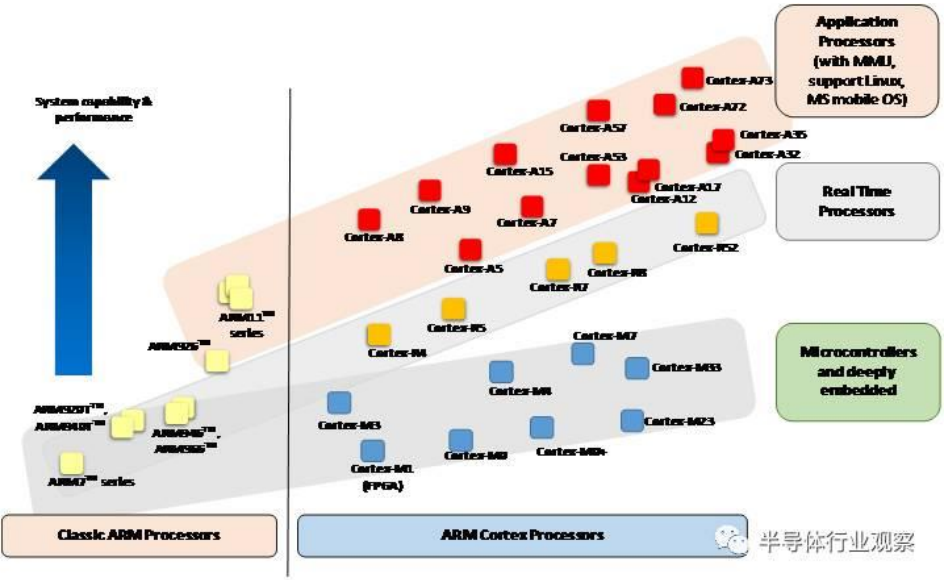
第4章 Cortex-Mx MCU [系统]

ARM Cortex



Application Processors（应用处理器）

–面向移动计算，智能手机，服务器等市场的的高端处理器。这类处理器运行在很高的时钟频率（超过1GHz），支持像Linux，Android，MS Windows和移动操作系统等完整操作系统需要的内存管理单元（MMU）

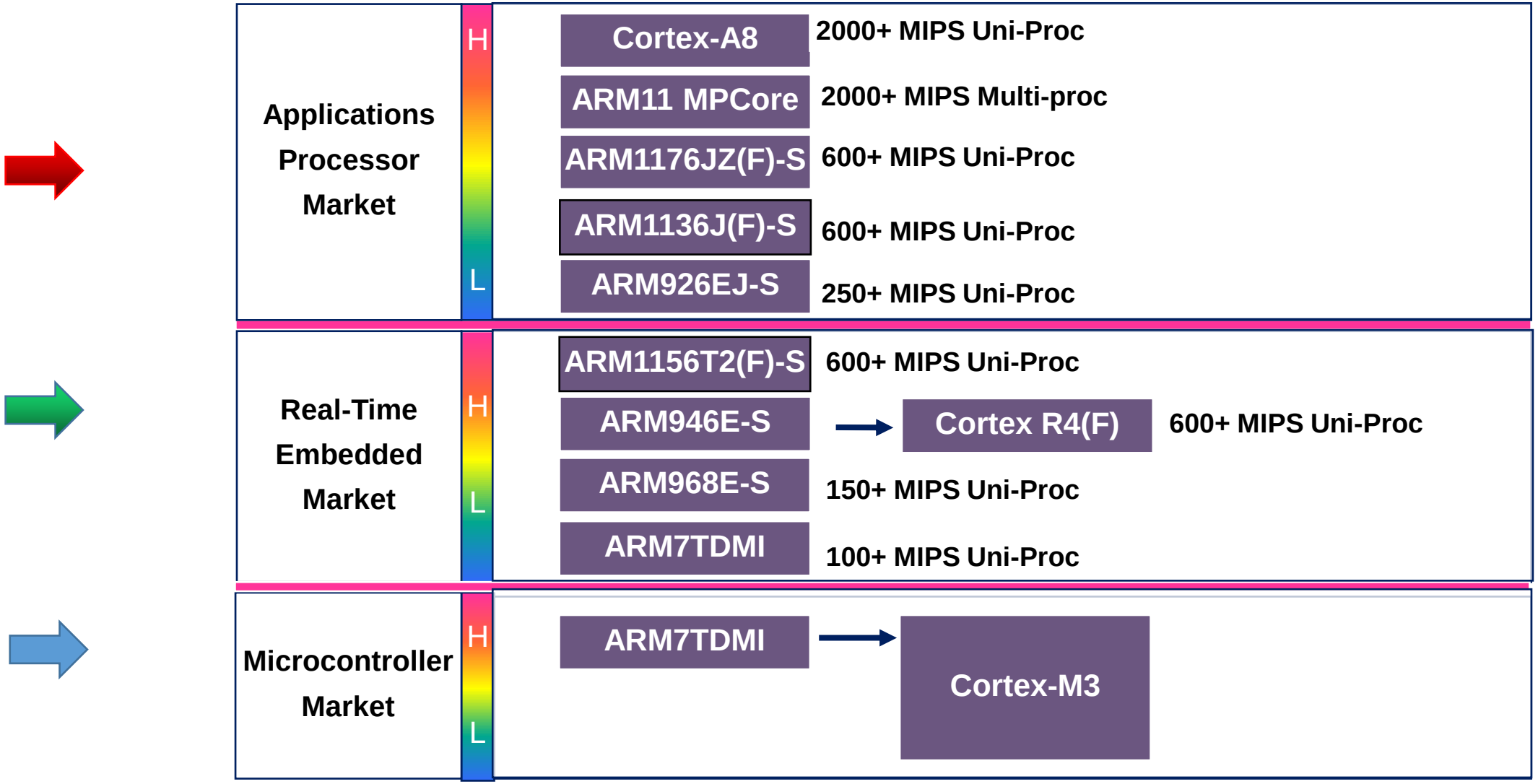
Real-time Processors （实时处理器）

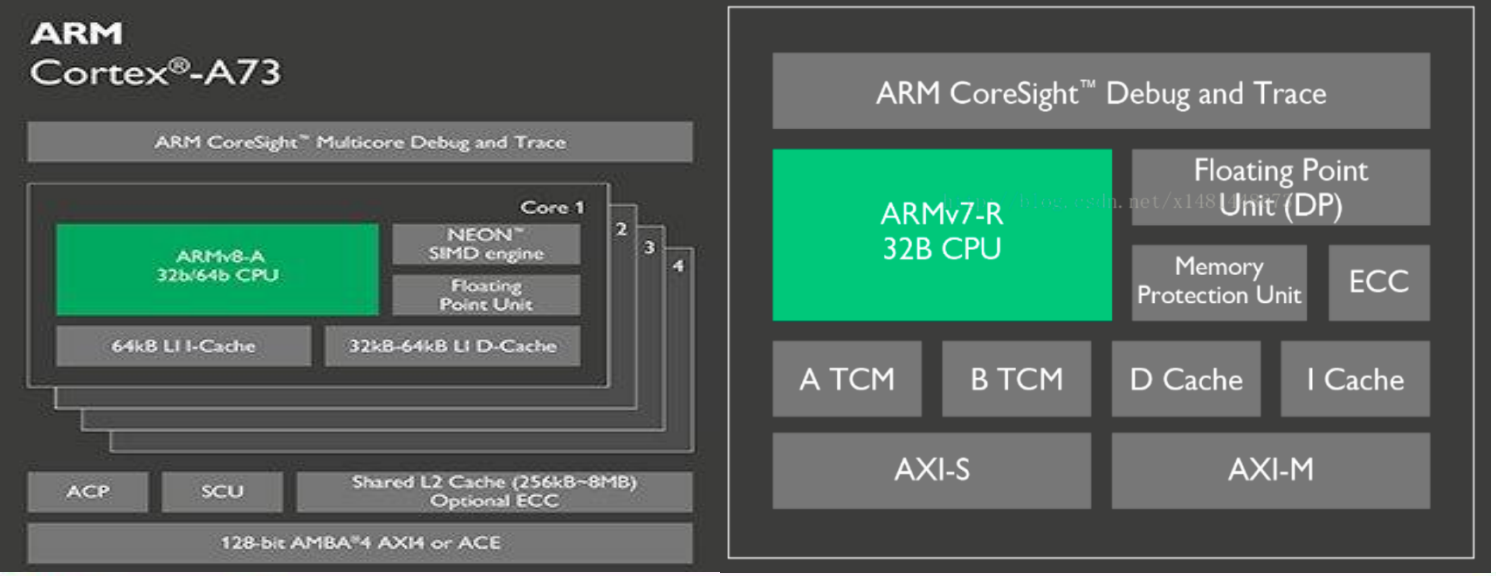
–面向实时应用的高性能处理器系列，例如硬盘控制器，汽车传动系统和无线通讯的基带控制。实时处理器运行在比较高的时钟频率（例如200MHz 到 >1GHz ），响应延迟非常低。可支持大量的实时操作系统（RTOS）

Microcontroller Processors（微控制器处理器）

–微控制器处理器通常设计成面积很小和能效比很高。通常这些处理器的流水线短，最高时钟频率较低(百MHz)，面向单片机和深度嵌入式系统市场

2019年全球Arm芯片的出货量达到64亿颗，其中Cortex-M处理器的出货量达到惊人的43亿颗，这主要是因为嵌入式终端智能需求的爆发性增长


Applications Processor Market
H
L
Cortex-A8
2000+ MIPS Uni-Proc
ARM11 MPCore
2000+ MIPS Multi-proc
ARM1176JZ(F)-S
600+ MIPS Uni-Proc
ARM1136J(F)-S
600+ MIPS Uni-Proc
ARM926EJ-S
250+ MIPS Uni-Proc
Real-Time Embedded Market
H
L
ARM1156T2(F)-S
600+ MIPS Uni-Proc
ARM946E-S
Cortex R4(F)
600+ MIPS Uni-Proc
ARM968E-S
150+ MIPS Uni-Proc
ARM7TDMI
100+ MIPS Uni-Proc
Microcontroller Market
H
L
ARM7TDMI
Cortex-M3



处理器	描述
Cortex-M0	面向低成本，超低功耗的微控制器和深度嵌入应用的非常小的处理器（最小 12K 门电路）
Cortex-M0+	针对小型嵌入式系统的最高能效的处理器，与 Cortex-M0 处理器接近的尺寸大小和编程模式，但是具有扩展功能，如单周期 I/O 接口和向量表重定位功能
Cortex-M1	针对 FPGA 设计优化的小处理器，利用 FPGA 上的存储器块实现了紧耦合内存（TCM）。和 Cortex-M0 有相同的指令集
Cortex-M3	针对低功耗微控制器设计的处理器，面积小但是性能强劲，支持可以处理器快速处理复杂任务的丰富指令集。具有硬件除法器和乘加指令（MAC）.并且，M3 支持全面的调试和跟踪功能，使软件开发者可以快速的开发他们的应用
Cortex-M4	不但具备 Cortex-M3 的所有功能，并且扩展了面向数字信号处理（DSP）的指令集，比如单指令多数据指令（SMID）和更快的单周期 MAC 操作。此外，它还有一个可选的支持 IEEE754 浮点标准的单精度浮点运算单元
Cortex-M7	针对高端微控制器和数据处理密集的应用开发的高性能处理器。具备 Cortex-M4 支持的所有指令功能，扩展支持双精度浮点运算，并且具备扩展的存储器功能，例如 Cache 和紧耦合存储器（TCM）
Cortex-M23	面向超低功耗，低成本应用设计的小尺寸处理器，和 Cortex-M0 相似，但是支持各种增强的指令集和系统层面的功能特性。M23 还支持 TrustZone 安全扩展
Cortex-M33	主流的处理器设计，与之前的 Cortex-M3 和 Cortex-M4 处理器类似，但系统设计更灵活，能耗比更高效，性能更高。M33 还支持 TrustZone 安全扩展

