

XPMPCK の GameBoy や SEGA MarkIII などの音源フォーマットを作成するために必要な
wla-dx を WindowsXP の Cygwin でコンパイルする手順をまとめてみた資料

How to compile WLA-DX 9.5a / Win32 binaries
(Japanese language)

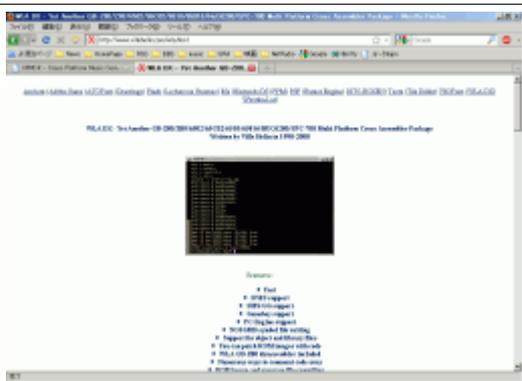
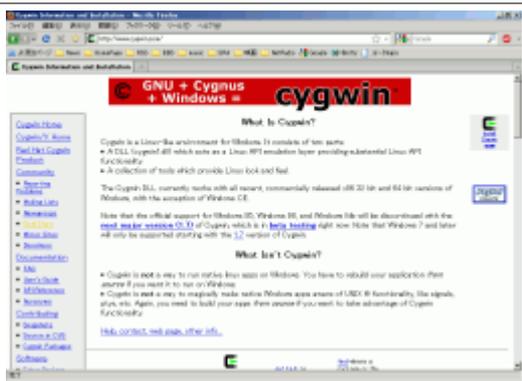
documentation by ROP.
2009-12-22

目次

必要なもの.....	3
cygwinの導入.....	4
wla-dxのコンパイル.....	9
XPMCKで実際にコンパイルしてみる.....	15
最後に.....	18

必要なもの

とりあえず xpmck があればメガドライブ、カプコン CP システムの音源は作れます。
本ドキュメントは、WLA-DX を導入してゲームボーイ、セガマーク III 等の音源も作成できるようにするための手順を記すので WLA-DX と CYGWIN も必要となります。

<h2>XPMCK</h2> <p>http://jiggawatt.org/muzak/xpmck/</p> <ul style="list-style-type: none">・これ単体で VGM を作成することが可能。 WLA-DX を導入することで他音源も作成できるようになる。	
<h2>WLA-DX</h2> <p>http://www.villehelin.com/wla.html</p> <ul style="list-style-type: none">・XPMCK にて GBS/CID/SMS/KSS を作成する場合は各音源用のバイナリが必要。	
<h2>CYGWIN</h2> <p>http://www.cygwin.com/</p> <ul style="list-style-type: none">・WLA-DX の各音源バイナリを生成する為導入する必要がある。 <p>ネット上からバイナリを拾える場合は別に使う必要はない。</p>	

作業の流れ

とりあえず作業の概要を説明します。

1 CYGWIN のインストール

最新の WLA-DX を使用するため、公式サイトからダウンロードすることが出来る最新のソースコードをコンパイルするための環境を整えます。

2 WLA-DX のコンパイル

今回説明するのは Win32 のバイナリを生成するための手順になります。
ソースコードを GCC でコンパイルし、WLA-GB.EXE 等の XPMCK で使用する音源バイナリを生成します。

3 コンパイルエラー時の対処

Win32 バイナリを作ろうとするとエラーするので、エラーメッセージの追いかけ方とソースコードの問題部分を洗い出して修正します。

4 XPMCK でサンプルのコンパイル

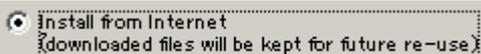
CYGWIN で作成した Win32 バイナリを XPMCK に導入し、実際にサンプルをコンパイルして GBS ファイルを生成します。

cygwin の導入

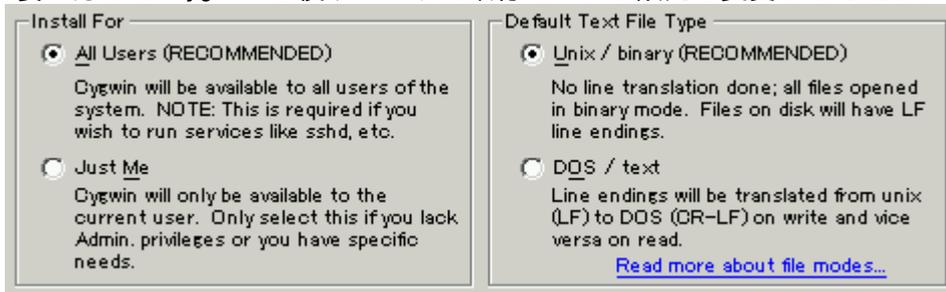
CYGWIN 公式サイトにある setup.exe をダウンロードして実行します。



