Учебное руководство по последовательному интерфейсу и UART

Аннотация

Эта статья рассказывает об использовании последовательного оборудования с FreeBSD.

Содержание

1. UART: Что это и как работает 1
2. Настройка драйвера sio 24
3. Настройка драйвера cy 29
4. Настройка драйвера si 30

1. UART: Что это и как работает

Copyright ® 1996 Frank Durda IV <uhclem@FreeBSD.org>*, All Rights Reserved. 13 января 1996 год*

Универсальный асинхронный приёмопередатчик (UART) — это ключевой компонент подсистемы последовательной передачи данных компьютера. UART принимает байты данных и передаёт отдельные биты последовательно. На стороне приёмника второй UART собирает биты обратно в полные байты.

Последовательная передача данных обычно используется с модемами и для не сетевого взаимодействия между компьютерами, терминалами и другими устройствами.

Существует две основные формы последовательной передачи данных: синхронная и асинхронная. В зависимости от режимов, поддерживаемых оборудованием, название подсистемы связи обычно включает букву A, если она поддерживает асинхронную передачу, и букву S, если поддерживается синхронная передача. Обе формы описаны ниже.

Некоторые распространённые сокращения:

UART Universal Asynchronous Receiver/Transmitter — Универсальный асинхронный приёмопередатчик

USART Universal Synchronous-Asynchronous Receiver/Transmitter — Универсальный синхронно-асинхронный приёмопередатчик

1.1. Синхронная последовательная передача

Синхронная последовательная передача данных требует, чтобы отправитель и получатель имели общий тактовый сигнал, либо чтобы отправитель предоставлял строб-сигнал или другой сигнал синхронизации, чтобы получатель знал, когда "считывать" следующий бит данных. В большинстве форм синхронной последовательной связи, если в данный момент нет доступных данных для передачи, вместо них должен быть отправлен заполняющий символ, чтобы передача данных не прерывалась. Синхронная связь обычно более эффективна, так как между отправителем и получателем передаются только биты данных, однако она может быть более затратной, если требуются дополнительные провода и схемы для обмена тактовым сигналом между отправителем и получателем.

Форма синхронной передачи используется с принтерами и устройствами с жёсткими дисками, где данные передаются по одному набору проводов, а тактовый сигнал или строб — по другому проводу. Принтеры и устройства с жёсткими дисками обычно не являются последовательными устройствами, так как большинство стандартов интерфейсов жёстких дисков передают целое слово данных для каждого тактового сигнала или строба, используя отдельный провод для каждого бита слова. В индустрии ПК такие устройства известны как параллельные.

Стандартное оборудование для последовательной связи в ПК не поддерживает синхронные операции. Этот режим описан здесь только для сравнения.

1.2. Асинхронная последовательная передача

Асинхронная передача позволяет передавать данные без необходимости отправки тактового сигнала от отправителя к получателю. Вместо этого отправитель и получатель заранее согласовывают параметры синхронизации, а к каждому слову добавляются специальные биты, которые используются для синхронизации передающего и принимающего устройств.

При передаче слова через UART в асинхронном режиме к началу каждого передаваемого слова добавляется бит, называемый "стартовым битом". Стартовый бит используется для оповещения приёмника о начале передачи слова данных, а также для синхронизации тактового сигнала приёмника с тактовым сигналом передатчика. Эти два тактовых сигнала должны быть достаточно точными, чтобы их расхождение по частоте не превышало 10% во время передачи оставшихся битов слова. (Данное требование было установлено во времена механических телетайпов и легко выполняется современным электронным оборудованием.)

После стартового бита передаются отдельные биты слова данных, начиная с младшего значащего бита (LSB). Каждый бит передаётся в течение точно такого же времени, как и все остальные биты, и приемник "проверяет" состояние линии примерно на середине интервала, отведенного для каждого бита, чтобы определить, является ли бит `1` или `0`. Например, если передача каждого бита занимает две секунды, приемник проверит сигнал, чтобы определить, является ли он `1` или `0`, через одну секунду, затем подождет две секунды и проверит значение следующего бита, и так далее.

Отправитель не знает, когда получатель «посмотрел» значение бита. Отправитель знает только, когда по тактовому сигналу нужно начать передачу следующего бита слова.

Когда все слово данных отправлено, передатчик может добавить бит чётности, который он генерирует. Бит чётности может быть использован приемником для выполнения простой проверки на ошибки. Затем передатчик отправляет как минимум один стоповый бит.

Когда приемник получил все биты в слове данных, он может проверить биты чётности (как отправитель, так и приемник должны договориться о том, будет ли использоваться бит чётности), а затем приемник ищет стоповый бит. Если стоповый бит не появляется, когда должен, UART считает все слово искаженным и сообщит об ошибке кадрирования главному процессору при чтении слова данных. Обычная причина ошибки кадрирования — несовпадение скорости тактовых сигналов отправителя и приемника или прерывание сигнала.

Независимо от того, были ли данные получены правильно или нет, UART автоматически отбрасывает бит чётности, стартовый и стоповый биты. Если отправитель и получатель настроены одинаково, эти биты не передаются хосту.

Если готово следующее слово для передачи, стартовый бит нового слова может быть отправлен сразу после того, как будет отправлен стоповый бит предыдущего слова.

Поскольку асинхронные данные являются "самосинхронизирующимися", если нет данных для передачи, линия передачи может быть неактивна.

1.3. Другие функции UART

Помимо основной задачи преобразования данных из параллельного формата в последовательный для передачи и из последовательного в параллельный при приеме, UART обычно предоставляет дополнительные схемы для сигналов, которые могут использоваться для указания состояния среды передачи и регулирования потока данных в случае, если удалённое устройство не готово принимать больше данных. Например, когда устройство, подключённое к UART, является модемом, модем может сообщать о наличии несущей на телефонной линии, в то время как компьютер может дать команду модему сбросить себя или не принимать вызовы, поднимая или опуская один или несколько из этих дополнительных сигналов. Функция каждого из этих дополнительных сигналов определена в стандарте EIA RS232-C.

1.4. Стандарты RS232-C и V.24

В большинстве компьютерных систем UART подключен к схеме, которая генерирует сигналы, соответствующие спецификации EIA RS232-C. Также существует стандарт CCITT под названием V.24, который отражает спецификации, включённые в RS232-C.

1.4.1. Назначения битов RS232-C (метки и пробелы)

В стандарте RS232-C значение `1` называется `Маркер` (Mark), а значение `0` — `Пробел` (Space). Когда линия связи находится в состоянии покоя, говорят, что она "маркирует" (Marking), то есть

передаёт непрерывные значения 1.

Стартовый бит всегда имеет значение 0 (пробел). Стоповый бит всегда имеет значение 1 (метка). Это означает, что на линии всегда будет переход от метки (1) к пробелу (0) в начале каждого слова, даже при передаче нескольких слов подряд. Это гарантирует, что отправитель и получатель могут синхронизировать свои тактовые сигналы независимо от содержимого передаваемых битов данных.

Время простоя между стоповым и стартовым битами не обязательно должно быть точным кратным (включая ноль) скорости передачи данных коммуникационного канала, однако большинство UART спроектированы таким образом для простоты.

В стандарте RS232-C сигнал «Marking» (логическая 1) представлен напряжением от -2 В до -12 В, а сигнал «Spacing» (логический 0) — напряжением от 0 В до +12 В. Передатчик должен выдавать +12 В или -12 В, а приёмник должен учитывать возможные потери напряжения в длинных кабелях. Некоторые маломощные передатчики (например, в портативных компьютерах) иногда используют только +5 В и -5 В, но эти значения всё ещё допустимы для приёмника RS232-C при условии использования коротких кабелей.

1.4.2. Сигнал Break в RS232-C

RS232-C также определяет сигнал под названием Break, который вызывается передачей непрерывных значений Spacing (без стартовых или стоповых битов). Когда на линии данных отсутствует напряжение, считается, что линия передаёт Break.

