XT32H05x

XT32 microcontroller LED

Application notes

Rev 0.0.0

Original Release Date: 24-Oct-2023

Revised :

Revision History

Release	Date	Author	Summary of Change
V0.0.0	24/10/2023	Shirling Liu	Initial

Contents

1 INTRODUCE 1

1.1 REQUIRED PERIPHERALS 1

1.2 COMPATIBLE DEVICES 2

2 DESIGN DESCRIPTION 2

2.1 FEATURE OVERVIEW 2

2.2 DESIGN STEPS 3

2.3 DESIGN CONSIDERATIONS 6

2.4 SOFTWARE FLOWCHART 6

2.5 REFERENCE CODE 8

2.6 ADDITIONAL RESOURCES 13

List of Figures

Figure 1. 8x8 LED matrix example 4

Figure 2. IO function selection as LED 4

Figure 3. Application flow—DMA mode 7

List of Tables

Table 1. Modules in example.....1

Table 2. Device list.....2

1 Introduce

This application note serves as a comprehensive guide for software developers, offering essential information on LED. It covers fundamental concepts and provides guidelines to ensure proper utilization of LED 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 LED in your applications.

1.1 Required peripherals

This application involves modules as table 1.

Table 1. Modules in example

Sub-module	Peripheral use	Note
PADI	Config the IO ports as LED ports	
LED	8 com-seg ports, 1 segment port	
DMA	DMA transfer the frame data	

1.2 Compatible devices

This example is compatible with the devices in Table 2.

Table 2. Device list

Product	EVB
XT32H050	XB002823
LED device demo board	XB003123

2 Design description

2.1 Feature overview

XT32H0xxx LED peripherals is an APB component for LED panel driver, the software provides the following features:

- support max 72 leds,
    - LED com pin: up to built-in 8 com pins
    - LED segment pin: up to built-in 9 segment pins
- Brightness: up to 256 brightness steps.
- Run mode: support On-off mode and Bright mode.
- Frame buff: LED driver support max buff size is 128 words.
- Frame format: two different formats for On-off and Bright mode.
    - On-off frame format: 1 frame buffer max size is 4 words. (the buffer size = max com index/2: com index is from 1 to 8)

Byte	Description	31-25	24	23	22	21	20	19	18	17	16	15-9	8	7	6	5	4	3	2	1	0
0x00	Frame	Reserved	COM1: Seg8~seg0 switch									Reserved	COM0: Seg8~seg0 switch								
0x04		Reserved	COM3: Seg8~seg0 switch									Reserved	COM2: Seg8~seg0 switch								
0x08		Reserved	COM5: Seg8~seg0 switch									Reserved	COM4: Seg8~seg0 switch								
0x0C		Reserved	COM7: Seg8~seg0 switch									Reserved	COM6: Seg8~seg0 switch								

- Bright mode: 1 com-frame buffer allocated 5 words. 1 frame buffer size = n*(com-frame buffer); (n is the number of used com)

Byte	Description	31 -- 24	23 -- 16	15 -- 8	7 -- 0
0x00	Frame COM0	SEG1_Start bright value	SEG1_Stop bright value	**0**	**0**
0x04		SEG3_Start bright value	SEG3_Stop bright value	SEG2_ Start bright value	SEG2_ Stop bright value
0x08		SEG5_Start bright value	SEG5_Stop bright value	SEG4_ Start bright value	SEG4_ Stop bright value
0x0C		SEG7_Start bright value	SEG7_Stop bright value	SEG6_ Start bright value	SEG6_ Stop bright value
0x10		Res	Res	SEG8_ Start bright value	SEG8_ Stop bright value
.....	Frame COM2-6	...	...	...	...
0x8C	Frame COM7	SEG1_ Start bright value	SEG1_ Stop bright value	SEG0_ Start bright value	SEG0_ Stop bright value
0x90		SEG3_ Start bright value	SEG3_ Stop bright value	SEG2_ Start bright value	SEG2_ Stop bright value
0x94		SEG5_ Start bright value	SEG5_ Stop bright value	SEG4_ Start bright value	SEG4_ Stop bright value
0x98		**0**	**0**	SEG6_ Start bright value	SEG6_ Stop] bright value
0x9C		Res	Res	SEG8_ Start bright value	SEG8_ Stop bright value

