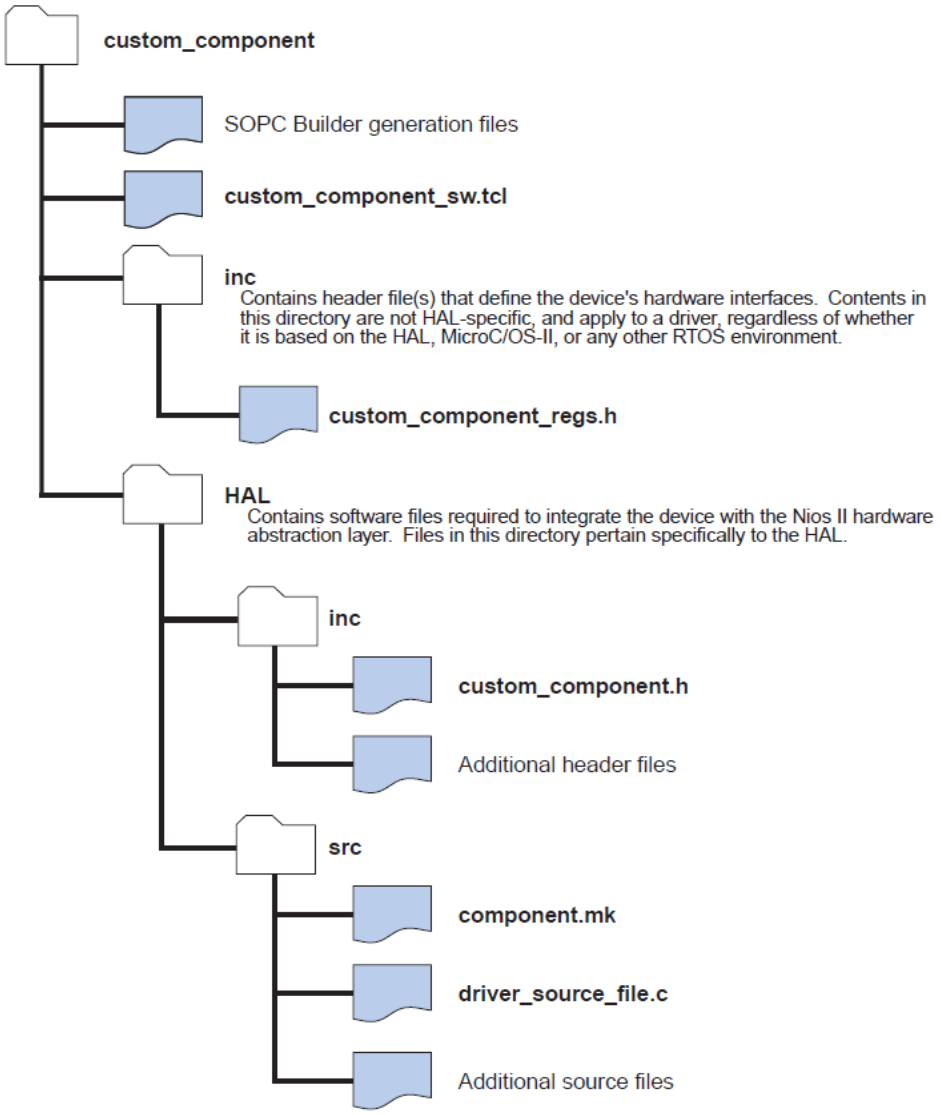
Создание Tcl скрипта для драйвера и пакета программ

В этой секции обсуждается написание Tcl скрипта для описания вашего пакета программ или драйвера. Содержимое Tcl скрипта зависит от структуры и сложности вашего драйвера или программы. Для большинства простых драйверов устройств, нам необходимо всего лишь написать несколько команд. Для более сложных программ, Nios II SBT предлагает полноценные средства, которые предоставляют конечному пользователю BSP контроль над работой вашей программы или драйвером.

Команды Tcl и описание аргументов в этой секции являются неполными. За подробным описанием каждой команды и аргумента обратитесь к главе "Справка по Nios II SBT" в настольной книге программиста под Nios II.

Для справки, при создании ваших собственных Tcl файлов драйверов или пакетов программ, вы можете также использовать Tcl скрипты драйверов и пакетов программ, содержащихся в библиотеках Nios II EDS и MegaCore IP. Эти скрипты находятся в папках <*Nios II EDS install path*>**/components** и <*MegaCore IP library install path*>**/sopc_builder_ip** соответственно.

Figure 7–1. Example Device Driver File Hierarchy and Naming



Tcl команды сквозного контроля для типового драйвера или пакета программ

Скрипты Tcl в этой секции описывают обычный драйвер устройства или пакет программ.

В примере в этой секции создаётся драйвер устройства для аппаратной периферии, являющейся компонентом SOPC Builder класса имени my_custom_component. Драйвер поддерживает BSP типов HAL и MicroC/OS-II. Он имеет единственный исходный файл Си (.**c**) и два заголовочных файла Си (.**h**), организованных как на рис. 7-1.

