AN1111E

Technology Innovator
Puya

Application Note

PY32F040-EP Application Note

Introduction

The series of PY32F040-EP microcontrollers features a high-performance 32-bit ARM® Cortex®-M0+ core, a wide voltage range MCU. They are embedded with up to 128 Kbytes of flash memory and 16 Kbytes of SRAM memory, operating at a maximum frequency of 72 MHz. A variety of products with different packaging types are included.

This document will help users understand the attentions for the application of various modules in PY32F040-EP and quickly start development.

Table 1 Applicable Products

Type	Product Series
Microcontroller Series	PY32F040-EP

PUYA CONFIDENTIAL

Contents

1. PWR Configuration 3
2. ADC Configuration 3
3. COMP Configuration 4
4. CRC Configuration 4
5. DMA Configuration 5
6. I2C Configuration 5
7. LCD Configuration 5
8. LPTIM Configuration 6
9. RCC Configuration 6
10. SPI Configuration 7
11. TIMER Configuration 7
12. FLASH Configuration 7
13. OPTION Configuration 7
14. Version History 10
Appendix 1 11
1.1 The routine of using timed wake-up to feed dog in the low power mode of PY32F040-EP (LL Library) 11
1.2 The routine of using timed wake-up to feed dog in the low power mode of PY32F040-EP (HAL Library) 16
Appendix 2 19
2.PY32F040-EP reads the Vreferint 1.2V actual value stored in the information area (see 2.3 for specific address). 19

PUYA CONFIDENTIAL

1. PWR Configuration

- In order to improve the stability of the system, the watchdog function must be enabled.
- It is recommended to enable the watchdog in the Option and set the watchdog overflow time according to actual conditions through software.
- Once the watchdog is enabled, it cannot be turned off by software. Therefore, in low power consumption modes, an LPTIM timer should be used to wake up and feed the watchdog. (refer to example routines in Appendix 1)
- When using event wakeup in Sleep mode, if the EXTI module clock and the CPU clock are one clock source, the divider frequency must not be used.

2. ADC Configuration

2.1 ADC Software Configuration

- Since the ADC only sets an EOC (End of Conversion) flag after all channels have completed conversion, it is not possible to use non-DMA multi-channel sampling. (You can set non-continuous mode to enable, and an EOC flag will be generated after each channel conversion.)
- Software should judge EOC pulled low before starting Start to avoid EOC and Start signals occurring simultaneously.
- ADC injected mode cannot be used.
- When the system clock is 8M and the ADC CLOCK is divided by two, the ADC cannot be calibrated.
- When configuring the sample time of channel 16 (OPA3_IN), channel 0 and channel 16 need to be configured as the same sample time.
- When using TIMER_CC/TIMER_TRGO to trigger ADC conversion, the ADC clock must not be divided by 8.

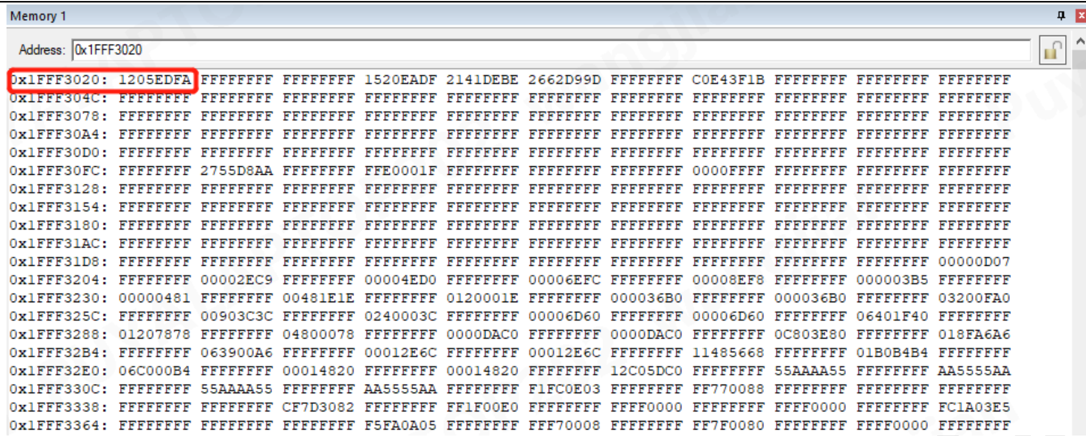
2.2 ADC Hardware Configuration

- The voltage of the ADC channel must not exceed VCC+0.3V (even if the ADC channel is not configured as an AD function), otherwise the ADC sampling will be inaccurate.

