0-15]	PTC14、PTC15
D	1	PTD2	
H	1	PTH3	PTH3
合计	50		

说明：本书中涉及的 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。

STM32L4(64 引脚 LQFP 封装)具有 50 个 I/O 引脚(包含两个 SWD 的引脚；两个外部低速晶振引脚；程序启动方式控制引脚，如 BOOT0 引脚)，这些引脚均具有多个功能，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。

【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 STM32L4 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。使用一个芯片，必须完全理解其硬件最小系统。当 MCU 工作不正常时，在硬件层面，应该检查硬件最小系统中可能出错的元件。芯片要能工作，必须有电源与工作时钟。至于复位电路则提供不掉电情况下 MCU 重新启动的手段。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。这样，硬件最小系统应该把写入器的接口电路也包含在其中。基于这个思路，STM32L4 芯片的硬件最小系统包括电源电路、复位电路、与写入器相连的 SWD 接口电路及可选晶振电路。图 3-2 给出了 STM32L4 硬件最小系统原理图(读者需彻底理解该原理图的基本内涵)。

1. 电源及其滤波电路

MCU 的电源类引脚较多，用来提供足够的电流容量，一些模块也有单独电源与地的引出脚。为了保持芯片电流平衡，电源分布于各边。为了保持进入 MCU 内部的电源稳定，所有电源引出脚必须外接适当的滤波电容，用来抑制电源波动。至于需要外接电容，是由于集成电路制造技术无法在集成电路内部通过光刻的方法制造这些电容。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。

需要强调的是，虽然硬件最小系统原理图中(图 3-2)的许多滤波电容被画在了一起，但实际布板时，需要各自接到靠近芯片的电源与地之间，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，意味着 MCU 一切重新开始，其引脚为 RESET。若复位引脚有效(低电平)，则会

