文档版本	V1.0
发布日期	20191108

APT32F172 EPWM 应用开发指南

目录

1 概述

本文介绍了在APT32F172中使用EPWM的应用范例。

2. 适用的硬件

该例程使用于 APT32F172 开发板 APT-DB172

3. 应用方案代码说明

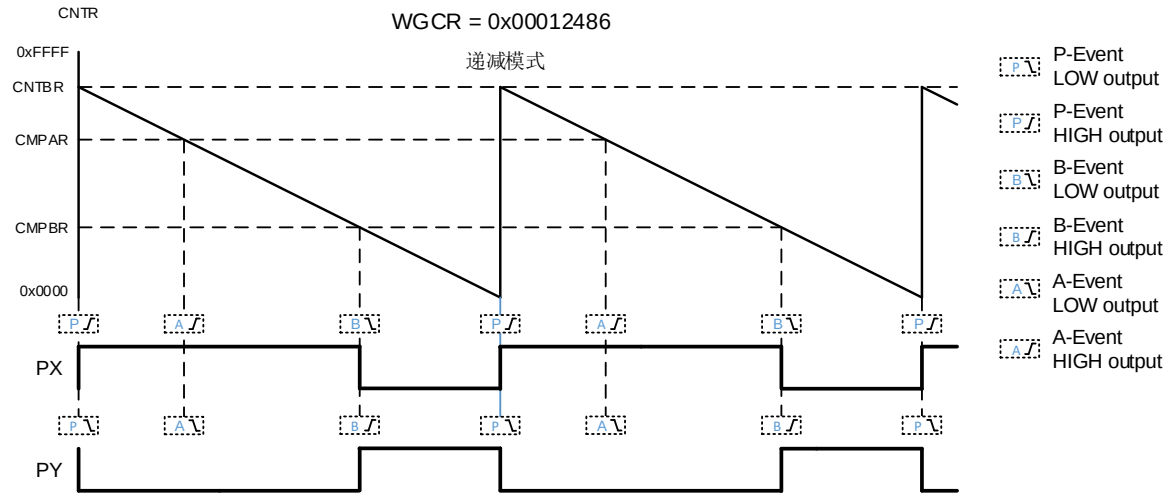
基于 APT32F172 完整的库文件系统，可以很方便的对 EPWM 进行配置。

EPWM 注意事项：

- EPWM 递增或递减递增模式下，计数开始前 SLPCNTx 的值需设置成跟 CNTR 一样
- EPWM 软锁止自增自减模式下， SLCON 的设置必须放在 SLPCMPxR 的后面。
- EPWM 设置触发延时功能时， TRGTDL 范围在： 0~14，若 TRGTDL=15,PWM 输出异常
- EPWM_CONTER_Configure();EPWM_PX_PY_Configure();EPWM_OUTPUT_Configure(); 在 主程序中不要配置 2 次以上，若要重新配置需要先设置 EPWM_software_reset();
- EPWM 单次触发时， 当 CMP 触发 PWM 时，此时 PWM 产生 STOP 事件，当次触发无效。解决办法：避免 CMP 触发周期小于 PWM 周期。
- EPWM 单次触发时， 当 CMP 触发 PWM 时，此时周期值比比较值小，当次触发无效。解决办法：在 PWM STOP 中断配置周期和比较寄存器值，且将 STOP 中断优先级设置最高。

3.1 六路 PWM 独立输出

波形输出示意图：



事件优先级：

优先级	触发事件	可以触发该事件的信号
1	S-Event	StartE（软件使能，或者比较器触发）/ StopE （软件停止，或者Hard Lock）
2	P-Event	PendE（周期结束）
3	C-Event	CENTERE （递增递减计数中间点）
4	B-Event	CMPBDME 和 CMPBUME（计数值与CMPB相等）
5	A-Event	CMPADME 和 CMPAUME（计数值与CMPA相等）

