Важный концепт NicheStack TCP/IP Stack

Следующие разделы могут оказать значительное влияние на ваш проект.

Обработка ошибок

Набор функций обработки ошибок определён в alt_error_handler(), он контролирует обработку ошибок в приложении Nios II Simple Socket Server, в NicheStack TCP/IP Stack и после вызова кодов ошибок системы MicroC-OS/II. Всё, система, гнездо и приложение, вызывает проверку на состояние ошибки, когда ошибка имела место быть.

Создание задачи по умолчанию в NicheStack TCP/IP Stack

NicheStack TCP/IP Stack создаёт одну и более задач системного уровня на стадии инициализации системы, когда вы вызываете функцию netmain(). Пользователи имеют полный контроль над этими задачами системного уровня через файл глобальной конфигурации **ipport.h**, расположенный в директории структуры BSP проекта в папке **iniche/src/h/nios2**.

Вы сможете отредактировать директивы #define в файле **ipport.h** для того, чтобы сконфигурировать следующие опции для NicheStack TCP/IP Stack:

- Module Inclusion (включение модулей) - определяет, какие модули NicheStack будут запущены.
- Module Priority (приоритет модулей) - определяет приоритет для задачи модуля, который будет использовать MicroC/OS-II.

- Module Stack Size (размер стека модуля) - определяет размер стека, выделяемого MicroC/OS-II, который будет использовать этот модуль.

За дополнительной информацией о других опциях NicheStack TCP/IP, которые могут быть разрешены на стадии прогона, обратитесь к документации NicheStack TCP/IP в **NicheStackRef.zip**, расположенной в директории <*Nios II EDS install path*>**\ components/altera_iniche/UCOSII/31src**.

В примере проекта Nios II Simple Socket Server для запуска сконфигурирован минимально необходимый набор задач NicheStack TCP/IP Stack:

- tk_netmain - инициализирует стек, включая сетевые интерфейсы,
- tk_nettick - задача управления временем, которое использует сетевой стек.

За дополнительной информацией об этих задачах NicheStack TCP/IP Stack обратитесь к главе "Приоритеты задач в проекте Nios II Simple Socket Server" на стр. 1-25.

Создание задач, использующих интерфейс гнезда NicheStack TCP/IP Stack

Вы должны использовать вызов функции TK_NEWTASK для создания любой задачи, использующей сетевой интерфейс NicheStack. Вы должны создавать такие задачи, которые не используют сетевые сервисы функции OSTaskCreate() в MicroC/OS-II.

NicheStack Networking Stack использует функцию TK_NEWTASK (определена в **osportco.c**) для запуска задач MicroC/OS-II, использующих сетевые сервисы. Функция TK_NEWTASK имеет единственный аргумент, struct inet_taskinfo * nettask (определён в **osport.h**), который задаёт имя задачи, очерёдность приоритета MicroC/OS-II и размер стека. Вы сможете найти эти файлы в директории**<***Nios II EDS install path***>\ components/ altera_iniche/UCOSII/src/nios2**. Структура struct inet_taskinfo определена следующим образом:

```
struct inet_taskinfo {
        TK_OBJECT_PTR(tk_ptr);/* pointer to static task object */
        char * name;/* name of task */
        TK_ENTRY_PTR(entry); /* pointer to code that starts task*/
        int priority; /* MicroC/OS-II priority of the task */
        int stacksize;/* size (bytes) of task's stack */
        char* stackbase; /* base of task's stack */
        };
```

Для каждой сетевой задачи, созданной вами в приложении, вы должны декларировать локальную структуру struct inet_taskinfo с определёнными элементами. Эти элементы представлены ниже:

- TK_OBJECT_PTR(tk_ptr) - Это указатель на статический объект задачи, определённый для данной задачи через макрос TK_OBJECT. NicheStack Networking Stack может использовать в работе элемент tk_ptr. После декларирования имени через TK_OBJECT и заполнения TK_OBJECT_PTR(tk_ptr), вам больше ничего не потребуется делать.
- char * name - Этот элемент содержит строку символов, которая соответствует имени задачи. Вы можете заполнить её любыми символами на свой вкус.

