

32 位微控制器

HC32M120 系列的通用定时器 TIMER0

适用对象

系列	产品型号
HC32M120	HC32M120J6TB
	HC32M120F6TB

目 录

1	摘要	3
2	TIMER0 简介	3
3	HC32M120 系列的 TIMER0	4
3.1	系统框图	4
3.2	功能说明	5
3.2.1	时钟源选择	5
3.2.2	硬件触发动作	5
3.2.3	中断及事件输出	5
3.2.4	功能模式	5
3.3	注意事项	7
3.4	寄存器说明	7
4	样例代码	8
4.1	代码介绍	8
4.2	代码运行	10
5	总结	11
6	版本信息 & 联系方式	12

1 摘要

本篇应用笔记主要介绍 HC32M120 系列芯片的通用定时器（TIMER0）模块，并通过展示 basetimer 样例代码简要说明如何使用 TIMER0 模块。

2 TIMER0 简介

HC32M120 系列的通用定时器（TIMER0）模块是一个可以实现同步计数、异步计数两种方式的基本定时器。定时器内含 1 个通道，可以在计数期间产生比较匹配事件。该事件可以触发中断、也可以作为事件输出来控制其它模块。本系列芯片搭载 1 个单元的 TIMER0。

TIMER0 主要特性：

- 同步计数方式异步计数方式可选
- 中断输出或事件输出
- 内部硬件触发事件输入可作为时钟源
- 可实现输入捕获功能
- 内部硬件触发事件可触发定时器启动、停止或清零动作

3 HC32M120 系列的 TIMER0

3.1 系统框图

TIMER0 的系统框图如下所示。

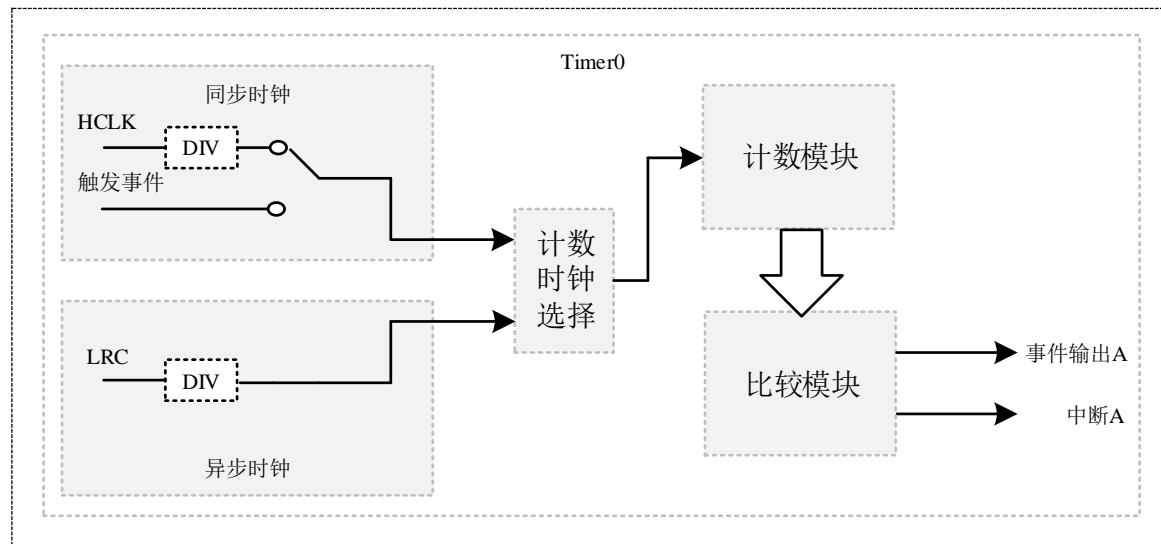


图 3-1 TIMER0 系统框图

3.2 功能说明

3.2.1 时钟源选择

时钟源分为两类：同步时钟源和异步时钟源。时钟源寄存器配置详情如下所示：

同步计数	BCONR.SYNS=0	BCONR.SYNCLK=0	HCLK 及 HCLK 的 2、4、8、16、32、64、128、256、512、1024 分频，分频系数由 BCONR.CKDIV[3:0]配置。
		BCONR.SYNCLK=1	内部硬件触发事件输入
异步计数	BCONR.SYNS=1		LRC 及 LRC 的 2、4、8、16、32、64、128、256、512、1024 分频，分频系数由 BCONR.CKDIV[3:0]配置。

