**Программирование com-порта**

Com-порт реализован на микросхеме UART

**Программная реализация com порта**

UART состоит из 12 регистров, к которым можно обратиться по восьми адресам портов ввода-вывода. Так как индивидуального адреса для каждого регистра не хватает, то используют расщепление адресного пространства с помощью следующих методов:

*1. Разделение одного адресного пространства на два регистра по записи/чтению*

По сигналу чтения RD=0 читается один регистр, по сигналу записи WR=0 записывается второй регистр.

То есть данные по одинаковому адресу записываются или читаются с разных регистров.

Таких регистров четыре:

THR, RBR - по адресу UART 00h(A2A1A0=000)

IIR, FOR - по адресу UART 02h(A2A1A0=010)

Эти регистры односторонние, то есть в одни можно только записывать, в другие только читать данные.

2. *Использование дополнительного адресного бита*

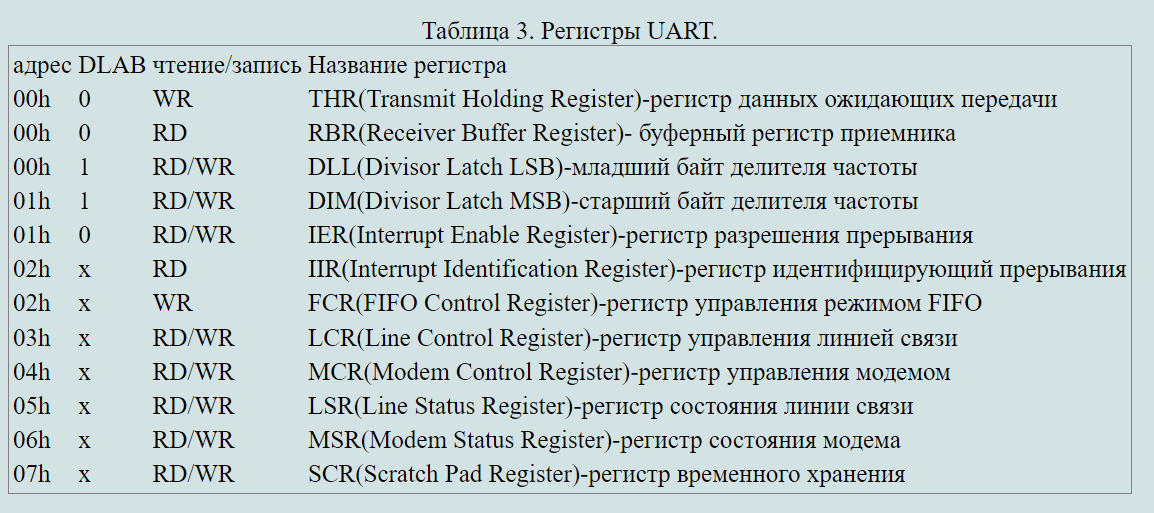
Используют 7-ой бит регистра LCR-находящегося по адресу UART 03h(A2A1A0=011).

Этот бит называют DLAB, если DLAB=0, то для чтения/записи используется один регистр,

если DLAB=1, то для чтения/записи используется второй регистр.

Таких регистров пять:

(THR & RBR), DLL - по адресу UART 00h(A2A1A0=000)

DIM, IER - по адресу UART 01h(A2A1A0=001)

**THR** *(Transmit Holding Register)* – для записи данных (помещения символа (5-8 бит) на линию связи)

Для указания того, что регистр THR пуст и в него можно загрузить очередной байт данных используют бит 5 регистра LSR. Этот бит называется THRE(Transmitter Holding Register Empty) -"регистр данных ожидающих передачи пуст". Если THRE=1, то в регистр THR можно посылать очередной байт данных, в режиме FIFO этот бит говорит о том, что регистр FIFO пуст и можно посылать следующий пакет байтов данных. Бит THRE может быть источником прерывания CPU.

**RBR***(Receiver Buffer Register)* – для чтения данных (принимает данные из приёмного сдвигающегося регистра).

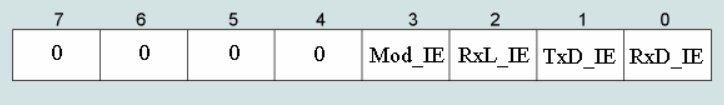
К моменту заполнения сдвигающего приёмного регистра регистр RBR должен быть освобожден для приема очередного байта, иначе произойдет ошибка переполнения. Освобождение регистра RBR происходит, когда данные из него читаются микропроцессором.

О том, что байт готов к прочтению микропроцессором (т.е. полностью выгрузился из приемного сдвигающегося регистра или FIFO) сообщает бит 0 регистра LSR. Этот бит называется DR (Receiver Data Ready) -"Данные приёмника готовы". DR=1 говорит о том, что регистр RBR(или FIFO) содержит принятый байт и его необходимо прочитать, DR сбрасывается в ноль после прочтения регистра RBR микропроцессором.

**DLL** *(Divisor Latch LSB)* – младший байт делителя частоты

**DIM** *(Divisor Latch MSB)* – старший байт делителя частоты

Далее см рис 1.

