```
/*
  libxbee - a C library to aid the use of Digi's Series 1 XBee modules
            running in API mode (AP=2).

  Copyright (C) 2009  Attie Grande (attie@attie.co.uk)

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#ifndef XBEE_H
#define XBEE_H

#if !defined(__GNUC__) && !defined(_WIN32)
#error "This library is only currently compatible with Linux and Win32"
#endif

#ifdef __cplusplus
extern "C" {
#endif

#ifndef __LIBXBEE_API_H
typedef void* xbee_hnd;
#endif

#include <stdarg.h>

#ifdef __GNUC__ /* ---- */
#include <semaphore.h>
typedef pthread_mutex_t    xbee_mutex_t;
typedef pthread_cond_t     xbee_cond_t;
typedef pthread_t          xbee_thread_t;
typedef sem_t              xbee_sem_t;
typedef FILE*              xbee_file_t;
#elif (defined(WIN32) || defined(_WIN32)) /* -------------- */
#include <Windows.h>
#define CALLTYPE   __stdcall
#define CALLTYPEVA __cdecl
typedef HANDLE             xbee_mutex_t;
typedef CONDITION_VARIABLE xbee_cond_t;
typedef HANDLE             xbee_thread_t;
typedef HANDLE             xbee_sem_t;
typedef HANDLE             xbee_file_t;
#else
#error "Unknown operating system or compiler"
#endif /* ------------- */

#ifndef CALLTYPE
#define CALLTYPE
#endif

#ifndef CALLTYPEVA
#define CALLTYPEVA
#endif

enum xbee_types {
  xbee_unknown,

  xbee_localAT,       /* frame ID */
  xbee_remoteAT,
  xbee_modemStatus,
  xbee_txStatus,

  /* XBee Series 1 stuff */
  xbee_16bitRemoteAT, /* frame ID */
  xbee_64bitRemoteAT, /* frame ID */

  xbee_16bitData,     /* frame ID for ACKs */
  xbee_64bitData,     /* frame ID for ACKs */

  xbee_16bitIO,
  xbee_64bitIO,

  /* XBee Series 2 stuff */
  xbee2_data,
  xbee2_txStatus
```

```
 86: };
 87: typedef enum xbee_types xbee_types;
 88:
 89: typedef struct xbee_sample xbee_sample;
 90: struct xbee_sample {
 91:   /* X  A5 A4 A3 A2 A1 A0 D8    D7 D6 D5 D4 D3 D2 D1 D0  */
 92:   unsigned short IOmask;          /*                  IO */
 93:   /* X  X  X  X  X  X  X  D8    D7 D6 D5 D4 D3 D2 D1 D0  */
 94:   unsigned short IOdigital;       /*                  IO */
 95:   /* X  X  X  X  X  D  D  D     D  D  D  D  D  D  D  D   */
 96:   unsigned short IOanalog[6];     /*                  IO */
 97: };
 98:
 99: typedef struct xbee_pkt xbee_pkt;
100: struct xbee_pkt {
101:   unsigned int sAddr64        : 1; /* TRUE / FALSE */
102:   unsigned int dataPkt        : 1;
103:   unsigned int txStatusPkt    : 1;
104:   unsigned int modemStatusPkt : 1;
105:   unsigned int remoteATPkt    : 1;
106:   unsigned int IOPkt          : 1;
107:
108:   unsigned int isBroadcastADR : 1;/* if TRUE, dest addr was 0xFFFF */
109:   unsigned int isBroadcastPAN : 1;/* if TRUE, dest PAN  was 0xFFFF */
110:
111:   unsigned char frameID;          /* AT        Status    */
112:   unsigned char atCmd[2];         /* AT                  */
113:
114:   unsigned char status;           /* AT  Data  Status    */ /* status / options */
115:   unsigned char samples;
116:   unsigned char RSSI;             /*     Data            */
117:
118:   unsigned char Addr16[2];        /* AT  Data            */
119:
120:   unsigned char Addr64[8];        /* AT  Data            */
121:
122:   unsigned char data[128];        /* AT  Data            */
123:   unsigned int datalen;
124:
125:   xbee_types type;
126:
127:   xbee_pkt *next;
128:
129:   xbee_sample IOdata[1];  /* this array can be extended by using a this trick:
130:                              p = calloc(sizeof(xbee_pkt) + (sizeof(xbee_sample) * (samples - 1))) */
131: };
132:
133: typedef struct xbee_con xbee_con;
134: struct xbee_con {
135:   unsigned int tAddr64       : 1;
136:   unsigned int atQueue       : 1; /* queues AT commands until AC is sent */
137:   unsigned int txDisableACK  : 1;
138:   unsigned int txBroadcastPAN: 1; /* broadcasts to PAN */
139:   unsigned int destroySelf   : 1; /* if set, the callback thread will destroy the connection
140:                                      after all of the packets have been processed */