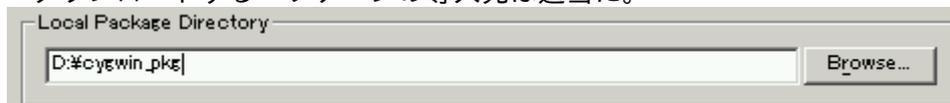
とりあえず internet からダウンロードするように設定



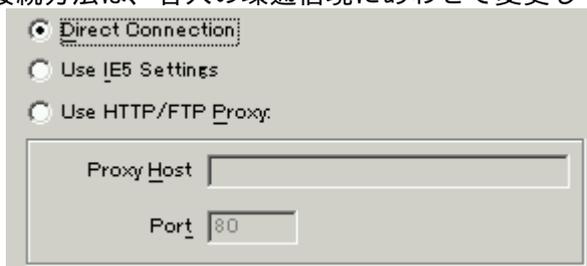
必要に応じて cygwin を使うユーザと改行コードの設定を変更してください。



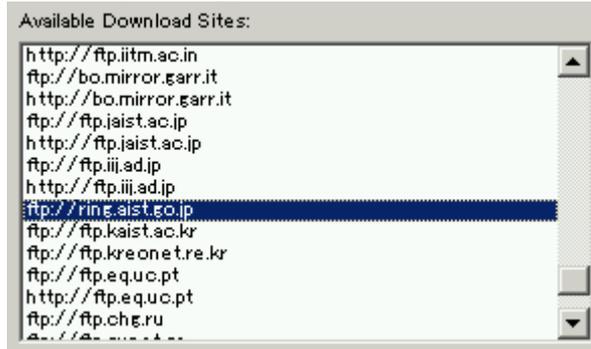
ダウンロードするパッケージの導入先は適当に。



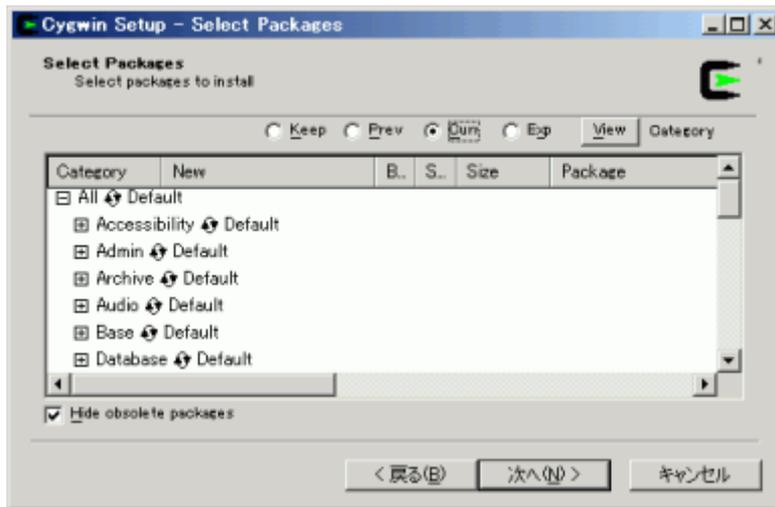
ネットへの接続方法は、各人の環境にあわせて変更してください。



ダウンロード先を指定します。まあ jp などどこでもいいんじゃないかと。



カタログ情報を読み取ると、下のスナップショットのような画面になります。

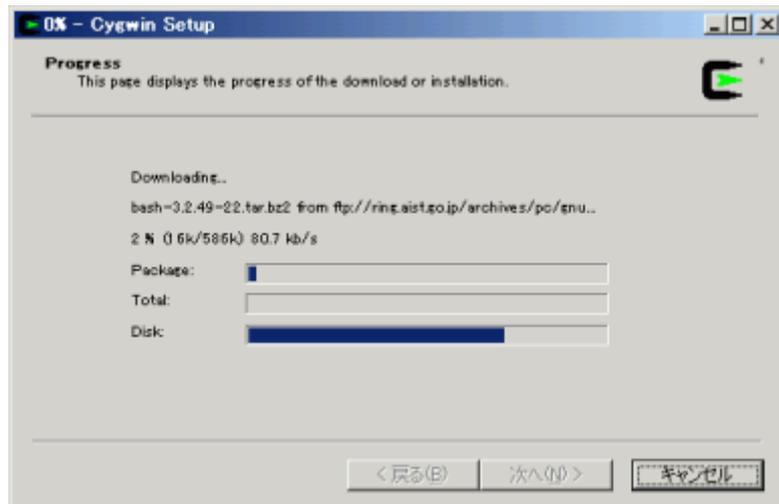


必須なものは以下のもの。

カテゴリ	名称	パッケージ説明
Devel	GCC-Core	GNU C コンパイラ
Devel	GCC-C++	GNU C++コンパイラ
Devel	make	make コマンド

