第3章 STM32F429微控制器概述

3.1 STM32系列微控制器

3.2 STM32F429微控制器结构

3.3 STM32F4系列微控制器存储器映射和寄存器

电子与通信工程系

华东理工大学信息学院

3.1 STM32系列微控制器

3.1 STM32系列微控制器

STM32系列32位微控制器，是意法半导体（ST Microelectronics）基于ARM Cortex-Mx内核开发的处理器。它能支持32位广泛的应用，支持包括高性能、实时功能、数字信号处理，和低功耗、低电压操作，同时拥有一个完全集成和易用的开发。

特点：

- 高性能
- 低电压
- 低功耗
- 丰富外设
- 软硬件开发资源多
- 简单易用、自由、低风险

3.1 STM32系列微控制器

3.1.1 STM32系列微控制器简介

STM32系列从内核上分，可分为：Cortex-M0/-M0+、Cortex-M3、Cortex-M4以及Cortex-M7。

3.1 STM32系列处理器简介

3.1.1 STM32系列微控制器简介

STM32系列从内核上分，可分为：Cortex-M0/-M0+、Cortex-M3、Cortex-M4以及Cortex-M7。

表3-1　STM32系列微控制器各系列简述表

系列	微控制器
主流级微控制器	STM32 F0系列：ARM Cortex-M0入门级微控制器
	STM32 F1系列：ARM Cortex-M3基础型微控制器
	STM32 F3系列：ARM Cortex-M4混合信号微控制器
高性能微控制器	STM32 F2系列：ARM Cortex-M3高性能微控制器
	STM32 F4系列：ARM Cortex-M4高性能微控制器
	STM32 F7系列：ARM Cortex-M7高性能微控制器
	STM32 H7系列：ARM Cortex-M7超高性能微控制器
超低功耗微控制器	STM32 L0系列：ARM Cortex M0+低功耗微控制器
	STM32 L1系列：ARM Cortex M3超低功耗微控制器
	STM32 L4系列：ARM Cortex M4超低功耗微控制器
	STM32 L4+系列：ARM Cortex-M4超低功耗高性能微控制器
无线微控制器	STM32 WB系列：ARM Cortex-M4和Cortex-M0+双核无线微控制器

3.1 STM32系列处理器简介

3.1.1 STM32系列微控制器简介

1. 通信外设：USART、SPI、I2C;
2. 定时器：Multiple general-purpose timers;
3. 直接内存存取：Multiple DMA;
4. 看门狗和实时时钟：2x watchdogs、RTC;
5. PLL和时钟电路：Integrated regulator PLL and clock circuit;
6. 数模转换：Up to 3x 12-bit DAC;
7. 模数转换：Up to 4x 12-bit ADC(Up to 5 MSPS);
8. 工作温度：-40 to +85 ° C and up to 125 ° C operating temperature range;
9. 低电压：Low voltage 2.0 to 3.6 Vor 1.65/1.7 to 3.6 V(depending on series);
10. 内部温度传感器：Temperature sensor;

3.1 STM32系列处理器简介

3.1.1 STM32系列微控制器简介

STM32F4XX简介

高效的信号处理功能。适用需要有效且易于使用的控制和信号处理功能混合的数字信号控制市场。

3.1 STM32系列处理器简介

3.1.1 STM32系列微控制器简介

STM32最具竞争力的优势：

1）极高的性能： 主流的Cortex内核。

2）丰富合理的外设，合理的功耗，合理的价格。

3）强大的软件支持：丰富的软件包。

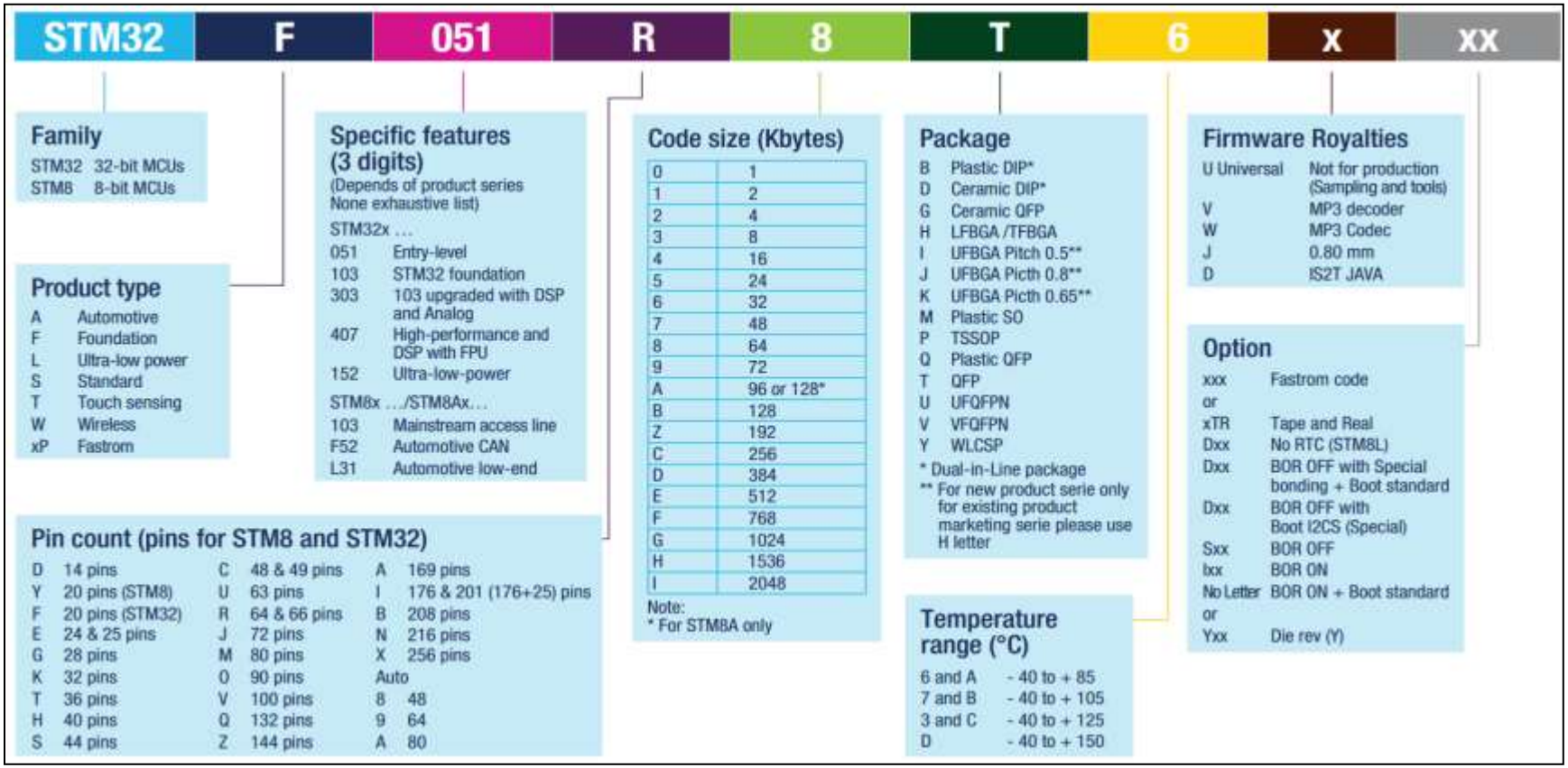
4）全面丰富的技术文档。

5）芯片型号种类多，覆盖面广。

6）强大的用户基础：最先成功试水CM4芯片的公司，积累了大批的用户群体，为其领先做铺垫。

3.1 STM32系列处理器简介

3.1.2 芯片命名规则



3.1 STM32系列处理器简介

3.1.2 芯片命名规则

STM32F429IGT6命名

—	ST M32	F	429	I	G	T	6
家族	STM32 表示 32bit 的 MCU						
产品类型	F 表示基础型						
具体特性	429 表示高性能且带 DSP 和 FPU						
引脚数目	I 表示 176pin，其他常用的为 C 表示 48，R 表示 64，V 表示 100，Z 表示 144，B 表示 208，N 表示 216						
FLASH 大小	G 表示 1024KB，其他常用的为 C 表示 256，E 表示 512，I 表示 2048						
封装	T 表示 QFP 封装，这个是最常用的封装						
温度	6 表示温度等级为 A： -40~85°						

