

Microprocessor I/O

(Chapter 5)

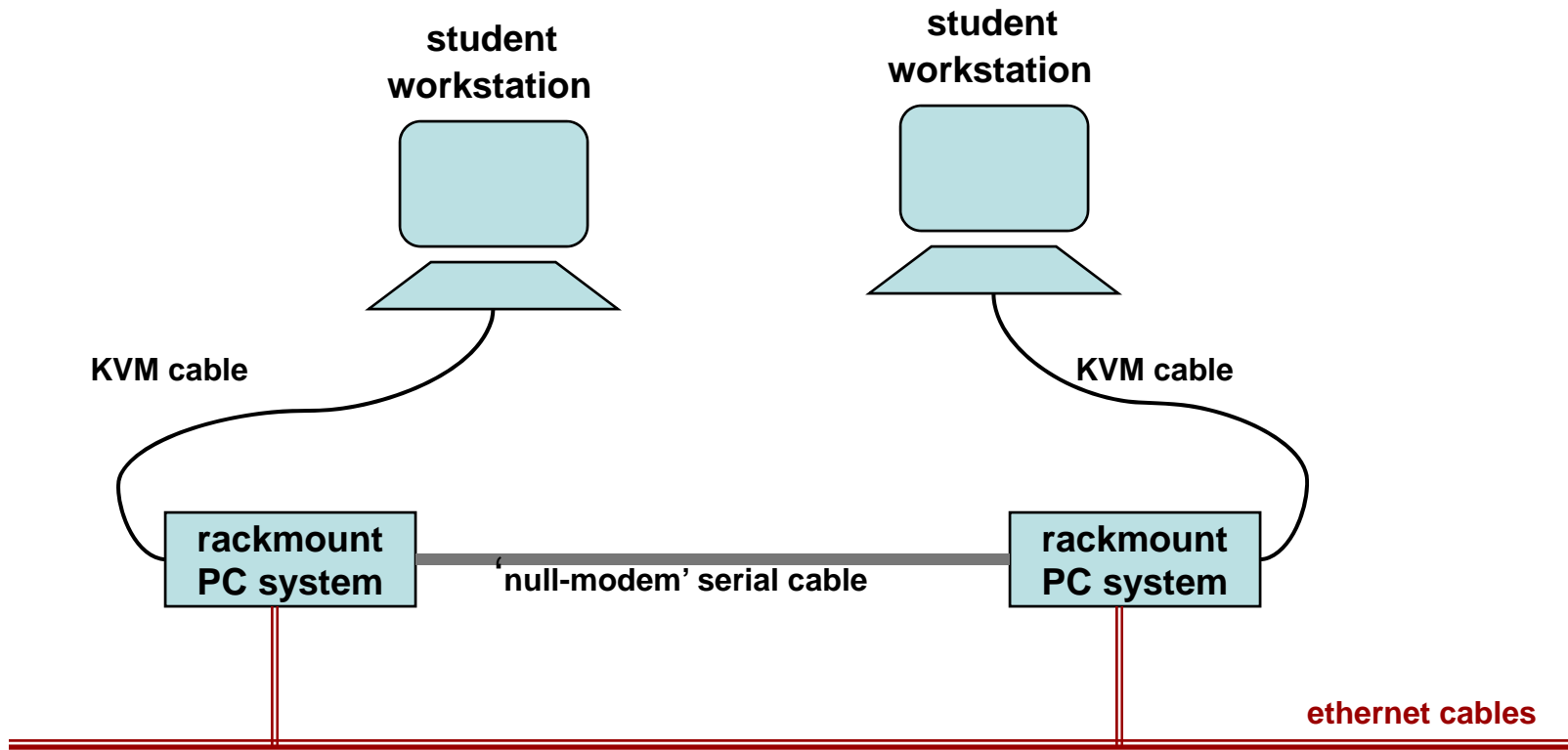
Microprocessor I/O

- Data Communication,
- Parallel I/O
- Serial communication
- Serial interface and UART modems
- I/O devices, D/A, A/D interface, special I/O devices.

Tx and Rx

- The UART has a transmission-engine, and also a reception-engine, which are able to operate simultaneously (i.e., “full-duplex”)
- Software controls the UART’s operations by accessing several registers, using the x86 processor’s ‘in’ and ‘out’ instructions
- Linux provides some convenient ‘macros’ that ‘hide’ the x86 machine-code details