Advanced 先进的

以最佳功耗
实现最高性能

应用领域：

汽车
工业
医学
调制解调器
存储

Real-Time 实时的

实时任务处理

应用领域：

汽车
相机
工业
医学

Microcontrollers 微处理器

最节能的
嵌入式设备

应用领域：

汽车
能源网
医学
嵌入式
智能卡
智能设备

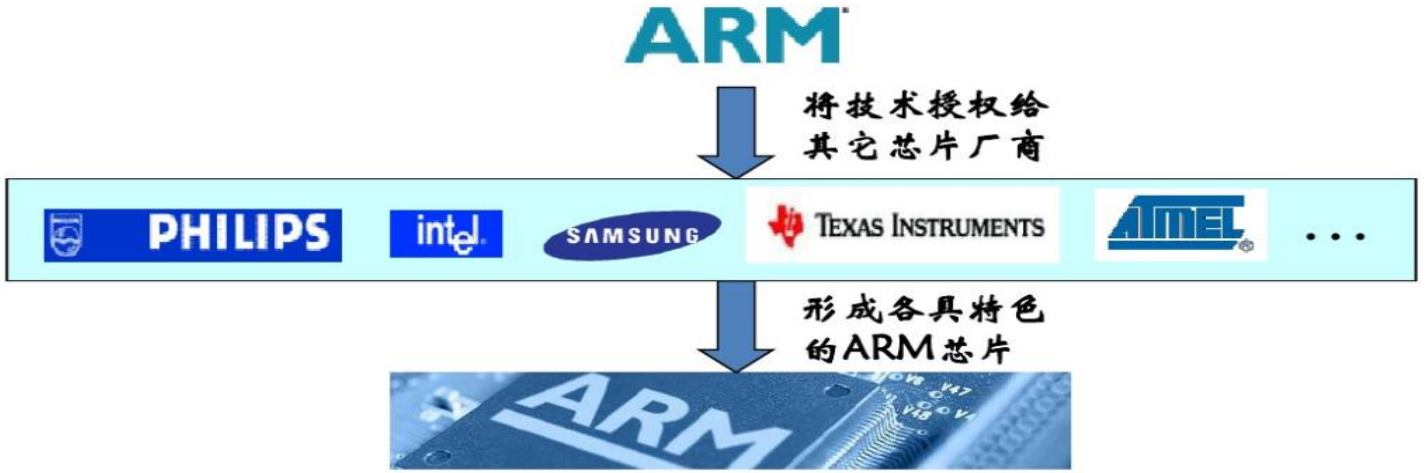
进一步的、不断的细分

ARM的发展（简）

Advanced RISC Machines

……
1993　ARM-----TI、APPLE(newton).....诺基亚Nokia
……

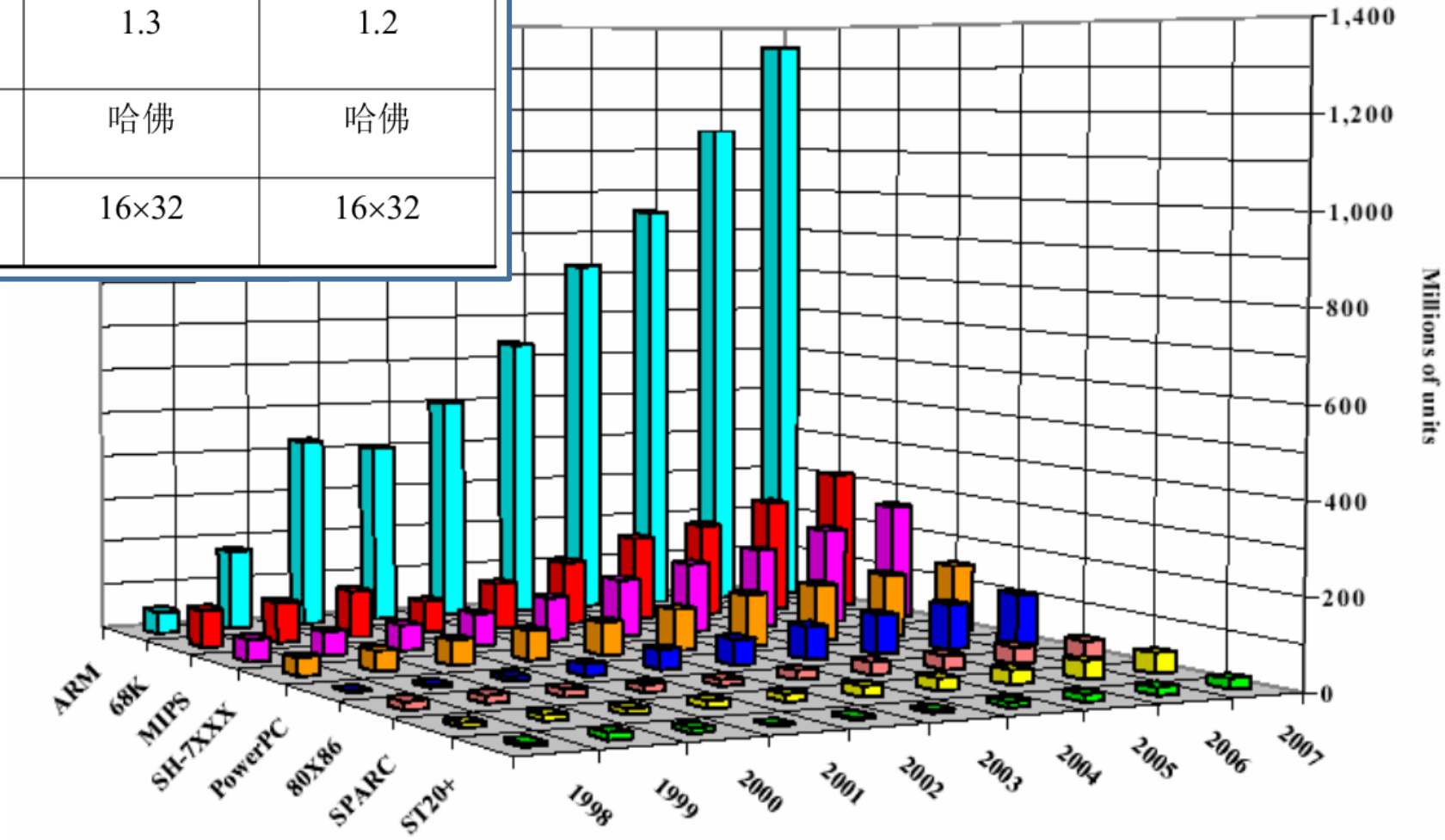
2007　iPhone，ARM设计/三星制造芯片
2008　谷歌推出Android系统，面向ARM指令集
　　摩托罗拉MOTO
　　恩智浦NXP，意法半导体ST
　　高通公司Qualcomm、苹果APPLE



ARM

X86

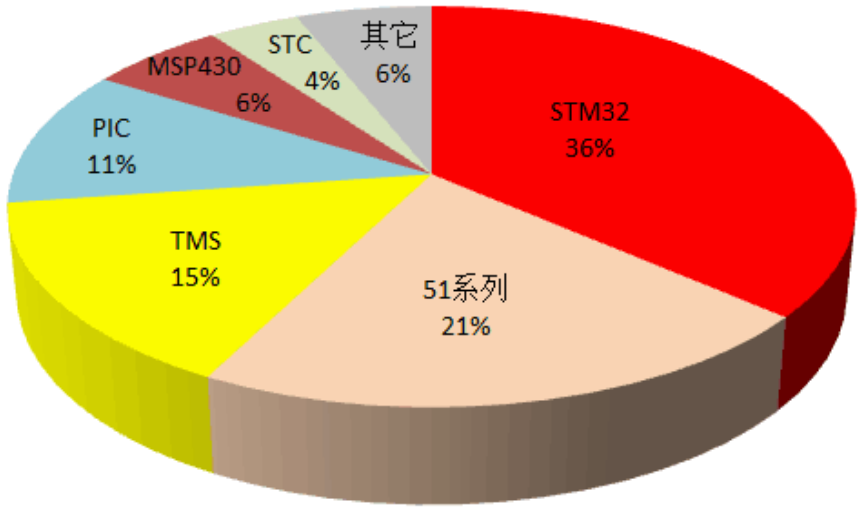
项目	ARM7	ARM9	ARM10	ARM11
流水线	3	5	6	8
典型频率（MHz）	80	150	260	335
功耗（mW/MHz）	0.06	0.19（+cache）	0.5（+cache）	0.4（+cache）
性能 MIPS**/MHz	0.97	1.1	1.3	1.2
架构	冯·诺伊曼	哈佛	哈佛	哈佛
乘法器	8×32	8×32	16×32	16×32



MCU主要厂商

Leading MCU Suppliers ($M)

2016 Rank	Company	2015	2016	% Change	% Marketshare
1	NXP*	1,350	2,914	116%	19%
2	Renesas	2,560	2,458	-4%	16%
3	Microchip**	1,355	2,027	50%	14%
4	Samsung	2,170	1,866	-14%	12%
5	ST	1,514	1,573	4%	10%
6	Infineon	1,060	1,106	4%	7%
7	Texas Instruments	820	835	2%	6%
8	Cypress***	540	622	15%	4%



