Build the libftdi-1.0 and new ftdi eeprom

From Milkymist Wiki

ftdi library variants

This is the chip support library, you will need it for your tools.

There are currently 3 independent variants. **You need to select one!**

method	comment
D2XX	Binary drivers from [ftdichip.com (http://www.ftdichip.com/Drivers/D2XX.htm)]
libftdi	Git repository at [libftdi (http://developer.intra2net.com/git/?p=libftdi;a=summary)] however this requires **libboost** to compile which pulls in a ton of additional code and is less portable than the method below.
libftdi-1.0	Git repository at [libftdi-1.0 (http://developer.intra2net.com/git/?p=libftdi-1.0;a=summary)] this is the preferred method, discussed in the next steps

1. Build the *libftdi-1.0*

1.1. First, get a clone of the repository :

1.1.1. Create and go to a directory where you want to store the git projects

```
# mkdir ~/git
# cd ~/git
```

1.1.2. Clone the repository

```
# git clone git://developer.intra2net.com/libftdi-1.0
Cloning into libftdi-1.0...
remote: Counting objects: 2008, done.
remote: Compressing objects: 100% (942/942), done.
remote: Total 2008 (delta 1285), reused 1643 (delta 1056)
Receiving objects: 100% (2008/2008), 902.65 KiB | 449 KiB/s, done.
Resolving deltas: 100% (1285/1285), done.
```

It'l create a directory called *libftdi-1.0* with some file in it.

1.2. Prepare to build with the autotools

1.2.1. Go to the directory of *libftdi-1.0*, you need some file

```
# ln -s /usr/share/libtool/config/config.guess
# ln -s /usr/share/libtool/config/config.sub
# ln -s /usr/share/libtool/config/install-sh
# ln -s /usr/share/libtool/config/ltmain.sh
# ln -s /usr/share/libtool/config/missing
# ln -s /usr/share/libtool/config/depcomp
```

1.2.2 Build the auto-conguration

```
# autoscan
# aclocal
# autoheader
# autoconf
# automake
```

1.3.1. Now, you can normally build and install the libftdi

```
# ./configure
...
```

```
# make
...
```

1.3.2. You need to be root for installing the lib

```
(root)# make install
...
```

1.3.3. Quit root, and if there's no error, you can continue to build *ftdi_eeprom*, you maybe needs run

```
# ldconfig.
```