3.1 STM32系列处理器简介

3.1.3 开发工具

进行STM32F系列微控制器的程序开发需要搭建一个交叉开发环境，其中包括计算机、开发软件、调试器、开发板或自己设计的电路板（包括STM32F系列微控制器）。

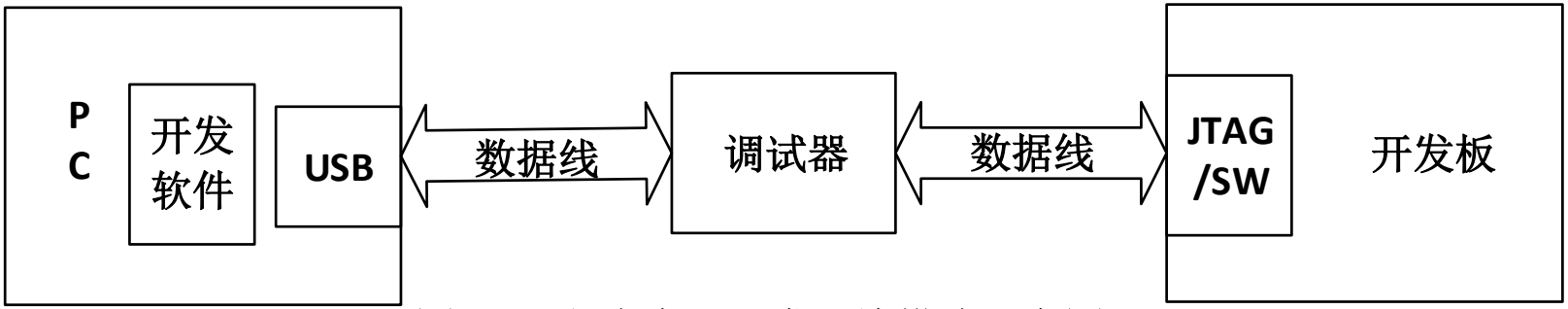


图3-2 程序交叉开发环境搭建示意图

1、开发板；

2、软件集成开发工具；

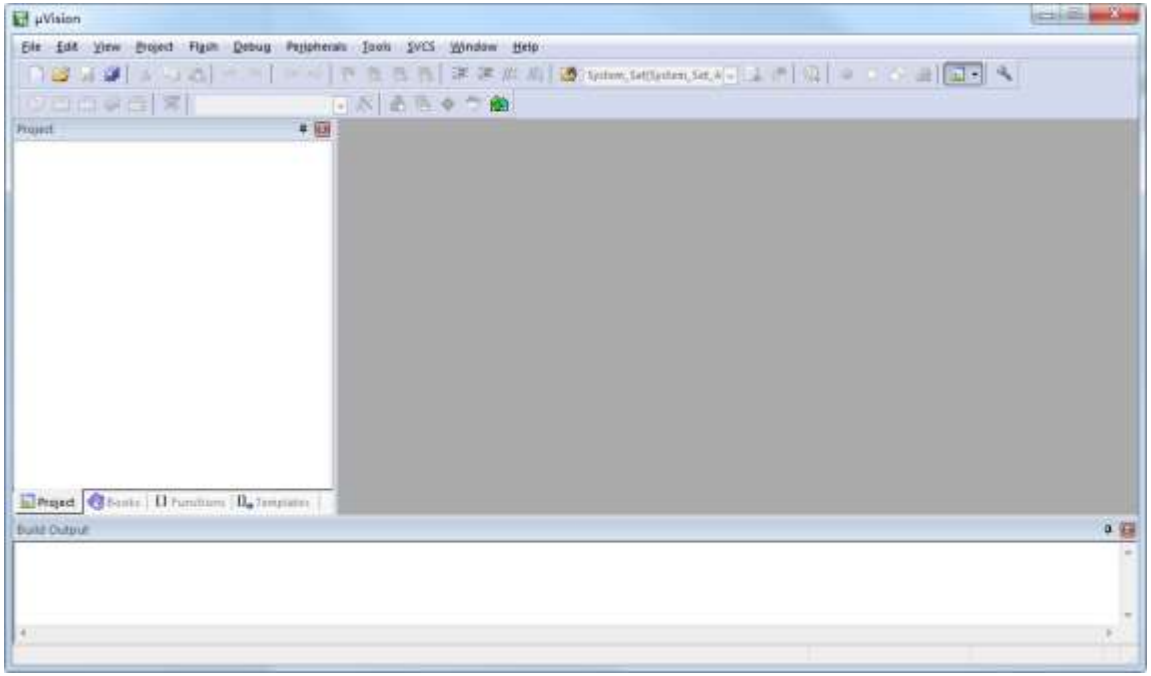
3、调试工具；

4、其他：RTOS、开源协议栈。

3.1 STM32系列处理器简介

3.1.3 开发工具

1. 开发软件



1）KEIL MDK

KEIL MDK 集成了业内最领先的技术，支持Cortex-M, Cortex-A等ARM内核处理器，集成Flash烧写模块等，具备针对不同调试器的在线调试功能，已经成为ARM软件开发工具的标准。最新版本： MDK-ARM Version 5.25

可访问www.keil.com获取更多内容。

2）IAR for ARM

www.iar.com

3.1 STM32系列处理器简介

3.1.3 开发工具

1）JLINK

J-Link 是SEGGER 公司为支持仿真ARM 内核芯片推出的JTAG 仿真器。是通用的开发工具，可以用于KEIL，IAR，ADS 等平台 速度，效率，功能均比ULINK强。需要安装驱动。

2．调试工具

2）ULINK

ULINK是KEIL公司开发的仿真器，专用于KEIL MDK平台。在KEIL MDK平台下无需驱动，可直接使用。

3.1 STM32系列处理器简介

3.1.3 开发工具

3）STlink

STLink 是ST 公司为STM8和STM32系列MCU设计的调试器。需要安装驱动。

4）调试接口

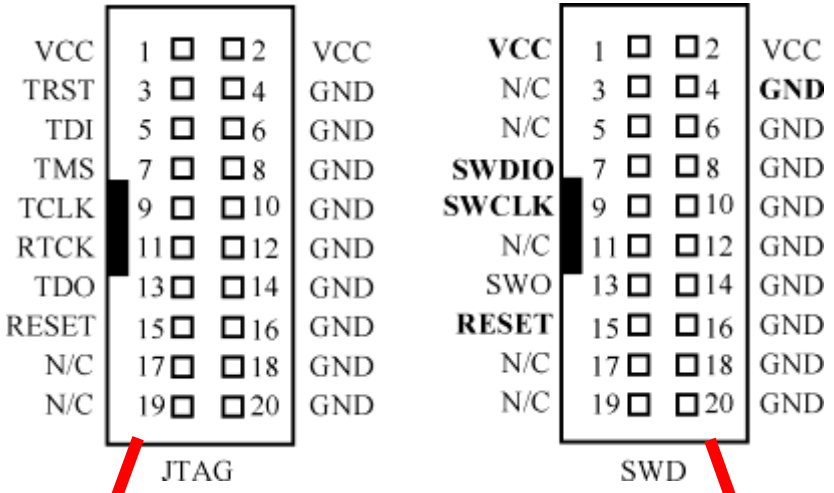
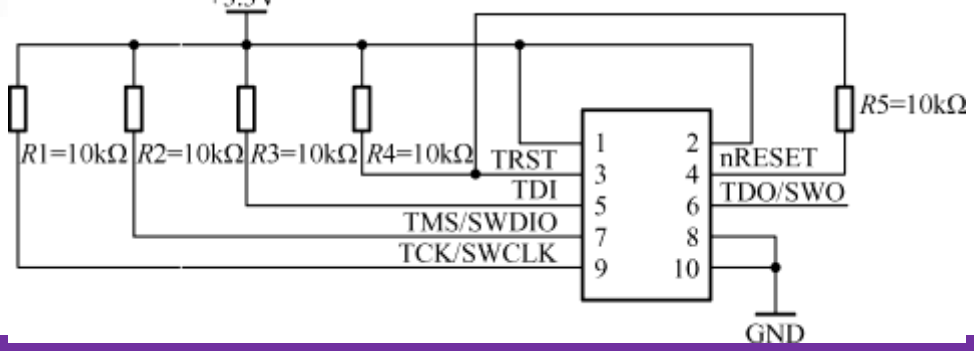
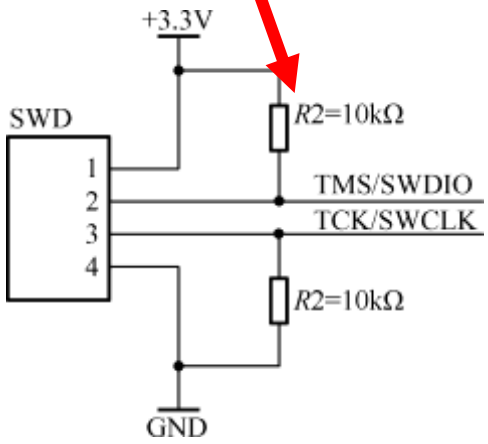


图3-4 STM32F4系列微控制器调试接口引脚图





3.1 STM32系列处理器简介

3.1.3 开发工具

3. 开发板

STM32F429I-DISC1

STM32 DISCOVERY套件是ST公司官方出品的开发板，它带有必要的硬件电路，可演示特定的设备特性，并且拥有全面的软件例程适合初学者学习。

其他常用的开发板：

安富莱

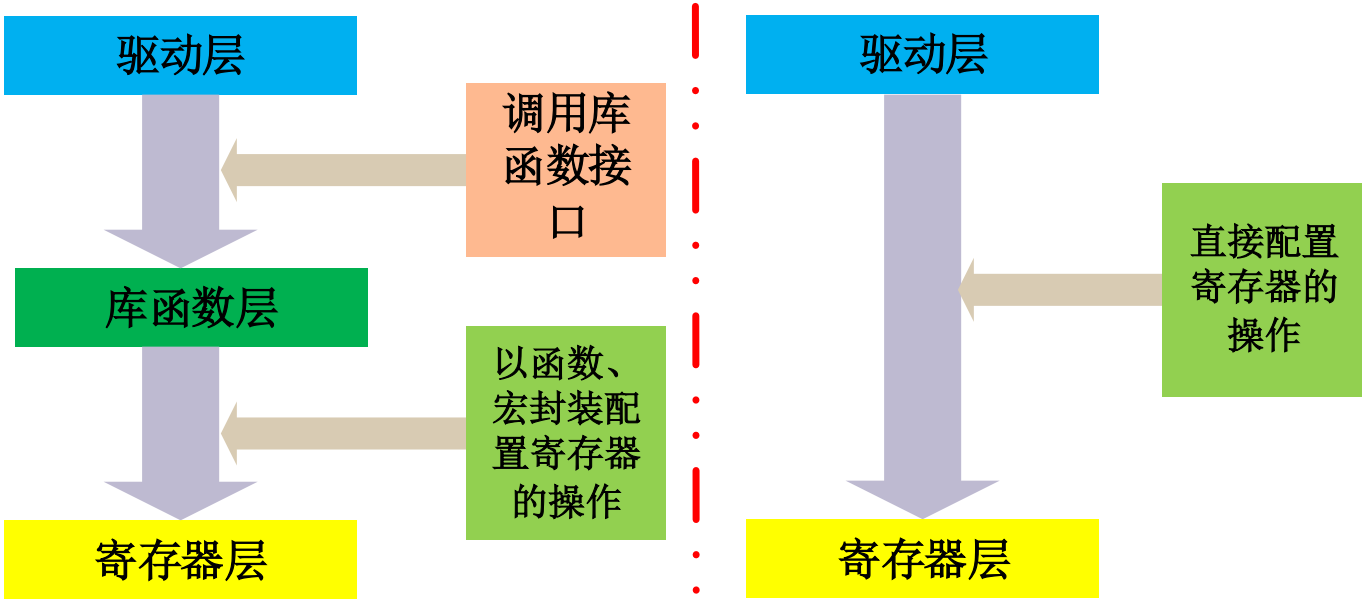
正点原子

秉火

3.1 STM32系列处理器简介

3.1.3 开发工具　　4. 软件开发形式

STM32库是由ST公司针对STM32提供的函数接口，即API (Application Program Interface)，开发者可调用这些函数接口来配置STM32的寄存器，使开发人员得以脱离最底层的寄存器操作，有开发快速，易于阅读，维护成本低等优点。



