10. MicroC/OS-II Real-Time Operating System

NII52008-11.0.0

10. Операционная система реального времени MicroC/OS-II

В этой главе описывается ядро реального времени MicroC/OS-II для встроенного процессора Nios II. Эта глава состоит из следующих секций:

Введение в MicroC/OS-II RTOS

MicroC/OS-II является популярным ядром реального времени, разработанного Micrium Inc. MicroC/OS-II является ядром: портативным, загружаемым из ROM, расширяемым, с приоритетными прерываниями, реального времени, многозадачным. Первое издание MicroC/OS-II 1992 года использовалось в сотнях коммерческих приложений. Оно поддерживает более сорока различных архитектур процессоров, а также процессор Nios II.

MicroC/OS-II предлагает следующие сервисы:

- Задачи (потоки)
- Флаги событий
- Передача сообщений
- Менеджер памяти
- Семафор (одно из средств синхронизации в многопоточных процессах, используемое для управления доступом к разделяемыми ресурсами)
- Менеджер времени

Ядро MicroC/OS-II работает над уровнем аппаратной абстракции (HAL) пакета поддержки платы (BSP) в процессоре Nios II. Используя подобную архитектуру, разработка на MicroC/OS-II для процессора Nios II имеет следующие преимущества:

- Программы могут переноситься на другие аппаратные системы Nios II.
- Программы не восприимчивы в изменениям в аппаратной части.
- Программы имеют доступ ко всем HAL сервисам, вызывая HAL API в стиле UNIX.
- Простота реализации программ обработки прерываний (ISR).

Дополнительная информация

В этой главе обсуждаются детали использования MicroC/OS-II только для процессора Nios II. Полные возможности MicroC/OS-II находятся в книге "MicroC/OS-II - ядро реального времени" Жана Лаброссе (Jean J. Labrosse) (CMP Books). Вы сможете получить дополнительную информацию на сайте Micrium (www.micrium.com).

Лицензирование

Altera предлагает MicroC/OS-II в Nios II EDS только в качестве ознакомления. Если вы планируете использовать MicroC/OS-II в коммерческом изделии, вы должны получить лицензию от Micrium (www.micrium.com).

Micrium предлагает свободную лицензию для университетов и студентов. Свяжитесь с Micrium для уточнения деталей.

Другие поставщики RTOS

Altera предлагает MicroC/OS-II, чтобы предоставить вам прямой доступ к простой в использовании RTOS. В дополнение к MicroC/OS-II некоторые разработчики предлагают другие RTOS. За подробной информацией о поддержке процессором Nios II других RTOS обратитесь к странице Embedded Software на сайте Altera.

Nios II вариант реализации MicroC/OS-II RTOS

Altera реализовала MicroC/OS-II для процессора Nios II. Altera предлагает MicroC/OS-II в Nios II EDS и поддерживает реализацию Nios II для ядра MicroC/OS-II. Законченные и готовые к использованию примеры программ на MicroC/OS-II уже инсталлированы в Nios II EDS. В дополнение, отладочные платы Altera имеют предустановленный проект веб сервера на базе MicroC/OS-II и NicheStack® TCP/IP Stack - Nios II Edition.

Altera версия MicroC/OS-II разработана для простоты в использовании. Используя настройки проекта Nios II, вы сможете контролировать конфигурацию всех модулей RTOS. Вам не потребуется модификация исходных файлов для прямого указания использовать или запретить средства ядра.

