ＣｂａｒによるプログラムIＩ　　Ｖｅｒ１．００　２００６．５．５
　　Ｖｅｒ１．０１　２００６．１１．０９

■ＣｂａｒをＨ８－３０５２ＣＰＵボードの開発例

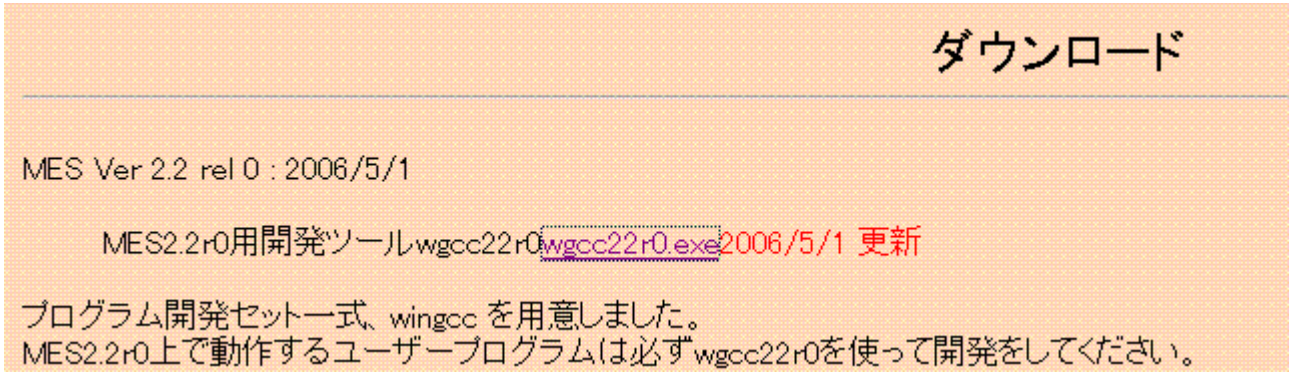
㈱秋月電子通商で販売している「ＡＫＩ－Ｈ８／３０６９ＦフラッシュマイコンｌＡＮボード」、「ＡＲＭマイコンボードキット」等を購入すると開発用ソフトウエアが添付しています。統合開発環境ＣｂａｒとＧＣＣのＣコンパイラ、ＲＯＭライタソフトとプログラム開発に必要なソフトウエアが同封されています。「ＧＣＣ　ｏｎ　Ｗｉｎｄｏｗｓ」としてＷｉｎｄｏｗｓ上で動くＧＣＣが添付しています。これはＡＲＭ、Ｈ８、ＰＩＣの開発が行えるようです。

また、それとは別にダウンロード無償で最新の開発環境が入手できます。これは開発できるＣＰＵにルネサス社のＳＨが追加されていて更に便利になっています。

今回はダウンロードした環境を使用して、当社製Ｈ８－３０５２ＣＰＵボードの開発方法について記述します。他のＨ８、ＳＨ、ＡＲＭ等ＣＰＵ開発をこの環境で行う場合でも参考になると思います。

