32 位微控制器

HC32M120 系列的通用定时器 TIMER4

适用对象

系列	产品型号
HC32M120	HC32M120J6TB HC32M120F6TB

目 录

1 摘要

本篇应用笔记主要介绍 HC32M120 系列通用控制定时器 4（Timer4）模块，并简要说明如何使用 Timer4 输出 PWM（Pulse-Width Modulation）波形。

2 Timer4 简介

通用控制定时器 4（Timer4）是一个用于三相电机控制的定时器模块，提供各种不同应用的三相电机控制方案。该定时器支持三角波和锯齿波两种波形模式，可生成 PWM 波形；支持缓存功能；支持 EMB 控制。

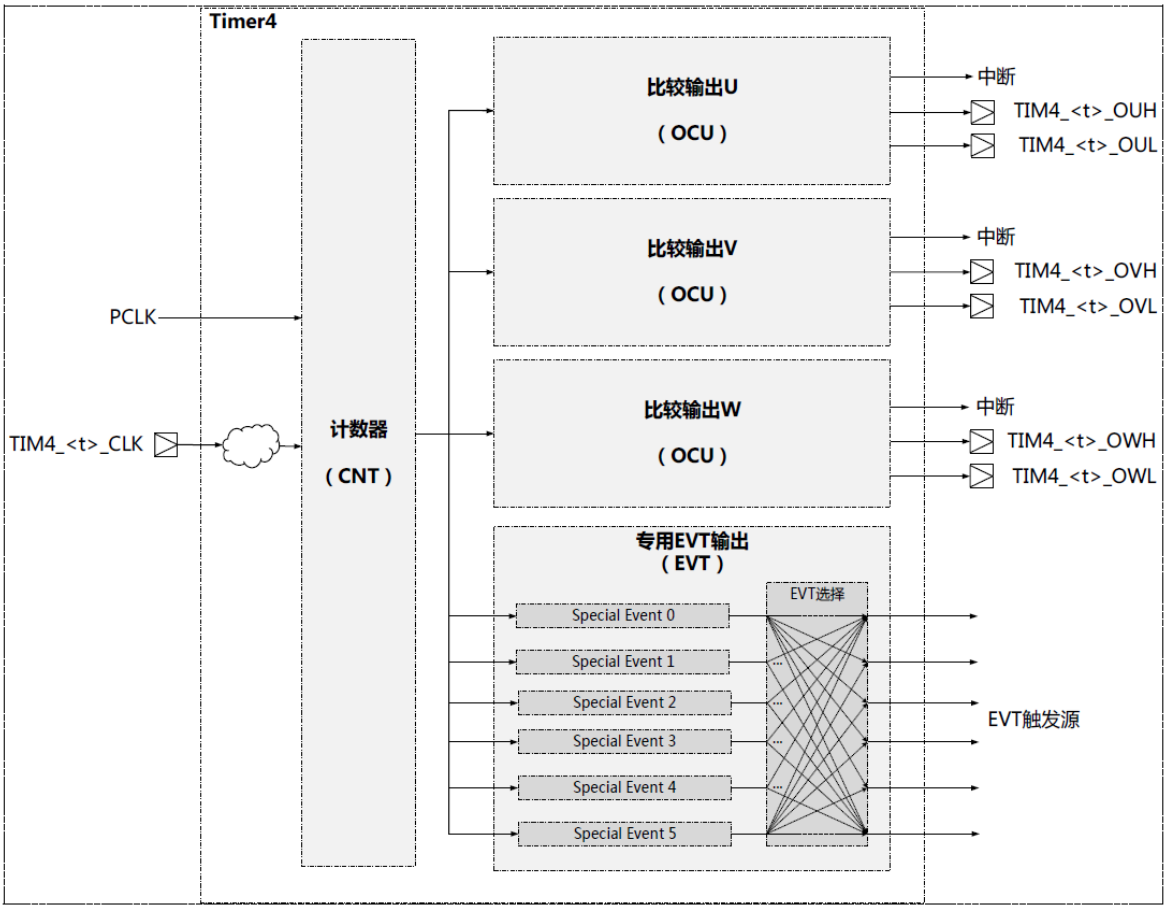
本系列产品搭载 1 个单元 Timer4。

2.1 主要特性

- 波形模式：锯齿波、三角波
- 递加、递减计数方向
- 缓存功能
- 通用 PWM 输出
- 专用事件输出
- EMB 控制

2.2 基本框图

Timer4 的基本框图如下图所示，图中”<t>”为单元编号，即”<t>”为 1；本文后续提到“<t>”时均指单元编号。



框图中的输入输出端口如下表：

端口名	方向	功能
TIM4_<t>_CLK	输入	计数时钟输入端口
TIM4_<t>_OUH	输出	PWM 输出端口
TIM4_<t>_OUL		
TIM4_<t>_OVH		
TIM4_<t>_OVL		
TIM4_<t>_OWH		
TIM4_<t>_OWL		