恩智浦半导体

NXP

- LPC1000系列： Cortex-M0~M3
- LPC2000系列： ARM7
- LPC3000系列： ARM9

流明诺瑞

TEXAS INSTRUMENTS LUMINARY

- AM35X,AM37X系列： Cortex-A8
- AM17X,AM18X 系列： ARM9
- LM3S1000~9000： Cortex-M3

意法半导体

STMicroelectronics

- STM32F系列：
- STM32L系列：
- STM32W 系列：

Cortex-M0~M7

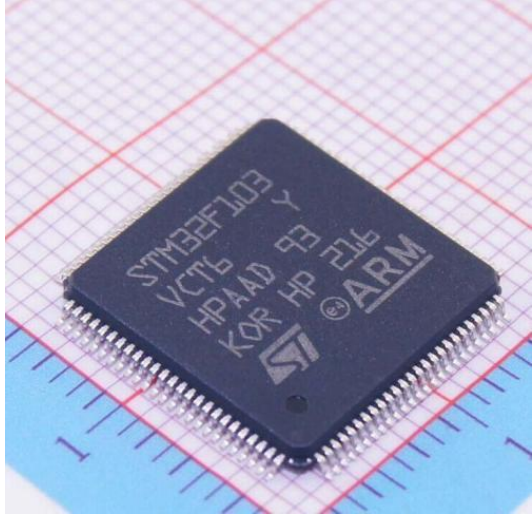
三星

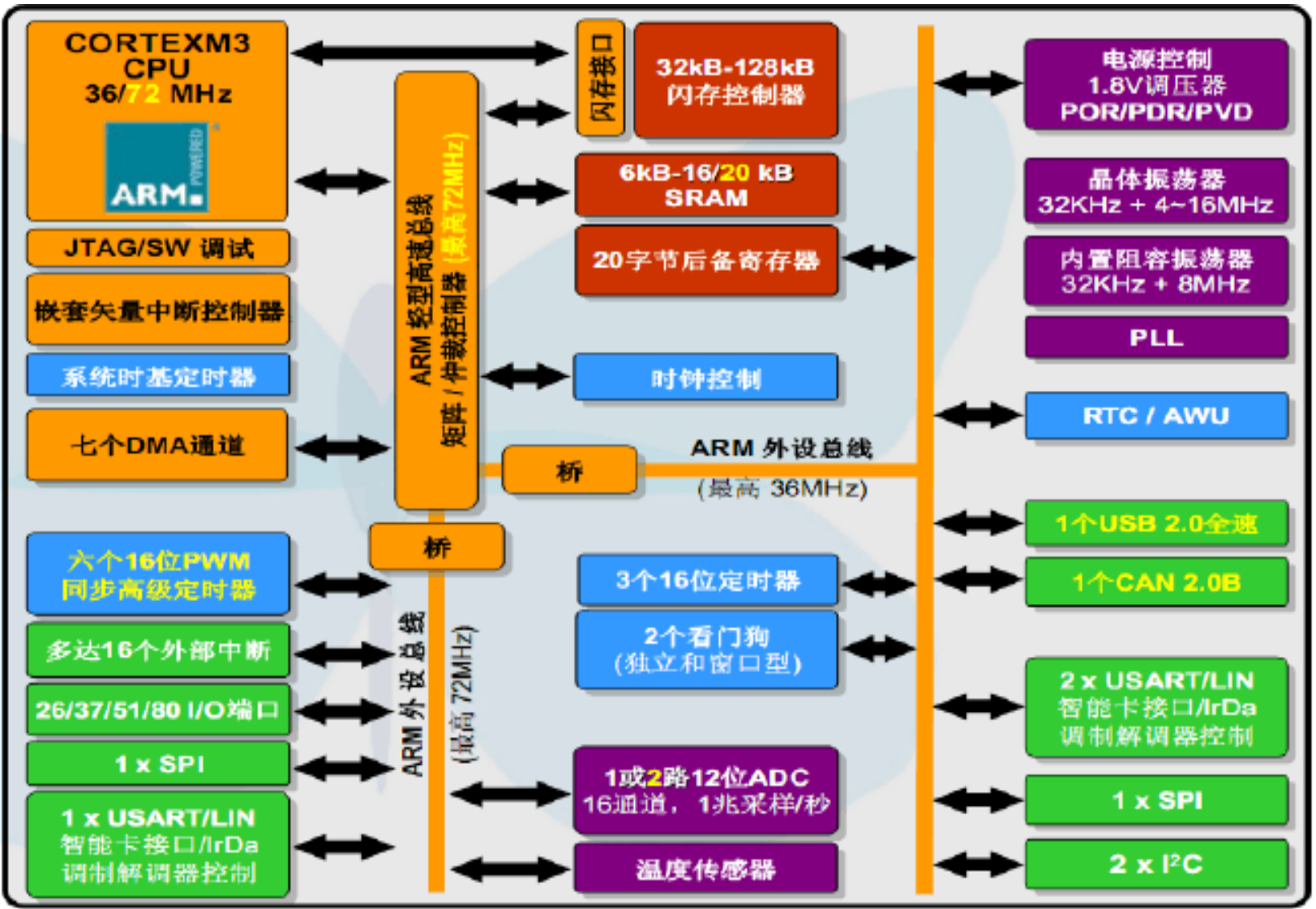
SAMSUNG

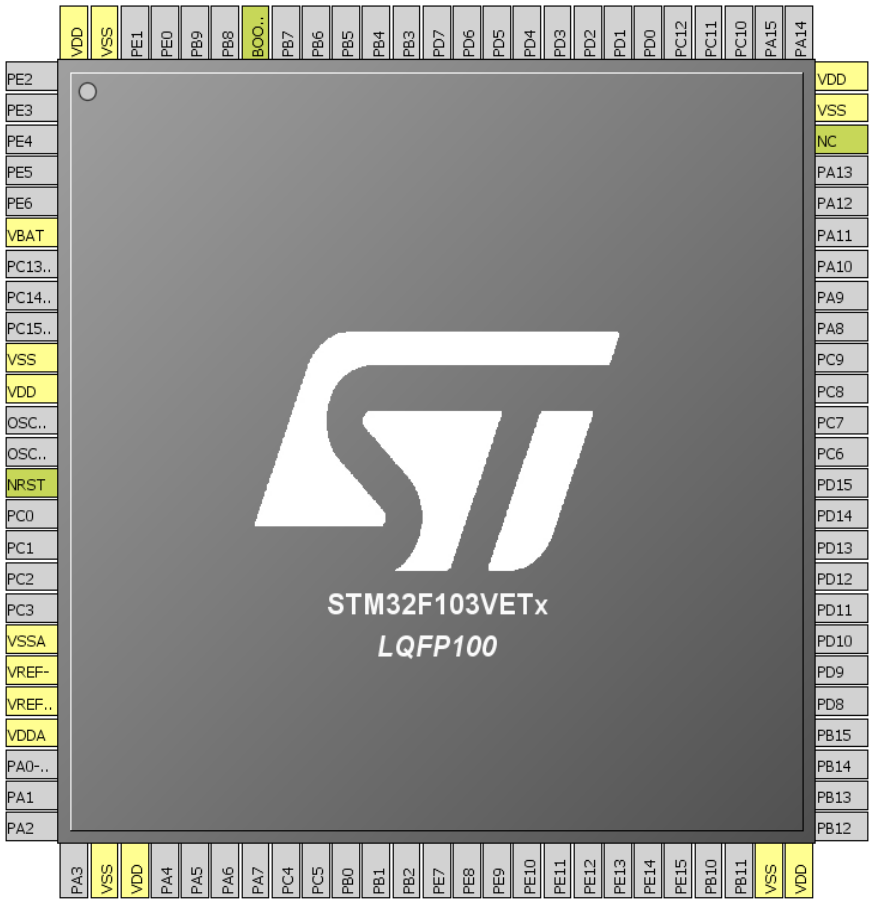
- S3C2410,2440： ARM9
- S3C6410： ARM11

M3 MCU

小而更完备的系统，而非一个功能元件（CPU仅是系统内一个组成部分）





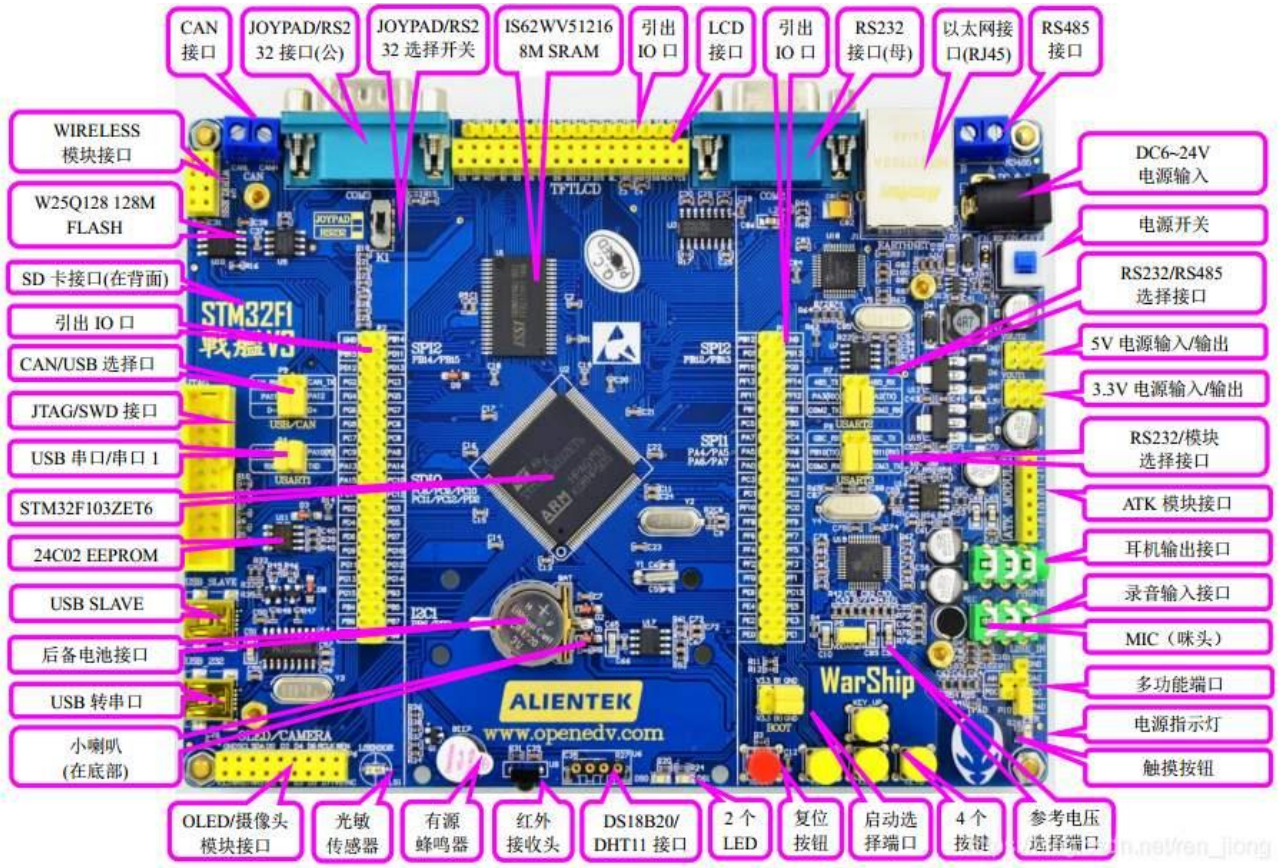


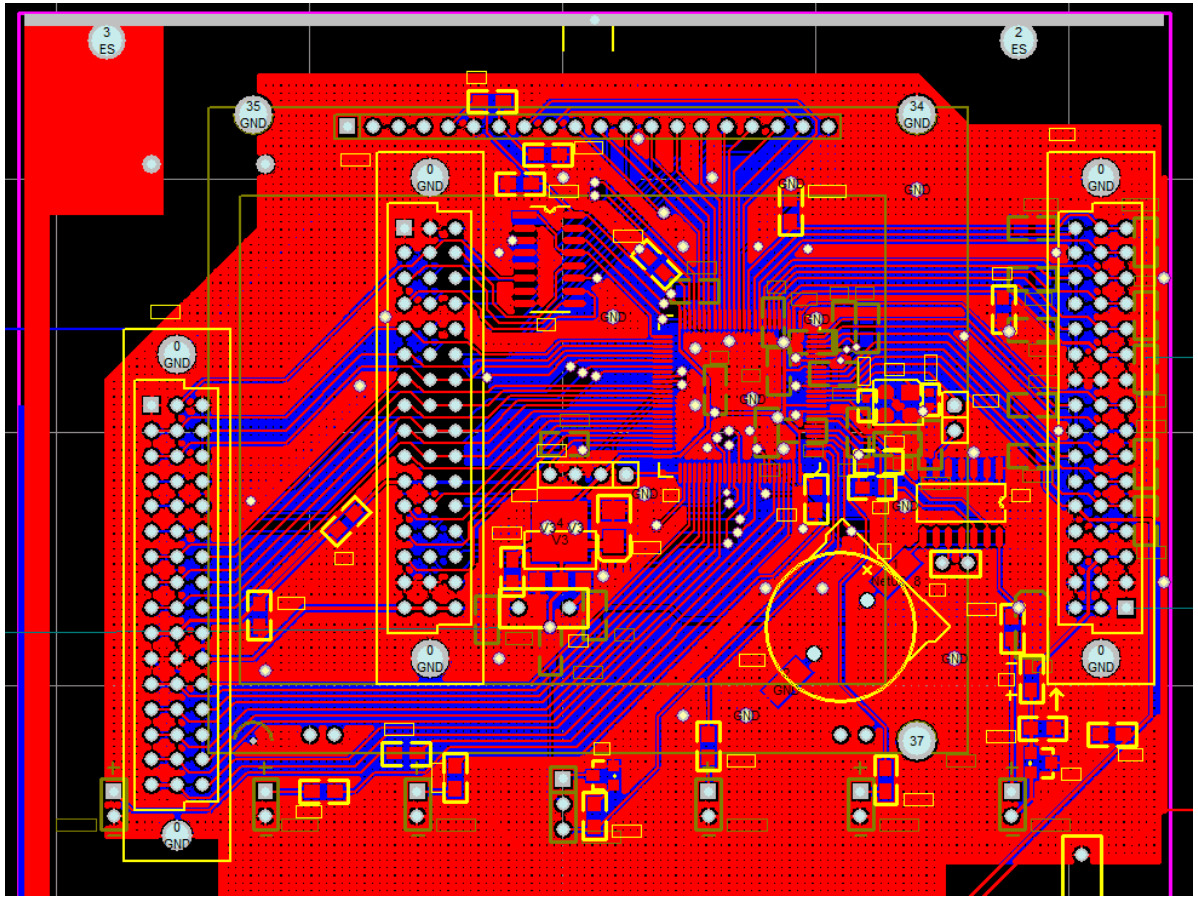
实例参考

基于某Mx 内核的MCU

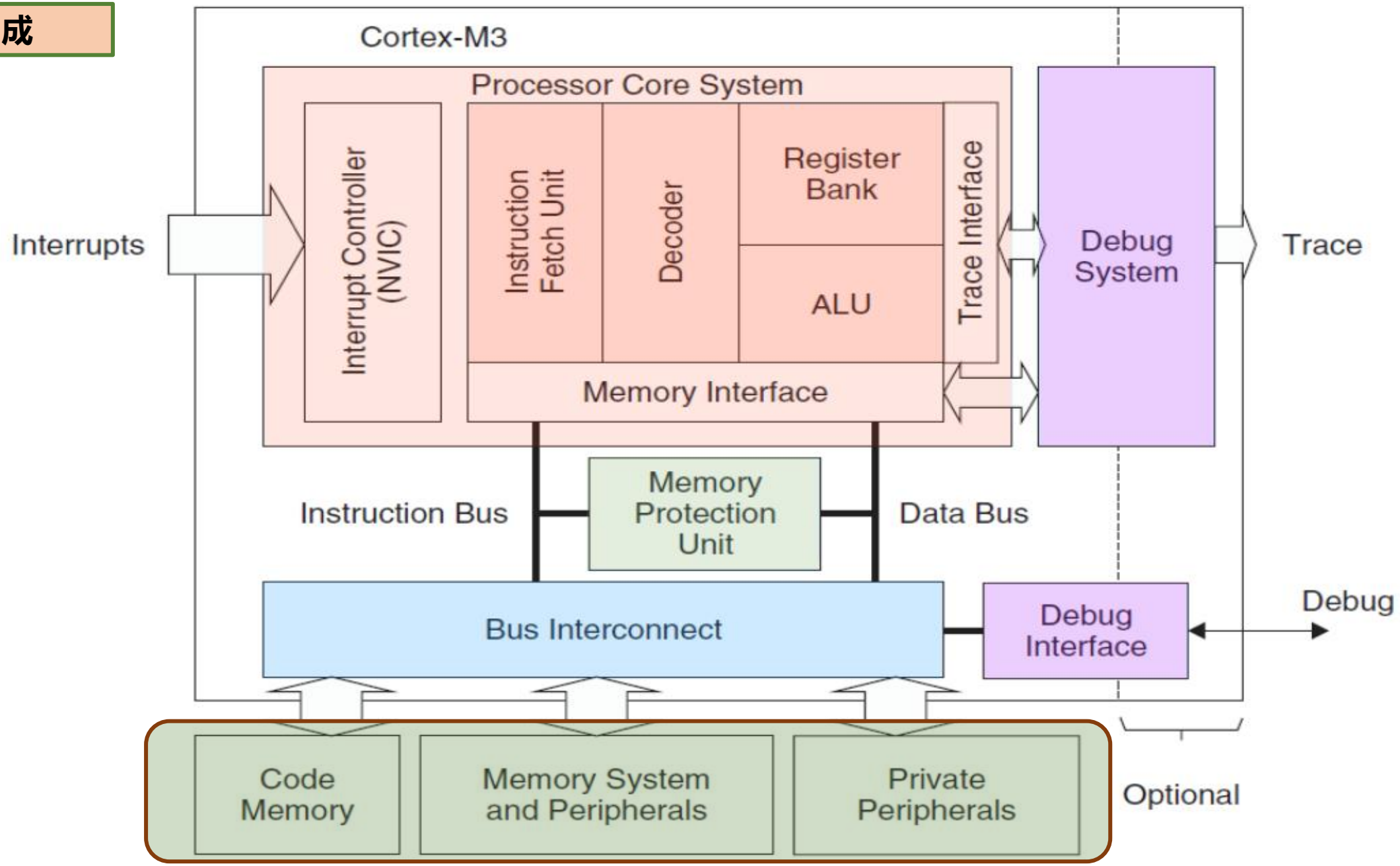
扩展少量外围元件

可构成一定变化、多种类型的、满足现实应用的微处理器系统

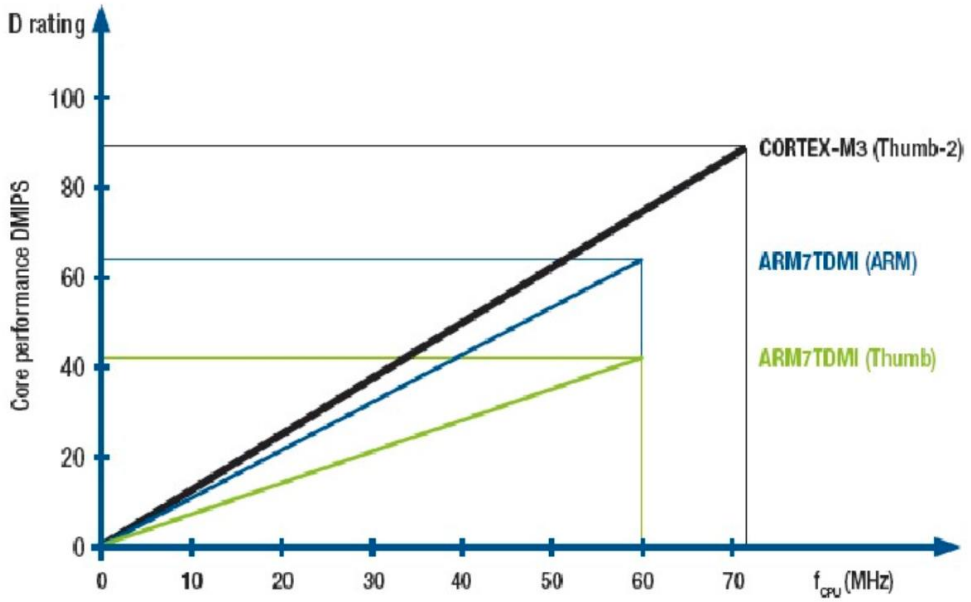




M3 系统构成


Cortex-M3
Processor Core System
Interrupts
Interrupt Controller (NVIC)
Instruction Fetch Unit
Decoder
Register Bank
ALU
Trace Interface
Memory Interface
Debug System
Trace
Instruction Bus
Memory Protection Unit
Data Bus
Bus Interconnect
Debug Interface
Debug
Code Memory
Memory System and Peripherals
Private Peripherals
Optional

M3 的进步



Cortex-M3和ARM7的比较

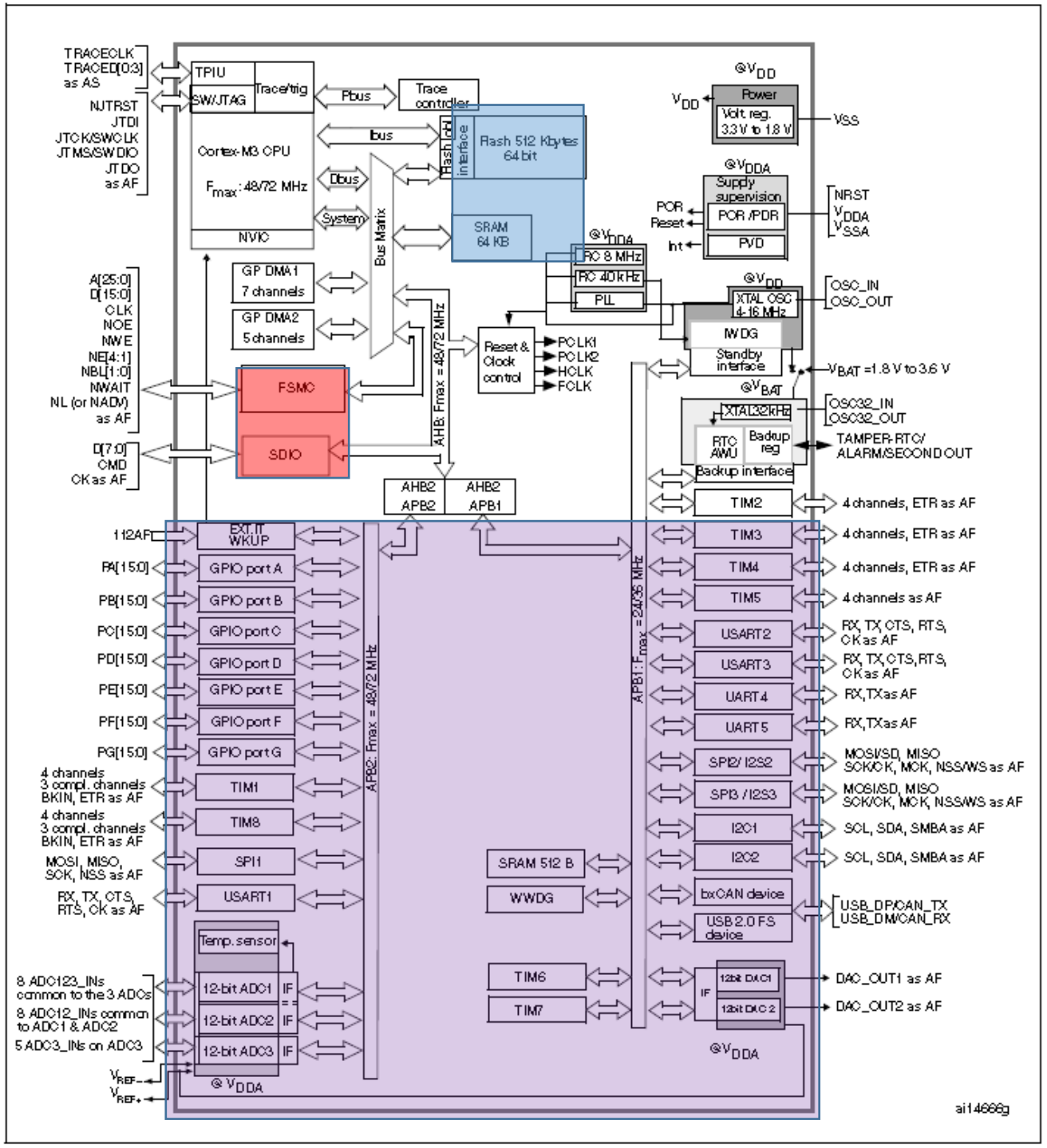
比较项目	ARM7	Cortex-M3
架构	ARMv4T（冯诺依曼） 指令和数据总线共用，会出现瓶颈	ARMv7-M（哈佛） 指令和数据总线分开，无瓶颈
指令集	32位ARM指令+16位Thumb指令 两套指令之间需要进行状态切换	Thumb/Thumb-2指令集 16位和32位 指令可直接混写，无需状态切换
流水线	3级流水线 若出现转移则需要刷新流水线，损失惨重	3级流水线+分支预测 出现转移时流水线无需刷新，几乎无损失
性能	0.95DMIPS/MHz（ARM模式）	1.25DMIPS/MHz
功耗	0.28mW/MHz	0.19mW/MHz
低功耗模式	无	内置睡眠模式
面积	0.62mm2（仅内核）	0.86mm2（内核+外设）
中断	普通中断IRQ和快速中断FIQ太少，大量外设不得不复用中断	不可屏蔽中断NMI+1-240个物理中断 每个外设都可以独占一个中断，效率高
中断延迟	24-42个时钟周期，缓慢	12个时钟周期，最快只需6个
中断压栈	软件手工压栈，代码长且效率低	硬件自动压栈，无需代码且效率高
存储器保护	无	8段存储器保护单元（MPU）
内核寄存器	寄存器分为多组、结构复杂、占核面积多	寄存器不分组（SP除外），结构简单
工作模式	7种工作模式，比较复杂	只有线程模式和处理模式两种，简单
乘除法指令	多周期乘法指令，无除法指令	单周期乘法指令，2-12周期除法指令
位操作	无 访问外设寄存器需分“读-改-写”3步走	先进的Bit-band位操作技术，可直接访问外设寄存器的某个值
系统节拍定时	无	内置系统节拍定时器，有利于操作系统移植

