2010年度送信機Ver2

呉高専電子情報工学科4年:中谷　寿洋

概要

2010年度送信機Ver2は2010年度全国大会終了後に開発された高専ロボコン用無線コントローラです。
実績:大会での実績はなし(現在部内ロボコンでテスト中)

外見

特徴

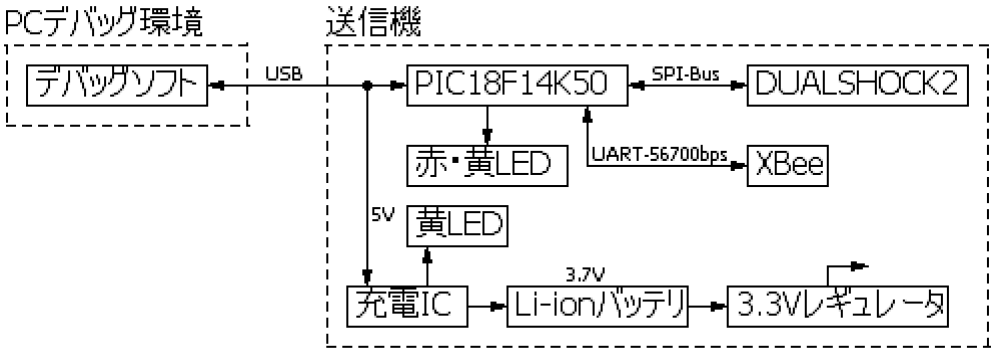
1. デュアルショック2を採用
プレステーション2用コントローラ、デュアルショック2を採用することにより、安価に高信頼性と多機能性を確保しています。また、デュアルショック2の機能をフルに活用しており、START,SELECT,R3,R3を除いたすべてのボタンで感圧機能を使うことができます。
(感圧機能:ボタンを押した力を256段階で送る機能)
2. 無線通信モジュールにXBeeを採用
高専ロボコンで多くのシェアを得ているDigi社のXBeeSeries1を採用しています。
通信距離、通信速度においては無線LANに対して若干不利ですが、安価かつ扱いやすいので高専ロボコン用の無線モジュールとしては十分な性能です。
3. 携帯用リチウムイオン電池を採用
携帯用リチウムイオン電池には過充電、過放電防止回路が内臓されています。そのため、万一回路がショートしたとしても高い安全性を期待することができます。
4. PIC18F14K50採用
USB対応マイコンであるPIC18F14K50を制御マイコンに採用し、パソコンからシリアルポートとして認識されます。この機能を利用してコントローラからXBeeを取り外さずにチャンネルの変更等を行うことができます。設定にはDigi社の設定ツールX-CTUを使うことができます。
5. 無線書き込み機能＆無線デバッグ機能
上記の機能を応用して、パソコンから無線でロボット側PICを書き換えることができます。
また、コントローラとして使いつつ、デバックすることもできるため開発効率を非常に高めることができます。
6. オマケ
送信機のプログラムを書き換えれば感圧ボタン対応のUSBゲームパッドとしてもご利用頂けます。
(赤字:大きな改良点,青字:小規模or前作からの引き継ぎ)

仕様

仕様	説明	備考
制御マイコン	PIC18F14K50	USB対応品,3.3V動作対応
電波帯,占有帯域	2.4GHz帯,5MHz	
通信規格	IEEE802.15.4	ZigBeeでも用いられている規格
通信モジュール	XB24-ACI-001	XBeeSeries1チップアンテナ型
	XBP24-ACI-001J	XBeeProSeries1チップアンテナ型

チャンネル(16進数)	11ch～26ch(0x0B～0x1A)	Series1
	12ch～23ch(0x0C～0x17)	Series1PRO,使用可能なチャンネルが少ない
出力	1mW	
	10mW	本来は60mWだが電波法の関係で10mWに制限される
暗号化	128-bit AES	未検証であるため、利用する際には要相談
通信距離	屋内30m屋外100m	環境によって大きく変わるので参考程度に
コントローラ	SONY(SCPH-10010)	デュアルショック2
ボタン数(感圧対応)	16ボタン(12ボタン)	すべてのボタンを使うことができます
アナログスティック	2本	誤差が大きいのでプログラム時は注意してください
送信周期	毎秒約60回	
バッテリー	PMBAT1(Li-ion,920mAh)	パナソニック製携帯用バッテリー
電圧	3.7V	
寿命	15時間程度	要検証
充電時間	2時間程度	要検証

ブロック図



LED の割り当

LED	説明	点灯時	消灯時
5mm黄LED1	パッドとの接続確認	接続	未接続
5mm黄LED2	充電確認	充電中	充電完了
5mm赤LED	USB接続確認	接続	未接続
3mm赤LED	電源確認	電源ON	電源OFF