Сигнал Break должен иметь длительность больше, чем время, необходимое для передачи полного байта, включая стартовый, стоповый и биты чётности. Большинство UART способны различить ошибку кадрирования и сигнал Break, но если UART не поддерживает эту функцию, для определения Break можно использовать обнаружение ошибки кадрирования.

Во времена телетайпов, когда множество принтеров по всей стране были соединены последовательно (например, в службах новостей), любое устройство могло вызвать Break, временно размыкая всю цепь, чтобы ток не протекал. Это использовалось для того, чтобы место с срочными новостями могло прервать устройство в другом месте, которое в данный момент передавало информацию.

В современных системах существует два типа сигналов Break. Если Break длится дольше 1,6 секунд, он считается "Модемным Break", и некоторые модемы можно запрограммировать на завершение соединения и переход в режим ожидания или вход в командный режим модема при обнаружении этого сигнала. Если Break короче 1,6 секунд, это означает "Break данных", и удалённый компьютер должен решить, как реагировать на этот сигнал. Иногда такая форма Break используется как сигнал "Внимание" или "Прерывание", а иногда принимается как замена символу ASCII CONTROL-C.

Метки и пробелы также эквивалентны "дыркам" и "отсутствию дырок" в системах с бумажной лентой.



Разрывы не могут быть сгенерированы с перфоленты или из любого другого

> байтового значения, поскольку байты всегда отправляются со стартовым и стоповым битами. UART обычно способен генерировать непрерывный сигнал Spacing в ответ на специальную команду от главного управляющего устройства (процессора передачи).

1.4.3. RS232-C устройства DTE и DCE

Спецификация RS232-C определяет два типа оборудования: оконечное оборудование данных (DTE — Data Terminal Equipment) и оборудование передачи данных (DCE — Data Carrier Equipment). Обычно устройство DTE — это терминал (или компьютер), а DCE — модем. На другом конце телефонной линии в разговоре принимающий модем также является устройством DCE, а компьютер, подключённый к этому модему, — устройством DTE. Устройство DCE принимает сигналы на тех контактах, на которых устройство DTE передаёт, и наоборот.

Когда два устройства, оба являющиеся DTE или DCE, должны быть соединены вместе без модема или аналогичного преобразователя среды между ними, необходимо использовать NULL модем. NULL модем электрически перестраивает кабель так, что выход передатчика подключается ко входу приемника на другом устройстве, и наоборот. Аналогичные преобразования выполняются для всех управляющих сигналов, чтобы каждое устройство видело то, что оно считает сигналами DCE (или DTE) от другого устройства.

Количество сигналов, генерируемых устройствами DTE и DCE, не симметрично. Устройство DTE генерирует меньше сигналов для устройства DCE, чем получает от него.

1.4.4. Назначение контактов RS232-C

Спецификация EIA RS232-C (и её эквивалент ITU, V.24) предусматривает использование двадцатипятиконтактного разъёма (обычно DB25) и определяет назначение большинства контактов в этом разъёме.

В IBM Personal Computer и подобных системах подмножество сигналов RS232-C предоставляется через девятиконтактные разъемы (DB9). Сигналы, которые не включены в разъем ПК, в основном связаны с синхронной работой, и этот режим передачи не поддерживается UART, выбранным IBM для использования в IBM PC.

В зависимости от производителя компьютера, для связи по RS232-C могут использоваться разъемы DB25, DB9 или оба типа. (В IBM PC также используется разъем DB25 для параллельного интерфейса принтера, что иногда вызывает путаницу.)

Ниже представлена таблица назначений сигналов RS232-C в разъемах DB25 и DB9.

Контакт в DB25 RS232-C	Контакт в DB9 IBM PC	Символ цепи по EIA	Символ цепи по CCITT	Общее имя	Источник сигнала	Описание
1	-	AA	101	PG/FG	-	Защитное заземление (Frame/Protective Ground)
2	3	BA	103	TD	DTE	Передача Данных (Transmit Data)
3	2	BB	104	RD	DCE	Прием данных (Receive Data)
4	7	CA	105	RTS	DTE	Запрос на передачу (Request to Send)
5	8	CB	106	CTS	DCE	Готовность к приёму (Clear to Send)
6	6	CC	107	DSR	DCE	Готовность терминального оборудования (Data Set Ready)
7	5	AV	102	SG/GND	-	Сигнальная земля (Signal Ground)
8	1	CF	109	DCD/CD	DCE	Обнаружение несущей (Data Carrier Detect)
9	-	-	-	-	-	Зарезервировано для Теста

Контакт в DB25 RS232-C	Контакт в DB9 IBM PC	Символ цепи по EIA	Символ цепи по CCITT	Общее имя	Источник сигнала	Описание
10	-	-	-	-	-	Зарезервировано для Теста
11	-	-	-	-	-	Зарезервировано для Теста
12	-	CI	122	SRLSD	DCE	Детектор сигнала вторичной линии приёма
13	-	SCB	121	SCTS	DCE	Вторичный сигнал готовности к приёму
14	-	SBA	118	STD	DTE	Вторичная линия передачи данных
15	-	DB	114	TSET	DCE	Тактирование элементов сигнала передатчика (Trans. Sig. Element Timing)
16	-	SBB	119	SRD	DCE	Вторичная линия приема данных
17	-	DD	115	RSET	DCE	Тактирование элементов сигнала приёмника (Receiver Signal Element Timing)

Контакт в DB25 RS232-C	Контакт в DB9 IBM PC	Символ цепи по EIA	Символ цепи по CCITT	Общее имя	Источник сигнала	Описание
18	-	-	141	LOOP	DTE	Локальная петля
19	-	SCA	120	SRS	DTE	Вторичный запрос на передачу
20	4	CD	108.2	DTR	DTE	Готовность терминального оборудования (Data Terminal Ready)
21	-	-	-	RDL	DTE	Режим удалённой цифровой петли (Remote Digital Loopback)
22	9	CE	125	RI	DCE	Индикатор передачи данных (Ring Indicator)
23	-	CH	111	DSRS	DTE	Селектор скорости передачи данных
24	-	DA	113	TSET	DTE	Тактирование элементов сигнала передатчика (Trans. Sig. Element Timing)
25	-	-	142	-	DCE	Режим тестирования

1.5. Биты, боды и символы

Скорость передачи данных (Baud) — это единица измерения скорости передачи в асинхронной связи. Из-за развития технологий модемной связи этот термин часто ошибочно используют для описания скорости передачи данных в современных устройствах.

Традиционно, скорость передачи (Baud Rate) представляет количество битов, фактически передаваемых по среде, а не объём данных, которые действительно перемещаются от одного устройства DTE к другому. Подсчет Baud включает служебные биты — Start, Stop и Parity, которые генерируются передающим UART и удаляются принимающим UART. Это означает, что 7-битные слова данных на самом деле занимают 10 бит для полной передачи. Следовательно, модем, способный передавать 300 бит в секунду, обычно может передавать только 30 7-битных слов, если используется Parity и присутствуют один бит Start и один бит Stop.

Если используются 8-битные слова данных и биты чётности, скорость передачи данных снижается до 27,27 слов в секунду, так как теперь для передачи восьмибитных слов требуется 11 бит, а модем по-прежнему передаёт только 300 бит в секунду.

Формула преобразования байтов в секунду в бодовую скорость и наоборот была простой до появления модемов с коррекцией ошибок. Эти модемы принимают последовательный поток битов от UART в компьютере (даже внутренние модемы часто работают с последовательными данными) и преобразуют биты обратно в байты. Затем эти байты объединяются в пакеты и передаются по телефонной линии с использованием синхронного метода передачи. Это означает, что стоповые, стартовые и биты чётности, добавленные UART в DTE (компьютере), удаляются модемом перед передачей отправляющим модемом. Когда эти байты принимаются удалённым модемом, он добавляет стартовые, стоповые и биты чётности к словам, преобразует их в последовательный формат и отправляет на принимающий UART в удалённом компьютере, который затем удаляет стартовые, стоповые и биты чётности.

