API documentation for libcomm

Contents

1 Overview

This library provides interface to connect to the logical serial ports.

2 comm.eh — Access to the COMM, IrDA, USB ports.

```
use "comm.eh"
```

2.1 Description

This header defines interface for logical serial port connections. A ”logical” serial port is a connection through which bytes are transferring serially. The logical serial port is not necessarily correspond to a physical RS-232 serial port. For instance, IrDA IRCOMM port can commonly be configured so that it can act as a ”logical” serial port.

Only one application may be connected to a particular serial port at a given time. An attempt to open connection with busy serial port ends with I/O error.

2.1.1 Connection parameters

Connection may be given additional parameters which are specified in `CommCfg`. The convenient way to pass these parameters is through the structure constructor:

```
var comm = new Comm("usb0", new CommCfg{baudrate=19200, parity="even"})
```

If some of parameters are skipped, the defaults will be used.

Parameter	Default	Description
`baudrate`	`platform dependent`	The speed of the port.
`bitsperchar`	`8`	The number bits per character (`7` or `8`).
`stopbits`	`1`	The number of stop bits per char (`1` or `2`).
`parity`	`"none"`	The parity can be `"odd"`, `"even"`, or `"none"`.
`blocking`	`true`	If `true`, wait for a full buffer when reading.
`autocts`	`true`	If `true`, wait for the CTS line to be on before writing.
`autorts`	`true`	If `true`, turn on the RTS line when the input buffer is not full. If `false`, the RTS line is always on.

2.1.2 Example

The following example shows how a `Comm` connection would be used in a simple loopback program.

```
var comm = new Comm("usb0", new CommCfg{baudrate=19200})
var baudrate = comm.baudrate
var in = comm.open_input()
var out = comm.open_output()
var ch = 0
while (ch != 'Z') {
  out.write(ch)
  ch = in.read()
  ...
}
in.close()
out.close()
comm.close()
```

2.2 Types

```
type Comm < StreamConnection;
```

COMM connection.

```
type CommCfg = {
  baudrate: Int,
  bitsperchar: Int,
  stopbits: Int,
  parity: String,
  blocking: Bool,
  autocts: Bool,
  autorts: Bool
}
```

Configuration for the COMM connection. If some fields are unset, the defaults will be used when creating COMM connection.

2.3 Functions

```
def list_commports(): [String];
```

Returns list of available COMM ports. If no ports are available on the device, then array with zero length is returned.

```
def Comm.new(port: String, cfg: CommCfg): Comm;
```

Creates new COMM connection with given port and configuration.

```
def Comm.get_baudrate(): Int;
```

Returns the baudrate of the serial port connection.

```
def Comm.set_baudrate(baudrate: Int);
```

Sets the baudrate for the serial port connection. If the requested *baudrate* is not supported on the platform, then the system uses an alternate valid setting. The set baudrate can be determined with get_baudrate.