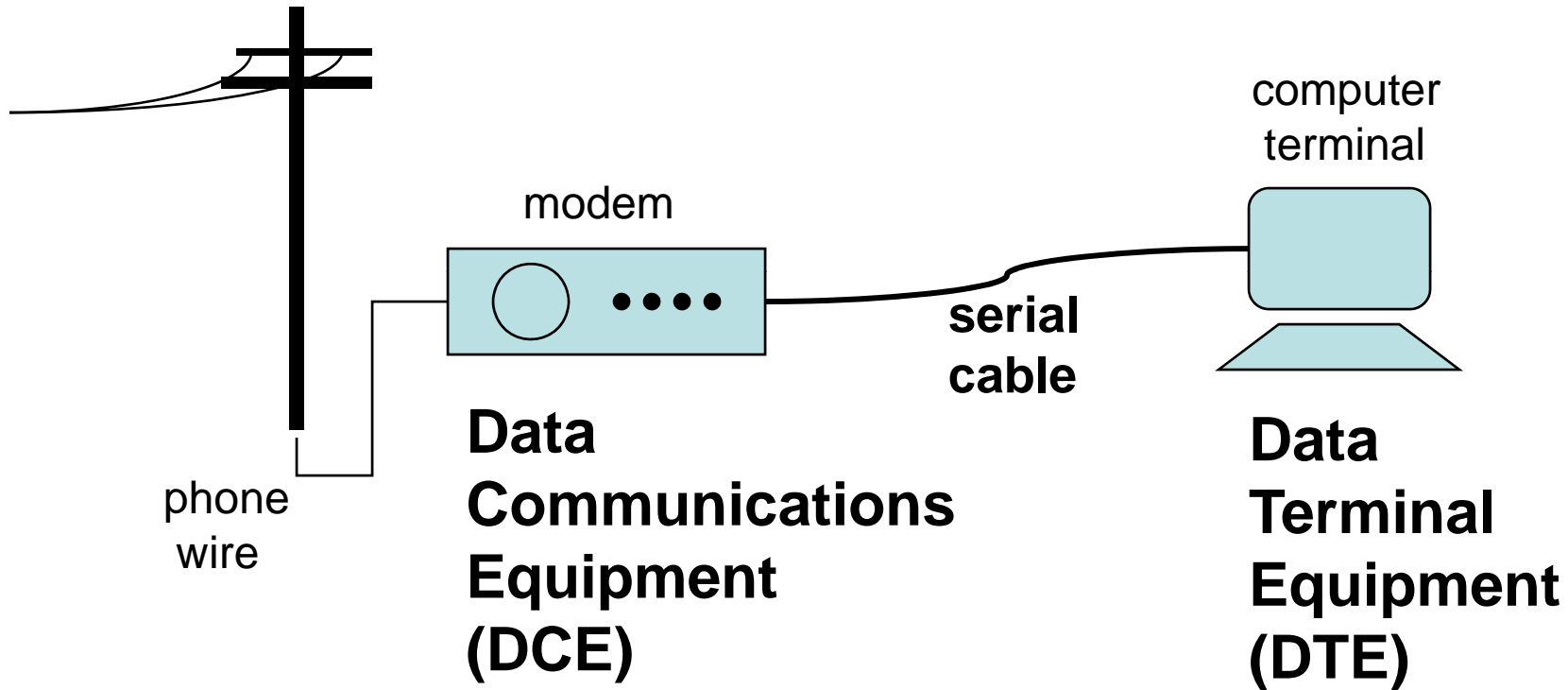
PC-to-PC communications



DCE and DTE

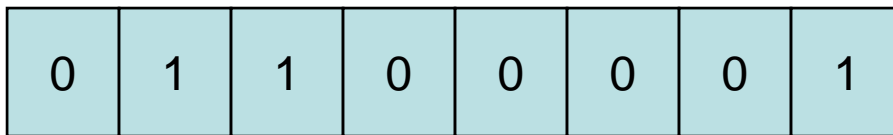
- Original purpose of the UART was for PCs to communicate via the telephone network
- Telephones were for voice communication (analog signals) whereas computers need to exchange discrete data (digital signals)
- Special 'communication equipment' was needed for doing the signal conversions (i.e. a modulator/demodulator, or **modem**)

PC with a modem



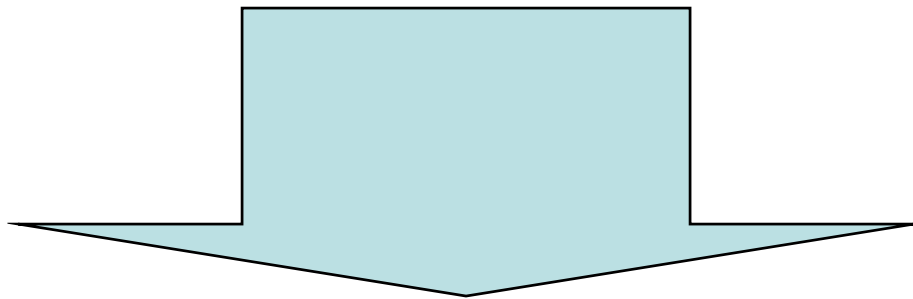
Serial data-transmission

The Transmitter Holding Register (8-bits)

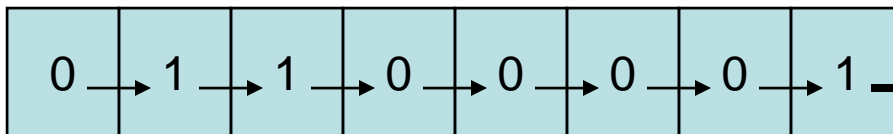


Software outputs a byte of data to the THR

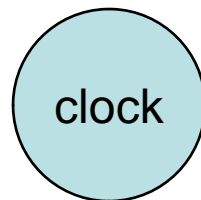
The bits are immediately copied into an internal 'shift'-register



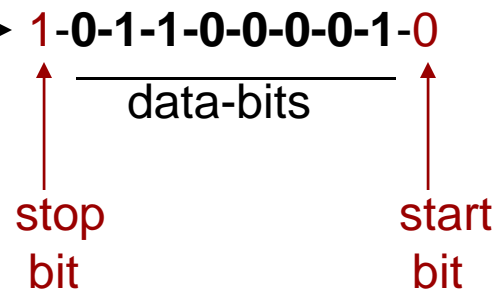
The bits are shifted out, one-at-a-time, in sync with a clock-pulse



