

stm32 TIM定时器 PWM脉冲输出[操作寄存器+库函数]

前一篇：[stm32 TIM定时器\[操作寄存器+库函数\]](#)

作者：Changing 发表时间：07-04 09:45 分类：电子相关 No Comments

后一篇：[stm32 独立看门狗\[操作寄存器+库函数\]](#)

脉冲调制（PWM）是利用微处理器对数字输出来对模拟电路的一种非常有效的技术。简单点说就是对确定频率的信号，调整其占空比。

stm32的定时器除了TIM6和TIM7外，其他定时器都可以产生PWM输出。其中高级定时器TIM1和TIM8可以产生多达7路的PWM输出。通用定时器可以产生4路的PWM输出。

在[stm32 TIM定时器\[操作寄存器+库函数\]](#) 中我们是通过在中断中，翻转指定引脚的电平。在stm32中可以通过配置一个**捕获/比较模式寄存器(TIMx_CCMR)**，设置通道引脚输出模式为PWM脉冲模式，在计时器计数到捕获/比较模式寄存器的值，指定引脚会输出一个有效电平，这样就可以通过定时器直接产生 PWM脉冲。**这种方式下不需要开启中断。**

- 这里说有效电平是因为这个电平不一定为1，这个在 **捕获/比较使能寄存器(TIMx_CCER)**中可以设置有效电平的极性。
- 指定引脚不是任意的，这个stm32对每个定时器通道有特定的引脚对应 对应关系如下

