Программа обработки прерываний

Программа часто обращается к периферийным устройствам, используя аппаратные прерывания (IRQ). Когда периферия назначает свой IRQ, она отклоняет процессор от нормального процесса исполнения. Когда происходит прерывание, соответствующая ему программа обработки прерываний (ISR) должна обработать это прерывание и возвратить процессор полностью в состояние, предшествующее прерыванию.

Когда вы создаёте проект пакета поддержки платы (BSP), инструменты сборки содержат все необходимые драйверы устройств. Вам не потребуется писать HAL ISR, пока вы не организуете интерфейс с собственной периферией. Для справки, в этой секции описываются конструкции, предлагаемые HAL BSP для обработки аппаратных прерываний.

Обратитесь к существующим обработчикам компонентов Altera SOPC Builder за примерами написания HAL ISR.

За подробной информацией о предлагаемых Altera HAL обработчиков обратитесь к "Процессор Nios II Настольная книга программиста – 6. Разработка программ с использованием слоя аппаратной абстракции".

Функции HAL API для аппаратных прерываний

HAL предлагает усовершенствованный API (интерфейс прикладного программирования) для написания, регистрации и управления ISR. Этот API совместим с внешними и внутренними контроллерами прерываний.

Altera также поддерживает API устаревших аппаратных прерываний. Этот API поддерживает только IIC. Если у вас есть собственный драйвер, написанный для версии Nios II 9,1 и ранее, он использует устаревший API.

Оба API прерываний содержат следующие типы процедур:

- Процедура, вызываемая драйвером устройства для регистрирования ISR
- Процедура, вызываемая ISR для управления её оборудованием
- Процедура, вызываемая BSP или кодом приложения для контроля над поведением ISR

Оба API прерываний содержат следующие типы BSP:

- HAL BSP без RTOS
- RTOS BSP, основанная на HAL, например MicroC/OS-II BSP

Не используйте устаревшие API. Пишите новые драйверы, используя расширенное API, даже если они предназначены для поддержки IIC. Драйверы для устройств, поддерживающих EIC должны использовать расширенную API. Существующие устаревшие драйверы будут поддерживаться до особого уведомления.

Когда присутствует EIC, драйвер контроллера предоставляет настройки драйвера для BSP, которые могут быть использованы для конфигурирования драйвера. Количество и тип настроек зависит от реализации EIC и количества присутствующих EIC.

Примеры настроек драйвера EIC находятся в главе "Ядро векторного контроллера прерываний" в руководстве пользователя по встроенной IP периферии.

Выбор API прерываний

Когда SBT создаёт BSP, он определяет, должен ли BSP реализовывать API устаревших прерываний. Каждый драйвер, который поддерживает расширенное API, сообщает это свойство SBT через его *<driver name>*_**sw.tcl** файл. BSP реализует расширенное API, если все драйверы поддерживают его. Он реализует устаревшее API только, если этого требуют драйверы.

В зависимости от используемого API прерываний, SBT игнорирует любые устройства, чьи прерывания не подключены к процессору Nios II, ассоциированному с BSP.

Драйвер может сообщить свои прерывания поддерживаемой API с помощью свойств программы. Файл драйвера *<driver name>*_**sw.tcl** использует команду *set_sw_property* для установки *supported_interrupt_apis* либо *legacy_interrupt_api*, либо *enhanced_interrupt_api*, либо обоих.

Драйверы, поддерживающие расширенное API всегда сообщают об этой поддержке. Если неопределенны *supported_interrupt_apis*, то SBT принимает решение, что драйвер поддерживает только устаревшее API.

Начиная с версии 9.1, все драйверы устройств Altera поддерживают обе API. Эти драйверы могут быть использованы в BSP вместе с устаревшими драйверами. SBT определяет, необходимо ли устаревшее API, и реализует его, только когда это требуется. Если нет драйверов, которым необходимо устаревшее API, BSP реализует расширенное API.

Драйвер может быть написан для поддержки только расширенного API. Однако, вы не можете скомбинировать этот драйвер с устаревшими драйверами.

За подробной информацией о написании драйвера с поддержкой обеих API, обратитесь к секции "Поддержка множества API прерываний" на стр. 8-11.

Расширенные HAL API прерываний

Расширенные HAL API прерываний определены функциями, перечисленными в табл. 8-1, управляющими процессами аппаратных прерываний.

Table 8–1. Enhanced HAL Interrupt API Functions

Function Name	Implemented By
`alt_ic_isr_register()`	Interrupt controller driver *(1)*
`alt_ic_irq_enable()`	Interrupt controller driver *(1)*
`alt_ic_irq_disable()`	Interrupt controller driver *(1)*
`alt_ic_irq_enabled()`	Interrupt controller driver *(1)*
`alt_irq_disable_all()`	HAL
`alt_irq_enable_all()`	HAL
`alt_irq_enabled()`	HAL

Примечание к табл. 8-1:

(1) Если система основана на EIC, эти функции должны быть реализованы драйвером EIC. Если система основана на IIC, эти функции реализованы в HAL. Подробное описание каждой функции находится в главе "Справка по HAL API" в настольной книге программиста под Nios II.

Функции в табл. 8-1 работают со встроенными и внешними контроллерами прерываний. За подробным описанием функций расширенного API прерываний обратитесь к главе "Справка по HAL API" в настольной книге программиста под Nios II.

Использование расширенного HAL API для реализации ISR требует, чтобы вы выполняли следующие пункты:

1. Напишите вашу ISR, которая обрабатывает аппаратные прерывания для заданных устройств.
2. Проследите, чтобы ваша программа регистрировала ISR с помощью HAL, вызывая функцию *alt_ic_isr_register()*. Функция *alt_ic_isr_register()* разрешает для вас аппаратные прерывания.