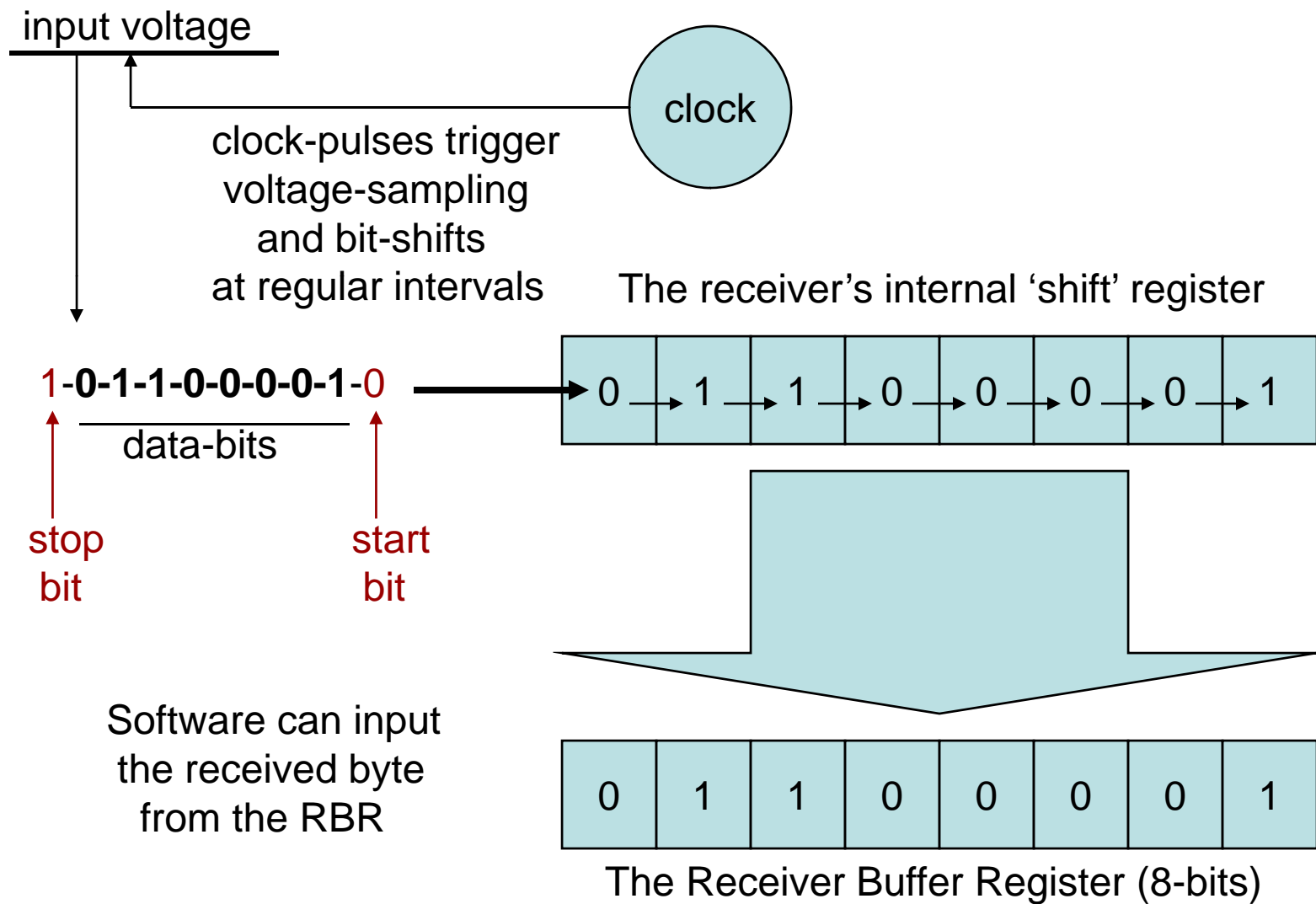
The transmitter's internal 'shift' register