**IER** (*Interrupt Enable Register*) – регистр разрешения прерываний

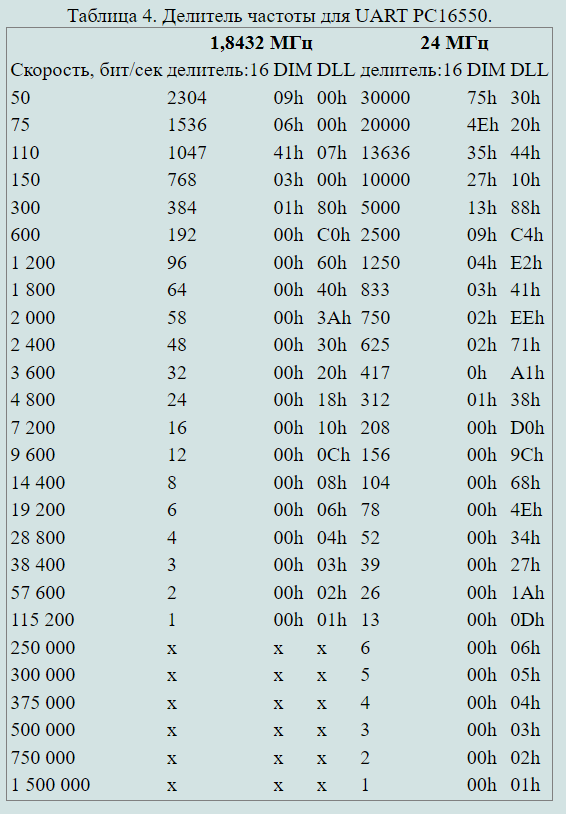
**RxD\_IЕ** – бит разрешения (если =1) прерывания приёма данных из RBR

**TxD\_IE** – если TxD\_IEЕ=1, то разрешено прерывание для передачи данных, это прерывание возникает когда передающий буфер пуст и необходимо загрузить байт в регистр THR

Бит 2. **RxL\_IЕ** – если RxL\_IЕ=1, то разрешено прерывание при обрыве линии связи или ошибке в приёме данных, это прерывание возникает когда в регистре состояния линии связи LSR будут выставлены биты этих ошибок.

Бит 3. **Mod\_IЕ** – если Mod\_IЕ =1, то разрешено прерывание при изменении состояния любого из входных сигналов RST, CTS, DCD, RI, это прерывание возникает когда состояние входных сигналов COM-порта изменились.

Биты 4…7 всегда 0

Рисунок 1

**IIR** – регистр идентифицирующий прерывания (чтение)

*(Interrupt Identification Register)*

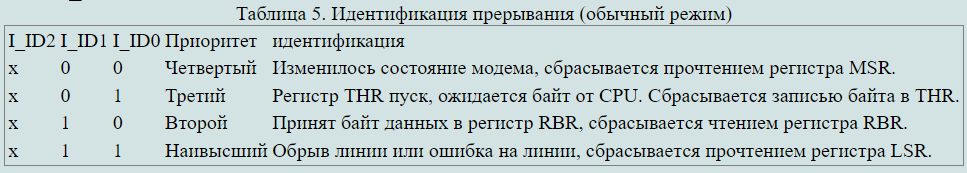
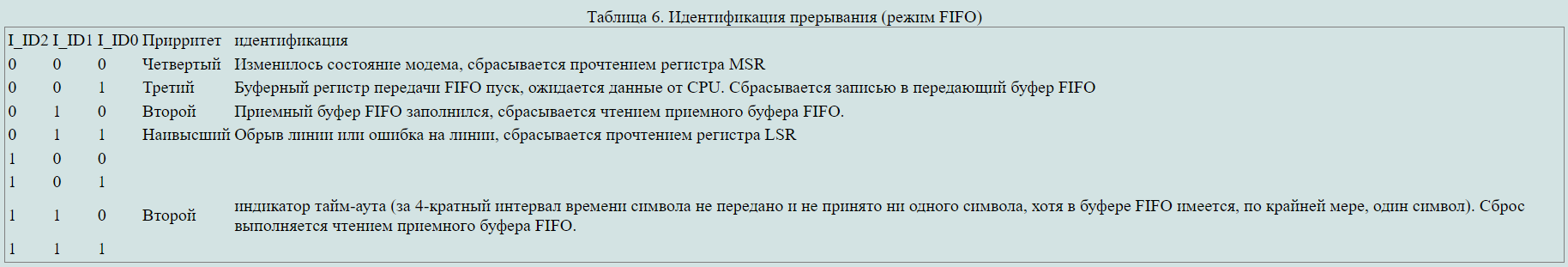
(Адрес=02h, RD)

Чтобы минимизировать программное обеспечение, UART располагает по приоритетам прерывания в четыре уровня и делает запись этих прерываний в IIR. Четыре уровня прерывания располагаются в порядке приоритета условий прерывания заданных регистрами - RLS; RDR; THR; и MSR. Когда CPU обращается к IIR, UART замораживает все прерывания и указывает самое высокое приоритетное отложенное прерывание для CPU. Во время обработки прерывания, UART делает запись новых прерываний, но не изменяет их текущий признак, до полной обработки.

Бит 0. **IP**(Interrupt Pending)— если IP=1, то все прерывания обработаны. Если IP=0,то есть необработанные прерывания.

