УДК 004.451.24
DOI: 10.15827/0236-235X.119.392-400
Дата подачи статьи: 30.03.17
2017. Т. 30. № 3. С. 392–400

РЕАЛИЗАЦИЯ КАНАЛОВ СПЕЦИФИКАЦИИ ARINC 653 В ОПЕРАЦИОННОЙ СИСТЕМЕ РЕАЛЬНОГО ВРЕМЕНИ БАГЕТ 3

А.Н. Годунов, к.ф.-м.н., зав. отделом, nkag@niisi.ras.ru;
В.А. Солдатов, к.т.н., старший научный сотрудник, nkvalera@niisi.ras.ru;
И.И. Хоменков, ведущий инженер, nkigor@niisi.ras.ru
(Федеральный научный центр Научно-исследовательский институт системных исследований РАН (ФНЦ НИИСИ РАН), Нахимовский просп., 36, корп. 1, г. Москва, 117218, Россия)

В статье рассматриваются каналы спецификации ARINC 653, предназначенные для взаимодействия между процессами. Предлагаются основные принципы построения и методы взаимодействия драйверов каналов с отечественной операционной системой реального времени Багет 3, в которой интерфейс с пользователем базируется на спецификации ARINC 653 и стандарте POSIX. Доступ к каналам осуществляется через порты. У каждого канала имеются только один порт, передающий сообщения, а также один или несколько портов, принимающих сообщения. Данные в каналах могут передаваться только в одном направлении.

Прикладная программа может использовать каналы в процессах, соответствующих как спецификации ARINC, так и стандарту POSIX. Формулируются требования, предъявляемые к драйверам каналов, и описываются средства управления уведомлениями, которые используются при создании конкретного драйвера. Рассматриваются этапы инициализации, приема и передачи сообщений, а также алгоритмы их реализации.

Предложенные способы построения взаимодействия драйверов каналов с операционной системой реального времени позволяют избежать записи данных в память чужого процесса, что существенно повышает надежность функционирования системы. Этот же интерфейс используется и в распределенных вычислительных системах, когда отдельные узлы системы соединяются между собой посредством шин с общим доступом (VME, RapidIO, Fibre Channel) или с помощью сети Ethernet. Таблица связей по каналам определяется при конфигурировании системы.

Драйвер каналов, разработанный для коммуникационной среды RapidIO, отличающейся высоким быстродействием, используется в отечественной многопроцессорной системе цифровой обработки сигналов.

Ключевые слова: *реальное время, операционная система, каналы, драйверы, многопроцессорные системы, RapidIO, POSIX, ARINC 653.*

В 2008 году состоялся первый выпуск отечественной *операционной системы реального времени* (ОСРВ) Багет 3.0 [1, 2]. Разработка системы была вызвана прежде всего высокими требованиями к надежности ее выполнения на основе концепции изолированных разделов, которая нашла свое отражение в спецификации ARINC 653 [3]. Известные зарубежные производители ОСРВ, такие как vxWorks [4], LynxOS-178 [5], Integrity-178B [6], также выпустили версии ОС, удовлетворяющие этой спецификации.

Таким образом, ОСРВ Багет 3 базируется на следующих международных стандартах: спецификация ARINC 653, POSIX 1003.1, стандарт на мобильные операционные системы (программный интерфейс).

Стандарты POSIX и ARINC существенно отличаются друга от друга по составу функций, их именам, используемой терминологии. В ОСРВ Багет 3 в качестве основной была выбрана спецификация ARINC 653. Стандарт POSIX используется в той мере, в какой это допускается спецификацией ARINC 653.

Спецификация ARINC 653 определяет интерфейс (APEX – APplication EXecutive) между ОС и прикладными программами. APEX предписывает пространственное и временное разделение ресурсов. Терминология, используемая в ARINC 653, отличается от терминологии, принятой в стандарте POSIX. В ARINC 653 единицей планирования ресурсов является раздел (partition). В стандарте POSIX этому термину соответствует понятие «процесс» (process). В рамках одного раздела возможно параллельное выполнение нескольких процессов, тогда как в POSIX в рамках одного процесса возможно параллельное выполнение нескольких потоков управления (thread). В настоящей статье использована терминология, принятая в стандарте POSIX.

Спецификация ARINC 653 предусматривает как пользовательские, так и системные процессы. Интерфейс пользовательских процессов должен соответствовать требованиям спецификации, а на интерфейс системных процессов в ARINC 653 не накладывается никаких ограничений. В ОСРВ Багет 3 для системных процессов используется интерфейс стандарта POSIX. Таким образом, под управлением ОСРВ Багет 3 могут одновременно выполняться как ARINC-процессы, так и POSIX-процессы. Все эти процессы, кроме главного системного процесса, будем называть пользовательскими. Все пользовательские процессы работают в пользовательском режиме процессора и используют виртуальную адресацию, что исключает доступ одних процессов к памяти других. Главный системный процесс, хотя и выполняется в привилегированном режиме работы процессора, в основном также использует виртуальную адресацию, доступ по физическим адресам применяется крайне редко. Организация виртуальной адресации позво-

ляет обеспечить контроль доступа к отдельным сегментам памяти, в частности, защиту пользовательских процессов друг от друга и защиту главного системного процесса от пользовательских процессов. Каждый процесс работает со своими виртуальными адресами, трансляцию которых в физические выполняет аппаратура компьютера. Таким образом, пользовательский процесс лишен возможности прямого обращения к страницам основной памяти, занятым информацией, относящейся к другим процессам.

