HDSC 华大半导体 HUADA SEMICONDUCTOR

32 -bit microcontroller

Base Timer for **HC32L110 / HC32F003 / HC32F005** series

Applicable object

series	Product number
HC32L110	HC32L110C6UA
	HC32L110C6PA
	HC32L110C4UA
	HC32L110C4PA
	HC32L110B6PA
	HC32L110B4PA
HC32F003	HC32F003C4UA
	HC32F003C4PA
HC32F005	HC32F005C6UA
	HC32F005C6PA
	HC32F005D6UA

content

1 Summary

This application note mainly introduces the function and usage of Base Timer of HC32L110 / HC32F003 / HC32F005 series

Law.

2 Introduction to Timer

What is a **Timer?**

Timer, timer, the function is to repeatedly trigger the timer event of the specified window within the specified time interval.

(Quoted from 'Baidu Encyclopedia', 'Interactive Encyclopedia', 'Wikipedia')

Timer basic principle?

The timer generally counts or measures the time by counting the number of pulses of the known input clock signal, and the internal structure of each timer is realized.

In fact, it is a programmable adding counter, plus some control logic, and is programmed to set its working mode and state.

Timer application?

A timer, a very common external device that can be used to measure time intervals, used to generate events or determine

The time interval between two events can also be used to count external events, generate specific waveforms, etc.

3 Base Timer

3.1 Introduction

The Base timer of BGI HC32L110 / HC32F003 / HC32F005 series MCU includes three timers TIM0/1/2.

The TIM0/1/2 functions are identical. TIM0/1/2 is a synchronous timer/counter, which can be used as a timer for the 16-bit auto-reload function

/counter, can also be used as a 32-bit timer/counter without reload function. TIM0/1/2 can count external pulses or

implement system timing.

The functions of this module include:

- Counting function

Used to measure the number of times an event occurs, the input signal is sampled by the internal Pclk.

- Timing function

Used to generate interval timing, the timer accumulates once every clock, and counts to the maximum value to generate an overflow interrupt.

- Buzzer function

The function of driving the Buzzer can be realized through the inversion output function of the timer.

- Interconnect function

Through the internal interconnection, the automatic identification of the UART baud rate can be realized, the width of the VC comparison output can be measured, and the external control counting can be realized.

3.2 Description

In this part, the Base Timer of the HC32L110 / HC32F003 / HC32F005 series will be mainly introduced, including registers and functions.

work process.

3.2.1 Register introduction

The operation of the Base Timer module is mainly carried out through the following registers:

abbreviation	register name
TIMx_ARR	TIM0/1/2 reload register
TIMx_CNT	TIM0/1/2 16-bit mode count register
TIMx_CNT32	TIM0/1/2 32-bit mode count register
TIMx_CR	TIM0/1/2 Control Register
TIMx_IFR	TIM0/1/2 Interrupt Flags
TIMx_ICLR	TIM0/1/2 Interrupt Clear Register

3.2.2 Workflow Introduction

Base Timer unified workflow:

1. Configure counting clock, working mode, gating, etc.;

2. Set initial count value or comparison value, reload value;

3. Start the timer;

4. Wait for the interrupt and process the result (if needed);

5. Stop the timer.

4 Sample code

4.1 Code introduction

Users can write their own code to learn and verify the module according to the above workflow, or directly through Huada Semiconductor

The sample code downloaded from the website to BT (Base Timer) directly uses the API functions provided by the BT driver library for coding and calibration.

test application.

