

32-bit microcontrollers

Universal Timer TIMERA for HC 32F460 Series

Applicable objects

| | |
|----------|-----------|
| F Series | HC 32F460 |
|----------|-----------|

Table of Contents Table of Contents

| | | |
|----------|--|-----------|
| 1 | Abstract | 3 |
| 2 | TIMERA Introduction | 3 |
| 2.1 | Main Features | 3 |
| 2.2 | Basic Block Diagram | 4 |
| 3 | TIMERA for HC32F460 Series | 5 |
| 3.1 | Waveform mode | 5 |
| 3.1.1 | Sawtooth wave mode | 5 |
| 3.1.2 | Triangular wave mode | 5 |
| 3.2 | Basic functions | 6 |
| 3.2.1 | Basic Count | 6 |
| 3.2.2 | Capture input | 6 |
| 3.2.3 | Compare Outputs | 7 |
| 3.3 | Orthogonal coding | 8 |
| 3.3.1 | Position counting mode | 8 |
| 3.3.2 | Rotation counting mode | 9 |
| 3.4 | Special Features | 10 |
| 3.4.1 | Digital filtering | 10 |
| 3.4.2 | Cache function | 11 |
| 3.4.3 | Synchronized start | 12 |
| 3.5 | Register Introduction | 13 |
| 4 | Sample Code | 14 |
| 4.1 | Code Introduction | 14 |
| 4.2 | Workflow | 17 |
| 4.3 | Code Run | 18 |
| 5 | Summary | 19 |
| 6 | Version Information & Contact | 20 |

1 Abstract

This application note introduces the HC32F460 series general-purpose timer (TIMERA) module and briefly explains how to use it by

How TIMERA's quadrature counting function implements 3-phase quadrature counting.

2 TIMERA Introduction

TIMERA is a timer with 16-bit count width and 8 PWM outputs. The timer supports both triangle waveform and sawtooth waveform modes to generate various PWM waveforms, synchronous counter start, cache function for comparison reference value register, 2-phase quadrature encoding count and 3-phase quadrature encoding count.

This series is equipped with 6 units of TIMERA, which can achieve a maximum of 48 PWM outputs.

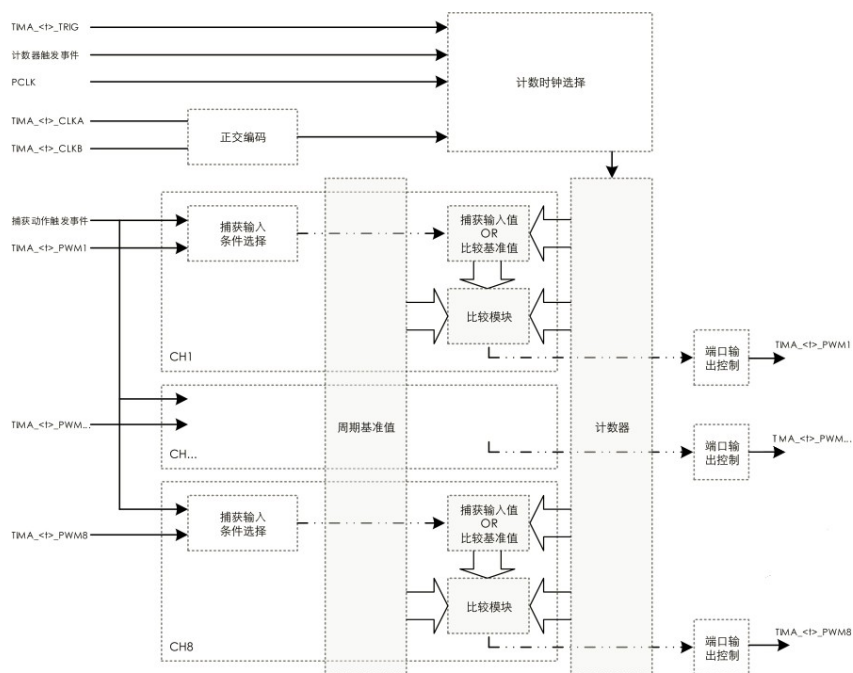
2.1 Main Features

- Waveform mode: sawtooth wave, triangle wave
- Adding and subtracting counting direction
- Base value caching function
- Orthogonal coding count
- Synchronous start counter
- Compare the output of matching events
- Compare match interrupts
- Periodic Matching Interruptions

2.2 Basic Block Diagram

The basic block diagram of TIMERA is shown below, in which "<t>" is the cell number, i.e. "<t>" is 1~6; the subsequent reference to "<t>" in this paper "

All refer to the unit number.