2.2 Design steps

This example uses 64 LED as display panel. In this example, the led driver uses 8 coms and 9 segments to configure the led matrix as below:

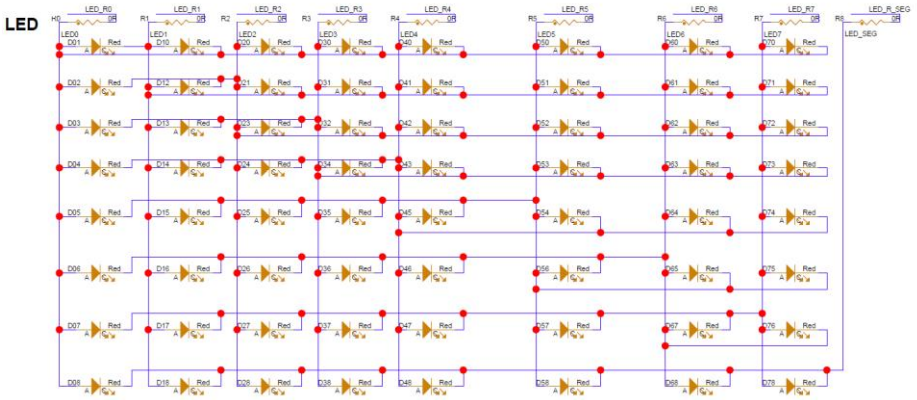

Figure 1. 8x8 LED matrix example

1. Configure pin alternate function as LED com/seg from Peripheral PADI through PADI_InitTypeDef structure.

   - PADI_IDX_IO1_LED_HIZ_0, means select and enable the IO1(pin 2).
   - PADI_CFG_IO1_LED_HIZ_0, means select LED0 function for IO1.

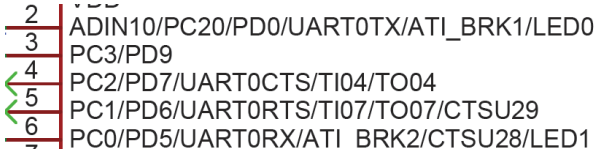

Figure 2. IO function selection as LED

Note: please refer to XT32H0xxB—reference manual document to find the assignment relationship between pin with IO.

2. Configure parameters for LED module and DMA module.
3. Assign system DMA handshaking interface to LED_FRM (LED frame DMA request), link DMA handle with LED handle, and enable

DMA1_IRQn interrupt configuration.

4. Set led running mode to bright mode, and configurate the frame buffer size and total frame number for bright mode.
5. Enable led interrupt, LED analog module and LED module.
6. Process to display the demo string or information on led panel.

