EFR32FG23 开发板试用报告

一、 摘要

EFR32FG23 开发板是一款紧凑、功能丰富的开发平台。它能够快速开发 1 GHz 以下的物联网产品，并完成原型设计。开发平台包括支持 FG23 的板载段式液晶控制器和其他关键功能，包括 LEESE 和脉冲计数器。

该板的亮点包括段式 LCD、Si7021 相对湿度和温度传感器以及用于金属检测的 LC 传感器。使用 USB Micro-B 电缆和板载 J-Link 调试器可以轻松对 EFR32FG23 进行编程。USB 虚拟 COM 端口(VCOM)提供与目标应用程序的串行连接，数据包跟踪接口(PTI)提供有关无线链路中传输和接收数据包的宝贵调试信息。是适用于智能家居、安防、照明、楼宇自动化和计量的 1 GHz 以下物联网无线连接的理想解决方案。

非常感谢 Silicon Labs 和 EDOM 官方提供的这次宝贵的试用机会，接下来将对这块开发板进行具体评测。

二、 开箱及资料获取

EFR32FG23 套件包括 1 个开发板，1 根 868MHz 天线，1 根 915MHz 天线以及 1 根 USB 连接线，开发板整体布局紧凑美观，并且十分小巧，板载天线接口和 J-Link，方便调试，同时有多种传感器设备，非常适合物联网应用开发设计。如图 1 所示。

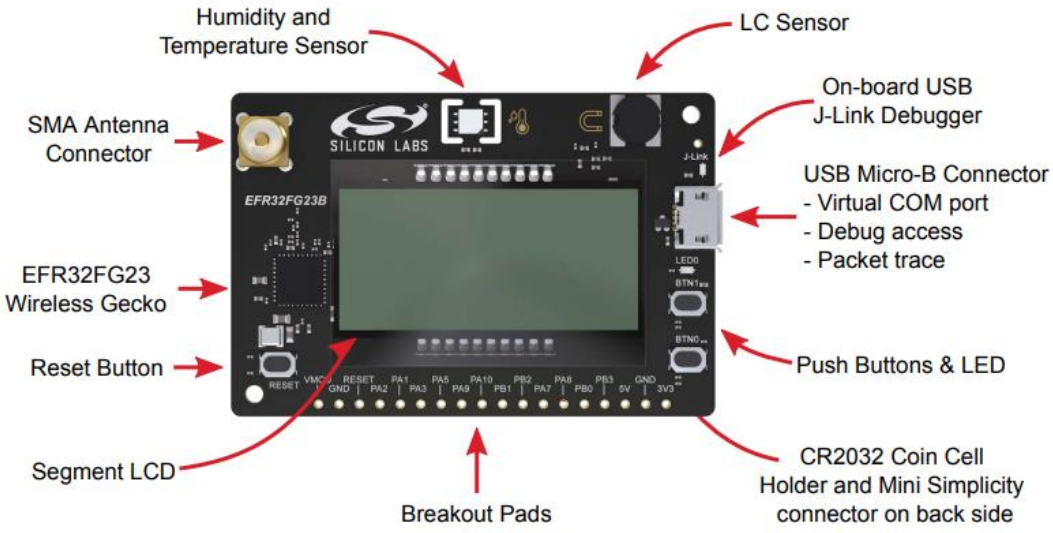

图 1 EFR32FG23 板载资源

该开发板支持 3 种供电方式：1.通过板载 USB 接口进行供电，典型电压为 5V，经过线性稳压器转换为 3.3V 为 MCU 供电；2.通过 CR2032 纽扣电池进行供电，典型电压值为 3V，可直接为 MCU 供电，若此时 USB 也插入供电，会通过一个自动选择电路将系统电源由电池供电选择为 USB 供电，同时保护电池免受反向电流作用；3.通过开发板背面的 Mini Simplicity connector 接口外接调试器进行供电，典型电压值为 3.3V，特别需要注意的是，此时系统不能外接其他任何电源，需要把 USB 线拔掉，纽扣电池取下，否则会导致电源冲突进一步可能损毁电路。如图 2 所示。