Бит 1. **I\_ID0**(Interrupt ID Bit0)- нулевой бит идентификатора прерываний

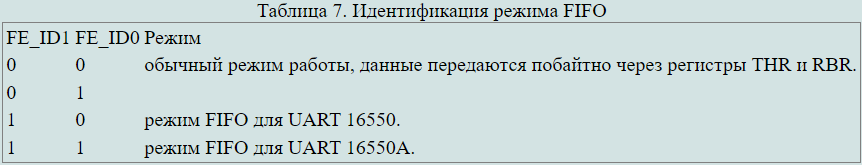
Бит 2. **I\_ID1**(Interrupt ID Bit1)- первый бит идентификатора прерываний

Бит 3. **I\_ID2**(Interrupt ID Bit2)- второй бит идентификатора прерываний

Бит 4..5.Зарезервированны

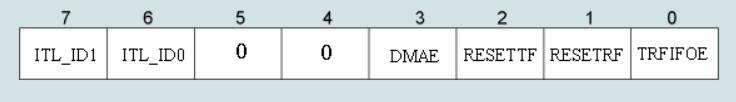
Бит 6. **FE\_ID0**(FIFOs Enabled ID Bit0)- нулевой бит идентификатора режима FIFO

Бит 7. **FE\_ID1**(FIFOs Enabled ID Bit1)- первый бит идентификатора режима FIFO



**FCR** – *регистр управления режимом FIFO (запись) (FIFO Control Register)*

Регистр FCR (Адрес=02h, WR)

Это регистр используется только для записи, его данные расположены как в регистре IIR. Этот регистр используется, чтобы разрешить режимы FIFO, очистить буферы FIFO, задать уровень заполнения буферов FIFO, и выбрать тип DMA(прямого обращения к памяти).

Бит 0. **TRFIFOE**(Transmit And Receive FIFO Enable)— Запись 1 в этот бит допускает оба режима FIFO передатчика(XMIT) и приемника(RCVR). Сброс бита в 0 очистит все байты в обоих буферов FIFO. При изменении режима FIFO к 16450 и наоборот, буферы FIFO автоматически очищаются. Этот бит должен быть в 1, когда производится запись других битов регистра FCR, иначе они не будут запрограммированы.

Бит 1. **RESETRF**(Reset Receiver FIFO)-Запись 1 в этот бит очищает все байты в приемном буфере FIFO и сбрасывает его счетчик в 0. Сдвиговый регистр при этом не очищается. После этого 1 в этом бите сбрасывается в 0.

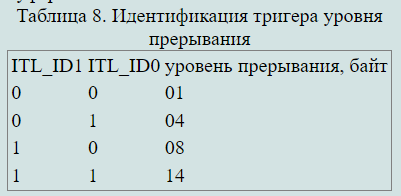
Бит 2. **RESETTF**(Reset Transmitter FIFO)- Запись 1 в этот бит очищает все байты в передающем буфере FIFO и сбрасывает его счетчик в 0. Сдвиговый регистр при этом не очищается. После этого 1 в этом бите сбрасывается в 0.

Бит 3. **DMAE**(DMA Enabled)- Запись 1 в этот бит приводит к изменению сигналов UART RxRDY и TxRDY с 0 к 1,при условии что FCR(bit0)=1. Эти аппаратные сигналы используются для организации правильной работы режима DMA в микропроцессорной системе.

Бит 4..5.Зарезервированные.

Бит 6. **ITL\_ID0** (Interrupt Trigger Level ID bit0) - нулевой бит идентификатора триггера уровня прерывания.

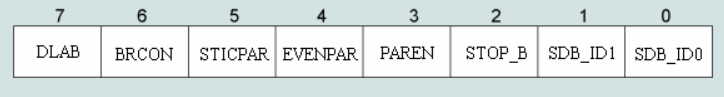
Бит 7. **ITL\_ID1**(Interrupt Trigger Level ID bit1)- первый бит идентификатора триггера уровня прерывания.



В этих двух битах задается идентификатор, который задает уровень при котором будет вырабатываться прерывание при приеме данных в режиме FIFO. Уровень задается количеством байт в приемном(RCVR) буфере FIFO.

**LCR***-регистр управления линией связи(запись/чтение)*

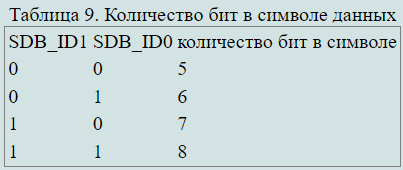
*(Line Control Register)*

Рис.21 Регистр LCR (Адрес=03h, RD/WR)

Данный регистр служит для определения(задания) формата асинхронного обмена передачи данных. Также в этом режиме устанавливается бит DLAB, который позволяет программисту записывать и читать данные из нужных регистров.. Программист может не только записывать, но и читать содержимое регистра LCR. Способность чтения упрощает системное программирование и устраняет потребность в отдельной области в системной памяти для хранения характеристик линии.

Бит 0. **SDB\_ID0**(Serial Data Bits ID0)- нулевой бит идентификатора количества бит в передаваемом символе.

Бит 1. **SDB\_ID1**(Serial Data Bits ID1)- первый бит идентификатора количества бит в передаваемом символе.

С помощью этих битов задают количество бит в передаваемом или принимаемом символе.

Бит 2. **STOP\_B**(Stop Bits)- Этот бит определяет число стоповых битов, переданных или полученных в каждом последовательном символе. Если бит STOP\_B=0, то передается один стоповый бит. Если бит STOP\_B=1, то стоповый бит равен двум для 6,7,8 битовых символов и полтора стоповых бита для 5-ти битовых символов. Приемник проверяет только первый стоповый бит, независимо от выставленных стоповых битов.

