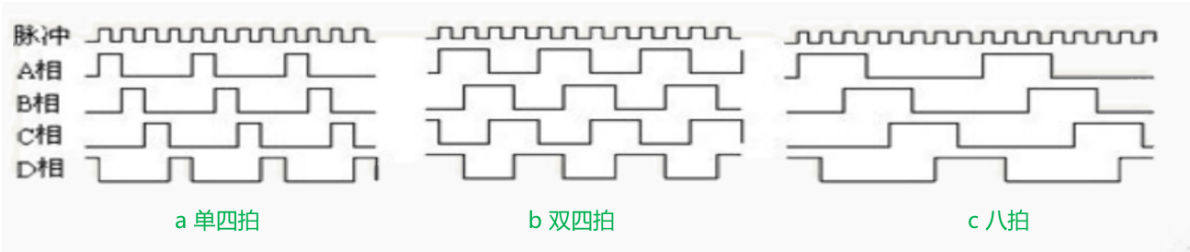
步进电机

步进电机是一种能够将**电脉冲信号转换成角位移或线位移**的机电元件， 是机电一体化的关键产品之一。

步进电机的驱动方式

四相步进电机按照**线圈通电顺序的不同**，可分为**单四拍、双四拍、八拍**三种驱动方式。

从一个时刻看上去如，有多少个高电平（有的是低电平有效，注意）为1是单，为2是双；一个周期占据多少脉冲周期为多少拍。



a 单四拍　　b 双四拍　　c 八拍

步距角

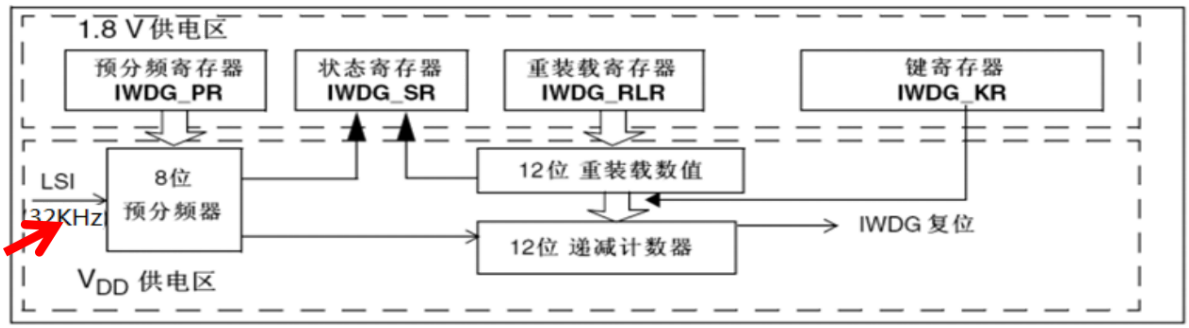
$$\theta = \frac{360^\circ}{\text{转子齿数}J * \text{运行拍数}n}$$

看门狗

在单片机系统中，由于工作常常会受到外界的干扰，如电磁场等，造成程序的正常运行被打断，程序跑飞、陷入死循环……系统无法继续工作，或整个系统陷入停滞状态、发生不可预料的后果。

一种专门用于监测单片机程序运行状态的模块或者芯片，俗称“看门狗”(watchdog)。

图157　独立看门狗框图



键值寄存器IWDG_KR:

0~15位有效。**可以发启用（0xCCCC）、喂狗（0xAAAA）、解锁（0x5555）等命令。**

预分频寄存器IWDG_PR

0~2位有效。　**有写保护功能，要操作，得先取消写保护。**

预分频寄存器(IWDG_PR)

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
保留													PR[2:0]		
													rw	rw	rw

位31:3	保留，始终读为0。
位2:0	**PR[2:0]**: 预分频因子 (Prescaler divider) 这些位具有写保护设置，参见17.3.2节。通过设置这些位来选择计数器时钟的预分频因子。要改变预分频因子，IWDG_SR寄存器的PVU位必须为0。 000: 预分频因子=4　　100: 预分频因子=64 001: 预分频因子=8　　101: 预分频因子=128 010: 预分频因子=16　　110: 预分频因子=256 011: 预分频因子=32　　111: 预分频因子=256 注意：对此寄存器进行读操作，将从VDD电压域返回预分频值。如果写操作正在进行，则读回的值可能是无效的。因此，只有当IWDG_SR寄存器的PVU位为0时，读出的值才有效。

