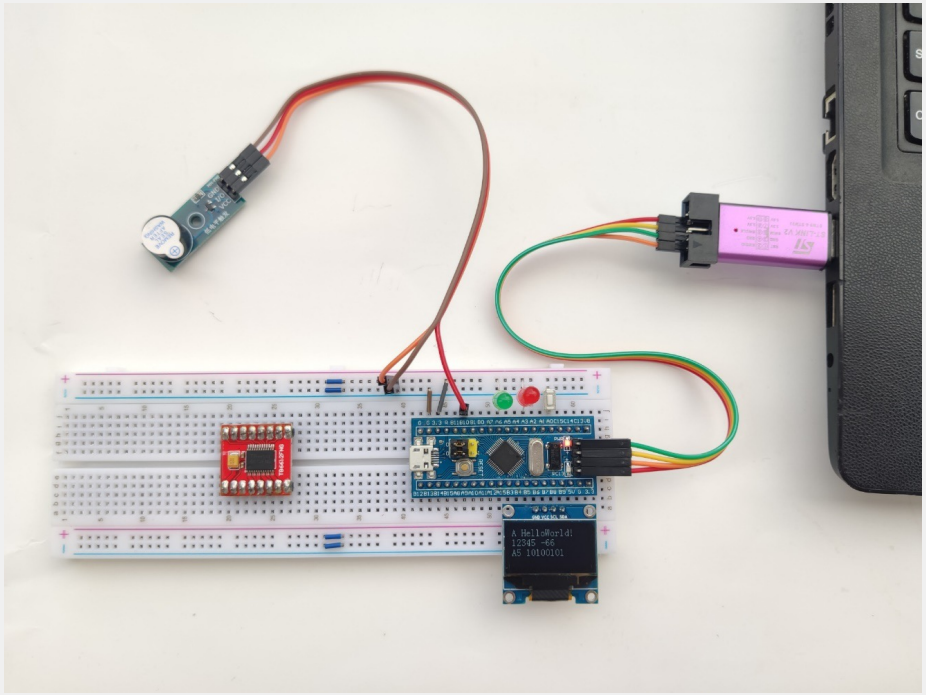
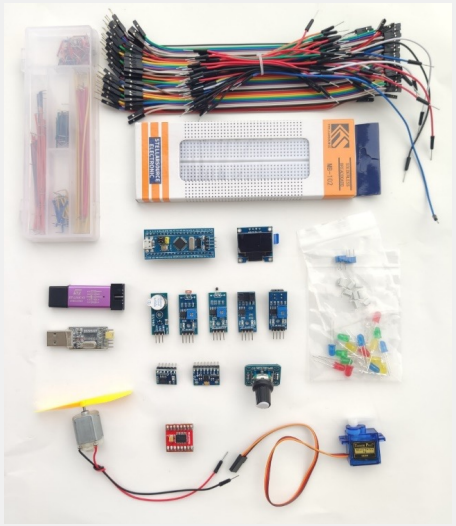
课程简介

- 程序纯手打，手把手教学
- STM32最小系统板+面包板硬件平台



硬件设备

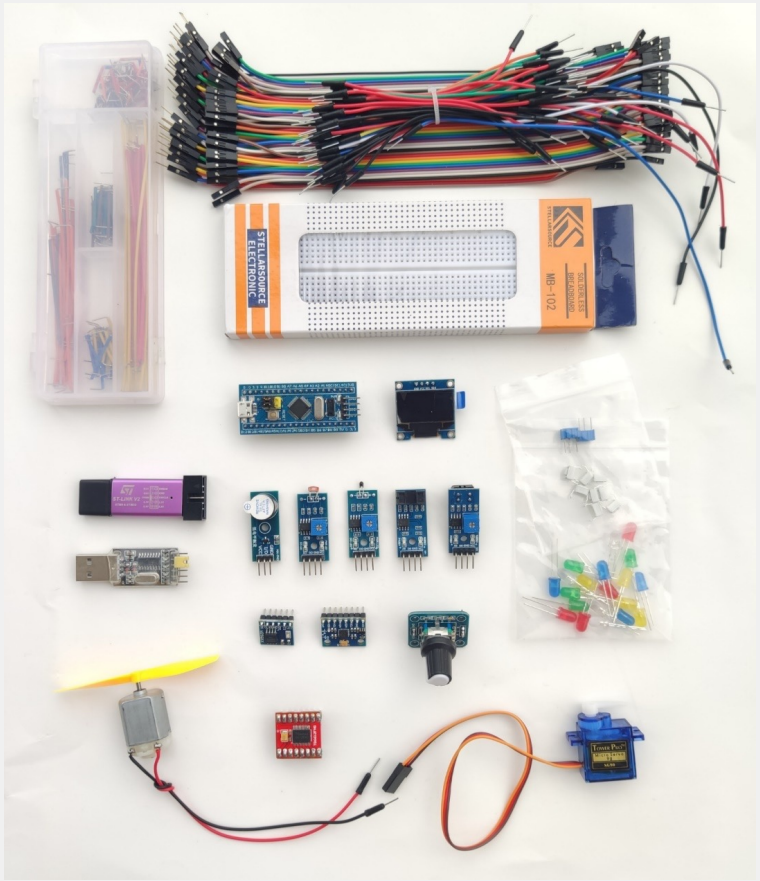
- STM32面包板入门套件
- Windows电脑
- 万用表、示波器、镊子、剪刀等



软件设备

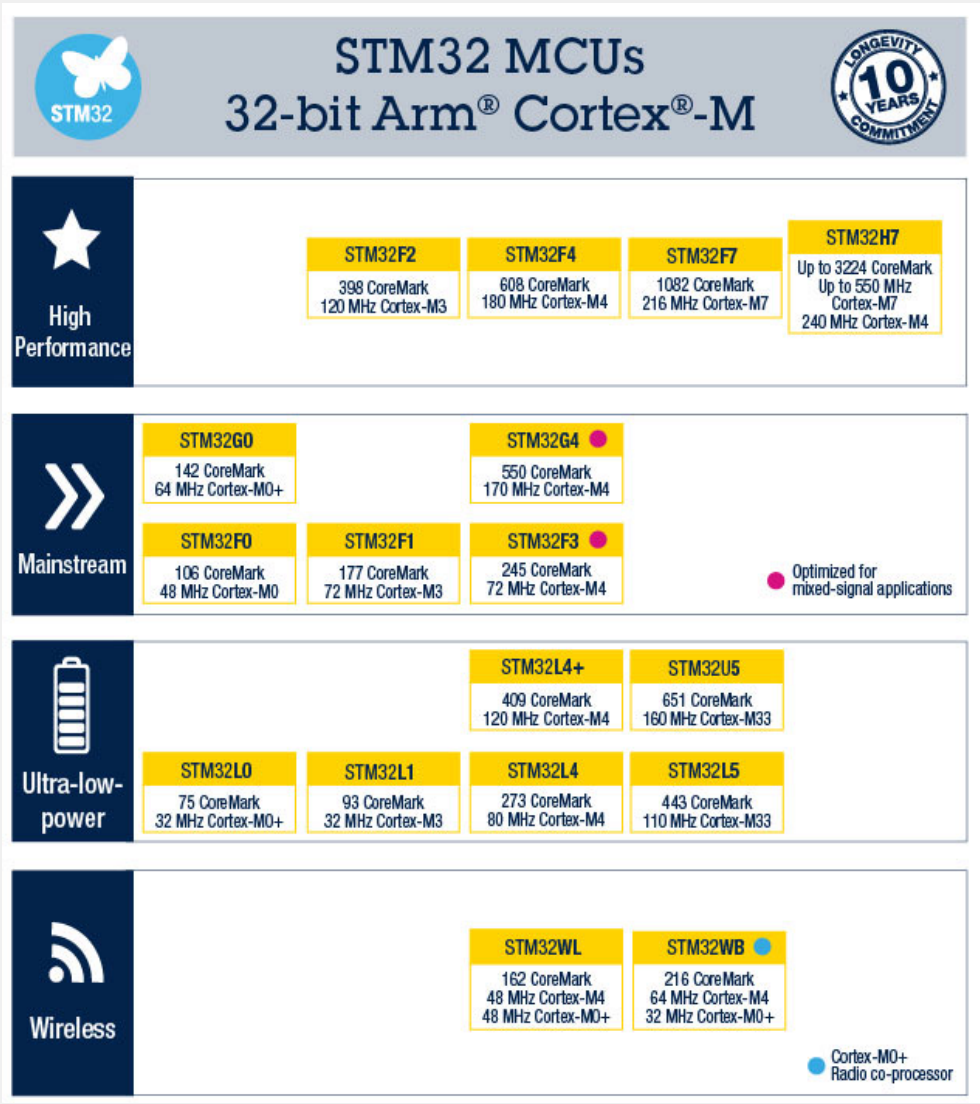
Keil5 MDK

套件介绍



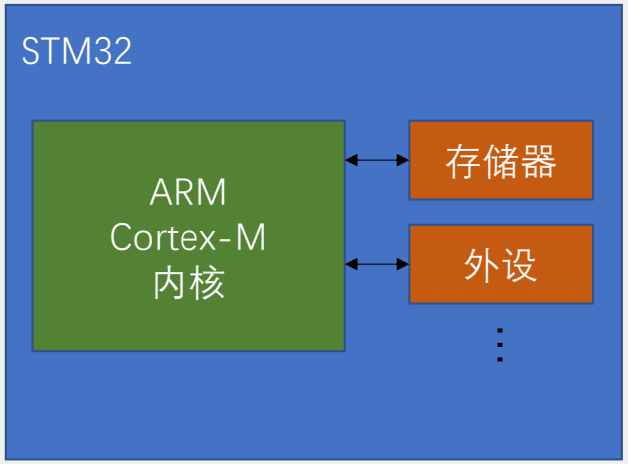
STM32简介

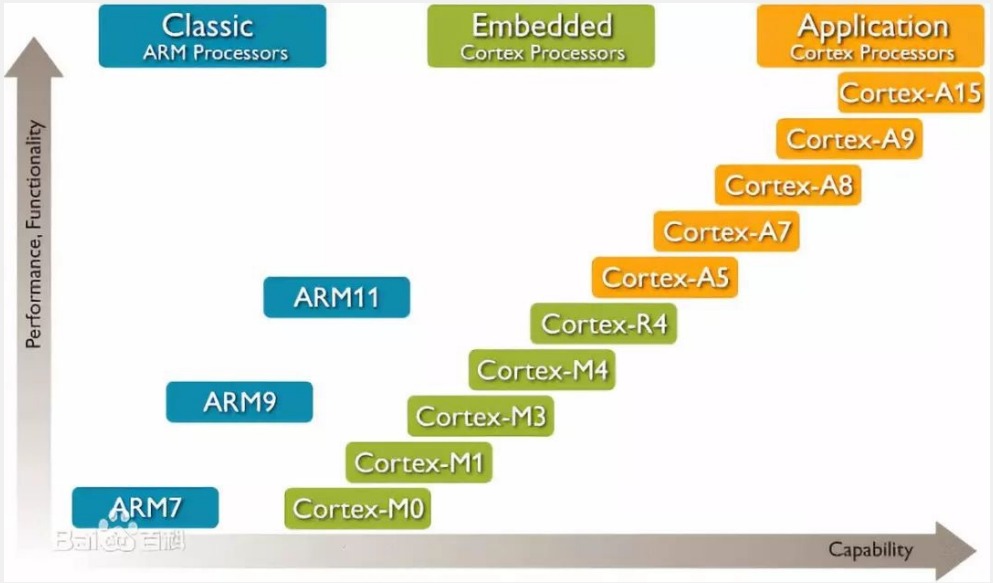
- STM32是ST公司基于ARM Cortex-M内核开发的32位微控制器
- STM32常应用在嵌入式领域，如智能车、无人机、机器人、无线通信、物联网、工业控制、娱乐电子产品等
- STM32功能强大、性能优异、片上资源丰富、功耗低，是一款经典的嵌入式微控制器



ARM

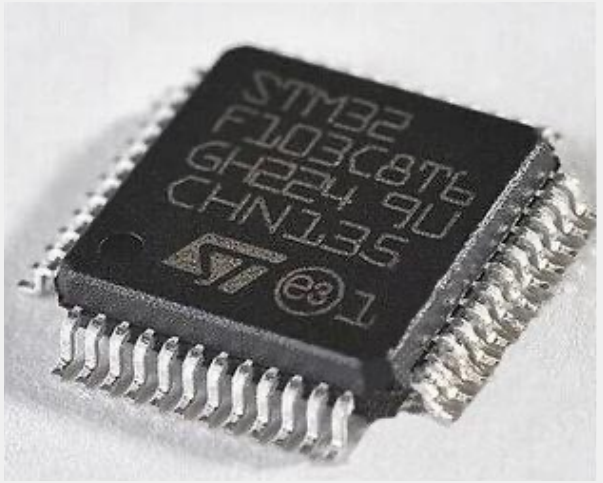
- ARM既指ARM公司，也指ARM处理器内核
- ARM公司是全球领先的半导体知识产权（IP）提供商，全世界超过95%的智能手机和平板电脑都采用ARM架构
- ARM公司设计ARM内核，半导体厂商完善内核周边电路并生产芯片





STM32F103C8T6

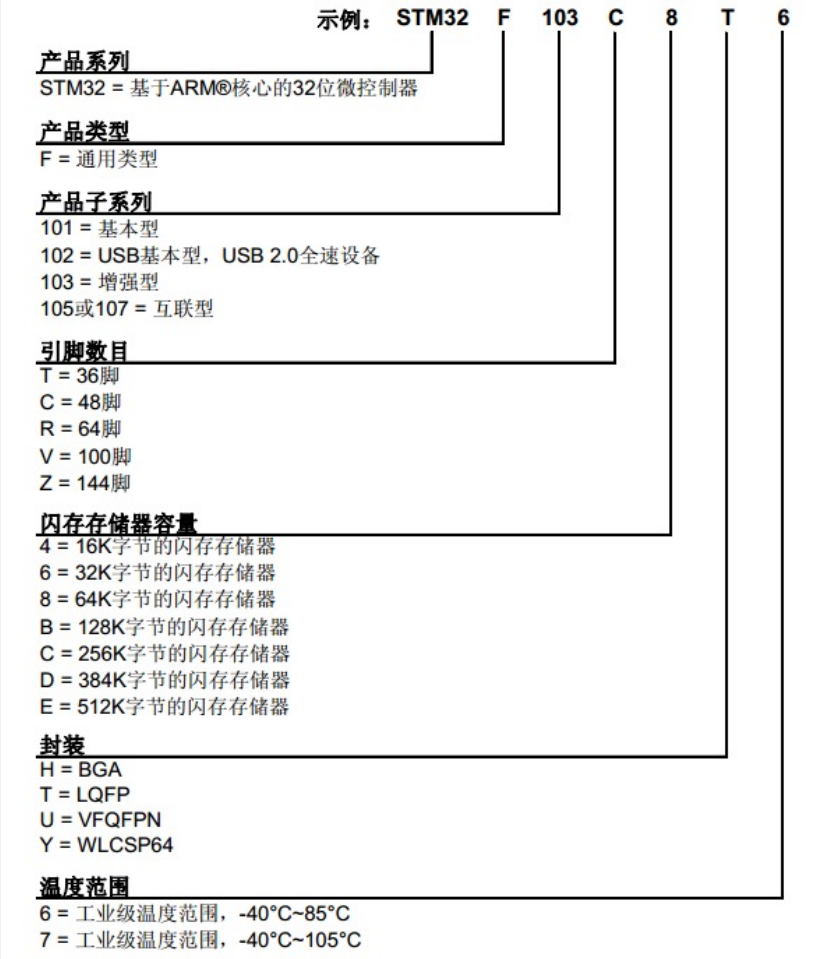
- 系列：主流系列STM32F1
- 内核：ARM Cortex-M3
- 主频：72MHz
- RAM：20K（SRAM）
- ROM：64K（Flash）
- 供电：2.0~3.6V（标准3.3V）
- 封装：LQFP48



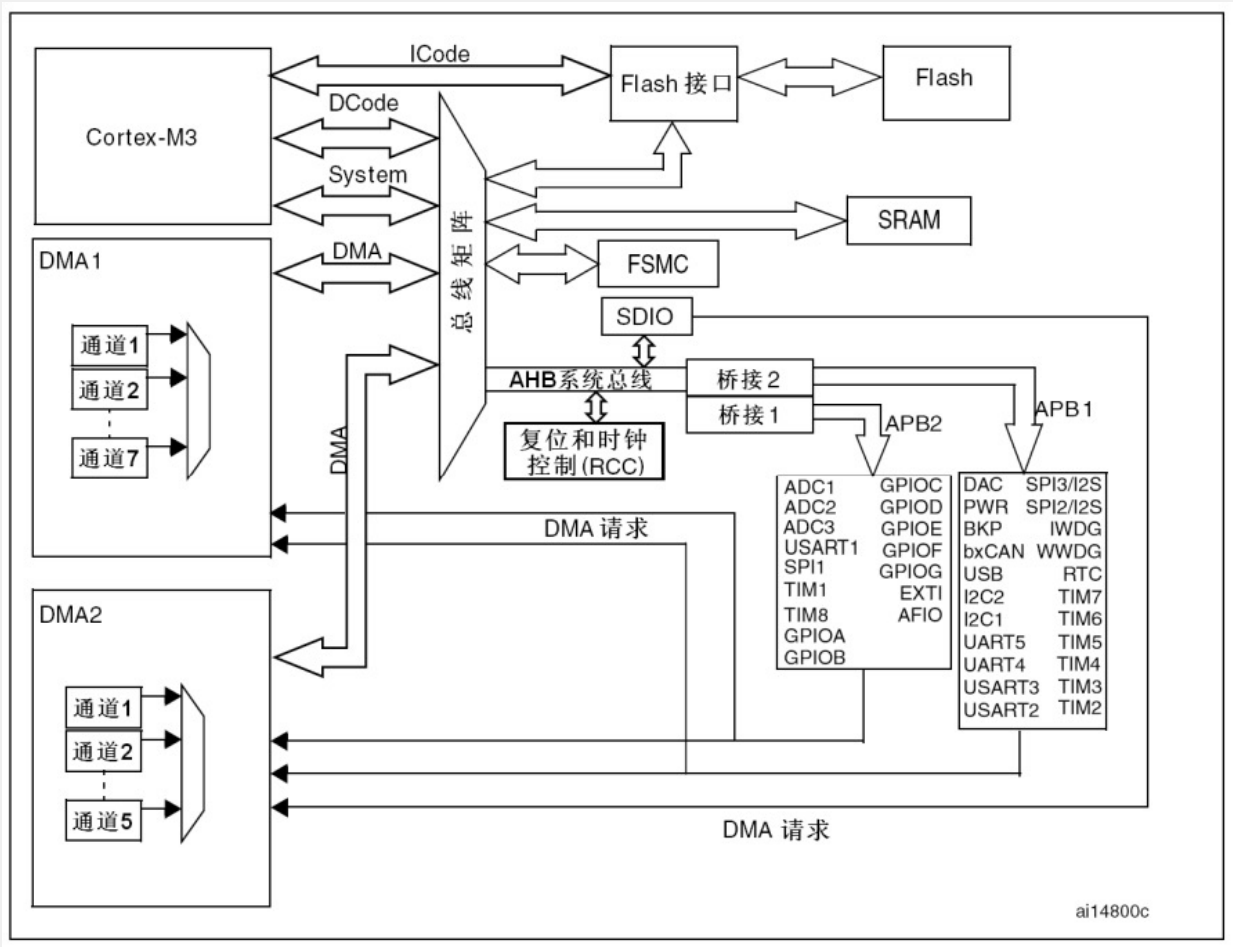
片上资源/外设

英文缩写	名称	英文缩写	名称
NVIC	嵌套向量中断控制器	CAN	CAN通信
SysTick	系统滴答定时器	USB	USB通信
RCC	复位和时钟控制	RTC	实时时钟
GPIO	通用IO口	CRC	CRC校验
AFIO	复用IO口	PWR	电源控制
EXTI	外部中断	BKP	备份寄存器
TIM	定时器	IWDG	独立看门狗
ADC	模数转换器	WWDG	窗口看门狗
DMA	直接内存访问	DAC	数模转换器
USART	同步/异步串口通信	SDIO	SD卡接口
I2C	I2C通信	FSMC	可变静态存储控制器
SPI	SPI通信	USB OTG	USB主机接口

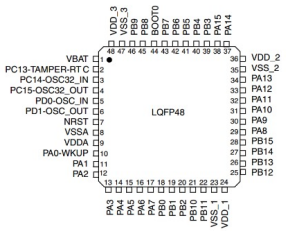
命名规则


示例： STM32 F 103 C 8 T 6
产品系列
STM32 = 基于ARM®核心的32位微控制器
产品类型
F = 通用类型
产品子系列
101 = 基本型
102 = USB基本型，USB 2.0全速设备
103 = 增强型
105或107 = 互联型
引脚数目
T = 36脚
C = 48脚
R = 64脚
V = 100脚
Z = 144脚
闪存存储器容量
4 = 16K字节的闪存存储器
6 = 32K字节的闪存存储器
8 = 64K字节的闪存存储器
B = 128K字节的闪存存储器
C = 256K字节的闪存存储器
D = 384K字节的闪存存储器
E = 512K字节的闪存存储器
封装
H = BGA
T = LQFP
U = VFQFPN
Y = WLCSP64
温度范围
6 = 工业级温度范围，-40°C~85°C
7 = 工业级温度范围，-40°C~105°C

系统结构


Cortex-M3
ICode
DCode
System
DMA
Flash 接口
Flash
SRAM
FSMC
SDIO
总线矩阵
DMA1
通道1
通道2
通道7
DMA2
通道1
通道2
通道5
DMA
AHB系统总线
桥接2
桥接1
APB2
APB1
复位和时钟控制(RCC)
DMA 请求
DMA 请求
ADC1
ADC2
ADC3
USART1
SPI1
TIM1
TIM8
GPIOA
GPIOB
GPIOC
GPIOD
GPIOE
GPIOF
GPIOG
EXTI
AFIO
DAC
PWR
BKP
bxCAN
USB
I2C2
I2C1
UART5
UART4
USART3
USART2
SPI3/I2S
SPI2/I2S
IWDG
WWDG
RTC
TIM7
TIM6
TIM5
TIM4
TIM3
TIM2
ai14800c

引脚定义



STM32F103C8T6
引脚定义

引脚号	引脚名称	类型	I/O口电平	主功能	默认复用功能	重定义功能
1	VBAT	S		VBAT		
2	PC13-TAMPER-RTC	I/O		PC13	TAMPER-RTC	
3	PC14-OSC32_IN	I/O		PC14	OSC32_IN	
4	PC15-OSC32_OUT	I/O		PC15	OSC32_OUT	
5	OSC_IN	I		OSC_IN		
6	OSC_OUT	O		OSC_OUT		
7	NRST	I/O		NRST		
8	VSSA	S		VSSA		
9	VDDA	S		VDDA		
10	**PA0-WKUP**	I/O		PA0	WKUP/USART2_CTS/ADC12_IN0/TIM2_CH1_ETR	
11	**PA1**	I/O		PA1	USART2_RTS/ADC12_IN1/TIM2_CH2	
12	**PA2**	I/O		PA2	USART2_TX/ADC12_IN2/TIM2_CH3	
13	**PA3**	I/O		PA3	USART2_RX/ADC12_IN3/TIM2_CH4	
14	**PA4**	I/O		PA4	SPI1_NSS/USART2_CK/ADC12_IN4	
15	**PA5**	I/O		PA5	SPI1_SCK/ADC12_IN5	
16	**PA6**	I/O		PA6	SPI1_MISO/ADC12_IN6/TIM3_CH1	TIM1_BKIN
17	**PA7**	I/O		PA7	SPI1_MOSI/ADC12_IN7/TIM3_CH2	TIM1_CH1N
18	**PB0**	I/O		PB0	ADC12_IN8/TIM3_CH3	TIM1_CH2N
19	**PB1**	I/O		PB1	ADC12_IN9/TIM3_CH4	TIM1_CH3N
20	PB2	I/O	FT	PB2/BOOT1		
21	**PB10**	I/O	FT	PB10	I2C2_SCL/USART3_TX	TIM2_CH3
22	**PB11**	I/O	FT	PB11	I2C2_SDA/USART3_RX	TIM2_CH4
23	VSS_1	S		VSS_1		
24	VDD_1	S		VDD_1		
25	**PB12**	I/O	FT	PB12	SPI2_NSS/I2C2_SMBAI/USART3_CK/TIM1_BKIN	
26	**PB13**	I/O	FT	PB13	SPI2_SCK/USART3_CTS/TIM1_CH1N	
27	**PB14**	I/O	FT	PB14	SPI2_MISO/USART3_RTS/TIM1_CH2N	
28	**PB15**	I/O	FT	PB15	SPI2_MOSI/TIM1_CH3N	
29	**PA8**	I/O	FT	PA8	USART1_CK/TIM1_CH1/MCO	
30	**PA9**	I/O	FT	PA9	USART1_TX/TIM1_CH2	
31	**PA10**	I/O	FT	PA10	USART1_RX/TIM1_CH3	
32	**PA11**	I/O	FT	PA11	USART1_CTS/USBDM/CAN_RX/TIM1_CH4	
33	**PA12**	I/O	FT	PA12	USART1_RTS/USBDP/CAN_TX/TIM1_ETR	
34	PA13	I/O	FT	JTMS/SWDIO		PA13
35	VSS_2	S		VSS_2		
36	VDD_2	S		VDD_2		
37	PA14	I/O	FT	JTCK/SWCLK		PA14
38	PA15	I/O	FT	JTDI		TIM2_CH1_ETR/PA15/SPI1_NSS
39	PB3	I/O	FT	JTDO		PB3/TRACESWO/TIM2_CH2/SPI1_SCK
40	PB4	I/O	FT	NJTRST		PB4/TIM3_CH1/SPI1_MISO
41	**PB5**	I/O		PB5	I2C1_SMBAI	TIM3_CH2/SPI1_MOSI
42	**PB6**	I/O	FT	PB6	I2C1_SCL/TIM4_CH1	USART1_TX
43	**PB7**	I/O	FT	PB7	I2C1_SDA/TIM4_CH2	USART1_RX
44	BOOT0	I		BOOT0		
45	**PB8**	I/O	FT	PB8	TIM4_CH3	I2C1_SCL/CAN_RX
46	**PB9**	I/O	FT	PB9	TIM4_CH4	I2C1_SDA/CAN_TX
47	VSS_3	S		VSS_3		
48	VDD_3	S		VDD_3		

启动配置

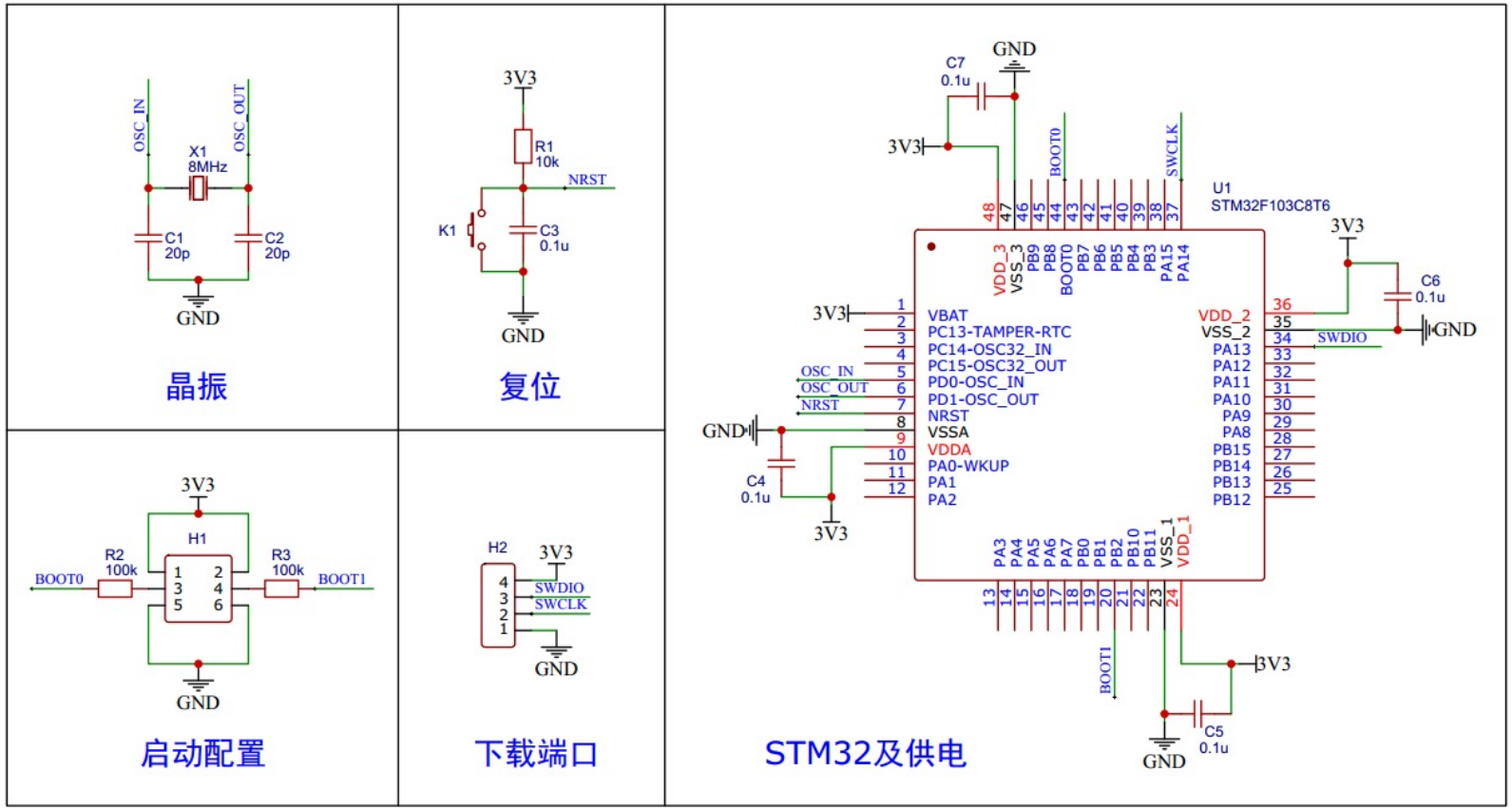
在STM32F10xxx里，可以通过BOOT[1:0]引脚选择三种不同启动模式。

表6　启动模式

启动模式选择引脚		启动模式	说明
BOOT1	BOOT0		
X	0	主闪存存储器	主闪存存储器被选为启动区域
0	1	系统存储器	系统存储器被选为启动区域
1	1	内置SRAM	内置SRAM被选为启动区域

在系统复位后，SYSCLK的第4个上升沿，BOOT引脚的值将被锁存。用户可以通过设置BOOT1和BOOT0引脚的状态，来选择在复位后的启动模式。

最小系统电路


OSC_IN
OSC_OUT
X1
8MHz
C1
20p
C2
20p
GND
晶振
3V3
R1
10k
NRST
K1
C3
0.1u
GND
复位
3V3
H1
R2
100k
R3
100k
BOOT0
BOOT1
1
2
3
4
5
6
GND
启动配置
H2
3V3
4
3
2
1
SWDIO
SWCLK
GND
下载端口
C7
0.1u
GND
3V3
BOOT0
SWCLK
U1
STM32F103C8T6
VDD_3
VSS_3
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PA15
PA14
48
47
46
45
44
43
42
41
40
39
38
37
3V3
C6
0.1u
GND
VBAT
PC13-TAMPER-RTC
PC14-OSC32_IN
PC15-OSC32_OUT
PD0-OSC_IN
PD1-OSC_OUT
NRST
VSSA
VDDA
PA0-WKUP
PA1
PA2
OSC_IN
OSC_OUT
NRST
GND
C4
0.1u
3V3
VDD_2
VSS_2
PA13
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12
SWDIO
36
35
34
33
32
31
30
29
28
27
26
25
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB10
PB11
VSS_1
VDD_1
13
14
15
16
17
18
19
20
21
22
23
24
BOOT1
3V3
C5
0.1u
GND
STM32及供电

软件安装

- 安装Keil5 MDK
- 安装器件支持包
- 软件注册
- 安装STLINK驱动
- 安装USB转串口驱动

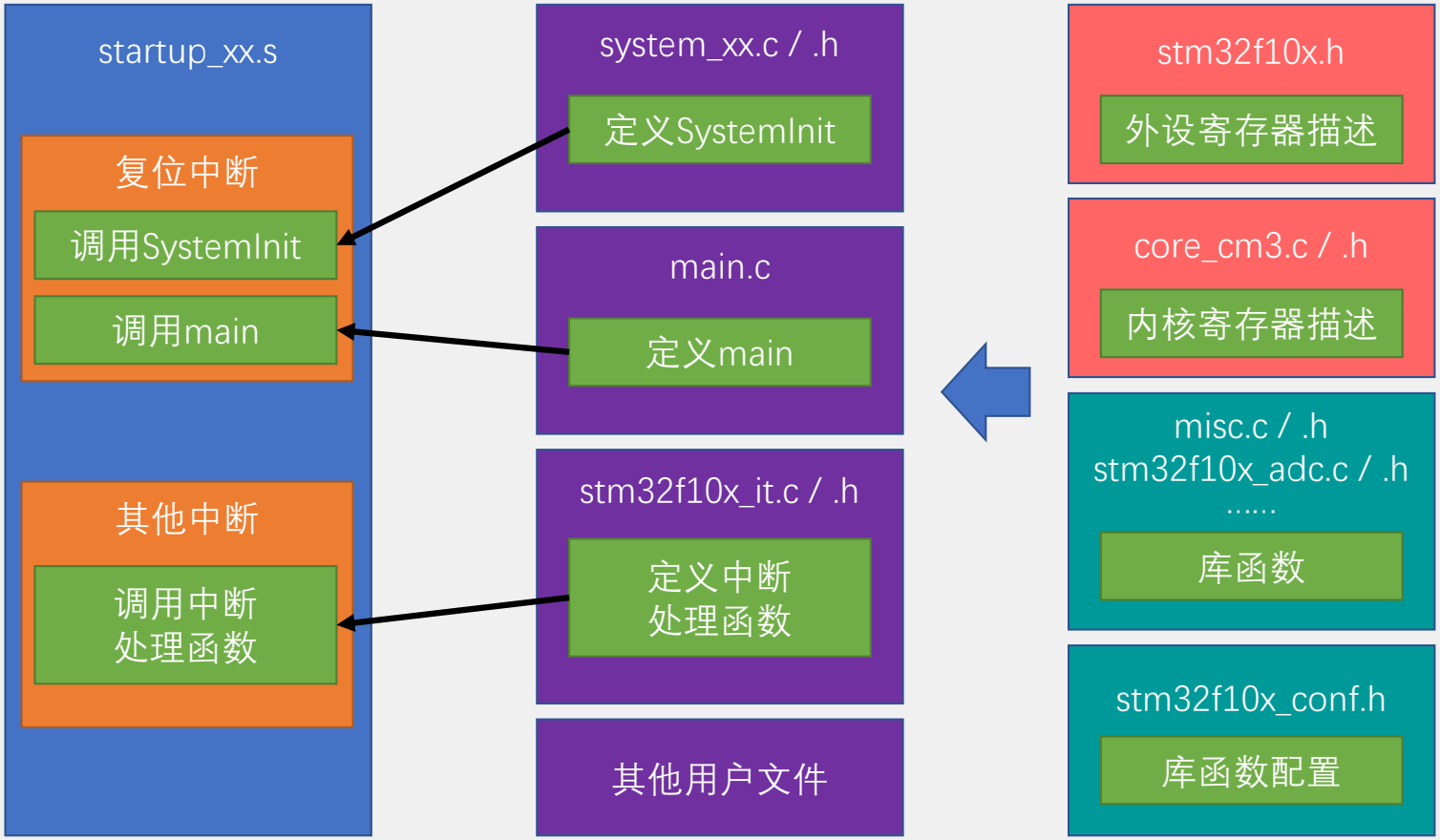
型号分类及缩写

缩写	释义	Flash容量	型号
LD_VL	小容量产品超值系列	16~32K	STM32F100
MD_VL	中容量产品超值系列	64~128K	STM32F100
HD_VL	大容量产品超值系列	256~512K	STM32F100
LD	小容量产品	16~32K	STM32F101/102/103
MD	中容量产品	64~128K	STM32F101/102/103
HD	大容量产品	256~512K	STM32F101/102/103
XL	加大容量产品	大于512K	STM32F101/102/103
CL	互联型产品	-	STM32F105/107

新建工程步骤

- 建立工程文件夹，Keil中新建工程，选择型号
- 工程文件夹里建立Start、Library、User等文件夹，复制固件库里面的文件到工程文件夹
- 工程里对应建立Start、Library、User等同名称的分组，然后将文件夹内的文件添加到工程分组里
- 工程选项，C/C++，Include Paths内声明所有包含头文件的文件夹
- 工程选项，C/C++，Define内定义USE_STDPERIPH_DRIVER
- 工程选项，Debug，下拉列表选择对应调试器，Settings，Flash Download里勾选Reset and Run

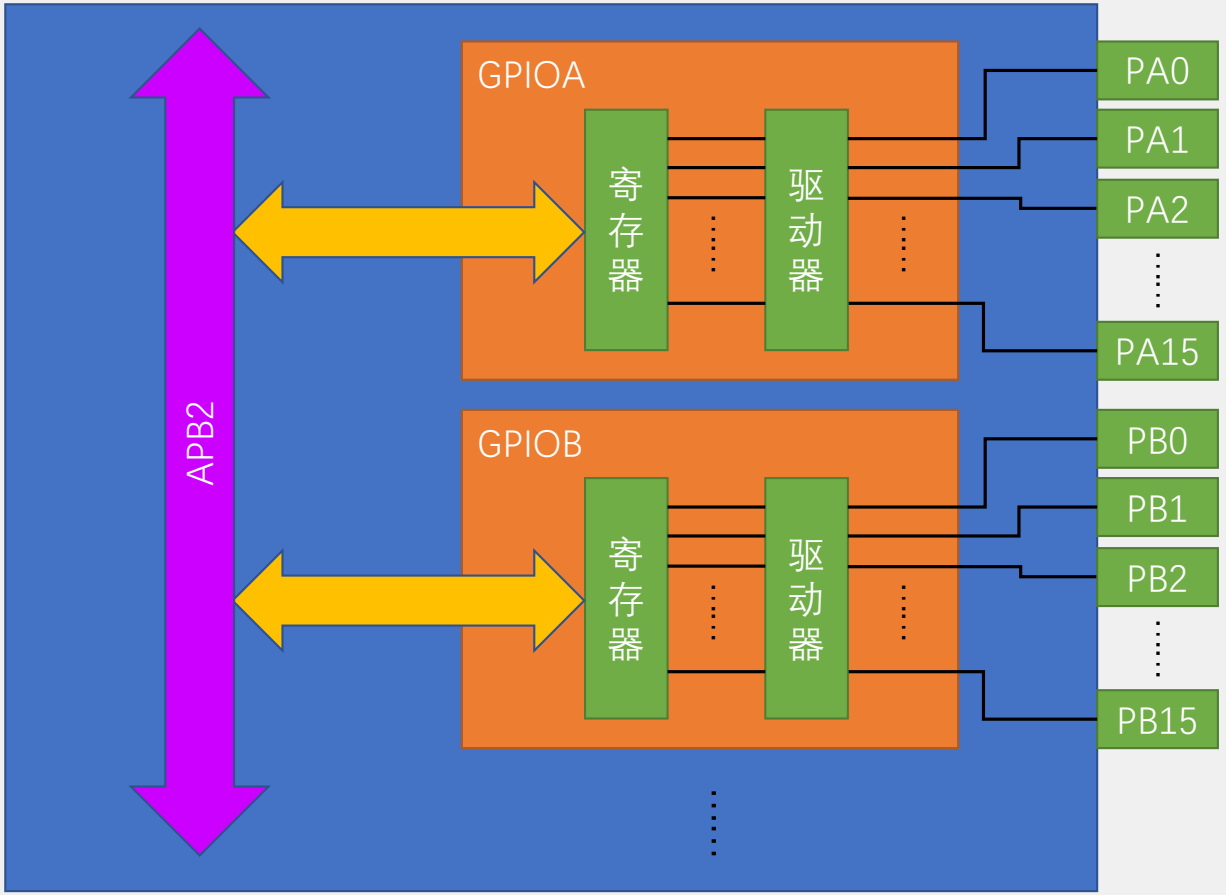
工程架构


startup_xx.s
复位中断
调用SystemInit
调用main
其他中断
调用中断
处理函数
system_xx.c / .h
定义SystemInit
main.c
定义main
stm32f10x_it.c / .h
定义中断
处理函数
其他用户文件
stm32f10x.h
外设寄存器描述
core_cm3.c / .h
内核寄存器描述
misc.c / .h
stm32f10x_adc.c / .h
.....
库函数
stm32f10x_conf.h
库函数配置

GPIO简介

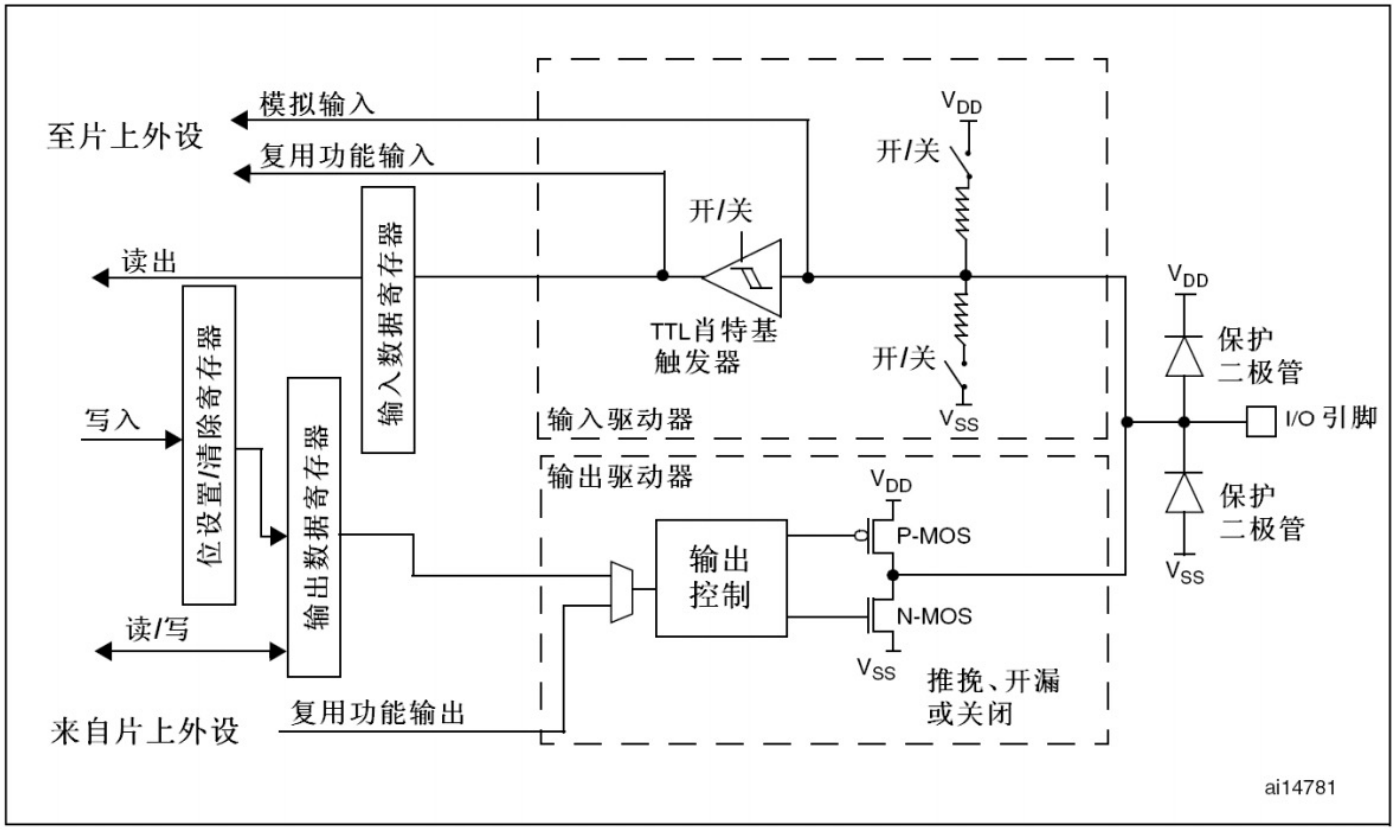
- GPIO（General Purpose Input Output）通用输入输出口
- 可配置为8种输入输出模式
- 引脚电平：0V~3.3V，部分引脚可容忍5V
- 输出模式下可控制端口输出高低电平，用以驱动LED、控制蜂鸣器、模拟通信协议输出时序等
- 输入模式下可读取端口的高低电平或电压，用于读取按键输入、外接模块电平信号输入、ADC电压采集、模拟通信协议接收数据等

GPIO基本结构


APB2
GPIOA
寄存器
驱动器
PA0
PA1
PA2
PA15
GPIOB
寄存器
驱动器
PB0
PB1
PB2
PB15

GPIO位结构

图13　I/O端口位的基本结构



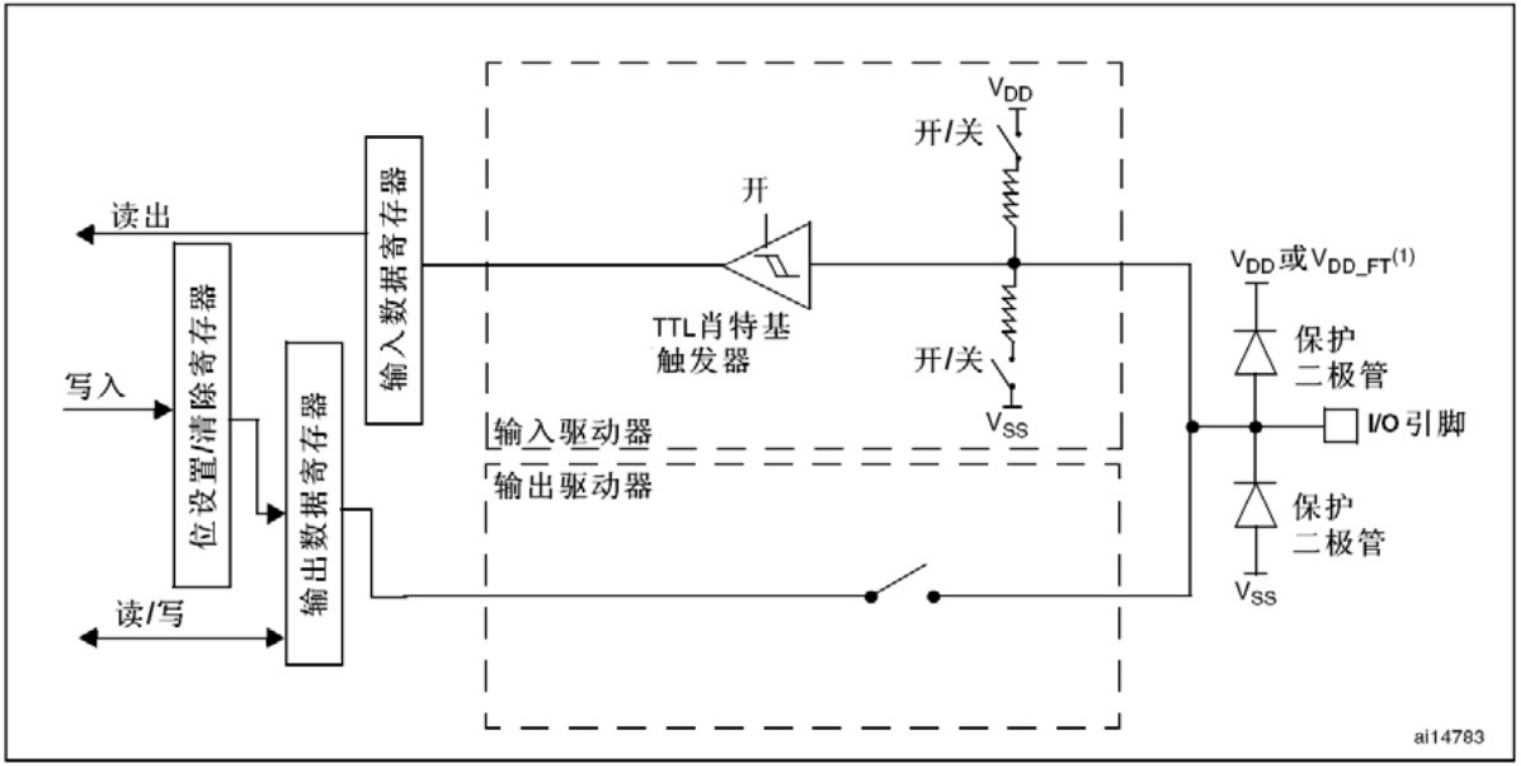
GPIO模式

- 通过配置GPIO的端口配置寄存器，端口可以配置成以下8种模式

模式名称	性质	特征
浮空输入	数字输入	可读取引脚电平，若引脚悬空，则电平不确定
上拉输入	数字输入	可读取引脚电平，内部连接上拉电阻，悬空时默认高电平
下拉输入	数字输入	可读取引脚电平，内部连接下拉电阻，悬空时默认低电平
模拟输入	模拟输入	GPIO无效，引脚直接接入内部ADC
开漏输出	数字输出	可输出引脚电平，高电平为高阻态，低电平接VSS
推挽输出	数字输出	可输出引脚电平，高电平接VDD，低电平接VSS
复用开漏输出	数字输出	由片上外设控制，高电平为高阻态，低电平接VSS
复用推挽输出	数字输出	由片上外设控制，高电平接VDD，低电平接VSS