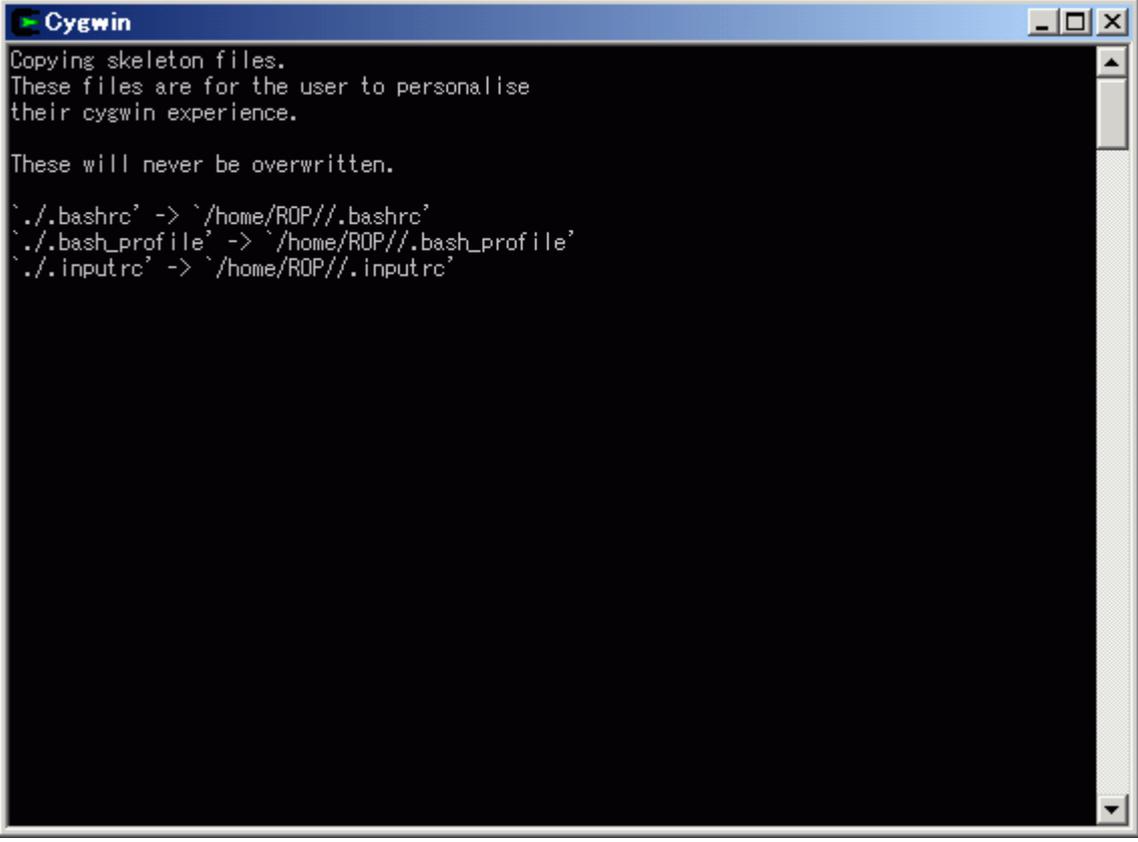
上記パッケージが *skip* になっていたら、クリックしてダウンロードするように変更してください。(私は vim とか more も使いたいのでそれらも入れてます) まあ、後から追加することもできるので上記パッケージの変更して後は必要に応じて追加する形でいいと思います。

必要なパッケージ選択が終わったら、選択したところからダウンロードします。
ダウンロードが完了すると、自動的にインストールが始まります。



これで cygwin のインストール完了です。

CYGWIN を起動すると以下のように初期設定が行われ、使用出来るようになります。



```
Cygwin
Copying skeleton files.
These files are for the user to personalise
their cygwin experience.

These will never be overwritten.

\'.bashrc' -> '/home/ROP/\\.bashrc'
\'.bash_profile' -> '/home/ROP/\\.bash_profile'
\'.inputrc' -> '/home/ROP/\\.inputrc'
```

実際にコンパイラが動くか、コマンドを入力して確認します。

```
$ gcc --version
gcc (GCC) 3.4.4 (cygming special, gdc 0.12, using dmd 0.125)
Copyright (C) 2004 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

GCC のバージョン表示が出たら大丈夫です。

wla-dx のコンパイル

それでは、WLA-DX のパッケージをコンパイルしていきます。

```
$ pwd
/home/ROP

$ ls -l
total 1056
-rwxrwxrwx 1 ROP なし 269144 Dec 21 21:18 wla_dx_9.4.tar.gz
-rwxrwxrwx 1 ROP なし 266208 Dec 21 21:18 wla_dx_9.5a.tar.gz
-rwxrwxrwx 1 ROP なし 544499 Dec 21 21:17 xpmck-22.zip
```

とりあえず、ユーザの home ディレクトリにダウンロードした wla_dx を放り込みます。
このマニュアルを作成時点での最新パッケージ・バージョンは上記のものとなります。

wla_dx パッケージをアンパックします。