TIMx_CHx 对应的I/O口就是此通道对应的引脚

GPIOA		GPIOB		GPIOC		GPIOD		GPIOE		GPIOF		GPIOG		GPIOH		GPIOI	
PA0	21	PA0-WKUP/USART2_CTS/ADC_IN0/TIM2_CH1_ETR	PB0	35	PB0/ADC_TIM8/TIM3_CH3	PC0	15	PC0/ADC_IN10	PD0	81	PD0	PE0	97	PE0	PF0	1	PF0
PA1	24	PA1/USART2_RTS/ADC_IN1/TIM2_CH2	PB1	36	PB1/ADC_TIM9/TIM3_CH4	PC1	16	PC1/ADC_IN11	PD1	82	PD1	PE1	98	PE1	PF1	2	PF1
PA2	25	PA2/USART2_TX/ADC_IN2/TIM2_CH3	PB2	37	PB2/BOOT1	PC2	17	PC2/ADC_IN12	PD2	83	PD2	PE2	1	PE2	PF2	3	PF2
PA3	26	PA3/USART2_RX/ADC_IN3/TIM2_CH4	PB3	38	PB3/ITDO	PC3	18	PC3/ADC_IN13	PD3	84	PD3	PE3	2	PE3	PF3	4	PF3
PA4	29	PA4/SPI1_NSS/USART2_CK/ADC_IN4	PB4	39	PB4/ITRST	PC4	19	PC4/ADC_IN14	PD4	85	PD4	PE4	3	PE4	PF4	5	PF4
PA5	30	PA5/SPI1_SCK/ADC_IN5	PB5	40	PB5/I2C1_SMBAL	PC5	20	PC5/ADC_IN15	PD5	86	PD5	PE5	4	PE5	PF5	6	PF5
PA6	31	PA6/SPI1_MISO/ADC_IN6/TIM3_CH1	PB6	41	PB6/I2C1_SCL/TIM4_CH1	PC6	21	PC6/ADC_IN16	PD6	87	PD6	PE6	5	PE6	PF6	7	PF6
PA7	32	PA7/SPI1_MOSI/ADC_IN7/TIM3_CH2	PB7	42	PB7/I2C1_SDA/TIM4_CH2	PC7	22	PC7/ADC_IN17	PD7	88	PD7	PE7	6	PE7	PF7	8	PF7
PA8	67	PA8/USART1_CK/TIM1_CH1/MCO	PB8	43	PB8/TIM4_CH3	PC8	23	PC8/ADC_IN18	PD8	89	PD8	PE8	7	PE8	PF8	9	PF8
PA9	68	PA9/USART1_TX/TIM1_CH2	PB9	44	PB9/TIM4_CH4	PC9	24	PC9/ADC_IN19	PD9	90	PD9	PE9	8	PE9	PF9	10	PF9
PA10	69	PA10/USART1_RX/TIM1_CH3	PB10	45	PB10/I2C2_SCL/USART3_TX	PC10	25	PC10/ADC_IN20	PD10	91	PD10	PE10	9	PE10	PF10	11	PF10
PA11	70	PA11/USART1_CTSCANRX/USBDP(2)/TIM1_CH4	PB11	46	PB11/I2C2_SDA/USART3_RX	PC11	26	PC11/ADC_IN21	PD11	92	PD11	PE11	10	PE11	PF11	12	PF11
PA12	71	PA12/USART1_RTSCANTX/USBDP(2)/TIM1_ETR	PB12	47	PB12/SPI2_NSS/I2C2_SMBAL/USART3_CK/TIM1_BKIN	PC12	27	PC12/ADC_IN22	PD12	93	PD12	PE12	11	PE12	PF12	13	PF12
PA13	72	PA13/TMS-SWDAT	PB13	48	PB13/SPI2_SCK/USART3_CTS/TIM1_CH1N	PC13	28	PC13/ADC_IN23	PD13	94	PD13	PE13	12	PE13	PF13	14	PF13
PA14	76	PA14/TCK-SWCLK	PB14	49	PB14/SPI2_MISO/USART3_RTS/TIM1_CH2N	PC14	29	PC14/ADC_IN24	PD14	95	PD14	PE14	13	PE14	PF14	15	PF14
PA15	77	PA15/ITDI	PB15	50	PB15/SPI2_MOSI/TIM1_CH3N	PC15	30	PC15/OSC32_OUT	PD15	96	PD15	PE15	14	PE15	PF15	16	PF15
PA16	23	PA16/USART2_TX/ADC_IN2/TIM2_CH3	PB16	37	PB16/BOOT1	PC16	17	PC16/ADC_IN12	PD16	83	PD16	PE16	1	PE16	PF16	3	PF16
PA17	24	PA17/USART2_RX/ADC_IN3/TIM2_CH4	PB17	38	PB17/ITDO	PC17	18	PC17/ADC_IN13	PD17	84	PD17	PE17	2	PE17	PF17	4	PF17
PA18	25	PA18/USART2_TX/ADC_IN2/TIM2_CH3	PB18	39	PB18/ITRST	PC18	19	PC18/ADC_IN14	PD18	85	PD18	PE18	3	PE18	PF18	5	PF18
PA19	26	PA19/USART2_RX/ADC_IN3/TIM2_CH4	PB19	40	PB19/I2C1_SMBAL	PC19	20	PC19/ADC_IN15	PD19	86	PD19	PE19	4	PE19	PF19	6	PF19
PA20	27	PA20/USART2_TX/ADC_IN2/TIM2_CH3	PB20	41	PB20/I2C1_SCL/TIM4_CH1	PC20	21	PC20/ADC_IN16	PD20	87	PD20	PE20	5	PE20	PF20	7	PF20
PA21	28	PA21/USART2_RX/ADC_IN3/TIM2_CH4	PB21	42	PB21/I2C1_SDA/TIM4_CH2	PC21	22	PC21/ADC_IN17	PD21	88	PD21	PE21	6	PE21	PF21	8	PF21
PA22	29	PA22/USART2_TX/ADC_IN2/TIM2_CH3	PB22	43	PB22/ITRST	PC22	23	PC22/ADC_IN18	PD22	89	PD22	PE22	7	PE22	PF22	9	PF22
PA23	30	PA23/USART2_RX/ADC_IN3/TIM2_CH4	PB23	44	PB23/I2C1_SMBAL	PC23	24	PC23/ADC_IN19	PD23	90	PD23	PE23	8	PE23	PF23	10	PF23
PA24	31	PA24/USART2_TX/ADC_IN2/TIM2_CH3	PB24	45	PB24/I2C1_SCL/TIM4_CH1	PC24	25	PC24/ADC_IN20	PD24	91	PD24	PE24	9	PE24	PF24	11	PF24
