NiosII 嵌入式系统软件设计

学习笔记（一）：

NiosII Software Build Tools 概述

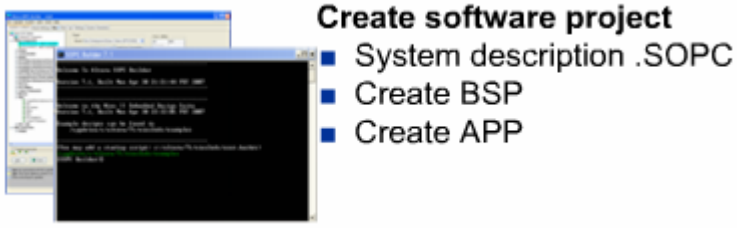
作者：**chactor**

一、 软件工程（Software Project）的构成

- **APP**
  - 源文件
    - ✓ C 源文件：.c
    - ✓ C++源文件：.cpp .cxx .cc
    - ✓ 汇编源文件： .s .S
  - 头文件（.h）
- **BSP（system library）**
  - 系统头文件
    - ✓ system.h
  - 操作系统
    - ✓ HAL
    - ✓ USOSII
  - 软件包
    - ✓ 文件系统（Host File System，Read-Only Zip File System）
    - ✓ 网络服务（NicheStack TCP/IP Stack）
    - ✓ 图形库
  - 设备驱动
    - ✓ Altera_Avalon_UART
    - ✓ Altera_Avalon_SPI
    - ✓ Altera_Avalon_timer
  - Boot Loader
  - 链接文件
    - ✓ link.x
    - ✓ link.h
  - Newlib（ANSI C Library）
- **User Library**

二、 Software Build Tools 开发流程

- **创建软件工程**

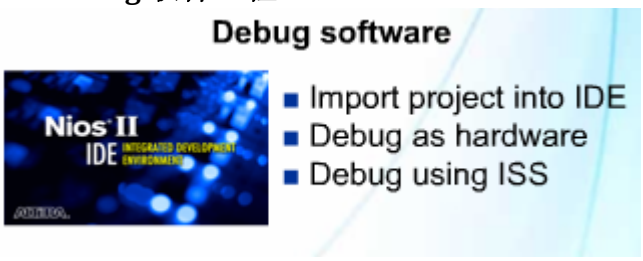


不同与 NiosII IDE 从.ptf 文件中获取所有的硬件信息，Software Build Tools 从.SOPC 文件中获取硬件信息。在 QuartusII 工程的文件夹中（即.SOPC 所在目录中）新建 software 文件夹，在该文件夹下新建两个空文件夹：APP，BSP。在 APP 文件夹下粘贴 Application Project 源文件以及所需头文件。

- **编译软件工程**

- 运行 nios2-bsp 命令创建 BSP Project，产生 Makefile 文件，再运行 make 命令 Build BSP project。
- 运行 nios2-app-generate-makefile 命令创建 Application project 以及 makefile，运行 make 编译整个工程，产生.elf 文件一共下载。

- **Debug 软件工程**



- 将 Software Build Tools 中创建的工程导入 NiosII IDE
- 在 NiosII IDE 中进行调试，下载，如果要重新编译，需要返回第二步重新 make。

- **下载程序**

在 NiosII IDE 中调用 flash programmer 将.elf 文件转化为.flash 文件便下载到 flash 或 EPCS 器件中。在下载.flash 文件之前需要先下载硬件镜像（hardware image），不然下载的软件镜像会被破坏。第二种下载方式是通过 Software Build Tools，命令行 nios2-configure-sof dir/*.sof 下载硬件镜像，命令行 nios2-download –g dir/*.elf 下载软件镜像。

三、BSP 的创建（creation），配置（configure）及编译（make）

- **nios2-bsp 命令详解**

Software Build Tools 通过运行 nios2-bsp 命令来创建，配置 BSP project。nios2-bsp 通过调用 nios2-bsp-create-setting 或者 nios2-bsp-update-setting 来创建或更新 bsp 设置文件，调用 nios2-bsp-generate-files 来创建 bsp 文件。