Бит 3. **PAREN**(Parity Enable) -Если PAREN=1, то разрешено использование бита паритета и данный бит вставляется между последним битом данных и стоповым битом. Если PAREN=0, то бит паритета не выставляется и не входит в состав передаваемого символа.

Бит 4. **EVENPAR**(Even Parity Select) - Бит выбора типа контроля паритета. Если EVENPAR=1, то происходит проверка на четность. Если EVENPAR=0, то происходит проверка на нечетность.

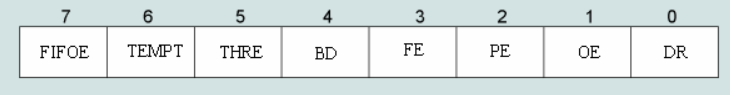
Бит 5. **STICPAR** (Sticky Parity)- Если STICPAR=0, то бит паритета бит генерируется в соответствии с паритетом выводимого символа. Если STICPAR=1, то постоянное значение контрольного бита: при EVENPAR=1 — нулевое, при EVENPAR=0 — единичное.

Бит 6. **BRCON**(Break Control)- Управляющий бит обрыва связи. Если BRCON=1, то вслучае возникновения перерыва в приеме данных, передатчик UART начнёт передавать в линию нули.

Бит 7. **DLAB**(Divisor Latch Access Bit)- Этот бит доступа к делителю частоты. Если DLAB=1, то можно обратиться к регистрам DIM, DLL в которых хранятся младший и старший байт делителя частоты :16.Если DLAB=0, то можно обратиться к регистрам THR,RBR,IER.

**LSR** – *регистр состояния линии связи (запись/чтение)*

*Line Status Register)*

Регистр LSR (Адрес=05h, RD/WR)

Данный регистр показывает состояние приемопередатчика.

Бит 0. **DR**(Receiver Data Ready) — Готовность данных приемника.DR=1 информирует о том, что данные приняты и загружены в регистр RBR или приемный буфер FIFO. Бит сбрасывается в ноль, когда все данные будут прочитаны CPU из регистра RBR или буфера FIFO.

Бит 1. **OE**(Overrun Error) — Бит ошибки переполнения. Бит указывает, что данные в регистре RBR не были прочитаны CPU прежде, чем следующий символ был передан в RBR, что привело к потере предыдущего символа. Бит устанавливается в OE=1 после обнаружения ошибки переполнения и сбрасывать всякий раз, когда SPU читает содержание регистра LSR.

Бит 2. **PE**(Parity Error) —Бит ошибки контрольного бита паритета.PE=1 если символ принят с ошибкой паритета.

Бит 3. **FE**(Framing Error) — ошибка кадра (неверный стопбит).

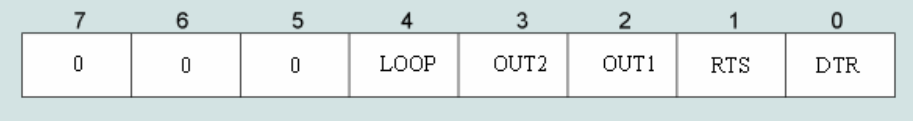
Бит 4. **BD**(Break Detected) — индикатор обрыва линии (вход приемника находится в состоянии 0 не менее чем время посылки символа).

Бит 5. **THRE**(Transmitter Holding Register Empty) — регистр передатчика готов принять байт для передачи. В режиме FIFO указывает на отсутствие символов в FIFO-буфере передачи. Может являться источником прерывания.

Бит 6. **TEMPT**(Transmitter Empty Status) — регистр передатчика пуст (нет данных для передачи ни в сдвиговом регистре, ни в буферных регистрах THR или FIFO).

Бит 7. **FIFOE**(FIFO Error Status) —ошибка принятых данных в режиме FIFO (буфер содержит хотя бы один символ, принятый с ошибкой формата, паритета или обрывом). В не FIFO-режиме всегда 0.

**MCR** *– регистр управления модемом или периферийным устройством (запись/чтение) (Modem Control Register)*

Регистр MCR (Адрес=04h, RD/WR)

Бит 0. **DTR**(Serial Data Bits ID0)(Data Terminal Ready)- Этот бит управляет выходным сигналом DTR (Готовность терминала данных).

Когда бит DTR=1, вывод DTR UART устанавливается в логический 0, в IBM XT этот сигнал инвертируется буферным инвертором DS1488(см.рис.12) в логическую 1 т.е. U= +12в (сигнал DTR COM-порта считается включенным)

Соответственно когда бит DTR=0, сигнал DTR COM-порта U= -12в логический 0 (сигнал DTR считается выключенным)

Бит 1. **RTS**(Request To Send )- Этот бит управляет выходным сигналом RTS (Запрос на передачу).

Когда бит RTS=1, вывод RTS UART устанавливается в логический 0, в IBM XT этот сигнал инвертируется буферным инвертором DS1488(см.рис.15) в логическую 1 т.е. U= +12в (сигнал RTS COM-порта считается включенным).