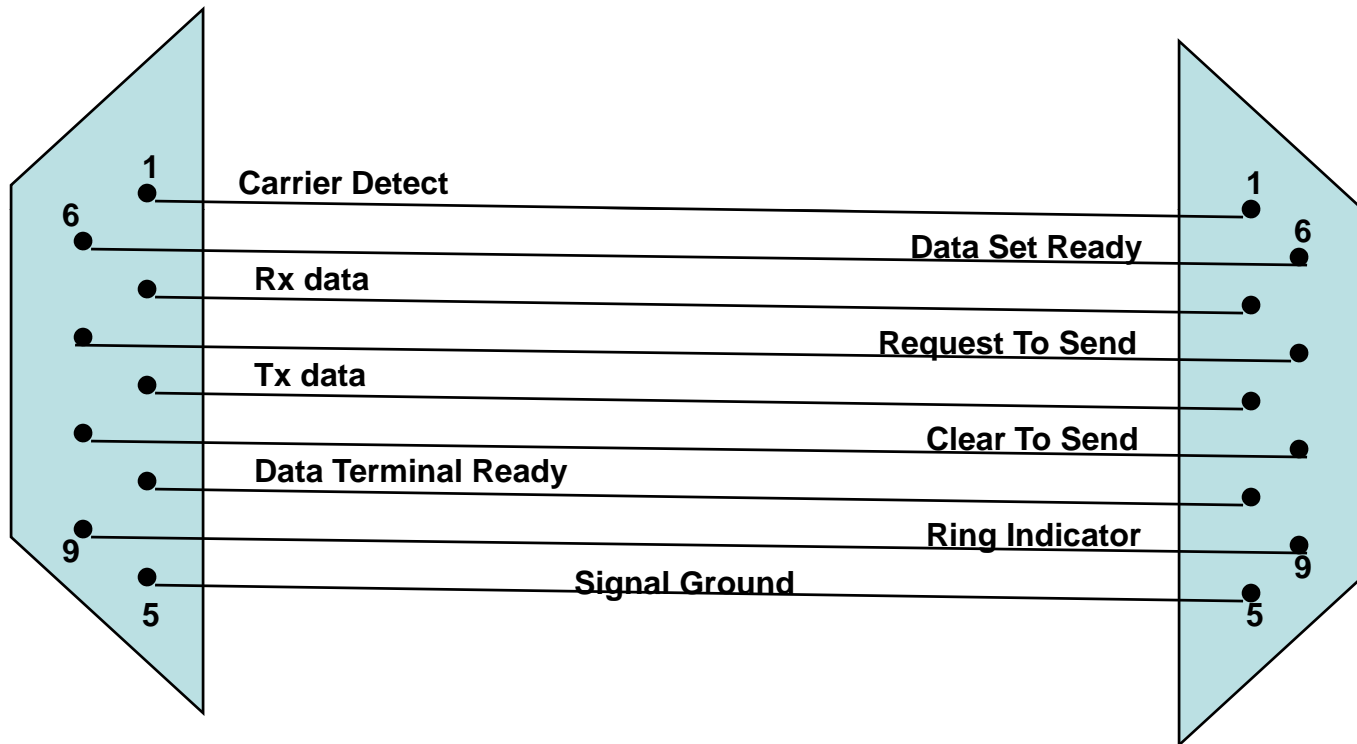
clock-pulses trigger bit-shifts



Serial data reception



Normal 9-wire serial cable



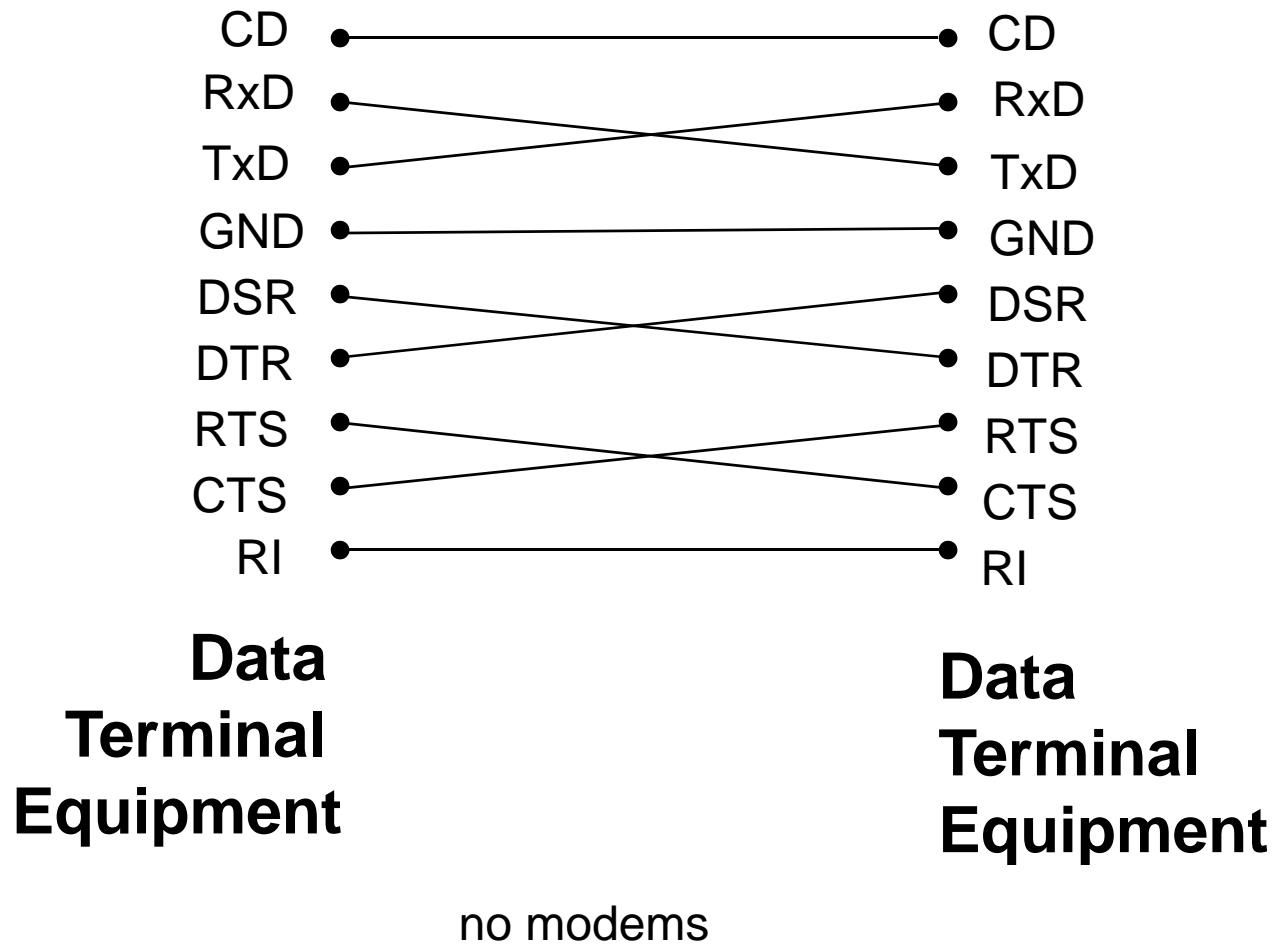
Signal functions

- **CD: Carrier Detect** The modem asserts this signal to indicate that it successfully made its connection to a remote device
- **RI: Ring Indicator** The modem asserts this signal to indicate that the phone is ringing at the other end of its connection
- **DSR: Data Set Ready** Modem to PC
- **DTR: Data Terminal Ready** PC to Modem

Signal functions (continued)

- **RTS: Request To Send** PC is ready for the modem to relay some received data
- **CLS: Clear To Send** Modem is ready for the PC to begin transmitting some data