3.1 STM32系列处理器简介

3.1.3 开发工具　　4. 软件开发形式

库是架设在寄存器与用户驱动层之间的代码，向下处理与寄存器直接相关的配置，向上为用户提供配置寄存器的接口。库开发方式与直接配置寄存器方式的区别。

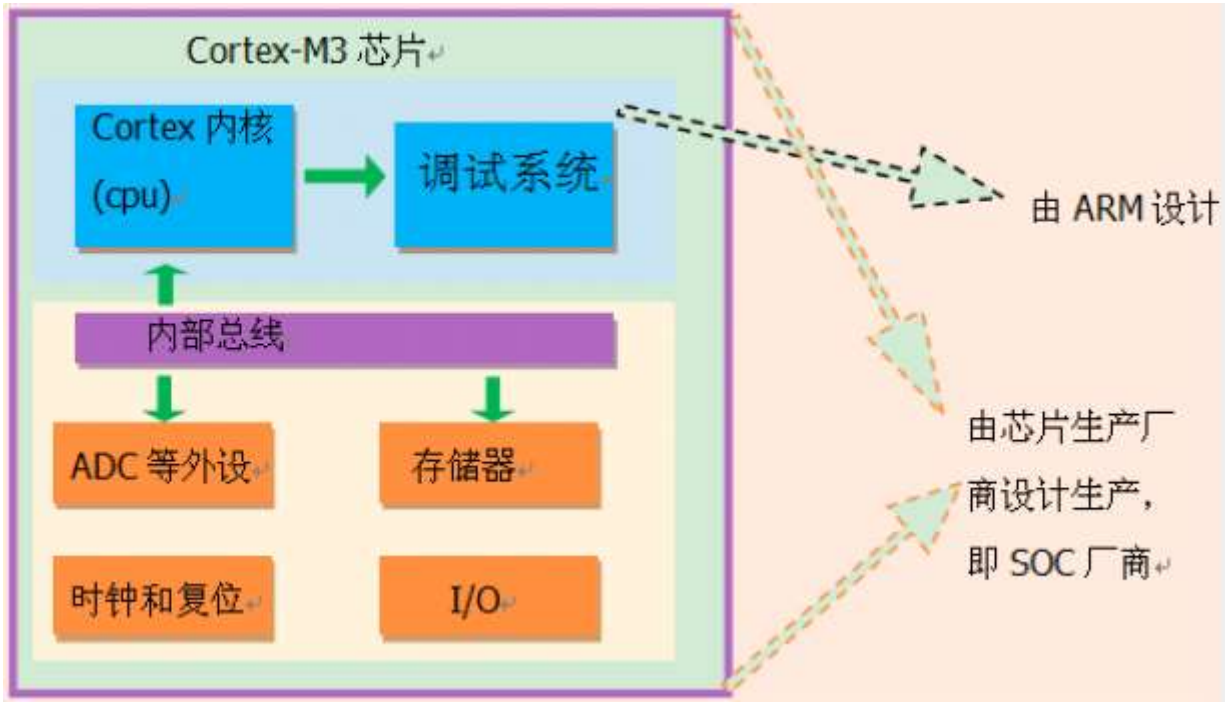
特点	库开发方式	特点	直接寄存器操作开发方式
更接程序员的思维	1）用结构体封装寄存器参数	更接近机械思维	直接针对寄存器的某些为进行置1或清0操作，能清晰看到驱动代码控制的底层对象。
	2）用宏表示参数，意义明确		
	3）用函数封装对寄存器的操作		
移植性好	代码的易读性好，使得驱动修改非常方便	运行效率	没有库函数层，省去代码为分层而消耗的资源

3.1 STM32系列处理器简介

3.1.4 STM32标准函数库介绍

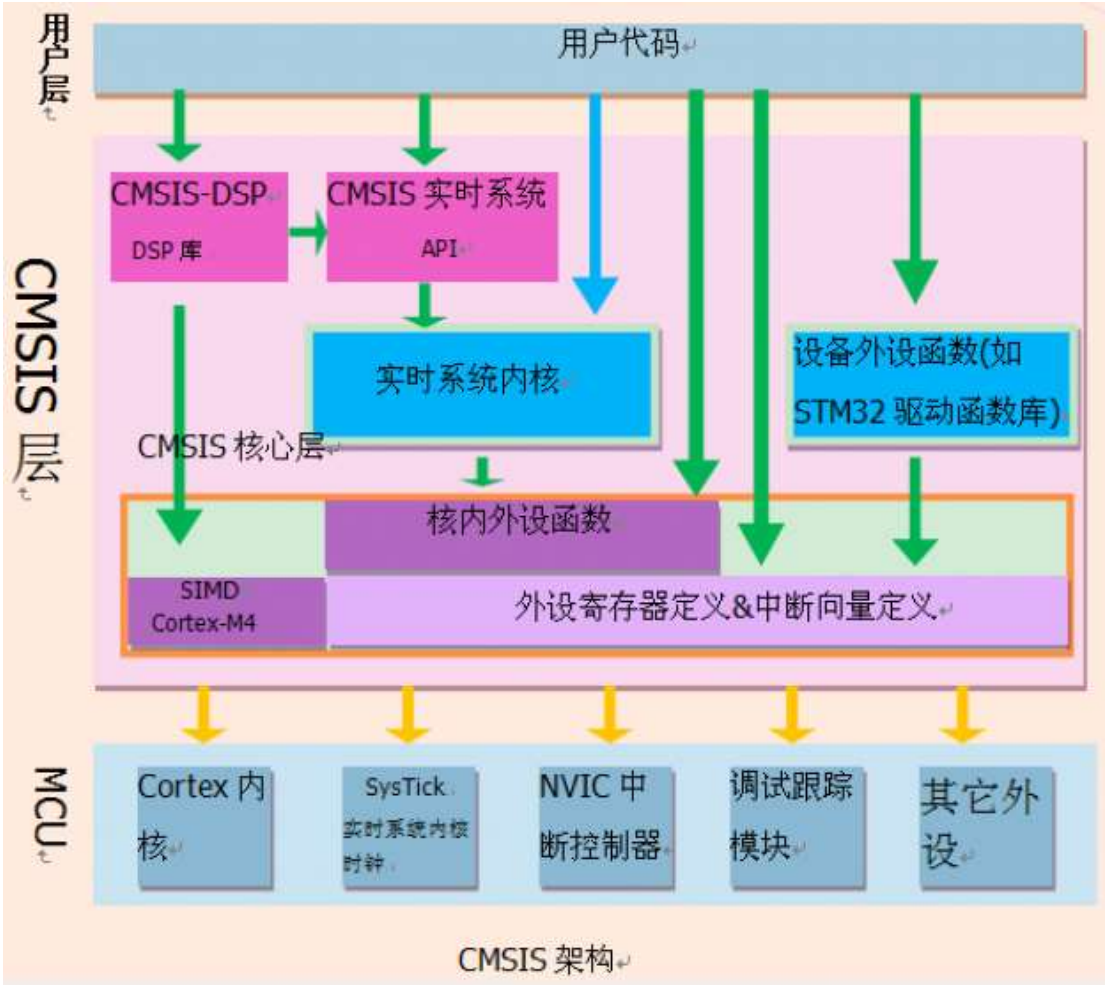
STM32的内核是**ARM**公司设计的处理器体系架构。**ST**公司或其它芯片生产厂商，负责设计的是在内核之外的部件，被称为核外外设或片上外设、设备外设。如芯片内部的模数转换外设**ADC**、串口**UART**、定时器**TIM**等。

为了解决不同的芯片厂商生产的**Cortex**微控制器软件 的兼容性问题，**ARM**与芯片厂商建立了**CMSIS**标准**(Cortex MicroController Software Interface Standard)**。



3.1 STM32系列处理器简介

CMSIS标准，实际是新建了一个软件抽象层。



CMSIS 架构

CMSIS标准中最主要的是CMSIS核心层，它包括了：

1、内核函数层：其中包含用于访问内核寄存器的名称、地址定义，主要由ARM公司提供。

2、设备外设访问层：提供了片上的核外外设的地址和中断定义，主要由芯片生产商提供。

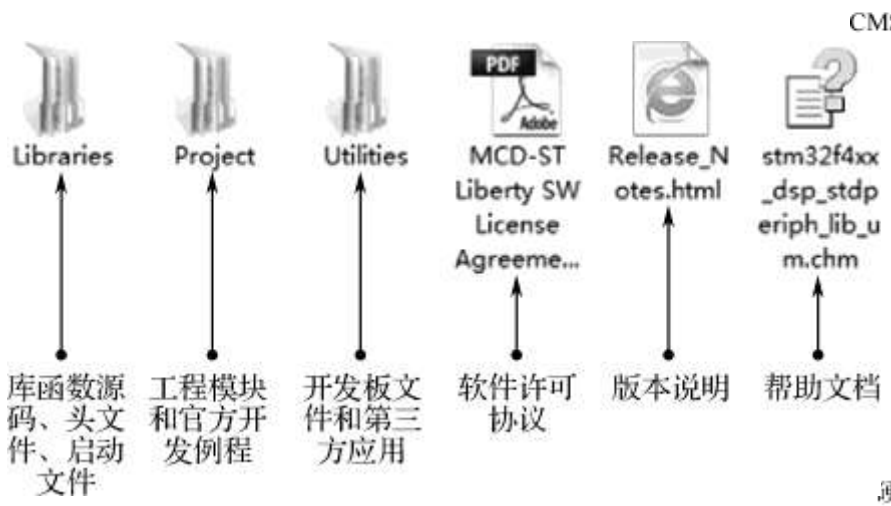
CMSIS层位于硬件层与操作系统或用户层之间，提供了与芯片生产商无关的硬件抽象层，可以为接口外设、实时操作系统提供简单的处理器软件接口，屏蔽了硬件差异，这对软件的移植是有极大的好处的。

