Getting Start GCC with UBUNTU

【このドキュメントについて】

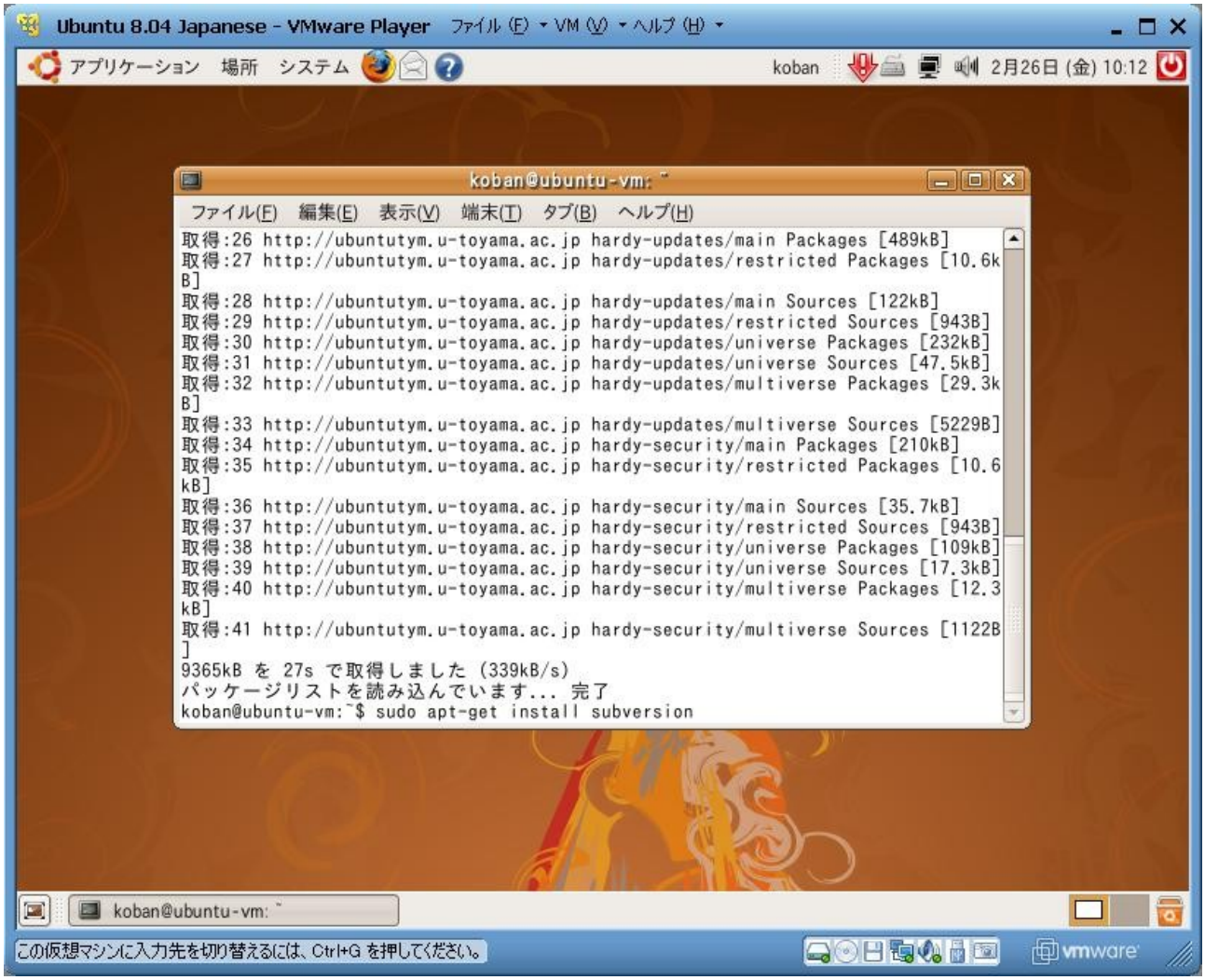
このドキュメントは GCC を使用時の SDK のスタート方法を示しています。
VMware による UBUNTU の起動で手軽に環境を整えられます。VMware で
UBUNTU 起動から、30 分程度で環境が構築できます。