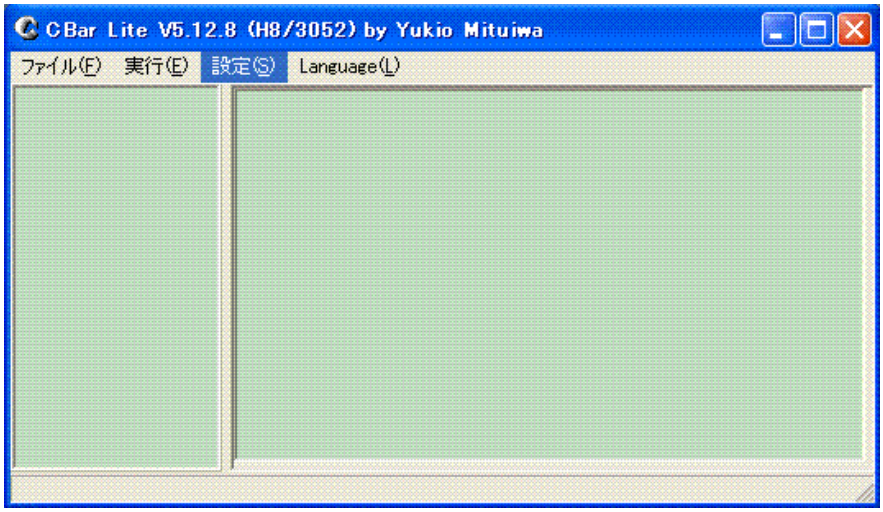
●ダウンロード

http://mes.sourceforge.jp/mes2/download-j.html
よりｗｇｃｃｘｘｘｘ．ｅｘｅをダウンロードします。（最新のもの）

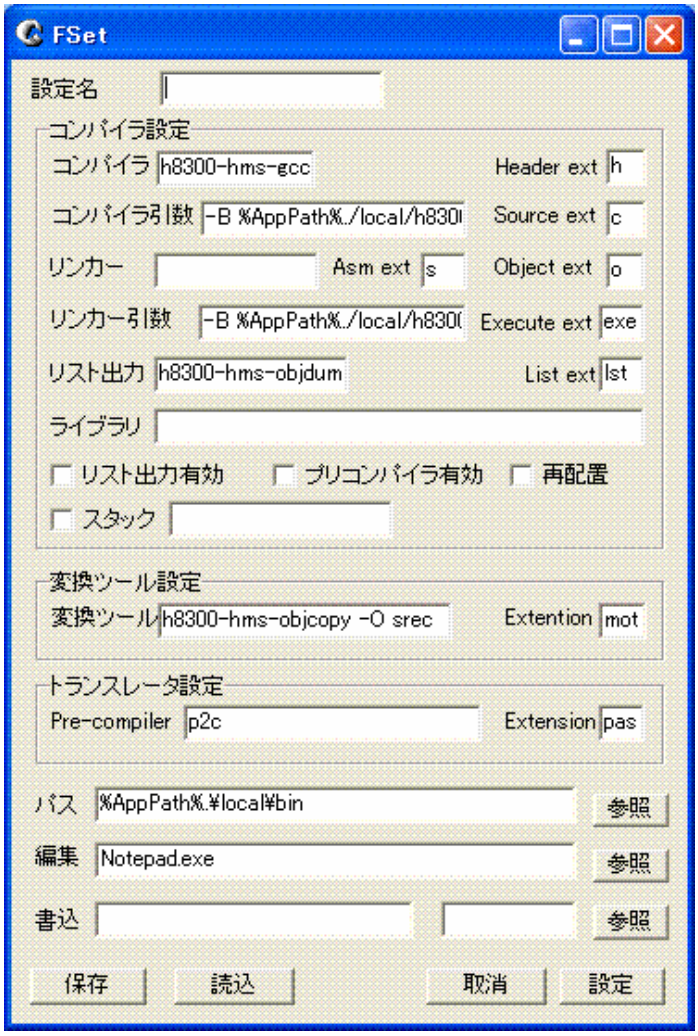


●　開発するＣＰＵを選択する

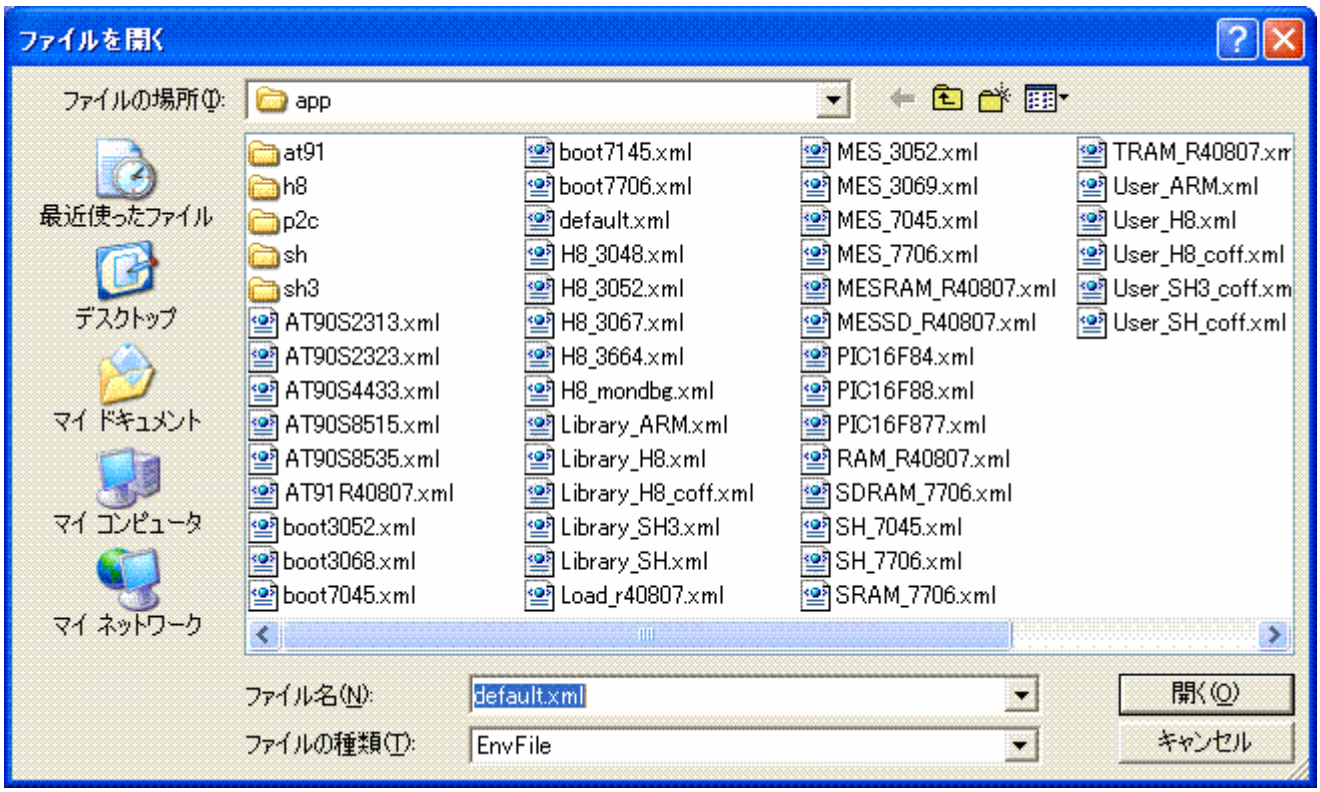
Ｈ８－３０５２のｘｍｌファイルはありませんので、製作します。「　設定　」をクリックします。



「　環境設定　」をクリックすると以下のＦＳｅｔ画面が表示されます。「　読込　」をクリックします。



各種ＣＰＵ別の設定ファイルが表示されます。



ここに見えるのがデホルトで開発できるマイコンの環境設定ファイル（．ｘｍｌ)です。ＡＴ９０、、はアトメル社のＡＲＭ７コアシングルチップマイコンです。Ｈ８、ＳＨ、ＰＩＣ等の開発ができることがわかります。
Ｈ８＿３０５２．ｘｌｍファイルを作るために雛形としてＨ８＿３０４８．ｘｌｍを読み込みます。

● Ｈ８＿３０５２．ｘファイルを作成

Ｈ８＿３０４８．ｘファイルをもとにＨ８＿３０５２．ｘファイルを作成します。ｘファイルはデホルトでは以下のディレクトリにあります。
Ｃ：￥ｗｉｎｇｃｃ￥ａｐｐ￥ｈ８＿３０４８．ｘ
エディタでファイルを開きます。

ＣＰＵによってなにを変えるか、というと　ｖｅｃｔｏｒｓの下にある
　　ｒｏｍ、ｒａｍ、ｓｔａｃｋの３つの右側にある数字です。それぞれＣＰＵハードウエア固有の数値です。Ｈ８－３０５２の場合、
ｒｏｍ　：　ｏ　＝　０ｘ０００１００、｜＝　０ｘ０７ＦＦＦＦ　とします。
左は０ｘ０００１００固定、右はそのＣＰＵのＲＯＭの最大アドレスを書きます。
ｒａｍ　：　ｏ　＝　０ｘｆｆｄｆ１０、｜＝　０ｘ００１ｆｆｆ
左はＲＡＭの開始アドレス、右は容量を書きます。
ｓｔａｃｋ　ｏ　＝　０ｘｆｆｆｆ０ｃ、｜＝　０ｘ０００００４
左はＲＡＭの最終アドレス　－　３の数値を、右は０ｘ０００００４固定です。

