```
/*
 *
 * Includes for cdc-acm.c
 *
 * Mainly take from usbnet's cdc-ether part
 *
 */

/*
 * CMSPAR, some architectures can't have space and mark parity.
 */

#ifndef CMSPAR
#define CMSPAR                  0
#endif

/*
 * Major and minor numbers.
 */

#define ACM_TTY_MAJOR       166
#define ACM_TTY_MINORS              256

/*
 * Requests.
 */

#define USB_RT_ACM          (USB_TYPE_CLASS | USB_RECIP_INTERFACE)

/*
 * Output control lines.
 */

#define ACM_CTRL_DTR        0x01
#define ACM_CTRL_RTS        0x02

/*
 * Input control lines and line errors.
 */

#define ACM_CTRL_DCD        0x01
#define ACM_CTRL_DSR        0x02
#define ACM_CTRL_BRK        0x04
#define ACM_CTRL_RI         0x08

#define ACM_CTRL_FRAMING    0x10
#define ACM_CTRL_PARITY             0x20
#define ACM_CTRL_OVERRUN    0x40

/*
 * Internal driver structures.
 */
```

```
/*
 * The only reason to have several buffers is to accommodate
assumptions
 * in line disciplines. They ask for empty space amount, receive
our URB size,
 * and proceed to issue several 1-character writes, assuming they
will fit.
 * The very first write takes a complete URB. Fortunately, this
only happens
 * when processing onlcr, so we only need 2 buffers. These values
must be
 * powers of 2.
 */
#define ACM_NW  16
#define ACM_NR  16

struct acm_wb {
      unsigned char *buf;
      dma_addr_t dmah;
      int len;
      int use;
      struct urb       *urb;
      struct acm       *instance;
};

struct acm_rb {
      int              size;
      unsigned char          *base;
      dma_addr_t       dma;
      int              index;
      struct acm       *instance;
};

struct acm {
      struct usb_device *dev;    /* the corresponding usb device */
      struct usb_interface *control;    /* control interface */
      struct usb_interface *data;q     /* data interface */
      unsigned in, out;                 /* i/o pipes */
      struct tty_port port;             /* our tty port data */
      struct urb *ctrlurb;              /* urbs */
      u8 *ctrl_buffer;                  /* buffers of urbs */
      dma_addr_t ctrl_dma;             /* dma handles of buffers */
      u8 *country_codes;           /* country codes from device */
      unsigned int country_code_size;  /* size of this buffer */
      unsigned int country_rel_date; /* release date of version */
      struct acm_wb wb[ACM_NW];
      unsigned long read_urbs_free;
      struct urb *read_urbs[ACM_NR];
      struct acm_rb read_buffers[ACM_NR];
      struct acm_wb *putbuffer;              /* for
acm_tty_put_char() */
```

```
	int rx_buflimit;
	spinlock_t read_lock;
	int write_used;					/* number of non-
empty write buffers */
	int transmitting;
	spinlock_t write_lock;
	struct mutex mutex;
	bool disconnected;
	unsigned long flags;
#		define EVENT_TTY_WAKEUP	0
#		define EVENT_RX_STALL 1
	struct usb_cdc_line_coding line;	/* bits, stop, parity
*/
	struct work_struct work;		/* work queue entry
for line discipline waking up */
	unsigned int ctrlin;			/* input control
lines (DCD, DSR, RI, break, overruns) */
	unsigned int ctrlout;			/* output control
lines (DTR, RTS) */
	struct async_icount iocount;		/* counters for
control line changes */
	struct async_icount oldcount;		/* for
comparison of counter */
	wait_queue_head_t wioctl;		/* for ioctl */
	unsigned int writesize;			/* max packet
size for the output bulk endpoint */
	unsigned int readsize,ctrlsize;		/* buffer sizes
for freeing */
	unsigned int minor;			/* acm minor number
*/
	unsigned char clocal;			/* termios CLOCAL */
	unsigned int ctrl_caps;			/* control
capabilities from the class specific header */
	unsigned int susp_count;		/* number of
suspended interfaces */
	unsigned int combined_interfaces:1;	/* control and
data collapsed */
	unsigned int throttled:1;		/* actually throttled
*/
	unsigned int throttle_req:1;		/* throttle
requested */
	u8 bInterval;
	struct usb_anchor delayed;		/* writes queued for
a device about to be woken */
	unsigned long quirks;
};

#define CDC_DATA_INTERFACE_TYPE	0x0a

/* constants describing various quirks and errors */
#define NO_UNION_NORMAL			BIT(0)
#define SINGLE_RX_URB			BIT(1)
```

```
#define NO_CAP_LINE              BIT(2)
#define NO_DATA_INTERFACE        BIT(4)
#define IGNORE_DEVICE            BIT(5)
#define QUIRK_CONTROL_LINE_STATE BIT(6)
#define CLEAR_HALT_CONDITIONS          BIT(7)
#define SEND_ZERO_PACKET         BIT(8)
```