软件配置：

```
/*****************************************************************/
//EPWM Init
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
/*********************BLOCK VIEW*************************************/
/* - PX --- -- PWM_X */
/* PWM Engine -- ---PWM output Control --- */
/* -PY --- --PWM_Y */
/*************************************************************************/
void EPWM_CONFIG(void)
{
EPWM_RESET_VALUE(); //EPWM 所有寄存器复位赋值
EPWM_software_reset(); //EPWM 软件复位
EPWM_IO_Init(PWM_X0,0); //PWM_X0 初始化
EPWM_IO_Init(PWM_Y0,0); //PWM_Y0 初始化
EPWM_IO_Init(PWM_X1,0); //PWM_X1 初始化
EPWM_IO_Init(PWM_Y1,0); //PWM_Y1 初始化
EPWM_IO_Init(PWM_X2,0); //PWM_X2 初始化
```

```
EPWM_IO_Init(PWM_Y2,0); //PWM_Y2 初始化
EPWM_CONTER_Configure(EPWM_ContMode_decrease,EPWM_Conter_three,EMP_Overflow_Mode_Cont
inue,1,9);
//递减计数, EPMW_CLK=PCLK/(2^DIVN)/(DINM+1)=20M/2/(9+1)=1M=1US, 单次触发
EPWM_PX_PY_Configure(EPWM_P0X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM
_CentralEvent_NoChange,EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P0X CNTR=1000,CMPAR=500,CMPBR=0
EPWM_PX_PY_Configure(EPWM_P0Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_
CentralEvent_NoChange,EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P0Y CNTR=1000,CMPAR=500,CMPBR=0
EPWM_PX_PY_Configure(EPWM_P1X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM
_CentralEvent_NoChange,EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P1X CNTR=1000,CMPAR=500,CMPBR=0
EPWM_PX_PY_Configure(EPWM_P1Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_
CentralEvent_NoChange,EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P1Y CNTR=1000,CMPAR=500,CMPBR=0
EPWM_PX_PY_Configure(EPWM_P2X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM51 / 69
_CentralEvent_NoChange,EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P2X CNTR=1000,CMPAR=500,CMPBR=0
EPWM_PX_PY_Configure(EPWM_P2Y,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_
CentralEvent_NoChange,EPWM_EqCMPAEvent_OutHigh,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P2Y CNTR=1000,CMPAR=500,CMPBR=0
EPWM_OUTPUT_Configure(EPWM_PWM_X0OrPWM_Y0,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10); //PWM_X
PWM_Y 直接输出模式,输出端电平保持丌变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
EPWM_OUTPUT_Configure(EPWM_PWM_X1OrPWM_Y1,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10); //PWM_X
PWM_Y 直接输出模式,输出端电平保持丌变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
EPWM_OUTPUT_Configure(EPWM_PWM_X2OrPWM_Y2,EPWM_OUTSE_PXPYOutputDirect,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_NoChange,EPWM_SRCSEL_PX,0x10,0x10); //PWM_X
PWM_Y 直接输出模式,输出端电平保持丌变;RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
EPWM_Conter0_START(); //Count0 开启
EPWM_Conter1_START(); //Count1 开启
EPWM_Conter2_START(); //Count2 开启
}
```

