# XT32H05x

# XT32 microcontroller TIMB

# Application notes

Rev 0.0.0

**Original Release Date: 10-Oct-2023**

**Revised :**

# Revision History

| Release | Date | Author | Summary of Change |
|---|---|---|---|
| V0.0.0 | 10/10/2023 | Shirling Liu | Initial |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Contents

# List of Figures

# List of Tables

# 1 Introduce

This application note serves as a comprehensive guide for software developers, offering essential information on TIMB configurations for basic timer (TIMB). It covers fundamental concepts and provides guidelines to ensure proper utilization of basic timers in software development projects. Whether you're a beginner or an experienced developer, this document will equip you with the necessary knowledge and best practices to effectively configure and utilize timer in your applications.

## 1.1 Required peripherals

This application involves PADI module and GPIO module.

Table 1. Modules in example

| Sub-module | Peripheral use | Note |
|---|---|---|
| PADI | 4 ports as GPIO | |
| TIMB | 2 basic timers | |
| GPIO | LEDs show the TIMB interrupt callback state; button trigger the timer to start | |

### 1.2 Compatible devices

This example is compatible with the devices in Table 2.

Table 2. Device list

| Product | EVB |
|---|---|
| XT32H050 | XB002823 |
| | |

## 2 Design description

### 2.1 Feature overview

XT32H0 microcontroller has two groups basic timer, TIMB1 (TIMB11~TIMB14) and TIMB2 (TIMB21~TIMB24), which provides total 8 basic timers within 32-bit down counter. These timers include the following features:

- Support two counter mode:
  - Free-running mode: Timer loads the maximum value(0xFFFFFFFF) and counter wrapping to its maximum value.
  - User-defined count mode: Timer loads count from register TIMERx_LDR which is any value by user defined.

- Interrupt generation with mask
- Load initial value when Timer is enabled after reset or disabled or timer counts to 0.

## 2.2 Design steps

Here, the example uses TIMB13 and TIMB22 as basic timers.

1. Set TIMB1 and TIMB2 source clock and clock divider.
2. Configure TIMB1 parameters: counter mode, set load count.
3. Configure TIMB1 parameters: counter mode, set load count.
4. Configure port to be TIMB13 and TIMB22 toggle output.
   - PADI_IDX_IO37_BTOUT13, means select and enable the IO37(pin 41) as TIMB13 output.
   - PADI_IDX_IO27_BTOUT22, means select and enable the IO27(pin 31) as TIMB22 output.

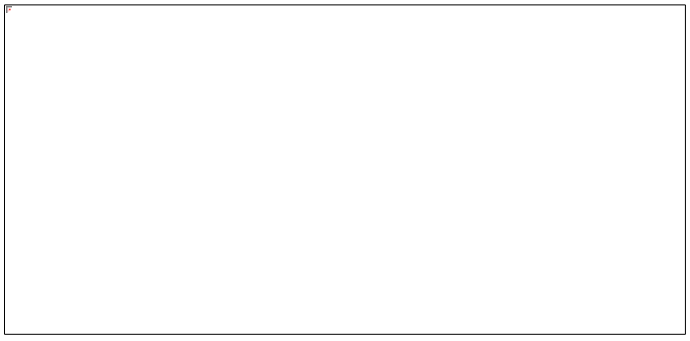

Figure 1. IO function selection

Note: please refer to XT32H0xxB—reference manual document to find the assignment relationship between pin41 with IO37, pin31 with IO27

5. Enable timer group1 (TIMB1) and timer group1 (TIMB2) Interrupt。
6. Process button1 to start the TIMB13 and TIMB22, button2 to stop them.