2.3 Vreferint 1.2V

- The actual value of the Vreferint 1.2V which the chip uses is plasced in the information area (0x1FFF3020) in FLASH. (The high 16 bits are the actual value and the low 16 bits are the inverse code.) The procedure for reading Vreferint 1.2V is shown in Appendix 2:

PUYA CONFIDENTIAL



- When sampling the internal reference voltage of 1.2V, the result calculated through the ADC sampling time conversion formula should be at least 20 microseconds. The methods are as follows:

a) Reduce the resolution;

b) Decrease the ADC clock frequency;

c) Increase the ADC sampling period.

The total conversion time is calculated as follows:

tCONV = Sampling Time 采样时间 + (Conversion Resolution+0.5) x ADC Clock Cycle

For example:

When ADC_CLK = 12MHz，the resolution is 12 bits, and the sampling time is 239.5 ADC clock cycles:

tCONV = （239.5 + 12.5）x ADC Clock Cycle=252 x ADC Clock Cycle= 21 us

3. COMP Configuration

- When the COMPx_INM input signal to the comparator is an internal analogue voltage source (e.g. VREFINT, TSVIN, VREF1P2), the external input channel COMPx_INP requires an addition of a capacitor (1nF) to ground.

4. CRC Configuration

- DMA mode is prohibited.
- Continuous operations on the CRC_DR register are prohibited, it is recommended to use the library, which contains mitigations.

PUYA CONFIDENTIAL

5. DMA Configuration

- When transferring data using DMA, you must wait for the DMA transfer to complete before you can disable DMA; otherwise, you will need to perform a forced Reset and reinitialize DMA to use it properly.

6. I2C Configuration

- When using DMA for I2C data transfer, you need to configure the DMA source address and destination address before enabling the I2C's DMA_EN.

7. LCD Configuration

- When writing data to the same LCD_RAMx register, it is necessary to write the data within 2 lcd clk cycles, and then wait for 2 pclk + 1 lcd clk cycles before continuing to write the data. (refer to the following)

```
   #define   Delay   40*2
   LCD_HandleTypeDef LcdHandle;

int main()
{
   RatioNops = Delay * (SystemCoreClock / 1000000U) / 4;
   HAL_LCD_Write(&LcdHandle,  LCD_RAM_REGISTER0,  0x0f0f0f0f);
   APP_DelayNops(RatioNops);/* Delay 2 pclk + 1 lcd clk cycles, about 80us
   HAL_LCD_Write(&LcdHandle,  LCD_RAM_REGISTER0,  0xf0f0f0f0);
}

static void APP_DelayNops(uint32_t Nops)
{
   for(uint32_t i=0; i<Nops;i++)
   {
      __NOP();
   }
}
```

- After writing to RAM, you must wait for 2 LCD clock cycles before entering STOP mode. (refer to the following)

```
  #define   Delay   40*2
   LCD_HandleTypeDef LcdHandle;

int main()
{
   RatioNops = Delay * (SystemCoreClock / 1000000U) / 4;

   HAL_LCD_Write(&LcdHandle,  LCD_RAM_REGISTER0,  0x0f0f0f0f);

   APP_DelayNops(RatioNops);/* Delay 2 lcd clk cycles, about 80us
}
```

PUYA CONFIDENTIAL

```
while(1)
{
/* Suspend SysTick interrupt */
HAL_SuspendTick();
/* Enter Stop Mode and Wakeup by WFI */
HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON,
PWR_STOPENTRY_WFI);
/* Resume Systick */
HAL_ResumeTick();

}
static void APP_DelayNops(uint32_t Nops)
{
  for(uint32_t i=0; i<Nops;i++)
  {
    __NOP();
  }
}
```

8. LPTIM Configuration

- When LPTIM uses RCC_CCIPR->LPTIMSEL to select PCLK as the clock source, the pre-scaler cannot be set to 1, otherwise LPTIM has a probability of running abnormally.
- When LPTIM uses the RSTARE function, the interval between two reads of the CNT registers should satisfy 4 LSI clocks.

