ノードアダプタの使用方法

© 安田 聖 7M3TJZ

(2008.04.18 V00.05)

本ドキュメントは、現在加筆修正中のものです。また、このドキュメントの元となる PIC プログラムも機能の追加、修正を行っています。使用される場合は、必ず本ドキュメントと PIC プログラムの対応が合っていることを確認の上、ご使用下さい。

本ドキュメントの対応 PIC プログラムのバージョンは、V00.22 です。

変更履歴

2008.04.11	初版
2008.04.12	サンプルプログラム echotest.c を修正しました。
2008.04.13	初期値設定プログラム valueset.c を追加しました。
2008.04.14	山掛けレピータについてを追加しました。
2008.04.15	音声パケットの表示プログラム getbuf.c を追加しました。また、サンプルプログラムの COS の検出ビットの位置が間違っていましたので修正しました
2008.04.16	
2008.04.17	再同期信号の挿入について

本ノードアダプターは、USB インターフェースを使用してデータのやり取りや設定を行います。このためには、USB インターフェースのためのドライバーをインストールします。

１．デバイスドライバーのインストール

・ダウンロード

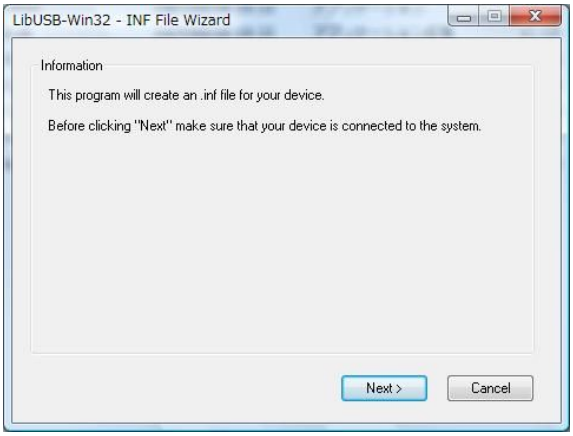
製作したアダプターは、USB のインターフェースを使用してデータのやり取りや、各種の設定を行うようにしてあります。これらのデバイスドライバーとしては、国内の場合、柏野政弘氏の汎用 USB ドライバーを使用されることが多いのですが、このアダプターでは海外で実績がある libusb の win32 版の libusb-win32 を使用しています。このため、本アダプターを使用するには、デバイスドライバーとして libusb-win32 をインストールする必要があります。

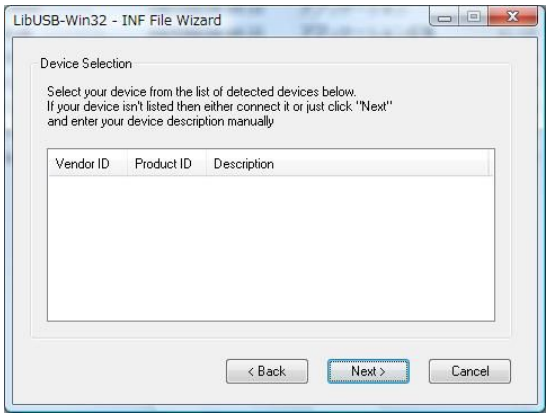
http://libusb-win32.sourceforge.net/

から、libusb-win32-device-bin-0.1.12.1.tar.gz をダウンロードします。

・INF ファイルの作成

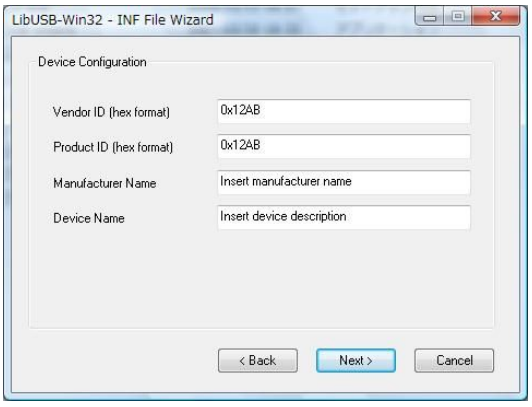
ダウンロードしたファイルを展開後、ディレクトリ bin の下にある、inf-wizard を実行してデバイスドライバーのインストールのための INF ファイルを作成します。Inf-wizard を実行しますと右の図のように表示されますので、Next で先に進めます。



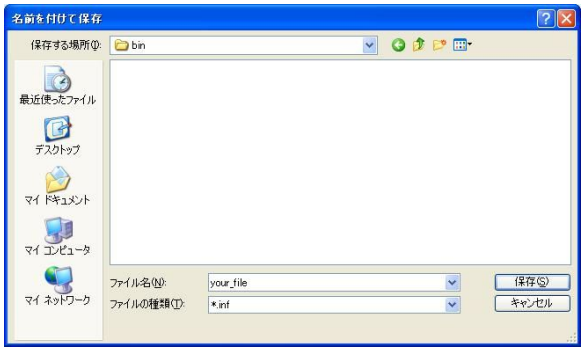


ここで、Vendor ID(hex format)には 04D8、Product ID(hex format)には 0004 そして、Manufacturer Name と Device Name を指定します。指定後 Next で次に移ります。

この画面で、VenderID や ProductID が表示されていることがありますが、その中に Vendor ID に 04D8、Product ID に 0004 のデバイスが表示されている場合は、そのデバイスを、されていない場合は何も指定せず、Next で先に進めます。



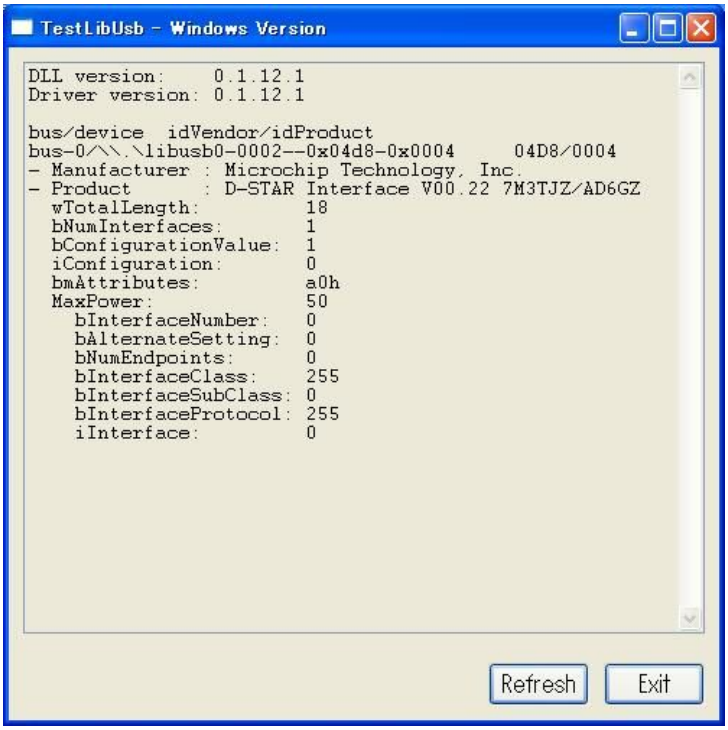
ここで、ファイル名を指定して inf-wizard と同じディレクトリに保存します。これらの一連の作業をしたファイルが、
http://d-star.dyndns.org/program/libusb-win32-device-bin-0.1.12.1.zip
に置いてありますので、このファイルを展開しても使用できます。