はじめからあるｈ８＿３０４８．ｘファイル。Ｈ８－３０４８のハードウエアに沿ったデータが記載されています。

h8_3048.x - メモ帳
ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

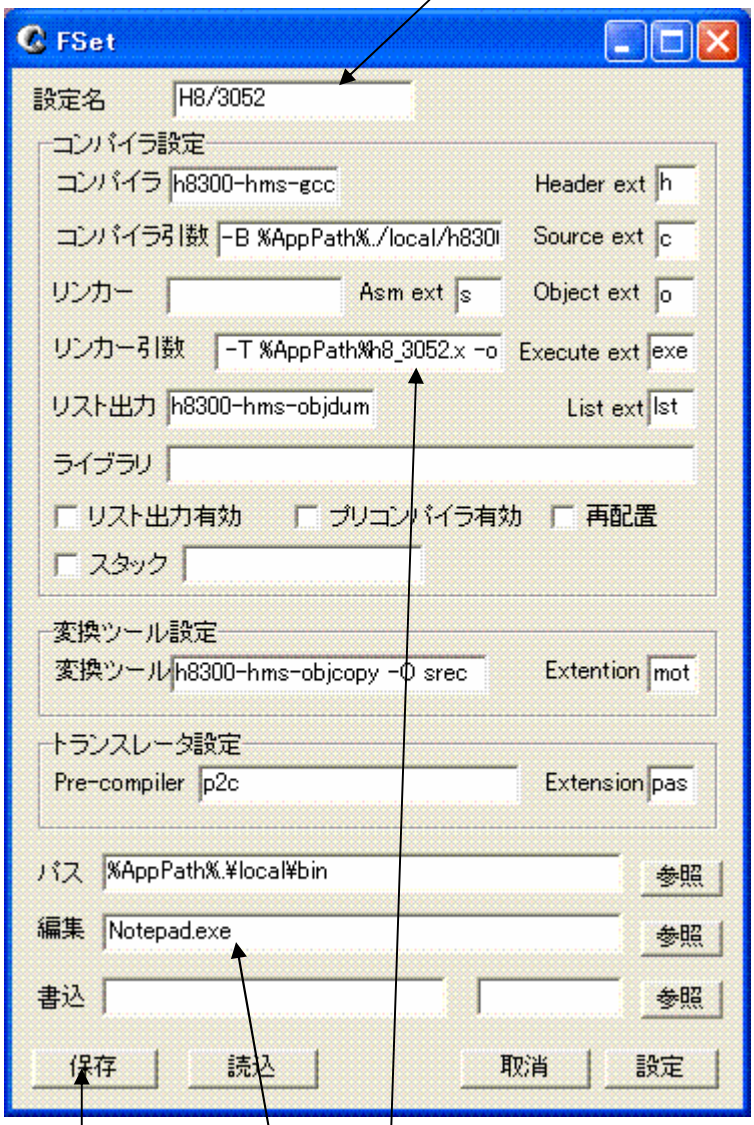
```
OUTPUT_FORMAT("coff-h8300")
OUTPUT_ARCH(h8300h)
ENTRY("_start")
MEMORY
{
        vectors(r)      : o = 0x000000, l = 0x000100
        rom(rx)         : o = 0x000100, l = 0x01ff00
        ram(rwx)        : o = 0xffef10, l = 0x000a00
        stack(rw)       : o = 0xffff0c, l = 0x000004
}
SECTIONS
{
.vectors : {
        /* Use something like this to place a specific function's address
```

以下がＨ８＿３０４８．ｘをもとに製作したＨ８＿３０５２．ｘファイルです。ｒｏｍ、ｒａｍ、ｓｔａｃｋの数値が書き換えられています。ファイル名を「Ｈ８＿３０５２．ｘ」としてセーブします。

h8_3052.x - メモ帳
ファイル(F)　編集(E)　書式(O)　表示(V)　ヘルプ(H)

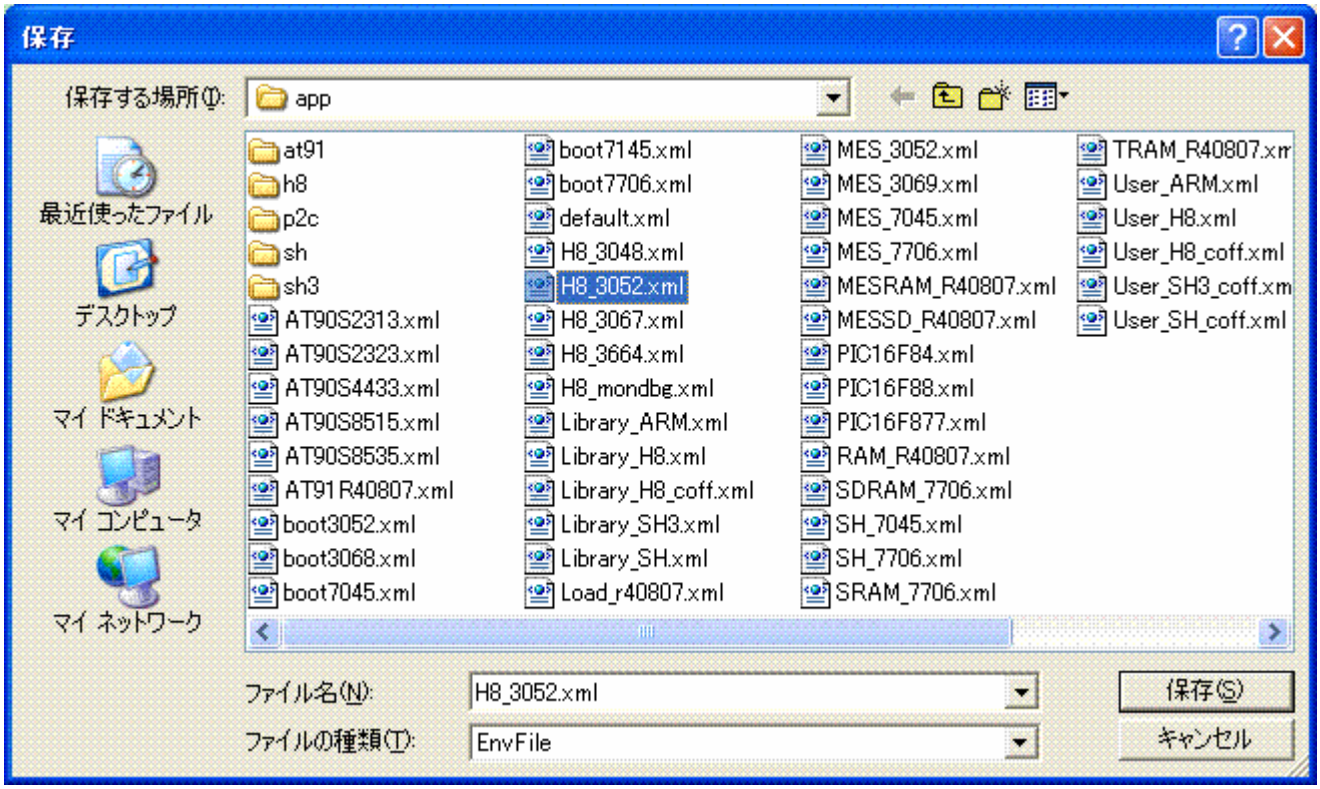
```
OUTPUT_FORMAT("coff-h8300")
OUTPUT_ARCH(h8300h)
ENTRY("_start")
MEMORY
{
        vectors(r)      : o = 0x000000, l = 0x000100
        rom(rx)         : o = 0x000100, l = 0x07ffff
        ram(rwx)        : o = 0xffdf10, l = 0x001fff
        stack(rw)       : o = 0xffff0c, l = 0x000004
}
SECTIONS
{
.vectors : {
        /* Use something like this to place a specific function's address
```