3.2.2 硬件触发动作

当 TIMER0 配置为比较输出功能时，内部硬件触发源事件可以触发计数启动、停止、清零动作；当 TIMER0 工作在输入捕获功能时，TIMER0 对内部硬件触发源事件进行输入捕获。

硬件触发源通过寄存器 HTSSR 进行选择，具体说明请参考本系列芯片用户手册 INTC 章节。

如采用外部中断事件 EVT_PORT_EIRQ1 作为触发源对 TIMER0 进行事件触发时，需要的配置步骤如下：

1. 配置产生事件的外部中断 ExtiCh01
2. 使能芯片的 AOS 功能，将 FCG 相关寄存器位清零
3. 配置 TIMER0 硬件触发使能，配置寄存器 TMR0_BCONR.HSTPA 触发动作为 stop
4. 配置寄存器 TMR0_HTSSR 选择硬件触发源为 EVT_PORT_EIRQ1
5. TIMER0 中断使能，启动定时器开始计数

完成以上配置后，通过外部中断事件可以触发 TIEMR0 停止计数。

3.2.3 中断及事件输出

当 TIMER0 产生了比较匹配事件或者输入捕获事件时，如 TIMER0 的中断功能使能则会产生相应 TIMER0 中断。详细说明请参考芯片用户手册。

3.2.4 功能模式

TIMER0 可以配置为比较输出功能或者捕获输入功能。

3.2.4.1 比较输出功能

当寄存器 TMR0_BCONR.CAPMDA 设置为 0 时 TIMER0 配置为比较输出功能，通过计数器 TMR0_CNTAR 对 TIMER0 计数时钟进行累加，当计数值与基准值寄存器 TMR0_CMPAR 匹配时产生匹配中断或事件。

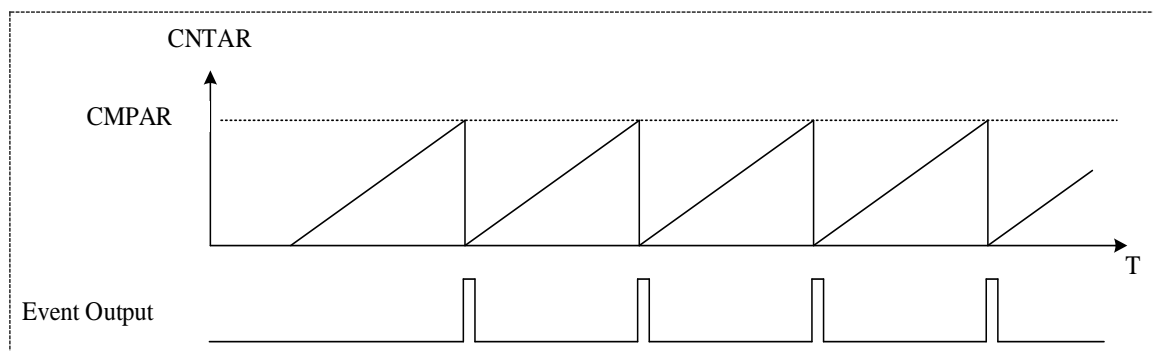


图 3-2 Timer0 比较输出功能

3.2.4.2 输入捕获功能

当寄存器 TMR0_BCONR.CAPMDA 设置为 1 并且 TMR0_BCONR.HICPA 也为 1 时 TIMER0 配置为输入捕获功能，对输入的硬件触发源进行捕获，当有内部硬件触发源事件时，产生 TIMER0 输入捕获中断，并且将 TMR0_CNTAR 当前的计数值存入 TMR0_CMPAR 寄存器中。

3.3 注意事项

使用 TIMER0 模块时，需要注意以下几点：

- 1) 在异步计数方式时，读计数值寄存器（CNTRE）和基准值寄存器（CMPAR）必须在计数停止状态下实现。
- 2) 在异步计数动作时，需先设定 BCONR.SYNSA 位选择异步计数方式，然后再启动 TIMER0。
- 3) 在异步计数的情况下，修改计数值（CNTAR）、基准值（CMPAR）、启动位（BCONR.CSTA）、状态位（STFLR.CMFA）时，TIMER0 从接收到写动作后经过 3 个异步计数时钟才将修改值写入对应的寄存器中。
- 4) 在选择异步计数的情况下，连续对计数值（CNTAR）、基准值（CMPAR）、启动位（BCONR.CSTA）、状态位（STFLR.CMFA）进行写动作时，需间隔至少 3 个异步计数时钟。