Причина, по которой выполняются все эти дополнительные преобразования, заключается в том, чтобы два модема могли осуществлять коррекцию ошибок. Это означает, что принимающий модем может запросить у передающего модема повторную отправку блока данных, который был получен с некорректной контрольной суммой. Эта проверка обрабатывается модемами, и устройства DTE обычно не осознают, что этот процесс происходит.

Удаляя стартовые, стоповые и биты чётности, дополнительные биты данных, которые два модема должны обмениваться между собой для выполнения коррекции ошибок, в основном скрываются от эффективной скорости передачи, наблюдаемой отправляющим и принимающим оборудованием DTE. Например, если модем отправляет десять 7-битных слов другому модему без включения стартовых, стоповых и битов чётности, отправляющий модем сможет добавить 30 бит своей собственной информации, которую принимающий модем может использовать для коррекции ошибок, не влияя на скорость передачи реальных данных.

Использование термина "Бод" дополнительно осложняется модемами, выполняющими

сжатие. Одно 8-битное слово, переданное по телефонной линии, может представлять собой дюжину слов, переданных на отправляющий модем. Принимающий модем развернёт данные обратно в их исходное содержимое и передаст эти данные принимающему DTE.

Современные модемы также включают буферы, которые позволяют скорости передачи битов по телефонной линии (DCE к DCE) отличаться от скорости передачи битов между DTE и DCE на обоих концах соединения. Обычно скорость между DTE и DCE выше, чем скорость между DCE и DCE, из-за использования сжатия модемами.

Поскольку количество битов, необходимых для описания байта, менялось во время передачи между двумя машинами, а также из-за различающихся скоростей передачи в битах в секунду на линиях DTE-DCE и DCE-DCE, использование термина «Бод» для описания общей скорости связи вызывает проблемы и может искажать реальную скорость передачи. Таким образом, термин «Биты в секунду» (bps) является корректным для описания скорости передачи на интерфейсе DCE-DCE, а термины «Бод» или «Биты в секунду» допустимы, когда соединение устанавливается между двумя системами с проводным подключением или используется модем, не выполняющий коррекцию ошибок или сжатие.

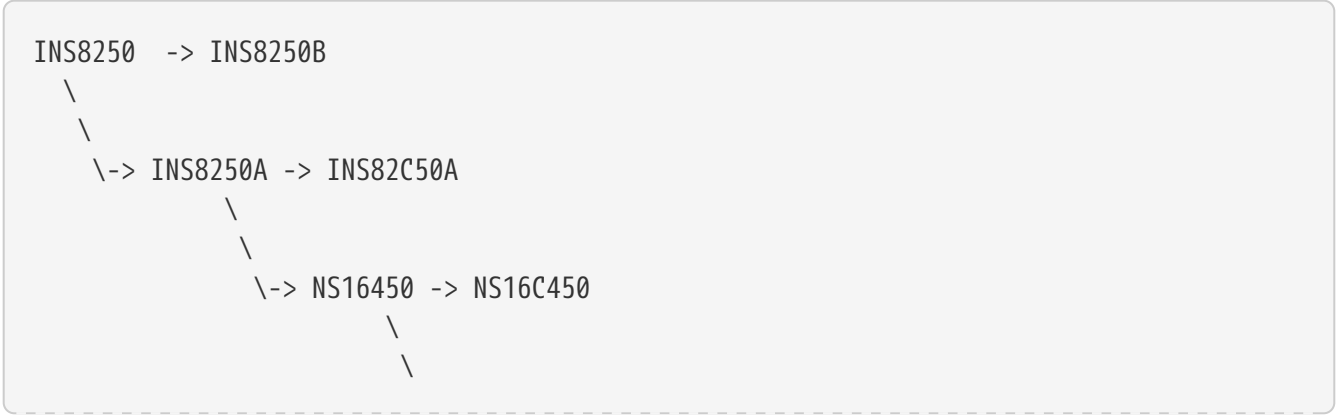
Современные высокоскоростные модемы (2400, 9600, 14,400 и 19,200 бит/с) на самом деле всё ещё работают на скорости 2400 бод или ниже, или, точнее, 2400 символов в секунду. Высокоскоростные модемы способны кодировать больше бит данных в каждый символ с использованием техники, называемой "Заполнение созвездия (Constellation Stuffing)", поэтому эффективная скорость передачи данных в битах в секунду у модема выше, но модем продолжает работать в ограниченной полосе пропускания звуковых частот, предоставляемой телефонной системой. Модемы, работающие на скоростях 28,800 и выше, имеют переменную скорость передачи символов, но техника остаётся той же.

1.6. UART в IBM PC

Начиная с оригинального IBM Personal Computer, IBM выбрала UART INS8250 от National Semiconductor для использования в адаптере Parallel/Serial IBM PC. Последующие поколения совместимых компьютеров от IBM и других производителей продолжали использовать INS8250 или улучшенные версии UART из семейства National Semiconductor.

1.6.1. Генеалогическое дерево National Semiconductor UART

Существует несколько версий и последующих поколений UART INS8250. Основные версии описаны ниже.



```
\-> NS16550 -> NS16550A -> PC16550D
```

INS8250

Эта часть использовалась в оригинальном IBM PC и IBM PC/XT. Первоначальное название этой части — INS8250 ACE (Asynchronous Communications Element), и она изготовлена по NMOS-технологии.

8250 использует восемь портов ввода-вывода и имеет однобайтовый буфер передачи и однобайтовый буфер приема. Этот оригинальный UART имеет несколько состояний гонки и другие недостатки. Оригинальный BIOS IBM включает код для обхода этих недостатков, но это сделало BIOS зависимым от их наличия, поэтому последующие модели, такие как 8250A, 16450 или 16550, не могли быть использованы в оригинальном IBM PC или IBM PC/XT.

INS8250-B

Это более медленная скорость INS8250, созданная по NMOS-технологии. Она имеет те же проблемы, что и оригинальный INS8250.

INS8250A

Улучшенная версия INS8250 с использованием технологии XMOS, в которой исправлены различные функциональные недостатки. INS8250A изначально использовалась в клонах ПК от производителей, применявших "чистые" проекты BIOS. Из-за исправлений в микросхеме этот чип не мог использоваться с BIOS, совместимой с INS8250 или INS8250B.

INS82C50A

Это CMOS-версия (с низким энергопотреблением) INS8250A и имеет схожие функциональные характеристики.

NS16450

Так же, как NS8250A, но с улучшениями для работы с более быстрыми шинами CPU. IBM использовала этот компонент в IBM AT и обновила IBM BIOS, чтобы она больше не зависела от ошибок в INS8250.

NS16C450

Это версия NS16450 с технологией CMOS (низкое энергопотребление).

NS16550

То же, что и NS16450, с 16-байтовым буфером передачи и приема, но конструкция буфера была неудачной и не могла быть надёжно использована.

NS16550A

То же, что и NS16550, но с исправленными недостатками буфера. 16550A и его преемники стали наиболее популярными UART-устройствами в индустрии ПК, в основном благодаря их способности надёжно работать на высоких скоростях передачи данных в операционных системах с медленным временем отклика прерываний.

NS16C552

Этот компонент состоит из двух CMOS UART NS16C550A в одном корпусе.

PC16550D

Так же, как NS16550A, с исправленными незначительными недостатками. Это ревизия D семейства 16550 и последняя доступная версия от National Semiconductor.

1.6.2. NS16550AF и PC16550D — это одно и то же

Компания National реорганизовала свою систему нумерации деталей несколько лет назад, и чип NS16550AFN больше не существует под этим названием. (Если у вас есть NS16550AFN, посмотрите на дату изготовления на корпусе — это четырёхзначное число, обычно начинающееся с девятки. Первые две цифры обозначают год, а последние две — неделю года, когда чип был упакован. Если у вас есть NS16550AFN, скорее всего, он уже довольно старый.)