データフレーム

Byte＼Bit	7	6	5	4	3	2	1	0	説明	
1Byte目	1	1	1	1	0	0	0	0	先頭識別子	
2Byte目	1	0	1	0	1	0	1	0	接続識別子	パッド接続時
	0	1	0	1	0	1	0	1		パッド未接続時
3Byte目	←	↓	→	↑	ST	R3	L3	SE	デジタルボタン1	押したとき0
4Byte目	△	×	○	□	R1	L1	R2	L2	デジタルボタン2	押したとき0
5Byte目	0 - 255								右スティック左右	右:255, 左:0
6Byte目	0 - 255								右スティック上下	下:255, 上:0
7Byte目	0 - 255								左スティック左右	右:255, 左:0
8Byte目	0 - 255								左スティック上下	下:255, 上:0
9Byte目	0 - 255								→ボタン感圧	強押:255, 無押:0
10Byte目	0 - 255								←ボタン感圧	強押:255, 無押:0
11Byte目	0 - 255								↑ボタン感圧	強押:255, 無押:0
12Byte目	0 - 255								↓ボタン感圧	強押:255, 無押:0
13Byte目	0 - 255								△ボタン感圧	強押:255, 無押:0
14Byte目	0 - 255								○ボタン感圧	強押:255, 無押:0
15Byte目	0 - 255								×ボタン感圧	強押:255, 無押:0
16Byte目	0 - 255								□ボタン感圧	強押:255, 無押:0
17Byte目	0 - 255								L1ボタン感圧	強押:255, 無押:0
18Byte目	0 - 255								R1ボタン感圧	強押:255, 無押:0
19Byte目	0 - 255								L2ボタン感圧	強押:255, 無押:0
20Byte目	0 - 255								R2ボタン感圧	強押:255, 無押:0
21Byte目	1-20バイト目の合計(0-255)								チェックサム	

※ST:START ボタン, SE:SELECT ボタン

使い方-充電編

このコントローラは USB ポートから充電を行うことができます。充電中は 5mm 黄色 LED1 が点灯、充電完了時には消灯します。バッテリー切れになった場合に電源スイッチを ON にしても電源 LED は完全に消灯したままになります。これはバッテリー内部の保護回路が作動し、放電を完全にストップしたためで充電すれば元のように動くようになります。ただし、バッテリー切れの状態が長く続くとバッテリーを痛めてしまうのでなるべく充電してから保管するようにしてください。長期間使わない場合は満充電した状態で保存すると寿命が短くなるので、ラジコン用の放電器等で放電しておいてください。
※送信機の電源を入れた状態で充電した場合、充電確認 LED は点灯したままになります。

使い方-操縦

マシンの操縦用コントローラとして使うにはパソコンと接続せずに電源を ON にしてください。
パソコンと接続している時は操縦用コントローラとして動作しない仕様になっていますが、「DebugTool->DebugTool.exe」を起動して接続後に「送信機起動」のチェックボックスにチェックを入れることで、パソコンに接続した状態でデュアルショック 2 のデータを送ることが出来ます。
無線書込みと併用して利用してください。

使い方-プログラミング(透過モード)

受信側(ロボット側)の XBee を透過モードで使った場合、比較的簡単にプログラミングを行うことが出来ます。(API モードでの使い方は省略します。)ただし以下の制約があります。

1. 送信元を判別できないため、1 対多の通信には向かない。
2. データの歯抜けを検知しにくいので信頼性が低い。
3. 無線ブートローダを使う場合、書込みする度に電源の再投入が必要。
4. 無線ブートローダを使う場合(API モード)と使わない場合(透過モード)で XBee の設定を変える必要がある(ユーザープログラム起動時に API から透過モードに切り替えることは可)

サンプル(XBee 実験基板用) PIC の書込みや XBee の設定は無線書込み＆無線デバッグ編を参考に！

```
#include <18f4520.h>
#device ADC=10
#fuses HS,NOWDT, PUT, NOPROTECT, NOLVP, BROWNOUT, NOCPD, NOWRT, MCLR, BORV42, NOFCMEN, NOCPB,
NOPBADEN, CCP2C1//,CCP2B3
#use delay(clock = 20000000)//20MHz
#use RS232(BAUD=57600, XMIT=PIN_C6, RCV=PIN_C7, PARITY=N, BITS=8,ERRORS)
#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)
//プロトタイプ宣言
void init();
void error();
void receive();
//変数宣言
int state;//ステート
int data_cnt;//受信したデータ数
int data[20];//データ
int check_sum;//チェックサム
void main(){
   init();
   printf("起動しました。¥r¥n");
   state = 0;
   while(TRUE){
      int temp;
      if(kbhit()){
         temp = getc();
         switch(state){
            case 0://先頭識別子受信
               if(temp == 0b11110000){
                  check_sum = temp;//チェックサムを初期化
                  data[0] = temp;
                  state = 1;
               }else{
                  error();
```