Соответственно когда бит RTS=0, сигнал RTS COM-порта U= -12в логический 0 (сигнал RTS считается выключенным)

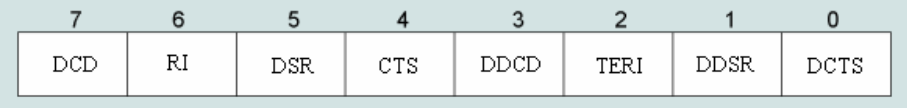
Бит 2. **OUT1**(OUT1 Bit Control) - Управление вспомогательным выходом OUT1.

Бит 3. **OUT2**(OUT2 Bit Control) - Управление вспомогательным выходом OUT2.

Бит 4. **LOOP**(Loopback Mode Enable)-Бит режима диагностики. Если LOOP=0, то UART работает в обычном режиме. Если LOOP=1, то URAT работет в режиме диагностики с обратной связью, в этом режиме используются вспомогательные сигналы OUT1 и OUT2.

Бит 5..7. Зарезервированы.

**MSR***-регистр состояния модема (Modem Status Register)*

Регистр MSR (Адрес=06h, RD/WR)

Этот регистр позволяет CPU контролировать текущее состояние линий управления модема или периферийного устройства. В дополнение к этому, четыре бита (0..3) регистра MSR контролируют изменения сигналов на входах CTS,RTS,RI,DCD микросхемы и вырабатывают прерывание микропроцессора.

Бит 0. **DCTS**(Delta Clear To Send) — Изменение состояния сигнала CTS(очищен для передачи).Бит устанавливается в DCTS=1 при изменении сигнала CTS на входе микросхемы и сбрасывается при прочтении регистра MSR микропроцессором. При установке бита в 1 генерируется прерывание микропроцессора.

Бит 1. **DDSR**(Delta Data Set Ready) — Изменение состояния сигнала DSR(установка данных готова).Бит устанавливается в DDSR=1 при изменении сигнала DSR на входе микросхемы и сбрасывается при прочтении регистра MSR микропроцессором. При установке бита в 1 генерируется прерывание микропроцессора.

Бит 2. **ТЕRI**(Trailing Edge Of Ring Indicator) — Детектор заднего фронта сигнала RI(индикатор звонка). Бит устанавливается в TERI=1, когда сигнал на выводе микросхемы RI изменяет свой уровень с низкого на высокий. Бит сбрасывается в TERI=0 при прочтении регистра MSR микропроцессором. При установке бита в 1 генерируется прерывание микропроцессора.

Бит 3. **DDCD**(Delta Data Carrier Detect) — Изменение состояния сигнала DCD(обнаружен носитель информации).Бит устанавливается в DDCD=1 при изменении сигнала DCD на входе микросхемы и сбрасывается при прочтении регистра MSR микропроцессором. При установке бита в 1 генерируется прерывание микропроцессора.

Бит 4. **CTS**(Clear To Send) — Состояние линии CTS. Если CTS=1, то на вход CTS СОМ-порта подано напряжение +12в(сигнал CTS активен).Если CTS=0, то вход СОМ-порта подано напряжение -12В(сигнал CTS пассивен). В режиме диагностики этот бит эквивалентен биту RTS регистра MCR.

Бит 5. **DSR**(Data Set Ready) — Состояние линии DSR. Если DSR=1, то на вход DSR СОМ-порта подано напряжение +12в(сигнал DSR активен).Если DSR=0, то вход СОМ-порта подано напряжение -12В(сигнал DSR пассивен). В режиме диагностики этот бит эквивалентен биту DTR регистра MCR.

Бит 6. **RI**(Ring Indicator) — Состояние линии RI. Если RI=1, то на вход DSR СОМ-порта подано напряжение +12в(сигнал RI активен).Если RI=0, то вход СОМ-порта подано напряжение -12В(сигнал RI пассивен). В режиме диагностики этот бит эквивалентен биту OUT1 регистра MCR.

Бит 7. **DCD**(Data Carrier Detect) — Состояние линии DCD. Если DCD=1, то на вход DCD СОМ-порта подано напряжение +12в(сигнал DCD активен).Если DCD=0, то вход СОМ-порта подано напряжение -12В(сигнал DCD пассивен). В режиме диагностики этот бит эквивалентен биту OUT2 регистра MCR.

**SCR** *– регистр временного хранения (чтение/запись)*

*(Scratch Pad Register)*

Регистр SCR (Адрес=07h, RD/WR)

Регистр временного хранения, на работу UART не влияет, предназначен для временного хранения данных (в UART i8250 отсутствует)

*1. Программирование com порта прямым кодом процессора*

Команды OUT и IN микропроцессора, позволяют читать/записать байт по указанному адресу порта ввода/вывода. Необходимо просто записать нужные данные в эти регистры, чтобы заставить СОМ порт выполнить нужные действия. При программировании регистров UART нужно учитывать, что в памяти BIOS находятся адреса портов ввода/вывода для СОМ1...СОМ4. По умолчанию они равны COM1=3F8h, COM2=2F8h, COM3=3E8h, COM4=2E8h , но бывают "чудики" которые могут их поменять в настройках BIOS. Поэтому перед началом программирования портов в MS-DOS желательно проверить адреса СОМ портов. В BIOS адрес СОМ порта занимает 2 байта, и находятся в ячейках памяти по адресам СОМ1: 40...41h, СОМ2: 42...43h, СОМ3: 44...45h, СОМ4: 46...47h.