浮空/上拉/下拉输入

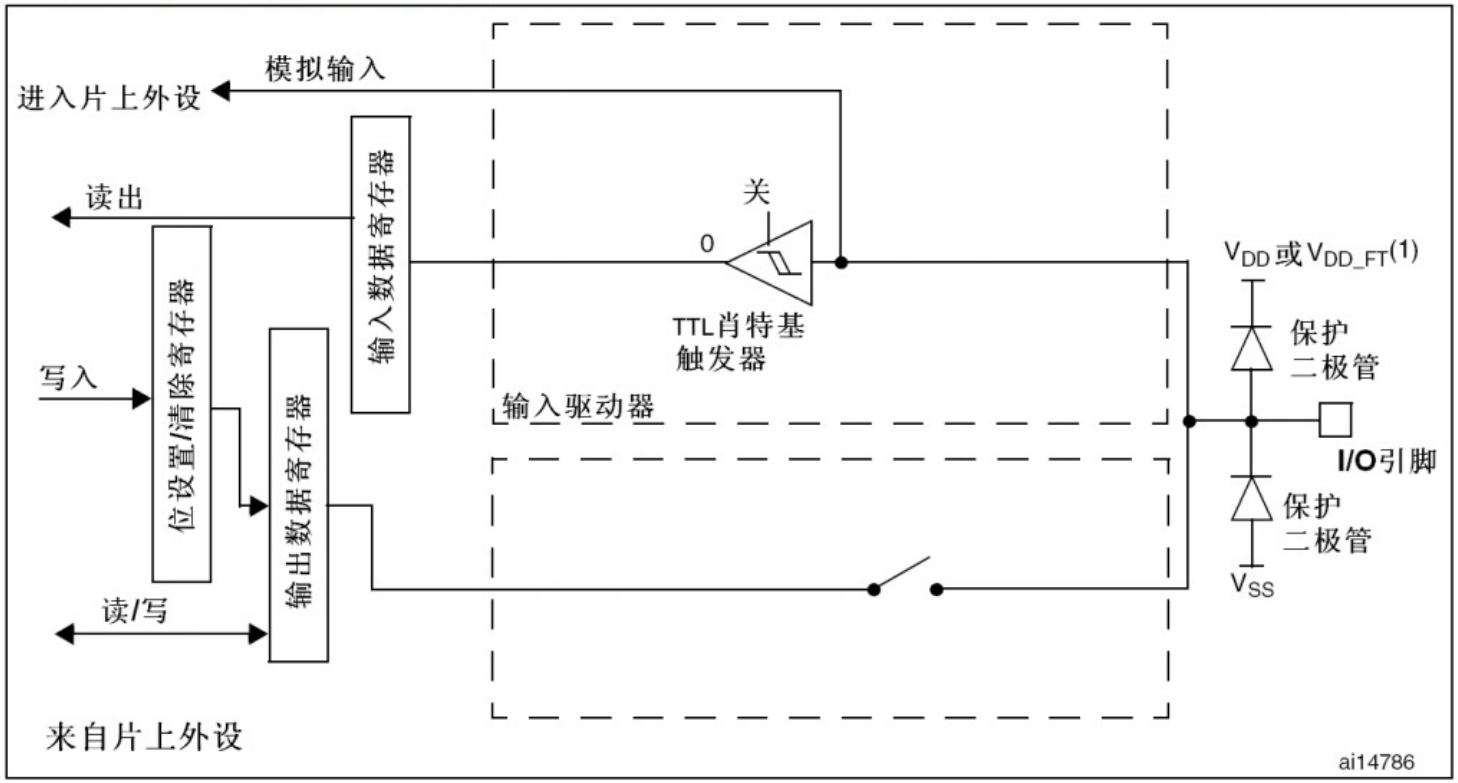
图15　输入浮空/上拉/下拉配置



(1) V_{DD_FT} 对5伏容忍I/O脚是特殊的，它与V_{DD}不同

模拟输入

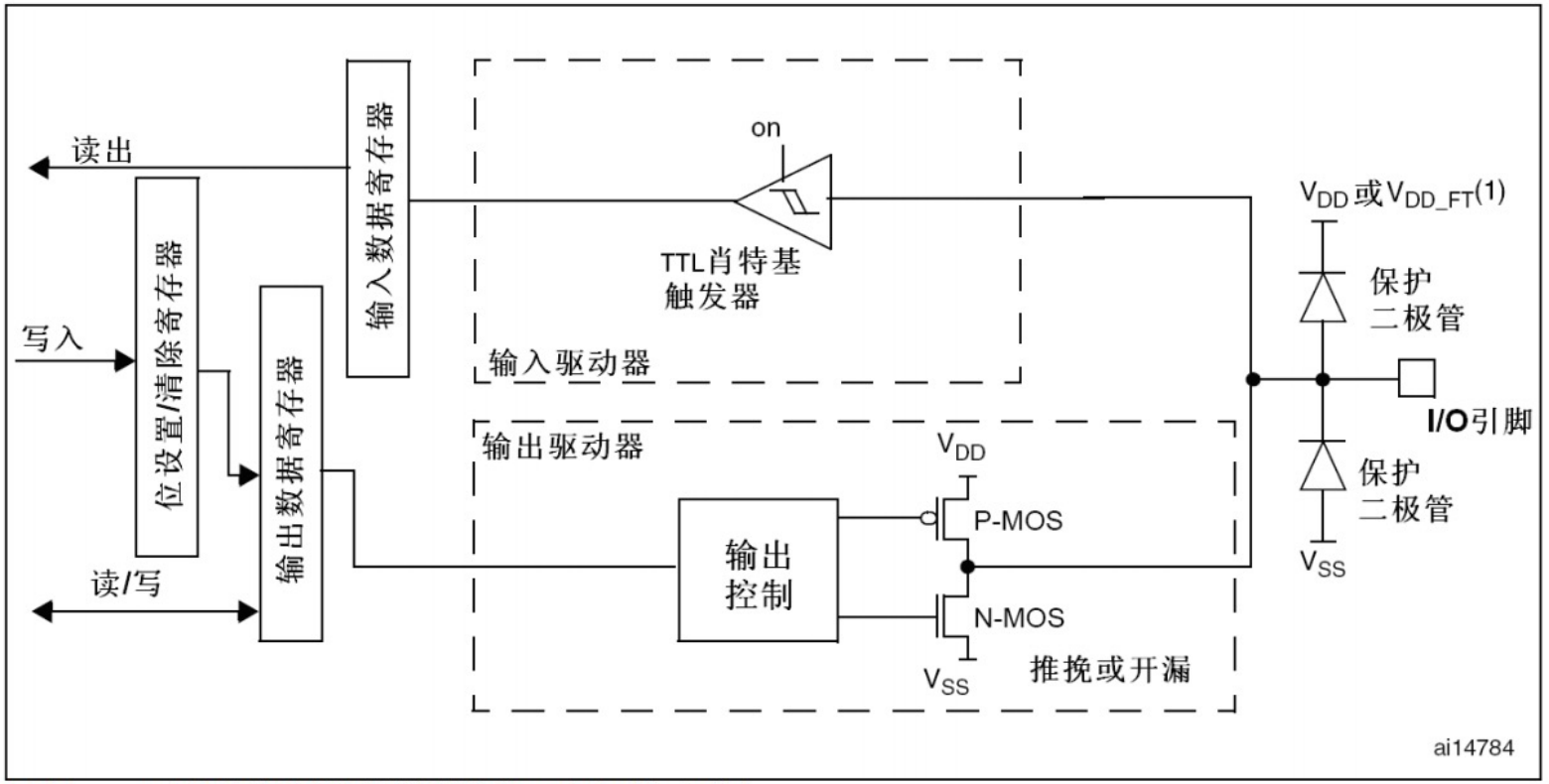
图18　高阻抗的模拟输入配置



(1) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

开漏/推挽输出

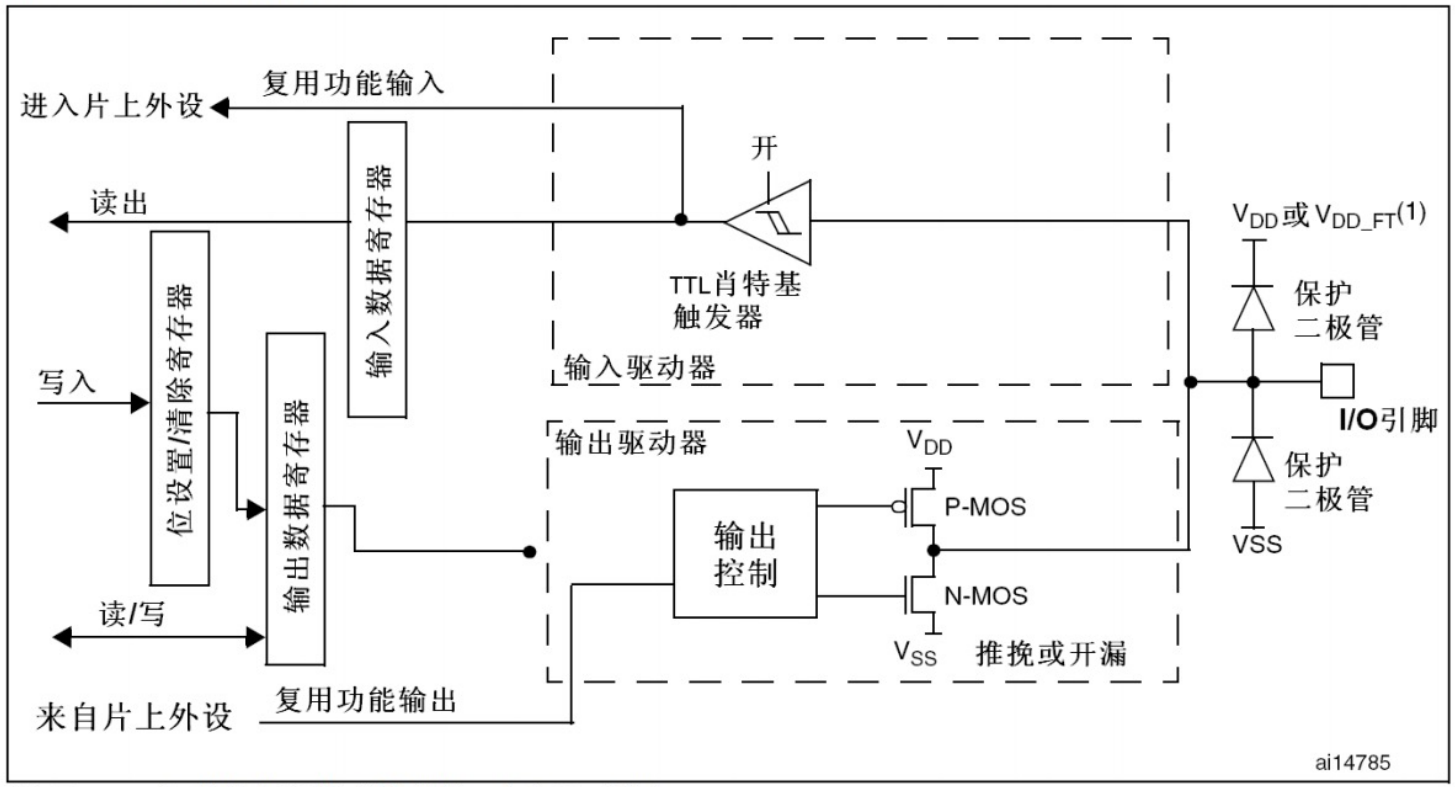
图16　输出配置



(1) V_{DD_FT} 对 5 伏兼容 I/O 脚是特殊的，它与 V_{DD} 不同

复用开漏/推挽输出

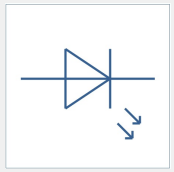
图17　复用功能配置

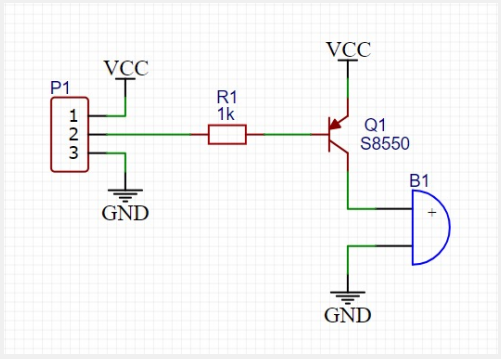


(1) V_{DD_FT} 对5伏兼容I/O脚是特殊的，它与V_{DD}不同

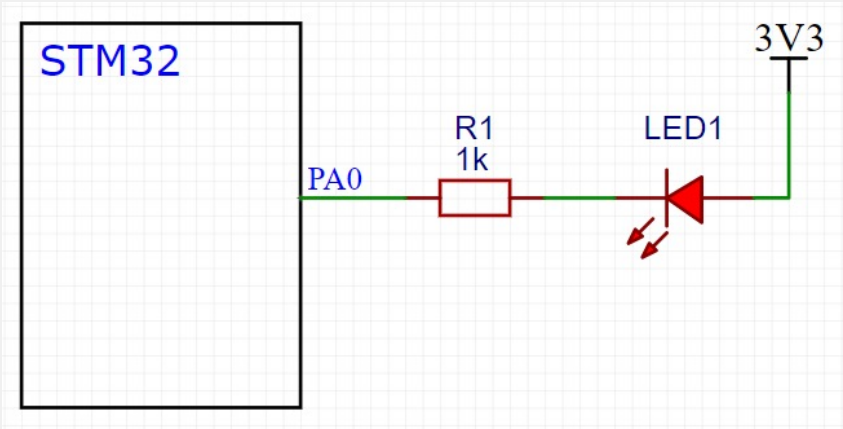
LED和蜂鸣器简介

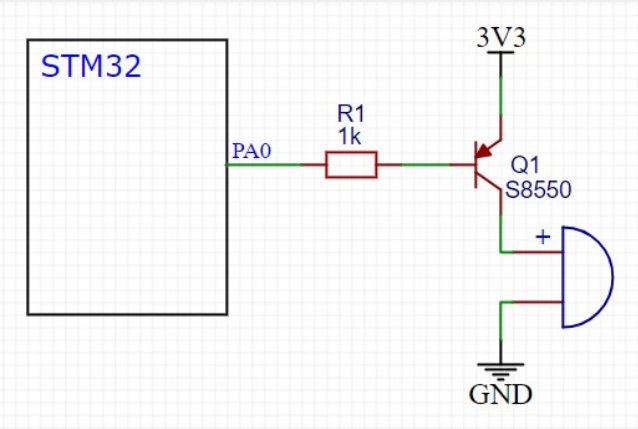
- LED：发光二极管，正向通电点亮，反向通电不亮
- 有源蜂鸣器：内部自带振荡源，将正负极接上直流电压即可持续发声，频率固定
- 无源蜂鸣器：内部不带振荡源，需要控制器提供振荡脉冲才可发声，调整提供振荡脉冲的频率，可发出不同频率的声音

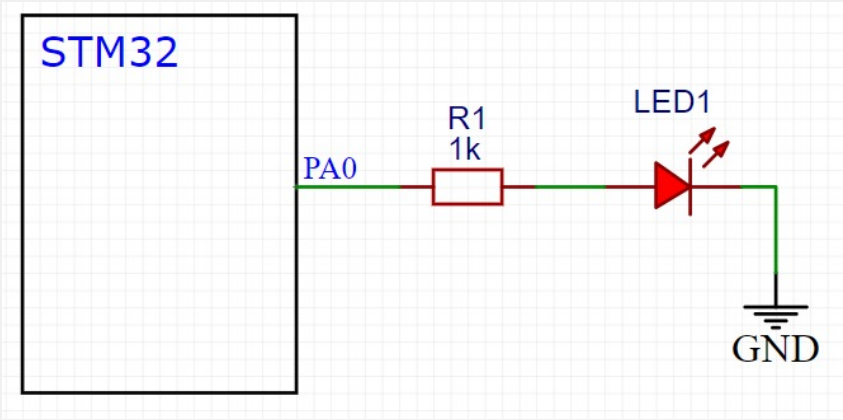


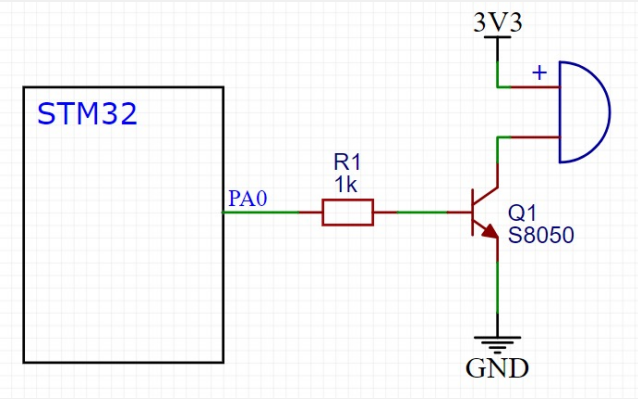


硬件电路

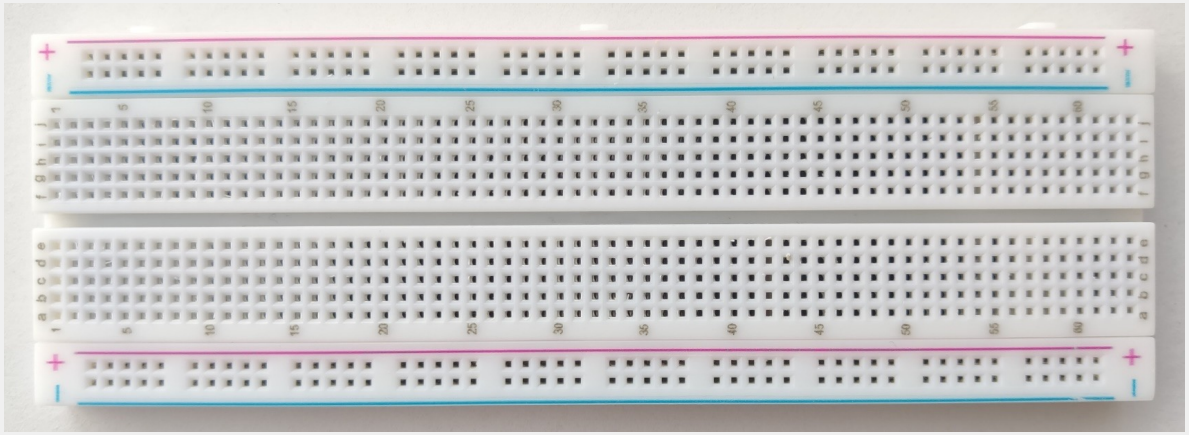

STM32
3V3
R1
1k
LED1
PA0

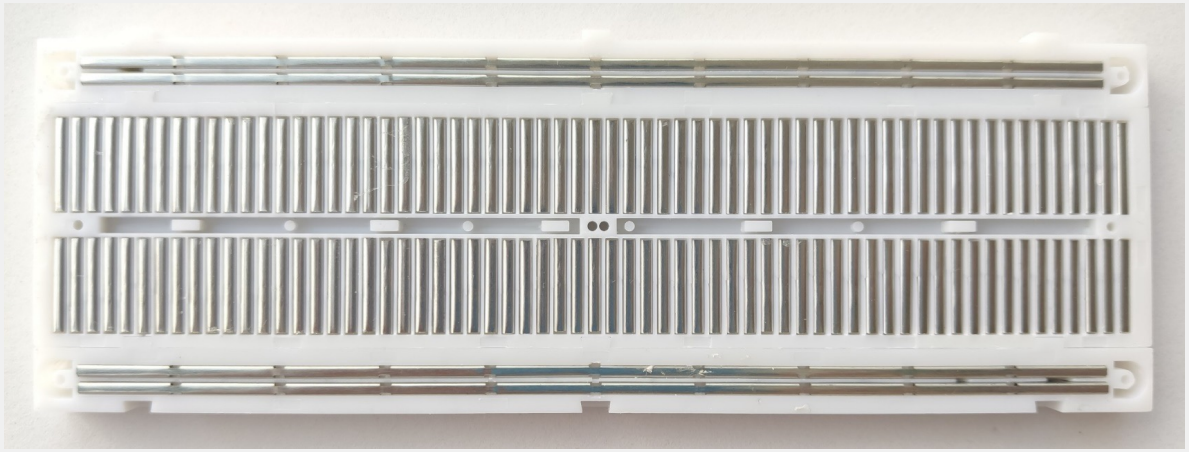

STM32
3V3
R1
1k
PA0
Q1
S8550
+
GND

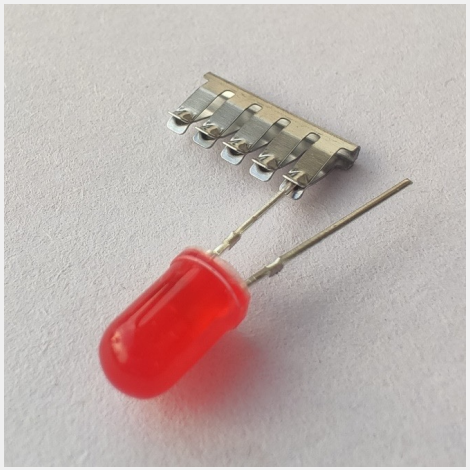

STM32
R1
1k
LED1
PA0
GND


3V3
+
STM32
R1
1k
PA0
Q1
S8050
GND

面包板

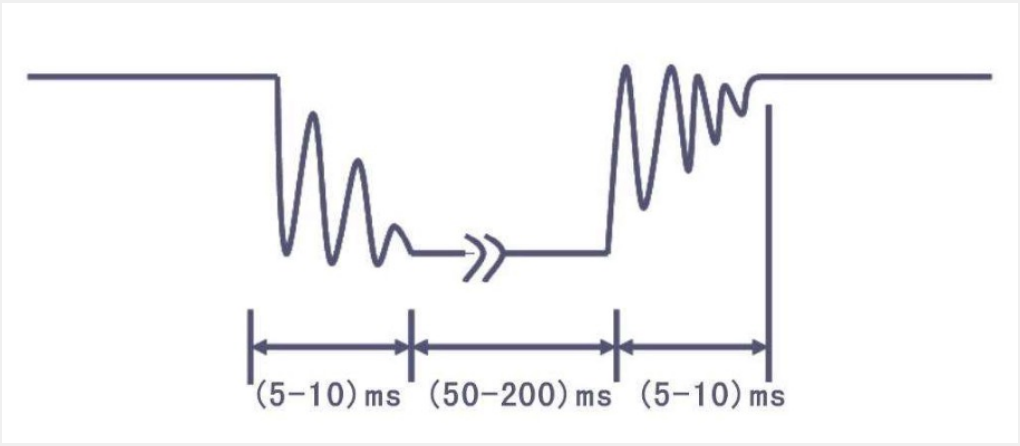






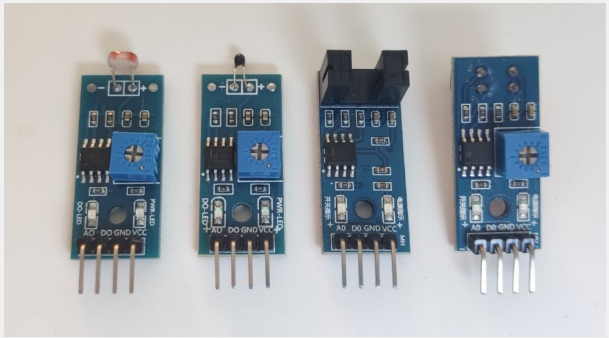
按键简介

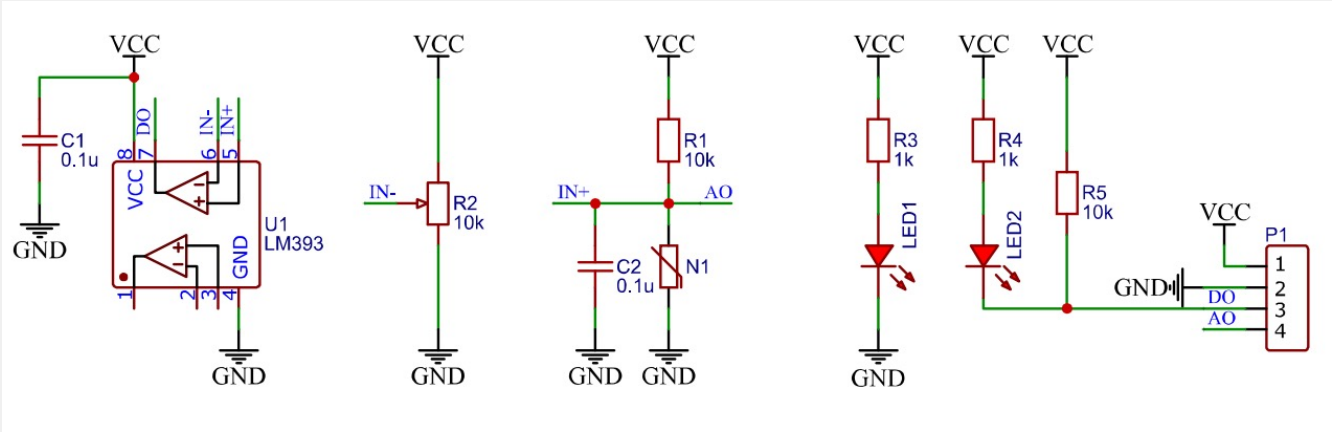
- 按键：常见的输入设备，按下导通，松手断开
- 按键抖动：由于按键内部使用的是机械式弹簧片来进行通断的，所以在按下和松手的瞬间会伴随有一连串的抖动



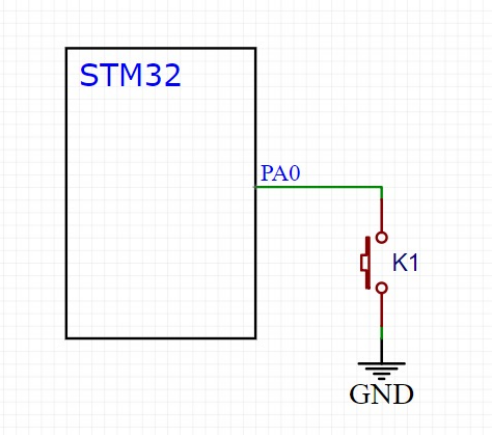
传感器模块简介

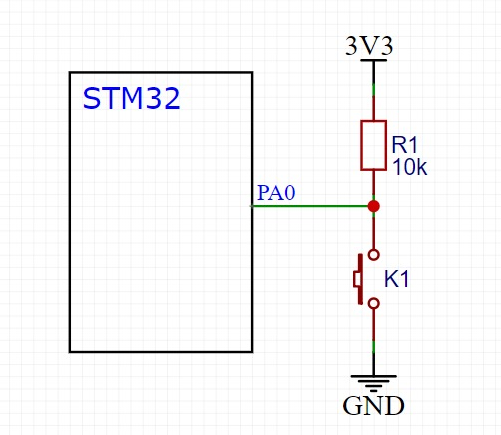
- 传感器模块：传感器元件（光敏电阻/热敏电阻/红外接收管等）的电阻会随外界模拟量的变化而变化，通过与定值电阻分压即可得到模拟电压输出，再通过电压比较器进行二值化即可得到数字电压输出

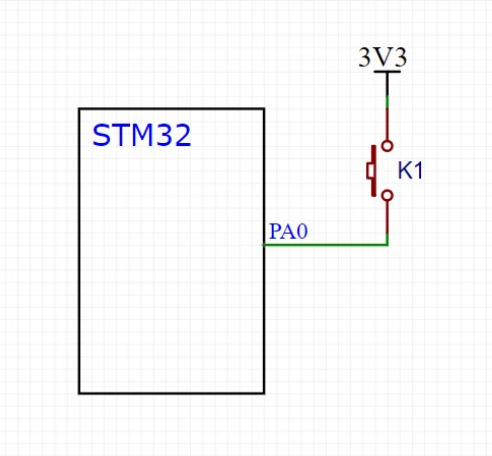


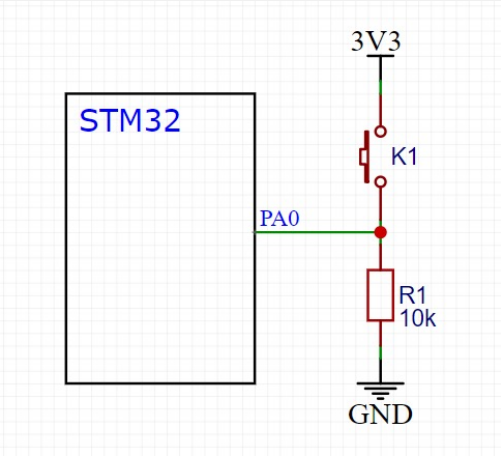


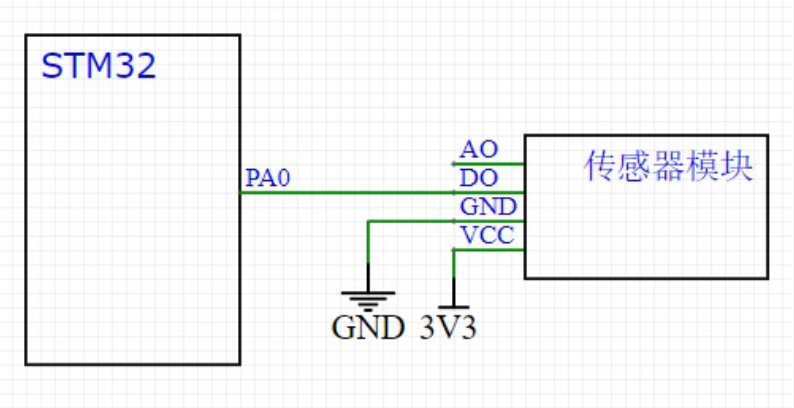
硬件电路


STM32
PA0
K1
GND


3V3
STM32
R1
10k
PA0
K1
GND


3V3
STM32
K1
PA0


3V3
STM32
K1
PA0
R1
10k
GND


STM32
PA0
AO
DO
GND
VCC
传感器模块
GND
3V3

C语言数据类型

关键字	位数	表示范围	stdint关键字	ST关键字
char	8	-128 ~ 127	int8_t	s8
unsigned char	8	0 ~ 255	uint8_t	u8
short	16	-32768 ~ 32767	int16_t	s16
unsigned short	16	0 ~ 65535	uint16_t	u16
int	32	-2147483648 ~ 2147483647	int32_t	s32
unsigned int	32	0 ~ 4294967295	uint32_t	u32
long	32	-2147483648 ~ 2147483647		
unsigned long	32	0 ~ 4294967295		
long long	64	-(2^64)/2 ~ (2^64)/2-1	int64_t	
unsigned long long	64	0 ~ (2^64)-1	uint64_t	
float	32	-3.4e38 ~ 3.4e38		
double	64	-1.7e308 ~ 1.7e308		

C语言宏定义

- 关键字：#define
- 用途：用一个字符串代替一个数字，便于理解，防止出错；提取程序中经常出现的参数，便于快速修改
- 定义宏定义：

```
#define ABC 12345
```

- 引用宏定义：

```
int a = ABC;  //等效于int a = 12345;
```

C语言typedef

- 关键字：typedef
- 用途：将一个比较长的变量类型名换个名字，便于使用
- 定义typedef：

```
typedef unsigned char uint8_t;
```

- 引用typedef：

```
uint8_t a;   //等效于unsigned char a;
```

C语言结构体

- 关键字：struct
- 用途：数据打包，不同类型变量的集合
- 定义结构体变量：

```
struct{char x; int y; float z;} StructName;
```

因为结构体变量类型较长，所以通常用typedef更改变量类型名

- 引用结构体成员：

```
StructName.x = 'A';
StructName.y = 66;
StructName.z = 1.23;
```

或

```
pStructName->x = 'A';   //pStructName为结构体的地址
pStructName->y = 66;
pStructName->z = 1.23;
```

C语言枚举

- 关键字：enum
- 用途：定义一个取值受限制的整型变量，用于限制变量取值范围；宏定义的集合
- 定义枚举变量：

  ```
  enum{FALSE = 0, TRUE = 1} EnumName;
  ```

  因为枚举变量类型较长，所以通常用typedef更改变量类型名
- 引用枚举成员：

  ```
  EnumName = FALSE;
  EnumName = TRUE;
  ```

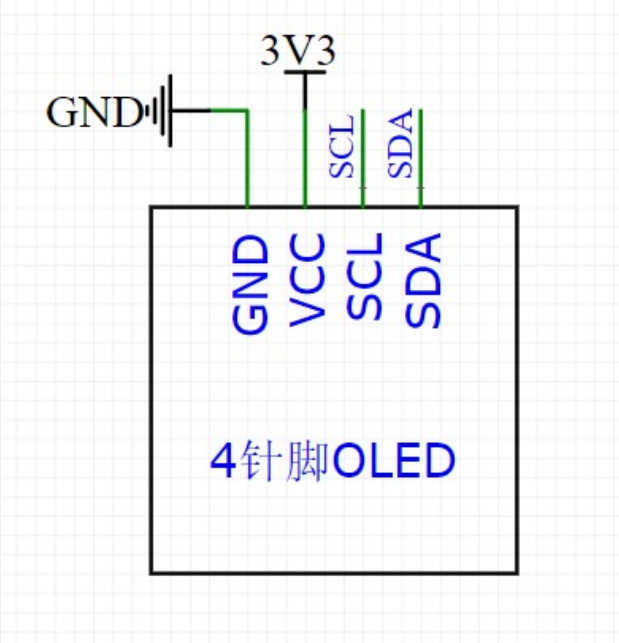
调试方式

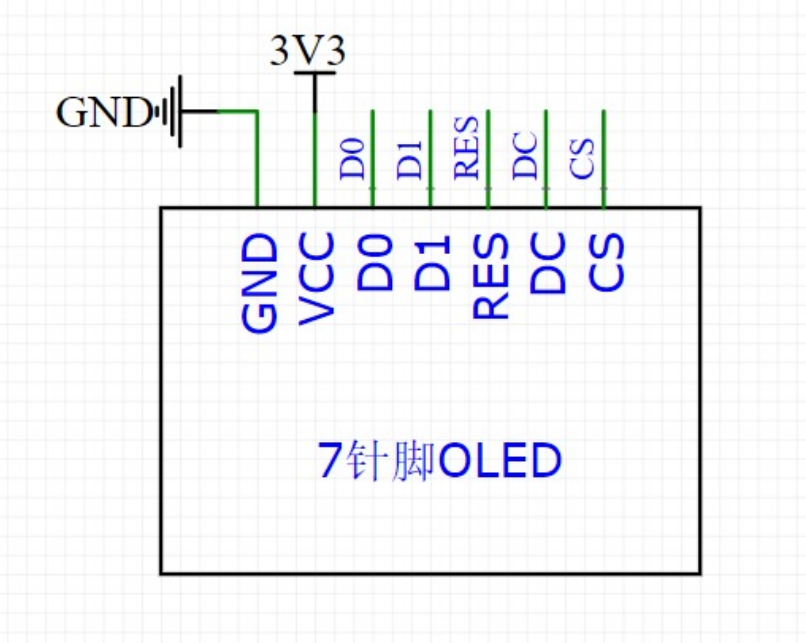
- 串口调试：通过串口通信，将调试信息发送到电脑端，电脑使用串口助手显示调试信息
- 显示屏调试：直接将显示屏连接到单片机，将调试信息打印在显示屏上
- Keil调试模式：借助Keil软件的调试模式，可使用单步运行、设置断点、查看寄存器及变量等功能

OLED简介

- OLED（Organic Light Emitting Diode）：有机发光二极管
- OLED显示屏：性能优异的新型显示屏，具有功耗低、相应速度快、宽视角、轻薄柔韧等特点
- 0.96寸OLED模块：小巧玲珑、占用接口少、简单易用，是电子设计中非常常见的显示屏模块
- 供电：3~5.5V，通信协议：I2C/SPI，分辨率：128*64

硬件电路


3V3
GND
SCL
SDA
GND
VCC
SCL
SDA
4针脚OLED


3V3
GND
D0
D1
RES
DC
CS
GND
VCC
D0
D1
RES
DC
CS
7针脚OLED

OLED驱动函数

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	A		H	e	l	l	o	W	o	r	l	d	!			
2	1	2	3	4	5		-	6	6							
3	A	A	5	5												
4	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1

函数	作用
OLED_Init();	初始化
OLED_Clear();	清屏
OLED_ShowChar(1, 1, 'A');	显示一个字符
OLED_ShowString(1, 3, "HelloWorld!");	显示字符串
OLED_ShowNum(2, 1, 12345, 5);	显示十进制数字
OLED_ShowSignedNum(2, 7, -66, 2);	显示有符号十进制数字
OLED_ShowHexNum(3, 1, 0xAA55, 4);	显示十六进制数字
OLED_ShowBinNum(4, 1, 0xAA55, 16);	显示二进制数字

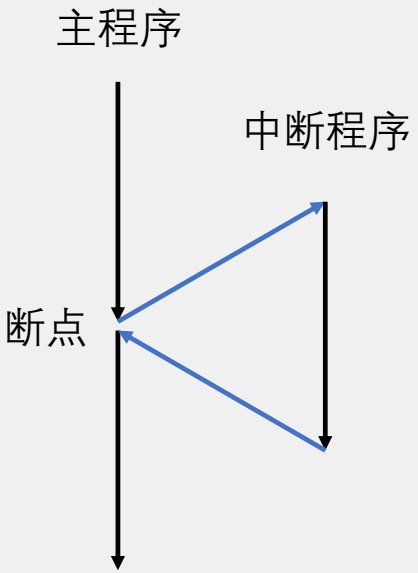
中断系统

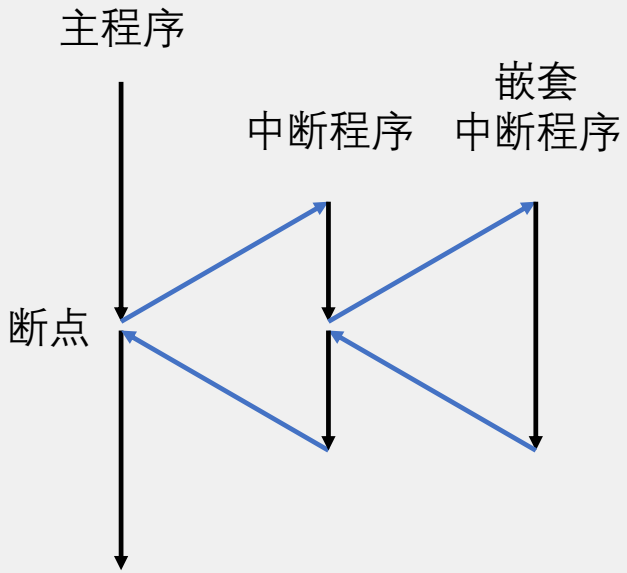
- 中断：在主程序运行过程中，出现了特定的中断触发条件（中断源），使得CPU暂停当前正在运行的程序，转而去处理中断程序，处理完成后又返回原来被暂停的位置继续运行

- 中断优先级：当有多个中断源同时申请中断时，CPU会根据中断源的轻重缓急进行裁决，优先响应更加紧急的中断源

- 中断嵌套：当一个中断程序正在运行时，又有新的更高优先级的中断源申请中断，CPU再次暂停当前中断程序，转而去处理新的中断程序，处理完成后依次进行返回

中断执行流程






```
1
2  int main(void)
3  {
4      while (1)
5      {
6          //主程序
7          //...
8          //主程序
9      }
10 }
11
12 void EXTI0_IRQHandler(void)
13 {
14     //中断程序
15     //...
16     //中断程序
17 }
18
```

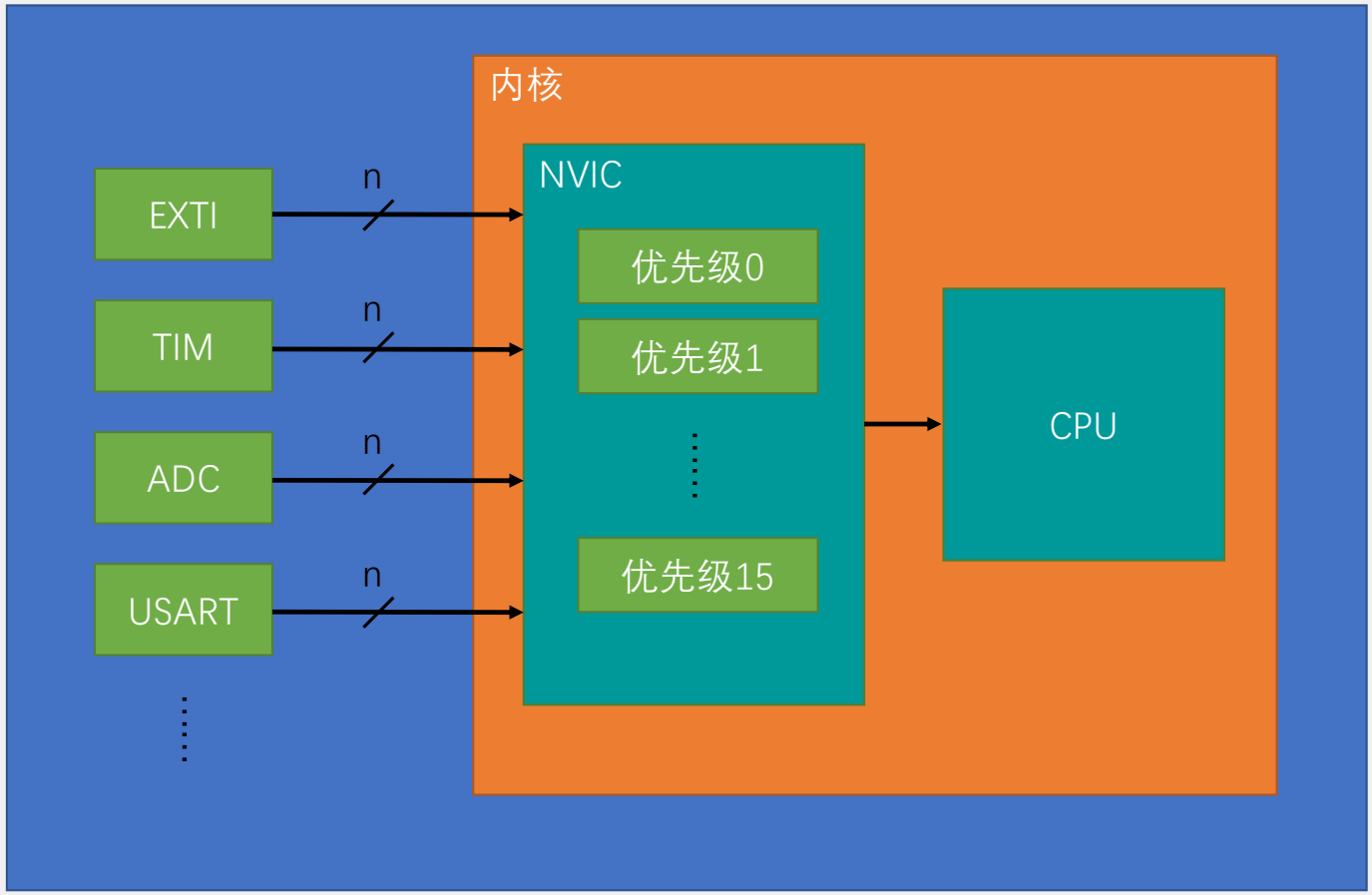
STM32中断

- 68个可屏蔽中断通道，包含EXTI、TIM、ADC、USART、SPI、I2C、RTC等多个外设
- 使用NVIC统一管理中断，每个中断通道都拥有16个可编程的优先等级，可对优先级进行分组，进一步设置抢占优先级和响应优先级

表55　其它STM32F10xxx产品(小容量、中容量和大容量)的向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000_0000
	-3	固定	Reset	复位	0x0000_0004
	-2	固定	NMI	不可屏蔽中断 RCC时钟安全系统(CSS)联接到NMI向量	0x0000_0008
	-1	固定	硬件失效(HardFault)	所有类型的失效	0x0000_000C
	0	可设置	存储管理(MemManage)	存储器管理	0x0000_0010
	1	可设置	总线错误(BusFault)	预取指失败，存储器访问失败	0x0000_0014
	2	可设置	错误应用(UsageFault)	未定义的指令或非法状态	0x0000_0018
	-	-	-	保留	0x0000_001C ~0x0000_002B
	3	可设置	SVCall	通过SWI指令的系统服务调用	0x0000_002C
	4	可设置	调试监控(DebugMonitor)	调试监控器	0x0000_0030
	-	-	-	保留	0x0000_0034
	5	可设置	PendSV	可挂起的系统服务	0x0000_0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000_003C
0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	连到EXTI的电源电压检测(PVD)中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048
3	10	可设置	RTC	实时时钟(RTC)全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	RCC	复位和时钟控制(RCC)中断	0x0000_0054
6	13	可设置	EXTI0	EXTI线0中断	0x0000_0058
7	14	可设置	EXTI1	EXTI线1中断	0x0000_005C
8	15	可设置	EXTI2	EXTI线2中断	0x0000_0060
9	16	可设置	EXTI3	EXTI线3中断	0x0000_0064
10	17	可设置	EXTI4	EXTI线4中断	0x0000_0068
11	18	可设置	DMA1通道1	DMA1通道1全局中断	0x0000_006C
12	19	可设置	DMA1通道2	DMA1通道2全局中断	0x0000_0070
13	20	可设置	DMA1通道3	DMA1通道3全局中断	0x0000_0074
14	21	可设置	DMA1通道4	DMA1通道4全局中断	0x0000_0078
15	22	可设置	DMA1通道5	DMA1通道5全局中断	0x0000_007C
16	23	可设置	DMA1通道6	DMA1通道6全局中断	0x0000_0080
17	24	可设置	DMA1通道7	DMA1通道7全局中断	0x0000_0084
18	25	可设置	ADC1_2	ADC1和ADC2的全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN_TX	USB高优先级或CAN发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN_RX0	USB低优先级或CAN接收0中断	0x0000_0090
21	28	可设置	CAN_RX1	CAN接收1中断	0x0000_0094
22	29	可设置	CAN_SCE	CAN SCE中断	0x0000_0098
23	30	可设置	EXTI9_5	EXTI线[9:5]中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1刹车中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1捕获比较中断	0x0000_00AC
28	35	可设置	TIM2	TIM2全局中断	0x0000_00B0
29	36	可设置	TIM3	TIM3全局中断	0x0000_00B4
30	37	可设置	TIM4	TIM4全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I^2C1事件中断	0x0000_00BC
32	39	可设置	I2C1_ER	I^2C1错误中断	0x0000_00C0
33	40	可设置	I2C2_EV	I^2C2事件中断	0x0000_00C4
34	41	可设置	I2C2_ER	I^2C2错误中断	0x0000_00C8
35	42	可设置	SPI1	SPI1全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2全局中断	0x0000_00D0
37	44	可设置	USART1	USART1全局中断	0x0000_00D4
38	45	可设置	USART2	USART2全局中断	0x0000_00D8
39	46	可设置	USART3	USART3全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI线[15:10]中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到EXTI的RTC闹钟中断	0x0000_00E4
42	49	可设置	USB唤醒	连到EXTI的从USB待机唤醒中断	0x0000_00E8
43	50	可设置	TIM8_BRK	TIM8刹车中断	0x0000_00EC
44	51	可设置	TIM8_UP	TIM8更新中断	0x0000_00F0
45	52	可设置	TIM8_TRG_COM	TIM8触发和通信中断	0x0000_00F4
46	53	可设置	TIM8_CC	TIM8捕获比较中断	0x0000_00F8
47	54	可设置	ADC3	ADC3全局中断	0x0000_00FC
48	55	可设置	FSMC	FSMC全局中断	0x0000_0100
49	56	可设置	SDIO	SDIO全局中断	0x0000_0104
50	57	可设置	TIM5	TIM5全局中断	0x0000_0108
51	58	可设置	SPI3	SPI3全局中断	0x0000_010C
52	59	可设置	UART4	UART4全局中断	0x0000_0110
53	60	可设置	UART5	UART5全局中断	0x0000_0114
54	61	可设置	TIM6	TIM6全局中断	0x0000_0118
55	62	可设置	TIM7	TIM7全局中断	0x0000_011C
56	63	可设置	DMA2通道1	DMA2通道1全局中断	0x0000_0120
57	64	可设置	DMA2通道2	DMA2通道2全局中断	0x0000_0124
58	65	可设置	DMA2通道3	DMA2通道3全局中断	0x0000_0128
59	66	可设置	DMA2通道4_5	DMA2通道4和DMA2通道5全局中断	0x0000_012C

NVIC基本结构



NVIC优先级分组

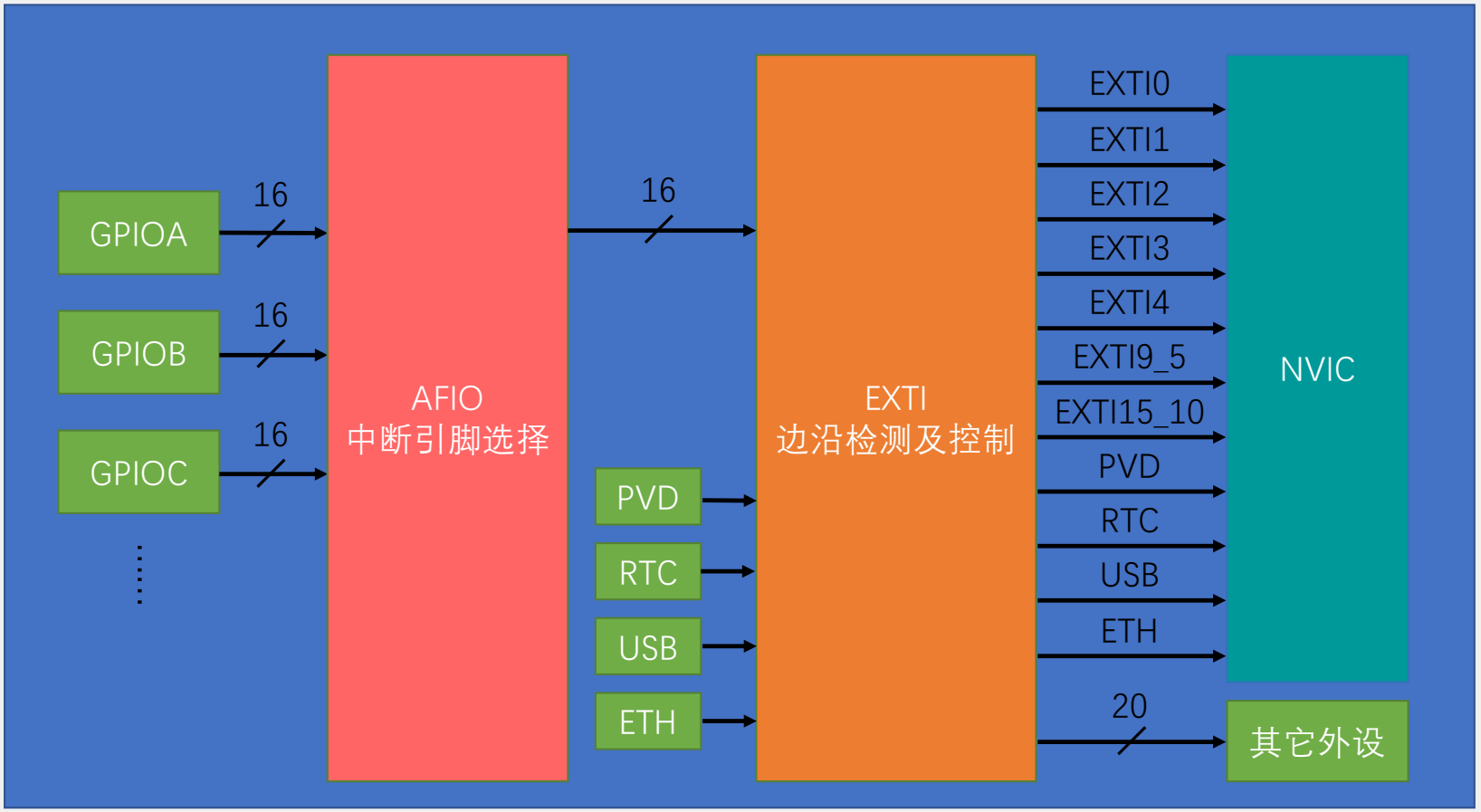
- NVIC的中断优先级由优先级寄存器的4位（0~15）决定，这4位可以进行切分，分为高n位的抢占优先级和低4-n位的响应优先级
- 抢占优先级高的可以中断嵌套，响应优先级高的可以优先排队，抢占优先级和响应优先级均相同的按中断号排队

分组方式	抢占优先级	响应优先级
分组0	0位，取值为0	4位，取值为0~15
分组1	1位，取值为0~1	3位，取值为0~7
分组2	2位，取值为0~3	2位，取值为0~3
分组3	3位，取值为0~7	1位，取值为0~1
分组4	4位，取值为0~15	0位，取值为0

