1.5 基于硬件构件的嵌入式底层软件构件的编程方法

嵌入式系统是软件与硬件的综合体，硬件设计和软件设计相辅相成。嵌入式系统中的驱动程序是直接工作在各种硬件设备上的软件，是硬件和高层软件之间的桥梁。正是通过驱动程序，各种硬件设备才能正常运行，达到既定的工作效果。

1.5.1 嵌入式硬件构件和软件构件的层次模型

嵌入式软件构件（Embedded Software Component, ESC）是实现一定嵌入式系统功能的一组封装的、规范的、可重用的、具有嵌入特性的软件单元，是组织嵌入式系统的功能单位。

嵌入式软件构件分为高层软件构件和底层软件构件（以下简称高层构件和底层构件）。高层构件与硬件无关。而底层构件与硬件密不可分，是硬件驱动程序的封装。前面提到，在硬件构件中，核心构件为 MCU 的最小系统。通常，MCU 内部包含有 GPIO（即通用 IO）口和一些内置功能模块，可将通用 I/O 口的驱动程序封装为 GPIO 构件，各内置功能模块的驱动程序封装为功能构件，如芯片内含模块的功能构件有串行通信构件、Flash 构件、定时器构件等。

在硬件构件层中，相对于核心构件而言，中间构件和终端构件是核心构件的“外设”。由这些“外设”的驱动程序封装而成的软件构件称为底层外设构件。注意，并不是所有的中间构件和终端构件都可以作为编程对象。例如：键盘、LED、LCD 等硬件构件与编程有关，而电平转换硬件构件就与编程无关，因而不存在相应的底层驱动程序，当然也就没有相应的软件构件。嵌入式硬件构件与软件构件的层次模型如图 4-5 所示。

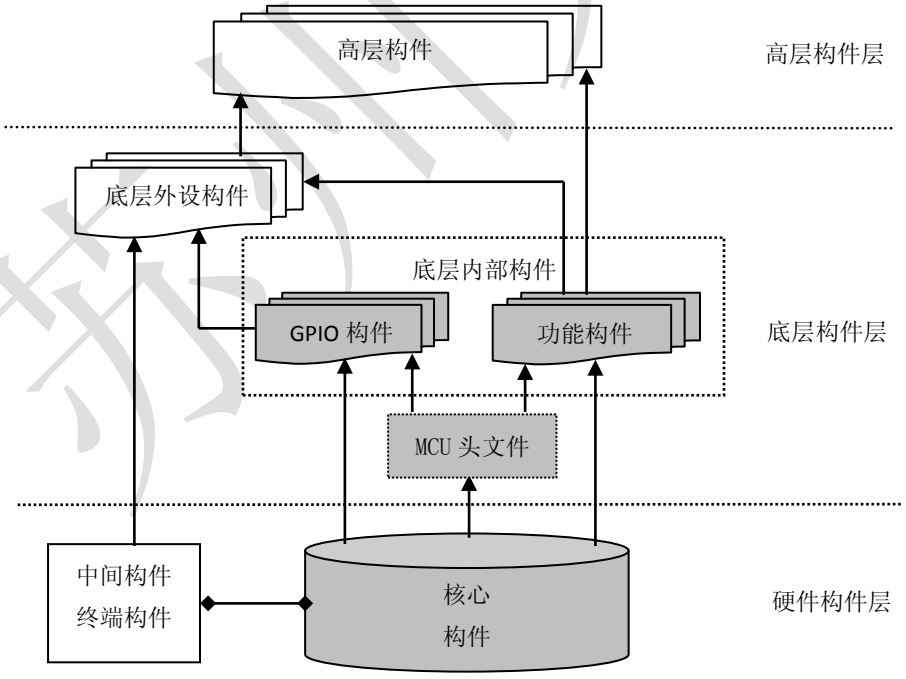


图 1-6 嵌入式硬件构件与软件构件的层次模型

由图 1-6 可看出，底层外设构件可以调用底层内部构件，如 LCD 构件可以调用 GPIO 构件、PCF8563 构件（时钟构件）可以调用 I2C 构件等。而高层构件可以调用底层外设构件和底层内部构件中的功能构件，而不能直接调用 GPIO 构件。另外，考虑到几乎所有的底层内部构件都涉及到 MCU 各种寄存器的使用，因此将 MCU 的所有寄存器定义组织在一起，形成 MCU 头文件，以便其它构件头文件中包含该头文件。

1.5.2 底层构件的实现方法与编程思想

底层构件是与硬件直接打交道的软件，由头文件和源程序文件两部分组成。

头文件中的内容主要有：包含下层构件头文件的#include 语句、用以描述构件属性的宏定义语句以及对外接口函数原型说明。在头文件中使用函数原型，对于建立代码模块和外部接口的规范，便于他人使用，都是很有帮助的。使用这些函数的用户，不需要查找源代码去了解参数的具体类型，直接查看函数原型即可。

源程序文件中存放构件的内部函数和外部函数的定义，即函数的实现代码，以完成函数所要实现的功能。

在对底层构件进行设计时，最关键的工作是要对构件的共性和个性进行分析，抽取出构件的属性和对外接口函数。**尽量做到：当一个底层构件应用到不同系统中时，仅需修改构件的头文件，对于构件的源程序文件则不必修改或改动很小**。

例如，串行通信模块 SCI 是大多数 MCU 都具有的内部模块。仔细分析各种 MCU 串行通信程序发现：在查询方式下，各种 MCU 都是根据状态寄存器中的两个标志位来判断是否接收到数据和数据是否发送完毕，这就是 SCI 模块的共性。对于不同的 MCU，该状态寄存器的名称可能不同，这两个标志位的位号也有可能不同。此外，用以设置波特率、通信格式、是否校验、是否允许中断等参数的寄存器也不同，这就是 SCI 模块的个性。分析出了共性和个性之后，就可以抽取出 SCI 构件的属性和操作，编制构件头文件和程序文件了。有关内容参见第五章。

在编写构件时，一般应注意以下几方面的内容：

（1）构件的头文件和源程序文件的主文件名一致，且为构件名。

（2）属性和操作的命名统一以构件名开头。这样做的好处是：当使用底层构件组装软件系统时，避免构件之间出现同名现象。同时，名称要使人有“顾名思义”的效果。

（3）对 MCU 内的模块寄存器名和端口名进行重定义，在其它的代码里面都将使用宏名对模块寄存器和端口进行操作。这样，当底层驱动程序移植到其它 MCU 时，只要修改重定义语句就可以了。

（4）内部函数与外部函数要设计合理，函数参数个数及类型要考虑全面。内部函数仅提供给同一构件中的其它内部函数或外部函数调用，作用域仅限于定义该函数的文件。外部函数是对外接口函数，供上层应用程序调用。在定义外部函数时，应该对函数名、函数功能、入口参数、函数返回值、使用说明、函数适用范围等进行详细描述，以增强程序的可读性。上层应用程序不能直接对构件的属性进行读取或设置，必须借助于该构件提供的接口操作函数来实现。

（5）应用程序在使用底层构件时，严格禁止通过全局变量来传递参数，所有的数据传递都要通过函数的形式参数来接收。这样做不但使得接口简洁，更加避免了全局变量可能引发的安全隐患。

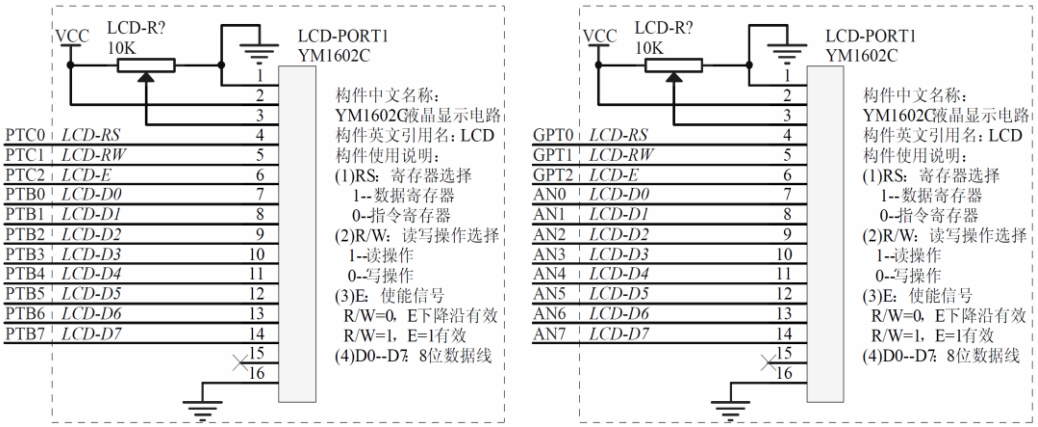
1.5.3 硬件构件及底层软件构件的重用与移植方法

重用是指在一个系统中，同一构件可被重复使用多次。移植是指将一个系统中使用到的构件应用到另外一个系统中。

1．硬件构件的重用与移植

对于以单 MCU 为核心的嵌入式应用系统而言，当用硬件构件“组装”硬件系统时，核心构件（即最小系统）有且只有一个，而中间构件和终端构件可有多个，并且相同类型的构

件可出现多次。下面以终端构件 LCD 为例，介绍硬件构件的移植方法。



(a) LCD 构件在系统 A 中的应用　　(b) LCD 构件在系统 B 中的应用

图 1-7 LCD 构件在实际系统中的应用

在应用系统 A 中，若 LCD 的数据线（LCD-D0～LCD-D7）与芯片 AW60（8 位 MCU）芯片的通用 IO 口的 B 口相连，C 口作为 LCD 的控制信号传送口，其中，LCD 寄存器选择信号 LCD-RS 与 C 口第 0 引脚连接，读写信号 LCD-RW 与 C 口第 1 引脚连接，使能信号 LCD_E 与 C 口第 2 引脚连接，则 LCD 硬件构件实例如图 1-7(a)所示。虚线框左边的文字（如 PTC0、PTC1 等）为接口网标，虚线框右边的文字（如 LCD-RS、LCD-RW 等）为接口注释。

在应用系统 B 中，若 LCD 的数据线（LCD-D0～LCD-D7）与 MCF52233（32 位 MCU）芯片的通用 IO 口的 AN 口相连，TA 口的第 0、1、2 引脚分别作为寄存器选择信号 LCD-RS、读写信号 LCD-RW、使能信号 LCD_E，则 LCD 硬件构件实例如图 1-7(b)所示。