3 HC32M120 系列的 Timer4

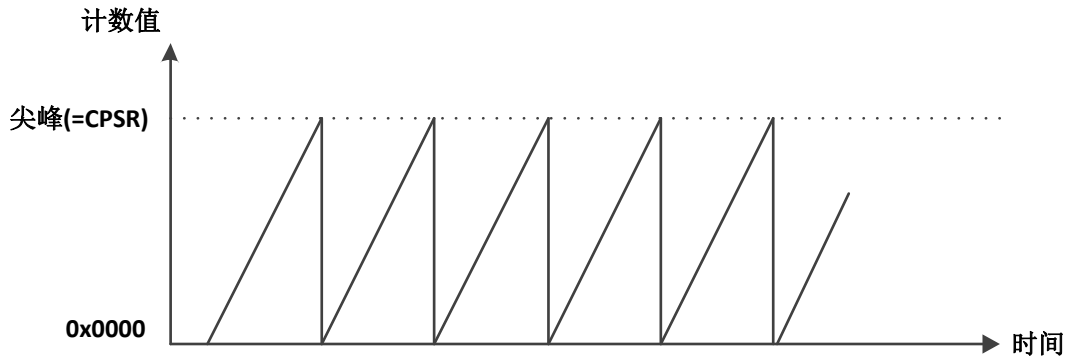
3.1 计数功能

计数器（CNT）时钟预分频器对外设时钟（PCLK1）信号进行分频，产生 16 位递增/递减计数器的操作时钟。当计数器（CNT）的计数值为 0x0000 时，过零检测标志(CCSR.IRQZF)位置“1”；当计数器（CNT）计数到峰值 (=CPSR)时，峰值检测标志(CCSR.ICLR)位置“1”。可设置是否将该中断信号输出到 CPU。也可设置中断屏蔽计数器将希望输出的中断(CCSR.IRQZF)和(CCSR.ICLR)标志数减少。

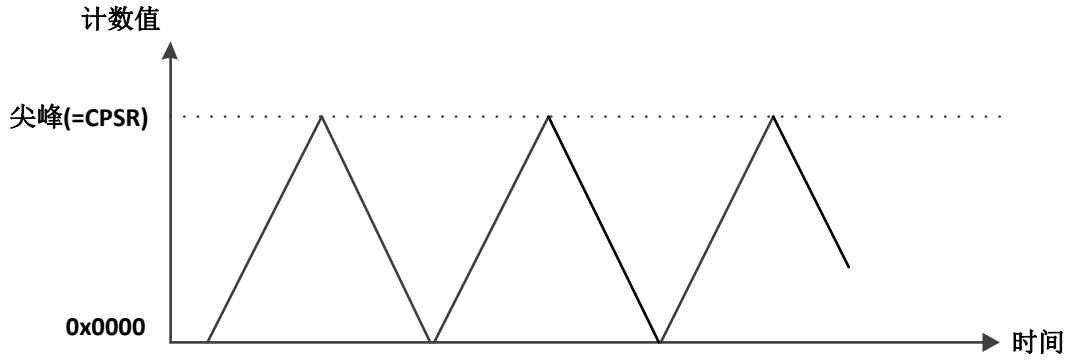
计数器（CNT）可设置两种工作模式：锯齿波模式和三角波模式。两种模式计数周期如下表所示：