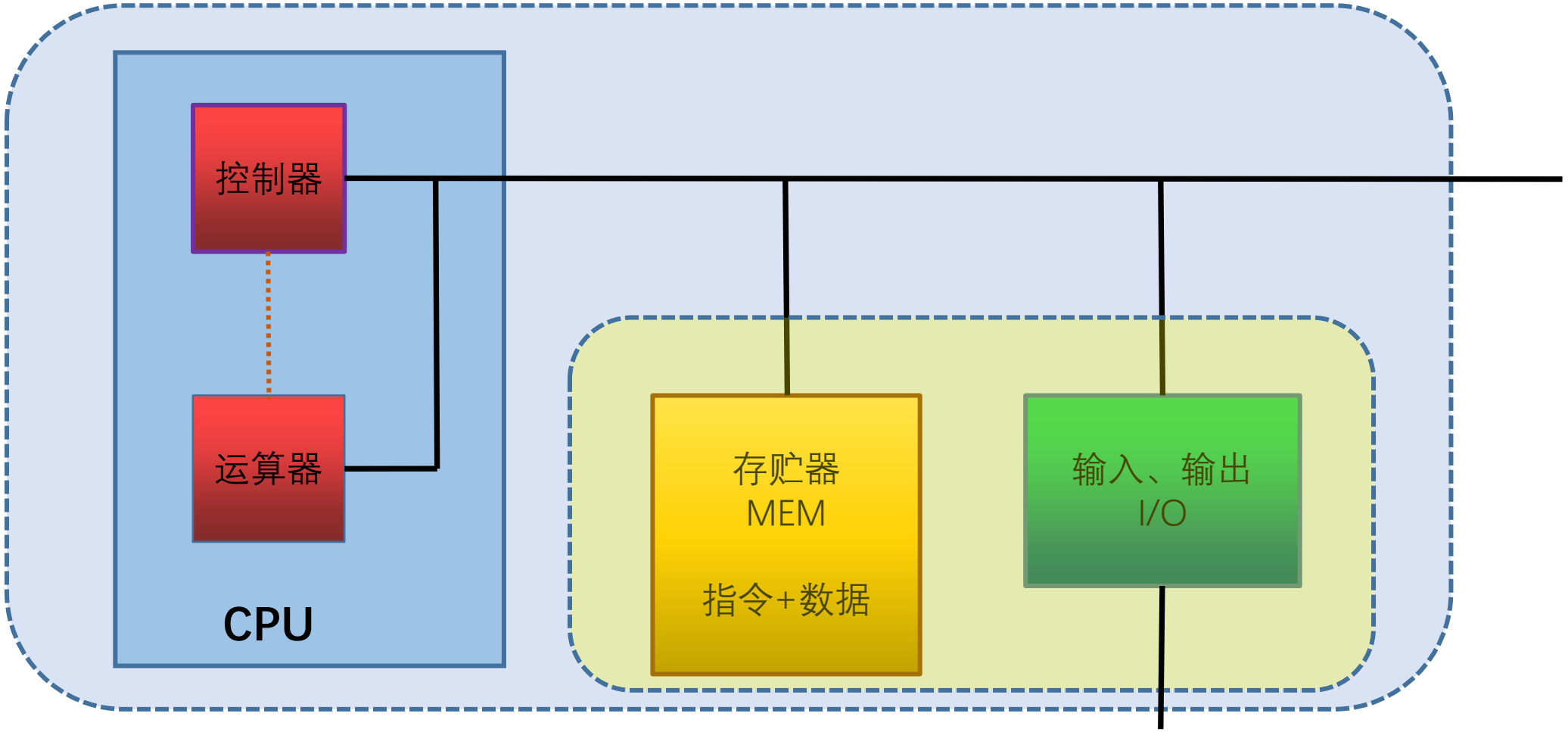
M3 MCU资源（示例）

配备了一定数量存贮MEM资源（RAM，Flash）

配备了常用、基本的IO资源（GPIO、串行类接口、定时器、模入模出AD/DA等

Peripherals		STM32F103Rx			STM32F103Vx			STM32F103Zx		
Flash memory in Kbytes		256	384	512	256	384	512	256	384	512
SRAM in Kbytes		48	64(1)		48	64		48	64	
FSMC		No			Yes(2)			Yes		
Timers	General-purpose	4								
	Advanced-control	2								
	Basic	2								
Comm	SPI(I²S)(3)	3(2)								
	I²C	2								
	USART	5								
	USB	1								
	CAN	1								
	SDIO	1								
GPIOs		51			80			112		
12-bit ADC Number of channels		3 16			3 16			3 21		
12-bit DAC Number of channels		2 2								
CPU frequency		72 MHz								