Взаимодействие между процессами

Согласно спецификации ARINC 653, процессы могут взаимодействовать между собой только путем передачи сообщений через каналы. Такое ограничение делает процессы слабо взаимосвязанными. Это в значительной мере уменьшает влияние ошибок в одних процессах на работу других процессов, а также упрощает восстановление работоспособности системы в случае обнаружении ошибок при работе пользовательских процессов. Если в результате обработки ошибки, возникшей при выполнении одного из пользовательских процессов, этот процесс будет повторно запущен (произведен рестарт), то для других процессов, которые взаимодействовали с перезапущенным процессом, это может привести лишь к потере нескольких сообщений.

Доступ к каналам производится через порты. Каждый канал имеет только один порт, передающий сообщения, и один или несколько портов, принимающих сообщения. Данные в канале могут передаваться только в одном направлении.

Каналы могут использоваться для передачи сообщений

– между процессами, которые выполняются на одном и том же процессорном модуле;

– между процессами, выполняемыми на разных процессорных модулях;

– между процессами и внешними устройствами.

В первом случае передача данных производится только средствами ОС, во втором и третьем случаях – драйверами и ОС.

Каждый канал может работать в одном из двух режимов: с очередью сообщений (queuing-канал) и без очереди сообщений (sampling-канал). Интерфейс взаимодействия прикладных программ с каналами не зависит от способа передачи данных, но зависит от режима работы канала. Для каждого режима работы используется свой набор функций.

Независимость интерфейса от способа передачи данных значительно повышает мобильность прикладных программ и позволяет выбирать способы передачи данных на этапе конфигурирования системы без внесения изменений в прикладные программы.

Каналы и порты должны быть описаны при конфигурировании системы. Указанные при конфигурировании параметры каналов (используемые порты, режим работы, направление передачи данных и др.) нельзя изменить во время работы системы.

Канал, работающий в режиме с очередью сообщений, имеет один порт приема сообщений (порт-получатель) и один порт отправки сообщений (порт-отправитель). Порт-отправитель содержит очередь отправляемых сообщений, а порт-получатель – очередь принимаемых сообщений. Размер очередей ограничен и указывается при создании порта. Каждое отправляемое сообщение доставляется получателю и может быть получено только один раз.

Канал, работающий в режиме без очереди сообщений, может иметь только один порт отправки сообщений и один или несколько портов приема сообщений. Каждое сообщение, получаемое портом приема, замещает ранее полученные сообщения (они просто теряются). Таким образом, из порта всегда считывается сообщение, полученное последним. Прочитанное сообщение считается годным, если с момента поступления его в порт приема до момента чтения из порта прошло времени не больше, чем на обновление порта.

Если при отправке сообщения оказалось, что по каким-то причинам предыдущее сообщение не было отправлено, новое сообщение замещает старое. Таким образом, отправка сообщения в порт всегда возможна и прикладная программа всегда может выводить сообщения в порт с нужной частотой (это не относится к процессу передачи данных от порта приема к порту передачи).

На рисунке 1 представлена модель интерфейса драйверов каналов с ОСРВ.

Как видно из рисунка, для передачи данных между двумя модулями создаются два канала: один на модуле, передающем данные, другой на модуле, принимающем данные. Канал, который находится на модуле, передающем данные, имеет порт-отправитель в пользовательском процессе ОСРВ, а порт-получатель – в псевдоразделе. Канал на модуле, принимающем данные, имеет порт-отправитель в псевдоразделе, а порт-получатель – в стандартном процессе ОСРВ.

Передача данных между портом-отправителем в процессе-отправителе и портом-получателем в процессе-получателе происходит в несколько этапов. На первом этапе сообщение из порта-отправителя в процессе-отправителе по каналу передается в порт-получатель псевдораздела модуля-отправителя. Передача данных по этому каналу производится операционной системой. На втором этапе полученное сообщение отправляется драйвером по линии связи модулю-получателю. На третьем этапе драйвер модуля-получателя принимает сообщение по линии связи и помещает его в порт-отправитель

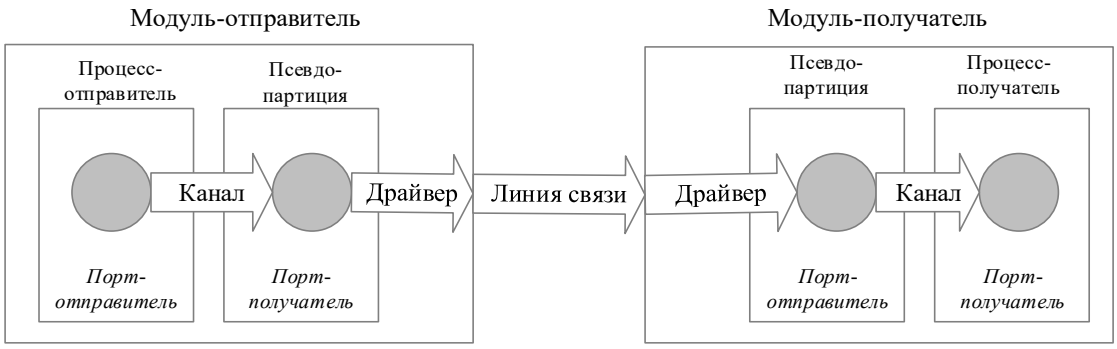

Рис. 1. Передача данных между двумя модулями

Fig. 1. Data transfer between two modules

псевдораздела модуля-получателя. На четвертом этапе это сообщение передается в порт-получатель в процессе-получателе.

