《手把手教你学STM32》

主讲人　：正点原子团队
硬件平台：正点原子STM32开发板
版权所有：广州市星翼电子科技有限公司
淘宝店铺：http://eboard.taobao.com
技术论坛：www.openedv.com 开源电子网
公众平台：“正点原子”
官方网站：www.alientek.com
联系电话：13922348612

ALIENTEK

《手把手教你学STM32》

■ 定时器中断

适用平台

- ~~✓ STM32F1xx 开发板（正点原子）~~
- ✓ STM32F4xx 开发板（正点原子）

✓ 定时器中断

■ 参考资料：

● 探索者STM32F4开发板：

《STM32F4开发指南-库函数版本》-第13章 定时器中断实验

□ STM32F4xx官方资料：

《STM32F4xx中文参考手册》-第15章 通用定时器

目录

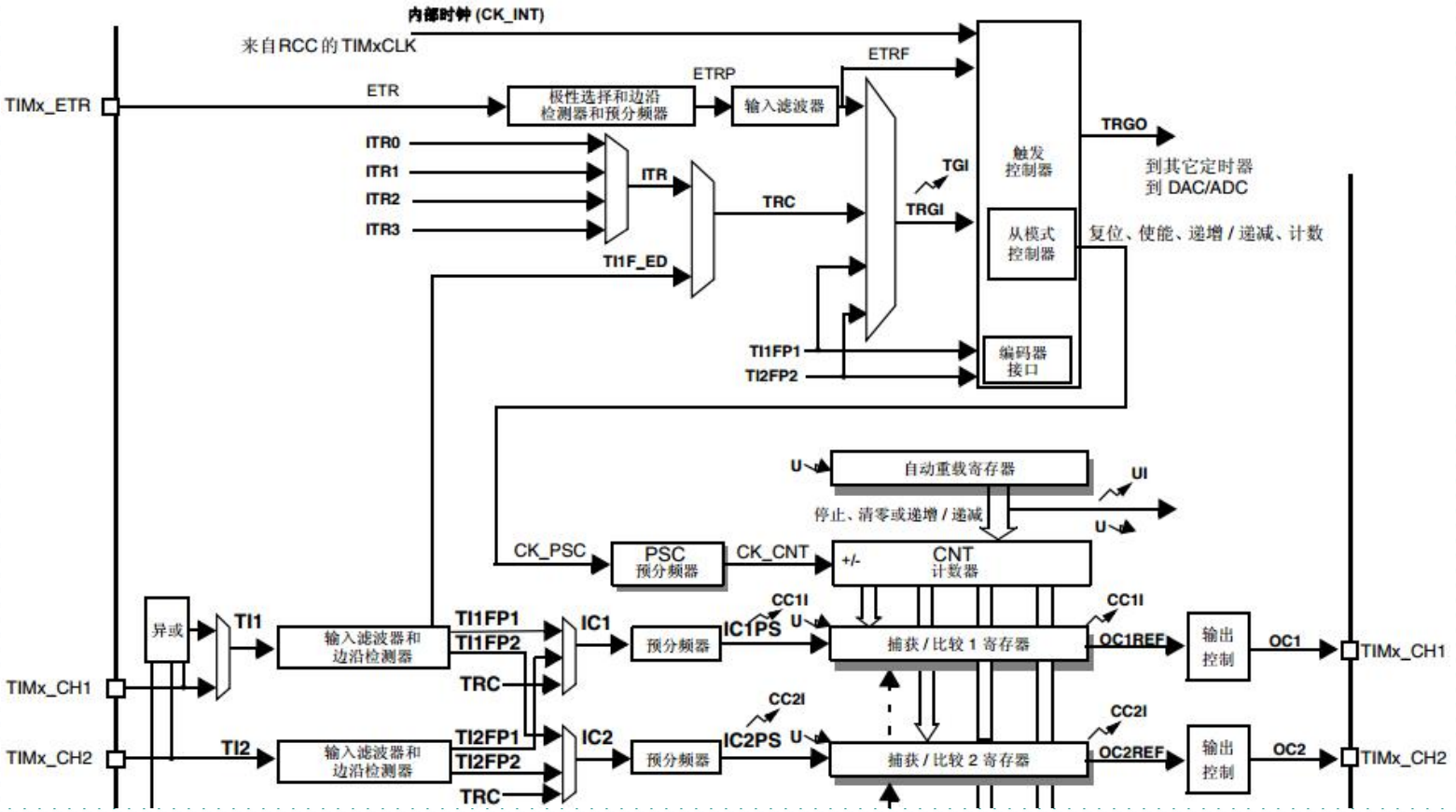
1 通用定时器知识回顾