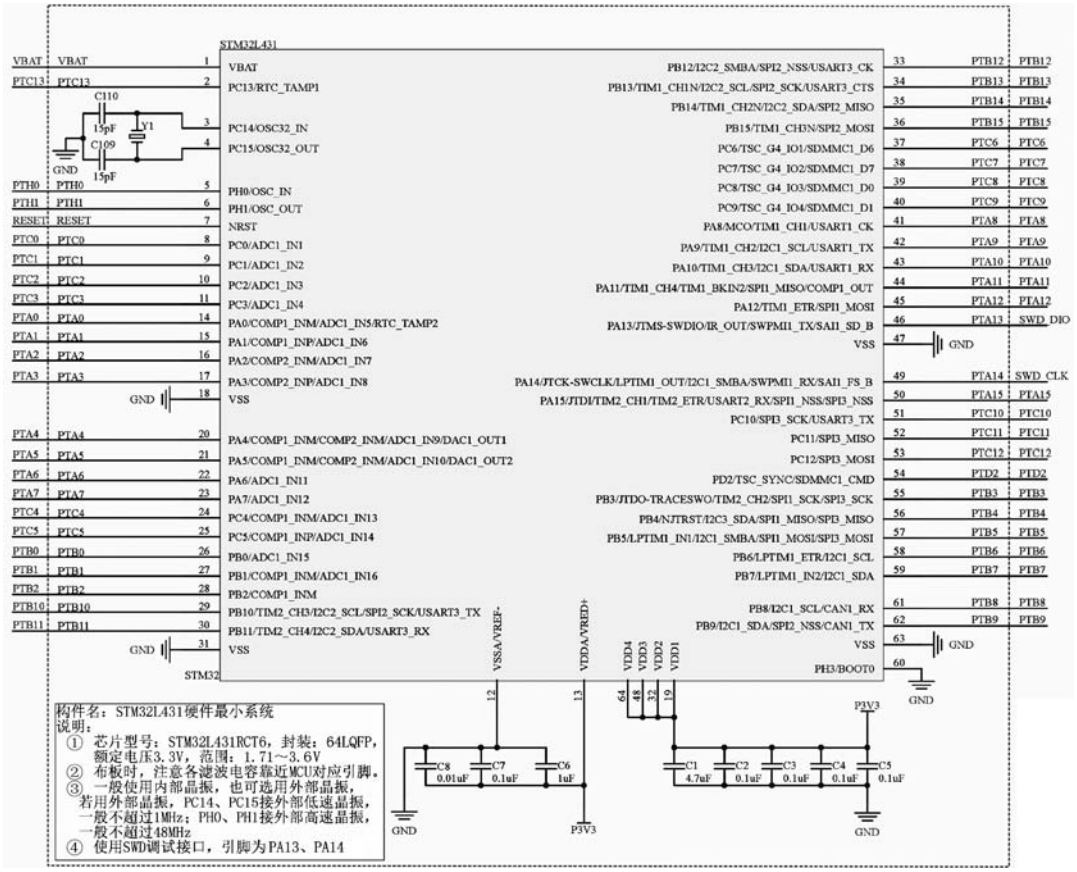

图 3-2 STM32L4 硬件最小系统原理图

引起 MCU 复位。一般芯片的复位引脚内部含有上拉电阻，若外部悬空，则上电的一瞬间，引脚为低电平，随后为高电平，这就是上电复位了。若外接一个按钮一端，按钮的另一端接地，这个按钮就称为**复位按钮**，可以从不同角度对复位进行基本分类。

（1）**外部复位和内部复位**。根据引起 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。外部复位有上电复位、按下“复位”按钮复位。内部复位有看门狗定时器复位、低电压复位、软件复位等。

（2）**冷复位和热复位**。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属于热复位。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。

（3）**异步复位与同步复位**。根据 CPU 响应快慢来区分，复位还可分为异步复位和同步复位。异步复位源的复位请求一般表示一种紧要的事件，因此复位控制逻辑会立即有效，不会等到当前总线周期结束后再复位。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才起作用，这是为了保护数据的完整性。在该总线周期结束后的下一个系统时钟的上升沿时，复位才有效。同步复位源有看门狗定时器、软件等。

【思考】 实际编程时,有哪些方式可判定热复位与冷复位?

3. 晶振电路

计算机的工作需要一个时间基准,这个时间基准由晶振电路提供。STM32L4 芯片可使用内部晶振和外部晶振两种方式为 MCU 提供工作时钟。

STM32L4 系列芯片含有内部时钟源,可以通过编程产生最高 48MHz 时钟频率,供系统总线及各个内部模块使用。CPU 使用的频率是其频率的 2 倍,可达 96MHz。使用内部时钟源时可略去外部晶振电路。

若时钟源需要更高的精度,可自行选用外部晶振。例如,图 3-2 给出外接 8MHz 无源晶振的晶振电路接法,晶振连接在芯片的晶振输入引脚(3 引脚)与晶振输出引脚(4 引脚)之间,根据芯片手册要求,它们均通过 15pF 电容接地。实际上,两个电容有一定偏差,否则晶振不会起振,而电容制造过程中总会有一个微小的偏差,满足起振条件。若使用内部时钟源,这个外接晶振电路就可以不焊接。

芯片启动时,需要运行芯片时钟初始化程序,随后才能正常工作。这个程序比较复杂,放在本书第 12 章阐述。从第 4 章开始的所有程序,均有芯片工作时钟初始化过程,大家先用起来,然后再理解其编程细节。

【思考】 通过查阅资料,了解晶振有哪些类型,简述其工作原理。

4. SWD 接口引脚

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。STM32L4 芯片的 SWD 是基于 CoreSight 架构[①],该架构在限制输出引脚和其他可用资源情况下,提供了最大的灵活性。通过 SWD 接口可以实现程序的下载和调试功能。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。在 STM32L4 芯片中,DIO 为引脚 PTA13,CLK 为引脚 PTA14。

在本书中,SWD 写入器用于写入 BIOS,随后在 BIOS 支持下,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架构,因此本书所附硬件系统不包含 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

一般来说,嵌入式计算机是一个微型计算机,目前嵌入式系统开发模式大多数是从“零”做起,也就是硬件从 MCU(或 MPU)芯片做起,软件从自启动开始,增加了嵌入式系统的学习与开发难度,存在软硬件开发存在颗粒度低、可移植性弱等问题。随着 MCU 性能的不断提高及软件工程概念的普及,给解决这些问题提供了契机。若能向通用计算机那样,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将会形成有力推动。