9-wire null-modem cable



The 16550 UART registers

Base+0	Divisor Latch Register	16-bits (R/W)
Base+0	Transmit Data Register	8-bits (Write-only)
Base+0	Received Data Register	8-bits (Read-only)
Base+1	Interrupt Enable Register	8-bits (Read/Write)
Base+2	Interrupt Identification Register	8-bits (Read-only)
Base+2	FIFO Control Register	8-bits (Write-only)
Base+3	Line Control Register	8-bits (Read/Write)
Base+4	Modem Control Register	8-bits (Read/Write)
Base+5	Line Status Register	8-bits (Read-only)
Base+6	Modem Status Register	8-bits (Read-only)
Base+7	Scratch Pad Register	8-bits (Read/Write)

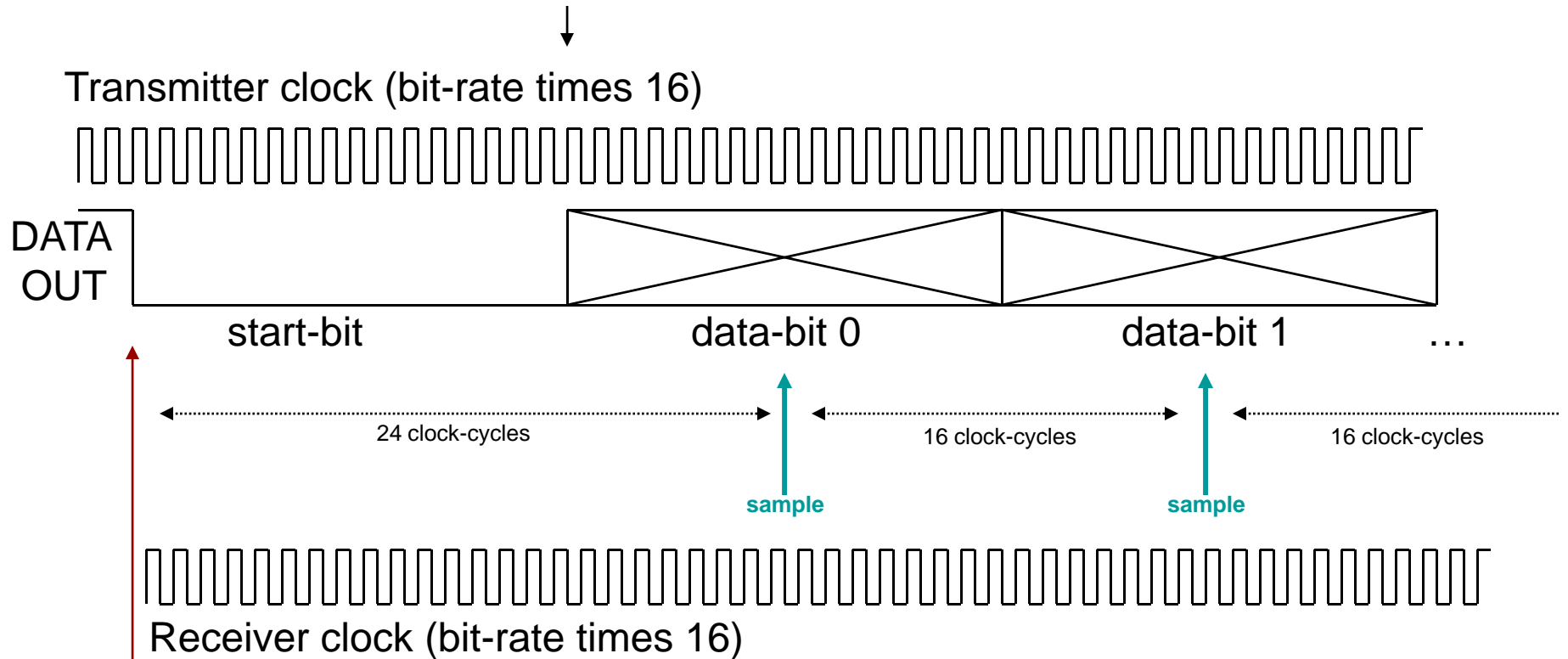
Rate of data-transfer

- The standard UART clock-frequency for PCs equals 1,843,200 cycles-per-second
- Each data-bit consumes 16 clock-cycles
- So the fastest serial bit-rate in PCs would be $1843200/16 = 115200$ bits-per-second
- With one 'start' bit and one 'stop' bit, ten bits are required for each 'byte' of data
- Rate is too fast for 'teletype' terminals

Divisor Latch

- The 'Divisor Latch' may be used to slow down the UART's rate of data-transfer
- Clock-frequency gets divided by the value programmed in the 'Divisor Latch' register
- Older terminals often were operated at a 'baud rate' of 300 bits-per-second (which translates into 30 characters-per-second)
- So Divisor-Latch was set to 0x0180