2．底层构件的移植

当一个已设计好的底层构件移植到另外一个嵌入式系统中时，其头文件和程序文件是否需要改动呢？这要视具体情况而定。例如：系统的核心构件发生改变（即 MCU 型号改变）时，底层内部构件头文件和某些对外接口函数也要随之改变，例如模块初始化函数。

而对于外接硬件构件，希望不改动程序文件，而只改动头文件，那么，头文件就必须充分设计。以 LCD 构件为例，与图 1-7(a)相对应的底层构件头文件 LCD.h 可如下编写。

```
#include “GPIO.h”            //包含 GPIO 构件头文件
#define   LCD_Data      GPIO_PORTB      //显示数据传送口
#define   LCD_Ctrl      GPIO_PORTC      //控制信号传送口
#define   LCD_RS        0               //寄存器选择信号
#define   LCD_RW        1               //读写信号
#define   LCD_E         2               //使能信号
void LCD_Init(void);                    //液晶显示初始化
void LCD_Fill(INT8U a);                 //填充 LCD，可实现清屏
void LCD_PutDot(INT8U x,INT8U y);  //画点
void LCD_PutLine(INT8U x1,INT8U y1,INT8U x2,INT8U y2);      //画线
void LCD_PutChar(INT8U a);              //显示一个字符
void LCD_PutString(INT8U str[]);        //显示一串字符
```

当 LCD 硬件构件发生如图 1-7(b)所示的移植时，显示数据传送口和控制信号传送口发生了改变，此时，只要将上面的第 1 和第 2 条宏定义语句修改成：

```
#define   LCD_Data      GPIO_PORTAN           //显示数据传送口
```

```
#define    LCD_Ctrl        GPIO_PORTTA             //控制信号传送口
```

必须申明的是，本书提出构件化设计方法的目的是，在进行软硬件移植时，设计人员所做的改动要尽量小，而不是不作任何改动。希望改动在头文件中进行，而不希望改动程序文件，事实上，不作任何改动是不现实的。

第2章 Kinetis概述与MK60N512VMD100硬件最小系统

本章简要概述 Kinetis 系列微处理器，给出 K60 系列微控制器存储器映像与编程结构、寻址方式、异常向量表；重点指出学习一个新 MCU 芯片比较快速的学习过程；给出 K60N512 的引脚功能与硬件最小系统电路。本章的重点是：存储空间的地址分配、中断结构、硬件最小系统电路。

2.1 ARM公司的发展史及ARM架构的发展

1.ARM 公司的历史

ARM 公司是世界领先的半导体知识产权供应商，其提供的产品是数字电子产品的核心。今天 ARM 为世界上四分之一的电子产品提供技术基础，被半导体及电子业界评为过去 30 年全球最有影响力的 10 家公司之一。ARM 公司于 1990 年 11 月成立于英国，原名 Advanced RISC Machine 有限公司，是苹果电脑、Acorn 电脑集团和 VLSI Technology 的合资企业。Acorn 曾推出世界首个商用单芯片 RISC 处理器，而苹果电脑当时希望将 RISC 技术应用于自身系统，ARM 的微处理器新标准因此应运而生。ARM 成功地研制了首个低成本 RISC 架构，迅速在市场上崭露头角，与此同时 RISC 结构的竞争对手都着眼于提高性能发展高端工作站处理器的 RISC 结构。

1991 年 ARM 推出首个嵌入式 RISC 核心——ARM6™ 系列处理器，不久 VLSI 率先获得授权，一年后夏普和 GEC Plessey 也成为授权用户，1993 年德州仪器和 Cirrus Logic 亦签署了授权协议。从此 ARM 的知识产权产品和授权用户急剧扩大，1993 年 Nippon Investment and Finance（NIF）成为 ARM 股东后，ARM 开始向全球拓展，分别在亚洲、美国和欧洲设立了办事处。1998 年 4 月 ARM 在伦敦证券交易所纳斯达克交易所上市。

ARM Holdings 在半导体革新过程中初露峥嵘，被 Dataquest 誉为世界第一的知识产权供应商。20 世纪 90 年代初 ARM 率先推出 32 位 RISC 微处理器芯片系统（SoC）知识产权公开授权概念。

现在，ARM 芯片的出货量每年都比上一年多 20 亿片以上。不像很多其他的半导体公司，ARM 从不制造和销售具体的处理器芯片，而是把处理器的设计授权给相关的商务合作伙伴，让他们根据自己的强项设计具体的芯片。更重要的是 ARM 开创了电子新纪元：采用 ARM 技术的微处理器遍及各类电子产品，在汽车、消费、娱乐、成像、工业控制、网络、存储、安保和无线等市场中，ARM 技术无处不在。

2.ARM 架构发展史

ARM 的设计历史起源是 Acorn 电脑公司（Acorn Computers Ltd）于 1983 年开始的开发计划。这个团队由 Roger Wilson 和 Steve Furber 带领，着手开发一种新架构，类似进阶的 MOS Technology 6502 处理器。设计团队在 1985 年时开发出 ARM1 Sample 版，而首颗“真

正”的产能型 ARM2 于次年量产。ARM2 具有 32 位的数据总线、26 位的寻址空间，并提供 64MB 的寻址范围与 16 个 32 位的寄存器。这些寄存器其中有一个作为程序计数器，其前面 6 位和后面 2 位用来保存处理器状态标记。ARM2 可能是全世界最简单实用的 32 位微处理器，其仅容纳了 30,000 个晶体管。之所以精简的原因在于它不含微码，而与现今大多数的 CPU 不同，它没有包含任何高速缓存。这个精简的特色使它只需消耗很少的电能，却能发挥比 Intel 80286 更好的效能。后继的处理器 ARM3 更备有 4KB 的告诉缓存，使它能发挥更佳的效能。

1991 年，ARM 技术首次发布，然后苹果电脑使用 ARM6 架构的 ARM610 当作其 Apple Newton PDA 的基础。1994 年，Acorn 使用 ARM610 作为其 RISC PC 电脑内的 CPU。

随着 ARM2 到 ARM6 的技术进阶发展，内核部分却大多维持一样的大小。ARM2 有 30,000 个晶体管，但 ARM6 只增长到 35,000。主要概念是以 ODM 的方式，使 ARM 核心能搭配一些选配的零件而制成一颗完整的 CPU，而且可在现有的晶圆制造厂里制作并以低成本的方式达到很大的效能。

随着技术的不断发展，ARM 公司不断推陈出新，发展至今最新的是 ARMv7 架构，要说明的是，架构版本号和名字中的数字并不是一码事。比如，ARM7TDMI 是基于 ARMv4T 架构的（T 表示支持 Thumb 指令）。ARMv7 架构采用 Thumb-2 技术，是在 ARM 的 Thumb 代码压缩技术的基础上发展起来的，并且保持了对现存 ARM 解决方案的完整代码兼容性。Thumb-2 技术比纯 32 位代码少使用 31%的内存，减小了系统开销。同时能够提供比已有的基于 Thumb 技术的解决方案高出 38%的性能。ARMv7 架构还采用了 NEON 技术，将 DSP 和媒体处理能力提高了近 4 倍，并支持改良的浮点运算，满足下一代 3D 图形、游戏物理应用以及传统嵌入式控制应用的需求。此外，在这个版本中，内核架构首次从单一款式变成三种款式。

- 款式 A：设计用于高性能的“开方应用平台”——越来越接近电脑。
- 款式 R：用于高端的嵌入式系统，尤其是那些带有实时要求的——既要快又要实时。
- 款式 M：用于深度嵌入的、单片机风格的系统中——本书的主角。

让我们再近距离的考察这三种模式：

- 款式 A（ARMv7-A）：需要运行复杂应用程序的“应用处理器”。支持大型嵌入式操作系统，比如 Symbian（诺基亚智能手机用）、Linux，以及微软的 Windows CE 和智能手机操作系统 Windows Mobile。
- 款式 R（ARMv7-R）：硬实时且高性能的处理器。目标是高端实时市场。像高档轿车的组件、大型发电机控制器、机器手臂控制器等，它们使用的处理器不但要很好很强大，还要极其可靠，对事件的反应也要极其敏捷。
- 款式 M（ARMv7-M）：认准了旧时代单片机应用而量身定制。在这些应用中，尤其是对于实时控制系统，低成本、低功耗、极速中断反应以及高处理效率都是至关重要的。Cortex 系列是 v7 架构的第一次亮相。其中 Cortex-M4 就是按款式 M 设计的。

ARM 架构发展版本如表 2-1 所示。

表 2-0-1 ARM 架构发展版本

ARM 版本系列	架构	内核名	特色	高速缓存（I/D）/MMU	常规 MIPS 与 MHz	代表芯片与应用
ARM1	ARM v1	ARM1		无		
ARM2	ARM v2	ARM2	Architecture2	无	4MIPS@8 MHZ	Acorn Architmedes,Chessmachine

ARM 版本系列	架构	内核名	特色	高速缓存（I/D）/MMU	常规 MIPS 与 MHz	代表芯片与应用
			加入了 MUL（乘法）指令			
	ARM v2a	ARM250	集成 MEMC(MMU),图像与 I/O 处理器。Architecture2a 加入了 SWP 和 SWPB（置换）指令	无，MEMC1a	7MIPS@12MHz	Acorn Architmedes
ARM3	ARM v2a	ARM2a	首次在 ARM 架构上使用处理器高速缓存	均为 4KB	12MIPS@25MHz	Acorn Architmedes
ARM6	ARM v3	ARM610	V3 架构首创支援寻址 32 位的内存(针对 26 位）	均为 4KB	28MIPS@33MHz	Acorn Risc PC 600, Apple Newton
ARM7	ARM v3					
ARM7 TDMI	ARM v4T	ARM7TDMI(-S)	三级流水线	无	15MIPS@16.8MHZz	Game Boy Advance, Nintendo DS,iPod
		ARM710T		均为 8KB,MMU	36MIPS@40MHz	Acorn Risc PC 700,Psion5 series, Apple eMate 300
		ARM720T		均为 8KB,MMU	60MIPS@59.8MHz	Zipit
		ARM740T		MPU		
	ARM v5TEJ	ARM7EJ-S	Jazelle DBX	无		
Strong ARM	ARM v4					
ARM8	ARM v4					
ARM9 TDMI	ARM v4T	ARM9TDMI	五级流水线	无		
		ARM920T		16KB/16KB, MMU	200MIPS @180MHz	Armadillo,GP32,GP2X(第一颗内核)，TapwareZodiac(Motorola/Freescale i.MX1)
		ARM922T		8KB/8KB,M		