The input and output ports in the block diagram are listed in the following table:

| Port Name | Direction | Function |
|---------------|-----------------|--|
| TIMA_<t>_PWMm | Input or output | Capture input event port or PWM output port (m=1~8) |
| TIMA_<t>_CLKA | Input | Capture input event port or PWM output port (m=1~8) |
| TIMA_<t>_CLKB | | |
| TIMA_<t>_TRIG | Input | Hardware triggered start, stop, and clear event input port |

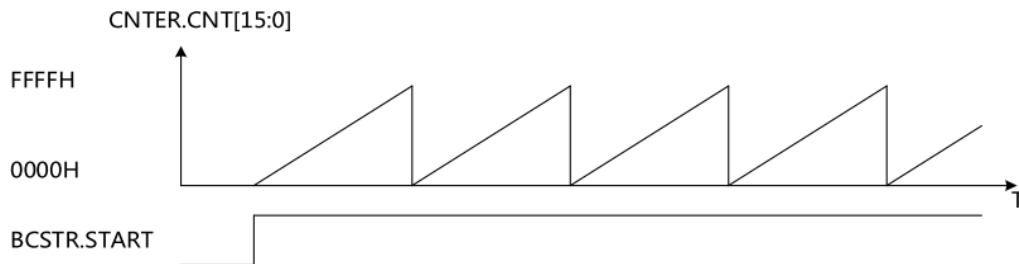
3 TIMERA for HC32F460 Series

3.1 Waveform mode

TIMERA has 2 basic counting waveform modes, sawtooth mode and triangular waveform mode.

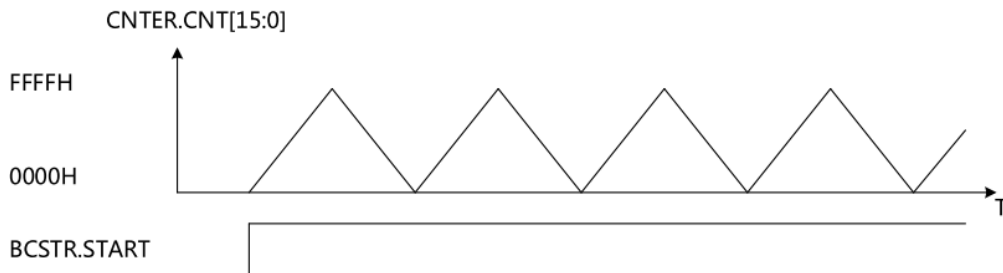
3.1.1 Sawtooth wave mode

The counter is set to sawtooth mode and the basic waveform is shown below:



3.1.2 Triangular wave mode

The counter is set to triangle wave mode and the basic waveform is as follows:



3.2 Basic Functions

3.2.1 Basic Count

Each TIMERA unit can be set to a reference count value, and depending on the configured mode can be selected to count up, count down

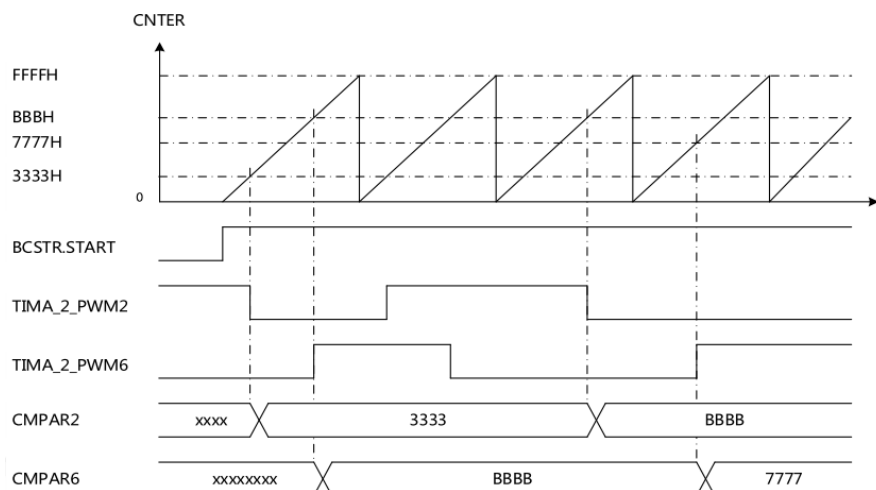
Each channel can be set to generate a count comparison match event when the count value and the reference value are equal.

The OVFF or UDFB bit of the Control Status Register (BCSTR) is set to 1 when the ramp mode is incrementally counted to the upper overflow point, the ramp mode is decrementally counted to the lower overflow point, and the delta mode is counted to the valley or peak point.