PA25	32	PA25/USART2_RX/ADC_IN3/TIM2_CH4	PB25	46	PB25/I2C1_SDA/TIM4_CH2	PC25	26	PC25/ADC_IN21	PD25	92	PD25	PE25	10	PE25	PF25	12	PF25
PA26	33	PA26/USART2_TX/ADC_IN2/TIM2_CH3	PB26	47	PB26/ITRST	PC26	27	PC26/ADC_IN22	PD26	93	PD26	PE26	11	PE26	PF26	13	PF26
PA27	34	PA27/USART2_RX/ADC_IN3/TIM2_CH4	PB27	48	PB27/I2C1_SMBAL	PC27	28	PC27/ADC_IN23	PD27	94	PD27	PE27	12	PE27	PF27	14	PF27
PA28	35	PA28/USART2_TX/ADC_IN2/TIM2_CH3	PB28	49	PB28/I2C1_SCL/TIM4_CH1	PC28	29	PC28/ADC_IN24	PD28	95	PD28	PE28	13	PE28	PF28	15	PF28
PA29	36	PA29/USART2_RX/ADC_IN3/TIM2_CH4	PB29	50	PB29/I2C1_SDA/TIM4_CH2	PC29	30	PC29/ADC_IN25	PD29	96	PD29	PE29	14	PE29	PF29	16	PF29
PA30	37	PA30/USART2_TX/ADC_IN2/TIM2_CH3	PB30	51	PB30/ITRST	PC30	31	PC30/ADC_IN26	PD30	97	PD30	PE30	15	PE30	PF30	17	PF30
PA31	38	PA31/USART2_RX/ADC_IN3/TIM2_CH4	PB31	52	PB31/I2C1_SMBAL	PC31	32	PC31/ADC_IN27	PD31	98	PD31	PE31	16	PE31	PF31	18	PF31
PA32	39	PA32/USART2_TX/ADC_IN2/TIM2_CH3	PB32	53	PB32/I2C1_SCL/TIM4_CH1	PC32	33	PC32/ADC_IN28	PD32	99	PD32	PE32	17	PE32	PF32	19	PF32
PA33	40	PA33/USART2_RX/ADC_IN3/TIM2_CH4	PB33	54	PB33/I2C1_SDA/TIM4_CH2	PC33	34	PC33/ADC_IN29	PD33	100	PD33	PE33	18	PE33	PF33	20	PF33
PA34	41	PA34/USART2_TX/ADC_IN2/TIM2_CH3	PB34	55	PB34/ITRST	PC34	35	PC34/ADC_IN30	PD34	101	PD34	PE34	19	PE34	PF34	21	PF34
PA35	42	PA35/USART2_RX/ADC_IN3/TIM2_CH4	PB35	56	PB35/I2C1_SMBAL	PC35	36	PC35/ADC_IN31	PD35	102	PD35	PE35	20	PE35	PF35	22	PF35
PA36	43	PA36/USART2_TX/ADC_IN2/TIM2_CH3	PB36	57	PB36/I2C1_SCL/TIM4_CH1	PC36	37	PC36/ADC_IN32	PD36	103	PD36	PE36	21	PE36	PF36	23	PF36
PA37	44	PA37/USART2_RX/ADC_IN3/TIM2_CH4	PB37	58	PB37/I2C1_SDA/TIM4_CH2	PC37	38	PC37/ADC_IN33	PD37	104	PD37	PE37	22	PE37	PF37	24	PF37
PA38	45	PA38/USART2_TX/ADC_IN2/TIM2_CH3	PB38	59	PB38/ITRST	PC38	39	PC38/ADC_IN34	PD38	105	PD38	PE38	23	PE38	PF38	25	PF38
PA39	46	PA39/USART2_RX/ADC_IN3/TIM2_CH4	PB39	60	PB39/I2C1_SMBAL	PC39	40	PC39/ADC_IN35	PD39	106	PD39	PE39	24	PE39	PF39	26	PF39
PA40	47	PA40/USART2_TX/ADC_IN2/TIM2_CH3	PB40	61	PB40/I2C1_SCL/TIM4_CH1	PC40	41	PC40/ADC_IN36	PD40	107	PD40	PE40	25	PE40	PF40	27	PF40
PA41	48	PA41/USART2_RX/ADC_IN3/TIM2_CH4	PB41	62	PB41/I2C1_SDA/TIM4_CH2	PC41	42	PC41/ADC_IN37	PD41	108	PD41	PE41	26	PE41	PF41	28	PF41
PA42	49	PA42/USART2_TX/ADC_IN2/TIM2_CH3	PB42	63	PB42/ITRST	PC42	43	PC42/ADC_IN38	PD42	109	PD42	PE42	27	PE42	PF42	29	PF42
PA43	50	PA43/USART2_RX/ADC_IN3/TIM2_CH4	PB43	64	PB43/I2C1_SMBAL	PC43	44	PC43/ADC_IN39	PD43	110	PD43	PE43	28	PE43	PF43	30	PF43
PA44	51	PA44/USART2_TX/ADC_IN2/TIM2_CH3	PB44	65	PB44/I2C1_SCL/TIM4_CH1	PC44	45	PC44/ADC_IN40	PD44	111	PD44	PE44	29	PE44	PF44	31	PF44
PA45	52	PA45/USART2_RX/ADC_IN3/TIM2_CH4	PB45	66	PB45/I2C1_SDA/TIM4_CH2	PC45	46	PC45/ADC_IN41	PD45	112	PD45	PE45	30	PE45	PF45	32	PF45
PA46	53	PA46/USART2_TX/ADC_IN2/TIM2_CH3	PB46	67	PB46/ITRST	PC46	47	PC46/ADC_IN42	PD46	113	PD46	PE46	31	PE46	PF46	33	PF46
PA47	54	PA47/USART2_RX/ADC_IN3/TIM2_CH4	PB47	68	PB47/I2C1_SMBAL	PC47	48	PC47/ADC_IN43	PD47	114	PD47	PE47	32	PE47	PF47	34	PF47
PA48	55	PA48/USART2_TX/ADC_IN2/TIM2_CH3	PB48	69	PB48/I2C1_SCL/TIM4_CH1	PC48	49	PC48/ADC_IN44	PD48	115	PD48	PE48	33	PE48	PF48	35	PF48