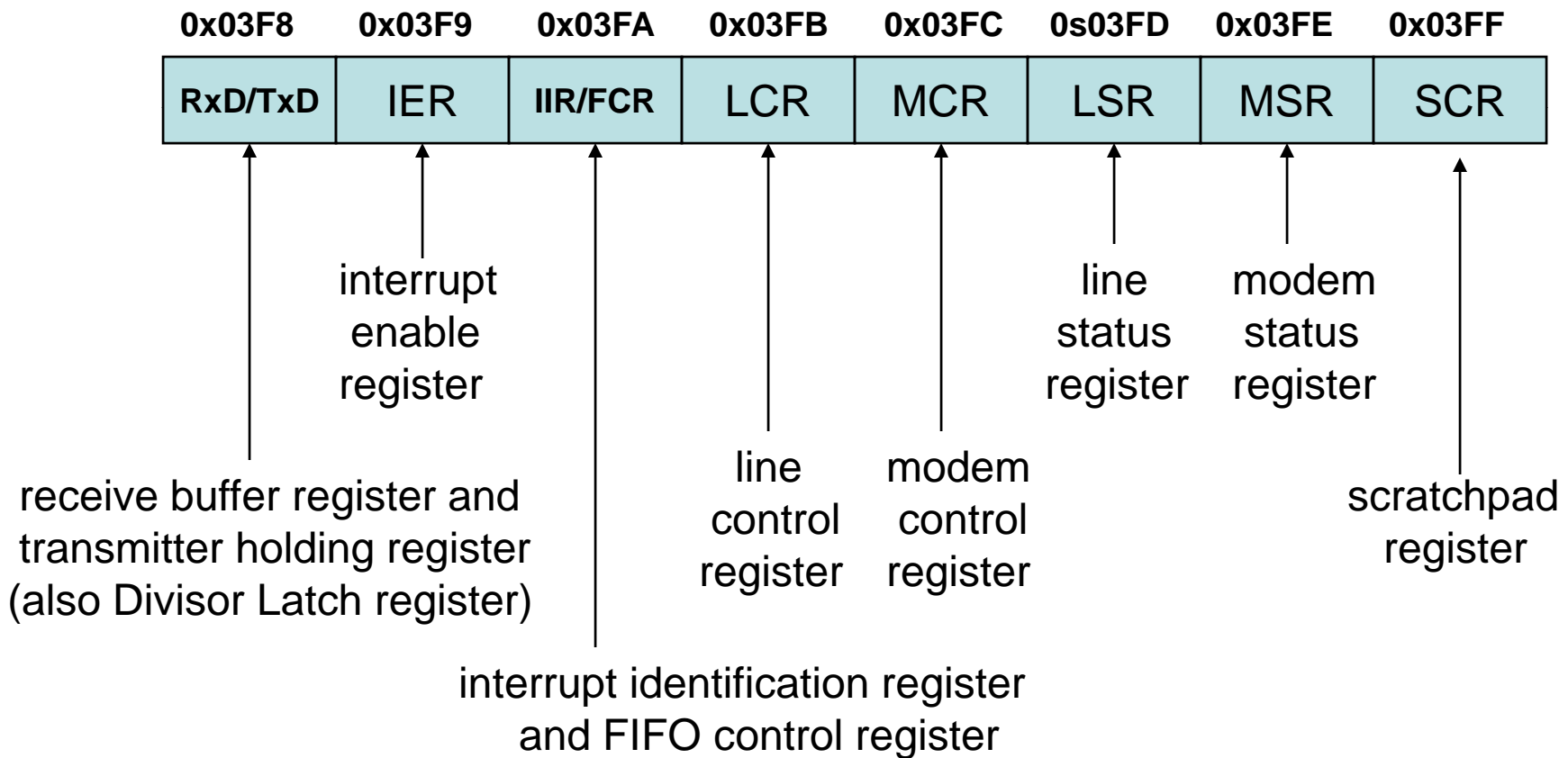
How timing works



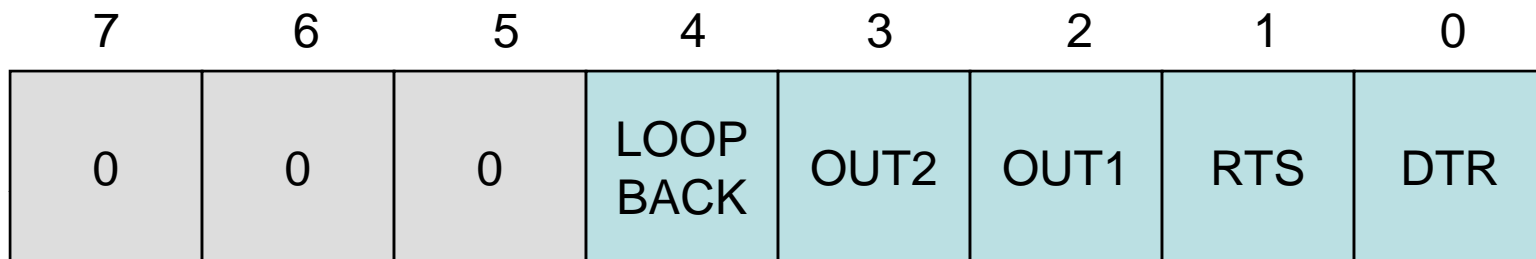
receiver detects this high-to-low transition,
so it waits 24 clock-cycles,
then samples the data-line's voltage
every 16 clock-cycles afterward

Programming interface

The PC uses eight consecutive I/O-ports to access the UART's registers



Modem Control Register



Legend:

DTR = Data Terminal Ready (1=yes, 0=no)

