modbus\_rtu\_set\_rts(3)

=====================

NAME

----

modbus\_rtu\_set\_rts - set the RTS mode in RTU

SYNOPSIS

--------

\*int modbus\_rtu\_set\_rts(modbus\_t \*'ctx', int 'mode')\*

DESCRIPTION

-----------

The \*modbus\_rtu\_set\_rts()\* function shall set the Request To Send mode to

communicate on a RS485 serial bus. By default, the mode is set to

`MODBUS\_RTU\_RTS\_NONE` and no signal is issued before writing data on the wire.

To enable the RTS mode, the values `MODBUS\_RTU\_RTS\_UP` or `MODBUS\_RTU\_RTS\_DOWN`

must be used, these modes enable the RTS mode and set the polarity at the same

time. When `MODBUS\_RTU\_RTS\_UP` is used, an ioctl call is made with RTS flag

enabled then data is written on the bus after a delay of 1 ms, then another

ioctl call is made with the RTS flag disabled and again a delay of 1 ms occurs.

The `MODBUS\_RTU\_RTS\_DOWN` mode applies the same procedure but with an inverted

RTS flag.

This function can only be used with a context using a RTU backend.

RETURN VALUE

------------

The function shall return 0 if successful. Otherwise it shall return -1 and set

errno to one of the values defined below.

ERRORS

------

\*EINVAL\*::

The libmodbus backend isn't RTU or the mode given in argument is invalid.

EXAMPLE

-------

.Enable the RTS mode with positive polarity

[source,c]

-------------------

modbus\_t \*ctx;

uint16\_t tab\_reg[10];

ctx = modbus\_new\_rtu("/dev/ttyS0", 115200, 'N', 8, 1);

modbus\_set\_slave(ctx, 1);

modbus\_rtu\_set\_serial\_mode(ctx, MODBUS\_RTU\_RS485);

modbus\_rtu\_set\_rts(ctx, MODBUS\_RTU\_RTS\_UP);

if (modbus\_connect(ctx) == -1) {

fprintf(stderr, "Connexion failed: %s\n", modbus\_strerror(errno));

modbus\_free(ctx);

return -1;

}

rc = modbus\_read\_registers(ctx, 0, 7, tab\_reg);

if (rc == -1) {

fprintf(stderr, "%s\n", modbus\_strerror(errno));

return -1;

}

modbus\_close(ctx);

modbus\_free(ctx);

-------------------

SEE ALSO

--------

linkmb:modbus\_rtu\_get\_rts[3]

AUTHORS

-------

The libmodbus documentation was written by Stéphane Raimbault

<stephane.raimbault@gmail.com>