工作模式	计数器周期
锯齿波计数模式	(CPSR+1) × 计数时钟周期
三角波计数模式	CPSR × 2 × 计数时钟周期

计数器（CNT）设置为锯齿波模式，操作示例如下：



计数器（CNT）设置为三角波模式，操作示例如下：

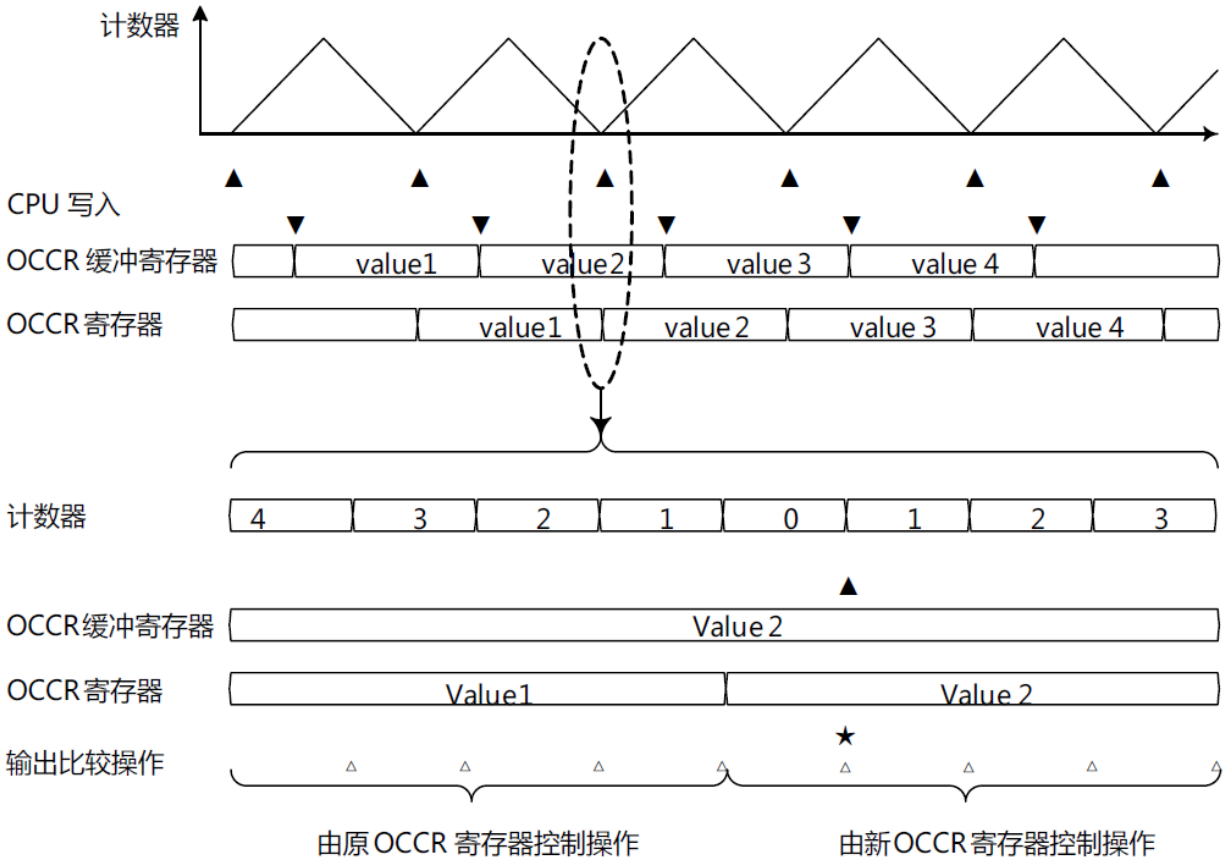


3.2 比较输出

比较输出（Output Compare, OCO）是根据 CNT 的计数器值产生和输出 PWM 信号的功能模块。OCO 输出的 PWM 信号名。这些信号经过 PWM 输出至 TIM4_<t>_OXH/ TIM4_<t>_OXH(X=U,V,W)。

- 通用比较基准寄存器（OCCR）指定 PWM 信号的变更时序，作为 CNT 计数器值的比较值。具有缓冲寄存器，使数据能通过异步 CNT 计数操作将数据异步写入 OCCR 寄存器。
- 通用模式控制寄存器（OCMR）用于指定 PWM 信号的变更条件。具有缓冲寄存器，使数据能通过 CNT 计数操作将数据异步写入 OCMR 寄存器。