nios2-bsp 命令解析：

格式：nios2-bsp <bsp-typr> <bsp-dir> <sopc> --<option>

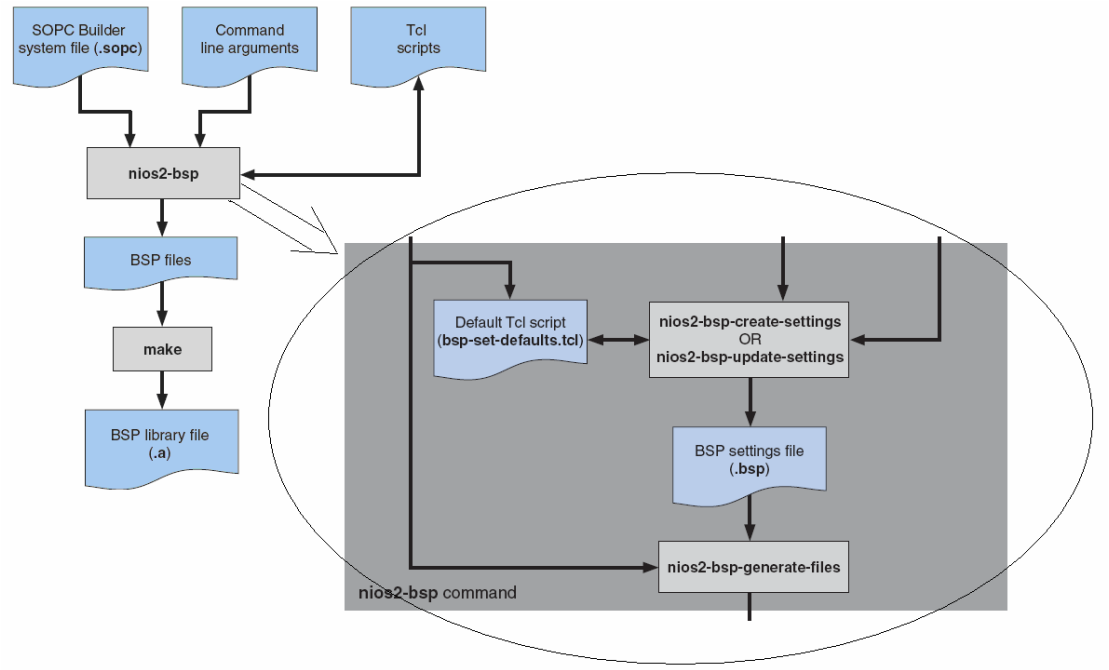
- bsp-type：bsp 类型，HAL 或者 UCOSII，必选
- sopc：.sopc 文件地址，如果省略则为当前目录
- bsp-dir：bsp 生成文件目标地址，一个“.”代表目标地址为当前目录，必选
- option：可选的 bsp 配置
  - ✓ 默认配置：如果 option 选项省略，则调用默认配置，默认配置文件在<Nios II

EDS install path>/sdk2/bin/bsp-set-defaults.tcl 中，默认配置中对 stdio 字符设备，系统时钟设备，链接存储器区域，链接 section 区域，boot loader 设置进行了默认设置。

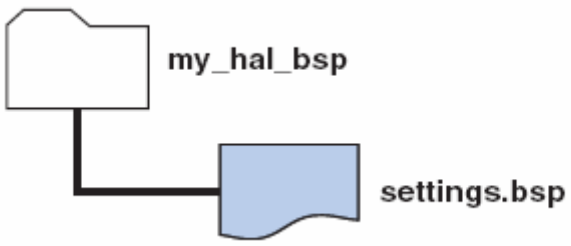
- ✓ 添加软件包：通过 enable_sw_package 命令使能相应的软件包，并将该软件包相应的源文件和设置加入 BSP 中，比如：enable_sw_package altera_hostfs
- ✓ 添加软件设置：可以在命令行中直接通过 –cmd <option>调用相应的软件设置，也可以将所有设置写在一个 tcl script（.tcl）文件中，通过 –script <*.tcl> 调用

option 中的各种配置命令可以参考 NiosII Software develop's handbook 的十四章：NiosII Software Build tools Reference。

- **BSP 创建编译流程**



运行 nios2-bsp 前，假设已经存在一个 my_hal_bsp 的空文件夹，用来生成 BSP project，运行 nios2-bsp 时，nios2-bsp 首先调用 nios2-bsp-creat-settings 或 nios2-bsp-update-settings，产生 settting.bsp 文件。



setting.bsp

setting.bsp 文件中包含了所有的 BSP 设置命令，该文件有 nios2-bsp-creat-settings 命令产生，可以有 nios2-bsp-update-settings 命令更新。nios2-bsp-query-settings 可以查询 setting.bsp 文件的内容。该文件有 nios2-bsp-generate-files 调用产生其他文件。

nios2-bsp 再调用 nios2-bsp-generate-files 产生如下文件：

summary.html

summary.html 概括了 BSP project 的基本信息

Makefile

Makefile 文件用于编译 BSP project。有 make 命令调用 makefile 来执行编译任务。

public.mk

public.mk 文件是 makefile 文件的一部分，提供了 makefile 中公有的部分，供其他 makefile 文件调用，比如 application makefile。