PA49	56	PA49/USART2_RX/ADC_IN3/TIM2_CH4	PB49	70	PB49/I2C1_SDA/TIM4_CH2	PC49	50	PC49/ADC_IN45	PD49	116	PD49	PE49	34	PE49	PF49	36	PF49
PA50	57	PA50/USART2_TX/ADC_IN2/TIM2_CH3	PB50	71	PB50/ITRST	PC50	51	PC50/ADC_IN46	PD50	117	PD50	PE50	35	PE50	PF50	37	PF50
PA51	58	PA51/USART2_RX/ADC_IN3/TIM2_CH4	PB51	72	PB51/I2C1_SMBAL	PC51	52	PC51/ADC_IN47	PD51	118	PD51	PE51	36	PE51	PF51	38	PF51
PA52	59	PA52/USART2_TX/ADC_IN2/TIM2_CH3	PB52	73	PB52/I2C1_SCL/TIM4_CH1	PC52	53	PC52/ADC_IN48	PD52	119	PD52	PE52	37	PE52	PF52	39	PF52
PA53	60	PA53/USART2_RX/ADC_IN3/TIM2_CH4	PB53	74	PB53/I2C1_SDA/TIM4_CH2	PC53	54	PC53/ADC_IN49	PD53	120	PD53	PE53	38	PE53	PF53	40	PF53
PA54	61	PA54/USART2_TX/ADC_IN2/TIM2_CH3	PB54	75	PB54/ITRST	PC54	55	PC54/ADC_IN50	PD54	121	PD54	PE54	39	PE54	PF54	41	PF54
PA55	62	PA55/USART2_RX/ADC_IN3/TIM2_CH4	PB55	76	PB55/I2C1_SMBAL	PC55	56	PC55/ADC_IN51	PD55	122	PD55	PE55	40	PE55	PF55	42	PF55
PA56	63	PA56/USART2_TX/ADC_IN2/TIM2_CH3	PB56	77	PB56/I2C1_SCL/TIM4_CH1	PC56	57	PC56/ADC_IN52	PD56	123	PD56	PE56	41	PE56	PF56	43	PF56
PA57	64	PA57/USART2_RX/ADC_IN3/TIM2_CH4	PB57	78	PB57/I2C1_SDA/TIM4_CH2	PC57	58	PC57/ADC_IN53	PD57	124	PD57	PE57	42	PE57	PF57	44	PF57
PA58	65	PA58/USART2_TX/ADC_IN2/TIM2_CH3	PB58	79	PB58/ITRST	PC58	59	PC58/ADC_IN54	PD58	125	PD58	PE58	43	PE58	PF58	45	PF58
PA59	66	PA59/USART2_RX/ADC_IN3/TIM2_CH4	PB59	80	PB59/I2C1_SMBAL	PC59	60	PC59/ADC_IN55	PD59	126	PD59	PE59	44	PE59	PF59	46	PF59
PA60	67	PA60/USART2_TX/ADC_IN2/TIM2_CH3	PB60	81	PB60/I2C1_SCL/TIM4_CH1	PC60	61	PC60/ADC_IN56	PD60	127	PD60	PE60	45	PE60	PF60	47	PF60
PA61	68	PA61/USART2_RX/ADC_IN3/TIM2_CH4	PB61	82	PB61/I2C1_SDA/TIM4_CH2	PC61	62	PC61/ADC_IN57	PD61	128	PD61	PE61	46	PE61	PF61	48	PF61
PA62	69	PA62/USART2_TX/ADC_IN2/TIM2_CH3	PB62	83	PB62/ITRST	PC62	63	PC62/ADC_IN58	PD62	129	PD62	PE62	47	PE62	PF62	49	PF62
PA63	70	PA63/USART2_RX/ADC_IN3/TIM2_CH4	PB63	84	PB63/I2C1_SMBAL	PC63	64	PC63/ADC_IN59	PD63	130	PD63	PE63	48	PE63	PF63	50	PF63
PA64	71	PA64/USART2_TX/ADC_IN2/TIM2_CH3	PB64	85	PB64/I2C1_SCL/TIM4_CH1	PC64	65	PC64/ADC_IN60	PD64	131	PD64	PE64	49	PE64	PF64	51	PF64
PA65	72	PA65/USART2_RX/ADC_IN3/TIM2_CH4	PB65	86	PB65/I2C1_SDA/TIM4_CH2	PC65	66	PC65/ADC_IN61	PD65	132	PD65	PE65	50	PE65	PF65	52	PF65
PA66	73	PA66/USART2_TX/ADC_IN2/TIM2_CH3	PB66	87	PB66/ITRST	PC66	67	PC66/ADC_IN62	PD66	133	PD66	PE66	51	PE66	PF66	53	PF66
PA67	74	PA67/USART2_RX/ADC_IN3/TIM2_CH4	PB67	88	PB67/I2C1_SMBAL	PC67	68	PC67/ADC_IN63	PD67	134	PD67	PE67	52	PE67	PF67	54	PF67
PA68	75	PA68/USART2_TX/ADC_IN2/TIM2_CH3	PB68	89	PB68/I2C1_SCL/TIM4_CH1	PC68	69	PC68/ADC_IN64	PD68	135	PD68	PE68	53	PE68	PF68	55	PF68
PA69	76	PA69/USART2_RX/ADC_IN3/TIM2_CH4	PB69	90	PB69/I2C1_SDA/TIM4_CH2	PC69	70	PC69/ADC_IN65	PD69	136	PD69	PE69	54	PE69	PF69	56	PF69
PA70	77	PA70/USART2_TX/ADC_IN2/TIM2_CH3	PB70	91	PB70/ITRST	PC70	71	PC70/ADC_IN66	PD70	137	PD70	PE70	55	PE70	PF70	57	PF70
PA71	78	PA71/USART2_RX/ADC_IN3/TIM2_CH4	PB71	92	PB71/I2C1_SMBAL	PC71	72	PC71/ADC_IN67	PD71	138	PD71	PE71	56	PE71	PF71	58	PF71
PA72	79	PA72/USART2_TX/ADC_IN2/TIM2_CH3	PB72	93	PB72/I2C1_SCL/TIM4_CH1	PC72	73	PC72/ADC_IN68	PD72	139	PD72	PE72	57	PE72	PF72	59	PF72
PA73	80	PA73/USART2_RX/ADC_IN3/TIM2_CH4	PB73	94	PB73/I2C1_SDA/TIM4_CH2												

