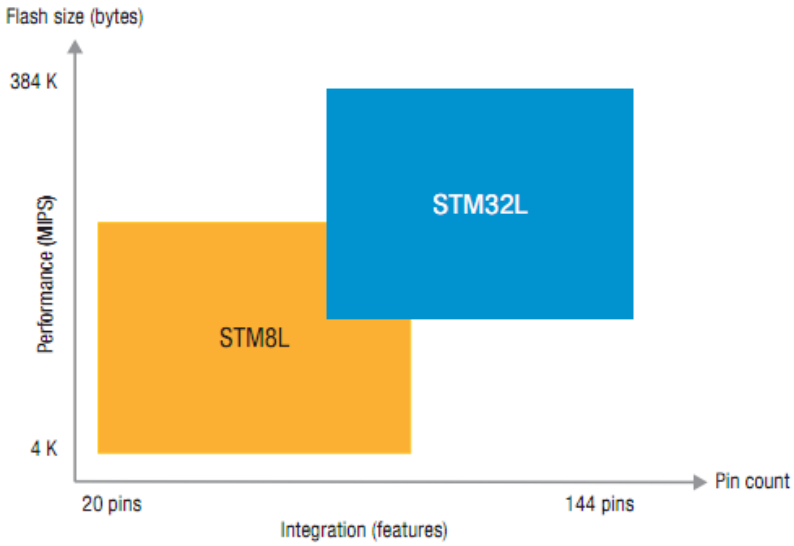
基于 STM8L152-Discovery 平台功耗测试

1 STM8L 系列

8/32-bit ultra-low-power range

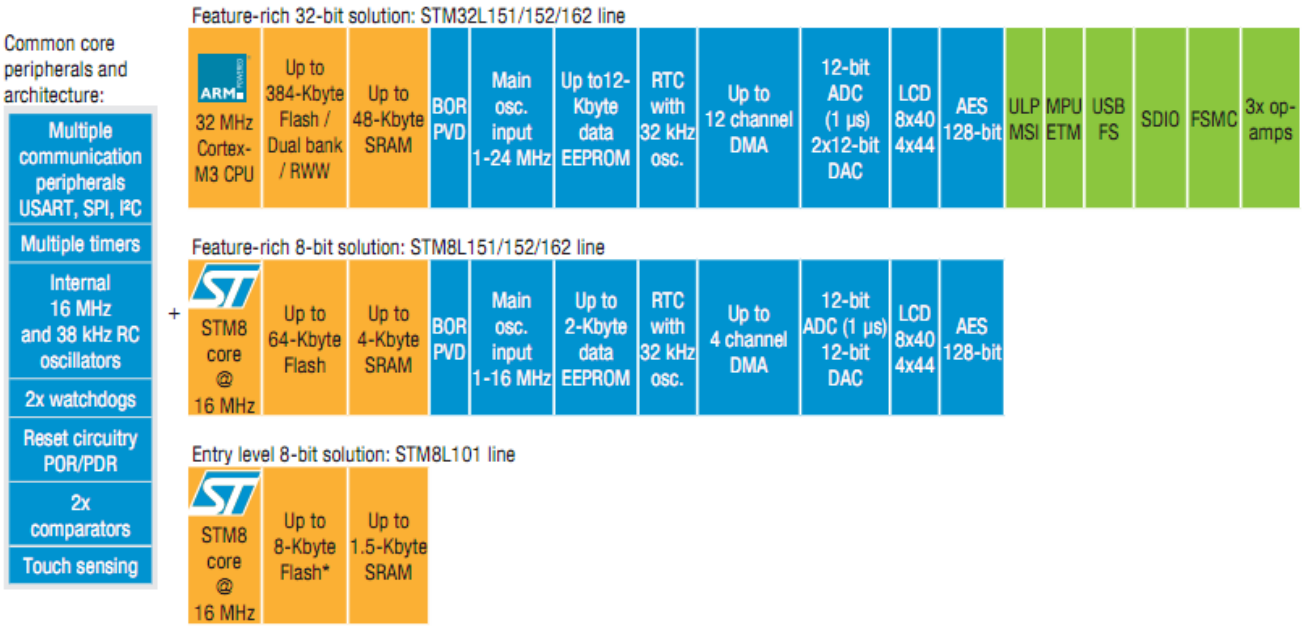
STMicroelectronics' ultra-low-power portfolio includes a full range of 8-bit and 32-bit MCUs, and so addresses most applications requiring reduced current consumption, from ultra-simple, cost-optimized feature needs to complex, high-performance requirements.



