The Linux LAPB Module Interface

Version 1.3

Jonathan Naylor 29.12.96

Changed (Henner Eisen, 2000-10-29): int return value for data_indication()

The LAPB module will be a separately compiled module for use by any parts of the Linux operating system that require a LAPB service. This document defines the interfaces to, and the services provided by this module. The term module in this context does not imply that the LAPB module is a separately loadable module, although it may be. The term module is used in its more standard meaning.

The interface to the LAPB module consists of functions to the module, callbacks from the module to indicate important state changes, and structures for getting and setting information about the module.

Structures

Probably the most important structure is the skbuff structure for holding received and transmitted data, however it is beyond the scope of this document.

The two LAPB specific structures are the LAPB initialisation structure and the LAPB parameter structure. These will be defined in a standard header file, <linux/lapb.h>. The header file <net/lapb.h> is internal to the LAPB module and is not for use.

LAPB Initialisation Structure

This structure is used only once, in the call to lapb_register (see below). It contains information about the device driver that requires the services of the LAPB module:

```
struct lapb_register_struct {
        void (*connect_confirmation)(int token, int reason);
        void (*connect_indication)(int token, int reason);
        void (*disconnect_confirmation)(int token, int reason);
        void (*disconnect_indication)(int token, int reason);
        int  (*data_indication)(int token, struct sk_buff *skb);
        void (*data_transmit)(int token, struct sk_buff *skb);
};
```

Each member of this structure corresponds to a function in the device driver that is called when a particular event in the LAPB module occurs. These will be described in detail below. If a callback is not required (!!) then a NULL may be substituted.

LAPB Parameter Structure

This structure is used with the lapb_getparms and lapb_setparms functions (see below). They are used to allow the device driver to get and set the operational parameters of the LAPB implementation for a given connection:

```
struct lapb_parms_struct {
        unsigned int t1;
        unsigned int t1timer;
        unsigned int t2;
        unsigned int t2timer;
        unsigned int n2;
        unsigned int n2count;
        unsigned int window;
        unsigned int state;
        unsigned int mode;
};
```

T1 and T2 are protocol timing parameters and are given in units of 100ms. N2 is the maximum number of tries on the link before it is declared a failure. The window size is the maximum number of outstanding data packets allowed to be unacknowledged by the remote end, the value of the window is between 1 and 7 for a standard LAPB link, and between 1 and 127 for an extended LAPB link.

The mode variable is a bit field used for setting (at present) three values. The bit fields have the following meanings:

Bit	Meaning
0	LAPB operation (0=LAPB_STANDARD 1=LAPB_EXTENDED).
1	[SM]LP operation (0=LAPB_SLP 1=LAPB=MLP).
2	DTE/DCE operation (0=LAPB_DTE 1=LAPB_DCE)
3-31	Reserved, must be 0.

Extended LAPB operation indicates the use of extended sequence numbers and consequently larger window sizes, the default is standard LAPB operation. MLP operation is the same as SLP operation except that the addresses used by LAPB are different to indicate the mode of operation, the default is Single Link Procedure. The difference between DCE and DTE operation is (i) the addresses used for commands and responses, and (ii) when the DCE is not connected, it sends DM without polls set, every T1. The upper case constant names will be defined in the public LAPB header file.

Functions

The LAPB module provides a number of function entry points.

```
int lapb_register(void *token, struct lapb_register_struct);
```

This must be called before the LAPB module may be used. If the call is successful then LAPB_OK is returned. The token must be a unique identifier generated by the device driver to allow for the unique identification of the instance of the LAPB link. It is returned by the LAPB module in all of the callbacks, and is used by the device driver in all calls to the LAPB module. For multiple LAPB links in a single device driver, multiple calls to lapb_register must be made. The format of the lapb_register_struct is given above. The return values are:

LAPB_OK	LAPB registered successfully.
LAPB_BADTOKEN	Token is already registered.
LAPB_NOMEM	Out of memory

```
int lapb_unregister(void *token);
```

This releases all the resources associated with a LAPB link. Any current LAPB link will be abandoned without further messages being passed. After this call, the value of token is no longer valid for any calls to the LAPB function. The valid return values are:

LAPB_OK	LAPB unregistered successfully.
LAPB_BADTOKEN	Invalid/unknown LAPB token.

```
int lapb_getparms(void *token, struct lapb_parms_struct *parms);
```

