



32-bit Microcontrollers

HC 32 F 460 Series General Purpose

Timer T IMERO

Applicable objects

Series	Product Model
HC32F460	HC32F460JEUA
	HC32F460JETA
	HC32F460KEUA
	HC32F460KETA
	HC32F460PETB

Table of Contents Table of Contents

1	Abstract	3
2	TIMER0 Introduction.....	3
3	TIMER0 of HC32F460 series	4
	3.1 System Block Diagram.....	4
	3.2 Function Description.....	5
	3.2.1 Clock source selection.....	5
	3.2.2 Basic counting function.....	5
	3.2.3 Hardware triggered action.....	5
	3.2.4 Interrupt and event output.....	6
	3.3 Cautions	7
	3.4 Register Description	8
4	Sample Code.....	9
	4.1 Code Introduction	9
	4.2 Code Run.....	11
5	Summary	12
6	Version Information & Contact	13

1 Abstract

This application note introduces the general-purpose timer (TIMER0) module of the HC32F460 series chip, and shows the following

The BaseTimer sample code briefly illustrates how to use the TIMER0 module.

2 TIMER0 Introduction

The general-purpose timer (TIMER0) module of HC32F460 series is a basic timer that can implement both synchronous counting and asynchronous counting. The timer contains 2 channels and can generate a compare match event during counting. The event can trigger an interrupt, or be used as an event output to control other modules, etc. There are two independent TIMER0 units, TMR01 and TMR02, in this series.

TIMER0 Main features:

- Synchronous counting method asynchronous counting method is optional
- Interrupt output or event output
- Two channels share an internal hardware trigger source

3 TIMER0 of HC32F460 series

3.1 System Block Diagram

The system block diagram of one unit TIMER0 is shown below. Each unit consists of two channel timers CH_A and CH_B.