В ОСРВ Багет 3 каналы могут использоваться не только в ARINC-процессах, но и в POSIX-процессах.

Драйвер канала для ОСРВ Багет 3 обычно состоит из обработчика прерываний и, возможно, одного или нескольких потоков главного (привилегированного) POSIX-процесса. Рассмотрим правила создания драйвера, разработанные для ОСРВ Багет 3.

Назначение и принципы работы канального драйвера

Как уже отмечалось, прием и передача сообщений между пользовательскими процессами происходит в несколько этапов. Драйвер каналов, который выполняется на 2-м и 3-м этапах, обеспечивает передачу и прием сообщений по физическим линиям связи между портом-получателем псевдораздела передающего модуля и портом-отправителем псевдораздела принимающего модуля. Управление устройством (прием и передача сообщений) производится драйвером в контексте специального потока привилегированного POSIX-процесса и/или в контексте функций обработки прерываний, которые также используют адресное пространство привилегированного POSIX-процесса.

Поток, в котором выполняется драйвер, может активизироваться либо периодически (периодический режим), либо по требованию (непериодический режим). В первом случае драйвер через равные интервалы времени выполняет требуемые действия по приему и/или передаче сообщений. Во втором случае драйвер активизируется с помощью функций уведомления о появлении нового сообщения или свободного буфера. Уведомления могут быть посланы из ARINC- или POSIX-процесса, а также из функции обработки прерываний с помощью соответствующих функций ОСРВ.

Накладные расходы при работе драйвера уменьшаются, если передавать и принимать не по одному сообщению, а сразу группу сообщений. Периодический режим позволяет накопить несколько сообщений и потом передать сразу группу сообщений. В этом случае драйвер не извещается о появлении ни новых сообщений, ни новых свободных буферов (что также существенно снижает накладные расходы). Периодический драйвер не обрабатывает прерывания от аппаратуры и не заказывает их порождения. Он позволяет экономить время на обработке прерываний и уведомлений, но уступает непериодическому по скорости реакции. Интерфейс взаимодействия с драйвером, используемый в ОС РВ Багет, предполагает поддержку обоих режимов. Конкретный вариант может быть выбран на этапе конфигурирования.

Сообщения, отправляемые или получаемые драйвером, могут передаваться вместе с заголовком. Если в псевдоразделе находится порт-отправитель (порт-отправитель принадлежит драйверу), ОС игнорирует заголовок и передает прикладной программе сообщение без заголовка. Если в псевдоразделе находится порт-получатель, ОС формирует сообщение с заголовком или без него в зависимости от атрибутов, указанных при конфигурировании. Формируемый ОС (стандартный) заголовок содержит длину и номер сообщения.

Для размещения сообщений в памяти используются буфера. Все буфера одного канала имеют одинаковый размер, который указывается при конфигурировании. Для каждого буфера создается его описатель. Описатель буфера содержит указатель на буфер, длину сообщения и др. Буфера и их описатели создаются процессом-отправителем.

Функции драйвера должны обеспечивать выполнение следующих действий:

– инициализация драйвера;

– инициализация отдельного порта;

– чтение сообщений из порта и передача их по каналам связи;

– прием сообщений по каналам связи и отправка их через порт.

При инициализации ОСРВ вызывается специальная функция ОС инициализации каналов, при выполнении которой разбираются параметры кон-

фигурации каналов и инициализируются соответствующие канальные драйверы. В настоящее время в зависимости от оборудования и параметров конфигурации могут быть проинициализированы канальные драйверы, обеспечивающие межмодульную связь по сети Ethernet по протоколу UDP, по шине VME, по шине RapidIO.

Инициализация драйвера

Каждый драйвер должен иметь свою функцию инициализации, которая вызывается при инициализации. В процессе инициализации драйвера он регистрирует в ОС функции драйвера, которые может использовать ОС для взаимодействия с драйвером. К ним относятся:

– функция probe(), которая позволяет определить, какие порты обслуживаются данным драйвером (вызывается в привилегированном *POSIX*-процессе);

– функция инициализации порта, которая вызывается при создании каждого порта, обслуживаемого данным драйвером.

Функция инициализации драйвера должна зарегистрировать драйвер в ОС с помощью функции chanCreateDriver(). Эта функция создает описатель драйвера в ОС и возвращает указатель на него. Доступ к описателю драйвера в ОС со стороны драйвера возможен только с помощью функций ОС (непосредственный доступ запрещен).

Далее драйвер должен поместить информацию о драйвере в описатель драйвера в ОС (с помощью функций ОС):

– указатель на функцию probe() драйвера;

– указатель на функцию инициализации порта драйвера;

– указатель на функцию запуска (старта) драйвера;

– имя драйвера.

Драйвер может создать свою управляющую таблицу (описатель) и зарегистрировать ее в управляющей таблице драйвера в ОС.