ｘファイルをもとにｘｍｌファイルを作成します。Ｆｓｅｔを修正します。
　設定名をＨ８／３０５２に、リンカー引数の中のｘファイルを「ｈ８＿３０５２ｘ」に書き換え「　保存　」します。これでＨ８＿３０５２．ｘｌｍファイルが作成されました。他のＣＰＵを使用した後でもこのファイルを読み込めばＨ８－３０５２の開発環境に設定されます。　　設定名をＨ８／３０５２にします。



セーブするときはここ　　ここをｈ８＿３０４８からｈ８＿３０５２に書き換えます。

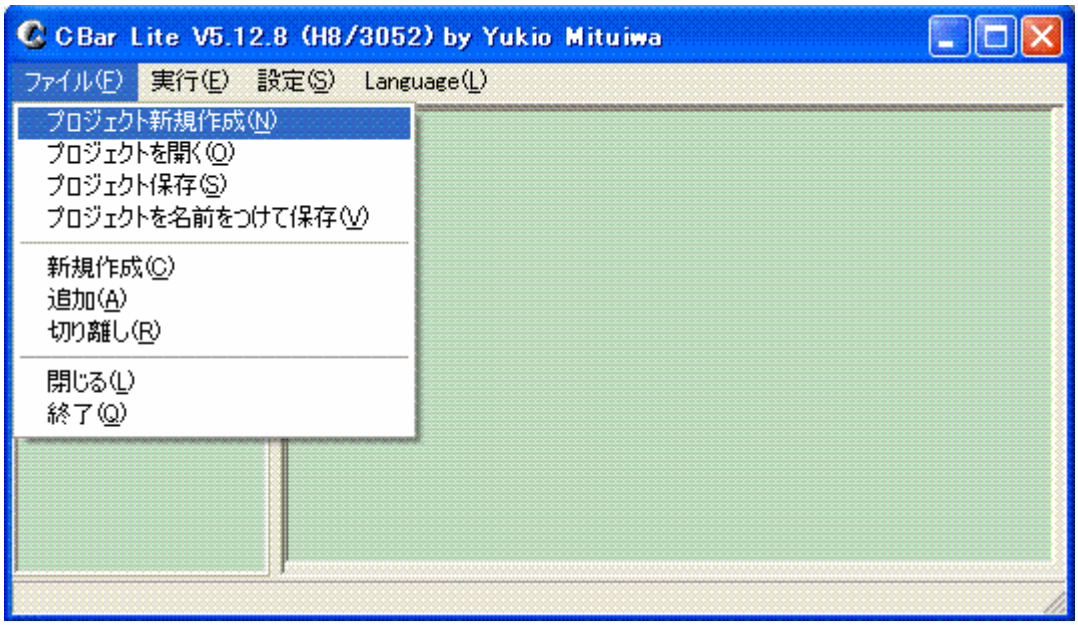
ちなみに、デホルトの「Ｅｄｉｔｏｒ」、「Ｗｒｉｔｅｒ」等も自分の好きなものに変更できます。



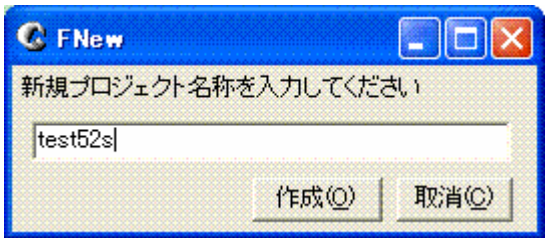
これでＨ８－３０５２の開発環境が整いました。以下、プログラム作成について説明します。