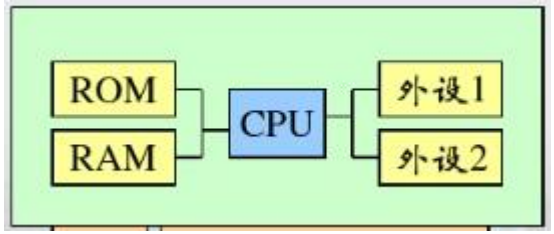


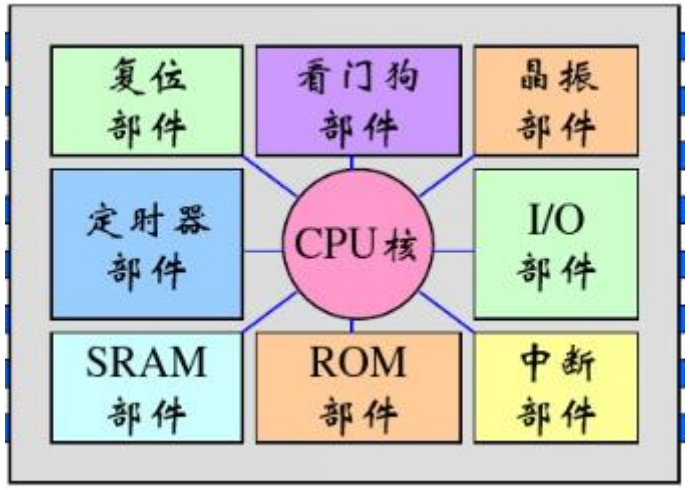


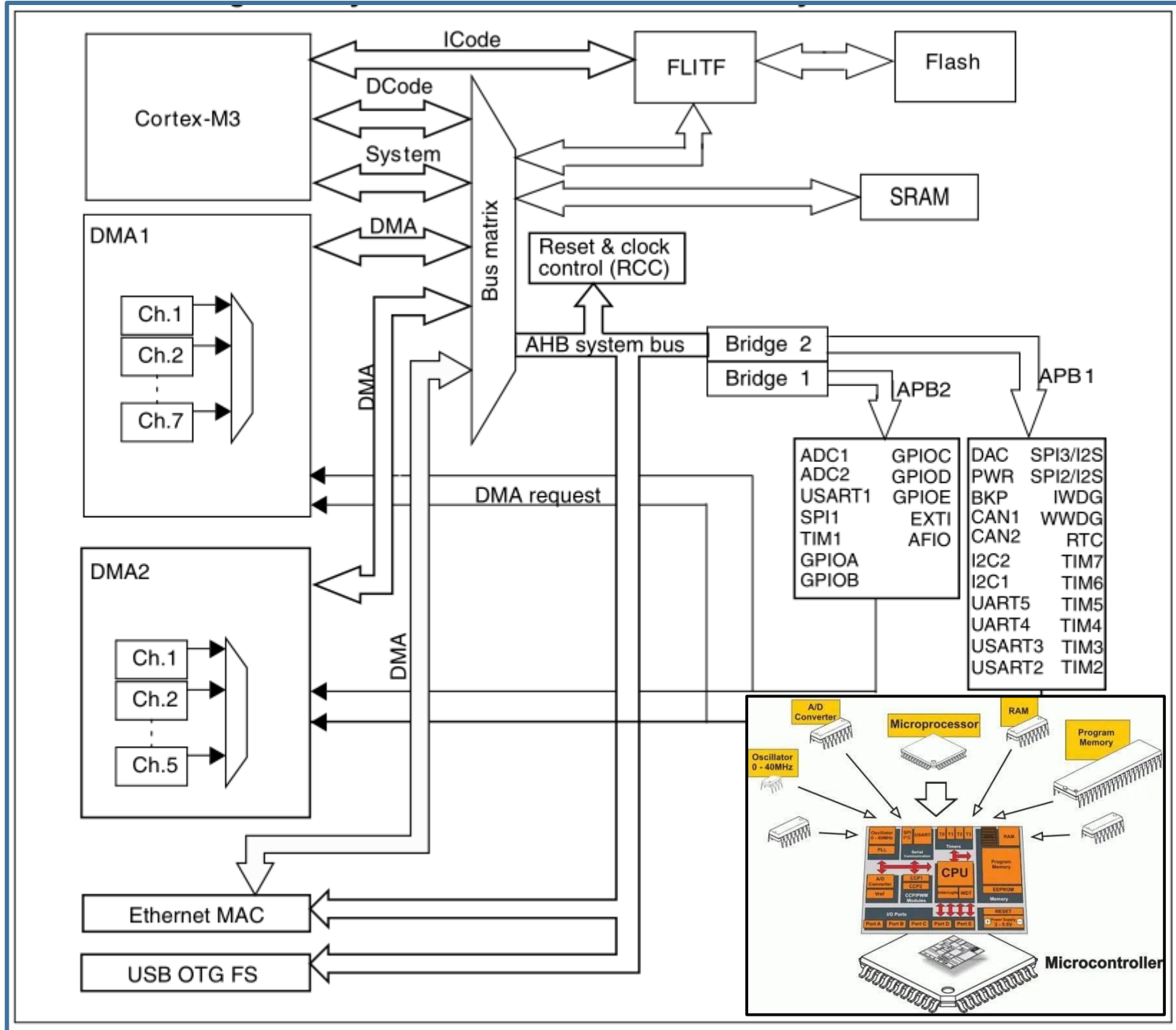
冯诺依曼体系——五大功能部分，存贮器 与 输入、输出是重要功能部分

传统基于CPU搭建的小规模微处理器系统，需要外围存贮器、IO接口等

MCU可在尺寸空间、可靠性、功耗、开发周期、综合成本等方面有优势



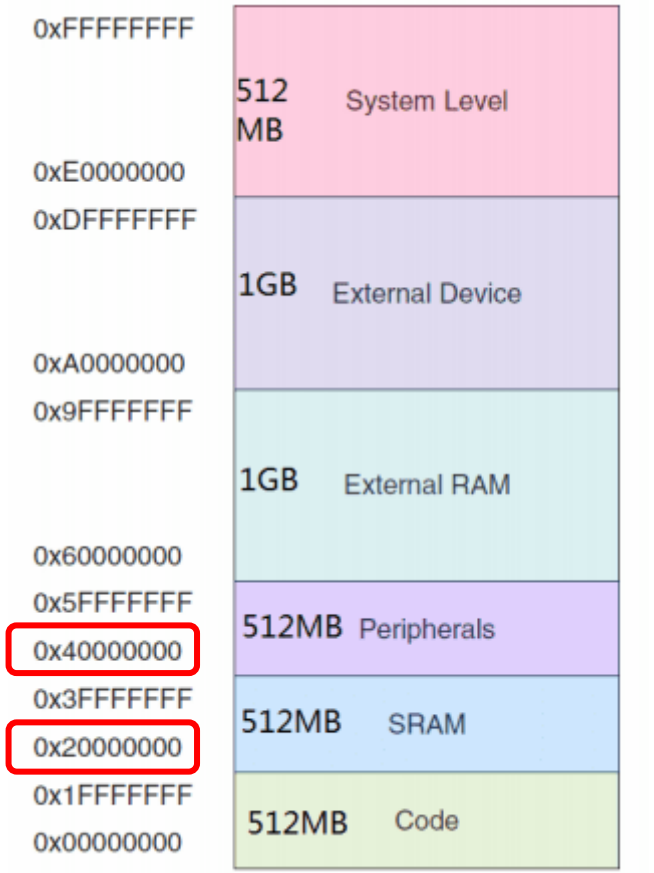


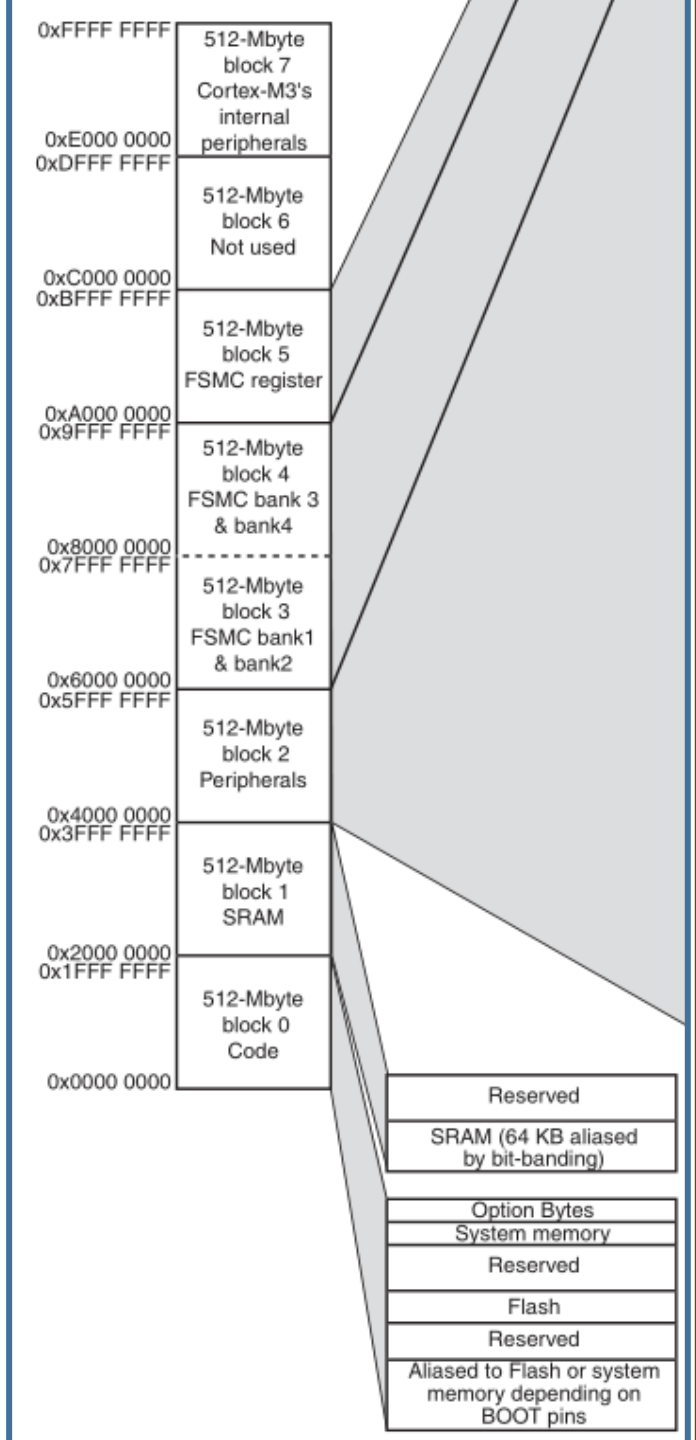


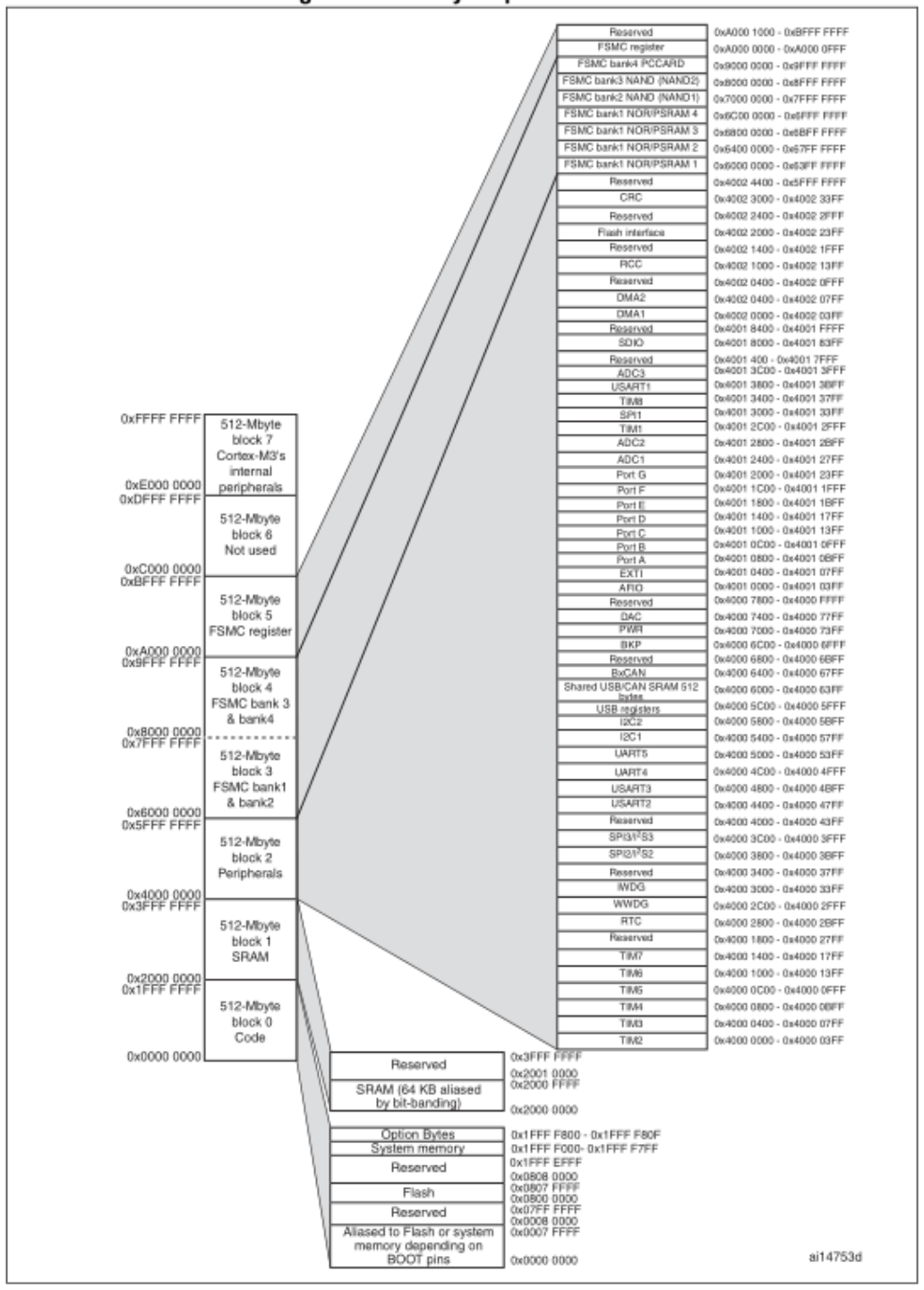
M3 存贮资源

目前可具备
数百 kB 级 Flash
数十 kB 级 RAM

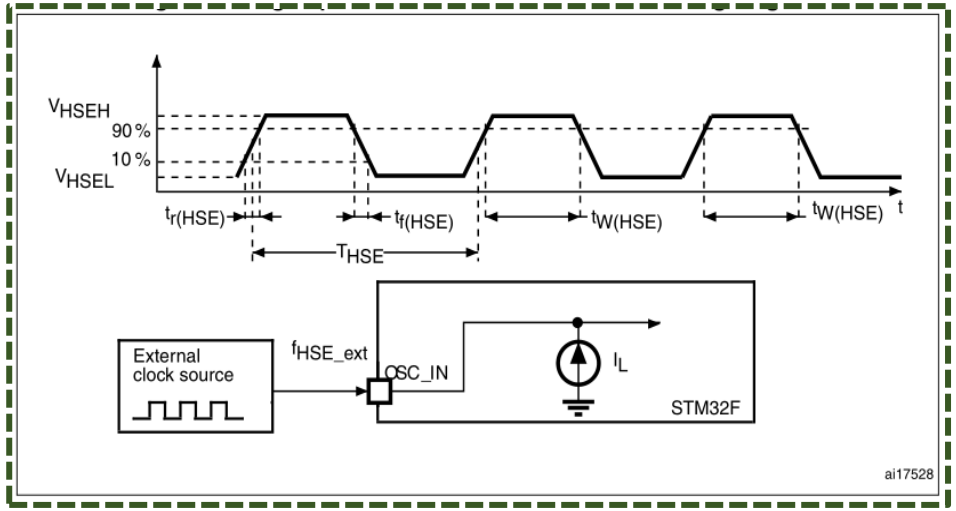
ARM M系列体系可支持访问、寻址GB级空间

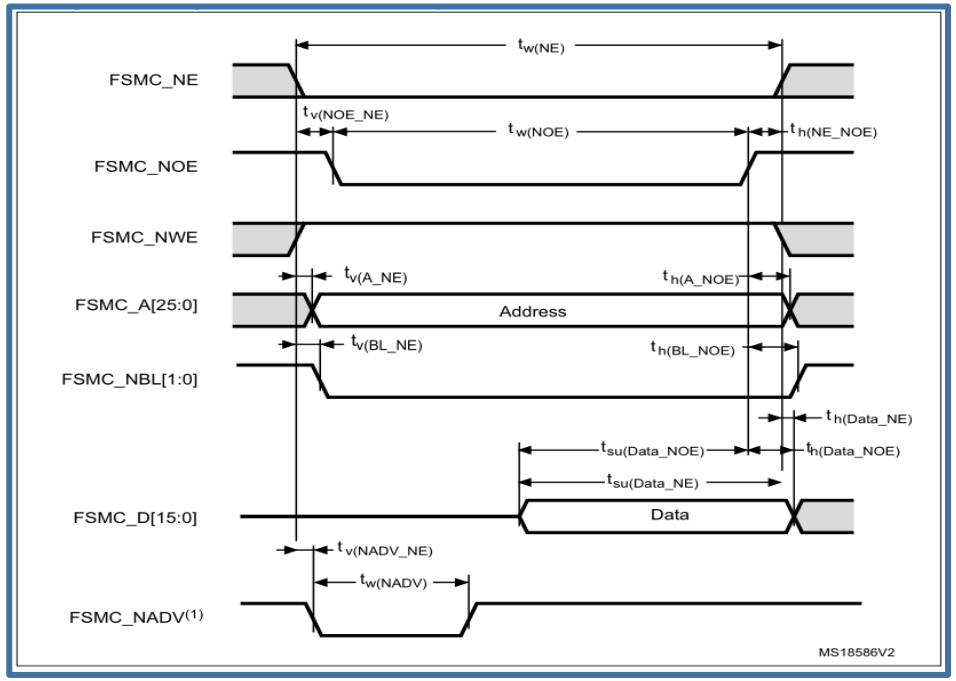


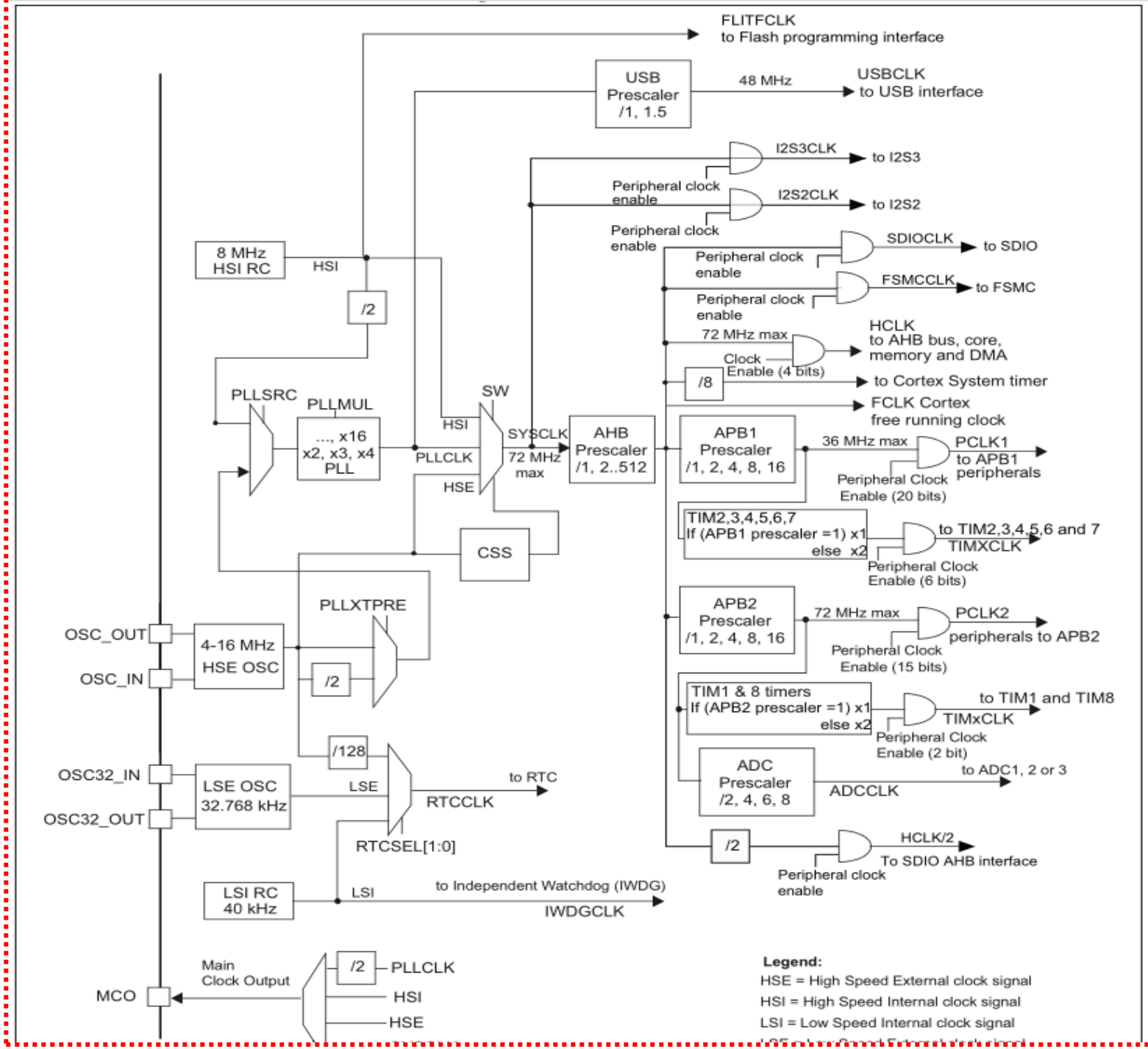




时钟相关

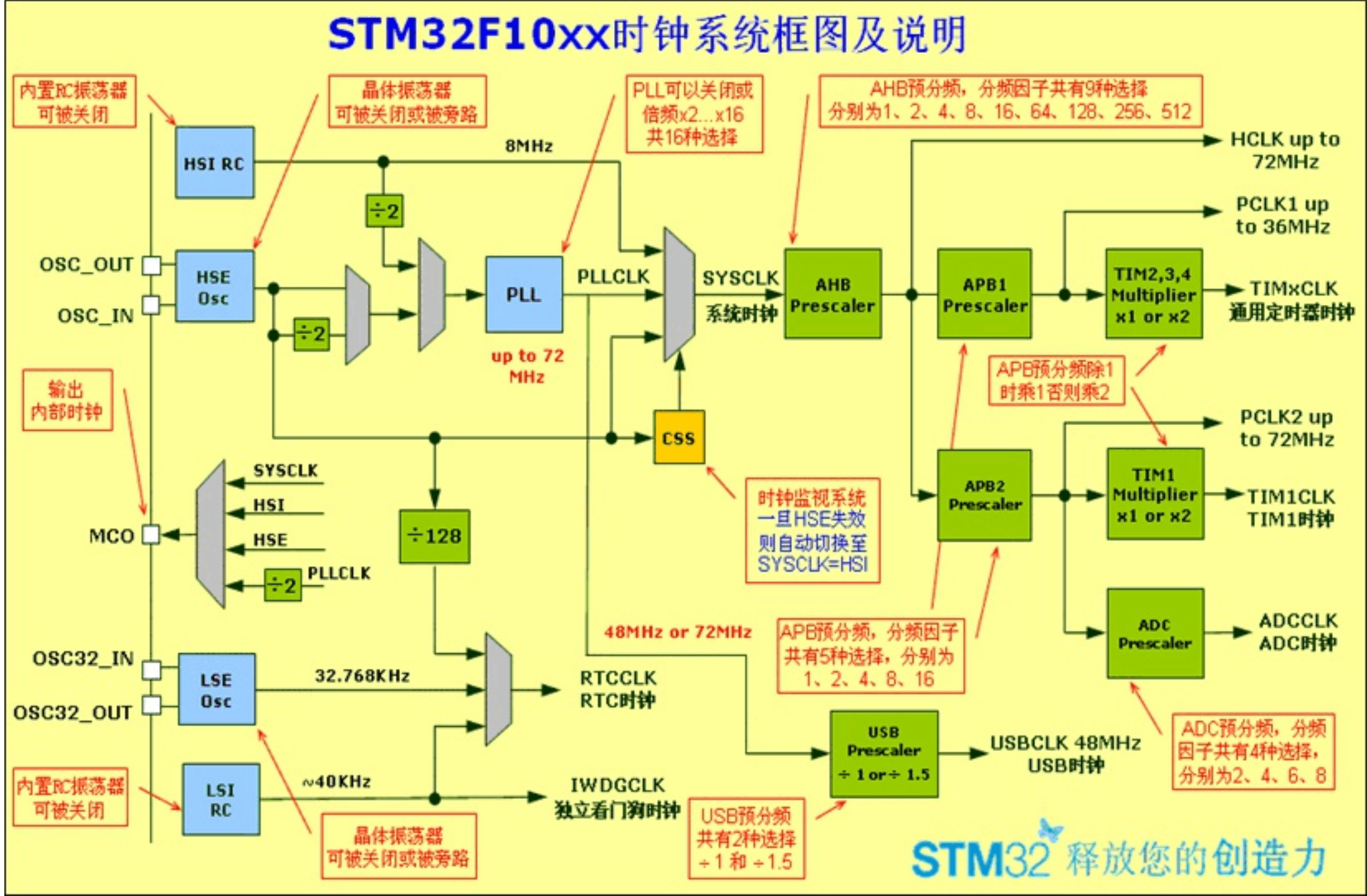

VHSEH
90 %
10 %
VHSEL
tr(HSE)
tf(HSE)
tW(HSE)
tW(HSE)
t
THSE
External
clock source
fHSE_ext
OSC_IN
IL
STM32F
ai17528


tw(NE)
FSMC_NE
tv(NOE_NE)
tw(NOE)
th(NE_NOE)
FSMC_NOE
FSMC_NWE
tv(A_NE)
th(A_NOE)
FSMC_A[25:0]
Address
tv(BL_NE)
th(BL_NOE)
FSMC_NBL[1:0]
th(Data_NE)
tsu(Data_NOE)
th(Data_NOE)
tsu(Data_NE)
FSMC_D[15:0]
Data
tv(NADV_NE)
tw(NADV)
FSMC_NADV(1)
MS18586V2

