在Keil RTX下以多工的方式存取EEPROM

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	本代码示范在 Keil RTX 程序下以多工的方式存取 EEPROM。
BSP 版本	NUC240 Series BSP CMSIS V3.01.001
开发平台	NuEdu-EVB-NUC240 V2.1 NuEdu-Basic01 V2.1

1 功能介绍

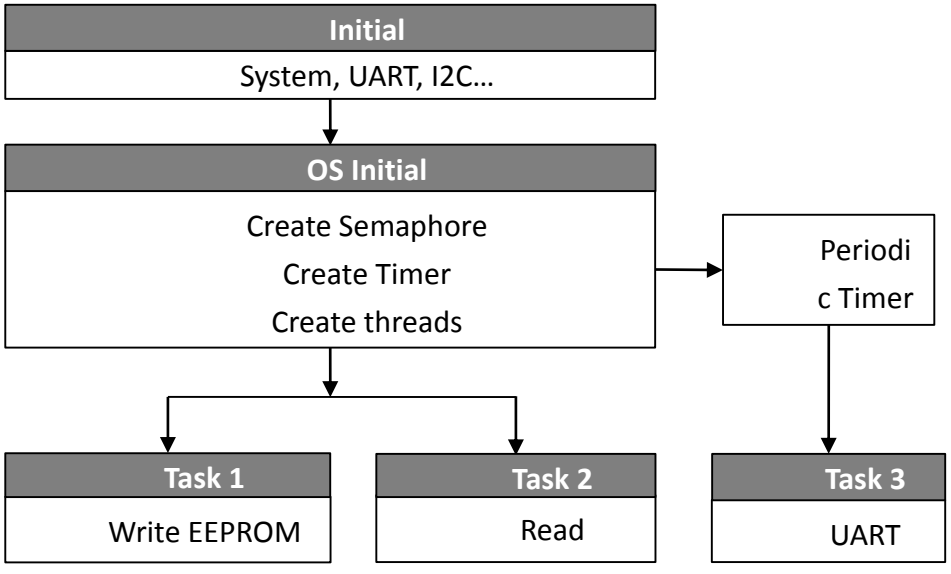
1.1 简介

当微控制器程序需同时执行许多任务时，可使用实时操作系统(real-time operating system，RTOS)来管理这些任务。RTOS将CPU的处理时间切割成许多时隙 (time slots)，每个时隙进行不同的任务，并且在每个时隙之间进行任务的切换。因为任务间的切换在一秒钟内可以达到上百次，整体看起来像是这些任务是同时在执行。当程序比较复杂需要许多任务同时执行，就可使用RTOS协助建立和管理这些任务。

Keil MDK (Microcontroller Development Kit) 内建的CMSIS-RTOS，为底层 RTOS和上层应用程序提供一套标准API，包括Keil免费的RTX (Real-Time eXecutive) 核心。当其他RTOS也支持CMSIS-RTOS，透过共同标准的API，则可轻易将程序移植到其他RTOS平台。本范例代码利用Keil里提供的CMSIS-RTOS建立RTX核心的系统，进行EEPROM的读写。

