Драйвер устройств Ethernet

Групповая модель устройств HAL для устройств Ethernet предоставляет доступ к стеку TCP/IP NicheStack® - Nios II Edition, запускаемому под операционной системой

MicroC/OS-II. Вы можете предоставить доступ к новым устройствам Ethernet, связав их с функциями драйвера, определенными в этой секции.

Прежде чем приступить к написанию нового Ethernet драйвера устройства, вам необходимо понять принципы реализации стека TCP/IP NicheStack® и их использование.

За дополнительной информацией обратитесь к главе "Ethernet и стек TCP/IP NicheStack®" в настольной книге программиста Nios II.

Простейший способ написания нового Ethernet драйвера устройства – это начать написания с реализации Altera драйвера для устройства SMSC lan91c111, чтобы затем модифицировать его для вашего контроллера доступа к среде (MAC) Ethernet. В этой секции подразумевается, что вы овладеете этими методами. Начало работы с понимания рабочего примера сделает проще изучение более сложных подробностей реализации стека TCP/IP NicheStack®.

Исходный код для драйвера lan91c111 поставляется с программой Quartus II в <*Altera installation*>**/ip/altera/sopc_builder_ip/altera_avalon_lan91c111/UCOSII**. Для краткости в этой секции будем ссылаться на эту директорию как на <*SMSC path*>. Исходные файлы находятся в директориях <*SMSC path*>**/src/iniche** и <*SMSC path*>**/inc/iniche**.

Полное количество файлов стека NicheStack TCP/IP, инсталлированных вместе с Nios II Embedded Design Suite (EDS), находятся в директории <*Nios II EDS install path*>**/components/altera_iniche/UCOSII**. Для краткости в этой главе будем ссылаться на эту директорию как на <*iniche path*>.

За дополнительной информацией о реализации стека NicheStack TCP/IP, обратитесь к технической документации NicheStack, доступной на веб-сайте Altera на странице Литература: Процессор Nios II.

Вам не нужно редактировать исходный код стека NicheStack TCP/IP, чтобы реализовать совместимый драйвер с NicheStack. Тем не менее, Altera предлагает вам исходный код для справки. Эти файлы инсталлированы вместе с Nios II EDS в директории <*iniche path*>. Интерфейс драйвера устройства Ethernet определён в <*iniche path*>**/inc/alt_iniche_dev.h**.

В следующих секциях описано, как предложить драйвер для нового устройства Ethernet.

Создание процедуры аппаратного интерфейса NicheStack

Архитектуре NicheStack TCP/IP необходимо несколько процедур сетевого аппаратного интерфейса:

- инициализация устройства
- посылка пакета
- приём пакета
- закрытие
- статистика dump ("снимок памяти")

Эти процедуры полностью задокументированы в главе "Функции предлагаемые инженерам порта" в технической документации NicheStack. Соответствующие функции для SMSC lan91c111 драйвера устройства показаны в табл. 7-4.

Табл. 7-4. Процедуры аппаратного интерфейса SMSC lan91c111

Прототип функции	**Функция lan91c111**	**Файл**	**Примечание**
n_init()	s91_init()	**smsc91x.c**	Процедура инициализации может быть инсталлирована как ISR.
pkt_send()	s91_pkt_send()	**smsc91x.c**	
Механизм приёма пакета	s91_isr()	**smsc91x.c**	Приём пакета включает в себя три ключевых действия: ▪ pk_alloc() – локализация структуры netbuf ▪ putq() – размещение структуры netbuf в rcvdq ▪ SignalPktDemux() – извещение интернет протокола (IP) о том, что он смог демультиплексировать пакет
	s91_rcv()	**smsc91x.c**	
	s91_dma_rx_done()	**smsc_mem.c**	
n_close()	s91_close()	**smsc91x.c**	
n_stats()	s91_stats()	**smsc91x.c**	

Код системы стека NicheStack TCP/IP использует изнутри структуру сети для определения ей интерфейса с драйверами устройств. Структура сети определена в **net.h**, в *<iniche path>***/src/downloads/30src/h**. Кроме этого, структура сети содержит следующее:

- Поле для IP адресов интерфейса
- Указатель функции на низкоуровневую функцию для инициализации MAC устройства
- Указатели функций на низкоуровневые функции для посылки пакетов

Обычный код NicheStack ссылается на тип NET, определённый как *net.