Altera также предлагает специфические исходные коды для процессора Nios II на тот случай, если вы захотите их изучить. Исходные коды MicroC/OS-II находятся в следующих директориях:

- Специфический код процессора: *<Nios II EDS путь установки>***/ components/ altera_nios2/ UCOSII**
- Независимый от процессора код: *<Nios II EDS путь установки>***/ components/ micrium_uc_osii**

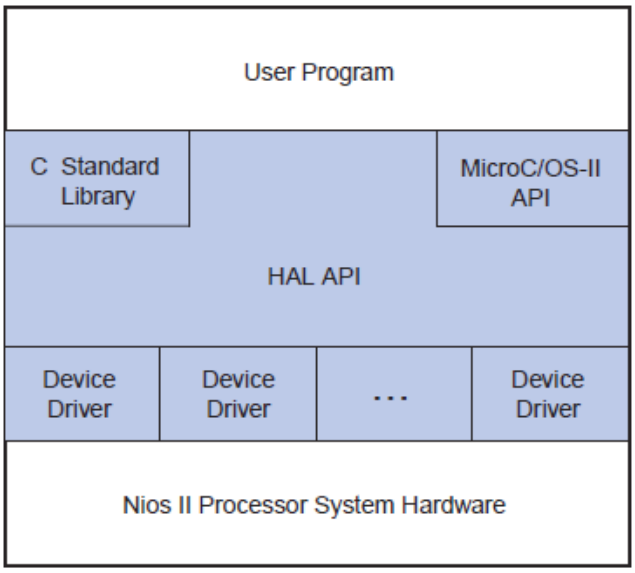
Пакет программ MicroC/OS-II похож на драйверы для аппаратных компонентов: когда MicroC/OS-II включен в проект Nios II, заголовочный и исходный файлы из **components/micrium_uc_osii** включаются в путь к проекту, принуждая компилировать ядро MicroC/OS-II и связывать его как часть проекта.

Архитектура MicroC/OS-II

Altera реализует MicroC/OS-II для Nios II процессора, включая некоторые элементы среды HAL в планировщик MicroC/OS-II и ассоциируя с MicroC/OS-II API. Завершённая HAL API доступна для все проектов MicroC/OS-II.

На рис. 10-1 показана архитектура программы на MicroC/OS-II и её взаимосвязь с HAL API.

Figure 10–1. Architecture of MicroC/OS-II Programs



Многопоточная среда влияет на определённые функции HAL.

За подробной информацией о последствиях вызова определённых функций HAL в многопоточной среде, обратитесь к главе "Справка по HAL API" в настольной книге программиста Nios II.

Многопоточная отладка в MicroC/OS-II

Когда вы отлаживаете MicroC/OS-II приложение, отладчик может отображать текущее состояние всех потоков в приложении, включая следы (backtraces) и значения регистров. Вы не сможете использовать отладчик для изменения текущего потока, поскольку невозможно использовать отладчик для смены потока или для пошагового исполнения другого потока.

Многопоточная отладка никак не может изменить поведение выбранного приложения.

Драйверы устройств MicroC/OS-II

Каждая периферия (то есть, каждый аппаратный компонент) предлагает файлы включений и исходные файлы в подпапках **inc** и **src** в папке **HAL** компонента.

За информацией обратитесь к главе "Разработка драйверов устройств для слоя аппаратной абстракции" в настольной книге программиста.

В дополнение к папке **HAL** компонент может дополнительно иметь папку **UCOSII**, в которой содержится специальный код для среды MicroC/OS-II. Так же как и в папке **HAL** в папке **UCOSII** находятся подпапки **inc** и **src**.

Когда вы создаёте MicroC/OS-II проект, эти папки добавляются в путь поиска исходных и файлов включений.

Nios II SBT копирует эти файлы в вашу подпапку BSP **obj**.

За дополнительной информацией о задании путей в Nios II SBT обратитесь к разделу "Встраиваемые программные проекты Nios II" в главе "Nios II SBT" в настольной книге программиста Nios II.

Вы можете использовать папку **UCOSII** для передачи кода, который будет использован только в многозадачной среде. Если вы используете другие папки в качестве дополнительного места поиска, то вы должны использовать механизм поиска драйвера устройств MicroC/OS-II, идентичный механизму поиска любого другого драйвера.