・デバイスドライバーのインストール

作成したアダプターを USB ケーブルで PC と接続しますと新しいデバイスのインストールが実行されますので、先に作成した inf を保存したディレクトリを指定してドライバーのインストールを実行します。デバイスマネージャで右側の図のように LibUSB-Win32 Device が表示されれば正常にインストールされています。（右側の図は、d-star.dyndns.org からダウンロードしてインストールした場合です。）

デバイスドライバーが正常にインストールされれば、アダプターが使用できます。
bin の下にある、testlibusb-win.exe を実行しますと次のように表示されます。



また、このダウンロードしたファイルの中には、gcc、bcc、msvc 用のライブラリーファイルも含まれていますのでこれらのCコンパイラーの内から、各自の環境に合ったコンパイラーを使用してください。

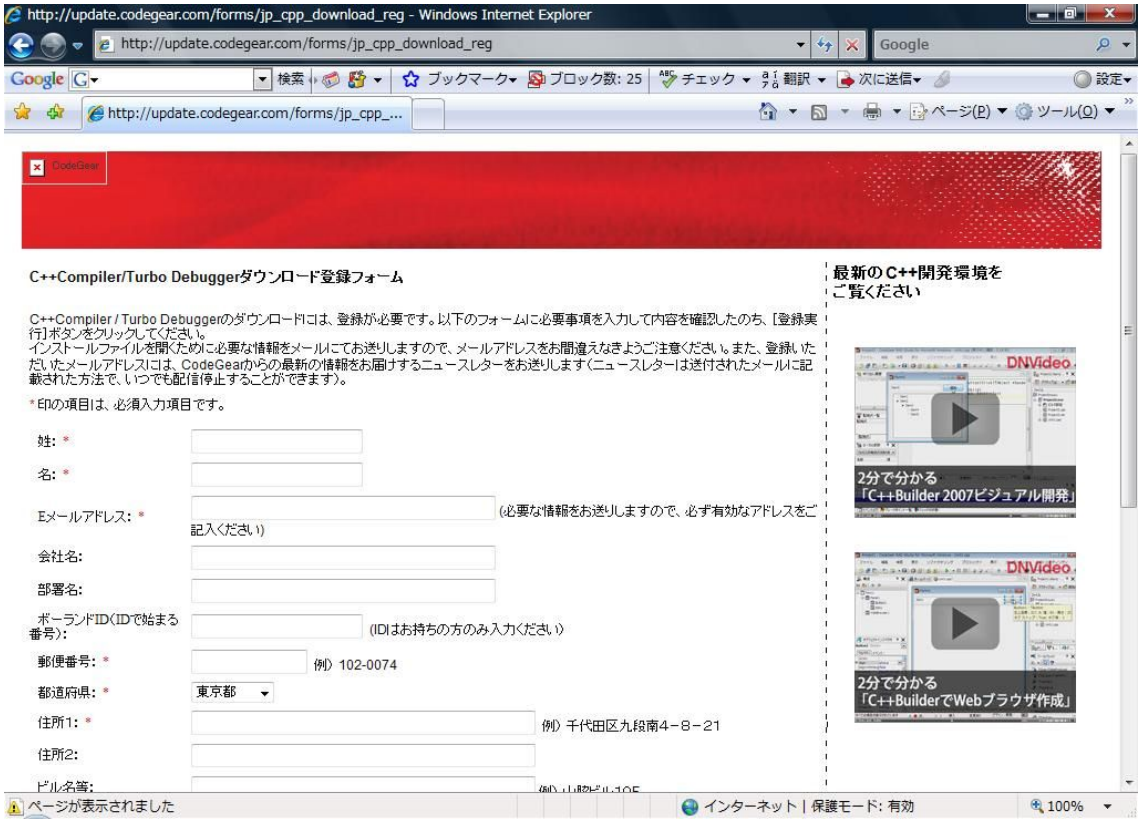
２．コンパイラーの準備

既に gcc、bcc、msvc のどれかの環境が設定されている場合は、libusb-win32 で提供されている include ファイルと lib ファイルを各自の環境に合ったディレクトリにコピーすることで独自のアプリケーションを作成する環境が整います。以下に個人で使用する場合、無料で使用できる Borland 社の bcc を使用する方法を説明します。既に、C コンパイラー環境をお持ちの方は、「３．プログラムの作成」に進んでください。

・bcc のダウンロード

http://www.codegear.com/jp/downloads/free/cppbuilder から C++Compiler/Turbo Debugger を選択します。（下記図で赤色の□で囲った部分）

C++Compiler/Turbo Debugger を選択しますと



が表示されますので、必要な情報を入力の上登録します。この登録が受け付けられると、メールで展開時に必要なパスワードが送られてきます。また、次の画面に移動し、ダウンロードができるようになります。

この画面から、ダウンロードし、ダウンロードしたファイルを展開します。展開時にパスワードを要求されますので、メールで送られてきたパスワードを指定します。展開後、同梱されている ReadmeInstall に従ってインストールを行って下さい。

インストール後、各種設定を行う必要があるのですが、この設置を簡単に行うプログラム、setbcc が vector で公開されています。http://www.vector.co.jp/soft/win95/prog/se149182.html からダウンロードできます。

上記 setbcc を使用してセットアップした場合、ilink32.cfg には、-L"c:¥Borland¥Bcc55¥lib;c:¥Borland¥Bcc55¥lib¥PSDK" の一行がセットされます。Libusb-win32 がインストールされているディレクトリが C:¥libusb-win32-device-bin-0.1.12.1 の

場合、ilink32.cfg の一行目の最後に；C:¥libusb-win32-device-bin-0.1.12.1¥lib¥bcc を次のように追加します。
-L"c:¥Borland¥Bcc55¥lib;c:¥Borland¥Bcc55¥lib¥PSDK;C:¥libusb-win32-device-bin-0.1.12.1¥lib¥bcc”
これで、bcc を使用する準備は完了です。

コンパイル

　これらの一連の設定が終われば、C/C++で作成したアプリケーションのコンパイルができます。ソースプログラムがあるディレクトリで例えば、echotest.c の場合

　bcc32 echotest.c libusb.lib

と指定しますと、コンパイルとリンクが実行され、実行形式のファイルが作成されます。なお、libusb.lib を付けるのを忘れますと、リンク時に未解決シンボルでエラーになりますので、必ず付けてください。