Key features

- Platform for 8-bit STM8L and 32-bit STM32L MCUs
- ST 130 nm ultra-low-leakage process technology – speed and power consumption are independent of MCU power supply
- Ultra-low-power modes: down to 300 nA
- Ultra-low voltage supply: 1.65 to 3.6 V
- Advanced analog functions down to 1.8 V
- Fast wake up
- On-board security and safety features for critical applications
- 33.3 DMIPS at 32 MHz (STM32L) and up to 16 MIPS at 16 MHz (STM8L)

Ultra-low-power product lines



由下图可以看出，STM8L 是一款基于 STM8S 系列单片机又加入了超低功耗功能的芯片，其丰富的低功耗模式和超小的功耗都为客户的功耗敏感项目提供了非常出色的解决方案。

STM8L ultra-low power consumption values

Operating mode	STM8L101	STM8L15x/STM8L16x	
	Typ 1.8 V – 3.3 V, 25 °C	Typ 1.8 V, 25 °C	Typ 3.0 V, 25 °C
Run from Flash mode	150 µA/MHz	192 µA/MHz	192 µA/MHz
Run from RAM mode	75 µA/MHz	90 µA/MHz	90 µA/MHz
Low-power Run from RAM	n.a	5.1 µA	5.1 µA
Low-power Wait	n.a	3.0 µA	3.0 µA
Active Halt with RTC	n.a	1.2 µA	1.35 µA
Active Halt with AWU	0.8 µA	1 µA	1 µA
Halt mode	0.35 µA	0.4 µA	0.4 µA

2 Discovery 平台

如图所示，STM8L-DISCOVERY 开发板能够帮助用户测试、开始开发 STM8L 系列超低功耗 MCU，

它是基于 STM8L152C6T6 芯片的平台，板子还内置了 ST-Link 模块、LED、LCD（24segments，4commons）和按键模块。

2.1 开机

连接 STM8L-DISCOVERY 的 mini USB 接口到 PC 机，可以看到 LCD 显示 STM8L DISCOVERY 等等字样，然后进入测量模式，通过按 USER Button 来切换这几种功能，几种功能如下图所示：