```
$ tar xzvf wla_dx_9.5a.tar.gz
:
wla_dx_9.5/icopy/SCOPTIONS
wla_dx_9.5/icopy/makefile.msdos
wla_dx_9.5/icopy/makefile.win32
wla_dx_9.5/icopy/makefile.unix
wla_dx_9.5/icopy/main.c
wla_dx_9.5/icopy/LICENSE
wla_dx_9.5/icopy/defines.h
```

どっさりファイルがアンパックされます。
プロンプトが戻ってきたら、解凍されたパッケージのディレクトリに移動します。

```
$ cd wla_dx_9.5
$ ls
CHANGELOG      defines.h      opcodes_6510_tables.c  pass_2.h
INSTALL        examples      opcodes_65816.c        pass_3.c
LICENSE        icopy         opcodes_65816_tables.c  pass_3.h
README         include_file.c  opcodes_65c02.c        pass_4.c
SCOPTIONS      include_file.h  opcodes_65c02_tables.c  pass_4.h
TODO           listfile.c     opcodes_gb.c           print_opcodes
amiga          listfile.h     opcodes_gb_tables.c    stack.c
amigaos4       main.c         opcodes_huc6280.c      stack.h
binaries       main.c.orig    opcodes_huc6280_tables.c  txt
dasm_opcodes   main.h         opcodes_spc700.c        unix.sh
decode_6502.c  makefile      opcodes_spc700_tables.c  win32
decode_6510.c  makefiles     opcodes_z80.c          win32.orig
decode_65816.c  memorymaps    opcodes_z80_tables.c    win32.rej
decode_65c02.c  msdos.bat     parse.c                 wlab
decode_gb.c    opcode_table_generator  parse.h                 wlad
decode_huc6280.c  opcodes_6502.c  pass_1.c                wlad_new
decode_spc700.c  opcodes_6502_tables.c  pass_1.h                wlalink
decode_z80.c    opcodes_6510.c  pass_2.c
```

とりあえず win32 バイナリを作ってみます。

win32 向けのバイナリは、前ページ一覧の win32 ファイルに作成手順が書かれています。
これを実行すれば自動的に各音源用のバイナリが作成されるようになっているはずですが。

では早速実行してみましょう。

```
$ ./win32
gcc -m486 -c -mno-cygwin -ansi -O3 -pedantic -Wall main.c
gcc -mno-cygwin main.o -o wlab.exe
rm -f main.o *~ wlab.exe
gcc -m486 -c -mno-cygwin -ansi -O3 -pedantic -Wall -DMSDOS -DGB main.c
gcc -mno-cygwin main.o -o wlad.exe
rm -f main.o *~ wlad.exe
rm -f main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o stack.o listfile.o
core *~ wla-huc6280
:
:
```

こんな感じでバイナリのコンパイルが実行されていきます。

しかし、途中でエラーしているようです。

```
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB stack.c
gcc listfile.c
/cygdrive/c/DOCUME~1/ROP/LOCALS~1/Temp/cc20fohd.o:listfile.c:(.text+0x35): undefined reference
to `_gba_tmp_name'
/cygdrive/c/DOCUME~1/ROP/LOCALS~1/Temp/cc20fohd.o:listfile.c:(.text+0x50): undefined reference
to `_gba_tmp_name'
/cygdrive/c/DOCUME~1/ROP/LOCALS~1/Temp/cc20fohd.o:listfile.c:(.text+0x10f): undefined reference
to `_sections_first'
/cygdrive/c/DOCUME~1/ROP/LOCALS~1/Temp/cc20fohd.o:listfile.c:(.text+0x3cc): undefined reference
to `_get_file_name'
/usr/lib/gcc/i686-pc-cygwin/3.4.4/../../../../libcygwin.a(libcmain.o):(.text+0xab): undefined reference
to `_WinMain@16'
collect2: ld returned 1 exit status
make: *** [listfile.o] Error 1
cp: cannot stat `wla-gb.exe': No such file or directory
```

正常にバイナリが作成されると /binaries にコピーされるようになっています。

該当ディレクトリを見ると、やはり各音源向けの exe は出来てません。残念。

```
$ ls -l binaries/
total 208
-rwxr-xr-x 1 ROP なし 47541 Dec 21 21:31 wlab.exe
-rwxr-xr-x 1 ROP なし 63993 Dec 21 21:31 wlad.exe
-rwxr-xr-x 1 ROP なし 95599 Dec 21 21:33 wlalink.exe
```

こうなってしまうと、exe を作る素になっているプログラムを見て、間違ったりエラーを出力しているところを修正していく必要があります。

では、作りこんでいく win32 ファイルを見てみます。
とりあえず頭 20 行ほど表示してみると、こんな記述になっています。

```
$ head -20 win32
#!/bin/sh

cd wlab
cp makefile.win32 makefile
make
cp wlab.exe ../binaries
make clean
cd ..