EXTI简介

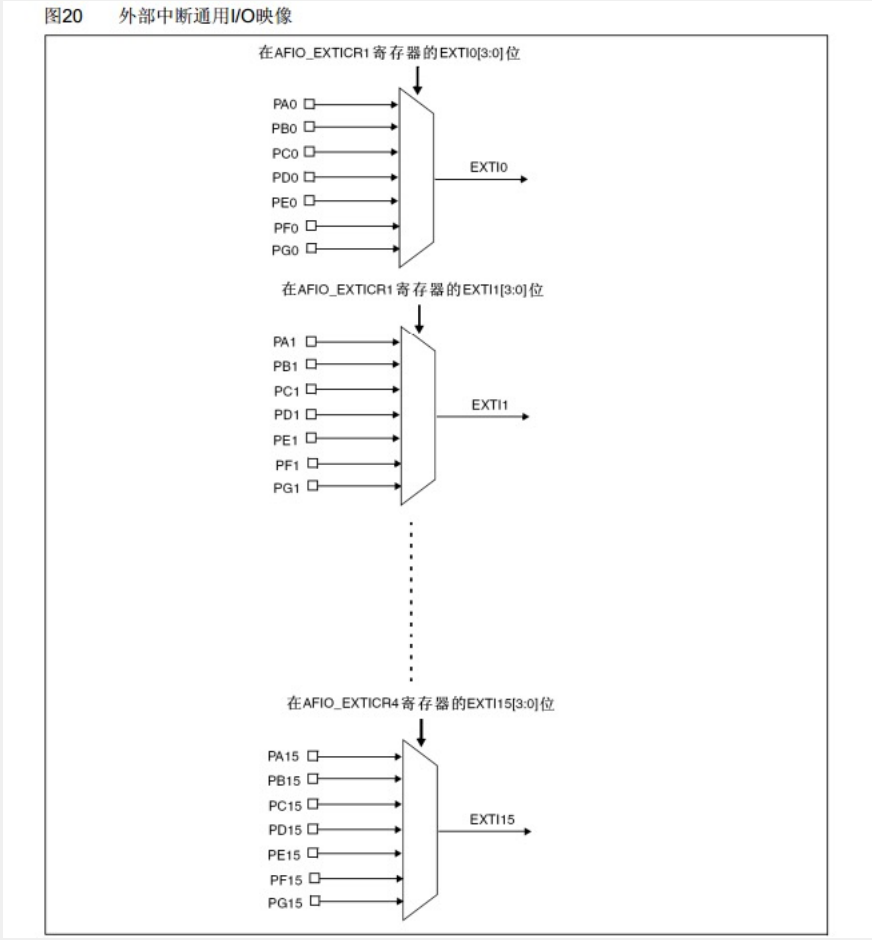
- EXTI（Extern Interrupt）外部中断
- EXTI可以监测指定GPIO口的电平信号，当其指定的GPIO口产生电平变化时，EXTI将立即向NVIC发出中断申请，经过NVIC裁决后即可中断CPU主程序，使CPU执行EXTI对应的中断程序
- 支持的触发方式：上升沿/下降沿/双边沿/软件触发
- 支持的GPIO口：所有GPIO口，但相同的Pin不能同时触发中断
- 通道数：16个GPIO_Pin，外加PVD输出、RTC闹钟、USB唤醒、以太网唤醒
- 触发响应方式：中断响应/事件响应

EXTI基本结构


GPIOA
GPIOB
GPIOC
16
16
16
AFIO
中断引脚选择
16
PVD
RTC
USB
ETH
EXTI
边沿检测及控制
EXTI0
EXTI1
EXTI2
EXTI3
EXTI4
EXTI9_5
EXTI15_10
PVD
RTC
USB
ETH
NVIC
20
其它外设

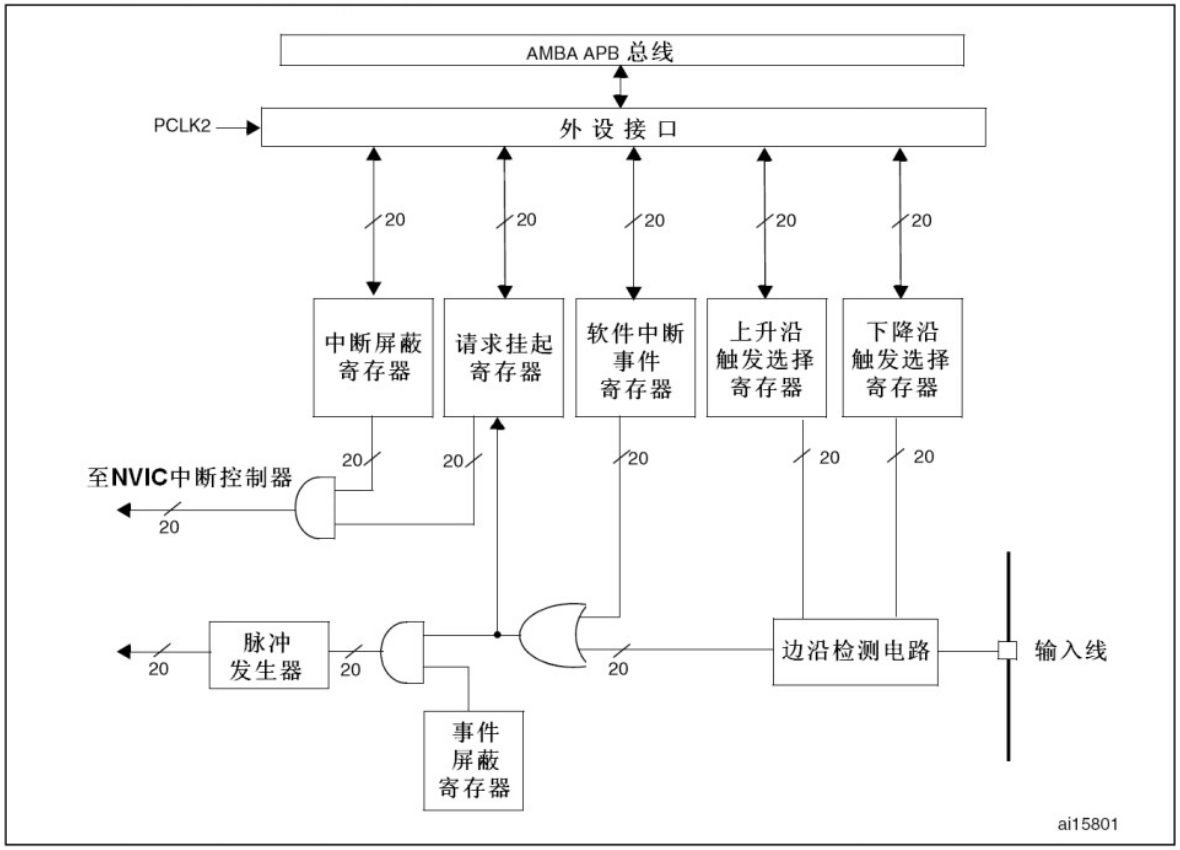
AFIO复用IO口

- AFIO主要用于引脚复用功能的选择和重定义
- 在STM32中，AFIO主要完成两个任务：复用功能引脚重映射、中断引脚选择



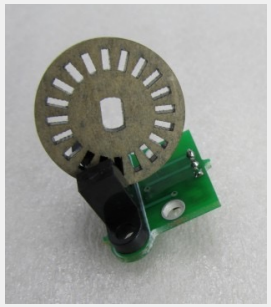
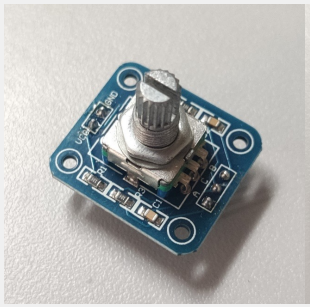
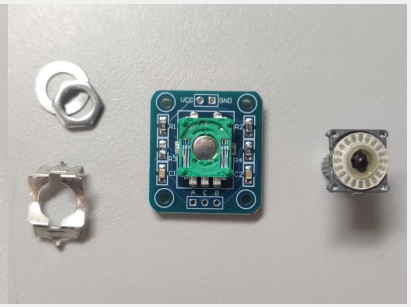
EXTI框图

图19　外部中断/事件控制器框图

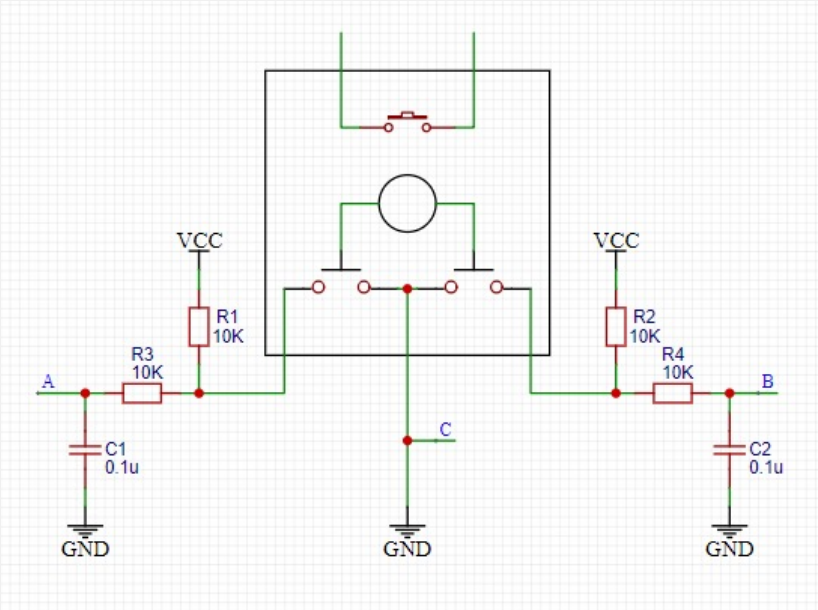


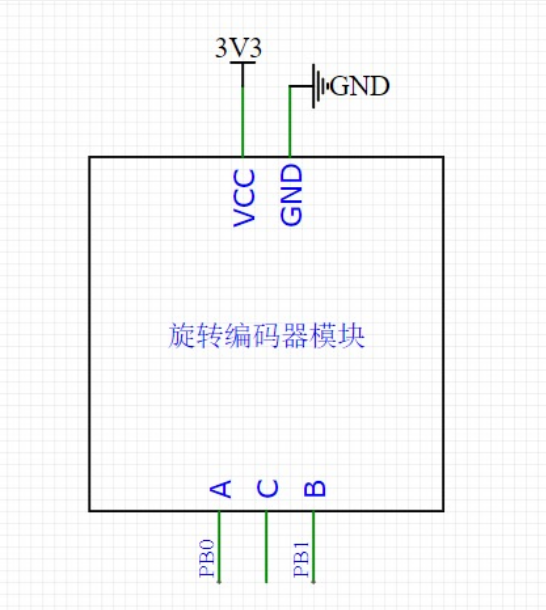
旋转编码器简介

- 旋转编码器：用来测量位置、速度或旋转方向的装置，当其旋转轴旋转时，其输出端可以输出与旋转速度和方向对应的方波信号，读取方波信号的频率和相位信息即可得知旋转轴的速度和方向
- 类型：机械触点式/霍尔传感器式/光栅式

硬件电路


VCC
R1
10K
R3
10K
A
C1
0.1u
GND
C
GND
VCC
R2
10K
R4
10K
B
C2
0.1u
GND


3V3
GND
VCC
GND
旋转编码器模块
A
C
B
PB0
PB1

TIM简介

- TIM（Timer）定时器
- 定时器可以对输入的时钟进行计数，并在计数值达到设定值时触发中断
- 16位计数器、预分频器、自动重装寄存器的时基单元，在72MHz计数时钟下可以实现最大59.65s的定时
- 不仅具备基本的定时中断功能，而且还包含内外时钟源选择、输入捕获、输出比较、编码器接口、主从触发模式等多种功能
- 根据复杂度和应用场景分为了高级定时器、通用定时器、基本定时器三种类型

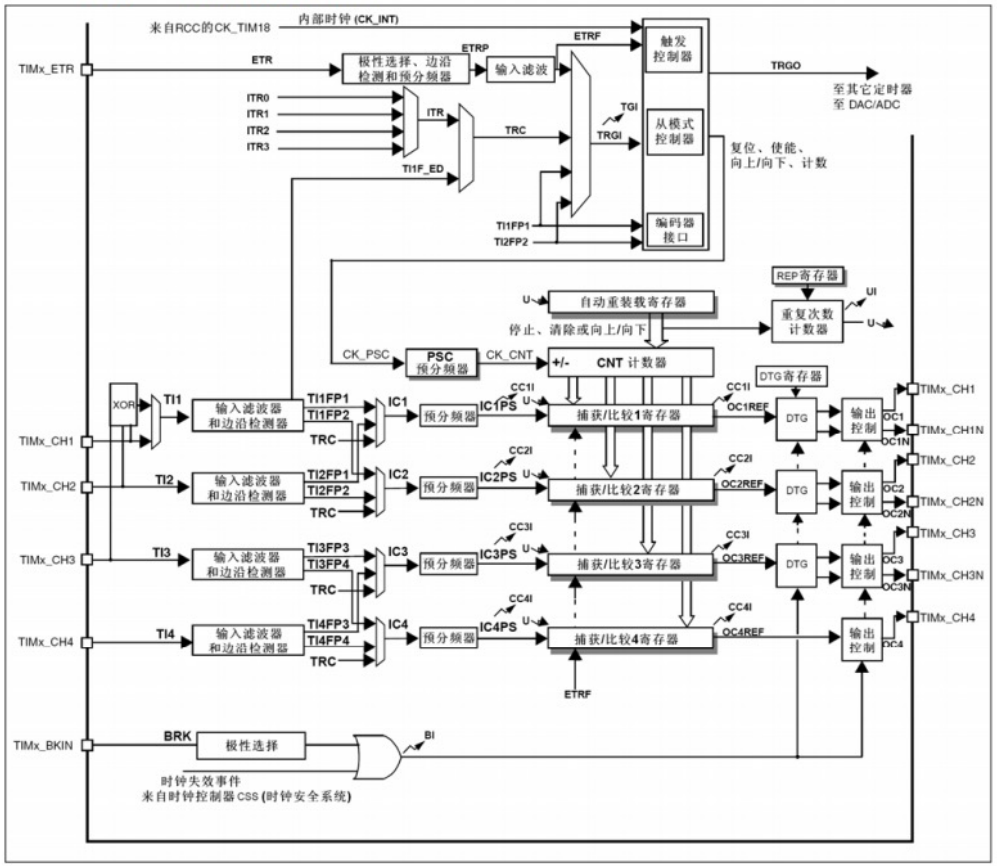
定时器类型

类型	编号	总线	功能
高级定时器	TIM1、TIM8	APB2	拥有通用定时器全部功能，并额外具有重复计数器、死区生成、互补输出、刹车输入等功能
通用定时器	TIM2、TIM3、TIM4、TIM5	APB1	拥有基本定时器全部功能，并额外具有内外时钟源选择、输入捕获、输出比较、编码器接口、主从触发模式等功能
基本定时器	TIM6、TIM7	APB1	拥有定时中断、主模式触发DAC的功能

- STM32F103C8T6定时器资源：TIM1、TIM2、TIM3、TIM4

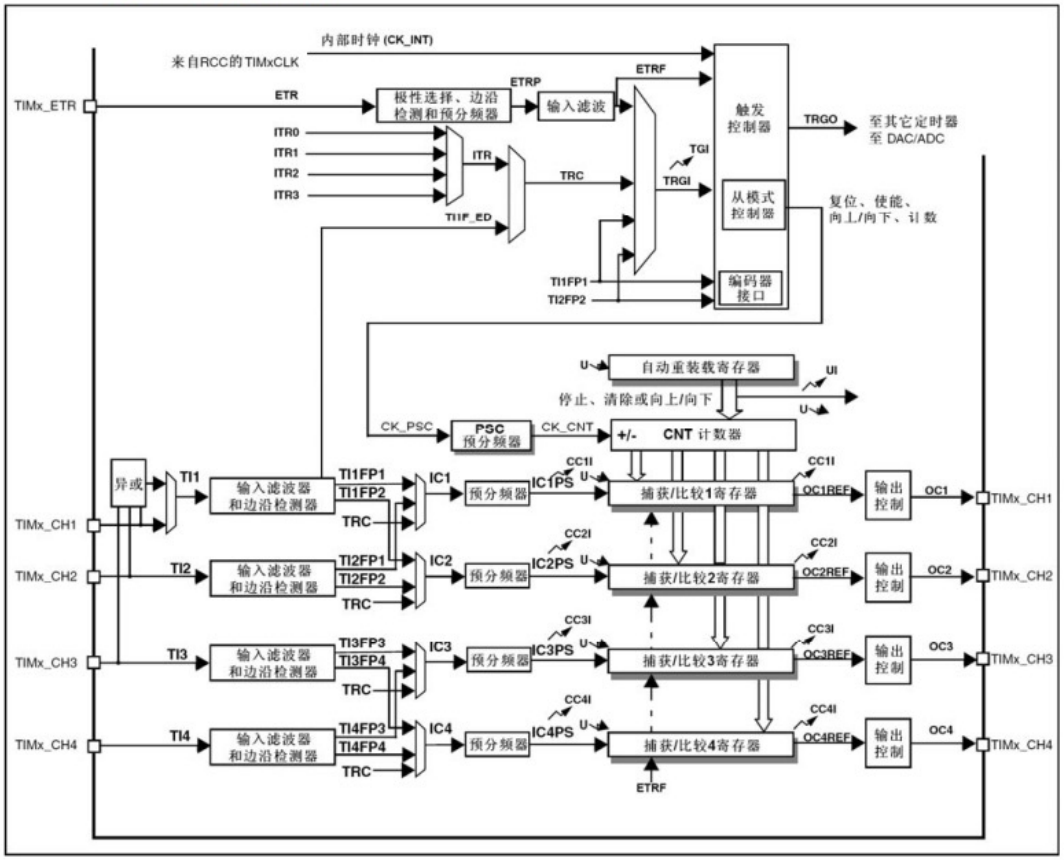
高级定时器

图50　高级控制定时器框图



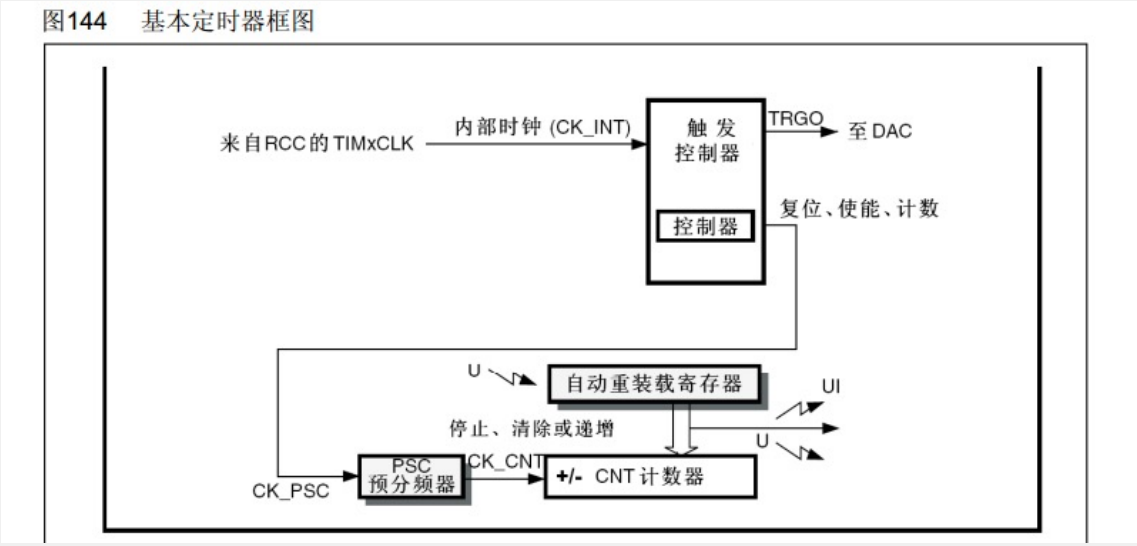
通用定时器

图98　通用定时器框图

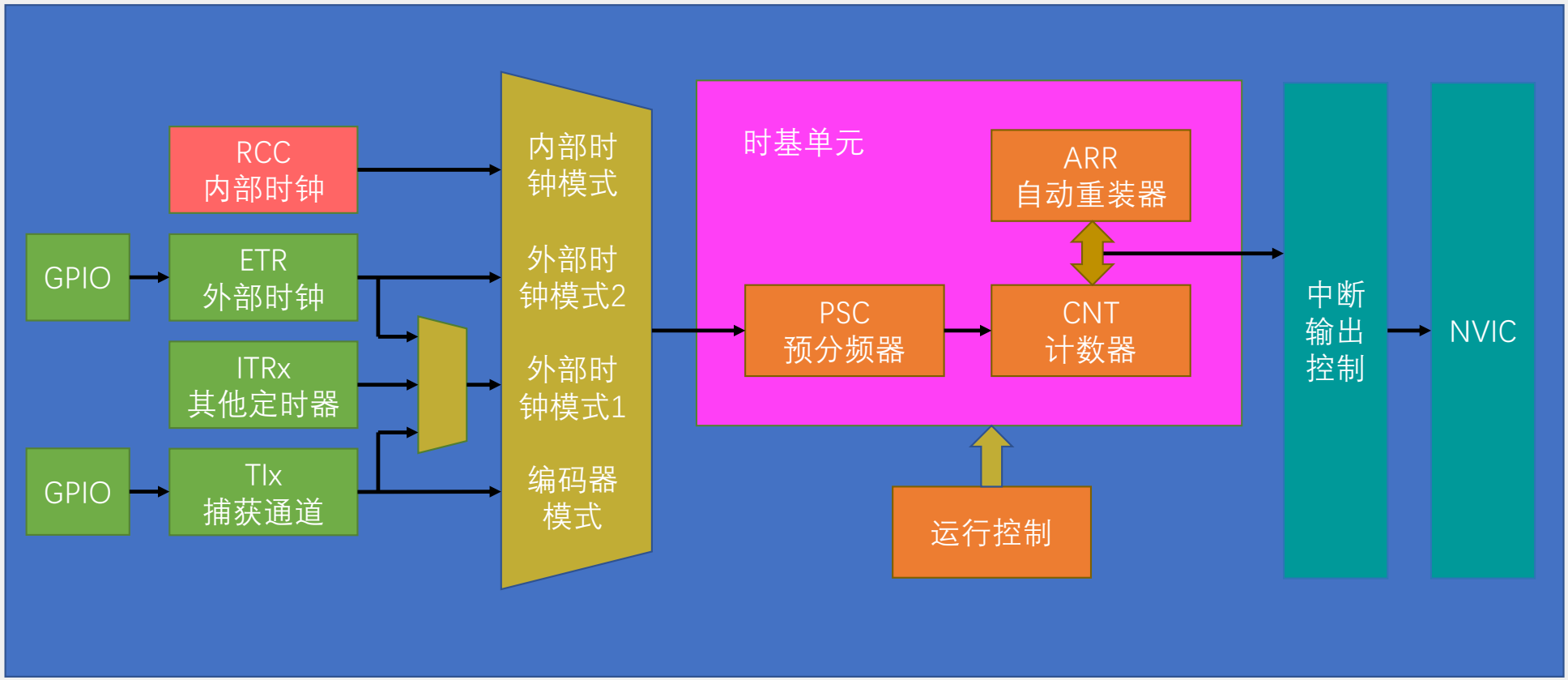


基本定时器

图144　基本定时器框图

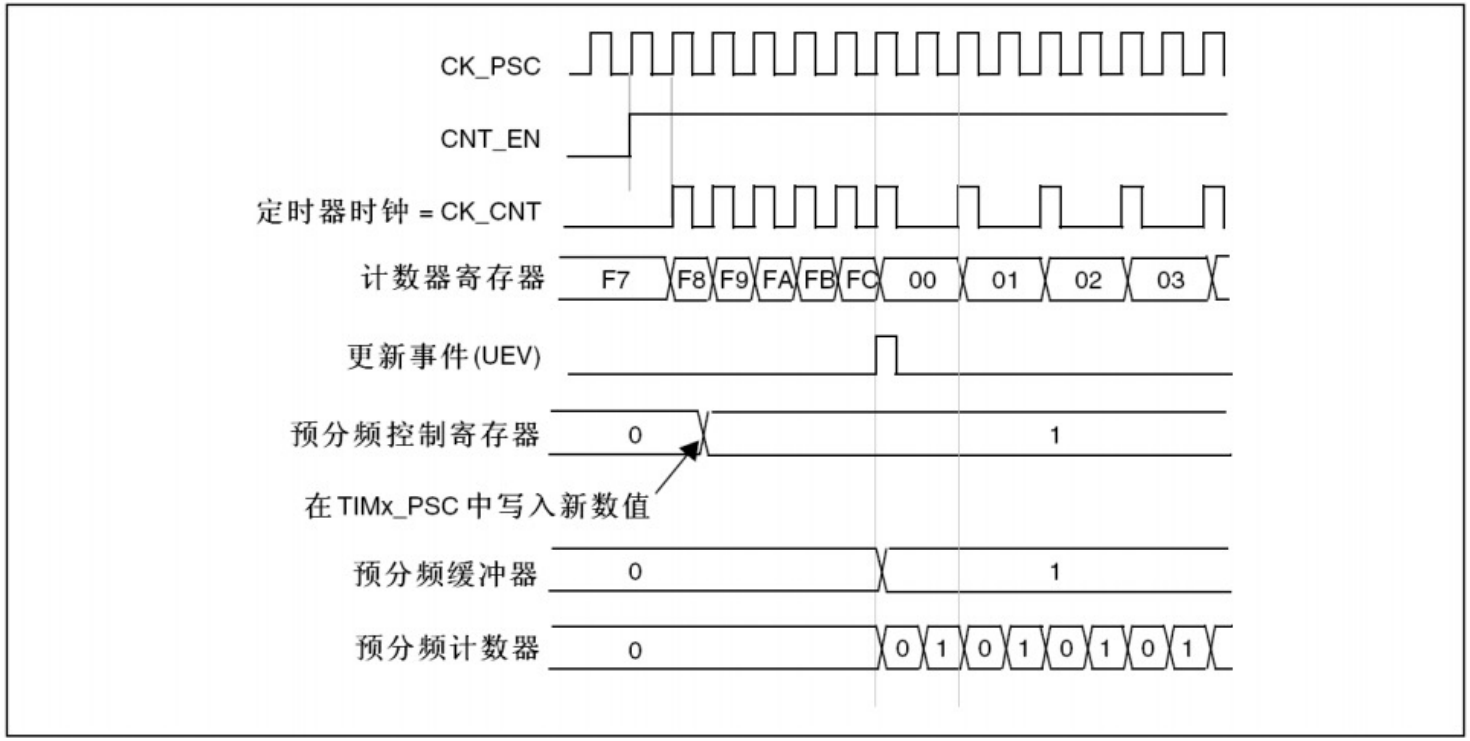


定时中断基本结构


RCC
内部时钟
GPIO
ETR
外部时钟
ITRx
其他定时器
GPIO
TIx
捕获通道
内部时钟模式
外部时钟模式2
外部时钟模式1
编码器模式
时基单元
ARR
自动重装器
PSC
预分频器
CNT
计数器
运行控制
中断输出控制
NVIC

预分频器时序

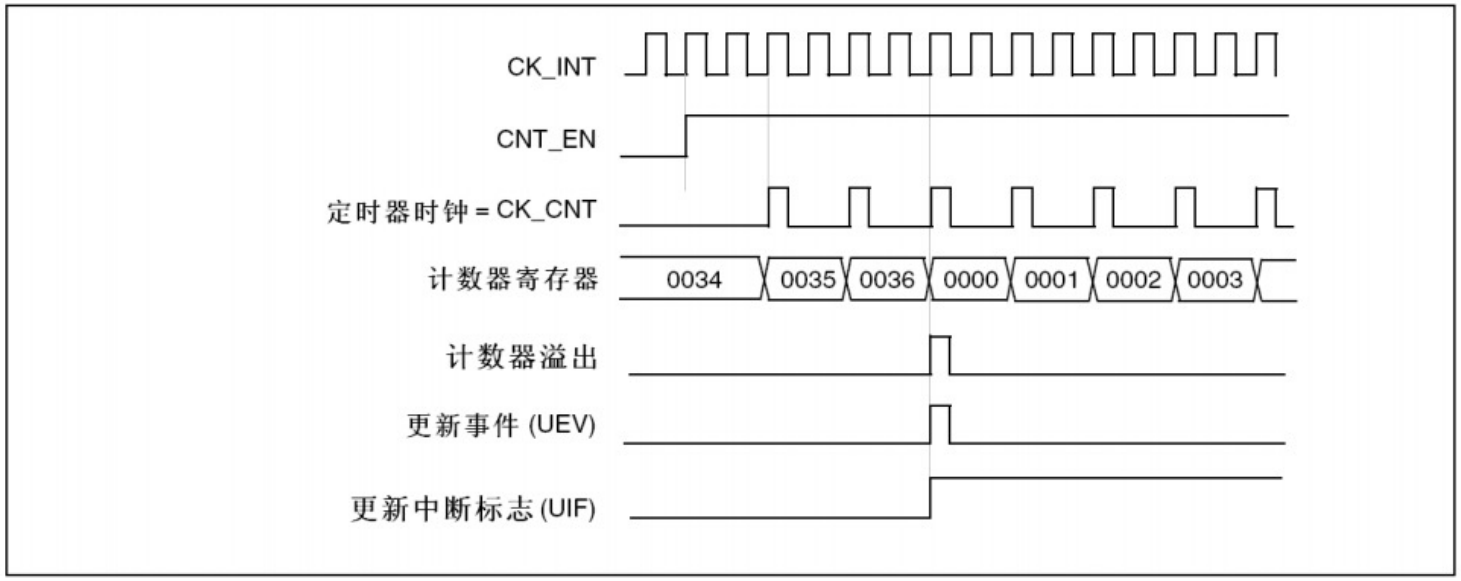
图99　当预分频器的参数从1变到2时，计数器的时序图



- 计数器计数频率：CK_CNT = CK_PSC / (PSC + 1)

计数器时序

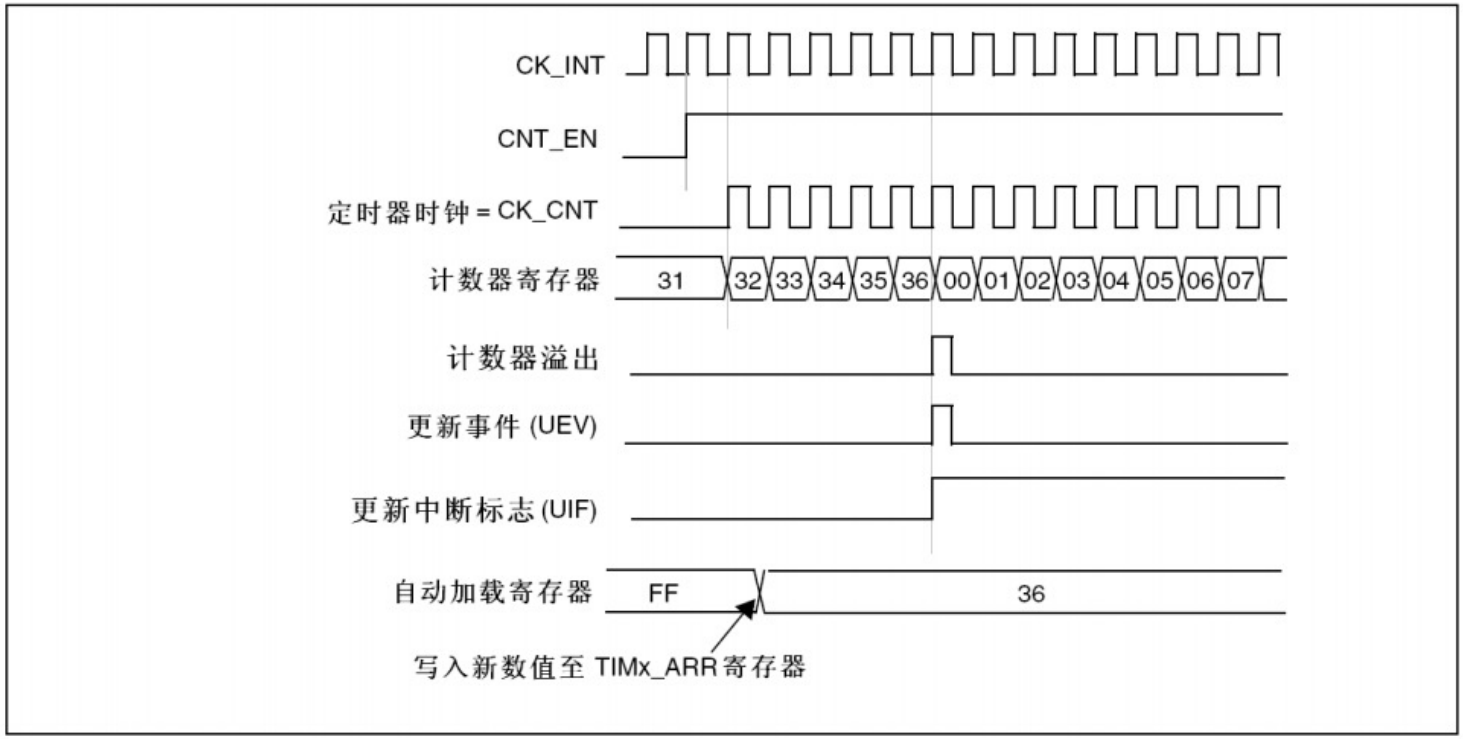
图102　计数器时序图，内部时钟分频因子为2



- 计数器溢出频率：CK_CNT_OV = CK_CNT / (ARR + 1)
  = CK_PSC / (PSC + 1) / (ARR + 1)

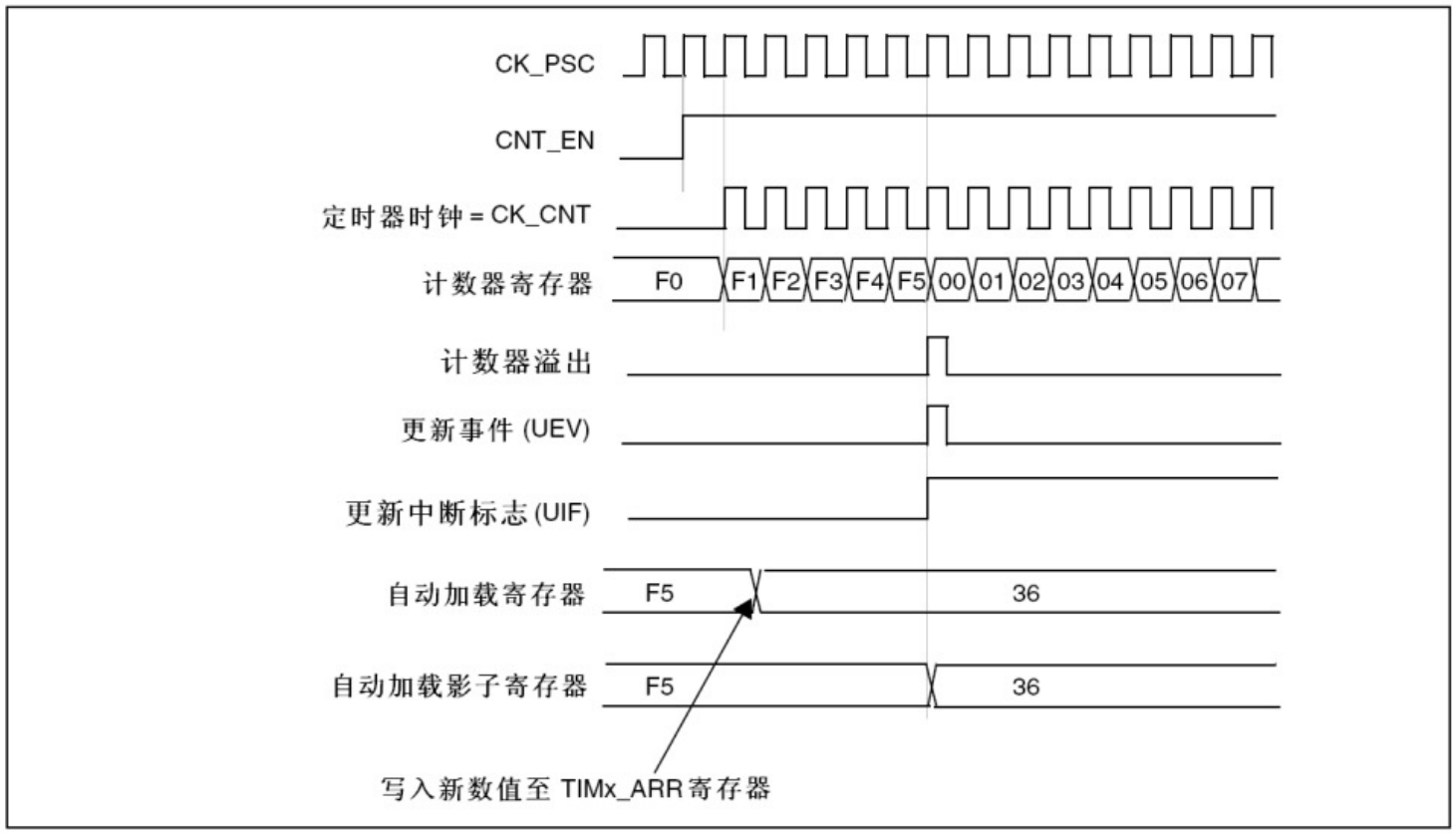
计数器无预装时序

图105　计数器时序图，当ARPE=0时的更新事件(TIMx_ARR没有预装入)



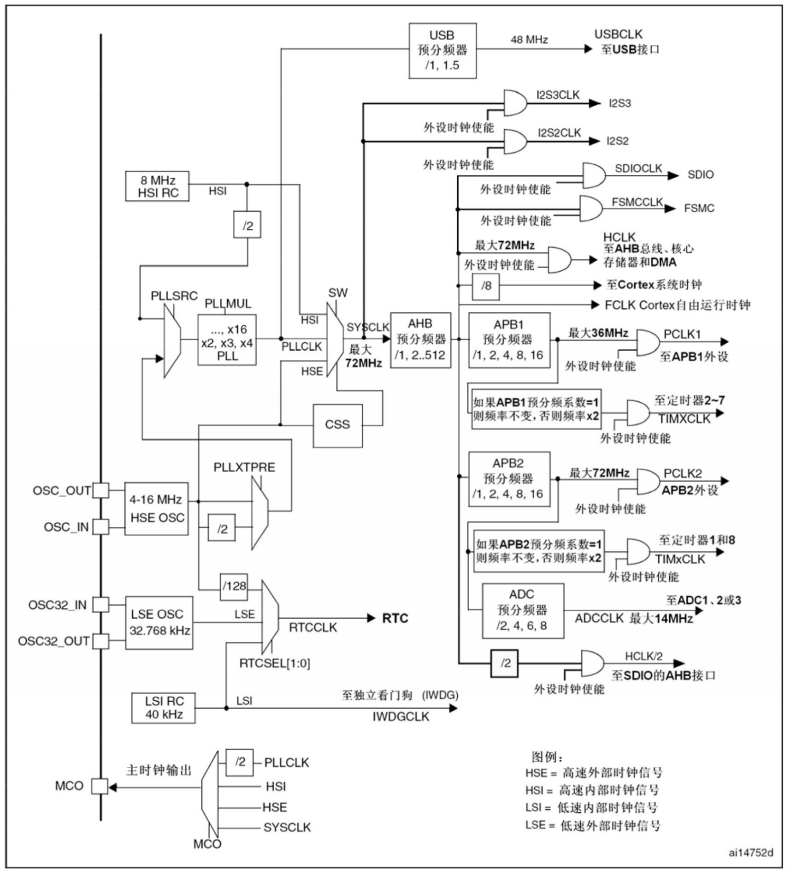
计数器有预装时序

图106　计数器时序图，当ARPE=1时的更新事件(预装入了TIMx_ARR)



RCC时钟树

图8　　时钟树



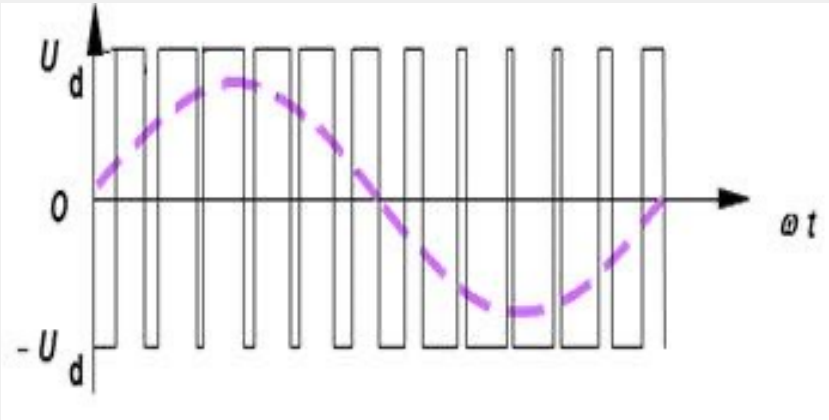
输出比较简介

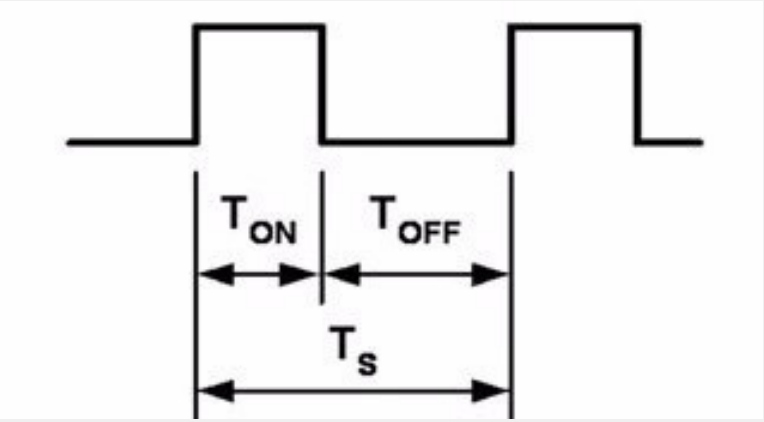
- OC（Output Compare）输出比较
- 输出比较可以通过比较CNT与CCR寄存器值的关系，来对输出电平进行置1、置0或翻转的操作，用于输出一定频率和占空比的PWM波形
- 每个高级定时器和通用定时器都拥有4个输出比较通道
- 高级定时器的前3个通道额外拥有死区生成和互补输出的功能

PWM简介

- PWM（Pulse Width Modulation）脉冲宽度调制
- 在具有惯性的系统中，可以通过对一系列脉冲的宽度进行调制，来等效地获得所需要的模拟参量，常应用于电机控速等领域
- PWM参数：

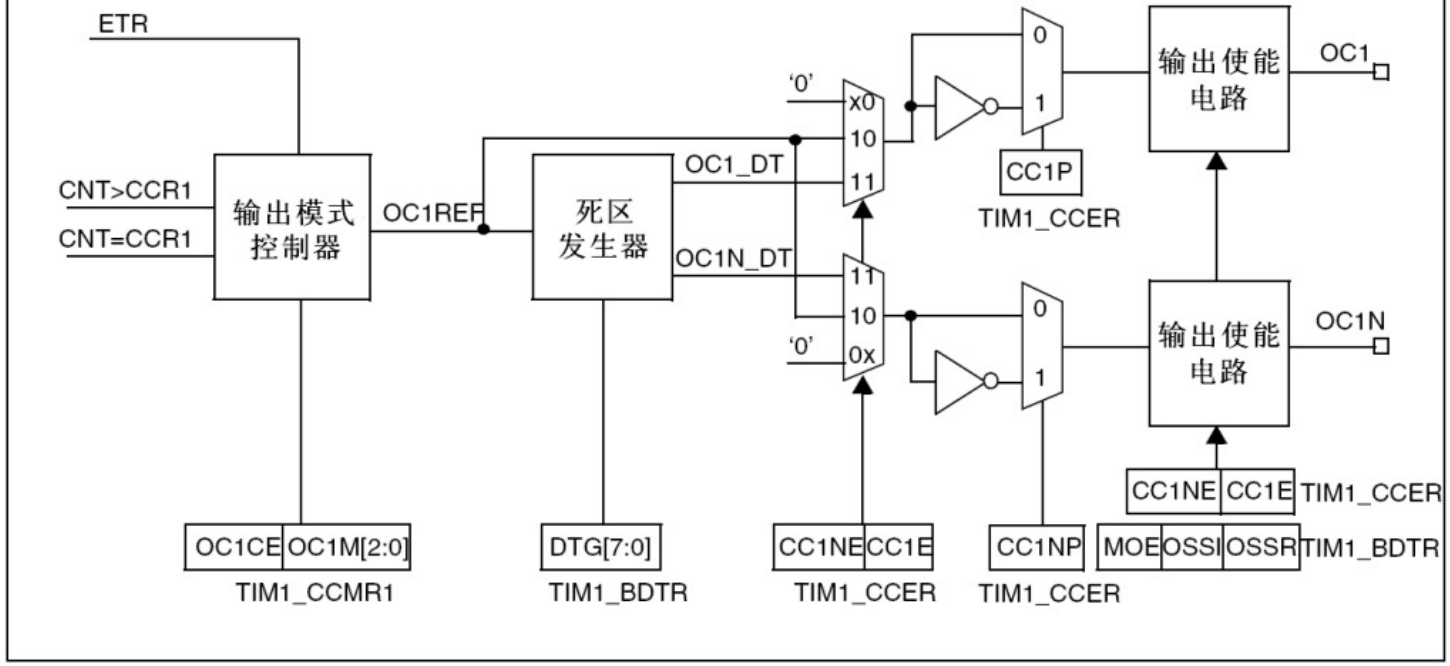
  频率 = 1 / T_S　　占空比 = T_{ON} / T_S　　分辨率 = 占空比变化步距





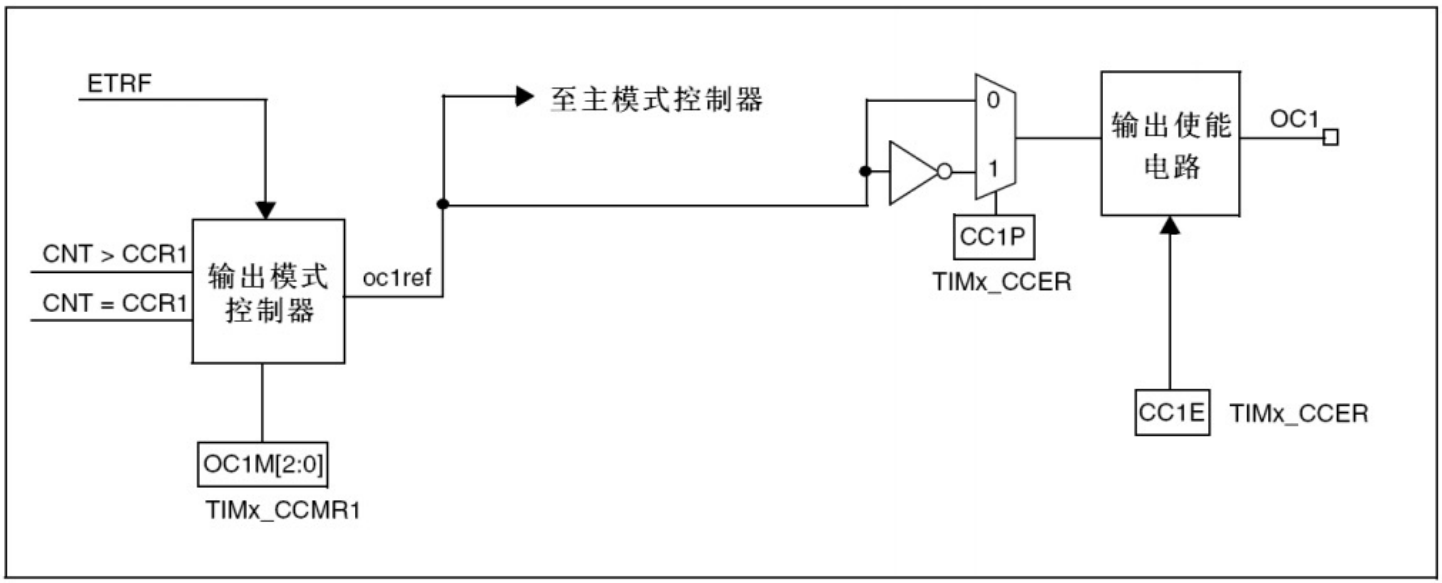
输出比较通道(高级)

图78　捕获/比较通道的输出部分(通道1至3)



输出比较通道(通用)

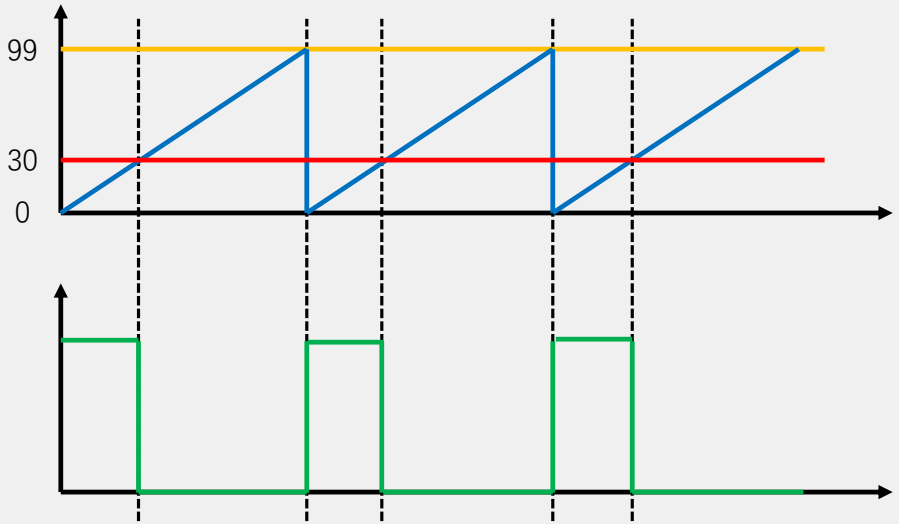
图125　捕获/比较通道的输出部分(通道1)