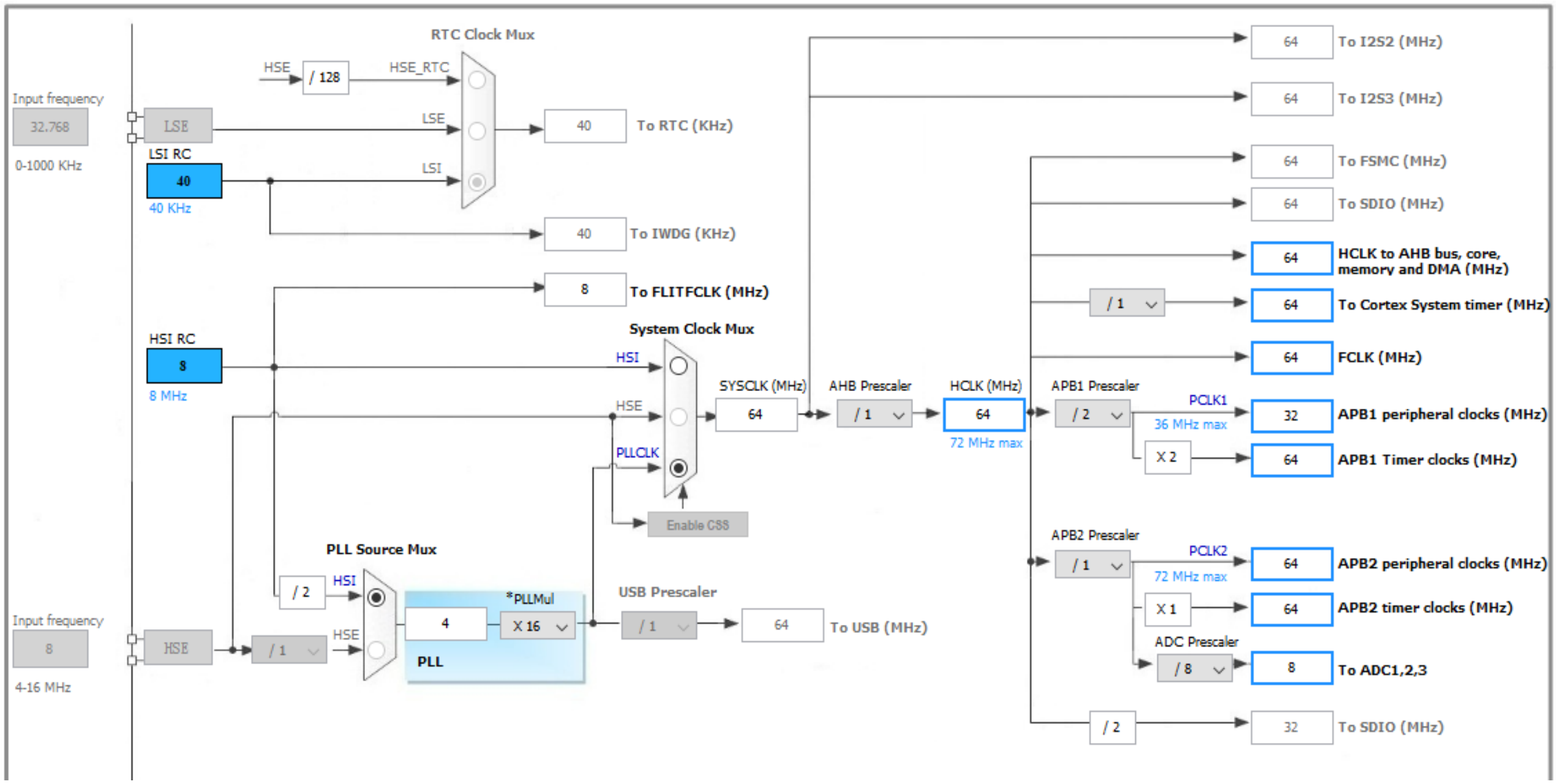

FLITFCLK
to Flash programming interface
USB
Prescaler
/1, 1.5
48 MHz
USBCLK
to USB interface
I2S3CLK
to I2S3
Peripheral clock
enable
I2S2CLK
to I2S2
Peripheral clock
enable
SDIOCLK
to SDIO
Peripheral clock
enable
FSMCCLK
to FSMC
Peripheral clock
enable
8 MHz
HSI RC
HSI
/2
72 MHz max
HCLK
to AHB bus, core,
memory and DMA
Clock
Enable (4 bits)
/8
to Cortex System timer
FCLK Cortex
free running clock
PLLSRC
PLLMUL
..., x16
x2, x3, x4
PLL
HSI
PLLCLK
HSE
SW
SYSCLK
72 MHz
max
AHB
Prescaler
/1, 2..512
APB1
Prescaler
/1, 2, 4, 8, 16
36 MHz max
PCLK1
to APB1
peripherals
Peripheral Clock
Enable (20 bits)
TIM2,3,4,5,6,7
If (APB1 prescaler =1) x1
else x2
to TIM2,3,4,5,6 and 7
TIMXCLK
Peripheral Clock
Enable (6 bits)
CSS
PLLXTPRE
APB2
Prescaler
/1, 2, 4, 8, 16
72 MHz max
PCLK2
peripherals to APB2
Peripheral Clock
Enable (15 bits)
OSC_OUT
OSC_IN
4-16 MHz
HSE OSC
/2
TIM1 & 8 timers
If (APB2 prescaler =1) x1
else x2
to TIM1 and TIM8
TIMxCLK
Peripheral Clock
Enable (2 bit)
/128
OSC32_IN
OSC32_OUT
LSE OSC
32.768 kHz
LSE
to RTC
RTCCLK
RTCSEL[1:0]
ADC
Prescaler
/2, 4, 6, 8
to ADC1, 2 or 3
ADCCLK
/2
HCLK/2
To SDIO AHB interface
Peripheral clock
enable
LSI RC
40 kHz
LSI
to Independent Watchdog (IWDG)
IWDGCLK
Main
Clock Output
MCO
/2
PLLCLK
HSI
HSE
Legend:
HSE = High Speed External clock signal
HSI = High Speed Internal clock signal
LSI = Low Speed Internal clock signal

时钟相关

（参考）


STM32F10xx时钟系统框图及说明
内置RC振荡器 可被关闭
晶体振荡器 可被关闭或被旁路
PLL可以关闭或 倍频x2...x16 共16种选择
AHB预分频，分频因子共有9种选择 分别为1、2、4、8、16、64、128、256、512
HSI RC
8MHz
÷2
OSC_OUT
OSC_IN
HSE Osc
÷2
PLL
PLLCLK
SYSCLK
系统时钟
AHB Prescaler
HCLK up to 72MHz
APB1 Prescaler
PCLK1 up to 36MHz
TIM2,3,4 Multiplier x1 or x2
TIMxCLK 通用定时器时钟
up to 72 MHz
APB预分频除1 时乘1否则乘2
输出 内部时钟
CSS
PCLK2 up to 72MHz
SYSCLK
HSI
HSE
PLLCLK
MCO
÷128
÷2
时钟监视系统 一旦HSE失效 则自动切换至 SYSCLK=HSI
APB2 Prescaler
TIM1 Multiplier x1 or x2
TIM1CLK TIM1时钟
48MHz or 72MHz
APB预分频，分频因子 共有5种选择，分别为 1、2、4、8、16
ADC Prescaler
ADCCLK ADC时钟
OSC32_IN
OSC32_OUT
LSE Osc
32.768KHz
RTCCLK RTC时钟
USB Prescaler ÷1 or ÷1.5
USBCLK 48MHz USB时钟
ADC预分频，分频 因子共有4种选择， 分别为2、4、6、8
内置RC振荡器 可被关闭
LSI RC
≈40KHz
IWDGCLK 独立看门狗时钟
晶体振荡器 可被关闭或被旁路
USB预分频 共有2种选择 ÷1 和 ÷1.5
STM32 释放您的创造力

作为包含多个功能部件及不同类型IO接口的系统，需要提供多样选择的时钟发生与倍频、分频。以满足各种应用需要（参考）



总线相关

总线（bus）是系统内各功能部件之间访问、控制的信号线总称

往往有共享性、多根、多组特征。

如数据总线（Data Bus）可连接多个部件。 多根常见8、16、32等

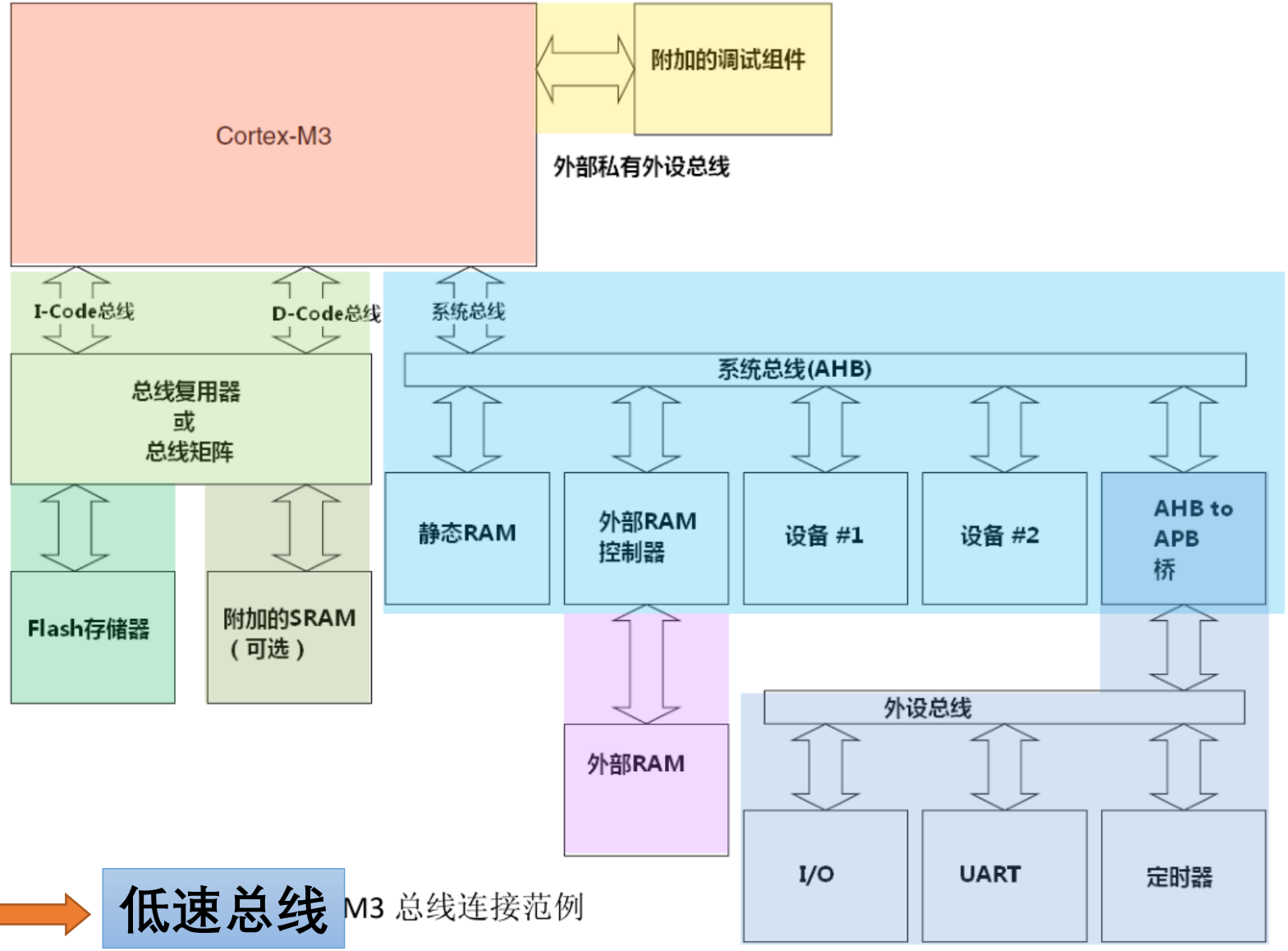
总线的主要参数

- 总线宽度 → 指总线能同时传送数据的位数。如32位总线具有32位数据传送能力。
- 总线频率 → 指总线的工作速度，频率越高，速度越快。
- 总线带宽 → 指一定时间内总线传送的数据总量，用每秒最大传送数据量来衡量。