2 常用寄存器和库函数配置

3 手把手写定时器中断实验

✓ 通用定时器概述

◆ 通用定时器工作过程：



◆时钟选择

计数器时钟可以由下列时钟源提供：

① 内部时钟(CK_INT)

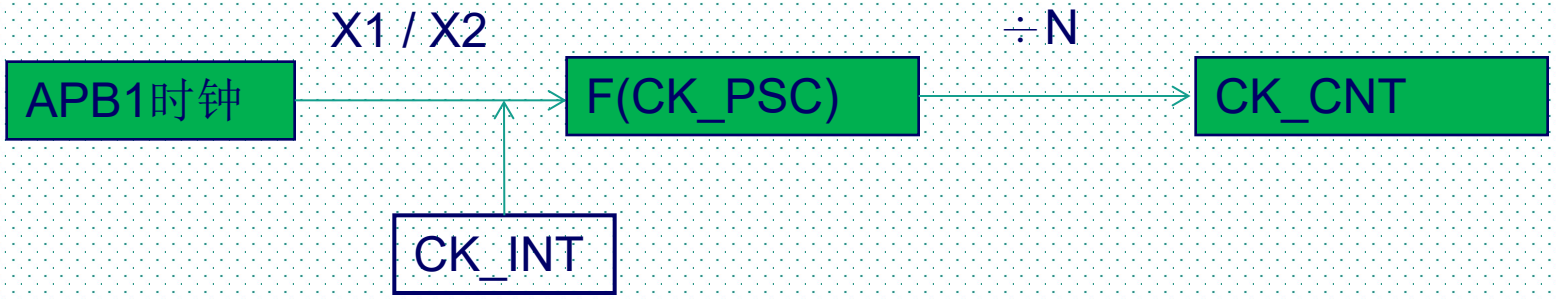
② 外部时钟模式1：外部输入脚(TIx)

③ 外部时钟模式2：外部触发输入(ETR)(仅适用TIM2,3,4)

④ 内部触发输入(ITRx)：使用一个定时器作为另一个定时器的预分频器，如可以配置一个定时器Timer1而作为另一个定时器Timer2的预分频器。

✓ 定时器中断实验

◆内部时钟选择



除非APB1的分频系数是1，否则通用定时器的时钟等于APB1时钟的2倍。

默认调用SystemInit函数情况下：
SYSCLK=168M
AHB时钟=168M
APB1时钟=42M
所以APB1的分频系数=AHB/APB1时钟=4
所以，通用定时器时钟CK_INT=2*42M=84M