输出比较模式

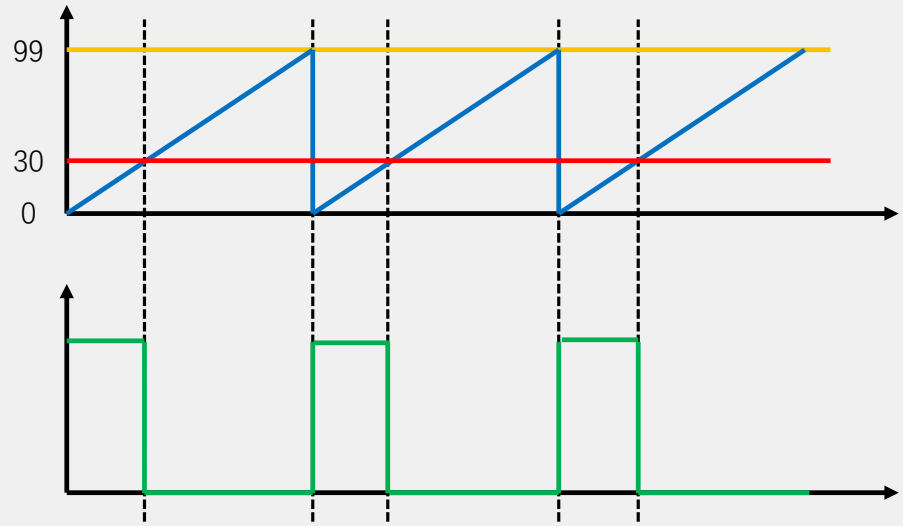
模式	描述
冻结	CNT=CCR时，REF保持为原状态
匹配时置有效电平	CNT=CCR时，REF置有效电平
匹配时置无效电平	CNT=CCR时，REF置无效电平
匹配时电平翻转	CNT=CCR时，REF电平翻转
强制为无效电平	CNT与CCR无效，REF强制为无效电平
强制为有效电平	CNT与CCR无效，REF强制为有效电平
PWM模式1	向上计数：CNT<CCR时，REF置有效电平，CNT≥CCR时，REF置无效电平 向下计数：CNT>CCR时，REF置无效电平，CNT≤CCR时，REF置有效电平
PWM模式2	向上计数：CNT<CCR时，REF置无效电平，CNT≥CCR时，REF置有效电平 向下计数：CNT>CCR时，REF置有效电平，CNT≤CCR时，REF置无效电平

PWM基本结构


99
30
0

时基单元
ARR
自动重装器
PSC
预分频器
CNT
计数器
运行控制
CCR
捕获/比较器
CNT<CCR时，REF置有效电平
CNT≥CCR时，REF置无效电平
REF
极性选择
输出使能
GPIO
输出比较单元×4

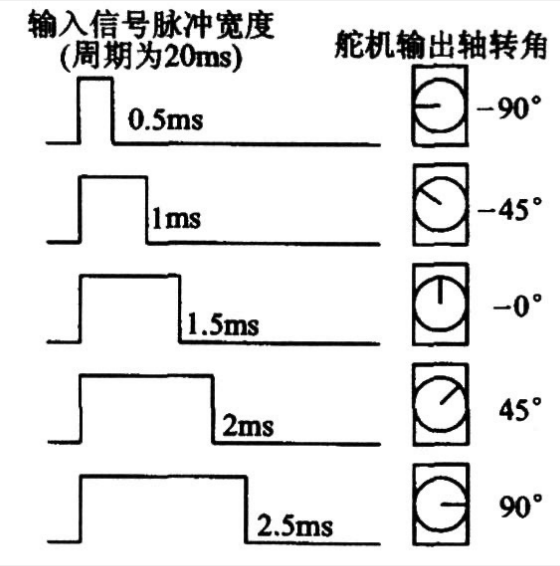
参数计算



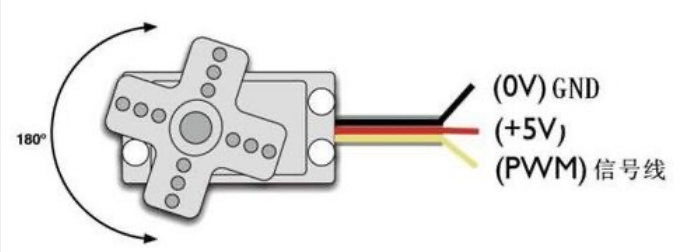
- PWM频率：　Freq = CK_PSC / (PSC + 1) / (ARR + 1)
- PWM占空比：Duty = CCR / (ARR + 1)
- PWM分辨率：Reso = 1 / (ARR + 1)

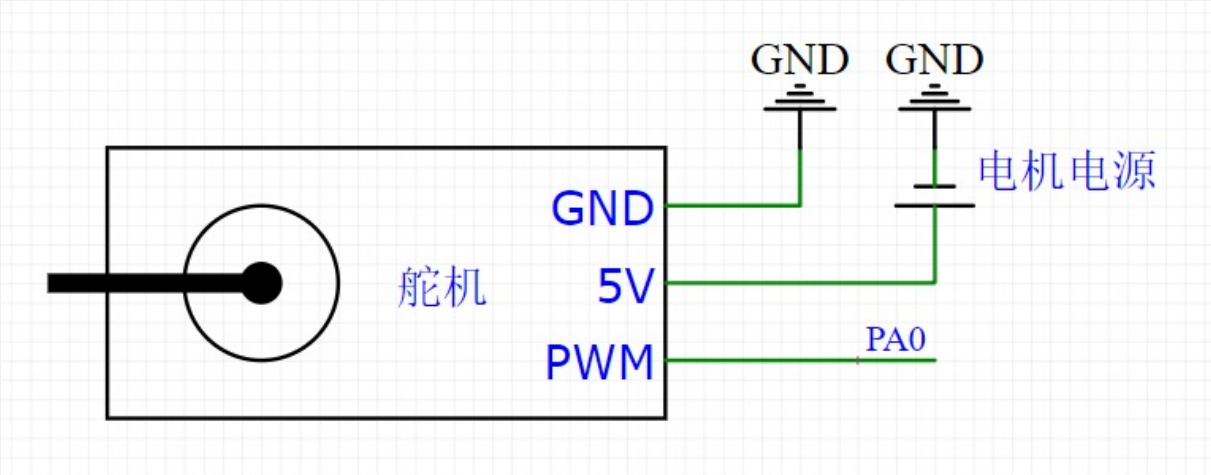
舵机简介

- 舵机是一种根据输入PWM信号占空比来控制输出角度的装置
- 输入PWM信号要求：周期为20ms，高电平宽度为0.5ms~2.5ms



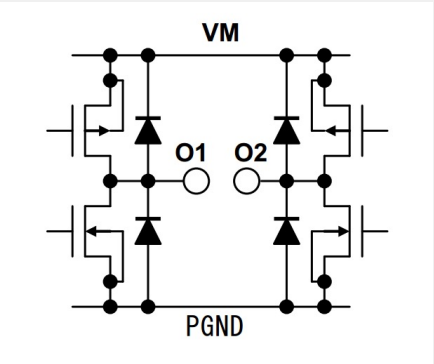
硬件电路



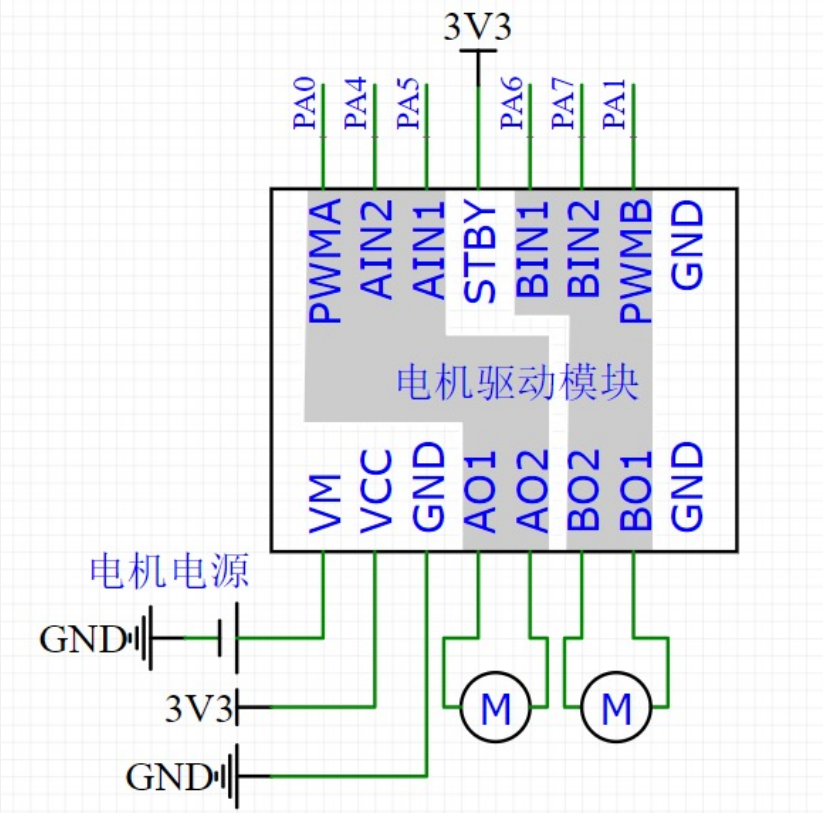


直流电机及驱动简介

- 直流电机是一种将电能转换为机械能的装置，有两个电极，当电极正接时，电机正转，当电极反接时，电机反转
- 直流电机属于大功率器件，GPIO口无法直接驱动，需要配合电机驱动电路来操作
- TB6612是一款双路H桥型的直流电机驱动芯片，可以驱动两个直流电机并且控制其转速和方向



硬件电路



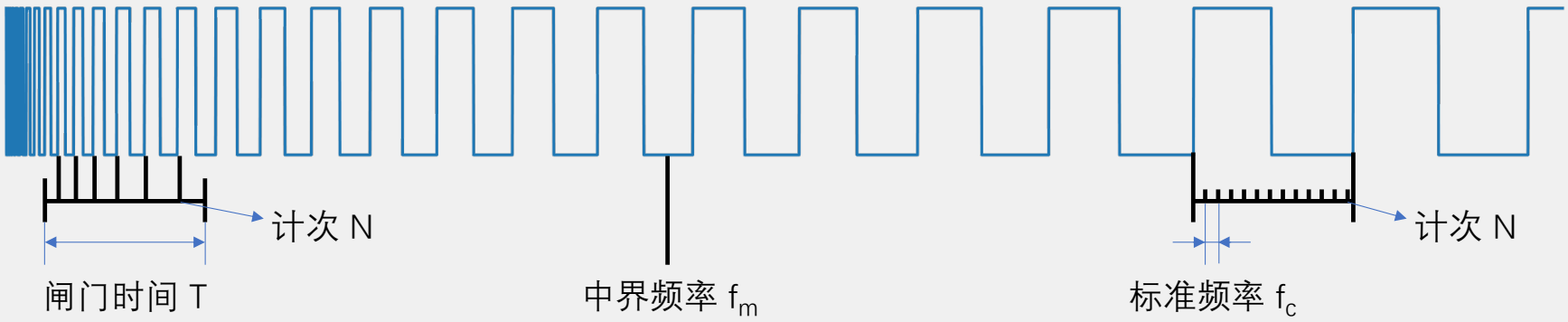
	引脚	定义
	VM	驱动电压输入端（4.5-10V）
	VCC	逻辑电平输入端（2.7-5.5V）
	GND	电源地端
	STBY	正常工作/待机状态控制输入端
1路电机	PWMA	PWM信号输入端
	AIN1	电机控制模式输入端
	AIN2	
	A01	电机驱动输出端
	A02	
2路电机	PWMB	PWM信号输入端
	BIN1	电机控制模式输入端
	BIN2	
	B01	电机驱动输出端
	B02	

输入				输出		
IN1	IN2	PWM	STBY	O1	O2	模式状态
H	H	H/L	H	L	L	制动
L	H	H	H	L	H	反转
L	H	L	H	L	L	制动
H	L	H	H	H	L	正转
H	L	L	H	L	L	制动
L	L	H	H	OFF		停止
H/L	H/L	H/L	L	OFF		待机

输入捕获简介

- IC（Input Capture）输入捕获
- 输入捕获模式下，当通道输入引脚出现指定电平跳变时，当前CNT的值将被锁存到CCR中，可用于测量PWM波形的频率、占空比、脉冲间隔、电平持续时间等参数
- 每个高级定时器和通用定时器都拥有4个输入捕获通道
- 可配置为PWMI模式，同时测量频率和占空比
- 可配合主从触发模式，实现硬件全自动测量

频率测量



- 测频法：在闸门时间T内，对上升沿计次，得到N，则频率

$$f_x = N / T$$

- 测周法：两个上升沿内，以标准频率f_c计次，得到N，则频率

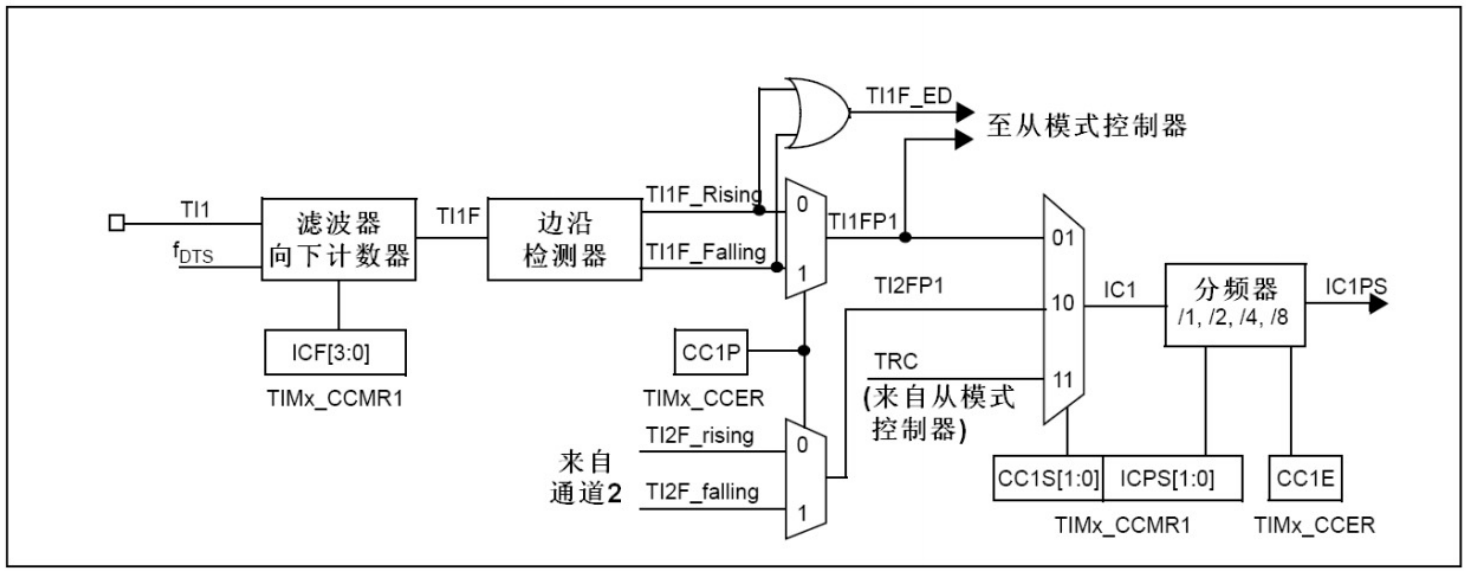
$$f_x = f_c / N$$

- 中界频率：测频法与测周法误差相等的频率点

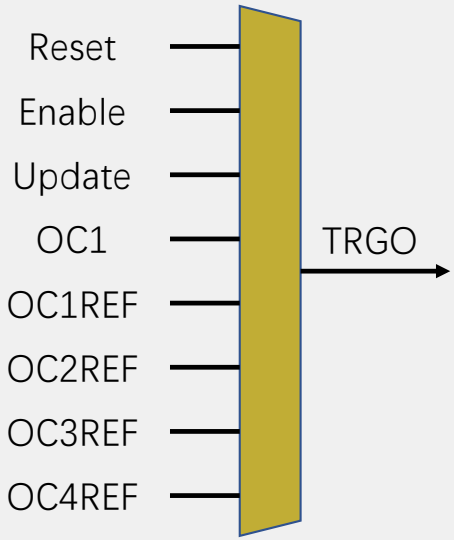
$$f_m = \sqrt{f_c / T}$$

输入捕获通道

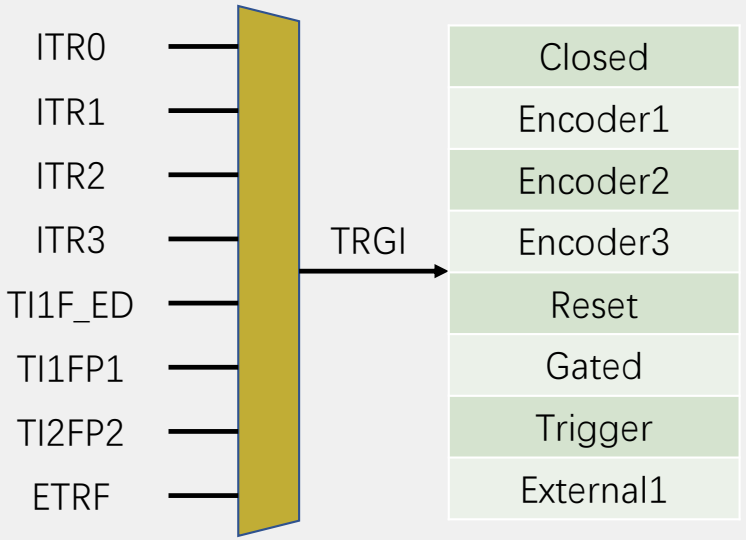
图123　捕获/比较通道(如：通道1输入部分)



主从触发模式



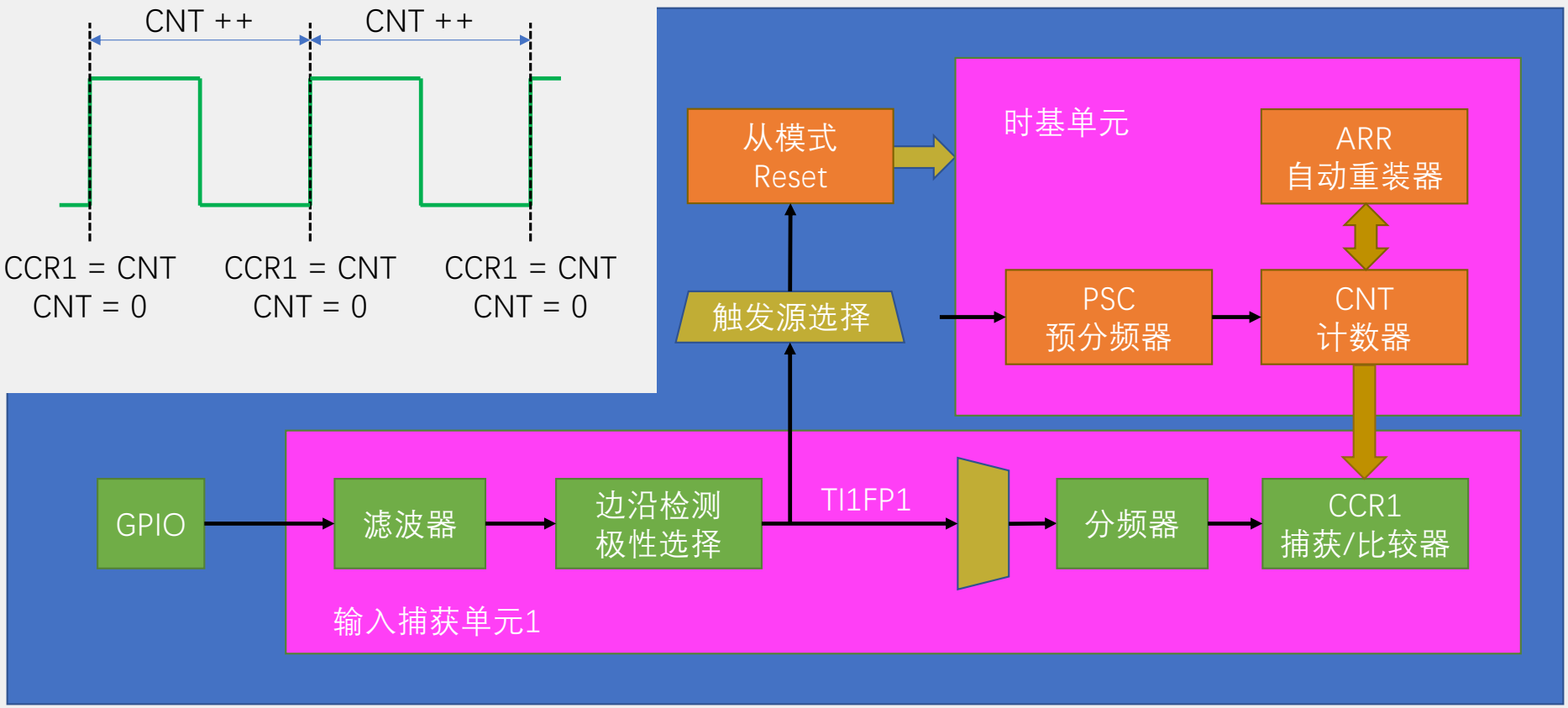
主模式



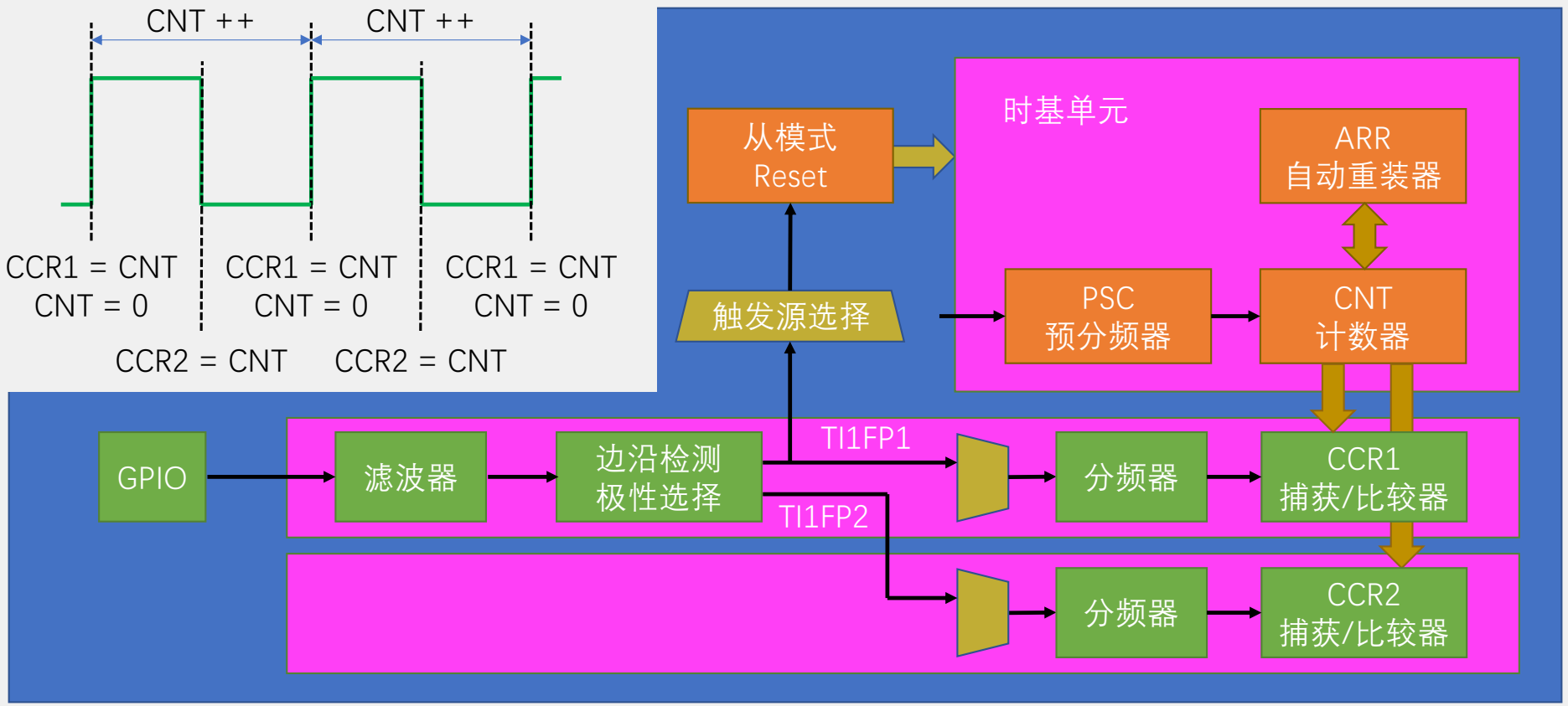
触发源选择

从模式

输入捕获基本结构


CNT ++
CNT ++
CCR1 = CNT
CNT = 0
CCR1 = CNT
CNT = 0
CCR1 = CNT
CNT = 0
从模式
Reset
时基单元
ARR
自动重装器
触发源选择
PSC
预分频器
CNT
计数器
GPIO
滤波器
边沿检测
极性选择
TI1FP1
分频器
CCR1
捕获/比较器
输入捕获单元1

PWMI基本结构

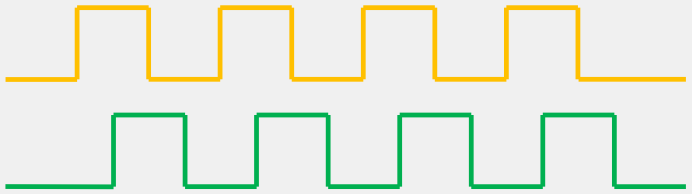

CNT ++
CNT ++
CCR1 = CNT
CNT = 0
CCR1 = CNT
CNT = 0
CCR1 = CNT
CNT = 0
CCR2 = CNT
CCR2 = CNT
从模式
Reset
时基单元
ARR
自动重装器
触发源选择
PSC
预分频器
CNT
计数器
GPIO
滤波器
边沿检测
极性选择
TI1FP1
TI1FP2
分频器
CCR1
捕获/比较器
分频器
CCR2
捕获/比较器

编码器接口简介

- Encoder Interface 编码器接口
- 编码器接口可接收增量（正交）编码器的信号，根据编码器旋转产生的正交信号脉冲，自动控制CNT自增或自减，从而指示编码器的位置、旋转方向和旋转速度
- 每个高级定时器和通用定时器都拥有1个编码器接口
- 两个输入引脚借用了输入捕获的通道1和通道2

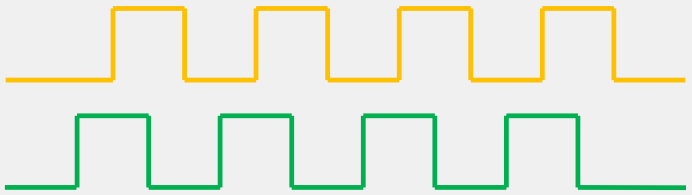
正交编码器

正转



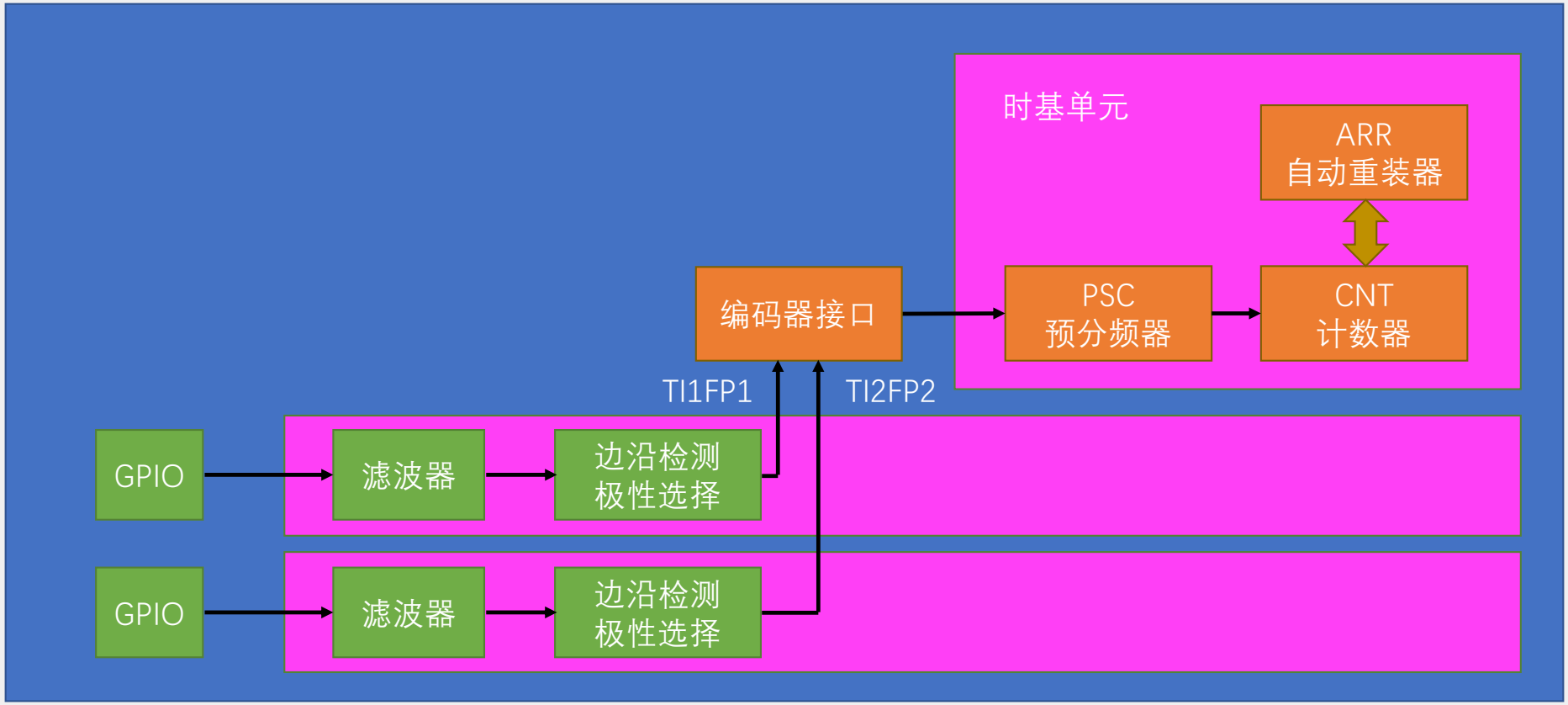
边沿	另一相状态
A相↑	B相低电平
A相↓	B相高电平
B相↑	A相高电平
B相↓	A相低电平

反转



边沿	另一相状态
A相↑	B相高电平
A相↓	B相低电平
B相↑	A相低电平
B相↓	A相高电平

编码器接口基本结构


时基单元
ARR
自动重装器
PSC
预分频器
CNT
计数器
编码器接口
TI1FP1
TI2FP2
GPIO
滤波器
边沿检测
极性选择
GPIO
滤波器
边沿检测
极性选择

工作模式

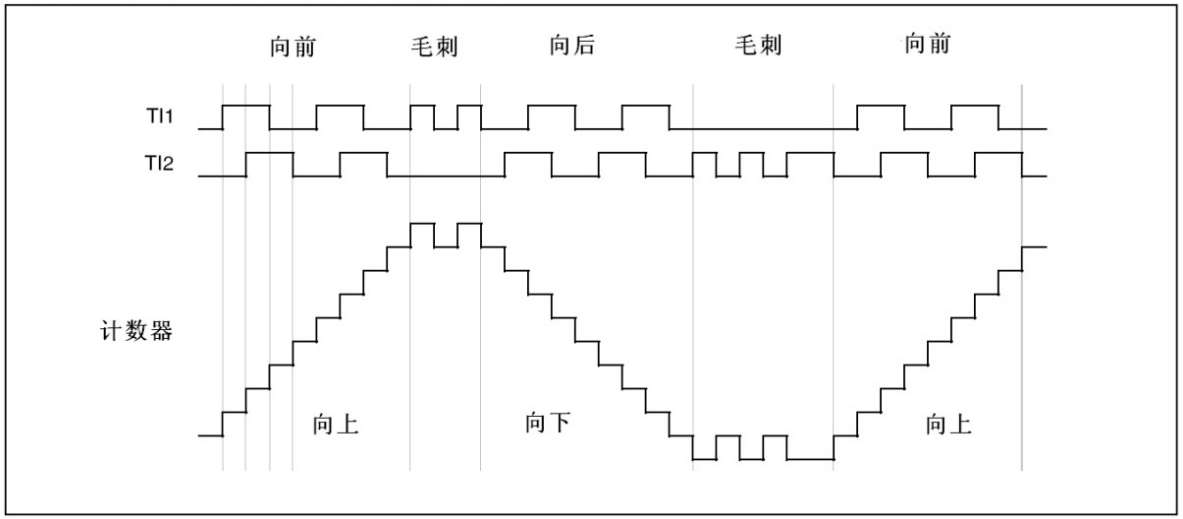
表77　计数方向与编码器信号的关系

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
仅在TI1计数	高	向下计数	向上计数	不计数	不计数
	低	向上计数	向下计数	不计数	不计数
仅在TI2计数	高	不计数	不计数	向上计数	向下计数
	低	不计数	不计数	向下计数	向上计数
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

实例（均不反相）

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

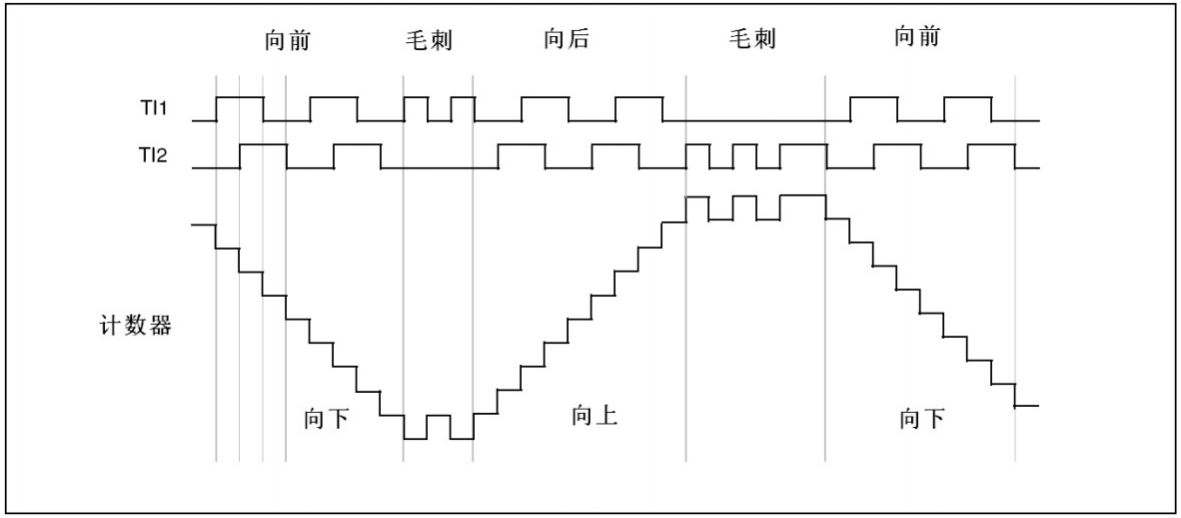
图132　编码器模式下的计数器操作实例



实例（TI1反相）

有效边沿	相对信号的电平 (TI1FP1对应TI2, TI2FP2对应TI1)	TI1FP1信号		TI2FP2信号	
		上升	下降	上升	下降
在TI1和TI2上计数	高	向下计数	向上计数	向上计数	向下计数
	低	向上计数	向下计数	向下计数	向上计数

图133　IC1FP1反相的编码器接口模式实例

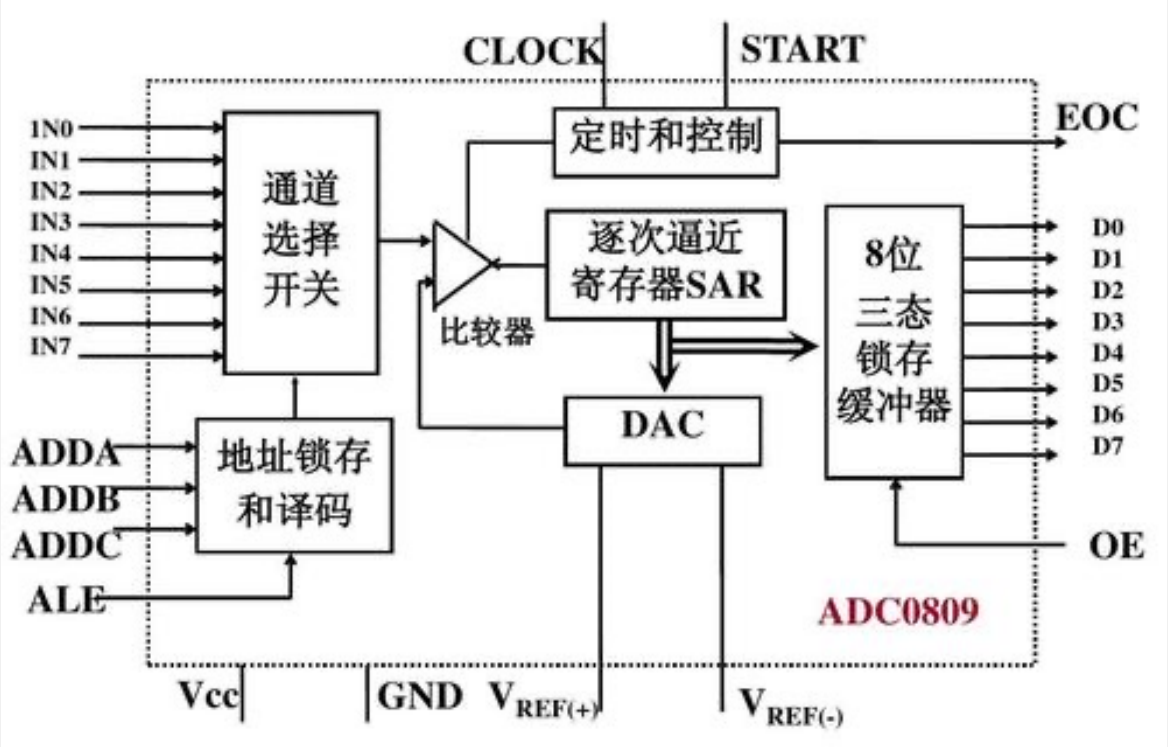


ADC简介

- ADC（Analog-Digital Converter）模拟-数字转换器
- ADC可以将引脚上连续变化的模拟电压转换为内存中存储的数字变量，建立模拟电路到数字电路的桥梁
- 12位逐次逼近型ADC，1us转换时间
- 输入电压范围：0~3.3V，转换结果范围：0~4095
- 18个输入通道，可测量16个外部和2个内部信号源
- 规则组和注入组两个转换单元
- 模拟看门狗自动监测输入电压范围

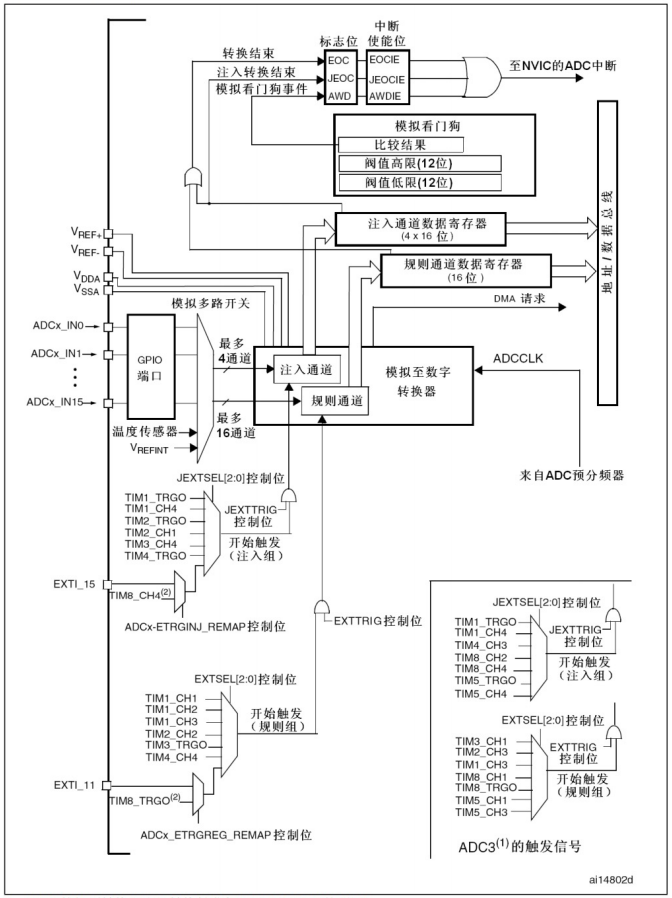
- STM32F103C8T6 ADC资源：ADC1、ADC2，10个外部输入通道

逐次逼近型ADC

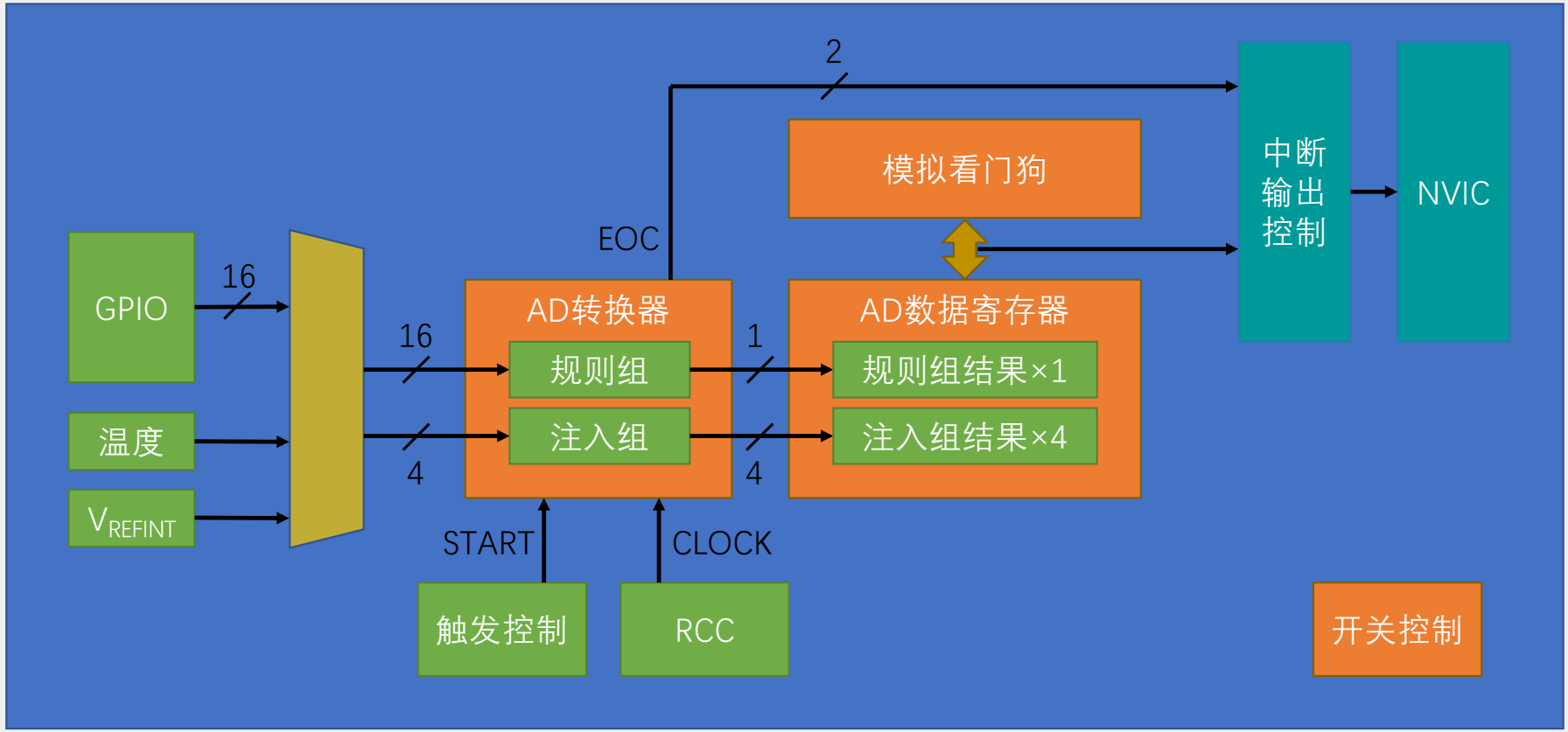

CLOCK
START
EOC
IN0
IN1
IN2
IN3
IN4
IN5
IN6
IN7
通道
选择
开关
定时和控制
逐次逼近
寄存器SAR
比较器
8位
三态
锁存
缓冲器
D0
D1
D2
D3
D4
D5
D6
D7
DAC
ADDA
ADDB
ADDC
ALE
地址锁存
和译码
OE
ADC0809
Vcc
GND
$V_{REF(+)}$
$V_{REF(-)}$

ADC框图

图24　单个ADC框图



ADC基本结构

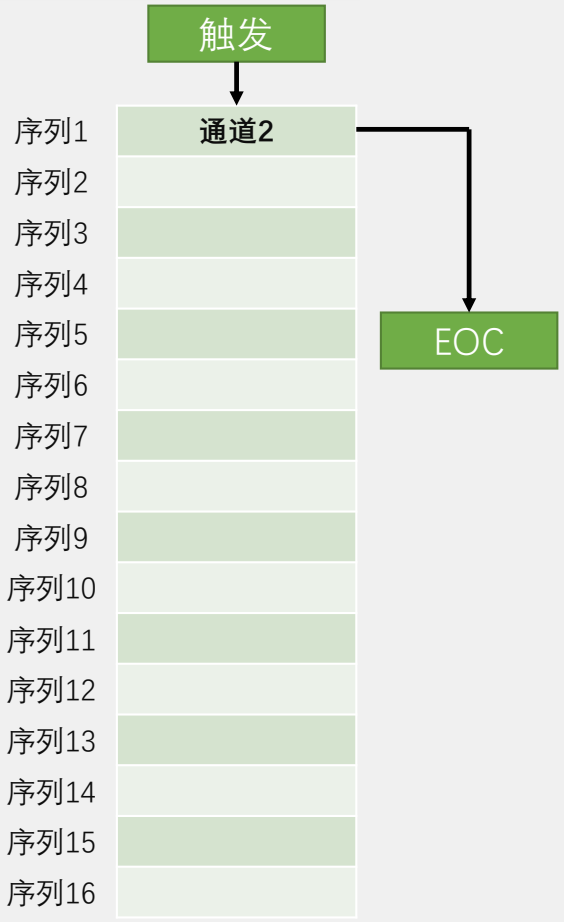

2
模拟看门狗
中断输出控制
NVIC
EOC
GPIO
16
16
AD转换器
规则组
注入组
1
AD数据寄存器
规则组结果×1
注入组结果×4
温度
4
4
VREFINT
START
CLOCK
触发控制
RCC
开关控制

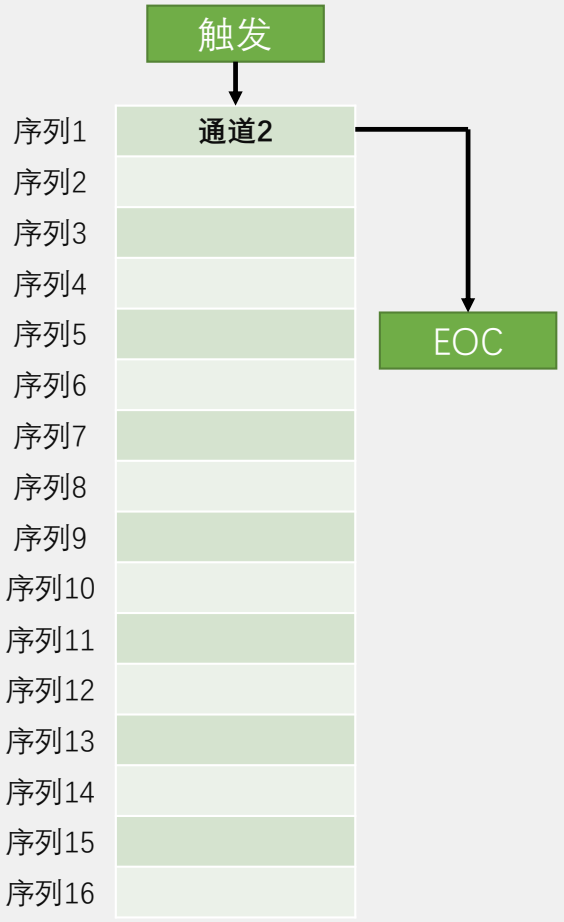
输入通道

通道	ADC1	ADC2	ADC3
通道0	PA0	PA0	PA0
通道1	PA1	PA1	PA1
通道2	PA2	PA2	PA2
通道3	PA3	PA3	PA3
通道4	PA4	PA4	PF6
通道5	PA5	PA5	PF7
通道6	PA6	PA6	PF8
通道7	PA7	PA7	PF9
通道8	PB0	PB0	PF10
通道9	PB1	PB1	
通道10	PC0	PC0	PC0
通道11	PC1	PC1	PC1
通道12	PC2	PC2	PC2
通道13	PC3	PC3	PC3
通道14	PC4	PC4	
通道15	PC5	PC5	
通道16	温度传感器		
通道17	内部参考电压		

