UART Details

Background

A Universal Asynchronous Receiver/Transmitter (UART) is used to implement serial communication. It is a standard piece of hardware, although manufacturers make slightly different versions that may have some functionality beyond the standard UART functionality described below.

A *data sheet* for the UART chip we are using can be found on the supporting materials page. The data sheet provides a more detailed and lengthy description than that provided below.

Finally, a word about terminology. When talking about serial transmission, a logical "1" on the line is called a *mark state* (or condition) while a logical "0" on the line is called a *space state* (or condition).

Interface

The CPU communicates with the UART by reading or writing one of eight bytes called *ports*. A computer system normally has more than one UART, so the port addresses depend on the particular UART being accessed. Each UART is associated with a different *base address*, and a particular port is specified by adding a specific *index* to that base address. The index for a particular port is independent of the UART, so we can characterize the ports by indices `0` through `7`.

A program is more understandable if it uses symbols rather than ``magic numbers" for things like port addresses and bits within a register. File `uart.ah` defines such symbols, and we will use them throughout this document. You should include file `uart.ah` in any assembly language program accessing the UART, and use the names it defines to refer to the ports and bits within the ports.

Some of the UART ports can only be read, others can only be written, and both accesses are possible on some. Even when both accesses are allowed, however, they may be unrelated. For example, the UART has a data-in buffer register and a data-out buffer register. Both of these registers, each of which holds one byte, are accessed via port `0`. If the UART has assembled a byte from the bits it has received, then the CPU can get that byte by reading port `0`. Similarly, if the UART has completely disposed of a byte then the CPU can output another byte by writing it to port `0`. Notice that reading and writing are totally unrelated -- if the CPU writes a byte to port `0` and then immediately reads from that port, it will *not* get the byte it wrote.

File `uart.ah` gives different names to the same port when that port can be used for different purposes. Thus both `RBR` (*R*ead *B*uffer *R*egister) and `THR` (*T*ransmit *H*olding *R*egister) are names for port `0`. You should essentially ignore the port numbers, and think of the UART only in terms of the symbols describing its functionality.

In addition to the data buffers, the UART's ports allow you to access three control registers and three status registers. Each control register is paired with one status register to deal with a different aspect of the UART:

- its data transmission (both directions),
- its handshaking with a modem, and
- its interrupt behavior.

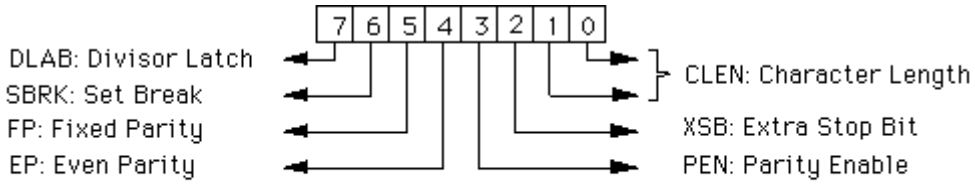
Port `7`, named `SCR` in file `uart.ah`, accesses a ``scratch register" that acts exactly like a memory location. It has no control, status, or data function related to the UART.

Data Transmission Control

The controllable characteristics of the data transmission are:

- Baud rate
- Number of information bits per character
- Type of parity checking
- Number of stop bits
- Breaking the transmission

These characteristics are controlled by the line control register (`LCR`, accessed via port `3`).



LCR - Line Control Register (Address 3)

Baud rate

The baud rate is established by storing the value 115200/(baud rate) into a 16-bit register inside the UART. When the `DLAB` bit of `LCR` is `1`, the least-significant byte of this register can be accessed via port `0` (using symbol `DLL`), and the most-significant byte can be accessed via port `1` (using symbol `DLM`). When `DLAB` bit of `LCR` is `0`, the data-in and data-out buffer registers are accessed via port `0` (using symbols `RBR` and `THR` as discussed above) and the interrupt enable register `IER` is accessed via port `1`.

Number of information bits per character

`CLEN` encodes the number of information bits in each character:

- `CLEN=0` for 5 information bits
- `CLEN=1` for 6 information bits
- `CLEN=2` for 7 information bits
- `CLEN=3` for 8 information bits

Type of parity checking

If `PEN=0` then the UART neither generates a parity bit for outgoing characters nor checks parity on incoming characters. If `PEN=1` and `FP=0` then even parity is generated and checked if `EP=1`, and odd parity is generated if `EP=0`. Finally, if `PEN=1` and `FP=1` then the generated parity bit is equal to `EP`, and the parity bit of an incoming character must be equal to `EP`.