３．プログラムの作成

ドライバーのインストールのためにダウンロードしたlibusb-win32-device-bin-0.1.12.1.tar.gzを展開したディレクトリの中のincludeの下のusb.hとlib¥bccの下のlibusb.libをプログラムを作成するディレクトリにコピーするか、bccがインストールされているディレクトリの中の includeとlibのディレクトリにコピーします。(Borlandの bccを使用する場合の例ですので、他のコンパイラーを使用する場合は、各自の環境に合わせてください。)

以下のプログラム例では、usb.hとlibusb.libをbccの対応するディレクトリにコピーして使用した例です。他の場所にコピーした場合は、対応する箇所を修正してください。

現在、アダプターで使用できるlibusb-win32の関数は、下記の通りです。

```
void usb_init(void);
int usb_find_busses(void);
int usb_find_devices(void);
struct usb_bus *usb_get_busses(void);
usb_dev_handle *usb_open(struct *usb_device dev);
int usb_close(usb_dev_handle *dev);
int usb_set_configuration(usb_dev_handle *dev, int configuration);
int usb_control_msg(usb_dev_handle *dev, int requesttype, int request, int value, int index,
char *bytes, int size, int timeout);
```

- **USBドライバーの初期化**

下記の３つの関数を順次呼び、USBドライバーの初期化を行います。

```
#include <usb.h>
        /* usb.hのコピー位置によっては #include “usb.h” に変更してください */
usb_init();
usb_find_busses();
usb_find_devices();
```

- **USBデバイスの確認**

本アダプターのVendor IDは04D8、Product IDは0004で設定してあります。このため、次のような手順でデバイスが接続されているどうかを確認します。

```
struct usb_bus *bus;
struct usb_device *dev;
usb_dev_handle *udev;
int dev_found;

for (bus = usb_get_busses(); bus && !dev_found; bus = bus->next) {
    for (dev = bus->devices; dev && !dev_found; dev = dev->next) {
        if((dev->descriptor.idVendor == 0x04D8) &&
            (dev->descriptor.idProduct == 0x0004)) {
        dev_found = TRUE;
        udev = usb_open(dev);
```

```
                }
            }
        }
```

上記の例は、接続されているデバイスの中に Vendor ID が 04D8 そして Product ID が 0004 のデバイス（本アダプターです）が見つかれば、その時点でデバイスを open し確認を終了します。

・アダプターの初期設定

アダプターが見つかった場合は、必ず下記関数を呼び出し、アダプターの初期化を行ってください。この関数が呼ばれないと、この関数以外の関数は無視されます。

```
usb_set_configuration (udev, 1);
```

・　アダプターとの情報交換

本アダプターでは、PC との情報のやり取りは

```
int usb_control_msg(usb_dev_handle *dev, /* デバイスハンドラー
                                            usb_open(dev)　の値 */
                    int requesttype,    /* リクエストタイプ　0x40　もしくは 0xC0 */
                    int request,        /* リクエストの内容 */
                    int value,          /* リクエストに対応した値 */
                    int index,          /* インデックスの値 */
                    char *bytes,        /* 受け渡しをする文字列のポインター */
                    int size,           /* 上記の文字列の長さ 最大８バイト*/
                    int timeout);       /* タイムアウト時間　ミリ秒 */
```

で行います。関数が正常に実行された場合は、０もしくは、指定された値が返ります。失敗した場合は、負の値が返ります。

現在サポートされているリクエストの内容を以下に示します。**（これらの内容には、0～255 の値が割り当てられていますが、変更される場合がありますので、最新の定義を node.h で提供していますので、PIC のプログラムのバージョンに対応したファイルを使用してください。）**

以下に示すリクエストの場合は、タイプに 0xC0 をセットし、内容に下記値のいずれかをセット、bytes と size で転送すべき値を指定します。なお、転送できるバイト数は、最大８バイトですので、必要に応じて繰り返し実行します。また、関数の戻り値に、実際に転送したバイト数をセットしてもとります。何も転送する情報がない場合は、この値に０をセットして戻ります。

```
GET_DATA            /* アダプターからデータを受け取ります
                       受け取ったバイト数を関数の戻り値で返します*/
GET_HEADER          /* 無線部ヘッダーの情報を受け取ります */
GET_AD_STATUS       /* アダプターの現在の状態を受け取ります */
```

ステイタスの読み方については、「アダプターのステイタスの読み込み」の項を参照してください。

```
GET_VERSION         /* アダプターのバージョン情報を受け取ります
                       文字列で返しますので、文字列が無くなるまで繰り返し
```

読み取ります*/

以下に示すリクエストの場合は、タイプに 0xC0 をセットし、size に 1 を指定します。要求した情報を bytes の最初のバイトにセットして戻ります。

```
GET_TimeOut                 /* 現在の連続送信時間を読み出す */
GET_DelayTime               /* 現在の遅延時間を読み出す */
GET_KeepAlive               /* 現在の受信時の中断継続時間を読み出す */
GET_RESYNC_ERROR_BITS   /* 現在の再同期ビットパターンの許容ビット数を読み出す */
GET_MODE                    /* 現在の各種 SW の状態を読み出す */
```

以下に示すリクエストの場合は、タイプに 0x40 をセットし、内容に下記値のいずれかをセットし、bytes と size で転送すべき値を指定します。なお、転送できるバイト数は、最大８バイトですので、必要に応じて繰り返し実行します。また、アダプター側で、指定されたバイト数を受け取れない場合は、何もせずエラーで戻ります。（戻り値に負の値がセットされます。）

```
PUT_DATA                    /* アダプターにデータを渡します
                               アダプターのバッファーに指定されたバイト数の余裕が
                               ない場合は、何もせず、エラーを返します*/
SET_MyCALL                  /* 送信の為の MyCall を設定します */
SET_MyCALL2                 /* 送信の為の MyCall2 を設定します */
SET_YourCALL                /* 送信の為の YourCall を設定します */
SET_RPT1CALL                /* 送信の為の RPT1Call を設定します */
SET_RPT2CALL                /* 送信の為の RPT2Call を設定します */
SET_FLAGS                   /* 送信の為のフラグを設定します */
SET_MyRPTCALL               /* 山掛けレピータモードの時のコールサインを
                                                    設定します */
```

