## UCOSII在SWORD-MIPS上的移植指南

Table of Contents

[UCOSII在SWORD-MIPS上的移植指南 1](#_Toc458759340)

[1. 代码目录: 1](#_Toc458759341)

[2. 硬件特性 4](#_Toc458759342)

[3. 工具指导 5](#_Toc458759343)

[4. 注意事项 5](#_Toc458759344)

### 1. 代码目录:

#### 1.1 概览

|  |
| --- |
| styx\_uvga/ (硬件部分：vivado工程)  ucosii移植代码/ (软件部分：ucosii移植)  include/ (头文件)  common/ (应用程序部分)  port/ (与SWORD-FPGA紧密相关的汇编代码，详见后文)  ucos/ (ucos的核心代码，实现任务调度等功能)  Makefile (编译指导文件)  config.mk (编译指导文件)  ram.ld (链接脚本，与内存地址分配相关)  ucosii.bin (用于下载的mips二进制机器代码)  ucosii.asm (ucosii.bin的反汇编内容)  ucosii.om (ucosii.bin的ELF文件格式)  zread.c (串口读入程序的C程序，zread是相应的可执行文件)  zsend.c (将ucosii.bin通过发送到fpga板的C程序)  (发送完毕后读入串口输出，zsend是其可执行文件)  ……  styxcpu\_soc.bit (fpga下载二进制文件) |

#### 1.2 include 文件夹

|  |
| --- |
| include/  includes.h (include其他头文件)  os\_cpu.h (C语言类型定义，函数原型声明)  app.h (与app.c对应)  (不同系统可能要相应修改系统时钟频率，串口波特率)  …… |

#### 1.3 common 文件夹

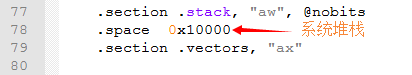
|  |
| --- |
| common/ (应用程序部分)  app.c (应用程序代码，可以在这里编写库函数等)  app.o (编译生成的OBJS文件)  common.o (编译生成的LIB文件)  Makefile (编译指导文件)  .depend (make自动生成的文件，第一次make的时候应该**删除**) |

#### 1.4 port 文件夹

a) **cpu\_a.s**

包括CPU\_SR\_Save和CPU\_SR\_Restore两个汇编函数，分别用于进入临界区域（关中断）和退出临界区域（恢复状态寄存器，开中断）

b) **os\_cpu\_a.S**

可以在这个文件中修改ucosii的堆栈长度：

该文件中包含以下函数:

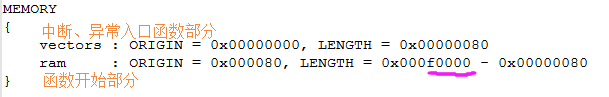
* OSStartHighRdy：切换到ready队列中最高的优先级的任务执行，直接进行弹栈操作，被调用时相当于函数的另一个出口
* OSIntCtxSw：在中断服务程序结束以后进行任务切换，直接进行弹栈操作，被调用时相当于函数的另一个出口
* OS\_CPU\_SR\_Save：关闭中断，进入临界区域
* OS\_CPU\_SR\_Restore：打开中断
* InterruptHandler：调用 BSP\_Interrupt\_Handler 依次处理多个中断，直到清空所有中断挂起标志，然后调用OSIntExit函数退出中断
* ExceptionHandler：调用 BSP\_Exception\_Handler 处理异常
* TickInterruptClear：清除时钟中断
* CoreTmrInit：设置时钟中断的周期
* TickISR：时钟中断服务程序，调用OSTimeTick通知操作系统发生了一次时钟中断
* DisableInterruptSource：用于关闭 src\_nbr($4) 的硬件中断
* EnableInterruptSource：打开src\_nbr($4)硬件中断

**c) os\_cpu\_c.c**

* OSTaskStkInit：初始化任务堆栈
* BSP\_Interrupt\_Handler：实际的中断处理函数，目前只是简单地调用了TickISR来处理时钟中断
* BSP\_Exception\_Handler：目前对所有的异常都只是简单地调用OSIntCtxSw做任务切换

#### 1.5 ram.ld

此文件为链接脚本，可以针对系统内存大小修改ram大小



#### 1.6 common/app.c

a) 此文件包含了这个工程的主函数，主函数创建了两个进程Task1和Task2，使用两个全局数组作为任务的堆栈：Task1Stk 和 Task2Stk

b) 该文件还包含初始化uart的函数：uart\_init()，目前这个函数关闭了中断，只能发送不能接受。封装了uart\_putc和uart\_print\_str两个函数用于打印

#### 1.7 zsend.c

将styxfpga\_soc.bit下载到SWORD开发板上以后，开发板将运行一个简单的bootloader: 从串口读入指令和数据并写到内存（从0x0000开始， 波特率为9600）读到0xFFFF。而zsend则是一个简单地C语言程序，它将打开串口，然后：

