Finsh for Nios II

目录

Author: aozima
2011-3-4
http://www.rt-thread.org

1.创建一个基本内核工程

finsh 基于 RT-Thread 内核,因些在使用 finsh 前,应该创建一个基本内核的工程,并保证其正常运行.具体方法和细节请参考: 基本内核工程_for_Nios_II

2.添加 finsh 源文件

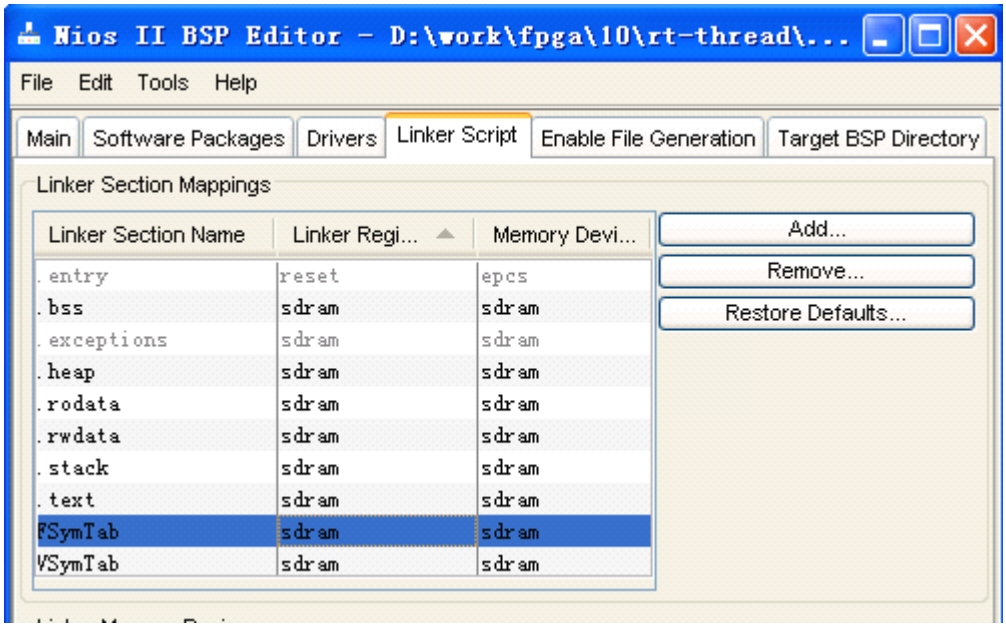
finsh 源代码在 RT-Thread 发布包的 components\finsh 目录下,建议工程目录结构如下:

```
|工程所在目录
└──rt-thread
    ├──components
    |    └──finsh
    ├──include
    ├──libcpu
    |    └──nios
    |         └──nios_ii
    └──src
```

3.修改 Linker script 添加 FSymTab 和 VSymTab 两个 section

finsh 需要两个独立的 section 用于保存函数和变量列表.因此需要修改链接脚本.Nios II IDE 封装了链接脚本管理,但我们可以使用 BSP Editer 来实现自定义操作(9.1 版本以后才有).

菜单 NIOS II --> BSP Editer 启动 BSP Editer,然后在 linker script 界面下添加 section.



根据硬件实现存储器的配置.为新加的两个 section 选择合适的目标存放.然后执行 generate
生成新的链接脚本.

4.修改 rtconfilg.h

要使用 finsh 组件,需要在 rtconfig.h 里面打开
#define RT_USING_FINSH

在一般可以自定义修改链接脚本的系统中,我们还会为 FSymTab 和 VSymTab 两个 section 添加始末地址变量.用于应用程序获取 FSymTab 和 VSymTab 两个 section 的位置和大小.
NIOS II IDE,封装了这一操作,为我们自动生成了
_alt_partition_**_start _alt_partition_**_end 这样的变量.
因此,我们还需要修改 finsh 中原来使用的变量名称:

```
#define __fsymtab_start    _alt_partition_FSymTab_start
#define __fsymtab_end      _alt_partition_FSymTab_end
#define __vsymtab_start    _alt_partition_VSymTab_start
#define __vsymtab_end      _alt_partition_VSymTab_end
```

5. 编译,调试,运行

finsh 主要注意的是 finsh 使用的输入输出的设备函数程序的编写,默认 RT-Thread 发布包中使用的是串口,一般针对具体的开发板小做修改即可正常使用.