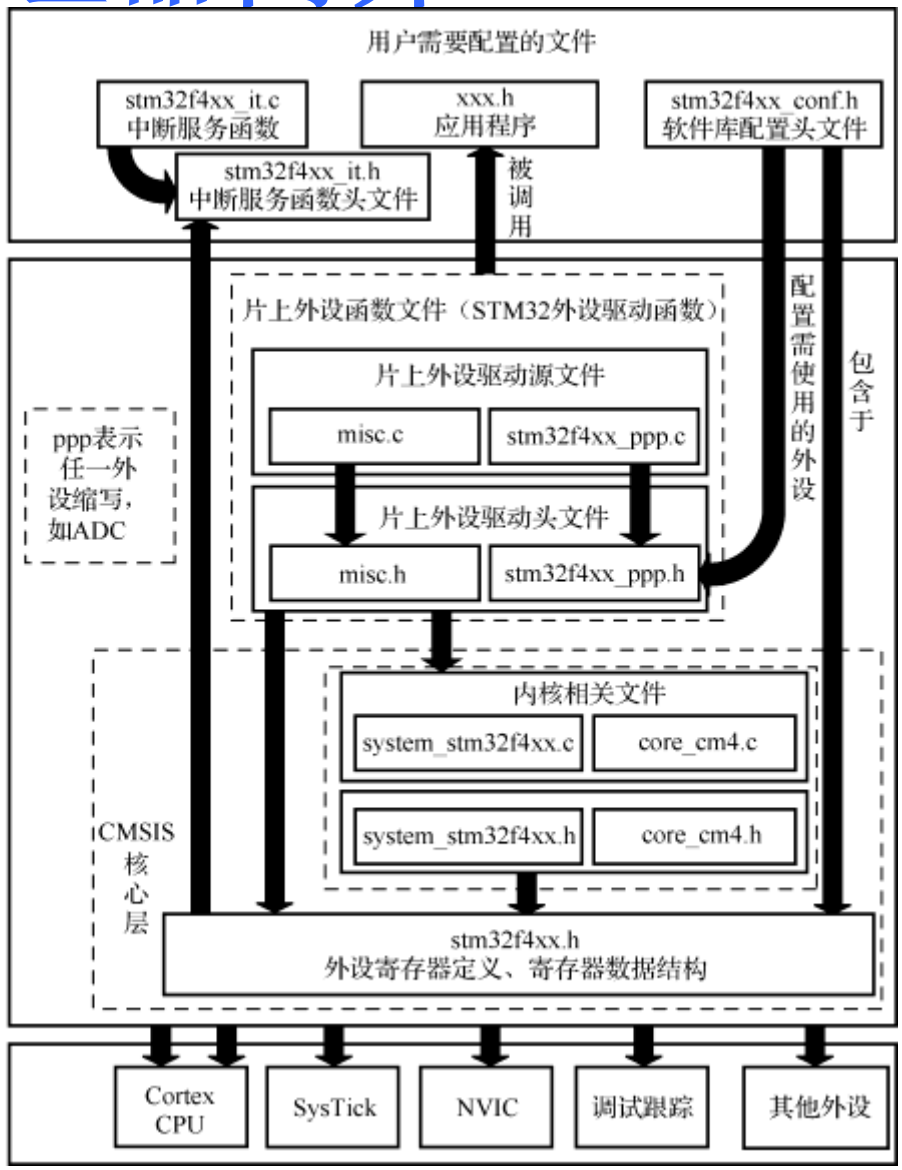
STM32的库，就是按照CMSIS标准建立的。

3.1 STM32系列处理器简介

3.1.4 STM32标准函数库介绍

http://www.stmcu.org/document/list/index/category-524





3.1 STM32系列处理器简介

3.1.4 STM32标准函数库介绍

标准函数库文件分析

STM32F4xx标准函数库文件分析			
启动文件	startup_stm32f429_439xx.s	启动文件	1必须
外设相关	stm32f4xx.h	外设寄存器定义	
	system_stm32f4xx.h	用于系统初始化	
	system_stm32f4xx.c	用于配置系统时钟	2必须
	stm32f4xx_xx.h	外设标准函数库头文件	
	stm32f4xx_xx.c	外设标准函数库源件	3必须，与外设相关
	misc.h		
	misc.c	NVIC、SysTick相关函数	4必须
内核相关	core_cm4.h	内核寄存器定义	
	core_cmFunc.h	操作内核相关，不常用	
	core_cmInstr.h	定义了内核指令	
	core_cmSimd.h	定义SIMD指令	
用户相关	stm32f4xx_it.h		
	stm32f4xx_it.c	用户编写的中断服务函数	5
	main.c	用户应用程序主程序入口	6必须
	其他应用子程序	用户自定义应用功能	7

3.1 STM32系列处理器简介

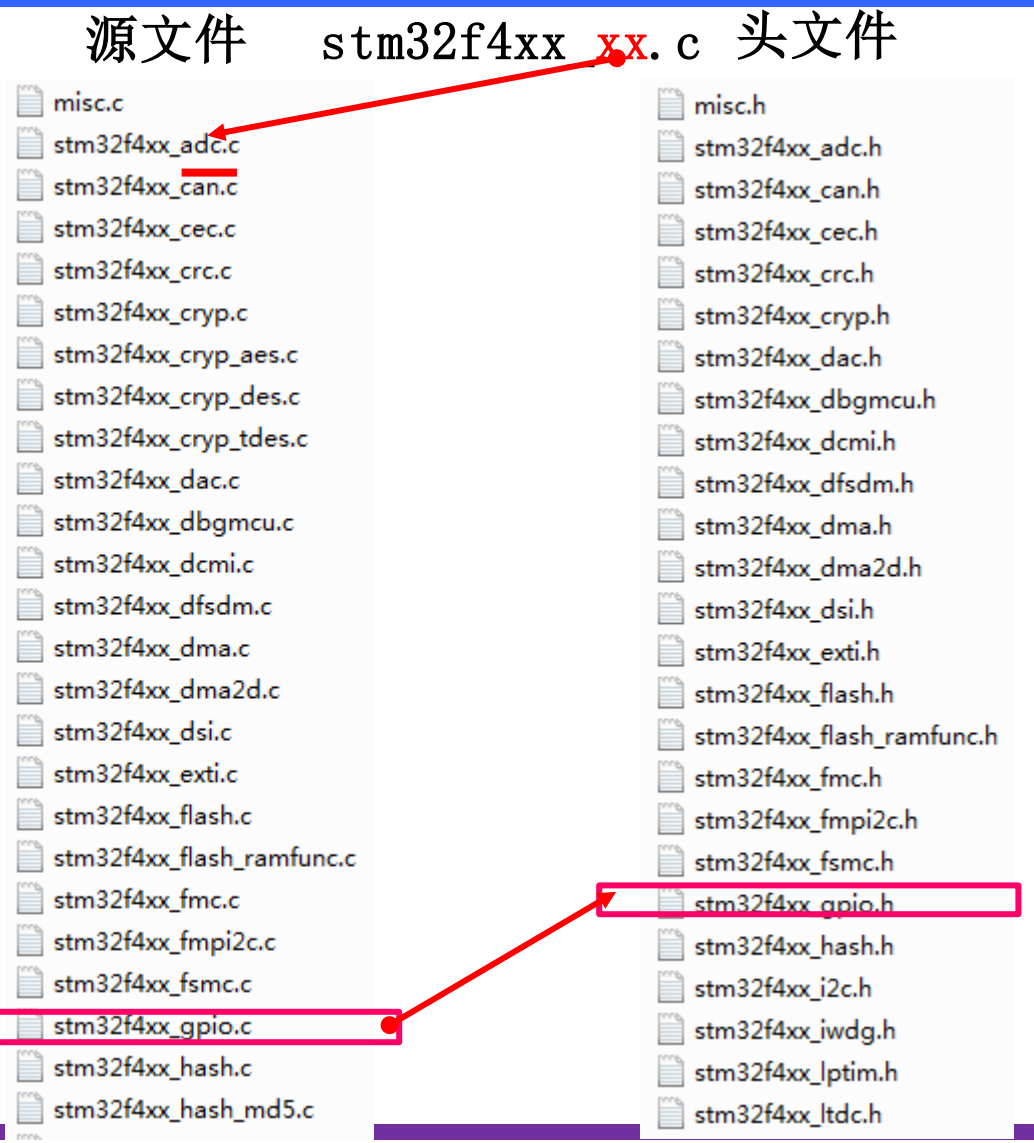
3.1.4 STM32标准函数库介绍

标准函数库使用

需要使用哪一个偏上外设即将库中，其对应的源文件添加到工程中。

同时，在**stm32f4xx_conf.h**包含对应头文件。

GPIO相关的操作函数



3.1 STM32系列处理器简介

3.1.4 STM32标准函数库介绍

标准函数库使用

stm32f4xx_it.c

```
/*****************************************************
/*                 STM32F4xx Peripherals Interrupt H
/*  Add here the Interrupt Handler for the used peri
/*  available peripheral interrupt handler's name pl
/*  file (startup_stm32f429_439xx.s).
/*****************************************************

/**
  * @}
  */
void EXTI2_IRQHandler(void)
{
  //确保是否产生了EXTI Line中断
  if(EXTI_GetITStatus(KEY1_INT_EXTI_LINE) != RESET)
  {
    // LED 取反
    LED_TOGGLE;
    //清除中断标志位
    EXTI_ClearITPendingBit(KEY1_INT_EXTI_LINE);
  }
}
```

外部中断2的服务程序

中断服务程序名字与启动文件中定义的要保持一致！

用来编写中断服务程序的一个文件，已经提供了相关片上外设的中断服务程序的框架，用户只需要填写响应中断的程序。

startup_stm32f426_439xx.s

```
EXPORT  WWDG_IRQHandler                 [WEAK]
EXPORT  PVD_IRQHandler                  [WEAK]
EXPORT  TAMP_STAMP_IRQHandler           [WEAK]
EXPORT  RTC_WKUP_IRQHandler             [WEAK]
EXPORT  FLASH_IRQHandler                [WEAK]
EXPORT  RCC_IRQHandler                  [WEAK]
EXPORT  EXTI0_IRQHandler                [WEAK]
EXPORT  EXTI1_IRQHandler                [WEAK]
EXPORT  EXTI2_IRQHandler                [WEAK]
EXPORT  EXTI3_IRQHandler                [WEAK]
EXPORT  EXTI4_IRQHandler                [WEAK]
EXPORT  DMA1_Stream0_IRQHandler         [WEAK]
EXPORT  DMA1_Stream1_IRQHandler         [WEAK]
EXPORT  DMA1_Stream2_IRQHandler         [WEAK]
EXPORT  DMA1_Stream3_IRQHandler         [WEAK]
EXPORT  DMA1_Stream4_IRQHandler         [WEAK]
EXPORT  DMA1_Stream5_IRQHandler         [WEAK]
EXPORT  DMA1_Stream6_IRQHandler         [WEAK]
EXPORT  ADC_IRQHandler                  [WEAK]
EXPORT  CAN1_TX_IRQHandler              [WEAK]
EXPORT  CAN1_RX0_IRQHandler             [WEAK]
EXPORT  CAN1_RX1_IRQHandler             [WEAK]
EXPORT  CAN1_SCE_IRQHandler             [WEAK]
EXPORT  EXTI9_5_IRQHandler              [WEAK]
EXPORT  TIM1_BRK_TIM9_IRQHandler        [WEAK]
```

