```c
1: /*
2:     libxbee - a C library to aid the use of Digi's Series 1 XBee modules
3:              running in API mode (AP=2).
4:
5:     Copyright (C) 2009  Attie Grande (attie@attie.co.uk)
6:
7:     This program is free software: you can redistribute it and/or modify
8:     it under the terms of the GNU General Public License as published by
9:     the Free Software Foundation, either version 3 of the License, or
10:    (at your option) any later version.
11:
12:    This program is distributed in the hope that it will be useful,
13:    but WITHOUT ANY WARRANTY; without even the implied warranty of
14:    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15:    GNU General Public License for more details.
16:
17:    You should have received a copy of the GNU General Public License
18:    along with this program.  If not, see <http://www.gnu.org/licenses/>.
19: */
20:
21: /* ################################################################ */
22: /* ### Linux Code ################################################# */
23: /* ################################################################ */
24:
25: /*  this file contains code that is used by Linux ONLY */
26: #ifndef __GNUC__
27: #error "This file should only be used on a Linux system"
28: #endif
29:
30: #include "linux.h"
31:
32: int init_serial(int baudrate) {
33:   struct flock fl;
34:   struct termios tc;
35:   speed_t chosenbaud;
36:
37:   /* select the baud rate */
38:   switch (baudrate) {
39:   case 1200:  chosenbaud = B1200;   break;
40:   case 2400:  chosenbaud = B2400;   break;
41:   case 4800:  chosenbaud = B4800;   break;
42:   case 9600:  chosenbaud = B9600;   break;
43:   case 19200: chosenbaud = B19200;  break;
44:   case 38400: chosenbaud = B38400;  break;
45:   case 57600: chosenbaud = B57600;  break;
46:   case 115200:chosenbaud = B115200; break;
47:   default:
48:     fprintf(stderr,"%s(): Unknown or incompatiable baud rate specified... (%d)\n",__FUNCTION__,baudrate);
49:     return -1;
50:   };
51:
52:   /* open the serial port as a file descriptor */
53:   if ((xbee.ttyfd = open(xbee.path,O_RDWR | O_NOCTTY | O_NONBLOCK)) == -1) {
54:     perror("xbee_setup():open()");
55:     xbee_mutex_destroy(xbee.conmutex);
56:     xbee_mutex_destroy(xbee.pktmutex);
57:     xbee_mutex_destroy(xbee.sendmutex);
58:     Xfree(xbee.path);
59:     return -1;
60:   }
61:
62:   /* lock the file */
63:   fl.l_type = F_WRLCK | F_RDLCK;
64:   fl.l_whence = SEEK_SET;
65:   fl.l_start = 0;
66:   fl.l_len = 0;
67:   fl.l_pid = getpid();
68:   if (fcntl(xbee.ttyfd, F_SETLK, &fl) == -1) {
69:     perror("xbee_setup():fcntl()");
70:     xbee_mutex_destroy(xbee.conmutex);
71:     xbee_mutex_destroy(xbee.pktmutex);
72:     xbee_mutex_destroy(xbee.sendmutex);
73:     Xfree(xbee.path);
74:     close(xbee.ttyfd);
75:     return -1;
76:   }
77:
78:   /* open the serial port as a FILE* */
79:   if ((xbee.tty = fdopen(xbee.ttyfd,"r+")) == NULL) {
80:     perror("xbee_setup():fdopen()");
81:     xbee_mutex_destroy(xbee.conmutex);
82:     xbee_mutex_destroy(xbee.pktmutex);
83:     xbee_mutex_destroy(xbee.sendmutex);
84:     Xfree(xbee.path);
85:     close(xbee.ttyfd);
```

```c
 86:        return -1;
 87:    }
 88:
 89:    /* flush the serial port */
 90:    fflush(xbee.tty);
 91:
 92:    /* disable buffering */
 93:    setvbuf(xbee.tty,NULL,_IONBF,BUFSIZ);
 94:
 95:    /* setup the baud rate and other io attributes */
 96:    tcgetattr(xbee.ttyfd, &tc);
 97:    /* input flags */
 98:    tc.c_iflag &= ~ IGNBRK;              /* enable ignoring break */
 99:    tc.c_iflag &= ~(IGNPAR | PARMRK); /* disable parity checks */
100:    tc.c_iflag &= ~ INPCK;              /* disable parity checking */
101:    tc.c_iflag &= ~ ISTRIP;             /* disable stripping 8th bit */
102:    tc.c_iflag &= ~(INLCR | ICRNL);   /* disable translating NL <-> CR */
103:    tc.c_iflag &= ~ IGNCR;              /* disable ignoring CR */
104:    tc.c_iflag &= ~(IXON | IXOFF);    /* disable XON/XOFF flow control */
105:    /* output flags */
106:    tc.c_oflag &= ~ OPOST;              /* disable output processing */
107:    tc.c_oflag &= ~(ONLCR | OCRNL);   /* disable translating NL <-> CR */
108:    tc.c_oflag &= ~ OFILL;              /* disable fill characters */
109:    /* control flags */
110:    tc.c_cflag |=   CREAD;              /* enable reciever */
111:    tc.c_cflag &= ~ PARENB;             /* disable parity */
112:    tc.c_cflag &= ~ CSTOPB;             /* disable 2 stop bits */
113:    tc.c_cflag &= ~ CSIZE;              /* remove size flag... */
114:    tc.c_cflag |=   CS8;                /* ...enable 8 bit characters */
115:    tc.c_cflag |=   HUPCL;              /* enable lower control lines on close - hang up */
116:    /* local flags */
117:    tc.c_lflag &= ~ ISIG;               /* disable generating signals */
118:    tc.c_lflag &= ~ ICANON;             /* disable canonical mode - line by line */
119:    tc.c_lflag &= ~ ECHO;               /* disable echoing characters */
120:    tc.c_lflag &= ~ ECHONL;             /* ??? */
121:    tc.c_lflag &= ~ NOFLSH;             /* disable flushing on SIGINT */
122:    tc.c_lflag &= ~ IEXTEN;             /* disable input processing */
123:    /* control characters */
124:    memset(tc.c_cc,0,sizeof(tc.c_cc));
125:    /* i/o rates */
126:    cfsetspeed(&tc, chosenbaud);      /* set i/o baud rate */
127:    tcsetattr(xbee.ttyfd, TCSANOW, &tc);
128:    tcflow(xbee.ttyfd, TCOON|TCION); /* enable input & output transmission */
129:
130:    return 0;
131: }
132:
133: static int xbee_select(struct timeval *timeout) {
134:    fd_set fds;
135:
136:    FD_ZERO(&fds);
137:    FD_SET(xbee.ttyfd, &fds);
138:
139:    return select(xbee.ttyfd+1, &fds, NULL, NULL, timeout);
140: }
```