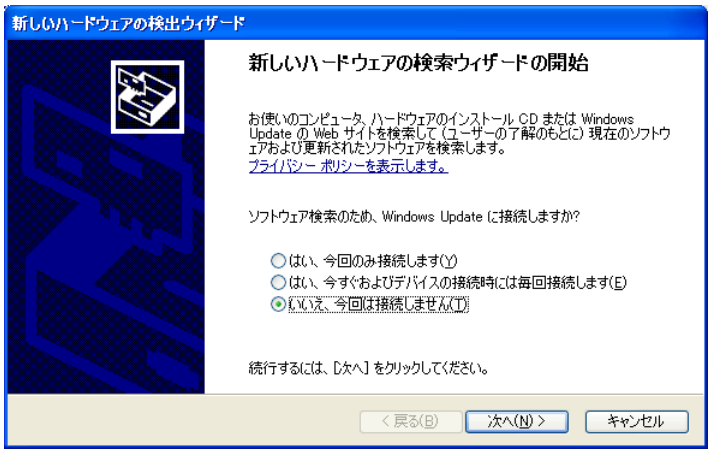
```
                }
                break;
            case 1://接続識別子受信
                if(temp == 0b10101010){//デュアルショック接続
                    data[1] = temp;
                    data_cnt = 2;
                    check_sum += temp;
                    state = 2;
                }else if(temp == 0b01010101){//デュアルショック未接続
                    data[1] = temp;
                    data_cnt = 2;
                    check_sum += temp;
                    state = 2;
                }else{//エラー
                    state = 0;
                    error();
                }
                break;
            case 2://データ受信
                data[data_cnt] = temp;
                data_cnt++;
                check_sum += temp;
                if(data_cnt == 20){//受信完了
                    state = 3;
                }
                break;
            case 3://チェックサム受信
                if(check_sum != temp){//チェックサムエラー
                    error();
                }else if(data[1] != 0b10101010){//デュアルショック 2 未接続
                    error();
                }else{
                    receive();
                }
                state = 0;
                break;
        }
    }
  }
}
void error(){
  static int16 cnt = 0;
  cnt++;
  printf("ErrorCnt=%lu¥r¥n",cnt);
}
void receive(){
  static int16 cnt = 0;
  cnt++;
  printf("ReceiveCnt=%luX-button=%u¥r¥n",cnt,data[14]);
}
void init(){
  output_a(0);//ポートの初期化
  output_b(0);
  output_c(0);
  output_d(0);
  output_e(0);
  //TRIS の設定
  set_tris_a(0b00000000);
  set_tris_b(0b00000000);
  set_tris_c(0b10000000);
  set_tris_d(0b00000000);
  set_tris_e(0b00000000);
}
```