3.2.2 Capture input

The capture input function is available for each PWM output channel of each TIMERA unit to save the captured count value. $TIMA_{<t>_CLKA}$, $TIMA_{<t>_CLKB}$, $TIMA_{<t>_TRIG}$, and $TIMA_{<t>_PWMn}$ (when the capture input function is active) port inputs of each unit have digital filtering function.

When the capture input condition is valid, the current count value is stored in the corresponding register ($CMPARn$) ($n=1\sim8$), and the capture input condition can be selected from the internal capture action trigger event (selected via HTSSR1 register), $TIMA_{<t>_PWMn}$ port input, etc. The specific condition selection can be set via the HICP bit of the capture control register ($CCONRn$) ($n=1\sim8$). The specific conditions can be set by the HICP bits in the Capture Control Register ($CCONRn$) ($n=1\sim8$), and the following figure shows an example of capture input action for Unit 2:

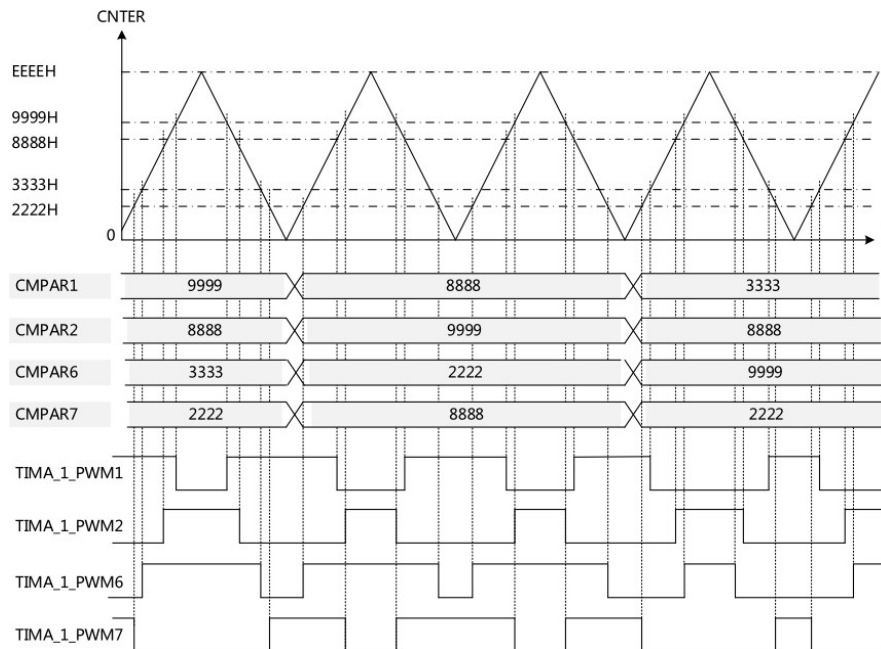


3.2.3 Compare Outputs

Each TIMERA unit contains 8 internal comparison outputs (TIMA_<t>_PWMn), which can output the specified level when the count value matches the comparison reference value. When the timer count value and CMPARn are equal, the TIMA_<t>_PWMn port outputs the specified level (n=1~8).

TIMERA's internal 8 outputs TIMA_<t>_PWMn, each of which can be accessed via the port control register

(The following figure shows an example of PWM output waveforms for channels 1, 2, 6 and 7 in the triangle mode of cell 1:



3.3 Orthogonal coding

Consider the TIMA_<t>_CLKA input as AIN input, the TIMA_<t>_CLKB input as BIN input, and the TIMA_<t>_CLKA input as BIN input,

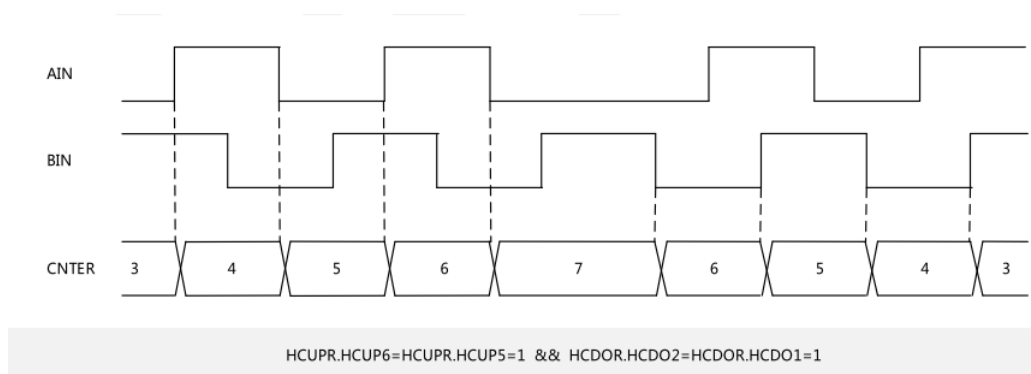
The TIMA_<t>_TRIG input is treated as a ZIN input, and TIMERA is able to implement a quadrature encoded count of the three inputs.