cd wlad
cp makefile.win32 makefile
make
cp wlad.exe ../binaries
make clean
cd ..

make clean
cp makefiles/makefile.win32.gb makefile
make
cp wla-gb.exe binaries
```

/binaries を見る限り、wlab.exe と wlad.exe は出来ているので、wla-gb.exe を作る部分を
実際に手入力してみます。

```
$ make clean
rm -f main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o stack.o listfile.o
*~ wla-huc6280.exe
$ cp makefiles/makefile.win32.gb makefile
```

ここまではエラーも出ずに進めますね。

```
$ make
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB main.c
:
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB pass_4.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB stack.c
gcc listfile.c
:
collect2: ld returned 1 exit status
make: *** [listfile.o] Error 1
```

あいやー。

このメッセージを見ると、とりあえず main.c から stack.c までは普通に処理されていて、listfile.c
を処理するときにエラーを出しているようだ、と分かります。

それでは作りこんでいる makefile を見てみましょう。

```
$ cat makefile

CC = gcc
CFLAGS = -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB
LD = gcc
LDFLAGS = -mno-cygwin -lkernel32

CFILES = main.c parse.c include_file.c pass_1.c pass_2.c pass_3.c pass_4.c stack.c listfile.c
HFILES = main.h parse.h include_file.h pass_1.h pass_2.h pass_3.h pass_4.h stack.h listfile.h defines.h
OFILES = main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o stack.o listfile.o

all: $(OFILES) makefile
    $(LD) $(LDFLAGS) $(OFILES) -o wla-gb.exe
main.o: main.c defines.h main.h makefile
    $(CC) $(CFLAGS) main.c
parse.o: parse.c defines.h parse.h makefile
    $(CC) $(CFLAGS) parse.c
include_file.o: include_file.c defines.h include_file.h makefile
    $(CC) $(CFLAGS) include_file.c
pass_1.o: pass_1.c defines.h pass_1.h parse.h makefile opcodes_gb.c opcodes_z80.c
    $(CC) $(CFLAGS) pass_1.c
pass_2.o: pass_2.c defines.h pass_2.h makefile
    $(CC) $(CFLAGS) pass_2.c
pass_3.o: pass_3.c defines.h pass_3.h makefile
    $(CC) $(CFLAGS) pass_3.c
pass_4.o: pass_4.c defines.h pass_4.h makefile
    $(CC) $(CFLAGS) pass_4.c
stack.o: stack.c defines.h stack.h makefile
    $(CC) $(CFLAGS) stack.c

listfile.o: listfile.c defines.h makefile
    $(CC) $(WLAFLAGS) listfile.c

$(OFILES): $(HFILES)

clean:
    rm -f $(OFILES) *~ wla-gb.exe
install:
    make ; cp wla-gb.exe /usr/local/bin
```

エラーを出しているのはここ(赤字の部分)みたいです。

で、よく見ると **WLAFLAGS** というパラメータを設定しているようですが、makefile 内にこれを設定している部分がありません。orz

それじゃエラー出るはずだよ。

そんな訳で、とりあえずその上で普通に動いてる CFLAGS を使うように makefile を書き換えます。秀丸エディタとかの unix 改行をちゃんと処理できるテキストエディタ等で、WLAFLAGS と書かれた部分を CFLAGS に変更してください。

修正したら、再度コンパイルします。

```
$ make clean
rm -f main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o stack.o listfile.o
*~ wla-gb.exe
$ make
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB main.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB parse.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB include_file.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB pass_1.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB pass_2.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB pass_3.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB pass_4.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB stack.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS -DWIN32 -DGB listfile.c
gcc -mno-cygwin -lkernel32 main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o
stack.o listfile.o -o wla-gb.exe
$
```

今度はエラー出力されずに終わったようです。
それではバイナリが作成されているか確認してみましょう。

```
$ ls -l *.exe
-rwxr-xr-x 1 ROP なし 203314 Dec 21 21:59 wla-gb.exe
```

作成されてますねえ。
それでは、本当に動くのか試してみましょう。

```
$ ./wla-gb.exe

WLA GB-Z80 Macro Assembler v9.5
Written by Ville Helin in 1998-2008
USAGE: D:%cygwin%home%ROP%wla_dx_9.5%wla-gb.exe -[iMqtvx]{lo} [DEFINITIONS] <ASM FILE> [OUTPUT FILE]
Commands:          Options:
l Library file      i Add list file information
o Object file       M Output makefile rules
                    q Quiet
                    t Test compile
                    v Verbose messages
                    x Extra compile time definitions
```

オプションの説明とか出てきました。
これでwla-gb.exeは一応動くことが確認出来ました。

XPMCK release 22 で使っているのは以下の音源なので、これらのmakefileを修正します。

```
./makefiles/makefile.win32.6510
./makefiles/makefile.win32.gb
./makefiles/makefile.win32.z80
```

これ以外の音源はお好みで修正してやってください。

ファイルを修正したら、もういちどコンパイルしてみましょう。