Table 1. Functions

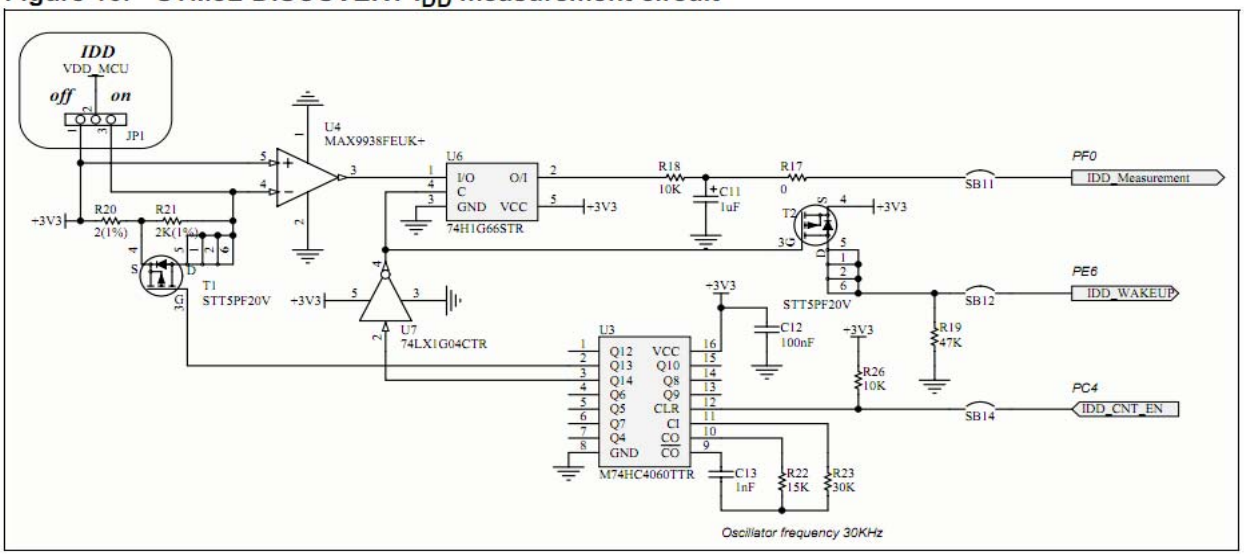
Function	LED LD3/4	Bars	Value displayed
1	Blink		STM8L V_{DD} voltage measured
2	Off		STM8L consumption measured in Run mode
3	Off		STM8L consumption measured in Low power mode, LCD on
4	Off		STM8L consumption measured in Low power mode, LCD off
5	Off		STM8L consumption measured in Halt mode, LCD off

2.2 功耗分析

我测试得到的功耗分别为：Run Mode=1.15mA；Low Power Wait Mode（LCD On）=7.43uA；Low Power Wait Mode（LCD Off）=3.80uA；Active-halt Mode=0.37uA。这几种模式是 STM8L152 芯片的 5 中低功耗模式中有代表性的几种。

那么，功耗是如何被测量出来的呢？我们看它的原理图：主要是利用 MAX9938 这颗芯片和

Figure 10. STM8L-DISCOVERY I_{DD} measurement circuit



R20（2 欧姆精度 1%）来测量经过电路的电流的。通过 JP1 跳线选择模式，当选择 1-2 相连的时候就是不测量模式，2-3 相连就是测量模式。

2.3 STM8L152 低功耗模式

STM8L152 共有 7 种工作模式：

2.3.1 run from flash

程序在片内 flash 里面运行，外设正常启动。

2.3.2 run from ram

程序在片内 ram 里面运行，外设正常启动。

2.3.3 low power run

低功耗运行模式，程序在 ram 中运行，关闭 flash 电源，外设可以选择性关闭，利用 LSI 或 LSE 工作，电压调节器从 MVR 切换到低功耗模式。

2.3.4 low power wait

在 low power run 模式运行的基础上加上 WFE（wait for even）或者 WFI（wait for interrupt）。

2.3.5 active-halt with RTC

选择 LSI 为主时钟，关闭大部分外设电源。使能 RTC 作为唤醒源，运行 halt 程序。

2.3.6 active-halt with AWU

和 active-halt with 模式唯一不同就是选择 AWU 作为唤醒源。

2.3.7 halt mode

外设和 CPU 已经关闭，靠外部中断唤醒。运行 halt 程序。

3 测试软件分析

首先看看芯片资料里面对各种模式的描述：

Table 13. Low power mode summary

Mode	Entry	Oscillator	CPU	Peripheral	Wakeup	Voltage regulator mode
Wait	WFI	On	Off	On	All internal or external interrupts, reset	MVR
	WFE	On	Off	On	All internal or external interrupts, wakeup events, reset	MVR
Low power run mode	Software sequence	LSI or LSE clock	On	On	Software sequence, reset	ULP
Low power wait mode	Software sequence +WFE	LSI or LSE clock	Off	On	Internal or external event, reset	ULP
Active-halt	HALT(1)	Off except LSI or LSE clock	Off	Off except RTC and possibly LCD	External interrupts, RTC interrupt, reset	MVR/ULP depending on CLK_ICKCR register
Halt	HALT(1)	Off	Off	Off	External interrupts, reset	ULP