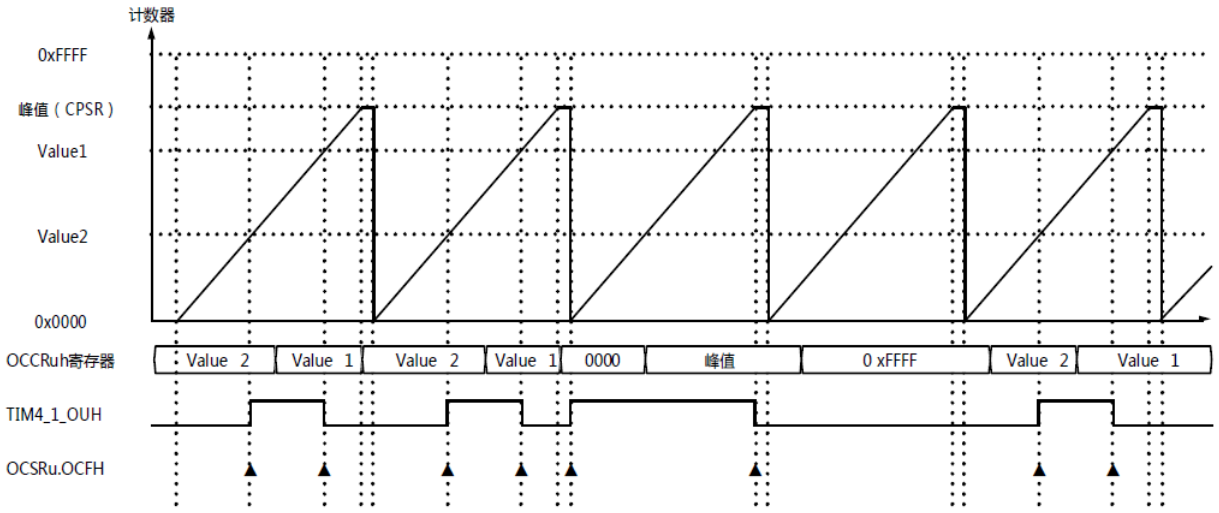
下图为 OCCR 缓存功能使能，OCCR 寄存器更新时序示例：



3.3 通用 PWM 输出

PWM 波形产生可以选择以下模式：直通模式、死区定时器模式和死区定时器滤波模式。

下图为 Timer4 的锯齿波的独立 PWM 输出示例，通用比较基准寄存器（OCCRxh、OCCRxl）的值发生比较匹配所产生的内部输出信号（in_opxh、in_opxl）直接输出至对应的端口（TIM4_<t>_OXH、TIM4_<t>_OXL）上。