下記に、送信時の無線部ヘッダーの情報をセットする例を示します。

```
/* Call Sign set */
        ret = usb_control_msg(udev, 0x40, SET_MyCALL, 0, 0, "7M3TJZ E", 8, 100);
        ret = usb_control_msg(udev, 0x40, SET_MyCALL2, 0, 0, "NODE", 4, 100);
        ret = usb_control_msg(udev, 0x40, SET_YourCALL, 0, 0, mycall, 8, 100);
        ret = usb_control_msg(udev, 0x40, SET_RPT2CALL, 0, 0, rpt1call, 8, 100);
        ret = usb_control_msg(udev, 0x40, SET_RPT1CALL, 0, 0, rpt2call, 8, 100);
         flags[0] = 0x40;     /* set for repeater */
         flags[1] = 0x00;
         flags[2] = 0x00;
        ret = usb_control_msg(udev, 0x40, SET_FLAGS, 0, 0, flags, 3, 100);
```

下記に、読み込んだ音声＋簡易データ部分を受け取り、ファイルに書き込む例を示します。

```
        voice = fopen ("D-STAR.voice","wb");
        while (ret <= 8)
        {
                ret = usb_control_msg(udev, 0x40, GET_DATA, 0, 0, buffer, 8, 100);
```

```
                if (ret != 0)
                {
                        for (i = 0 ; i < ret; i++)
                        {
                                fputc (buffer[i], voice);
                        }
                }
                if (ret == 0)
                {
                 ret = usb_control_msg(udev, 0x40, GET_AD_STATUS, 0, 0, buffer, 1, 100);
                        if (!(buffer[0] & COS_OnOff)) goto PlayBack;
                }
        }
PlayBack:
        fclose (voice);
```

以下のリクエストの場合は、タイプに 0x40 をセットし、内容に下記値のいずれかをセット、value にセットする値をセットします。

```
SET_TimeOut             /*  連続送信時間   x 10 秒  */
SET_DelayTime           /*  送信遅延時間   x 10 ミリ秒  */
SET_KeepAlive           /*  受信時の中断継続時間   x 10 ミリ秒  */
SET_RESYNC_ERROR_BITS       /*  再同期ビットパターンの許容ビット数  */
SET_PTT                 /* On:1   Off:0 */
SET_COS                 /* COS をチェックするかどうかの SW   ON:1   OFF:0 */
SET_CRC_CHECK           /* CRC をチェックするかどうかの SW   ON:1   OFF:0 */
SET_RAW_OUTPUT          /*  受信したビット列をそのまま転送するかどうかの SW
                                                        ON:1   OFF:0 */
SET_LastFrame           /* PTT OFF 時にラストフレームを送信するかどうかの SW
                                                        ON:1   OFF:0 */
```

以下に送信遅延時間をセットする場合の例を示します。

```
/* delay time TxDelay   nn x 10mSec. */
        ret = usb_control_msg(udev, 0x40, SET_DelayTime, 30, 0,buffer, 0, 100);
```

value でセットされた値の 10 ミリ秒倍された時間が、遅延時間となります。なお、設定に失敗しますと ret に 0 以外の値が返ります。

・アダプターのステイタスの読み込み

ret = usb_control_msg(udev, 0x40, GET_AD_STATUS, 0, 0, buffer, 1, 100);

で、buffer[0]にステイタスをセットして返ってきます。Buffer[0]の各ビットのの意味は
０ビット： 無線部ヘッダーを読み込んでいる場合 ON
読み込みが実行されますと OFF になり新規に無線部ヘッダーを読み込むと ON に変更されます
１ビット： COS(carrier operated squelch)のを検出しているとときは ON
２ビット： CRC チェックでエラーの場合 ON

・アダプターの現在の動作モードの読み込み

ret = usb_control_msg(udev, 0x40, GET_MODE, 0, 0, buffer, 1, 100);

で、buffer[0]にアダプターの現在の動作モードの設定値を読み出します。

０ビット　RAW モードで出力の設定
１の場合：読み込んだデータをそのまま PC に渡します
０の場合：無線部ヘッダーの解析を行います
１ビット　CRC チェックを行うかどうかの設定　１：ON　０：OFF
１の場合：ビット同期およびフレーム同期を検出後、無線部のヘッダーの CRC が間違っていると、この音声パケットは、破棄されます。
０の場合：ビット同期およびフレーム同期を検出後、無線部のヘッダーの CRC が間違っていても、無線部ヘッダーそれに続く音声データを読み出します。
２ビット　COS（スケルチ） チェックを行うかどうかの設定　１：ON　０：OFF
１の場合：スケルチに従って音声パケットを取り込みます。また、Keep Alive の時間が設定されていますと、スケルチが閉じても、この時間はスケルチが開いているとして処理されます。
０の場合：スケルチとは無関係に、常に音声パケットを取り込みます
３ビット　LastFrame 送信を行うかどうかの設定　　１：ON　０：OFF
１の場合：PTT OFF のコマンドを受け取った後、音声パケットの最後にラストフレームを自動的に付加します。
０の場合：ラストフレームの付加は行われません
４ビット　簡易データ通信のフィールドにスクランブル処理を行うかどうかの設定
１：ON　０：OFF
１の場合：簡易データフィールドに足して、スクランブル／アンスクランブル処理を行います
０の場合：何もしません

４．ヘッダーファイル

本アダプターのアプリケーションを作成するための C/C++用のヘッダーファイルです。最新版が、http://d-star.dyndns.org/program/node.h と http://d-star.dyndns.org/program/dstar.h からダウンロードできます。

node.h

```
/*   Header file for Node adapter V00.21 or later */
/*                                   Satoshi Yasuda      */


#define   SET_TimeOut                 0x01
#define   SET_DelayTime               0x02
#define   SET_KeepAlive               0x03
#define   SET_RESYNC_ERROR_BITS             0x04
#define   SET_PTT                           0x05
#define   SET_COS                           0x06
#define   SET_CRC_CHECK                     0x07
#define   SET_RAW_OUTPUT                    0x08
#define   SET_LastFrame               0x0a
#define   SET_SD_CONV                 0x0b
#define   PUT_DATA                    0x10
#define   GET_DATA                    0x11
#define   GET_HEADER                  0x21
#define   GET_AD_STATUS                     0x30
#define   SET_MyCALL                  0x40
#define   SET_MyCALL2                 0x41
#define   SET_YourCALL                0x42
#define   SET_RPT1CALL                0x43
#define   SET_RPT2CALL                0x44
#define   SET_FLAGS                   0x45
#define   SET_MyRPTCALL                     0x46
#define   GET_TimeOut                 0x51
#define   GET_DelayTime               0x52
#define   GET_KeepAlive               0x53
#define   GET_RESYNC_ERROR_BITS             0x54
#define   GET_MyRPTCALL                     0x55
#define   GET_MODE                    0x56
#define   GET_VERSION                 0xFF


#define   ON        1
#define   OFF       0
```

```
/* Mode definition */
#define   RAW_SW                0x01
#define   CRC_SW                0x02
#define   COS_SW                0x04
#define   LastFrame_SW   0x08
#define   SD_CONV_SW     0x10


/* AD_STATUS definition    */
#define   READ_RF_HEADER        0x01
#define   COS_OnOff       0x02
#define   CRC_ERROR       0x04
```

