**API of FTM**

默认使用总线时钟,若要调低PWM的频率则可以改用MCGFFCLK时钟

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 函数定义 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**PWM**

**void FTM\_PWM\_Init(**uint8 **Mode,** uint16 **Ftmn\_CHx\_PTnx,** uint32 **Freq,** float **Duty);**

**void FTM\_PWM\_Set\_Freq(**uint8**Ftmn,** uint8 **CHx,** uint32 **Freq);**

**void FTM\_PWM\_Set\_Duty(**uint8 **Ftmn,** uint8 **CHx,** float **Duty);**

//中断使能函数

**void FTM\_TimeOut\_Int\_Enable(**uint8 **Ftmn,** uint8 **CHx);**

**void FTM\_TimeOut\_Int\_Disable(**uint8 **Ftmn,** uint8 **CHx)**

//中断服务中用到的语句

**void FTM\_Clear\_TimeOut(**uint8 **Ftmn);**

**Input Capture**

**void FTM\_IC\_Init(**uint16 **Ftmn\_CHx\_PTnx,** uint8 **cfg);**

//中断使能函数

**void FTM\_Ch\_Int\_Enable(**uint8 **Ftmn,** uint8 **CHx);**

**void FTM\_Ch\_Int\_Dis(**uint8 **Ftmn,** uint8 **CHx);**

//中断服务中用到的语句

**void FTM\_Clear\_Channal\_Event(**uint8 **Ftmn,** uint8 **CHx);**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END of函数定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PWM函数功能介绍 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**Ftmn\_CHx\_PTnx:** //选择相应模块,通道及其引脚

**FTM0\_CH0\_PTC1 FTM1\_CH0\_PTA12 FTM1\_CH0\_PTB0**

**FTM0\_CH0\_PTA3 FTM0\_CH4\_PTA7 FTM1\_CH1\_PTA13**

**FTM0\_CH1\_PTA4 FTM0\_CH4\_PTD4 FTM1\_CH1\_PTA9**

**FTM0\_CH1\_PTC2 FTM0\_CH5\_PTD5 FTM1\_CH1\_PTB1**

**FTM0\_CH2\_PTA5 FTM0\_CH5\_PTA0 FTM2\_CH0\_PTA10**

**FTM0\_CH2\_PTC3 FTM0\_CH6\_PTD6 FTM2\_CH0\_PTB18**

**FTM0\_CH3\_PTA6 FTM0\_CH6\_PTA1 FTM2\_CH1\_PTA11**

**FTM0\_CH3\_PTC4 FTM0\_CH7\_PTD7 TM2\_CH1\_PTB19**

**FTM0\_CH7\_PTA2 FTM1\_CH0\_PTA8**

**FTM3\_CH0\_PTD0 FTM3\_CH0\_PTE5**

**FTM3\_CH1\_PTD1 FTM3\_CH1\_PTE6**

**FTM3\_CH2\_PTD2 FTM3\_CH2\_PTE7**

**FTM3\_CH3\_PTD3 FTM3\_CH3\_PTE8**

**FTM3\_CH4\_PTC8 FTM3\_CH4\_PTE9**

**FTM3\_CH5\_PTC9 FTM3\_CH5\_PTE10**

**FTM3\_CH6\_PTC10 FTM3\_CH6\_PTE11**

**FTM3\_CH7\_PTC11 FTM3\_CH7\_PTE12**

**void FTM\_PWM\_Init(**uint8 **Mode,** uint16 **Ftmn\_CHx\_PTnx,** uint32**Freq,** float**Duty);**

**功能:**设置PWM的模式是边沿对齐,还是中间对齐, 选择FTM的哪一个模块,哪个通道及其对应的引脚, 设置PWM的频率, 设置占空比

**参数:Mode : EPWM\_MODE,CPWM\_MODE**//中间对齐模式,边沿对齐模式

**Ftmn\_CHx\_PTnx: 在上面**

**Freq:** //选择PWM的频率

**CPWM EPWM**

建议:  **5~ 375000 10~7500000**

波形开始失真: **375~2000000 7500000~2500000**

**Duty: 0~1**

**void FTM\_PWM\_Set\_Freq(**uint8 **Ftmn,** uint8**CHx,** uint32 **Freq);**

**功能:**选择FTM的哪一个模块,哪个通道

设置PWM的频率

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**CHx: FTM\_CH0**~**FTM\_CH7**//选择FTM的通道,有0~7

**void FTM\_PWM\_Set\_Duty(**uint8 **Ftmn,** uint8 **CHx,** float **Duty);**

**功能:**选择FTM哪一个模块,哪个通道

设置占空比

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**CHx: FTM\_CH0**~**FTM\_CH7**//选择FTM的通道,有0~7

**Duty: 0~1**

**void FTM\_TimeOut\_Int\_Enable(**uint8 **Ftmn,** uint8 **CHx);**

**功能:**使能PWM溢出中断, 每个PWM周期进入一次中断

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**CHx: FTM\_CH0**~**FTM\_CH7**//选择FTM的通道,有0~7

**void FTM\_Clear\_TimeOut(**uint8 **Ftmn);**

**功能:**清除PWM溢出中断标志位,用在PWM的溢出中断服务函数中第一条语句

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END of PWM函数定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PWM例程 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

现象:先生成一个中间对齐,频率为100Hz,占空比为0.5的方波

然后变为频率为1000Hz,占空比为0.4的方波

再然后,每个PWM周期改变占空比(在PWM周期溢出中断服务函数中改变占空比)