我们根据板子的测试情况重点将两种。

3.1 Low Power Wait Mode

```
/* Displays current in Low Power with LCD */
case STATE_LPR_LCD:
        Icc_measure_LPR_LCD();
break;

/* Displays current in Low Power without LCD */
case STATE_LPR:
        Icc_measure_LPR();
break;
```

源代码中这两个分支分别测试了开 LCD 和关 LCD 情况，虽然表明了是 LPR，但是，是处于 LPW 模式。如何进入 low power wait 模式呢？看看资料上面的说明：

4.7.1 Entering Low power run mode

This mode is entered by executing the following software sequence:

1. Switch off all unused peripherals, oscillators (except LSI or LSE) and analog blocks
2. Mask all interrupts
3. Jump to RAM
4. Switch system clock to LSI or LSE clock sources
5. Configure the Flash memory in I_{DDQ} mode by setting the EEPM bit in the FLASH_CR1 register
6. Add a software delay loop to make sure the Flash/Data EEPROM are off
7. Configure the ultralow power mode for the regulator by setting the REGOFF bit in the CLK_REGCSR register (do not confuse with the ULP bit in the PWR_CSR2 which configures the behaviour of the internal reference voltage).

4.7.2 Exiting Low power run mode

The only way to exit this mode is to follow these steps:

1. Switch on the main regulator by resetting the REGOFF bit in the CLK_REGCSR register. The REGREADY flag in the CLK_REGCSR register is set when the regulator is ready.
2. Switch on the Flash/Data EEPROM by resetting the EEPM bit in the FLASH_CR1 register. The EEREADY flag in the CLK_REGCSR register is set when the Flash/Data EEPROM is ready.
3. Switch the clock to HSI (or HSE).
4. Reset interrupt mask.
5. Switch on what is necessary and jump to Flash/Data EEPROM if needed.

4.8 Low power wait mode

This mode is entered by executing a WFE instruction, while the MCU is in Low power run mode. It can be exited only by means of an external or internal event, in this case the MCU returns to Low power run mode. WFI instruction cannot be used, because interrupts have to be disabled in Low power run mode.

Low power wait 就是在运行 low power run 模式的基础之上，增加了运行 WFI、WFE 程序，编

写程序的重点就在于那 7 点，在此再重复一下：

（1） 关闭不使用的外设
（2） 关闭中断
（3） 程序在 ram 中执行
（4） 将主时钟切换到 LSE 或 LSI
（5） 关闭 flash 电源
（6） 延时一段时间，等待 flash 电源关闭
（7） 关闭 MVR，自动切换到低功耗电源

软件代码如下所示：

```
/* Set STM8 in low power */
  PWR->CSR2 = 0x2;

  LCD_Cmd(DISABLE);

  /* To wait LCD disable */
  while ((LCD->CR3 & 0x40) != 0x00);

/* Set GPIO in low power*/
  GPIO_LowPower_Config();

/* Stop RTC Source clock */
  CLK_RTCClockConfig(CLK_RTCCLKSource_Off, CLK_RTCCLKDiv_1);

  #ifdef USE_LSE
    CLK_LSEConfig(CLK_LSE_OFF);
    while ((CLK->ECKCR & 0x04) != 0x00);
  #else
    CLK_LSICmd(DISABLE);
    while ((CLK->ICKCR & 0x04) != 0x00);
  #endif

  /* Stop clock RTC and LCD */
  CLK_PeripheralClockConfig(CLK_Peripheral_RTC, DISABLE);
  CLK_PeripheralClockConfig(CLK_Peripheral_LCD, DISABLE);
```

```
/*Switch the clock to LSE and disable HSI*/
  #ifdef USE_LSE
    CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_LSE);
    CLK_SYSCLKSourceSwitchCmd(ENABLE);
    while (((CLK->SWCR)& 0x01)==0x01);
    CLK_HSICmd(DISABLE);
  #else
    CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
    CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_LSI);
    CLK_SYSCLKSourceSwitchCmd(ENABLE);
    while (((CLK->SWCR)& 0x01)==0x01);
    CLK_HSICmd(DISABLE);
  #endif

/*Configure event for WAKEUP and FUNCTION, disable the interrupts*/

  sim();

/* To copy function LPR_Ram in RAM section LPRUN*/
#ifdef _COSMIC_
  if (!(_fctcpy('L')))
    while(1);
#endif

  LPR_Ram(); // Call in RAM

  /*Switch the clock to HSI*/
  CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_8);
  CLK_HSICmd(ENABLE);
  while (((CLK->ICKCR)& 0x02)!=0x02);

  CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSI);
  CLK_SYSCLKSourceSwitchCmd(ENABLE);
  while (((CLK->SWCR)& 0x01)==0x01);
```