За подробной информацией о разработке драйверов устройств обратитесь к главе "Разработка драйверов устройств для слоя аппаратной абстракции" в настольной книге программиста.

Процесс инициализации системы HAL вызывает MicroC/OS-II функцию OSInit() перед вызовом alt_sys_init(), которая инсталлирует и инициализирует каждое устройство в системе. Поэтому полная MicroC/OS-II API доступна драйверам устройств, несмотря на то, что система запущена в однозадачном режиме, пока программа не вызовет OSStart() внутри тела main().

Драйверы сохранения потока HAL

Чтобы разрешить драйвер переноса между средой HAL и MicroC/OS-II, Altera определяет набор рабочих макросов, не зависимых от системы, которые предоставляют доступ к средствам операционной системы. Эти макросы реализуют функциональность, относящуюся к многозадачной среде. Когда макрос компилируется для проекта MicroC/OS-II, он распространяется на вызовы MicroC/OS-II API. Когда он компилируется для однозадачного HAL проекта, он распространяется на легкие пустые реализации. Этот макрос используется в коде драйвера устройства, предлагаемом Altera, и вы сможете использовать его, если вам потребуется написать драйвер со схожей портативностью.

В табл. 10-1 отображаются возможные драйверы и их функции.

За дополнительной информацией о функционировании в среде MicroC/OS-II обратитесь к "MicroC/OS-II: Ядро реального времени".

Путь, описанный для заголовочного файла, относится к директории: <*Nios II EDS путь установки*>**/ components/micrium_uc_osii/UCOSII/inc.**

Табл. 10-1. Независимые от ОС макросы для HAL драйверов сохранения потока (часть 1 из 2)

Макрос	**Определён в заголовке**	**MicroC/OS-II реализация**	**Однопоточная HAL реализация**
ALT_FLAG_GRP (группа)	**os/alt_flag.h**	Создаёт указатель на группу флагов с именем группы.	Пустой оператор.
ALT_EXTERN_FLAG_GRP (группа)	**os/alt_flag.h**	Создаёт внешнюю ссылку на указатель на группу флагов с именем группы.	Пустой оператор.
ALT_STATIC_FLAG_GRP (группа)	**os/alt_flag.h**	Создаёт статичный указатель на группу флагов с именем группы.	Пустой оператор.
ALT_FLAG_CREATE(группа, флаги)	**os/alt_flag.h**	Вызывает OSFlagCreate() для инициализации указателя на группу флагов, группу с флагами значения флагов. Код ошибки возвращает значение макроса	Возвращает 0 (успешно).

Табл. 10-1. Независимые от ОС макросы для HAL драйверов сохранения потока (часть 2 из 2)

Макрос	Определён в заголовке	MicroC/OS-II реализация	Однопоточная HAL реализация
ALT_FLAG_PEND(группа, флаги, тип ожидания, таймаут)	**os/alt_flag.h**	Вызывает OSFlagPend() с первыми четырьмя входными аргументами установками для групп, флагов, типа ожидания и таймаута соответственно.	Возвращает 0 (успешно).
ALT_FLAG_POST (группа, флаги, опции)	**os/alt_flag.h**	Вызывает OSFlagPost() с первыми тремя входными аргументами установками для групп, флагов и опций соответственно.	Возвращает 0 (успешно).
ALT_SEM (sem)	**os/alt_sem.h**	Создаёт OS_EVENT указатель с именем sem.	Пустой оператор.
ALT_EXTERN_SEM(sem)	**os/alt_sem.h**	Создаёт внешнюю ссылку на OS_EVENT указатель с именем sem.	Пустой оператор.
ALT_STATIC_SEM(sem)	**os/alt_sem.h**	Создаёт статичный OS_EVENT указатель с именем sem.	Пустой оператор.
ALT_SEM_CREATE(sem, значение)	**os/alt_sem.h**	Вызывает OSSemCreate() с аргументом значения для инициализации OS_EVENT указателя sem. Возвращает нуль, если успешно, или отрицательное, если нет.	Возвращает 0 (успешно).
ALT_SEM_PEND(sem, таймаут)	**os/alt_sem.h**	Вызывает OSSemPend() с первыми двумя аргументами установками для sem и таймаута соответственно. Код ошибки возвращает значение макроса.	Возвращает 0 (успешно).
ALT_SEM_POST(sem)	**os/alt_sem.h**	Вызывает OSSemPost() с входным аргументом sem.	Возвращает 0 (успешно).