OCMRuh.OCFUCH=1 OCMRuh.OPNPKH=00 OCMRuh.OPUCH=OCMRuh.OPPKH=11

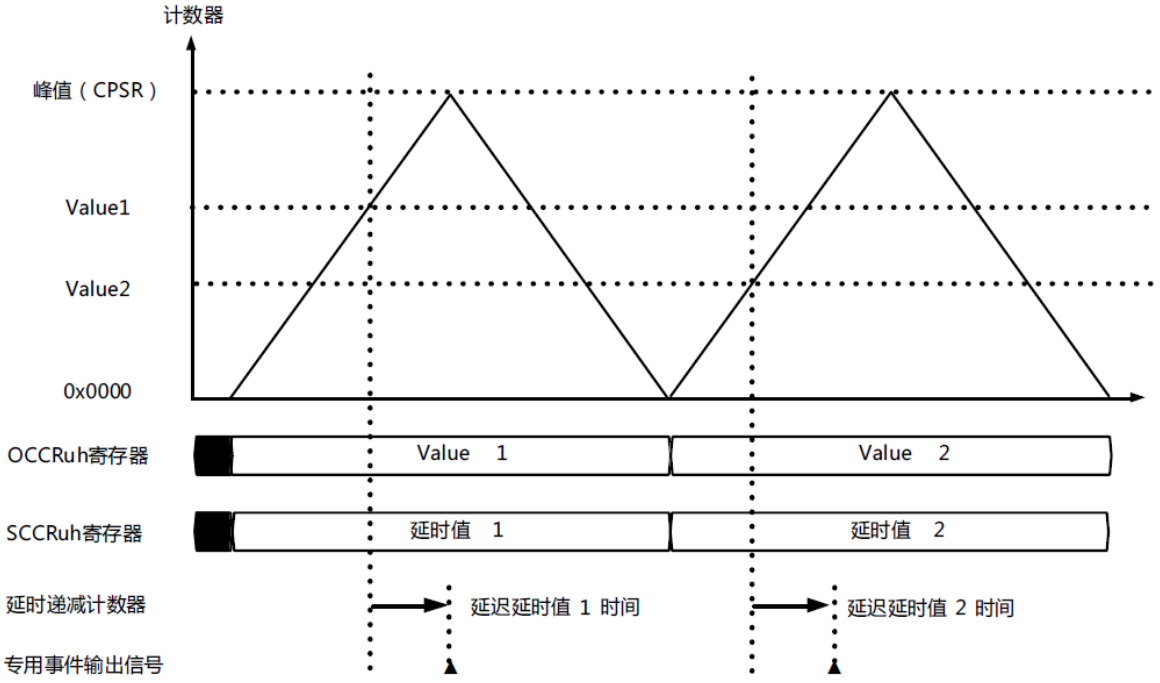
3.4 专用事件输出

专用比较基准寄存器（SCCR）和专用模式控制寄存器（SCMR）

专用事件输出功能模块根据 CNT 计数器值产生 AD 转换启动信号。

- 在比较启动模式下，SCCR 寄存器用于指定事件输出的启动时序，用作 FRT 计数器值的比较值。
- 在延时启动模式下，SCCR 寄存器用于指定 OCO 检测到匹配后事件输出之前的延迟时间。
- SCMR 具有缓冲寄存器，因此可通过 FRT 计数操作将数据异步写入 SCMR 寄存器。

下图为延时启动模式下的专用事件输出信号的请求输出示例。



3.5 EMB 控制

Timer4 单元有一个输出无效事件接口，链接 EMB 模块输出的 EMB 事件。该接口上选通的一场状况事件可从 EMB 侧设定。监测到从 EMB 过来的 EMB 异常事件，则端口的输出状态可变为预先设定好的状态。该预设端口状态可以为输出高阻态、输出低电平或输出高电平。

3.6 寄存器说明

通用定时器 Timer4 模块的寄存器如下表所示，若需了解具体细节，请参考用户手册：

寄存器简称	寄存器功能	备注
TMR4_CNTER	计数值寄存器	
TMR4_CPSR	周期基准寄存器	
TMR4_CCSR	控制状态寄存器	
TMR4_CVPR	有效周期寄存器	
TMR4_OCCRxh	通用比较基准寄存器 xh	x=u/v/w
TMR4_OCCRxl	通用比较基准寄存器 xl	x=u/v/w
TMR4_OCSRx	通用控制状态寄存器 x	x=u/v/w
TMR4_OCERx	通用扩展控制寄存器 x	x=u/v/w
TMR4_OCMRxh	通用模式控制寄存器 xh	x=u/v/w
TMR4_OCMRxl	通用模式控制寄存器 xl	x=u/v/w
TMR4_SCCRxh	专用比较基准寄存器 xh	x=u/v/w
TMR4_SCCRxl	专用控制状态寄存器 xl	x=u/v/w
TMR4_SCSRxh	专用控制状态寄存器 xh	x=u/v/w
TMR4_SCSRxl	专用控制状态寄存器 xl	x=u/v/w
TMR4_SCMRxh	专用模式控制寄存器 xh	x=u/v/w
TMR4_SCMRxl	专用模式控制寄存器 xl	x=u/v/w
TMR4_POCRx	PWM 基本控制寄存器 x	x=u/v/w
TMR4_PFSRx	PWM 滤波控制寄存器 x	x=u/v/w
TMR4_PFSRAx	PWM 死区控制寄存器 Ax	x=u/v/w
TMR4_PFSRBx	PWM 死区控制寄存器 Bx	x=u/v/w
TMR4_RCSR	重载控制状态寄存器	
TMR4_ECSR	EMB 控制状态寄存器	

4 样例代码

4.1 代码介绍

用户可以根据上述的介绍编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到 HC32M120 系列 MCU 的设备驱动库（Device Driver Library，DDL）来体验 Timer4 的正交编码计数功能。

以下部分主要基于 DDL 的 Timer4 模块的输入捕获样例 timer4_pwm_through_mode 代码，简要介绍 Timer4 PWM 功能使用方法：

1) 配置系统时钟

```
/* Configure system clock. */
SystemClockConfig();
```

2) 配置 Timer4 计数器功能

```
/* Enable peripheral clock */
CLK_FcgPeriphClockCmd(FUNCTION_CLK_GATE, Enable);

/* Initialize TIMER4 Counter */

/* Configure RGB LED. */
TIMER4_CNT_StructInit(&stcTimer4CntInit);
stcTimer4CntInit.u16ClkDiv = TIMER4_CNT_CLK_DIV512;
/* Period_Value(500ms) = SystemClock(SystemCoreClock) / TIMER4_CNT_Clock_Division(512) /
Frequency(2) */
stcTimer4CntInit.u16CycleVal = TIMER4_CNT_CYCLE_VAL;
TIMER4_CNT_Init(&stcTimer4CntInit);
```