141:   unsigned int waitforACK    : 1; /* waits for the ACK or NAK after transmission */
142:   unsigned int noFreeAfterCB : 1; /* prevents libxbee from free'ing the packet after
143:                                      the callback has completed */
144:   unsigned int __spare__     : 1;
145:   xbee_types type;
146:   unsigned char frameID;
147:   unsigned char tAddr[8];         /* 64-bit 0-7   16-bit 0-1 */
148:   void *customData;               /* can be used to store data related to this connection */
149:   void (*callback)(xbee_con*,xbee_pkt*); /* callback function */
150:   void *callbackList;
151:   xbee_mutex_t callbackmutex;
152:   xbee_mutex_t callbackListmutex;
153:   xbee_mutex_t Txmutex;
154:   xbee_sem_t waitforACKsem;
155:   volatile unsigned char ACKstatus; /* 255 = waiting, 0 = success, 1 = no ack, 2 = cca fail, 3 = purged */
156:   xbee_con *next;
157: };
158:
159: int CALLTYPE xbee_setup(char *path, int baudrate);
160: int CALLTYPE xbee_setuplog(char *path, int baudrate, int logfd);
161: int CALLTYPE xbee_setupAPI(char *path, int baudrate, char cmdSeq, int cmdTime);
162: int CALLTYPE xbee_setuplogAPI(char *path, int baudrate, int logfd, char cmdSeq, int cmdTime);
163: xbee_hnd CALLTYPE _xbee_setup(char *path, int baudrate);
164: xbee_hnd CALLTYPE _xbee_setuplog(char *path, int baudrate, int logfd);
165: xbee_hnd CALLTYPE _xbee_setupAPI(char *path, int baudrate, char cmdSeq, int cmdTime);
166: xbee_hnd CALLTYPE _xbee_setuplogAPI(char *path, int baudrate, int logfd, char cmdSeq, int cmdTime);
167:
168: int CALLTYPE xbee_end(void);
169: int CALLTYPE _xbee_end(xbee_hnd xbee);
170:
```

```
171: void CALLTYPE xbee_logitf(char *format, ...);
172: void CALLTYPE _xbee_logitf(xbee_hnd xbee, char *format, ...);
173: void CALLTYPE xbee_logit(char *str);
174: void CALLTYPE _xbee_logit(xbee_hnd xbee, char *str);
175:
176: xbee_con * CALLTYPEVA xbee_newcon(unsigned char frameID, xbee_types type, ...);
177: xbee_con * CALLTYPEVA _xbee_newcon(xbee_hnd xbee, unsigned char frameID, xbee_types type, ...);
178: xbee_con * CALLTYPE _xbee_vnewcon(xbee_hnd xbee, unsigned char frameID, xbee_types type, va_list ap);
179:
180: void CALLTYPE xbee_purgecon(xbee_con *con);
181: void CALLTYPE _xbee_purgecon(xbee_hnd xbee, xbee_con *con);
182:
183: void CALLTYPE xbee_endcon2(xbee_con **con, int alreadyUnlinked);
184: void CALLTYPE _xbee_endcon2(xbee_hnd xbee, xbee_con **con, int alreadyUnlinked);
185: #define xbee_endcon(x) xbee_endcon2(&(x),0)
186: #define _xbee_endcon(xbee,x) _xbee_endcon2((xbee),&(x),0)
187:
188: int CALLTYPE xbee_nsenddata(xbee_con *con, char *data, int length);
189: int CALLTYPE _xbee_nsenddata(xbee_hnd xbee, xbee_con *con, char *data, int length);
190: int CALLTYPEVA xbee_senddata(xbee_con *con, char *format, ...);
191: int CALLTYPEVA _xbee_senddata(xbee_hnd xbee, xbee_con *con, char *format, ...);
192: int CALLTYPE xbee_vsenddata(xbee_con *con, char *format, va_list ap);
193: int CALLTYPE _xbee_vsenddata(xbee_hnd xbee, xbee_con *con, char *format, va_list ap);
194:
195: #if defined(WIN32)
196: /* oh and just 'cos windows has rubbish memory management rules... this too */
197: void CALLTYPE xbee_free(void *ptr);
198: #endif /* ------------- */
199:
200: xbee_pkt * CALLTYPE xbee_getpacket(xbee_con *con);
201: xbee_pkt * CALLTYPE _xbee_getpacket(xbee_hnd xbee, xbee_con *con);
202: xbee_pkt * CALLTYPE xbee_getpacketwait(xbee_con *con);
203: xbee_pkt * CALLTYPE _xbee_getpacketwait(xbee_hnd xbee, xbee_con *con);
204:
205: int CALLTYPE xbee_hasdigital(xbee_pkt *pkt, int sample, int input);
206: int CALLTYPE xbee_getdigital(xbee_pkt *pkt, int sample, int input);
207:
208: int CALLTYPE xbee_hasanalog(xbee_pkt *pkt, int sample, int input);
209: double CALLTYPE xbee_getanalog(xbee_pkt *pkt, int sample, int input, double Vref);
210:
211: const char * CALLTYPE xbee_svn_version(void);
212: const char * CALLTYPE xbee_build_info(void);
213:
214: void CALLTYPE xbee_listen_stop(xbee_hnd xbee);
215:
216: #ifdef __cplusplus
217: } /* cplusplus */
218: #endif
219:
220: #endif
```