转换模式

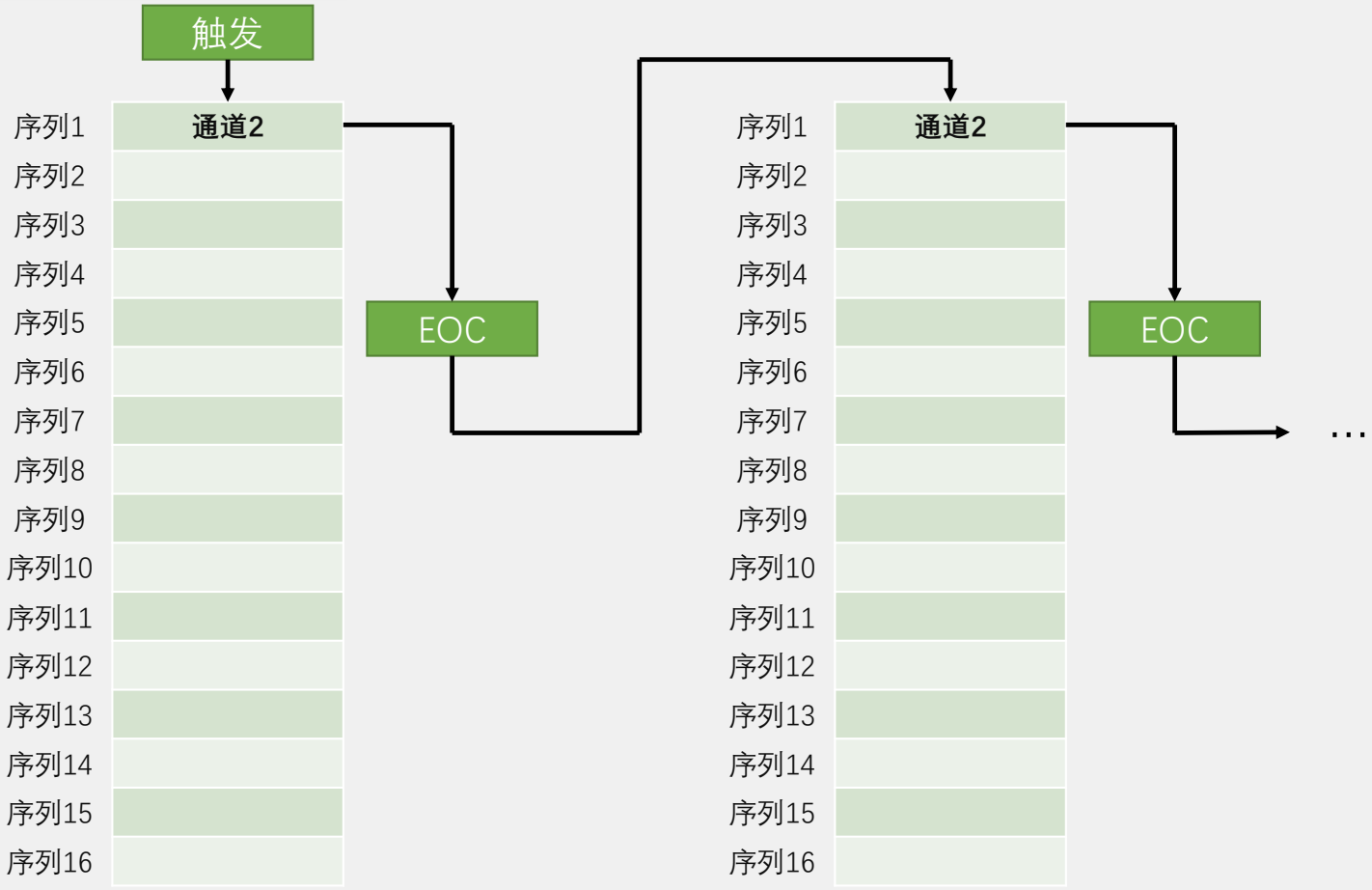
- 单次转换，非扫描模式





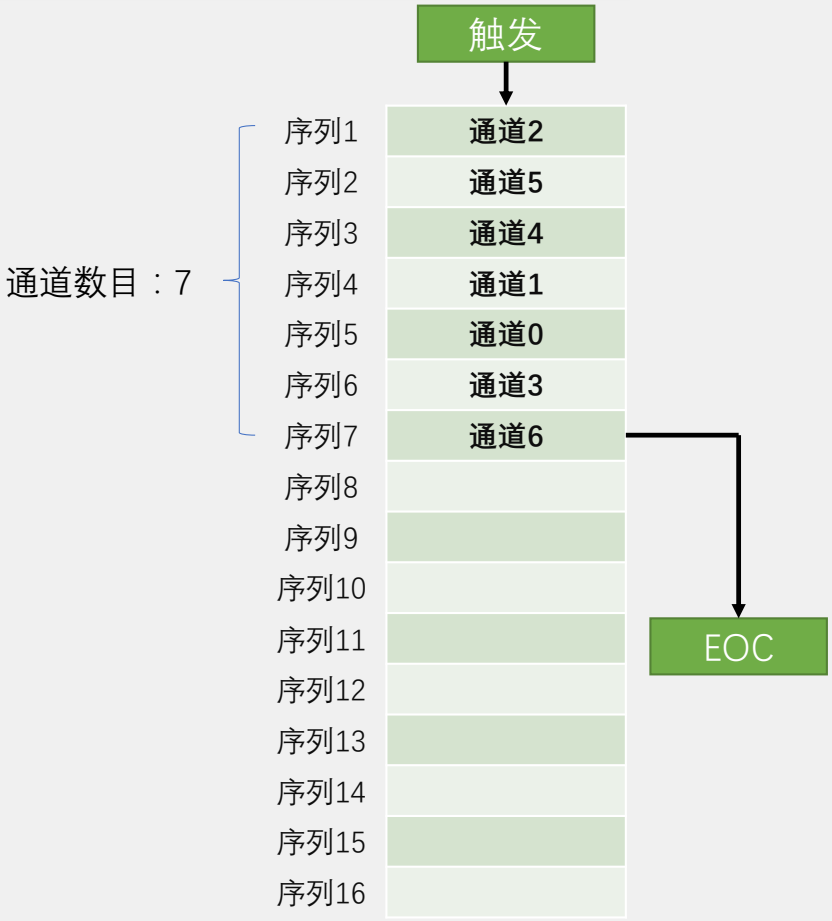
转换模式

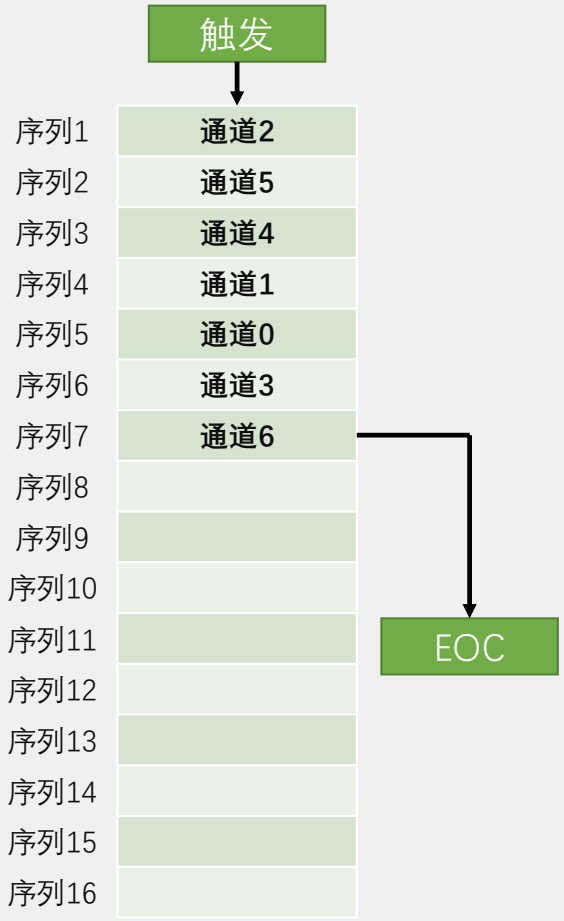
- 连续转换，非扫描模式



转换模式

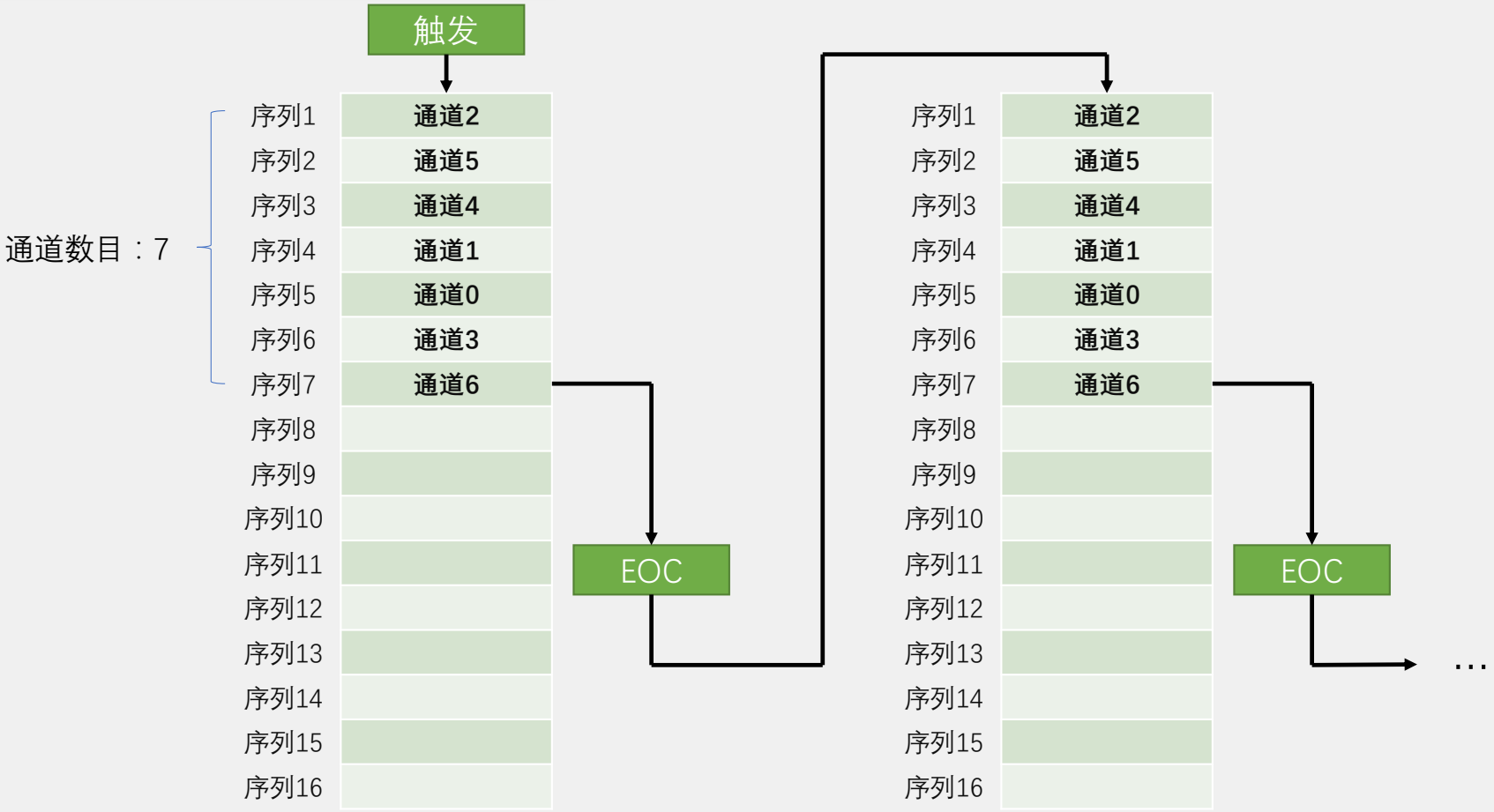
- 单次转换，扫描模式





转换模式

- 连续转换，扫描模式



触发控制

表64　ADC1和ADC2用于规则通道的外部触发

触发源	类型	EXTSEL[2:0]
TIM1_CC1事件	来自片上定时器的内部信号	000
TIM1_CC2事件		001
TIM1_CC3事件		010
TIM2_CC2事件		011
TIM3_TRGO事件		100
TIM4_CC4事件		101
EXTI线11/TIM8_TRGO事件(1)(2)	外部引脚/来自片上定时器的内部信号	110
SWSTART	软件控制位	111

数据对齐

- 数据右对齐：

规则组

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

- 数据左对齐：

规则组

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

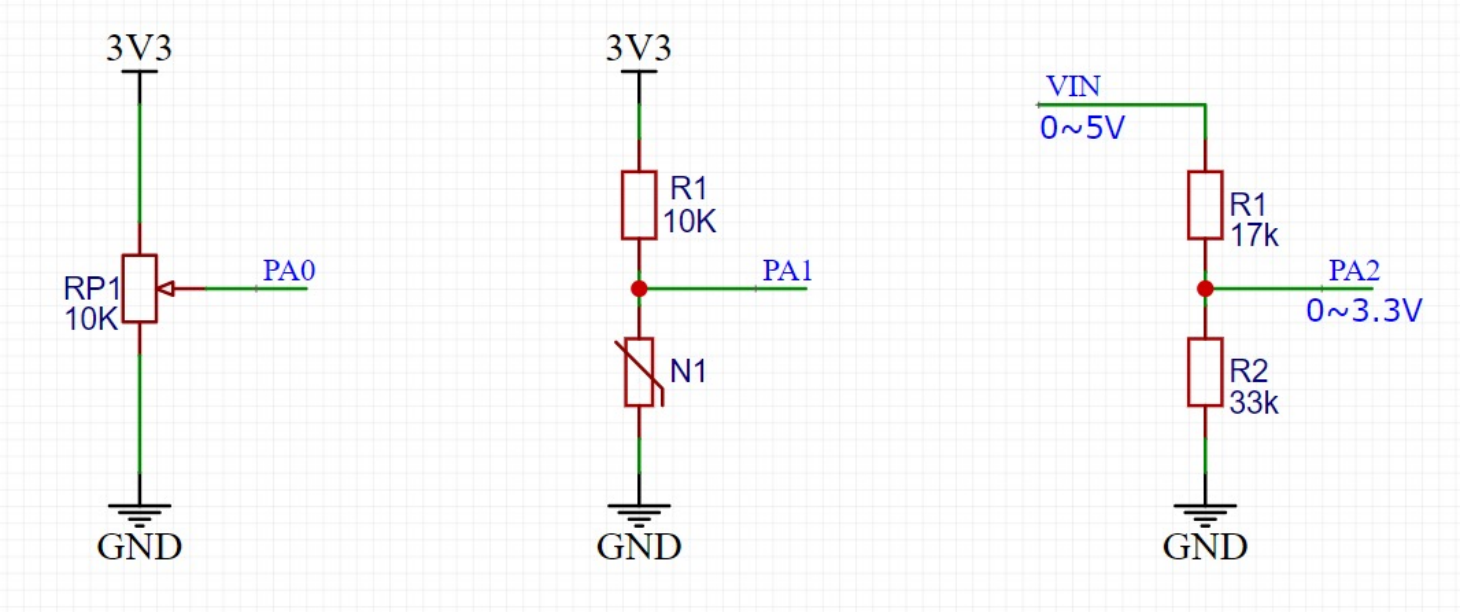
转换时间

- AD转换的步骤：采样，保持，量化，编码
- STM32 ADC的总转换时间为：
  T_{CONV} = 采样时间 + 12.5个ADC周期
- 例如：当ADCCLK=14MHz，采样时间为1.5个ADC周期
  T_{CONV} = 1.5 + 12.5 = 14个ADC周期 = 1μs

校准

- ADC有一个内置自校准模式。校准可大幅减小因内部电容器组的变化而造成的准精度误差。校准期间，在每个电容器上都会计算出一个误差修正码(数字值)，这个码用于消除在随后的转换中每个电容器上产生的误差
- 建议在每次上电后执行一次校准
- 启动校准前， ADC必须处于关电状态超过至少两个ADC时钟周期

硬件电路


3V3
RP1
10K
PA0
GND
3V3
R1
10K
PA1
N1
GND
VIN
0~5V
R1
17k
PA2
0~3.3V
R2
33k
GND

DMA简介

- DMA（Direct Memory Access）直接存储器存取
- DMA可以提供外设和存储器或者存储器和存储器之间的高速数据传输，无须CPU干预，节省了CPU的资源
- 12个独立可配置的通道： DMA1（7个通道）， DMA2（5个通道）
- 每个通道都支持软件触发和特定的硬件触发

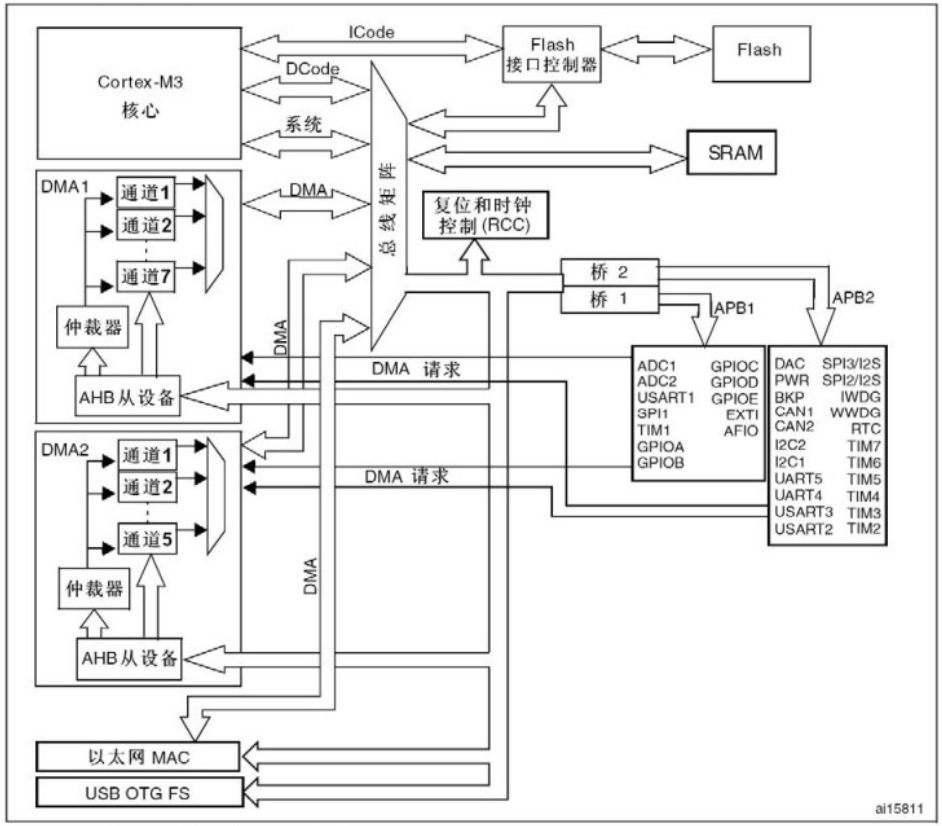
- STM32F103C8T6 DMA资源：DMA1（7个通道）

存储器映像

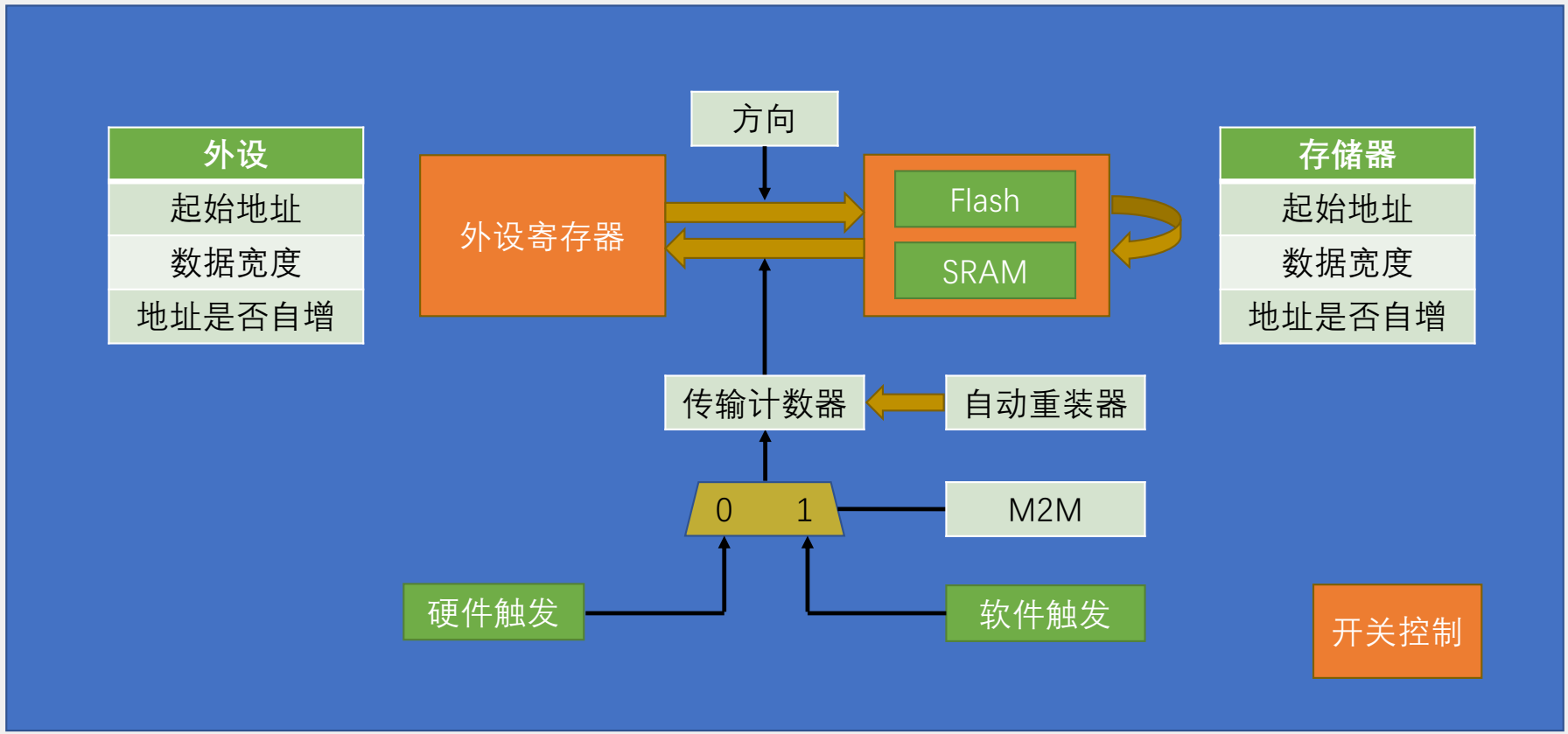
类型	起始地址	存储器	用途
ROM	0x0800 0000	程序存储器Flash	存储C语言编译后的程序代码
	0x1FFF F000	系统存储器	存储BootLoader，用于串口下载
	0x1FFF F800	选项字节	存储一些独立于程序代码的配置参数
RAM	0x2000 0000	运行内存SRAM	存储运行过程中的临时变量
	0x4000 0000	外设寄存器	存储各个外设的配置参数
	0xE000 0000	内核外设寄存器	存储内核各个外设的配置参数

DMA框图

图21　DMA框图

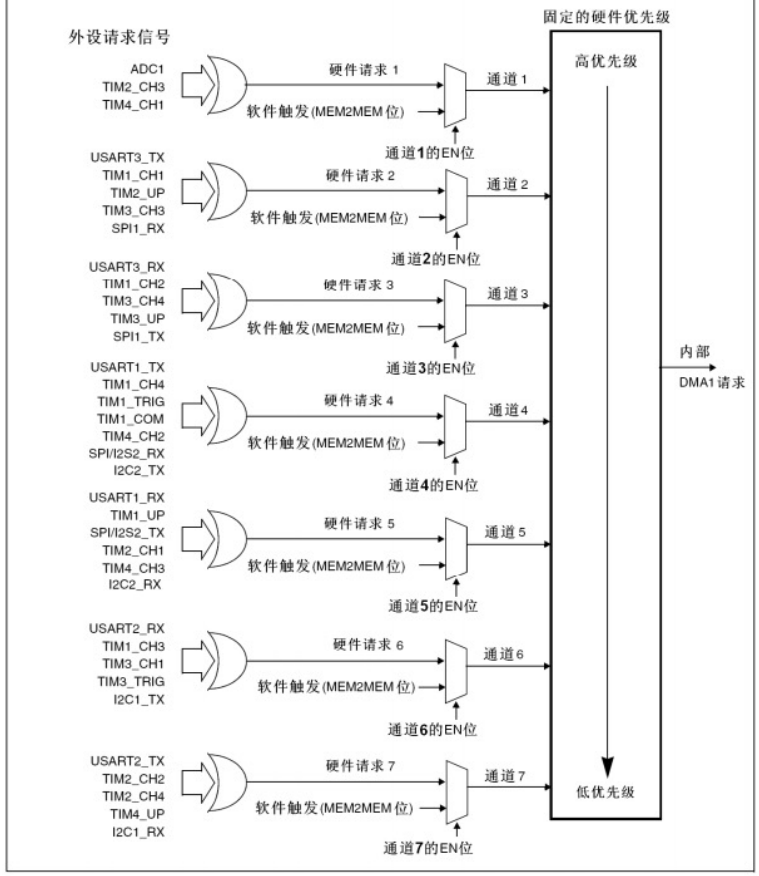


DMA基本结构


外设
起始地址
数据宽度
地址是否自增
方向
外设寄存器
Flash
SRAM
存储器
起始地址
数据宽度
地址是否自增
传输计数器
自动重装器
0
1
M2M
硬件触发
软件触发
开关控制

DMA请求

图22　DMA1请求映像

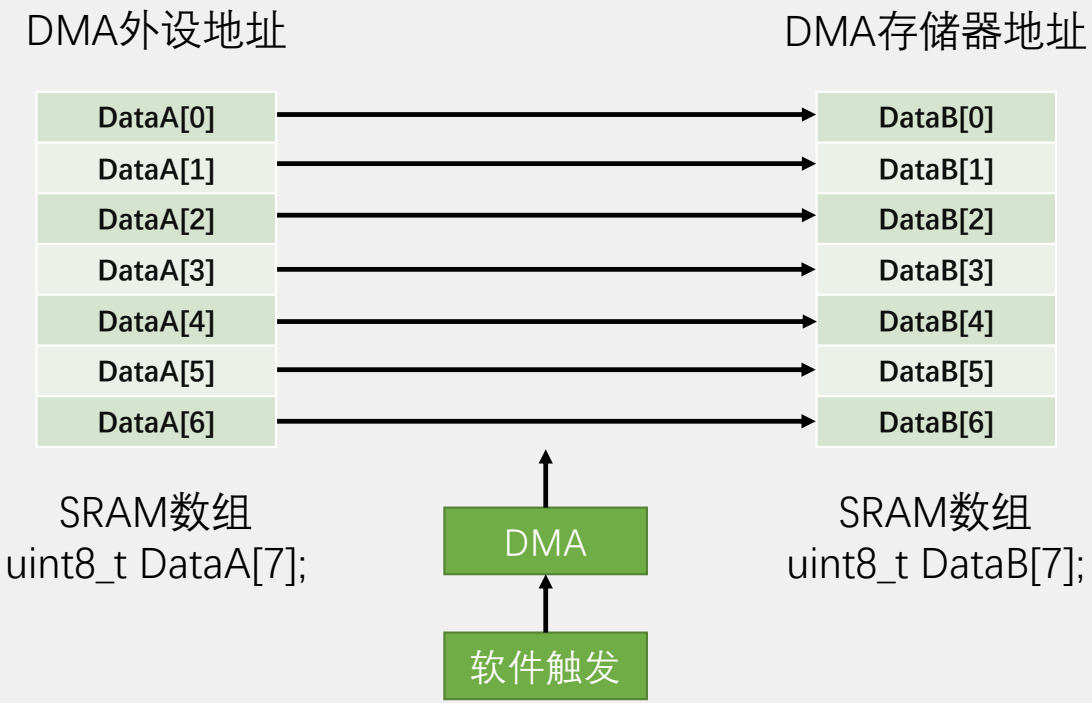


数据宽度与对齐

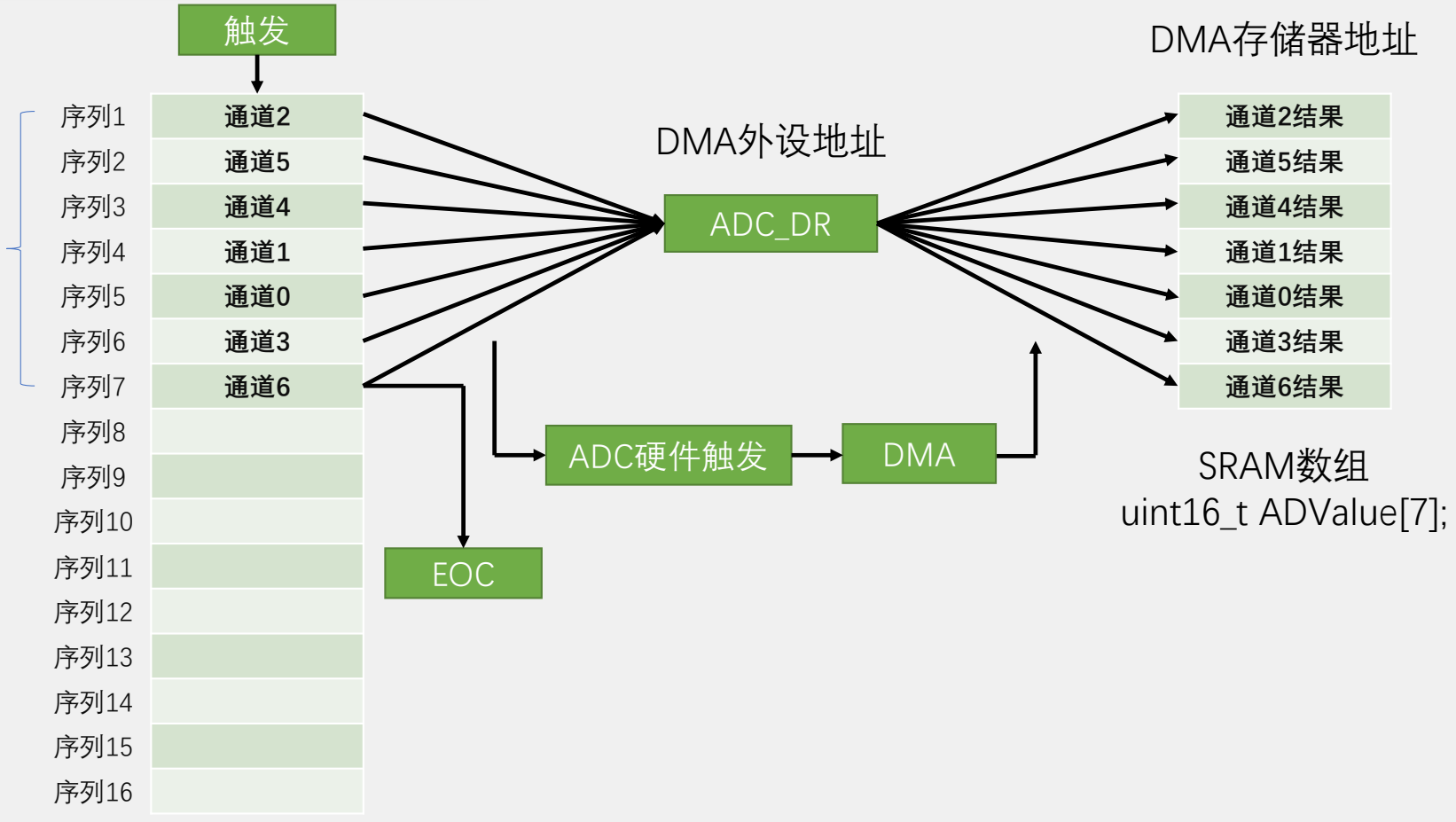
表57　可编程的数据传输宽度和大小端操作(当PINC = MINC = 1)

源端宽度	目标宽度	传输数目	源：地址/数据	传输操作	目标：地址/数据
8	8	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7:0]，在0x0写B0[7:0] 2: 在0x1读B1[7:0]，在0x1写B1[7:0] 3: 在0x2读B2[7:0]，在0x2写B2[7:0] 4: 在0x3读B3[7:0]，在0x3写B3[7:0]	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3
8	16	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7:0]，在0x0写00B0[15:0] 2: 在0x1读B1[7:0]，在0x2写00B1[15:0] 3: 在0x2读B2[7:0]，在0x4写00B2[15:0] 4: 在0x3读B3[7:0]，在0x6写00B3[15:0]	0x0 / 00B0 0x2 / 00B1 0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	1: 在0x0读B0[7:0]，在0x0写000000B0[31:0] 2: 在0x1读B1[7:0]，在0x4写000000B1[31:0] 3: 在0x2读B2[7:0]，在0x8写000000B2[31:0] 4: 在0x3读B3[7:0]，在0xC写000000B3[31:0]	0x0 / 000000B0 0x4 / 000000B1 0x8 / 000000B2 0xC / 000000B3
16	8	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15:0]，在0x0写B0[7:0] 2: 在0x2读B3B2[15:0]，在0x1写B2[7:0] 3: 在0x4读B5B4[15:0]，在0x2写B4[7:0] 4: 在0x6读B7B6[15:0]，在0x3写B6[7:0]	0x0 / B0 0x1 / B2 0x2 / B4 0x3 / B6
16	16	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15:0]，在0x0写B1B0[15:0] 2: 在0x2读B3B2[15:0]，在0x2写B3B2[15:0] 3: 在0x4读B5B4[15:0]，在0x4写B5B4[15:0] 4: 在0x6读B7B6[15:0]，在0x6写B7B6[15:0]	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	1: 在0x0读B1B0[15:0]，在0x0写0000B1B0[31:0] 2: 在0x2读B3B2[15:0]，在0x4写0000B3B2[31:0] 3: 在0x4读B5B4[15:0]，在0x8写0000B5B4[31:0] 4: 在0x6读B7B6[15:0]，在0xC写0000B7B6[31:0]	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31:0]，在0x0写B0[7:0] 2: 在0x4读B7B6B5B4[31:0]，在0x1写B4[7:0] 3: 在0x8读BBBAB9B8[31:0]，在0x2写B8[7:0] 4: 在0xC读BFBEBDBC[31:0]，在0x3写BC[7:0]	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31:0]，在0x0写B1B0[15:0] 2: 在0x4读B7B6B5B4[31:0]，在0x2写B5B4[15:0] 3: 在0x8读BBBAB9B8[31:0]，在0x4写B9B8[15:0] 4: 在0xC读BFBEBDBC[31:0]，在0x6写BDBC[15:0]	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	32	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	1: 在0x0读B3B2B1B0[31:0]，在0x0写B3B2B1B0[31:0] 2: 在0x4读B7B6B5B4[31:0]，在0x4写B7B6B5B4[31:0] 3: 在0x8读BBBAB9B8[31:0]，在0x8写BBBAB9B8[31:0] 4: 在0xC读BFBEBDBC[31:0]，在0xC写BFBEBDBC[31:0]	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC

数据转运+DMA


DMA外设地址
DMA存储器地址
DataA[0]
DataA[1]
DataA[2]
DataA[3]
DataA[4]
DataA[5]
DataA[6]
DataB[0]
DataB[1]
DataB[2]
DataB[3]
DataB[4]
DataB[5]
DataB[6]
SRAM数组
uint8_t DataA[7];
DMA
软件触发
SRAM数组
uint8_t DataB[7];

ADC扫描模式+DMA


触发
序列1
序列2
序列3
序列4
序列5
序列6
序列7
序列8
序列9
序列10
序列11
序列12
序列13
序列14
序列15
序列16
通道2
通道5
通道4
通道1
通道0
通道3
通道6
DMA外设地址
ADC_DR
ADC硬件触发
DMA
EOC
DMA存储器地址
通道2结果
通道5结果
通道4结果
通道1结果
通道0结果
通道3结果
通道6结果
SRAM数组
uint16_t ADValue[7];

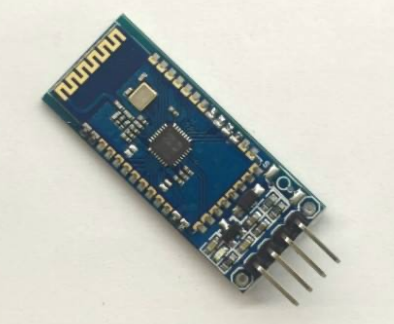
通信接口

- 通信的目的：将一个设备的数据传送到另一个设备，扩展硬件系统
- 通信协议：制定通信的规则，通信双方按照协议规则进行数据收发

名称	引脚	双工	时钟	电平	设备
USART	TX、RX	全双工	异步	单端	点对点
I2C	SCL、SDA	半双工	同步	单端	多设备
SPI	SCLK、MOSI、MISO、CS	全双工	同步	单端	多设备
CAN	CAN_H、CAN_L	半双工	异步	差分	多设备
USB	DP、DM	半双工	异步	差分	点对点

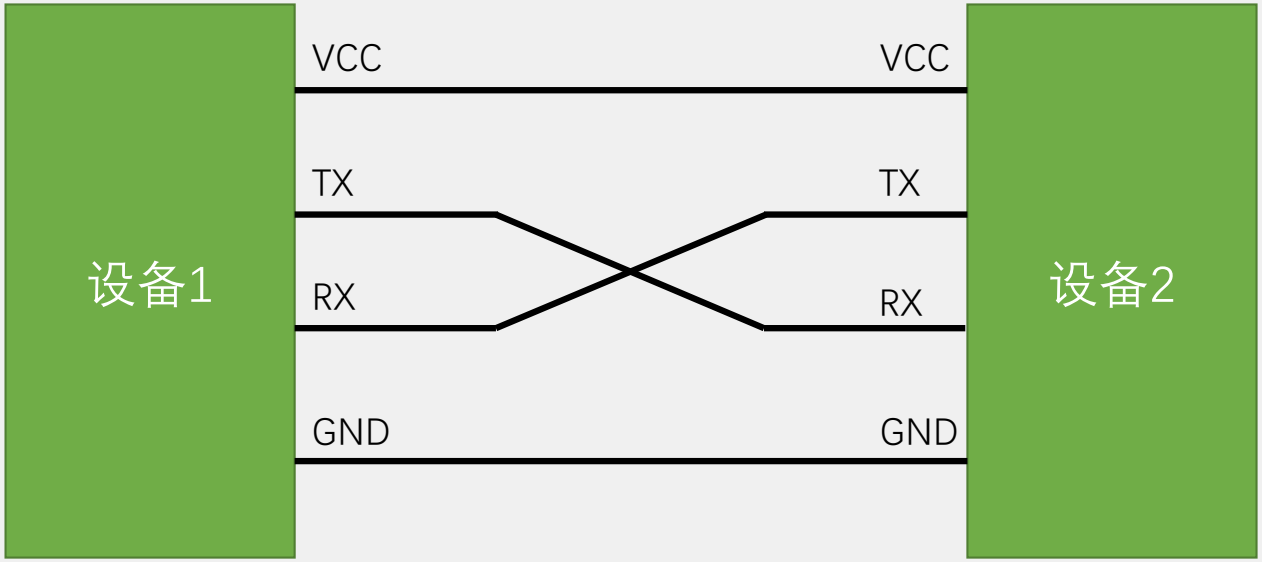
串口通信

- 串口是一种应用十分广泛的通讯接口，串口成本低、容易使用、通信线路简单，可实现两个设备的互相通信
- 单片机的串口可以使单片机与单片机、单片机与电脑、单片机与各式各样的模块互相通信，极大地扩展了单片机的应用范围，增强了单片机系统的硬件实力



硬件电路

- 简单双向串口通信有两根通信线（发送端TX和接收端RX）
- TX与RX要交叉连接
- 当只需单向的数据传输时，可以只接一根通信线
- 当电平标准不一致时，需要加电平转换芯片



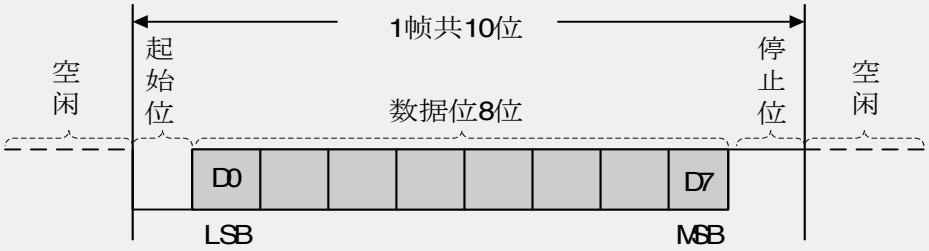
电平标准

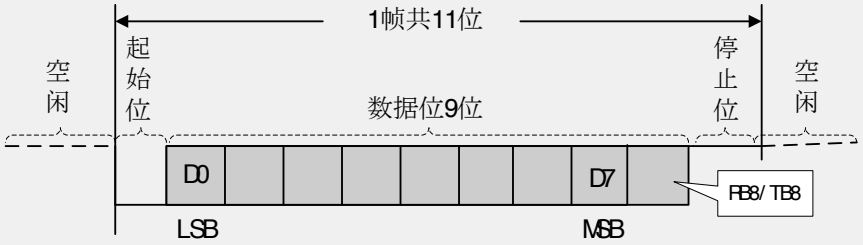
- 电平标准是数据1和数据0的表达方式，是传输线缆中人为规定的电压与数据的对应关系，串口常用的电平标准有如下三种：

- TTL电平：+3.3V或+5V表示1，0V表示0
- RS232电平：-3~-15V表示1，+3~+15V表示0
- RS485电平：两线压差+2~+6V表示1，-2~-6V表示0（差分信号）

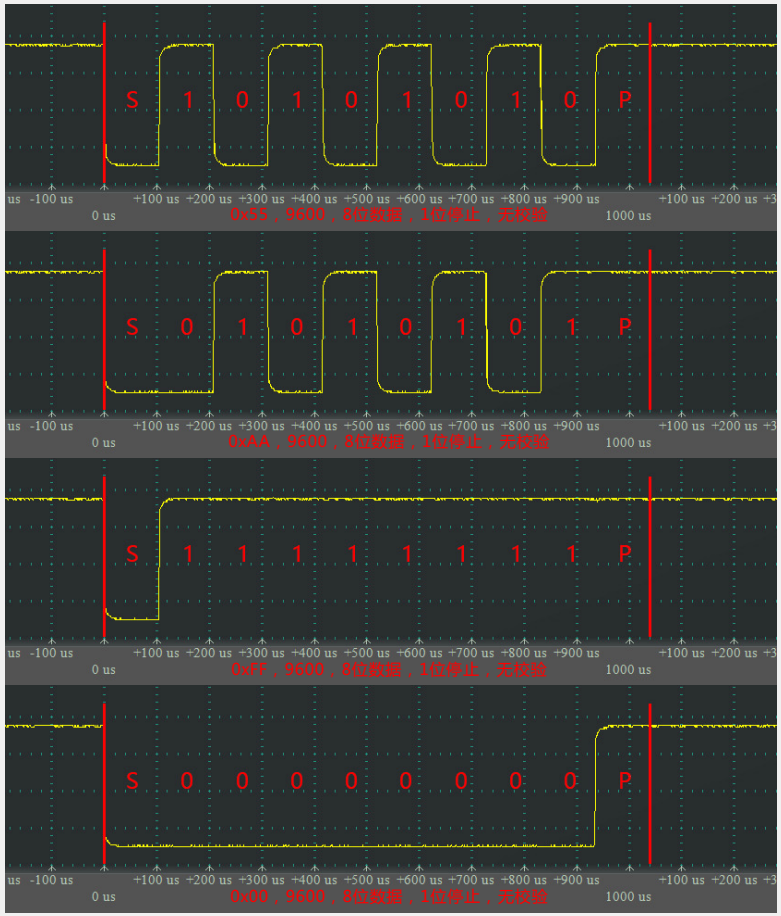
串口参数及时序

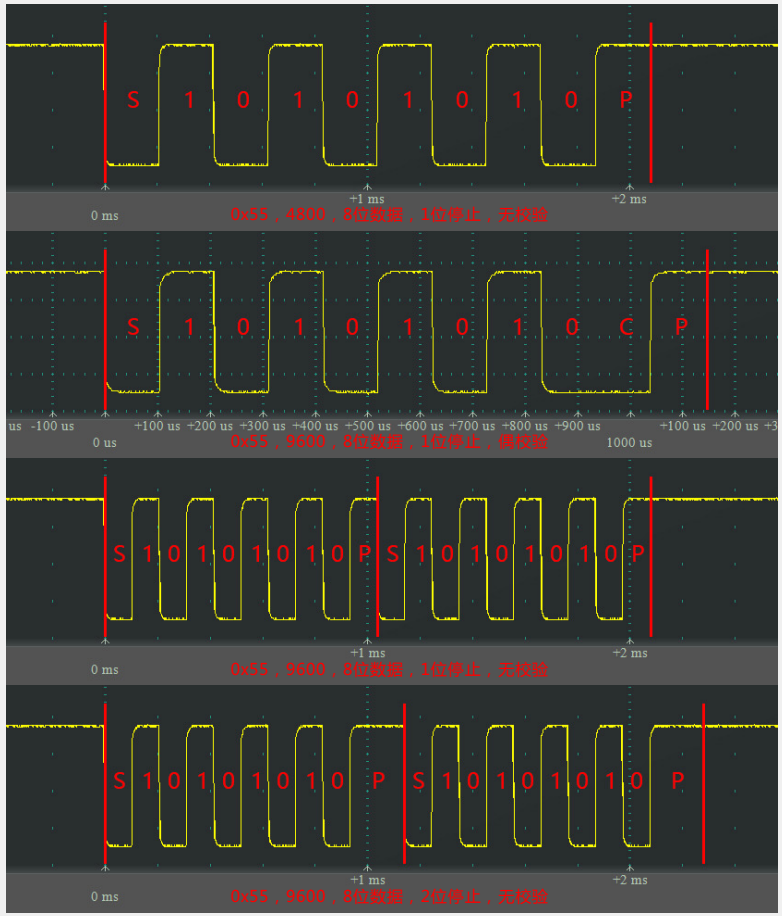
- 波特率：串口通信的速率
- 起始位：标志一个数据帧的开始，固定为低电平
- 数据位：数据帧的有效载荷，1为高电平，0为低电平，低位先行
- 校验位：用于数据验证，根据数据位计算得来
- 停止位：用于数据帧间隔，固定为高电平





串口时序


S 1 0 1 0 1 0 1 0 P
0x55，9600，8位数据，1位停止，无校验
S 0 1 0 1 0 1 0 1 P
0xAA，9600，8位数据，1位停止，无校验
S 1 1 1 1 1 1 1 1 P
0xFF，9600，8位数据，1位停止，无校验
S 0 0 0 0 0 0 0 0 P
0x00，9600，8位数据，1位停止，无校验


S 1 0 1 0 1 0 1 0 P
0x55，4800，8位数据，1位停止，无校验
S 1 0 1 0 1 0 1 0 C P
0x55，9600，9位数据，1位停止，偶校验
S 1 0 1 0 1 0 1 0 P S 1 0 1 0 1 0 1 0 P
0x55，9600，8位数据，1位停止，无校验
S 1 0 1 0 1 0 1 0 P S 1 0 1 0 1 0 1 0 P
0x55，9600，8位数据，2位停止，无校验

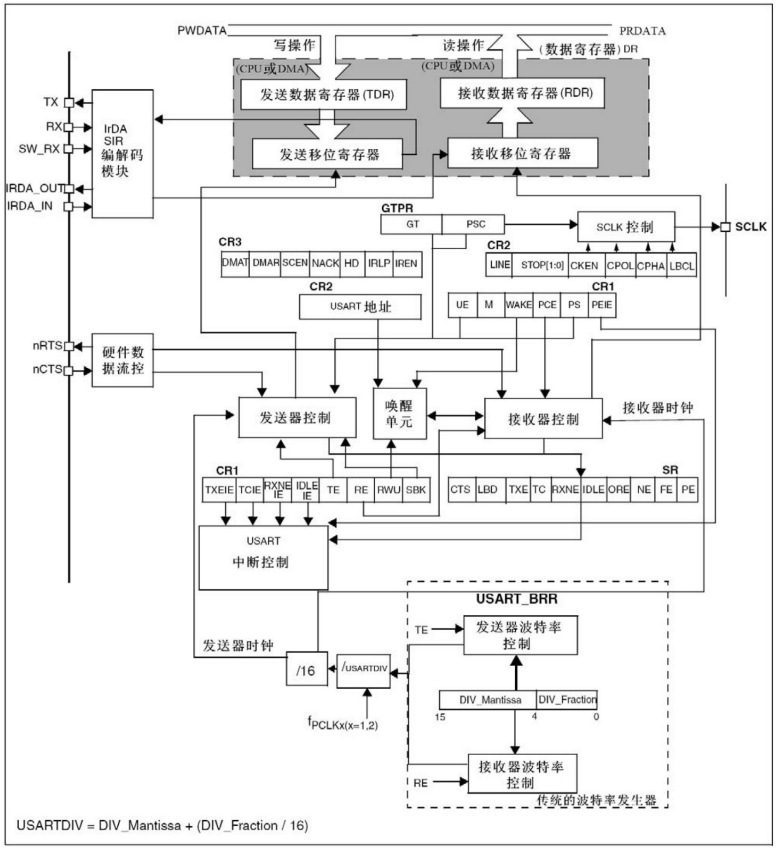
USART简介

- USART（Universal Synchronous/Asynchronous Receiver/Transmitter）通用同步/异步收发器
- USART是STM32内部集成的硬件外设，可根据数据寄存器的一个字节数据自动生成数据帧时序，从TX引脚发送出去，也可自动接收RX引脚的数据帧时序，拼接为一个字节数据，存放在数据寄存器里
- 自带波特率发生器，最高达4.5Mbits/s
- 可配置数据位长度（8/9）、停止位长度（0.5/1/1.5/2）
- 可选校验位（无校验/奇校验/偶校验）
- 支持同步模式、硬件流控制、DMA、智能卡、IrDA、LIN

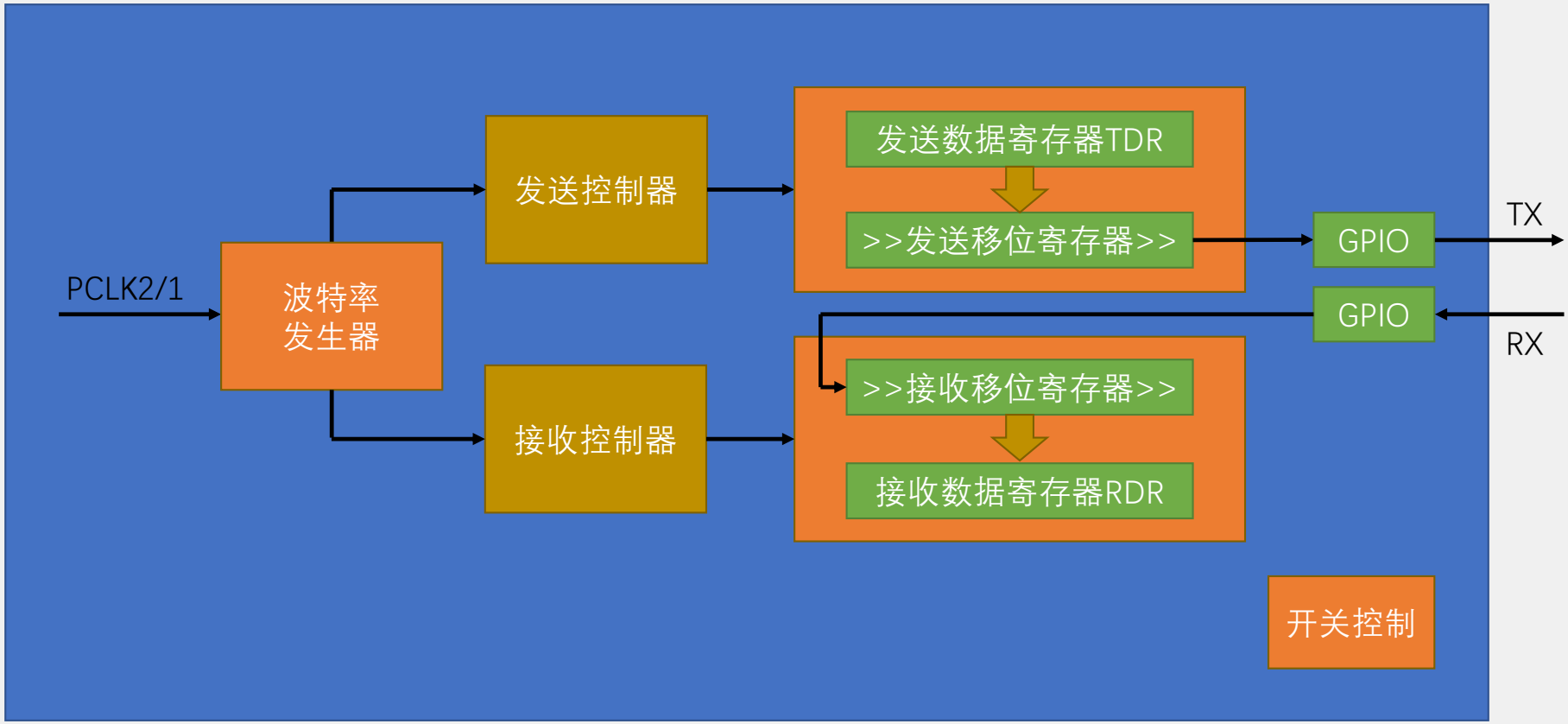
- STM32F103C8T6 USART资源：USART1、USART2、USART3

