AN1071

应用笔记

PY32C616 系列应用指南

前言

PY32C616 系列微控制器采用高性能的 32 位 ARM® Cortex®-M0+内核，宽电压工作范围的 MCU。嵌入 24Kbytes Flash 和 2Kbytes SRAM 存储器，最高工作频率 24MHz。包含多种不同封装类型多款产品。

本应用笔记将帮助用户了解 PY32C616 各个模块应用的注意事项，并快速着手开发。

表1. 适用产品

类型	产品系列
微型控制器系列	PY32C616

目录

1 PWR

- 为了提供系统稳定性一定要使能看门狗功能
- 推荐客户在Option中使能看门狗并根据实际情况软件设置看门狗溢出时间
- 一旦使能看门狗，软件无法关闭。所以在低功耗模式下，需使用LPTIM定时唤醒，对看门狗进行喂狗。(例程参考附录1)

2 GPIO引脚设计注意事项

- 所有GPIO不能有超过-0.3V的负压
- BOOT0 在任何复位产生的时候都不能为高电平(包括在被配置为普通GPIO状态后)，否则会进BOOTLOADER。
- 初始化GPIO等其他的结构体都需要赋值为0，避免初始值不固定。
- 合封引脚两个脚都设置为输出时，一个PIN脚设定为推挽输出，另一个引脚设定为开漏输出，输出1的时候，开漏PIN先输出1，然后推挽PIN再输出1;输出0的时候，推挽脚先输出0，开漏脚再输出0(例程参考附录3)

3 LED使用大电流脚注意事项

- LED在单独使用大电流脚时，需要打开LED模块时钟，置位LED_CR_EHS = 1，并且配置GPIO速度为VERY_HIGH。(例程参考附录4)

4 RCC

- E版本芯片HSI不支持分频

5 ADC硬件设计注意事项

- ADC通道电压不能高于（VCC+0.3V）(即使ADC通道未配置为AD功能),否则ADC采样异常

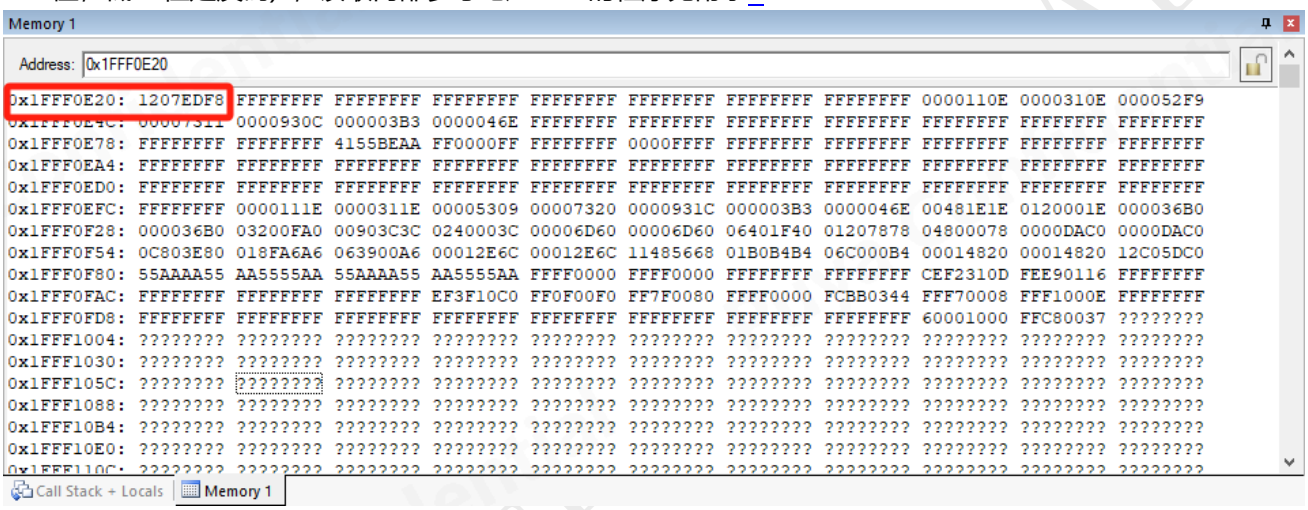
6 ADC配置

- ADC初始化前添加ADC_FORCE_RESET，确保初始化成功。
- ADC需要在使能前配置通道，若在使能后配置则会失败。
- ADC时钟需要配置到16MHz以下，确保ADC采样精度。