The following sections briefly describe the functionality of the various parts of the code:

1) BT data declaration and initialization:

```
//BT TEST DATA INIT
stc_bt_config_t stcConfig;
en_result_t enResult = Error; uint32_t
u32InitCntData = 0xFFFF0000;
```

2) Interrupt and initialization configuration code:

```
//INT ENABLE
    EnableNvic(TIM0_IRQn, 3, TRUE);
    Bt_EnableIrq(TIM0);

    //Count external ext0, GPIO configuration
    Gpio_SetFunc_EXT0_P34();

    stcConfig.enGateP = BtPositive;
    stcConfig.enGate = BtGateDisable;
    stcConfig.enPRS = BtPCLKDiv8;
    stcConfig.enTog = TogDisable;
    stcConfig.enCT = BtCounter;
    stcConfig.enMD = BtMode1;

    stcConfig.pfnTim0Cb = Bt0Int;

    if (Ok != Bt_Init(TIM0, &stcConfig)) {

        enResult = Error;
}

Bt_Cnt32Set(TIM0, u32InitCntData);
```

3) BT starts and runs:

```
        Bt_Run(TIM0);
```

4) BT waits for the interruption to stop and process the result:

```
//Enter the interrupt...
        while(1) {

              if (0x01 == u32BtTestFlag)
              {
                    u32BtTestFlag = (u32BtTestFlag & (~0x01));
                    Bt_Stop(TIM0);
                    enResult = Ok;
                    break;
              }
        }
```

The counting function operation of external 32K clk using BT's TIM0 can be completed through the above code.

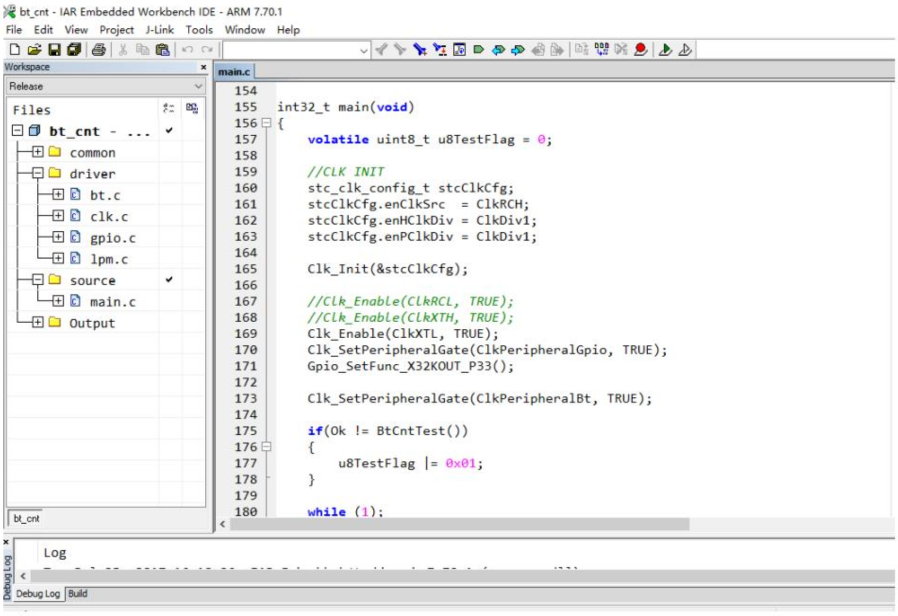
4.2 Code running

Users can download the BT sample code through the website of Huada Semiconductor, and run the relevant code with the evaluation board to learn and use BT module.

The following sections describe how to run the BT sample code on the evaluation board and observe the results:

- Make sure to install the correct IAR (or Keil, here IAR is used as a sample description, the operation method in Keil is similar) tool (please download the corresponding installation package from Huada Semiconductor, and refer to the user manual for installation).
- Download the BT sample code from the Huada Semiconductor website.
- Download and run the sample code:

1) Open the bt_cnt project, and open the 'main.c' view as follows:

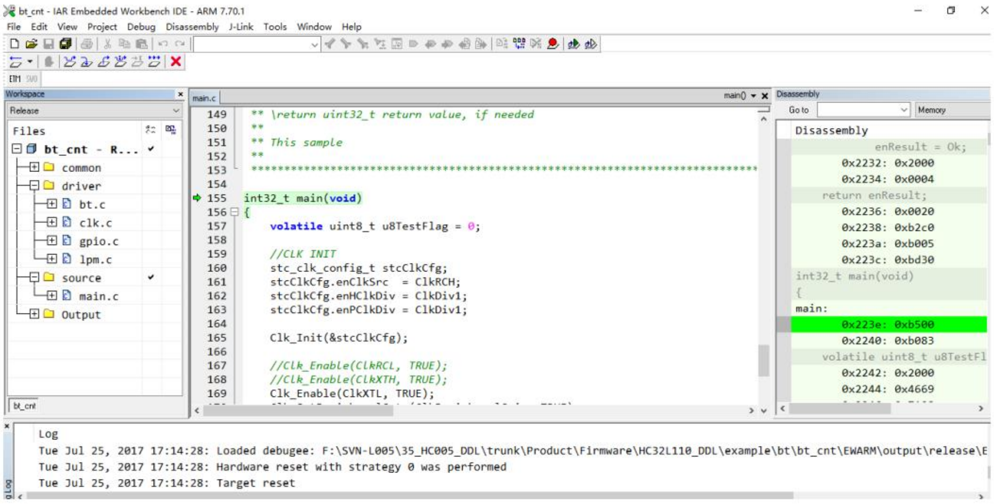


```
int32_t main(void)
{
    volatile uint8_t u8TestFlag = 0;

    //CLK INIT
    stc_clk_config_t stcClkCfg;
    stcClkCfg.enClkSrc  = ClkRCH;
    stcClkCfg.enHClkDiv = ClkDiv1;
    stcClkCfg.enPClkDiv = ClkDiv1;

    Clk_Init(&stcClkCfg);

    //Clk_Enable(ClkRCL, TRUE);
    //Clk_Enable(ClkXTH, TRUE);
    Clk_Enable(ClkXTL, TRUE);
    Clk_SetPeripheralGate(ClkPeripheralGpio, TRUE);
    Gpio_SetFunc_X32KOUT_P33();

    Clk_SetPeripheralGate(ClkPeripheralBt, TRUE);

    if(Ok != BtCntTest())
    {
        u8TestFlag |= 0x01;
    }

    while (1);
```

2) Click Recompile to link the entire project.

3) Click to download the code to the evaluation board.

4) You can see a view similar to the following:



```
** \return uint32_t return value, if needed
**
** This sample
**
******************************************************************************

int32_t main(void)
{
    volatile uint8_t u8TestFlag = 0;

    //CLK INIT
    stc_clk_config_t stcClkCfg;
    stcClkCfg.enClkSrc  = ClkRCH;
    stcClkCfg.enHClkDiv = ClkDiv1;
    stcClkCfg.enPClkDiv = ClkDiv1;

    Clk_Init(&stcClkCfg);

    //Clk_Enable(ClkRCL, TRUE);
    //Clk_Enable(ClkXTH, TRUE);
    Clk_Enable(ClkXTL, TRUE);
```

```
Disassembly
            enResult = Ok;
        0x2232: 0x2000
        0x2234: 0x0004
    return enResult;
        0x2236: 0x0020
        0x2238: 0xb2c0
        0x223a: 0xb005
        0x223c: 0xbd30
int32_t main(void)
{
main:
        0x223e: 0xb500
        0x2240: 0xb083
    volatile uint8_t u8TestFl
        0x2242: 0x2000
        0x2244: 0x4669
```

```
Log
Tue Jul 25, 2017 17:14:28: Loaded debugee: F:\SVN-L005\35_HC005_DDL\trunk\Product\Firmware\HC32L110_DDL\example\bt\bt_cnt\EWARM\output\release\E
Tue Jul 25, 2017 17:14:28: Hardware reset with strategy 0 was performed
Tue Jul 25, 2017 17:14:28: Target reset
```

5) Set a breakpoint on the line after 'main(void)':

6) This example needs to use jumper to connect P33 and P34 ports on the Demo board.