3.1 STM32系列处理器简介

3.1.4 STM32标准函数库介绍

标准函数库使用

1、startup_stm32f429_439xx.s

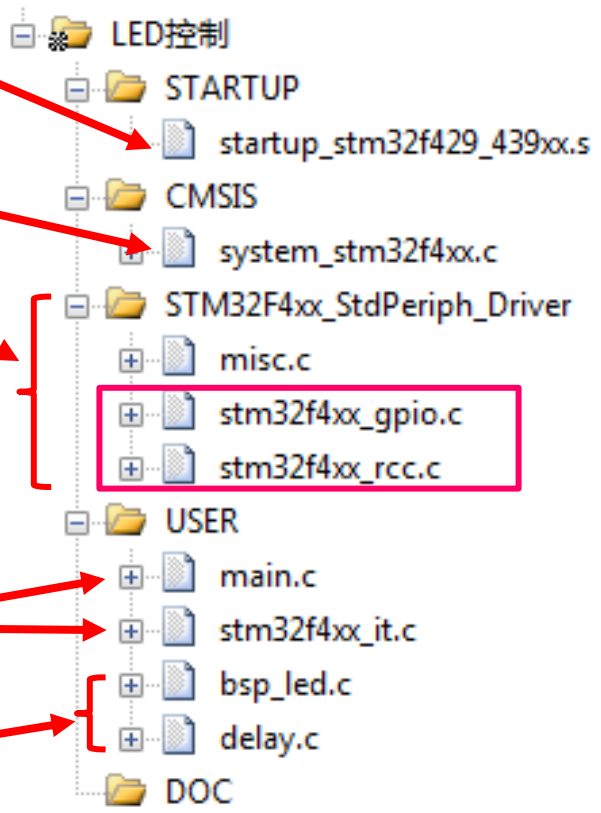
2、system_stm32f4xx.c

3、stm32f4xx_xx.c ：LED控制需要使用到GPIO，并用到了系统时钟定时生成的延时功能，因此在工程中需要包含需要包含stm32f4xx_gpio.c和stm32f4xx_rcc.c

4、stm32f4xx_it.c

5、main.c

6、其他应用程序



3.2 STM32F429微控制器结构

3.2 STM32F429微控制器结构

STM32F429微控制器属于STM32F4系列微控制器，采用了最新的180MHz的Cortex-M4处理器内核，可取代当前基于微控制器和中低端独立数字信号处理器的双片解决方案，或者将两者整合成一个基于标准内核的数字信号控制器。

STM32F429微控制器的封装形式有
LQFP100（14mm× 14mm）、
LQFP144（20mm×20mm）、
UFBGA169（7mm×7mm）、
LQFP176（24mm× 24mm）、
LQFP208（28mm×28mm）、
UFBGA176（10mm×10mm）、
TFBGA216（13mm×13mm）、
WLCSP143。

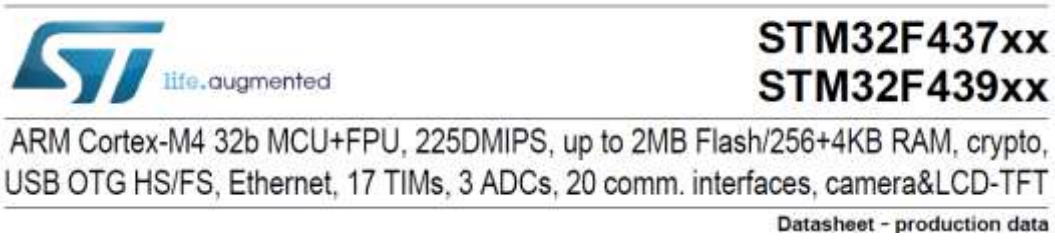
封装：LQFP176

图3-15 STM32F429T6芯片的外观图

3.2 STM32F429微控制器结构

3.2.1 芯片资源

1、对应具体芯片的数据手册一般是英文的，一般说明具体芯片的基本描述、电气特性等。



2、STM32F4XX有完整的中文编程参考手册，说明了这一系列芯片的所有功能和使用方法，一般查阅的是这一个手册。

3.2 STM32F429微控制器结构

3.2.1 芯片资源

1、内核：

-32位 高性能ARM Cortex-M4处理器

-时钟：高达180M

-支持FPU（浮点运算）和DSP指令

2、IO口：

STM32F429IGT6：176引脚 140个IO

-大部分IO口都耐5V(模拟通道除外)

-支持调试：SWD和JTAG，SWD只要2根数据线

3、存储器容量：1024K FLASH，256K SRAM

4、LCD控制器：800x600分辨率，具备专用于图像处理的专业 DMA：Chrom-Art Accelerator™ (DMA2D)

3.2 STM32F429微控制器结构

3.2.1 芯片资源

5、时钟，复位和电源管理：

① 1.8~3.6V电源和IO电压

② 上电复位，掉电复位和可编程的电压监控

③ 强大的时钟系统

-4~26M的外部高速晶振

-内部16MHz的高速RC振荡器

-内部32KHz低速RC振荡器，看门狗时钟

-内部锁相环（PLL，倍频），一般系统时钟都是外部或者内部高速时钟经过PLL倍频后得到

- 外部低速32.768K的晶振，主要做RTC时钟源

3.2 STM32F429微控制器结构

3.2.1 芯片资源

6、低功耗：

-睡眠，停止和待机三种低功耗模式

-可用电池为RTC和备份寄存器供电

7、模数转换器-ADC:

-3个12位ADC【多达24个外部测量通道】

-内部通道可以用于内部温度测量

-内置参考电压

8、数模转换器-DAC:

2个12位DAC

9、直接内存存取-DMA:

16个DMA通道，带FIFO和突发支持

支持外设：定时器，ADC,DAC，SDIO,I2S,SPI,I2C,和USART

3.2 STM32F429微控制器结构

3.2.1 芯片资源

10、定时器：多达17个定时器

-10个通用定时器（TIM2和TIM5是32位）

-2个基本定时器

-2个高级定时器

-1个系统定时器

-2个看门狗定时器

3.2 STM32F429微控制器结构

3.2.1 芯片资源

11、通信接口：多达21个通信接口

-3个I2C接口

-8个串口

-6个SPI接口

-1个SAI接口

-2个CAN2.0

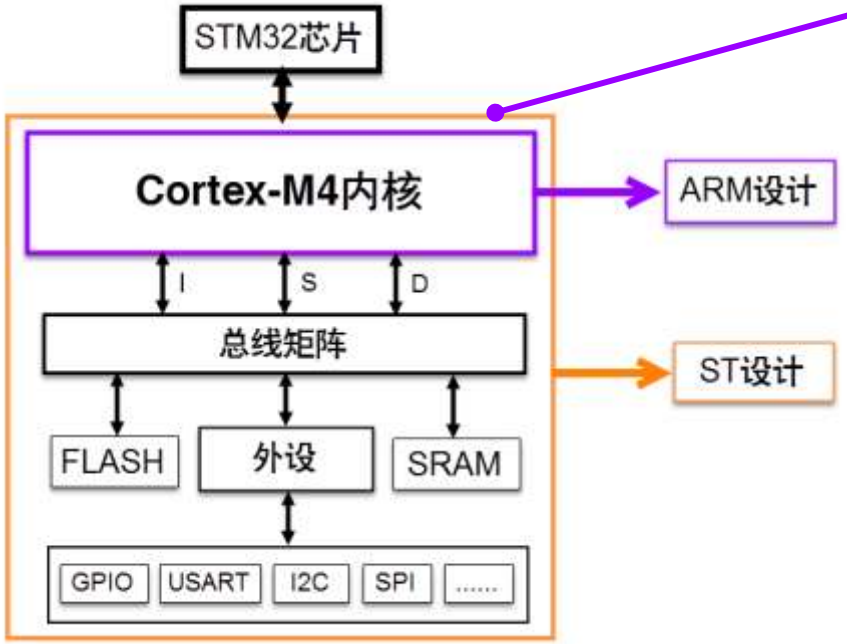
-1个SDIO

12、2个USB OTG

13、1个以太网媒体接入控制器(MAC)

3.2 STM32F429微控制器结构

3.2.2 芯片内部结构



STM32 芯片架构简图

3.2 STM32F429微控制器结构

3.2.2 芯片内部结构

1．8条主控总线
S0：I总线。
S1：D总线。
S2：S总线。
S3、S4：DMA存储器总线。
S5：DMA2外设总线。
S6：以太网DMA总线。
S7：USB OTG HS DMA总线。

2．7条被控总线
（1）内部Flash I总线（M0）。
（2）内部Flash D总线（M1）。
（3）主要内部SRAM1（112KB）总线（M2）。
（4）辅助内部SRAM2（16KB）总线（M3）。
（5）辅助内部SRAM3（64KB）总线
（6）AHB1外设。
（7）AHB2外设总线（M4）。
（8）FSMC总线（M6）。