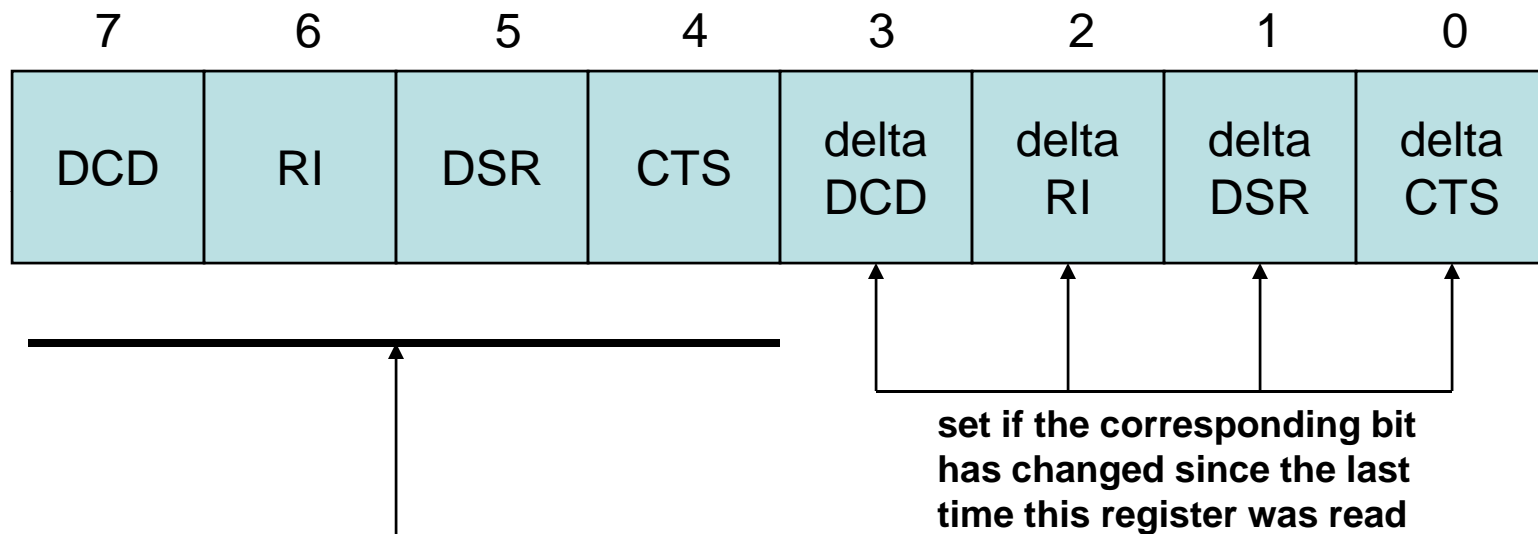
RTS = Request To Send (1=yes, 0=no)

OUT1 = not used (except in loopback mode)

OUT2 = enables the UART to issue interrupts

LOOPBACK-mode (1=enabled, 0=disabled)

Modem Status Register



Legend:

CTS = Clear To Send (1=yes, 0=no)

DSR = Data Set Ready (1=yes, 0=no)

RI = Ring Indicator (1=yes, 0=no)

DCD = Data Carrier Detected (1=yes, 0=no)

[---- loopback-mode ----]

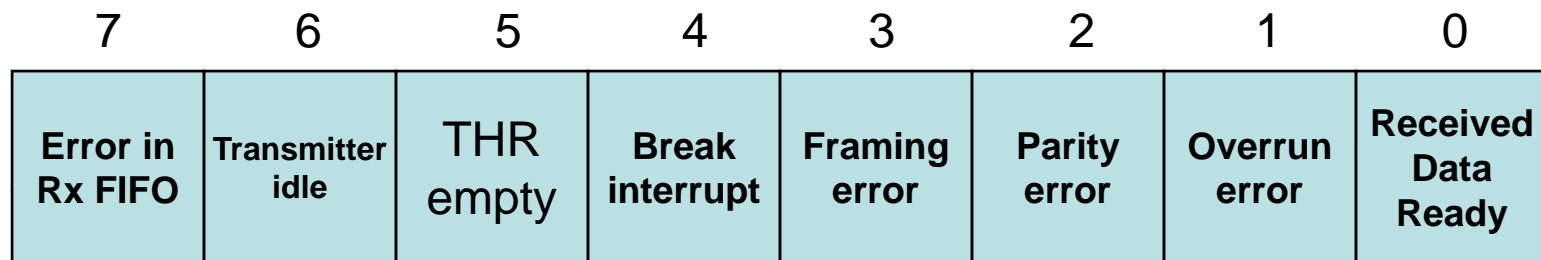
[bit 0 in Modem Control]

[bit 1 in Modem Control]

[bit 2 in Modem Control]

[bit 3 in Modem Control]

Line Status Register



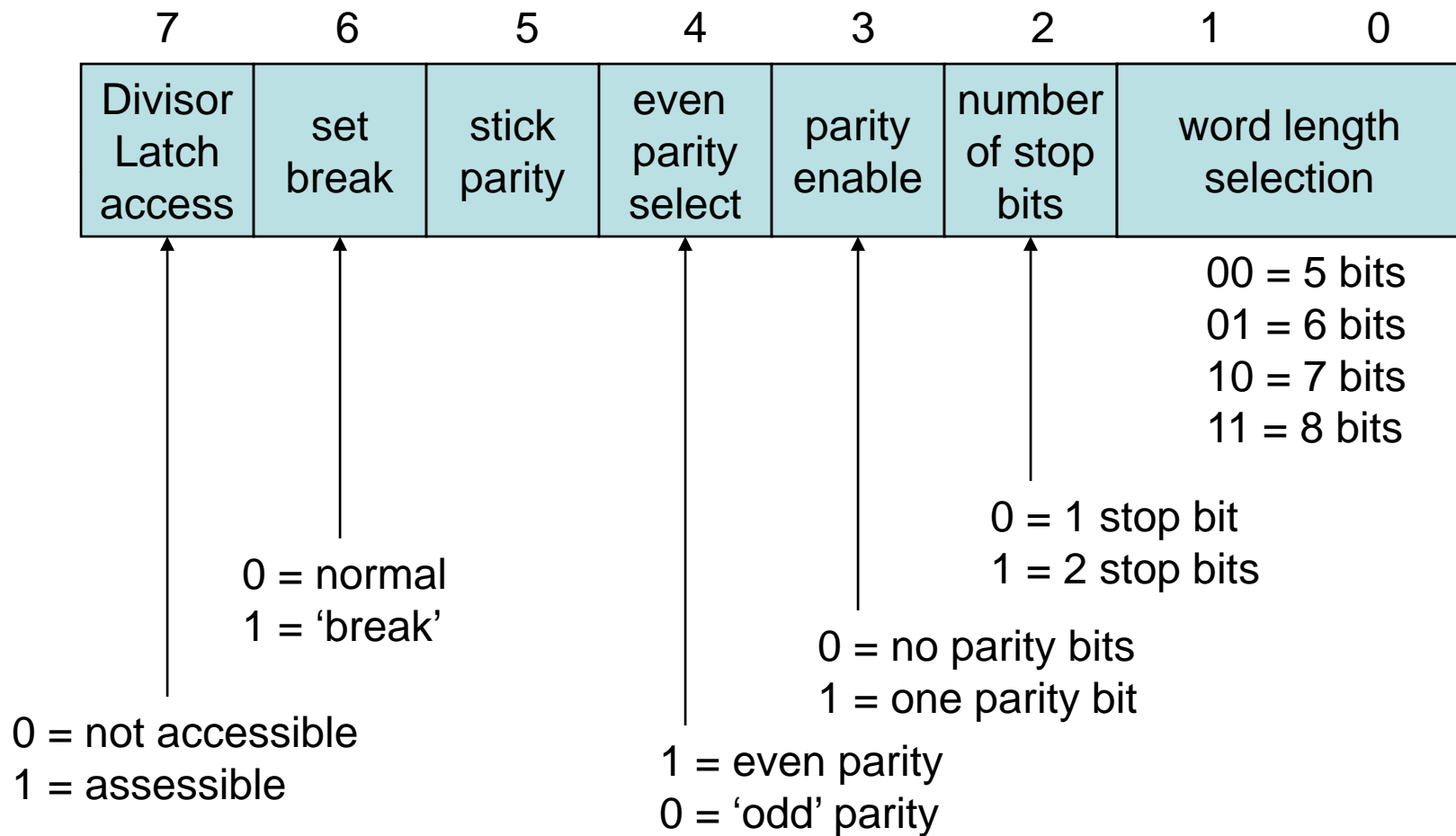
These status-bits indicate errors in the received data