Адрес регистра com порта вычисляется так:

***адрес регистра com порта = адрес самого com порта + относительный адрес его регистра***

Относительные адреса всех регистров указаны в таблице всех регистров com порта (см ближе к началу)

Команды ввода/вывода микропроцессора:

* **IN AL,port8**- ввод байта в регистр AL из указанного порта;
* **IN AL,DX**- ввод байта в регистр AL из порта по адресу указанному в DX;
* **OUT port8,AL**- вывод байта из регистр AL указанного порта;
* **OUT DX,AL**- вывод байта из регистр AL порта по адресу указанному в DX;

Перед записью байта данных в регистр передатчика необходимо убедиться в том, что регистр хранения передатчика свободен, то есть убедиться в том, что передача предыдущего символа завершена. Признаком того, что регистр передатчика свободен, является установленный бит 5 (THRE=1) регистра состояния линии LSR.

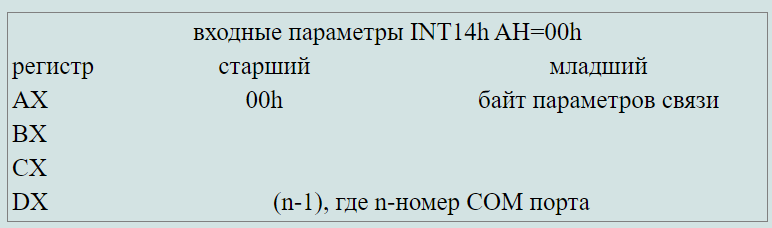
Аналогично тому как это делается при передаче данных, перед вводом символа из порта приемника необходимо убедиться в том, что бит 0 регистра LSR установлен (т.е. DR=1). Это означает, что символ принят из линии и находится в буферном регистре приемника.

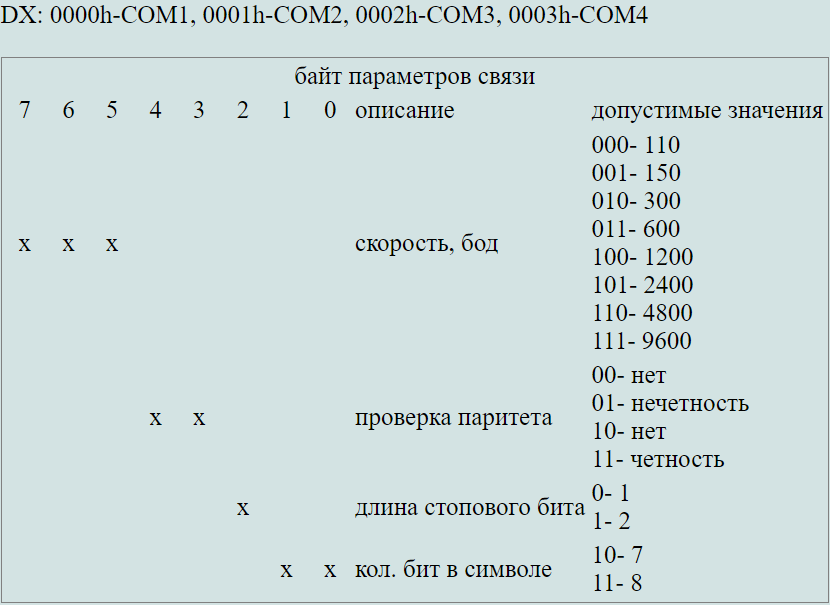
*2. Программирование com порта с помощью прерываний BIOS*

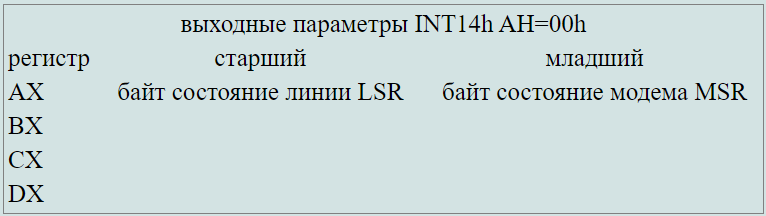
Используем функции прерывания BIOS ***14h***

*INT 14h AH=00h* – инициализация СОМ порта.

Под инициализацией порта (также применяют термин "открытие") понимают установку всех его параметров: номер порта, длину символа, число стоп-бит, установку четности и скорость обмена.



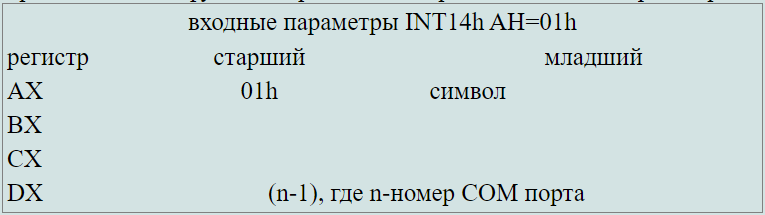


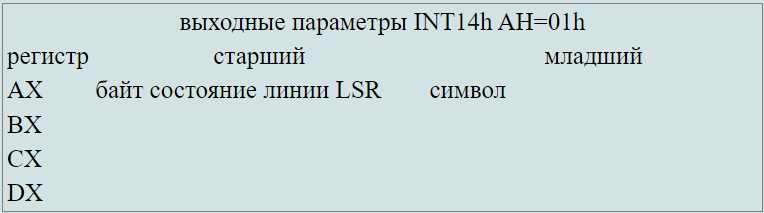


В качестве выходных параметром в регистр аккумулятора копируются регистры UART LSR(см.рис.23) и MSR(см.рис.24). Из выше сказанного видно, что работа с СОМ портом через функцию BIOS INT14h ограничена по скорости и по количеству бит в символе.