**#include"all.h"**

uint32 **busclk = 75000000;**

uint8 **i=3;**

**void FTM1\_IRQHandler()**

**{**

**FTM\_Clear\_TimeOut(FTM1);**

**FTM\_PWM\_Set\_Duty(FTM1, FTM\_CH0, (float)(i)/100);**

**i = i + 2;**

**if(i>95)i=3;**

**}**

**void main(void)**

**{**

**SYS\_CLOCK\_SET(SYS\_CLOCK\_150M, 1, 2, 3, 6);**

**FTM\_PWM\_Init(CPWM\_MODE, FTM1\_CH0\_PTB0, 100, 0.5);**

**FTM\_PWM\_Set\_Duty(FTM1, FTM\_CH0, 0.4);**

**FTM\_PWM\_Set\_Freq(FTM1, FTM\_CH0,1000);**

**FTM\_TimeOut\_Int\_Enable(FTM1, FTM\_CH0);**

**while(1);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END of PWM例程 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Input Capture函数功能介绍 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**void FTM\_IC\_Init(**uint16 **Ftmn\_CHx\_PTnx,** uint8 **cfg);**

**功能:**选择FTM的哪一个模块,哪个通道及其对应的引脚

选择触发事件是上升沿,还是下降沿,还是双边沿

**参数:Ftmn\_CHx\_PTnx: 在上面**

**cfg: FTM\_Rising,FTM\_Falling,FTM\_Rising\_or\_Falling** //上升沿触发,下降沿触发,双边沿触发

**void FTM\_Ch\_Int\_Enable(**uint8 **Ftmn,** uint8 **CHx);**

**功能:**使能IC通道事件中断,每捕捉到一次事件(上升沿,下降沿,双边沿)进入一次中断

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**CHx: FTM\_CH0**~**FTM\_CH7**//选择FTM的通道,有0~7

**void FTM\_Clear\_Channal\_Event(**uint8 **Ftmn,** uint8 **CHx);**

**功能:**清除IC通道事件中断标志,用在IC捕捉到相应沿的中断服务函数中第一条语句

**参数: Ftmn: FTM0, FTM1, FTM2, FTM3**//选择FTM哪一个模块

**CHx: FTM\_CH0**~**FTM\_CH7**//选择FTM的通道,有0~7

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END of Input Capture函数定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Input Capture例程 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

现象:测得**Frequency**为10000Hz

**#include"all.h"**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  Input Capture的变量 **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

//没函数发生器,所以还没测试10Hz以下的,2MHz以上的.10Kz到2MHz的精度达到99%

//连接B0,C2.既可测试,B0作为PWM输出,C2捕捉

uint32 **current=0, original=0,Freq[10],tick = 0,Frequency;**

uint32 **FTM\_TimerOut\_flag = 0,QuTou\_QuWei\_NUM = 4;**

uint32 **busclk = 75000000;**

uint32 **Fre\_Frequece\_Index=0;**  //用于统计序列

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

uint32 **average(uint32 p[10]) //功能：去毛刺求平均值**

**{**

**int i = 0,j = 0;**

uint32 **sum = 0, temp = 0;**

**for (i = 0; i < 10; i++) {**

**for (j = 0; j < 10 - i; j++) {**

**if (p[j] > p[j + 1]) {**

**temp = p[j];**

**p[j] = p[j + 1];**

**p[j + 1] = temp;**

**}**

**}**

**}**

**for (i = QuTou\_QuWei\_NUM; i < (10 - QuTou\_QuWei\_NUM);i++)**

**sum = sum + p[i];**

**return sum / (10 - 2\*QuTou\_QuWei\_NUM);**

**}**

//the maximum frequency for the channel input signal to be

//detected correctly is system clock divided by 4,

**void FTM0\_IRQHandler()**

**{**

**//FTM0\_SC &= ~FTM\_SC\_TOF\_MASK;** //清除溢出中断标志位

**FTM\_Clear\_Channal\_Event(FTM0, FTM\_CH1);**

**current = FTM0\_C1V;**

// 当两次中断之间的时间差超过了一个计数周期时，需要补加FTM\_CountOut\_flag个0xFFFF值

**if(FTM\_TimerOut\_flag >= 1)**

**{**

**tick = (0xFFFF - original + (FTM\_TimerOut\_flag-1)\*0xFFFF + current);**

**FTM\_TimerOut\_flag = 0;**

**}**

// 若两次中断的响应在一个周期内，则直接用前一次的计数器值减去后一次计数器值

**else**

**tick =(current-original);**

**Freq[Fre\_Frequece\_Index] = busclk / tick;**

**Fre\_Frequece\_Index++;**

**if(Fre\_Frequece\_Index >= 10)**

**{**

**Frequency = average(Freq);**

**Fre\_Frequece\_Index = 0;**

**}**

**original=current;**

**}**

**void main(void)**

**{**

**SYS\_CLOCK\_SET(SYS\_CLOCK\_150M, 1, 2, 3, 6);**

**FTM\_PWM\_Init(EPWM\_MODE, FTM1\_CH0\_PTB0, 10000, 0.5);**

**FTM\_IC\_Init(FTM0\_CH1\_PTC2, FTM\_Falling);**

**FTM\_Ch\_Int\_Enable(FTM0, FTM\_CH1);**

**while(1)**

**{**

**if(FTM0\_SC & FTM\_SC\_TOF\_MASK)**

**{**

**FTM0\_SC &= ~FTM\_SC\_TOF\_MASK;**

**FTM\_TimerOut\_flag++;**

**}**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END of Input Capture例程 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**