Новые номера выглядят как PC16550DV, с незначительными отличиями в суффиксных буквах в зависимости от материала корпуса и его формы. (Описание системы нумерации можно найти ниже.)

Важно понимать, что в некоторых магазинах можно заплатить $15 (США) за микросхему NS16550AFN, выпущенную в 1990 году, а в соседнем ящике могут лежать новые PC16550DN с небольшими исправлениями, которые National внесла с момента выпуска AFN. PC16550DN, вероятно, произведены в последние полгода и стоят вдвое дешевле (от $5 (США) при оптовой покупке), чем NS16550AFN, поскольку они легко доступны.

Поскольку поставки чипов NS16550AFN продолжают сокращаться, цена, вероятно, будет расти до тех пор, пока больше людей не узнают и не примут тот факт, что PC16550DN действительно выполняет ту же функцию, что и старый номер детали.

1.6.3. Система нумерации компонентов National Semiconductor

Старые номера деталей NSnnnnnrqp теперь имеют формат PCnnnnnrgp.

r — это поле ревизии. Текущая ревизия 16550 от National Semiconductor — D.

p — это поле типа пакета. Типы:

"F"	QFP	(quad flat pack - квадратный плоский корпус) с L-образными выводами
"N"	DIP	(dual inline package — корпус с двусторонним расположением выводов) для сквозного монтажа с прямыми выводами
"V"	LPCC	(lead plastic chip carrier — пластиковый корпус) с J-образными выводами

Поле *g* обозначает класс изделия. Если перед буквой типа пакета стоит I, это указывает на

«промышленный» класс детали, который имеет более высокие характеристики, чем стандартная деталь, но не такие высокие, как компонент военного назначения (Milspec). Это необязательное поле.

То, что мы раньше называли NS16550AFN (DIP-корпус), теперь называется PC16550DN или PC16550DIN.

1.7. Другие производители и аналогичные UART

На протяжении многих лет чипы 8250, 8250A, 16450 и 16550 лицензировались или копировались другими производителями. В случае с 8250, 8250A и 16450 точная схема ("мегаячейка") была лицензирована многими производителями, включая Western Digital и Intel. Другие производители проводили обратную разработку чипа или создавали эмуляции с аналогичным поведением.

Во внутренних модемах разработчик модема часто эмулирует 8250A/16450 с помощью микропроцессора модема, и эмулированный UART часто имеет скрытый буфер размером в несколько сотен байт. Благодаря размеру буфера, эти эмуляции могут быть такими же надёжными, как 16550A, в способности обрабатывать высокоскоростные данные. Однако большинство операционных систем по-прежнему сообщают, что UART является только 8250A или 16450, и могут не эффективно использовать дополнительную буферизацию, присутствующую в эмулированном UART, если не используются специальные драйверы.

Некоторые производители модемов под давлением рыночных сил отказываются от конструкции с буфером в сотни байт и вместо этого используют UART 16550A, чтобы их продукция выглядела выигрышно в рыночных сравнениях, даже если это может снизить фактическую производительность.

Распространённое заблуждение заключается в том, что все микросхемы с маркировкой "16550A" одинаковы по производительности. Однако между ними существуют различия, а в некоторых клонах 16550A даже встречаются серьёзные недостатки.

Когда компания National Semiconductor разработала NS16550, она получила несколько патентов на эту конструкцию и также ограничила лицензирование, что затруднило для других производителей выпуск чипов с аналогичными характеристиками. В результате патентов обратно спроектированные конструкции и эмуляции должны были избегать нарушения пунктов, охватываемых патентами. Впоследствии эти копии почти никогда не работают точно так же, как NS16550A или PC16550D, которые являются компонентами, наиболее востребованными производителями компьютеров и модемов, но иногда они не готовы платить цену, необходимую для получения оригинальных деталей.

Некоторые различия в клонах микросхем 16550A несущественны, в то время как другие могут полностью препятствовать использованию устройства с определённой операционной системой или драйвером. Эти различия могут проявиться при использовании других драйверов или при возникновении определённых комбинаций событий, которые не были хорошо протестированы или учтены в драйвере Windows®. Это происходит потому, что большинство производителей модемов и клонов 16550 используют драйверы Microsoft из Windows® for Workgroups 3.11 и утилиту Microsoft® MS-DOS® в качестве основных тестов на совместимость с NS16550A. Этот чрезмерно упрощенный

критерий означает, что при использовании другой операционной системы могут возникнуть проблемы из-за тонких различий между клонами и оригинальными компонентами.

National Semiconductor предоставила программу под названием COMTEST, которая выполняет тесты совместимости независимо от каких-либо драйверов ОС. Следует помнить, что цель такого типа программ — демонстрация недостатков в продуктах конкурентов, поэтому программа будет сообщать как о значительных, так и о крайне незначительных различиях в поведении тестируемого компонента.

В серии тестов, проведенных автором этого документа в 1994 году, компоненты производства National Semiconductor, TI, StarTech и CMD, а также мегаячейки и эмуляции, встроенные во внутренние модемы, были протестированы с помощью COMTEST. Ниже приведён счетчик различий для некоторых из этих компонентов. Поскольку эти тесты проводились в 1994 году, они могут не отражать текущую производительность данного продукта от поставщика.

Следует отметить, что COMTEST обычно завершает работу при обнаружении чрезмерного количества или определённых типов проблем. В рамках этого тестирования COMTEST был изменён так, чтобы он не завершал работу независимо от количества обнаруженных различий.

Поставщик	**Номер детали**	**Ошибки (также известные как "различия" в отчётах)**
National	(PC16550DV)	0
National	(NS16550AFN)	0
National	(NS16C552V)	0
TI	(TL16550AFN)	3
CMD	(16C550PE)	19
StarTech	(ST16C550J)	23
Rockwell	Стандартный модем с внутренним 16550 или его эмуляцией (RC144DPi/C3000-25)	117
Sierra	Модем с внутренним 16550 (SC11951/SC11351)	91



На сегодняшний день автор данного документа не обнаружил ни одного не-National компонента, который бы показывал нулевые различия при использовании программы COMTEST. Также следует отметить, что у National было пять версий 16550 за эти годы, и новейшие компоненты ведут себя несколько иначе, чем классический NS16550AFN, который считается эталоном функциональности. COMTEST, по-видимому, закрывает глаза на различия внутри линейки продуктов National и не сообщает об ошибках в компонентах National (за исключением оригинальной 16550), даже когда

> существуют официальные errata, описывающие ошибки в ревизиях A, B и C этих компонентов, поэтому эту предвзятость COMTEST необходимо учитывать.

Важно понимать, что простое подсчитывание различий с COMTEST не даёт полного представления о том, какие различия существенны, а какие нет. Например, около половины различий, обнаруженных в двух вышеупомянутых модемах с внутренними UART, были вызваны тем, что клоновые UART не поддерживают режимы пяти- и шестибитных символов. Настоящие UART 16550, 16450 и 8250 поддерживают эти режимы, и COMTEST проверяет их функциональность, поэтому фиксируется более пятидесяти различий. Однако почти ни один современный модем не поддерживает пяти- или шестибитные символы, особенно те, что обладают функциями коррекции ошибок и сжатия. Это означает, что различия, связанные с режимами пяти- и шестибитных символов, можно не учитывать.

Многие различия, о которых сообщает COMTEST, связаны с временными характеристиками. Во многих клонированных конструкциях, когда хост читает из одного порта, статусные биты в другом порте могут обновляться с иной скоростью (быстрее или медленнее), чем у *настоящего* NS16550AFN, и COMTEST выявляет эти различия. Это означает, что количество различий может вводить в заблуждение: одно устройство может иметь всего одно или два различия, но они крайне критичны, тогда как другое устройство, обновляющее статусные регистры быстрее или медленнее эталонной части (что, вероятно, никогда не повлияет на работу правильно написанного драйвера), может иметь десятки зарегистрированных различий.