3) 配置 Timer4 输出比较功能

```
/* Initialize TIMER4 OCO high&&low channel */
TIMER4_OCO_StructInit(&stcTimer4OcoInit);
stcTimer4OcoInit.enOcoCmd = Enable;
stcTimer4OcoInit.enOcoIntCmd = Enable;
stcTimer4OcoInit.u16OcoInvalidOp = TIMER4_OCO_INVAILD_OP_LOW;
stcTimer4OcoInit.u16OccrVal = TIMER4_CNT_CYCLE_VAL/2;
TIMER4_OCO_Init(TIMER4_OCO_HIGH_CH, &stcTimer4OcoInit);
TIMER4_OCO_Init(u32OcoLowCh, &stcTimer4OcoInit);
```

4) 配置 Timer4 输出比较高低通道模式：

```
  if (!(TIMER4_OCO_HIGH_CH % 2ul))
  {
    /* OCMR[15:0] = 0x0FFF = b 0000 1111 1111 1111 */
    stcHighChCmpMode.OCMRx_f.OCFDCH = TIMER4_OCO_OCF_SET;   /* bit[0] 1 */
    stcHighChCmpMode.OCMRx_f.OCFPKH = TIMER4_OCO_OCF_SET;   /* bit[1] 1 */
    stcHighChCmpMode.OCMRx_f.OCFUCH = TIMER4_OCO_OCF_SET;   /* bit[2] 1 */
    stcHighChCmpMode.OCMRx_f.OCFZRH = TIMER4_OCO_OCF_SET;   /* bit[3] 1 */

    stcHighChCmpMode.OCMRx_f.OPDCH = TIMER4_OCO_OP_INVERT;  /* Bit[5:4]    11 */
    stcHighChCmpMode.OCMRx_f.OPPKH = TIMER4_OCO_OP_INVERT;  /* Bit[7:6]    11 */
    stcHighChCmpMode.OCMRx_f.OPUCH = TIMER4_OCO_OP_INVERT;  /* Bit[9:8]    11 */
    stcHighChCmpMode.OCMRx_f.OPZRH = TIMER4_OCO_OP_INVERT;  /* Bit[11:10]  11 */
    stcHighChCmpMode.OCMRx_f.OPNPKH = TIMER4_OCO_OP_HOLD;   /* Bit[13:12]  00 */
    stcHighChCmpMode.OCMRx_f.OPNZRH = TIMER4_OCO_OP_HOLD;   /* Bit[15:14]  00 */

    stcHighChCmpMode.enExtendMatchCondCmd = Disable;

    TIMER4_OCO_SetHighChCompareMode(TIMER4_OCO_HIGH_CH, &stcHighChCmpMode);  /* Set
OCO high channel compare mode */
  }
  else
  {
  }

  if (u32OcoLowCh % 2ul)
  {
    /* OCMR[31:0] 0x0FF0 0FFF = b 0000 1111 1111 0000   0000 1111 1111 1111 */
    stcLowChCmpMode.OCMRx_f.OCFDCL = TIMER4_OCO_OCF_SET;    /* bit[0] 1 */
    stcLowChCmpMode.OCMRx_f.OCFPKL = TIMER4_OCO_OCF_SET;    /* bit[1] 1 */
    stcLowChCmpMode.OCMRx_f.OCFUCL = TIMER4_OCO_OCF_SET;    /* bit[2] 1 */
    stcLowChCmpMode.OCMRx_f.OCFZRL = TIMER4_OCO_OCF_SET;    /* bit[3] 1 */

    stcLowChCmpMode.OCMRx_f.OPDCL = TIMER4_OCO_OP_INVERT;   /* bit[5:4]    11 */
    stcLowChCmpMode.OCMRx_f.OPPKL = TIMER4_OCO_OP_INVERT;   /* bit[7:6]    11 */
    stcLowChCmpMode.OCMRx_f.OPUCL = TIMER4_OCO_OP_INVERT;   /* bit[9:8]    11 */
    stcLowChCmpMode.OCMRx_f.OPZRL = TIMER4_OCO_OP_INVERT;   /* bit[11:10]  11 */
    stcLowChCmpMode.OCMRx_f.OPNPKL = TIMER4_OCO_OP_HOLD;    /* bit[13:12]  00 */
    stcLowChCmpMode.OCMRx_f.OPNZRL = TIMER4_OCO_OP_HOLD;    /* bit[15:14]  00 */
    stcLowChCmpMode.OCMRx_f.EOPNDCL = TIMER4_OCO_OP_HOLD;   /* bit[17:16]  00 */
    stcLowChCmpMode.OCMRx_f.EOPNUCL = TIMER4_OCO_OP_HOLD;   /* bit[19:18]  00 */
    stcLowChCmpMode.OCMRx_f.EOPDCL = TIMER4_OCO_OP_INVERT;  /* bit[21:20]  11 */
    stcLowChCmpMode.OCMRx_f.EOPPKL = TIMER4_OCO_OP_INVERT;  /* bit[23:22]  11 */
    stcLowChCmpMode.OCMRx_f.EOPUCL = TIMER4_OCO_OP_INVERT;  /* bit[25:24]  11 */
    stcLowChCmpMode.OCMRx_f.EOPZRL = TIMER4_OCO_OP_INVERT;  /* bit[27:26]  11 */
    stcLowChCmpMode.OCMRx_f.EOPNPKL = TIMER4_OCO_OP_HOLD;   /* bit[29:28]  00 */
    stcLowChCmpMode.OCMRx_f.EOPNZRL = TIMER4_OCO_OP_HOLD;   /* bit[31:30]  00 */

    stcLowChCmpMode.enExtendMatchCondCmd = Disable;

    TIMER4_OCO_SetLowChCompareMode(u32OcoLowCh, &stcLowChCmpMode);  /* Set OCO low
channel compare mode */
  }
  else
  {
  }
```