2.3 Design considerations

2.4 Software flowchart

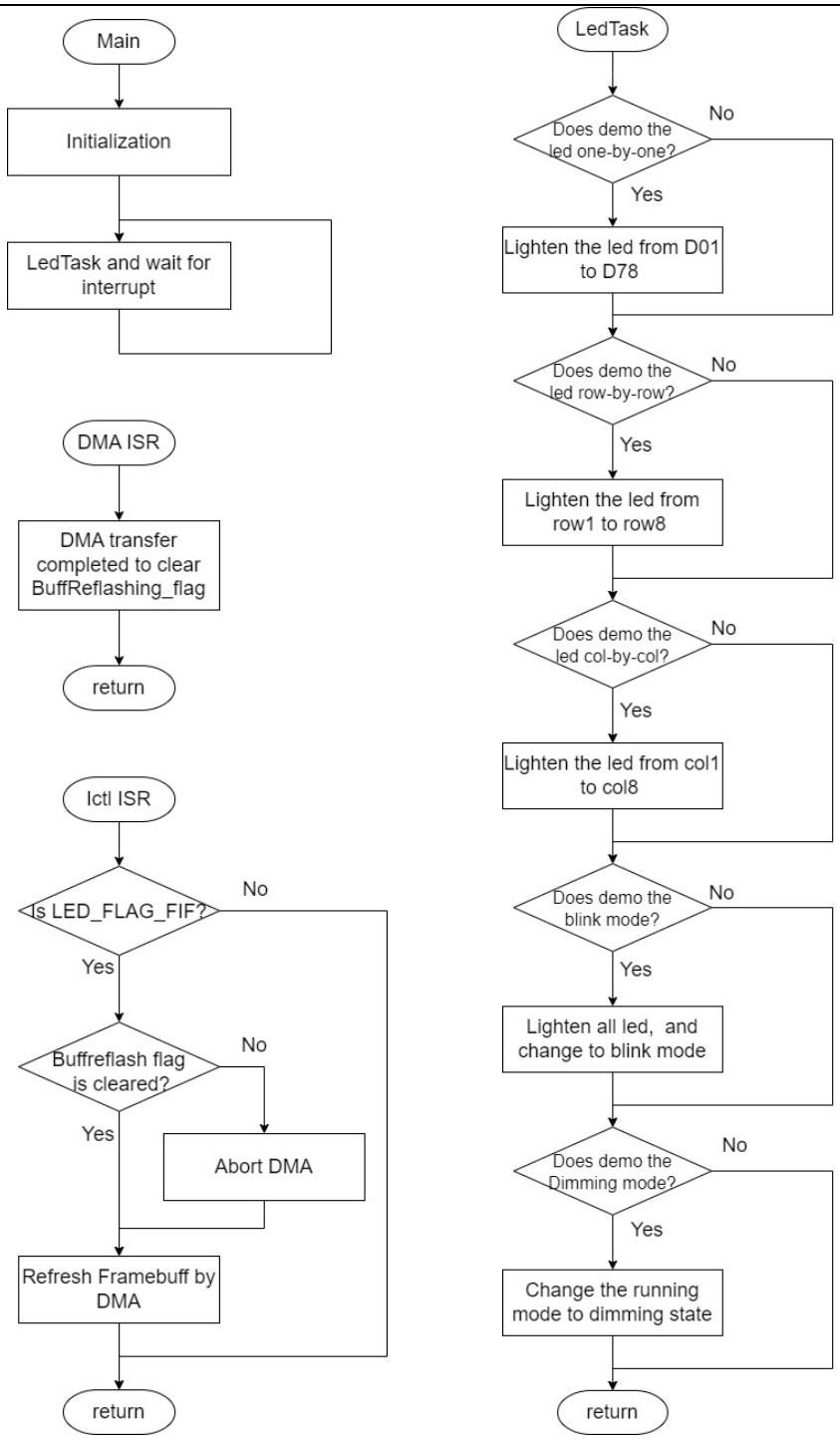

Figure 3. Application flow—DMA mode

2.5 Reference code

Configure Peripheral PADI through PADI_InitTypeDef structure to select alternate function as LED interface as bellowing code:

```
XT_IO_Option_Assigned(PADI_IDX_IO1_LED_HIZ_0,  PADI_CFG_IO1_LED_HIZ_0,  PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO5_LED_HIZ_1,  PADI_CFG_IO5_LED_HIZ_1,  PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO9_LED_HIZ_2,  PADI_CFG_IO9_LED_HIZ_2,  PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO10_LED_HIZ_3, PADI_CFG_IO10_LED_HIZ_3, PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO13_LED_HIZ_4, PADI_CFG_IO13_LED_HIZ_4, PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO14_LED_HIZ_5, PADI_CFG_IO14_LED_HIZ_5, PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO15_LED_HIZ_6, PADI_CFG_IO15_LED_HIZ_6, PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO17_LED_HIZ_7, PADI_CFG_IO17_LED_HIZ_7, PADI_PULLNO);
XT_IO_Option_Assigned(PADI_IDX_IO25_LED_SEG_HIZ,PADI_CFG_IO25_LED_SEG_HIZ,PADI_PULLNO);
```