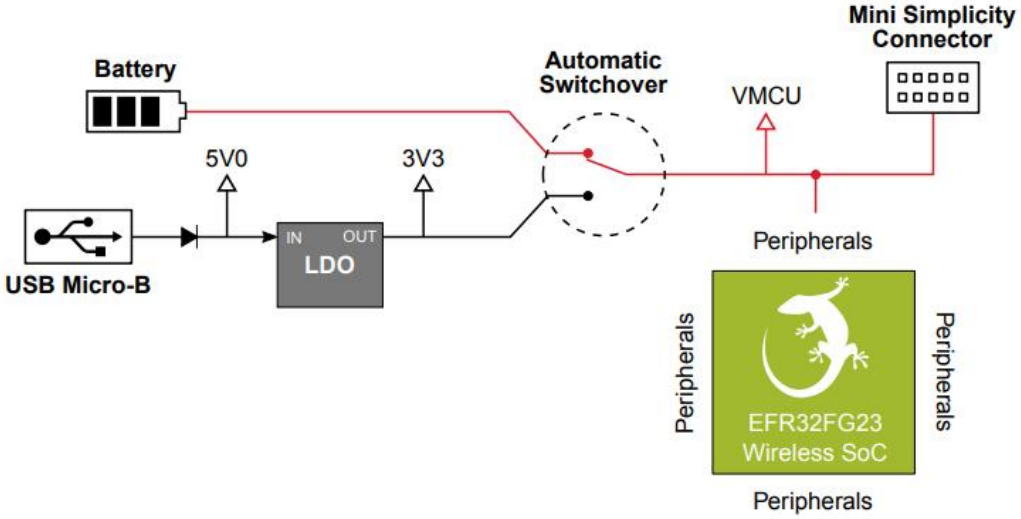

图 2 EFR32FG23 系统电源结构

应用开发需要通过 Simplicity Studio 进行，之前接触过 Silicon Labs 的 EFR32BG22 开发套件，同样也使用上面的 IDE 进行开发，不得不说 Simplicity Studio 真的极为方便，针对 Silicon Labs 技术、SoC 和模块完全可以做到一站式开发。从 https://cn.silabs.com/developers/simplicity-studio 下载该 IDE 进行安装后将 EFR32FG23 通过 USB 线接入 PC，IDE 可以自动识别板卡型号，根据提示可下载相应的 SDK 进行开发。

三、 PWM 呼吸灯

本试验通过使用定时器产生占空比可调的 PWM 方波，以此控制板载 LED0 的亮度。

通过 sl_status_t sl_pwm_init(sl_pwm_instance_t *pwm, sl_pwm_config_t *config);函数实现 PWM 初始化，设置相应的频率、通道、端口、引脚等信息，具体代码如图 3 所示。

```
sl_pwm_instance_t sl_pwm_led0 = {
  .timer = SL_PWM_LED0_PERIPHERAL,
  .channel = (uint8_t)(SL_PWM_LED0_OUTPUT_CHANNEL),
  .port = (uint8_t)(SL_PWM_LED0_OUTPUT_PORT),
  .pin = (uint8_t)(SL_PWM_LED0_OUTPUT_PIN),
#if defined(SL_PWM_LED0_OUTPUT_LOC)
  .location = (uint8_t)(SL_PWM_LED0_OUTPUT_LOC),
#endif
};


void sl_pwm_init_instances(void)
{

  sl_pwm_config_t pwm_led0_config = {
    .frequency = SL_PWM_LED0_FREQUENCY,
    .polarity = SL_PWM_LED0_POLARITY,
  };

  sl_pwm_init(&sl_pwm_led0, &pwm_led0_config);

}
```

图 3 PWM 初始化

之后通过引入 for 循环实现 PWM 占空比由小变大，再由大变小的过程，映射到 LED0 上，即出现灯先由暗变亮，再由亮变暗的过程，即呼吸灯。具体代码如图 4 所示。

```
void blink_pwm_process_action(void)
{
  for (uint8_t i = 0; i < 100; i++) {
    sl_pwm_set_duty_cycle(&sl_pwm_led0, pwm_lut[i]);
    sl_sleeptimer_delay_millisecond(6);
    if (i == 0) {
      sl_sleeptimer_delay_millisecond(190);
    }
  }
  for (uint8_t i = 100; i > 0; i--) {
    sl_pwm_set_duty_cycle(&sl_pwm_led0, pwm_lut[i]);
    sl_sleeptimer_delay_millisecond(6);
    if (i == 100) {
      sl_sleeptimer_delay_millisecond(190);
    }
  }
}
```