- TK_ENTRY_PTR(entry) - Этот элемент связан с точкой входа или определением имени функции задачи, которую вы хотите запустить.
- int priority - Уровень приоритета задачи в MicroC/OS-II.
- int stacksize - Размер стека под задачу в MicroC/OS-II.
- char* stackbase - Этот элемент в структуре используется программой NicheStack. Вы не должны изменять этот элемент структуры.

В дополнение к декларированию структуры struct inet_taskinfo, вы должны вызвать два макроса: TK_OBJECT и TK_ENTRY. Эти макросы используются следующим образом:

- TK_OBJECT(name) - Создаёт статический объект задачи с именем name, который использует в работе NicheStack. Статический объект задачи также задаётся в TK_OBJECT_PTR(tk_ptr). NicheStack конвертирует имя для параметра name, добавляя строку "to_" перед именем, декларированным в структуре struct inet_taskinfo.
- TK_ENTRY(name) - Используется для создания декларирования точки входа задачи или имени функции. Параметр name идентичен имени функции, заданной для задачи, которую вы хотите создать. Функция должна иметь вид: void name (void). Параметр name задаёт макрос TK_ENTRY_PTR(entry).

Для создания собственных задач в приложении, которые используют сервисы, предлагаемые NicheStack TCP/IP Stack, следуйте этим пунктам:

1. **Вызовете макросы задач** - это макросы TK_OBJECT и TK_ENTRY с информацией о вашей задаче.
2. **Определите параметры задачи** - определите программу задачи, заполнив локальную структуру inet_taskinfo в вашем коде.
3. **Ожидайте инициализации стека** - прежде чем запускать свою задачу, подождите, пока внешняя переменная iniche_net_ready установится в TRUE. Эта переменная устанавливается в FALSE на стадии прогона и изменяется на TRUE, когда начинает работать NicheStack TCP/IP Networking Stack.
4. **Запустите задачу** - вызовите TK_NEWTASK, пока передаёте указатель на структуру inet_taskinfo для вашей задачи.

Ниже приведён пример кода для создания собственной задачи для приложения:

```
// Declaration of SSSNiosIISimpleSocketServerTask
void SSSNiosIISimpleSocketServerTask(void){
// task specific code
}
// Creation of NicheStack networking task
TK_OBJECT(to_ssstask);
TK_ENTRY(SSSNiosIISimpleSocketServerTask);
struct inet_taskinfo ssstask = {
                                &to_ssstask,
```

```
                                "simple socket server",
                                SSSNiosIISimpleSocketServerTask,
                                TASK_PRIORITY,
                                APP_STACK_SIZE,
                                };
    while (!iniche_net_ready)
            TK_SLEEP(1);
    /* Create the main simple socket server task. */
    TK_NEWTASK(&ssstask);
```

Сетевые задачи могут передавать работу с длительными процессами, которые не зависят от сети, другим задачам. Эта задача загрузки сегмента имеет преимущество в увеличении контроля над использованием памяти под задачи стека, например, больший контроль над расстановкой приоритетов работы.

Избегайте чрезмерного распределения работы между несколькими одновременно запущенными задачами по следующим причинам:

- Дополнительные задачи требуют дополнительного времени исполнения процессором, чтобы сделать их контекстно-переключаемыми.
- Ограниченный набор приоритетов. Каждая задача должна иметь собственный уникальный приоритет в MicroC/OS-II, и вы не можете выходить за границы приоритетов задач.