The AIN and BIN of each unit alone can realize the position counting mode; the AIN, BIN and ZIN of two units can realize the revolution counting mode by combining their actions, where the unit for position counting is called position counting unit and the unit for revolution counting is called revolution counting unit. In the metric counting mode, every two units are combined with each other (unit 1, 2 combination; unit 3, 4 combination; unit 5, 6 combination), and the position counting unit and the metric counting unit can be specified arbitrarily within the combination.

3.3.1 Position counting mode

The orthogonal encoding position counting mode means that the basic counting function, phase difference counting function and direction counting function are implemented according to the input of AIN and BIN.

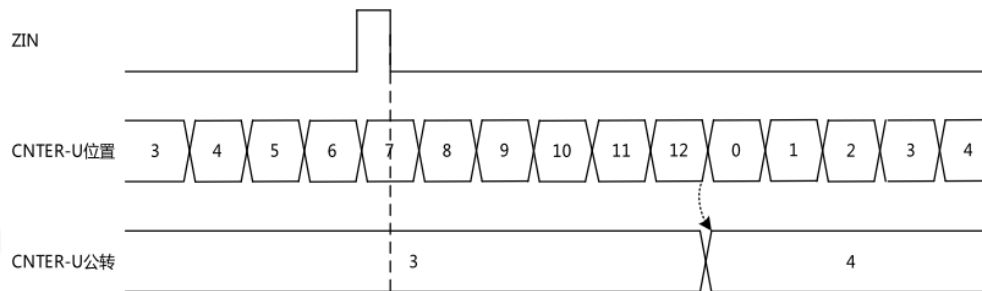
Phase difference counting means counting according to the phase relationship between AIN and BIN. Depending on the setting, 1X counting, 2X counting, 4X counting, etc. can be realized, and the following figure shows the phase difference 2X counting:



3.3.2 Rotation counting mode

The orthogonal coding revolution counting mode refers to adding ZIN input events to the AIN and BIN counting to realize the judgment of revolution number, etc. The revolution counting mode can realize Z-phase counting function, position overflow counting function and mixed counting function according to the counting mode of revolution timer.

The position overflow count is an overflow event generated when the count of the position counter unit overflows or underflows, which triggers the timer of the revolution counter unit to perform a count (the input of ZIN does not perform the count action of the revolution counter unit and the zero action of the position counter unit during this counting method). The hardware incremental (decremental) event selection register of the revolution counter unit (HCUPR or HCDOR) the incremental (decremental) event bit12~11 enables, the overflow event of the position counting unit can trigger the revolution counting unit to achieve one count, as shown in the following figure:

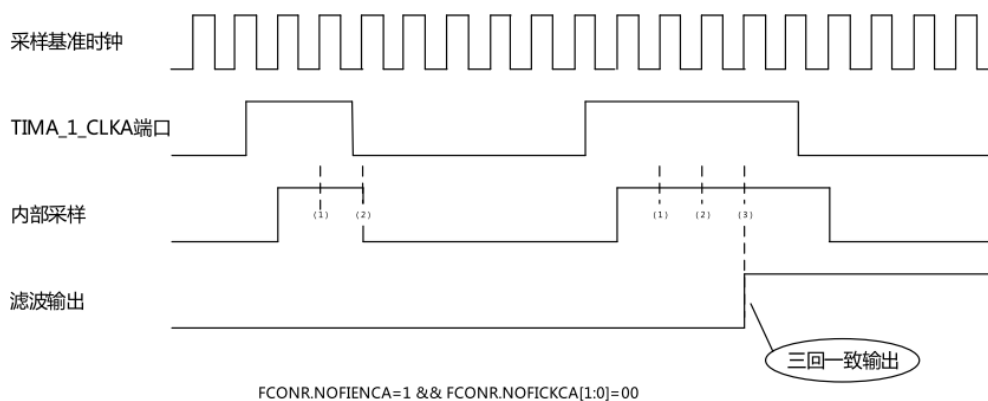


3.4 Special Features

3.4.1 Digital filtering

The TIMA_<i>t>_CLKA, TIMA_<t>_CLKB, TIMA_<t>_TRIG, and TIMA_<t>_PWMn (at the capture input function) port inputs of each unit have digital filtering functions, and the enable of the filtering function and the selection of the filtering clock for each port can be achieved by setting The enable of the filter function and the selection of the filter clock for each port can be achieved by setting the corresponding bits of the filter control register (FCONR) and the capture control register (CCONRn) (n=1~8).