LPR_Ram（）函数是在 ram 里面运行的，要注意，WFI、WFE 不能超出 ram 范围，否则会出错。

```
#ifdef _COSMIC_
#pragma section (LPRUN)
void    LPR_Ram(void)
#endif
#ifdef _RAISONANCE_
void    LPR_Ram(void) inram
#endif
#ifdef _IAR_
#pragma location="MY_RAM_FUNC"
void    LPR_Ram(void)
#endif
{
  uint8_t i = 0;

/* To reduce consumption to minimal
  Swith off the Flash */
  FLASH->CR1 = 0x08;
  while(((CLK->REGCSR)&0x80)==0x80);

/* Swith off the Regulator*/
  CLK->REGCSR = 0x02;
  while(((CLK->REGCSR)&0x01)==0x01);

/* Set trigger on GPIOE pin6*/
  WFE->CR2 = 0x04;
  GPIOE->CR2 = 0x44;

  for (i=0; i<100; i++);

/* To start counter on falling edge*/
  GPIO_LOW(CTN_GPIO_PORT,CTN_CNTEN_GPIO_PIN);

/*Wait for end of counter */
  wfe();

  EXTI->SR1 |= 0x40;
  WFE->CR2 = 0x00;

  //Switch on the regulator
  CLK->REGCSR = 0x00;
  while(((CLK->REGCSR)&0x1) != 0x1);
}
```

3.2 Active-halt Mode

```
/* Displays current in Halt without LCD */
case STATE_HALT:
        Icc_measure_HALT();
break;
```

源代码中这一分支用来测试 Active-halt 模式。如何进入 Active-halt Mode，我们看看资料上面的说明：

7.8 Halt mode

In this mode the system clock is stopped. This means that the CPU and all the peripherals clocked by SYSCLK or by derived clocks are disabled, except for the following cases:

- The HSI clock is not stopped if used by SWIM
- The system clock source is not stopped if a Flash/Data EEPROM write operation is in progress
- The LSI clock is not stopped if used by the SWIM, the IWDG or if the “IWDG_HALT” option bit is disabled.

In Halt mode, none of the peripherals are clocked and the digital part of the MCU consumes almost no power.

7.8.1 Entering Halt mode

The MCU enters Halt mode when a HALT instruction is executed.

Note: *It is recommended not to enter Halt/Active-halt mode from the Low power run mode. Otherwise, the only safe way to exit one of these two modes is to reset the MCU.*

Before executing a HALT instruction, the application must clear all pending peripheral interrupts by clearing the interrupt pending bit in the corresponding peripheral configuration register. Otherwise, the HALT instruction is not executed and program execution continues.

However, the Halt procedure can be delayed if one of the following flags is set:

- SWBSY flag in the CLK_SWCR register
- EEBUSY flag in the CLK_CLK_REGCSR register
- RTCSWBSY flag in the CLK_CRTCR register
- BEEPSWBSY flag in the CLK_CBEEPR register when BEEP in Active-halt mode enabled.

If SAHALT bit is set in the CLK_ICKCR register the main regulator (MVR) will be switched off without taking into account that some high-speed clock may be used by the system.