*INT 14h AH=01h*-запись символа в СОМ порт.

При вызове этой функции происходит передача символа из регистра AL в порт с номером заданным в регистре DX





*3. Программирование COM порта с помощью API Windows*

1. Открытие COM порта (инициализация)

Открыть порт можно при помощи API функции *CreateFile*

*HANDLE CreateFile(*

*LPCTSTR lpFileName,*

*DWORD dwDesiredAccess,*

*DWORD dwShareMode,*

*LPSECURITY\_ATTRIBUTES lpSecurityAttributes,*

*DWORD dwCreationDistribution,*

*DWORD dwFlagsAndAttributes,*

*HANDLE hTemplateFile*

*);*

Эта функция создает новый объект и присваивает ему дескриптор, по которому с этим объектом можно будет работать.

Параметры функции CreateFile:

* **lpFileName** - имя СОМ-порта. Может принимать значения: "СОМ1","СОМ2","СОМ3","СОМ4","СОМ5","СОМ6","СОМ7","СОМ8","СОМ9", если более одной цифры, то в формате "\\.\СОМ47"
* **dwDesiredAccess** - режим доступа к файлу.Это четырехбайтовое число, которое задает различные режимы доступа к файлу. Нас интересует только режим чтение и запись, этот режим задаётся числом: С0000000hex в СИ можно вместо числа записать константу с именем "GENERIC\_READ|GENERIC\_WRITE".
* **dwShareMode** - режим совместного доступа. СОМ-порты ПК не поддерживают совместный доступ, только одна программа может открыть порт. Поэтому этот параметр должен быть равен 0(режим запрещен).
* **lpSecurityAttributes** - атрибуты защиты файла. Для СОМ-портов не используется поэтому всегда равны 0 ("NULL").
* **dwCreationDistribution** - управление режимом автосоздания файла. Это четырехбайтовое число, которое для СОМ портов всегда должно быть 00000003hex ("OPEN\_EXISTING")
* **dwFlagsAndAttributes** - задает атрибуты создаваемого файла.Это четырехбайтовое число, которое для СОМ портов всегда должно быть 0 ("NULL")
* **hTemplateFile** - описатель файла "шаблона" по которому создавался файл.Для СОМ-портов не используется поэтому всегда равен 0 ("NULL").

1. Чтение и запись

Чтение: ReadFile()

*BOOL ReadFile(*

*HANDLE hFile, // описатель сОМ порта*

*LPVOID lpBuffer, // Указатель на буфер, который принимает прочитанные данные из порта*

*DWORD nNumberOfBytesToRead,// Число байтов, которые читаются из порта*

*LPDWORD lpNumberOfBytesRead, // Указатель на переменную, которая получает число прочитанных байтов*

*LPOVERLAPPED lpOverlapped // Указатель на структуру OVERLAPPED.*

Запись: WriteFile()

*BOOL WriteFile(*

*HANDLE hFile,// описатель сом порта*

*LPCVOID lpBuffer,// Указатель на буфер, содержащий данные, которые будут записаны в файл.*

*DWORD nNumberOfBytesToWrite,// Число байтов, которые будут записаны в файл.*

*LPDWORD lpNumberOfBytesWritten,// Указатель на переменную, которая получает число записанных байтов*

*LPOVERLAPPED lpOverlapped // Указатель на структуру OVERLAPPED*

*);*

1. Закрытие COM порта

*BOOL CloseHandle(*

*HANDLE hObject // описатель порта*

*);*

***Ответы на вопросы***

***1. Назначение сигналов СОМ порта по стандарту RS-232C***

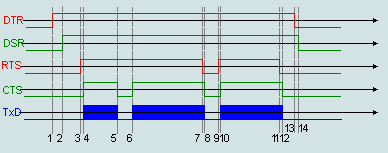
* GND- Ground, (общий) второй провод для всех сигналов.(Сигналы передаются всегда по двум проводам!)
* TxD- Transmited Data, асинхронный канал для передачи данных.
* RxD- Received Data, асинхронный канал для приема данных.
* RTS- Request To Send (запрос на передачу), Выход который говорит о том, что у компьютера есть данные для передачи по каналу TxD для конечного устройства.
* DTR- Data Terminal Ready(готовность терминала данных), Выход который говорит о том, что компьютер(терминал) готов к обмену данными с конечным устройством
* CTS- Clear To Send (очищен для передачи) Вход, который говорит о том, что конечное устройство готово принимать данные от терминала по каналу TxD. Обычно этот сигнал выставляет конечное устройство после того, как оно получит от компьютера сигнал RTS=True(запрос на передачу) и будет готово принять данные от компьютера. Если конечное устройство не выставит сигнал CTS=True, то передача по каналу TxD не начнется. Данный сигнал используется для аппаратного управления потоками данных
* DSR- Data Set Ready(установка данных готова), Вход который говорит о том, что конечное устройство выполнило все установки и готово начать передавать и принимать данные от компьютера. Если конечное устройство модем, то установка DSR=True воспринимается компьютером(терминалом) так, что модем уже установил связь с другим модемом и готов начать процедуру обмена между двумя компьютерами оснащенных модемами
* DCD- Data Carrier Detected(обнаружен носитель информации), Вход который информирует компьютер(терминал) об обнаружении другого терминала, то есть конечное устройство , например модем, обнаружил другой модем, который хочет инициализировать обмен данных между терминалами. Модем выставляет сигнал DCD=True, который обнаруживается на входе компьютера(терминала). Если терминал готов к обмену данными, то он на сигнал DCD=True должен выставить сигнал готовности терминала к обмену данными DTR=True, после чего начинается обмен данными между двумя терминалами.
* RI- Ring Indicator(индикатор звонка), Вход который говорит компьютеру(терминалу) что на конечное устройство поступает сигнал вызова. Например, на модем поступил сигнал вызова с телефонной станции, совсем не обязательно, что этот вызов закончится обменом данных.
* PG - Protective Ground. Защитное заземление.