COMTEST можно использовать в качестве инструмента проверки, чтобы предупредить администратора о наличии потенциально несовместимых компонентов, которые могут вызвать проблемы или потребуют особого подхода.

Если вы запускаете COMTEST на 16550, который находится в модеме или к модему подключён последовательный порт, необходимо сначала отправить модему команду ATE0&W, чтобы модем не эхо-повторял ни один из тестовых символов. Если вы забудете это сделать, COMTEST сообщит как минимум об одном различии:

```
Error (6)...Timeout interrupt failed: IIR = c1  LSR = 61
```

1.8. Регистры 8250/16450/16550

UART 8250/16450/16550 занимает восемь последовательных адресов портов ввода-вывода. В IBM PC определены два расположения для этих восьми портов, которые вместе известны как COM1 и COM2. Производители PC-клонов и дополнительных карт создали два дополнительных области, известных как COM3 и COM4, но эти дополнительные COM-порты конфликтуют с другим оборудованием на некоторых системах. Наиболее распространённый конфликт возникает с видеоадаптерами, обеспечивающими эмуляцию IBM 8514.

COM1 находится в диапазоне от 0x3f8 до 0x3ff и обычно использует IRQ 4. COM2 находится в диапазоне от 0x2f8 до 0x2ff и обычно использует IRQ 3. COM3 находится в диапазоне от 0x3e8

до 0x3ef и не имеет стандартного IRQ. COM4 находится в диапазоне от 0x2e8 до 0x2ef и не имеет стандартного IRQ.

Описание портов ввода-вывода UART 8250/16450/16550 представлено ниже.

Порт ввода/вывода	**Доступ Разрешен**	**Описание**
+0x00	запись (DLAB==0)	Регистр передачи данных (THR). Информация, записанная в этот порт, обрабатывается как слова данных и передаётся через UART.
+0x00	чтение (DLAB==0)	Регистр буфера приема (RBR). Любые слова данных, полученные UART из последовательного соединения, доступны для чтения хостом через этот порт.
+0x00	запись/чтение (DLAB==1)	Младший байт защелки делителя (DLL — Divisor Latch LSB) Это значение будет поделено от основного входного тактового сигнала (в IBM PC основной тактовый сигнал равен 1,8432 МГц), и полученный тактовый сигнал будет определять скорость передачи UART. Этот регистр содержит биты с 0 по 7 делителя.
+0x01	запись/чтение (DLAB==1)	Старший байт защелки делителя (DLH — Divisor Latch MSB) Это значение будет разделено от основного входного тактового сигнала (в IBM PC основной тактовый сигнал равен 1,8432 МГц), и полученный тактовый сигнал будет определять скорость передачи данных UART. Этот регистр содержит биты с 8 по 15 делителя.

Порт ввода/вывода	Доступ Разрешен	Описание
+0x01	запись/чтение (DLAB==0)	Регистр разрешения прерываний (IER) UART 8250/16450/1655 классифицирует события на четыре категории. Каждая категория может быть настроена на генерацию прерывания при возникновении любого из событий. UART 8250/16450/16550 генерирует единый внешний сигнал прерывания независимо от того, сколько событий в разрешённых категориях произошло. Задача главного процессора — обработать прерывание и затем опросить разрешённые категории прерываний (обычно прерывания разрешены для всех категорий), чтобы определить истинную причину(ы) прерывания. Бит 7 → Зарезервирован, всегда 0. Бит 6 → Зарезервирован, всегда 0. Бит 5 → Зарезервирован, всегда 0. Бит 4 → Зарезервирован, всегда 0. Бит 3 → Разрешение прерывания по состоянию модема (EDSSI). Установка этого бита в "1" позволяет UART генерировать прерывание при изменении состояния одной или нескольких линий статуса. Бит 2 → Разрешение прерывания по состоянию линии приёмника (ELSI). Установка этого бита в "1" приводит к генерации прерывания UART при обнаружении ошибки (или сигнала BREAK) во входящих данных. Бит 1 → Разрешение прерывания по опустошению регистра передатчика (ETBEI). Установка этого бита в "1" приводит к генерации прерывания UART, когда в UART появляется место для одного или более дополнительных символов, предназначенных для передачи. Бит 0 → Разрешение прерывания по наличию принятых данных (ERBFI). Установка этого бита в "1" приводит к генерации прерывания UART, когда UART принял достаточное количество символов для превышения порога FIFO, или истекло время ожидания FIFO (устаревшие данные), или принят одиночный символ при отключённом FIFO.

Порт ввода/вывода	Доступ Разрешен	Описание
+0x02	запись	Регистр управления FIFO (FCR — FIFO Control Register) (Этот порт отсутствует в UART 8250 и 16450.) Бит 7 → Бит триггера приемника #1 Бит 6 → Бит триггера приемника #0 Эти два бита определяют, при каком количестве данных приемник должен генерировать прерывание, когда FIFO активен. 7 6 Количество слов перед генерацией прерывания 0 0 1 0 1 4 1 0 8 1 1 14 Бит 5 → Зарезервирован, всегда 0. Бит 4 → Зарезервирован, всегда 0. Бит 3 → Выбор режима DMA. Если бит 0 установлен в "1" (FIFO включены), установка этого бита изменяет работу сигналов -RXRDY и -TXRDY с режима 0 на режим 1. Бит 2 → Сброс передающего FIFO. При записи "1" в этот бит содержимое FIFO очищается. Любое слово, которое передаётся в данный момент, будет отправлено полностью. Эта функция полезна для прерывания передачи. Бит 1 → Сброс приемного FIFO. При записи "1" в этот бит содержимое FIFO очищается. Любое слово, которое в данный момент собирается в сдвиговом регистре, будет принято полностью. Бит 0 → Включение FIFO 16550. При установке этого бита активируются как передающий, так и приемный FIFO. Любое содержимое в регистре хранения, сдвиговых регистрах или FIFO теряется при включении или отключении FIFO.

Порт ввода/вывода	Доступ Разрешен	Описание
+0x02	чтение	Регистр идентификации прерываний Бит 7 → FIFO включены. На UART 8250/16450 этот бит равен нулю. Бит 6 → FIFO включены. На UART 8250/16450 этот бит равен нулю. Бит 5 → Зарезервирован, всегда 0. Бит 4 → Зарезервирован, всегда 0. Бит 3 → Бит идентификатора прерывания №2. На UART 8250/16450 этот бит равен нулю. Бит 2 → Бит идентификатора прерывания №1 Бит 1 → Бит идентификатора прерывания №0.Эти три бита объединяются для указания категории события, вызвавшего текущее прерывание. Эти категории имеют приоритеты, поэтому, если несколько категорий событий происходят одновременно, UART сообщит о более важных событиях первыми, и хост должен обрабатывать события в порядке их поступления. Все события, вызвавшие текущее прерывание, должны быть обработаны до генерации новых прерываний. (Это ограничение архитектуры ПК.) 2 1 0 Приоритет Описание 0 1 1 Первый Принятая ошибка (OE, PE, BI или FE) 0 1 0 Второй Доступны принятые данные 1 1 0 Второй Идентификация уровня триггера (Устаревшие данные в буфере приема) 0 0 1 Третий Передатчик готов принять больше данных (THRE) 0 0 0 Четвертый Изменение состояния модема (-CTS, -DSR, -RI или -DCD) Бит 0 → Бит ожидания прерывания. Если этот бит установлен в "0", то как минимум одно прерывание ожидает обработки.