以上注意事项 2) ~ 4) 在 DDL（Device Driver Library）TIMER0 库函数中已经进行处理，用户在使用过程中也需要注意时间间隔。

3.4 寄存器说明

以下为 TIMER0 模块的寄存器列表，详细寄存器说明请查看本系列芯片用户手册相关章节。

BASE ADDR: 0x40005800

寄存器名	符号	偏移量	位宽	复位值
计数值寄存器	TMR0_CNTAR	0x0000h	32	0x00000000h
基准值寄存器	TMR0_CMPAR	0x0008h	32	0x0000FFFFh
基本控制寄存器	TMR0_BCONR	0x0010h	32	0x00000000h
触发选择寄存器	TMR0_HTSSR	(0x40000C20h)	32	0x0000007Fh
状态标志寄存器	TMR0_STFLR	0x0014h	32	0x00000000h

触发选择寄存器 TMR0_HTSSR 和 TIMER2 模块共用一个触发源选择寄存器，使用中需要注意。

4 样例代码

4.1 代码介绍

用户可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设备驱动库（Device Driver Library, DDL）的样例代码并使用其中的 **TIMER0** 的样例进行验证。

以下部分简要介绍本 AN 基于 DDL 的 **TIMER0** 模块样例 **basetimer** 代码所涉及的各项配置。

1) 系统时钟、测试 LED 端口初始化

```
/* Configure system clock. */  
SystemClockConfig();  
/* Configure RGB LED. */  
LedConfig();  
/* LRC ON */  
CLK_LRCInit(CLK_LRC_ON);
```

2) Exint1 功能使能及 AOS 功能使能

```
/* Config EXINT (INT1 SW2)function for hardware trigger */  
GPIO_StructInit(&stcGpioIni);  
stcGpioIni.u16ExInt = PIN_EXINT_ON;  
GPIO_Init(SW2_PORT, SW2_PIN, &stcGpioIni);  
EXINT_StructInit(&stcExintCfg);  
stcExintCfg.u16ExIntCh = EXINT_CH01;  
stcExintCfg.u8ExIntFE = EXINT_FILTER_ON;  
stcExintCfg.u8ExIntFClk = EXINT_FCLK_HCLK_DIV32;  
stcExintCfg.u8ExIntLvl = EXINT_TRIGGER_FALLING;  
EXINT_Init(&stcExintCfg);  
/* Enable AOS function */  
CLK_FcgPeriphClockCmd(CLK_FCG_AOS, Enable);
```

3) TIMER0 比较输出功能配置及使能

```
/* Enable timer0 peripheral clock */  
CLK_FcgPeriphClockCmd(CLK_FCG_TIM0, Enable);  
/* TIMER0 basetimer function initialize */  
TIMER0_StructInit(&stcTimer0Ini);  
stcTimer0Ini.u32ClockDivision = TIMER0_CLK_DIV256; /* Config clock division */  
stcTimer0Ini.u32ClockSource = TIMER0_CLK_SRC_LRC; /* Chose clock source */  
stcTimer0Ini.u32Tmr0Fun = TIMER0_FUNC_CMP; /* Timer0 compare mode */  
stcTimer0Ini.u32HwTrigFunc = TIMER0_BT_HWTRG_FUNC_STOP; /* Config Hardware trigger  
function */  
stcTimer0Ini.u16CmpValue = LRC_FRQ / 2u / 256u; /* Set compara register data */  
TIMER0_Init(&stcTimer0Ini);  
/* Set internal hardware capture source */  
TIMER0_SetTriggerSrc(EVT_PORT_EIRQ1);
```


4) TIMER0 NVIC 中断配置

```
/* Register IRQ handler && configure NVIC. */
stcIrqRegiCfg.enIRQn = TIMER0_IRQn;
stcIrqRegiCfg.enIntSrc = TIMER0_SOURCE;
stcIrqRegiCfg.pfnCallback = Timer0CompareIrqCallback;
INTC_IrqRegistration(&stcIrqRegiCfg);
NVIC_ClearPendingIRQ(stcIrqRegiCfg.enIRQn);
NVIC_SetPriority(stcIrqRegiCfg.enIRQn, DDL_IRQ_PRIORITY_02);
NVIC_EnableIRQ(stcIrqRegiCfg.enIRQn);
```

5) 计数功能使能及主循环

```
/* Timer0 interrupt function enable */
TIMER0_IntCmd(Enable);
/* Timer function kick start */
TIMER0_Cmd(Enable);

while(1)
{
};
```