USART框图

图248　USART框图

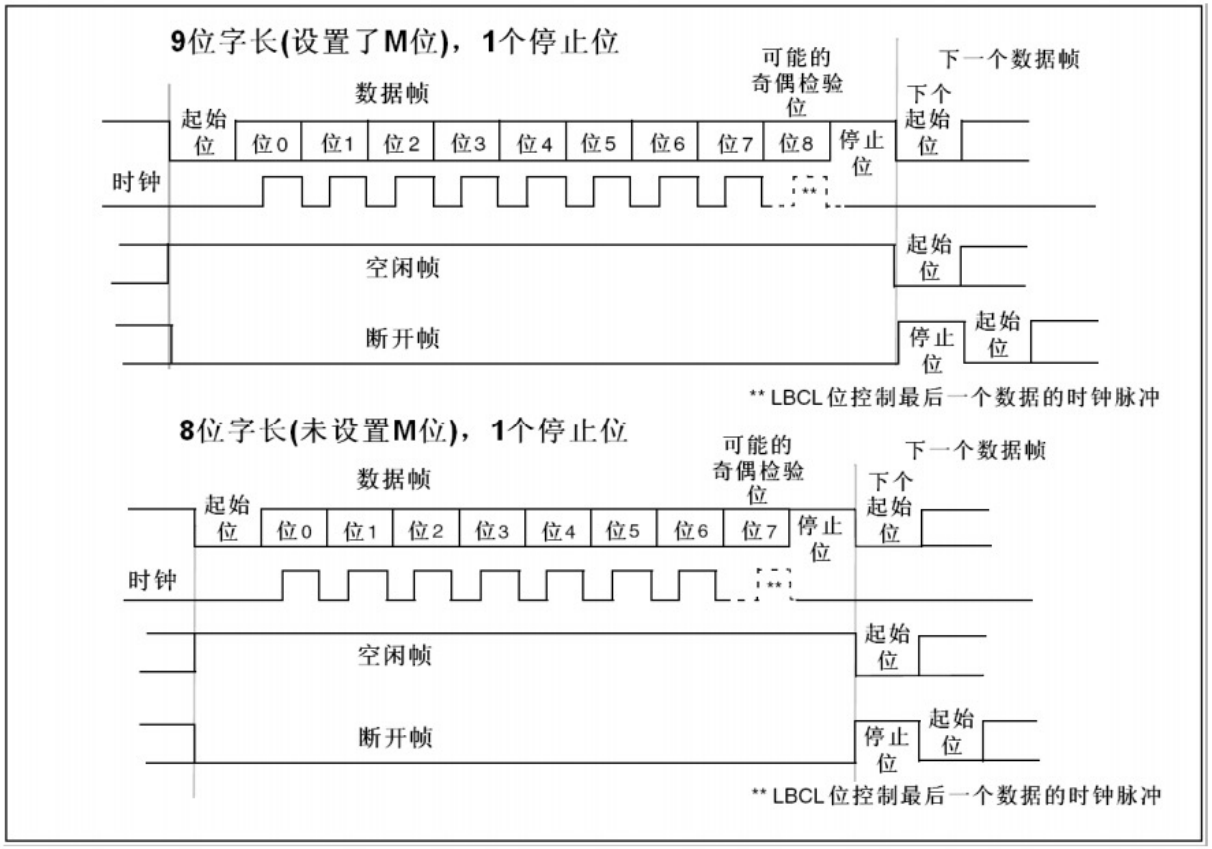


USART基本结构


PCLK2/1
波特率
发生器
发送控制器
接收控制器
发送数据寄存器TDR
>>发送移位寄存器>>
>>接收移位寄存器>>
接收数据寄存器RDR
GPIO
GPIO
TX
RX
开关控制

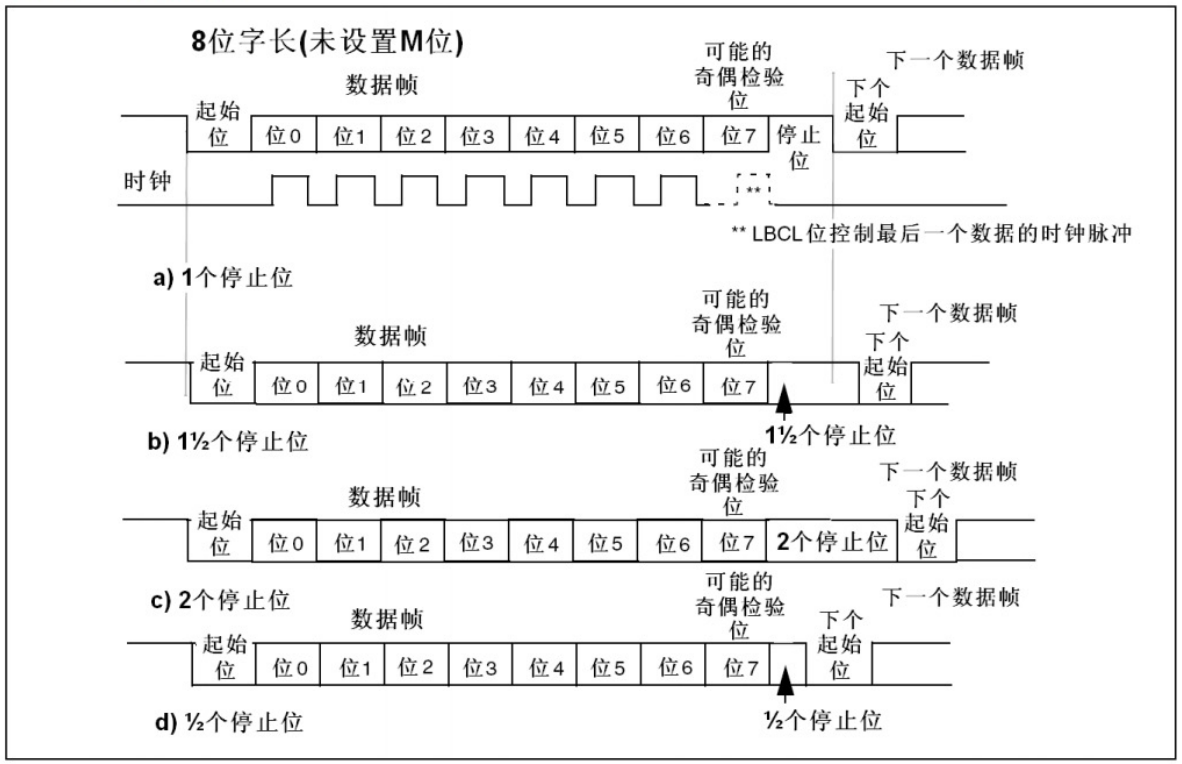
数据帧

图249　字长设置



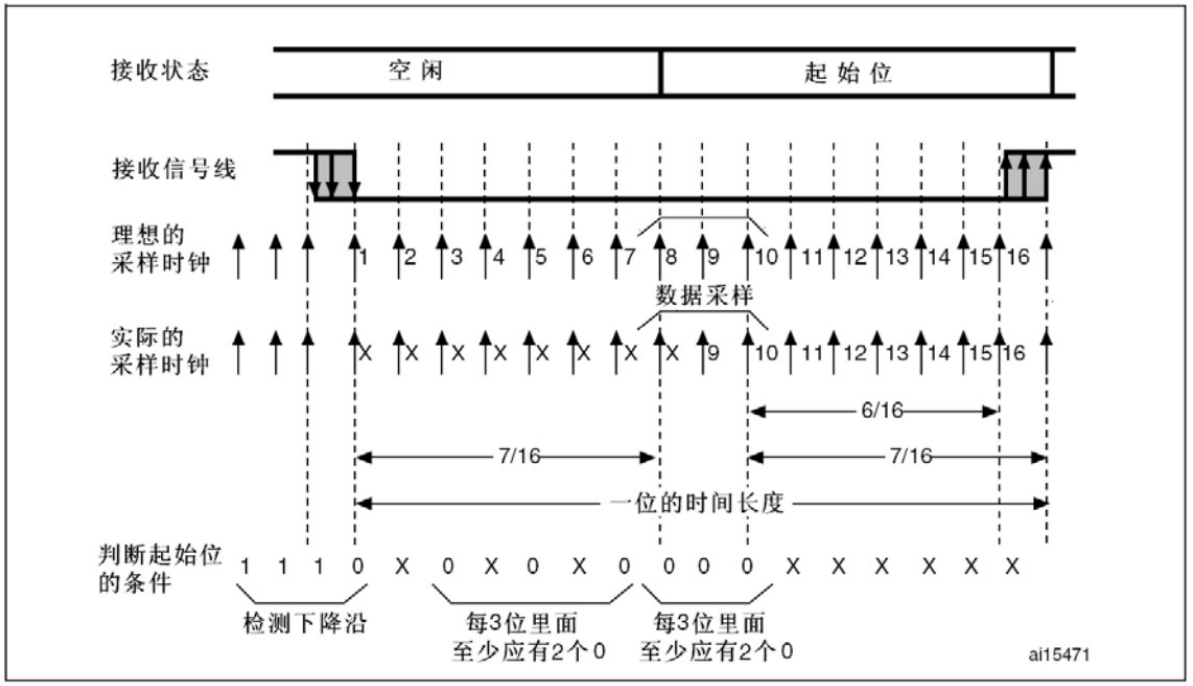
数据帧

图250　配置停止位



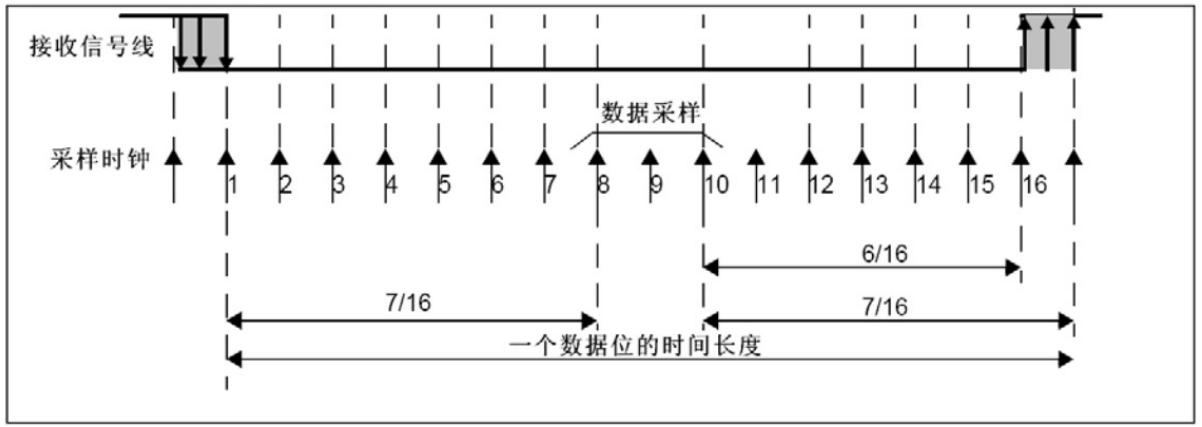
起始位侦测

图252　起始位侦测



数据采样

图253　检测噪声的数据采样



波特率发生器

- 发送器和接收器的波特率由波特率寄存器BRR里的DIV确定
- 计算公式：波特率 = $f_{PCLK2/1}$ / (16 * DIV)

25.6.3 波特比率寄存器(USART_BRR)

注意： *如果TE或RE被分别禁止，波特计数器停止计数*

地址偏移：0x08

复位值：0x0000

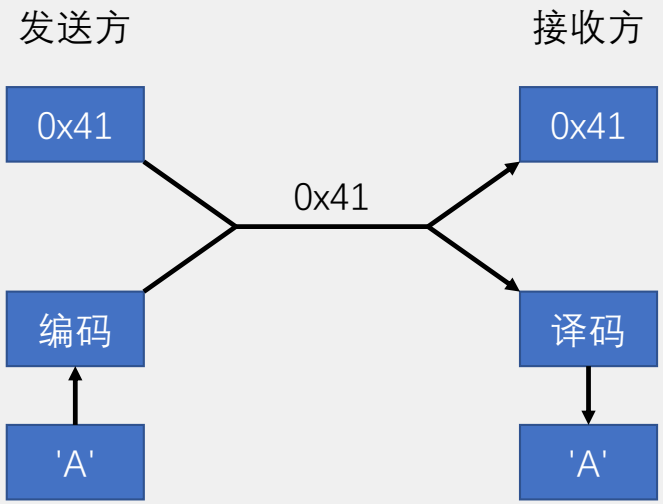
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:16	保留位，硬件强制为0
位15:4	**DIV_Mantissa[11:0]**：USARTDIV的整数部分 这12位定义了USART分频器除法因子(USARTDIV)的整数部分。
位3:0	**DIV_Fraction[3:0]**：USARTDIV的小数部分 这4位定义了USART分频器除法因子(USARTDIV)的小数部分。

数据模式

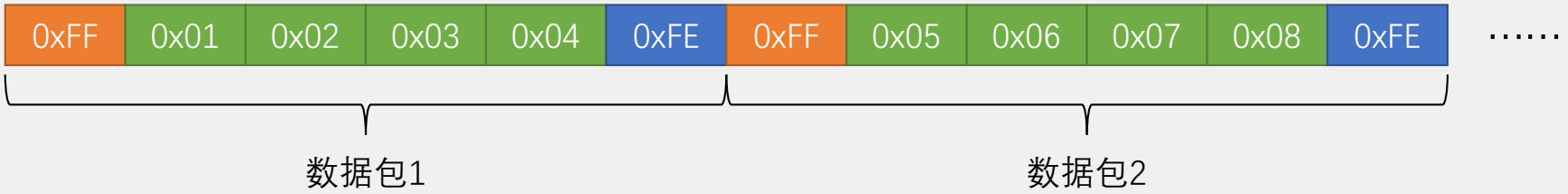
- HEX模式/十六进制模式/二进制模式：以原始数据的形式显示
- 文本模式/字符模式：以原始数据编码后的形式显示

DEC	HEX	字符	解释	DEC	HEX	字符	解释	DEC	HEX	字符	解释	DEC	HEX	字符	解释
0	0x00	(NUL)	空字符	32	0x20	(space)	空格	64	0x40	@	电子邮件符号	96	0x60	`	反单引号
1	0x01	(SOH)	标题开始	33	0x21	!	叹号	65	0x41	A	大写字母A	97	0x61	a	小写字母a
2	0x02	(STX)	正文开始	34	0x22	"	双引号	66	0x42	B	大写字母B	98	0x62	b	小写字母b
3	0x03	(ETX)	正文结束	35	0x23	#	井号	67	0x43	C	大写字母C	99	0x63	c	小写字母c
4	0x04	(EOT)	传输结束	36	0x24	$	美元符	68	0x44	D	大写字母D	100	0x64	d	小写字母d
5	0x05	(ENQ)	请求	37	0x25	%	百分号	69	0x45	E	大写字母E	101	0x65	e	小写字母e
6	0x06	(ACK)	收到通知	38	0x26	&	与号	70	0x46	F	大写字母F	102	0x66	f	小写字母f
7	0x07	(BEL)	响铃 (\a)	39	0x27	'	单引号	71	0x47	G	大写字母G	103	0x67	g	小写字母g
8	0x08	(BS)	退格 (\b)	40	0x28	(	左括号	72	0x48	H	大写字母H	104	0x68	h	小写字母h
9	0x09	(HT)	水平制表符 (\t)	41	0x29	)	右括号	73	0x49	I	大写字母I	105	0x69	i	小写字母i
10	0x0A	(LF)	换行键 (\n)	42	0x2A	*	星号	74	0x4A	J	大写字母J	106	0x6A	j	小写字母j
11	0x0B	(VT)	垂直制表符 (\v)	43	0x2B	+	加号	75	0x4B	K	大写字母K	107	0x6B	k	小写字母k
12	0x0C	(FF)	换页键 (\f)	44	0x2C	,	逗号	76	0x4C	L	大写字母L	108	0x6C	l	小写字母l
13	0x0D	(CR)	回车键 (\r)	45	0x2D	-	减号	77	0x4D	M	大写字母M	109	0x6D	m	小写字母m
14	0x0E	(SO)	不用切换	46	0x2E	.	句号	78	0x4E	N	大写字母N	110	0x6E	n	小写字母n
15	0x0F	(SI)	启用切换	47	0x2F	/	斜杠	79	0x4F	O	大写字母O	111	0x6F	o	小写字母o
16	0x10	(DLE)	数据链路转义	48	0x30	0	字符0	80	0x50	P	大写字母P	112	0x70	p	小写字母p
17	0x11	(DC1)	设备控制1	49	0x31	1	字符1	81	0x51	Q	大写字母Q	113	0x71	q	小写字母q
18	0x12	(DC2)	设备控制2	50	0x32	2	字符2	82	0x52	R	大写字母R	114	0x72	r	小写字母r
19	0x13	(DC3)	设备控制3	51	0x33	3	字符3	83	0x53	S	大写字母S	115	0x73	s	小写字母s
20	0x14	(DC4)	设备控制4	52	0x34	4	字符4	84	0x54	T	大写字母T	116	0x74	t	小写字母t
21	0x15	(NAK)	拒绝接收	53	0x35	5	字符5	85	0x55	U	大写字母U	117	0x75	u	小写字母u
22	0x16	(SYN)	同步空闲	54	0x36	6	字符6	86	0x56	V	大写字母V	118	0x76	v	小写字母v
23	0x17	(ETB)	结束传输块	55	0x37	7	字符7	87	0x57	W	大写字母W	119	0x77	w	小写字母w
24	0x18	(CAN)	取消	56	0x38	8	字符8	88	0x58	X	大写字母X	120	0x78	x	小写字母x
25	0x19	(EM)	媒介结束	57	0x39	9	字符9	89	0x59	Y	大写字母Y	121	0x79	y	小写字母y
26	0x1A	(SUB)	代替	58	0x3A	:	冒号	90	0x5A	Z	大写字母Z	122	0x7A	z	小写字母z
27	0x1B	(ESC)	换码	59	0x3B	;	分号	91	0x5B	[	左方括号	123	0x7B	{	左花括号
28	0x1C	(FS)	文件分隔符	60	0x3C	<	小于	92	0x5C	\	反斜杠	124	0x7C	|	竖线
29	0x1D	(GS)	分组符	61	0x3D	=	等号	93	0x5D	]	右方括号	125	0x7D	}	右花括号
30	0x1E	(RS)	记录分隔符	62	0x3E	>	大于	94	0x5E	^	次方号	126	0x7E	~	波浪号
31	0x1F	(US)	单元分隔符	63	0x3F	?	问号	95	0x5F	_	下划线	127	0x7F	(DEL)	删除

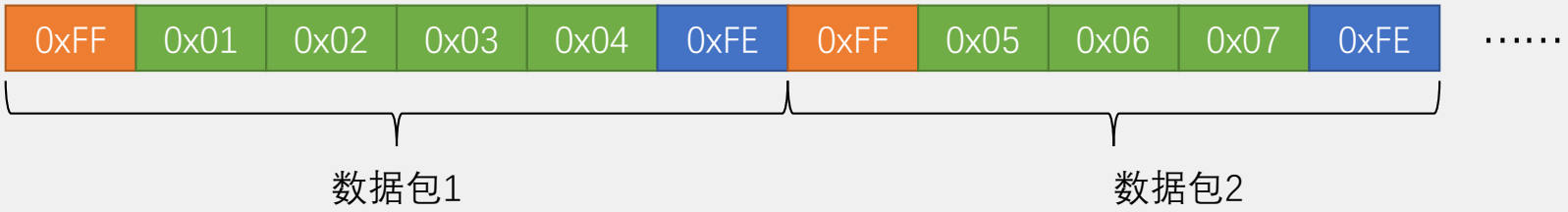


HEX数据包

- 固定包长，含包头包尾

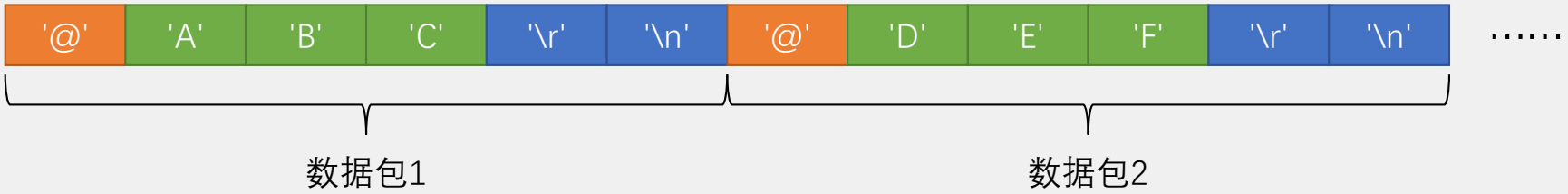


- 可变包长，含包头包尾



文本数据包

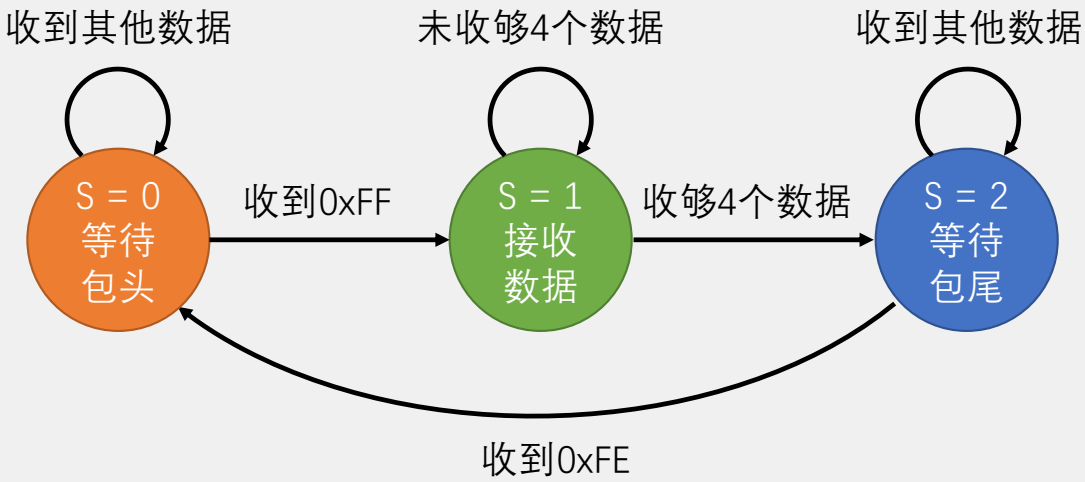
- 固定包长，含包头包尾



- 可变包长，含包头包尾

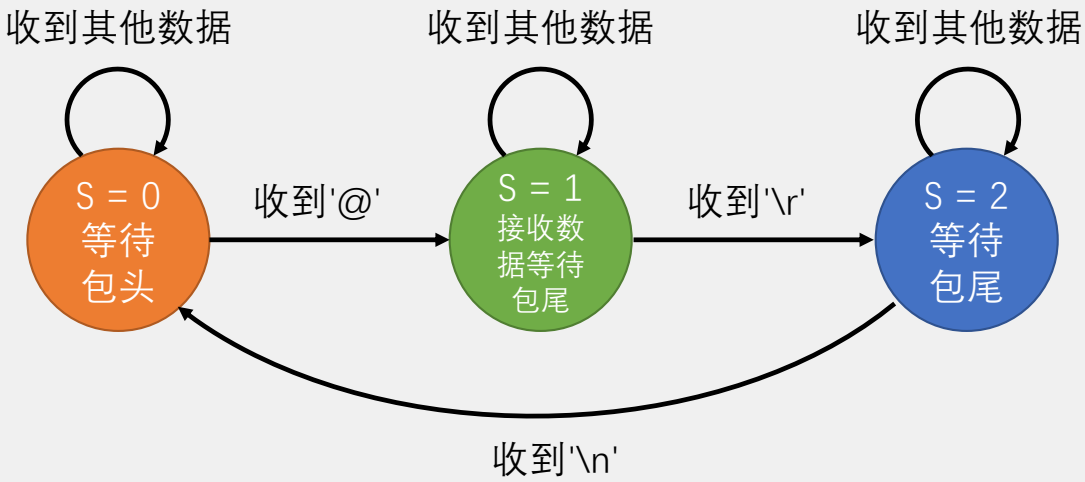
HEX数据包接收


0xFF
0x01
0x02
0x03
0x04
0xFE
0xFF
0x05
0x06
0x07
0x08
0xFE
.....


收到其他数据
未收够4个数据
收到其他数据
S = 0
等待
包头
收到0xFF
S = 1
接收
数据
收够4个数据
S = 2
等待
包尾
收到0xFE

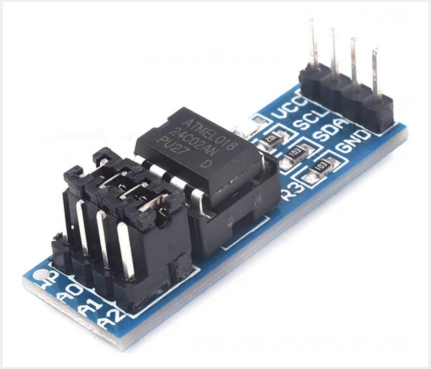
文本数据包接收


'@'
'A'
'B'
'C'
'\r'
'\n'
'@'
'D'
'E'
'\r'
'\n'
.....


收到其他数据
收到其他数据
收到其他数据
S = 0
等待
包头
收到'@'
S = 1
接收数
据等待
包尾
收到'\r'
S = 2
等待
包尾
收到'\n'

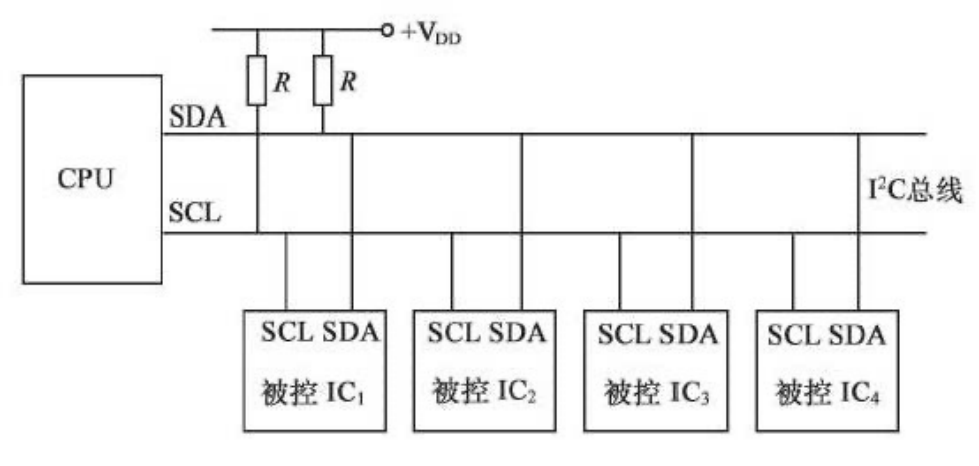
I2C通信

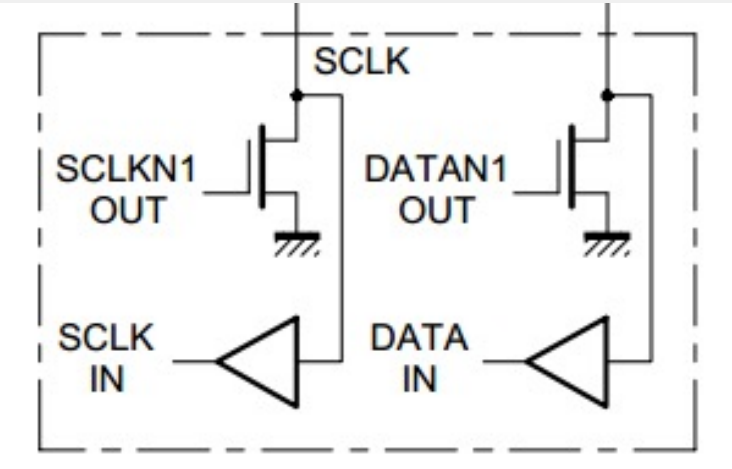
- I2C（Inter IC Bus）是由Philips公司开发的一种通用数据总线
- 两根通信线：SCL（Serial Clock）、SDA（Serial Data）
- 同步，半双工
- 带数据应答
- 支持总线挂载多设备（一主多从、多主多从）



硬件电路

- 所有I2C设备的SCL连在一起，SDA连在一起
- 设备的SCL和SDA均要配置成开漏输出模式
- SCL和SDA各添加一个上拉电阻，阻值一般为4.7KΩ左右



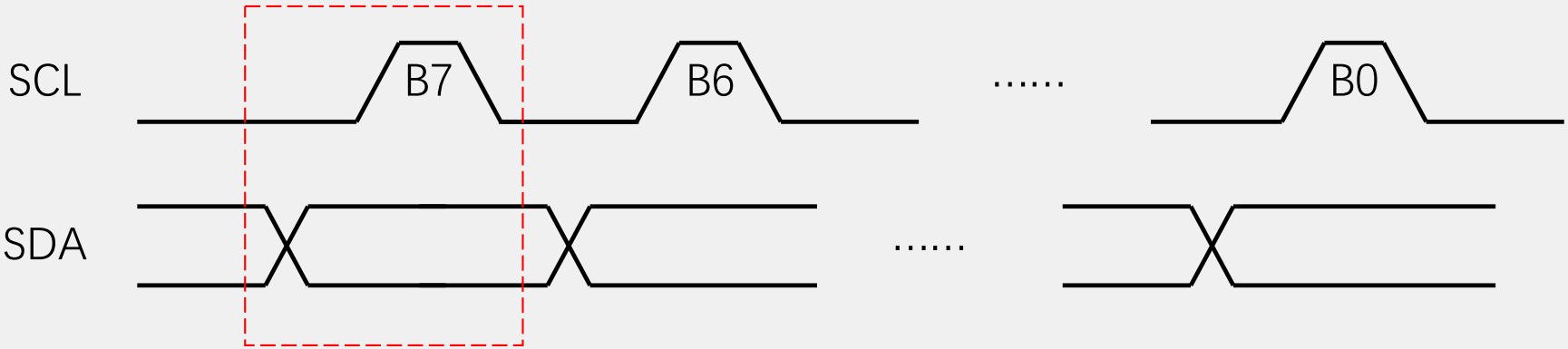


I2C时序基本单元

- 起始条件：SCL高电平期间，SDA从高电平切换到低电平
- 终止条件：SCL高电平期间，SDA从低电平切换到高电平

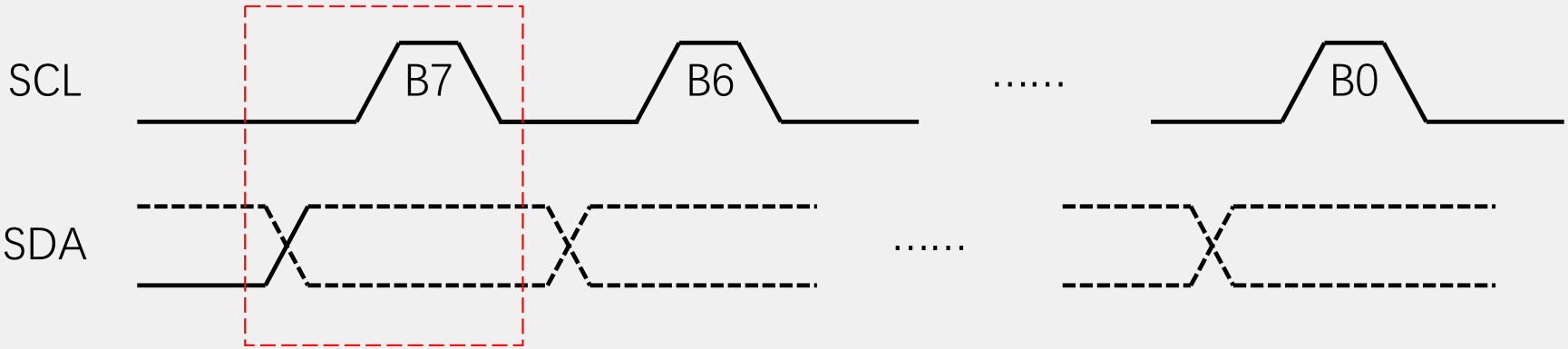
I2C时序基本单元

- 发送一个字节：SCL低电平期间，主机将数据位依次放到SDA线上（高位先行），然后释放SCL，从机将在SCL高电平期间读取数据位，所以SCL高电平期间SDA不允许有数据变化，依次循环上述过程8次，即可发送一个字节



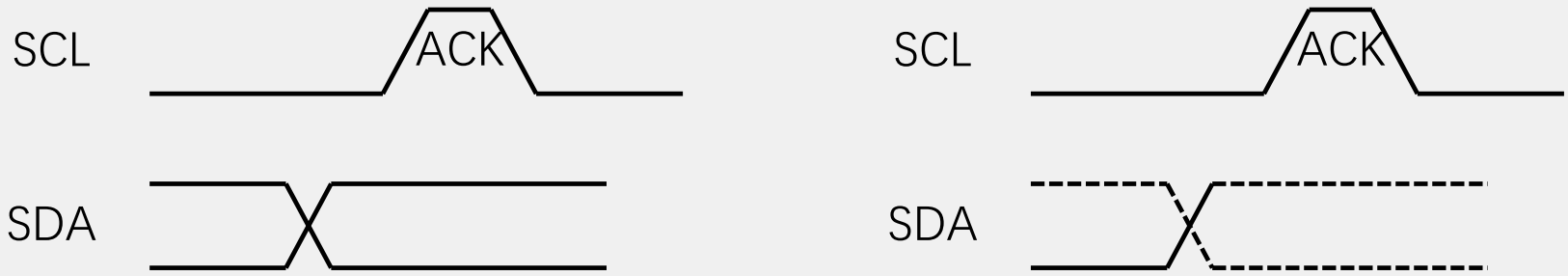
I2C时序基本单元

- 接收一个字节：SCL低电平期间，从机将数据位依次放到SDA线上（高位先行），然后释放SCL，主机将在SCL高电平期间读取数据位，所以SCL高电平期间SDA不允许有数据变化，依次循环上述过程8次，即可接收一个字节（主机在接收之前，需要释放SDA）



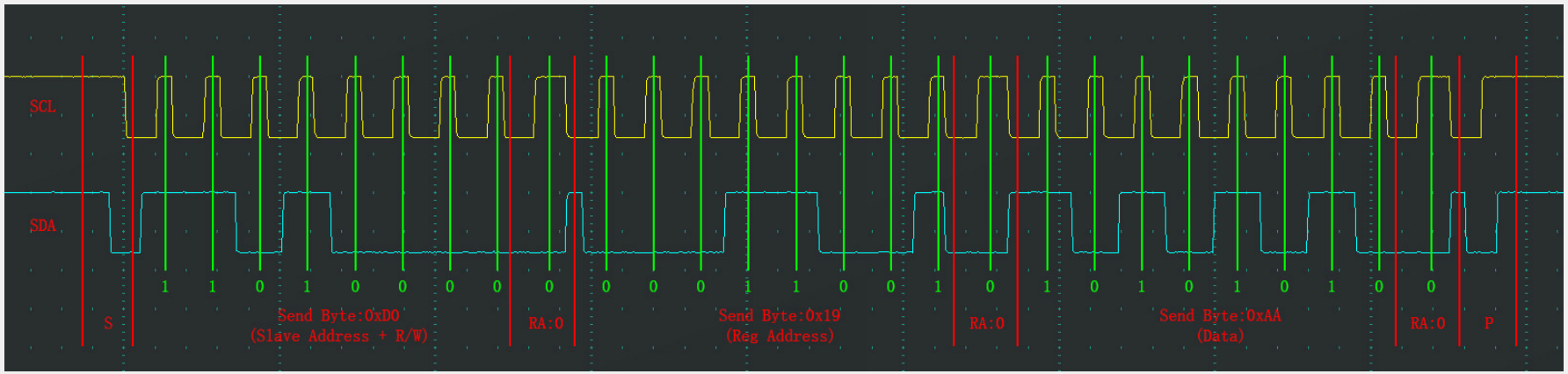
I2C时序基本单元

- 发送应答：主机在接收完一个字节之后，在下一个时钟发送一位数据，数据0表示应答，数据1表示非应答
- 接收应答：主机在发送完一个字节之后，在下一个时钟接收一位数据，判断从机是否应答，数据0表示应答，数据1表示非应答（主机在接收之前，需要释放SDA）



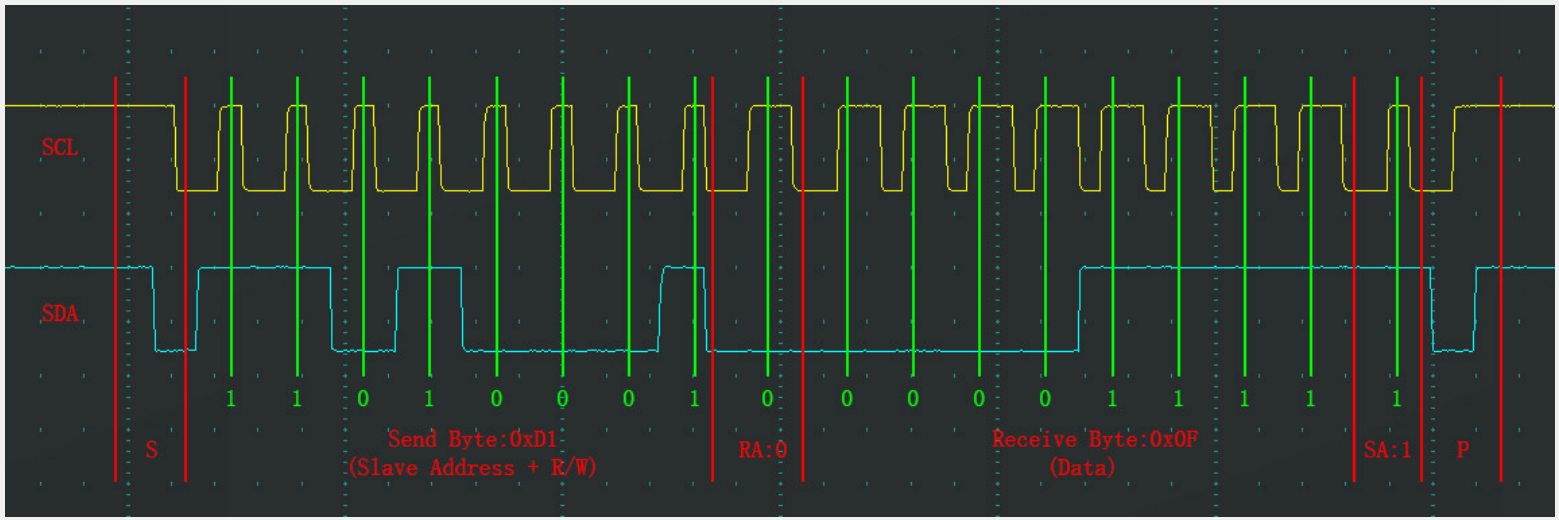
I2C时序

- 指定地址写
- 对于指定设备（Slave Address），在指定地址（Reg Address）下，写入指定数据（Data）



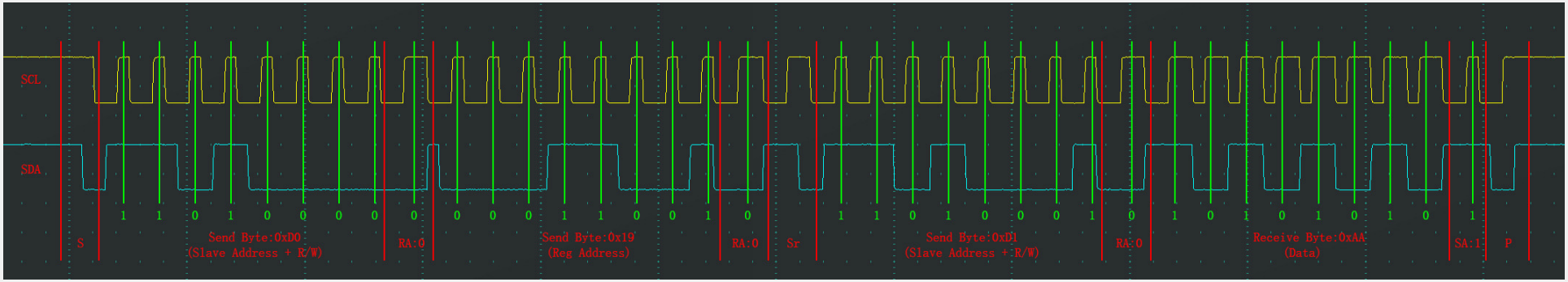
I2C时序

- 当前地址读
- 对于指定设备（Slave Address），在当前地址指针指示的地址下，读取从机数据（Data）



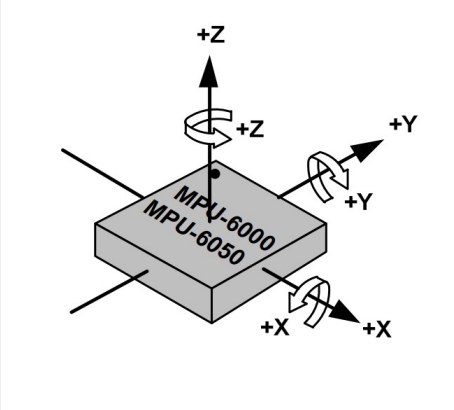
I2C时序

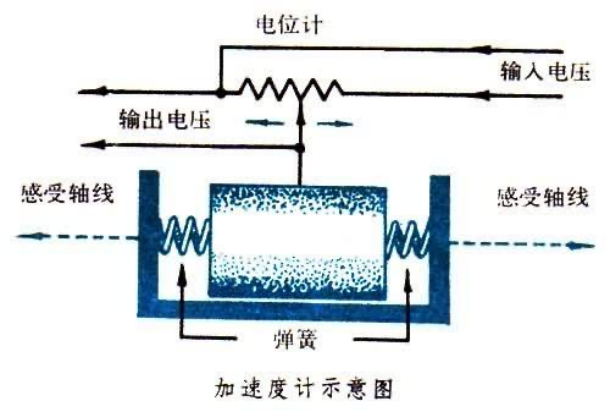
- 指定地址读
- 对于指定设备（Slave Address），在指定地址（Reg Address）下，读取从机数据（Data）



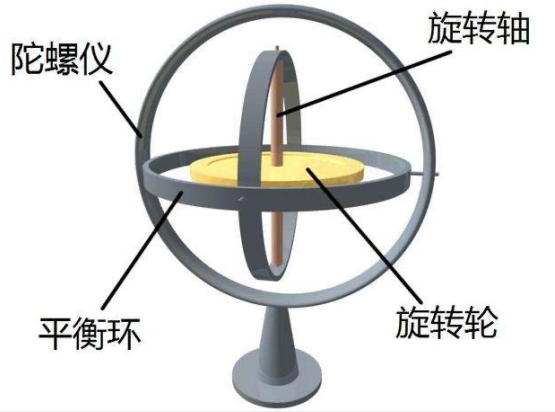
MPU6050简介

- MPU6050是一个6轴姿态传感器，可以测量芯片自身X、Y、Z轴的加速度、角速度参数，通过数据融合，可进一步得到姿态角，常应用于平衡车、飞行器等需要检测自身姿态的场景
- 3轴加速度计（Accelerometer）：测量X、Y、Z轴的加速度
- 3轴陀螺仪传感器（Gyroscope）：测量X、Y、Z轴的角速度





加速度计示意图

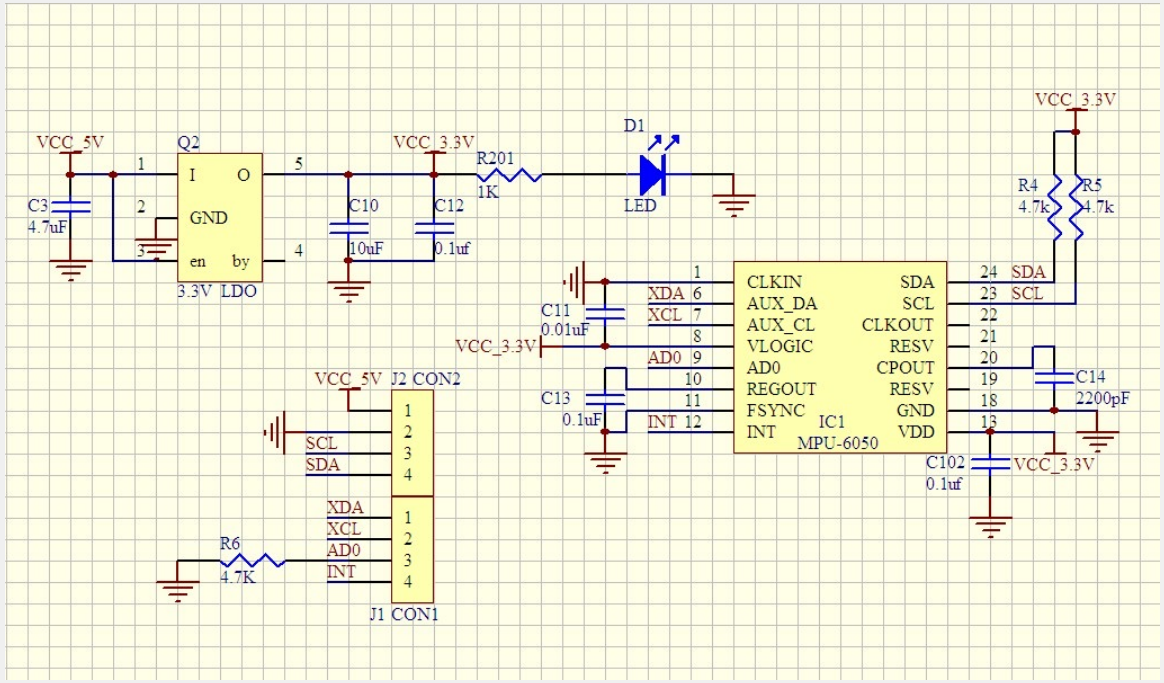


MPU6050参数

- 16位ADC采集传感器的模拟信号，量化范围：-32768~32767
- 加速度计满量程选择：±2、±4、±8、±16（g）
- 陀螺仪满量程选择： ±250、±500、±1000、±2000（°/sec）
- 可配置的数字低通滤波器
- 可配置的时钟源
- 可配置的采样分频

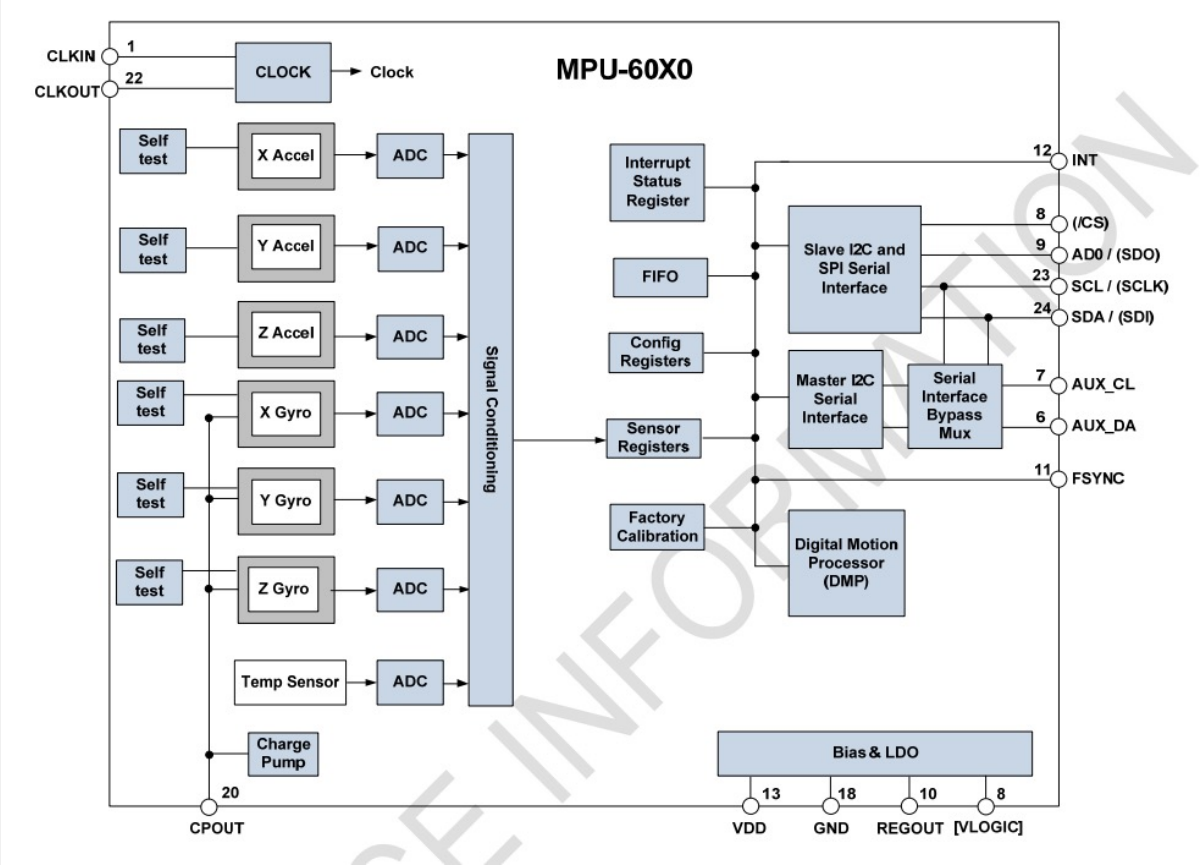
- I2C从机地址：1101000（AD0=0）
  1101001（AD0=1）

硬件电路



引脚	功能
VCC、GND	电源
SCL、SDA	I2C通信引脚
XCL、XDA	主机I2C通信引脚
AD0	从机地址最低位
INT	中断信号输出

MPU6050框图


MPU-60X0
CLKIN
CLKOUT
1
22
CLOCK
Clock
Self test
X Accel
Y Accel
Z Accel
X Gyro
Y Gyro
Z Gyro
ADC
Temp Sensor
Signal Conditioning
Charge Pump
20
CPOUT
Interrupt Status Register
FIFO
Config Registers
Sensor Registers
Factory Calibration
Slave I2C and SPI Serial Interface
Master I2C Serial Interface
Serial Interface Bypass Mux
Digital Motion Processor (DMP)
12
INT
8
(/CS)
9
AD0 / (SDO)
23
SCL / (SCLK)
24
SDA / (SDI)
7
AUX_CL
6
AUX_DA
11
FSYNC
Bias & LDO
13
18
10
8
VDD
GND
REGOUT
[VLOGIC]