If DMA mode to transfer data, should assign system DMA handshaking interface and enable LED frame DMA request as below code:

```
 //DMA ModeDMA Handshaking Interface:
 HAL_LED_DMAHSIFConfig(hled, &hLedDmatx,LED_FRM_DMAHSIF_IDX,LED_FRM_DMAHSIF_CFG);
//HAL_LED_DMAHSIFConfig(hled,&hLedDmatx,LED_UDT_DMAHSIF_IDX,LED_UDT_DMAHSIF_CFG);
 HAL_LED_LinkDMA(hled, &hLedDmatx);
 __HAL_LED_DMAREQ_ENABLE(hled, LED_DMAREQ_FDE);
```

Configure Peripheral LED, and enable LED :

```
ledInit.ComEnable = LED_ENABLE_COM0 | LED_ENABLE_COM1 | LED_ENABLE_COM2 | LED_ENABLE_COM3|
                    LED_ENABLE_COM4 | LED_ENABLE_COM5 | LED_ENABLE_COM6 | LED_ENABLE_COM7;
ledInit.SegEnable = LED_ENABLE_SEG0 | LED_ENABLE_SEG1 | LED_ENABLE_SEG2 | LED_ENABLE_SEG3|
       LED_ENABLE_SEG4 | LED_ENABLE_SEG5 | LED_ENABLE_SEG6 | LED_ENABLE_SEG7|LED_ENABLE_SEG8;
ledInit.Scan_Feq     = 60;
ledInit.Display_Mode = LED_RMODE_BRIGHT;//LED_RMODE_ONOFF;  //
ledInit.Dead_Time    = 2;
```

```
ledInit.Current_Bias = 4;
ledInit.Bright_Step  = 8;
ledInit.SegBright_Max[0] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[1] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[2] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[3] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[4] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[5] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[6] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[7] = 0x1f;                /*!< Segment brightness maximum */
ledInit.SegBright_Max[8] = 0x1f;                /*!< Segment brightness maximum */

hled1.Instance = LED;
hled1.Init     = ledInit;
hled1.hdma     = NULL;
HAL_LED_Init(&hled1);
HAL_LED_Off(&hled1);
if(ledInit.Display_Mode == LED_RMODE_BRIGHT)
    XT_LED_Rmode_SetBrightMode();
else
    XT_LED_Rmode_SetOnOFFMode();

__HAL_LED_CLEAR_ALLFLAGS(&hled1);
__HAL_LED_INTERRUPT_ENABLE(&hled1, LED_CTRL_FIE);

HAL_LED_AnaOn(&hled1);
HAL_LED_On(&hled1);
```

XT_Led_Task handles the demonstration to lighten 64 led.

```
void XT_Led_Task(void)
{
    uint8_t i=0, j=0;

    if(u4DemoPattern_idx == LED_DEMO_SCANONE)
```

```
    {
        for(j=0; j<8; j++)
        {
            for(i=LED_COM1; i<= LED_COM8; i++ )
            {
                XT_LED_ScreenAllFrames_Clear();
                XT_LED_DisplayLed(i, 0x1<<j, 30, LED_TEXT_MODE_REWRITE);
                XT_LED_DisplayRefresh();
                HAL_Delay(300);
            }
        }
    }
    else if(u4DemoPattern_idx == LED_DEMO_SCANROW)
    {
        XT_LED_ScreenAllFrames_Clear();
        for(j=0; j<8; j++)
        {
            for(i=LED_COM1; i<=LED_COM8; i++ )
            { XT_LED_DisplayLed(i, 0x1<<j, 30, LED_TEXT_MODE_REWRITE);    }
            XT_LED_DisplayRefresh();
            HAL_Delay(300);
        }
    }
    else if(u4DemoPattern_idx == LED_DEMO_SCANCOL)
    {
        for(i=LED_COM1; i<=LED_COM8; i++ )
        {
            XT_LED_ScreenAllFrames_Clear();
            XT_LED_DisplayLed(i, LED_D1|LED_D2|LED_D3|LED_D4|LED_D5|LED_D6|LED_D7|LED_D8, 60,
LED_TEXT_MODE_REWRITE);
            XT_LED_DisplayRefresh();
            HAL_Delay(300);
        }
    }
    else if(u4DemoPattern_idx == LED_DEMO_NUMBLINK)
    {
```

```
        for(i=LED_COM1; i<=LED_COM8; i++ )
        {
            XT_LED_DisplayLed(i, LED_D1|LED_D2|LED_D3|LED_D4|LED_D5|LED_D6|LED_D7|LED_D8, 60,
LED_TEXT_MODE_REWRITE);
        }
        XT_LED_DisplayRefresh();
        HAL_LED_SetRunningMode(&hled1, LED_DMODE_DIMMING, LED_DIMMING_STEP_1, 2);

    }
    else if(u4DemoPattern_idx == LED_DEMO_DIMMING)
    {
        HAL_LED_SetRunningMode(&hled1, LED_DMODE_DIMMING, 1, 16);
    }

    u4DemoPattern_idx++;
    HAL_Delay(1000);
}
```