8.1 LPTIM Continuous Mode

- LPTIM continuous mode must clear ARRMCF each time before entering STOP and wait for 1 LSI clock cycle*PSC factor.(Approx. 40us*PSC including programme execution time.)
- Changing the LPTIM reload value requires a wait of 4 LSI clock cycles * PSC factor.(Approx. 160us*PSC including programme execution time.)

8.2 LPTIM Once Mode

- LPTIM once mode wake-up from STOP and wait 4 LSI clock cycles before entering STOP again. (Takes about 160us, including programme execution time.)
- Wait 4 LSI clock cycles when changing the reload value LPTIM_ARR. (Approx. 160us including programme execution time.)

9. RCC Configuration

- When the APB crossover coefficient is greater than 1, after reseting the module, (n+2)__nop() null instructions is required to be added before performing read and write operations on the module registers. n is the APB crossover coefficient.

PUYA CONFIDENTIAL

10. SPI Configuration

- When SPI is used as a slave transmitter, it is necessary to clear SPI_EN to 0 and then set SPI_EN to 1 before sending each frame of data.

11. TIMER Configuration

- When enabling the CC interrupt, the corresponding prescaler coefficient PSC must not be higher than 80.
- TIMER prescaler must be set to 1; otherwise, it will cause the MCU to enter interrupts repeatedly.

12. FLASH Configuration

- FLASH only supports Page erase and Page write, a Page is 256 bytes, the start address can only be Page aligned; (e.g. start address 0x08005000, 0x08005100, etc.)
- Every time Page write must be preceded by Page erase.

13. OPTION Configuration

- When mass production, Option operation needs to be configured in the Option Byte configuration of the porgammer, and the function that operates Option in the program should be blocked.
- It is recommended that the customer program enables write protection, which is set in Option, as shown in Figure 13-1 and Figure 13-2.