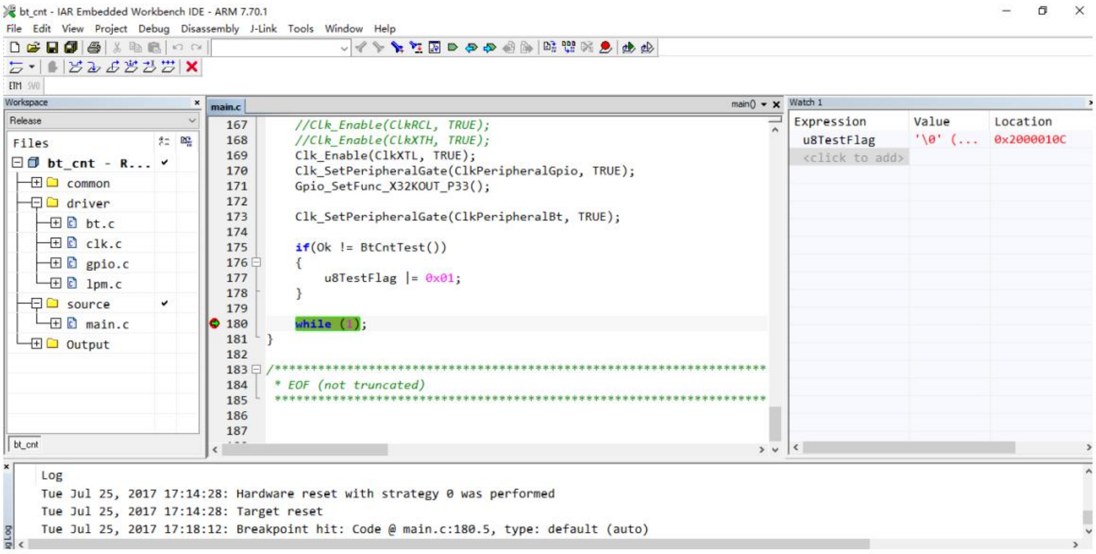
7) Click "View -> Watch -> Watch1" to open a 'watch1' window, and add the 'u8TestFlag' variable to observe its

numerical value.

8) Click Run.

9) The code runs and stops at the breakpoint of 'main(void)'. If 'u8TestFlag = 0', it means that the encoding and verification functions are working correctly.

Do it correctly, as shown below:


bt_cnt - IAR Embedded Workbench IDE - ARM 7.70.1
File Edit View Project Debug Disassembly J-Link Tools Window Help
Workspace
Release
Files
bt_cnt - R...
common
driver
bt.c
clk.c
gpio.c
lpm.c
source
main.c
Output
bt_cnt
main.c
main()
Watch 1

```
    //Clk_Enable(ClkRCL, TRUE);
    //Clk_Enable(ClkXTH, TRUE);
    Clk_Enable(ClkXTL, TRUE);
    Clk_SetPeripheralGate(ClkPeripheralGpio, TRUE);
    Gpio_SetFunc_X32KOUT_P33();

    Clk_SetPeripheralGate(ClkPeripheralBt, TRUE);

    if(Ok != BtCntTest())
    {
        u8TestFlag |= 0x01;
    }

    while (1);
}

/******************************************************************************
 * EOF (not truncated)
 ******************************************************************************
```

Expression	Value	Location
u8TestFlag	'\0' (...	0x2000010C
<click to add>		

```
Log
Tue Jul 25, 2017 17:14:28: Hardware reset with strategy 0 was performed
Tue Jul 25, 2017 17:14:28: Target reset
Tue Jul 25, 2017 17:18:12: Breakpoint hit: Code @ main.c:180.5, type: default (auto)
```

10) You can close the project file after running.

11) Users can also learn the timing, Buzzer and other functions of Base Timer through other samples or their own research.

5 Summary

The above chapters briefly introduced the Base Timer of the HC32L110 / HC32F003 / HC32F005 series, and explained in detail

The registers and operation flow of the Base Timer module demonstrate how to use the relevant sample code for operation and use.

During development, users can configure and use other functions of Base Timer according to their own needs.

6 Version Information & Contact Information

date	Version revision record
2018/6/4	Rev1.0 initial release

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

Email: mcu@hdsc.com.cn

Website: www.hdsc.com.cn

Mailing address: No. 39, Lane 572, Bibo Road, Zhangjiang Hi-Tech Park, Shanghai

Postcode: 201203