Функция проверки драйвера probe() входит в состав драйвера и используется ОС при поиске драйвера, который обслуживает данный (межмодульный) канал. Если драйвер обслуживает указанный канал, функция probe() возвращает 0, в противном случае не ноль.

Инициализация порта

При создании порта, находящегося в псевдоразделе, ОС вызывает функцию инициализации порта драйвера и передает ей указатель на описатель порта ОС. Доступ к описателю порта в ОС со стороны драйвера возможен только с помощью функций ОС (непосредственный доступ запрещен).

Функция инициализации порта драйвера может получить указатель на управляющую таблицу (описатель) драйвера и информацию об инициализируемом порте (канале). С помощью функций ОС драйвер может получить следующую информацию:

– идентификатор модуля-партнера;

– номер логической связи (подканала);

– имя канала;

– идентификатор канала;

– имя порта;

– размер буферов;

– число буферов;

– направление передачи данных (ввод или вывод);

– период обновления (для sampling-портов);

– тип канала (queuing, sampling, SAP).

Драйвер может создать свою управляющую таблицу (описатель) порта и зарегистрировать ее в управляющей таблице порта ОС (с помощью функции ОС). При инициализации канала (порта) драйвер также определяет функцию уведомления о появлении сообщения или свободного буфера в канале, а также наличие заголовков сообщений.

Обслуживание буферов

Для приема и передачи сообщений через порт в ОСРВ Багет 3 был разработан нестандартный интерфейс взаимодействия с драйвером канала. Для работы с буферами и очередями буферов предназначены следующие функции:

– chanGetBuffer() − получить свободный буфер;

– chanPutBuffer() − освободить буфер;

– chanGetMessage() − получить сообщение;

– chanPutMessage() − отправить сообщение.

Эти функции можно использовать как для порта queuing, так и для sampling, но в зависимости от типа порта (queuing или sampling) функции используют разный алгоритм.

Обслуживание буферов для queuing-каналов

Для каждого queuing-канала ОСРВ создает две очереди: очередь сообщений и очередь свободных буферов. Процесс-отправитель помещает отправляемые сообщения в очередь сообщений, а процесс-получатель извлекает сообщения из этой очереди. Очередь свободных буферов используется для передачи освободившихся буферов от процесса-получателя к процессу-отправителю. Процесс-получатель помещает свободные буфера в очередь, а процесс-отправитель извлекает их из очереди.

Все очереди организованы как циклические буфера, то есть состоят из массива идентификаторов буферов и двух указателей: указателя свободной области и указателя занятой области. Такая организация очередей упрощает синхронизацию при уста-

новке и извлечении элементов очереди. Длина каждого массива равна числу буферов плюс один. Так как в любой очереди буфер может содержаться не более одного раза, то его всегда можно поместить в очередь.

Массив очереди сообщений находится в памяти процесса-отправителя, там же находится и указатель свободной области (номер первого элемента в области свободных элементов) этой очереди. Указатель занятой области (номер первого элемента в области занятых элементов) находится в памяти процесса-получателя.

Массив очереди свободных буферов находится в памяти процесса-получателя, там же находится и указатель свободной области (номер первого элемента в области свободных элементов). Указатель занятой области (номер первого элемента в области занятых элементов) находится в памяти процесса-отправителя.

Если номер первого элемента в области занятых элементов меньше номера первого элемента в области свободных, то область занятых элементов находится между первым занятым (включительно) и первым свободным (не включая его). Все остальные элементы считаются свободными.

Если номер первого элемента в области занятых элементов больше номера первого элемента в области свободных, то область свободных элементов находится между первым свободным (включительно) и первым занятым (не включая его). Занятые элементы располагаются от первого занятого до конца массива и далее от начала массива до первого свободного (не включая его).

Если номер первого элемента в области занятых элементов равен номеру первого элемента в области свободных, считается, что все элементы свободны. Таким образом, всегда в массиве имеется хотя бы один свободный элемент (его нельзя использовать).

Занятый элемент массива содержит идентификатор буфера. В случае очереди отправленных сообщений это идентификатор буфера, содержащего сообщение, а в случае очереди свободных номеров – идентификатор свободного буфера.

Если нужно добавить буфер в очередь, выбирается первый элемент в области свободных (он лежит сразу после области занятых элементов). В этот элемент помещается идентификатор буфера. После этого указатель свободной области передвигается на 1 элемент. Если же он указывал на конец массива, то перемещается на начало массива.

Если нужно извлечь буфер из очереди, выбирается первый элемент в области занятых. Из этого элемента считывается идентификатор буфера. После этого указатель области занятых элементов передвигается на 1 элемент. Если же он указывал на конец массива, то перемещается на начало массива.

Функции chanGetBuffer() и chanPutMessage() используются при обслуживании портов-отправителей, а функции chanGetMessage() и chanPutBuffer() при обслуживании портов-получателей.

Функция chanGetBuffer() извлекает буфер из очереди свободных буферов и возвращает его вызвавшей программе. Если очередь пуста, функция формирует запрос на уведомление о появлении свободного буфера и возвращает управление.

Функция chanPutMessage() помещает сообщение в очередь сообщений. Функция также посылает уведомление о появлении сообщения, если был неудовлетворенный запрос на уведомление.