使い方-パソコンとの接続

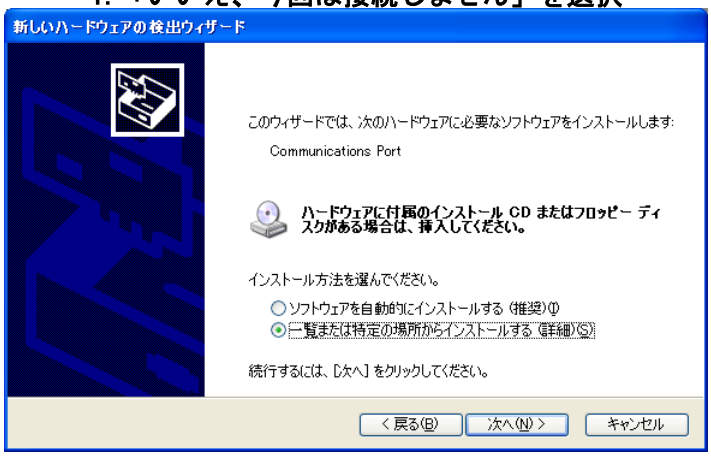
項目が多いので分けて解説します。

使い方-パソコンとの接続-ドライバのインストール編

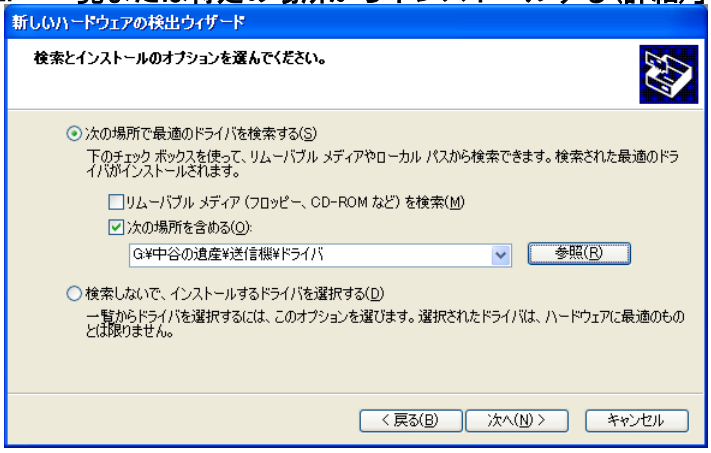
パソコンとの接続初回時には右ようなバルーンが表示され、ハードウェアの追加ウィザードが表示されます。以下の手順にしたがってドライバをインストールしてください。