---

① CoreSight 架构是 ARM 定义的一个开放体系结构,以使 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。

3.3.1 嵌入式终端开发方式存在的问题与解决办法

1. 嵌入式终端开发方式存在的问题

微控制器 MCU 是嵌入式终端的核心,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;许多厂家也给出软件开发包(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。在通常 NB-IoT 终端的硬件设计中,首先选一款 MCU 和一款通信模组,再选一家 eSIM 卡,根据终端的功能,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,这里有许多共性可以抽取。

(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,要完成一个串行通信,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,工程师针对所使用的芯片封装其驱动。即使利用厂家给出的 SDK,也需要一番周折。无论如何,是有一定技术门槛的,也需要花费不少时间。此外,工程师面向个性产品制作,不具备社会属性,常常弱化可移植性。又比如,对于 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,则需要花费一番功夫。

(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个 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,如何做到更换其芯片型号,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

针对嵌入式终端开发方式存在颗粒度低、可移植性弱的问题,必须探讨如何提高硬件颗粒度、如何提高软件颗粒度、如何提高可移植性,做到这三个“提高”,就可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**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。这里以 GPIO 为例阐述这个问题。共性知识要素是:引脚复用成 GPIO 功能、初始化引脚方向;若定义成输出,则设置引脚电平;若定义成输入,则获得引脚电平,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、功能而不同。嵌入式开发人员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,也需要注意可移植性。但是,由于厂家之间的竞争关系,其社会属性被弱化。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,有些勉为其难。科学界必须从共性知识要素本身角度研究这个问题。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,这样才能使得应用层编程具有可移植性。在硬件方面,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、GEC定义与特点

1. 提出GEC概念的时机

要能够做到提高编程颗粒度、提高可移植性,可以借鉴通用计算机(General Computer)的概念与做法,在一定条件下,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。GEC虽然不能涵盖所有嵌入式开发,但可涵盖其中大部分。

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,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。

另外,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方面,也为嵌入式人工智能提供先导基础**。

2. GEC定义及基本特点

一个具有特定功能的通用嵌入式计算机,体现在硬件与软件两方面。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和用户软件可移植性3个基本特点。

(1) **GEC的硬件直接可测性**。与一般MCU不同,GEC类似PC,通电后可直接运行内部BIOS程序,BIOS驱动保留使用的小灯引脚,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PC程序调试的printf语句,通过串口向PC输出信息,实现了GEC的硬件直接可测性。

(2) **GEC的用户软件编程快捷性**。与一般MCU不同,GEC内部驻留的BIOS与PC上电过程类似,需完成系统总线时钟初始化;需提供一个系统定时器、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，而是在相应框架基础上，充分应用BIOS资源，实现快捷编程。

(3) **GEC的用户软件可移植性**。与一般MCU软件不同，GEC的BIOS软件由GEC提供者研发完成，随GEC芯片一起提供给用户，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用户软件就具有较高的可移植性。

3.3.3 由STM32L431芯片构成的GEC

本书以STM32L431为核心构建一种通用嵌入式计算机，命名为AHL-STM32L431，作为本书的主要实验平台，在此基础上可以构建各种类型的GEC。

1. AHL-STM32L431硬件系统基本组成

图3-3给出了AHL-STM32L431硬件图，内含STM32L431芯片及其硬件最小系统、三色灯、复位按钮、温度传感器、触摸区、两路TTL-USB串口，基本组成如表3-6所示。