dstar.h

```
/*   Header file for D-STAR  */
/*                    Satoshi Yasuda     */
/*                    7m3tjz/ad6gz       */

struct     dv_header{
           unsigned char      flags[3];
           unsigned char      RPT2Call[8];
           unsigned char      RPT1Call[8];
           unsigned char      YourCall[8];
           unsigned char      MyCall[8];
           unsigned char      MyCall2[4];
           unsigned char      CRC[2];
};


struct     back_bone{
           unsigned char      id;
           unsigned char      dest_repeater_id;
           unsigned char      send_repeater_id;
           unsigned char      send_terminal_id;
           unsigned char      com_id_high;
           unsigned char      com_id_low;
           unsigned char      control;
};

struct     inet_header{
           unsigned char      id[4];
           unsigned char      flags[2];
           unsigned char      reserve[2];
           struct     back_bone bk_bone_flags;
};

struct     voice_header{
           struct     inet_header inet_fg;
           struct     dv_header rf_header;
};

struct     voice_data{
           struct     inet_header inet_fg;
           unsigned char      voice_segment[9];
           unsigned char      data_segment[3];
```

```
};
```

５．エコーサーバーのプログラム例

本プログラムでは、受信した情報（音声＋簡易データ）を PC のハードディスクに書き込んでいます。このときのファイル名の拡張子に txt にしますと、情報の中に含まれています 0x0d が落ちますので、拡張子を .txt 以外にしてください。下記例では、.voice にしてあります。

```
#include <stdio.h>
#include <usb.h>
#include <fcntl.h>
#include "node.h"
#include "dstar.h"

#define FALSE     0
#define TRUE      1

int       CallSignPlay(usb_dev_handle *udev, unsigned char CallSign[], int resync);
int       ReplyMsgPlay(usb_dev_handle *udev, int resync);
int       NullVoicePlay(usb_dev_handle *udev, int resync);

void main(void) {
        struct usb_bus *bus;
        struct usb_device *dev;
        usb_dev_handle *udev;
        struct dv_header *dv;
        int dev_found, ret;
        char buffer[8];
        char radio_header[41];
        char flags[3];
        FILE *voice;

        int       resync;

        int i,k;
        int       HeaderLength;

        usb_init();
        usb_find_busses();
        usb_find_devices();


        resync = 0;
        udev = NULL;
        dev_found = FALSE;
        for (bus = usb_get_busses(); bus && !dev_found; bus = bus->next) {
```

```
        for (dev = bus->devices; dev && !dev_found; dev = dev->next) {
                if ((dev->descriptor.idVendor == 0x04D8) && (dev->descriptor.idProduct ==
0x0004)) {
                dev_found = TRUE;
                udev = usb_open(dev);
                }
        }
        }

        if (!dev_found) {
                printf("No matching device found...¥n");
        }

        usb_set_configuration (udev, 1);


/* PIC VERSION READ */
        ret = 8;
        while (ret == 8)
        {
        ret = usb_control_msg(udev, 0xC0, GET_VERSION, 0, 0, buffer, 8, 100);
                for (i = 0 ; i < ret ; i++)
                {
                        printf ("%c",buffer[i]);
                }
        }
        printf ("¥n");


    while (1)
    {
        k = 0;
        HeaderLength = 0;
        while (HeaderLength < 41)
        {
                ret = usb_control_msg(udev, 0xC0, GET_HEADER, 0, 0, buffer, 8, 100);
                if (ret > 0)
                {
                        HeaderLength += ret;
                        for (i = 0 ; i < ret; i++)
                        {
                                radio_header[k] = buffer[i];
                                k++;
                        }
                }
```

```
	}

	dv = (struct dv_header *) radio_header;

	printf ("¥n");

	printf ("Flags    : %2.2x %2.2x %2.2x¥n",dv->flags[0],dv->flags[1],dv->flags[2]);
	printf ("RPT2     : %.8s¥n",dv->RPT2Call);
	printf ("RPT1     : %.8s¥n",dv->RPT1Call);
	printf ("YuCall : %.8s¥n",dv->YourCall);
	printf ("MyCall : %.8s¥n",dv->MyCall);
	printf ("MyCall2: %.4s¥n",dv->MyCall2);
	printf ("CRC       : %2.2x %2.2x¥n",dv->CRC[0],dv->CRC[1]);


	voice = fopen ("D-STAR.voice","wb");

	while (ret <= 8)
	{

	ret = usb_control_msg(udev, 0xC0, GET_DATA, 0, 0, buffer, 8, 100);
		if (ret != 0)
		{
			for (i = 0 ; i < ret; i++)
			{
				fputc (buffer[i], voice);
			}
		}
		if (ret == 0)
		{
		ret = usb_control_msg(udev, 0xC0, GET_AD_STATUS, 0, 0, buffer, 1, 100);
			if (!(buffer[0] & COS_OnOff)) break;
		}
	}

	fclose (voice);

	voice = fopen ("D-STAR.voice","rb");

	/* Call Sign set */
	usb_control_msg(udev, 0x40, SET_MyCALL, 0, 0, "7M3TJZ E", 8, 100);	/*  change
your callsign */
	usb_control_msg(udev, 0x40, SET_MyCALL2, 0, 0, "NODE", 4, 100);
	usb_control_msg(udev, 0x40, SET_YourCALL, 0, 0, dv->MyCall, 8, 100);
	usb_control_msg(udev, 0x40, SET_RPT2CALL, 0, 0, dv->RPT1Call, 8, 100);
```

```
usb_control_msg(udev, 0x40, SET_RPT1CALL, 0, 0, dv->RPT2Call, 8, 100);
 flags[0] = 0x40;      /* set for repeater */
 flags[1] = 0x00;
 flags[2] = 0x00;
usb_control_msg(udev, 0x40, SET_FLAGS, 0, 0, flags, 3, 100);


 /* delay time TxDelay    nn x 10mSec. */
usb_control_msg(udev, 0x40, SET_DelayTime, 30, 0,buffer, 0, 100);

 /* Check COS */