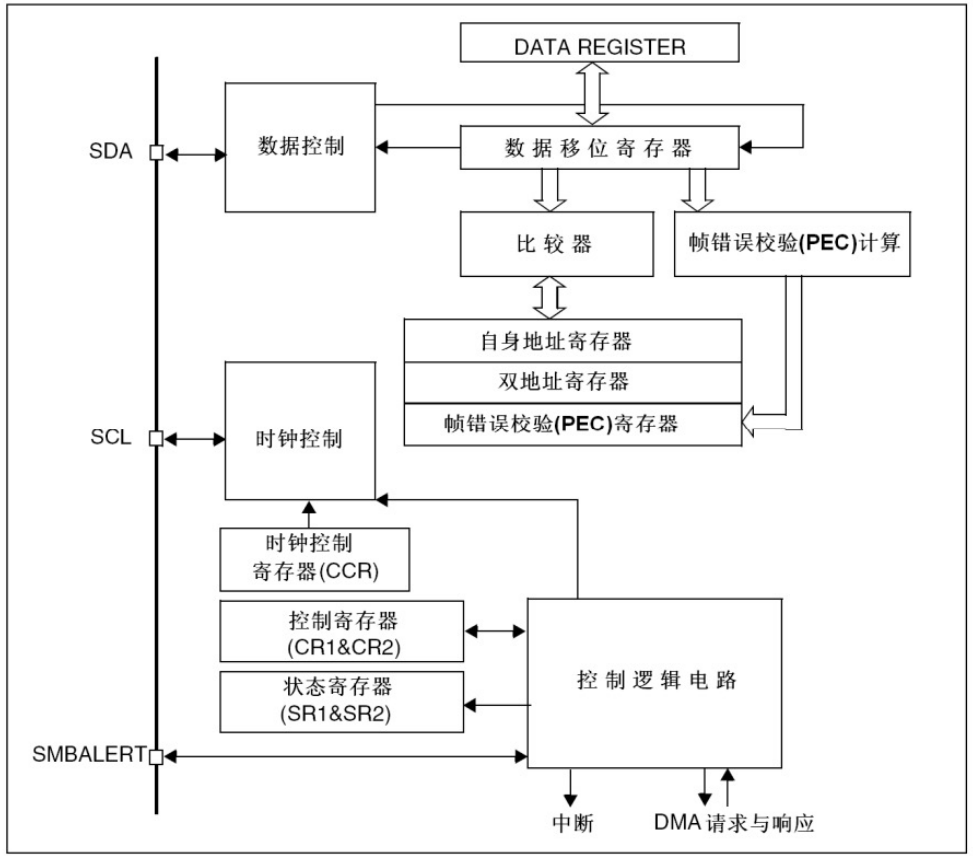
I2C外设简介

- STM32内部集成了硬件I2C收发电路，可以由硬件自动执行时钟生成、起始终止条件生成、应答位收发、数据收发等功能，减轻CPU的负担
- 支持多主机模型
- 支持7位/10位地址模式
- 支持不同的通讯速度，标准速度(高达100 kHz)，快速(高达400 kHz)
- 支持DMA
- 兼容SMBus协议

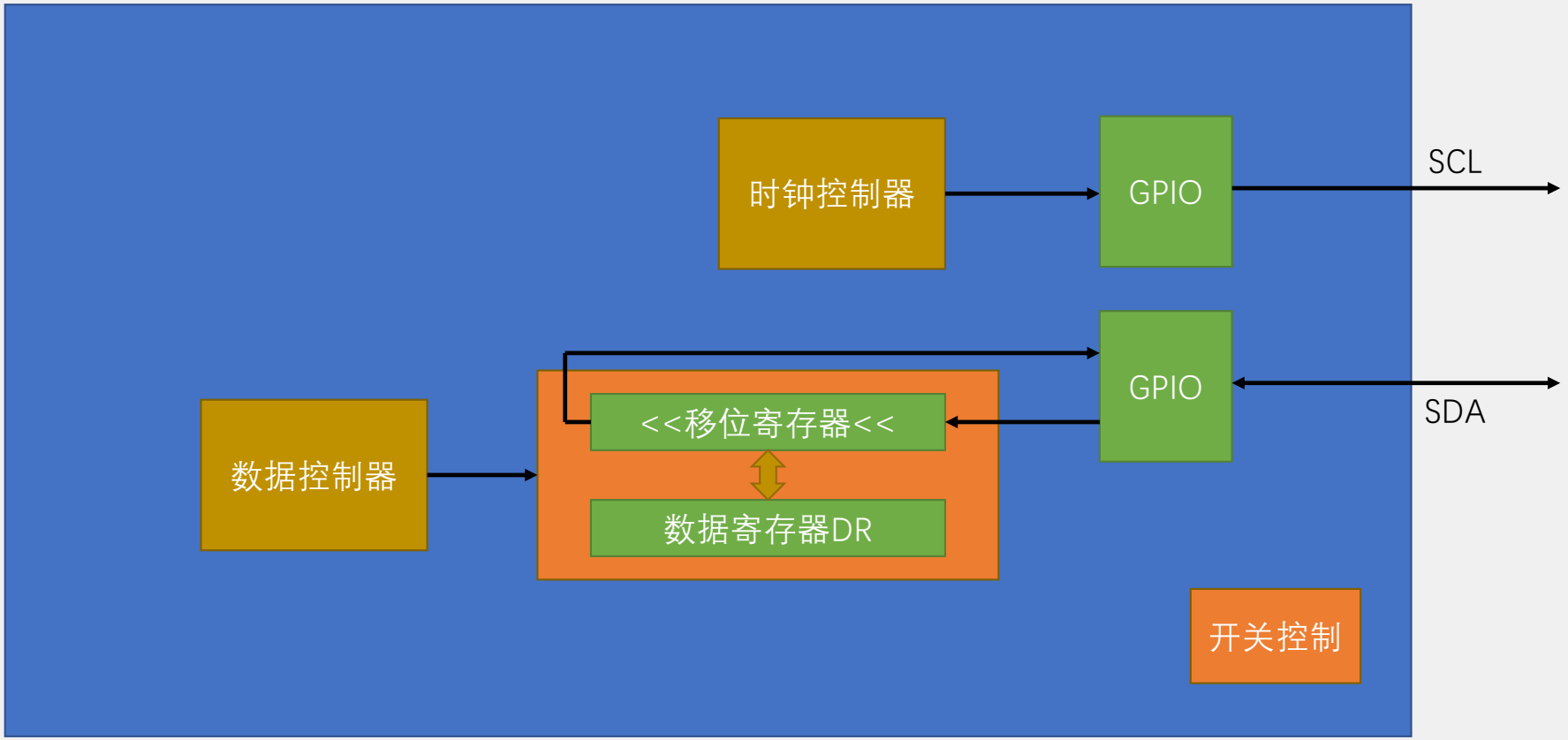
- STM32F103C8T6 硬件I2C资源：I2C1、I2C2

I2C框图

图242　I²C的功能框图

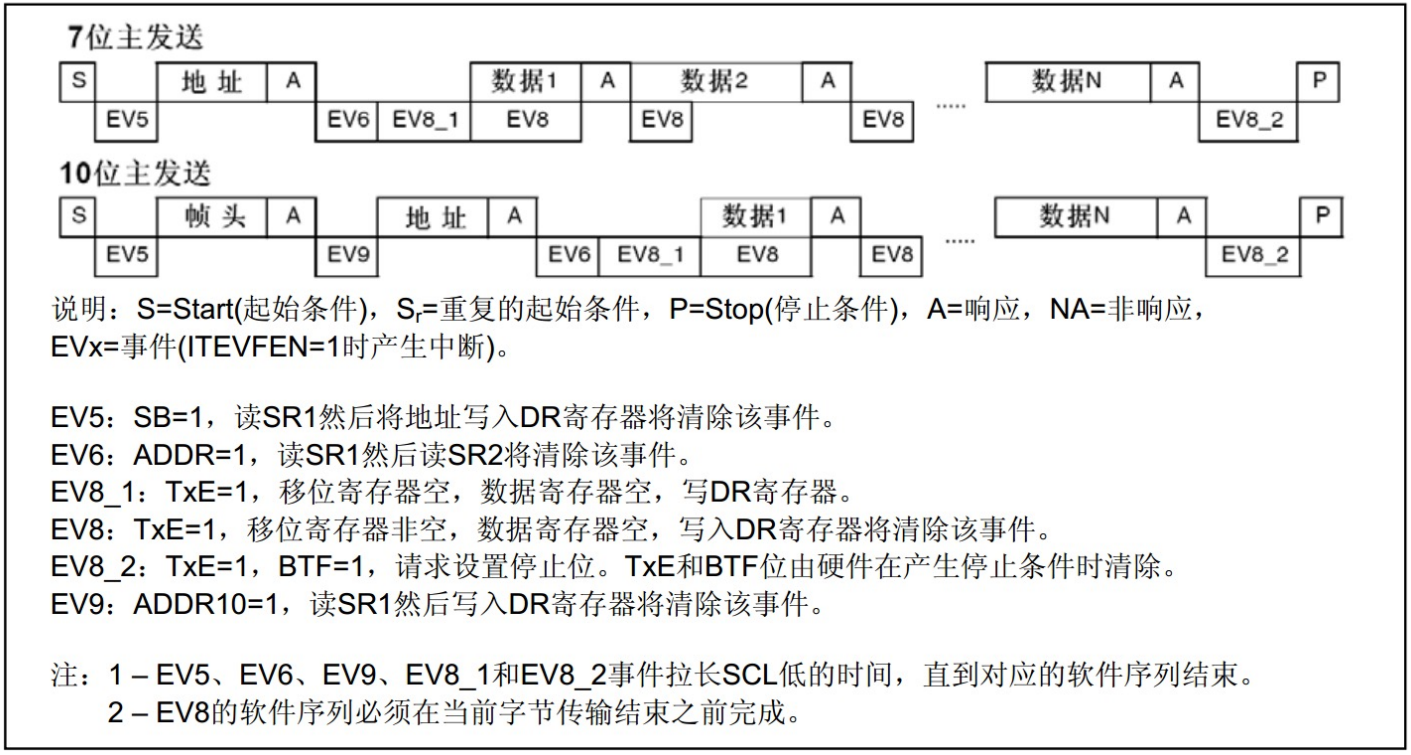


I2C基本结构



主机发送

图245　主发送器传送序列图



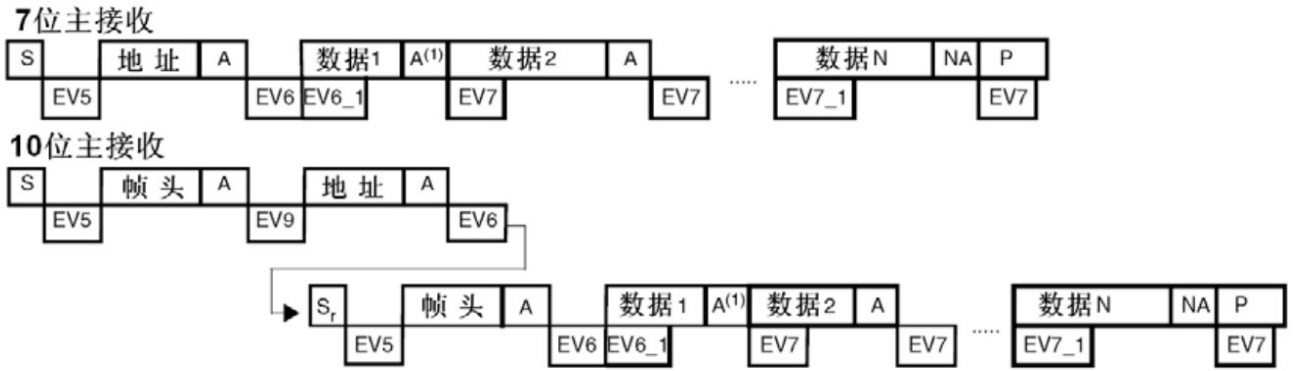
说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，
EVx=事件(ITEVFEN=1时产生中断)。

EV5：SB=1，读SR1然后将地址写入DR寄存器将清除该事件。
EV6：ADDR=1，读SR1然后读SR2将清除该事件。
EV8_1：TxE=1，移位寄存器空，数据寄存器空，写DR寄存器。
EV8：TxE=1，移位寄存器非空，数据寄存器空，写入DR寄存器将清除该事件。
EV8_2：TxE=1，BTF=1，请求设置停止位。TxE和BTF位由硬件在产生停止条件时清除。
EV9：ADDR10=1，读SR1然后写入DR寄存器将清除该事件。

注：1 – EV5、EV6、EV9、EV8_1和EV8_2事件拉长SCL低的时间，直到对应的软件序列结束。
　　2 – EV8的软件序列必须在当前字节传输结束之前完成。

主机接收

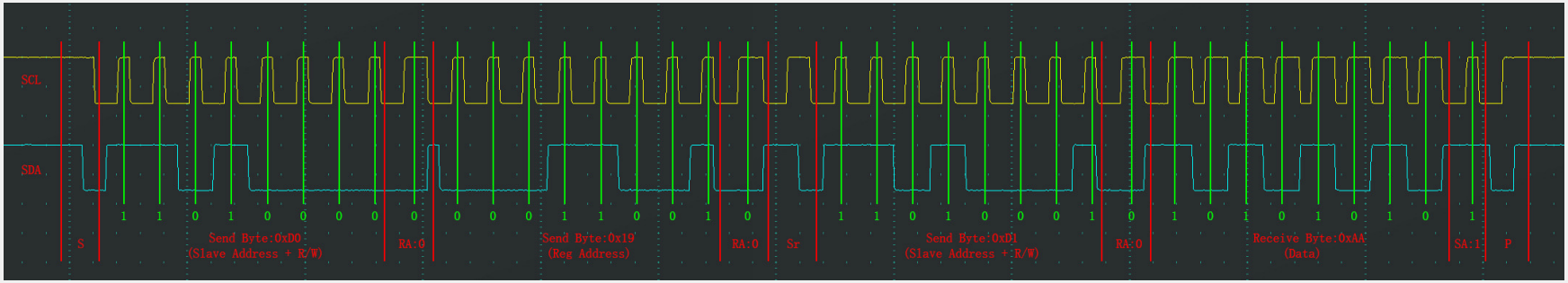
图246　主接收器传送序列图

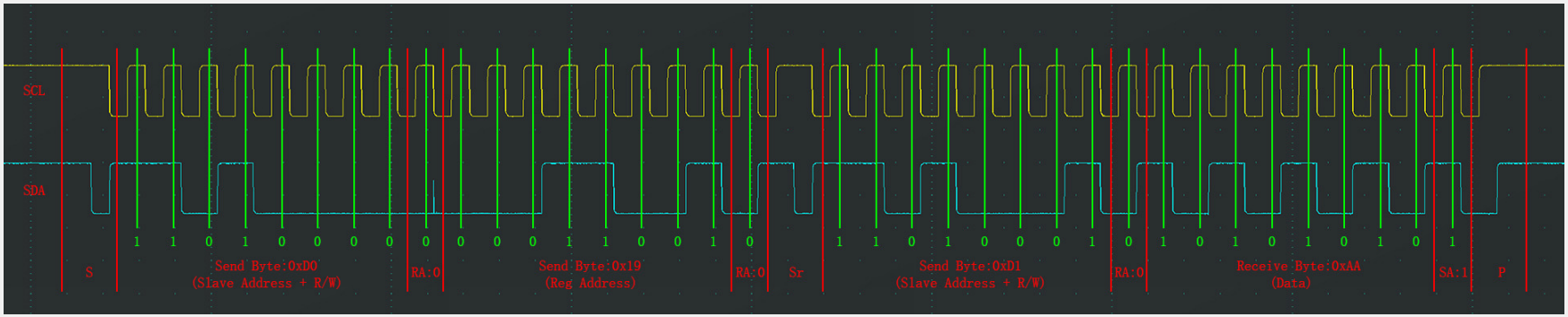


说明：S=Start(起始条件)，S_r=重复的起始条件，P=Stop(停止条件)，A=响应，NA=非响应，
EVx=事件(ITEVFEN=1时产生中断)

EV5：SB=1，读SR1然后将地址写入DR寄存器将清除该事件。
EV6：ADDR=1，读SR1然后读SR2将清除该事件。在10位主接收模式下，该事件后应设置CR2的START=1。
EV6_1：没有对应的事件标志，只适于接收1个字节的情况。恰好在EV6之后(即清除了ADDR之后)，要清除响应和停止条件的产生位。
EV7：RxNE=1，读DR寄存器清除该事件。
EV7_1：RxNE=1，读DR寄存器清除该事件。设置ACK=0和STOP请求。
EV9：ADDR10=1，读SR1然后写入DR寄存器将清除该事件。

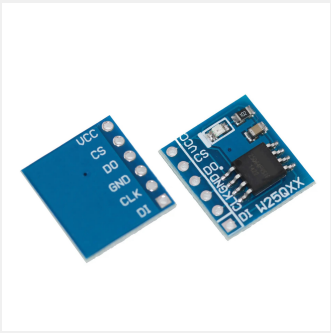
软件/硬件波形对比


SCL
SDA
1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1
S
Send Byte:0xD0
(Slave Address + R/W)
RA:0
Send Byte:0x19
(Reg Address)
RA:0
Sr
Send Byte:0xD1
(Slave Address + R/W)
RA:0
Receive Byte:0xAA
(Data)
SA:1
P


SCL
SDA
1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1
S
Send Byte:0xD0
(Slave Address + R/W)
RA:0
Send Byte:0x19
(Reg Address)
RA:0
Sr
Send Byte:0xD1
(Slave Address + R/W)
RA:0
Receive Byte:0xAA
(Data)
SA:1
P

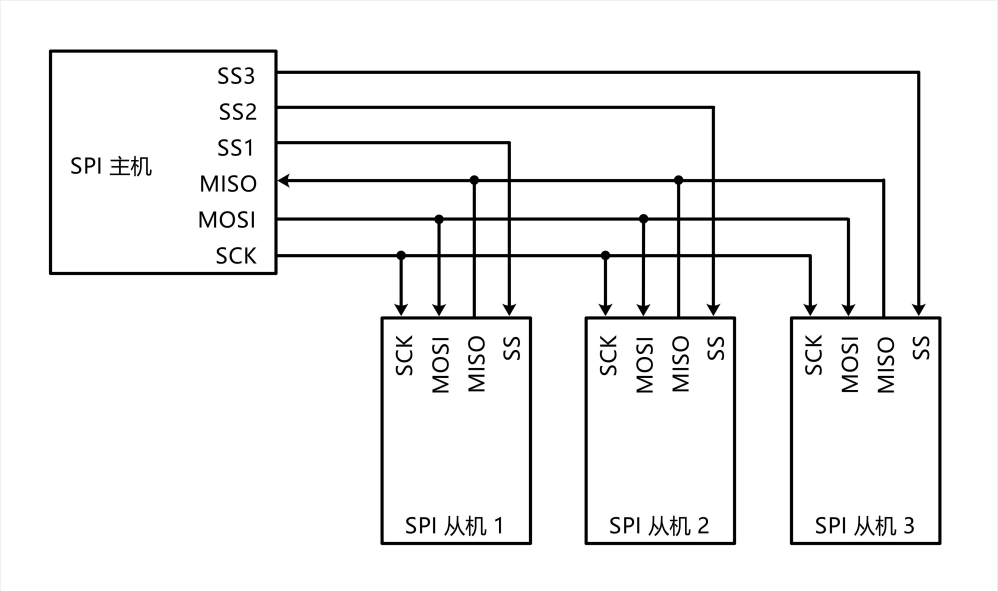
SPI通信

- SPI（Serial Peripheral Interface）是由Motorola公司开发的一种通用数据总线
- 四根通信线：SCK（Serial Clock）、MOSI（Master Output Slave Input）、MISO（Master Input Slave Output）、SS（Slave Select）
- 同步，全双工
- 支持总线挂载多设备（一主多从）

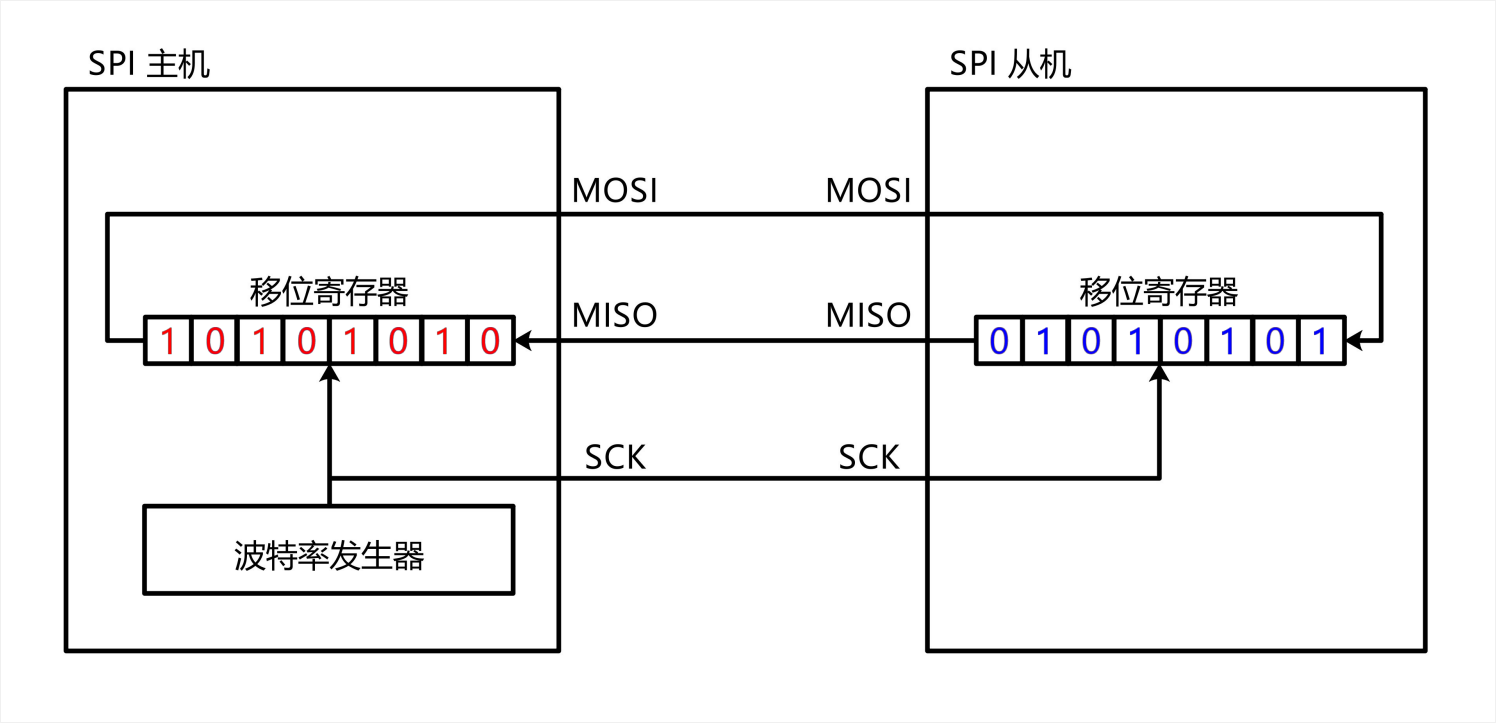


硬件电路

- 所有SPI设备的SCK、MOSI、MISO分别连在一起
- 主机另外引出多条SS控制线，分别接到各从机的SS引脚
- 输出引脚配置为推挽输出，输入引脚配置为浮空或上拉输入

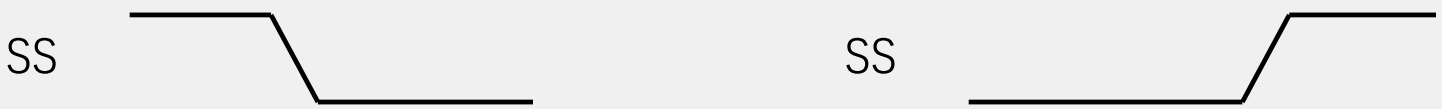


移位示意图


SPI 主机
SPI 从机
MOSI
MOSI
移位寄存器
MISO
MISO
移位寄存器
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
SCK
SCK
波特率发生器

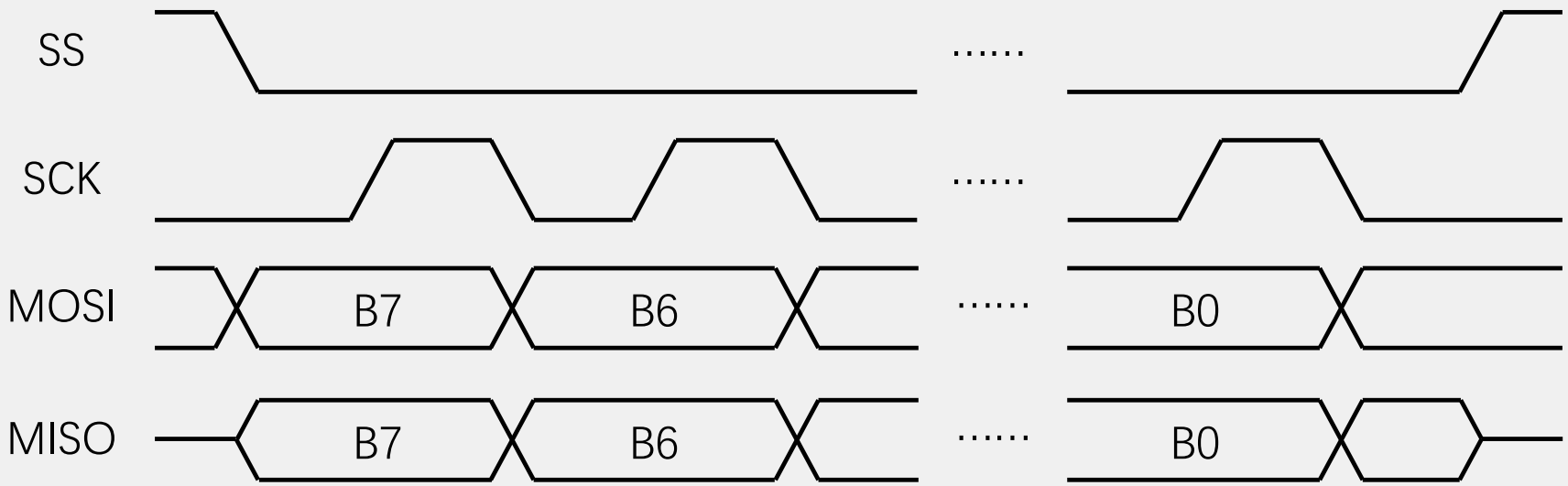
SPI时序基本单元

- 起始条件：SS从高电平切换到低电平
- 终止条件：SS从低电平切换到高电平



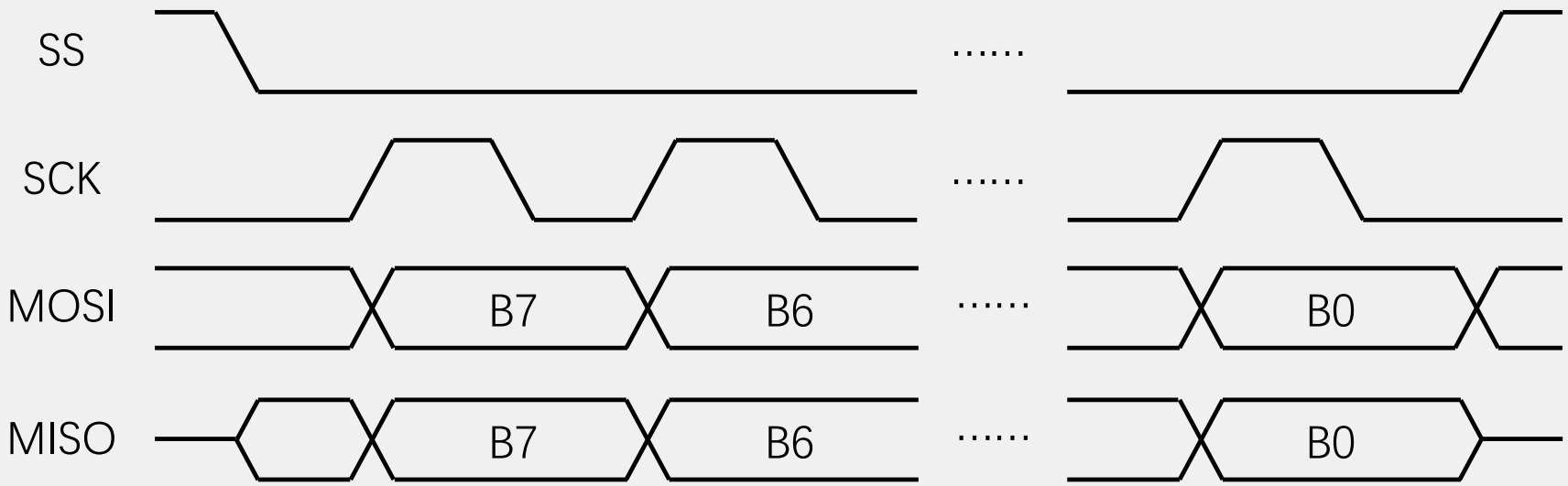
SPI时序基本单元

- 交换一个字节（模式0）
- CPOL=0：空闲状态时，SCK为低电平
- CPHA=0：SCK第一个边沿移入数据，第二个边沿移出数据



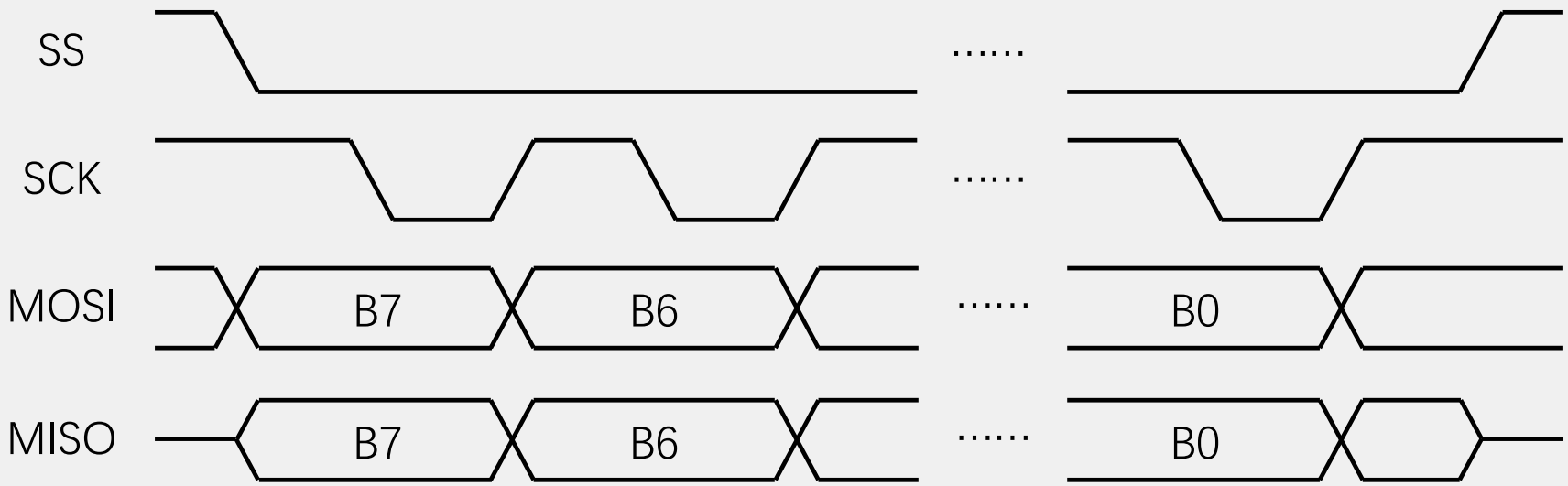
SPI时序基本单元

- 交换一个字节（模式1）
- CPOL=0：空闲状态时，SCK为低电平
- CPHA=1：SCK第一个边沿移出数据，第二个边沿移入数据



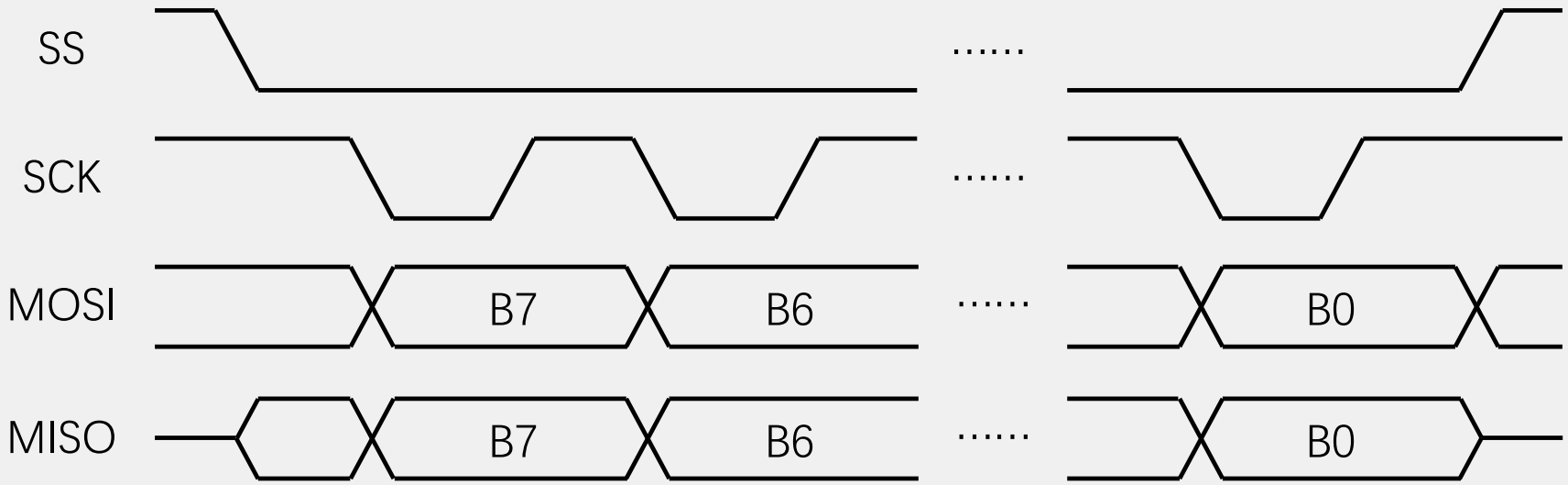
SPI时序基本单元

- 交换一个字节（模式2）
- CPOL=1：空闲状态时，SCK为高电平
- CPHA=0：SCK第一个边沿移入数据，第二个边沿移出数据



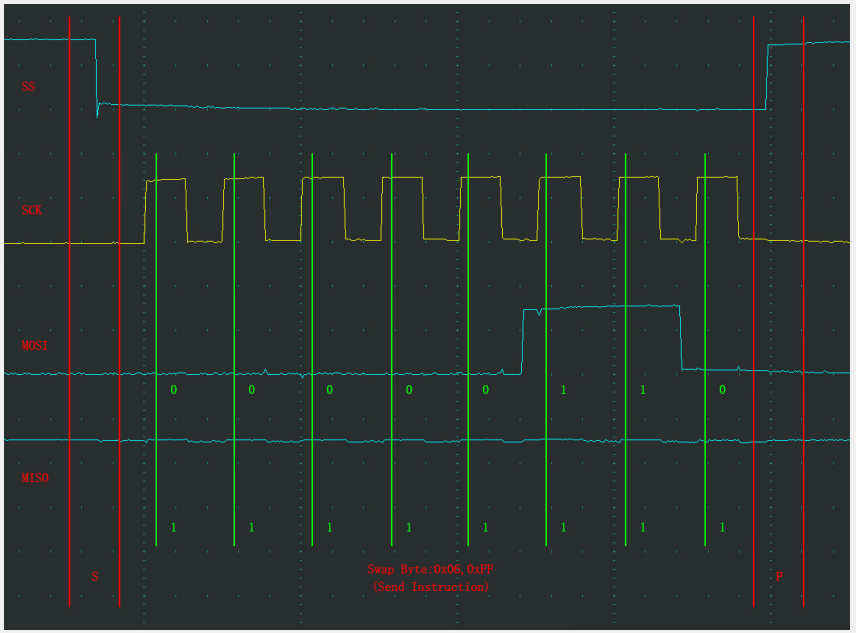
SPI时序基本单元

- 交换一个字节（模式3）
- CPOL=1：空闲状态时，SCK为高电平
- CPHA=1：SCK第一个边沿移出数据，第二个边沿移入数据



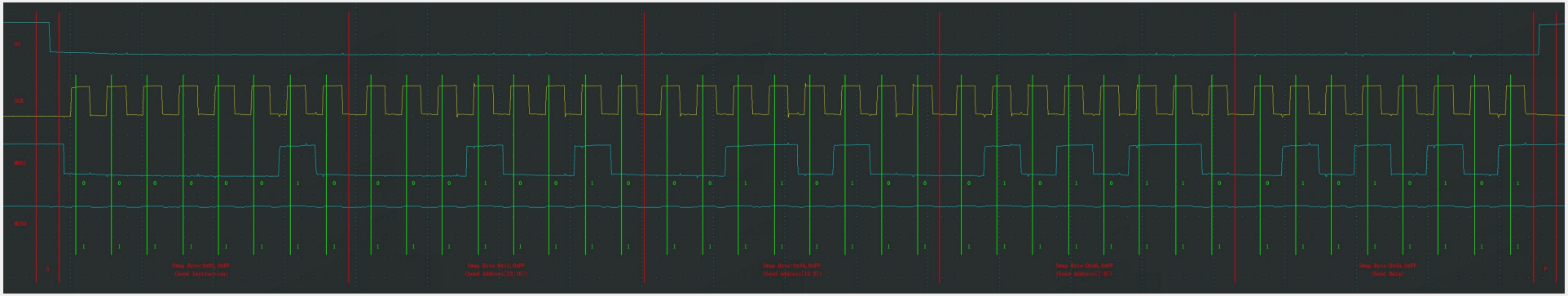
SPI时序

- 发送指令
- 向SS指定的设备，发送指令（0x06）



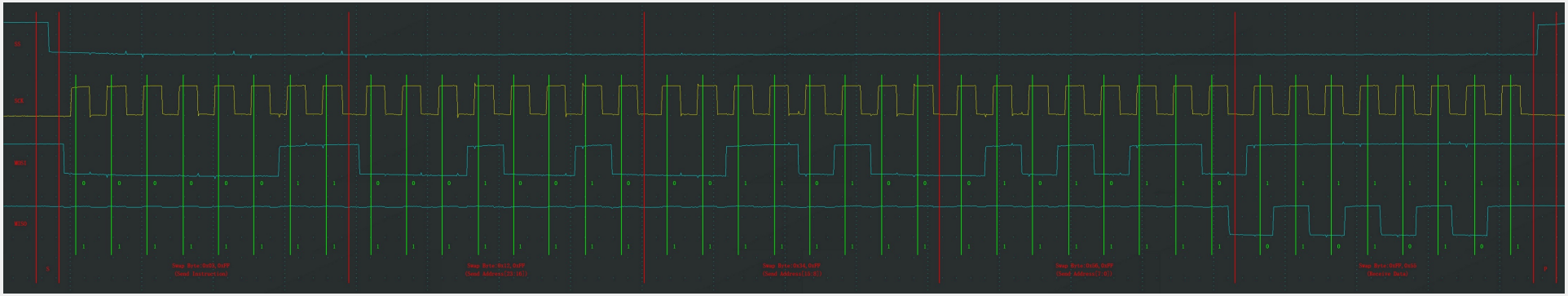
SPI时序

- 指定地址写
- 向SS指定的设备，发送写指令（0x02），
  随后在指定地址（Address[23:0]）下，写入指定数据（Data）



SPI时序

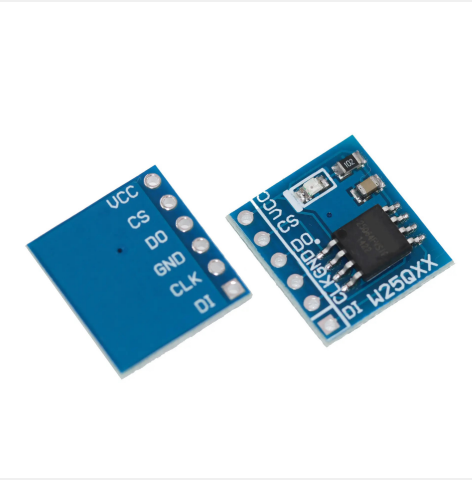
- 指定地址读
- 向SS指定的设备，发送读指令（0x03），
  随后在指定地址（Address[23:0]）下，读取从机数据（Data）



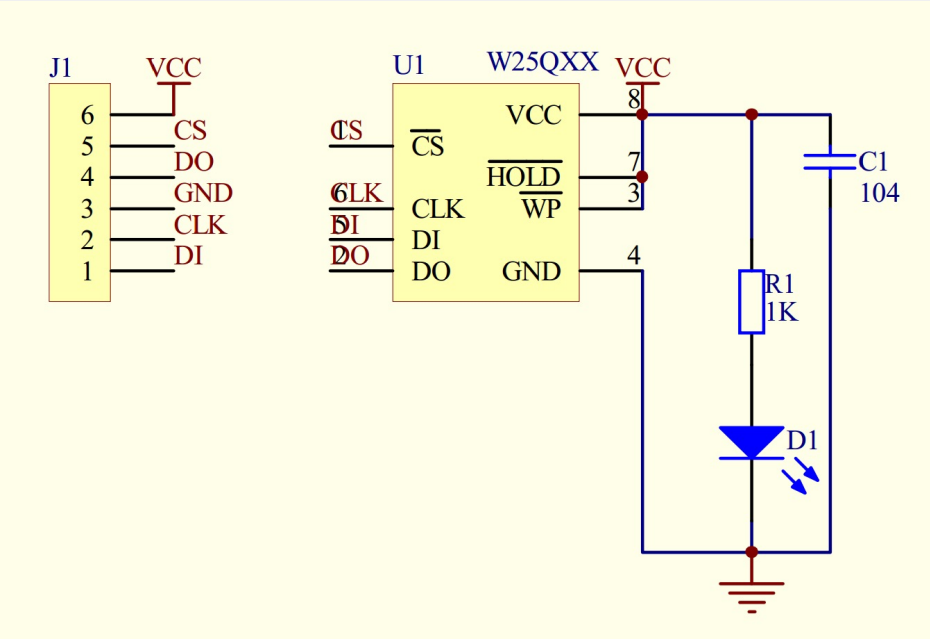
W25Q64简介

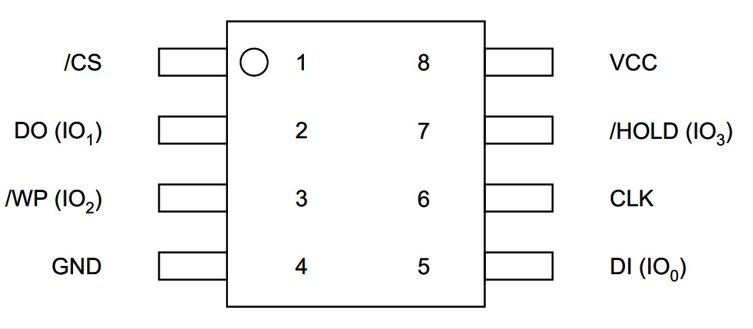
- W25Qxx系列是一种低成本、小型化、使用简单的非易失性存储器，常应用于数据存储、字库存储、固件程序存储等场景
- 存储介质：Nor Flash（闪存）
- 时钟频率：80MHz / 160MHz (Dual SPI) / 320MHz (Quad SPI)
- 存储容量（24位地址）：

  W25Q40： 4Mbit / 512KByte
  W25Q80： 8Mbit / 1MByte
  W25Q16： 16Mbit / 2MByte
  W25Q32： 32Mbit / 4MByte
  W25Q64： 64Mbit / 8MByte
  W25Q128： 128Mbit / 16MByte
  W25Q256： 256Mbit / 32MByte



硬件电路





引脚	功能
VCC、GND	电源（2.7~3.6V）
CS（SS）	SPI片选
CLK（SCK）	SPI时钟
DI（MOSI）	SPI主机输出从机输入
DO（MISO）	SPI主机输入从机输出
WP	写保护
HOLD	数据保持

W25Q64框图

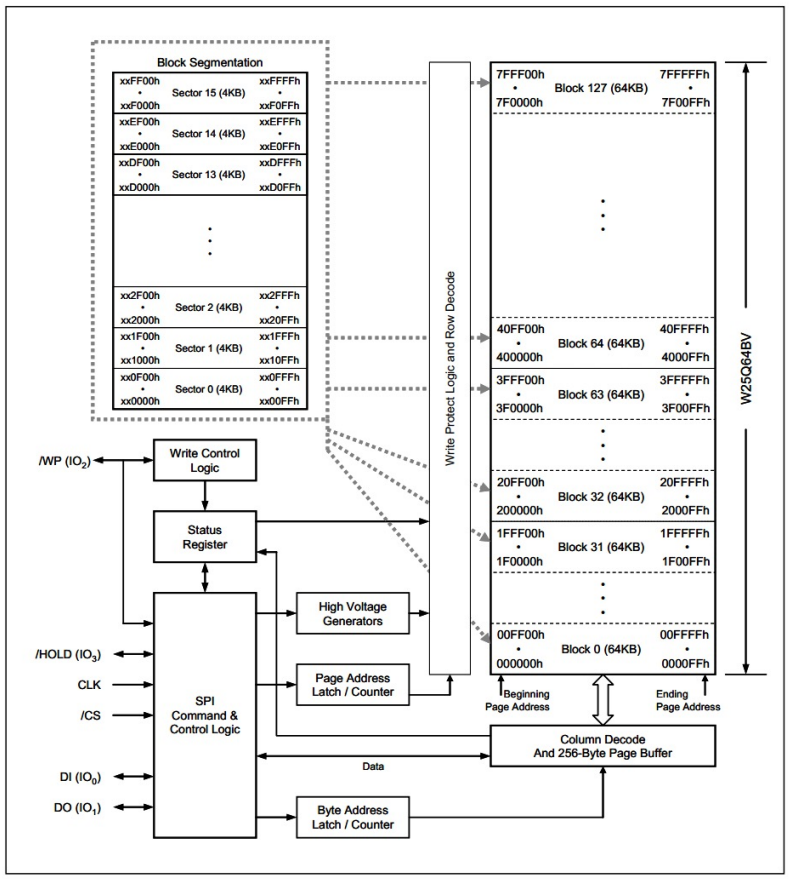


Figure 2. W25Q64BV Serial Flash Memory Block Diagram

Flash操作注意事项

写入操作时：

- 写入操作前，必须先进行写使能
- 每个数据位只能由1改写为0，不能由0改写为1
- 写入数据前必须先擦除，擦除后，所有数据位变为1
- 擦除必须按最小擦除单元进行
- 连续写入多字节时，最多写入一页的数据，超过页尾位置的数据，会回到页首覆盖写入
- 写入操作结束后，芯片进入忙状态，不响应新的读写操作

读取操作时：

- 直接调用读取时序，无需使能，无需额外操作，没有页的限制，读取操作结束后不会进入忙状态，但不能在忙状态时读取

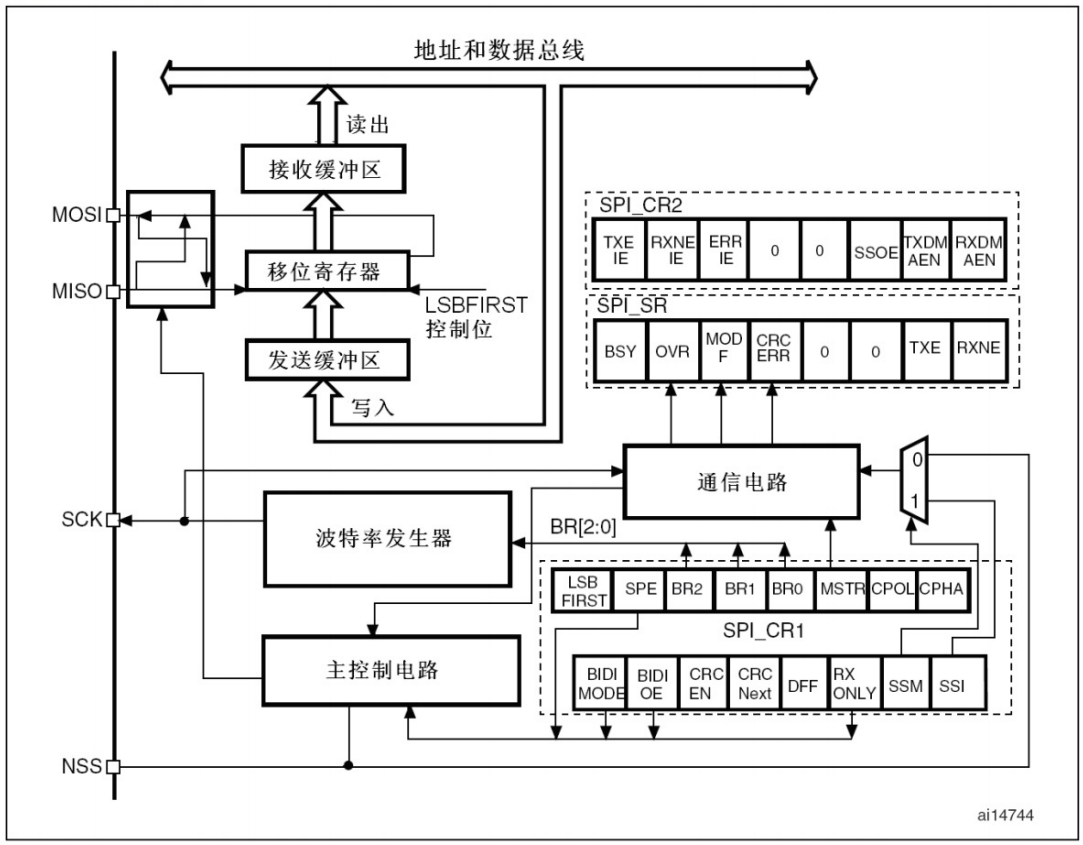
SPI外设简介

- STM32内部集成了硬件SPI收发电路，可以由硬件自动执行时钟生成、数据收发等功能，减轻CPU的负担
- 可配置8位/16位数据帧、高位先行/低位先行
- 时钟频率：f_{PCLK} / (2, 4, 8, 16, 32, 64, 128, 256)
- 支持多主机模型、主或从操作
- 可精简为半双工/单工通信
- 支持DMA
- 兼容I2S协议