ARM 版本系列	架构	内核名	特色	高速缓存（I/D）/MMU	常规 MIPS 与 MHz	代表芯片与应用
				MU		
		ARM940T		4KB/4KB,MMU		GP2X(第二颗内核)
ARM9E	ARM v5TE	ARM946E-S		可变动，tightly coupled memories,MPU		Nintendo DS,Nokia N-Gage Conexant 802.11 chips
		ARM966E-S		无高速缓存，TCMS		ST Micro STR91xF
		ARM968E-S		无高速缓存，TCMS		
	ARM v5TEJ	ARM926EJ-S	Jazelle DBX	可变动，TCMS,MMU	200MIPS @200MHz	移动电话：Sony Ericsson(K、W 系列)，Siemens 和 Benq(x65 系列)
	ARM v5TE	ARM996HS	无振荡器处理器	无高速缓存，TCMS,MMU		
ARM10E	ARM v5TE	ARM1020E	(VFP),六级流水线	32KB/32KB,MMU		
		ARM1022E	(VFP)	16KB/16KB,MMU		
	ARM v5TEJ	ARM1026EJ-S	Jazelle DBX	可变动，MMU，可包含 MPU		
Xscale	ARM v5TE	80200/IOP310/IOP315	I/O 处理器			
		80219			400/600MHz	Thecus N2100
		IOP321			600 BogoMips @600MHz	Iyonix
		IOP33x				
		IOP34x	1~2 核，RAID 加速器	32K/32KL1,512KL2,MMU		
		PXA210/PXA250	应用处理器，七级流水线			Zaurus SL-5600
		PXA255		32KB/32KB,MMU	400 BogoMips @400MHz	Gumstix,Palm TungstenE2

ARM 版本系列	架构	内核名	特色	高速缓存（I/D）/MMU	常规 MIPS 与 MHz	代表芯片与应用
		PXA26x			可达 400MHz	Palm Tungsten T3
		PXA27x			800MIPS @624MHz	HTC Universal,Zaurus SL-C1000,3000,3100,3200,Dell Axim x30,x50 和 x51 系列
		PXA800(E)F				
		Monahans			1000MIPS @1.25GHz	Mavell PXA300/PXA310/PXA320,Max frequency:PXA300@624MHz, PXA310/PXA320@806MHz
		PXA900				Blackberry 8700 Blackberry Pearl(8100)
ARM11	ARM v6	ARM1136J(F)-S	SIMD,Jazelle DBX,(VFP),八级流水线	可变动，MMU	532-665MHz(i.MX31 SoC)	Nokia N93,Microsoft Zune(Freescale i.MX31),Nokia N800
	ARM v6T2	ARM1156T2(F)-S	SIMD,Thumb-2,(VFP),九级流水线	可变动，MPU		
	ARM v6KZ	ARM1176JZ(F)-S	SIMD,Jazelle,(VFP)	可变动，MMU+TrustZone		Freescale i.MX37
	ARM v6K	ARM11 MPCore	1~4 核对称多处理器，SIMD,Jazelle,(VFP)	可变动，MMU		
Cortex	ARM v7-A	Cortex-A8	Application profile,VFP,NEON,Jazelle RCT,Thumb-2,13-stage pipeline	可变动(L1+L2),MMU+TrustZone	2.0DMIPS/MHz 从 600MHz 到超过 1GHz 的速度	Freescale i.MX51,Texas Instruments OMAP3
		Cortex-A9				
		Cortex-A9 MPCore				
	ARM v7-R	Cortex-R4(F)	Embedded profile,(FPU)	可变动高速缓存，MMU 可选配	600 DMIPS	
	ARM	Cortex-M3	Microcontrol	无高速缓存，	120	Luminary Micro 微控制器家

ARM 版本系列	架构	内核名	特色	高速缓存（I/D）/MMU	常规 MIPS 与 MHz	代表芯片与应用
	v7-M		ler profile	(MPU)	DMIPS @ 100MHz	族
	ARM v6-M	Cortex-M0				
		Cortex-M1				
	ARM v7-M E	Cortex-M4		Optional 8region MPU with sub regions and background region	1.25DMIPS/MHz	

3.ARM 处理器命名方法

以前，ARM 使用一种基于数字的命名法。在早期(1990s)，还在数字后面添加字母后缀，用来进一步明细该处理器支持的特性。就拿 ARM7TDMI 来说，T 代表 Thumb 指令集，D 是说支持 JTAG 调试(Debugging)，M 意指快速乘法器，I 则对应一个嵌入式 ICE 模块。后来，这 4 项基本功能成了任何新产品的标配，于是就不再使用这 4 个后缀——相当于默许了。但是新的后缀不断加入，包括定义存储器接口的，定义高速缓存的，以及定义“紧耦合存储器(TCM)”的，于是形成了新一套命名法，这套命名法也是一直在使用的。

到了架构 7 时代，ARM 改革了一度使用的，冗长的、需要“解码”的数字命名法，转到另一种看起来比较整齐的命名法。比如，ARMv7 的三个款式都以 Cortex 作为主名。这不仅更加澄清并且“精装”了所使用的 ARM 架构，也避免了新手对架构号和系列号的混淆。例如，ARM7TDMI 并不是一款 ARMv7 的产品，而是辉煌起点——v4T 架构的产品。

4.ARM 专有技术

（1）Thumb

ARM 处理器的一种 16 位指令模式，称为 Thumb。Thumb 指令集可以看作是 ARM 指令压缩形式的子集，它是为减小代码量而提出，具有 16bit 的代码密度。Thumb 指令体系并不完整，只支持通用功能，必要时仍需要使用 ARM 指令，如进入异常时。其指令的格式与使用方式与 ARM 指令集类似，而且使用并不频繁

（2）Jazelle

Jazelle 是 ARM 体系结构的一种相关技术，用于在处理器指令层次对 JAVA 加速。ARM 还开发出一项技术，Jazelle DBX (Direct Bytecode eXecution)，允许它们在某些架构的硬件上加速执行 Java bytecode，就如其他执行模式般，当呼叫一些无法支援 bytecodes 的特殊软件时，能提供某些 bytecodes 的加速执行。它能在现存的 ARM 与 Thumb 模式之间互相执行。

首颗具备 Jazelle 技术的处理器是 ARM926EJ-S：Jazelle 以一个英文字母'J'标示于 CPU

名称中。它用来让手机制造商能够加速执行 Java ME 的游戏和应用程式，也因此促使了这项技术不断地发展。

（3）Thumb-2

Thumb-2 技术首先见于 ARM1156 核心，并于 2003 年发表。Thumb-2 扩充了受限的 16 位 Thumb 指令集，以额外的 32 位指令让指令集的使用更广泛。因此 Thumb-2 的预期目标是要达到近乎 Thumb 的编码密度，但能表现出近乎 ARM 指令集在 32 位内存下的效能.

Thumb-2 至今也从 ARM 和 Thumb 指令集中派生出多种指令，包含位段操作、分支跳转和条件执行等功能。

（4）ThumbEE

ThumbEE，也就是所谓的 Thumb-2EE,，业界称为 Jazelle RCT 技术，于 2005 年发表，首见于 Cortex-A8 处理器。ThumbEE 提供从 Thumb-2 而来的一些扩充性，在所处的执行环境（Execution Environment）下，使得指令集能特别适用于执行阶段（Runtime）的编码产生（例如即时编译）。Thumb-2EE 是专为一些语言如 Limbo、Java、C#、Perl 和 Python，并能让 即时编译器 能够输出更小的编译码却不会影响到效能。

ThumbEE 所提供的新功能，包括在每次存取指令时自动检查是否有无效指标，以及一种可以执行阵列范围检查的指令，并能够分支到分类器（handlers），其包含一小部份经常呼叫的编码，通常用于高阶语言功能的实作，例如对一个新物件做内存配置。

（5）高级 SIMD（NEON）

高级 SIMD 延伸集，业界称为 NEON 技术，它是一个结合 64 和 128 bit 的 SIMD（Single Instruction Multiple Data 单指令多重数据）指令集，其针对多媒体和讯号处理程式具备标准化加速的能力。NEON 可以在 10 MHz 的 CPU 上执行 MP3 音效解码，且可以执行 13 MHz 频率以下的 GSM AMR (Adaptive Multi-Rate) 语音编码。NEON 具有一组广泛的指令集、各自的寄存器阵列，以及独立执行的硬件。NEON 支援 8-, 16-, 32- 和 64-bit 的整数及单精度浮点数据，并以 SIMD 的方式运算，执行图形和游戏处理中关于语音 / 视讯的部分。SIMD 在 向量超级处理机 中是个决定性的要素，它具备同时多项处理功能。在 NEON 技术中，SIMD 最高可支援到同时 16 个运算。

（6）VFP

VFP 是在协同处理器针对 ARM 架构的衍生技术。它提供低成本的单精度和倍精度浮点运算能力，并完全相容于 ANSI/IEEE Std 754-1985 二进制浮点算数标准。VFP 提供大多数适用于浮点运算的应用，例如 PDA、智慧手机、语音压缩与解压、3D 图像以及数位音效、打印机、机上盒，和汽车应用等。VFP 架构也支援 SIMD（单指令多重数据）平行化的短向量指令执行。这在图像和讯号处理等应用上，非常有助于降低编码大小并增加输出效率。在 ARM-based 处理器中，其他可见的浮点、或 SIMD 的协同处理器还包括了 FPA, FPE, iwMMXt。他们提供类似 VFP 的功能但在 opcode 层面上来说并不具有相容性。