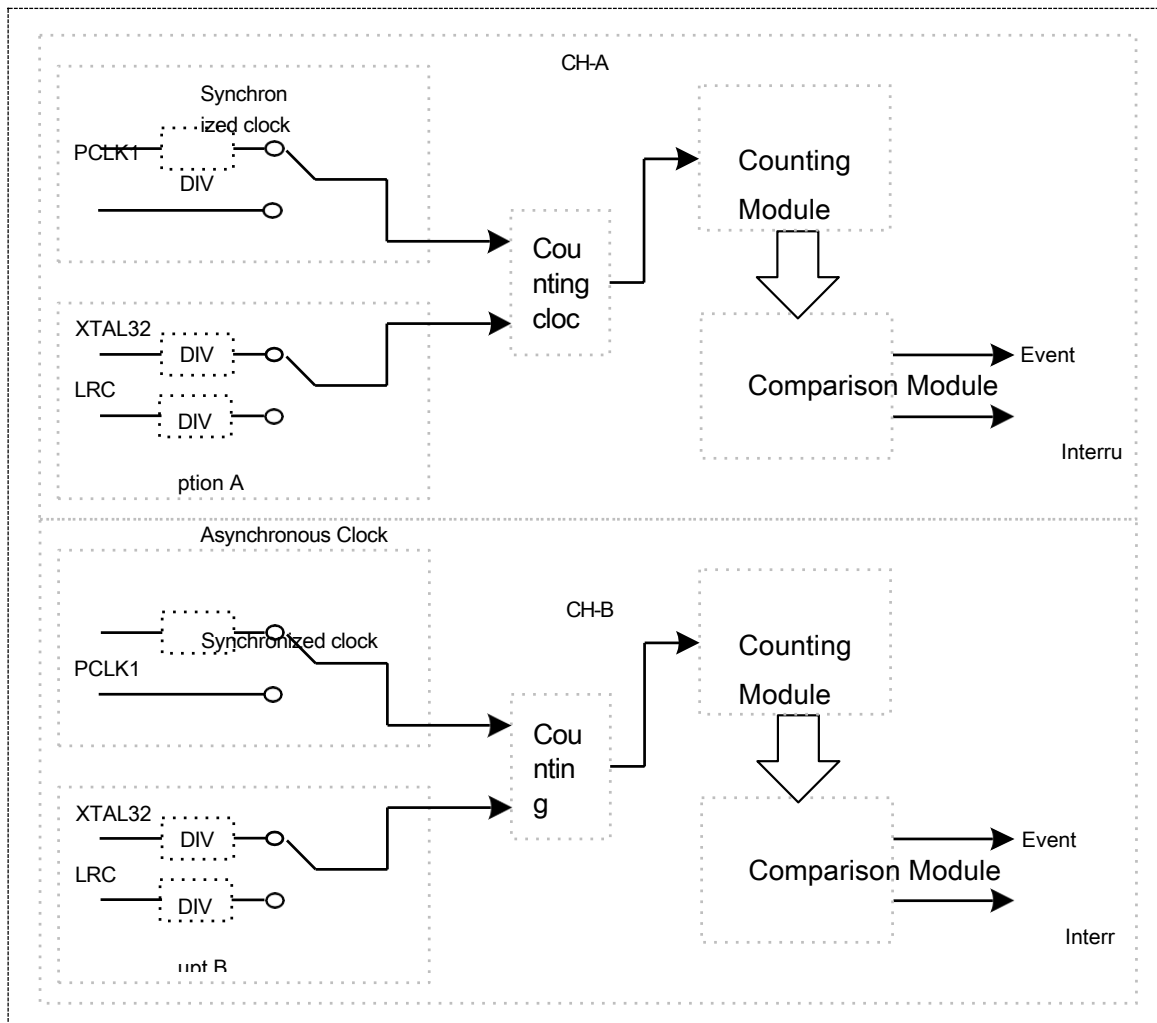


Figure 1 TIMER0 System Block Diagram

3.2 Function Description

3.2.1 Clock source selection

Separate clock sources can be selected for the counter units of Channel A and Channel B.

There are two types of clock sources: synchronous clock source and asynchronous clock source. Details of the clock source register configuration are shown below:

Synchronization Counting	BCONR.SYNS=0	BCONR.SYNCLK =0	PCLK1 and the 2, 4, 8, 16, 32, and 64, 128, 256, 512, 1024 crossover frequencies with crossover coefficients configured by BCONR.CKDIV[3:0].
		BCONR.SYNCLK =1	Internal hardware triggered event input
Asynchronous Counting	BCONR.SYNS=1	BCONR.ASYNCLK=0	The LRC clock source input and its 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 divisions are used as input clocks with the division factor configured by BCONR.CKDIV[3:0].
		BCONR.ASYNCLK= 1	The XTAL32 clock source input and its 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 divisions are used as input clocks with the division coefficients configured by BCONR.

3.2.2 Basic counting function

Each channel in the TIMER0 unit can independently set the reference count value and generate a compare match event when the count value and the reference count value match. For details, please refer to the relevant section of the user manual of this series.

3.2.3 Hardware triggered actions

The two channels in the TIMER0 unit have a common internal hardware trigger source, which can trigger timer counting, start, stop, clear and capture input actions. The hardware trigger source is selected via register HTSSR, please refer to the INTC chapter of the user manual of this chip series for details.

If an external interrupt event is used as a trigger source to trigger the event on CH_B of cell 1 of TIMER0, the configuration steps required are as follows:

- 1) Configure external interrupts that generate events ExtiCh03
- 2) Configure CH_B of TIMER0 unit 1 as hardware trigger mode, hardware trigger enable, trigger action is start

- 3) Configure register TMR0_HTSSR to select the hardware trigger source as EVT_PORT_EIRQ3.
- 4) Configure TIMER0 interrupt enable
- 5) Enable TIMER0 Unit 1 Counting Function

After completing the above configuration, an external interrupt event can trigger an interrupt to generate TIEMR0.

3.2.4 Interrupt and event output

Each unit contains 2 interrupt outputs, channel A and channel B interrupt outputs. For example, the compare match interrupt and the input capture interrupt of channel A share one interrupt output.

The event outputs of the TIMER0 function correspond to the interrupt outputs, as shown below.

Unit number	Interrupt output	Event Output
TIMER0 Unit 1 (TMR01)	INT_TMR01_GCMA INT_TMR01_GCMB	EVT_TMR01_GCMA EVT_TMR01_GCMB
TIMER0 Unit 2 (TMR02)	INT_TMR02_GCMA INT_TMR02_GCMB	EVT_TMR02_GCMA EVT_TMR02_GCMB

3.3 Cautions

When using the TIMER0 module, the following points need to be noted:

- 1) In the asynchronous counting method, the register read operation must be implemented in the count-stop state.
- 2) CSTA and BCONR.CSTB must be read to determine a successful write before the register flags can be written.
- 3) The compare match interrupt TMR_U1_GCMA for Unit 1 Channel A is only available in asynchronous counting mode.

3.4 Register Description

The following is the register list of TIMER0 module. For detailed register descriptions, please refer to the relevant chapters of this series chip user manual.