3.2 两路 PWM 互补输出支持死区可调

软件配置：

```
/*************************************************************/
//EPWM Init
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
```

```
/**********************BLOCK VIEW*************************************/
/* - PX --- -- PWM_X */
/* PWM Engine -- ---PWM output Control --- */
/* -PY --- --PWM_Y */
/*************************************************************************/
void EPWM_CONFIG(void)
{
EPWM_RESET_VALUE(); //EPWM 所有寄存器复位赋值
EPWM_software_reset(); //EPWM 软件复位
EPWM_IO_Init(PWM_X0,0); //PWM_X0 初始化
EPWM_IO_Init(PWM_Y0,0); //PWM_Y0 初始化
EPWM_CONTER_Configure(EPWM_ContMode_decrease,EPWM_Conter_three,EMP_Overflow_Mode_Cont
inue,1,9);
//递减计数, EPMW_CLK=PCLK/(2^DIVN)/(DINM+1)=20M/2/(9+1)=1M=1US, 单次触发
EPWM_PX_PY_Configure(EPWM_P0X,EPWM_StartStopEvent_OutHigh,EPWM_PendEvent_OutHigh,EPWM
_CentralEvent_NoChange,EPWM_EqCMPAEvent_OutLow,EPWM_EqCMPBEvent_NoChange,1000,500,0);
//P0X:CNTR=1000,CMPAR=500,CMPBR=0
EPWM_OUTPUT_Configure(EPWM_PWM_X0OrPWM_Y0,EPWM_OUTSE_OutputComplementary,EPWM_X
_POLARITY_NoChange,EPWM_Y_POLARITY_Negate,EPWM_SRCSEL_PX,0x10,0x10);
//PX 为输入源互补输出,PWM_X 输出保持丌变,PWM_Y 输出反向,
//RED=EPMW_CLK*16=16us,FED=EPMW_CLK*16=16us
EPWM_Conter0_START(); //Count0 开启
}
```

3.3 六路 PWM 互补中心对齐输出支持死区可调、外部 EP0 输入下降沿触发硬锁止、PEND 时间触发 ADC

- IO 配置 PA0.9->EPWM0_X、PA0.10->EPWM0_Y
  PA0.7->EPWM1_X、PA0.8->EPWM1_Y
  PA0.5->EPWM2_X、PA0.6->EPWM2_Y
- EPWMclk= PCLK/(2^DIVN)/(DINM+1)，DIVN=0，DINM=0，EPWMclk=0.025us。
- EPWM0 周期=1000* EPWMclk=25us， 占空比=100* EPWMclk=2.5us。
- EPWM1 周期=1000* EPWMclk=25us， 占空比=100* EPWMclk=5us。
- EPWM2 周期=1000* EPWMclk=25us， 占空比=100* EPWMclk=7.5us。
- PA0.2 作为紧急制动，触发事件为下降沿触发。
- 紧急制动后所有 PWM 口配置为高阻态。
- 死区配置参数 RED=1us，FED=1uS。
- 配置 PWM PEND 事件（周期结束）事件触发 ADC 开启。