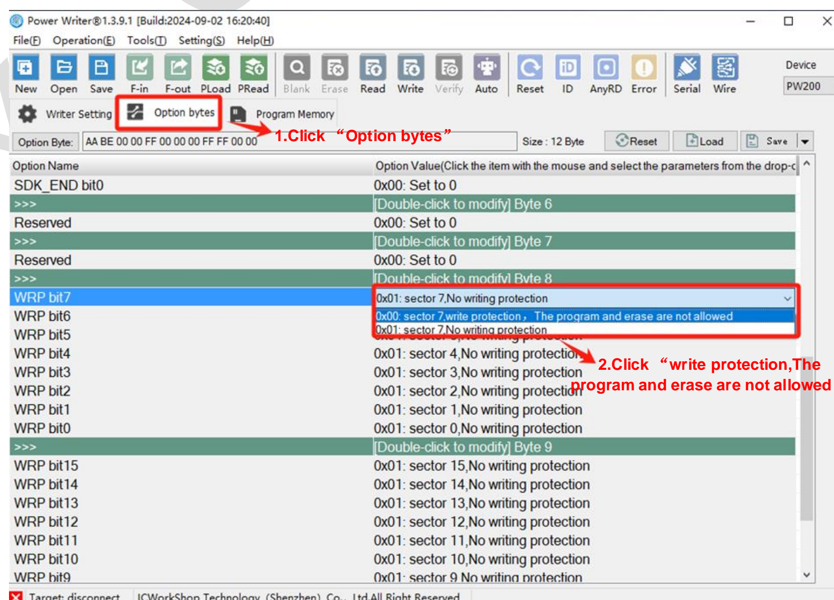


Figure 13-1 Power Write Operation Option Write Protection

PUYA CONFIDENTIAL

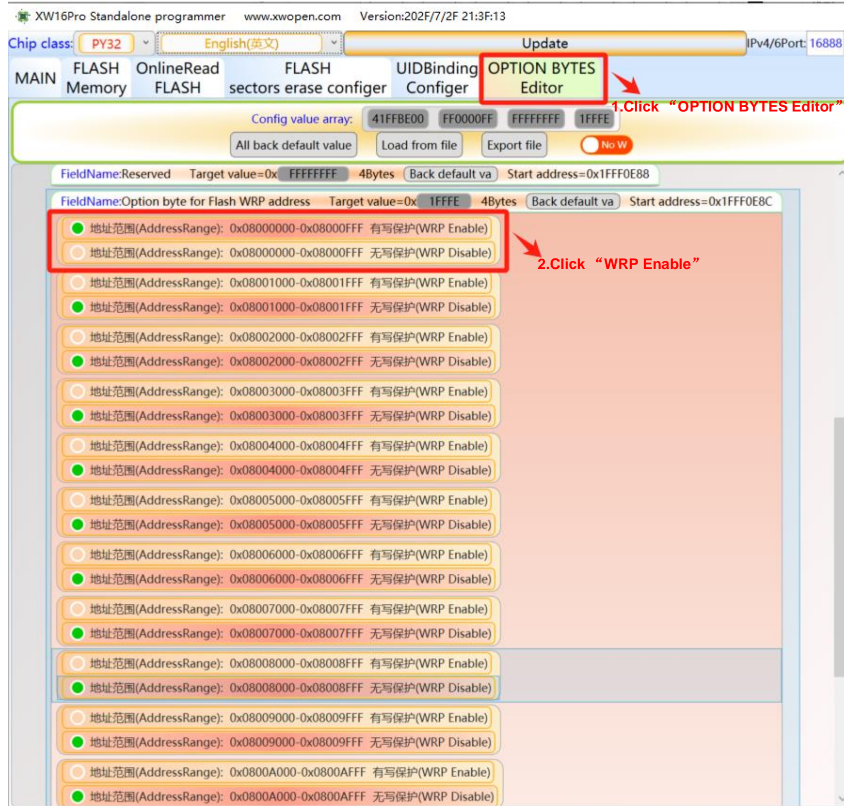


Figure 13-2 XW operation Option write protection

- When configuring the Option of the programmer, you need to check the "Smart Reset" function or "Restart the chip after programming" (programmers typically have similar options that need to be selected). The specific steps are shown in Figures 13-3, Figure 13-4.

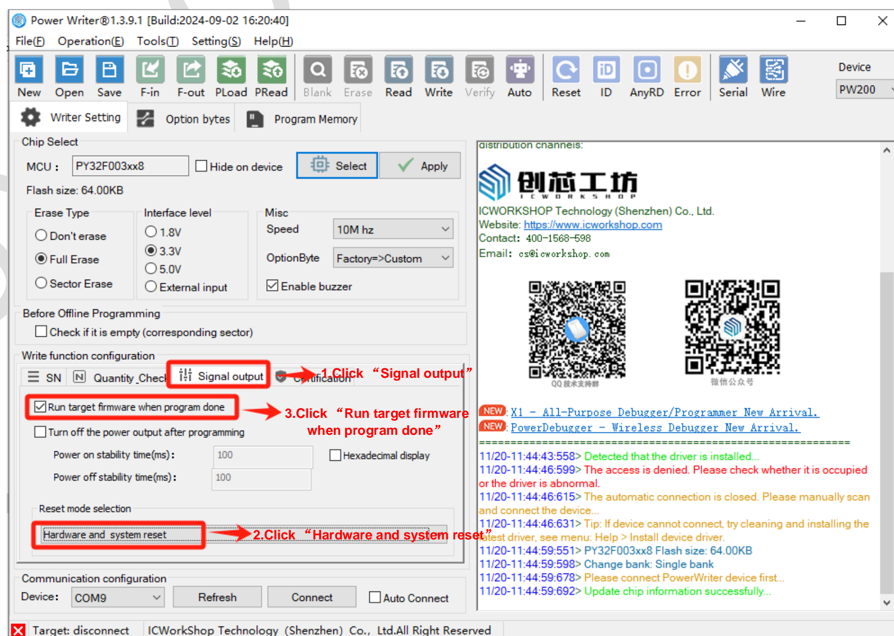


Figure 13-3 Power Write Operation Tick 'Reboot Chip After Programming"