【プログラム作成】

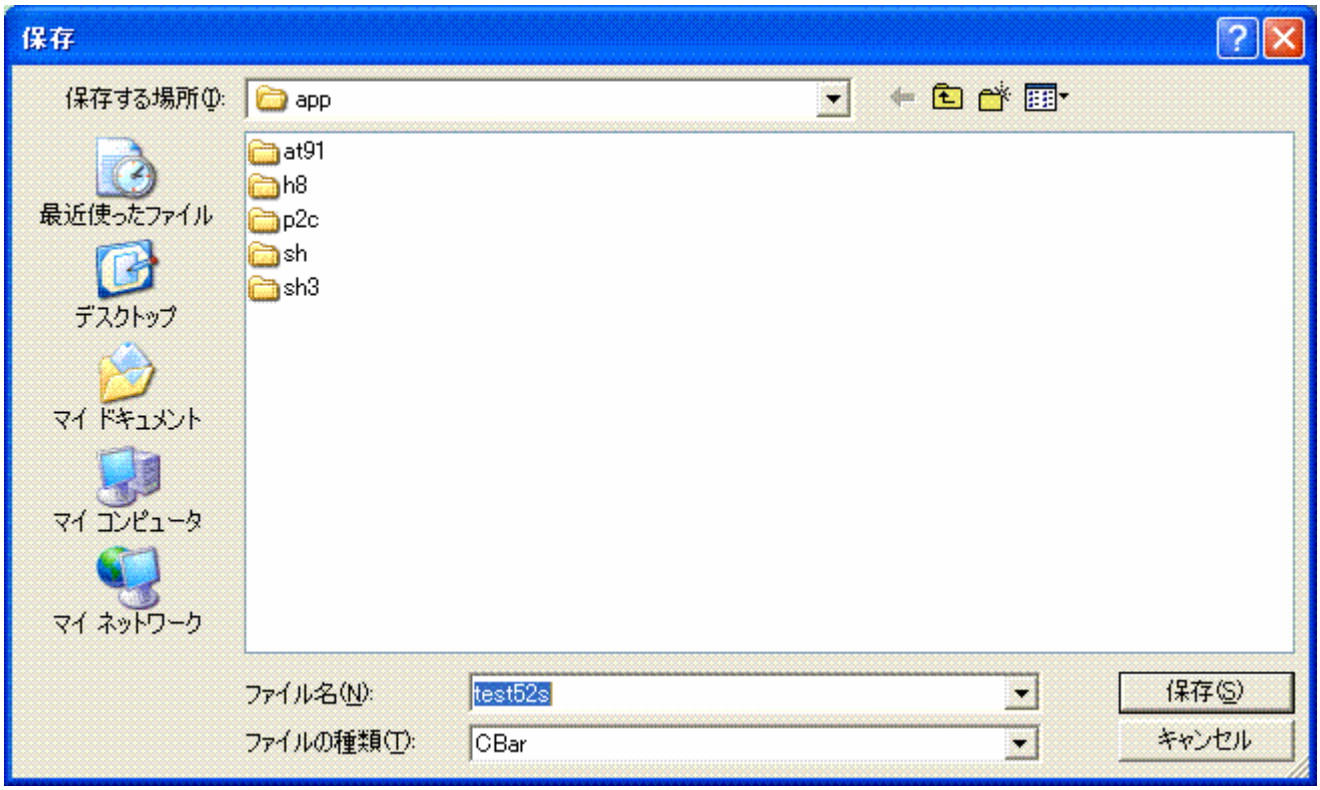
初めにプロジェクトを作成します。



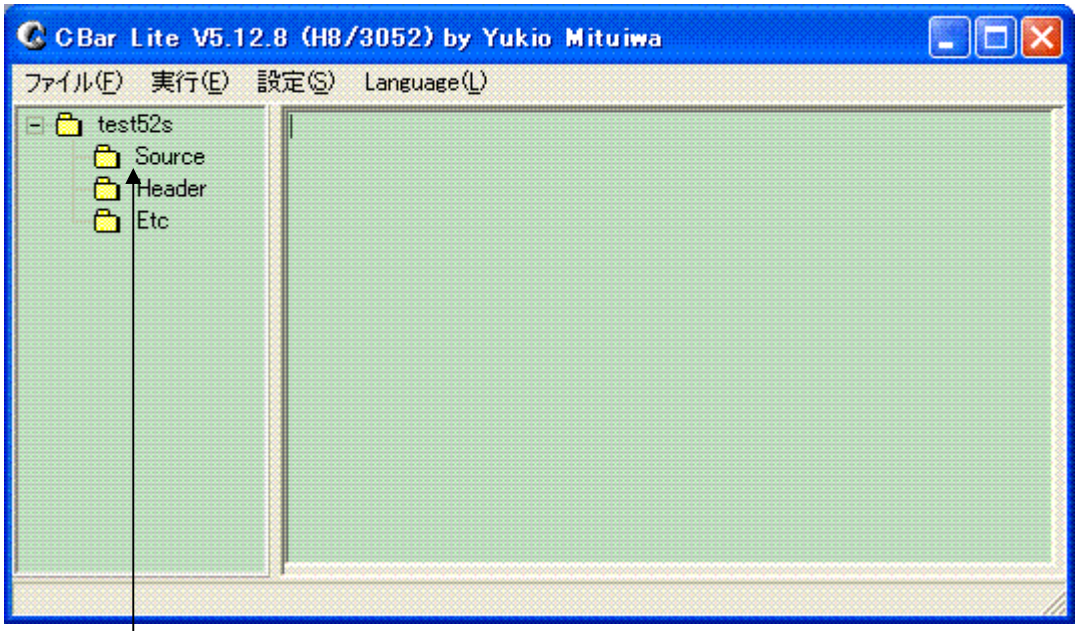
ここでは「ｔｅｓｔ５２ｓ」とします。



プロジェクトを保存します。ここではＣ：￥ｗｉｎｇｃｃ￥ａｐｐ￥ｈ８￥の中に保存します。新規にソースファイルホルダを製作してもいいでしょう。
【注意】Ｃ：￥ｗｉｎｇｃｃ￥ａｐｐ￥にセーブするとコンパイル時に「共有違反」とコメントが出て正常に動きません。