Создание и название драйвера или пакета

Первой командой в любом Tcl скрипте драйвера или пакета программ должна быть create_driver или create_sw_package. Остальные команды могут идти в любой последовательности. Используйте соответствующую команду **create** только один раз в Tcl файле.

Выберите уникальное имя драйвера или пакета программ. Для драйверов, Altera рекомендует добавлять _**driver** к ассоциированному имени класса устройства. В следующем примере показано это обозначение:

```
create_driver my_custom_component_driver
```

Идентификация класса аппаратного компонента

Каждый драйвер должен идентифицировать класс аппаратного компонента, чей драйвер ассоциирован в команде set_sw_property с аргументом hw_class_name. В следующем примере ассоциируется драйвер с классом устройства, называемым my_custom_component:

```
set_sw_property hw_class_name my_custom_component
```

Команде set_sw_property доступны несколько типов аргументов. Каждый вызов set_sw_property устанавливает или перезаписывает свойства на значения, заданные во втором аргументе.

За дополнительной информацией о команде set_sw_property, обратитесь к главе "Справка по Nios II SBT" в настольной книге программиста под Nios II.

Аргумент hw_class_name не применяется к пакету программ.

Если вы создаёте свой собственный драйвер вместо уже существующего (например, собственный драйвер UART для компонента altera_avalon_uart), задайте имя драйвера, отличающееся от имени стандартного драйвера. Nios II SBT будет использовать ваш драйвер только, если вы зададите его явно.

За дополнительной информацией, обратитесь к главе "Справка по Nios II SBT" в настольной книге программиста под Nios II.

Выбирайте имя для вашего драйвера или пакета программ, которое не конфликтует с другими именами, поставляемых программ или IP Altera или поставляемых сторонними разработчиками, в вашей хост системе. Генератор BSP использует имя, заданное вами, чтобы видеть пакет программ или драйвер во время создания BSP. Если Nios II SBT находит несколько совместимых драйверов или пакетов программ с одним именем, он может выбрать любой из них.

Если вы планируете разместить ваш драйвер или пакет программ, Altera рекомендует создать префикс для всех имен, допустим именем вашей организации.

Установка типа BSP

Вы должны задать каждую операционную систему (или тип BSP), которую поддерживает ваш драйвер или пакет программ. Используйте команды add_sw_property с аргументом supported_bsp_type для задания каждой совместимой операционной системы. В большинстве случаев, драйвер или пакет программ поддерживает BSP типов Altera HAL (hal) и Micrium MicroC/OS-II (ucosii), как в следующем примере:

```
add_sw_property supported_bsp_type hal
add_sw_property supported_bsp_type ucosii
```

Команде add_sw_property доступны несколько типов аргументов. Каждый вызов add_sw_property добавляет конечный аргумент в свойства, заданные во втором аргументе.

Поддержка дополнительных операционных систем и типов BSP не описана в этой документации Nios II SBT.

Задание операционной системы

Некоторым драйверам и пакетам программ не требуется особенная операционная система. Однако вы можете структурировать вашу программу, чтобы предоставить другие исходные файлы, в зависимости от используемой операционной системы.

Если ваш драйвер или пакет программ имеет различные исходные файлы, пути или настройки, которые зависят от используемой операционной системы, напишите Tcl скрипт для каждого варианта драйвера или пакета программ. Каждый скрипт должен задавать один и то же пакет программ или имя драйвера в команде create_driver или create_sw_package, и одно и тоже hw_class_name в случае с драйверами устройств. Каждый скрипт должен задать только файлы, пути и прочие настройки, которые относятся к операционной системе. Во время генерации, в BSP будут добавлены только драйверы и пакеты программ, в которых задана совместимость с выбранной операционной системой (OS).

Задание исходных файлов

Используя интерфейс команд Tcl, вы должны задать все исходные файлы в вашем драйвере или пакете программ, который вы хотите иметь в генерированном BSP. Команды, обсуждаемые в этой секции, добавляют исходные файлы драйвера и задают их размещение в файловой системе и в генерированном BSP.

Команда add_sw_property с аргументами c_source и asm_source добавляет один .**c** или один Nios II ассемблерный (**.s** or .**S**) файл в ваш драйвер или пакет программ. Вы должны выразить информацию о пути к исходному файлу по отношению к корневой директории драйвера (размещению Tcl файла). Команда add_sw_property копирует исходные файлы в BSP, в которую интегрируется драйвер, используя заданную информацию о пути, и добавляет их в список исходных файлов в генерируемом BSP makefile. Когда вы собираете BSP, используя make, исходные файлы драйвера копируются следующим образом:

```
add_sw_property c_source HAL/src/my_driver.c
```

Аргумент include_source команды add_sw_property добавляет один заголовочный файл в пути, заданном для драйвера. Путь является относительным к корню драйвера. Команда add_sw_property копирует заголовочные файлы в BSP во время генерации, используя информацию о пути, заданную на стадии генерации. Она не может включить заголовочные файлы в makefile.

```
add_sw_property include_source inc/my_custom_component_regs.h
add_sw_property include_source HAL/inc/my_custom_component.h
```

Задание поддиректории

Вы можете опционально задать поддиректорию в генерированном BSP для вашего драйвера или пакета программ, используя аргумент bsp_subdirectory для команды set_sw_property. Все исходные и заголовочные файлы драйвера копируются в эту директорию, вместе с информацией о любом пути или иерархии, заданной для каждого исходного или заголовочного файла. Если bsp_subdirectory не задана, ваш драйвер или пакет программ размещается в папке **drivers** генерированного BSP. Задайте поддиректорию следующим образом:

```
set_sw_property bsp_subdirectory my_driver
```

Если путь начинается с типа BSP (т.е. HAL или UCOSII), то тип BSP удаляется и замещается значением из свойства bsp_subdirectory.