Приоритеты задач в проекте Nios II Simple Socket Server

Приоритеты задач в программе напрямую влияют на то, как будет работать программа, если функции задачи полностью корректны. Операционная система MicroC/OS-II использует уникальную схему нумерации приоритетов для запуска своих задач, при этом, задачи, которым задан низший номер приоритета, рассматриваются как наиболее приоритетные задачи. Поскольку Altera версия NicheStack TCP/IP Stack использует в работе MicroC/OS-II RTOS, всем задачам, запущенным в системе, должен быть назначен уникальный номер приоритета. Для демо-версии программы Nios II Simple Socket Server всем задачам должны быть заданы не конфликтные приоритеты. Для вашего собственного приложения вы можете проверить наличие свободных приоритетов в системе на стадии прогона.

В табл. 1-4 перечислены задачи, которые должны быть запущены в вашей системе, а также механизм настройки приоритетов для этих задач.

Табл. 1-4. Задачи Simple Socket Server и механизм настройки

Тип задачи	**Механизм настройки**
Внутренние задачи MicroC/OS-II	Настройки **ucosii**, находятся на вкладке **Main** в Nios II BSP Editor.
Внутренние задачи NicheStack TCP/IP Stack	Файл **ipport.h**, находится в директории **iniche/src/h/nios2** вашего BSP проекта.
Задачи инициализации сети	Файл **iniche_init.c**, находится в директории *<tutorial_files>***\ nichestack_tutorial**
Сетевые задачи пользователя (вызов TK_NEWTASK)	Создаётся в коде пользовательского приложения.
Несетевые задачи пользователя (вызов OSTaskCreate)	Создаётся в коде пользовательского приложения.
Задачи мониторинга PHY (физического уровня)	Создаётся в коде пользовательского приложения.

В следующих параграфах обсуждаются приоритеты задач в проекте Nios II Simple Socket Server:

Внутренние задачи MicroC/OS-II

Приложение Nios II Simple Socket Server сконфигурировано таким образом, чтобы не использовать внутренние задачи MicroC/OS-II.

Внутренние задачи NicheStack TCP/IP Stack

Макрос TK_NETMAIN_TPRIO, определённый в файле **ipport.h**, задаёт приоритет под номером 2 для основной задачи NicheStack TCP/IP Stack, запущенной функцией netmain(). Эта задача реализует функционирование ядра NicheStack TCP/IP Stack. Чтобы увеличить скорость прохождения пакетов, приоритет этой задачи должен быть выше приоритетов приложения, которые использует NicheStack TCP/IP Networking Stack.

Макрос TK_NETTICK_TPRIO, определённый в файле **ipport.h**, задаёт приоритет под номером 3 для задачи хронометража NicheStack TCP/IP Stack, запущенной функцией netmain().NicheStack TCP/IP Stack использует эту задачу для отслеживания временных событий сетевого стека. Altera рекомендует вам задать приоритет для этой задачи на один уровень ниже, чем у TK_NETMAIN_TPRIO.

Задачи инициализации сети

Макрос SSS_INITIAL_TASK_PRIORITY задаёт приоритет 5 для первой задачи, запускаемой MicroC/OS-II. Эта задача создаёт ресурсы и все остальные задачи прежде, чем само удалиться. Эта задача получает наивысший приоритет не из-за требований к высокой частоте работы или низкой задержке, а для создания ресурсов системы реального времени и задач до того, как другие задачи начнут использовать эти ресурсы.

Сетевые задачи пользователя

Макрос SSS_NIOS_II_SIMPLE_SOCKET_SERVER_TASK_PRIORITY задаёт приоритет 10, который ниже, чем приоритет задачи LEDManagementTask(). Приоритет этой задачи приложения задан меньше, чем приоритет сервисных задач для всех программных компонентов системы. Подобная практика лучше подходит для планируемой задержки, поскольку задачи программных компонентов разрабатываются для работы в максимально возможных коротких промежутках времени.

Несетевые задачи пользователя

Макрос LED_MANAGEMENT_TASK_PRIORITY задаёт приоритет 7. Задача этой функции в приёме сообщения о команде для светодиодов от SSSNiosIISimpleSocketServerTask.