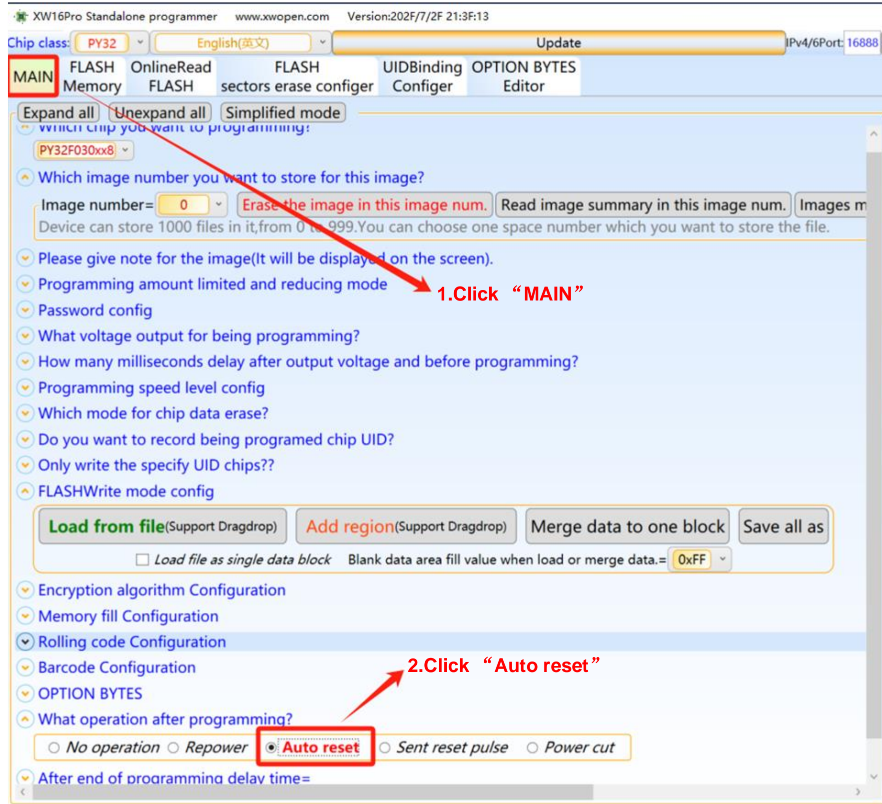


Figure 13-4 XW Operation ‘Smart Reset’

PUYA CONFIDENTIAL

14. Version History

Version	Date	Update Records
V1.0	2025.06.10	Initial release
V1.1	2025.07.28	Modify LPTIM module contents

PUYA CONFIDENTIAL

Puya Semiconductor Co., Ltd.

IMPORTANT NOTICE

Puya reserve the right to make changes, corrections, enhancements, modifications to Puya products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information of Puya products before placing orders.

Puya products are sold pursuant to terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice and use of Puya products. Puya does not provide service support and assumes no responsibility when products that are used on its own or designated third party products.

Puya hereby disclaims any license to any intellectual property rights, express or implied.

Resale of Puya products with provisions inconsistent with the information set forth herein shall void any warranty granted by Puya.

Any with Puya or Puya logo are trademarks of Puya. All other product or service names are the property of their respective owners.

The information in this document supersedes and replaces the information in the previous version.

Puya Semiconductor Co., Ltd. – All rights reserved

Appendix 1

1.1 The routine of using timed wake-up to feed dog in the low power mode of PY32F040-EP (LL Library)

```
#define Delay                    40*4
int main(void)
{
  /* Configure system clock */
  APP_SystemClockConfig();

  /* Initialize LED */
  BSP_LED_Init(LED_GREEN);

  APP_GpioConfig();
 /* Initialize button */
  BSP_PB_Init(BUTTON_KEY, BUTTON_MODE_GPIO);

  /* Configure EXTI Line29 corresponding to LPTIM as interrupt wake-up mode */
  LL_EXTI_EnableIT(LL_EXTI_LINE_29);     /* Enable EXTI Line 29 interrupt wakeup */
  LL_EXTI_DisableEvent(LL_EXTI_LINE_29); /* Disable EXTI Line 29 event wakeup */

  /* Configure LPTIM clock source as LSI */
  APP_ConfigLptimClock();
    APP_IwdgConfig();
  /* Initialize LPTIM */
  LPTIM_InitStruct.Prescaler = LL_LPTIM_PRESCALER_DIV128;          /* Prescaler: 128 */
  LPTIM_InitStruct.UpdateMode = LL_LPTIM_UPDATE_MODE_IMMEDIATE;   /* Immediate update
mode */
  if (LL_LPTIM_Init(LPTIM,   &LPTIM_InitStruct) != SUCCESS)
  {
    APP_ErrorHandler();
  }

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

  /*   Wait for the button to be pressed   */
  while (BSP_PB_GetState(BUTTON_USER) != 0)
  {
  }

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

  /* Calculate the value required for a delay of macro-defined(Delay) */
  RatioNops = Delay * (SystemCoreClock / 1000000U) / 4;

  /* Configure LPTIM and enable interrupt */
  APP_ConfigLptim();

  while (1)
  {
    /* LPTIM must be disabled to restore internal state before next time enter stop mode */
    LL_LPTIM_Disable(LPTIM);
```