 usb_control_msg(udev, 0xC0, GET_AD_STATUS, 0, 0, buffer, 1, 100);
 while (buffer[0] & COS_OnOff)                    /* until COS OFF */
 {
              usb_control_msg(udev, 0xC0, GET_AD_STATUS, 0, 0, buffer, 1, 100);
 }


 /* PTT ON */
usb_control_msg(udev, 0x40, SET_PTT, ON, 0, buffer, 0, 100);

 /* Message send your call */
 resync = CallSignPlay (udev, dv->MyCall, resync);

 /* Reply Message send */
 resync = ReplyMsgPlay (udev, resync);

 /* Reply Message send */                  /* padding for Null Voice data */
 resync = NullVoicePlay (udev, resync);

 /* Send Voice & Data STREAM */
 k = 0;
 while ((i = fgetc (voice)) != EOF)
 {
              buffer[k] = i;
              k++;
              if (k == 8)
              {
                            k = 0;
                           ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buffer, 8, 100);
                            while (ret < 0)
                            {
                            ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buffer, 8, 100);
                            }
              }
```

```
		}

		if (k > 0)
		{
				ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buffer, k, 100);
				while (ret < 0)
				{
				ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buffer, k, 100);
				}

		}

		/* PTT OFF */
		ret = usb_control_msg(udev, 0x40, SET_PTT, OFF, 0, buffer, 0, 100);

		fclose (voice);

	}		/* end while (1) */

/*
		usb_close(udev);
*/
}

int		CallSignPlay (usb_dev_handle *udev, unsigned char CallSign[], int resync)
{
		FILE	*wave;
		char	file_name[] = {"audio¥¥ .bin"};
		char	buff[12];
		int	i,k,l,ret;

		for (i = 0 ; i < 6 ; i++)
		{
			if (CallSign[i] != 0x20)
			{
				file_name[6] = CallSign[i];
				wave = fopen (file_name,"rb");
				l = 0;
				while ((k = fgetc (wave)) != EOF)
				{
					buff[l] = k;
					l++;
					if (l == 12)
					{
						l = 0;
```

```
                                        if (resync == 0)
                                        {
                                                buff[ 9] = 0xaa;
                                                buff[10] = 0xb4;
                                                buff[11] = 0x68;
                                        } else {
                                                buff[ 9] = 0x0e;
                                                buff[10] = 0xf2;
                                                buff[11] = 0xc9;
                                        }
                                        resync++;
                                        if (resync == 21) resync = 0;

                                        ret = usb_control_msg(udev, 0x40, PUT_DATA,
0, 0, buff, 8, 100);

                                        while (ret < 0)
                                        {
                                        ret = usb_control_msg(udev, 0x40, PUT_DATA,
0, 0, buff, 8, 100);
                                        }

                                        ret = usb_control_msg(udev, 0x40, PUT_DATA,
0, 0, &buff[8], 4, 100);
                                        while (ret < 0)
                                        {
                                        ret = usb_control_msg(udev, 0x40, PUT_DATA,
0, 0, &buff[8], 4, 100);
                                        }
                                                }
                                        }
                                        fclose (wave);
                                }
                        }
                        return resync;

}

int     ReplyMsgPlay (usb_dev_handle *udev, int resync)
{
        FILE    *reply;
        char    file_name[] = {"audio¥¥replymsg.bin"};
        char    buff[12];
        int     k,l,ret;
```

```
	reply = fopen (file_name,"rb");
	l = 0;

	while ((k = fgetc (reply)) != EOF)
	{
		buff[l] = k;
		l++;
		if (l == 12)
		{
			 l = 0;
			if (resync == 0)
			{
				buff[ 9] = 0xaa;
				buff[10] = 0xb4;
				buff[11] = 0x68;
			} else {
				buff[ 9] = 0x0e;
				buff[10] = 0xf2;
				buff[11] = 0xc9;
			}
			resync++;
			if (resync == 21) resync = 0;
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buff, 8, 100);
			while (ret < 0)
			{
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, buff, 8, 100);
			}
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, &buff[8], 4,
100);
			while (ret < 0)
			{
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, &buff[8], 4,
100);
			}
		}
	}

	fclose (reply);

	return resync;
}

int	NullVoicePlay (usb_dev_handle *udev, int resync)
{
	int	ret;
```

```
	char	NullVoice[12]	=
{0x79,0xb1,0x4c,0x11,0x64,0x58,0xfc,0x88,0x17,0x0e,0xf2,0xc9};

	while (resync != 0)
	{
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, NullVoice, 8,
100);
			while (ret < 0)
			{
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, NullVoice, 8,
100);
			}
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, &NullVoice[8],
4, 100);
			while (ret < 0)
			{
			ret = usb_control_msg(udev, 0x40, PUT_DATA, 0, 0, &NullVoice[8],
4, 100);
			}
			resync++;
			if (resync == 21) resync = 0;
	}

	return resync;
}
```