2. Build the *ftdi_eeprom* tool

2.1. Go to the directory *ftdi_eeprom*

```
# cd ftdi_eeprom
```

2.2. Replace it in *libftdi-1.0/ftdi_eeprom/main.c* :

```
/***************************************************************************
                               main.c  -  description
                             -------------------
       begin                : Mon Apr  7 12:05:22 CEST 2003
       copyright            : (C) 2003,2008 by Intra2net AG
       email                : opensource@intra2net.com
```

```
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License version 2 as     *
 *   published by the Free Software Foundation.                            *
 *                                                                         *
 ***************************************************************************/

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <confuse.h>
#include "../src/ftdi.h"
#include "ftdi_eeprom_version.h.in"

int str_to_cbus(char *str, int max_allowed)
{
        #define MAX_OPTION 14
        const char* options[MAX_OPTION] = {
         "TXDEN", "PWREN", "RXLED", "TXLED", "TXRXLED", "SLEEP",
         "CLK48", "CLK24", "CLK12", "CLK6",
         "IO_MODE", "BITBANG_WR", "BITBANG_RD", "SPECIAL"};
        int i;
        max_allowed += 1;
        if (max_allowed > MAX_OPTION) max_allowed = MAX_OPTION;
        for (i=0; i<max_allowed; i++) {
            if (!(strcmp(options[i], str))) {
                return i;
            }
        }
        printf("WARNING: Invalid cbus option '%s'\n", str);
        return 0;
}

int main(int argc, char *argv[])
{
        /*
        configuration options
        */
        cfg_opt_t opts[] =
        {
            CFG_INT("vendor_id", 0, 0),
            CFG_INT("product_id", 0, 0),
            CFG_BOOL("self_powered", cfg_true, 0),
            CFG_BOOL("remote_wakeup", cfg_true, 0),
            CFG_STR_LIST("chip_type", "{AM,BM,2232C,R,2232H,4232H,other}", 0),
            CFG_BOOL("in_is_isochronous", cfg_false, 0),
            CFG_BOOL("out_is_isochronous", cfg_false, 0),
            CFG_BOOL("suspend_pull_downs", cfg_false, 0),
            CFG_BOOL("use_serial", cfg_false, 0),
            CFG_BOOL("change_usb_version", cfg_false, 0),
            CFG_INT("usb_version", 0, 0),
            CFG_INT("max_power", 0, 0),
            CFG_STR("manufacturer", "Acme Inc.", 0),
            CFG_STR("product", "USB Serial Converter", 0),
            CFG_STR("serial", "08-15", 0),
            CFG_BOOL("high_current", cfg_false, 0),
            CFG_STR_LIST("cbus0", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus1", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus2", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus3", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus4", "{TXDEN,PWRON,RXLED,TXLED,TX_RX_LED,SLEEP,CLK48,CLK24,CLK12,CLK6}", 0),
            CFG_BOOL("invert_txd", cfg_false, 0),
            CFG_BOOL("invert_rxd", cfg_false, 0),
```

```
    CFG_BOOL("invert_rts", cfg_false, 0),
    CFG_BOOL("invert_cts", cfg_false, 0),
    CFG_BOOL("invert_dtr", cfg_false, 0),
    CFG_BOOL("invert_dsr", cfg_false, 0),
    CFG_BOOL("invert_dcd", cfg_false, 0),
    CFG_BOOL("invert_ri", cfg_false, 0),
    CFG_END()
};
cfg_t *cfg;

/*
normal variables
*/
unsigned char           _read = 0, _erase = 0, _flash = 0;
int                                     size_check;
int                                     i;
char *                          type;
unsigned int            invert;
FILE *                          fp;
unsigned short          device_vid = 0;
unsigned short          device_pid = 0;
FILE *                          device;
struct ftdi_context     ftdi;
struct ftdi_eeprom *eeprom;

printf("\nFTDI eeprom generator v%s\n", EEPROM_VERSION_STRING);
printf ("(c) Intra2net AG <opensource@intra2net.com>\n");

if (argc < 3)
{
        printf("Syntax sample :\n");
        printf("           %s --read-eeprom  /dev/bus/usb/002/003 ftdi-backup.conf\n", argv[0]);
        printf("           %s --erase-eeprom /dev/bus/usb/002/003\n", argv[0]);
        printf("           %s --flash-eeprom /dev/bus/usb/002/003 ftdi-source.conf\n", argv[0]);