このドキュメントは以下の事項について説明しています。
・VMWare ＋UBUNTU のインストール
・UBUNTU での必要なツールのインストール
・ビルド方法
・プログラム転送方法

※デバッグについては言及していません。今後言及する予定です。

基本的な UNIX のコマンドの知識があることを前提に書いています。

Vmware + UBUNTU の画面



【VMware + UBUNTU について】

GCC で開発をするには基本は Linux での開発になります。お好きな Linux Distribution をインストールし開発を行うことができますが、ここでは VMware + UBUNTU で開始する手法を示しています。VMware を使用すれば Windows 上から UBUNTU をシステムを汚すことなく起動できます。最大のメリットはドライバーがしっかりしているということです。ネットワークやシリアル等は Windows で動作していればそのまま VMware + UBUNTU でも使えます。

UBUNTU は 2010 年 2 月 26 日現在、VMware 用配布版がある UBUNTU 8.04LTS で動作確認しています。

【用意するもの】

このドキュメントの流れに沿ってスタートするために用意しなければならないものは以下のものです。

- 開発 PC　(Windows XP SP3 にて動作確認)。
- KOBANZAME Board　※１
- USB Serial (バッファローコクヨサプライ SRC06USB＋オスメス変換アダプターで動作確認)※２

開発 PC は筆者は Atom1.6GHz の Netbook で動作させていますが、ストレスなく動作しています。PC を別にすればかなり安価に環境が整います。

※１　KOBANZAME Board は UBOOT がすでに書かれていることを前提にしています。
※２　PC 付属の RS232C でも動作しますが、若干設定を変えなければなりません。

【手順１　VMware + UBUNTU のインストール】

こちらのサイトを参考にダウンロードとインストールを行ってください。WMware Player はユーザー登録が必要です。
http://www.ubuntulinux.jp/products/JA-Localized/vmware

ファイルは説明にもあるとおり
ubuntu-ja-8.04-vmware-i386.zip
をダウンロードし、適当なディレクトリに展開してください。
必ずパッケージのアップデートと追加の項目を実行してください。
現状のところ、コンソールで以下のコマンドを打てばよいはずです。

```
$ sudo apt-get update
```

【手順２　**Subversion** のインストール】

Sorceforge からソースコードを取得ために Subversion をインストールします。
UBUNTU のコマンドライン上の適当なディレクトリで以下のコマンドを実行してください。

```
$ sudo apt-get install subversion
```

もし失敗した場合はネットワークが切断されているか、apt-get がアップデートされていません。apt-get をアップデートするには手順 1 のサイトのパッケージのアップデートと追加の項目を実行してください。

【手順３　ソースファイルの取得】

Sourceforge から KOBANZAME SDK のソースファイルを取得します。適当なワーキングディレクトリを作成し、そのディレクトリ上から以下のコマンドを実行してください。最新ソースコードが取得できます。

```
$ svn export http://svn.sourceforge.jp/svnroot/kobanzame-sdk/trunk
```

【手順４　**Blackfin Tool Chain** など をインストール】

ビルドに必要なツールをインストールします。このスクリプトを実行すると以下のツールが自動でインストールされます。完了までにしばらく時間がかかります。
（途中 sudo のパスワードやインストール実行の選択などのユーザー入力が一部あります)

・gcc tool-chain for Blackfin 2008R1.5 (最新版での動作確認はまだしていません)
・nkf (Shift JIS -> UTF8 変換に使用します)
・ckermit (Serial 転送プログラム)

```
$ cd trunk/maketool/gcc
$ chmod a+x install.script
$ ./install.script
```

【手順５　ビルド】

5-1 premake.script 実行

まずは、KOBANZAME SDK のコードは Shift-JIS でエンコードされていますので、UTF8 に変換します。以下のコマンドを trunk/maketool/gcc 上で実行します。実行に

数分かかります。またスクリプト系のプログラムの実行権を変更しています。

```
$ chmod a+x premake.script
$ ./premake.script
```

※ premake.script は毎回のビルドで必要ではありません。

5-2 configuration exe の make

TOPPERS JSP で必要な、タスクや資源などを生成する静的コンフィギュレータのビルドを行います。以下のコマンドを　trunk/jsp/cfg 上で実行します。エラーの文字が見えなければ成功しました。

```
$ make depend
$ make
```