LED driver lighten a led:

```
static void XTMW_DrawLedPos(LED_ComTypeDef Com, uint8_t segs, uint32_t Bright, uint32_t Mode)
{
    uint16_t segpattern = aSegCom_patternlib[Com];
    uint8_t segctrl  = segs;                        /*column*/
    uint8_t i = 0;
    uint16_t mask = 0x01;

    for(i=0; i<SEGMENT_MAX; i++)
    {
        if(segpattern & (((uint32_t)0x01UL)<<i)) {
            if ((segctrl & mask)) {
                XTMW_WriteScreen(Com, i, HAL_LED_PackSegData(&hled1, Com, i, Bright));
            }
            else  if(Mode == LED_TEXT_MODE_REWRITE) {
```

```
                XTMW_WriteScreen(Com, i, HAL_LED_PackSegData(&hled1, Com, i, 0));
            }
            mask <<= 1;
        }
    }
    return;
}
```

Write the data to screen frame buffer:

```
static void XTMW_WriteScreen(LED_ComTypeDef Comid, uint32_t Seg, uint32_t Data)
{
    if (hled1.Init.Display_Mode == LED_RMODE_BRIGHT){
        uint32_t framebase = u1Scrn_UpdateFrameIdx*(u4Scrn_FrameWDataLen);
        uint32_t comoffset = HAL_LED_GetBufferComIndex(&hled1,
Comid)*BRT_COMF_WDATALEN_DEFAULT;  /* a COM is assigned 5 word buffer, it is fixed space for a
com buffer in bright mode*/
        uint32_t segoffset = Seg>>1;     /*two segments occupy a word buff*/
        uint32_t segvalue = aScrnBuf[framebase + comoffset + segoffset];
        segvalue &= ~(0xFFFF << (16*(Seg&0x01)));   /*Seg1/3/5/7:[32:16],Seg0/2/4/6/8:[16:0]*/
        segvalue |= (Data << (16*(Seg&0x01)));  /*the data should be 0 or 255*/
        aScrnBuf[framebase + comoffset + segoffset] = segvalue;
    }
    else {
        uint32_t comid = HAL_LED_GetBufferComIndex(&hled1, Comid);
        uint32_t framebase = u1Scrn_UpdateFrameIdx*(u4Scrn_FrameWDataLen);
        uint32_t comoffset = comid >> 1;       /*two coms occupy a word buff*/
        uint32_t segoffset = 1 << Seg;
        uint32_t segvalue = aScrnBuf[framebase + comoffset];
        segvalue &= ~(segoffset << (16*(comid&0x01))); /*Com1/3/5/7:[25:16],com0/2/4/6:[8:0]*/
        if(Data > 0)
        segvalue |= ((segoffset)  << (16*(comid&0x01))); /*the data should be 0 or 1*/
        aScrnBuf[framebase + comoffset] = segvalue;
    }
}
```

2.6 Additional resources

- XT32H0xxB--reference manual
- XT32H0xxB--dma-AN230800