        exit (-1);
}
else
{
        if (strcmp(argv[1], "--read-eeprom") == 0)
            _read = 1;
        else if (strcmp(argv[1], "--erase-eeprom") == 0)
            _erase = 1;
        else if (strcmp(argv[1], "--flash-eeprom") == 0)
            _flash = 1;
}

ftdi_init(&ftdi);
ftdi_eeprom_initdefaults (&ftdi, "Acme Inc.", "FTDI Chip", NULL);
eeprom = ftdi.eeprom;

if ((device = fopen(argv[2], "r")) == NULL)
{
        printf ("Can't open device file\n");
        exit (-1);
}

fseek(device, 8, SEEK_SET);
fread(&device_vid, 1, 2, device);
fread(&device_pid, 1, 2, device);

fclose(device);

printf("\nFound device with VID:PID : 0x%X:0x%X\n", device_vid, device_pid);

i = ftdi_usb_open(&ftdi, device_vid, device_pid);

if (i == 0)
{
    printf("EEPROM size: %d\n", eeprom->size);
}
else
```

```
        {
            printf("Unable to find FTDI devices under given vendor/product id: 0x%X/0x%X\n", device_vid, device_
            printf("Error code: %d (%s)\n", i, ftdi_get_error_string(&ftdi));
            exit (-1);
        }


        if (_read)
        {
            printf("FTDI read eeprom: %d\n", ftdi_read_eeprom(&ftdi));

            ftdi_eeprom_decode(&ftdi, 0);

            const char* chip_types[] = {"other", "", "AM", "", "BM", "2232C", "R", "2232H", "4232H"};

            FILE *fp = fopen (argv[3], "wb");
/*
To implement
            CFG_BOOL("high_current", cfg_false, 0),
            CFG_STR_LIST("cbus0", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus1", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus2", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus3", "{TXDEN,PWREN,RXLED,TXLED,TXRXLED,SLEEP,CLK48,CLK24,CLK12,CLK6,IO_MODE,BITBANG
            CFG_STR_LIST("cbus4", "{TXDEN,PWRON,RXLED,TXLED,TX_RX_LED,SLEEP,CLK48,CLK24,CLK12,CLK6}", 0),
            CFG_BOOL("invert_txd", cfg_false, 0),
            CFG_BOOL("invert_rxd", cfg_false, 0),
            CFG_BOOL("invert_rts", cfg_false, 0),
            CFG_BOOL("invert_cts", cfg_false, 0),
            CFG_BOOL("invert_dtr", cfg_false, 0),
            CFG_BOOL("invert_dsr", cfg_false, 0),
            CFG_BOOL("invert_dcd", cfg_false, 0),
            CFG_BOOL("invert_ri", cfg_false, 0),
*/
                fprintf(fp, "vendor_id=0x%04x\n", eeprom->vendor_id);
                fprintf(fp, "product_id=0x%04x\n", eeprom->product_id);
                fprintf(fp, "self_powered=%s\n", eeprom->self_powered?"true":"false");
                fprintf(fp, "remote_wakeup=%s\n", eeprom->remote_wakeup?"true":"false");
                fprintf(fp, "chip_type=%s\n", chip_types[ftdi.type]);
                fprintf(fp, "max_power=%d\n", eeprom->max_power);
                fprintf(fp, "in_is_isochronous=%s\n", eeprom->in_is_isochronous?"true":"false");
                fprintf(fp, "out_is_isochronous=%s\n", eeprom->out_is_isochronous?"true":"false");
                fprintf(fp, "suspend_pull_downs=%s\n", eeprom->suspend_pull_downs?"true":"false");
                fprintf(fp, "use_serial=%s\n", eeprom->use_serial?"true":"false");
                fprintf(fp, "change_usb_version=%s\n", eeprom->use_usb_version?"true":"false");
                fprintf(fp, "usb_version=%d\n", eeprom->usb_version);
                fprintf(fp, "manufacturer=\"%s\"\n", eeprom->manufacturer);
                fprintf(fp, "product=\"%s\"\n", eeprom->product);
                fprintf(fp, "serial=\"%s\"\n", eeprom->serial);

            fclose (fp);

            goto cleanup;
        }
        else if (_erase)
        {
            printf("FTDI erase eeprom: %d\n", ftdi_erase_eeprom(&ftdi));
        }
        else if (_flash)
        {
                if ((fp = fopen(argv[3], "r")) == NULL)
                {
                        printf ("Can't open configuration file\n");
                        exit (-1);
                }
                fclose (fp);

                cfg = cfg_init(opts, 0);
                cfg_parse(cfg, argv[3]);

                if (cfg_getbool(cfg, "self_powered") && cfg_getint(cfg, "max_power") > 0)
                        printf("Hint: Self powered devices should have a max_power setting of 0.\n");