SBT вставляет следующий символ определения в файл **system.h**, отображая в конфигурации процессора связанные с прерыванием аппаратные опции:

- NIOS2_EIC_PRESENT – если определено, отображает, что представлено одно или несколько EIC.
- NIOS2_NUM_OF_SHADOW_REG_SETS – отображает, что представлено множество теневых регистров. Максимальное значение – 63. Если теневых регистров нет, его значение – 0.

Драйвер внешнего контроллера прерываний

Для совместимости с расширенными HAL API прерываний, драйвер для EIC должен поддерживать функции, перечисленные в секции "Расширенные HAL API прерываний". Вы можете дополнить его функциями для поддержки специальных аппаратных средств. За примерами обратитесь к секции "Использование HAL API прерываний совместно с VIC".

Использование HAL API прерываний совместно с VIC

Драйвер Altera для VIC компонента поддерживает расширенные HAL API прерываний. Драйвер VIC предлагает поддержку для нескольких последовательно соединённых VIC устройств. Он также включает в себя поддержку для наборов теневых регистров. Настройки драйвера BSP позволяют вам разрешить автоматический приоритет (быстрые вложенные прерывания). Автоматический приоритет – это средство, позволяющее процессору Nios II оставлять маскируемые исключения разрешёнными, когда происходит аппаратное прерывание.

За дополнительной информацией о быстрых вложенных прерываниях обратитесь к секции "Процессы исключений" в главе "Программная модель" настольной книги по процессору Nios II.

Драйвер устройства VIC также предлагает следующие специальные функции устройств:

- int alt_vic_sw_interrupt_set(alt_u32 ic_id, alt_u32 irq);
- int alt_vic_sw_interrupt_clear(alt_u32 ic_id, alt_u32 irq);
- alt_u32 alt_vic_sw_interrupt_status(alt_u32 ic_id, alt_u32 irq);
- int alt_vic_irq_set_level(alt_u32 ic_id, alt_u32 irq, alt_u32 level);

За подробным описанием специальных процедур VIC драйвера устройств обратитесь к главе "Ядро векторного контроллера прерываний" в руководстве пользователя по встроенной IP периферии.

Драйвер EIC контролирует расположение таблицы аппаратного вектора прерываний. Например, драйвер Altera VIC по умолчанию размещает таблицу вектора в секции **.text**, но позволяет вам разместить таблицу вектора в другой секции с помощью настроек драйвера.

Память, в которой вы разместите таблицу вектора, должна быть подключена к обоим мастер портам данных и инструкций процессора Nios II.

Устаревшие HAL API прерываний

Устаревшие HAL API прерываний определяют следующие функции для управления процессами аппаратных прерываний:

- alt_irq_register()
- alt_irq_disable()
- alt_irq_enable()
- alt_irq_disable_all()
- alt_irq_enable_all()
- alt_irq_interruptible()
- alt_irq_non_interruptible()
- alt_irq_enabled()

Подробное описание этих функций в главе "Справка по HAL API" в настольной книге программиста Nios II.

Устаревшие драйверы не могут определять свойства supported_interrupt_apis. Отсутствие этих свойств показывает, что SBT необходимы устаревшие API прерываний.

Использование устаревших HAL API для реализации ISR требует, чтобы вы выполнили следующие пункты:

1. Написали свою ISR, которая обрабатывает аппаратные прерывания для заданного устройства.
2. Проследили, чтобы ваша программа регистрировала ISR с помощью HAL, вызывая функцию alt_irq_register(). Функция alt_irq_register() разрешает для вас аппаратные прерывания, вызовом alt_irq_enable_all().

Поддержка нескольких API прерываний

Когда вы пишите или обновляете собственный драйвер устройства, Altera рекомендует вам писать его одним из следующих способов:

- Пишите его с поддержкой расширенных HAL API прерываний. – Пишите драйвер этим способом, если вы собираетесь использовать его только вместе с другими драйверами, поддерживающими расширенные HAL API прерываний
- Пишите его для поддержки расширенных и устаревших API прерываний. – Пишите этим способом, если вы будете использовать его в комбинации с устаревшими драйверами, поддерживающими только устаревшее API.

Altera рекомендует использовать расширенное API всегда, когда ваш процессор реализует IIC. Поддержка расширенным API двух типов контроллеров прерываний и устаревших API не рекомендуется.

Когда SBT выбирает API прерываний, он определяет один из следующих символов в **system.h** для идентификации доступного API:

- ALT_ENHANCED_INTERRUPT_API_PRESENT – Определяет реализацию расширенного API
- ALT_LEGACY_INTERRUPT_API_PRESENT – Определяет реализацию устаревшего API

В вашем коде драйвера используйте эти символы для определения, какое API нужно вызвать. Для поддержки обоих API, ваш драйвер должен объявить поддержку API прерываний способом программных средств. В файле вашего драйвера *<driver name>*_**sw.tcl**, используйте команду set_sw_property для установки supported_interrupt_apis на оба legacy_interrupt_api и enhanced_interrupt_api.

За подробной информацией о команде set_sw_property, обратитесь к секции "Tcl команды" в главе "Справка по инструменту создания программ под Nios II" в настольной книге программиста Nios II.

Ограничения HAL ISR

Когда в вашей системе есть EIC, поддержка прерываний HAL накладывает следующие ограничения:

- Немаскируемые аппаратные прерывания должны использовать набор теневых регистров.
- Немаскируемые аппаратные прерывания не могут совместно использовать набор регистров с маскируемыми аппаратными прерываниями.