Функция chanGetMessage() извлекает буфер из очереди сообщений и возвращает его вызвавшей программе. Если очередь пуста, функция формирует запрос на уведомление о появлении сообщения и возвращает управление.

Функция chanPutBuffer() помещает буфер в очередь свободных буферов. Функция также посылает уведомление о появлении свободного буфера, если был неудовлетворенный запрос на уведомление.

Обслуживание буферов для sampling-каналов

В случае sampling-каналов буфера используются портами-отправителями для размещения в них отправляемых сообщений, а портами-получателями для доступа к полученным сообщениям. Один и тот же буфер может одновременно использоваться несколькими портами-получателями, но не может одновременно использоваться портом-отправителем и портом-получателем. Поэтому, если один из портов-получателей перестал использовать какой-либо буфер, это не означает, что буфер стал свободным и может использоваться портом-отправителем для нового сообщения.

Порт, принадлежащий псевдоразделу (драйверу), может одновременно использовать (закрепить за собой) несколько буферов. Максимальное число буферов, которое может быть закреплено за портом, обслуживаемым данным драйвером, должно быть определено при инициализации порта функцией chanSetPortMaxBuffNum(). Порты, принадлежащие пользовательскому процессу (ARINC или POSIX), не закрепляют за собой буфера.

Учет буферов, используемых в настоящее время портом (закрепленных за портом), производится ОС, управление буферами – функциями ОС. Функции chanGetBuffer() и chanPutMessage() используются при обслуживании портов-отправителей, а функции chanGetMessage() и chanPutBuffer() при обслуживании портов-получателей.

Ввод сообщений

Queuing-каналы (прием сообщений). Для ввода сообщений через устройство создается канал, порт-получатель которого находится в обычном процессе, а порт-отправитель – в псевдораз-

деле. Сообщения, получаемые через обсуживаемое драйвером устройство, драйвер выводит через порт-отправитель канала.

Для получения сообщений из queing-порта прикладная программа (процесс-получатель) использует функцию ОСРВ Багет 3 RECEIVE_QUEUING_MESSAGE. Эта функция запрашивает сообщение из канала с помощью функции ОС chanGetMessage(). Если в очереди есть (непрочитанные) сообщения, эта функция возвращает буфер с самым старым (непрочитанным) сообщением. Полученное сообщение копируется в буфер, указанный при вызове функции RECEIVE_QUEUING_MESSAGE. Освободившийся буфер помещают в очередь с помощью функции chanPutBuffer(). Эта функция отправляет уведомление о появлении свободного буфера драйверу, если это требуется.

Если сообщение получить не удалось, функция формирует запрос на уведомление о получении сообщения и возвращает соответствующий код возврата. Если при вызове функции RECEIVE_QUEUING_MESSAGE указано ненулевое время ожидания сообщения, поток объявляется неработоспособным.

Буфера для ввода сообщений с устройства канальный драйвер должен брать из очереди свободных буферов канала с помощью функции ОС chanGetBuffer(). Если свободных буферов нет, функция возвращает соответствующий код возврата и формирует запрос на уведомление о появлении свободных буферов в канале. Соответствующая функция уведомления драйвера должна быть зарегистрирована заранее (в противном случае уведомления не будут отправляться).

Для каждого выводимого в канал сообщения драйвер указывает его номер. Если сообщения нумерует (модуль) отправитель, используется его номер, в противном случае драйвер нумерует сообщения при получении. Если драйвер обнаружил потерю сообщений, он уведомляет об этом ОС (поле cb_overflow в описателе буфера).

Буфера, содержащие полученные с устройства сообщения, драйвер помещает в очередь сообщений канала с помощью функции ОС chanPutMessage(). Перед вызовом функции chanPutMessage() драйвер должен поместить в описатель буфера следующую информацию: длина сообщения, порядковый номер сообщения, признак переполнения (если какие-то сообщения были утеряны).

Функция chanPutMessage() также отправит уведомление о появлении сообщения процессу-получателю, если это требуется. Функция обработки этого уведомления объявляет работоспособными потоки, которые ждут сообщений (столько потоков, сколько есть сообщений).

При периодической работе драйвера прерывания не порождаются аппаратурой и не обрабатываются драйвером.

Sampling-каналы (чтение сообщений). Если для ввода сообщений через устройство используется sampling-канал, то порты-получатели канала находятся в обычных процессах, а порт-отправитель – в псевдоразделе. Порт-отправитель обслуживается драйвером. Описатель порта находится в памяти привилегированного POSIX-процесса, там же находятся буфера и их описатели.

Поступающие с устройства сообщения драйвер выводит через порт-отправитель канала. Прикладная программа получает сообщение из порта-получателя канала с помощью функции ОС READ_SAMPLING_MESSAGE. Эта функция находит буфер, содержащий самое свежее сообщение, запоминает номер этого сообщения и копирует сообщение из буфера в память прикладной программы. После копирования функция проверяет, находится ли в буфере то же сообщение, что и до чтения. Если сообщение было затерто более свежим, снова запрашивается самое свежее сообщение и т.д.

Несколько потоков могут одновременно читать сообщения из одного и того же sampling-порта функцией READ_SAMPLING_MESSAGE. В силу этого sampling-порты обычных процессов не закрепляют буфера на время чтения сообщения. Функция READ_SAMPLING_MESSAGE не запрашивает и не посылает никаких уведомлений (так как это не требуется).