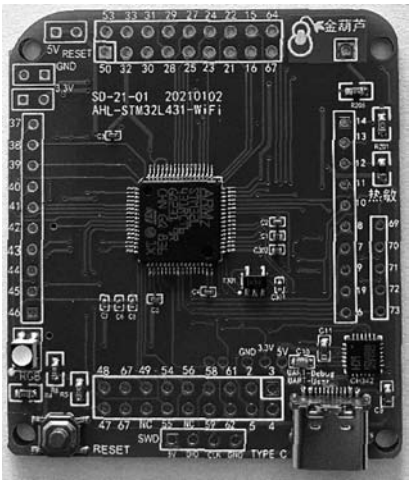

图3-3 AHL-STM32L431硬件图

表3-6 AHL-STM32L431的基本组成

序号	部 件	功 能 说 明
1	三色灯	红、绿、蓝
2	温度传感器	测量环境温度
3	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
4	触摸区	进行初步的触摸实验
5	复位按钮	用户程序不能写入时，按此按钮6次以上，绿灯闪烁，可继续下载用户程序
6	MCU	STM32L431芯片
7	SWD	图3-3中最下方的接口，供利用SWD写入器写入BIOS使用
8	5V转3.3V电路	实验时通过Type-C线接PC，5V引入本板，在板上转为3.3V给MCU供电
9	引出脚编号	1～73，把MCU的基本引脚全部再次引出，供应用开发者使用

下面对 AHL-STM32L431 中的 LED 三色灯、温度传感器、TTL-USB 串口、触摸区和复位按钮等做简要说明。

(1) **LED 三色灯**。红(R)、绿(G)、蓝(B)三色灯电路原理图，如图 3-4 所示。三色灯的型号为 1SC3528VGB01MH08，内含红、绿、蓝 3 个发光二极管。图 3-4 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，对应的发光二极管被点亮，达到软件控制硬件的目的。

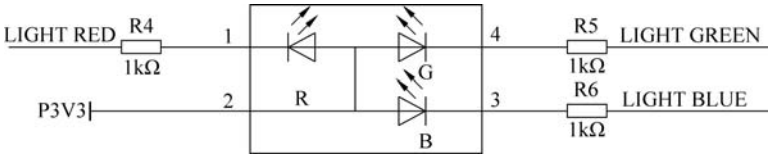

图 3-4　三色灯电路图

【思考】 根据三色灯 1SC3528VGB01MH08 的芯片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，还是适当减少？限流电阻增大或减少带来的影响是什么？

(2) **温度传感器**。AHL-STM32L431 除了其 MCU 内部有温度传感器外，图 3-3 的右侧还有一个区域标有“热敏”字样，这是一个外接温度传感器，即热敏电阻，用于测量环境温度。其电路及编程方法将在本书第 9 章 ADC 一节中阐述。

(3) **TTL-USB 串口**。这个用于使用 Type-C 线将 GEC 与 PC 的 USB 连接起来，实质是串行通信连接，PC 使用 USB 接口模拟串口是为了方便，现在的 PC 和笔记本电脑已经逐步没有串行通信接口，将在本书第 6 章对此进行阐述。这个 TTL-USB 串口提供了两个串口，一个用于下载用户与调试程序，一个供用户使用，本书第 6 章将阐述其编程方法。

(4) **触摸区**。图 3-3 的右上部标有“金葫芦”字样的小铜板，是个可以模拟触摸按键效应的区域，本书第 4 章将阐述其编程方法。

(5) **复位按钮**。图 3-3 的左下部有个按钮，其作用是热复位。特别功能是，在短时间内连续按 6 次以上，GEC 将进入 BIOS 运行状态，可以进行用户程序下载，仅用于解决 GEC 与开发环境连接不上时的写入操作问题。

2. AHL-STM32L431 的对外引脚

AHL-STM32L431 具有 73 个引出脚，如表 3-7 所示。大部分是 MCU 的引脚直接引出，有的引脚功能被固定下来，在进行具体应用的硬件系统设计时查阅此表。

表 3-7　AHL-STM32L431 的引脚复用功能

编号	特定功能	MCU 引脚名	复 用 功 能
1	GND	GND	
2		PTC9	TSC_G4_IO4/SDMMC1_D1
3		PTC8	TSC_G4_IO3/SDMMC1_D0
4		PTC7	TSC_G4_IO2/SDMMC1_D7
5		PTC6	TSC_G4_IO1/SDMMC1_D6
6		PTC5	COMP1_INP/ADC1_IN14/WKUP5/USART3_RX
7		PTC4	COMP1_INM/ADC1_IN13/ USART3_TX