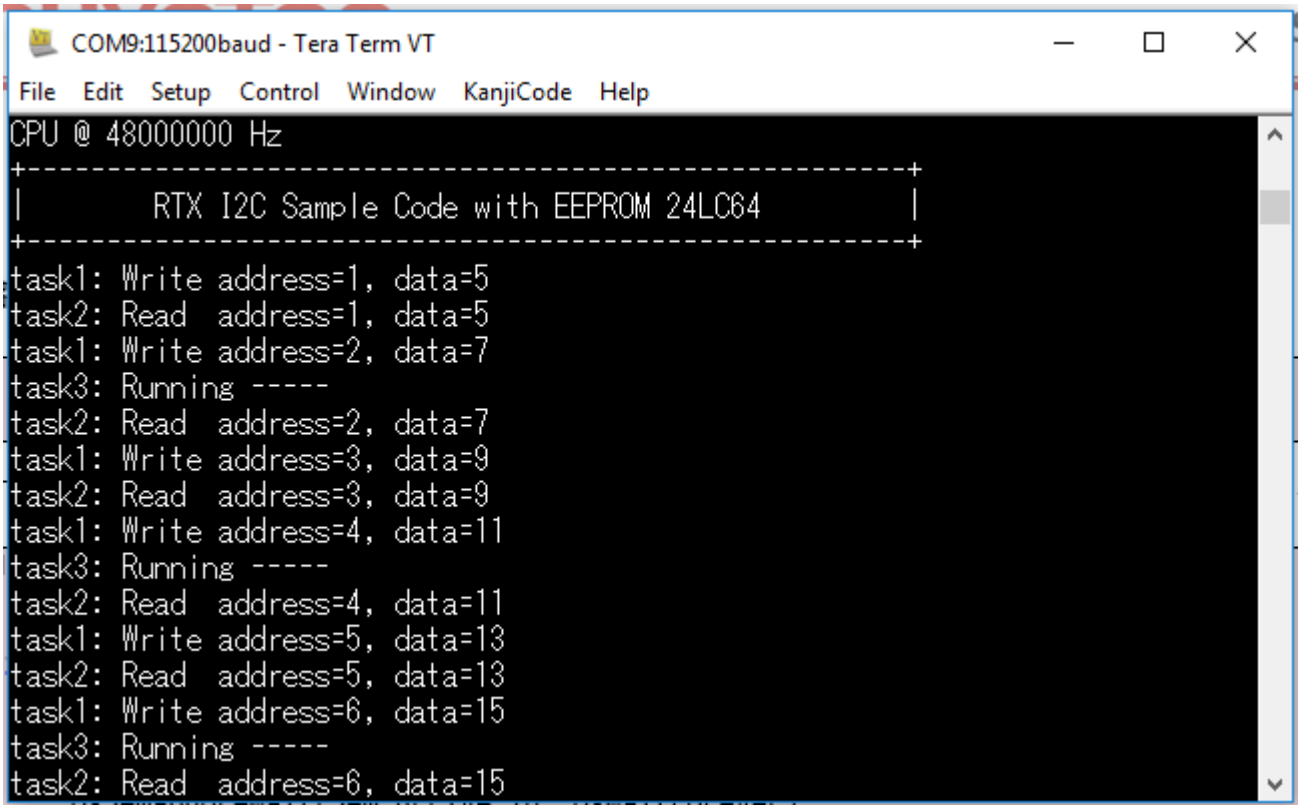
1.2 原理

程序首先做初始化，包括打印讯息的UART和存取EEPROM的I2C。建立一个Semaphore、一个Timer和两个Thread。两个Thread分别是对EEPROM的读写：其中一个 Thread先写入资料，释放一个Semaphore token；另外一个任务则等待完成EEPROM写入的Semaphore token，再读取EEPROM的数据。设定周期性的Timer，定时触发UART打印讯息。