5) 配置 Timer4 PWM 功能：

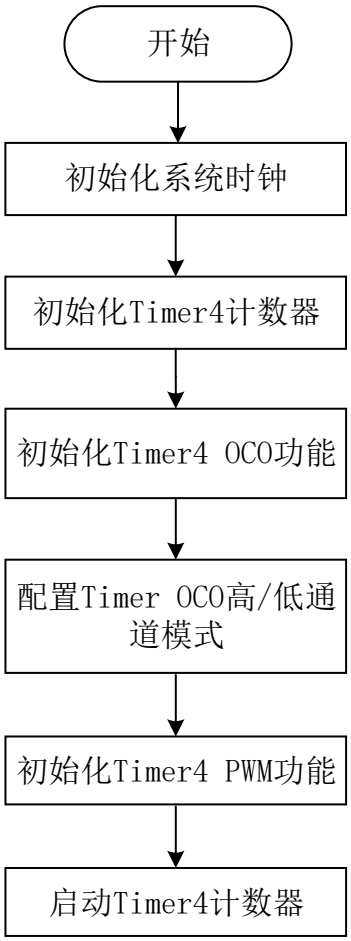
```
/* Initialize PWM I/O */
GPIO_SetFunc(TIM4_1_OXH_PORT, TIM4_1_OXH_PIN, TIM4_1_OXH_GPIO_FUNC);
GPIO_SetFunc(TIM4_1_OXL_PORT, TIM4_1_OXL_PIN, TIM4_1_OXL_GPIO_FUNC);

/* Timer4 PWM: Get pwm couple channel */
u32PwmCh = TIMER4_PWM_CH(TIMER4_OCO_HIGH_CH);

/* Initialize Timer4 PWM */
TIMER4_PWM_StructInit(&stcTimer4PwmInit);
stcTimer4PwmInit.enRtIntMaskCmd = Enable;
stcTimer4PwmInit.u16PwmOutputPolarity = TIMER4_PWM_OP_OXH_HOLD_OXL_INVERT;
TIMER4_PWM_Init(u32PwmCh, &stcTimer4PwmInit);
```