BASE ADDR: 0x40024000 (TMR01) 0x40024400 (TMR02)

Register Name	Symbols	Offset	Bit width	Reset value
Count value register	TMR0_CNTAR	0x0000h	32	0x00000000h
Count value register	TMR0_CNTBR	0x0004h	32	0x00000000h
Base value register	TMR0_CMPAR	0x0008h	32	0x0000FFFFh
Base value register	TMR0_CMPBR	0x000ch	32	0x0000FFFFh
Basic control register	TMR0_BCONR	0x0010h	32	0x00000000h
Trigger selection register	TMR0_HTSSR	(0x40010840h)	32	0x000001FFh
Status Flag Register	TMR0_STFLR	0x0014h	32	0x00000000h

4 Sample Code

4.1 Code Introduction

Users can write their own code to learn to verify the module according to the above workflow, or download the sample code of Device Driver Library (DDL) directly through the website of UW Semiconductors and use the sample of TIMER0 to verify.

The following section briefly describes the configuration involved in this sample BaseTimer code for the AN DDL-based TIMER0 module.

1) System clock, test LED port initialization

```
SysClkIni().
/*initialize LED port*/
stcPortInit.enPinMode = Pin_Mode_Out;
stcPortInit.enExInt = Enable;
stcPortInit.enPullUp = Enable.
/* LED0 and LED1 Port/Pin initialization */
PORT_Init(LED0_PORT, LED0_PIN, &stcPortInit);
PORT_Init(LED1_PORT, LED1_PIN,
&stcPortInit) .
/* Get pclk1 */
CLK_GetClockFreq(&stcClkTmp);
```

2) TIMER0 CH_A Peripheral enable and basic counter function configuration

```
/* Timer0 peripheral enable */
ENABLE_TMR0().
/*config register for channel A */
stcTimerCfg.Tim0_CounterMode = Tim0_Async;
stcTimerCfg.Tim0_AsyncClockSource = Tim0_XTAL32;
stcTimerCfg.Tim0_ClockDivision = Tim0_ClkDiv4;
stcTimerCfg.Tim0_CmpValue = 32000/4 - 1;
TIMER0_BaseInit(TMR_UNIT, Tim0_ChannelA, &stcTimerCfg).
```

3) TIMER0 CH_A Interrupt function configuration and enable

```
/* Enable channel A interrupt */
TIMER0_IntCmd(TMR_UNIT, Tim0_ChannelA, Enable).
/* Register TMR_INI_GCMA Int to Vect.No.001
*/ stcIrqRegiConf.enIRQn = Int001_IRQn.
/* Select Event interrupt function */
stcIrqRegiConf.enIntSrc = TMR_INI_GCMA.
/* Callback function */
stcIrqRegiConf.pfnCallback = Timer0A_CallBack.
/* Registration IRQ */
enIrqRegistration(&stcIrqRegiConf).
```

```
/* Clear Pending */
NVIC_ClearPendingIRQ(stcIrqRegiConf.enIRQn).
/* Set priority */
NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_15).
/* Enable NVIC */
NVIC_EnableIRQ(stcIrqRegiConf.enIRQn).
```

4) TIMER0 CH_B Counting function

and interrupt configuration omitted

5) Counting function enable

```
/*start timer0*/
TIMER0_Cmd(TMR_UNIT,Tim0_ChannelA,Enable);
TIMER0_Cmd(TMR_UNIT,Tim0_ChannelB,Enable).
```

6) Test code, count value reading

```
while(1)
{
    /* Read counter register of channelB*/
    u16cnt = TIMER0_GetCntReg(TMR_UNIT,Tim0_ChannelB);
    u16cmp = TIMER0_GetCntReg(TMR_UNIT, Tim0_ChannelB).

    /* Read counter register of channel A, need stop counter function for asynchronous mode*/
    TIMER0_Cmd(TMR_UNIT,Tim0_ChannelA,Disable).
    u16cnt = TIMER0_GetCntReg(TMR_UNIT,Tim0_ChannelA);
    u16cmp = TIMER0_GetCntReg(TMR_UNIT,Tim0_ChannelA);
    TIMER0_Cmd(TMR_UNIT,Tim0_ChannelA, Enable).

    u32tmp = 0xFFFFF;
    while(u32tmp--).
}
```

7) Interrupting service procedures

```
void Timer0A_CallBack(void)
{
    LED0_TOGGLE().
}

void Timer0B_CallBack(void)
{
    LED1_TOGGLE().
}
```