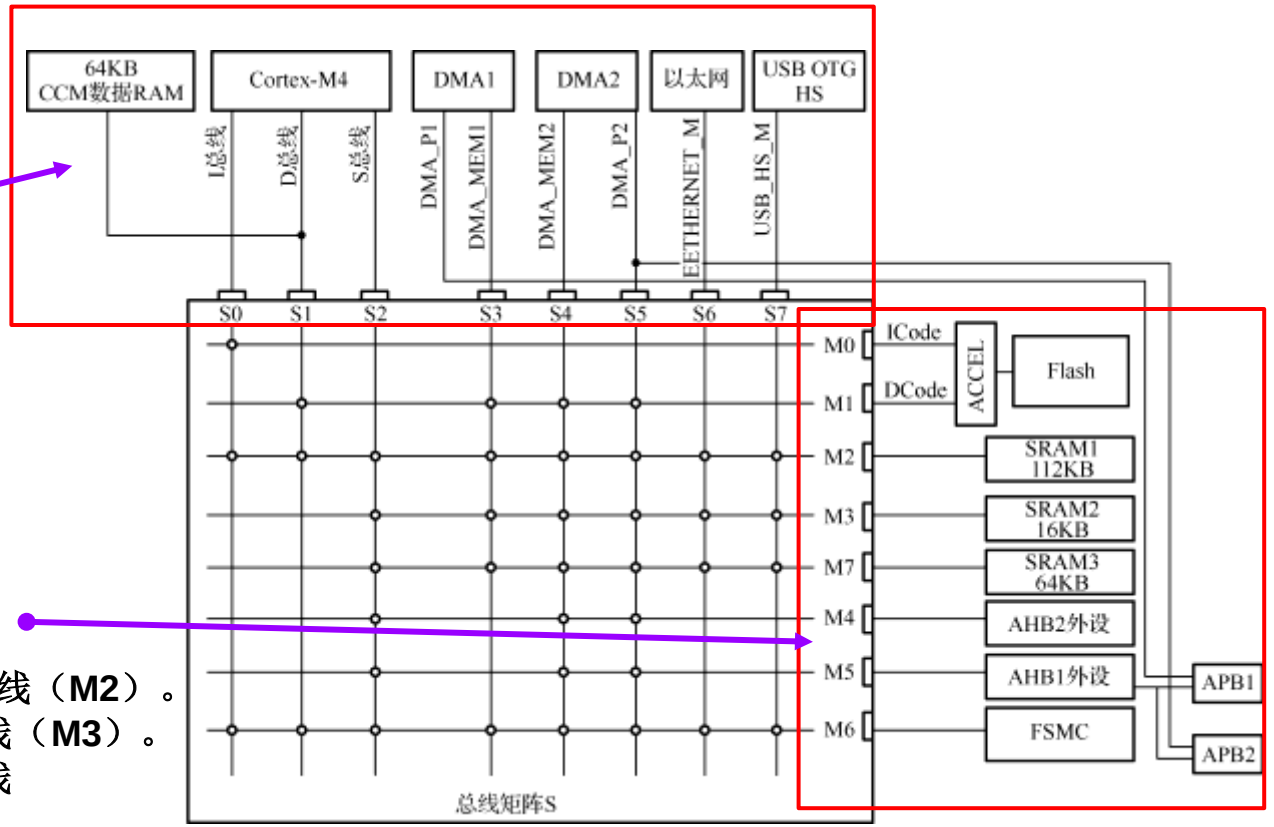


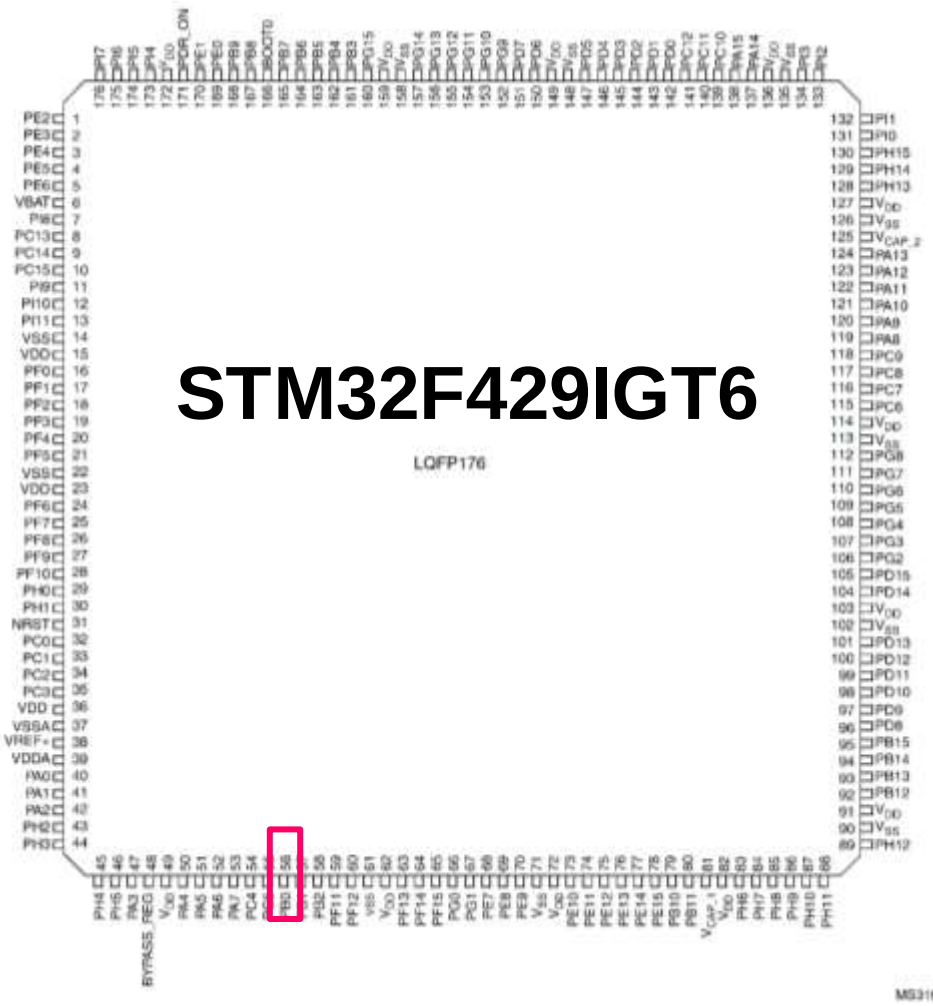
图3-16　STM32F429 IGT6芯片总线矩阵结构图

3.2 STM32F429微控制器结构

3.2.3 芯片引脚和功能

只列出了引脚的基本功能。

由于芯片内部集成功能较多，但是实际引脚有限，因此多数引脚为*复用引脚*（一个引脚可复用为多个功能）。



例如：56号引脚可以作为**PB0**、*TIM1_CH2N,TIM3_CH3,TIM8_CH2N,LCD_R3,OTG_HS_ULPI_D1,ETH_MII_RXD2,EVENTOUT,ADC12_IN8*。

56	PB0/TIM1 CH2N/TIM3 CH3/TIM8 CH2N/LCD R3/OTG HS ULPI D1/ETH MII RXD2/EVENTOUT/ADC12 IN8

3.2 STM32F429微控制器结构

3.2.3 芯片引脚和功能

手册	主要内容	说明
参考手册	片上外设的功能说明和寄存器描述	对片上每一个外设的功能和使用做了详细的说明，包含寄存器的详细描述。编程的时候需要反复查询这个手册。
数据手册	功能概览	主要讲这个芯片有哪些功能，属于概括性的介绍。芯片选型的时候首先看这个部分。
	引脚说明	**详细描述每一个引脚的功能，设计原理图的时候和写程序的时候需要参考这部分。**
	内存映射	讲解该芯片的内存映射，列举每个总线的地址和包含有哪些外设。
	封装特性	讲解芯片的封装，包含每个引脚的长度宽度等，我们画PCB封装的时候需要参考这部分的参数。

3.2 STM32F429微控制器结构

3.2.3 芯片引脚和功能

名称	缩写	说明
① 引脚序号	阿拉伯数字表示 LQFP 封装，英文字母开头的表示 BGA 封装。引脚序号	
② 引脚名称	指复位状态下的引脚名称	
③ 引脚类型	S	电源引脚
	I	输入引脚
	I/O	输入/输出引脚
	FT	兼容 5V
	TTa	只支持 3V3，且直接到 ADC
④ I/O 结构	B	BOOT 引脚
	RST	复位引脚，内部带弱上拉
⑤ 注意事项	对某些 IO 要注意的事项的特别说明	
⑥ 复用功能	IO 的复用功能，过 GPIOx_AFR 寄存器来配置选择。一个 IO 口可以复用	
⑦ 额外功能	IO 的额外功能，通过直连的外设寄存器配置来选择。	

3.2 STM32F429微控制器结构

3.2.3 芯片引脚和功能

数据手册：P51

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
1	1	B2	A2	1	D8	1	A3	PE2	I/O	FT		TRACECLK, SPI4_SCK, SAI1_MCLK_A, ETH_MII_TXD3, FMC_A23, EVENTOUT	
2	2	C1	A1	2	C10	2	A2	PE3	I/O	FT		TRACED0, SAI1_SD_B, FMC_A19, EVENTOUT	
3	3	C2	B1	3	B11	3	A1	PE4	I/O	FT		TRACED1, SPI4_NSS, SAI1_FS_A, FMC_A20, DCMI_D4, LCD_B0, EVENTOUT	

3.2 STM32F429微控制器结构

3.2.3 芯片引脚和功能

引脚分类	引脚说明说明
电源	(VBAT)、(VDD VSS)、(VDDA VSSA)、(VREF+ VREF-)等
晶振 IO	主晶振 IO，RTC 晶振 IO
下载 IO	用于 JTAG 下载的 IO：JTMS、JTCK、JTDI、JTDO、NJTRST
BOOT IO	BOOT0、BOOT1，用于设置系统的启动方式
复位 IO	NRST，用于外部复位
GPIO	专用器件接到专用的总线，比如 I2C，SPI，SDIO，FSMC，DCMI 这些总线的器件需要接到专用的 IO普通的元器件接到 GPIO，比如蜂鸣器，LED，键等元器件用普通的GPIO即如果还有剩下的 IO，可根据项目需要引出或者不引出

3.2 STM32F429微控制器结构

3.2.4 电源系统

STM32F429微控制器的工作电压（V_{DD}）范围为1.8～3.6V。嵌入式线性调压器用于提供内部1.2V数字电源。当主电源V_{DD}断电时，可通过V_{BAT}引脚为实时时钟（RTC）、RTC备份寄存器和备份SRAM（BKP SRAM）供电。电源系统主要分为备份电路、ADC电路及调压器主供电电路三部分