图 4 PWM 呼吸灯

实际效果如图 5 所示。

图 5 呼吸灯实际效果

四、 Si7021 温湿度检测

Si7021 是一个集成温湿度传感器，模数转换器，信号处理，数据校准，

具有标准 IIC 接口的 CMOS 器件，本试验通过 MCU 的 IIC 接口与 Si7021 进行通信以获取 Si7021 检测到的温度和湿度。

通过 void I2CSPM_Init(I2CSPM_Init_TypeDef *init);对 MCU 的 IIC 接口进行初始化，具体代码如图 6 所示。

```
I2CSPM_Init_TypeDef init_sensor = {
  .port = SL_I2CSPM_SENSOR_PERIPHERAL,
  .sclPort = SL_I2CSPM_SENSOR_SCL_PORT,
  .sclPin = SL_I2CSPM_SENSOR_SCL_PIN,
  .sdaPort = SL_I2CSPM_SENSOR_SDA_PORT,
  .sdaPin = SL_I2CSPM_SENSOR_SDA_PIN,
  .i2cRefFreq = 0,
  .i2cMaxFreq = SL_I2CSPM_SENSOR_MAX_FREQ,
  .i2cClhr = SL_I2CSPM_SENSOR_HLR
};

void sl_i2cspm_init_instances(void)
{
  CMU_ClockEnable(cmuClock_GPIO, true);
  I2CSPM_Init(&init_sensor);
}
```

图 6 IIC 初始化

然后对 Si7021 温湿度传感器进行相应初始化，并测量温湿度值

```
sl_status_t sl_si70xx_init(sl_i2cspm_t *i2cspm, uint8_t addr)
{
  sl_status_t status;

  status = SL_STATUS_OK;

  if (!sl_si70xx_present(i2cspm, addr, NULL)) {
    /* Wait for sensor to become ready */
    sl_sleeptimer_delay_millisecond(80);

    if (!sl_si70xx_present(i2cspm, addr, NULL)) {
      status = SL_STATUS_INITIALIZATION;
    }
  }

  return status;
}

sl_status_t sl_si70xx_measure_rh_and_temp(sl_i2cspm_t *i2cspm, uint8_t addr, uint32_t *rhData,
                                          int32_t *tData)
{
  sl_status_t retval;
  retval = sl_si70xx_send_command(i2cspm, addr, rhData, SI70XX_MEASURE_RH);

  if (retval != SL_STATUS_OK) {
    return retval;
  }

  *rhData = sl_si70xx_get_percent_relative_humidity(*rhData);

  retval = sl_si70xx_send_command(i2cspm, addr, (uint32_t *) tData, SI70XX_READ_TEMP);

  if (retval != SL_STATUS_OK) {
    return retval;
  }

  *tData = sl_si70xx_get_celcius_temperature(*tData);

  return retval;
}
```

图 7 温湿度值检测

对所测得的温湿度值进行相应的数据处理转换为℃与 RH。

```
void process_Si7021(void)
{
    u16 TEMP,HUMI;
    u8 curI;
    Multiple_read_Si7021(TEMP_NOHOLD_MASTER,&TEMP);//NOHOLD_MASTER
    si7021.temp=(((((float)TEMP)*175.72f)/65536.0f) - 46.85f);
    Multiple_read_Si7021(HUMI_NOHOLD_MASTER,&HUMI);//NOHOLD_MASTER
    si7021.humi=(((((float)HUMI)*125.0f)/65535.0f) - 6.0f);

    if(MEAN_NUM > si7021_filter.curI)
    {
        si7021_filter.tBufs[si7021_filter.curI] = si7021.temp;
        si7021_filter.hBufs[si7021_filter.curI] = si7021.humi;
        si7021_filter.curI++;
    }
    else
    {
        si7021_filter.curI = 0;
        si7021_filter.tBufs[si7021_filter.curI] = si7021.temp;
        si7021_filter.hBufs[si7021_filter.curI] = si7021.humi;
        si7021_filter.curI++;
    }
    if(MEAN_NUM <= si7021_filter.curI)
    {
        si7021_filter.thAmount = MEAN_NUM;
    }
    if(0 == si7021_filter.thAmount)
    {
        for(curI = 0; curI < si7021_filter.curI; curI++)
        {
            si7021.temp += si7021_filter.tBufs[curI];
            si7021.humi += si7021_filter.hBufs[curI];
        }
        si7021.temp = si7021.temp / si7021_filter.curI;
        si7021.humi = si7021.humi / si7021_filter.curI;
        TEMP_buf = si7021.temp;
        Humi_buf = si7021.humi;
    }
}
```