```

```
                eeprom->vendor_id = cfg_getint(cfg, "vendor_id");
                eeprom->product_id = cfg_getint(cfg, "product_id");

                type = cfg_getstr(cfg, "chip_type");
                if (!strcmp(type, "AM")) {
                        ftdi.type = TYPE_AM;
                } else if (!strcmp(type, "BM")) {
                        ftdi.type = TYPE_BM;
                } else if (!strcmp(type, "2232C")) {
                        ftdi.type = TYPE_2232C;
                } else if (!strcmp(type, "R")) {
                        ftdi.type = TYPE_R;
                } else if (!strcmp(type, "2232H")) {
                        ftdi.type = TYPE_2232H;
                } else if (!strcmp(type, "4232H")) {
                        ftdi.type = TYPE_4232H;
                }

                eeprom->self_powered = cfg_getbool(cfg, "self_powered");
                eeprom->remote_wakeup = cfg_getbool(cfg, "remote_wakeup");
                eeprom->max_power = cfg_getint(cfg, "max_power");
                eeprom->in_is_isochronous = cfg_getbool(cfg, "in_is_isochronous");
                eeprom->out_is_isochronous = cfg_getbool(cfg, "out_is_isochronous");
                eeprom->suspend_pull_downs = cfg_getbool(cfg, "suspend_pull_downs");
                eeprom->use_serial = cfg_getbool(cfg, "use_serial") == 0 ? 0 : USE_SERIAL_NUM;
                eeprom->use_usb_version = cfg_getbool(cfg, "change_usb_version");
                eeprom->usb_version = cfg_getint(cfg, "usb_version");
                eeprom->manufacturer = cfg_getstr(cfg, "manufacturer");
                eeprom->product = cfg_getstr(cfg, "product");
                eeprom->serial = cfg_getstr(cfg, "serial");
                eeprom->high_current = cfg_getbool(cfg, "high_current");
                eeprom->cbus_function[0] = str_to_cbus(cfg_getstr(cfg, "cbus0"), 13);
                eeprom->cbus_function[1] = str_to_cbus(cfg_getstr(cfg, "cbus1"), 13);
                eeprom->cbus_function[2] = str_to_cbus(cfg_getstr(cfg, "cbus2"), 13);
                eeprom->cbus_function[3] = str_to_cbus(cfg_getstr(cfg, "cbus3"), 13);
                eeprom->cbus_function[4] = str_to_cbus(cfg_getstr(cfg, "cbus4"), 9);

                invert = 0;
                if (cfg_getbool(cfg, "invert_rxd")) invert |= INVERT_RXD;
                if (cfg_getbool(cfg, "invert_txd")) invert |= INVERT_TXD;
                if (cfg_getbool(cfg, "invert_rts")) invert |= INVERT_RTS;
                if (cfg_getbool(cfg, "invert_cts")) invert |= INVERT_CTS;
                if (cfg_getbool(cfg, "invert_dtr")) invert |= INVERT_DTR;
                if (cfg_getbool(cfg, "invert_dsr")) invert |= INVERT_DSR;
                if (cfg_getbool(cfg, "invert_dcd")) invert |= INVERT_DCD;
                if (cfg_getbool(cfg, "invert_ri")) invert |= INVERT_RI;
                eeprom->invert = invert;

                size_check = ftdi_eeprom_build(&ftdi);

                if (size_check == -1)
                {
                        printf ("Sorry, the eeprom can only contain 128 bytes (100 bytes for your strings).\n");
                        printf ("You need to short your string by: %d bytes\n", size_check);
                        goto cleanup;
                } else if (size_check < 0) {
                        printf ("ftdi_eeprom_build(): error: %d\n", size_check);
                }
                else
                {
                        printf ("Used eeprom space: %d bytes\n", 128-size_check);
                }

            printf ("FTDI write eeprom: %d\n", ftdi_write_eeprom(&ftdi));

                cfg_free(cfg);
        }

cleanup:
/*
```

```
fp = fopen("eeprom.img", "w");
fwrite(eeprom->buf, 1, 128, fp);
fclose (fp);
*/
        printf("FTDI close: %d\n", ftdi_usb_close(&ftdi));

        ftdi_deinit (&ftdi);

        printf("\n");

        return 0;
}
```

2.3. Compile the source

```
# gcc main.c -I/usr/include/libusb-1.0/ -lconfuse -L/usr/local/lib/ -lftdi -o ftdi_eeprom
```

That's all.

Retrieved from "http://milkymist.org/wiki/index.php?title=Build_the_libftdi-1.0_and_new_ftdi_eeprom"