可以看出 TIM2的 OC通道 1-4 对应的就是 GPIOA 0-3

此例直接操作寄存器实现 Led灯由暗到亮再由亮到暗的呼吸灯效果。库函数实现用PWM脉冲输出模式，产生4个不同频率的脉冲，让led闪烁。

直接操作寄存器

通用定时器的每个通道都有6种输出模式，其中有两种PWM模式。通过捕获/比较模式寄存器1(TIMx_CCMR1)设定，由OC1M[2:0]三位决定。6种模式如下：

- 000：冻结。输出比较寄存器TIMx_CCR1与计数器TIMx_CNT间的比较对OC1REF不起作用；
- 001：匹配时设置通道1为有效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1)相同时，强制OC1REF为高。

记时设置通道1为无效电平。当计数器TIMx_CNT的值与捕获/比较寄存器1 (TIMx_CCR1)相同时，强制OC1REF为低。

专。当TIMx_CCR1=TIMx_CNT时，翻转OC1REF的电平。

引为无效电平。强制OC1REF为低。

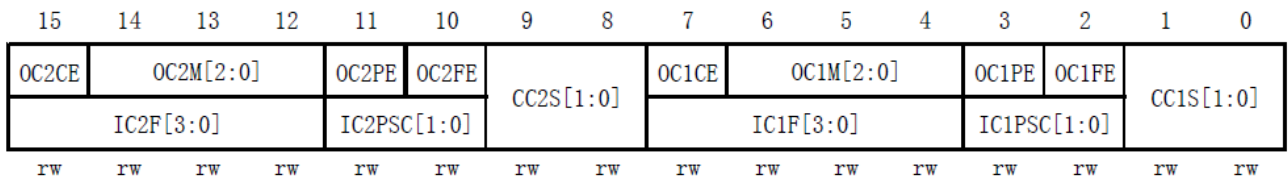
引为有效电平。强制OC1REF为高。

- 110：PWM模式1 - 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。

- 111：PWM模式2 - 在向上计数时，一旦TIMx_CNT<TIMx_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx_CNT>TIMx_CCR1时通道1为有效电平，否则为无效电平。

两种PWM模式，区别在于通道的电平极性是相反的。

首先需要设定TIMx_CCMR1寄存器：



OCxM[2:0]已经做了介绍，OC2CE：输出比较2清0使能 OC2PE：输出比较2预装载使能
通过设定OC2M[2:0]为 110/111 为PWM脉冲输出模式。

设定TIMx_CCER寄存器相关位，使能通道输出，还可以设置有效电平极性。

最后一个就是调整占空比的关键寄存器，捕获/比较寄存器(TIMx_CCRx)，低16位有效，这个寄存器已经使用过，**要实现PWM脉冲的占空比可调的原理就是不断改变这个寄存器的值。**

要实现led亮暗的渐变，PWM的频率不能太低，低于50Hz的时候就会明显感觉到闪烁。这里用8khz的频率，调整PWM输出占空比，从0到不断增大其占空比，再递减为0。

代码如下：（system.h 和 stm32f10x_it.h 等相关代码参照 [stm32 直接操作寄存器开发环境配置](#)）

User/main.c

```
01 #include <stm32f10x_lib.h>
02 #include "system.h"
03 #include "tim.h"
04
05 void Gpio_Init(void);
06
07 int main(void)
08 {
09     u32 var=0,flag=0;
10
11     Rcc_Init(9);           //系统时钟设置
12
13     // 相关TIM_x,CCR_x参数定义tim.h文件
14
15     Tim_Init(TIM_3,900,0); //初始化TIM3定时器，设定重装值和分频值
16
17     Tim_OC_Set(TIM_3,OC_2,7); //设定TIM3 通道1为PWM输出模式
18
19     Gpio_Init();
20
21     while(1){
22
23         delay(5000);      //延时5ms
24
25         if(flag){
26             var--;
27         }else{
28             var++;
29         }
30
31         if(var>300) flag = 1;
32
33         if(var == 0) flag = 0;
34
35         Tim_CCR_Set(TIM_3,OC_2,var);
36     }
37
38 }
39
40
41 void Gpio_Init(void)
42 {
43     RCC->APB2ENR|=1<<2;    //使能PORTA时钟
44
45 }
```

```
GPIOA->CRL&=0X0FFFFFFF;//PA7输出
GPIOA->CRL|=0XB0000000;//复用功能输出
```