The following figure shows the CLKA port filtering action for Unit 1:

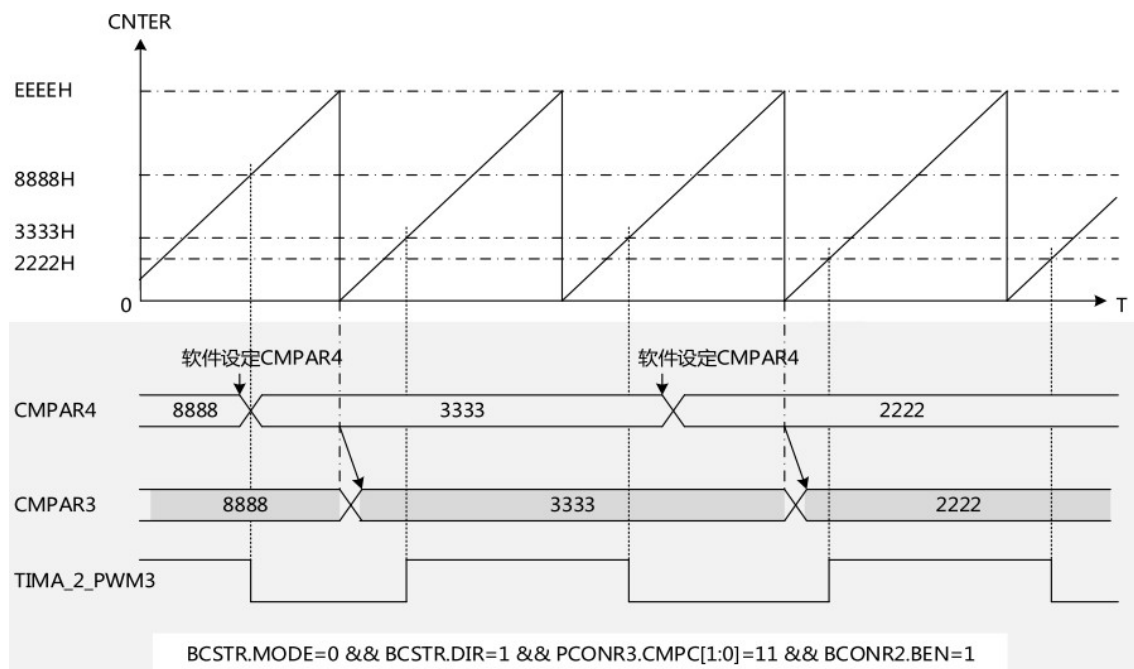


3.4.2 Cache function

A total of eight comparison reference registers (CMPARn) of TIMERA can implement cache functions in pairs (n=1~8), i.e. CMPAR2 as cache reference value for CMPAR1, CMPAR8 as cache reference value for CMPAR7, and cache control registers (BCONRm) for each of the four groups of cache functions (m=1~4).

When the BEN bit of the cache control register (BCONRm) is set, the cache function becomes active (m=1~4), and a cache transfer occurs when the counter counts a specific time point (CMPAR8/6/4/2->CMPAR7/5/3/1), and this "specific time point" has There are several cases, see the reference manual for details.

The following figure shows the cache transfer for the sawtooth wave mode of cell 2:

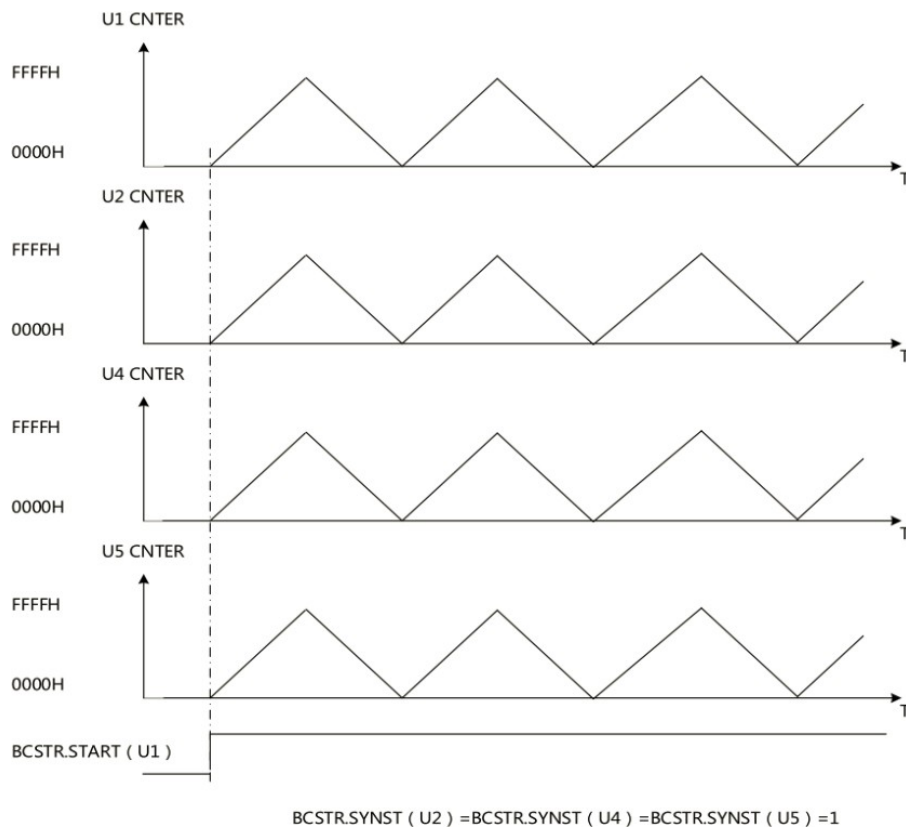


3.4.3 Synchronized start

TIMERA can realize software synchronous start or hardware synchronous start, unit 2 to unit 6 can choose to realize synchronous start with unit 1, when the BCSTR. START bit of Unit 1 is set to 1, the counter of the synchronized unit (Unit 2~Unit 6) starts software synchronous counting.

If any bit in HCONR.HSTA1~0 of unit 1 is 1 in hardware setting, and the counter of the synchronized unit (unit 2~unit 6) starts hardware synchronous counting when the corresponding hardware event of unit 1 occurs, the corresponding bit in HCONR.HSTA1~0 of the synchronized unit (unit 2~unit 6) must also be set to be valid when the hardware synchronous counting start function is selected.

The following figure shows the software synchronization start when BCSTR.SYNST=1 is set for Unit 2, Unit 4 and Unit 5:



3.5 Register Introduction

The registers of the general-purpose timer TIMERA module are shown in the following table, for more details, please refer to the user manual:

| Register abbreviation | Register Function | Remarks |
|-----------------------|---|---------|
| TMRA_CNTER | General purpose count value register | |
| TMRA_PERAR | Cycle reference value register | |
| TMRA_CMPARn | Compare reference value register | n=1~8 |
| TMRA_BCSTR | Control Status Register | |
| TMRA_ICONR | Interrupt control register | |
| TMRA_ECONR | Event Control Register | |
| TMRA_FCONR | Filter control register | |
| TMRA_STFLR | Status Flag Register | |
| TMRA_BCONRn | Cache Control Register | n=1~4 |
| TMRA_CCONRn | Capture Control Register | n=1~8 |
| TMRA_PCONRn | Port Control Register | n=1~8 |
| TMRA_HCONR | Hardware trigger event selection register | |
| TMRA_HCUPR | Hardware recursive event selection register | |
| TMRA_HCDOR | Hardware decrement event selection register | |
| TMRA_HTSSRn | Internal trigger event selection register | n=0~1 |

4 Sample Code

4.1 Code Introduction

Users can write their own code according to the above workflow to learn and verify the module, or they can download the Device Driver Library (DDL) of HC32F460 series MCUs directly from the website of UW Semiconductors to experience the quadrature coding counting function of TIMERA.

The following section is based on a sample of the orthogonal coding function of the TIMERA module of the DDL

timera_position_overflow_count code, which briefly describes the use of TIMERA's orthogonal encoding count:

- 1) Turn on the TIMERA clock:

```
/* Initialize I/O */
Led_Init().
```

- 2) Configure the IO used by the TIMERA orthogonal encoding counting function:

```
/* Initialize I/O */
/* Configuration TIMERA coding pin */
PORT_SetFunc(TIMERA_UNIT1_CLKA_PORT, TIMERA_UNIT1_CLKA_PIN,
TIMERA_UNIT1_CLKA_FUNC, Disable).
PORT_SetFunc(TIMERA_UNIT1_CLKB_PORT, TIMERA_UNIT1_CLKB_PIN,
TIMERA_UNIT1_CLKB_FUNC, Disable).
```

- 3) Initialize the TIMERA Unit 1 and Unit 2 basic counting functions:

```
/* Configuration timera unit 1 structure */
stcTimerInit.enCntMode = TimerCountModeSawtoothWave;
stcTimerInit.enSyncStartupEn = Disable.
stcTimerInit.u16PeriodVal = 1000;
TIMERA_Baselnit(TIMERA_UNIT1, &stcTimerInit).
TIMERA_IrqCmd(TIMERA_UNIT1, TimerIrqOverflow, Enable).

/* Configure timera unit 2 structure */
stcTimerInit.u16PeriodVal = 6;
TIMERA_Baselnit(TIMERA_UNIT2, &stcTimerInit).
TIMERA_IrqCmd(TIMERA_UNIT2, TimerIrqOverflow, Enable).
```