Порт ввода/вывода	Доступ Разрешен	Описание
+0x03	запись/чтение	Регистр управления линией (LCR — Line Control Register) Бит 7 → Бит доступа к защелке делителя (DLAB). При установке доступ к регистру передачи/приема данных (THR/RBR) и регистру разрешения прерываний (IER) отключается. Любой доступ к этим портам перенаправляется к регистрам защелки делителя. Установка этого бита, загрузка регистров делителя и сброс DLAB должны выполняться при отключённых прерываниях. Бит 6 → Установка прерывания. При установке в "1" передатчик начинает передавать непрерывный интервал (Spacing), пока этот бит не будет сброшен в "0". Это переопределяет любые передаваемые биты символов. Бит 5 → Фиксированный бит чётности. При включённой проверке чётности установка этого бита приводит к тому, что бит чётности всегда будет "1" или "0" в зависимости от значения бита 4. Бит 4 → Выбор чётности (EPS). При включённой проверке чётности и если бит 5 равен "0", установка этого бита приводит к использованию и ожиданию четной чётности. В противном случае используется нечетная чётность. Бит 3 → Разрешение проверки чётности (PEN). При установке в "1" бит чётности вставляется между последним битом данных и стоповым битом. UART также ожидает наличие бита чётности в принимаемых данных. Бит 2 → Количество стоповых битов (STB). Если установлен в "1" и используются 5-битные слова данных, передаётся и ожидается 1.5 стоповых бита в каждом слове данных. Для 6, 7 и 8-битных слов данных передаётся и ожидается 2 стоповых бита. Если этот бит сброшен в "0", используется один стоповый бит в каждом слове данных. Бит 1 → Бит выбора длины слова #1 (WLSB1) Бит 0 → Бит выбора длины слова #0 (WLSB0) Вместе эти биты определяют количество битов в каждом слове данных. 1 0 Длина слова 0 0 5 бит данных 0 1 6 бит данных 1 0 7 бит данных 1 1 8 бит данных

Порт ввода/вывода	Доступ Разрешен	Описание
+0x04	запись/чтение	Регистр управления модемом (MCR — Modem Control Register) Бит 7 → Зарезервирован, всегда 0. Бит 6 → Зарезервирован, всегда 0. Бит 5 → Зарезервирован, всегда 0. Бит 4 → Режим петли (Loop-Back). При установке в "1" передатчик и приёмник UART соединяются внутри для диагностики. Также выходы управления модемом UART подключаются к его входам: CTS к RTS, DTR к DSR, OUT1 к RI, а OUT2 к DCD. Бит 3 → OUT2. Вспомогательный выход, который процессор может установить в высокий или низкий уровень. В адаптере IBM PC (и большинстве клонов) OUT2 используется для отключения сигнала прерывания от UART 8250/16450/16550. Бит 2 → OUT1. Вспомогательный выход, который процессор может установить в высокий или низкий уровень. На адаптере IBM PC не используется. Бит 1 → Запрос на передачу (RTS). При установке в "1" выход линии -RTS UART переходит в низкий уровень (активное состояние). Бит 0 → Готовность терминала данных (DTR). При установке в "1" выход линии -DTR UART переходит в низкий уровень (активное состояние).

Порт ввода/вывода	Доступ Разрешен	Описание
+0x05	запись/чтение	Регистр состояния линии (LSR — Line Status Register) Бит 7 → Ошибка в FIFO приемника. На UART 8250/16450 этот бит равен нулю. Этот бит устанавливается в «1», когда любой из байтов в FIFO имеет одно или несколько из следующих условий ошибки: PE, FE или BI. Бит 6 → Передатчик пуст (TEMT). Когда установлен в «1», в FIFO передатчика или сдвиговом регистре передатчика не осталось слов. Передатчик полностью бездействует. Бит 5 → Регистр хранения передатчика пуст (THRE). Когда установлен в «1», в FIFO (или регистре хранения) теперь есть место для передачи как минимум одного дополнительного слова. Передатчик может все ещё передавать данные, когда этот бит установлен в «1». Бит 4 → Прерывание по Break (BI). Приемник обнаружил сигнал Break. Бит 3 → Ошибка кадрирования (FE). Обнаружен стартовый бит, но стоповый бит не появился в ожидаемое время. Принятое слово, вероятно, искажено. Бит 2 → Ошибка чётности (PE). Бит чётности для принятого слова был некорректен. Бит 1 → Ошибка переполнения (OE). Было получено новое слово, но в буфере приема не было места. Вновь поступившее слово в сдвиговом регистре отбрасывается. На UART 8250/16450 слово в регистре хранения отбрасывается, а вновь поступившее слово помещается в регистр хранения. Бит 0 → Данные готовы (DR). Одно или несколько слов находятся в FIFO приемника, которые хост может прочитать. Слово должно быть полностью принято и перемещено из сдвигового регистра в FIFO (или регистр хранения для 8250/16450) до того, как этот бит будет установлен.

Порт ввода/вывода	Доступ Разрешен	Описание
+0x06	запись/чтение	Регистр состояния модема (MSR — Modem Status Register) Бит 7 → Обнаружение несущей данных (DCD). Отражает состояние линии DCD на UART. Бит 6 → Индикатор вызова (RI). Отражает состояние линии RI на UART. Бит 5 → Готовность передатчика данных (DSR). Отражает состояние линии DSR на UART. Бит 4 → Готовность к приёму (CTS). Отражает состояние линии CTS на UART. Бит 3 → Изменение состояния обнаружения несущей данных (DDCD). Устанавливается в "1", если линия -DCD изменила состояние ещё раз с момента последнего чтения MSR хостом. Бит 2 → Фронт сигнала вызова (TERI). Устанавливается в "1", если линия -RI перешла из низкого уровня в высокий с момента последнего чтения MSR хостом. Бит 1 → Изменение состояния готовности передатчика данных (DDSR). Устанавливается в "1", если линия -DSR изменила состояние ещё раз с момента последнего чтения MSR хостом. Бит 0 → Изменение состояния готовности к приёму (DCTS). Устанавливается в "1", если линия -CTS изменила состояние ещё раз с момента последнего чтения MSR хостом.
+0x07	запись/чтение	Регистр Scratch (SCR — Scratch Register). Этот регистр не выполняет никакой функции в UART. Хост может записать любое значение в это место и позднее считать его.

1.9. За пределами UART 16550A

Хотя National Semiconductor не предлагала никаких компонентов, совместимых с 16550 и предоставляющих дополнительные функции, другие производители сделали это. Некоторые из этих компонентов описаны ниже. Следует понимать, что для эффективного использования этих улучшений могут потребоваться драйверы от производителя чипа, поскольку большинство популярных операционных систем не поддерживают функции, выходящие за рамки возможностей 16550.

ST16650

По умолчанию эта часть аналогична NS16550A, но дополнительно можно включить расширенный 32-байтовый буфер отправки и приёма. Производитель — StarTech.

TIL16660

По умолчанию эта часть ведёт себя аналогично NS16550A, но дополнительно может быть включён расширенный 64-байтный буфер передачи и приёма. Производится Texas Instruments.

Hayes ESP

Эта проприетарная внешняя карта содержит буфер передачи и приема размером 2048 байт и поддерживает скорость передачи данных до 230,4 Кбит/с. Произведено компанией Hayes.

В дополнение к этим "простым" UART многие производители выпускают интеллектуальные платы для последовательной связи. Такой тип конструкции обычно включает микропроцессор, который взаимодействует с несколькими UART, обрабатывает и буферизует данные, а затем при необходимости уведомляет основной процессор ПК. Поскольку в такой системе связи UART не доступны напрямую процессору ПК, производителю не обязательно использовать UART, совместимые с 8250, 16450 или 16550. Это даёт разработчику свободу выбора компонентов с лучшими характеристиками производительности.

2. Настройка драйвера sio

Драйвер sio обеспечивает поддержку интерфейсов связи EIA RS-232C (CCITT V.24) на основе NS8250, NS16450, NS16550 и NS16550A. Также поддерживаются несколько многопортовых карт. Подробную техническую документацию смотрите на sio(4).