续表

编号	特定功能	MCU引脚名	复用功能
8	用户串口	PTA3	**UART_2_RX(UART_User)**
9	GND	GND	
10	用户串口	PTA2	**UART_2_TX(UART_User)**
11		PTB1	COMP1_INM/ADC1_IN16/TIM1_CH3N/USART3_RTS_DE/LPUART1_RTS_DE/QUADSPI_BK1_IO0/LPTIM2_IN1/
12		PTB0	ADC1_IN15/TIM1_CH2N/SPI1_NSS/USART3_CK/QUADSPI_BK1_IO1/COMP1_OUT/SAI1_EXTCLK
13	调试串口	PTC11	**UART_3_RX (UART_Debug,BIOS保留使用)**
14	调试串口	PTC10	**UART_3_TX (UART_Debug,BIOS保留使用)**
15		PTA7	ADC1_IN12/IM1_CH1N/I2C3_SCL/SPI1_MOSI/QUADSPI_BK1_IO2/COMP2_OUT
16		PTA6	ADC1_IN11/TIM1_BKIN/SPI1_MISO/COMP1_OUT/USART3_CTS/LPUART1_CTS/QUADSPI_BK1_IO3/TIM1_BKIN_COMP2/TIM16_CH1
17	GND	GND	
18	GND	GND	
19	GNSS-ANT		GPS/北斗天线接入(保留)
20	GND	GND	
21		PTA5	COMP1_INM/COMP2_INM/ADC1_IN10/DAC1_OUT2/TIM2_CH1/ TIM2_ETR/SPI1_SCK/LPTIM2_ETR
22		PTA15	JTDI/TIM2_CH1/TIM2_ETR/USART2_RX/SPI1_NSS/SPI3_NSS/USART3_RTS_DE/TSC_G3_IO1/SWPMI1_SUSPEND
23		PTB3	COMP2_INM/JTDO-TRACESWO/TIM2_CH2/SPI1_SCK/SPI3_SCK/USART1_RTS_DE/SAI1_SCK_B
24		PTB4	COMP2_INP/NJTRST/I2C3_SDA/SPI1_MISO/SPI3_MISO/USART1_CTS/TSC_G2_IO1/SAI1_MCLK_B
25		PTB5	LPTIM1_IN1/I2C1_SMBA/SPI1_MOSI/SPI3_MOSI/USART1_CK/TSC_G2_IO2/COMP2_OUT/SAI1_SD_B/TIM16_BKIN
26	GND	GND	
27		PTB6	COMP2_INP/LPTIM1_ETR/I2C1_SCL/USART1_TX/TSC_G2_IO3/SAI1_FS_B/TIM16_CH1N
28		PTB14	TIM1_CH2N/I2C2_SDA/SPI2_MISO/USART3_RTS_DE/TSC_G1_IO3/SWPMI1_RX/SAI1_MCLK_A/TIM15_CH1
29		PTB15	RTC_REFIN/TIM1_CH3N/SPI2_MOSI/TSC_G1_IO4/SWPMI1_SUSPEND/SAI1_SD_A/TIM15_CH2
30		PTB13	TIM1_CH1N/I2C2_SCL/SPI2_SCK/USART3_CTS/LPUART1_CTS/TSC_G1_IO2/SWPMI1_TX/SAI1_SCK_A/TIM15_CH1N
31		PTB12	TIM1_BKIN/TIM1_BKIN_COMP2/I2C2_SMBA/SPI2_NSS/USART3_CK/LPUART1_RTS_DE/TSC_G1_IO1/SWPMI1_IO/SAI1_FS_A/TIM15_BKIN
32	GND	GND	
33	保留		保留无线通信模组的天线接入使用