- ADC使能后需要增加8个ADC时钟的延时，才可以使能转换，否则会影响采样精度。
- 数码管显示会影响ADC采样结果(数码管显示的时候不采样ADC,或者在数码管的各个数据线上面串入10-100欧姆电阻，可根据实际情况进行调整)。

7 内部参考电压1.2V

- 芯片的内部参考电压1.2V实测值放置在FLASH中的information区域(0x1FFF0E20)。(高16位是实际值，低16位是反码) ，读取内部参考电压1.2V的程序见附录3:



- 在采样内部参考电压1.2V的时候，通过ADC采样时间转换公式算出来的结果要至少10us，方法如下:

  a) 降低分辨率。

  b) 降低ADC的时钟频率。

  c) 提高ADC采样周期。

  总转换时间计算如下:

  tCONV = 采样时间 + (转换分辨率+0.5) x ADC 时钟周期

  例如:

  当 ADC_CLK = 16MHz，分辨率为12位，且采样时间为 3.5个ADC 时钟周期:

  tCONV = (3.5 + 12.5) x ADC 时钟周期= 16 x ADC 时钟周期 = 1 μs

8 I2C PF0,PF1作为I2C引脚使用流程

- I2C 在初始化引脚 PF0、PF1 做 SCL、SDA 后，BUSY 位状态位受 IO 口影响置 1，影响 I2C 使用。软件必须在 IO 口初始化后复位一次 I2C 模块，使 BUSY 位清零。

9 I2C从机通讯

- I2C从机在发送一帧数据后，主机重新发地址后buffer指针会加1，所以从机需在地址中断中重新初始化buffer指针。
- 在I2C从机接收到每一个字节都需要时钟延长时，I2C主机发地址到从机的前两个字节无法时钟延长。

10 COMP

- 使用比较器2需要同时使能比较器1的SCALER_EN。

11 SPI发送和接收(库已经规避此问题，寄存器操作时需要注意此项)

- SPI作为主机接收一串数据会多一个字节，软件需要丢弃第一个字节。
- 使用SPI主机发送时不推荐使用硬件片选，推荐使用软件片选。
- SPI做主机发送时每个字节前会多发一个0x00,需要对DR寄存器做一下强制转换*（(__IO uint8_t *)&SPI1->DR)）= byte，可避免这个问题。
- SPI作为主机直接写DR寄存器发送数据的时候，需要在写DR后面添加四个NOP();确保发送成功。

12 SPI TFT屏应用

- 建议SPI使用单工模式，TX使用polling模式，RX使用DMA模式。

13 IAP升级

- APP程序必须修改VECT_TAB_OFFSET，例如#define VECT_TAB_OFFSET 0x1000。

14 LED使用注意事项

- 在单独使用大电流脚时，需要打开LED模块时钟，置位LED_CR_EHS = 1，并且配置GPIO速度为VERY_HIGH

15 OPTION操作

- 量产时，option操作需要在烧写器选项字节中配置，并把程序中操作option的函数屏蔽

16 版本历史

版本	日期	更新记录
V1.0	2024.03.28	初版
V1.1	2024.06.06	修改1、6、11章节内容，新增4章节

Puya Semiconductor Co., Ltd.