（7）安全性扩充（TrustZone）

TrustZone(TM) 技术出现在 ARMv6KZ 以及较晚期的应用核心架构中。它提供了一种低成本的方案，针对系统单芯片（SoC）内加入专属的安全核心，由硬件建构的存取控制方式支援两颗虚拟的处理器。这个方式可使得应用程式核心能够在两个状态之间切换（通常改称为领域（worlds）以避免和其他功能领域的名称混淆），在此架构下可以避免资讯从较可信的核心领域泄漏至较不安全的领域。这种内核领域之间的切换通常是与处理器其他功能完全无关联性（orthogonal），因此各个领域可以各自独立运作但却仍能使用同一颗内核。内存和周边装置也可因此得知目前内核运作的领域为何，并能针对这个方式来提供对装置的机密和编码进行存取控制。典型的 TrustZone 技术应用是要能在一个缺乏安全性的环境下完整地执行操作系统，并在可信的环境下能有更少的安全性的编码。

Cortex-M4 处理器是由 ARM 专门开发的最新嵌入式处理器，它完美融合了高效的信号处理能力以及 Cortex-M 系列处理器诸多无可比拟的优势，包括低功耗、低成本和易于使用，旨在满足那些新兴的、灵活多变的解决方案的需求。飞思卡尔 Kinetis 系列使用 ARM Cortex-M4 处理器，下一节将阐述 Kinetis 系列。

2.2 Kinetis系列微处理器概述

Kinetis系列微控制器是飞思卡尔公司于2010年下半年推出的基于ARM Cortex-M4内核的微控制器，是业内首款Cortex-M4内核芯片。Kinetis系列微控制器采用了飞思卡尔 90纳米薄膜存储器 (TFS) 闪存技术和 Flex存储器功能(可配置的内嵌EEPROM)，支持超过1000万次的擦写。Kinetis 微控制器系列融合了最新的低功耗革新技术，具有高性能、高精度的混合信号能力，宽广的互连性，人机接口和安全外设。

第一阶段产品由五个微控制器系列组成，见图2-1所示，包含超过两百种器件，在引脚、外设和软件上可兼容。每个系列提供了不同的性能，存储器和外设特性。通过通用外设、存储器映射和封装的一致性来实现系列内和各系列间的便捷移植。

系列	程序闪存	封装	关键特性
K60 系列	256KB-1MB	100-256引脚	
K40 系列	64-512KB	64-144 引脚	
K30 系列	64-512KB	64-144 引脚	
K20 系列	32KB-1MB	32-144 引脚	
K10 系列	32KB-1MB	32-144 引脚	

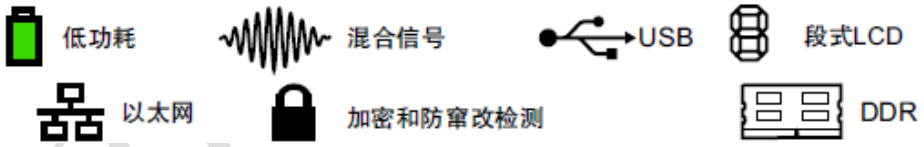


图2-1 **Kenetis** 微控制器产品组合

由图2-1可见，所有的 Kinetis 系列都包含丰富的模拟、通信和定时控制外设，提供多种闪存容量和输入输出引脚数量。所有Kinetis 系列都具有以下特性：

- **内核：**
  - ARM Cortex-M4 内核带DSP指令，性能可达1.25 DMIPS/MHz (部分Kinetis 系列提供浮点单元)
  - 多达32通道的DMA可用于外设和存储器数据传输并减少CPU干预
  - 提供不同级别的CPU频率, 有50 MHz、72 MHz 和100 MHz（部分Kinetis系列提供120 MHz 和150 MHz)
- **极低的功耗：**
  - 10 种低功耗操作模式用于优化外设活动和唤醒时间以延长电池的寿命
  - 低漏唤醒单元、低功耗定时器和低功耗RTC可以更加灵活地实现低功耗
  - 行业领先的快速唤醒时间

- **存储器：**
  - 内存空间可扩展，从32 KB闪存/ 8 KB RAM 到 1 MB 闪存 / 128 KB RAM。多个独立的闪存模块使同时进行代码执行和固件升级成为可能
  - 可选的16 KB 缓存用于优化总线带宽和闪存执行性能
  - Flex 存储器具有高达512 KB的FlexNVM 和高达16 KB的FlexRAM。FlexNVM 能够被分区以支持额外的程序闪存（例如引导加载程序）、数据闪存 (例如存储大表) 或者EEPROM 备份。FlexRAM 支持EEPROM 字节写/ 字节擦除操作，并且指示最大 EEPROM 空间
  - EEPROM 最高超过一千万次的使用寿命
  - EEPROM 擦除/ 写速度远高于传统的EEPROM
- **模拟混合信号：**
  - 快速、高精度的16位ADC、12位DAC、可编程增益放大器、高速比较器和内部电压参考。提供强大的信号调节、转换和分析性能的同时降低了系统成本
- **人机接口 (HMI)：**
  - 低功耗感应触摸传感接口在所有低功耗模式均可工作
- **连接性和通信：**
  - UART支持ISO7816 和 IrDA， I^2S、CAN、I^2C 和 SPI
- **可靠性和安全性：**
  - 硬件循环冗余校验引擎用于验证存储器内容、通信数据和增加的系统可靠性
  - 独立时钟工作的COP 用于防止代码跑飞
  - 外部看门狗监控
- **定时和控制：**
  - 强大的FlexTimers 支持通用、PWM 和电机控制功能
  - 载波调制器发射器用于产生红外波形
  - 可编程中断定时器用于RTOS 任务调度或者为ADC 转换和可编程延迟模块提供触发源
- **外部接口：**
  - 多功能外部总线接口提供和外部存储器、门阵列逻辑或LCD 的接口
- **系统：**
  - 5 V 容限的GPIO 带引脚中断功能
  - 从 1.71 V 到 3.6 V 的宽操作电压范围，闪存编程电压低至 1.71 V ，并且此时闪存和模拟外设所有功能正常
  - 运行温度 -40 °C 到105 °C

除了以上共性，图 2-2列出了各Kinetis 系列所特有的性能。

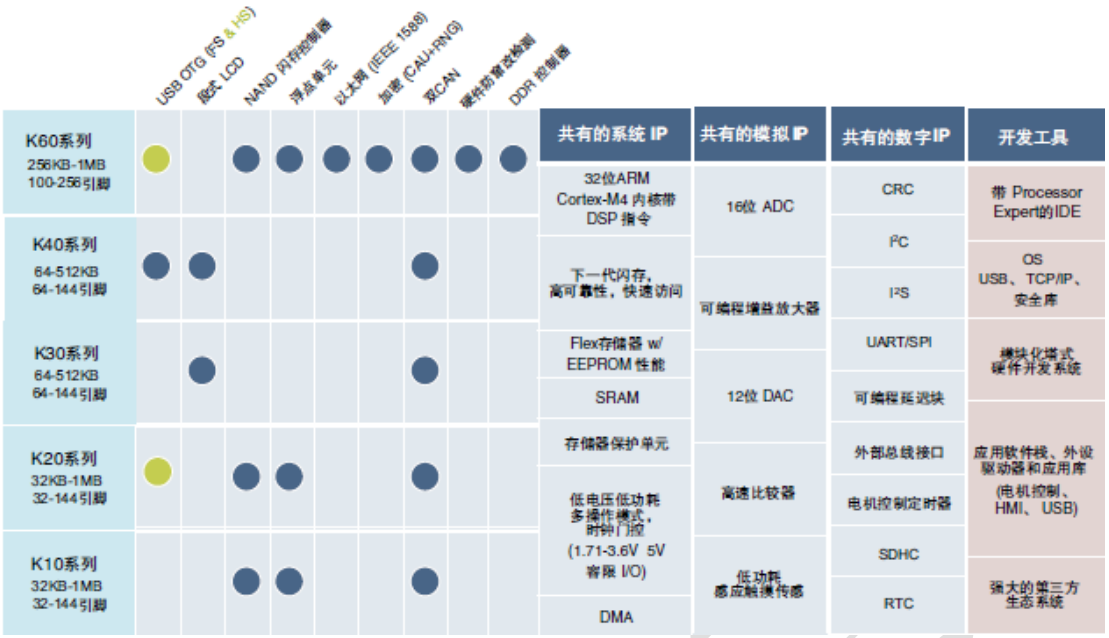


图 2-2 **Kinetis** 系列微控制器特性

2.3 Kinetis系列微控制器存储器映像与编程结构

2010年11月，Freescale开始提供K60的样片，2011年上半年K60批量上市。MK60N512VMD100是K60系列 MCU，是Kinetis系列的代表芯片。本书以该芯片为主要蓝本阐述嵌入式开发所必备的硬件设计、软件设计及相关技术。**一般来说，学习一个新的MCU芯片，若用C语言进行编程，比较快速的学习过程是：**

（1）了解性能及内部主要功能模块与存储空间的地址分配。

（2）了解基本的编程结构、编程模式及寻址方式。

（3）了解中断结构。

（4）了解芯片的引脚的总体布局情况，掌握硬件最小系统电路。

（5）理解第一个工程的结构，理解工程中各个文件的基本功能。一般来说，第一个工程为一个简单的小程序，如利用通用 I/O 模块编程控制几个发光二极管，主要目的是给出程序框架和工作过程。

（6）进行实际环境的编译（compile）、链接（link）生成可以下载到芯片内部 Flash 存储器中的程序（可以运行的机器码），基本理解列表文件、机器码文件。

（7）一定要有硬件评估环境，这是学习新 MCU 的必需品。这样就可将程序利用写入调试器下载到目标 MCU 中，在目标板上，观察运行情况。随后，可进一步利用嵌入式软件的打桩调试技术，即在被测程序代码中插入一些函数或语句，利用这些函数或语句产生可在硬件板上显示物理现象，供观察程序运行情况之用。

（8）从整个工程组成、各个文件、写入 Flash 存储器的机器码等角度，透彻理解第一工程的执行过程。

（9）理解第一个带有中断过程的 C 语言工程结构，理解主循环与中断两条程序执行路线各自的作用。

至此，以上学习过程已经覆盖学习一个新 MCU 硬件设计与软件编程的基本要素。完成了以上学习步骤，就完成了“基本入门”过程。随后，可以在此框架下，结合嵌入式构件方

法，逐个模块学习就方便了。

本章给出上述过程的（1）—（4）步，下一章给出上述过程的（5）—（8）步。这两章完成了“基本入门”的前 8 步。“基本入门”的第 9 步（第一个中断例程）将在第 4 章阐述。为了规范编程，符合嵌入式软件工程的基本要求，提高硬件及底层驱动软件的可复用与可移植性。