1. 如果没有提供命令行参数的话，那么zsend将尝试打开inst\_rom.bin文件并将其中的内容发送到SWORD板上。

2. 如果提供了命令行参数的话，那么zsend将第一个参数作为文件名，并尝试打开该文件，然后发送到SWORD板上。

3. **注意：**发送的文件（inst\_rom.bin中不应该包含0xFFFF，否则SWORD板将提前结束接收过程）。文件发送完毕以后，zsend程序会自动发送0xFFFF，从而使SWORD结束接受，并跳转到0x0000 0000开始运行。

另外，读写串口需要相应的权限，在安装好串口转usb的驱动以后，在ubuntu 操作系统上，可以尝试以下命令修改该设备的权限:

sudo chown 666 /dev/ttyUSB0

### 2. 硬件特性

#### 2.1 硬件参数

|  |  |
| --- | --- |
| 属性 | 参数 |
| CPU频率 | 25MHz快速时钟 OR 4Hz 慢速时钟 |
| SRAM大小 | 4M |
| UART波特率 | 9600（可配置） |
| UART软核版本 | 16550A |
| Wishbone总线 | 版本: B2，交叉互联型 |
| 字节序 | 大端 |
| 延时槽 | 有 |

#### 2.2 地址映射

本系统物理地址映射说明如下：

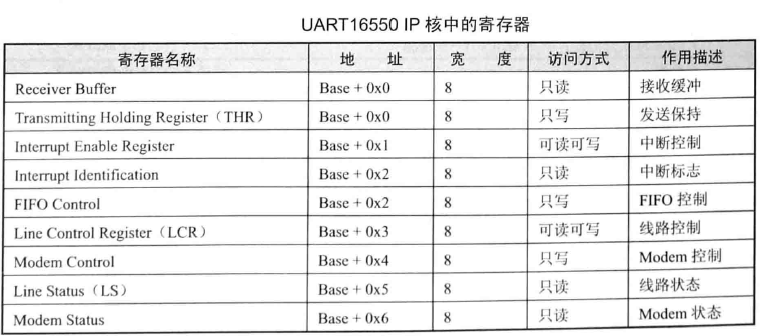
* **0x0000 0000 ~ 0x0FFF FFFF**: SRAM空间，实际可用只有4M

复位地址：0x0000 0000

0x0000 0020 硬件中断处理程序入口地址

0x0000 0040 异常处理程序入口地址

* **0x1000 0000 ~ 0x1FFF FFFF**: UART寄存器访问。由于使用了Opencores提供的IP核，所以地址映射遵循以下规范。



* **0x2000 0000 ~ 0x2FFF FFFF**

留作GPIO使用（包括VRAM和键盘），尚未实现

* **0x3000 0000~0x3FFF FFFF**

ROM 空间，不可写，本系统的bootloader，执行从串口拷贝代码到SRAM的功能。0x3000 0000 为上电后第一条指令的执行地址。

* **0x4000 0000 ~ 0xFFFF FFFF**

没有使用

* 特别注意，由于没有MMU，访问非法地址可能会导致Wishbone总线无应答，最终使CPU卡死。

#### 2.3 按钮功能

|  |  |
| --- | --- |
| SW【0】 | reset 系统复位 |
| SW【1】 | **0为快速时钟，1为慢速时钟** |
| SW【4:2】 | **100-111: 显示时钟计数，000: 显示PC**  **001: 显示代码，010: 显示CPU读取到的数据**  **011：显示CPU输出的数据** |

#### 2.4 支持指令

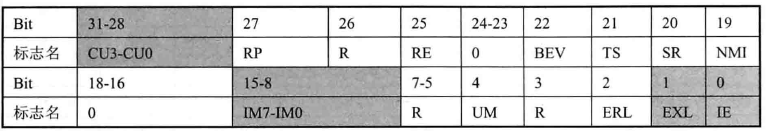
支持MIPS所有的整数指令：

* 逻辑操作指令AND、ANDI、LUI、NOR、OR、ORI、XOR、XORI
* 移位操作指令SLL、SLLV、SRA、SRAV、SRL、SRLV
* 算术操作指令ADD、ADDI、ADDIU、ADDU、CLO、CLZ、SLT、SLTI、SLTIU、SLTU、SUB、SUBU、MADD、MADDU、MSUB、MSUBU、MUL、MULT、MULTU、DIV、DIVU
* 移动操作指令MFHI、MFLO、MOVN、MOVZ、MTHI、MTLO
* 控制指令NOP、SSNOP
* 跳转指令J、JAL、JALR、JR
* 分支指令B、BAL、BEQ、BGEZ、BGEZAL、BGTZ、BLEZ、BLTZ、BLTZAL、BNE
* 加载类指令LB、LBU、LH、LHU、LL、LW、LWL、LWR
* 存储类指令SB、SC、SH、SW、SWL、SWR
* 协处理器访问指令MFC0、MTC0
* 自陷指令SYSCALL、TEQ、TEQI、TGE、TGEI、TGEIU、TGEU、TLT、TLTI、TLTIU、TLTU、TNE、TNEI
* 异常返回指令ERET
* 空指令有NOP，SSNOP，SYNC，PREF（全都实现为空指令）