IWDG的时钟预分频值与超时时间极限值:

预分频器	PR[2:0] 位	最短超时 (ms) RL[11:0]= 0x000	最长超时 (ms) RL[11:0]= 0xFFF
/4	0	0.125	512
/8	1	0.25	1024
/16	2	0.5	2048
/32	3	1	4096
/64	4	2	8192
/128	5	4	16384
/256	6	8	32768

超时时间计算公式：

$$T_{out} = \frac{(4 \times 2^{prer}) \times rlr}{32} \tag{1}$$

时钟频率LSI=32K， 一个看门狗时钟周期就是最短超时时间。\
最长超时时间= (IWDG_RLR寄存器最大值0xFFF) x 看门狗时钟周期

重装载寄存器IWDG_RLR

0~11位有效。**有写保护功能，要操作，得先取消写保护。**

状态寄存器IWDG_SR

0~1位有效。

IWDG操作步骤

1. 初始化编程
    1. 允许写入PR、RLR ： WDG_WriteAccessCmd();
    2. 设置独立看门狗的预分频系数，确定时钟: WDG_SetPrescaler();
    3. 置看门狗重装载值rlr，确定溢出时间:$T_{out} = \frac{(4 \times 2^{prer}) \times rlr}{32} ms$ ： WDG_SetReload();
    4. 使能看门狗 ：WDG_Enable();
2. 操作编程
    1. 应用程序喂狗: WDG_ReloadCounter();

1.看门狗初始化

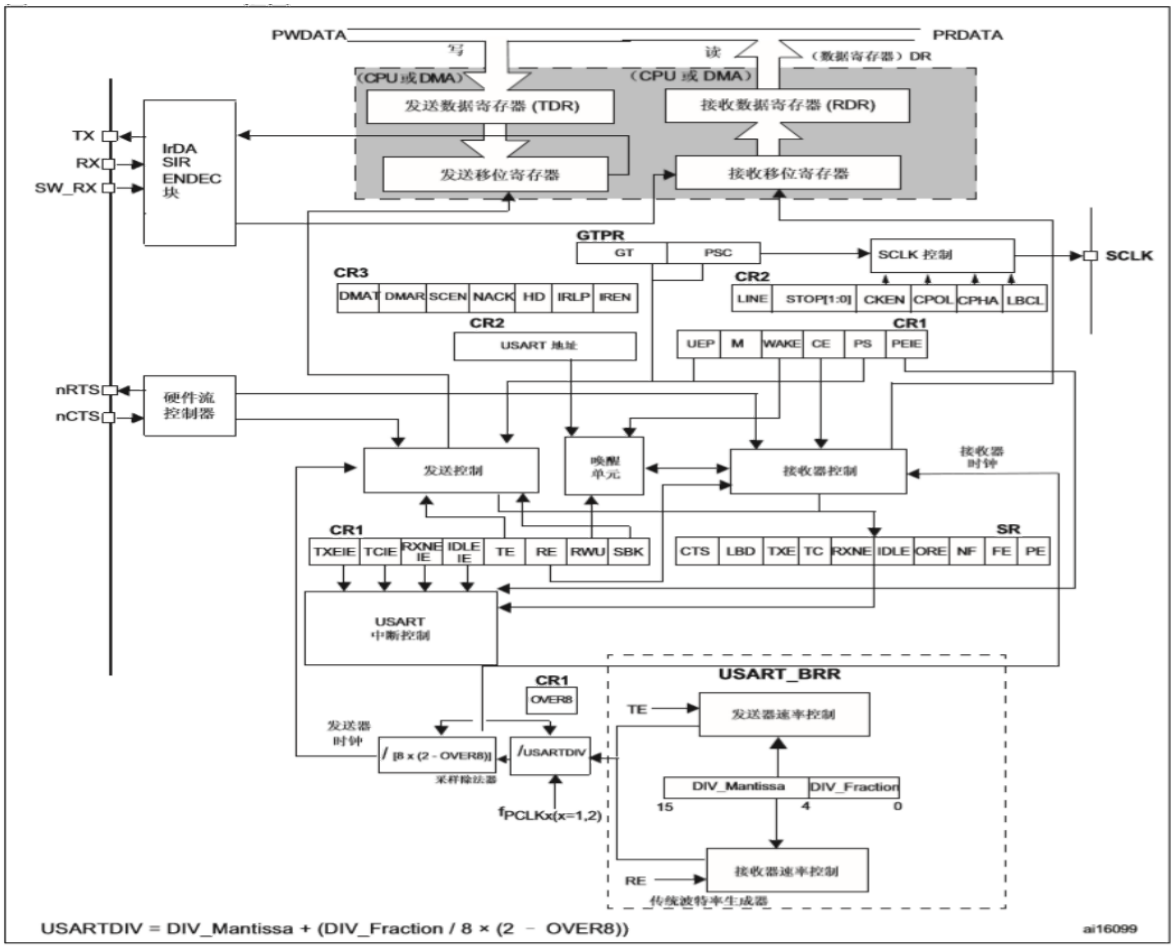
```
/*************************************************************
函数名称：看门狗初始化函数
输入参数：无
输出参数：无
看门狗超时时间：2s
*************************************************************/
void IWDG_Configuration(void)
{
    //  写入键值0x5555，  即允许狗狗寄存器写入功能
    IWDG_WriteAccessCmd(IWDG_WriteAccess_Enable);

    //  狗狗时钟分频,40K/256=156HZ(6.4ms)
    IWDG_SetPrescaler(IWDG_Prescaler_256);

    //  喂狗时间 2s/6.4MS = 312 ，  注意不能大于0xfff
    IWDG_SetReload(312);

    //  写入键值0xAAAA，即喂狗
    IWDG_ReloadCounter();

    //  写入键值0xCCCC，即使能狗狗
    IWDG_Enable();
}
```