PUYA CONFIDENTIAL

```
        /* Wait at least three LSI times for the completion of the disable operation */
        APP_DelayNops(RatioNops);

        /* Enable LPTIM */
        LL_LPTIM_Enable(LPTIM);

        /* Set auto-reload value */
        LL_LPTIM_SetAutoReload(LPTIM,  51);

        /* Start in once mode */
        LL_LPTIM_StartCounter(LPTIM,  LL_LPTIM_OPERATING_MODE_ONESHOT);

        /* Enable STOP mode */
        APP_EnterStop();

        /* PB1 toggle */
        LL_GPIO_TogglePin(GPIOB,  LL_GPIO_PIN_1);
    }
}

/**
  * @brief  Configure LPTIM clock
  * @param  None
  * @retval None
  */
static void APP_ConfigLptimClock(void)
{
  /* Enable LSI */
  LL_RCC_LSI_Enable();
  while(LL_RCC_LSI_IsReady() != 1)
  {
  }

  /* Select LTPIM clock source as LSI */
  LL_RCC_SetLPTIMClockSource(LL_RCC_LPTIM1_CLKSOURCE_LSI);

  /* Enable LPTIM clock */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_LPTIM1);
}

/**
  * @brief  Delayed by NOPS
  * @param  None
  * @retval None
  */
static 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 */
```

PUYA CONFIDENTIAL

```
    LL_IWDG_SetPrescaler(IWDG, LL_IWDG_PRESCALER_32);

    /* Set watchdog reload counter */
    LL_IWDG_SetReloadCounter(IWDG, 1024); /* T*1024=1s */

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

    /* Feed the watchdog */
    LL_IWDG_ReloadCounter(IWDG);
}
static void APP_DelayNops(uint32_t Nops)
{
    for(uint32_t i=0; i<Nops;i++)
    {
        __NOP();
    }
}

/**
    * @brief  GPIO configuration program
    * @param  None
    * @retval None
    */
static void APP_GpioConfig(void)
{
    /* Enable GPIOB clock */
    LL_IOP_GRP1_EnableClock(LL_IOP_GRP1_PERIPH_GPIOB);

    /* Configure PB1 in output mode */
    LL_GPIO_SetPinMode(GPIOB, LL_GPIO_PIN_1, LL_GPIO_MODE_OUTPUT);

}

/**
    * @brief  Configure LPTIM
    * @param  None
    * @retval None
    */
static void APP_ConfigLptim(void)
{
    /* Enable LPTIM1 interrupt */
    NVIC_SetPriority(TIM6_LPTIM1_DAC_IRQn, 0);
    NVIC_EnableIRQ(TIM6_LPTIM1_DAC_IRQn);

    /* Enable LPTIM ARR match interrupt */
    LL_LPTIM_EnableIT_ARRM(LPTIM);
}

/**
    * @brief  Enter STOP mode
    * @param  None
    * @retval None
    */
static void APP_EnterStop(void)
{
    /* Enable PWR clock */
    LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_PWR);
```