MCU 包含了较多类型差异、速度多样的部件

可能存在需要采用不同速度的总线访问

也是基于具体类型及综合性能的考虑



M3 总线连接范例

AMBA：Advanced Microcontroller Bus Architecture

AMBA是ARM 公司研发的一种总线规范。

- **AHB(Advanced High-performance Bus)：**

  用于高性能系统模块的连接，支持突发模式数据传输和数据分割传输；可以有效地连接处理器、片上和片外存储器，支持流水线操作。

- **APB（Advanced Peripheral Bus）：**

  用于较低性能外设的简单连接，一般是接在AHB系统总线上的第二级总线。

MCU 系统搭建与扩展可能的选择

如前所示，MCU包含了一定的MEM与IO资源，但实际应用系统往往有各自需求与变化。尽管目前的MCU有多样的系列与型号，但其定位依然是提供基本资源

MCU首先提供了一定数量、不同类型的串行接口（SPI、I2C、SCI、USB、CAN…）

其中SPI可扩展很多外围功能部件

具有SPI、I2C接口的芯片在目前有较大的选择范围

以串行方式搭建、扩展，满足实际应用系统的需要，是主要考虑方向

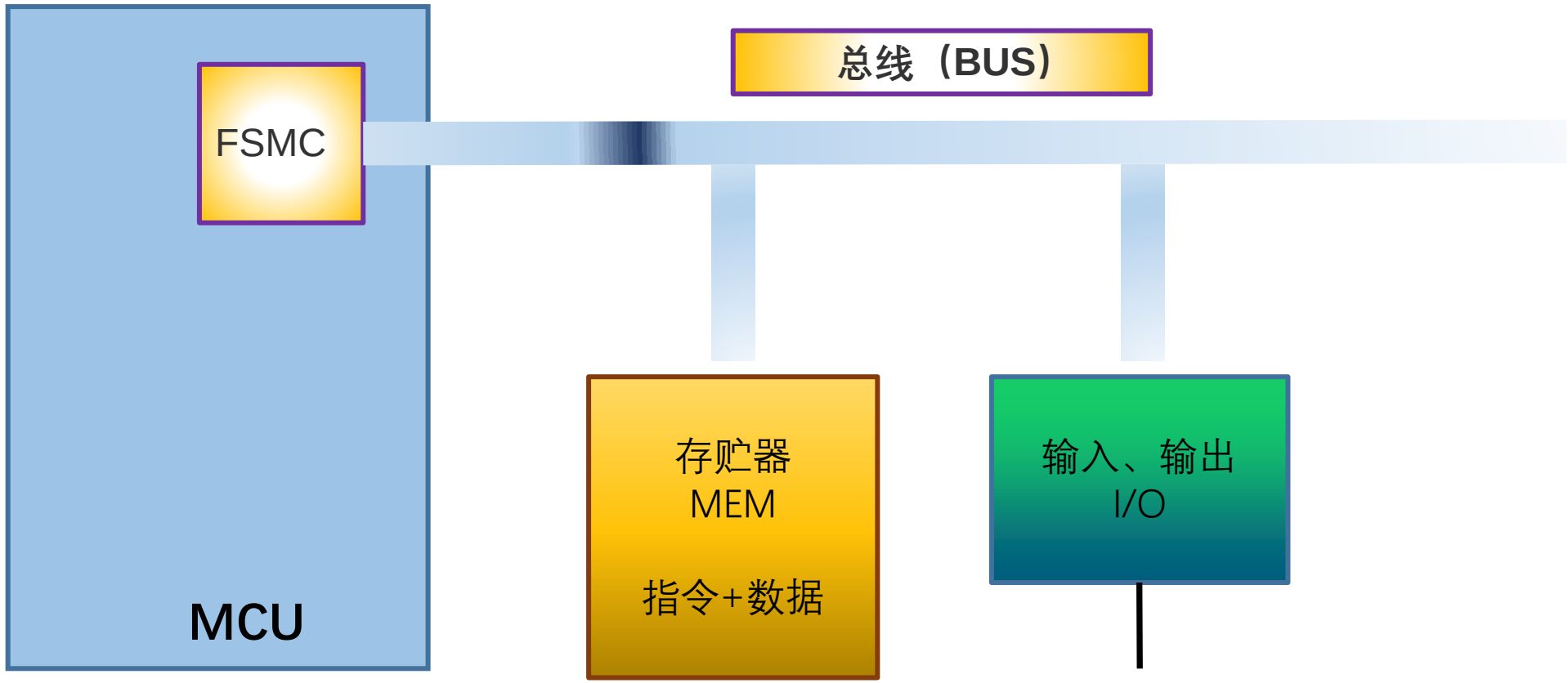
但在一些应用中，与外部设备（功能元件）有大量、快速、频繁的数据传输与访问时

以并行总线方式，实现CPU与外围部件（大容量存贮器、快速ADC、LCD部件）等，也是必要的

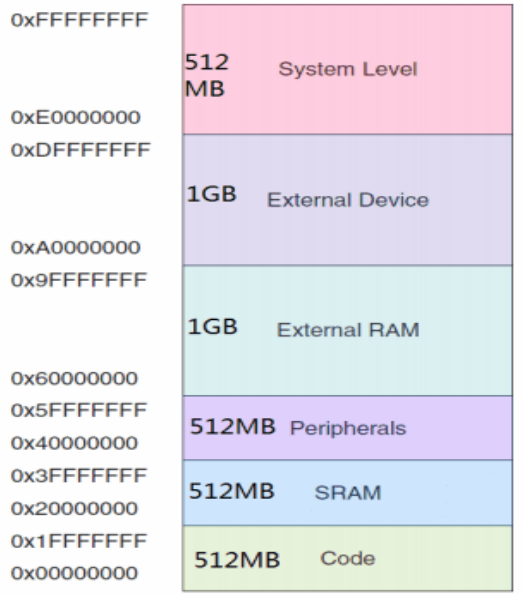
MCU 可否提供外部总线方式，访问、连接、扩展外部部件（MEM、IO）

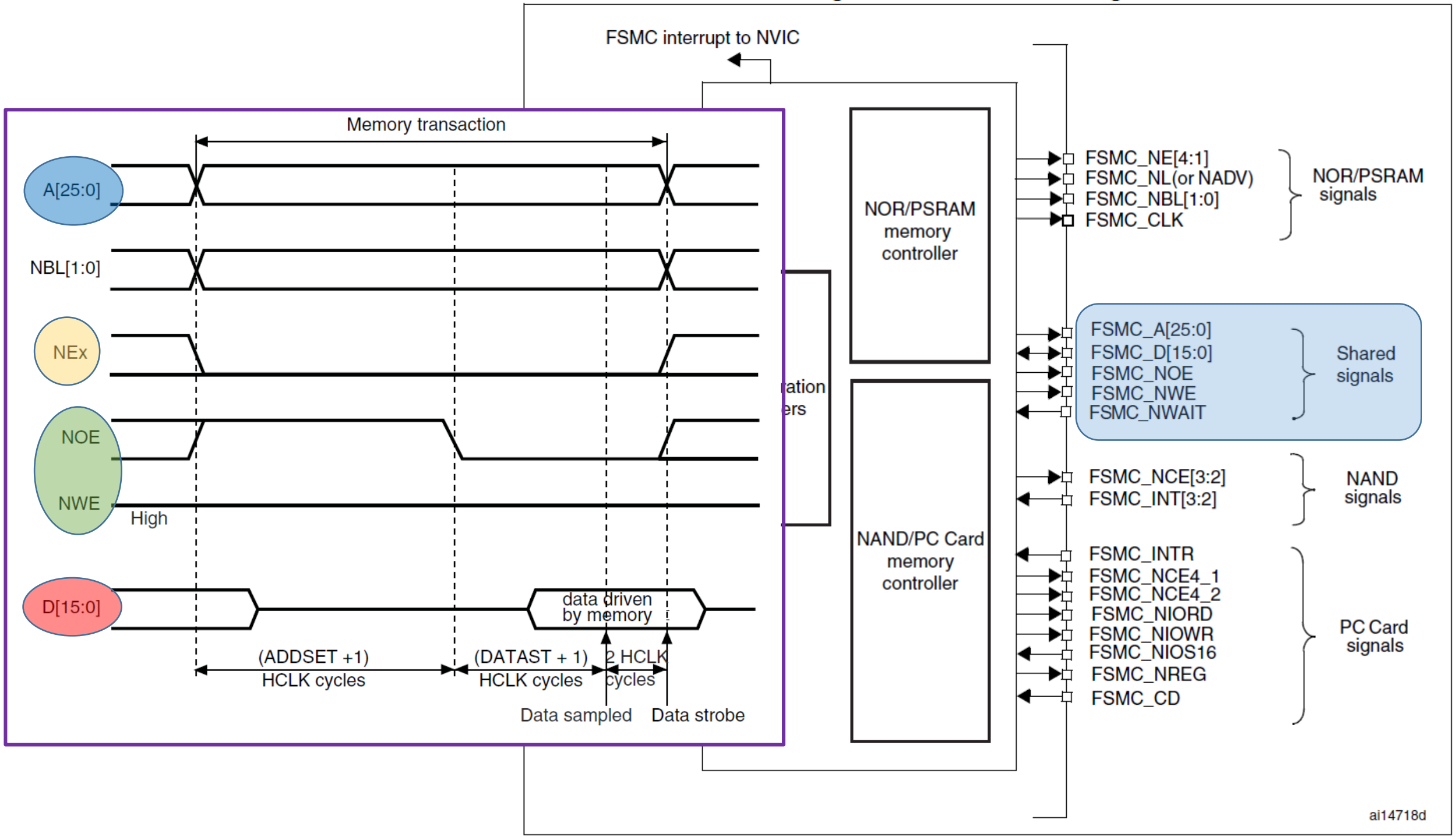
并行外部总线扩展------ FSMC

FSMC(Flexible Static Memory Controller，灵活静态存储控制器)是一种总线扩展技术。用于外部存储器、外部元件、快速、大量数据传输的系统扩展



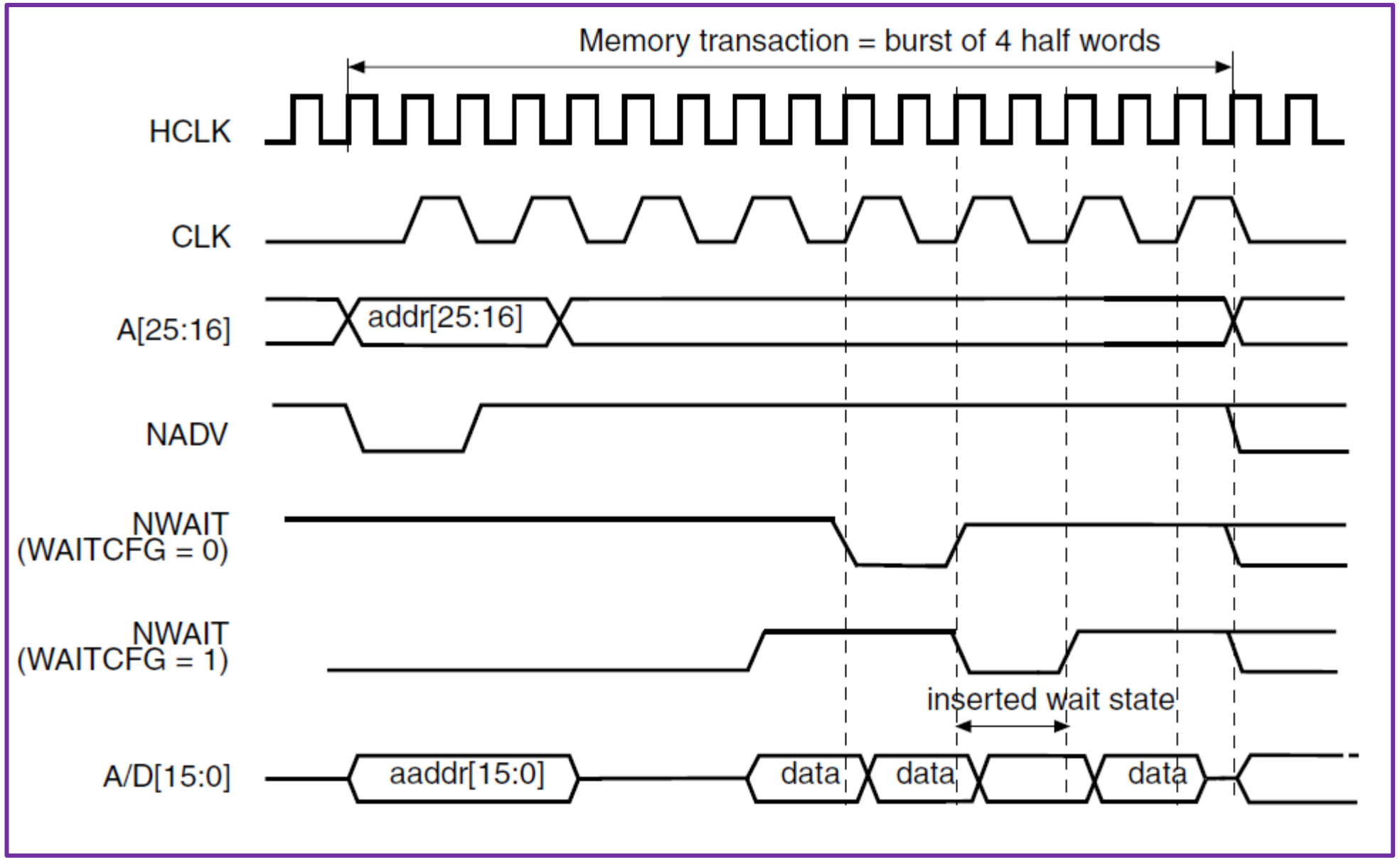
- 支持多种静态存储器类型。通过FSMC可以与SRAM、ROM、PSRAM、NOR Flash和NANDFlash存储器的引脚直接相连。
- 支持丰富的存储操作方法。FSMC不仅支持多种数据宽度的异步读/写操作，而且支持对NOR/PSRAM/NAND存储器的同步突发访问方式。
- 支持同时扩展多种存储器。FSMC的映射地址空间中，不同的BANK是独立的，可用于扩展不同类型的存储器。当系统中扩展和使用多个外部存储器时，FSMC会通过总线悬空延迟时间参数的设置，防止各存储器对总线的访问冲突。
- 支持更为广泛的存储器型号。通过对FSMC的时间参数设置，扩大了系统中可用存储器的速度范围，为用户提供了灵活的存储芯片选择空间。
- 支持代码从FSMC扩展的外部存储器中直接运行，而不需要首先调入内部SRAM。




FSMC interrupt to NVIC
Memory transaction
A[25:0]
NBL[1:0]
NEx
NOE
NWE
High
D[15:0]
data driven by memory
(ADDSET +1) HCLK cycles
(DATAST + 1) HCLK cycles
2 HCLK cycles
Data sampled
Data strobe
NOR/PSRAM memory controller
NAND/PC Card memory controller
FSMC_NE[4:1]
FSMC_NL(or NADV)
FSMC_NBL[1:0]
FSMC_CLK
NOR/PSRAM signals
FSMC_A[25:0]
FSMC_D[15:0]
FSMC_NOE
FSMC_NWE
FSMC_NWAIT
Shared signals
FSMC_NCE[3:2]
FSMC_INT[3:2]
NAND signals
FSMC_INTR
FSMC_NCE4_1
FSMC_NCE4_2
FSMC_NIORD
FSMC_NIOWR
FSMC_NIOS16
FSMC_NREG
FSMC_CD
PC Card signals
ai14718d

灵活、多种的总线类型，如数据、地址线复用，可插入等待周期的总线访问形式

其中一种方式示意



NOR/PSRAM 1 | User Constants | GPIO Settings

Configure the below parameters :

Search (Crtl+F)