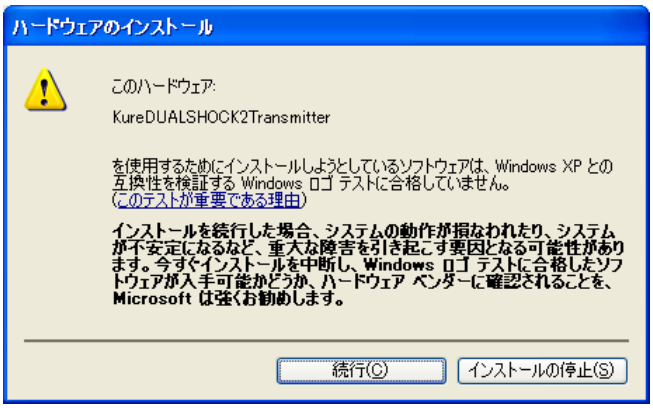
1.「いいえ、今回は接続しません」を選択



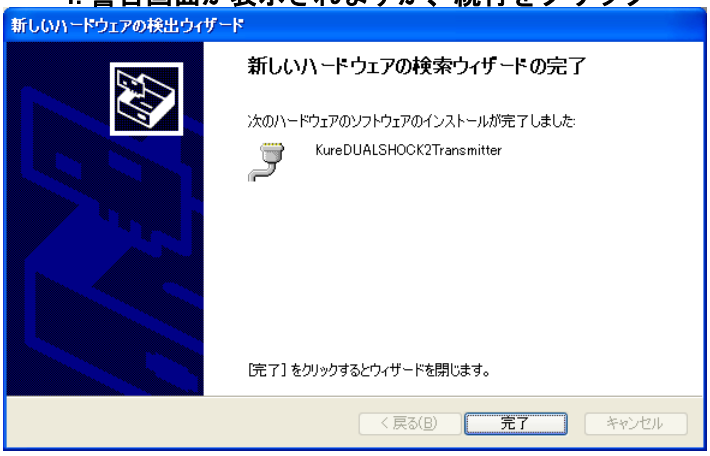
2.「一覧または特定の場所からインストールする(詳細)」



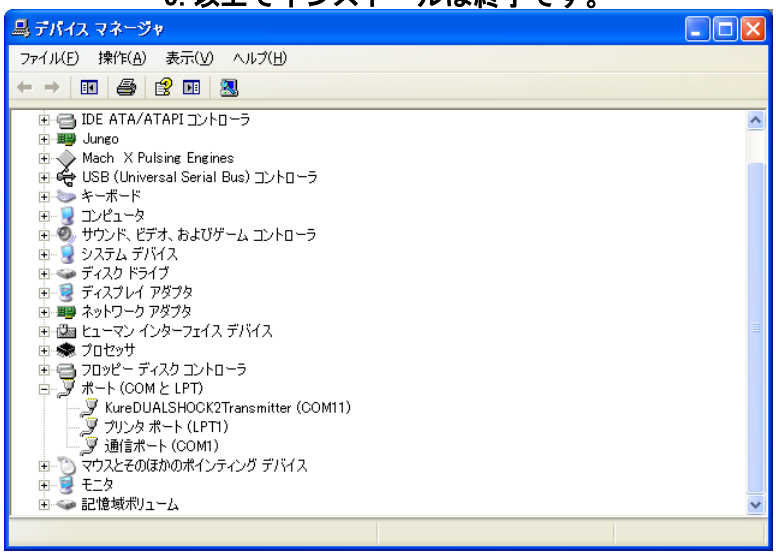
3. ドライバが入っているフォルダを選択



4. 警告画面が表示されますが、続行をクリック



5. 以上でインストールは終了です。

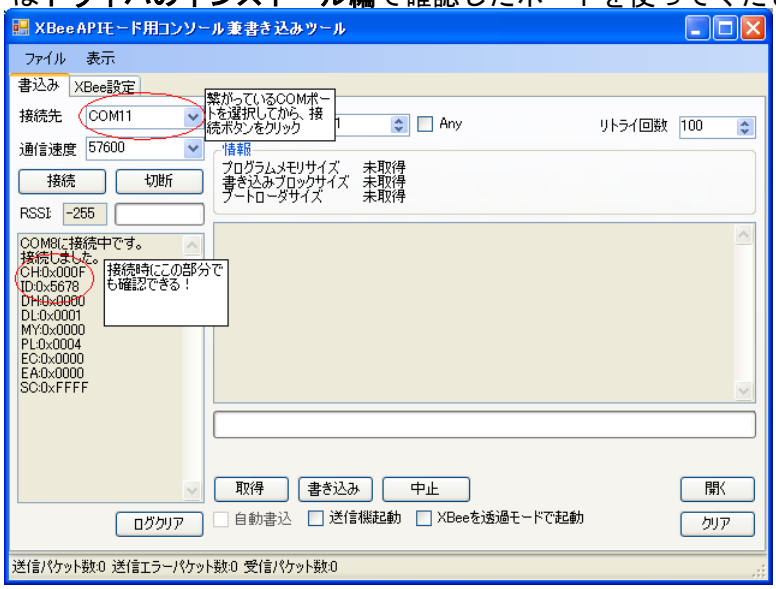


6. インストール後はデバイスマネージャにて COM 番号を確認。

KureDUALSHOCK2Transmitter として認識されているのが送信機です。
今回は COM11 として認識されました。

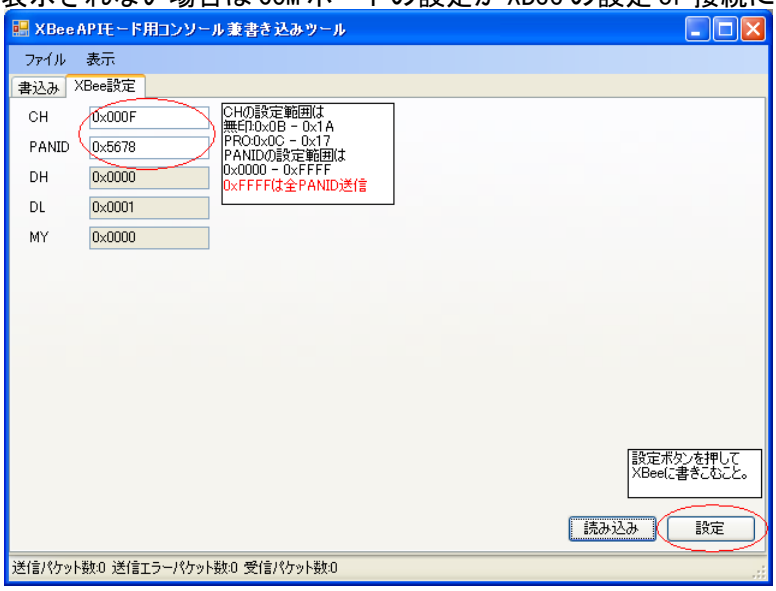
使い方-パソコンとの接続-XBee の設定

Xbee の設定には Digi 社の設定ツール X-CTU を利用します。
同梱の「XBee 設定ファイル->PAD.pro」ファイルを X-CTU で書きこんでください。
CH パラメータと ID パラメータは混信を防ぐためにコントローラごとに分けるようにしてください。
絶対に忘れないように！
なお、一回書き込んだあとは同梱の「DebugTool->Debug.exe」を使って CH, ID を変更することが出来ます。
接続する COM ポートは**ドライバのインストール編**で確認したポートを使ってください。



1. COM ポートを選択して接続ボタンをクリック

接続すると左のテキストボックスに今のパラメータが表示されます。
表示されない場合は COM ポートの設定か XBee の設定 or 接続に問題があります。



2. 設定が終わったら設定ボタンをクリック

XBee と XBeePro ではチャンネル設定の範囲が違うので注意してください。
注意:送信機側かマシン側に Pro を使った場合、両方の XBee を
Pro の設定範囲内に収める必要があります。

使い方-パソコンとの接続-無線書き込み＆無線デバッグ

無線ブートローダとこの送信機を利用して、無線書込みと無線デバッグを行う手順を説明します。
今回は printf の結果を PC に表示させるのを目標とします。（コントローラとしての利用は解説が長くなるので省略します。）