***2. Назовите регистры последовательных портов и их назначение***

Описано выше.

***3. Алгоритмы режимов синхронизации приема - передачи данных***

***Аппаратный режим синхронизации обмена данными RTS/CTS*** (hardware flow control) Handshaking =2, использует сервисные сигналы RS-232C для управления потоком данных.

Рис.10 Организация обмена при аппаратной синхронизации.

Порядок выставления сервисных сигналов при обмене через модем:

1. DTR=True компьютер указывает на желание использовать модем
2. В ответ модем сигнализирует о установлении соединения с другим модемом и своей готовности выставив DSR=True.
3. Сигналом RTS=True компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от модема.
4. Сигналом CTS=True модем уведомляет о своей готовности к приему данных от компьютера и передаче их в линию. По этому сигналу начинается обмен данными между терминалами через модемы.
5. Установкой CTS=False модем сигнализирует о невозможности дальнейшего приема, компьютер должен приостановить передачу данных.
6. Установкой CTS=True, модем сообщает, что он может продолжать обмен данными.
7. Установкой RTS=False компьютер сообщает модему о временной приостановке обмена.
8. Модем получив сигнал остановки обмена RTS=False, сообщает о приостановки обмена сигналом CTS=False
9. Компьютер вновь готов принимать данные и он выставляет сигнал RTS=True
10. Модем получив сигнал от компьютера о готовности к обмену выставляет свою готовность CTS=True. После чего возобновляется обмен данных.
11. Компьютер указывает на завершение обмена выставив RTS=False
12. Модем подтверждает завершение обмена сигналом CTS=False
13. Компьютер снимает сигнал DTR, что является сообщением для модема разорвать соединение и повесить трубку.
14. Модем подтверждает разрыв соединения установкой сигнала DSR=False

Как видно из рис.10 модем использует сигнал CTS, который позволяет остановить передачу данных, если приемник не готов к их приему. Передатчик «выпускает» очередной байт только при включенной линии CTS. Байт, который уже начал передаваться, задержать сигналом CTS невозможно (это гарантирует целостность посылки). Аппаратный протокол обеспечивает самую быструю реакцию передатчика на состояние приемника.

***Организация обмена данных при программном режиме синхронизации***

Программный протокол управления потоком XON/XOFF( Handshaking =1). Работает протокол следующим образом: если устройство, принимающее данные, обнаруживает причины, по которым оно не может их дальше принимать, оно по обратному последовательному каналу посылает байт-символ XOFF (13hex). Противоположное устройство, приняв этот символ, приостанавливает передачу. Когда принимающее устройство снова становится готовым к приему данных, оно посылает символ XON (11hex), приняв который противоположное устройство возобновляет передачу. Время реакции передатчика на изменение состояния приемника по сравнению с аппаратным протоколом увеличивается, по крайней мере, на время передачи символа (XON или XOFF) плюс время реакции программы передатчика на прием символа. Преимущество программного протокола заключается в отсутствии необходимости передачи управляющих сигналов интерфейса — минимальный кабель для двустороннего обмена может иметь только 3 провода. Недостатком данного метода является большее время реагирования и исключения из передаваемого потока двух символов (13hex, 11hex).

Существует смешанный метод синхронизации обмена данными RTS/XOn/Xoff (Handshaking =3), который представляет собой объединение двух предыдущих методов.

***4. Для чего используется режим диагностики и как включается?***

Режим диагностики UART позволяет проверить работоспособность СОМ-портов без подключения к ним периферийных устройств. Режим диагностики включается битом LOOP=1 регистра MCR.

При этом внутри UART организуется аппаратная "заглушка":

* выход передатчика переводится в состояние логической единицы;
* вход приемника отключается;
* выход сдвигающего регистра передатчика соединяется со входом приемника;
* входы DSR, CTS, RI, DCD отключаются от входных линий COM-порта и данные в них поступают из регистра MCR (биты RTS, DTR, OUT1, OUT2);
* выходы управления модемом переводятся в пассивное состояние (логический ноль).

Аппаратная "заглушка" позволяет передавать и сразу принимать данные СОМ-порта, без каких-либо подключений к нему. В результате этого возможно проверить работу сдвигающего регистра, отработку системы прерываний и т.д.

***5. Назовите преимущества последовательных портов.***

- Снижение числа соединений

- Нет проблемы электромагнитных помех (на высоких скоростях передачи данных каждый провод/канал становится антенной, излучающей электромагнитные волны; при параллельном интерфейсе т.к. эти каналы находятся рядом, то и влияние на соседние каналы они оказывают более значительное)