透过SVC取得目前程序执行的PC值

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	本范例代码透过 SVC 读取，并打印 MCU 目前程序执行的 pc，stack， r0， r1， r12 等缓存器值。
BSP 版本	NUC230_240 Series BSP CMSIS V3.01.001
开发平台	NuTiny-EVB-NUC240 V1.2

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

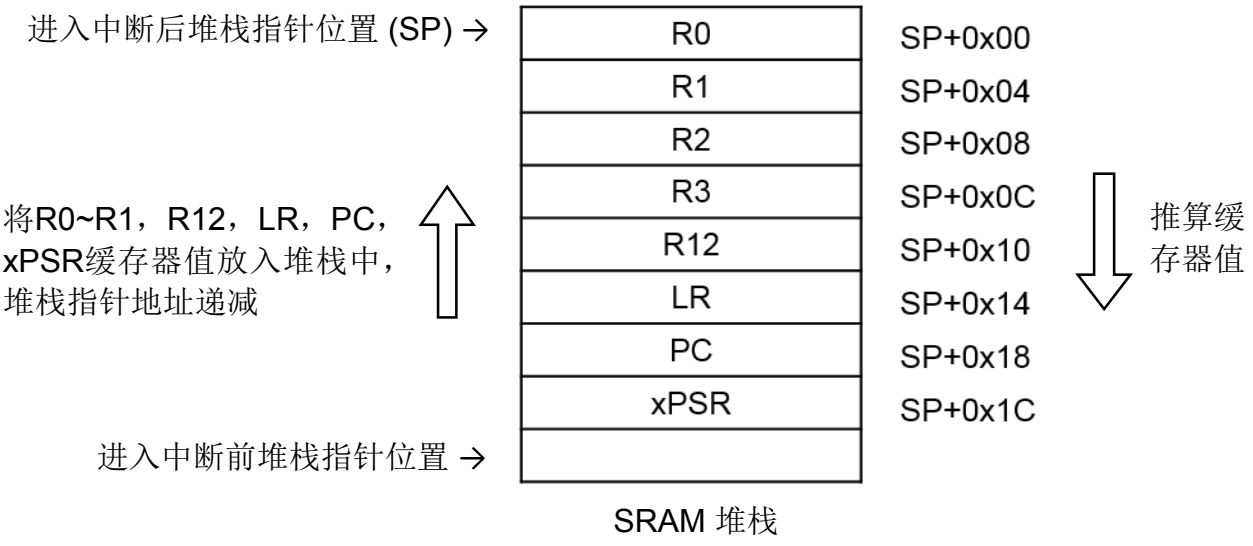
1 功能介绍

1.1 简介

透过SVC读取程序执行当下的PC值，执行SVC指令时，系统会产生SVC中断，可用于OS系统服务程序的操作。产生SVC中断时，系统会将目前缓存器的状态放入堆栈中，本范例代码利用这些信息来获取呼叫产生SVC中断时的程序地址(PC)，并将该PC值打印出来，其他可获取的信息包括堆栈(stack)，R0~R3，R12，xPSR。