附录1

1.1 PY32C616低功耗模式下，定时唤醒喂狗例程(LL库)

```
int main(void)
{
  /* Configure system clock */
  APP_SystemClockConfig();

  /* Enable LPTIM and PWR clock */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_LPTIM1);
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);

  /* Initialize LED and button */
  BSP_LED_Init(LED3);
  BSP_PB_Init(BUTTON_USER,BUTTON_MODE_GPIO);

  /* Configure LPTIM clock source as LSI */
  APP_LPTIMClockconf();
  APP_IwdgConfig();
  /* Configure and enable LPTIM */
  APP_ConfigLPTIMOneShot();

  /* Turn on LED */
  BSP_LED_On(LED3);

  /* Wait for button press */
  while(BSP_PB_GetState(BUTTON_USER) != 0)
  {}

  /* Turn off LED */
  BSP_LED_Off(LED3);

  while (1)
  {
    /* Enable low power run mode */
    LL_PWR_EnableLowPowerRunMode();
    /* Disable LPTIM */
    LL_LPTIM_Disable(LPTIM1);
    APP_uDelay(120);                          //必须在此处增加120us以上延迟
    /* Enable LPTIM */
    LL_LPTIM_Enable(LPTIM1);
    /* Set autoreload value */
    LL_LPTIM_SetAutoReload(LPTIM, 51);
    /* Start LPTIM in one-shot mode */
    LL_LPTIM_StartCounter(LPTIM1,LL_LPTIM_OPERATING_MODE_ONESHOT);

    /* Set SLEEPDEEP bit of Cortex System Control Register */
    LL_LPM_EnableDeepSleep();

    /* Request Wait For Interrupt */
    __WFI();
    LL_IWDG_ReloadCounter(IWDG);
    LL_GPIO_TogglePin(GPIOA,LL_GPIO_PIN_3);
  }
}
void APP_IwdgConfig(void)
{
  /* Enable LSI */
```

```
    LL_RCC_LSI_Enable();
    while (LL_RCC_LSI_IsReady() == 0U) {;}

    /* Enable IWDG */
    LL_IWDG_Enable(IWDG);

    /* Enable write access */
    LL_IWDG_EnableWriteAccess(IWDG);

    /* Set IWDG prescaler */
    LL_IWDG_SetPrescaler(IWDG, LL_IWDG_PRESCALER_32); /* T=1MS */

    /* Set watchdog reload counter */
    LL_IWDG_SetReloadCounter(IWDG, 1000); /* 1ms*1000=1s */

    /* IWDG initialization */
    while (LL_IWDG_IsReady(IWDG) == 0U) {;}

    /* Feed watchdog */
    LL_IWDG_ReloadCounter(IWDG);
}
/**
    * @brief    LPTIM clock configuration
    * @param    None
    * @retval None
    */
static void APP_LPTIMClockconf(void)
{
    /* Enable LSI */
    LL_RCC_LSI_Enable();

    /* Wait for LSI to be ready */
    while(LL_RCC_LSI_IsReady() == 0)
    {}

    /* Configure LSI as LPTIM clock source */
    LL_RCC_SetLPTIMClockSource(LL_RCC_LPTIM1_CLKSOURCE_LSI);
}

/**
    * @brief    Configure LPTIM in one-shot mode
    * @param    None
    * @retval None
    */
static void APP_ConfigLPTIMOneShot(void)
{
    /* Configure LPTIM */
    /* LPTIM prescaler: divide by 128 */
    LL_LPTIM_SetPrescaler(LPTIM1,LL_LPTIM_PRESCALER_DIV128);

    /* Update ARR at the end of LPTIM counting period */
    LL_LPTIM_SetUpdateMode(LPTIM1,LL_LPTIM_UPDATE_MODE_ENDOFPERIOD);

    /* Enable ARR interrupt */
    LL_LPTIM_EnableIT_ARRM(LPTIM1);

    /* Enable NVIC interrupt request */
    NVIC_EnableIRQ(LPTIM1_IRQn);
    NVIC_SetPriority(LPTIM1_IRQn,0);
```

附录1

```
    /* Enable LPTIM */
    LL_LPTIM_Enable(LPTIM1);

    /* Configure auto-reload value: 51 */
    LL_LPTIM_SetAutoReload(LPTIM1,51);
}
void APP_LPTIMCallback(void)
{
    /*Toggle LED*/
    BSP_LED_Toggle(LED3);
}
static void APP_uDelay(uint32_t Delay)
{
    uint32_t temp;
    SysTick->LOAD=Delay*(SystemCoreClock/1000000);
    SysTick->VAL=0x00;
    SysTick->CTRL|=SysTick_CTRL_ENABLE_Msk;
    do
    {
       temp=SysTick->CTRL;
    }
    while((temp&0x01)&&!(temp&(1<<16)));
    SysTick->CTRL=0x00;
    SysTick->VAL =0x00;
}
static void APP_SystemClockConfig(void)
{
    /* Enable HSI */
    LL_RCC_HSI_Enable();
    while(LL_RCC_HSI_IsReady() != 1)
    {
    }

    /* Set AHB prescaler */
    LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);

    /* Configure HSISYS as system clock source */
    LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_HSISYS);
    while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_HSISYS)
    {
    }

    /* Set APB1 prescaler */
    LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_1);
    LL_Init1msTick(8000000);

    /* Update system clock global variable SystemCoreClock (can also be updated by calling
SystemCoreClockUpdate function) */
    LL_SetSystemCoreClock(8000000);
}
void APP_ErrorHandler(void)
{
    /* Infinite loop */
    while(1)
    {
    }
}
```