Library/src/tm.c

```
001 #include <stm32f10x_lib.h>
002 #include "tim.h"
003
004 //通用定时器初始化
005 //参数说明: TIM_x 为选择定时器 TIM_1为通用寄存器1又一次类推(定义于tim.h), arr为自动重装值; psc 为时钟预分频数
006 //要使用定时器的其他函数, 必须先调用此函数, 因为时钟在这个函数中开启
007 //TIM3用于PWM输出已测试
008 //待完善 目前只支持TIM2
009 //其他定时器只做了开启时钟处理
010 void Tim_Init(u8 TIM_x,u16 arr,u16 psc)
011 {
012     switch(TIM_x)
013     {
014         case 1 :{ RCC->APB2ENR |=1<<11; break; } //TIM1高级定时器设置
015         case 2 :{ //TIM2通用定时器设置
016
017             RCC->APB1ENR |=1<<0;
018
019             TIM2->ARR = arr; //设定自动重装值
020             TIM2->PSC = psc; //设定预分频值
021             TIM2->DIER |= 1<<0; //允许更新中断
022             TIM2->DIER |= 1<<6; //允许触发中断
023
024             TIM2->CR1 |= 0x81; //使能定时器, 自动重装允许
025
026             break;
027         }
028
029         case 3 :{
030
031             RCC->APB1ENR |=1<<1;
032
033             TIM3->ARR = arr; //设定自动重装值
034             TIM3->PSC = psc; //设定预分频值
035             TIM3->DIER |= 1<<0; //允许更新中断
036             TIM3->DIER |= 1<<6; //允许触发中断
037
038             TIM3->CR1 |= 0x81; //使能定时器
039
040             break;
041         }
042
043         case 4 :{
044
045             RCC->APB1ENR |=1<<2;
046
047             TIM4->ARR = arr; //设定自动重装值
048             TIM4->PSC = psc; //设定预分频值
049             TIM4->DIER |= 1<<0; //允许更新中断
050             TIM4->DIER |= 1<<6; //允许触发中断
051
052             TIM4->CR1 |= 0x01; //使能定时器
053
054             break;
055         }
056
057         case 5 :{
058
059             RCC->APB1ENR |=1<<3;
060
061             TIM5->ARR = arr; //设定自动重装值
062             TIM5->PSC = psc; //设定预分频值
063             TIM5->DIER |= 1<<0; //允许更新中断
064             TIM5->DIER |= 1<<6; //允许触发中断
065
066             TIM5->CR1 |= 0x01; //使能定时器
067
068             break;
069         }
070
071         case 6 :{ RCC->APB1ENR |=1<<4; break; }
072         case 7 :{ RCC->APB1ENR |=1<<5; break; }
073         case 8 :{ RCC->APB2ENR |=1<<13; break; }
074     }
075 }
076
077 //捕获比较值设定函数
078 //参数说明:
079 // TIM_x 为选择定时器 TIM_1为通用寄存器1又一次类推(定义于tim.h)
080 // OC_x 为选择通道, 以确定捕获/比较寄存器(1~4)(定义于tim.h)
081 // val 为要设定的捕获/比较寄存器的值
082 // TIM3,OC_2 用于PWM输出已测试
083 // 待完善, 目前只支持TIM2
084 void Tim_CCR_Set(u8 TIM_x,u8 OC_x,u32 val)
085 {
086     switch(TIM_x)
087     {
088         case 1 :{ break;}
089         case 2 :{
090
091             TIM2->DIER |= 1 << OC_x; //开启相应允许捕获/比较中断
092         }
093     }
094 }
```

```

097
098
099
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187

switch(OC_x){
    case 1: {
        TIM2 ->CCR1 = val;          //设置捕获/比较1的值
        break;
    }
    case 2: {
        TIM2 ->CCR2 = val;          //设置捕获/比较2的值
        break;
    }
    case 3: {
        TIM2 ->CCR3 = val;          //设置捕获/比较3的值
        break;
    }
    case 4: {
        TIM2 ->CCR4 = val;          //设置捕获/比较4的值
        break;
    }
}
break;
}
case 3 :{
    //TIM3->DIER |= 1 << OC_x;      //开启相应允许捕获/比较中断
    switch(OC_x){
        case 1: {
            TIM3 ->CCR1 = val;      //设置捕获/比较1的值
            break;
        }
        case 2: {
            TIM3 ->CCR2 = val;      //设置捕获/比较2的值
            break;
        }
        case 3: {
            TIM3 ->CCR3 = val;      //设置捕获/比较3的值
            break;
        }
        case 4: {
            TIM3 ->CCR4 = val;      //设置捕获/比较4的值
            break;
        }
    }
    break;
}
case 4 :{ break;}
case 5 :{ break;}
case 6 :{ break;}
case 7 :{ break;}
case 8 :{ break;}
}
}
//定时器通道引脚输出模式设定函数
//参数说明:
// TIM_x 为选择定时器 TIM_1为通用寄存器1又一次类推（定义于tim.h）
// OC_x 为选择输出通道选择（1~4）（定义于tim.h）
// Mode 为选择通道对应引脚输出模式（0~7）
// TIM3,OC_2 用于PWM输出已测试
// 待完善，目前只支持TIM2
void Tim_OC_Set(u8 TIM_x,u8 OC_x,u8 Mode)
{
    switch(TIM_x)
    {
        case 1 :{ break;}
        case 2 :{
            switch(OC_x){
                case 1: {
                    TIM2 ->CCMR1 |= Mode <<4;      //设定引脚输出模式
                    TIM2 ->CCMR1 |= 1<<3;          //允许预装载
                    //TIM2 ->CCER |= 1<<2;          //引脚输出低电平为有效
                    TIM2 ->CCER |= 1<<0;          //OC1 输出使能
                    break;
                }
                case 2: {
                    TIM2 ->CCMR1 |= Mode <<12;      //设定引脚输出模式
                    TIM2 ->CCMR1 |= 1<<11;          //允许预装载
                    //TIM2 ->CCER |= 1<<5;          //引脚输出低电平为有效
                    TIM2 ->CCER |= 1<<4;          //OC2 输出使能
                    break;
                }
            }
        }
    }
}