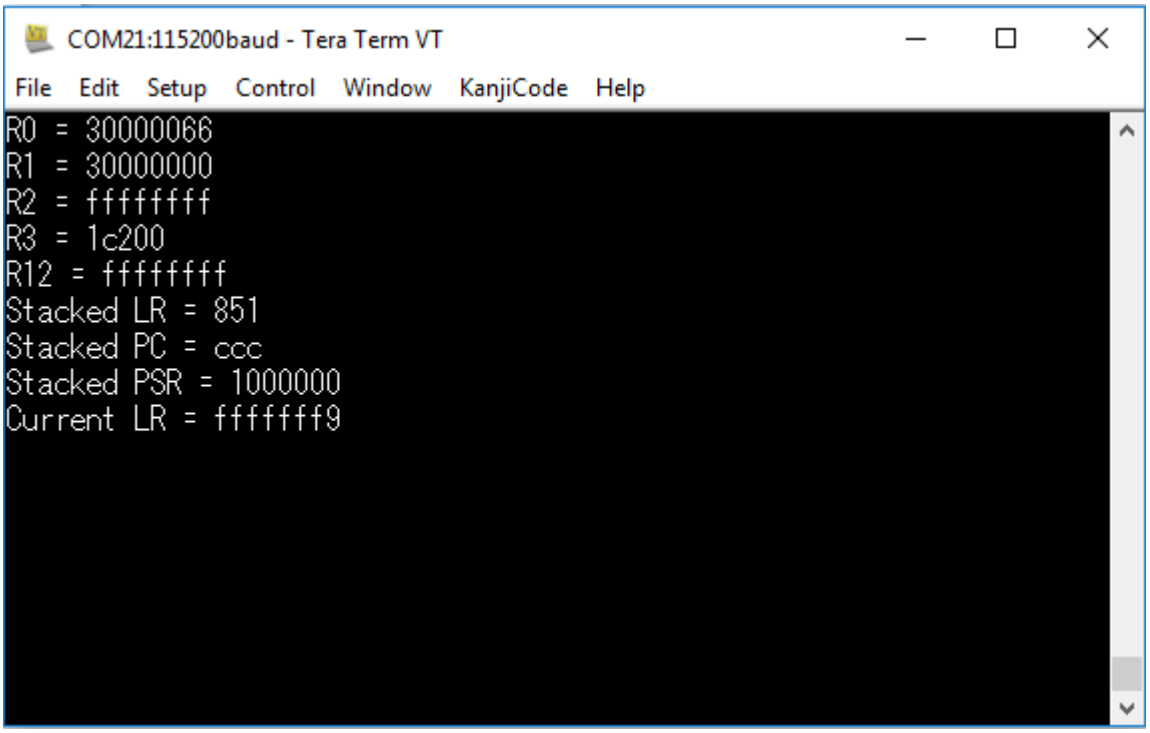
1.2 原理

中断发生时，硬件会自动将下列8个缓存器数值放入堆栈中：R0，R1，R2，R3，R12，LR，PC，xPSR。在SCV中断里，读取目前堆栈指针的位置，即可推算得知这8个数值，包括执行SVC指令时的程序地址(PC)



1.3 执行结果

使用NUC240开发板，PB.0(UART RX)及PB.1(UART TX)连接到连接至NuBridge，从UART0打印讯息。讯息显示呼叫SVC当下的PC值和其他信息。



2 代码介绍

产生SVC中断

执行SVC指令产生SVC中断，利用Keil MDK提供的 __svc 关键词定义SVC函式，定义数字0的SVC函式，没有传入和传回参数。

```
    /* Define SVC function */
    void __svc(0x00) svc_PC(void);
```

使用时直接呼叫定义的SVC函式即可。

```
    /* Supervisor Call */
    svc_PC();
```

SVC中断处理

读取LR的第二个位，检查使用的堆栈是哪一个，PSP或MSP。呼叫打印函式，传入参数：堆栈指针和返回地址。

```
__asm void SVC_Handler(void)
{
    MOVS    r0, #4
    MOV     r1, LR
    TST     r0, r1
    BEQ     stacking_used_MSP
    MRS     R0, PSP         ; first parameter - stacking was using PSP
    B       get_LR_and_branch
stacking_used_MSP
    MRS     R0, MSP         ; first parameter - stacking was using MSP
get_LR_and_branch
    MOV     R1, LR          ; second parameter is LR current value
    LDR     R2,=__cpp(dump_handler_c)
    BX      R2
    NOP
    BX      LR
}
```

打印PC值

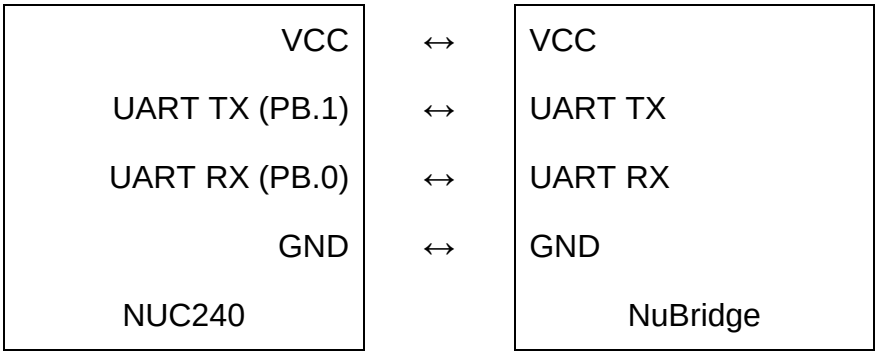
从堆栈里读取R0~R3，R12，LR，PC和xPSR数值，并打印出来。

```
void dump_handler_c(unsigned int * hardfault_args, unsigned lr_value)
{
    static volatile unsigned int stacked_r0;
    static volatile unsigned int stacked_r1;
    static volatile unsigned int stacked_r2;
    static volatile unsigned int stacked_r3;
    static volatile unsigned int stacked_r12;
    static volatile unsigned int stacked_lr;
    static volatile unsigned int stacked_pc;
    static volatile unsigned int stacked_psr;

    stacked_r0 = ((unsigned long) hardfault_args[0]);
    stacked_r1 = ((unsigned long) hardfault_args[1]);
    stacked_r2 = ((unsigned long) hardfault_args[2]);
    stacked_r3 = ((unsigned long) hardfault_args[3]);
    stacked_r12 = ((unsigned long) hardfault_args[4]);
    stacked_lr = ((unsigned long) hardfault_args[5]);
    stacked_pc = ((unsigned long) hardfault_args[6]);
    stacked_psr = ((unsigned long) hardfault_args[7]);

    printf ("R0 = %x\n", stacked_r0);
    printf ("R1 = %x\n", stacked_r1);
    printf ("R2 = %x\n", stacked_r2);
    printf ("R3 = %x\n", stacked_r3);
    printf ("R12 = %x\n", stacked_r12);
    printf ("Stacked LR = %x\n", stacked_lr);
    printf ("Stacked PC = %x\n", stacked_pc);
    printf ("Stacked PSR = %x\n", stacked_psr);
    printf ("Current LR = %x\n", lr_value);
}
```

3 软件与硬件环境

- 软件环境
  - BSP 版本
    - NUC230_240 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
    - NuBridge
  - 示意图

NUC240开发版连接NuBridge，用来打印讯息。



4 目录信息

- EC_NUC240_READ_PC_BY_SVC_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
    - 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_READ_PC_BY_SVC.uvproj。
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
Jul. 31, 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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.