5-3 SDK Projecrt の make

いよいよ、SDK の MAIN プログラムをビルドします。以下のコマンドを trunk/projects/main/sdkproject 上で実行します。

```
$ make depend
$ make
```

出力ファイルに　jsp.dxe が生成されれば成功しました。make depend に失敗した場合は、ツールのパスが反映されていない可能性があります。コンソールを一旦終了し、あたらにコンソールを立ち上げなおし、再度実行してください。

【手順６　プログラムの転送】

jsp.dxe ができたところで、kermit を用いて KOBANZAME Board にプログラムの転送を行い実行します。install.srcpt によりデフォルトでは USB Serial が設定されています。他のデバイスを選びたい場合は、~/.kermrc を適宜編集してください。

※KOBANZAME に UBOOT が書き込まれていることが前提条件です。

trunk/projects/main/sdkproject 上で以下のコマンドを入力し kermit を立ち上げます。

```
$ kermit
C-Kermit 8.0.211, 10 Apr 2004, for Linux
 Copyright (C) 1985, 2004,
  Trustees of Clumbia University in the City of New York.
Type ? or HELP for help.
```

```
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>
```

kermit はコマンドモードで起動されています。 connect (or c) と実行するとシリアルモードになります。c を入力した後、KOBANZAME Board をリセットをかけ、何かのキーを押し、Auto Boot をスキップした状態にします。

```
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit> connect
Connecting to /dev/ttyUSB0, speed 57600
 :
 : 省略
 :
U-Boot 1.1.6 (ADI-2008R1.5) (Dec 22 2008 – 22:57:37)

CPU:       ADSP bf533-0.4 (Detected Rev: 0.4 )
:
: 省略
:
Hit any key to stop autoboot:    0
bfin>
```

KOBANZAME SDK のメインプロジェクトは Cache を使用しないので、以下のコマンドを入力しあらかじめ Cache を OFF にします

```
bfin> icache off
Instruction Cache is OFF
bfin> dcache off
Data  (writethrough) Cache is OFF
```

KOBANZAME Board をデータ受信 Ready 状態にします。

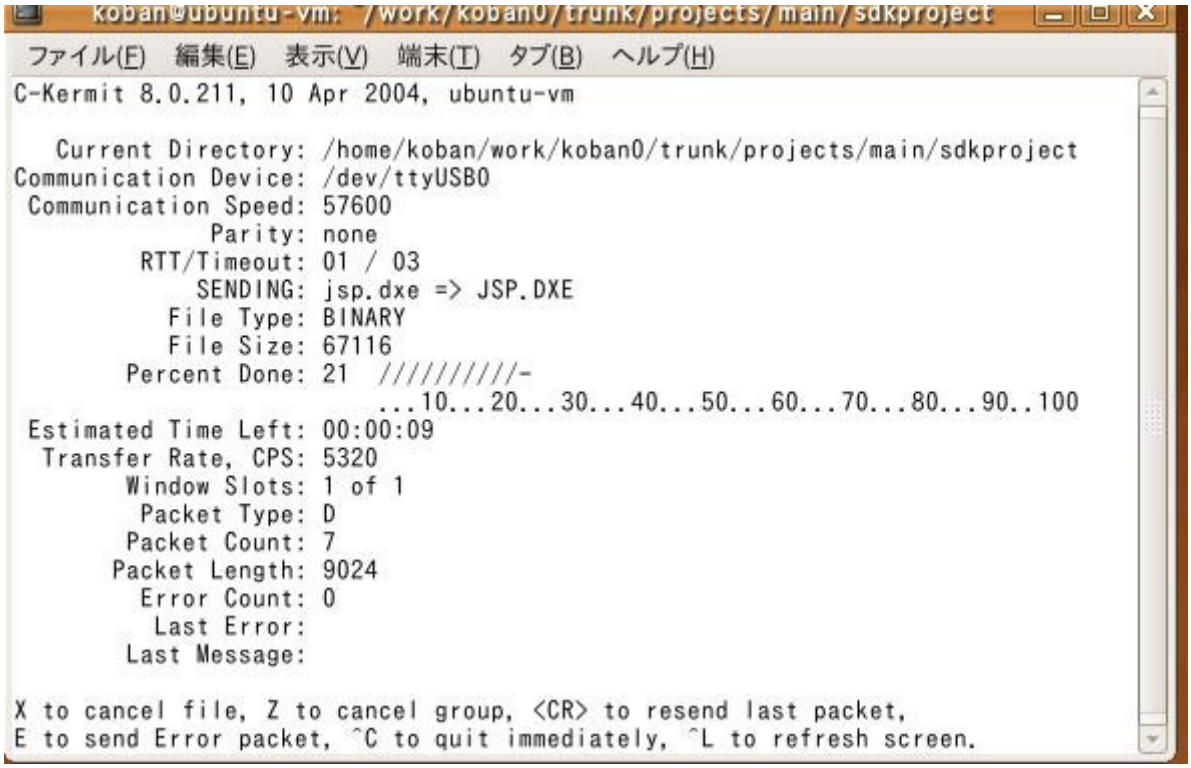
```
bfin> loadb
## Ready for binary (kermit) download to 0x01000000 at 57600 bps...
```