```
$ ./win32
gcc -m486 -c -mno-cygwin -ansi -O3 -pedantic -Wall main.c
gcc -mno-cygwin main.o -o wlab.exe
rm -f main.o *~ wlab.exe
gcc -m486 -c -mno-cygwin -ansi -O3 -pedantic -Wall -DMSDOS -DGB main.c
gcc -mno-cygwin main.o -o wlad.exe
rm -f main.o *~ wlad.exe
rm -f main.o parse.o include_file.o pass_1.o pass_2.o pass_3.o pass_4.o stack.o listfile.o *~ wla-gb.exe
:
:
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS discard.c
gcc -m486 -c -mno-cygwin -O3 -ansi -pedantic -Wall -DMSDOS listfile.c
gcc -mno-cygwin main.o memory.o parse.o files.o check.o analyze.o write.o compute.o discard.o
listfile.o -o wlalink.exe
rm -f main.o memory.o parse.o files.o check.o analyze.o write.o compute.o discard.o
listfile.o *~ wlalink.exe
```

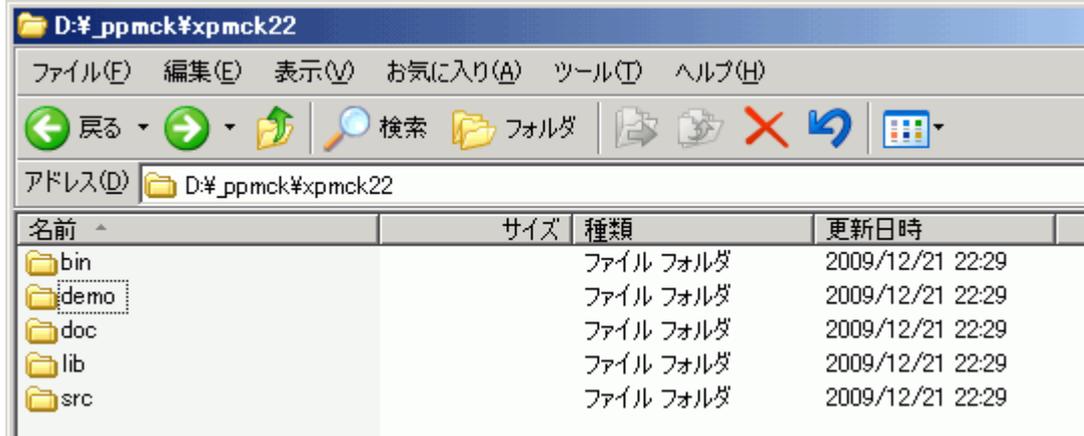
今度はちゃんと作れたようです。