This status-bit indicates that the data-transmission has been completed

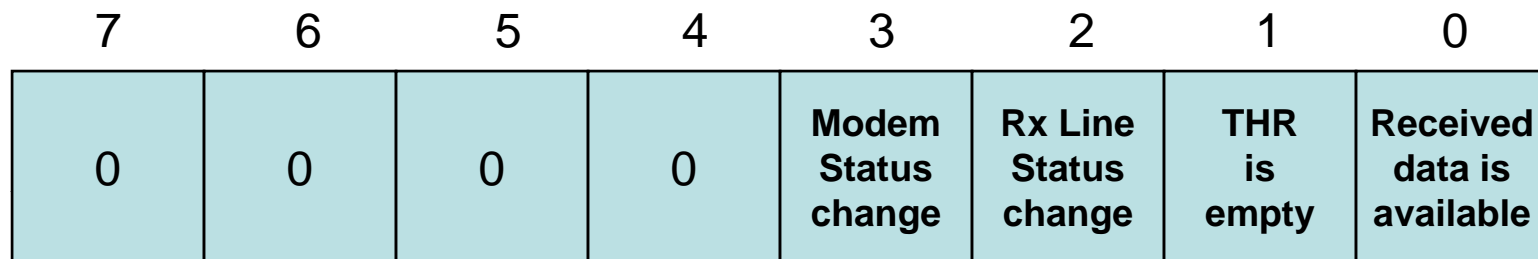
This status-bit indicates that the Transmitter Holding Register is ready to accept a new data byte

This status-bit indicates that a new byte of data has arrived (or, in FIFO-mode, that the receiver-FIFO has reached its threshold)

Line Control Register



Interrupt Enable Register



If **enabled** (by setting the bit to 1),
the UART will generate an interrupt:

(bit 3) whenever modem status changes

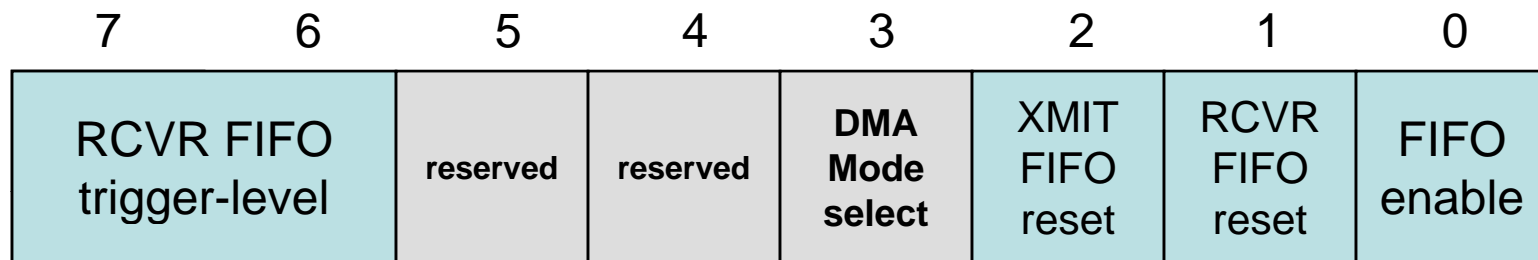
(bit 2) whenever a receive-error is detected

(bit 1) whenever the transmit-buffer is empty

(bit 0) whenever the receive-buffer is nonempty

Also, in FIFO mode, a 'timeout' interrupt will be generated if neither FIFO has been 'serviced' for at least four character-clock times

FIFO Control Register



00 = 1 byte
01 = 4 bytes
10 = 8 bytes
11 = 14 bytes

NOTE: DMA is unsupported for the UART on our systems

Writing 1 empties the FIFO, writing 0 has no effect

Writing 0 will disable the UART's FIFO-mode, writing 1 will enable FIFO-mode

Interrupt Identification Register