Number of stop bits

`XSB` specifies the number of stop bits transmitted with each serial character. If `XSB=0` then one stop bit is generated in the transmitted data. Otherwise 1.5 stop bits are generated for characters with 5 information bits and 2 stop bits are generated for all other characters. The receiver checks the first stop bit only, regardless of the number of stop bits selected.

Breaking the transmission

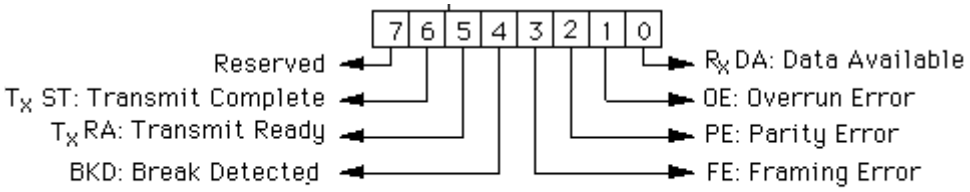
When `SBRK=1`, the serial output line is held in the spacing state; no characters are transmitted. This condition is detected by the UART at the other end of the line, and causes `BKD=1` in the line status register port `5` of that UART.

Data Transmission Status

The status of the data transmission involves

- Availability of an input character
- Completion of character output
- Errors
- Break detection

This status is reported in the line status register (`LSR`, accessed via port `5`). All bits in this register are reset to 0 when the register is read, except as noted below.



LSR - Line Status Register (Address 5)

Availability of an input character

`RxDA=1` when a character is available in the data-in buffer register.

Completion of character output

`TxRA=1` when the data-out buffer register does *not* contain a character, and `TxST=1` when transmission of all characters is complete. These bits remain `1` if no transmission is in progress when the line status register is read.

Errors

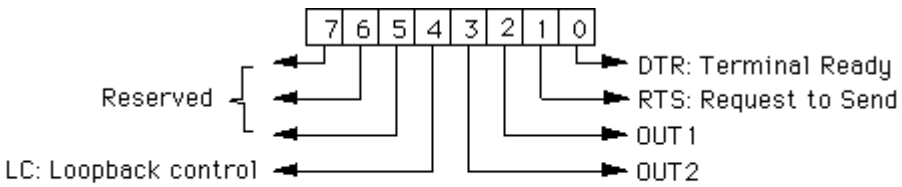
`PE=1` indicates a parity error, `OE=1` an overrun error, and `FE=1` a framing error.

Break detection

`BKD=1` if the input line has been held in a spacing condition for two or more character times.

Modem Handshaking Control

An protocol called RS-232C describes how a computer and a modem should interact to ensure that they agree on who is ready to do what. The modem control register (`MCR`, accessed via port `4`), is used to control the outgoing signals used for this protocol. It also allows the machine to perform self-tests without using the modem at all.



MCR - Modem Control Register (Address 4)

Handshaking protocol

The computer should set DTR=1 when it is ready for communication. (Setting DTR=0, for example, will cause a modem to hang up the telephone line at the end of a transmission.)

RTS=1 indicates that the computer desires to transmit information. Setting RTS=0 would indicate that the modem should turn the line around when using half-duplex mode. For full-duplex mode, RTS should be permanently 1.

Loopback control

It is useful to be able to test communication software without having to have another device to communicate with. If LC=1 then transmitted characters are sent directly to the receiver -- they ``loop back'' within the UART itself. When LC=0, characters are transmitted normally over the serial line.

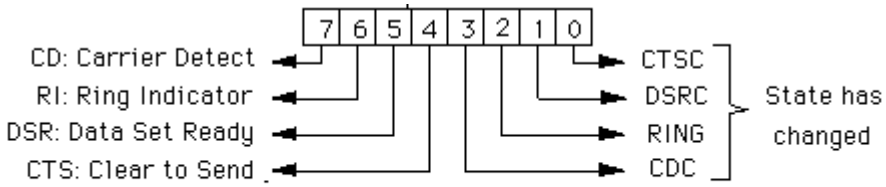
Miscellaneous control bits

OUT1 is used only by specialized hardware (like the Hayes SmartModem internal board). The meaning of this bit is determined by the hardware using it. For example, if the computer has a Hayes SmartModem board then OUT1=1 resets the modem.