## 2.3 Design considerations

## 2.4 Software flowchart

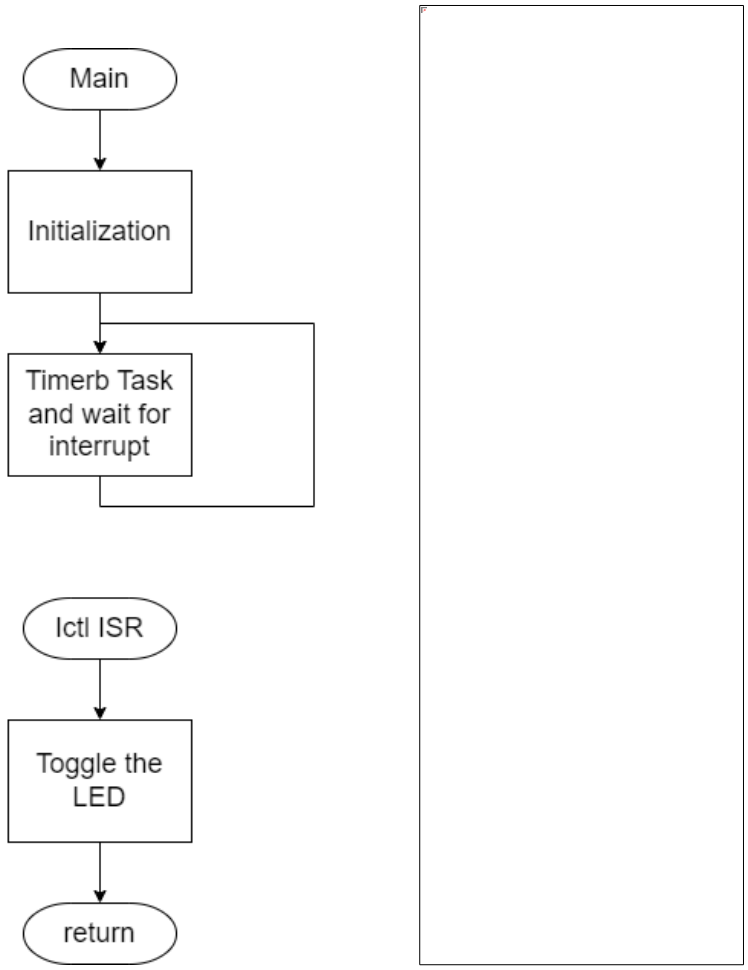


Figure 2. Application flow

## 2.5 Reference code

Configure Peripheral IO to select alternate function as TIMRB13 output

toggle port. As IO37 default used to be Jlink, so here must release the Jlink default IO37 first and reassign other IO as Jlink interface, then assign IO37 to TIMB13 output

```
static void XT_JLink_Remapping_portcfg(void)
{
  /*assign port IO29,IO30 as Jlink port swdio and swclk*/
  XT_PADI_Option_Assigned(PADI_IDX_IO29_SWDIO, PADI_CFG_IO29_SWDIO, PADI_PULLNO);
  XT_PADI_Option_Assigned(PADI_IDX_IO30_SWCLK, PADI_CFG_IO30_SWCLK, PADI_PULLNO);

  /*release default Jlink port PAD37,PAD40 and reassign as GPIO*/
  XT_PADI_Option_Assigned(PADI_IDX_IO40_PA30, PADI_CFG_IO40_PA30, PADI_PULLNO);
  XT_PADI_Option_Assigned(PADI_IDX_IO37_PA31, PADI_CFG_IO37_PA31, PADI_PULLNO);
}
```

Assign IO37 to TIMB13 output

```
      HAL_TIMR_OutputPortConfig(htim, TIMRB13_TOGGLE_PIN_IDX);
```

Configure Peripheral IO27 to select alternate function as TIMRB22 output toggle port.

```
     HAL_TIMR_OutputPortConfig(htim, TIMRB22_TOGGLE_PIN_IDX);
```

Enable interrupt code:

```
    HAL_NVIC_SetPriority(ICTL_IRQn, 2, 0);
    HAL_NVIC_EnableIRQ(ICTL_IRQn);
```

```
     HAL_TIMR_InterruptEnable(TIMB13);
```

```
    HAL_TIMR_InterruptEnable(TIMB22);
```

Configure Peripheral TIMB using HAL_TIMR_Base_Init.

```
  /* Initialize TIMB */
  htimb3.Instance = TIMB13;

  htimb3.Init.CounterMode = TIME_MODE_USERDEFINED;
  htimb3.Init.LoadCount   = TIMERB1_PERIOD_1ms;
  if(HAL_TIMR_Base_Init(&htimb3) != HAL_OK)
  {
    /* Error_Handler */
  }
```

```
  /* Initialize TIMB */
  htimb6.Instance = TIMB22;

  /* Initialize TIMx peripheral as follow:
  */
  htimb6.Init.CounterMode = TIME_MODE_USERDEFINED;
  htimb6.Init.LoadCount   = TIMERB2_PERIOD_10ms;
  if(HAL_TIMR_Base_Init(&htimb6) != HAL_OK)
  {
    /* Error_Handler */
  }
```

XT_TimerB_Task handle to start and stop timers.

```
void XT_TimerB_Task(void)
{
    /* USER CODE */
    if(XT_EVB_Button1_State_Get())
    {
```

```
        /* Start the TIM time Base generation in interrupt mode */
        if(HAL_TIMR_Base_Start_IT(&htimb3) != HAL_OK)
        {
            /* Starting Error */
            Error_Handle();
        }

        if(HAL_TIMR_Base_Start_IT(&htimb6) != HAL_OK)
        {
            /* Starting Error */
            Error_Handle();
        }

    }

    if(XT_EVB_Button2_State_Get())
    {
        /* Stop the TIM time Base generation in interrupt mode */
        if(HAL_TIMR_Base_Stop_IT(&htimb3) != HAL_OK)
        {
            /* Starting Error */
            Error_Handle();
        }
        if(HAL_TIMR_Base_Stop_IT(&htimb6) != HAL_OK)
        {
            /* Starting Error */
            Error_Handle();
        }
    }
}
```

## 2.6 Additional resources

- XT32H0xxB--reference manual