6.受信パケットの表示プログラム

受信したパケットの無線部ヘッダーを解析し、その結果を表示すると共に、無線部ヘッダーに引き続き送られてくる音声＋簡易データ部分（合計１２バイトを単位として）の情報を編集して 16 進数で表示するプログラムです。簡易データ部分については、アンスクランブル処理をした値について、16 進数の表示とキャラクターによる表示も行っています。

```
#include <stdio.h>
#include <usb.h>
#include <fcntl.h>
#include "node.h"


#define FALSE     0
#define TRUE      1

void        main(void) {
            struct usb_bus *bus;
            struct usb_device *dev;
            struct dv_header *dv;
            usb_dev_handle *udev;
            int dev_found, ret;
            char string[256];
            char buffer[8];
            char radio_header[41];
            char c;
            char SlowData[3];
            int i,k,l,n;
            int         HeaderLength;

            FILE *voice;



            usb_init();

            usb_find_busses();
            usb_find_devices();

            udev = NULL;
            dev_found = FALSE;
            for (bus = usb_get_busses(); bus && !dev_found; bus = bus->next) {
            for (dev = bus->devices; dev && !dev_found; dev = dev->next) {
                        if ((dev->descriptor.idVendor == 0x04D8) && (dev->descriptor.idProduct ==
```

```
0x0004)) {
                dev_found = TRUE;
                  udev = usb_open(dev);
                }
        }
        }
        if (!dev_found) {
        printf("No matching device found...¥n");
        }


        usb_set_configuration (udev, 1);


/* PIC VERSION READ */
        ret = 8;
        while (ret == 8)
        {
        ret = usb_control_msg(udev, 0xC0, GET_VERSION, 0, 0, buffer, 8, 100);
                for (i = 0 ; i < ret ; i++)
                {
                        printf ("%c",buffer[i]);
                }
        }
        printf ("¥n");
while (1)
{

        k = 0;
        HeaderLength = 0;

        while (HeaderLength < 41)
        {
        ret = usb_control_msg(udev, 0xC0, GET_HEADER, 0, 0, buffer, 8, 100);
                if (ret < 0) {
                printf("Unable to send vendor request, ret = %d...¥n", ret);
        } else
                {
                        if (ret != 0)
                        {
                                HeaderLength += ret;
                                for (i = 0 ; i < ret; i++)
                                {
                                        radio_header[k] = buffer[i];
                                        k++;
```

```
                                        }
                                }
                        }
                }

                dv = (struct dv_header *) radio_header;

                printf ("¥n¥n");

                printf ("Flags    : %2.2x %2.2x %2.2x¥n",dv->flags[0],dv->flags[1],dv->flags[2]);
                printf ("RPT2     : %.8s¥n",dv->RPT2Call);
                printf ("RPT1     : %.8s¥n",dv->RPT1Call);
                printf ("YuCall : %.8s¥n",dv->YourCall);
                printf ("MyCall : %.8s¥n",dv->MyCall);
                printf ("MyCall2: %.4s¥n",dv->MyCall2);
                printf ("CRC      : %2.2x %2.2x¥n",dv->CRC[0],dv->CRC[1]);

                printf ("¥nDV STREAM ¥n");

                k = 0;
                l = 0;
                while (ret <= 8)
                {

                ret = usb_control_msg(udev, 0xC0, GET_DATA, 0, 0, buffer, 8, 100);
                        if (ret < 0) {
                      printf("Unable to send vendor request, ret = %d...¥n", ret);
                        } else if (ret > 0)
                        {
                                for (i = 0 ; i < ret; i++)
                                {
                                                printf ("%2.2x ",buffer[i] & 0xff);
                                                if (k == 11)   SlowData[2] = (buffer[i] & 0xff) ^
0xc9;;
                                                if (k == 10)   SlowData[1] = (buffer[i] & 0xff) ^
0xf2;
                                                if (k == 9)    SlowData[0] = (buffer[i] & 0xff) ^
0x0e;
                                                k++;

                                        if (k == 12)
                                        {

                                                c = SlowData[0];
                                                SlowData[0] = 0x00;
```

```
                                                for (n = 0 ; n < 8 ; n++)
                                                {
                                                if (c & ( 0x01 << n)) SlowData[0] |= (0x80 >> n);
                                                }
                                                c = SlowData[1];
                                                SlowData[1] = 0x00;
                                                for (n = 0 ; n < 8 ; n++)
                                                {
                                                        if (c & ( 0x01 << n)) SlowData[1] |=
(0x80 >> n);
                                                }
                                                c = SlowData[2];
                                                SlowData[2] = 0x00;
                                                for (n = 0 ; n < 8 ; n++)
                                                {
                                                if (c & ( 0x01 << n)) SlowData[2] |= (0x80 >> n);
                                                }
                                                printf ("      %2.2x %2.2x %2.2x %c %c %c
¥n",SlowData[0] & 0xff, SlowData[1] & 0xff, SlowData[2] & 0xff, SlowData[0], SlowData[1],
SlowData[2]);
                                                k = 0;
                                                l++;
                                        }
                                        if (l == 21)
                                        {
                                                printf ("¥n");
                                                l = 0;
                                        }
                                }
                        } else if (ret == 0)
                        {
                        ret = usb_control_msg(udev, 0xC0, GET_AD_STATUS, 0, 0, buffer, 1, 100);
                        if (!(buffer[0] & COS_OnOff)) break;                /* check COS_OFF */
                        }
                }
}                       /* end while (1) */
/*
        usb_close(udev);
*/
}
```