OUT2 controls interrupt servicing for the UART: OUT2=0 blocks the interrupt generated by the UART. This allows the user to prevent UART interrupts from reaching the computer's priority interrupt controller, while still generating them within the UART.

Modem Handshaking Status

The modem status register (MSR, accessed via port 6), is used to sense incoming signals used for the handshaking RS-232C protocol. All bits in this register are reset to 0 when the register is read, except as noted below.



MSR - Modem Status Register (Address 6)

Current State

DSR=1 indicates that the modem is ready for communication. This bit remains 1 after the register is read if the modem remains ready.

CTS=1 indicates that the computer is allowed to transmit information. In half-duplex mode (Section 9-1), the modem responds to a signal RTS=1 from the computer by turning the line around and then setting CTS=1. This bit remains 1 after the register is read if the computer is still allowed to transmit information.

CD=1 if the modem believes that there actually is an incoming signal. For example, if a computer is communicating with a remote terminal over a telephone line, and the modem detects that the other party has hung up, it will set CD=0. This bit remains 1 after the register is read if the modem still believes that there actually is an incoming signal.

The modem sets RI=1 when it detects a ringing signal on the telephone line. Thus RI=1 is a request for service from a remote site.

State Changes

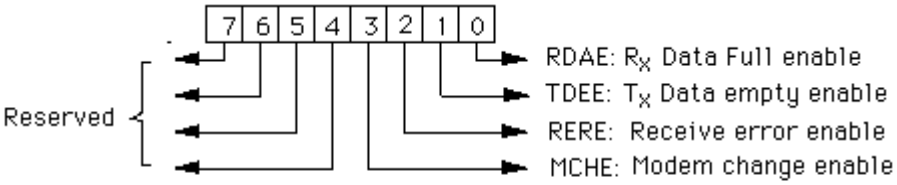
The remaining bits in the register are set to `1` when a state change is detected. Each of these bits is associated with one of the state bits mentioned above, and becomes `1` when that state bit changes. Thus these bits allow the computer to determine which state bits have changed since the last time the register was read, without having to keep the old values and compare them.

Interrupt Behavior

The UART is capable of generating an interrupt when any one of a number of situations arises. These situations are grouped into four classes:

1. Receive machine error or break condition
2. Receive data register full
3. Transmit data register empty
4. Change in the state of the modem input pins

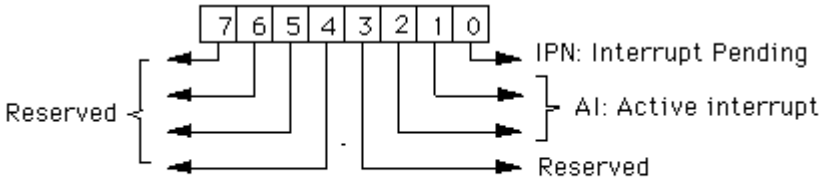
Interrupts in each of these classes must be explicitly enabled by writing a `1` to the appropriate bit in the interrupt enable register (`IER`, accessed via port `1`).



IER - Interrupt Enable Register (Address 1)

In addition to enabling the particular kind of interrupt, interrupt servicing for the UART must be requested by setting bit `OUT2` in the modem control register (`MCR`, accessed via port `4`).

If an interrupt occurs, the class of interrupt can be determined by reading the interrupt identification register (`IIR`, accessed via port `2`).



IIR - Interrupt Identification Register (Address 2)

`IPN=0` if an interrupt is pending. Thus the `AI` field of the interrupt identification register is relevant only if `IPN=0`.

The interrupt classes are prioritized, with `AI=0` being the lowest and `AI=3` the highest priority:

- `AI=3` for a receive machine error or break condition
- `AI=2` when received data is available
- `AI=1` when the transmit register is empty
- `AI=0` when there is a change in the state of the modem input pins

If the interrupt identification register indicates that there was a receive machine error or break condition then the program must examine the line status register to determine the precise condition. Similarly, if the interrupt identification register indicates that there was a change in the state of the modem input pins the program must determine the change by reading the modem status register.

Only the highest priority interrupt appears in the interrupt identification register at any time. After dealing with a UART interrupt, therefore, the program must read the interrupt identification register again in order to

determine whether a lower-priority interrupt also occurred. This cycle stops when `IPN=1`.

Instructor Revision 1.13 (2004/02/11 20:18:36)