1.2 PY32C616低功耗模式下，定时唤醒喂狗例程(HAL库)

```
int main(void)
{
  /* Reset of all peripherals, Initializes the Systick */
  HAL_Init();

  /* Clock configuration */
  APP_RCCOscConfig();

  /* Initialize LED */
  BSP_LED_Init(LED3);

  /* Initialize button */
  BSP_PB_Init(BUTTON_USER, BUTTON_MODE_GPIO);

  /* LPTIM initialization */
  APP_LPTIMInit();

  /* IWDG initialization */
  APP_IWDGInit();

  /* Enable PWR */
  __HAL_RCC_PWR_CLK_ENABLE();

  /* Turn on LED */
  BSP_LED_On(LED_GREEN);

  /* Wait for button press */
  while (BSP_PB_GetState(BUTTON_USER) != 0)
  {
  }

  /* Turn off LED */
  BSP_LED_Off(LED_GREEN);

  while (1)
  {
    /* Disable LPTIM */
    __HAL_LPTIM_DISABLE(&LPTIMCONF);

    /* Enable LPTIM and interrupt, and start in single count mode */
    APP_LPTIMStart();

     /* Suspend Systick interrupt */
    HAL_SuspendTick();
    /* Enter STOP mode with interrupt wakeup */

    HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON,PWR_STOPENTRY_WFI
    );
     /* Resume Systick interrupt */
         HAL_ResumeTick();
     if (HAL_IWDG_Refresh(&IwdgHandle) != HAL_OK)
     {
       Error_Handler();
     }
          /* LED Toggle */
     BSP_LED_Toggle(LED_GREEN);
  }
}
```

附录1

```
static void APP_RCCOscConfig(void)
{
  RCC_OscInitTypeDef OSCINIT;
  RCC_PeriphCLKInitTypeDef LPTIM_RCC;

  /* LSI clock configuration */
  OSCINIT.OscillatorType = RCC_OSCILLATORTYPE_LSI;   /* Set the oscillator type to LSI */
  OSCINIT.LSIState = RCC_LSI_ON;                     /* Enable LSI */
  /* Clock initialization */
  if (HAL_RCC_OscConfig(&OSCINIT) != HAL_OK)
  {
    Error_Handler();
  }

  /* LPTIM clock configuration */
  LPTIM_RCC.PeriphClockSelection = RCC_PERIPHCLK_LPTIM;        /* Select peripheral clock: LPTIM */
  LPTIM_RCC.LptimClockSelection = RCC_LPTIMCLKSOURCE_LSI;      /* Select LPTIM clock source: LSI */
  /* Peripheral clock initialization */
  if (HAL_RCCEx_PeriphCLKConfig(&LPTIM_RCC) != HAL_OK)
  {
    Error_Handler();
  }

  /* Enable LPTIM clock */
  __HAL_RCC_LPTIM_CLK_ENABLE();
}
static void APP_LPTIMInit(void)
{
  /* LPTIM configuration */
  LPTIMCONF.Instance = LPTIM;                                  /* LPTIM */
  LPTIMCONF.Init.Prescaler = LPTIM_PRESCALER_DIV128;   /* Prescaler: 128 */
  LPTIMCONF.Init.UpdateMode = LPTIM_UPDATE_IMMEDIATE; /* Immediate update mode */
  /* Initialize LPTIM */
  if (HAL_LPTIM_Init(&LPTIMCONF) != HAL_OK)
  {
    Error_Handler();
  }
}

static void APP_LPTIMStart(void)
{
  /* Enable autoreload interrupt */
  __HAL_LPTIM_ENABLE_IT(&LPTIMCONF, LPTIM_IT_ARRM);

  __HAL_LPTIM_DISABLE(&LPTIMCONF);
  /* Delay 120us */
  APP_delay_us(120);                                  //必须在此处增加120us以上延迟

  /* Enable LPTIM */
  __HAL_LPTIM_ENABLE(&LPTIMCONF);

  /* Load autoreload value */
  __HAL_LPTIM_AUTORELOAD_SET(&LPTIMCONF, 51);

  /* Start single count mode */
  __HAL_LPTIM_START_SINGLE(&LPTIMCONF);
}
```