7. EEPROM の初期値の変更プログラム

PIC18F2550 の eeprom エリアに連続送信のタイムアウト時間、送信遅延時間、再同期信号を検出する場合の不一致ビット数、処理に関するスイッチ、山掛けレピータとして使用する場合のレピータのコールサインを記憶しています。これらの値を変更するプログラムの例を下記に示しておきますので参考にしてください。

```
#include <stdio.h>
#include <stdlib.h>
#include <usb.h>
#include "node.h"
#include "dstar.h"

int onoff(void);

void main() {

    struct usb_bus *bus;
    struct usb_device *dev;
    usb_dev_handle *udev;
    int dev_found, ret, i, k;
    unsigned char buffer[8];
    char string[10];

    usb_init();

    usb_find_busses();
    usb_find_devices();

    udev = NULL;
    dev_found = FALSE;
    for (bus = usb_get_busses(); bus && !dev_found; bus = bus->next) {
        for (dev = bus->devices; dev && !dev_found; dev = dev->next) {
            if ((dev->descriptor.idVendor == 0x04D8) && (dev->descriptor.idProduct ==
0x0004)) {
                dev_found = TRUE;
                udev = usb_open(dev);
            }
        }
    }


    if (!dev_found) {
        printf("No matching device found...¥n");
    }
```

```
        usb_set_configuration (udev,1);

/* PIC VERSION READ */
        ret = 8;
        while (ret == 8)
        {
        ret = usb_control_msg(udev, 0xC0, GET_VERSION, 0, 0, buffer, 8, 100);
                for (i = 0 ; i < ret ; i++)
                {
                        printf ("%c",buffer[i]);
                }
        }
        printf ("¥n¥n");


        i = 99;
        while (i != 0)
        {

                usb_control_msg(udev, 0xC0, GET_TimeOut, 0, 0, buffer, 1, 100);
                 printf ("Time Out               : %3d Sec.¥n",buffer[0]*10);

                usb_control_msg(udev, 0xC0, GET_DelayTime, 0, 0, buffer, 1, 100);
                 printf ("Delay Time             : %3d mSec.¥n",buffer[0]*10);

                usb_control_msg(udev, 0xC0, GET_KeepAlive, 0, 0, buffer, 1, 100);
                 printf ("Keep Alive             : %3d mSec.¥n",buffer[0]);

                usb_control_msg(udev, 0xC0, GET_RESYNC_ERROR_BITS, 0, 0, buffer, 1,
100);
                 printf ("ReSync Error Bits : %3d¥n",buffer[0]);

                usb_control_msg(udev, 0xC0, GET_MODE, 0, 0, buffer, 1, 100);
                 printf ("MODE RAW OUTPUT    :   ");
                 if (buffer[0] & 0x01)
                 {
                        printf ("ON¥n");
                 } else {
                        printf ("OFF¥n");
                 }

                 printf ("          CRC Check    :   ");
                 if (buffer[0] & 0x02)
                 {
```

```
                printf ("ON¥n");
} else {
                printf ("OFF¥n");
}

printf ("          COS Check     :   ");
if (buffer[0] & 0x04)
{
                printf ("ON¥n");
} else {
                printf ("OFF¥n");
}

printf ("    Last Frame Send :   ");
if (buffer[0] & 0x08)
{
                printf ("ON¥n");
} else {
                printf ("OFF¥n");
}

printf ("    Slow Data Conv. :   ");
if (buffer[0] & 0x10)
{
                printf ("ON¥n");
} else {
                printf ("OFF¥n");
}



usb_control_msg(udev, 0xC0, GET_MyRPTCALL, 0, 0, buffer, 8, 100);
printf ("My Repeater Call   : %.8s¥n¥n",buffer);

printf ("Time Out                : 1¥n");
printf ("Delay Time              : 2¥n");
printf ("Keep Alive              : 3¥n");
printf ("ReSync Error Bits : 4¥n");
printf ("Mode Set                : 5¥n");
printf ("My Repeater Call   : 6¥n");
printf ("Exit                    : 0¥n");

printf ("¥nEnter Select Number : ");

gets(string);
```

```
i = atoi(string);
switch (i)
{
        case 0:
                break;

        case 1:         /* 1 Time Out */
                printf ("Enter New Time Out Value : ");
                gets(string);
                k = atoi(string);
                usb_control_msg(udev, 0x40, SET_TimeOut, k/10, 1, buffer,
0, 100);
                break;
        case 2:         /* 2 Delay Time */
                printf ("Enter New Delay Time Value : ");
                gets(string);
                k = atoi(string);
                usb_control_msg(udev, 0x40, SET_DelayTime, k/10, 1,
buffer, 0, 100);
                break;
        case 3:         /* 3 Keep Alive */
                printf ("Enter New Keep Alive Value : ");
                gets(string);
                k = atoi(string);
                usb_control_msg(udev, 0x40, SET_KeepAlive, k, 1, buffer,
0, 100);
                break;
        case 4:         /* 4 ReSync Error Bits */
                printf ("Enter New ReSync Error Bits Value : ");
                gets(string);
                k = atoi(string);
                usb_control_msg(udev,                                   0x40,
SET_RESYNC_ERROR_BITS, k, 1, buffer, 0, 100);
                break;
        case 5:         /* 5 Mode Set */
                printf ("¥n¥n¥n");
                printf ("RAW OutPut       : 1¥n");
                printf ("CRC Check        : 2¥n");
                printf ("COS Check        : 3¥n");
                printf ("Last Frame Send : 4¥n");
                printf ("Slow data Conv. : 5¥n");
                printf ("Return           : 0¥n");

                printf ("¥nEnter Select Number : ");
```

```
                gets(string);
                k = atoi(string);
                switch (k)
                {
                        case 1:
                                if (onoff())
                                {
                                        usb_control_msg(udev,   0x40,
SET_RAW_OUTPUT, ON, 1, buffer, 0, 100);
                                } else {
                                        usb_control_msg(udev,   0x40,
SET_RAW_OUTPUT, OFF, 1, buffer, 0, 100);
                                }
                                break;
                        case 2:
                                if (onoff())
                                {
                                        usb_control_msg(udev,   0x40,
SET_CRC_CHECK, ON, 1, buffer, 0, 100);
                                } else {
                                        usb_control_msg(udev,   0x40,
SET_CRC_CHECK, OFF, 1, buffer, 0, 100);
                                }
                                break;
                        case 3:
                                if (onoff())
                                {
                                        usb_control_msg(udev,   0x40,
SET_COS, ON, 1, buffer, 0, 100);
                                } else {
                                        usb_control_msg(udev,   0x40,
SET_COS, OFF, 1, buffer, 0, 100);
                                }
                                break;
                        case 4:
                                if (onoff())
                                {
                                        usb_control_msg(udev,   0x40,
SET_LastFrame, ON, 1, buffer, 0, 100);
                                } else {
                                        usb_control_msg(udev,   0x40,
SET_LastFrame, OFF, 1, buffer, 0, 100);
                                }
                                break;
                        case 5:
```

```
                                        if (onoff())
                                        {
                                                usb_control_msg(udev,   0x40,
SET_SD_CONV, ON, 1, buffer, 0, 100);
                                        } else {
                                                usb_control_msg(udev,   0x40,
SET_SD_CONV, OFF, 1, buffer, 0, 100);
                                        }
                                        break;
                                case 0:
                                        break;
                        }

                        break;
                case 6:         /* 6 My repeater Call */
                        printf ("Enter New My Repeater Call : ");
                        gets(string);
                        while ((k = strlen(string)) < 8)
                        {
                                string[k] = 0x20;
                                string[k+1] = 0x00;
                        }
                        for (k = 0 ; k < 8 ; k++)
                        {
                                string[k] = toupper(string[k]);
                        }
                       usb_control_msg(udev,   0x40,   SET_MyRPTCALL,   0,   0,
string, 8, 100);
                        break;
                default:
                        break;
        }
        printf ("¥n¥n");
    }

    usb_close(udev);

}

int     onoff (void)
{
    char string[10];
    int i,k;
    printf (" Enter ON/OFF : ");
    gets (string);
```

```
        k = strlen(string);
        for (i = 0 ; i < k ; i++) string[i] = toupper(string[i]);
        if (strcmp (string,"ON")) return 0;
        else return 1;
}
```

8. 山掛けレピータ

本アダプターは、PC のアプリケーションによっては、現在運用されていますゲートウエイと同等なレピータを作成することができますが、ここでは、「山掛けレピータ」の構成方法について説明することにします。

レピータを構成する場合、

1．送受信が同時（少し遅延して送信されます）に、実行できなければ機能しません。本アダプターでは、使用している GMSK 用の IC が送受信同時に使用できることから、送受信同時に使用できるように作成されています。
2．レピータは、自分自身が呼ばれたのかどうかを判定し、自分自身が呼ばれているのであれば、無線部ヘッダーのレピータ使用のフラグをクリアーした後、再送信することになります。このため、無線部ヘッダーの情報（660 ビット）を受信した後、この内容を解析し、解析結果に基づき、送信の為の無線部ヘッダーを構成する必要があります。
3．音声＋簡易データの情報は、無線部ヘッダーに引き続き送られてくるため、無線部ヘッダーの解析、再構成が終わるまで一時的に蓄積する必要があります。通常これらに必要な時間は、100mSec.から 300mSec.です。

の機能が必要です。PC を使用すれば、これらの機能を容易に実現できるのですが、山掛けレピータを実現するためだけに PC を用意するのは、大げさすぎますので、アダプターにオプションを装着することで、この機能を実現できるようにしました。なお、オプションで遅延できる最大の長さは、1．25 秒です。オプションボードをメインボードに装着し、電源を入れ直せば、自動的にレピータモードになります。

なお、必要な遅延時間は使用する無線機で異なりますので、送信した音声パケットが、ダウンリンクの受信機側で無線部ヘッダーが正常に表示される時間に設定してください。

なお、アダプターと無線機（受信、送信用各一台）を繋ぐ場合は、アダプターのパケット（データ）端子を、受信と送信を分離して接続してください。

９．再同期信号

D-STAR では、QRM や QSB で受信している信号の同期が外れた場合の対策として、簡易データのフィールドを利用して再同期信号を挿入しています。無線部ヘッダーに引き続き送信されてくる音声データの最初の簡易データフィールドに再同期信号を挿入し、その後 21 個目毎に再同期信号を送信してきます。この手順が守られていないと、現在の ICOM の D-STAR 対応無線機では、正規のタイミングでない再同期信号を受信した時点で（約１秒ほど遅れます）、ピーという音が出ます。このため、送信側では必ず 21 個目毎に再同期信号（16 進数で aa b4 68）を挿入して下さい。