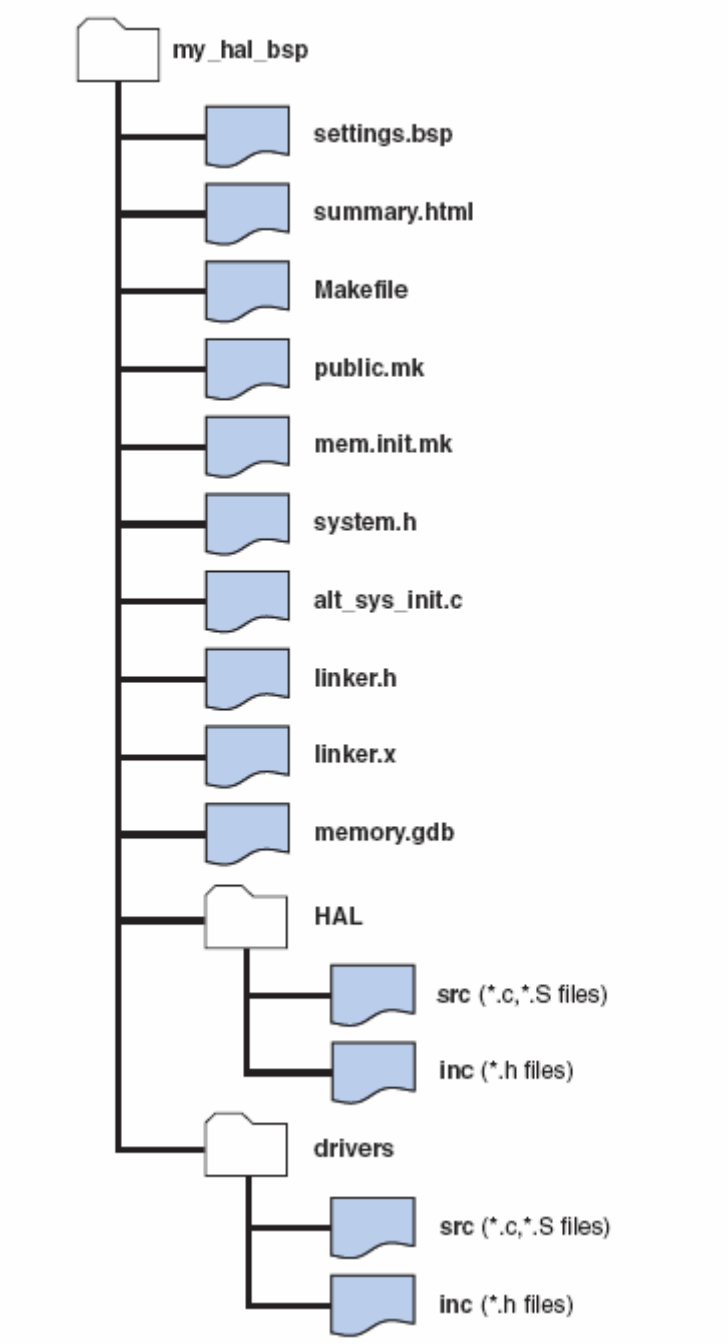


图 nios2-bsp 运行后的目录

mem_init.mk

mem_init.mk 文件也是 makefile 的一部分，用于将应用可执行文件转化为存储器初始化文件（.dat，.hex 以及.flash），应用工程的 makefile 文件将包含该文件。