Предоставление макросов *INSTANCE и *INIT

Чтобы разрешить HAL использовать ваш драйвер, вы должны предоставить два HAL макроса. Имена этих макросов основаны на имени вашего компонента сетевого интерфейса, в соответствии со следующими шаблонами:

- *<component name>_INSTANCE*
- *<component name>_INIT*

За примерами обратитесь к ALTERA_AVALON_LAN91C111_INSTANCE и к ALTERA_AVALON_LAN91C111_INIT в *<SMSC path>***/inc/iniche/ altera_avalon_lan91c111_iniche.h**, который включен в *<iniche path>***/inc/ altera_avalon_lan91c111.h.**

Вы можете скопировать **altera_avalon_lan91c111.h** и изменить его под свой собственный драйвер. HAL выполняет поиск макросов *INIT и *INSTANCE в файле **<***component name***>.h**, как описано в секции "Заголовочные файлы и alt_sys_init.c" на стр. 7-16. Вы можете завершить это с помощью директивы #include как в файле **altera_avalon_lan91c111.h**, или сможете определить макрос прямо в файле **<***component name***>.h.**

Ваш макрос *INSTANCE декларирует структуру данных, необходимую для элемента MAC. Эта структура данных должна содержать структуру alt_iniche_dev. Макрос *INSTANCE должен инициализировать первые три поля структуры alt_iniche_dev следующим образом:

- Первое поле, llist, для внутреннего использования, ему всегда должно быть установлено значение ALT_LLIST_ENTRY.
- Второму полю, name, должно быть установлено имя устройства, как оно определено в **system.h**. Например, **altera_avalon_lan91c111_iniche.h** использует Си оператор предпроцессора ## (конкатенация, сцепление) для ссылки на символ LAN91C111_NAME, определённый в **system.h**.
- В третьем поле, init_func, должен быть указатель на вашу функцию программной инициализации, как это описано в секции "Предоставление функции программной инициализации". Например, в файле **altera_avalon_lan91c111_iniche.h** установлен указатель на alt_avalon_lan91c111_init().

Ваш макрос *INIT инициализирует программу драйвера. Инициализация должна включать в себя вызов макроса alt_iniche_dev_reg(), определённого в **alt_iniche_dev.h**. Этот макрос регистрирует устройство в HAL, добавляя элемент драйвера в alt_iniche_dev_list.

Когда ваш драйвер включен в проект Nios II BSP, HAL автоматически инициализирует ваш драйвер, запуская макросы *INSTANCE и *INIT из функции alt_sys_init(). Обратитесь к секции "Заголовочные файлы и alt_sys_init.c" на стр. 7-16 за подробной информацией о макросах *INSTANCE и *INIT.

Предоставление функции программной инициализации

Макрос *INSTANCE() вставляет указатель на вашу функцию инициализации в структуру alt_iniche_dev, как это описано в секции "Предоставление макросов *INSTANCE и *INIT" на стр. 7-14. Ваша функция программы инициализации должна выполнять одну из трёх задач:

- Инициализировать устройство и проверять его готовность
- Заканчивать инициализацию структуры alt_iniche_dev
- Вызывать get_mac_addr()

Функция инициализации должна выполнять любую необходимую инициализацию вашего драйвера, такую как создание и инициализацию собственных структур данных и ISR (процедуры обработки прерываний).

За подробной информацией о функции get_mac_addr(), обратитесь к главе "Ethernet и стек TCP/IP NicheStack®" в настольной книге программиста Nios II.

За примерами функции программной инициализации, обратитесь к alt_avalon_lan91c111_init() в <*SMSC path*>**/src/iniche/smsc91x.c**.