软件配置：

```
/*************************************************************/
//EPWM Init
//EntryParameter:NONE
```

```
//ReturnValue:NONE
/*************************************************************/
/**********************BLOCK VIEW*************************************/
/* - PX --- -- PWM_X */
/* PWM Engine -- ---PWM output Control --- */
/* -PY --- --PWM_Y */
/*****************************************************************************/
void EPWM_CONFIG(void)
{
EPWM_RESET_VALUE(); //EPWM 所有寄存器复位赋值
EPWM_software_reset();  //EPWM 软件复位

    EPWM_IO_Init(PWM_X0,0); //PWM_X0 初始化
    EPWM_IO_Init(PWM_Y0,0); //PWM_Y0 初始化
    EPWM_IO_Init(PWM_X1,0); //PWM_X1 初始化
    EPWM_IO_Init(PWM_Y1,0); //PWM_Y1 初始化
    EPWM_IO_Init(PWM_X2,0); //PWM_X2 初始化
    EPWM_IO_Init(PWM_Y2,0); //PWM_Y2 初始化
    GPIO_PullHigh_Init(GPIOA0,2);
    EPWM_IO_Init(PWM_EP0,0); //PWM_EP0 初始化 下降沿触发

    EPWM_CONTER_Configure(EPWM_ContMode_increaseTOdecrease,EPWM_Conter_one,EMP_Overflow_Mode_C
    ontinue,0,0);
    //递增计数, EPMW_CLK=PCLK/(2^DIVN)/(DINM+1)=40M/1/(0+1)=0.025, 连续触发
    EPWM_PX_PY_Configure(EPWM_P0X,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_Centr
    alEvent_NoChange,EPWM_EqCMPAEvent_Negate,EPWM_EqCMPBEvent_NoChange,1000,100,0);
    //P0X CNTR=1000,CMPAR=500,CMPBR=0
    EPWM->WGCR0&=0XFFF003FF;
    EPWM->WGCR0|=(0X02<<18)|(0X02<<16)|(0X02<<14)|(0X02<<12)|(0X02<<10);
    //P0Y 所有事件设置为高
    EPWM_PX_PY_Configure(EPWM_P1X,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_Centr
    alEvent_NoChange,EPWM_EqCMPAEvent_Negate,EPWM_EqCMPBEvent_NoChange,1000,200,0);
    //P1X CNTR=100,CMPAR=500,CMPBR=0
```

```
EPWM->WGCR1&=0XFFF003FF;
EPWM->WGCR1|=(0X02<<18)|(0X02<<16)|(0X02<<14)|(0X02<<12)|(0X02<<10);
//P1Y 所有事件设置为高
EPWM_PX_PY_Configure(EPWM_P2X,EPWM_StartStopEvent_OutLow,EPWM_PendEvent_OutLow,EPWM_Centr
alEvent_NoChange,EPWM_EqCMPAEvent_Negate,EPWM_EqCMPBEvent_NoChange,1000,300,0);
//P2X CNTR=100,CMPAR=500,CMPBR=0
EPWM->WGCR2&=0XFFF003FF;
EPWM->WGCR2|=(0X02<<18)|(0X02<<16)|(0X02<<14)|(0X02<<12)|(0X02<<10);
//P2Y 所有事件设置为高
EPWM_OUTPUT_Configure(EPWM_PWM_X0OrPWM_Y0,EPWM_OUTSE_OutputComplementary,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_Negate,EPWM_SRCSEL_PX,40,40);          //PX 作为输入源,互补输
出,EPMW_Y 输出反向,EPWM_X 输出保持,RED=EPMW_CLK*40=1us,FED=EPMW_CLK*40=1us
EPWM_OUTPUT_Configure(EPWM_PWM_X1OrPWM_Y1,EPWM_OUTSE_OutputComplementary,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_Negate,EPWM_SRCSEL_PX,40,40);          //PX 作为输入源,互补输
出,EPMW_Y 输出反向,EPWM_X 输出保持,RED=EPMW_CLK*40=1us,FED=EPMW_CLK*40=1us
EPWM_OUTPUT_Configure(EPWM_PWM_X2OrPWM_Y2,EPWM_OUTSE_OutputComplementary,EPWM_X_POL
ARITY_NoChange,EPWM_Y_POLARITY_Negate,EPWM_SRCSEL_PX,40,40);          //PX 作为输入源,互补输
出,EPMW_Y 输出反向,EPWM_X 输出保持,RED=EPMW_CLK*40=1us,FED=EPMW_CLK*40=1us
EPWM_LKCR_TRG_Configure(EXI0LKM,0x07); //选择 EP0 为外部触发源，硬锁止
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP0XS,EPWM_LK_output_OP);
//PWM_X0 产生硬锁止时，输出高阻
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP0YS,EPWM_LK_output_OP);
//PWM_Y0 产生硬锁止时，输出高阻
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP1XS,EPWM_LK_output_OP);
//PWM_X1 产生硬锁止时，输出高阻
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP1YS,EPWM_LK_output_OP);
//PWM_Y1 产生硬锁止时，输出高阻
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP2XS,EPWM_LK_output_OP);
//PWM_X2 产生硬锁止时，输出高阻
EPWM_SoftHardWare_OUTPUT_Configure(EPWM_LK_output_HLP2YS,EPWM_LK_output_OP);
```

```
    //PWM_Y2 产生硬锁止时，输出高阻
    EPWM_EXTRG_Configure(EPWM0_EXTRG_PEND,EPWM_EXTRG_Mode_ADC);//PWM0 PEND 事件触发 ADC
    EPWM_EXTRG_Configure(EPWM1_EXTRG_PEND,EPWM_EXTRG_Mode_ADC);//PWM1 PEND 事件触发 ADC
    EPWM_EXTRG_Configure(EPWM2_EXTRG_PEND,EPWM_EXTRG_Mode_ADC);//PWM2 PEND 事件触发 ADC

    EPWM_AllConter_START();                    //Count0~Count3 同时开启
}
```