Для получения свободного буфера для ввода сообщений канальный драйвер может использовать функцию ОСРВ chanGetBuffer().

Перед использованием буфера устройством драйвер должен сбросить поле порядкового номера сообщения (записать -1) в описателе буфера. По завершении записи сообщения в буфер (ввода сообщения устройством) драйвер должен поместить в описатель буфера следующую информацию: длина сообщения, порядковый номер сообщения.

Далее драйвер должен вывести буфер с сообщением в порт канала с помощью функции chanPutMessage(). В результате вызова этой функции указанный буфер будет объявлен содержащим самое свежее сообщение.

В данном случае (когда все порты-получатели находятся в стандартных процессах) драйвер может взять управление буферами полностью на себя (например, циклически использовать все имеющиеся буфера) и не использовать функцию chanGetBuffer().

Драйвер не запрашивает никаких уведомлений и не порождает уведомлений (так как это не требуется).

Вывод сообщений

Queuing-каналы (отправка сообщений). Для вывода сообщений через устройство создается канал, порт-отправитель которого находится в обычном процессе, а порт-получатель – в псевдоразделе.

Порт-получатель канала обслуживается драйвером. Сообщения, полученные через порт-получатель канала, драйвер выводит через устройство. Прикладная программа отправляет сообщения с помощью функции ОС SEND_QUEUING_MESSAGE. Эта функция запрашивает свободный буфер (из очереди свободных буферов канала) с помощью функции ОС chanGetBuffer(). Если в очереди есть свободные буфера, функция возвращает свободный буфер. Отправляемое сообщение копируется в полученный буфер, и буфер с сообщением помещается в очередь сообщений канала функцией ОС chanPutMessage(). Эта функция отправит уведомление об отправке сообщения драйверу, если это требуется.

Если свободных буферов нет, формируется запрос на уведомление о появлении свободного буфера в канале. Если после окончания указанного времени ожидания свободный буфер не появился, функция заканчивает свою работу с соответствующим кодом окончания.

Порт-получатель queuing-канала обслуживается драйвером канала. Драйвер извлекает буфера с сообщениями из очереди сообщений канала с помощью функции ОС chanGetMessage() и выводит их через устройство. Если очередь сообщений пуста, функция возвращает соответствующий код возврата и формирует запрос на уведомление о появлении сообщений в канале. Соответствующая функция уведомления драйвера должна быть зарегистрирована заранее (в противном случае уведомления не будут отправляться).

Буфера, содержащие сообщения, уже отправленные через устройство, драйвер помещает в очередь свободных буферов канала с помощью функции ОС chanPutBuffer(). Эта функция отправит уведомление о появлении свободного буфера процессу-получателю, если это требуется. Функция обработки этого уведомления объявляет работоспособными потоки, которые ждут свободных буферов (столько потоков, сколько свободных буферов).

Sampling-каналы (запись сообщений). При выводе сообщений через устройство посредством sampling-канала порт-отправитель канала находится в обычном процессе, а один или несколько портов-получателей – в псевдоразделе. Эти порты обслуживаются драйвером.

Поступающие через канал сообщения драйвер выводит через устройство; каждый раз выводятся самые свежие сообщения, так как более свежие сообщения замещают более старые (старые сообщения просто игнорируются).

Управление устройством производится драйвером в контексте обработчика прерываний или специального потока драйвера привилегированного POSIX-процесса.

Прикладная программа отправляет сообщения через порт-отправитель канала с помощью функции ОС WRITE_SAMPLING_MESSAGE. Функция вначале запрашивает свободный буфер функцией ОС chanGetBuffer(). Эта функция просматривает все порты и создает список всех используемых буферов (буфер, содержащий самое свежее сообщение, считается используемым), находит свободный буфер и возвращает его идентификатор.

В полученный таким образом буфер функция копирует сообщение и отправляет его в канал с помощью функции ОС chanPutMessage(). В результате буфер перестает быть закрепленным за портом-отправителем и становится буфером, содержащим самое свежее сообщение.

Функция chanPutMessage() отправляет уведомление об отправке сообщения всем портам-получателям, которым это требуется. Функция уведомления может, например, объявить поток драйвера работоспособным. Для портов, находящихся в стандартных процессах, функция уведомления не используется.

Если один поток попытается выводить сообщение функцией WRITE_SAMPLING_MESSAGE в порт, в который сейчас выводит сообщение другой поток, его сообщение будет проигнорировано.

Sampling-каналы могут иметь несколько портов-получателей. Находящиеся в псевдоразделах порты-получатели обслуживаются драйверами устройств.

Драйвер получает наиболее свежее сообщение, выведенное в канал, с помощью функции ОС chanGetMessage(). Если это сообщение не было выведено ранее, драйвер выводит его через обслуживаемое им устройство. После вывода сообщения драйвер освобождает содержащий его буфер с помощью функции chanPutBuffer() и он может быть повторно использован для вывода сообщения.

Функция chanGetMessage() всегда порождает запрос на уведомление о выводе сообщения. Драйвер должен заранее зарегистрировать функцию обработки уведомления, так как иначе уведомления не будут посылаться. При выводе очередного сообщения функция WRITE_SAMPLING_MESSAGE направляет извещение драйверу.