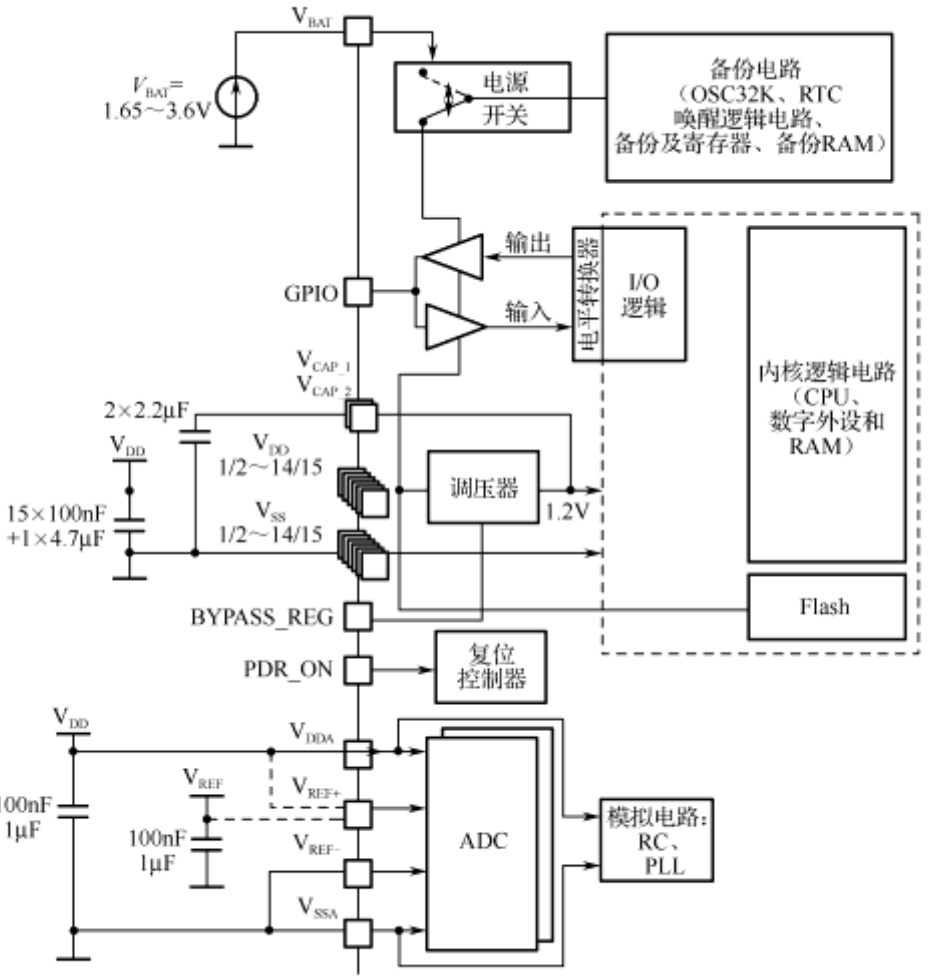


图3-19　STM32F429微控制器内部电源系统结构图

3.2 STM32F429微控制器结构

3.2.5 复位系统

STM32F429微控制器的复位共有三种类型，分别为系统复位、电源复位和备份域复位。

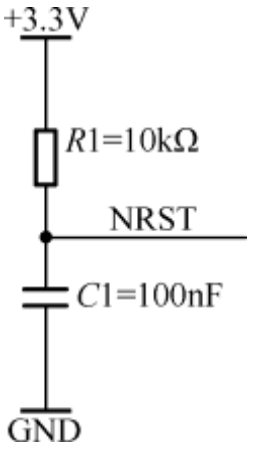


图3-23 RC复位电路

1. 系统复位

（1）NRST引脚低电平（外部复位）。
（2）窗口看门狗计数结束（WWDG复位）。
（3）独立看门狗计数结束（IWDG复位）。
（4）软件复位（SW复位）。
（5）低功耗管理复位。

2. 电源复位

当发生以下事件之一时，就会产生电源复位。

（1）上电/掉电（POR/PDR）复位或欠压（BOR）复位。
（2）在退出待机模式时。

3.2 STM32F429微控制器结构

3.2.5 复位系统

备份域复位会将所有RTC寄存器和RCC备份域控制寄存器（RCC_BDCR）复位为各自的复位值。

当发生以下事件之一时，就会产生备份域复位。

（1）软件复位，通过将RCC_BDCR中的BDRST位置1触发。

（2）在电源V_{DD}和V_{BAT}都已掉电后，其中任何一个又再上电。

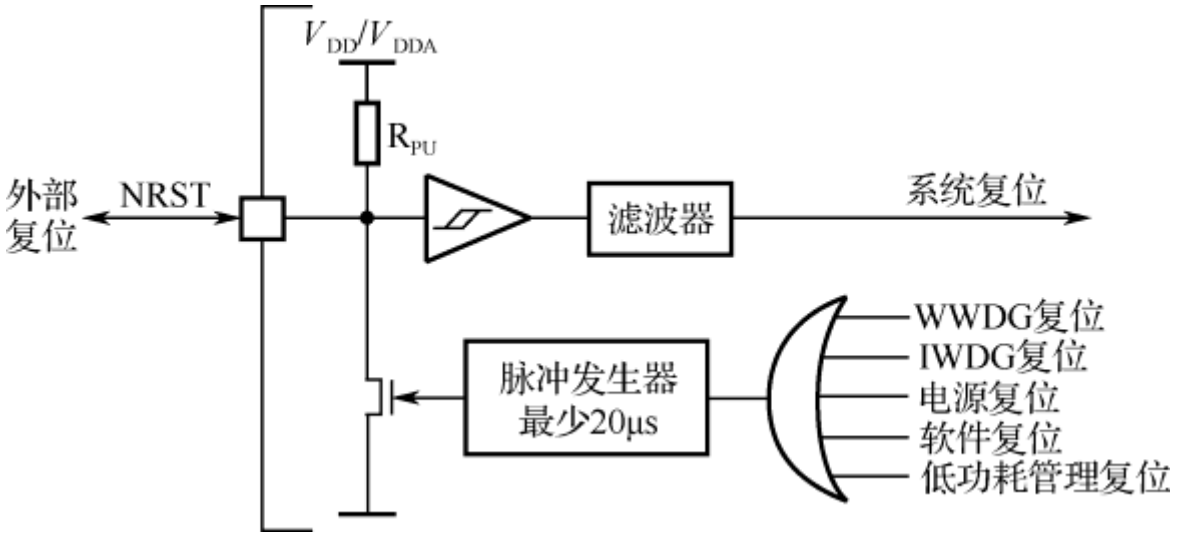


图3-24 备份域复位内部结构图

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.1 存储器映射

1、存储器内部各个块的功能由芯片厂商，给存储器分配地址的过程就称为存储器映射。

2、STM32F4XX在存储空间上采用冯诺依曼结构，在寻址上采用哈佛结构。可寻址的存储空间为2^{32}=4G Byte，共被分为8个块。每个存储单元都有一个地址，存储单位为字节。

3、编写应用程序一般固化在Flash中，程序运行过程中使用的变量主要杯定义在SRAM中（Block1），控制片上外设使用到的寄存器被配备在Block2上，当需要外部扩展存储空间时使用Block3、4。

序号	用途	地址范围
Block 0	SRAM（FLASH）	0x0000 0000 ~ 0x1FFF FFFF(512MB)
Block 1	SRAM	0x2000 0000 ~ 0x3FFF FFFF(512MB)
Block 2	片上外设	0x4000 0000 ~ 0x5FFF FFFF(512MB)
Block 3	FMC的bank1 ~ bank2	0x6000 0000 ~ 0x7FFF FFFF(512MB)
Block 4	FMC的bank3 ~ bank4	0x8000 0000 ~ 0x9FFF FFFF(512MB)
Block 5	FMC	0xA000 0000 ~ 0xCFFF FFFF(512MB)
Block 6	FMC	0xD000 0000 ~ 0xDFFF FFFF(512MB)
Block 7	Cortex-M4内部外设	0xE000 0000 ~ 0xFFFF FFFF(512MB)