附录1

```
  static void APP_IWDGInit(void)
  {
    IwdgHandle.Instance = IWDG;                              /* Select IWDG */
    IwdgHandle.Init.Prescaler = IWDG_PRESCALER_32;    /* Configure prescaler to 32 */
    IwdgHandle.Init.Reload = (1000);                         /* Set IWDG counter reload value to 1000, 1s
*/
    /* Initialize IWDG */
    if (HAL_IWDG_Init(&IwdgHandle) != HAL_OK)
    {
      APP_ErrorHandler();
    }
  }

  static void APP_delay_us(int us)
 {
      unsigned t1, t2, count, delta, sysclk; sysclk = 24 ; //Modify this according to the system clock

      t1 = SysTick->VAL;
      while(1)
      {
          t2 = SysTick->VAL;
          delta = t2<t1?(t1-t2):(SysTick->LOAD - t2 + t1);
          if(delta >= us * sysclk)
          break;
      }
 }

 void Error_Handler(void)
{
  /* 无限循环 */
  while (1)
  {
  }
}
```

附录2

PY32C616读取information区域中存放的内部参考电压1.2V实测值(具体地址见5)

```
#define HAL_VREF_INT                    (*(uint8_t *)(0x1fff0E23))
#define HAL_VREF_DEC                    (*(uint8_t *)(0x1fff0E22))
#define vref_int        (*(uint8_t *)(HAL_VREF_INT))        //存放参考电压整数部分
#define vref_dec        (*(uint8_t *)(HAL_VREF_DEC))        //存放参考电压小数部分
float vref;                 //参考电压值

static uint8_t Bcd2ToByte(uint8_t Value)
{
    uint32_t tmp = 0U;
    tmp = ((uint8_t)(Value & (uint8_t)0xF0) >> (uint8_t)0x4) * 10U;
    return (tmp + (Value & (uint8_t)0x0F));
}

float read_1_2V(void)
{
  uint8_t data_vref_int,data_vref_dec;
  data_vref_int = Bcd2ToByte(HAL_VREF_INT);
  data_vref_dec = Bcd2ToByte(HAL_VREF_DEC);

  //初始化所有外设，flash接口，systick
    vref = data_vref_int/10;      //计算参考电压
    vref = vref + ((data_vref_int%10)*0.1 + data_vref_dec*0.001);
    return vref;
}
```

附录3

输出1的时候，开漏PIN先输出1，然后推挽PIN再输出1;输出0的时候，推挽脚先输出0，开漏脚再输出0

```
int main(void)
{
  /* Reset of all peripherals, Initializes the Systick */
  HAL_Init();

  /* Initialize GPIO */
  APP_GpioConfig();

  HAL_GPIO_WritePin(GPIOA,GPIO_PIN_3,1);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,1);

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,0);
  HAL_GPIO_WritePin(GPIOA,GPIO_PIN_3,0);

}

/**
  * @brief  GPIO configuration
  * @param  None
  * @retval None
  */
static void APP_GpioConfig(void)
{
  GPIO_InitTypeDef  GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();                    /* Enable GPIOA、GPIOB
clock */
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_3;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;        /* Push-pull output */
  GPIO_InitStruct.Pull = GPIO_NOPULL;              /* Enable pull-up */
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;      /* GPIO speed */
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = GPIO_PIN_3;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;        /* Push-pull output */
  GPIO_InitStruct.Pull = GPIO_NOPULL;              /* Enable pull-up */
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;      /* GPIO speed */

  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

附录4

LED在单独使用大电流脚时，需要打开LED模块时钟，置位LED_CR_EHS = 1，并且配置GPIO速度为 VERY_HIGH。

```
int main(void)
{
  /* Reset of all peripherals, Initializes the Systick */
  HAL_Init();
  SET_BIT(RCC->APBENR2,RCC_APBENR2_LEDEN);
  SET_BIT(LED->CR,LED_CR_EHS);
  APP_GpioConfig();
}

/**
  * @brief  GPIO configuration
  * @param  None
  * @retval None
  */
static void APP_GpioConfig(void)
{
  GPIO_InitTypeDef  GPIO_InitStruct = {0};

  __HAL_RCC_GPIOB_CLK_ENABLE();                          /* Enable GPIOB clock */

  GPIO_InitStruct.Pin = GPIO_PIN_3;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;            /* Push-pull output */
  GPIO_InitStruct.Pull = GPIO_NOPULL;                    /* Enable pull-up */
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;     /* GPIO speed */
  /* GPIO Initialization */
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```