- 4) Initialization of the TIMERA Unit 1 and Unit 2 orthogonal code counting function:

```
/* Configure coding count structure */
stcTimerCondngInit.enIncClkBHighAndClkARisingEn = Enable;
stcTimerCondngInit.enClkAFilterEn = Enable;
stcTimerCondngInit.enClkAClkDiv = TimerFilterPclkDiv4;
stcTimerCondngInit.Enable; stcTimerCondngInit.enClkAClkDiv =
```

```
stcTimeraCondngInit.enClkBClkDiv = TimeraFilterPclkDiv4;
TIMERA_OrthogonalCodingInit(TIMERA_UNIT1, &stcTimeraCondngInit).

/* Configure position overflow count structure */
MEM_ZERO_STRUCT(stcTimeraCondngInit);
stcTimeraCondngInit.enIncAnotherUnitOverflowEn = Enable;
TIMERA_OrthogonalCodingInit(TIMERA_UNIT2, &stcTimeraCondngInit).
```

5) Initialize the TIMERA Unit 1 and Unit 2 overflow interrupt function:

```
/* Configure count overflow interrupt of timera unit 1 */
stcIrqRegiConf.enIntSrc = TIMERA_UNIT1_OVERFLOW_INT;
stcIrqRegiConf.enIRQn = Int006_IRQn;
stcIrqRegiConf.pfnCallback = TimeraUnit1Over_IrqCallback;
enIrqRegistration(&stcIrqRegiConf); NVIC_ClearPendingIRQ(
stcIrqRegiConf.enIRQn).
NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_15);
NVIC_EnableIRQ(stcIrqRegiConf.enIRQn).

/* Configure count overflow interrupt of timera unit 2 */
stcIrqRegiConf.enIntSrc = TIMERA_UNIT2_OVERFLOW_INT;
stcIrqRegiConf.enIRQn = Int007_IRQn;
stcIrqRegiConf.pfnCallback = TimeraUnit2Over_IrqCallback;
enIrqRegistration(&stcIrqRegiConf); NVIC_ClearPendingIRQ(
stcIrqRegiConf.enIRQn).
NVIC_SetPriority(stcIrqRegiConf.enIRQn, DDL_IRQ_PRIORITY_15);
NVIC_EnableIRQ(stcIrqRegiConf.enIRQn).
```

6) Start TIMERA Unit 1 and Unit 2 counts:

```
/* Timera unit 1 and unit 2 startup */
TIMERA_Cmd(TIMERA_UNIT1, Enable);
TIMERA_Cmd(TIMERA_UNIT2, Enable).
```

7) Configure the function generator, channel A and B output frequency (1000HZ), duty cycle is the same, connect the function generator output channel to PE9 and PE11 pins, modify the function generator phase configuration, channel B phase ahead of channel A 1/4 cycle, observe LED0 blink; LED0 state switch 6 times will trigger LED1 state switch.

```
/**
*****
** \brief Timera unit 1 count overflow callback function
**
** \param [in] None
**
** \retval None
*****/
void TimeraUnit1Over_IrqCallback(void)
{
    LED0_TOGGLE().
    TIMERA_ClearFlag(TIMERA_UNIT1, TimeraFlagOverflow).
}
```

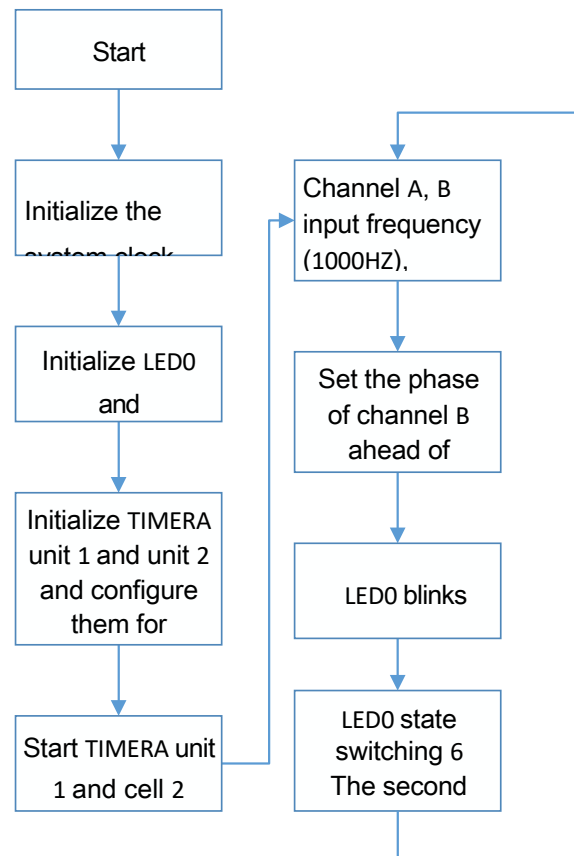
```

/**
*****
** \brief Timera unit 2 count overflow callback function
**
** \param [in] None
**
** \retval None
* * * * *
* * * * * void
TimeraUnit2Over_IrqCallback(void)
    LED1_TOGGLE().
    TIMERA_ClearFlag(TIMERA_UNIT2, TimeraFlagOverflow).
}