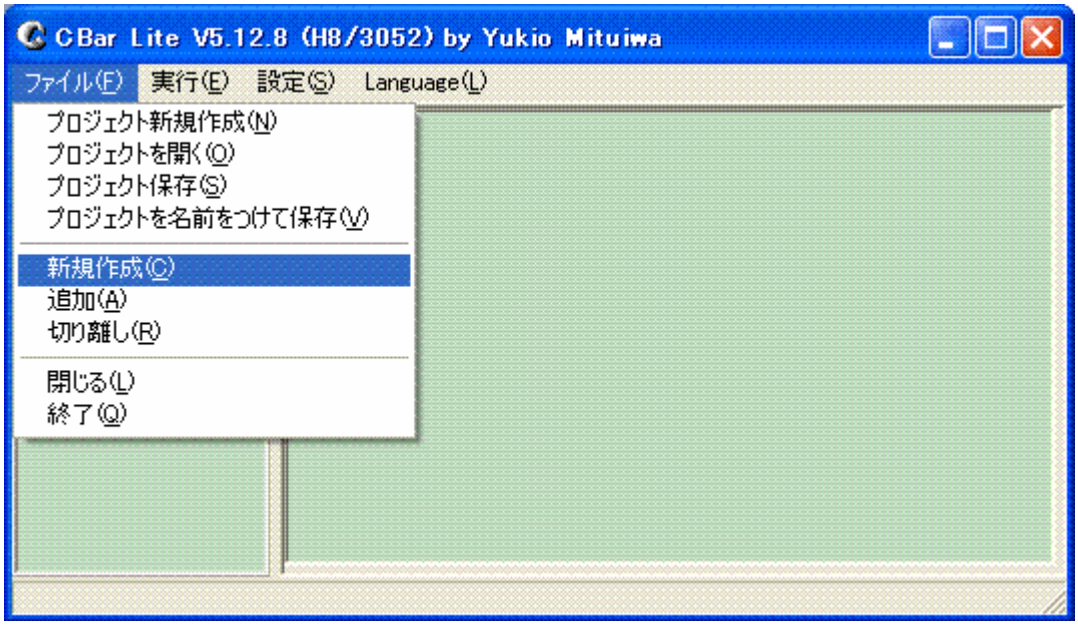


プロジェクト「ｔｅｓｔ５２ｓ］を作成するとその中に
「Ｓｏｕｒｃｅ]、「Ｈｅａｄｅｒ」、「Ｅｔｃ」のホルダが自動的に作成されます。

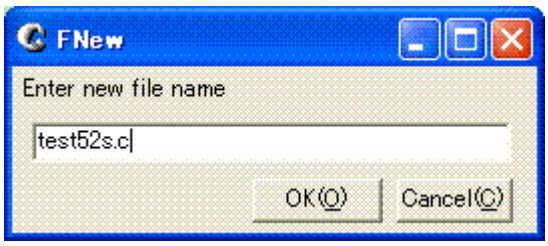


「Ｓｏｕｒｃｅ」をクリックします。

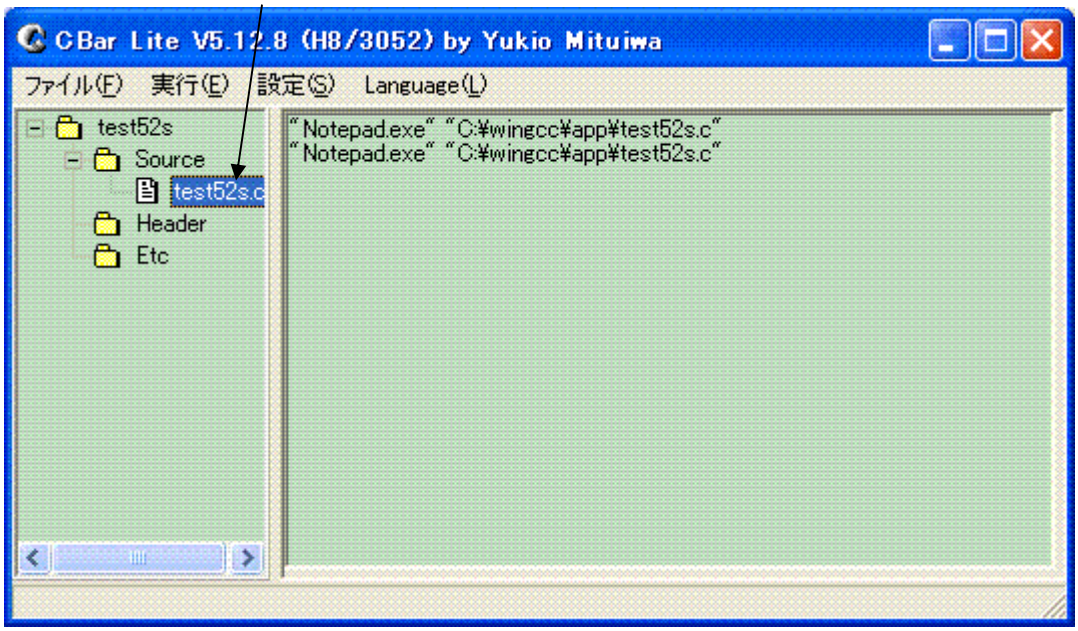
「ファイル（Ｆ)」をクリックし、「新規作成」をクリックし、ソースファイルを作成します。



ここでは「ｔｅｓｔ５２ｓ．ｃ」とします。「ＯＫ」をクリックします。



以降、ここをダブルクリックすると「Ｆｓｅｔ」で設定したエディタでソースファイルが開きます。



　ここでは弊社のサイトよりダウンロードできる「ｔｅｓｔ５２ｓ．ｃ」をｃ:\ｗｉｎｇｃｃ￥ａｐｐ￥ｈ８￥にコピーして開きます。

■ｔｅｓｔ５２ｓ．ｃについて

ｔｅｓｔ５２ｓ．ｃは弊社のＢＣＨ８３０５２　ＣＰＵボードでＤ／Ａコンバータ、Ａ／Ｄコンバータ、Ｉ／Ｏ、ＳＩＯテストが行えるサンプルソフトです。この環境以外でも動作します。詳細は別ファイルの「サンプルプログラムの使い方」をご参照ください。

以下ソースファイルです。

```
/* BCH83052 テスト１

        test52s.c

        ４つのサンプルプログラムが動作します。

        void test_da(void);         D/A コンバータテスト
        void test_ad(void);         A/D コンバータテスト
        void test_io(void);         I/O テスト
        void test_sio(void);        SIO テスト

*/


//port

#define P1DDR     (*((unsigned char *)0xffffc0))
#define P1        (*((unsigned char *)0xffffc2))

#define P2DDR     (*((unsigned char *)0xffffc1))
#define P2        (*((unsigned char *)0xffffc3))
#define P2PCR     (*((unsigned char *)0xffffd8))

#define P3DDR     (*((unsigned char *)0xffffc4))
#define P3        (*((unsigned char *)0xffffc6))

#define P4DDR     (*((unsigned char *)0xffffc5))
#define P4PCR     (*((unsigned char *)0xffffda))
#define P4        (*((unsigned char *)0xffffc7))

#define P5DDR     (*((unsigned char *)0xffffc8))
#define P5        (*((unsigned char *)0xffffca))

#define P6DDR     (*((unsigned char *)0xffffc9))
#define P6        (*((unsigned char *)0xffffcb))

#define P7        (*((unsigned char *)0xffffce))

#define P8DDR     (*((unsigned char *)0xffffcd))
#define P8        (*((unsigned char *)0xffffcf))

#define P9DDR     (*((unsigned char *)0xffffd0))
```

```
#define P9        (*((unsigned char *)0xffffd2))

#define PADDR     (*((unsigned char *)0xffffd1))
#define PA        (*((unsigned char *)0xffffd3))

#define PBDDR     (*((unsigned char *)0xffffd4))
#define PB        (*((unsigned char *)0xffffd6))

//wait

#define WCER      (*((unsigned char *)0xffffef))

//SI00

#define SMR       (*((unsigned char *)0xffffb0))
#define BRR       (*((unsigned char *)0xffffb1))
#define SCR       (*((unsigned char *)0xffffb2))

#define TDR       (*((unsigned char *)0xffffb3))
#define SSR       (*((unsigned char *)0xffffb4))
#define RDR       (*((unsigned char *)0xffffb5))

//SI01
#define SMR1      (*((unsigned char *)0xffffb8))
#define BRR1      (*((unsigned char *)0xffffb9))
#define SCR1      (*((unsigned char *)0xffffba))

#define TDR1      (*((unsigned char *)0xffffbb))
#define SSR1      (*((unsigned char *)0xffffbc))
#define RDR1      (*((unsigned char *)0xffffbd))

#define MSTCRH    (*((unsigned char *)0xfee01c))

//DA
#define DADR0     (*((unsigned char *)0xffffdc))
#define DADR1     (*((unsigned char *)0xffffdd))
#define DACR      (*((unsigned char *)0xffffde))
#define DASTCR    (*((unsigned char *)0xffff5c))

//A/D
#define ADDRAH    (*((unsigned char *)0xffffe0))
#define ADDRAL    (*((unsigned char *)0xffffe1))
#define ADDRBH    (*((unsigned char *)0xffffe2))
#define ADDRBL    (*((unsigned char *)0xffffe3))
#define ADDRCH    (*((unsigned char *)0xffffe4))
#define ADDRCL    (*((unsigned char *)0xffffe5))
#define ADDRDH    (*((unsigned char *)0xffffe6))
#define ADDRDL    (*((unsigned char *)0xffffe7))
#define ADCSR     (*((unsigned char *)0xffffe8))
#define ADCR      (*((unsigned char *)0xffffe9))
```

```
//関数

void char_out1(unsigned char);
unsigned char char_in1(void);
void wait(unsigned short);

void ascout1(unsigned char);
void hexascii(unsigned char);

void test_da(void);
void test_ad(void);
void test_io(void);
void test_sio(void);

unsigned char ascl,asch;


main()
{

unsigned char cf,bf,cyc;


        P8DDR = 0xfc;           //_CS0,1,2 active

        P1DDR = 0xff;           //
        P2DDR = 0xff;           //
        P3DDR = 0xff;           //
        P4DDR = 0x00;           //all_in
        P4PCR = 0xff;           //all_pull_up
        P5DDR = 0xff;           //
        P6DDR = 0xff;           //
        P8DDR = 0xff;           //
        P9DDR = 0xff;
        PADDR = 0xff;           //all_out
        PBDDR = 0xff;           //all_out

//SIO0 ｲﾆｼｬﾙ

        BRR = 19;          /* 38400bps */
        SMR = 0;
        SCR = 0x30;

//SIO1 ｲﾆｼｬﾙ

        BRR1 = 19;              /* 38400bps */
        SMR1 = 0;
        SCR1 = 0x30;
```

```
//SIO0 CK
        char_out1('H');
        char_out1('8');
        char_out1('/');
        char_out1('3');
        char_out1('0');
        char_out1('5');
        char_out1('2');
        char_out1(0x0d);
        char_out1(0x0a);

        cf = char_in1();            //動作番号読み込み

        char_out1('T');
        char_out1('E');
        char_out1('S');
        char_out1('T');
        char_out1(' ');
        char_out1('N');
        char_out1('o');
        char_out1('=');
        char_out1(cf);
        char_out1(0x0d);
        char_out1(0x0a);


        switch(cf & 0x0f)
        {
                case 0:test_da();break;
                case 1:test_ad();break;
                case 2:test_io();break;
                case 3:test_sio();break;
                default:test_io();
        }

}


/* subroutins */

void test_sio(void)
{
        while(1)
        {
                char_out1(char_in1());
        }
}

void test_io(void)
{
```