メモ:

無線書込み・デバッグを利用すると書き換えるたびに PIC を抜く必要が無いので何回も書き直すようなプログラムを書いているときに有用です。また、printf の結果をパソコンに表示できるのでプログラムのデバッグを効率的に行なえます。

1. 用意するもの

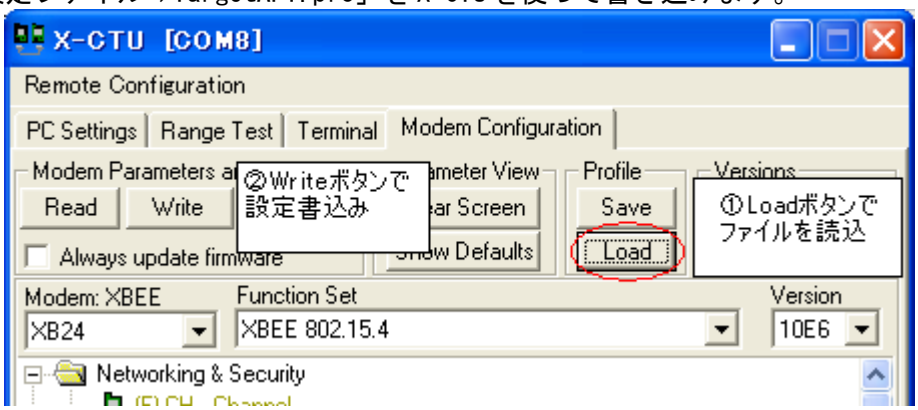
1. 無線書込みに対応した PIC と基板（今回は PIC18F4520 と XBee 実験ボードを使います）
2. XBee（XB24-ACI-001 等）
3. PIC ライター（ブートローダの書込みに必要）
4. この送信機（XBee の設定やドライバのインストールは終わっているものとします。）