PUYA CONFIDENTIAL

```
    /* VCORE = 0.8V when enter stop mode */
    LL_PWR_SetRegulVoltageScaling(LL_PWR_REGU_VOLTAGE_0P8V);

    /* Enable Low Power Run mode */
    LL_PWR_EnableLowPowerRunMode();

    /* Enter DeepSleep mode */
    LL_LPM_EnableDeepSleep();

    /* Request Wait For interrupt */
    __WFI();

    LL_LPM_EnableSleep();
}

/**
  * @brief   System clock configuration function
  * @param   None
  * @retval None
  */
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);
}

/**
  * @brief   LPTIM interrupt callback program
  * @param   None
  * @retval None
  */
void APP_LptimIRQCallback(void)
{
    if((LL_LPTIM_IsActiveFlag_ARRM(LPTIM) == 1) && (LL_LPTIM_IsEnabledIT_ARRM(LPTIM) == 1))
    {
        /* Clear autoreload match flag */
        LL_LPTIM_ClearFLAG_ARRM(LPTIM);

        LL_IWDG_ReloadCounter(IWDG);
```

PUYA CONFIDENTIAL

```
        /* LED Toggle */
        BSP_LED_Toggle(LED_GREEN);
    }
}

/**
  * @brief   This function is executed in case of error occurrence.
  * @param   None
  * @retval None
  */
void APP_ErrorHandler(void)
{
  /* Infinite loop */
  while (1)
  {
  }
}
```

PUYA CONFIDENTIAL

1.2 The routine of using timed wake-up to feed dog in the low power mode of PY32F040-EP (HAL Library)

```
#define Delay                    40*4
int main(void)
{
   EXTI_ConfigTypeDef            ExtiCfg = {0};

   /* Reset of all peripherals,   Initializes the Systick */
   HAL_Init();

   /* Clock configuration */
   APP_RCCOscConfig();

   /* Initialize LED */
   BSP_LED_Init(LED_GREEN);

   APP_GpioConfig();
   /* Initialize button */
   BSP_PB_Init(BUTTON_USER,   BUTTON_MODE_GPIO);
   /* LPTIM initialization */
   LPTIMConf.Instance = LPTIM1;                              /* LPTIM1 */
   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)
   {
      APP_ErrorHandler();
   }
   /* Configure EXTI Line as interrupt wakeup mode for LPTIM */
   ExtiCfg.Line = EXTI_LINE_29;
   ExtiCfg.Mode = EXTI_MODE_INTERRUPT;
   /* The following parameters do not need to be configured */
   /* ExtiCfg.Trigger */
   /* ExtiCfg.GPIOSel */
   HAL_EXTI_SetConfigLine(&ExtiHandle,   &ExtiCfg);
   /* Enable LPTIM1 interrupt */
   HAL_NVIC_SetPriority(TIM6_LPTIM1_DAC_IRQn,   0,   0);
   HAL_NVIC_EnableIRQ(TIM6_LPTIM1_DAC_IRQn);
   APP_IwdgConfig();
   /* Turn on LED */
   BSP_LED_On(LED_GREEN);
   /*   Wait for the button to be pressed   */
   while (BSP_PB_GetState(BUTTON_USER) != 0)
   {
   }
   /* Turn off LED */
   BSP_LED_Off(LED_GREEN);
  /* Calculate the value required for a delay of macro-defined(Delay) */
   RatioNops = Delay * (SystemCoreClock / 1000000U) / 4;
   while (1)
   {
      /* LPTIM must be disabled to restore internal state before next time enter stop mode */
      __HAL_LPTIM_DISABLE(&LPTIMConf);
      /* Wait at least three LSI times for the completion of the disable operation */
      APP_DelayNops(RatioNops);
```

