**class UART – duplex serial communication bus**

UART implements the standard UART/USART duplex serial communications protocol. At the physical level it consists of 2 lines: RXD and TXD. The unit of communication is a character (not to be confused with a string character) which can be 5, 6, 7 or 8 bits wide.

UART objects can be created and initialized using:

**from** machine **import** UART

uart **=** UART(1, 9600) *# init with given baudrate*

uart**.**init(9600, bits**=**8, parity**=None**, stop**=**1) *# init with given parameters*

Bits can be 5, 6, 7, 8. Parity can be None, UART.EVEN or UART.ODD. Stop can be 1, 1.5 or 2.

A UART object acts like a stream object therefore reading and writing is done using the standard stream methods:

uart**.**read(10) *# read 10 characters, returns a bytes object*

uart**.**readall() *# read all available characters*

uart**.**readline() *# read a line*

uart**.**readinto(buf) *# read and store into the given buffer*

uart**.**write('abc') *# write the 3 characters*

To check if there is anything to be read, use:

uart**.**any() *# returns the number of characters available for reading*

**Quick usage example**

**from** machine **import** UART

*# this uses the UART\_1 default pins for TXD and RXD (``P3`` and ``P4``)*

uart **=** UART(1, baudrate**=**9600)

uart**.**write('hello')

uart**.**read(5) *# read up to 5 bytes*

**Constructors**

***class*machine.UART(*bus*, *...*)**

Construct a UART object on the given bus. **bus** can be 0 or 1. If the bus is not given, the default one will be selected (0) or the selection will be made based on the given pins.

**Methods**

**uart.init(*baudrate=9600*, *bits=8*, *parity=None*, *stop=1*, *\**, *timeout\_chars=2*, *pins=(TXD*, *RXD*, *RTS*, *CTS)*)**

Initialize the UART bus with the given parameters:

* **baudrate** is the clock rate.
* **bits** is the number of bits per character. Can be 5, 6, 7 or 8.
* **parity** is the parity, **None**, **UART.EVEN** or **UART.ODD**.
* **stop** is the number of stop bits, 1 or 2.
* **timeout\_chars** Rx timeout defined in number of characters. The value given here will be multiplied by the time a characters takes to be transmitted at the configured baudrate.
* **pins** is a 4 or 2 item list indicating the TXD, RXD, RTS and CTS pins (in that order). Any of the pins can be None if one wants the UART to operate with limited functionality. If the RTS pin is given the the RX pin must be given as well. The same applies to CTS. When no pins are given, then the default set of TXD (**P1**) and RXD (**P0**) pins is taken, and hardware flow control will be disabled. If pins=None, no pin assignment will be made.

**uart.deinit()**

Turn off the UART bus.

**uart.any()**

Return the number of characters available for reading.

**uart.read([*nbytes*])**

Read characters. If **nbytes** is specified then read at most that many bytes.

Return value: a bytes object containing the bytes read in. Returns **None** on timeout.

**uart.readall()**

Read as much data as possible.

Return value: a bytes object or **None** on timeout.

**uart.readinto(*buf*[, *nbytes*])**

Read bytes into the **buf**. If **nbytes** is specified then read at most that many bytes. Otherwise, read at most **len(buf)** bytes.

Return value: number of bytes read and stored into **buf** or **None** on timeout.

**uart.readline()**

Read a line, ending in a newline character. If such a line exists, return is immediate. If the timeout elapses, all available data is returned regardless of whether a newline exists.

Return value: the line read or None on timeout if no data is available.

**uart.write(*buf*)**

Write the buffer of bytes to the bus.

Return value: number of bytes written or **None** on timeout.

**uart.sendbreak()**

Send a break condition on the bus. This drives the bus low for a duration of 13 bits. Return value: **None**.

**Constants**

**UART.EVEN**

**UART.ODD**

parity types (along with **None**)

**UART.RX\_ANY**

IRQ trigger sources