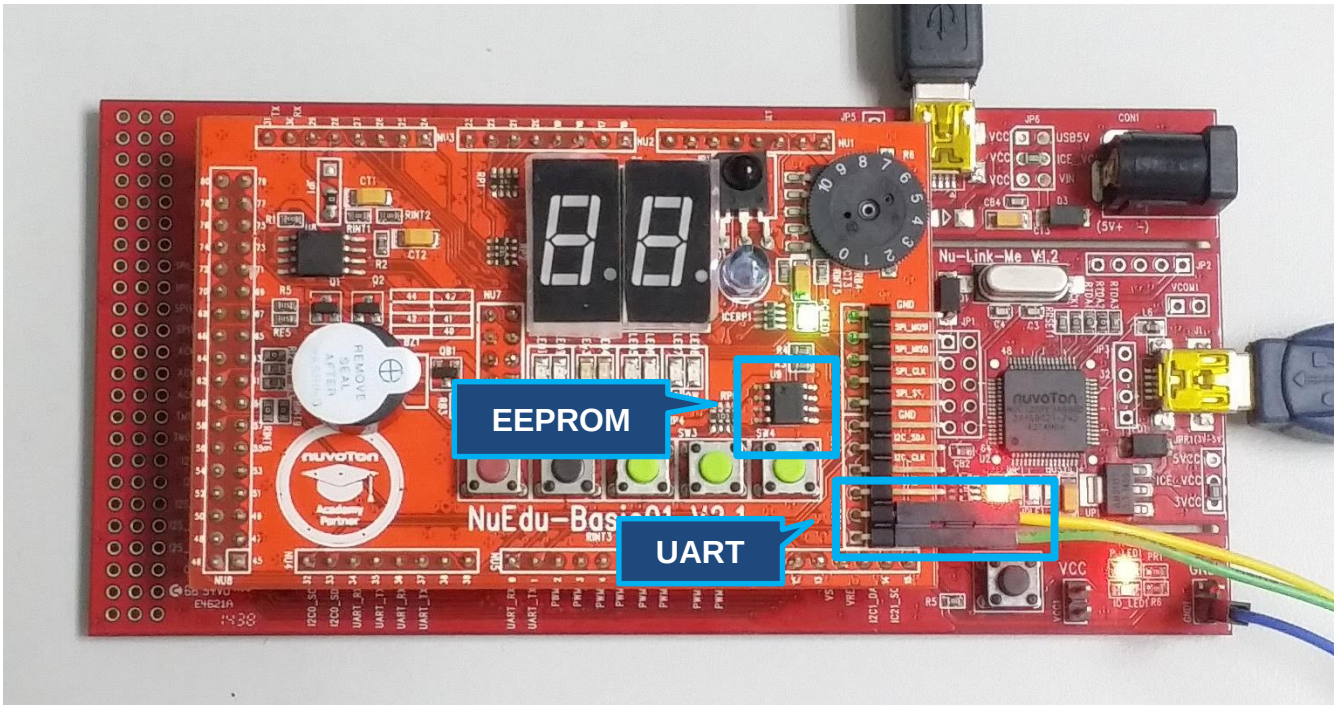


1.3 執行結果

NUC240開發板透過NuBridge連接到電腦，開啟電腦終端機，顯示UART打印的訊息。



下图为NUC240开发版上EEPROM和UART的位置。


EEPROM
UART
Nu-Link-Me V1.2

2 代码介绍

2.1 RTX

Thread

本范例代码有两个平行处理的任务，分别宣告其函式原型，并为每个任务宣告各自的ID(t_task1和t_task2)。使用osThreadDef宣告每个任务为thread，也可以用来设定任务的优先等级。

```
/* Thread IDs */
osThreadId t_task1;     // Declare a thread ID for task 1 (EEPROM Write)
osThreadId t_task2;     // Declare a thread ID for task 2 (EEPROM Read)

/* Function Declaration */
void task1_EEPROMwrite(void const *argument);
void task2_EEPROMread(void const *argument);

/* Thread Definition with function, priority, and stack requirements */
osThreadDef(task1_EEPROMwrite, osPriorityNormal, 1, 0);
osThreadDef(task2_EEPROMread, osPriorityNormal, 1, 0);
```

在主程序中使用osThreadCreate函式为每个任务建立新的thread，传回值指定给该任务的ID。

```
    /* Create task and assign thread IDs */
    t_task1 = osThreadCreate(osThread(task1_EEPROMwrite), NULL);
    t_task2 = osThreadCreate(osThread(task2_EEPROMread), NULL);
```

Semaphore

宣告作为任务之间内部互相沟通的Semaphore，名字为Sem_Arrive，以及宣告其ID变量，Sem_Arrive_id。

```
/* Semaphore ID */
osSemaphoreId Sem_Arrive_id;

/* Semaphore Declaration */
osSemaphoreDef(Sem_Arrive);
```

主程序中呼叫osSemaphoreCreate函式建立新的Semaphore，并将传回值指定给该Semaphore的ID(Sem_Arrive_id)，可用的token数目为1。

```
    /* Create Semaphore and configure token number */
    Sem_Arrive_id = osSemaphoreCreate(osSemaphore(Sem_Arrive), 1);
```

Timer

宣告一个Timer ID名字为Timer_Delay_id。定义这个Timer的名字为Timer_Delay，time-out时会触发task3_UARTprint函式。

```
/* Timer ID */
osTimerId Timer_Delay_id;

/* osTimer Declaration */
osTimerDef(Timer_Delay, task3_UARTprint);
```

主程序中呼叫osTimerCreate函式建立新的Timer，操作在周期的模式，没有传入参数，并将传回值指定给该Timer的ID(Timer_Delay_id)。呼叫osTimerStart函式开始Timer的计数，时间为50ms。每50ms就会触发task3_UARTprint函式执行。

```
    /* Create Timer in periodic mode and without argument */
    Timer_Delay_id = osTimerCreate(osTimer(Timer_Delay), osTimerPeriodic, (void *)NULL);

    /* Start Timer with period 50ms */
    osTimerStart(Timer_Delay_id, 50);
```

task3_UARTprint函式打印程序执行中讯息。

```
void task3_UARTprint(void const *argument)
{
    /* UART print message */
    printf("task3: Running -----\n");
}
```

OS Kernel 初始化和执行

呼叫osKernelInitialize函式，开始进行OS Kernel初始化。

```
    /* Initialize RTOS */
    osKernelInitialize();
```

呼叫osKernelStart函式开始执行OS Kernel以及任务切换。

```
    /* Start the RTOS */
    osKernelStart();
```

2.2 EEPROM

开启I2C，设定多功能管脚、时钟、传输速率和中断。

```
    /* Initial I2C */
    I2C_EEPROM_Init();
```

第一个任务里等有空的Semaphore token，呼叫I2C_EEPROM_Write函式写入数据，然后释出一个Semaphore token。

```
void task1_EEPROMwrite(void const *argument)
{
    while(1) {

        /* Wait for semaphore token */
        osSemaphoreWait(Sem_Arrive_id, osWaitForever);

        /* Delay 10 msec */
        osDelay(10);

        /* Access EEPROM address from 0~10 */
        u32Addr++;
        if (u32Addr>10)
            u32Addr=0;

        /* Write data */
         printf("task1: Write address=%d, data=%d \n\r",u32Addr,(u32Addr*2+3));
         I2C_EEPROM_Write(u32Addr,(u32Addr*2+3));

        /* Release semaphore token */
        osSemaphoreRelease(Sem_Arrive_id);
    }
}
```