图 8 温湿度值处理

实际效果如图 9 所示。

图 9 Si7021 温湿度值

五、 Micrium OS kernrl

第一眼看到这个操作系统的名字有点眼生，后来了解后发现 μC/OS 和 Micrium OS 都属于 RTOS 公司 Micrium，而 Silicon Labs 收购了 Micrium 并

开发了 Micrium OS 为了发展自己 MCU 的生态和业务。

本试验通过在 Micrium OS kernrl 系统上运行 EUSART，通过虚拟串口 VCOM 与电脑实现通信。

通过 void app_iostream_eusart_init(void)函数实现 EUSRT 初始化与任务创建功能，并分配相应的堆栈与优先级。具体代码如图 10 所示。

```
void app_iostream_eusart_init(void)
{
  RTOS_ERR err;

  /* Prevent buffering of output/input.*/
#if !defined(__CROSSWORKS_ARM) && defined(__GNUC__)
  setvbuf(stdout, NULL, _IONBF, 0);   /*Set unbuffered mode for stdout (newlib)*/
  setvbuf(stdin, NULL, _IONBF, 0);   /*Set unbuffered mode for stdin (newlib)*/
#endif

  // Create Blink Task
  OSTaskCreate(&tcb,
               "iostream terminal task",
               app_iostream_terminal_task,
               DEF_NULL,
               TERMINAL_TASK_PRIO,
               &stack[0],
               (TERMINAL_TASK_STACK_SIZE / 10u),
               TERMINAL_TASK_STACK_SIZE,
               0u,
               0u,
               DEF_NULL,
               (OS_OPT_TASK_STK_CLR),
               &err);
  EFM_ASSERT((RTOS_ERR_CODE_GET(err) == RTOS_ERR_NONE));
}
```

图 10 创建任务

任务创建完成后通过 void app_iostream_terminal_task(void *arg)进行任务的具体操作，具体代码如图 11 所示。

```
void app_iostream_terminal_task(void *arg)
{
  int8_t c = 0;
  uint8_t index = 0;

  (void)&arg;

  /* Output on vcom usart instance */
  const char str1[] = "IOstream EUSART example\r\n\r\n";
  sl_iostream_write(sl_iostream_vcom_handle, str1, strlen(str1));

  /* Setting default stream */
  sl_iostream_set_default(sl_iostream_vcom_handle);
  const char str2[] = "This is output on the default stream\r\n";
  sl_iostream_write(SL_IOSTREAM_STDOUT, str2, strlen(str2));

  /* Using printf */
  /* Writing ASCII art to the VCOM iostream */
  printf("Printf uses the default stream, as long as iostream_retarget_stdio is included.\r\n");

  printf("> ");
  /* Retrieve characters, print local echo and full line back */
  while (1) {
    c = getchar();
    if (c > 0) {
      if (c == '\r' || c == '\n') {
        buffer[index] = '\0';
        printf("\r\nYou wrote: %s\r\n>", buffer);
        index = 0;
      } else {
        if (index < BUFSIZE - 1) {
          buffer[index] = c;
          index++;
        }

        /* Local echo */
        putchar(c);
      }
    }
  }
}
```

图 11 任务操作

最后通过 osStatus_t　osKernelStart(void)启动系统内核，通过串口调试助手的打印信息，可以看到系统正常运行。具体代码如图 12 所示。

```
osStatus_t  osKernelStart(void)
{
  RTOS_ERR  err;

  if (CORE_InIrqContext() == true) {
    return osErrorISR;
  }

  if (OSRunning == OS_STATE_OS_RUNNING) {
    return osError;
  }

#if OS_CFG_PRIO_MAX >= osPriorityISR
  OSStart(&err);

  (void)err;   // There is no point in checking the error here
               // because the function will not return in case of success

  return osError;
#else
  RTOS_ASSERT_CRITICAL(DEF_FALSE, RTOS_ERR_NOT_AVAIL, osError);   // CMSIS-RTOS2 requires atleast 56 priority levels
  return osError;
#endif
}
```

图 12 启动系统

实际效果如图 13 所示。

图 13 EUSART 通信

六、 总结

非常荣幸能获得这次宝贵的试用机会，也非常感谢官方，通过本次试用，对芯科的无线专有产品 EFR32FG23 有了进一步的了解与学习，但由于本人能力有限，未能充分发挥该套件的性能，后续会继续改进学习，深入挖掘，设计出更多更好的物联网应用产品。谢谢！