2. ブートローダの書込み

「ブートローダ->18F4520_20MHz（OSC20MHz）.HEX」をライターで書込みます。

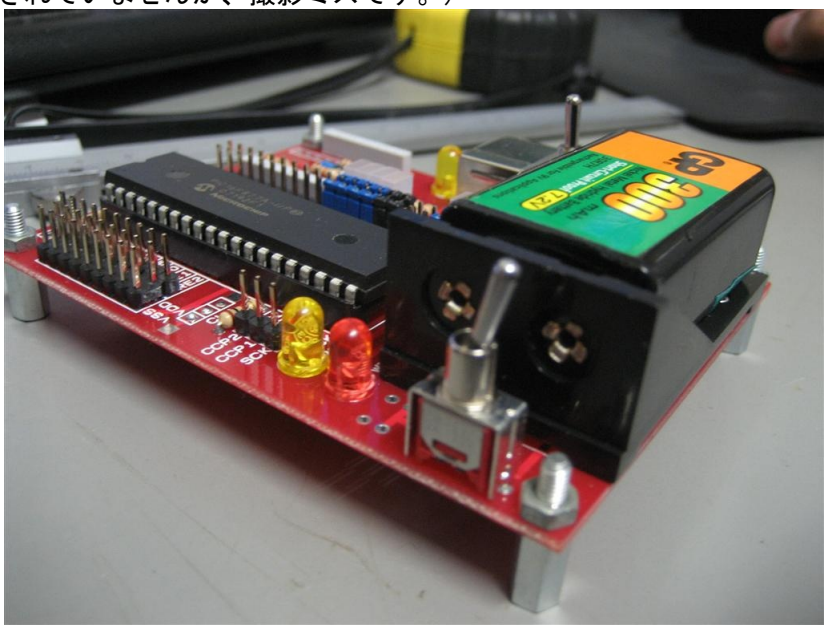
3. ターゲット側 XBee の設定

「XBee 設定ファイル->TargetAPI.pro」を X-CTU を使って書き込みます。



4. 基板にセット

書込みが終わった PIC18F4520 と XBee を基板にセットします。
（XBee がセットされていませんが、撮影ミスです。）

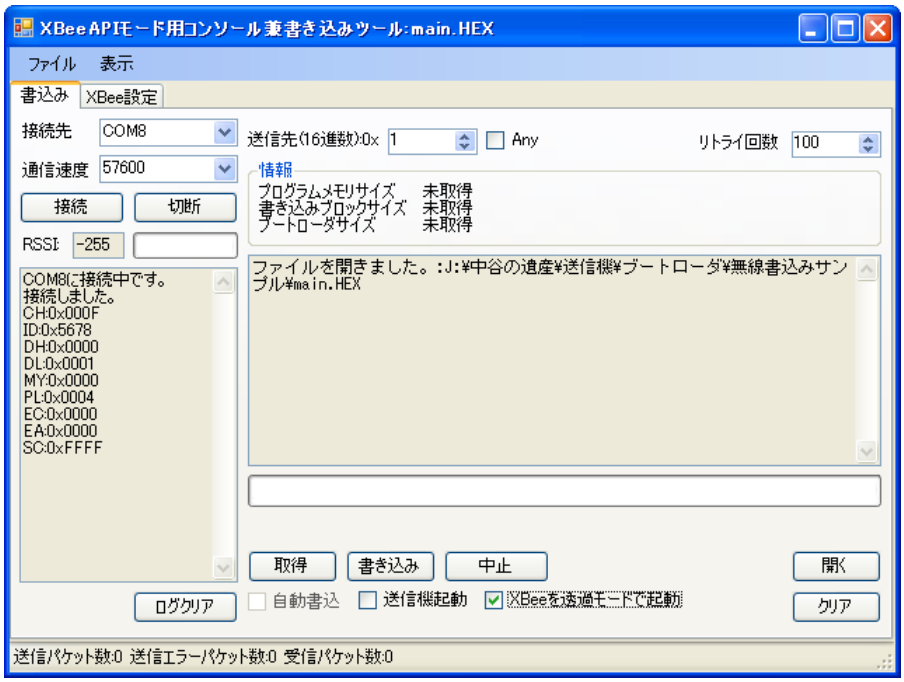


5.書き込みたいプログラムをコンパイル(XBee 実験基板用サンプル)

```
//XBee 実験ボード用の設定,セラロック 20MHz 用(PIC18F4520)
#include <18f4520.h>
#device ADC=10
#fuses HS,NOWDT, PUT, NOPROTECT, NOLVP, BROWNOUT, NOCPD,
NOWRT, MCLR, BORV42, NOFCMEN, NOCPB, NOPBADEN,
CCP2C1//,CCP2B3
#use delay(clock = 20000000)//20MHz
#use RS232(BAUD=57600, XMIT=PIN_C6, RCV=PIN_C7, PARITY=N,
BITS=8,ERRORS)
#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)
//プロトタイプ宣言
void init();
void main(){
  init();
  printf("起動しました。\r\n");
  while(TRUE){
    printf("テスト\r\n");
    delay_ms(1000);
  }
}
void init(){
  //ポートの初期化
  output_a(0);
  output_b(0);
  output_c(0);
  output_d(0);
  output_e(0);
  //TRIS の設定
  set_tris_a(0b00000000);
  set_tris_b(0b00000000);
  set_tris_c(0b10000000);
  set_tris_d(0b00000000);
  set_tris_e(0b00000000);
}
```

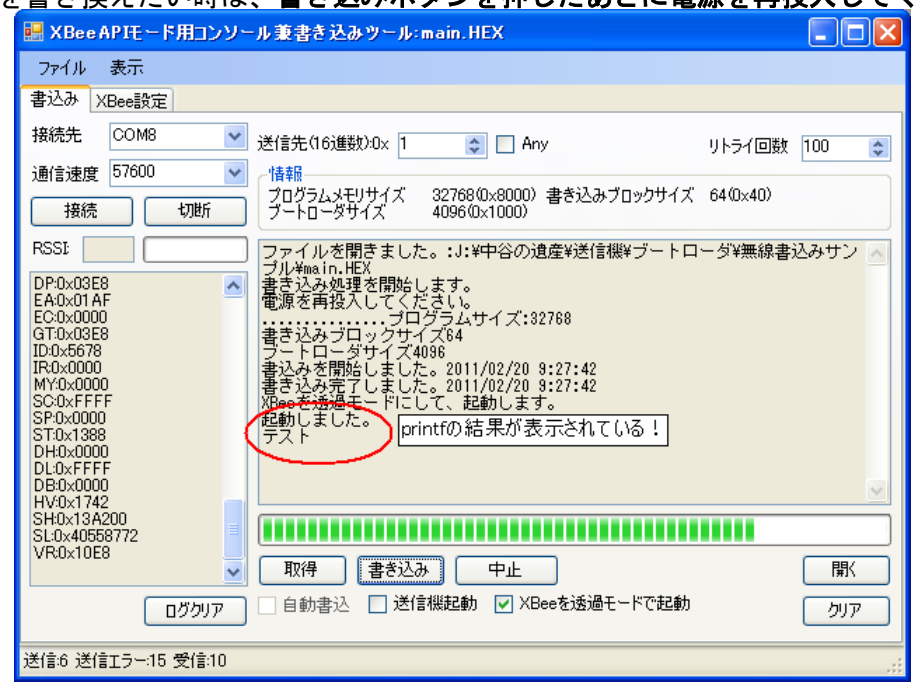
6.HEX ファイル無線書込み

1. Debug.exe を起動して接続ボタンをクリック。
2. 「Xbee を透過モードで起動」にチェックを入れておきます。
   ※透過モードではパケット化せずにデータを遅れるので簡単にデータを送受信できます。
   ただし RSSI や送信元の判別を行うことが出来ません。
3. 書き込みたい HEX ファイルを開きます。
4. XBee 実験ボードの電源を入れる準備をしておき、書込みボタンをクリックします。
   書込みボタンを押したら、電源を入れてください。書込みが始まります。