```
$ ls -l binaries/
total 1820
-rwxr-xr-x 1 ROP なし 169984 Dec 21 22:19 wla-6502.exe
-rwxr-xr-x 1 ROP なし 201496 Dec 21 22:20 wla-6510.exe
-rwxr-xr-x 1 ROP なし 225869 Dec 21 22:20 wla-65816.exe
-rwxr-xr-x 1 ROP なし 200984 Dec 21 22:20 wla-65c02.exe
-rwxr-xr-x 1 ROP なし 203314 Dec 21 22:19 wla-gb.exe
-rwxr-xr-x 1 ROP なし 204056 Dec 21 22:20 wla-huc6280.exe
-rwxr-xr-x 1 ROP なし 199840 Dec 21 22:20 wla-spc700.exe
-rwxr-xr-x 1 ROP なし 229195 Dec 21 22:19 wla-z80.exe
-rwxr-xr-x 1 ROP なし 47541 Dec 21 22:19 wlab.exe
-rwxr-xr-x 1 ROP なし 63993 Dec 21 22:19 wlad.exe
-rwxr-xr-x 1 ROP なし 95599 Dec 21 22:21 wlalink.exe
```

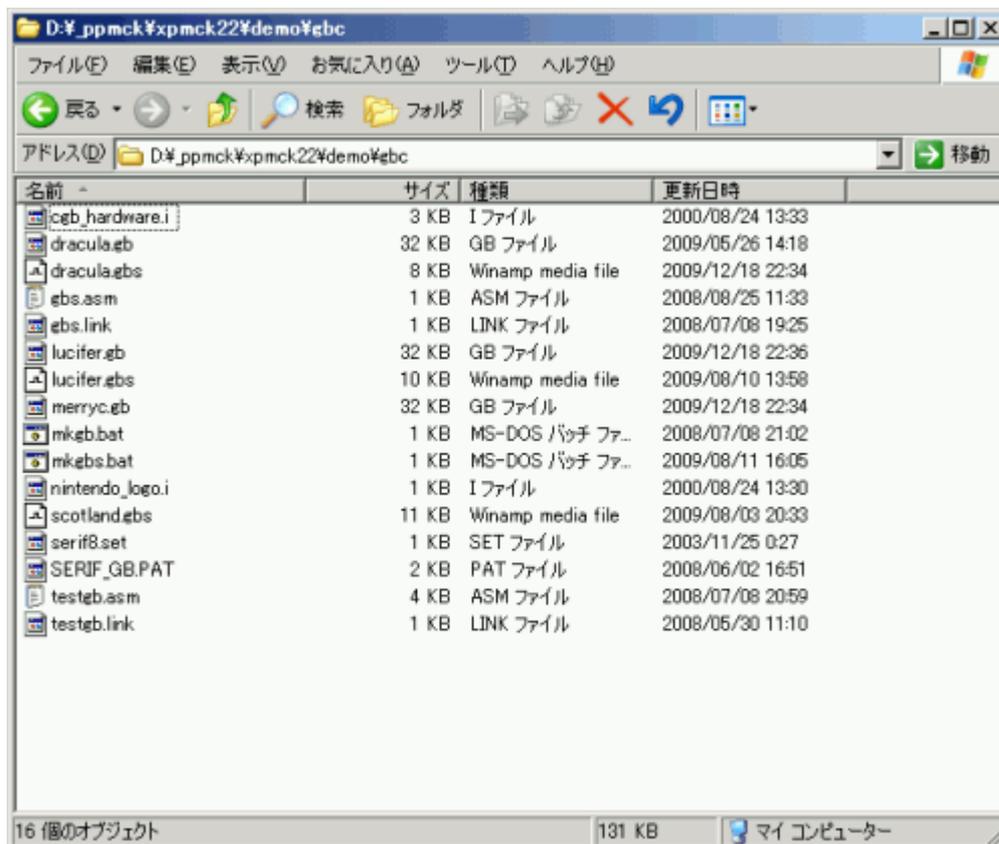
※ちなみに wla-6502 は WLAFLAGS の設定がされているので、listfile.o のところはそのまま。
しかし、CC , LC , CFLAGS の後ろに「?」が書かれていてコンパイルできないので、この ?
を削除することでコンパイル通りしました。

XPMCKで実際にコンパイルしてみる

XPMCK を適当なフォルダに解凍します。



とりあえず GameBoy 音源(GBS)の作成をしてみるため、demo ¥ gbc フォルダに移動します。

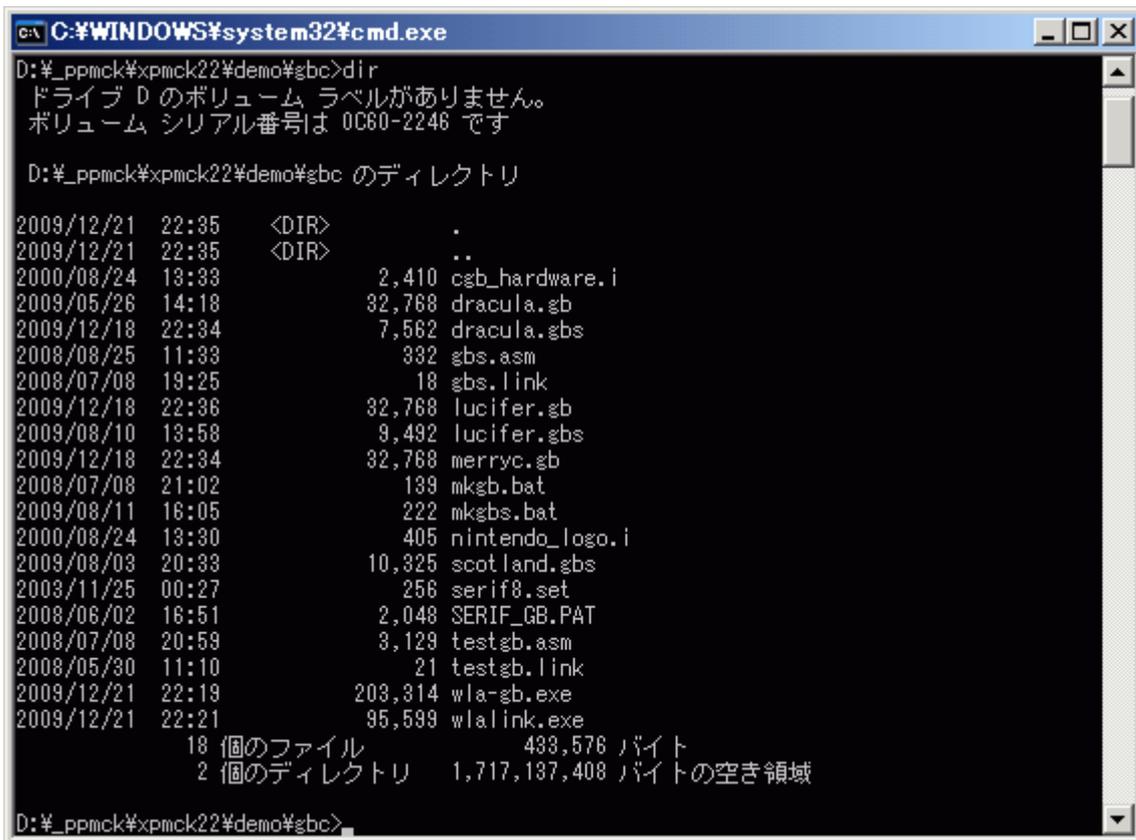
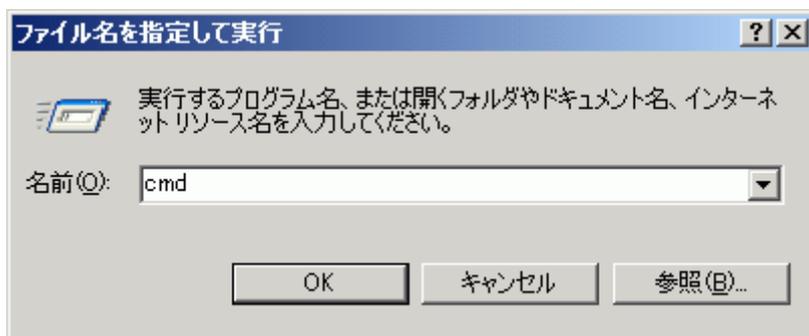


ゲームボーイ音源を作成するために使用する mkgbs.bat を見てみます。