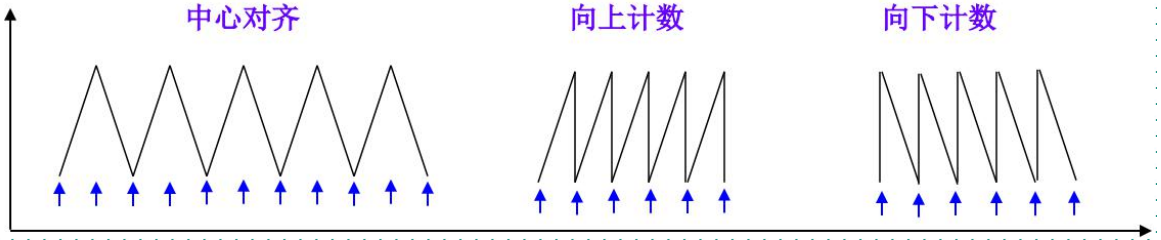
◆计数器模式

通用定时器可以向上计数、向下计数、向上向下双向计数模式。

①向上计数模式：计数器从0计数到自动加载值(TIMx_ARR)，然后重新从0开始计数并且产生一个计数器溢出事件。

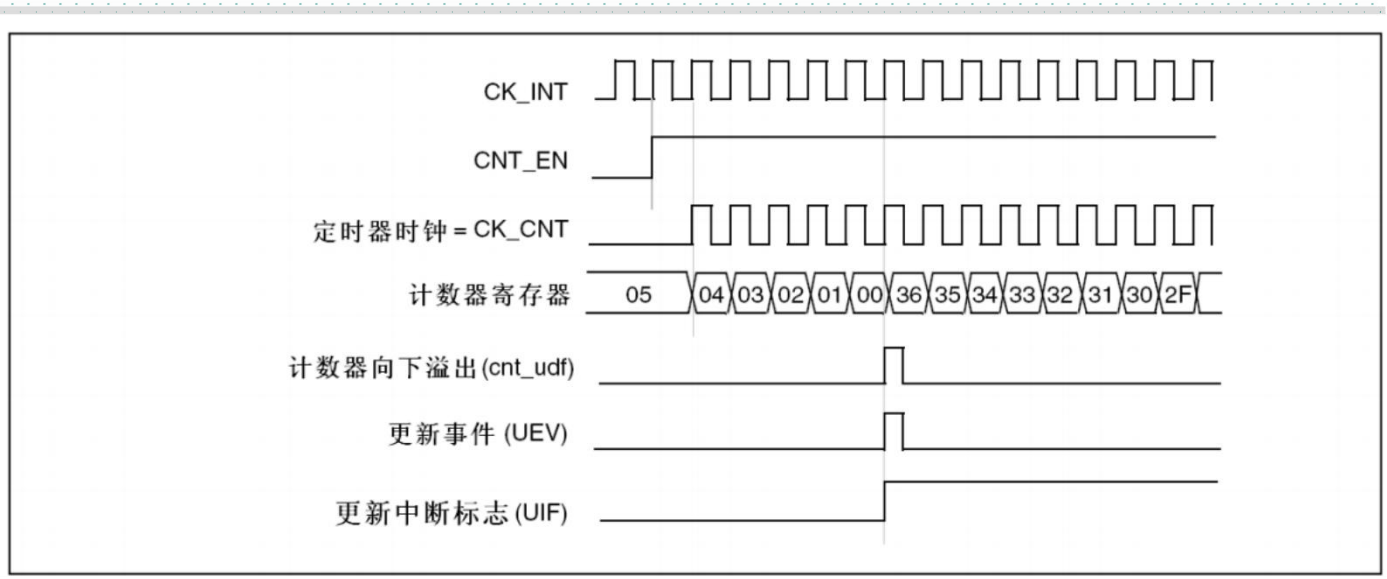
②向下计数模式：计数器从自动装入的值(TIMx_ARR)开始向下计数到0，然后从自动装入的值重新开始，并产生一个计数器向下溢出事件。

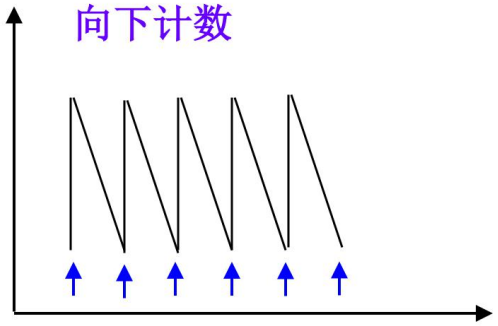
③中央对齐模式（向上/向下计数）：计数器从0开始计数到自动装入的值-1，产生一个计数器溢出事件，然后向下计数到1并且产生一个计数器溢出事件；然后再从0开始重新计数。