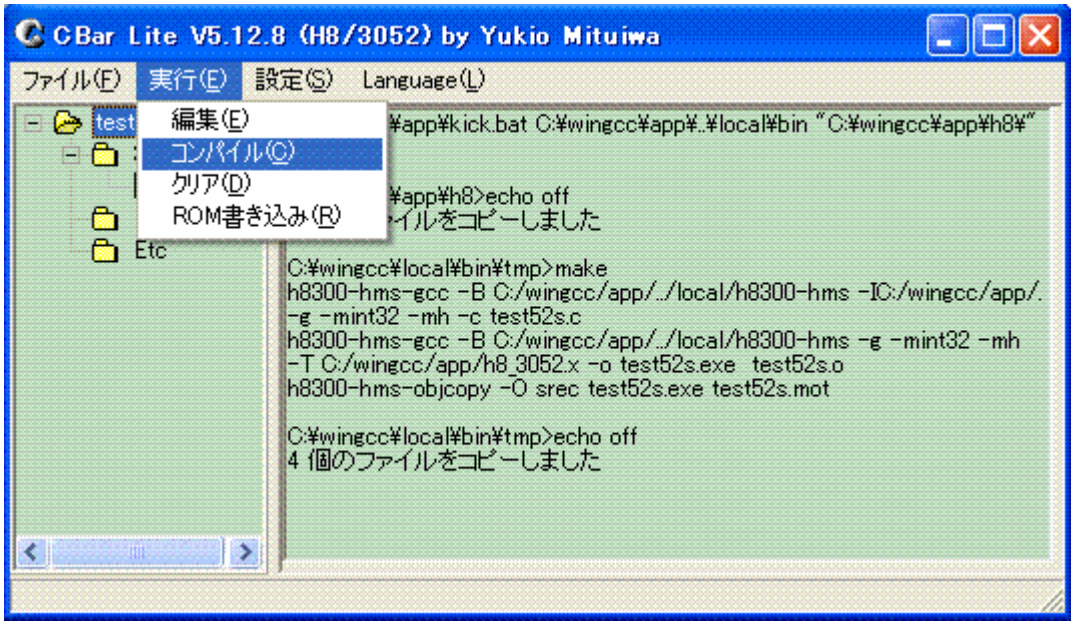
```
        while(1)
        {
                ascout1(~P4);
                PB = 0xff;
                wait(10000);
                PB = 0x00;
                wait(10000);
        }
}

void test_ad(void)
{
unsigned short data;
unsigned char cl,ch;

        while(1)
        {
                //ch0
                ADCSR = 0x00;               //ch0 a/d 変換,ｼﾝｸﾞﾙﾓｰﾄﾞ
                ADCSR |= 0x20;              //ch0 a/d 変換開始
                while((ADCSR & 0x80) == 0)
                        ;                                       //A/D ｴﾝﾄﾞﾌﾗｸﾞ まち
                ADCSR &= 0x7f;              //ADF ｸﾘｱｰ 0x7f の AND を取る*/
                data = ADDRAH;              //a/d 16bit(内ﾃﾞｰﾀ 10bit) 読み込み
                data <<= 8;
                data += ADDRAL;
                data >>= 6;                             //6bit 右ｼﾌﾄ

                cl = data;ch = data >> 8;
                char_out1('A');
                char_out1('D');
                char_out1('0');
                char_out1('=');
                ascout1(ch);
                ascout1(cl);
                char_out1(' ');

                //ch1

                ADCSR = 0x01;               //ch1 a/d 変換,ｼﾝｸﾞﾙﾓｰﾄﾞ
                ADCSR |= 0x20;              //ch1 a/d 変換開始
                while((ADCSR & 0x80) == 0)
                        ;                   //A/D ｴﾝﾄﾞﾌﾗｸﾞ まち
                ADCSR &= 0x7f;              //ADF ｸﾘｱｰ 0x7f の AND を取る*/
                data = ADDRBH;              //a/d 16bit(内ﾃﾞｰﾀ 10bit) 読み込み
                data <<= 8;
                data += ADDRBL;
                data >>= 6;                 //6bit 右ｼﾌﾄ

                cl = data;ch = data >> 8;
```

```
char_out1('A');
char_out1('D');
char_out1('1');
char_out1('=');
ascout1(ch);
ascout1(cl);
char_out1(' ');

wait(10000);

//ch2

ADCSR = 0x02;               //ch2 a/d 変換,ｼﾝｸﾞﾙﾓｰﾄﾞ
ADCSR |= 0x20;              //ch2 a/d 変換開始
while((ADCSR & 0x80) == 0)
        ;                                   //A/D ｴﾝﾄﾞﾌﾗｸﾞまち
ADCSR &= 0x7f;              //ADF ｸﾘｱｰ 0x7f の AND を取る*/
data = ADDRCH;              //a/d 16bit(内ﾃﾞｰﾀ10bit) 読み込み
data <<= 8;
data += ADDRCL;
data >>= 6;                         //6bit 右ｼﾌﾄ

cl = data;ch = data >> 8;
char_out1('A');
char_out1('D');
char_out1('2');
char_out1('=');
ascout1(ch);
ascout1(cl);
char_out1(' ');

//ch3

ADCSR = 0x03;               //ch2 a/d 変換,ｼﾝｸﾞﾙﾓｰﾄﾞ
ADCSR |= 0x20;              //ch2 a/d 変換開始
while((ADCSR & 0x80) == 0)
        ;                                   //A/D ｴﾝﾄﾞﾌﾗｸﾞまち
ADCSR &= 0x7f;              //ADF ｸﾘｱｰ 0x7f の AND を取る*/
data = ADDRDH;              //a/d 16bit(内ﾃﾞｰﾀ10bit) 読み込み
data <<= 8;
data += ADDRDL;
data >>= 6;                         //6bit 右ｼﾌﾄ

cl = data;ch = data >> 8;
char_out1('A');
char_out1('D');
char_out1('3');
char_out1('=');
ascout1(ch);
ascout1(cl);
```

```
                char_out1(' ');

                wait(10000);
        }
}


void test_da(void)
{
//DA Test
unsigned char dabuff;


        dabuff = 0;
        DACR = 0xff;        //DA0,1 出力

        while(1)
        {
                DADR0 = dabuff++;
                DADR1 = dabuff++;
        }
}




void wait(unsigned short loop)
{
int time;

        while(loop != 0)
        {
                loop--;
                for(time = 0;time < 50;time++)
                                ;
        }
}


unsigned char char_in1(void)
{
unsigned char data;

        while((SSR1 & 0x40) == 0)
                ;
        data = RDR1;
        SSR1 &= 0xbf;
        return (data);
}
```

```
void char_out1(unsigned char dat)
{
        while((SSR1 & 0x80) == 0)
                ;
        TDR1 = dat;                     //data set
        SSR1 &=~0x80;                   //tdre clear
}


void ascout1(unsigned char dat)
{
                hexascii(dat);
                char_out1(asch);
                char_out1(ascl);
}

void hexascii(unsigned char dat)
{
unsigned char     cx;

                cx = dat & 0x0f;
                if(cx > 9){ascl = cx + 0x37;}else{ascl = cx + 0x30;}
                cx = dat;
                cx >>= 4;
                if(cx > 9){asch = cx + 0x37;}else{asch = cx + 0x30;}

}
```