2.3.1 K60 系列 MCU 性能概述与内部结构简图

K60 微控制器系列具有IEEE 1588 以太网，全速和高速 USB 2.0 On-The-Go 带设备充电探测，硬件加密和防窜改探测能力，具有丰富的模拟、通信、定时和控制外设，从100 LQFP 封装 256 KB 闪存开始可扩展到256 MAPBGA 1MB 闪存。 大闪存的 K60 系列器件还可提供可选的单精度浮点单元、NAND 闪存控制器和DRAM 控制器。

图 2-3 给出了 K60 系列的模块结构框图。

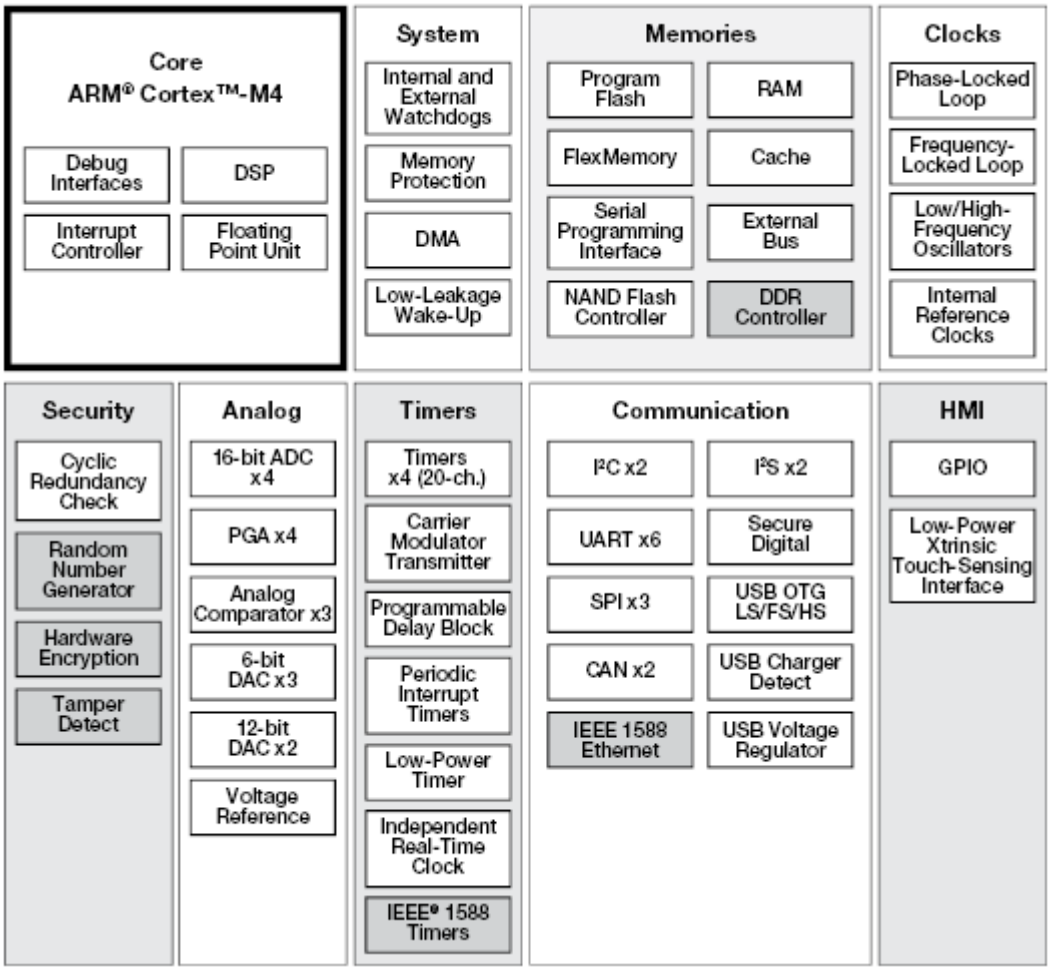


图 2-3 **K60 模块结构图**

可选的芯片类型有 MK60N256VLQ100、MK60X256VLQ100、MK60N512VLQ100、MK60N256VMD100、MK60X256VMD100 和 MK60N512VMD100。这些芯片的 CPU 频率与引脚数都一样，但是在封装、Flash 容量、程序空间等方面还是有差别的。表 2-2 给出了可选芯片的 MCU 简要描述，供选型参考之用。

表 2-2 K60 可选择的芯片类型

芯片类型	CPU 频率	引脚数	封装	Flash 容量	程序空间	EEPROM	SRAM	GPIO
MK60N256VLQ100	100 MHz	144	LQFP	256 KB	256 KB	—	64 KB	100
MK60X256VLQ100	100 MHz	144	LQFP	512 KB	256 KB	4 KB	64 KB	100
MK60N512VLQ100	100 MHz	144	LQFP	512 KB	512 KB	—	128 KB	100
MK60N256VMD100	100 MHz	144	MAP BGA	256 KB	256 KB	—	64 KB	100
MK60X256VMD100	100 MHz	144	MAP BGA	512 KB	256	4 KB	64 KB	100
MK60N512VMD100	100 MHz	144	MAP BGA	512 KB	512 KB	—	128 KB	100

表 2-3 总结了 K60 系列 MCU 的共性

表 2-3**. K60** 系列器件的共性

工作特性	•电压范围 1.71V - 3.6V • 闪存编程电压最低至1.71V • 温度范围(TA) -40 to 105°C • 灵活的工作模式
内核特性	• 32 位 ARM Cortex-M4 内核 • 支持DSP 指令 • 嵌套向量中断控制器(NVIC) • 异步唤醒中断控制器(AWIC) • 调试和跟踪 • 2 引脚串口调试 (SWD) • IEEE 1149.1 JTAG 调试 (JTAG) • IEEE 1149.7 简洁 JTAG (cJTAG) • 端口跟踪接口单元(TPIU) • 闪存片和断点单元 (FPB) • 数据检测和跟踪单元(DWT) • 指令跟踪宏单元 (ITM)
系统和功耗管理	• 带外部监控引脚的软件和硬件看门狗 • 带16 个通道的DMA 控制器 • 低漏唤醒单元 (LLWU) • 带10 种功耗模式的功耗管理控制器 • 不可屏蔽中断(NMI) • 每个芯片 128 位唯一标识(ID) 数
时钟	• 多用途时钟发生器 • PLL 和FLL • 内部参考时钟(32kHz 或 2MHz) • 4MHz 到 32MHz 晶振 • 32kHz 到 40kHz 晶振

	• 内部 1kHz 低功耗振荡器 • DC 到 50MHz 外部方波输入时钟
存储器和存储器接口	• Flex 存储器有FlexNVM (非易失闪存用于执行程序代码、存储数据或者备份EEPROM 数据) 或者FlexRAM (RAM 存储器被用作传统的RAM 或者高耐擦写EEPROM 存储和加快闪存程序运行) • 闪存安全性和保护特性 • 串行闪存编程接口(EzPort)
安全和集成性	• 循环冗余校检(CRC)
模拟	• 16 位 SAR ADC • 可编程的电压参考(VREF) • 12 位 DAC • 带 6 位 DAC 的高速模拟比较器 (CMP)
定时器	• 1x8ch 电机控制/ 通用/PWM 定时器(FTM) • 2x2ch 正交解码器/ 通用/PWM 定时器 (FTM) • 载波调制定时器(CMT) • 可编程延迟模块 (PDB) • 1x4ch 可编程中断定时器(PIT) • 低功耗定时器(LPT)
通信	• 支持IEEE 1588 的以太网接口 • USB 全速/ 低速 OTG/ 主机/ 从设备接口 • CAN • SPI • I2C，支持SMBUS • UART (带 ISO7816、IrDA 和硬件流控)
人机接口	• GPIO 支持引脚中断、DMA 请求、数字滤波和其他引脚控制选项 • 最大允许5V 输入 • 电容式触摸传感输入

2.3.2 K60 系列存储器映像

表 2-4 列出了不同频率、不同封装的 K60 系列微控制器的存储器大小。

表2-4 **K60 系列MCU 概述**

	存储器				封装					
CPU 频率 (MHz)	闪存 **(KB)**	**FlexNVM (KB)**	**SRAM (KB)**	**FlexRAM (KB)**	**100 LQFP (14x14)**	**104 BGA (8x8)**	**144 LQFP (20x20)**	**144 BGA (13x13)**	**196 BGA (15x15)**	**256 BGA (17x17)**
100	256	—	64	—	+	+	+	+	—	—
100	512	—	128	—	+	+	+	+	—	—
100	256	256	64	4	+	+	+	+	—	—
120	512	512	128	16	—	—	+	+	+	+
150	512	512	128	16	—	—	+	+	+	+
120	1024	—	128	—	—	—	+	+	+	+

150	1024	—	128	—	—	—	+	+	+	+

Flex存储器是飞思卡尔的新一代Flex存储器技术，为需要片上EEPROM、额外程序或数据的开发者提供非常多样化和强大的解决方案。Flex 存储器和SRAM 一样简单快速，当用作高耐久性擦写EEPROM 时，在完成程序运行和擦除功能时不需要用户或者系统干预。Flex 存储器同时能提供平行于主程序闪存的额外闪存 (FlexNVM) 用于数据或者程序存储。Flex 存储器使您能完全配置 FlexNVM 和 FlexRAM 模块，从而为应用提供最均衡的存储器资源。用户可配置的参数包括：EEPROM 大小、擦写次数、写大小和额外程序/ 数据闪存的大小。

FlexNVM能被用作EEPROM 配置的一部分、额外的程序或者数据闪存。也可以一部分用作闪存同时另一部分被用作增强型EEPROM 备份

FlexRAM能被用作EEPROM 配置的一部分或者额外的系统RAM

K60 系列的 MCU 为 32 位微控制器，可寻址 4GB 的地址空间，地址范围为 0x0000_0000~0xFFFF_FFFF。K60 系列的存储器空间地址映像见表 2-5 所示。