alt_sys_init.c

alt-sys_init.c 文件用于初始化设备驱动实例和软件包。

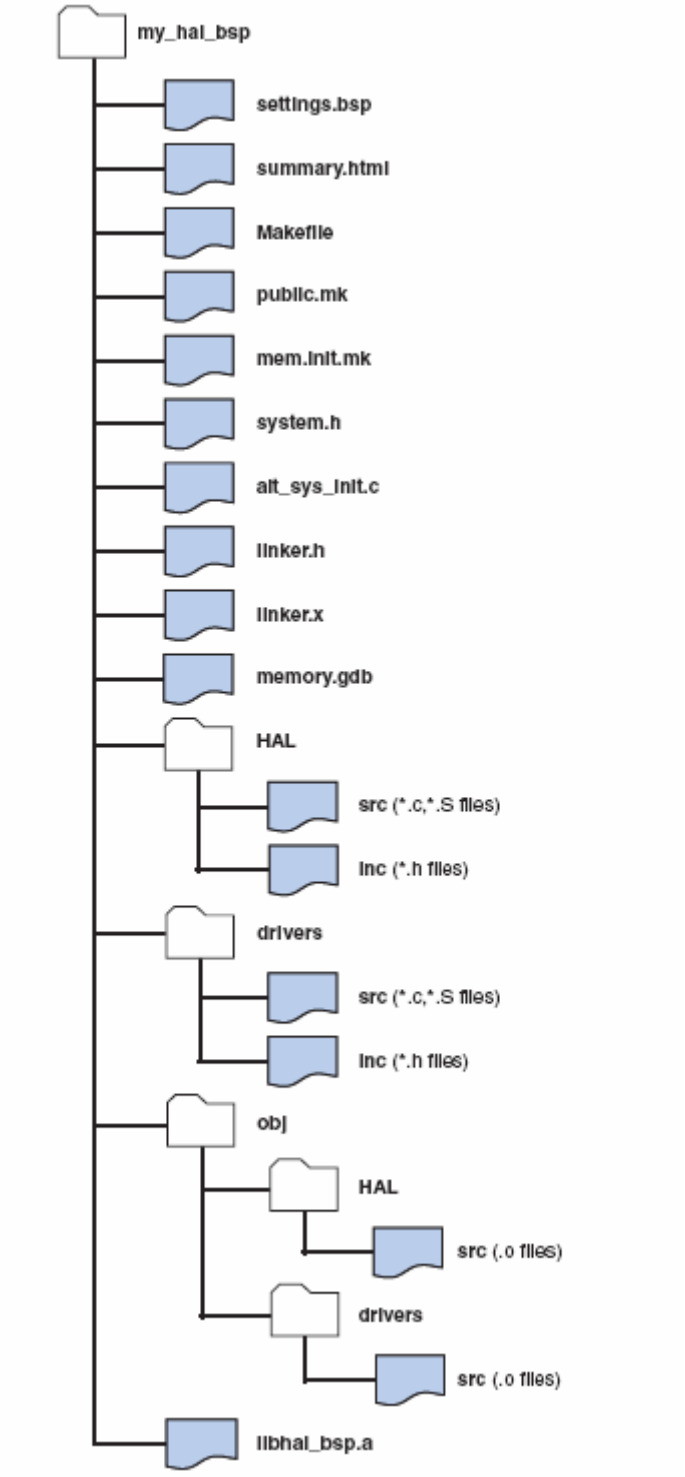


图 make 运行后的目录

system.h

system.h 文件有.sopc 文件和 BSP 设置文件生成，包括存储器映射和各种系统信息。

link.h

link.h 文件包含链接存储器布局信息，system.h 文件包含 link.h。

link.x

link.x 为 GNU 链接器提供链接脚本。与 NiosII IDE 中的 generated.x 文件相同。

memory.gdb

memory.gdb 为 GNU 调试器提供存储器区域声明。与 NiosII IDE 中的 memory.gdb 相同。

以上文件均为 software Build tools 产生的文件。

HAL 文件夹

HAL 文件夹包含了 HAL 源文件，这些文件都是从软件安装目录下拷贝的文件。src 目录下包含了 c 语言和汇编语言源文件，inc 目录下包含了头文件。Ctr0.S 文件为 HAL 启动代码，也在 src 目录下。