应用注意事项：

- 若需要在程序中强行关闭 PWM 做高阻态配置如下：

```
GPIOA0->CONLR=GPIOA0->CONLR & 0X000FFFFF;//PWM 对应 PIN 脚设置为高阻
GPIOA0->CONHR=GPIOA0->CONHR & 0XFFFFF000;
EPWM_AllConter_stop();  //关闭 EPWM
```

- 若重新开启 PWM，需重新初始化对应 PIN 作为 PWM 功能。

```
EPWM_IO_Init(PWM_X0,0);
EPWM_IO_Init(PWM_Y0,0)
EPWM_IO_Init(PWM_X1,0);
EPWM_IO_Init(PWM_Y1,0);
EPWM_IO_Init(PWM_X2,0);
EPWM_IO_Init(PWM_Y2,0);
EPWM_AllConter_START();
```

- 周期占空比配置函数。

```
EPWM_Set_CNTRX_CMPARX_CMPBRX(EPWM_CNTR0,1000,500,0);
//周期= 1000Tclk 占空比=500Tclk
EPWM_Set_CNTRX_CMPARX_CMPBRX(EPWM_CNTR1,1000,300,0);
EPWM_Set_CNTRX_CMPARX_CMPBRX(EPWM_CNTR2,1000,100,0);
```

注意周期一定不能大于占空比！

- 硬锁止清除配置，清除完成后才能重新开启 PWM，重新开启 PWM 需要调用 PWM 开启函数。

```
EPWM_Hardware_Clr();      //PWM 硬锁止清除函数
```

- 如果需要更新周期值或占空比值时，需要 3 路比较值同时更新且放在 start 事件中，否则

会出现不对称问题 。

4. 程序下载和运行

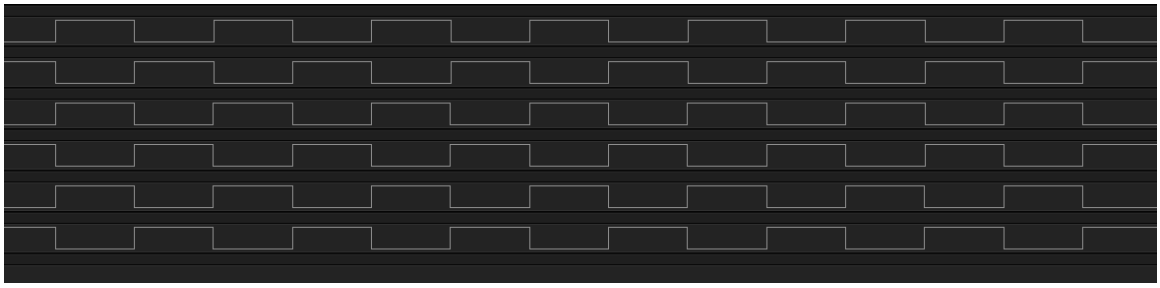
1．将目标板与仿真器连接，分别为 VDD SCLK SWIO GND

2. 将需要采集输出的口连接到波形采集器上

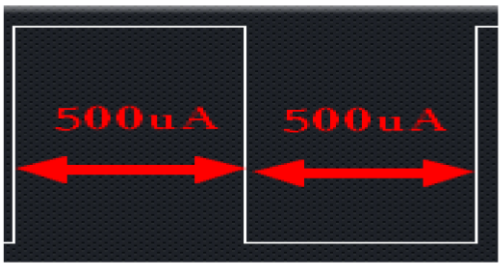
3. 程序编译后仿真运行

4. 从波形采集器上可观察到如下波形

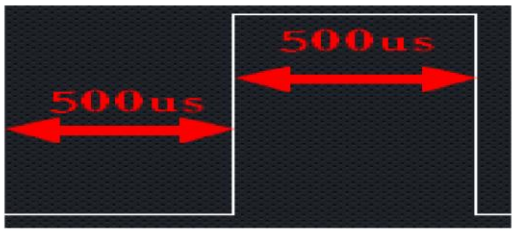
a. 六路 PWM 独立输出的波形



PA0.9(PWM_X0) /PA0.7(PWM_X1)/PA0.5(PWM_X2)