6) 启动 Timer 计数器

```
/* Start TIMER4 counter. */
TIMER4_CNT_Start();
```

4.2 工作流程

样例代码中 Timer4 操作流程如下图所示：



4.3 代码运行

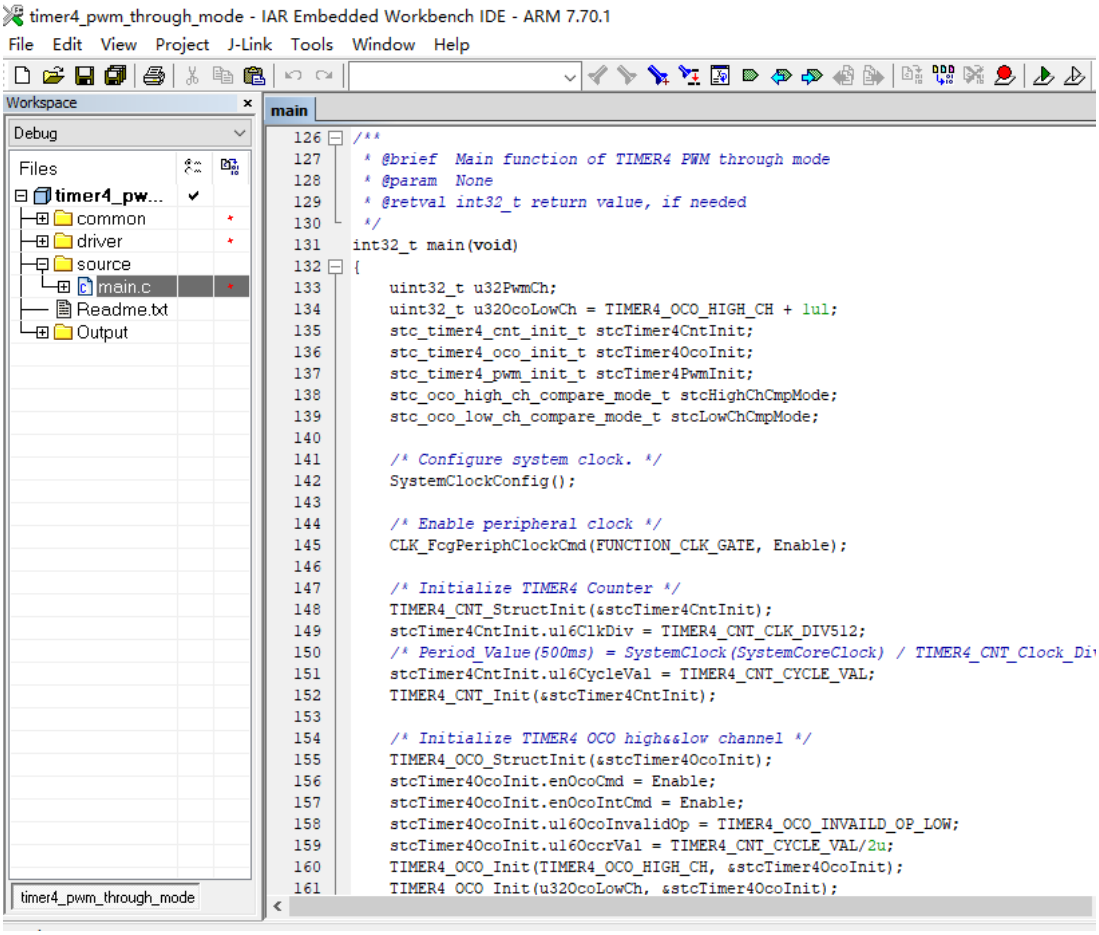
用户可以通过华大半导体的网站下载到 DDL 的样例代码（timer4_pwm_through_mode），并配合评估用板（‘STK_HC32M120_LQFP48_050_V11’）运行相关代码学习使用 Timer4 模块。

以下部分主要介绍如何在‘STK_HC32M120_LQFP48_050_V11’评估板上，通过 IAR EWARM 编译、运行 timer4_pwm_through_mode 样例代码并观察结果：

- 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 获取‘timer4_pwm_through_mode’评估板。
- 从华大半导体网站下载 HC32M120 DDL 代码。
- 下载并运行 timer4\timer4_pwm_through_mode\中的项目文件：

  1) 将测试板 J3-P61(TIM4_1_OUH)、J3-P60(TIM4_1_OUL)引脚与示波器相连；

  2) 打开 timer4_pwm_through_mode\工程，并打开‘main.c’如下视图：



  3) 点击  重新编译整个项目；

4) 点击  将代码下载到评估板上，全速运行；

5) 观察示波器，P61、P60 通道波形极性相反，频率相同 1Hz。

5 总结

以上章节简要介绍 HC32M120 系列的 Timer4 寄存器、功能模式。演示了如何操作 Timer4 PWM 直通模式样例代码，在开发中用户可以根据自己的实际需要使用 Timer4 模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2019/12/10	Rev1.0	初版发布

如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email：mcu@hdsc.com.cn

网址：www.hdsc.com.cn

通信地址：上海市浦东新区中科路 1867 号 A 座 10 层

邮编：201203