6) 中断服务程序

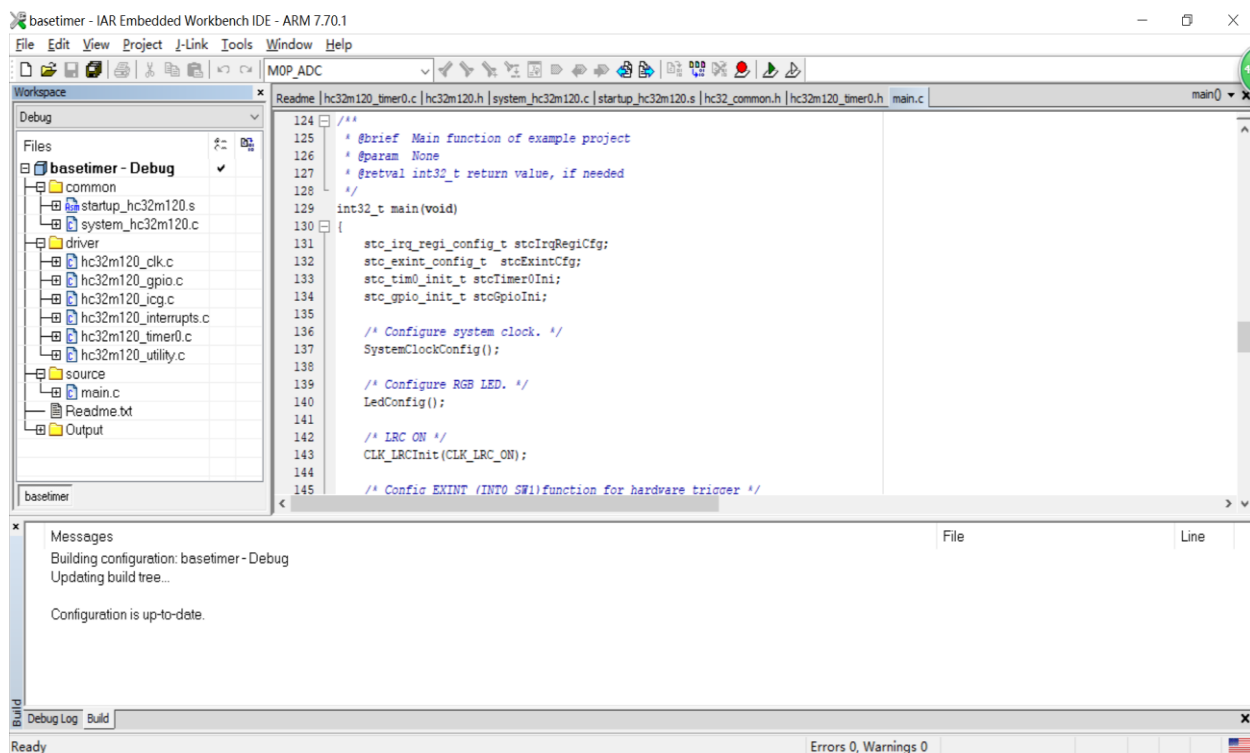
```
/**
 * @brief TIMER0 compare IRQ callback
 * @param None
 * @retval None
 */
static void Timer0CompareIrqCallback(void)
{
    LED_G_TOGGLE();
    TIMER0_ClearFlag();
}
```


4.2 代码运行


用户可以通过华大半导体的网站下载到 HC32M120 的 DDL 的样例代码（basetimer），并配合评估用板（STK-HC32M120-LQFP48-050-V11）运行相关代码学习使用 TIMER0 模块。

以下部分主要介绍如何在评估板上运行 basetimer 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）。
- 从华大半导体网站下载 HC32M120 DDL 代码。
- 下载并运行 basetimer\ 中的工程文件：
 - 1) 打开 basetimer\ 工程，并打开 ‘main.c’ 如下视图：



2) 点击  重新编译整个项目；

3) 点击  将代码下载到评估板上，全速运行；

4) 观测 LED 的亮灭变化，此时评估板上的绿灯以 1s 为周期闪烁，表示 TIMER0 的计数及中断功能正常运行；

5) 短按按键 SW2，通过内部触发事件使 TIMER0 计数停止，绿灯停止闪烁。

5 总结

以上章节简要介绍了 HC32M120 系列的 TIMER0，说明了 TIMER0 模块的寄存器及部分操作流程，并且演示了如何使用 TIMER0 样例代码，在实际开发中用户可以根据自己的需要配置和使用 TIMER0 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