Разрешение программы инициализации

Если ваш драйвер или пакет программ используют механизм авто инициализации HAL, ваш исходный код содержит макросы INSTANCE и INIT для создания места хранилища каждого драйвера, и для вызова любой процедуры инициализации. Генерируемый файл **alt_sys_init.c** вызывает эти макросы, которые должны быть определены в заголовочном файле с именем <*hardware component class*>**.h**.

За дополнительной информацией обратитесь к секции "Предоставление макросов *INSTANCE и *INIT" на стр. 7-14.

Для поддержки такой функциональности в Nios II BSP вы должны задать аргументу auto_initialize команды set_sw_property значение true, используя следующую Tcl команду:

```
set_sw_property auto_initialize true
```

Если вы не включите этот атрибут, файл **alt_sys_init.c** не вызовет макросы INIT и INSTANCE.

Добавление путей включений

По умолчанию, генерируемые BSP **Makefile** и **public.mk** добавляют пути включений для поиска заголовочных файлов в папках **/inc** or <*BSP type*>**/inc**.

Вам может понадобиться задать директорию иерархии для заголовочного файла для логической организации вашего кода. Вы можете добавить дополнительные пути включений для вашего драйвера или пакета программ, используя аргумент include_directory команды add_sw_property следующим образом:

```
add_sw_property include_directory UCOSII/inc/protocol/h
```

Если путь начинается с типа BSP (т.е. HAL или UCOSII), то тип BSP удаляется и замещается значением из свойства bsp_subdirectory.

Дополнительные пути включений добавляют флаги предпроцессора в файл BSP **public.mk**. Эти флаги предпроцессора позволяют исходным файлам BSP, а также приложениям и файлам пользовательских библиотек, связанных с BSP, находить пути включений, пока компилируется каждый исходный файл.

Добавление дополнительного пути включений не требуется, если ваш исходный код содержит заголовочные файлы с явными именами путей. Вы можете также задать размещение заголовочных файлов с помощью директивы #include следующим образом:

#include "protocol/h/*<filename>*"

Совместимость версий

Вашему драйверу устройства или пакету программ можно дополнительно задать информацию о версии с помощью команды интерфейса Tcl. Драйверы и пакеты программ с заданной подобным образом информацией о версии получают следующую функциональность:

- Вы можете запросить специальную версию вашего драйвера или пакета программ в настройках BSP.
- Вы можете сделать обновления для вашего драйвера устройств и установить, какой драйвер будет совместим с минимальной версией класса устройства или с заданной версией класса устройства. Такая возможность очень полезна в ситуации, когда аппаратный проект неизменен, а вы предусматриваете создание обновлений программы в течение времени.

Аргумент *<version>* в каждой следующей команде, контролирующей версии, может быть строкой, состоящей из цифр и символов. Примером строки версии являются: 8.0, 5.1.1, 6.1 и 6.1sp1. Символ "."(точка) это разделительный знак. Генератор BSP сравнивает версии друг с другом для определения новейшей или равной версии, путём сравнения строк между каждым разделителем. Например, 2.1 больше чем 2.0, а 2.1sp1 больше чем 2.1. Две версии идентичны, когда версии их аргументов равны.

Используйте аргумент version команды set_sw_property, чтобы назначить версию для вашего драйвера или пакета программ. Если вы не назначите версию для вашего драйвера или пакета программ, то будет назначена версия инсталлированного Nios II EDS (будут исполнены BSP команды Nios II):

set_sw_property version 7.1

Драйверы устройств (но не пакеты программ) могут использовать аргументы min_compatible_hw_version и specific_compatible_hw_version для установки совместимости с ассоциированным с ними классом устройств следующим образом:

set_sw_property min_compatible_hw_version 5.0.1add_sw_property
specific_compatible_hw_version 6.1sp1

Вы можете добавить несколько специальных версий совместимости. Такая функциональность позволит вам развёртывать новую версию драйвера устройства, которая несёт изменения, поддерживающие изменения в аппаратной периферии.

Для драйверов устройств, если не задана информация о совместимой версии, то версия драйвера устройства должна быть эквивалентна ассоциированному классу

устройства. Однако если вы не хотите использовать это свойство, Altera рекомендует установить настройку min_compatible_hw_version вашего драйвера на самую минимальную версию ассоциированного класса устройства, с которым совместим ваш драйвер.

.