2.喂狗

```
//  喂狗，程序不复位
IWDG_ReloadCounter();
```

51

USART串口原理



相关寄存器

偏移地址：0x0C

复位值：0x0000 0000

USART控制寄存器1：USART_CR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

偏移地址：0x10

复位值：0x0000 0000

USART控制寄存器2：USART_CR2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

USART中断

26.4 USART 中断

表 121. USART 中断请求

中断事件	事件标志	使能控制位
发送数据寄存器为空	TXE	TXEIE
CTS 标志	CTS	CTSIE
发送完成	TC	TCIE
准备好读取接收到的数据	RXNE	RXNEIE
检测到上溢错误	ORE	
检测到空闲线路	IDLE	IDLEIE
奇偶校验错误	PE	PEIE
断路标志	LBD	LBDIE
多缓冲区通信中的噪声标志、上溢错误和帧错误	NF 或 ORE 或 FE	EIE

USART 中断事件被连接到相同的中断向量（请参见*图 270*）。

- 发送期间：发送完成、清除以发送或发送数据寄存器为空中断。
- 接收期间：空闲线路检测、上溢错误、接收数据寄存器不为空、奇偶校验错误、LIN 断路检测、噪声标志（仅限多缓冲区通信）和帧错误（仅限多缓冲区通信）

如果相应的使能控制位置 1，则这些事件会生成中断。

USART/UART中断有哪几个？中断名是什么？

CTS change interrupt. \
LIN((local interconnection network) Break detection interrupt.\
Tansmit Data Register empty interrupt.\
Transmission complete interrupt.\
Receive Data register not empty interrupt.\
Idle line detection interrupt.\
Parity Error interrupt.\
Error interrupt(Frame error, noise error, overrun error).\
硬流控的发送允许状态改变中断\
本地互连网络破坏检测中断\
发送数据寄存器空中断\
接收数据寄存器不空中断\
uart空闲检测中断\
极性错误中断\
错误中断（帧错误，杂讯错误，溢出错误）

串口线与gpiox引脚的关系

GPIO引脚复用方式充当串口线！

AFRL、AFRH 这两个寄存器的作用

是将内部Usartx（等接口设备）使用的固定对应的GPIOx：配置成“复用方式”

```
//  开启GPIO_B的时钟
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

//  开启串口3的时钟
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);

GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_11;
GPIO_Init(GPIOB, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_10;
GPIO_Init(GPIOB, &GPIO_InitStructure);


GPIO_PinAFConfig(GPIOB, GPIO_PinSource10, GPIO_AF_USART3);
GPIO_PinAFConfig(GPIOB, GPIO_PinSource11, GPIO_AF_USART3);


USART_InitStructure.USART_BaudRate    = 115200;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits    = USART_StopBits_1;
USART_InitStructure.USART_Parity      = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
```