Стандартная библиотека newlib ANSI C

Программы для MicroC/OS-II могут вызывать функции стандартной библиотеки ANSI C. Необходимы некоторые пояснения для многозадачной среды, чтобы удостовериться в том, что функции стандартной библиотеки Си сохраняют потоки. Библиотека newlib ANSI C сохраняет все глобальные переменные в одной структуре, связанной через указатель _impure_ptr. Однако реализация Altera MicroC/OS-II создаёт новый элемент структуры под каждую задачу. Во время переключения контекста, значение _impure_ptr обновляется по указателю на текущую версию структуры. В этом случае содержимое структуры по указателю _impure_ptr рассматривается как локальная задача. Например, при таком механизме каждая задача будет иметь свою собственную версию errno.

Эти данные локальных потоков находятся в верху стека задач. Вы должны иметь доступ для данных локальных потоков, когда выделяете память под стек. В основном структура _reent занимает ококло 900 байт данных под обычную библиотеку Си или 90 байт под уменьшенную библиотеку Си.

За подробной информацией о содержимом структуры _reent, обратитесь к документации newlib. В меню **Пуск**, кликните **Все программы** > **Altera** > **Nios II** > **Nios II Documentation**.

Дополнительно, реализация MicroC/OS-II предоставляет необходимые функции блокировки задач, позволяющие получить доступ к куче (вызов malloc() и free()), которые также сохраняют потоки.

Программы обработки прерываний (ISR) для MicroC/OS-II

Реализация ISR для MicroC/OS-II обычно затрагивает несколько служебных функций, как это описано в книге "MicroC/OS-II: Ядро реального времени". Однако, поскольку реализация MicroC/OS-II для Nios II основана на HAL, некоторые из этих функций скрыты от вас. HAL выполняет следующие служебные функции для вашей программы обработки прерываний (ISR):

- сохраняет и восстанавливает регистры процессора,
- вызывает OSIntEnter()и OSIntExit().

HAL также позволяет вам описывать вашу ISR на Си, а не на Ассеблере.

За подробной информацией о написании ISR в HAL обратитесь к главе "Обработка исключений" в настольной книге программиста Nios II.

Реализация MicroC/OS-II RTOS проектов на процессоре Nios II

Для создания программ на MicroC/OS-II, начните с задания настроек для BSP, в которых необходимо указать, что это проект MicroC/OS-II. Вы сможете проконтролировать конфигурацию ядра MicroC/OS-II, используя настройки Nios II SBT на Eclipse™, или через командную строку Nios II.

Вам не требуется редактировать заголовочные файлы (например, **OS_CFG.h**) или исходные файлы для конфигурирования средства MicroC/OS-II. Настройки проекта отображаются в BSP файле **system.h**, а **OS_CFG.h** уже содержит **system.h**.

За списком возможных настроек MicroC/OS-II BSP, обратитесь к разделу "Управление настройками в SBT" в главе "Справка по Nios II SBT" в настольной книге программиста. Настройки MicroC/OS-II находятся по префиксу ucosii. За информацией о том, как сконфигурировать MicroC/OS-II с помощью настроек BSP, обратитесь к главам "Начало работы с графической оболочкой", "Nios II SBT" и "Справка по Nios II SBT" в настольной книге программиста Nios II. Значение каждой настройки полностью описаны в книге "MicroC/OS-II: Ядро реального времени".