```

```

    }

    case 3: {
        TIM2 ->CCMR2 |= Mode <<4; // 设定引脚输出模式
        TIM2 ->CCMR2 |= 1<<3; // 允许预装载

        //TIM2 ->CCER |= 1<<9; // 引脚输出低电平为有效
        TIM2 ->CCER |= 1<<8; // OC3 输出使能
        break;
    }

    case 4: {
        TIM2 ->CCMR2 |= Mode <<12; // 设定引脚输出模式
        TIM2 ->CCMR2 |= 1<<11; // 允许预装载

        //TIM2 ->CCER |= 1<<5; // 引脚输出低电平为有效
        TIM2 ->CCER |= 1<<4; // OC1 输出使能
        break;
    }
}

break;
}

case 3 :{
    switch(OC_x){
        case 1: {
            TIM3 ->CCMR1 |= Mode <<4; // 设定引脚输出模式
            TIM3 ->CCMR1 |= 1<<3; // 允许预装载

            //TIM3 ->CCER |= 1<<2; // 引脚输出低电平为有效
            TIM3 ->CCER |= 1<<0; // OC1 输出使能
            break;
        }

        case 2: {
            TIM3 ->CCMR1 |= Mode <<12; // 设定引脚输出模式
            TIM3 ->CCMR1 |= 1<<11; // 允许预装载

            TIM3 ->CCER |= 1<<5; // 引脚输出低电平为有效
            TIM3 ->CCER |= 1<<4; // OC2 输出使能
            break;
        }

        case 3: {
            TIM3 ->CCMR2 |= Mode <<4; // 设定引脚输出模式
            TIM3 ->CCMR2 |= 1<<3; // 允许预装载

            //TIM3 ->CCER |= 1<<9; // 引脚输出低电平为有效
            TIM3 ->CCER |= 1<<8; // OC3 输出使能
            break;
        }

        case 4: {
            TIM3 ->CCMR2 |= Mode <<12; // 设定引脚输出模式
            TIM3 ->CCMR2 |= 1<<11; // 允许预装载

            //TIM3 ->CCER |= 1<<5; // 引脚输出低电平为有效
            TIM3 ->CCER |= 1<<4; // OC1 输出使能
            break;
        }
    }

    break;
}

case 4 :{ break;}
case 5 :{ break;}
case 6 :{ break;}
case 7 :{ break;}
case 8 :{ break;}
}
}
}

```

Library/inc/tim.h

```

01 #include <stm32f10x_lib.h>
02
03 #define TIM_1 0x01
04 #define TIM_2 0x02
05 #define TIM_3 0x03
06 #define TIM_4 0x04
07 #define TIM_5 0x05
08 #define TIM_6 0x06
09 #define TIM_7 0x07
10 #define TIM_8 0x08
11
12 #define OC_1 0x01
13 #define OC_2 0x02
14 #define OC_3 0x03
15 #define OC_4 0x04
16
17
18 void Tim_Init(u8 TIM_x,u16 arr,u16 psc);
19 void Tim_CCR_Set(u8 TIM_x,u8 OC_x,u32 val);
20 void Tim_OC_Set(u8 TIM_x,u8 OC_x,u8 Mode);

```

注意的是 Led的连接方式，我的led是低电平亮的，如果你的Led是高电平点亮，可以设置通道引脚输出极性为高电平有效。在Tim_数中可以设置，此例中选用TIM3的OC2通道，只需要注释 TIM3 -> CCER |= 1<<5; //引脚输出低电平为有效 这句代码即可。