На рисунке 2 представлено графическое изображение описанного порядка ввода и вывода сообщений.

Уведомления

Средства управления уведомлениями, входящие в состав ОСРВ Багет 3, играют важную роль при создании канальных драйверов. Рассмотрим типы существующих уведомлений, их назначение и пример использования в основном цикле головной функции драйвера канала.

Существуют следующие типы уведомлений (извещений):

- извещение о выводе сообщения в канал;
- извещение об освобождении буфера.

Рис. 2. Схема взаимодействия между прикладной программой и драйвером

Fig. 2. The scheme of the interaction between an application program and a driver

Уведомления о выводе сообщения порождаются владельцем порта-отправителя (обычным процессом или псевдоразделом) и направляются владельцу порта-получателя. Уведомления об освобождении буфера порождаются владельцем порта-получателя и направляются владельцу порта-отправителя.

Драйвер получает уведомление о выводе сообщения при обслуживании порта-получателя queuing-канала при выполнении следующих условий:

– драйвер заранее зарегистрировал функцию уведомления (о выводе сообщения);

– драйвер заранее породил запрос на уведомление о выводе сообщения.

Драйвер, обслуживающий порт-отправитель queuing-канала, получает уведомление об освобождении буфера при выполнении следующих условий:

– драйвер заранее зарегистрировал функцию уведомления (об освобождении буфера);

– драйвер заранее породил запрос на уведомление об освобождении буфера.

Запрос на уведомление о выводе сообщения в queuing-канал драйвер порождает неявно при вызове функции ОС chanGetMessage(), если очередь сообщений пуста. Запрос на уведомление об освобождении буфера драйвер также порождает неявно при вызове функции ОС chanGetBuffer(), если очередь свободных буферов пуста.

Драйвер, обслуживающий порт-получатель sampling-канала, получает уведомление о выводе сообщения при выполнении следующих условий:

– драйвер заранее зарегистрировал функцию уведомления (о выводе сообщения);

– драйвер заранее породил запрос на уведомление о выводе сообщения.

Запрос на уведомление о выводе сообщения в sampling-канал драйвер порождает неявно при вызове функции ОС chanGetMessage().

Уведомления об освобождении буфера в sampling-канале никогда не порождаются, так как свободные буфера всегда есть.

Аппарат уведомлений является частью драйвера и создается разработчиком драйвера с помощью следующих функций ОСРВ:

– chanNotificationInit() − инициализация средств передачи уведомлений;

– chanNotificationWait() − ожидание уведомления или начала периода;

– chanNotificationPut() − постановка параметров уведомления в очередь;

– chanNotificationGet() − извлечение параметров уведомления из очереди.

Функция chanNotificationInit() должна быть вызвана потоком управления драйвера в начале его работы. Функция создает очередь уведомлений. Одному уведомлению соответствует один элемент очереди, который содержит два параметра. Функции уведомления драйвера помещают в очередь параметры уведомления с помощью функции chanNotificationPut(). Драйвер получает уведомления с помощью функции chanNotificationGet(). Если очередь уведомлений пуста, драйвер может использовать функцию chanNotificationWait() для ожидания появления уведомлений в очереди. Если драйвер работает в периодическом режиме, вызов функции chanNotificationInit() считается началом первого периода. Для ожидания начала следующего периода поток драйвера может использовать также функцию chanNotificationWait().

Заключение

В работе описана общая схема взаимодействия различных частей системы для обмена сообщениями в системах реального времени, сформулированы основные требования к драйверу, описаны этапы приема и передачи сообщений. Предложенные способы взаимодействия драйверов каналов с ОСРВ позволяют избежать записи данных в память чужого процесса, что существенно повышает надежность системы. Этот же интерфейс используется и в распределенных вычислительных системах, когда различные устройства системы соединяются между собой посредством шин с общим доступом.

Для ОСРВ Багет 3 разработаны и протестированы драйверы, обеспечивающие межмодульную связь по сети Ethernet по протоколу UDP, по шине VME, по шине RapidIO. Подготовлен прототип драйвера для Fibre Channel (FC).

Драйвер каналов, разработанный для коммуникационной среды RapidIO, которая отличается высоким быстродействием, используется в отечественной линейке процессоров [7] для решения задачи цифровой обработки сигналов [8] на многопроцессорной системе.

На описанную модель взаимодействия ОСРВ с драйвером канала получены патенты [9, 10].