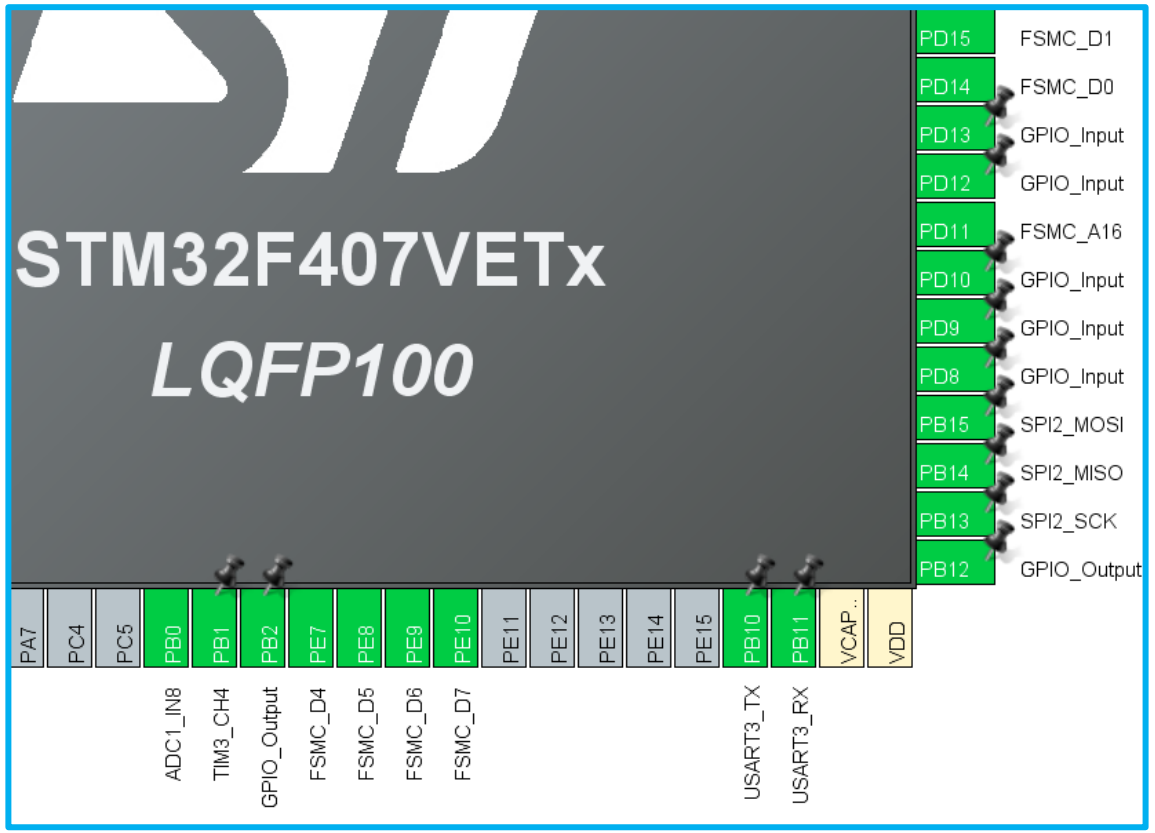
NOR/PSRAM control

Memory type	LCD Interface
Bank	Bank 1 NOR/PSRAM 1
Write operation	Enabled
Extended mode	Disabled

NOR/PSRAM timing

Address setup time in HCLK clock cycles	15
Data setup time in HCLK clock cycles	25
Bus turn around time in HCLK clock cycles	15

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...
PD0	FSMC_D2	n/a	Alternate Fu...	No pull-up an...	Very High
PD1	FSMC_D3	n/a	Alternate Fu...	No pull-up an...	Very High
PD4	FSMC_NOE	n/a	Alternate Fu...	No pull-up an...	Very High
PD5	FSMC_NWE	n/a	Alternate Fu...	No pull-up an...	Very High
PD7	FSMC_NE1	n/a	Alternate Fu...	No pull-up an...	Very High
PD11	FSMC_A16	n/a	Alternate Fu...	No pull-up an...	Very High
PD14	FSMC_D0	n/a	Alternate Fu...	No pull-up an...	Very High
PD15	FSMC_D1	n/a	Alternate Fu...	No pull-up an...	Very High
PE7	FSMC_D4	n/a	Alternate Fu...	No pull-up an...	Very High
PE8	FSMC_D5	n/a	Alternate Fu...	No pull-up an...	Very High
PE9	FSMC_D6	n/a	Alternate Fu...	No pull-up an...	Very High
PE10	FSMC_D7	n/a	Alternate Fu...	No pull-up an...	Very High




```
#define ADC06 *(volatile unsigned short int *)(0x60000000)
void TIM3_IRQHandler(void)        //62.5us
{  static  short Clicks,C15,*px,ADdd[6],VFct[5];

   iDelay(1100);                        GPIOD->BSRR|=0xc<<16;
ADdd[0]=(ADC06&0x1fe0)<<3; ADdd[0]=(((ADC06&0x1fe0)>>5)|ADdd[0])>>1;
ADdd[1]=(ADC06&0x1fe0)<<3; ADdd[1]=(((ADC06&0x1fe0)>>5)|ADdd[1])>>1;
ADdd[2]=(ADC06&0x1fe0)<<3; ADdd[2]=(((ADC06&0x1fe0)>>5)|ADdd[2])>>1;
ADdd[3]=(ADC06&0x1fe0)<<3; ADdd[3]=(((ADC06&0x1fe0)>>5)|ADdd[3])>>1;
ADdd[4]=(ADC06&0x1fe0)<<3; ADdd[4]=(((ADC06&0x1fe0)>>5)|ADdd[4])>>1;
```

bit band 位带------“非” 重要内容

应用中，有时会遇到按” 位”操作的应用

01100101　　如读取b2，或对其置1（set），或置0(clr)

常规做法是按字(节)访问，通过字(节)的 AND \ OR 操作，改变其取值

位应用常见吗?

Without Bit-Band

```
LDR    R0,=0x20000000 ; Setup address
LDR    R1, [R0]       ; Read
ORR.W  R1, #0x4       ; Modify bit
STR    R1, [R0] ; Write back result
```

With Bit-Band

```
LDR    R0,=0x22000008 ; Setup address
MOV    R1, #1         ; Setup data
STR    R1, [R0]       ; Write
```

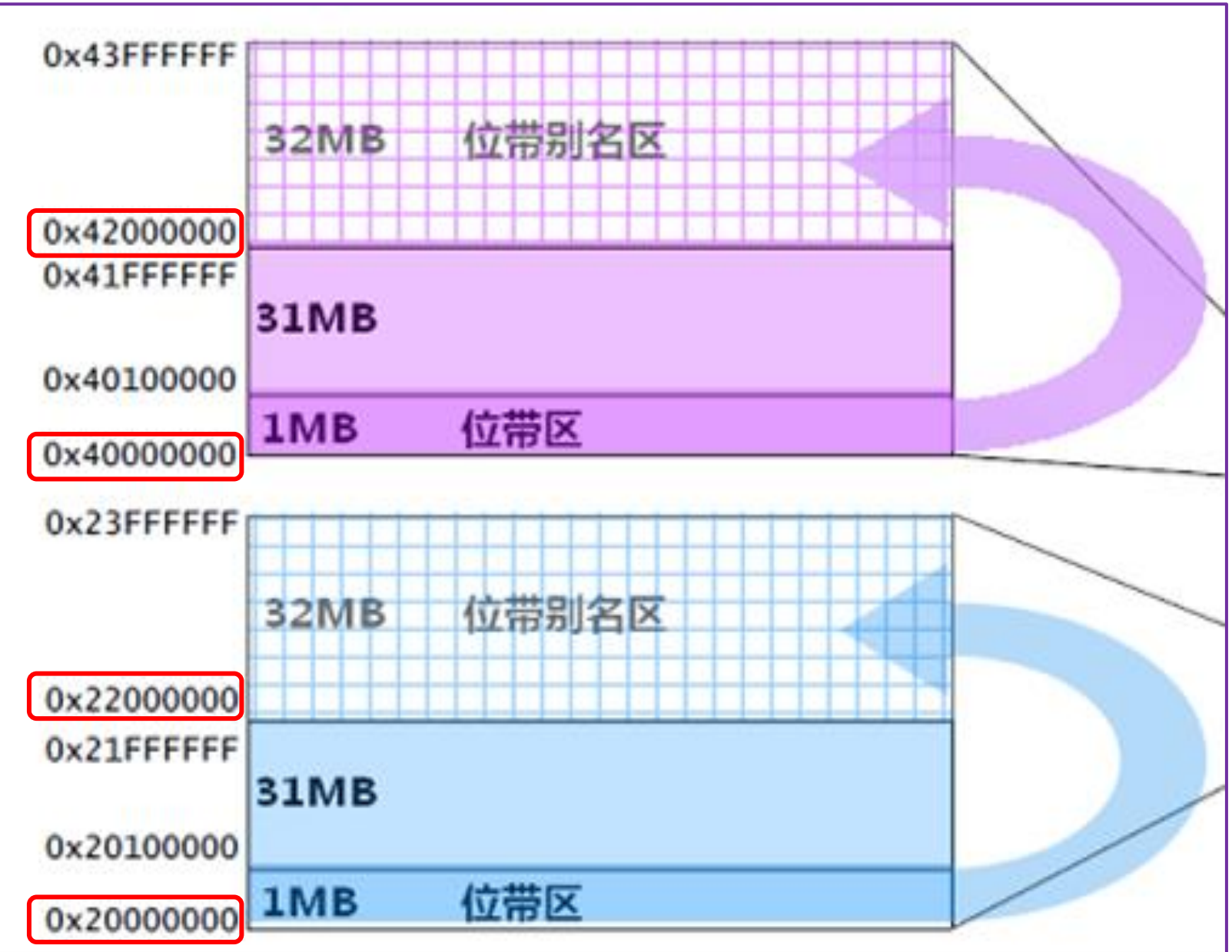
原子

无位带

```
LDR     R0,=0x20000000 ; 建立地址
LDR     R1, [R0]       ; Read
UBFX.W  R1,R1, #2, #1  ; 提取bit2
```

有位带

```
LDR    R0,=0x22000008 ; 建立地址
LDR    R1, [R0]       ; Read
```



Cortex-Mx

存储器系统支持“位带”（bit - band）操作，实现对单一比特的原子操作

0x4000_0000 - 0x400F_FFFF
（片上外设区中的最低1MB）

0x2000_0000 - 0x200F_FFFF
（SRAM 区中的最低1MB）

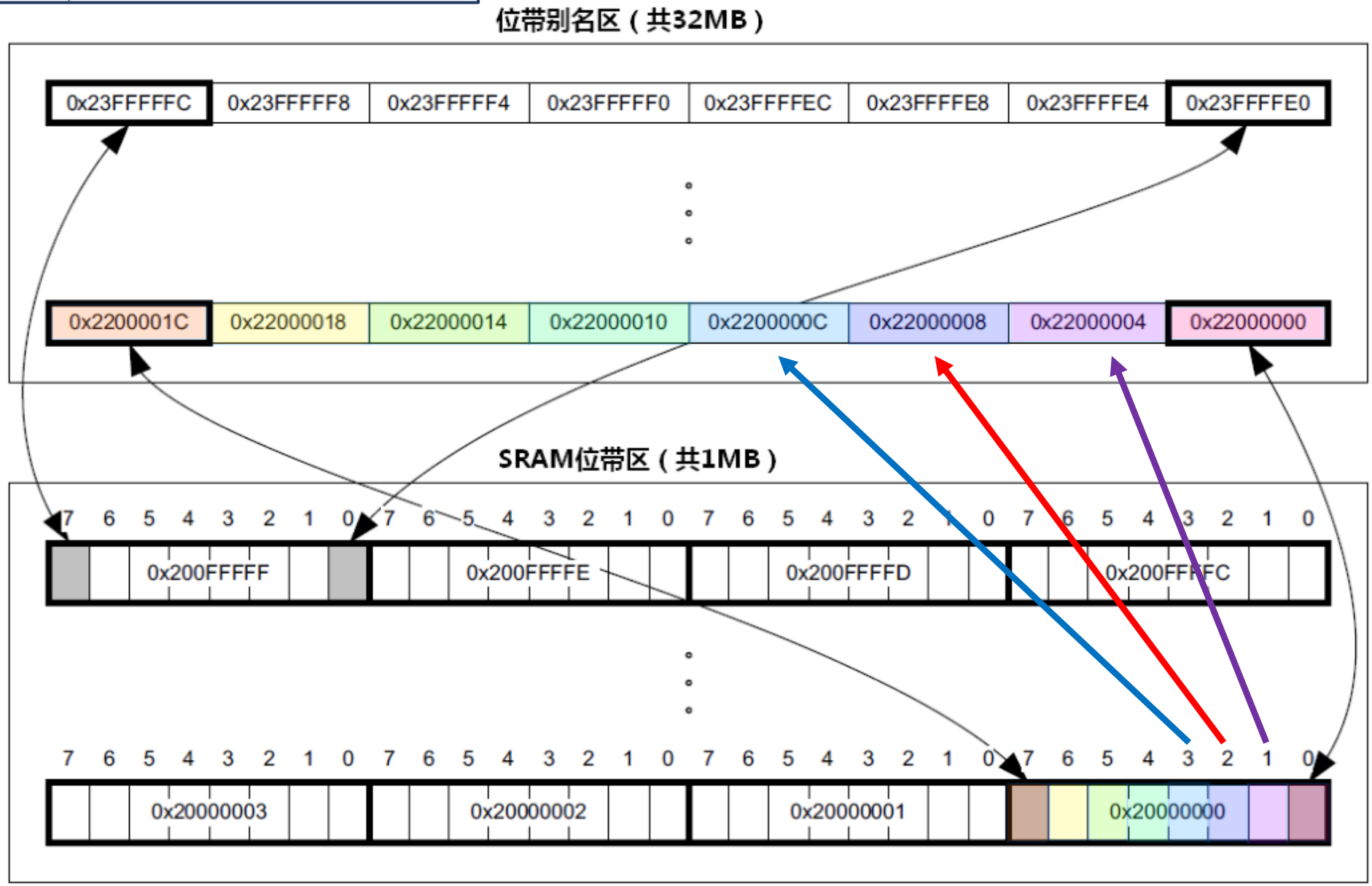
无位带

```
LDR     R0,=0x20000000 ; 建立地址
LDR     R1, [R0]       ; Read
UBFX.W R1,R1, #2, #1   ; 提取bit2
```

有位带

```
LDR     R0,=0x22000008 ; 建立地址
LDR     R1, [R0]       ; Read
```

位带(Bit-band)操作可以使用普通的加载/存储指令来对单一的比特进行读写。可通过“位带别名区”被访问。



位带(Bit-band)　C的应用示例

bit_word_offset = (byte_offset x 32) + (bit_number × 4)
bit_word_addr = bit_band_base + bit_word_offset

```
#define BITBAND(addr, bitnum)   ((addr & 0xF0000000)+0x2000000 + ((addr &0xFF FFF)<<5) + (bitnum<<2))

#define MEM_ADDR(addr)     *((volatile unsigned long *)(addr))

        MEM_ADDR(BITBAND( (u32)&CRCValue, 3 )) = 0x1;

        if(MEM_ADDR(BITBAND( (u32)&CRCValue, 23 ))==1)
        {
        }
```

对于IO--------例如亮、灭LED

使用STM32库:

```
        GPIO_ResetBits(GPIOC, GPIO_Pin_4);      //关LED4
        GPIO_SetBits(GPIOC, GPIO_Pin_7);        //开LED7
```

一般读操作:

```
        STM32_Gpioc_Regs->bsrr.bit.BR4 =1;      // 1：清除对应的ODRy位为0
        STM32_Gpioc_Regs->bsrr.bit.BS7 =1;      // 1：设置对应的ODRy位为1
```

如果使用位带别名区操作:

```
        STM32_BB_Gpioc_Regs->BSRR.BR[4] =1;   // 1：清除对应的ODRy位为0
        STM32_BB_Gpioc_Regs->BSRR.BS[7] =1;   // 1：设置对应的ODRy位为1
```

```
        /* Configure the interrupt mask */
        if ((GPIO_Init->Mode & GPIO_MODE_IT) == GPIO_MODE_IT)
        {
          SET_BIT(EXTI->IMR, iocurrent);
        }
        else
        {
          CLEAR_BIT(EXTI->IMR, iocurrent);
        }
```