第二个任务，等有空的Semaphore token，呼叫I2C_EEPROM_Read函式读取数据，检查数据数值，然后释出Semaphore token。

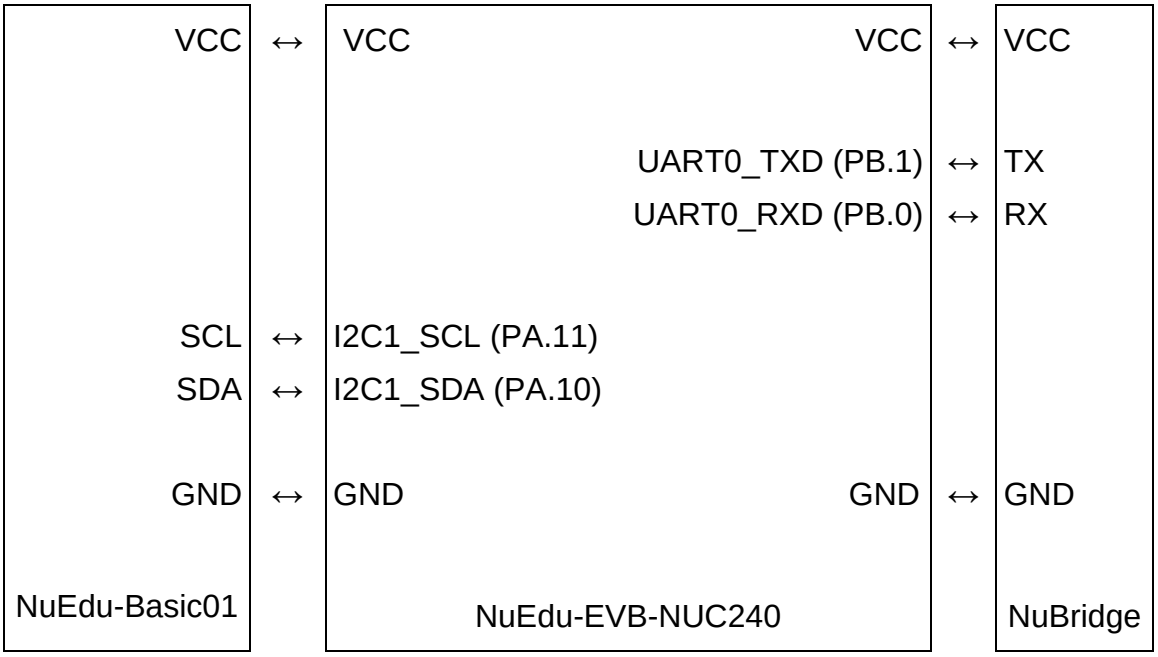
```
void task2_EEPROMread(void const *argument)
{
    uint32_t u32Data;
```

```
    while(1)
    {
        /* Wait for semaphore token */
        osSemaphoreWait(Sem_Arrive_id, osWaitForever);

        /* Read data */
        u32Data = I2C_EEPROM_Read(u32Addr);
        printf("task2: Read  address=%d, data=%d \n\r",u32Addr,u32Data);

        /* Check data is correct or not */
         if(u32Data!=(u32Addr*2+3)){
             printf("I2C Byte Write/Read Failed, Data 0x%x\n", u32Data);
             while(1);
         }

        /* Release semaphore token */
        osSemaphoreRelease(Sem_Arrive_id);

    }
}
```

3 软件与硬件环境

- 软件环境
  - BSP 版本
    - NUC240 Series BSP CMSIS V3.01.001
  - IDE 版本
    - Keil uVersion 5.28
- 硬件环境
  - 电路组件
    - NuTiny-EVB-NUC240 V1.2
    - NuEdu-Basic01 V2.1
    - Nubridge
  - 示意图

NuEdu-EVB-NUC240开发版上接NuEdu-Basic01版子和NuBridge。



4 目录信息

- EC_NUC240_RTX_EEPROM_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - NuEdu — Library for NuEdu-SDK-NUC240 board
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 NUC240_RTX_EEPROM.uvprojx。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
3. 进入除错模式接口
   a. 执行代码

6 修订纪录

Date	Revision	Description
Sep. 30, 2019	1.00	1. 初始发布.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.