续表

编号	特定功能	MCU引脚名	复用功能
34	GND	GND	
35	GND	GND	
36	P3V3_ME		输出检测用3.3V
37		PTA8	MCO/TIM1_CH1/USART1_CK/SWPMI1_IO/SAI1_SCK_A/LPTIM2_OUT/
38		PTB11	TIM2_CH4/I2C2_SDA/USART3_RX/LPUART1_TX/QUADSPI_BK1_NCS/COMP2_OUT
39		PTB10	TIM2_CH3/I2C2_SCL/SPI2_SCK/USART3_TX/LPUART1_RX/TSC_SYNC/QUADSPI_CLK/COMP1_OUT/SAI1_SCK_A
40		PTA4	COMP1_INM/COMP2_INM/ADC1_IN9/DAC1_OUT1/SPI1_NSS/SPI3_NSS/USART2_CK/SAI1_FS_B/LPTIM2_OUT
41		PTH1	OSC_OUT
42		PTH0	OSC_IN
43	GND	GND	
44		PTA1	OPAMP1_VINM/COMP1_INP/ADC1_IN6/TIM2_CH2/I2C1_SMBA/SPI1_SCK/USART2_RTS_DE/TIM15_CH1N
45		PTA0	OPAMP1_VINP/COMP1_INM/ADC1_IN5/RTC_TAMP2/WKUP1/TIM2_CH1/USART2_CTS/COMP1_OUT/SAI1_EXTCLK/TIM2_ETR
46		PTC1	ADC1_IN2/LPTIM1_OUT/I2C3_SDA/LPUART1_TX
47		PTC0	ADC1_IN1/LPTIM1_IN1/I2C3_SCL/LPUART1_RX/LPTIM2_IN1
48		PTC2	ADC1_IN3/LPTIM1_IN2/SPI2_MISO
49		PTC3	ADC1_IN4/LPTIM1_ETR/SPI2_MOSI，SAI1_SD_A/LPTIM2_ETR
50	RESET	NRST	
51	GND	GND	
52	GND	GND	
53	保留		
54	蓝灯引脚	PTB9	IR_OUT/I2C1_SDA/SPI2_NSS/CAN1_TX/SDMMC1_D5/SAI1_FS_A
55	绿灯引脚	PTB8	I2C1_SCL/CAN1_RX/SDMMC1_D4/SAI1_MCLK_A/TIM16_CH1
56	红灯引脚	PTB7	COMP2_INM/PVD_IN/LPTIM1_IN2/I2C1_SDA/USART1_RX/TSC_G2_IO4
57	RST	NRST	
58	SWD_DIO	PTA13	JTMS-SWDIO/IR_OUT/SWPMI1_TX/SAI1_SD_B
59		PTD2	USART3_RTS_DE/TSC_SYNC/SDMMC1_CMD
60	GND	GND	
61		PTC12	SPI3_MOSI/USART3_CK/TSC_G3_IO4/SDMMC1_CK
62	SWD_CLK	PTA14	JTCK-SWDCLK/LPTIM1_OUT/I2C1_SMBA/SWPMI1_RX/SAI1_FS_B
63	GND	GND	
64	3.3V		3.3V输出(150mA)

续表

编号	特定功能	MCU引脚名	复用功能
65	GND	GND	
66	5V输入		
67	5V输入		
68	GND	GND	
69		PTC13	RTC_TAMP1/RTC_TS/RTC_OUT/WKUP2
70		PTA12	TIM1_ETR/PI1_MOSI/USART1_RTS_DE/CAN1_TX
71		PTA11	TIM1_CH4/TIM1_BKIN2/SPI1_MISO/COMP1_OUT/USART1_CTS/CAN1_RX/TIM1_BKIN2_COMP1
72		PTA10	TIM1_CH3/I2C1_SDA/USART1_RX/SAI1_SD_A
73		PTA9	TIM1_CH2/I2C1_SCL/USART1_TX/SAI1_FS_A/TIM15_BKIN

本章小结

1. 关于初识一个MCU

初识一个MCU时,首先要从认识型号标识开始,可以从型号标识中获得芯片家族、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,这些信息是购买芯片的基本知识;其次了解内部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;最后了解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

一个芯片的硬件最小系统是指可以使内部程序运行所必需的最低规模的外围电路,也可以包括写入器接口电路。使用一个芯片,必须完全理解其硬件最小系统。硬件最小系统引脚是必须为芯片提供服务的引脚,包括电源、晶振、复位、SWD接口等,做好这些服务之后,其他引脚就为用户提供服务了。硬件最小系统电路中着重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是为了降低硬件设计复杂度,更重要目的是降低软件开发难度。硬件上,使其只要供电就可工作,关键是其内部有BIOS。BIOS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],等等。在最小的硬件系统基础上,辅以WiFi、NB-IoT、5G等,可形成不同应用的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。

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中,这里了解即可,不做深入阐述。

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 LED 三色灯，给出 MCU 引脚通过三极管控制 LED 三色灯的电路。