表 2-5 K60 系列的存储器空间地址映像

地址范围	空间大小	实际的物理对象
0x0000_0000–0x0FFF_FFFF	256MB	可编程 flash 和只读数据(包括一开始 1024 字节的异常中断向量)
0x1000_0000–0x13FF_FFFF	64MB	对 MK60N256VLQ100 芯片：未使用 对 MK60X256VLQ100 芯片：FlexNVM 对 MK60N512VLQ100 芯片：未使用 对 MK60N256VMD100 芯片：未使用 对 MK60X256VLQ100 芯片：FlexNVM 对 MK60N512VMD100 芯片：未使用
0x1400_0000–0x17FF_FFFF	64MB	对有 FlexNVM 的设备：FlexRAM 对只有可编程 Flash 的设备：可编程加速 RAM
0x1800_0000–0x1FFF_FFFF	128MB	SRAM_L
0x2000_0000–0x200F_FFFF	1MB	SRAM_U
0x2010_0000–0x21FF_FFFF	31MB	未使用
0x2200_0000–0x23FF_FFFF	32MB	SRAM_U 的混合位宽区
0x2400_0000–0x3FFF_FFFF	448MB	未使用
0x4000_0000–0x4007_FFFF	512KB	外设桥 0（AIPS-Lite0）
0x4008_0000–0x400F_EFFF	508KB	外设桥 1（AIPS-Lite1）
0x400F_F000–0x400F_FFFF	4KB	通用输入输出
0x4010_0000–0x41FF_FFFF	25MB	未使用
0x4200_0000–0x43FF_FFFF	32MB	外设桥（AIPS-Lite）与通用输入输出的混合位宽区
0x4400_0000–0x5FFF_FFFF	448MB	未使用
0x6000_0000–0xDFFF_FFFF	2GB	Flexbus
0xE000_0000–0xE00F_FFFF	1MB	私有外设
0xE010_0000–0xFFFF_FFFF	511MB	未使用

片上 RAM 分为 SRAM_L 和 SRAM_U，从表 2-5 可见它们是连续的存储映射。对于 SRAM_L 和 SRAM_U 如果处于片外时，在请求主机访问总线时将会产生一个错误。外设存储映射在 0x4000_0000 到 0x400F_FFFF 区有两个从机端口，实现了 2 个外设桥(AIPS-Lite0 和 AIPS-Lite 1)： AIPS-Lite0 占 512KB ，AIPS-Lite1 占 508KB，4KB 的 GPIO。 AIPS-Lite0

连接到从机端口 2，可访问区间是 0x4000_0000 到 0x4007_FFFF。 AIPS-Lite1 和 GPIO 共享从机端口 3，AIPS-Lite1 的可访问区间是 0x4008_0000 到 0x400F_EFFF，GPIO 的 4KB 可访问区间是 0x400F_F000 到 0x400F_FFFF。它可直接连接到闩开关提供的主机不需要等待 AIPS-Lite 控制器的状态。 本模块的时钟是由 SIM 寄存器的 AIPS 控制位控制的。访问任何未实现或未使能的外设桥信号将产生错误。 对于可编程模块访问外设桥，只有 4KB 的实现空间。

2.4 K60的引脚功能与硬件最小系统

本书以 144 引脚 LQFP 封装的 MK60N512VMD100 芯片为例介绍 K60 的编程和应用。

2.4.1K60 的引脚功能

图 2-4 是 144 引脚 LQFP 封装的 MK60N512VMD100 的引脚图。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，有的有四个复用功能，还有的有六个复用功能，系统设计时必须注意只能使用其中的一个功能。一般情况下，需要按引脚功能分类了解 MCU 引脚功能情况。

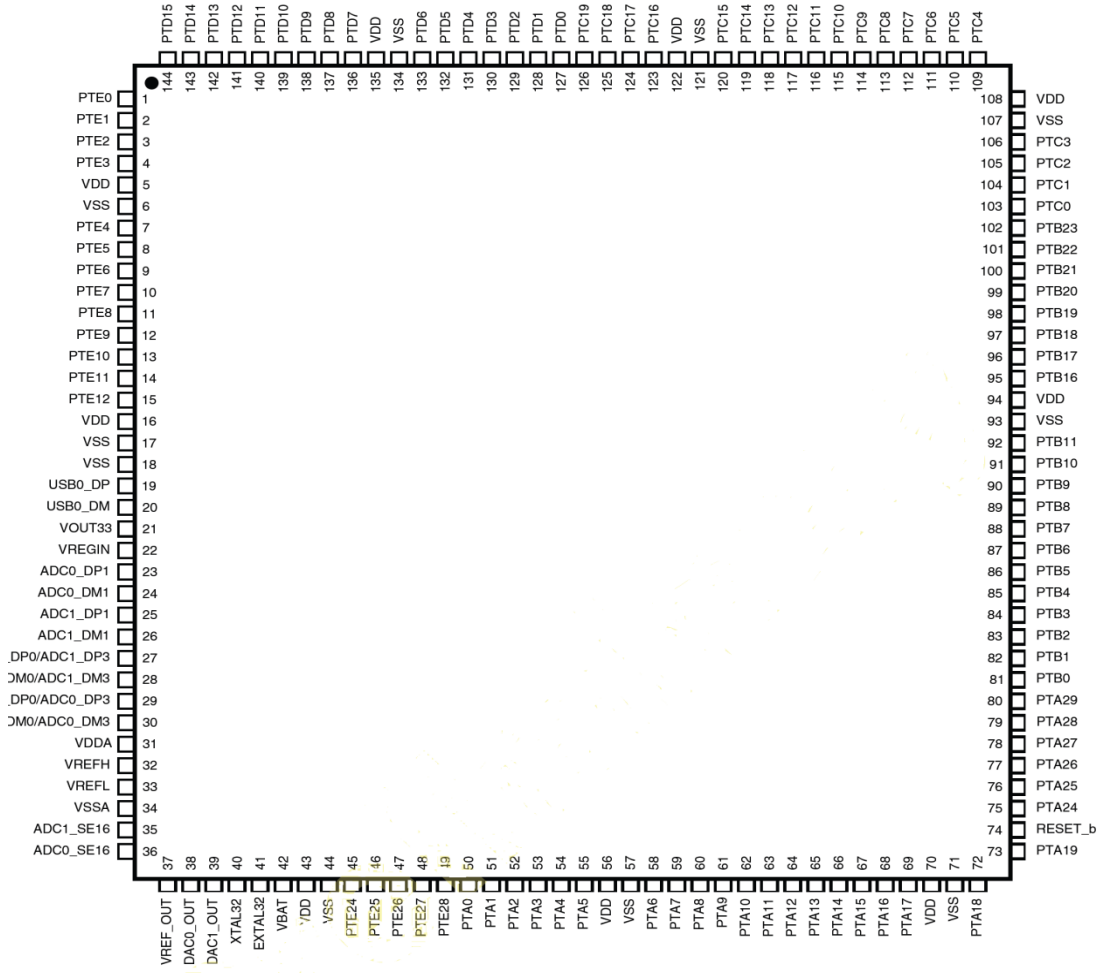


图 2-4 是 144 引脚 LQFP 封装的 MK60N512VMD100 的引脚图

表 2-6 按引脚功能分类列出了 MK60N512VMD100 的工作支撑引脚。

表 2-6　MK60N512VMD100 工作支撑引脚表

分类	引脚名	引脚号	典型值	功能描述	备注
电源类	VDD	5、16、43、56、70、94、108、122、135	3.3V	为 I/O 引脚提供电源	范围 1.71～3.6V
	VSS	6、17、18、44、57、71、93、107、121、134、	0V	为 I/O 引脚提供参考地	
	VDDA	31	3.3V	为 AD 转换模块供电源	范围 1.71～3.6V
	VSSA	34	0V	为 AD 转换模块供参考地	
	VREFH	32		AD 模块参考高电平	
	VREFL	33		AD 模块参考高低平	
	VREF_OUT	37		内部产生的参考输出电压	
	VREGIN	22		USB 调节器输入电源	2.7~5.5
	VOUT33	21		USB 调节器输出电源	
	VBAT	42		32KHz 晶振电源	1.71~3.6
复位	$\overline{\text{RESET}}$	74			低电平时芯片复位
晶振	EXTAL32	41		RTC 晶振输入引脚（32KHz）	
	XTAL32	40		RTC 晶振输出引脚（32KHz）	
	EXTAL	72		外部晶振输入引脚	
	XTAL	73		外部晶振输出引脚	
写入器	JTAG_TDI	51		JTAG 测试数据输入引脚	TCK 线上升沿时，从线上取数
	JTAG_TDO	52		JTAG 测试数据输出引脚	TCK 线下降沿时，数据上线
	JTAG_TCLK	50		JTAG 测试时钟线	
	JTAG_TMS	53		JTAG 测试模式选择线	用于设置 JTAG 控制器的状态
	JTAG_TRST	55		JTAG 复位	

除去需要服务的引脚外，其它引脚可以为实际系统提供 I/O 服务。芯片提供服务的引脚也可称为 I/O 端口资源类引脚。MK60N512VMD100（144 引脚 LQFP 封装）的有多达 100 个 I/O 引脚。其中 A 口 26 个，B 口 20 个，C 口 20 个，D 口 16 个，E 口 18 个，详细情况见表 2-7 所示。