This allows the device driver to get the values of the current LAPB variables, the lapb_parms_struct is described above. The valid return values are:

LAPB_OK	LAPB getparms was successful.
LAPB_BADTOKEN	Invalid/unknown LAPB token.

```
int lapb_setparms(void *token, struct lapb_parms_struct *parms);
```

This allows the device driver to set the values of the current LAPB variables, the lapb_parms_struct is described above. The values of t1timer, t2timer and n2count are ignored, likewise changing the mode bits when connected will be ignored. An error implies that none of the values have been changed. The valid return values are:

LAPB_OK	LAPB getparms was successful.
LAPB_BADTOKEN	Invalid/unknown LAPB token.
LAPB_INVALUE	One of the values was out of its allowable range.

```
int lapb_connect_request(void *token);
```

Initiate a connect using the current parameter settings. The valid return values are:

LAPB_OK	LAPB is starting to connect.
LAPB_BADTOKEN	Invalid/unknown LAPB token.
LAPB_CONNECTED	LAPB module is already connected.

```
int lapb_disconnect_request(void *token);
```

Initiate a disconnect. The valid return values are:

LAPB_OK	LAPB is starting to disconnect.
LAPB_BADTOKEN	Invalid/unknown LAPB token.
LAPB_NOTCONNECTED	LAPB module is not connected.

```
int lapb_data_request(void *token, struct sk_buff *skb);
```

Queue data with the LAPB module for transmitting over the link. If the call is successful then the skbuff is owned by the LAPB module and may not be used by the device driver again. The valid return values are:

LAPB_OK	LAPB has accepted the data.
LAPB_BADTOKEN	Invalid/unknown LAPB token.
LAPB_NOTCONNECTED	LAPB module is not connected.

```
int lapb_data_received(void *token, struct sk_buff *skb);
```

Queue data with the LAPB module which has been received from the device. It is expected that the data passed to the LAPB module has skb->data pointing to the beginning of the LAPB data. If the call is successful then the skbuff is owned by the LAPB module and may not be used by the device driver again. The valid return values are:

LAPB_OK	LAPB has accepted the data.
LAPB_BADTOKEN	Invalid/unknown LAPB token.

Callbacks

These callbacks are functions provided by the device driver for the LAPB module to call when an event occurs. They are registered with the LAPB module with lapb_register (see above) in the structure lapb_register_struct (see above).

```
void (*connect_confirmation)(void *token, int reason);
```

This is called by the LAPB module when a connection is established after being requested by a call to lapb_connect_request (see above). The reason is always LAPB_OK.

```
void (*connect_indication)(void *token, int reason);
```

This is called by the LAPB module when the link is established by the remote system. The value of reason is always LAPB_OK.

```
void (*disconnect_confirmation)(void *token, int reason);
```

This is called by the LAPB module when an event occurs after the device driver has called lapb_disconnect_request (see above). The reason indicates what has happened. In all cases the LAPB link can be regarded as being terminated. The values for reason are:

LAPB_OK	The LAPB link was terminated normally.
LAPB_NOTCONNECTED	The remote system was not connected.
LAPB_TIMEDOUT	No response was received in N2 tries from the remote system.

```
void (*disconnect_indication)(void *token, int reason);
```

This is called by the LAPB module when the link is terminated by the remote system or another event has occurred to terminate the link. This may be returned in response to a lapb_connect_request (see above) if the remote system refused the request. The values for reason are:

LAPB_OK	The LAPB link was terminated normally by the remote system.
LAPB_REFUSED	The remote system refused the connect request.
LAPB_NOTCONNECTED	The remote system was not connected.
LAPB_TIMEDOUT	No response was received in N2 tries from the remote system.

```
int (*data_indication)(void *token, struct sk_buff *skb);
```

This is called by the LAPB module when data has been received from the remote system that should be passed onto the next layer in the protocol stack. The skbuff becomes the property of the device driver and the LAPB module will not perform any more actions on it. The skb->data pointer will be pointing to the first byte of data after the LAPB header.

This method should return NET_RX_DROP (as defined in the header file include/linux/netdevice.h) if and only if the frame was dropped before it could be delivered to the upper layer.

```
void (*data_transmit)(void *token, struct sk_buff *skb);
```

This is called by the LAPB module when data is to be transmitted to the remote system by the device driver. The skbuff becomes the property of the device driver and the LAPB module will not perform any more actions on it. The skb->data pointer will be pointing to the first byte of the LAPB header.