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.1 存储器映射

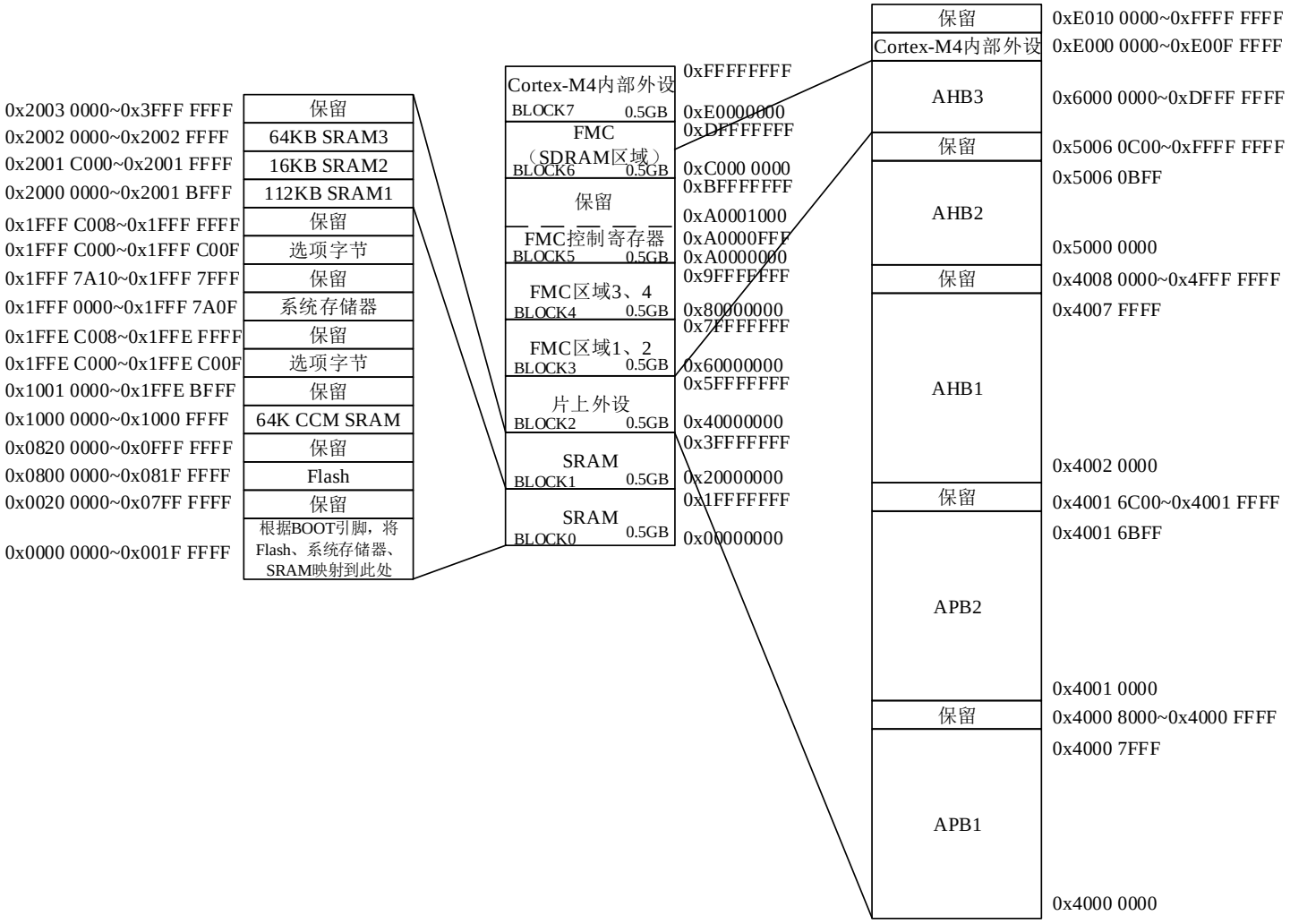


图3-25　STM32F429微控制器存储空间映射图

3.3.3 自举配置

1、自举配置确定控制器复位后，内核从什么地方读取执行指令。

2、复位后，在SYSCLK的第四个上升沿锁存BOOT引脚的值。复位后，用户可以通过设置BOOT1和BOOT0引脚来选择需要的自举模式。
BOOT0为专用引脚，而BOOT1则与GPIO引脚共用。一旦完成对BOOT1的采样，相应GPIO引脚即进入空闲状态，可用于其它用途。

在STM32F4xx中，可通过BOOT[1:0]引脚选择三种不同的自举模式，

自举模式选择引脚		自举模式	自举空间
BOOT1	BOOT0		
x	0	主 Flash	选择主 Flash 作为自举空间
0	1	系统存储器	选择系统存储器作为自举空间
1	1	嵌入式 SRAM	选择嵌入式 SRAM 作为自举空间

例如：通常现在主Flash作为复位后指令执行的位置，那么就需要将引脚BOOT0接到低电平上。

3.3.3 寄存器映射

把片上外设对应的寄存器在存储空间上分配地址的过程就叫寄存器映射。

与存储单元一样，每个寄存器（一般都是32位的）都有一个寻址地址。

寄存器是一个很重要的概念。

- 应用程序对片上外的设初始化和控制是通过对片上外设对应一系列寄存器的修改、读写来实现的。

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

- **STM32F429的各个片上外设通过APB1、APB2总线和AHB1、AHB2总线连接到内核上。**
- **每个总线都有对应的基地址，挂载在相应总线上的外设寄存器的地址，都以相应总线的基地址进行偏移。**

具体可以查看STM32F4XX参考手册（RM0090） p52-p54。

各总线基地址：

总线名称	总线基地址	相对外设基地址的偏移
APB1	0x4000 0000	0x0
APB2	0x4001 0000	0x0001 0000
AHB1	0x4002 0000	0x0002 0000
AHB2	0x5000 0000	0x1000 0000
AHB3	0x6000 0000	已不属于片上外设

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

GPIO(通用输入输出端口)挂载在AHB1总线上。

GPIO基地址:

外设名称	外设基地址	相对AHB1总线的地址偏移
GPIOA	0x4002 0000	0x0
GPIOB	0x4002 0400	0x0000 0400
GPIOC	0x4002 0800	0x0000 0800
GPIOD	0x4002 0C00	0x0000 0C00
GPIOE	0x4002 1000	0x0000 1000
GPIOF	0x4002 1400	0x0000 1400
GPIOG	0x4002 1800	0x0000 1800
GPIOH	0x4002 1C00	0x0000 1C00

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

1、GPIOH端口的寄存器列表

寄存器名称	寄存器地址	相对GPIOH基址的偏移
GPIOH_MODER	0x4002 1C00	0x00
GPIOH_OTYPER	0x4002 1C04	0x04
GPIOH_OSPEEDR	0x4002 1C08	0x08
GPIOH_PUPDR	0x4002 1C0C	0x0C
GPIOH_IDR	0x4002 1C10	0x10
GPIOH_ODR	0x4002 1C14	0x14
GPIOH_BSRR	0x4002 1C18	0x18
GPIOH_LCKR	0x4002 1C1C	0x1C
GPIOH_AFRL	0x4002 1C20	0x20
GPIOH_AFRH	0x4002 1C24	0x24

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

2、GPIOx端口数据输出寄存器ODR描述

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	③
Reserved																②
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	③
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0	②
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	①

位31:16 保留，必须保持复位值。
位15:0 ODR*y*[15:0]：端口输出数据（*y*=0～15）这些位可通过软件读取和写入。 ④

①是寄存器中位段的操作权限，r表示只读，w表示只写，rw表示可读可写。②是位段名。③是位段编号，从0开始。④是对寄存器各位段的使用说明，通过这一部分可以得到这一个寄存器所能实现的功能。

端口有16个引脚，编号对应0-15。数据输出寄存器的低16为对应于端口的没一个引脚。

3.3.3 寄存器映射

3、让GPIOH端口的16个引脚输出高电平

通过绝对地址访问内存单元

```
1 // GPIOH 端口全部输出 高电平
2 *(unsigned int*)(0x40021C14) = 0xFFFF;
```

问题：

1）、0X40021C14 是GPIOH输出数据寄存器ODR的地址，如何找到？

2）、(unsigned int*)的作用是什么?

2）、学会使用C语言的 * 号

3.3.3 寄存器映射

4、通过宏定义方式访问内存单元

```
1 // GPIOH 端口全部输出 高电平
2 #define GPIOH_ODR     (unsignedint*)(0x40021C14)
3 * GPIOH_ODR = 0xFFFF;
```

为了方便操作，我们干脆把指针操作“*”也定义到寄存器别名里面

```
1 // GPIOH 端口全部输出 高电平
2 #define GPIOH_ODR    *(unsignedint*)(0x40021C14)
3 GPIOH_ODR = 0xFFFF;
```

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

1、总线和外设基址宏定义

定义在头文件STM32F4xx.h中

```
/* 外设基地址 */
#define PERIPH_BASE           ((unsigned int)0x40000000)
/* 总线基地址 */
#define APB1PERIPH_BASE       PERIPH_BASE
#define APB2PERIPH_BASE       (PERIPH_BASE + 0x00010000)
#define AHB1PERIPH_BASE       (PERIPH_BASE + 0x00020000)
#define AHB2PERIPH_BASE       (PERIPH_BASE + 0x10000000)
/* GPIO外设基地址 */
#define GPIOA_BASE            (AHB1PERIPH_BASE + 0x0000)
#define GPIOB_BASE            (AHB1PERIPH_BASE + 0x0400)
#define GPIOC_BASE            (AHB1PERIPH_BASE + 0x0800)
#define GPIOD_BASE            (AHB1PERIPH_BASE + 0x0C00)
#define GPIOE_BASE            (AHB1PERIPH_BASE + 0x1000)
#define GPIOF_BASE            (AHB1PERIPH_BASE + 0x1400)
#define GPIOG_BASE            (AHB1PERIPH_BASE + 0x1800)
#define GPIOH_BASE            (AHB1PERIPH_BASE + 0x1C00)
/*使用把地址强制转换成GPIO_TypeDef类型地址*/
#define GPIOA                 ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                 ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                 ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                 ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                 ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOF                 ((GPIO_TypeDef *) GPIOF_BASE)
#define GPIOG                 ((GPIO_TypeDef *) GPIOG_BASE)
#define GPIOH                 ((GPIO_TypeDef *) GPIOH_BASE)
```

3.3 STM32F4系列微控制器存储器映射和寄存器

3.3.3 寄存器映射

2、使用结构体封装寄存器

定义在头文件
STM32F4xx.h中

```
typedef unsigned           int uint32_t; /*无符号32位变量*/
typedef unsigned short     int uint16_t; /*无符号16位变量*/

/* GPIO寄存器列表 */
typedef struct {
     uint32_t MODER;    /*GPIO模式寄存器                地址偏移：0x00        */
     uint32_t OTYPER;   /*GPIO输出类型寄存器            地址偏移：0x04        */
     uint32_t OSPEEDR;  /*GPIO输出速度寄存器            地址偏移：0x08        */
     uint32_t PUPDR;    /*GPIO上拉/下拉寄存器           地址偏移：0x0C        */
     uint32_t IDR;      /*GPIO输入数据寄存器            地址偏移：0x10        */
     uint32_t ODR;      /*GPIO输出数据寄存器            地址偏移：0x14        */
     uint16_t BSRRL;    /*GPIO置位/复位寄存器低16位部分 地址偏移：0x18        */
     uint16_t BSRRH;    /*GPIO置位/复位寄存器高16位部分 地址偏移：0x1A        */
     uint32_t LCKR;     /*GPIO配置锁定寄存器            地址偏移：0x1C        */
     uint32_t AFR[2];   /*GPIO复用功能配置寄存器        地址偏移：0x20-0x24   */
} GPIO_TypeDef;
```

注意：寄存器封装形式可以自定义！

3.3.3 寄存器映射

例：将GPIOH端口的10号引脚先输出低电平，再输出高电平。

```
// PH10输出输出低电平
GPIOH->ODR &= ~(1<<10);

// PH10输出输出高电平
GPIOH->ODR |= (1<<10);
```

注意：

1）在对寄存器操作的时候一般通过读-修改-写的方式实现。

常用操作有：位与&（清零），位或|（置位），异或^（取反）；

2）选择合适的操作和对应的屏蔽字。