表 2-7　I/O 端口资源表

口名	引脚数	引脚名	引脚号	功能描述					
				第一	第二	第三	第四	第五	第六
A	26	PTA0	50	TSI0_CH1	GPIO	UART0_CTS_b	FTM0_CH5	JTAG_TCLK/SWD_CLK	EZP_CLK
		PTA1	51	TSI0_CH2	GPIO	UART0_RX	FTM0_CH6	JTAG_TDI	EZP_DI
		PTA2	52	TSI0_CH3	GPIO	UART0_TX	FTM0_CH7	JTAG_TDO/TRACE_SWO	EZP_DO
		PTA3	53	TSI0_CH4	GPIO	UART0_RTS_b	FTM0_CH0	JTAG_TMS/SWD_DIO	
		PTA4	54	TSI0_CH5	GPIO	FTM0_CH1	NMI_b	EZP_CS_b	
		PTA5	55	GPIO	FTM0_CH2	RMII0_RXER/MII0_RXER	CMP2_OUT	I2S0_RX_BCLK	JTAG_TRST
		PTA6	58	GPIO	FTM0_CH3	TRACE_CLKOUT			
		PTA7	59	ADC0_SE10	GPIO	FTM0_CH4	TRACE_D3		
		PTA8	60	ADC0_SE11	GPIO	FTM1_CH0	FTM1_QD_PHA	TRACE_D2	
		PTA9	61	GPIO	FTM1_CH1	MII0_RXD3	FTM1_QD_PHB	TRACE_D1	
		PTA10	62	GPIO	FTM2_CH0	MII0_RXD2	FTM2_QD_PHA	TRACE_D0	
		PTA11	63	GPIO	FTM2_CH1	MII0_RXCLK	FTM2_QD_PHB		
		PTA12	64	CMP2_IN0	GPIO	CAN0_TX	FTM1_CH0	RMII0_RXD1/MII0_RXD1	I2S0_TXD
		PTA13	65	CMP2_IN1	GPIO	CAN0_RX	FTM1_CH1	RMII0_RXD0/MII0_RXD0	I2S0_TX_FS
		PTA14	66	GPIO	SPI0_PCS0	UART0_TX	RMII0_CRS_DV/MII0_RXDV	I2S0_TX_BCLK	

		PTA15	67	GPIO	SPI0_SCk	UART0_RX	RMII0_TXEN/ MII0_TXEN	I2S0_RXD	
		PTA16	68	GPIO	SPI0_SOUT	UART0_CTS_b	RMII0_TXD0/ MII0_TXD0	I2S0_RX_FS	
		PTA17	69	ADC1_SE17	GPIO	SPI0_SIN	UART0_RTS_b	RMII0_TXD1/ MII0_TXD1	I2S0_MCLK
		PTA18	72	EXTAL	GPIO	FTM0_FLT2	FTM_CLKIN0		
		PTA19	73	XTAL	GPIO	FTM1_FLT0	FTM_CLKIN1	LPT0_ALT1	
		PTA24	75	GPIO	MII0_TXD2	FB_A29			
		PTA25	76	GPIO	MII0_TXCLK	FB_A28			
		PTA26	77	GPIO	MII0_TXD3	FB_A27			
		PTA27	78	GPIO	MII0_CRS	FB_A26			
		PTA28	79	GPIO	MII0_TXER	FB_A25			
		PTA29	80	GPIO	MII0_COL	FB_A24			
B	20	PTB0	81	ADC0_SESE8/ ADC1_SE8/ TSI0_CH0	GPIO	I2C0_SCL	FTM1_CH0	RMII0_MDIO/ MII0_MDIO	FTM1_QD_PHA
		PTB1	82	ADC0_SE9/ ADC1_SE9/ TSI0_CH6	GPIO	I2C0_SDA	FTM1_CH1	RMII0_MDC/ MII0_MDC	FTM1_QD_PHB
		PTB2	83	ADC0_SE12/ TSI0_CH7	GPIO	I2C0_SCL	UART0_RTS_b	ENET0_1588_TMR0	FTM0_FLT3

		PTB3	84	ADC0_SE13/TSI0_CH8	GPIO	I2C0_SDA	UART0_CTS_b	ENET0_1588_TMR1	FTM0_FLT0
		PTB4	85	ADC1_SE10	GPIO	ENET0_1588_TMR2	FTM1_FLT0		
		PTB5	86	ADC1_SE11	GPIO	ENET0_1588TMR3	FTM2_FLT0		
		PTB6	87	ADC1_SE12	GPIO	FB_AD23			
		PTB7	88	ADC1_SE13	GPIO	FB_AD22			
		PTB8	89	GPIO	UART3_RTS	FB_AD21			
		PTB9	90	GPIO	SPI1_PCS1	UART3_CTS	FB_AD20		
		PTB10	91	ADC1_SE14	GPIO	SPI1_PCS0	UART3_RX	FB_AD19	FTM0_FLT1
		PTB11	92	ADC1_SE15	GPIO	SPI1_SCK	UART3_TX	FB_AD18	FTM0_FLT2
		PTB16	95	TSI0_CH9	GPIO	SPI1_SOUT	UART0_RX	FB_AD17	EWM_IN
		PTB17	96	TSI0_CH10	GPIO	SPI1_SIN	UART0_TX	FB_AD16	EWM_OUT_b
		PTB18	97	TSI0_CH11	GPIO	CAN0_TX	FTM2_CH0	I2S0_TX_BCLK	FB_AD15HA
		PTB19	98	TSI0_CH12	GPIO	CAN)_RX	FTM2_CH1	I2S0_TX_FS	FB_OE_b
		PTB20	99	GPIO	SPI2_PCS0	FB_AD31	CMP0_OUT		
		PTB21	100	GPIO	SPI2_SCK	FB_AD30	CMP1_OUT		
		PTB22	101	GPIO	SPI2_SOUT	FB_AD29	CMP2_OUT		
		PTB23	102	GPIO	SPI2_SIN	FB_AD28			
C	20	PTC0	103	ADC0_SE14/TSI0_CH13	GPIO	SPI0_PCS4	PDB0_EXTRG	I2S0_TXD	FB_AD14

		PTC1	104	ADC0_SE15/TSI0_CH14	GPIO	SPI0_PCS3	UART1_RTS_b	FTM0_CH0	FB_AD13
		PTC2	105	ADC0_SE4b/CMP1_IN0/TSI0_CH15	GPIO	SPI0_PCS2	UART1_CTS_b	FTM0_CH1	FB_AD12
		PTC3	106	CMP1_IN1	GPIO	SPI0_PCS1	UART1_RX	FTM0_CH2	FB_CLKOUT
		PTC4	109	GPIO	SPI0_PCS0	UART1_TX	FTM0_CH3	FB_AD11	CMP1_OUT
		PTC5	110	GPIO	SPI0_SCK	LPT0_ALT2	FB_AD10	CMP0_OUT	
		PTC6	111	CMP0_IN0	GPIO	SPI0_SOUT	PDB0_EXTRG	FB_AD9	
		PTC7	112	CMP0_IN1	GPIO	SPI0_SIN	FB_AD8		
		PTC8	113	ADC1_SE4b/CMP0_IN2	GPIO	I2S0_MCLK	I2S0_CLKIN	FB_AD7	
		PTC9	114	ADC1_SE5b/CMP0_IN3	GPIO	I2S0_RX_BCLK	FB_AD6	FTM2_FLT0	
		PTC10	115	ADC1_SE6b/CMP0_IN4	GPIO	I2C1_SCL	I2S0_RX_FS	FB_AD5	
		PTC11	116	ADC1_SE7b	GPIO	I2C1_SDA	I2S0_RXD	FB_RW_b	
		PTC12	117	GPIO	UART4_RTSb	FB_AD27			
		PTC13	118	GPIO	UART4_CTS_b	FB_AD26			
		PTC14	119	GPIO	UART4_RX	FB_AD25			

		PTC15	120	GPIO	UART4_TX	FB_AD24			
		PTC16	123	GPIO	CAN1_RX	UART3_RX	ENET0_1588_TMR0	FB_CS5_b/ FB_TSIZ1/ FB_BE23_16_BLS15_8_b	
		PTC17	124	GPIO	CAN1_TX	UART3_TX	ENET0_1588_TMR1	FB_CS4_b/ FB_TSIZ0/ FB_BE31_24_BLS7_0_b	
		PTC18	125	GPIO	UART3_RTS_b	ENET0_1588_TMR2	FB_TBST_b/ FB_CS2_b/ FB_BE15_8_BLS23_16_b		
		PTC19	126	GPIO	UART3_CTS_b	ENET0_1588_TMR3	FB_CS3_b/ FB_BE7_0_BLS31_24_b	FB_TA_b	
D	16	PTD0	127	GPIO	SPI0_PCS0	UART2_RTS_b	FB_ALE/FB_CS1_b/ FB_TS_b		
		PTD1	128	ADC0_SE5b	GPIO	SPI0_SCK	UART2_CTS_b	FB_CS0_b	
		PTD2	129	GPIO	SPI0_SOUT	UART2_RX	FB_AD4		
		PTD3	130	GPIO	SPI0_SIN	UART2_TX	FB_AD3		
		PTD4	131	GPIO	SPI0_PCS1	UART0_RTS_b	FTM0_CH4	FB_AD2	EWM_IN
		PTD5	132	ADC0_SE6b	GPIO	SPI0_PCS2	UART0_CTS_b	FTM0_CH5	FB_AD1

		PTD6	133	ADC0_SE7b	GPIO	SPI0_PCS3	UART0_RX	FTM0_CH6	FB_AD0
		PTD7	136	GPIO	CMT_IRO	UART0_TX	FTM0_CH7	FTM0_FLT1	
		PTD8	137	GPIO	I2C0_SCL	UART5_RX	FB_A16		
		PTD9	138	GPIO	I2C0_SDA	UART5_TX	FB_A17		
		PTD10	139	GPIO	UART5_RTS_b	FB_A18			
		PTD11	140	GPIO	SPI2_PCS0	UART5_CTS_b	SDHC0_CLKIN	FB_A19	
		PTD12	141	GPIO	SPI2_SCK	SDHC0_D4	FB_A20		
		PTD13	142	GPIO	SPI2_SOUT	SDHC0_D5	FB_A21		
		PTD14	143	GPIO	SPI2_SIN	SDHC0_D6	FB_A22		
		PTD15	144	GPIO	SPI2_PCS1	SDHC0_D7	FB_A22		
E	18	PTE0	1	ADC1_SE4a	GPIO	SPI1_PCS1	UART1_TX	SDHC0_D1	I2C1_SDA
		PTE1	2	ADC1_SE5a	GPIO	SPI1_SOUT	UART1_RX	SDHC0_D0	I2C1_SCL
		PTE2	3	ADC1_SE6a	GPIO	SPI1_SCK	UART1_CTS_b	SDHC0_DCLK	
		PTE3	4	ADC1_SE7a	GPIO	SPI1_SIN	UART1_RTS_b	SDHC0_CMD	
		PTE4	7	GPIO	SPI_PCS0	UART3_TX	SDHC0_D3		
		PTE5	8	GPIO	SPI_PCS2	UART3_RX	SDHC0_D2		