#### 2.5 协处理器

目前支持的协处理器寄存器有：COUNT、COMPARE、STATUS、CAUSE、EPC

* COUNT寄存器: 是一个不停计数的32寄存器，计数频率和CPU相同。当达到32位无符号数的上限时，会重新从0开始。该可读，可写。
* Compare寄存器: 用于完成定时中断，当Count==Compare时，产生定时中断，直到有数据被写入Compare寄存器。该寄存器可读可写
* Status寄存器也是32位可读可写的。用于显示CPU状态，其中字段如下图所示：



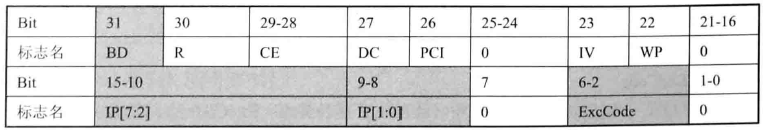
目前，我们实现了以下字段：

**CU3-CU0：**设置为4’b1，表示只有cp0可用

**IM7-IM0 :** 表示是否屏蔽相应中断，0表示屏蔽，1表示不屏蔽。处理器是否响应中断由Status寄存器和Cause寄存器共同决定，只有当Status寄存器的IM字段和Cause寄存器的IP相应位都为1，而且Staus寄存器的IE字段也为1时，处理器才响应相应中断。

**IM：**全局中断使能标志位。1表示中断使能，0表示禁止。

* Cause寄存器： 用于记录最近一次异常发生的原因，其各字段如下所示：



**BD：**当发生异常的指令处于分支延时槽时，该字段被置为1。

**IP：**中断挂起字段，用于表示中断是否发生。0-1bit为软件中断，2-7bit为硬件中断，包括：0号：定时中断 和 1号：UART

**ExcCode：**表示异常的类型

0： Int 中断

8： Sys 系统调用

10：RI 未定义指令

12：Ov 整数运算溢出

13：Tr 自陷指令

* EPC寄存器：存储异常返回地址

### 3. 工具指导

#### 3.1 软件环境

a) 硬件fpga工程：

Window7 + Vivado 2014.3

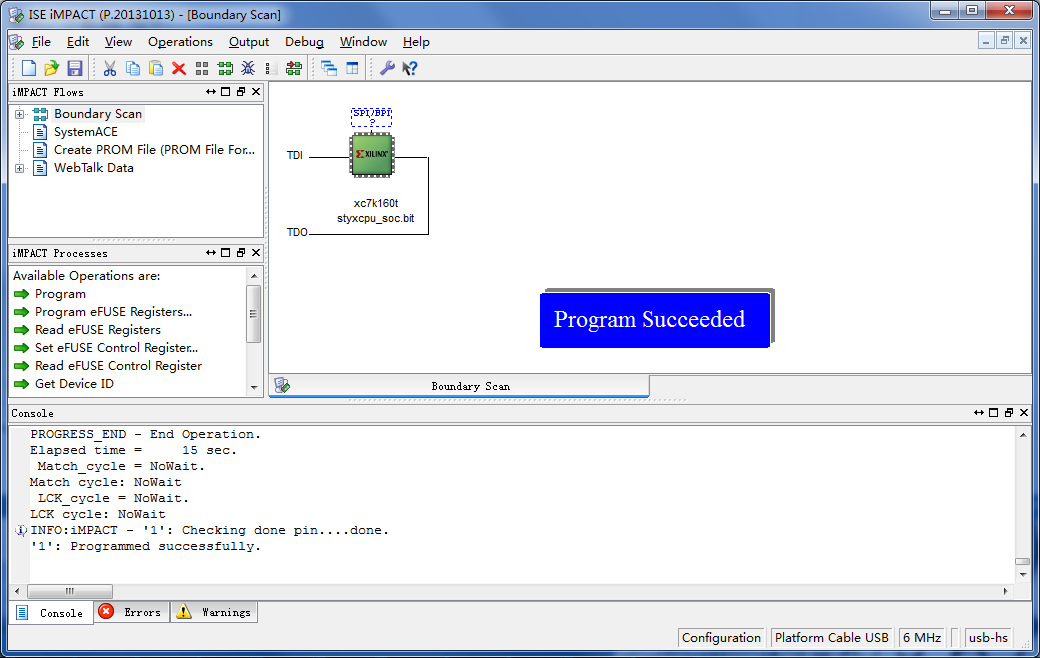
b) 软件环境：

linux内核版本：Linux kernel 3.19.0-32-generic

交叉编译器版本：mips-sde-elf-gcc 4.3.2（经测试ubuntu16.04不能直接运行该交叉编译器）

#### 3.2 分步指导

1. 在windows系统上下载.bit程序



2. 在linux上运行zsend（在这里它的程序名为serial）



3. 等串口发送完毕，SWORD将从0x0000 0000开始运行程序

### 4. 注意事项

如果在C代码中使用了除法的话，需要注意编译器会自动在除法指令后加上一些指令检查除数是否是0，如果是0就会引起一个异常。