FTDI Chip Commands

Introduction

The following list is of FTDI-specific commands sent via the ioctl system call to the USB driver. The usb_control_msg routine of libusb wraps these commands into the correct ioctl format. The 7 parameters of the usb_control_msg are:

```
struct usb_device                  The particular USB device to talk with
1-byte request type                Read (0xC0), Write(0x40), or read and write
1-byte command                     Defined by FTDI firmware
2-byte Value                       Defined by FTDI firmware
2-byte Index                       Defined by FTDI firmware
pointer to data structure          Defined by FTDI firmware
2-byte size of data structure      sizeof(data structure)
Timeout                            read or write Timeout (5000 milliseconds is typical)
```

These are the command messages issued by libftdi routines:

```
reset:             usb_control_msg(ftdi->usb_dev, 0x40, 0x00,       0,     Index,           NULL, 0, Timeout)

purge RX buffer:   usb_control_msg(ftdi->usb_dev, 0x40, 0x00,       1,     Index,           NULL, 0, Timeout)

purge TX buffer:   usb_control_msg(ftdi->usb_dev, 0x40, 0x00,       2,     Index,           NULL, 0, Timeout)

setflowctrl:       usb_control_msg(ftdi->usb_dev, 0x40, 0x01,   Value,     Index,           NULL, 0, Timeout)
                        Value encodes cc parameters, etc.  (See ftdi_sio.c in Linux Kernel for details.)
                        Index encodes the following flow control options:
                          0x00   Disable flow control on Channel A
                          0x01   Disable flow control on Channel B
                          0x10   Set RTS/CTS  flow control on Channel A
                          0x11   Set RTS/CTS  flow control on Channel B
                          0x20   Set DTR/DSR  flow control on Channel A
                          0x21   Set DTR/DSR  flow control on Channel B
                          0x40   Set XON/XOFF flow control on Channel A
                          0x41   Set XON/XOFF flow control on Channel B

set DTR:           usb_control_msg(ftdi->usb_dev, 0x40, 0x02,   Value, Interface,           NULL, 0, Timeout)
                        Value is 0x10 to set DTR Low, 0x11 to set DTR High
                        Interface is 0 for Channel A, 1 for Channel B

set RTS:           usb_control_msg(ftdi->usb_dev, 0x40, 0x02,   Value, Interface,           NULL, 0, Timeout)
                        Value is 0x20 to set RTS Low, 0x21 to set RTS High
                        Interface is 0 for Channel A, 1 for Channel B

set baudrate:      usb_control_msg(ftdi->usb_dev, 0x40, 0x03,   Value,     Index,           NULL, 0, Timeout)
                        Value and Index combined encode the divisor and output channel.
                               (See ftdi_convert_baudrate for details.)

set line property: usb_control_msg(ftdi->usb_dev, 0x40, 0x04,   Value, Interface,           NULL, 0, Timeout)
                        Value encodes the number of data bits, stop bits and parity.
                          Bits 0-7:   Number of data bits
                          Bits 8-10:  Parity: NONE=0x00, ODD=0x01, EVEN=0x02, MARK=0x03, SPACE=0x04
                          Bits 11-12: Stop bits: STOP_BIT_1=0x00, STOP_BIT_15=0x01, STOP_BIT_2=0x02
                        Interface is 0 for Channel A, 1 for Channel B

get modem status:  usb_control_msg(ftdi->usb_dev, 0xC0, 0x05,   Value, Interface, (char *)&Value, 0, Timeout)
                        Value is returned with the following bits set:
                          Bit 4:      CTS state
                          Bit 5:      DSR state
                          Bit 6:      RI state
                          Bit 7:      RLSD state
                        Interface is 0 for Channel A, 1 for Channel B

set event char:    usb_control_msg(ftdi->usb_dev, 0x40, 0x06,   Value, Interface,           NULL, 0, Timeout)
                        Value encodes the event character and whether it is to be used:
                          Bits 0-7:   Event character
                          Bit 8:      Event character enabled=1, disabled=0
```

```
                    Interface is 0 for Channel A, 1 for Channel B

set error char:     usb_control_msg(ftdi->usb_dev, 0x40, 0x07,   Value, Interface,          NULL, 0, Timeout)
                    Value encodes the error character and whether it is to be used:
                      Bits 0-7:   Error character
                      Bit 8:      Error character enabled=1, disabled=0

set latency time:   usb_control_msg(ftdi->usb_dev, 0x40, 0x09,   Value, Interface,          NULL, 0, Timeout)
                    Value has the latency time with Value between 1 and 255 milliseconds.)
                    Interface is 0 for Channel A, 1 for Channel B

get latency time:   usb_control_msg(ftdi->usb_dev, 0xC0, 0x0A,       0, Interface, (char *)&Value, 1, Timeout)
                    The current latency time will be placed in Value.
                    Interface is 0 for Channel A, 1 for Channel B

enable bitbang:     usb_control_msg(ftdi->usb_dev, 0x40, 0x0B,   Value,     Index,          NULL, 0, Timeout)
                    Value encodes the bitmask and bitbang mode (1=normal, 2=SPI bitbang mode).
                    Index encodes channel (1 = Channel A, 2 = Channel B)

disable bitbang:    usb_control_msg(ftdi->usb_dev, 0x40, 0x0B,       0,     Index,          NULL, 0, Timeout)
                    Index encodes channel (1 = Channel A, 2 = Channel B)

set bitmode:        usb_control_msg(ftdi->usb_dev, 0x40, 0x0B,   Value,     Index,          NULL, 0, Timeout)
                    Value encodes the bitmask in bits 0-7 and MPSSE bitmode in bits 8-12.
                      The bitmode bits are:
                        0x00   Reset I/O Bit Mode
                        0x01   Asynchronous Bit Bang Mode
                        0x02   Multi-Protocol Synchronous Serial Engine Mode
                        0x04   Synchronous Bit Bang Mode
                        0x08   MCU Host Bus Emulation Mode
                        0x10   Fast Opto-Isolated Serial Interface Mode
                    Index encodes channel (1 = Channel A, 2 = Channel B)

read pins:          usb_control_msg(ftdi->usb_dev, 0xC0, 0x0C,       0,     Index, (char *)&Value, 1, Timeout)
                    Index encodes channel (1 = Channel A, 2 = Channel B)
                    The bits of Value will contain the pin settings.

read eeprom:        usb_control_msg(ftdi->usb_dev, 0xC0, 0x90,       0,         i,   eeprom+(i*2), 2, Timeout)
                    Called for each byte pair read from EEPROM.  Results put in the 128-byte eeprom string.

write eeprom:       usb_control_msg(ftdi->usb_dev, 0x40, 0x91,   Value,         i,          NULL, 0, Timeout)
                    Called for each byte pair to be written to EEPROM address i.
                    Value contains the pair of bytes starting at EEPROM address i.

erase eeprom:       usb_control_msg(ftdi->usb_dev, 0x40, 0x92,       0,         0,          NULL, 0, Timeout)
```

Last updated: February 19, 2007

Contact Craig Van Degrift if you have problems or questions with this web site.