		PTE6	9	GPIO	SPI_PCS3	UART3_CTS_b	I2S0_MCLK	I2S0_CLKIN	
		PTE7	10	GPIO	UART3_RTS_b	I2S0_RXD			
		PTE8	11	GPIO	UART5_TX	I2S0_RX_FS			
		PTE9	12	GPIO	UART5_RX	I2S0_RX_BCLK			
		PTE10	13	GPIO	UART5_CTS_b	I2S0_TXD			
		PTE11	14	GPIO	UART5_RTS_b	I2S0_TX_FS			
		PTE12	15	GPIO	I2S0_TX_BCLK				
		PTE24	45	ADC0_SE17	GPIO	CAN1_TX	UART4_TX	EWM_OUT_b	
		PTE25	46	ADC0_SE18	GPIO	CAN1_RX	UART4_RX	EWM_IN	
		PTE26	47	GPIO	UART4_CTSb	ENET_1588_CLKIN	RTC_CLKOUT	USB_CLKIN	
		PTE27	48	GPIO	UART4_RTSb				
		PTE28	49	GPIO					
总	100								

2.4.2K60 硬件最小系统

MCU的硬件最小系统是指可以使内部程序运行的所必须的外围电路，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。当MCU工作不正常时，首先查找最小系统中可能出错的元件。一般情况下，MCU的硬件最小系统由电源、晶振及复位等电路组成。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。由于Flash存储器制造技术的发展，大部分芯片提供了在板或在系统（On System）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，MK60N512VMD100芯片的硬件最小系统包括电源电路、复位电路、晶振电路及JTAG接口电路。下面分别对这些电路给出简明分析。

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。所有的电源引脚必须外接适当的滤波电容抑制高频噪音。图2-5给出了MK60N512VMD100硬件最小系统的电源和地连接图。

电源（VDDx）与地（VSSx）包括很多引脚，如VDDA、VSSA、VDD1~VDD6、VSS1~VSS8、VREAFH和VREAFL等。一些电源与地引脚仅用于外接滤波电容，内部已经连接到电源与地，芯片参考手册指出不需要再外接电源。至于电容外接，是由于集成电路制造技术所限，无法在IC内部通过光刻的方法制造这些电容。电源滤波电路，用于改善系统的电磁兼容性，降低电源波动对系统的影响，增强电路工作稳定性。为标识系统通电与否，可以增加一个电源指示灯。

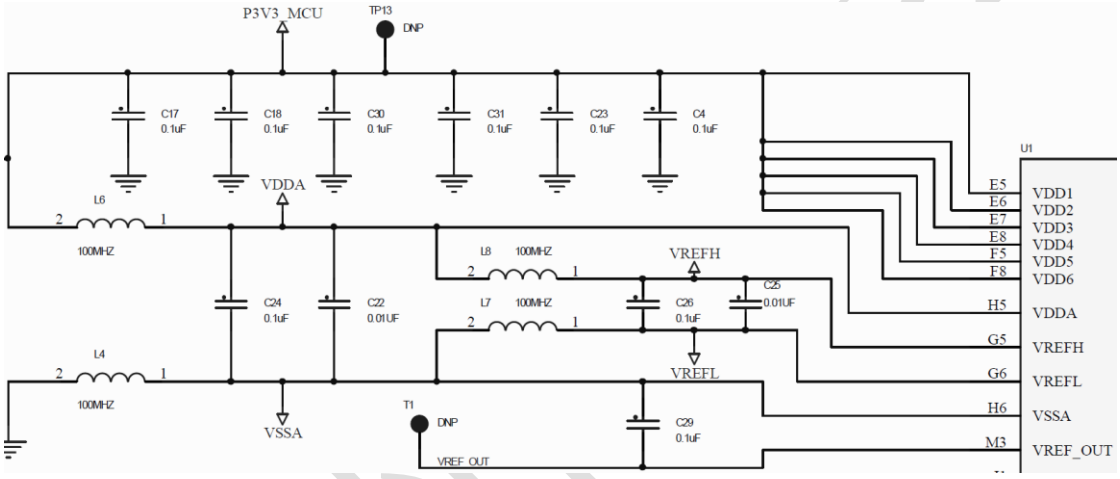


图 2-5　MK60N512VMD100 电源电路

2．复位电路

图2-6给出了MK60N512VMD100硬件最小系统的复位电路。复位，意味着MCU一切重新开始。复位引脚为$\overline{\text{RESET}}$。若$\overline{\text{RESET}}$信号有效（低电平），MCU复位。复位电路原理如下：正常工作时复位输入引脚$\overline{\text{RESET}}$通过一个10K的电阻接到电源正极，所以应为高电平。若按下复位按钮，则$\overline{\text{RESET}}$脚接地，为低电平，导致芯片复位。

从引起MCU复位的内部与外部因素区分，复位分为外部复位与内部复位两种。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、低漏唤醒(LLWU)复位、MCG丢失时钟复位、软件复位、锁定复位、EzPort复位等。

从复位时芯片是否处于上电状态区分，复位可分为冷复位与热复位。芯片从无电状态上电的复位属于冷复位，芯片处于带电状态的复位叫热复位。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响应快慢区分，复位可分为异步复位与同步复位。异步复位源提出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、丢失时钟、低压检测LVD、失去锁定等。同步复位的处理方法与异步复位不同，当一个同步复位源提出一个复位请求时，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，这是为了保护数据的完整性。在该总线周期结束后，下一个系统时钟的上升沿时，复位才有效。同步复位有看门狗定时器、软件等。

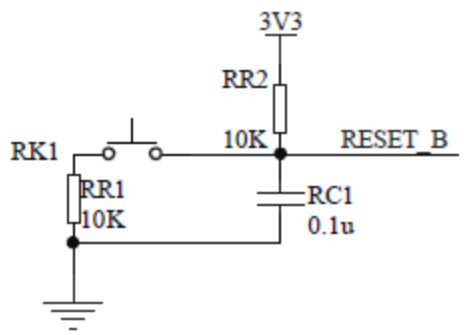


图 2-6 MK60N512VMD100 复位电路

3．晶振电路

晶振电路为芯片提供准确的工作时钟。

晶体振荡器分为无源晶振和有源晶振两种类型。需要外接电源的晶振称为有源晶振。无源晶振与有源晶振的英文名称不同，无源晶振为crystal（晶体），而有源晶振则叫做oscillator（振荡器）。无源晶振是有两个引脚的无极性元件，需要借助于时钟电路才能产生振荡信号，

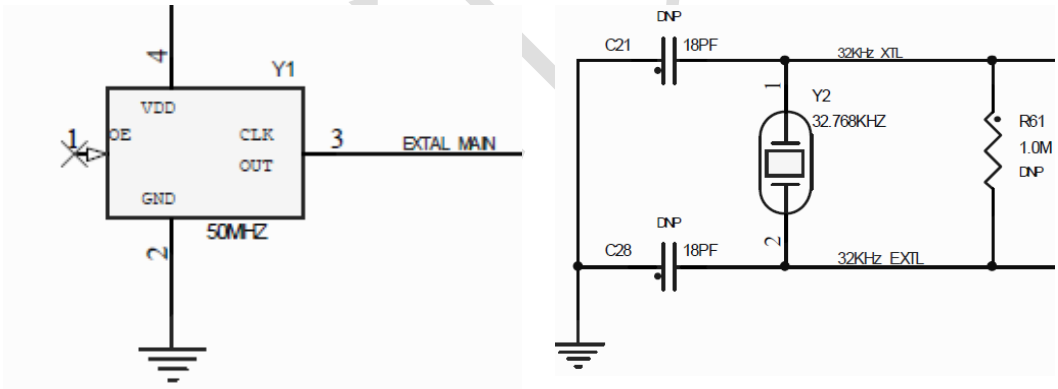


图 2-7 MK60N512VMD100 晶振电路

自身无法振荡起来。MK60N512VMD100内部集成多用途时钟产生器（Multipurpose Clock Generator ，MCG）模块，用于将晶振输入时钟倍频至系统所需时钟。MK60N512VMD100共需要两个晶振，一个是芯片的主晶振，用于产生芯片和外设的工作时钟，另一个是实时定时器的晶振（RTC）。苏州大学飞思卡尔嵌入式实验室开发的核心板主芯片时钟使用50MHZ有源晶振，RTC时钟使用32.768KHZ无源晶振，图2-7为系统晶振电路。在硬件布线时需要注意晶振附近不能走高频信号，晶振应该尽量靠近晶振输入引脚。晶振一旦不能正常工作，芯片将无法工作。晶振实际上负责给芯片提供心跳。

4．JTAG 电路

Kinetis 芯片使用的是 ARM Cortex-M4 内核，该内核内部集成了 JTAG（Joint Test Action Group）接口，通过 JTAG 接口可以实现程序下载和调试功能。图 2-8 为 JTAG 接口电路。

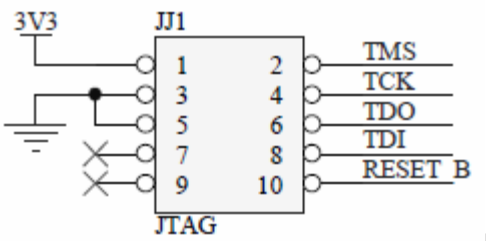


图 2-8 MK60N512VMD100 JTAG 电路

2.4.3 硬件最小系统测试方法

前面介绍了硬件最小系统的设计，给出了硬件最小系统元件的参考值。硬件最小系统电路原理图见光盘。

根据原理图制作了印刷电路板后，就开始硬件电路板的焊接和测试了。具体过程如下：

（1）焊接电源、复位电路、晶振电路、以及 JTAG 接口电路。注意：电源的滤波电容不可漏焊，否则芯片所受干扰较大，影响调试。

（2）在确保电源和地未短路的情况下接通电源，测量电压是否正常，检查按下复位按钮是否能够复位（观察复位指示灯）。

（3）将写入器与电路板连接，启动开发环境 IAR Embedded Workbench for RAM 6.10，对目标 MCU 进行擦除，如果成功则说明最小系统工作正常。

（4）将第一个样例程序下载到 Flash 中，观察小灯闪烁情况。

（5）硬件最小系统测试通过以后就可以进行其他模块焊接。正确的做法是，焊完一个模块后，应紧接着测试该模块工作是否正常，切忌焊接多个模块后再进行测试，因为一旦出现问题，就很难定位具体是哪个模块的问题。