2.1. Digi International (DigiBoard) PC/8

Предоставлено Andrew Webster <awebster@pubnix.net>. *26 августа 1995.*

Вот фрагмент конфигурации с машины, на которой установлена плата Digi International PC/8 с чипом 16550. К ней подключено 8 модемов, работающих на этих 8 линиях, и они отлично функционируют. Не забудьте добавить `options COM_MULTIPORT`, иначе работа будет нестабильной!

```
device          sio4    at isa? port 0x100 flags 0xb05
device          sio5    at isa? port 0x108 flags 0xb05
device          sio6    at isa? port 0x110 flags 0xb05
device          sio7    at isa? port 0x118 flags 0xb05
device          sio8    at isa? port 0x120 flags 0xb05
device          sio9    at isa? port 0x128 flags 0xb05
device          sio10   at isa? port 0x130 flags 0xb05
device          sio11   at isa? port 0x138 flags 0xb05 irq 9
```

Хитрость настройки заключается в том, что старший бит флагов представляет последний порт SIO, в данном случае 11, поэтому флаги равны 0xb05.

2.2. Boca 16

Предоставлено Don Whiteside <whiteside@acm.org>. *26 августа 1995.*

Процедуры по настройке платы Boca с 16 портами в FreeBSD довольно просты, но вам

понадобится несколько вещей для успешной работы:

1. Вам необходимо либо установить исходные коды ядра, чтобы перекомпилировать нужные опции, либо найти кого-то, кто сделает это за вас. Стандартное ядро версии 2.0.5 *не* включает поддержку нескольких портов, и в любом случае вам потребуется добавить запись устройства для каждого порта.
2. Два, вам нужно знать прерывание и настройку ввода-вывода для вашей платы Boca, чтобы правильно установить эти параметры в ядре.

Важное замечание — реальные микросхемы UART для Boca 16 находятся в соединительной коробке, а не на внутренней плате. Поэтому, если она отключена, попытки проверить эти порты завершатся неудачей. Я никогда не проверял загрузку с отключённой коробкой и последующим её подключением, и не рекомендую вам этого делать.

Если у вас ещё нет настроенного файла конфигурации пользовательского ядра, обратитесь к разделу Конфигурация ядра в руководстве FreeBSD для получения общих инструкций. Ниже приведены конкретные настройки для платы Boca 16, предполагается, что вы используете ядро с именем MYKERNEL и редактируете его с помощью vi.

1. Добавьте строку

   ```
   options COM_MULTIPORT
   ```

   в конфигурационный файл.

2. Где находятся текущие строки `device sio`, вам нужно добавить ещё 16 устройств. В следующем примере показана плата Boca Board с прерыванием 3 и базовым адресом ввода-вывода 100h. Адрес ввода-вывода для каждого порта увеличивается на 8 в шестнадцатеричной системе относительно предыдущего порта, поэтому адреса будут 100h, 108h, 110h…

   ```
   device sio1 at isa? port 0x100 flags 0x1005
   device sio2 at isa? port 0x108 flags 0x1005
   device sio3 at isa? port 0x110 flags 0x1005
   device sio4 at isa? port 0x118 flags 0x1005
   ...
   device sio15 at isa? port 0x170 flags 0x1005
   device sio16 at isa? port 0x178 flags 0x1005 irq 3
   ```

   Запись flags *обязательно* должна быть изменена по сравнению с этим примером, если вы не используете точно такие же назначения sio. Флаги устанавливаются в соответствии с 0x`MYY`, где *M* обозначает младший номер главного порта (последний порт на Boca 16), а *YY* указывает, включен или выключен FIFO (включен), используется ли разделение IRQ (да) и есть ли регистр управления IRQ, совместимый с AST/4 (нет). В этом примере,

```
flags
        0x1005
```

указывает, что основной порт - sio16. Если добавить другую плату и назначить порты с sio17 по sio28, флаги для всех 16 портов на *этой* плате будут 0x1C05, где 1C обозначает минорный номер основного порта. Не изменяйте значение 05.

3. Сохраните и завершите конфигурацию ядра, перекомпилируйте, установите и перезагрузитесь. Предполагая, что вы успешно установили перекомпилированное ядро и настроили правильный адрес и IRQ, сообщение при загрузке должно указывать на успешное обнаружение портов Boca следующим образом: (очевидно, номера sio, IO и IRQ могут отличаться)

```
sio1 at 0x100-0x107 flags 0x1005 on isa
sio1: type 16550A (multiport)
sio2 at 0x108-0x10f flags 0x1005 on isa
sio2: type 16550A (multiport)
sio3 at 0x110-0x117 flags 0x1005 on isa
sio3: type 16550A (multiport)
sio4 at 0x118-0x11f flags 0x1005 on isa
sio4: type 16550A (multiport)
sio5 at 0x120-0x127 flags 0x1005 on isa
sio5: type 16550A (multiport)
sio6 at 0x128-0x12f flags 0x1005 on isa
sio6: type 16550A (multiport)
sio7 at 0x130-0x137 flags 0x1005 on isa
sio7: type 16550A (multiport)
sio8 at 0x138-0x13f flags 0x1005 on isa
sio8: type 16550A (multiport)
sio9 at 0x140-0x147 flags 0x1005 on isa
sio9: type 16550A (multiport)
sio10 at 0x148-0x14f flags 0x1005 on isa
sio10: type 16550A (multiport)
sio11 at 0x150-0x157 flags 0x1005 on isa
sio11: type 16550A (multiport)
sio12 at 0x158-0x15f flags 0x1005 on isa
sio12: type 16550A (multiport)
sio13 at 0x160-0x167 flags 0x1005 on isa
sio13: type 16550A (multiport)
sio14 at 0x168-0x16f flags 0x1005 on isa
sio14: type 16550A (multiport)
sio15 at 0x170-0x177 flags 0x1005 on isa
sio15: type 16550A (multiport)
sio16 at 0x178-0x17f irq 3 flags 0x1005 on isa
sio16: type 16550A (multiport master)
```

Если сообщения проходят слишком быстро, чтобы их увидеть,

```
# dmesg | more
```

покажет вам сообщения загрузки.

4. Далее необходимо создать соответствующие записи в /dev для устройств с помощью скрипта /dev/MAKEDEV. Этот шаг можно пропустить, если вы используете FreeBSD 5.X с ядром, в котором включена поддержка devfs(5).

   Если вам необходимо создать записи в /dev, выполните следующую команду от имени `root`:

   ```
   # cd /dev
   # ./MAKEDEV tty1
   # ./MAKEDEV cua1

   (everything in between)
   # ./MAKEDEV ttyg
   # ./MAKEDEV cuag
   ```

   Если по какой-то причине вам не нужны или не требуются устройства исходящих соединений, вы можете обойтись без создания устройств cua*.

5. Если вам нужен быстрый и небрежный способ убедиться, что устройства работают, вы можете просто подключить модем к каждому порту и (как root)

   ```
   # echo at > ttyd*
   ```

   для каждого устройства, которое вы создали. Вы *должны* увидеть, как мигают индикаторы RX для каждого рабочего порта.

2.3. Поддержка дешёвых многоканальных UART-карт

Предоставлено Хельге Ольдахом hmo@sep.hamburg.com, сентябрь 1999 года

Вы когда-нибудь задумывались о поддержке FreeBSD вашей 20-долларовой многофункциональной карты с двумя (или более) COM-портами, разделяющими IRQ? Вот как это сделать:

Обычно единственный способ поддержки таких плат — использование отдельного IRQ для каждого порта. Например, если ваша материнская плата имеет встроенный порт COM1 (он же sio0 — адрес ввода-вывода 0x3F8 и IRQ 4), а у вас есть расширительная плата с двумя UART, то обычно их нужно настроить как COM2 (он же sio1 — адрес ввода-вывода 0x2F8 и IRQ 3), а третий порт (он же sio2) — с адресом 0x3E8 и IRQ 5. Очевидно, это расточительное использование ресурсов IRQ, так как в принципе возможно запустить оба порта

расширительной платы с одним IRQ, используя конфигурацию `COM_MULTIPORT`, описанную в предыдущих разделах.