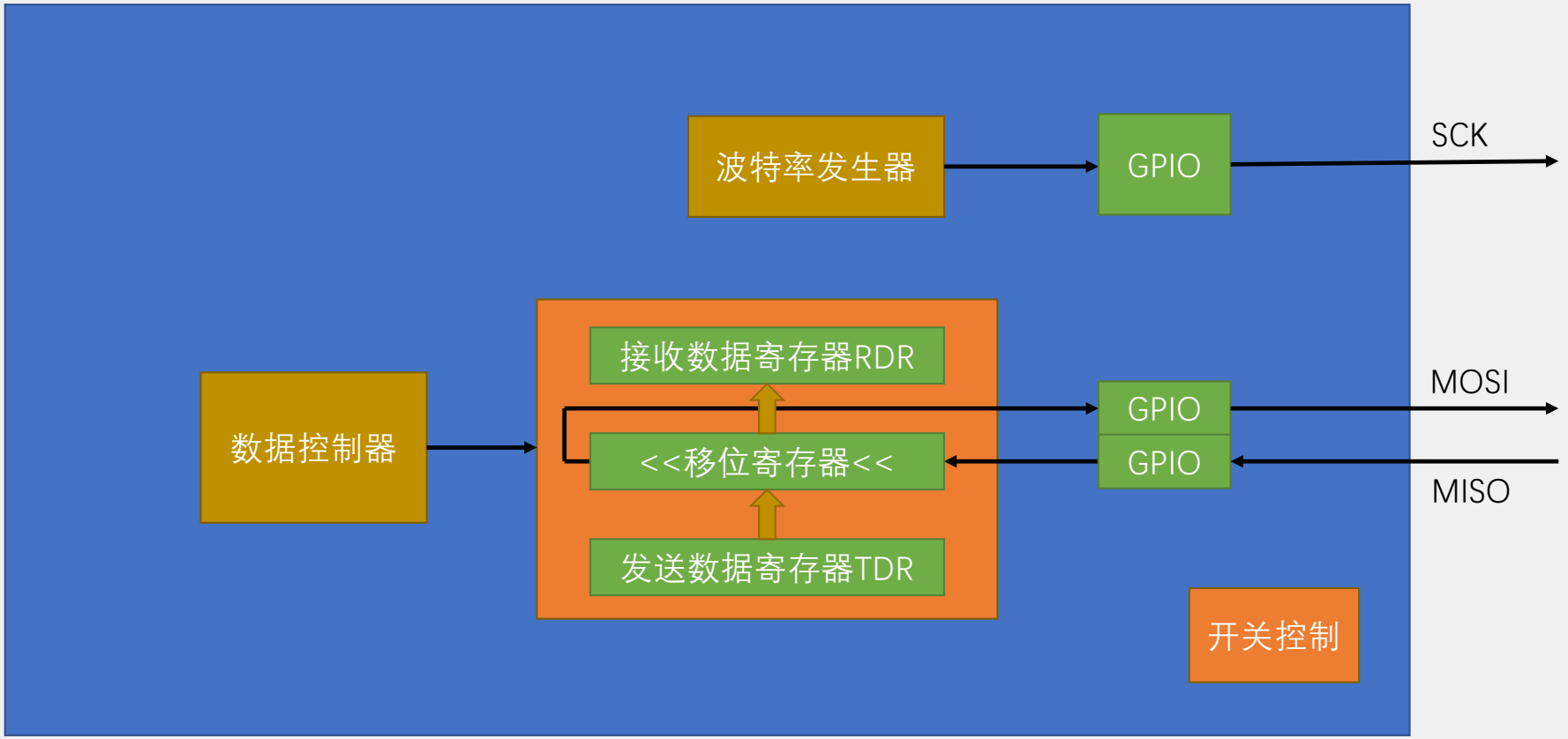
- STM32F103C8T6 硬件SPI资源：SPI1、SPI2

SPI框图

图209　SPI框图

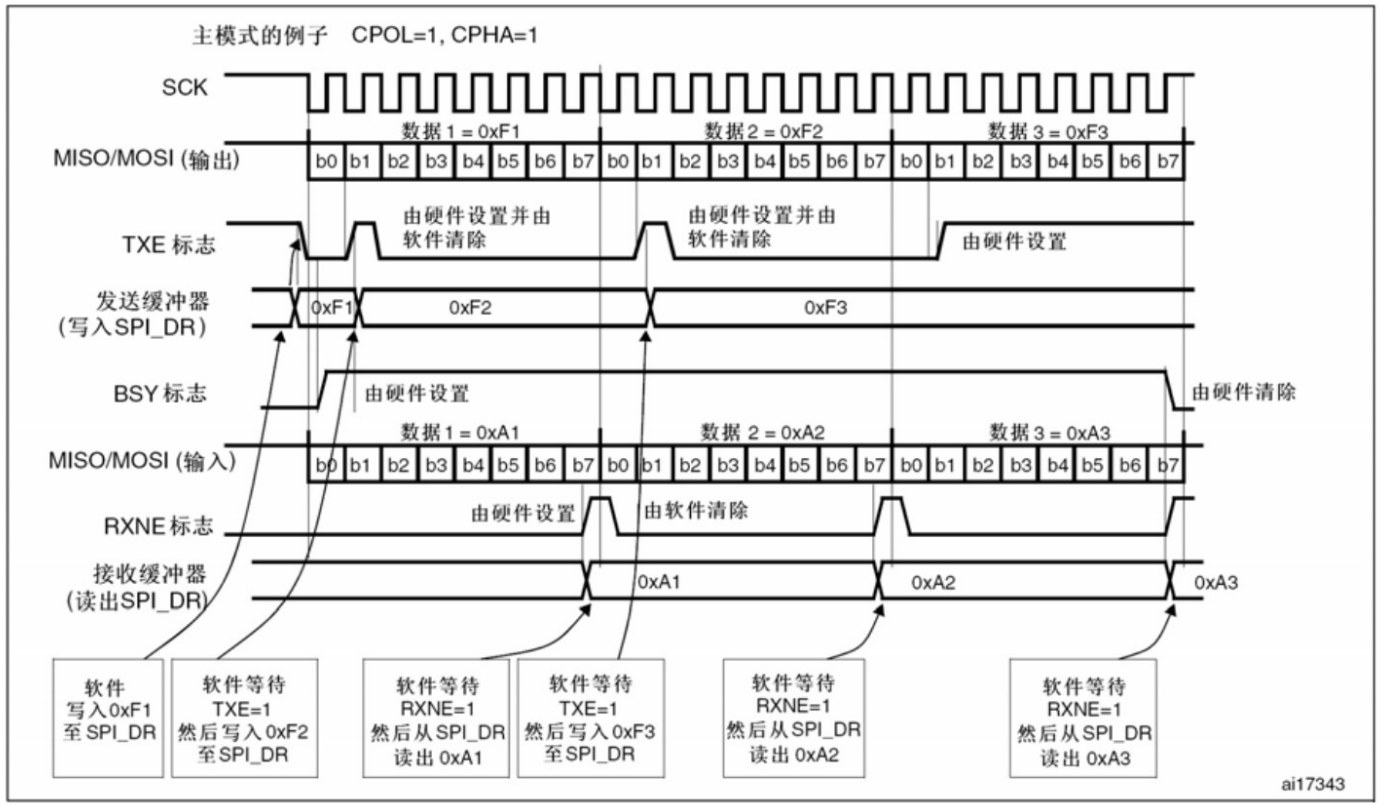


SPI基本结构


波特率发生器
GPIO
SCK
接收数据寄存器RDR
数据控制器
GPIO
MOSI
<<移位寄存器<<
GPIO
MISO
发送数据寄存器TDR
开关控制

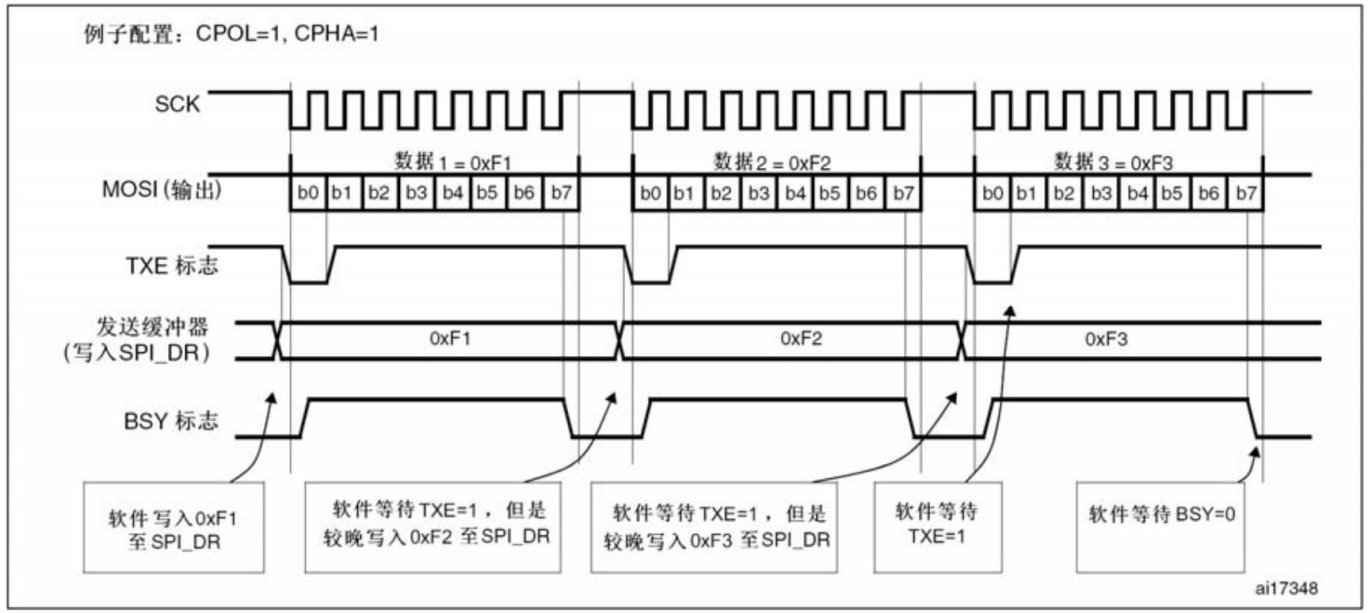
主模式全双工连续传输

图213　主模式、全双工模式下(BIDIMODE=0并且RXONLY=0)连续传输时，TXE/RXNE/BSY的变化示意图

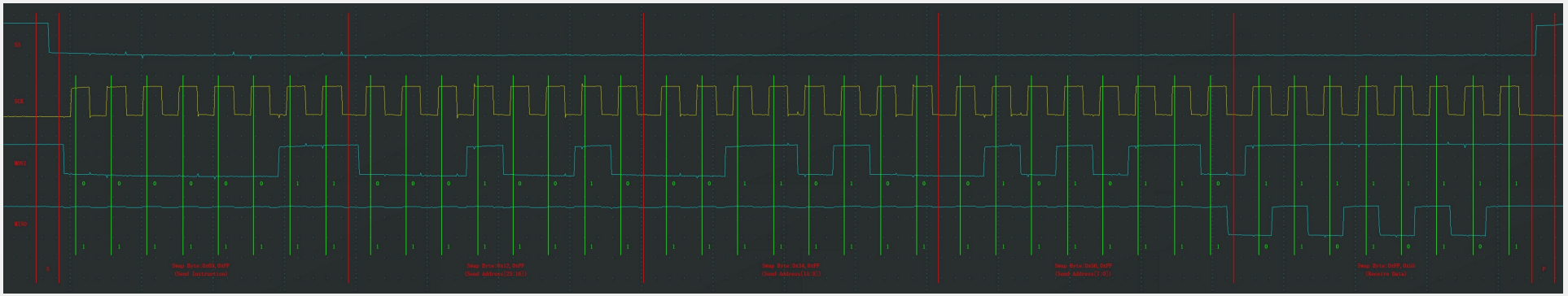


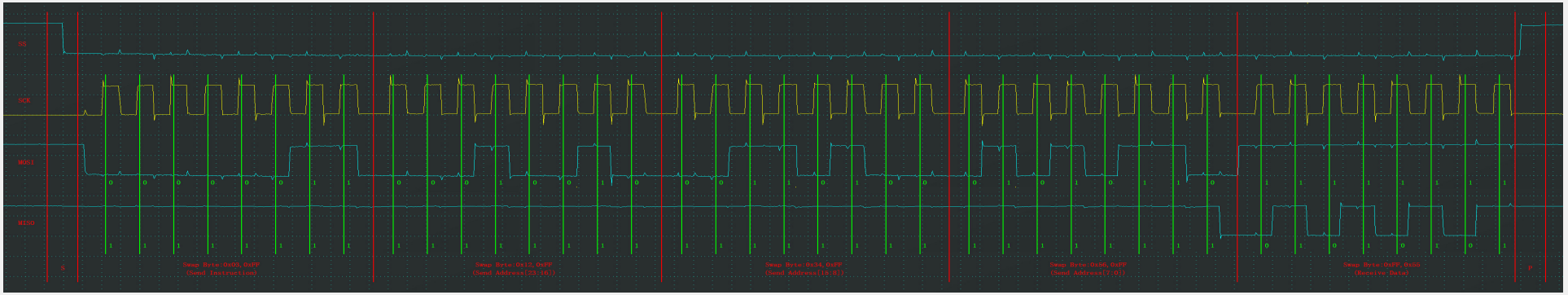
非连续传输

图218　非连续传输发送(BIDIMODE=0并且RXONLY=0)时，TXE/BSY变化示意图



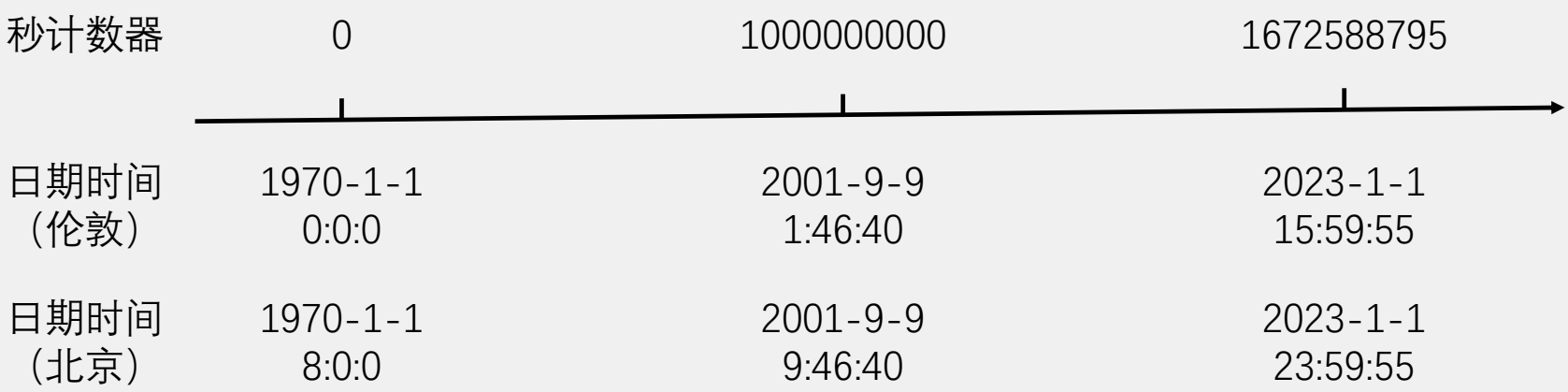
软件/硬件波形对比





Unix时间戳

- Unix 时间戳（Unix Timestamp）定义为从UTC/GMT的1970年1月1日0时0分0秒开始所经过的秒数，不考虑闰秒
- 时间戳存储在一个秒计数器中，秒计数器为32位/64位的整型变量
- 世界上所有时区的秒计数器相同，不同时区通过添加偏移来得到当地时间



UTC/GMT

- GMT（Greenwich Mean Time）格林尼治标准时间是一种以地球自转为基础的时间计量系统。它将地球自转一周的时间间隔等分为24小时，以此确定计时标准

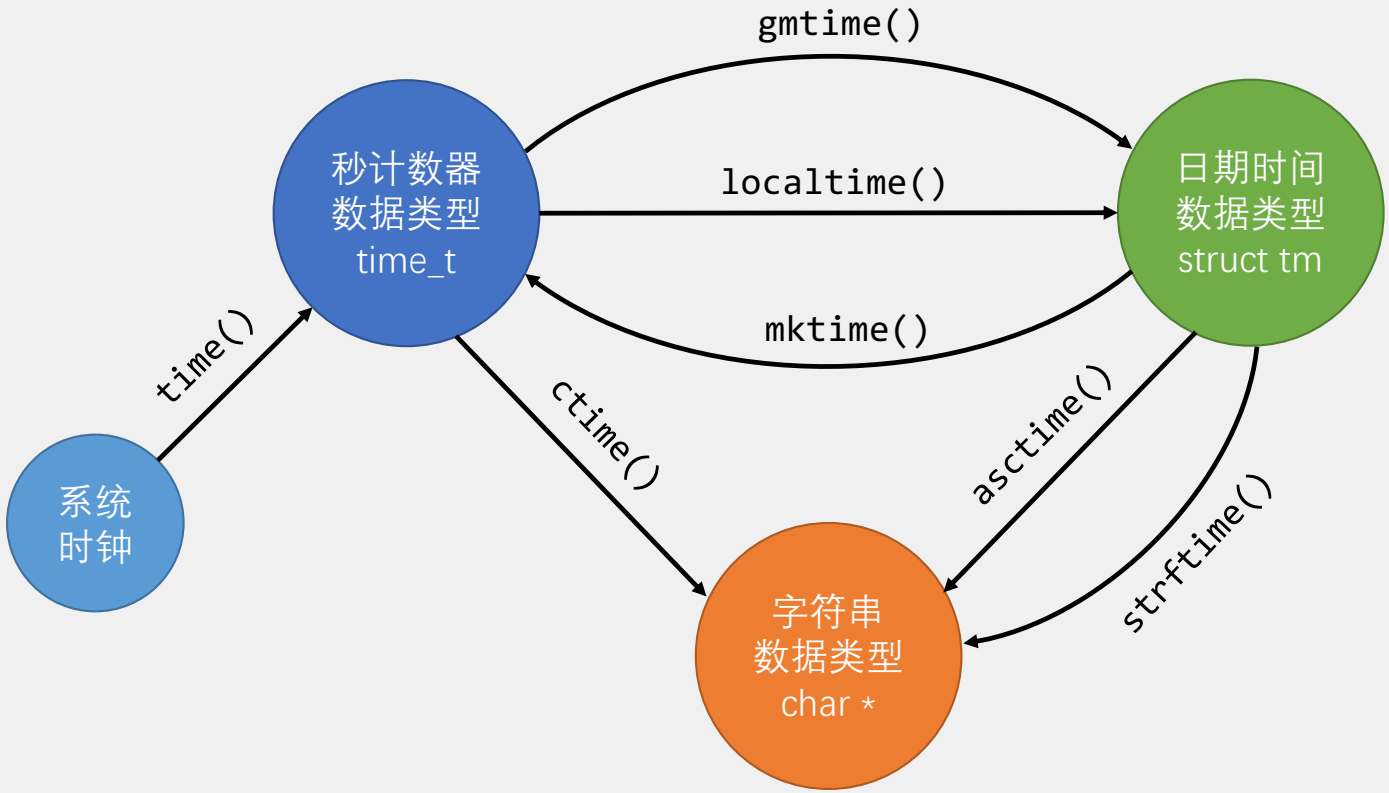
- UTC（Universal Time Coordinated）协调世界时是一种以原子钟为基础的时间计量系统。它规定铯133原子基态的两个超精细能级间在零磁场下跃迁辐射9,192,631,770周所持续的时间为1秒。当原子钟计时一天的时间与地球自转一周的时间相差超过0.9秒时，UTC会执行闰秒来保证其计时与地球自转的协调一致

时间戳转换

- C语言的time.h模块提供了时间获取和时间戳转换的相关函数，可以方便地进行秒计数器、日期时间和字符串之间的转换

函数	作用
`time_t time(time_t*);`	获取系统时钟
`struct tm* gmtime(const time_t*);`	秒计数器转换为日期时间（格林尼治时间）
`struct tm* localtime(const time_t*);`	秒计数器转换为日期时间（当地时间）
`time_t mktime(struct tm*);`	日期时间转换为秒计数器（当地时间）
`char* ctime(const time_t*);`	秒计数器转换为字符串（默认格式）
`char* asctime(const struct tm*);`	日期时间转换为字符串（默认格式）
`size_t strftime(char*, size_t, const char*, const struct tm*);`	日期时间转换为字符串（自定义格式）

时间戳转换

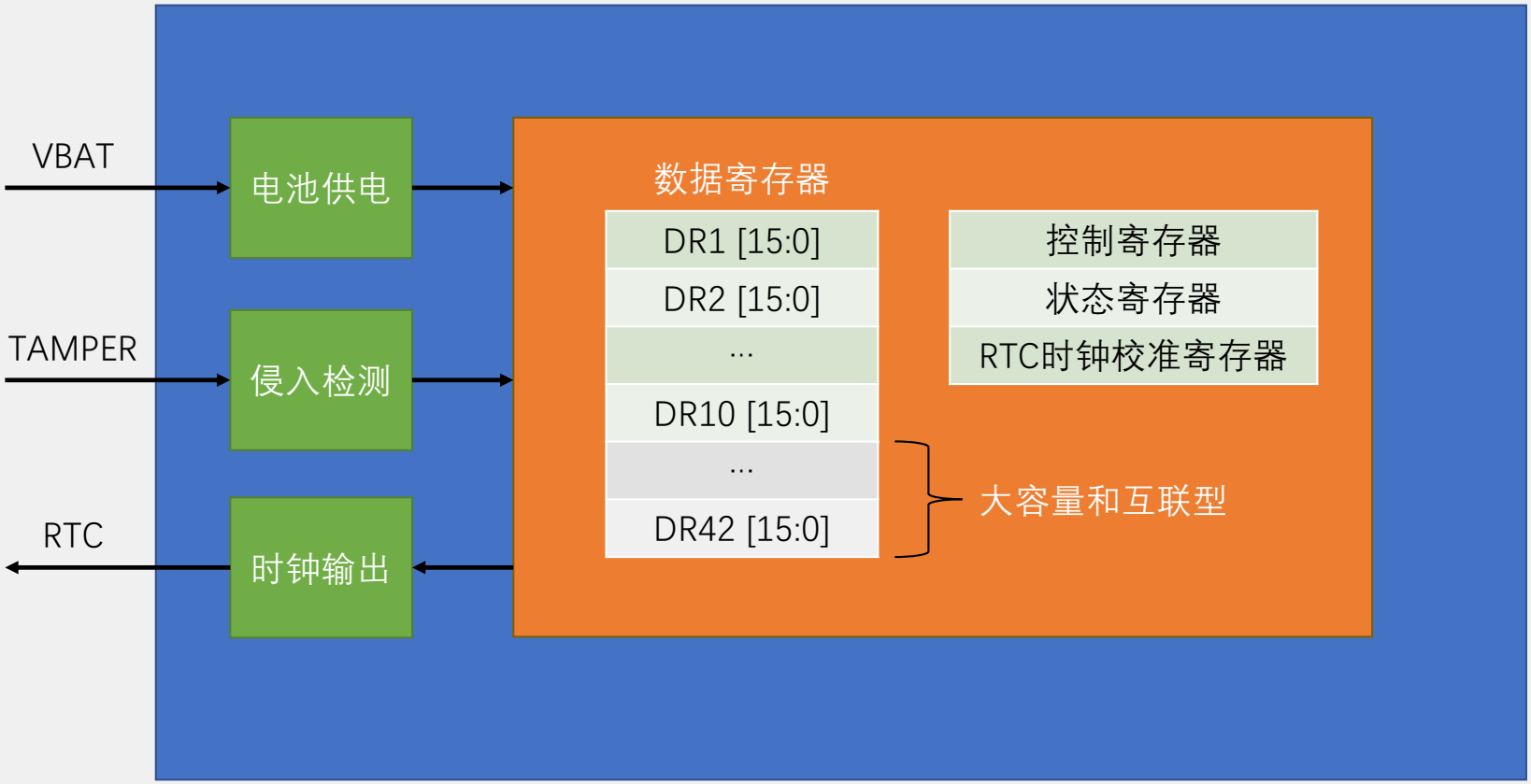

gmtime()
localtime()
mktime()
time()
ctime()
asctime()
strftime()
秒计数器
数据类型
time_t
日期时间
数据类型
struct tm
系统
时钟
字符串
数据类型
char *

BKP简介

- BKP（Backup Registers）备份寄存器
- BKP可用于存储用户应用程序数据。当VDD（2.0~3.6V）电源被切断，他们仍然由VBAT（1.8~3.6V）维持供电。当系统在待机模式下被唤醒，或系统复位或电源复位时，他们也不会被复位
- TAMPER引脚产生的侵入事件将所有备份寄存器内容清除
- RTC引脚输出RTC校准时钟、RTC闹钟脉冲或者秒脉冲
- 存储RTC时钟校准寄存器
- 用户数据存储容量：

  20字节（中容量和小容量）/ 84字节（大容量和互联型）

BKP基本结构

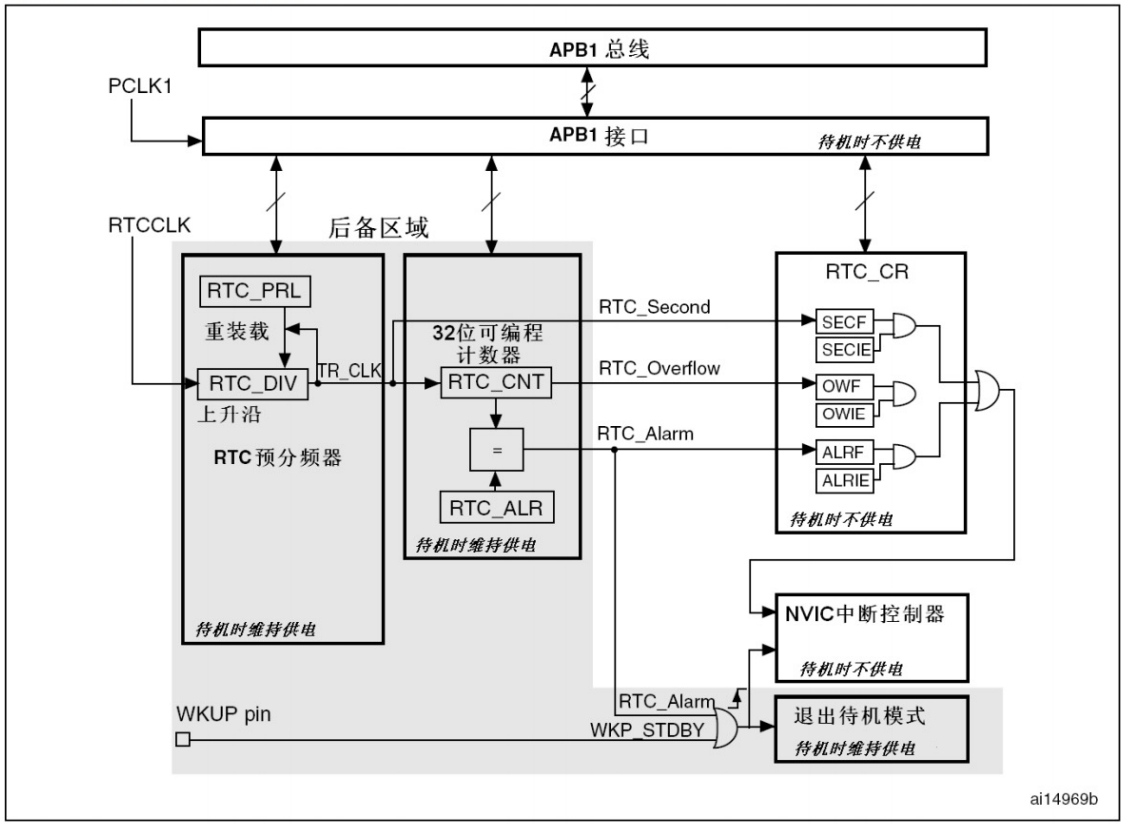

VBAT
电池供电
TAMPER
侵入检测
RTC
时钟输出
数据寄存器
DR1 [15:0]
DR2 [15:0]
…
DR10 [15:0]
…
DR42 [15:0]
控制寄存器
状态寄存器
RTC时钟校准寄存器
大容量和互联型

RTC简介

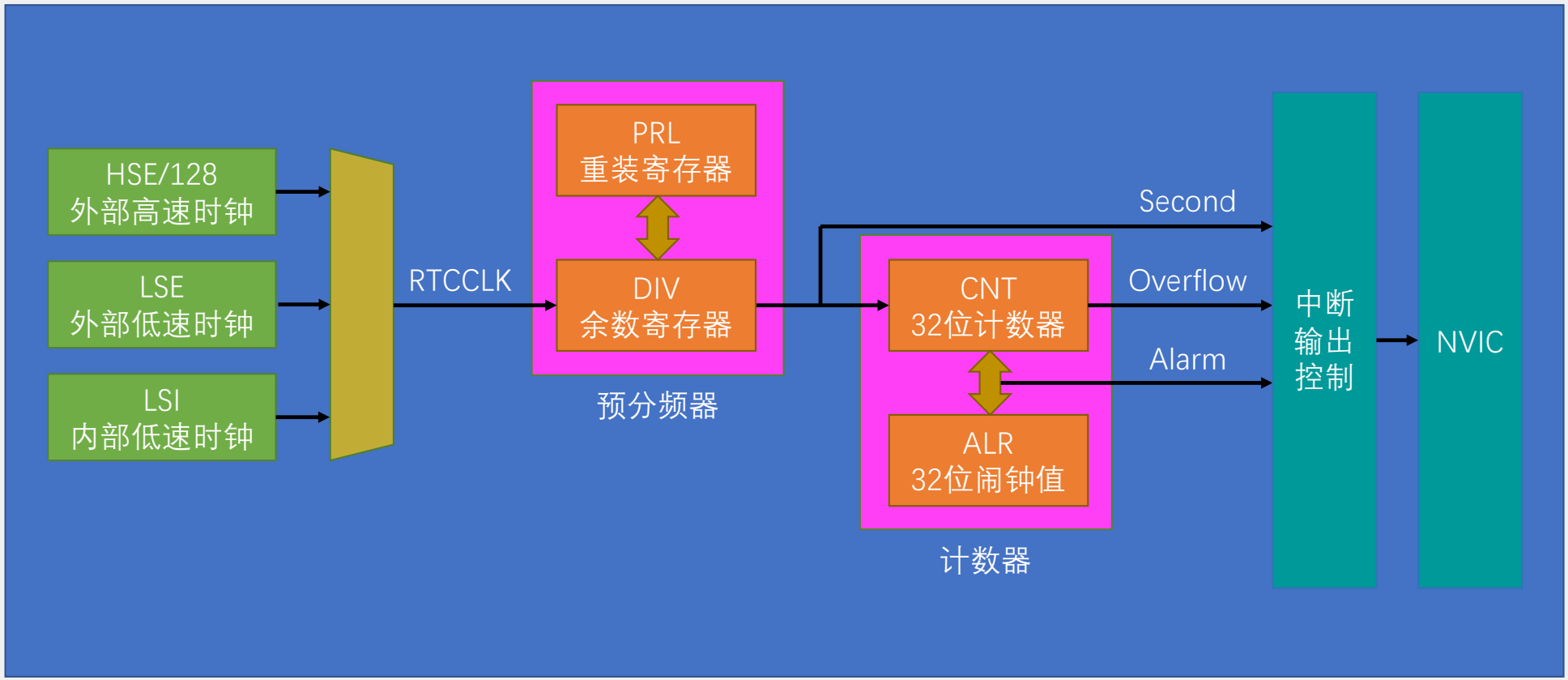
- RTC（Real Time Clock）实时时钟
- RTC是一个独立的定时器，可为系统提供时钟和日历的功能
- RTC和时钟配置系统处于后备区域，系统复位时数据不清零，VDD（2.0~3.6V）断电后可借助VBAT（1.8~3.6V）供电继续走时
- 32位的可编程计数器，可对应Unix时间戳的秒计数器
- 20位的可编程预分频器，可适配不同频率的输入时钟
- 可选择三种RTC时钟源：
  - HSE时钟除以128（通常为8MHz/128）
  - LSE振荡器时钟（通常为32.768KHz）
  - LSI振荡器时钟（40KHz）

RTC框图

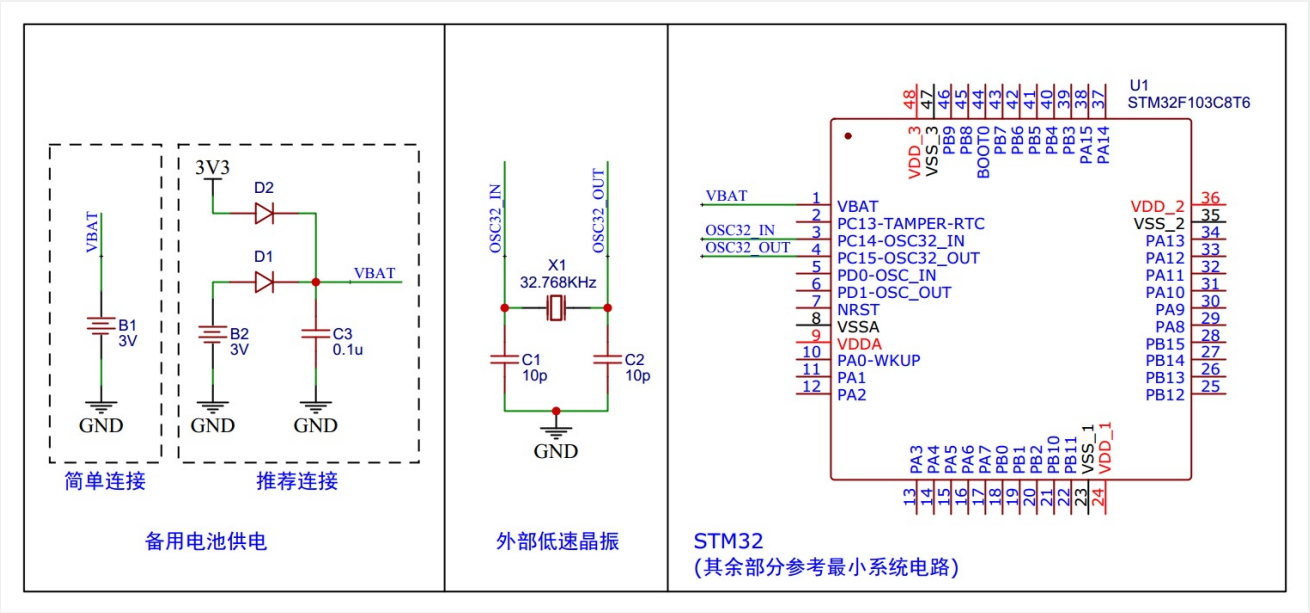
图154　简化的RTC框图

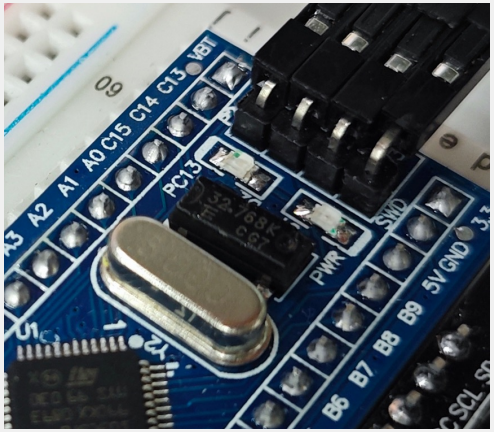


RTC基本结构


HSE/128
外部高速时钟
LSE
外部低速时钟
LSI
内部低速时钟
RTCCLK
PRL
重装寄存器
DIV
余数寄存器
预分频器
Second
CNT
32位计数器
Overflow
Alarm
ALR
32位闹钟值
计数器
中断
输出
控制
NVIC

硬件电路


3V3
D2
D1
VBAT
B1
3V
B2
3V
C3
0.1u
GND
GND
GND
简单连接
推荐连接
备用电池供电
OSC32_IN
OSC32_OUT
X1
32.768KHz
C1
10p
C2
10p
GND
外部低速晶振
U1
STM32F103C8T6
VBAT
PC13-TAMPER-RTC
PC14-OSC32_IN
PC15-OSC32_OUT
PD0-OSC_IN
PD1-OSC_OUT
NRST
VSSA
VDDA
PA0-WKUP
PA1
PA2
VDD_2
VSS_2
PA13
PA12
PA11
PA10
PA9
PA8
PB15
PB14
PB13
PB12
STM32
(其余部分参考最小系统电路)


PC13
PWR
SWD
5V
GND

RTC操作注意事项

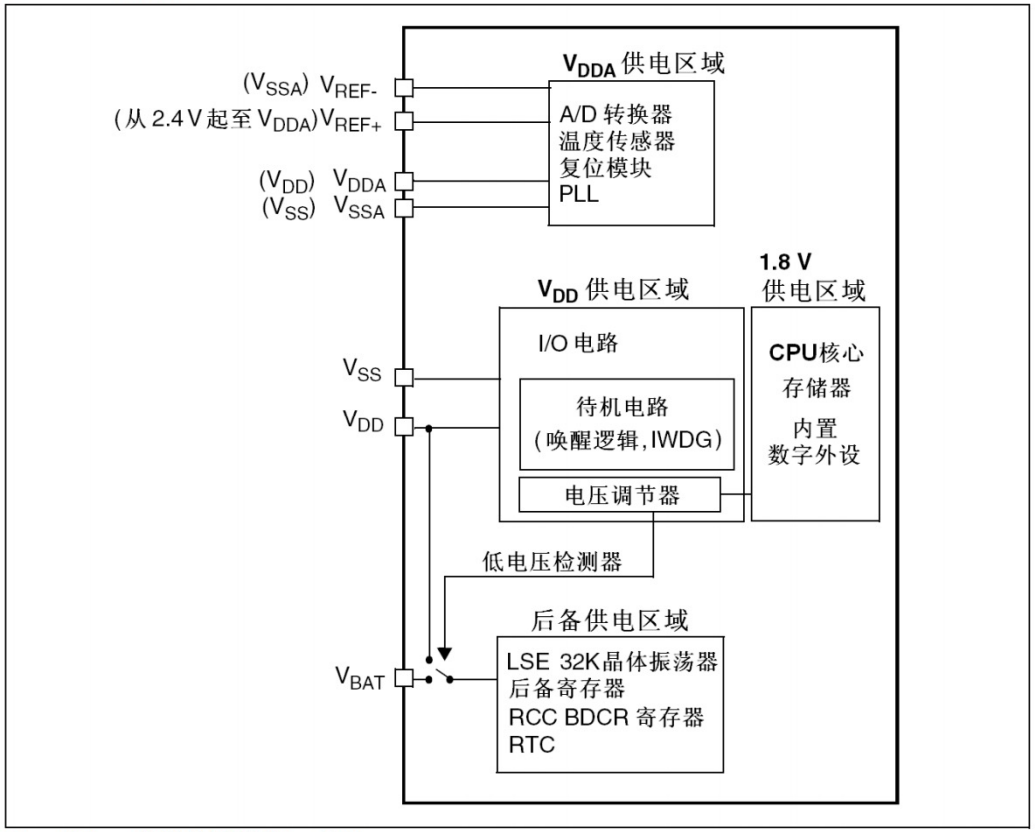
- 执行以下操作将使能对BKP和RTC的访问：
  设置RCC_APB1ENR的PWREN和BKPEN，使能PWR和BKP时钟
  设置PWR_CR的DBP，使能对BKP和RTC的访问
- 若在读取RTC寄存器时，RTC的APB1接口曾经处于禁止状态，则软件首先必须等待RTC_CRL寄存器中的RSF位（寄存器同步标志）被硬件置1
- 必须设置RTC_CRL寄存器中的CNF位，使RTC进入配置模式后，才能写入RTC_PRL、RTC_CNT、RTC_ALR寄存器
- 对RTC任何寄存器的写操作，都必须在前一次写操作结束后进行。可以通过查询RTC_CR寄存器中的RTOFF状态位，判断RTC寄存器是否处于更新中。仅当RTOFF状态位是1时，才可以写入RTC寄存器

PWR简介

- PWR（Power Control）电源控制
- PWR负责管理STM32内部的电源供电部分，可以实现可编程电压监测器和低功耗模式的功能
- 可编程电压监测器（PVD）可以监控VDD电源电压，当VDD下降到PVD阀值以下或上升到PVD阀值之上时，PVD会触发中断，用于执行紧急关闭任务
- 低功耗模式包括睡眠模式（Sleep）、停机模式（Stop）和待机模式（Standby），可在系统空闲时，降低STM32的功耗，延长设备使用时间

电源框图

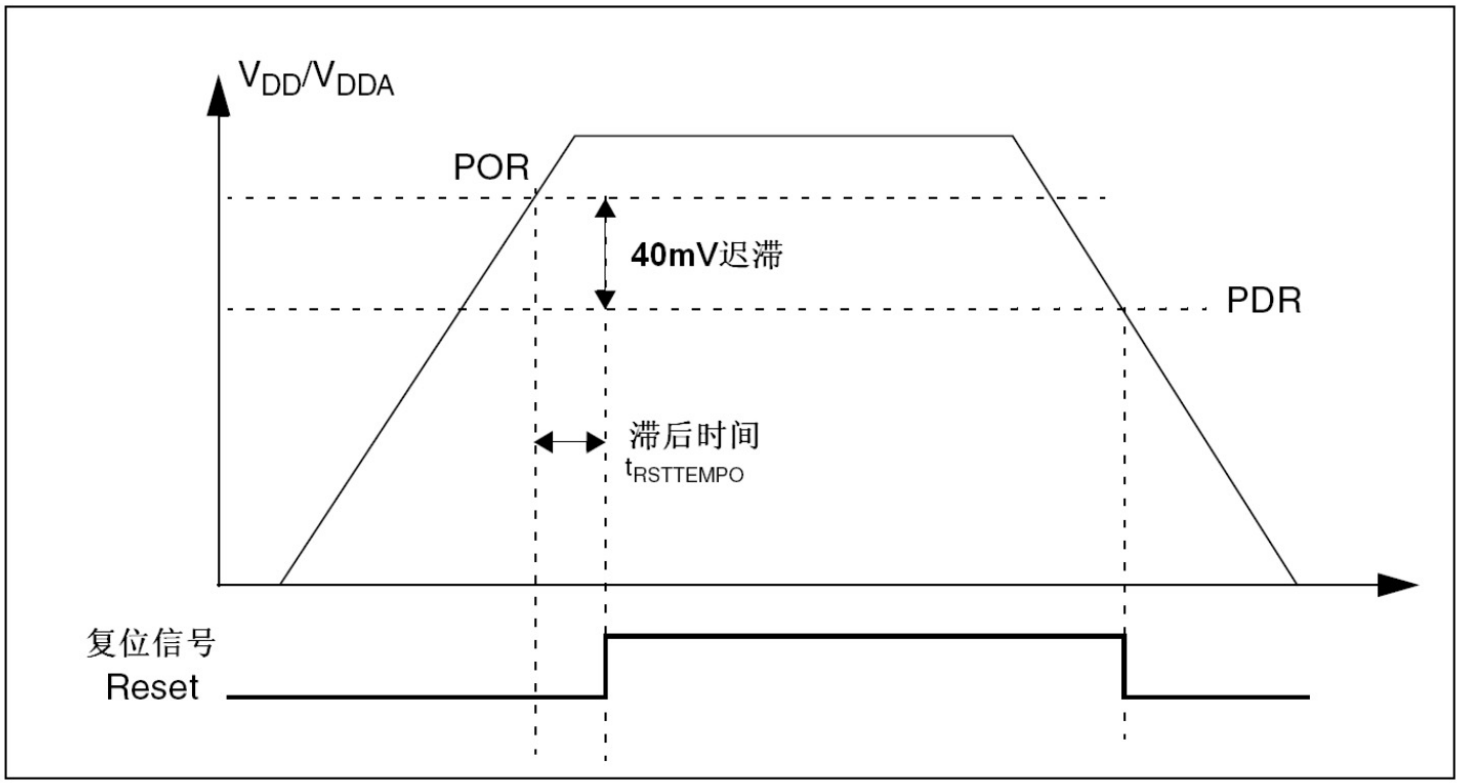
图4　　电源框图



V_{DDA}和V_{SSA}必须分别联到V_{DD}和V_{SS}。

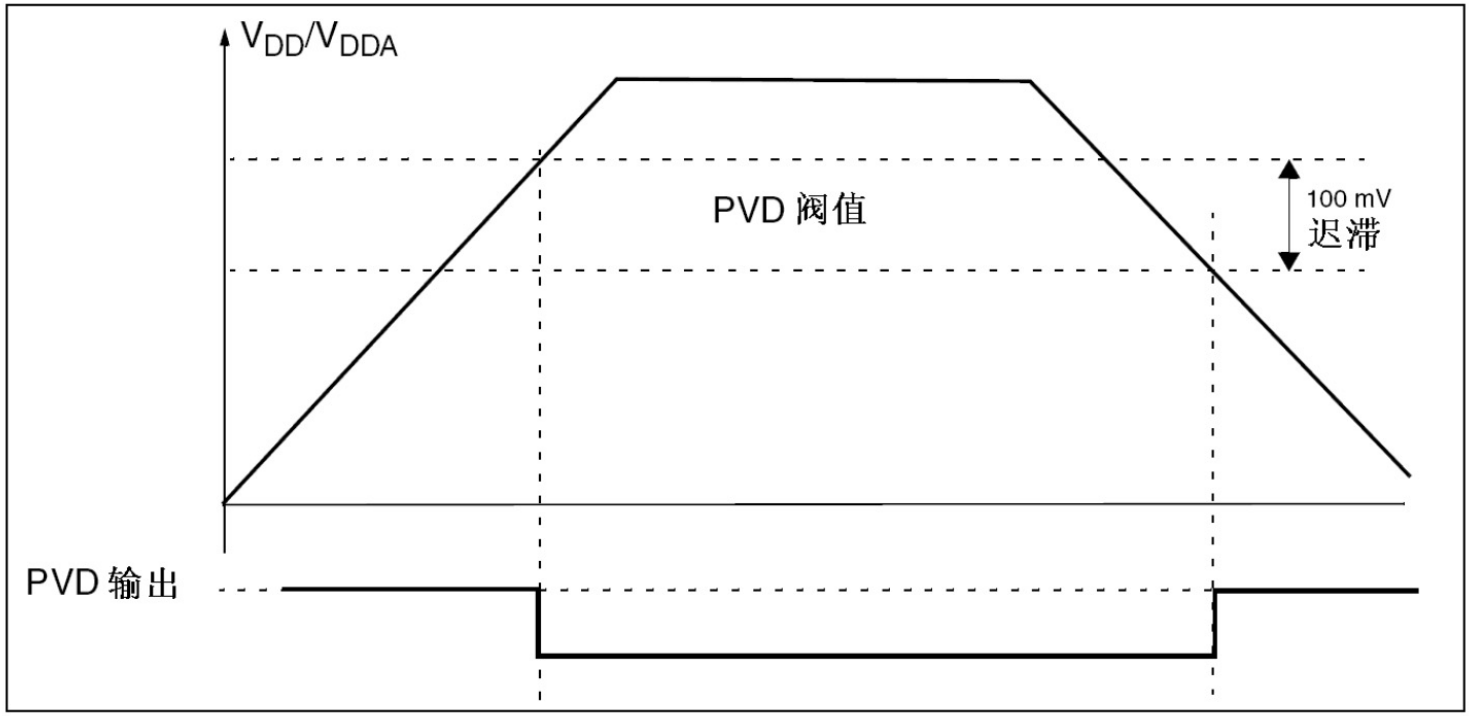
上电复位和掉电复位

图5　　上电复位和掉电复位的波形图



可编程电压监测器

图6　PVD的门限



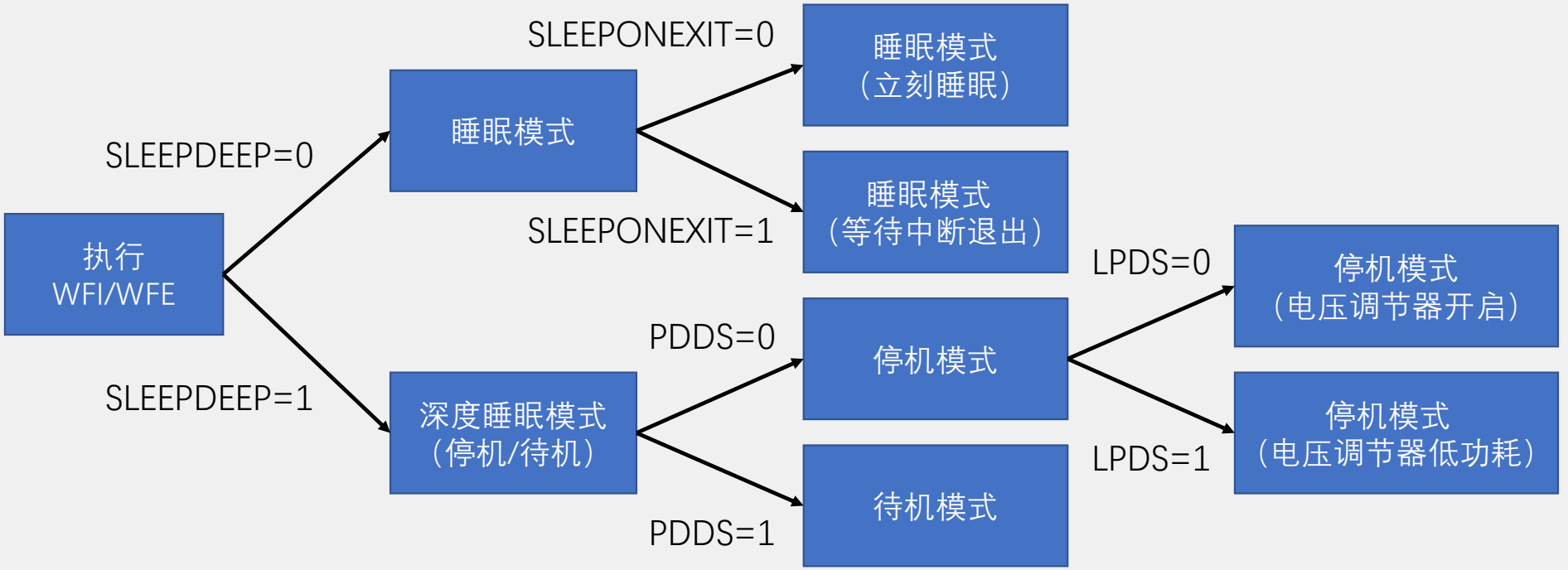
低功耗模式

表8　低功耗模式一览

模式	进入	唤醒	对1.8V区域时钟的影响	对V_{DD}区域时钟的影响	电压调节器
睡眠 (SLEEP-NOW或 SLEEP-ON-EXIT)	WFI	任一中断	CPU时钟关，对其他时钟和ADC时钟无影响	无	开
	WFE	唤醒事件			
停机	PDDS和LPDS位 +SLEEPDEEP位 +WFI或WFE	任一外部中断(在外部中断寄存器中设置)	关闭所有1.8V区域的时钟	HSI 和HSE的振荡器关闭	开启或处于低功耗模式(依据电源控制寄存器(PWR_CR)的设定)
待机	PDDS位 +SLEEPDEEP位 +WFI或WFE	WKUP引脚的上升沿、RTC闹钟事件、NRST引脚上的外部复位、IWDG复位			关

模式选择

- 执行WFI（Wait For Interrupt）或者WFE（Wait For Event）指令后，STM32进入低功耗模式



睡眠模式

- 执行完WFI/WFE指令后，STM32进入睡眠模式，程序暂停运行，唤醒后程序从暂停的地方继续运行
- SLEEPONEXIT位决定STM32执行完WFI或WFE后，是立刻进入睡眠，还是等STM32从最低优先级的中断处理程序中退出时进入睡眠
- 在睡眠模式下，所有的I/O引脚都保持它们在运行模式时的状态
- WFI指令进入睡眠模式，可被任意一个NVIC响应的中断唤醒
- WFE指令进入睡眠模式，可被唤醒事件唤醒

停止模式

- 执行完WFI/WFE指令后，STM32进入停止模式，程序暂停运行，唤醒后程序从暂停的地方继续运行
- 1.8V供电区域的所有时钟都被停止，PLL、HSI和HSE被禁止，SRAM和寄存器内容被保留下来
- 在停止模式下，所有的I/O引脚都保持它们在运行模式时的状态
- 当一个中断或唤醒事件导致退出停止模式时，HSI被选为系统时钟
- 当电压调节器处于低功耗模式下，系统从停止模式退出时，会有一段额外的启动延时
- WFI指令进入停止模式，可被任意一个EXTI中断唤醒
- WFE指令进入停止模式，可被任意一个EXTI事件唤醒

待机模式

- 执行完WFI/WFE指令后，STM32进入待机模式，唤醒后程序从头开始运行
- 整个1.8V供电区域被断电，PLL、HSI和HSE也被断电，SRAM和寄存器内容丢失，只有备份的寄存器和待机电路维持供电
- 在待机模式下，所有的I/O引脚变为高阻态（浮空输入）
- WKUP引脚的上升沿、RTC闹钟事件的上升沿、NRST引脚上外部复位、IWDG复位退出待机模式

- END