```
..%*.%bin%xpmc.exe -gbc ..%mml%1.mml gbmusic.asm  
wla-gb -o gbs.asm gbs.o  
wlalink -b gbs.link gbs.bin  
wla-gb -o -DXPMP_MAKE_GBS gbmusic.asm gbs.o  
wlalink -vb gbs.link %1.gbs  
del gbs.o  
del gbs.bin  
del gbmusic.asm
```

wla-gb と wlalink が必要になるので、%demo%gbc フォルダにバイナリをコピーします。

DOS プロンプトを開き、サンプルの Dolacura.mml をコンパイルさせてみます。



コンパイルするには以下のコマンドを実行します。

```
D:¥_ppmck¥xpmck22¥demo¥gbc>mkgbs ..¥mml¥dracula
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>..¥.¥bin¥xpmc.exe -gbc ..¥mml¥.¥mml¥dracula.mml gbmusic.asm
gbmusic, Warning: Mismatch in length between channels in song 1
Song 1, Channel A: 73 bytes, 128 / 128 ticks
Song 1, Channel B: 17 bytes, 128 / 128 ticks
Song 1, Channel C: 15 bytes, 32 / 32 ticks
Song 1, Channel D: 15 bytes, 32 / 32 ticks
```

Done.

```
D:¥_ppmck¥xpmck22¥demo¥gbc>wla-gb -o gbs.asm gbs.o
MEM_INSERT: 1. write into $0148 (old: $00, new: $7d).
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>wlalink -b gbs.link gbs.bin
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>wla-gb -o -DXPMP_MAKE_GBS gbmusic.asm gbs.o
MEM_INSERT: 1. write into $0148 (old: $00, new: $20).
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>wlalink -vb gbs.link ..¥mml¥dracula.gbs
Free space at $1d8a-$7fff.
Bank 00 has 08822 bytes (53.85%) free.
Bank 01 has 16384 bytes (100.0%) free.
0 unused bytes (0.00%) of total 7562.
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>del gbs.o
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>del gbs.bin
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>del gbmusic.asm
```

```
D:¥_ppmck¥xpmck22¥demo¥gbc>
```

エラーせずに終了したら、 mml フォルダに Dracula.gbs が作成されているはずです。

```
D:¥_ppmck¥xpmck22¥demo¥gbc>dir ..¥mml¥*.gbs
```

ドライブ D のボリューム ラベルがありません。
ボリューム シリアル番号は 0C60-2246 です

D:¥_ppmck¥xpmck22¥demo¥mml のディレクトリ

```
2009/12/21  22:41                7,562 dracula.gbs
             1 個のファイル                7,562 バイト
             0 個のディレクトリ  1,717,129,216 バイトの空き領域
```

出来上がった gbs を演奏させてみます。



最後に

2008/10/20

自分のblogに作成メモと称して、ソースの変更点をまとめて記事にする。
このとき XPMCK release 10、WLA-DX は 9.5a

2009/12/19

ピコピコ忘年会に出席した際に、上記記事が為になったと感謝される。
帰宅後、該当記事を参照して本当にメモだなあと思い直す。

2009/12/21

最新のアーカイブをダウンロード、自宅PCのCYGWIN環境をいじりすぎてgdgdになっていたので完全アンインストールから入れなおす。
この段階からスナップを採取しつつ、OpenOffice.orgで資料化する。
最低限、資料として使える形になったので公開。

2009/12/22

もうちょっと見栄えを整えて、ないしょでファイルを差し替える。