✓ 定时器中断实验

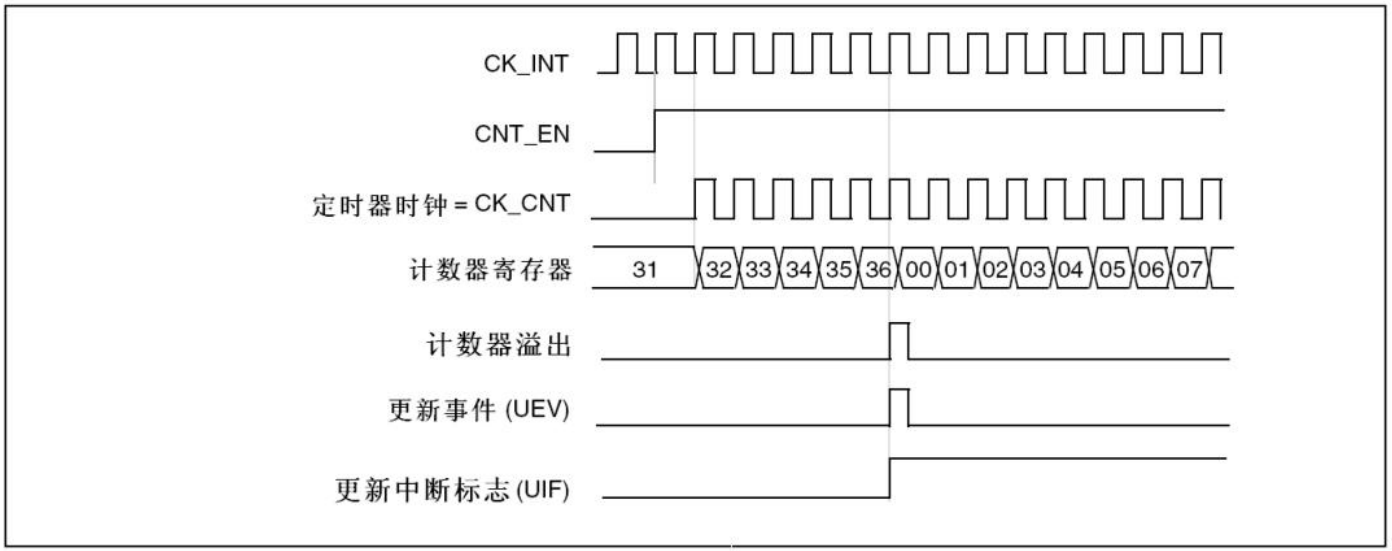
◆ 向下计数模式（时钟分频因子=1）

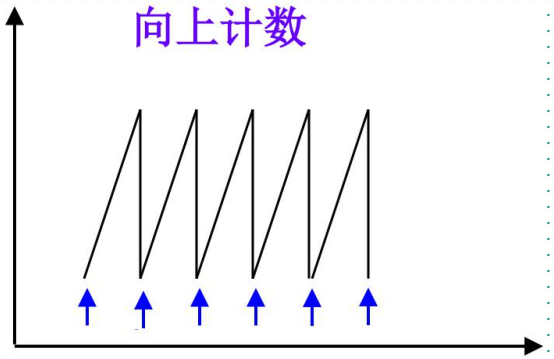




✓ 定时器中断实验

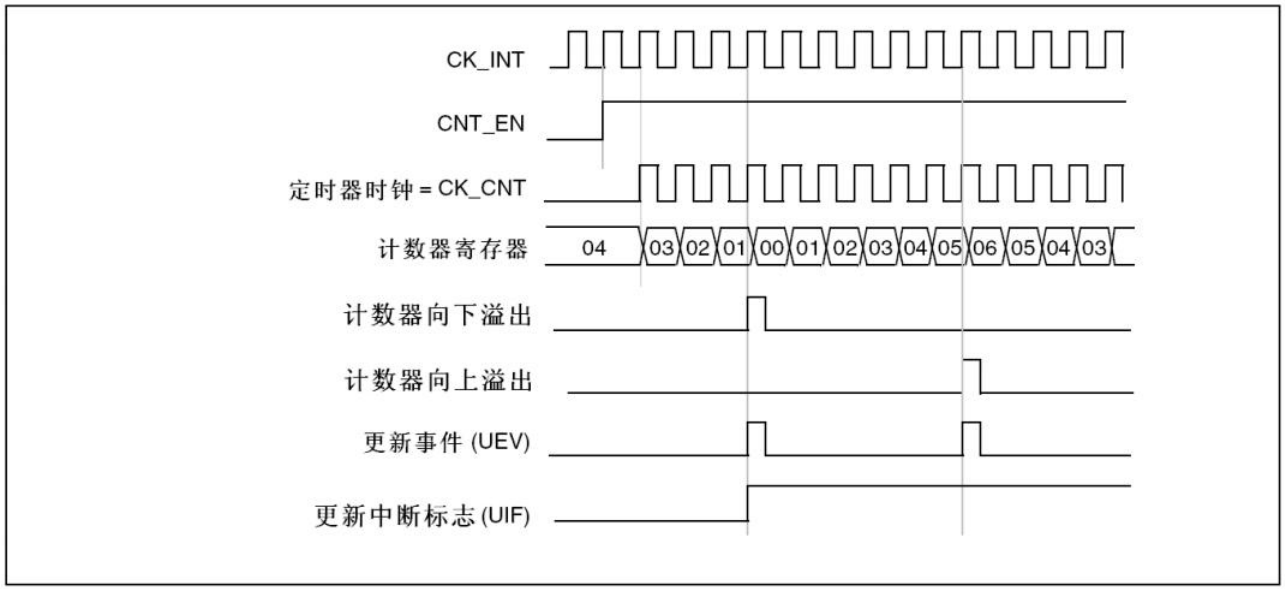
◆ 向上计数模式（时钟分频因子=1）

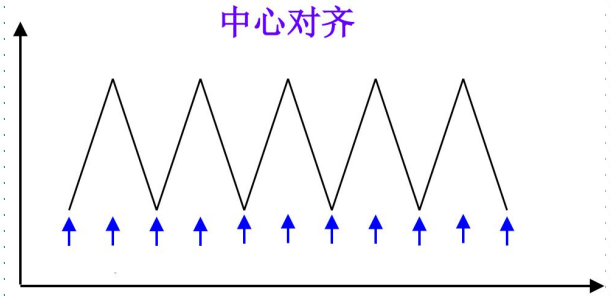




✓ 定时器中断实验

◆ 中央对齐计数模式（时钟分频因子=1 ARR=6）





定时器中断实验相关寄存器

✓ 通用定时器常用寄存器和库函数

◆计数器当前值寄存器CNT

计数器(TIMx_CNT)

偏移地址：0x24

复位值：0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15:0	**CNT[15:0]**：计数器的值 (Counter value)

✓ 通用定时器常用寄存器和库函数

◆预分频寄存器TIMx_PSC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15:0	**PSC[15:0]**：预分频器的值 (Prescaler value) 计数器的时钟频率CK_CNT等于f_{CK_PSC}/(PSC[15:0]+1)。 PSC包含了当更新事件产生时装入当前预分频器寄存器的值。

✓ 通用定时器常用寄存器和库函数

◆自动重装载寄存器（TIMx_ARR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位15:0	**ARR[15:0]**: 自动重装载的值 (Auto reload value) ARR包含了将要传送至实际的自动重装载寄存器的数值。 详细参考14.3.1节：有关ARR的更新和动作。 当自动重装载的值为空时，计数器不工作。