Макрос LED_BLINK_TASK_PRIORITY задаёт приоритет 11. Приоритет этой задачи приложения задан ниже, чем оставшиеся задачи системы, поскольку ей для выполнения требуется очень мало рабочих циклов процессора Nios II.

Задачи мониторинга PHY

Макрос SSS_MONITOR_PHY_TASK_PRIORITY задаёт приоритет 9. Эта задача контролирует статус единственной физической (PHY) сети и работает для поддержания сетевого подключения.

Размер стека задач

Требуемая область стека задачи зависит от конфигурации процессора Nios II, HAL, RTOS и индивидуальных программных компонентов. Быстрая эмпирическая проверка массива значений Stk[] в окне памяти Nios II SBT под Eclipse является простым способом оценки вершины стека задач. Рассмотрение массива стеков Stk[] выявляет различные значения, представляя часть, используемую стеком, и множество нулей в той части, где стек ещё не задействован. Количество нулей в области, граничащей со следующей задачей, показывает, насколько глубоко увеличился стек с момента последнего сброса системы.

Все задачи, которые сделали вызовы к библиотеке программ этапа исполнения, находятся от вершины стека до примерно 900 байтной структуры _reent. Каждая задача имеет собственную копию структуры, расположенную в стеке задач. Размер этой структуры единственный может быть уменьшен в зависимости от объёма стека.

За подробной информацией о структуре _reent, обратитесь к параграфам "Стандартная библиотека newlib ANSI C" и "Реализация MicroC/OS-II RTOS проектов на процессоре Nios II" в главе "Операционная система реального времени MicroC/OS-II" в настольной книге программиста Nios II.

Что делать дальше?

Функциональность этого примера можно просто расширить для обработки нескольких TCP соединений для одного порта. Задача SSSNiosIISimpleSocketServerTask() может быть измена для создания элементов задач отдельных соединений гнезда, для обработки нескольких telnet соединений.

Многие используют этот пример во встраиваемой системе в качестве подключения к Изернет. Для того, чтобы сделать подключение к Интернет, вы можете дополнить этот проект некоторыми функциональными узлами, такими как удалённое конфигурирование через веб-браузер, или удалённое обновление программ для коррекции или функционального расширения устройств физического уровня.

2. Дополнение A

Подробности установки аппаратных средств

Введение

Для выполнения этого учебного пособия вы должны установить Nios II SBT под Eclipse, а ваша оценочная плата Altera должна быть подключена к персональному компьютеру через Изернет и USB/ JTAG порты.

За информацией о загрузочных кабелях и драйверах, обратитесь на страницу Download Cables на сайте Altera.

Пример аппаратного проекта Nios II Ethernet Standard для оценочной платы Altera имеет устройство Изернет, необходимое для этого учебного пособия по NicheStack. Устройство Изернет в этом примере проекта уже имеет физический MAC/PHY на вашей оценочной плате Altera, и является периферией Altera Triple Speed Ethernet MAC. Настройки базового адреса периферии для примера проекта определены в файле **system.h**.

Сетевое подключение

Если вы уже используете DHCP сервер для задания IP адресов, подключите оценочную плату Altera в вашу Изернет сеть.

Если оценочная плата Altera подключена прямо к вашему персональному компьютеру с помощью Изернет кабеля, или если DHCP сервер не доступен, задайте адрес вручную в файле **niosII_simple_socket_server.h**.

IP адрес по умолчанию в файле **niosII_simple_socket_server.h** задан нулями, поэтому пакеты сервера DHCP могут проходить защиту роутеров. Если вы не используете DHCP сервер, задайте правильный статичный адрес, например, 192.168.1.234, со шлюзом 192.168.1.1 и маской подсети 255.255.255.0.

Проследите за тем, что настройка **enable_dhcp_client** соответственно включена или выключена на вкладке **Software Packages** в BSP Editor. Подробнее в параграфе "Конфигурирование BSP" на стр. 1-7.