要输出PWM脉冲 必须要 将io 设置为复用推挽

代码如下：main.c

```
001 | #include "stm32f10x.h"
002 |
003 | vu16 CCR1_Val = 60000;
004 | vu16 CCR2_Val = 30000;
005 | vu16 CCR3_Val = 15000;
006 | vu16 CCR4_Val = 7500;
007 |
008 | void RCC_Configuration(void);
009 | void GPIO_Configuration(void);
010 | void TIM_Configuration(void);
011 |
012 | int main(void)
013 | {
014 |
015 |     RCC_Configuration();
016 |     GPIO_Configuration();
017 |     TIM_Configuration();
018 |     while(1);
019 | }
020 |
021 | void TIM_Configuration(void)
022 | {
023 |     TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
024 |     TIM_OCInitTypeDef TIM_OCInitStructure;
025 |     TIM_TimeBaseStructure.TIM_Period = 65535;
026 |     TIM_TimeBaseStructure.TIM_Prescaler = 7199;
027 |     TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
028 |     TIM_TimeBaseInit(TIM2,&TIM_TimeBaseStructure);
029 |
030 |     //TIM_PrescalerConfig(TIM2,7199,TIM_PSCReloadMode_Immediate);
031 |
032 |     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
033 |     TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //使能TIM输出
034 |     TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
035 |     TIM_OCInitStructure.TIM_Pulse = CCR1_Val;
036 |     TIM_OC1Init(TIM2,&TIM_OCInitStructure);
037 |     TIM_OCInitStructure.TIM_Pulse = CCR2_Val;
038 |     TIM_OC2Init(TIM2,&TIM_OCInitStructure);
039 |     TIM_OCInitStructure.TIM_Pulse = CCR3_Val;
040 |     TIM_OC3Init(TIM2,&TIM_OCInitStructure);
041 |     TIM_OCInitStructure.TIM_Pulse = CCR4_Val;
042 |     TIM_OC4Init(TIM2,&TIM_OCInitStructure);
043 |
044 |     TIM_OC1PreloadConfig(TIM2,TIM_OCPreload_Enable);
045 |     TIM_OC2PreloadConfig(TIM2,TIM_OCPreload_Enable);
046 |     TIM_OC3PreloadConfig(TIM2,TIM_OCPreload_Enable);
047 |     TIM_OC4PreloadConfig(TIM2,TIM_OCPreload_Enable);
048 |
049 |     //TIM_ITConfig(TIM2,TIM_IT_CC1|TIM_IT_CC2|TIM_IT_CC3|TIM_IT_CC4,ENABLE);
050 |
051 |     TIM_Cmd(TIM2,ENABLE);
052 | }
053 |
054 |
055 |
056 |
057 | void GPIO_Configuration(void)
058 | {
059 |     GPIO_InitTypeDef GPIO_InitStructure;
060 |
061 |     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;
062 |     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
063 |     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; //设置为复用推挽
064 |     GPIO_Init(GPIOA , &GPIO_InitStructure);
065 | }
066 |
067 |
068 | void RCC_Configuration(void)
069 | {
070 |     /* 定义枚举类型变量 HSEStartUpStatus */
071 |     ErrorStatus HSEStartUpStatus;
072 |
073 |     /* 复位系统时钟设置*/
074 |     RCC_DeInit();
075 |     /* 开启HSE*/
076 |     RCC_HSEConfig(RCC_HSE_ON);
077 |     /* 等待HSE起振并稳定*/
078 |     HSEStartUpStatus = RCC_WaitForHSEStartUp();
079 |     /* 判断HSE起是否振成功，是则进入if()内部 */
080 |     if(HSEStartUpStatus == SUCCESS)
081 |     {
082 |         /* 选择HCLK (AHB) 时钟源为SYSCLK 1分频 */
083 |         RCC_HCLKConfig(RCC_SYSCLK_Div1);
084 |         /* 选择PCLK2时钟源为 HCLK (AHB) 1分频 */
```

```


RCC_PCLK2Config(RCC_HCLK_Div1);
/* 选择PCLK1时钟源为 HCLK (AHB) 2分频 */
RCC_PCLK1Config(RCC_HCLK_Div2);
/* 设置FLASH延时周期数为2 */
FLASH_SetLatency(FLASH_Latency_2);
/* 使能FLASH预取缓存 */
FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
/* 选择锁相环 (PLL) 时钟源为HSE 1分频, 倍频数为9, 则PLL输出频率为 8MHz * 9 = 72MHz */
RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
/* 使能PLL */
RCC_PLLCmd(ENABLE);
/* 等待PLL输出稳定 */
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
/* 选择SYSCLK时钟源为PLL */
RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
/* 等待PLL成为SYSCLK时钟源 */
while(RCC_GetSYSCLKSource() != 0x08);
}
/* 打开APB2总线上的GPIOA时钟*/
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA , ENABLE);

RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2 , ENABLE);
}

```

相关文章

- stm32 DA 数模转换
- stm32 驱动 触摸屏
- stm32 Fatfs 读写SD卡
- stm32 最小系统
- Uip + Stm32移植问题总结

 传播到腾讯微博

Tags: stm32, 库函数, TIM定时器, PWM, 占空比可调, 直接操作寄存器

前一篇：stm32 TIM定时器[操作寄存器+库函数]

后一篇：stm32 独立看门狗[操作寄存器+库函数]

Leave a Comment

<input type="text"/>	昵称 (必填*)
<input type="text"/>	电邮 (为保障隐私,将不被显示. [必填*])
<input type="text"/>	个人网站[要加上 http://]
<div></div>	

Submit Comment