Такие недорогие платы ввода-вывода обычно имеют перемычечную матрицу 4x3 для COM-портов, подобную следующей:

```
           o  o  o  *
Port A                |
           o  *  o  *
Port B        |
           o  *  o  o
IRQ        2  3  4  5
```

Показано, что порт A подключен для IRQ 5, а порт B — для IRQ 3. Столбцы IRQ на вашей конкретной плате могут отличаться — другие платы могут предоставлять перемычки для IRQ 3, 4, 5 и 7.

Можно было бы сделать вывод, что подключение обоих портов к IRQ 3 с помощью самодельной перемычки, замыкающей все три точки соединения в колонке IRQ 3, решит проблему, но это не так. Невозможно дублировать IRQ 3, потому что выходные драйверы каждого UART соединены по схеме "монтажное И", и если один из UART управляет IRQ 3, выходной сигнал будет не таким как ожидается. В зависимости от реализации платы расширения или материнской платы, линия IRQ 3 будет постоянно находиться в высоком уровне или всегда оставаться низкой.

Вам необходимо разделить драйверы прерываний для двух UART, чтобы линия прерывания платы поднималась только тогда (и только тогда), когда один из UART вызывает прерывание, и оставалась низкой в противном случае. Решение было предложено Йоргом Вуншем j@ida.interface-business.de: припаять монтажную схему "монтажное ИЛИ", состоящую из двух диодов (предпочтительно германиевых или типа Шоттки) и резистора на 1 кОм. Вот схема, начиная с контактного поля 4x3 выше:

```
                        Diode
              +---------->|-------+
             /                    |
           o  *  o  o             |     1 kOhm
Port A                            +----|######|-------+
           o  *  o  o             |                    |
Port B          `-----------------+                ==+==
           o  *  o  o             |                Ground
              \                   |
               +--------->|-------+
IRQ        2  3  4  5    Diode
```

Катоды диодов соединены в общей точке вместе с подтягивающим резистором 1 кОм. Важно подключить резистор к земле, чтобы избежать плавания линии IRQ на шине.

Теперь мы готовы настроить ядро. Продолжая этот пример, мы настроим:

```
# standard on-board COM1 port
device          sio0    at isa? port "IO_COM1" flags 0x10
# patched-up multi-I/O extension board
options         COM_MULTIPORT
device          sio1    at isa? port "IO_COM2" flags 0x205
device          sio2    at isa? port "IO_COM3" flags 0x205 irq 3
```

Обратите внимание, что настройка `flags` для sio1 и sio2 действительно важна; подробности смотрите в sio(4). (Обычно `2` в атрибуте "flags" относится к sio2, который содержит IRQ, и вам наверняка потребуется нижний ниббл `5`.) При включённом режиме подробного вывода ядра это должно дать что-то похожее на следующее:

```
sio0: irq maps: 0x1 0x11 0x1 0x1
sio0 at 0x3f8-0x3ff irq 4 flags 0x10 on isa
sio0: type 16550A
sio1: irq maps: 0x1 0x9 0x1 0x1
sio1 at 0x2f8-0x2ff flags 0x205 on isa
sio1: type 16550A (multiport)
sio2: irq maps: 0x1 0x9 0x1 0x1
sio2 at 0x3e8-0x3ef irq 3 flags 0x205 on isa
sio2: type 16550A (multiport master)
```

Хотя /sys/i386/isa/sio.c выглядит несколько загадочно из-за использования массива "irq maps" выше, основная идея заключается в том, что вы наблюдаете `0x1` на первой, третьей и четвертой позициях. Это означает, что соответствующий IRQ был установлен при выводе и сброшен после, что полностью соответствует ожиданиям. Если ваше ядро не демонстрирует такое поведение, скорее всего, проблема в вашей разводке.

3. Настройка драйвера cy

Предоставлено Алексом Нэшем. 6 июня 1996.

Многопортовые карты Cyclades основаны на драйвере cy, а не на обычном драйвере sio, используемом другими многопортовыми картами. Настройка сводится к простым действиям:

1. Добавьте устройство cy в конфигурацию ядра (обратите внимание, что параметры irq и iomem могут отличаться).

   ```
   device cy0 at isa? irq 10 iomem 0xd4000 iosiz 0x2000
   ```

2. Перестройте и установите новый образ ядра.
3. Создайте файлы устройств, введя (следующий пример предполагает 8-портовую плату):

```
# cd /dev
# for i in 0 1 2 3 4 5 6 7;do ./MAKEDEV cuac$i ttyc$i;done
```

4. Если необходимо, добавьте записи для коммутируемого доступа в /etc/ttys, дублируя записи для последовательных устройств (`ttyd`) и используя `ttyc` вместо `ttyd`. Например:

```
ttyc0   "/usr/libexec/getty std.38400"  unknown on insecure
ttyc1   "/usr/libexec/getty std.38400"  unknown on insecure
ttyc2   "/usr/libexec/getty std.38400"  unknown on insecure
...
ttyc7   "/usr/libexec/getty std.38400"  unknown on insecure
```

5. Перезагрузитесь с новым ядром.

4. Настройка драйвера si

Предоставлено Nick Sayer <nsayer@FreeBSD.org>. *25 марта 1998.*

Специальные мультипортные карты Specialix SI/XIO и SX используют драйвер si. На одной машине может быть установлено до 4 хост-карт. Поддерживаются следующие хост-карты:

- ISA SI/XIO host card (2 versions)
- EISA SI/XIO host card
- PCI SI/XIO host card
- ISA SX host card
- PCI SX host card

Хотя хост-карты SX и SI/XIO выглядят заметно по-разному, их функциональность практически одинакова. Хост-карты не используют порты ввода-вывода, а вместо этого требуют 32K сегмента памяти. Заводская конфигурация для карт ISA размещает этот сегмент по адресу `0xd0000-0xd7fff`. Также им требуется IRQ. Карты PCI, разумеется, настраиваются автоматически.

Вы можете подключить до 4 внешних модулей к каждой карте хоста. Внешние модули содержат либо 4, либо 8 последовательных портов. Они бывают следующих видов:

- Модули SI на 4 или 8 портов. Поддерживается скорость до 57600 бит/с на каждом порту.
- XIO 8-портовые модули. Поддерживается скорость до 115200 бит/с на каждом порту. Один из типов модулей XIO имеет 7 последовательных и 1 параллельный порт.
- Модули SXDC с 8 портами. Поддерживается скорость до 921600 бит/с на каждом порту. Как и в случае с XIO, доступен модуль с одним параллельным портом.

Для настройки карты хоста ISA добавьте следующую строку в файл конфигурации ядра,

изменив числа по мере необходимости:

```
device si0 at isa? iomem 0xd0000 irq 11
```

Допустимые номера IRQ: 9, 10, 11, 12 и 15 для SX ISA host cards и 11, 12 и 15 для SI/XIO ISA host cards.

Для настройки карты EISA или PCI используйте следующую строку:

```
device si0
```

После добавления записи конфигурации пересоберите и установите свое новое ядро.



Следующий шаг не обязателен, если вы используете devfs(5) в FreeBSD 5.*X*.

После перезагрузки с новым ядром необходимо создать файлы устройств в /dev. Скрипт MAKEDEV выполнит эту задачу за вас. Подсчитайте общее количество портов и введите:

```
# cd /dev
# ./MAKEDEV ttyAnn cuaAnn
```

(где *nn* — количество портов)

Если вы хотите, чтобы приглашения к входу отображались на этих портах, вам нужно добавить такие строки в /etc/ttys:

```
ttyA01  "/usr/libexec/getty std.9600"   vt100   on insecure
```

Измените тип терминала по необходимости. Для модемов подойдут `dialup` или `unknown`.