PUYA CONFIDENTIAL

```
        /* Configure LPTIM for once mode and enable interrupt */
        HAL_LPTIM_SetOnce_Start_IT(&LPTIMConf,  51);
        /* Suspend SysTick interrupt */
        HAL_SuspendTick();
        /* VCORE = 0.8V  when enter stop mode */
        PwrStopModeConf.LPVoltSelection       =  PWR_STOPMOD_LPR_VOLT_0P8V;
        PwrStopModeConf.FlashDelay            =  PWR_WAKEUP_HSIEN_AFTER_MR;
        PwrStopModeConf.WakeUpHsiEnableTime =  PWR_WAKEUP_FLASH_DELAY_1US;
        HAL_PWR_ConfigStopMode(&PwrStopModeConf);
        /* Enter Stop Mode and Wakeup by WFI */
        HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
        /* Resume Systick */
        HAL_ResumeTick();
        if (HAL_IWDG_Refresh(&IwdgHandle) != HAL_OK)
        {
          APP_ErrorHandler();
        }
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1);
    }
}
static void APP_IwdgConfig()
{
    IwdgHandle.Instance = IWDG;                           /* Select IWDG */
    IwdgHandle.Init.Prescaler = IWDG_PRESCALER_32;  /* Configure prescaler as 32 */
    IwdgHandle.Init.Reload = (1024);                     /* IWDG counter reload value is 1024,  1s */
    /* Initialize IWDG */
    if (HAL_IWDG_Init(&IwdgHandle) != HAL_OK)
    {
        APP_ErrorHandler();
    }
}
/**
    * @brief    LPTIM AutoReloadMatchCallback
    * @param    None
    * @retval   None
    */
void HAL_LPTIM_AutoReloadMatchCallback(LPTIM_HandleTypeDef *LPTIMConf)
{
    BSP_LED_Toggle(LED_GREEN);
}
/**
    * @brief    Clock configuration function
    * @param    None
    * @retval   None
    */
static void APP_RCCOscConfig(void)
{
    RCC_OscInitTypeDef OSCINIT = {0};
    RCC_PeriphCLKInitTypeDef LPTIM_RCC = {0};
    /* Oscillator configuration */
    OSCINIT.OscillatorType = RCC_OSCILLATORTYPE_LSI ;       /* Select oscillator LSI */
    OSCINIT.LSIState = RCC_LSI_ON;                            /* Enable LSI */
    OSCINIT.PLL.PLLState = RCC_PLL_NONE;                       /* PLL configuration unchanged */
    /*OSCINIT.PLL.PLLSource = RCC_PLLSOURCE_HSI;*/
    /*OSCINIT.PLL.PLLMUL = RCC_PLL_MUL2;*/
    /* Configure oscillator */
    if (HAL_RCC_OscConfig(&OSCINIT) != HAL_OK)
    {
```

PUYA CONFIDENTIAL

```
    APP_ErrorHandler();
  }
  /* 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)
  {
    APP_ErrorHandler();
  }
  /* Enable LPTIM clock */
  __HAL_RCC_LPTIM_CLK_ENABLE();
}
/**
  * @brief  Configure GPIO
  * @param  None
  * @retval None
  */
static void APP_GpioConfig(void)
{
  /* Configuration pins */
  GPIO_InitTypeDef  GPIO_InitStruct = {0};
  __HAL_RCC_GPIOB_CLK_ENABLE();                       /* Enable the GPIO clock*/
  GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;     /* GPIO mode is OutputPP */
  GPIO_InitStruct.Pull  = GPIO_PULLUP;                 /* pull up */
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;   /* The speed is high */
  GPIO_InitStruct.Pin = GPIO_PIN_1;
  HAL_GPIO_Init(GPIOB,  &GPIO_InitStruct);
}
/**
  * @brief  Delayed by NOPS
  * @param  None
  * @retval None
  */
static void APP_DelayNops(uint32_t Nops)
{
  for(uint32_t i=0; i<Nops;i++)
  {
    __NOP();
  }
}
/**
  * @brief   This function is executed in case of error occurrence.
  * @param   None
  * @retval  None
  */
void APP_ErrorHandler(void)
{
  /* Infinite loop */
  while (1)
  {
  }
}
```

PUYA CONFIDENTIAL

Appendix 2

2.PY32F040-EP reads the Vreferint 1.2V actual value stored in the information area (see 2.3 for specific address).

```
#define HAL_VREF_INT                        (*(uint8_t *)(0x1fff3023))
#define HAL_VREF_DEC                        (*(uint8_t *)(0x1fff3022))
#define vref_int          (*(uint8_t *)(HAL_VREF_INT))              //Store the integer part of the
reference voltage
#define vref_dec          (*(uint8_t *)(HAL_VREF_DEC))          // Store the fractional part of the
reference voltage
float vref;                //Reference voltage value

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);


  //Initialize all peripherals, flash interface， systick
    vref = data_vref_int/10;       //Calculate the reference voltage
    vref = vref + ((data_vref_int%10)*0.1 + data_vref_dec*0.001);
    return vref;
}
```

PUYA CONFIDENTIAL