7.8.2 Exiting Halt mode

Wakeup from Halt mode is triggered by an external interrupt sourced by a general purpose I/O port configured as interrupt input or by an alternate function pin capable of triggering a peripheral interrupt.

The system clock is restarted with a frequency depending on the FHW bit in the CLK_ICKCR register. If the FHW bit is set, HSI/8 is the system clock, otherwise the system clock is the last selected clock source before entering Halt mode.

In an interrupt based application, where most of the processing is done through the interrupt routines, the main program may be suspended by setting the activation level bit (AL) in the CPU configuration register. Setting this bit causes the CPU to return to Halt mode when executing the return from interrupt, without restoring the main execution context.

Power consumption is reduced as there is no save/restore context activity and no need for a main software loop execution for power management (in order to return to WFI mode).

After a wake up from Halt mode, the LCD clock switches from RTCCLK to SYSCLK. To have a stable clock signal without glitches, 2 RTCCLK cycles are needed for synchronization reasons. Consequently, read/write access to LCD registers is not possible during this period.

7.9 Active-halt mode

Active-halt mode is similar to Halt mode.

Note: *It is recommended not to enter Halt/Active-Halt mode from the Low power run mode. Otherwise, the only safe way to exit one of these two modes is to reset the MCU.*

In Active-halt mode, the main oscillator, the CPU and almost all peripherals are stopped. Only oscillator or the LSE crystal is running to drive the SWIM, beeper, IWDG, RTC and LCD if enabled.

由上面的图片可以看出，进入 halt 模式，只需要在程序的适当位置运行 halt（）；程序就可以了，不过要注意提到的清中断标志位，另外，不要试图在 low power run 模式进入 halt 模式，否则就只能复位才能唤醒了。而进入 Active-halt Mode 模式，只需要在 halt 的基础之上，设定唤醒源即可，比如 RTC。软件代码如下：

```
/* Set STM8 in low power */
  PWR->CSR2 = 0x2;

  LCD_Cmd(DISABLE);

  /* To wait LCD disable */
  while ((LCD->CR3 & 0x40) != 0x00);

/* Set GPIO in low power*/
  GPIO_LowPower_Config();

/* Stop RTC Source clock */
  CLK_RTCClockConfig(CLK_RTCCLKSource_Off, CLK_RTCCLKDiv_1);

  #ifdef USE_LSE
    CLK_LSEConfig(CLK_LSE_OFF);
    while ((CLK->ECKCR & 0x04) != 0x00);
  #else
    CLK_LSICmd(DISABLE);
    while ((CLK->ICKCR & 0x04) != 0x00);
  #endif

  /* Stop clock RTC and LCD */
  CLK_PeripheralClockConfig(CLK_Peripheral_RTC, DISABLE);
  CLK_PeripheralClockConfig(CLK_Peripheral_LCD, DISABLE);
```

```
  sim();

  /* To prepare to start counter */
  GPIO_HIGH(CTN_GPIO_PORT,CTN_CNTEN_GPIO_PIN);

  /* Falling edge for start counter */
  GPIO_LOW(CTN_GPIO_PORT,CTN_CNTEN_GPIO_PIN);
  delay_10us(1);

  /* MCU in halt during measurement */
  /* Wake up by Interrupt done counter Input Port E pin 6 */
  halt();
```

唤醒之后从 halt（）；代码后面开始执行，用户要注意正确的设置时钟和系统初始化。

总之，用户可以参考两部分代码来设计自己的程序。至于，wait mode 就是指采用了 WFI、WFE 代码的程序，halt mode 都可以从这两部分中演化出相应的设计方法。用户在设计初期还要参考 STM8L15x_StdPeriph_Lib_V1.5.1 固件库和芯片的参考手册、datasheet 和 AN3147 STM8L family power management.pdf 手册，综合考虑各种情况，比如：客户板子的 IO 口设置成何种模式，是根据板子的具体情况而定的，如果没有特别要求，设置成输出低电平是最理想的状态。在设计中要多注意此类问题。