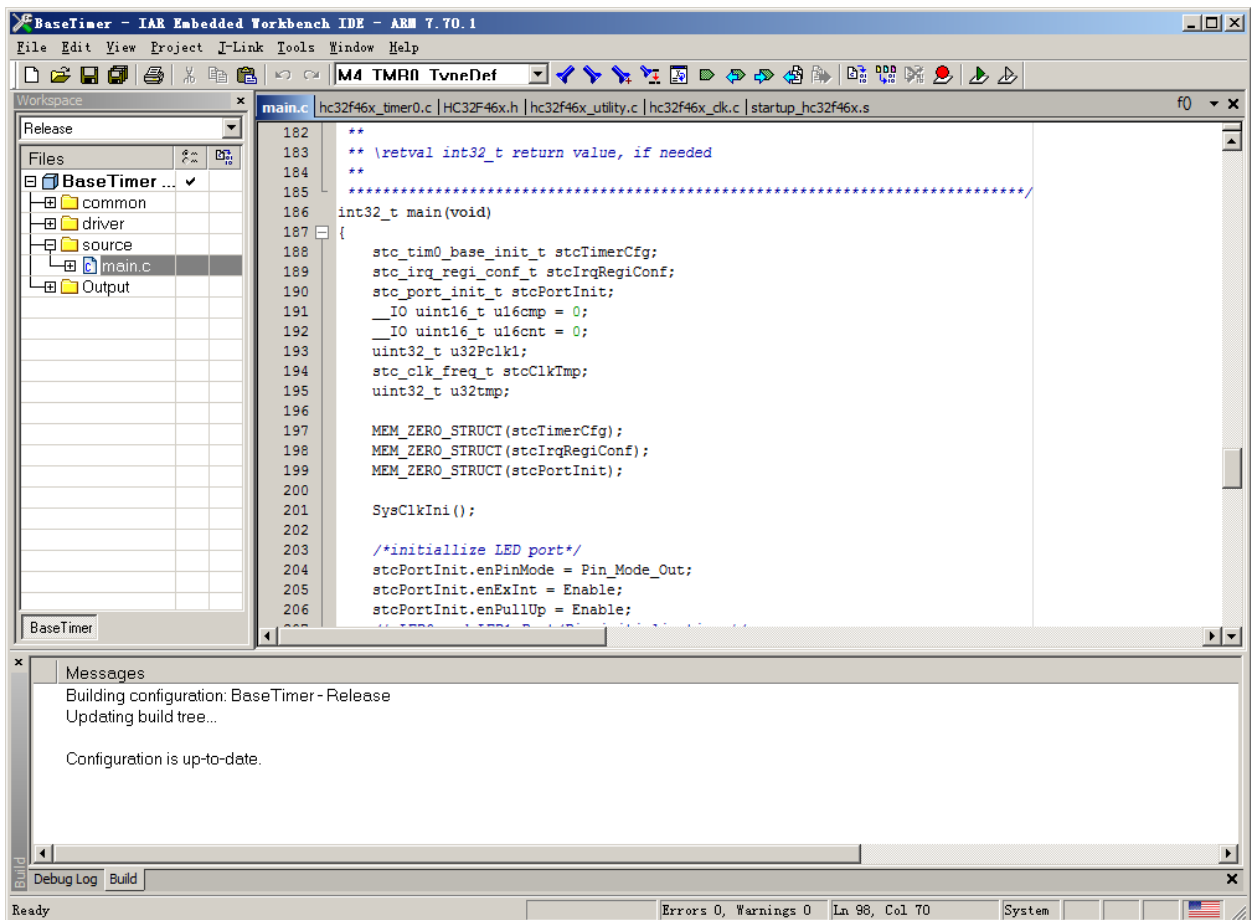
4.2 Code Run

Users can download the sample code (BaseTimer) of the HC32F460 DDL from the UW website and run the code with the evaluation board (EV-HC32F460-LQFP100-050-V1.1) to learn how to use the TIMER0 module.


The following section describes how to run the BaseTimer sample code on the evaluation board and observe the results:

- Verify that the correct IAR EWARM v7.7 tool is installed (please download the appropriate installation package from the official IAR website and refer to the user manual for installation).
- Download the HC32F460 DDL code from the UW Semiconductors website.
- Download and run the project file in BaseTimer\ at

1) Open the BaseTimer\ project and open the 'main.c' view as follows:



2) Click  to recompile the entire project.

3) Click  Download the code to the evaluation board and run it at full speed.

- 4) Observe the change of the LED on and off, at this time the red and green lights on the evaluation board flash at different frequencies, indicating the TIMER0
The counting and interrupt functions of CH_A and CH_B are operating normally.

5 Summary

The above section briefly introduces the TIMER0 of HC32F460 series, explains the registers and part of the operation flow of TIMER0 module, and demonstrates how to use the TIMER0 sample code, so that users can configure and use the TIMER0 module according to their needs in the actual development.

6 Version Information & Contact

Date	Versions	Modify records
2019/3/15	Rev1.0	Initial Release



If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: <http://www.hdsc.com.cn/mcu.htm>

Address: 39, Lane 572, Bibo Road, Zhangjiang Hi-

Tech Park, Shanghai, 201203, P.R. China