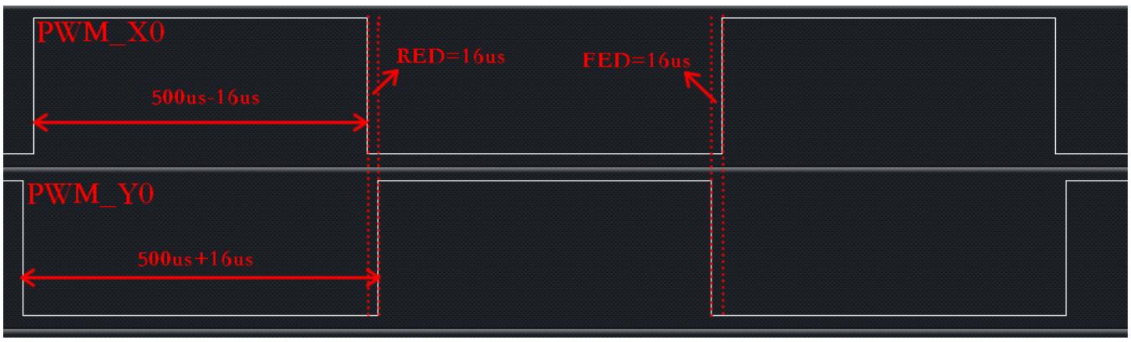


PA0.10(PWM_Y0)/PA0.8(PWM_Y1)/PA0.6(PWM_Y2)

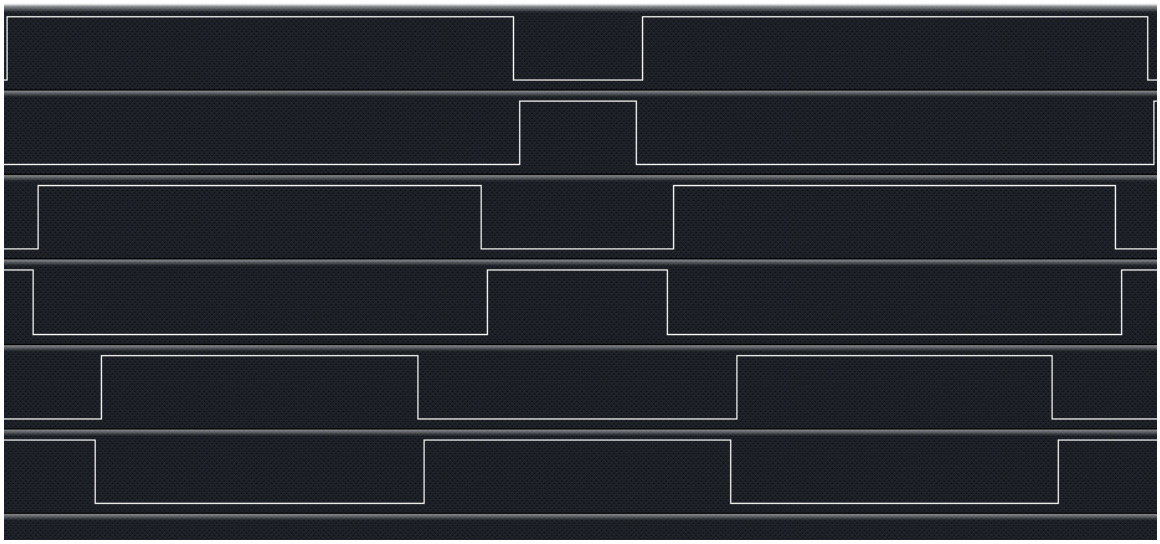


b. 两路 PWM 互补输出支持死区可调输出的波形

PWM_X0 和 PWM_Y0 波形



C．6 路互补输出且中心对齐



5 改版历史

版本	修改日期	修改概要
V1.0	2019-11-08	初版