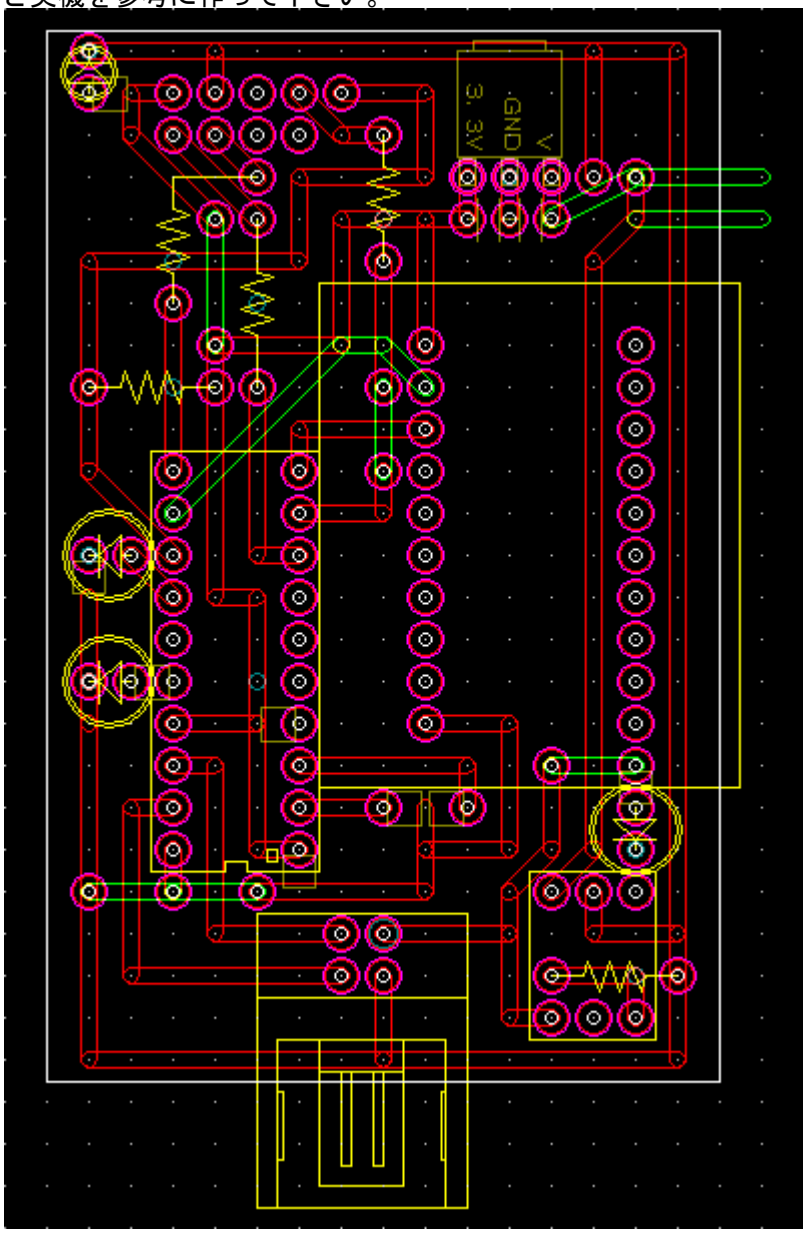
7. 無線デバッグ

書込みが終わると、プログラムが自動的に起動します。また、printf の結果が表示されます。
この機能を利用して無線で printf デバッグを行うことが出来ます。
プログラムを書き換えたい時は、**書き込みボタンを押したあとに電源を再投入してください。**



製作-基板製作

以下のパターン図と実機を参考に作って下さい。



製作-HEX ファイルの書込み

送信機.HEX ファイルを書きこむ際に、EEPROM を送信機ごとに書き換えておく必要があります。
下図を参考にしてください。
※USB 機器のシリアル番号を分けるため。同じシリアル番号の USB 機器を 2 台つなげるとブルースクリーンで落ちる。