```


4.2 Workflow

The TIMERA operation flow in the sample code is shown in the following diagram:

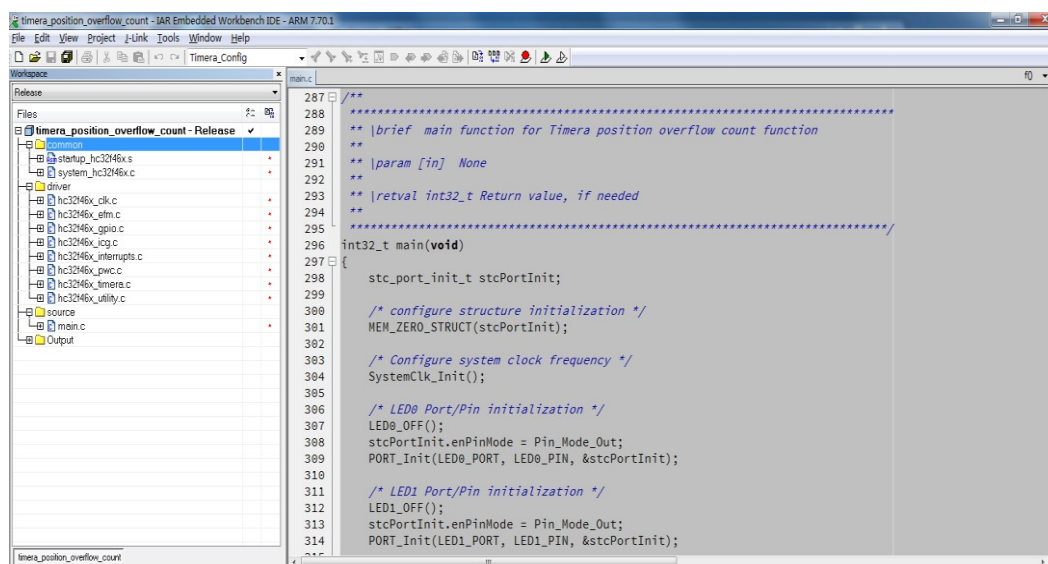




4.3 Code Run

Users can download sample DDL code (timera_sawtooth_wave_base_timer, timera_sawtooth_wave_capture_input, timera_triangular_wave_compare_output, timera_position_overflow_count) from the UW Semiconductors website. , timera_position_overflow_count), and the evaluation board (e.g. 'EV-HC32F460-LQFP100-050-V1.1') to run the relevant code to learn to use the TIMERA module.

The following section describes how to compile and run the timera_position_overflow_count sample code on the 'EV-HC32F460-LQFP100-050-V1.1' evaluation board with IAR EWARM 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).
- Get the 'EV-HC32F460-LQFP100-050-V1.1' evaluation board.
- Download the HC32F460 DDL code from the UW Semiconductors website.
- Download and run the project file in timera\timera_position_overflow_count \:
- 1) Open the timera_position_overflow_count \ project and open the 'main.c' view as follows:



- 2) Click  Recompile the entire project;
- 3) Click  Download the code to the evaluation board and run it at full speed;
- 4) Configure the function generator with the same output frequency (1000HZ) and duty

- cycle for channels A and B. Connect the function generator output channels to pins PE9 and PE11;
- 5) Modify the function generator phase configuration so that the phase of channel B overruns channel A by 1/4 cycle and observe LED0 blinking;

6) Observe the test board, LED0 state switch 6 times will trigger LED1 state switch.

5 Summary

The above section briefly introduces the TIMERA registers and function modes of HC32F460 series. It demonstrates how to operate the TIMERA

Sample code for orthogonal coding counts, in development users can use the TIMERA module according to their actual needs.

6 Version Information & Contact

| Date | Versions | Modify records |
|-----------|----------|-------------------------|
| 2019/3/15 | Rev1.0 | Initial Release |
| 2020/8/26 | Rev1.1 | Update supported models |
| | | |



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: 10/F, Block A, 1867 Zhongke Road, Pudong

New Area, Shanghai, 201203, P.R. China