✓ 通用定时器常用寄存器和库函数

◆控制寄存器1（TIMx_CR1）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位4	**DIR**：方向 (Direction) 0：计数器向上计数； 1：计数器向下计数。 注：当计数器配置为中央对齐模式或编码器模式时，该位为只读。
位0	**CEN**：使能计数器 0：禁止计数器； 1：使能计数器。 注：在软件设置了CEN位后，外部时钟、门控模式和编码器模式才能工作。触发模式可以自动地通过硬件设置CEN位。 在单脉冲模式下，当发生更新事件时，CEN被自动清除。

◆DMA中断使能寄存器（TIMx_DIER）

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留	TDE	保留	CC4DE	CC3DE	CC2DE	CC1DE	UDE	保留	TIE	保留	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

位5	保留，始终读为0。
位4	**CC4IE**：允许捕获/比较4中断 (Capture/Compare 4 interrupt enable) 0：禁止捕获/比较4中断； 1：允许捕获/比较4中断。
位3	**CC3IE**：允许捕获/比较3中断 (Capture/Compare 3 interrupt enable) 0：禁止捕获/比较3中断； 1：允许捕获/比较3中断。
位2	**CC2IE**：允许捕获/比较2中断 (Capture/Compare 2 interrupt enable) 0：禁止捕获/比较2中断； 1：允许捕获/比较2中断。
位1	**CC1IE**：允许捕获/比较1中断 (Capture/Compare 1 interrupt enable) 0：禁止捕获/比较1中断； 1：允许捕获/比较1中断。
位0	**UIE**：允许更新中断 (Update interrupt enable) 0：禁止更新中断； 1：允许更新中断。

✓ 通用定时器常用寄存器和库函数

◆常用库函数:stm32f4xx_tim.c/.h

定时器参数初始化：

void TIM_TimeBaseInit(TIM_TypeDef* TIMx,
TIM_TimeBaseInitTypeDef* TIM_TimeBaseInitStruct);

```
typedef struct
{
  uint16_t TIM_Prescaler;
  uint16_t TIM_CounterMode;
  uint16_t TIM_Period;
  uint16_t TIM_ClockDivision;
  uint8_t TIM_RepetitionCounter;
} TIM_TimeBaseInitTypeDef;
```

```
TIM_TimeBaseStructure.TIM_Period = 4999;
TIM_TimeBaseStructure.TIM_Prescaler =7199;
TIM_TimeBaseStructure.TIM_ClockDivision =   TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_CounterMode =   TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
```

✓ 通用定时器常用寄存器和库函数

◆ 定时器使能函数：

void TIM_Cmd(TIM_TypeDef TIMx, FunctionalState NewState)*

◆ 定时器中断使能函数：

void TIM_ITConfig(TIM_TypeDef TIMx, uint16_t TIM_IT, FunctionalState NewState);*

◆ 状态标志位获取和清除

FlagStatus TIM_GetFlagStatus(TIM_TypeDef TIMx, uint16_t TIM_FLAG);*

void TIM_ClearFlag(TIM_TypeDef TIMx, uint16_t TIM_FLAG);*

ITStatus TIM_GetITStatus(TIM_TypeDef TIMx, uint16_t TIM_IT);*

void TIM_ClearITPendingBit(TIM_TypeDef TIMx, uint16_t TIM_IT);*

✓ 通用定时器常用寄存器和库函数

◆ 定时器中断实现步骤

① 能定时器时钟。

RCC_APB1PeriphClockCmd();

② 初始化定时器，配置ARR,PSC。

TIM_TimeBaseInit();

③ 开启定时器中断，配置NVIC。

NVIC_Init();

④ 使能定时器。

TIM_Cmd();

⑥ 编写中断服务函数。

TIMx_IRQHandler();

✓ 手把手写定时器中断实验

◆ 程序要求

通过定时器中断配置，每500ms中断一次，然后中断服务函数中控制LED实现LED1状态取反（闪烁）。

Tout（溢出时间）=（ARR+1)(PSC+1)/Tclk

GO!!

谢谢您对“正点原子”团队的支持

硬件平台：正点原子STM32开发板
版权所有：广州市星翼电子科技有限公司
淘宝店铺：http://eboard.taobao.com
技术论坛：www.openedv.com