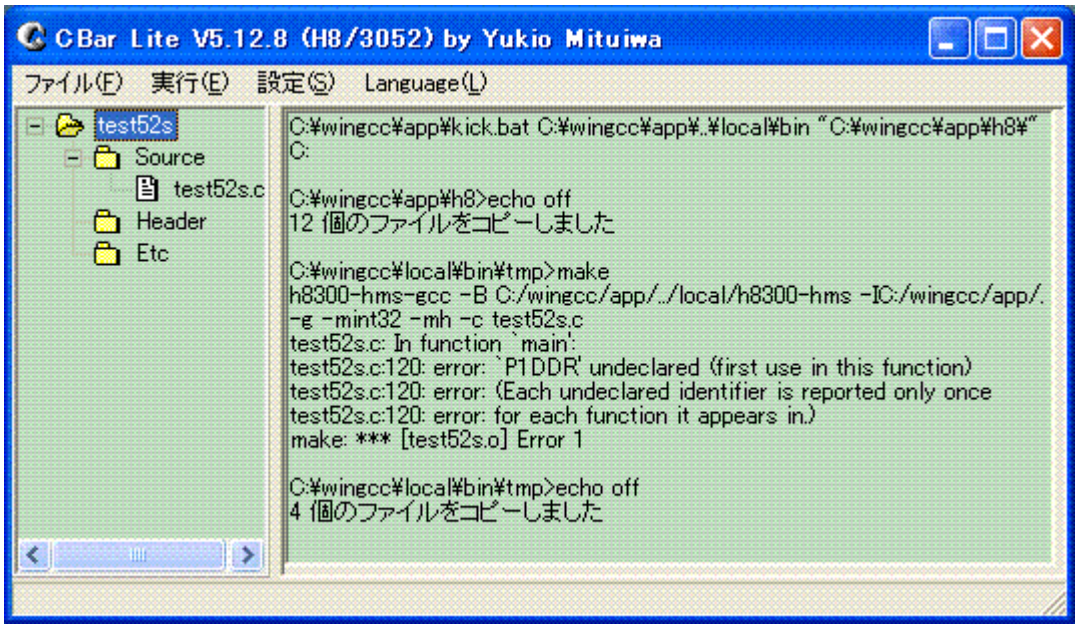
■コンパイルしてみます



問題なければフラッシュＲＯＭに書き込むファイル「ｔｅｓｔ５２ｓ．ｍｏｔ」が作成されています。

test52s.c	8 KB	C ファイル	2006/05/05 12:24
test52s.cba	0 KB	cbar.exe	2006/05/05 11:41
test52s.exe	6 KB	アプリケーション	2006/05/05 12:24
test52s.mot	5 KB	MOT ファイル	2006/05/05 12:24
test52s.o	7 KB	O ファイル	2006/05/05 12:24

エラーがある場合、以下のようにエラーの数とエラー発生行、エラーの種類等が表示されますので、ソースファイルを直し、再びコンパイルしてみてください。「　Ｅｒｒｏｒ　」という文字が出なくなったらＯＫです。



これで完成した「ｔｅｓｔ５２ｓ．ｍｏｔ」ファイルをフラッシュＲＯＭライタで書き込みます。

●フラッシュＲＯＭの書き込み

弊社の「フォースライタ」を使用しますので、別ファイル「フォースライタＨ８－３０５２の使い方」ご参照ください。

ご注意

■ＣｂａｒはＫＥＮＣＨ氏によるフリーソフトです。

■ＧＣＣ（ＧＮＵ　Ｃｏｍｐｉｌｅｒ　Ｃｏｌｌｅｃｔｉｏｎ）Ｃ　はＧＮＵプロジェクトによるフリーＣコンパイラです。

■Ｗｉｎｄｏｗｓは米国マイクロソフト社の登録商標です。

１．本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。

２．本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。

３．本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。

４．本文章は許可なく転載、複製することを堅くお断りいたします。

〒350-1213　埼玉県日高市高萩 1141-1
TEL　042（985）6982　FAX　042（985）6720
Homepage：http//beriver.co.jp
e-mail：support@beriver.co.jp

有限会社ビーリバーエレクトロニクス