Литература

1. Годунов А.Н. Операционная система реального времени Багет 3.0 // Программные продукты и системы. 2010. № 4. С. 15–19.
2. Годунов А.Н., Солдатов В.А. Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. С. 68–76.
3. Годунов А.Н., Солдатов В.А. Спецификация ARINC 653 и ее реализация в операционной системе реального времени Багет 3 // Программная инженерия. 2015. № 6. С. 3–17.
4. WIND RIVER VXWORKS 653 PLATFORM. URL: http://www.windriver.com/products/product-overviews/PO_VE_6_9_Platform_0211.pdf (дата обращения: 29.03.2017).
5. LynxOS-178 RTOS for DO-178B Software Certification. URL: http://www.lynx.com/products/real-time-operating-systems/lynxos-178-rtos-for-do-178b-software-certification/ (дата обращения: 29.03.2017).
6. Safety Critical Products: INTEGRITY®-178B RTOS. URL: http://www.ghs.com/products/safety_critical/integrity-do-178b.html (дата обращения: 29.03.2017).
7. Бобков С.Г. Импортозамещение элементной базы вычислительных систем // Вестн. РАН. 2014. Т. 84. № 11. С. 1010–1016.
8. Райко Г.О., Павловский Ю.А., Мельканович В.С. Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства «КОМДИВ» // Гидроакустика. 2014. № 20. С. 85–92.
9. Бетелин В.Б., Годунов А.Н., Грюнталь А.И., Солдатов В.А., Хоменков И.И. Способ передачи данных между процессами. Патент № 2592461, Российская Федерация; заявл. 12.05.2014, опубл. 20.07.2016.
10. Аряшев С.И., Бетелин В.Б., Аракелов А.А., Солдатов В.А., Годунов А.Н., Малофеев Ю.Г., Хоменков И.И., Бобков С.Г. Способ передачи сообщений между вычислительными устройствами. Патент № 2611337, Российская Федерация; заявл. 13.01.2016, опубл. 21.02.2017.

Software & Systems Received 30.03.17
DOI: 10.15827/0236-235X.119.392-400 2017, vol. 30, no. 3, pp. 392–400

CHANNELS IMPLEMENTATION OF ARINC 653 SPECIFICATION IN RTOS BAGET 3

A.N. Godunov [1], *Ph.D. (Physics and Mathematics), Head of Department, nkag@niisi.ras.ru*
V.A. Soldatov [1], *Ph.D. (Engineering), Senior Researcher, nkvalera@niisi.ras.ru*
I.I. Homenkov [1], *Leading Engineer, nkigor@niisi.ras.ru*

[1] *Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA RAS), Nakhimovsky Ave. 36/1, Moscow, 117218, Russian Federation*

Abstract. The article considers ARINC specification 653 channels, which are intended for interprocess communication. The paper proposes the basic methods of designing channel driver, as well as their interaction with the Russian real-time operating system Baget 3 (RTOS Baget 3). RTOS Baget 3 user interface is based on ARINC 653 Specification and POSIX standard. The channel access is performed through ports. Every channel has the only port for sending messages and a single or several destination ports for receiving messages. Messages can be sent through a channel in only one direction. An application program can use the channels in the processes that correspond both ARINC and POSIX.

The paper specifies the requirements for channel drivers and describe notification control means used for driver development. It also considers the stages of initialization, sending/receiving messages and implemented algorithms. The proposed methods of implementing drivers interaction with RTOS allow avoiding data recording to the other process memory that significantly enhances the system reliability. Distributed computing systems use the same interface when separate subsystems are interconnected by means of shared access buses (VME, RapidIO, Fibre Channel) or Ethernet. The connection table for processor units is specified at the stage of system configuration. The channel driver, designed for RapidIO communication environment that offers fast operation speed, is used in a multiprocessor digital signal processing system.

Keywords: real-time, operating system, channels, drivers, multiprocessor systems, RapidIO. POSIX, ARINC 653.

References

1. Godunov A.N. Real-time operating system baget 3.0. *Programmnye produkty i sistemy* [Software & Systems]. 2010, no. 4, pp. 15–19 (in Russ.).
2. Godunov A.N., Soldatov V.A. Operating Systems of the Baget Family (Likeness, Differences and Perspectives). *Programmirovanie* [Programming and Computer Software]. 2014, no. 5, pp. 68–76 (in Russ.).
3. Godunov A.N., Soldatov V.A. ARINC Specification 653 and its Implementation in RTOS Baget 3. *Programmnaya inzheneriya* [Software Engineering]. 2015, no. 6, pp. 3–17 (in Russ.).
4. *WIND RIVER VXWORKS 653 PLATFORM*. Available at: http://www.windriver.com/ products/product-overviews/PO_VE_6_9_Platform_0211.pdf (accessed March 29, 2017).
5. *LynxOS-178 RTOS for DO-178B Software Certification*. Available at: http://www.lynx.com/products/real-time-operating-systems/lynxos-178-rtos-for-do-178b-software-certification/ (accessed March 29, 2017).
6. *Safety Critical Products: INTEGRITY®-178B RTOS*. Available at: http://www.ghs.com/products/safety_critical/integrity-do-178b.html (accessed March 29, 2017).
7. Bobkov S.G. Import substitution of the element base of computer systems. *Herald of the Russian Academy of Sciences*. 2014, vol. 84, iss. 11, pp. 1010–1016 (in Russ.).
8. Raiko G.O., Pavlovsky Yu.A., Melkanovich V.S. Technology of programming of multiprocessor processing of hydroacoustic signals on “KOMDIV” computer set. *Hydroacoustics*. 2014, no. 20 (2), pp. 85–92 (in Russ.).
9. Betelin V.B., Godunov A.N., Gryuntal A.I., Soldatov V.A., Homenkov I.I. *The Method for Transferring Data Between Processes*. Patent RF, no. 2692461, 2016 (in Russ.).
10. Aryashev S.I., Betelin V.B., Arakelov A.A., Soldatov V.A., Godunov A.N., Malofeev Yu.G., Homenkov I.I., Bobkov S.G. *A Method for Transmitting Messages Between Computing Devices*. Patent RF, no. 2611337, 2017 (in Russ.).