driver 文件夹

driver 文件夹包含所有的驱动源文件，这些文件也是从其他目录拷贝的，该目录下也有 src 和 inc 两个子文件

obj 文件夹

obj 文件夹下包含所有 BSP 源文件的目标代码文件。

libhal_bsp.a 文件夹

libhal_bsp.a 包含了 HAL BSP 库。

四、APP 的创建（creation）配置（configure）及编译（make）

- **nios2-app-genegate-makefile 命令详解**

nios2-app-genegate-makefile 命令创建及配置 APP Project，产生 Makefile 文件让 make 命令调用。

nios2-app-genegate-makefile 命令解析

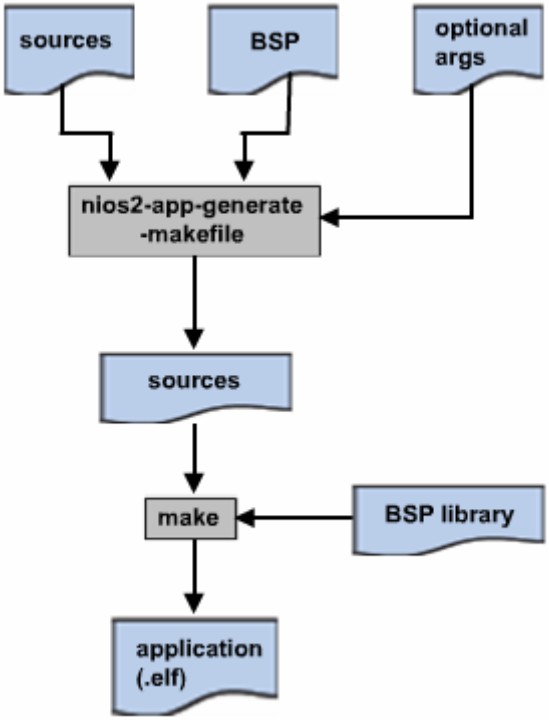
格式：nios2-app-generate-makefile [--app-dir <directory>] --bsp-dir <directory> [--c2h] [--debug] [--elf-name <filename>] [--extended-help] [--help] [--log <filename>] [--set <name value>] [--silent] [--src-dir<directory>] [--src-files <filenames>] [--src-rdir<directory>] [--use-lib-dir <directory>] [--verbose] [--version]

- --app-dir <directory>：makefile 文件和 elf 文件目标地址，省略则为当前地址
- --bsp-dir <directory>：bsp project 目录
- --c2h：使能 C2H，将一个静态的 C2H makefile 包含进 application makefile，并拷贝一个空的 c2h.mk 到 makefile 目录。
- --debug：在 stdout 中输出调试，exception traces, verbose 以及默认信息。
- --elf-name <filename>：被创建的可执行文件（.elf）的文件名，如果省略，则为第一个源文件的文件名。
- --extended-help：显示该命令的所有信息。
- --help：显示该命令的基本信息。
- --log <filename>：创建 log 文件。
- --set <name value>：将 makefile 文件中名为<name>的变量的值设置为<value>，如果该变量存在 makefile 中，则<value>值代替默认值，如果变量不存在，该变量被添加，支持多个--set <name value>。
- --silent：stdout 输出缩减的命令行执行信息。
- --src-files <filenames>：需要编译的源文件名，多个源文件用空格隔开。
- --src-rdir <directory>：与--src-files <filenames>功能相同，不过其同时在该目录下查

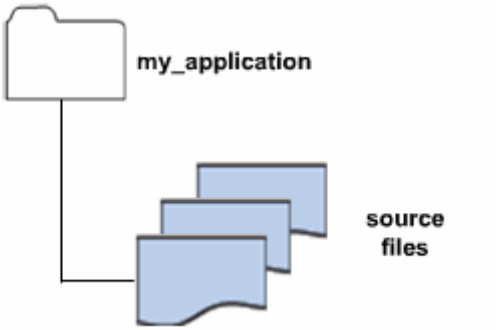
找源文件，支持多个--src-rdir <directory>一起使用，也可以和--src-files <filenames>混合使用。

- --use-lib-dir <directory>：library 文件夹路径，该路径需包含 makefile 文件：public.mk，支持多个--use-lib-dir <directory>一起使用。
- --verbose：stdout 中详细输出命令行执行信息。
- --version：显示该命令的版本。

- **APP 创建编译流程**

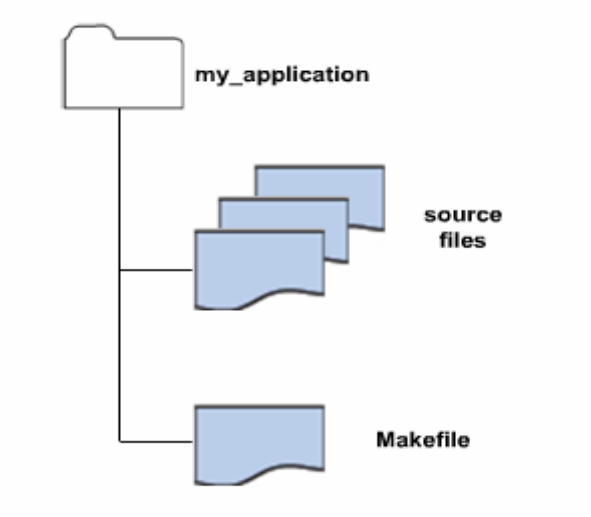


运行 nios2-app-generate-makefile 前在 BSP 文件夹同一个根目录下创建 APP 文件夹，名为 my_application，在该文件夹下放置应用工程的源文件以及头文件，如下所示：



运行 nios2-app-generate-makefile 后，产生 makefile 文件，如下图所示。

再通过 make 编译整个工程。


my_application
source
files
Makefile