ここで、コマンドモードに一旦戻ります。コマンドモードに戻るには　CTRL-\の後に CTRL-c を入力します。

その後、転送エラーを避けるために robus 命令を入れ、send 命令でプログラムを送信します。

```
(Back at ubuntu-vm)
------------------------------------------------
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>robust
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>send jsp.dxe
```

この命令のあとは以下のような画面が現れます。



転送後は　connect (or c) でシリアルモードに戻り、bootelf でプログラムを起動します。KOBANZAME Board に SD-Card を挿入した状態で起動すると以下のようなメッセージが表れます。

```
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>robust
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>send jsp.dxe
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>connect
Connecting to /dev/ttyUSB0, speed 56700
 Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or following by ? to see other options.
---------------------------------------------------------
## Total Size       = 0x0001062C = 67116 Bytes
## Start Addr       = 0x01000000
bfin> bootelf
Loading .init @ 0xffa00000 (272 bytes)
Loading .text @ 0xffa00110 (49640 bytres)
Loading .rodate @ 0xff800000 (7056 bytes)
Loading .data @ 0xff801b90 (2548 bytes)
Loading .bss @ 0xff900000 (30488 bytes)
Loading .l1code @ 0xffa0c2f8 (448 bytes)
## Starting applications at 0xffa00000 ...
System logging task is started on port 1.
Start KOBANZAME SDK Sample Project
KOBANZAME SDK Version      : 1.40(0008)
                  Build Date  : Unkonwn build date
                  Copyright   : KOBANZAME SDK Project
```

```
SD-Card Ready, Filesystem start
/>
```

help と入力すると、コマンド一覧が表示されます。

以上で、GCC のインストールからビルド、プログラムの実行までの一連の流れを説明しました。

【手順 **7**　プログラムの焼きこみ】

KOBANZAME Board には 2MByte の Serial Flash がついています。これにプログラムを焼きこみ、ブート時に自動でそのプログラムを実行する方法を示します。手順 6 の手順でプログラムをロードします。ただし、手順 6 最後の **bootelf** は実行しないでください。下の状態まで実行してください。

```
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>robust
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>send jsp.dxe
(/home/koban/work/koban0/trunk/projects/main/sdkproject/) C-Kermit>connect
Connecting to /dev/ttyUSB0, speed 56700
 Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or following by ? to see other options.
---------------------------------------------------------------------------
## Total Size       = 0x0001062C = 67116 Bytes
## Start Addr       = 0x01000000
bfin>
```

EEPROM にプログラムを書き込みます。

```
bfin> eeprom write ${loadaddr} 0x100000 ${filesize}

EEPROM @0x0 write: addr 01000000  off 100000  count 67116 ... ....done
bfin>
```

printenv コマンドを使用して環境変数の filesize を控えておきます。

```
bfin> printenv
    : (省略)
stdin=serial
stdout=serial
stderr=serial
filesize=1062C
```

ブートコマンドを書き込みます。

```
bfin> setenv 'icache off;dcache off;eeprom read ${loadaddr} 0x100000
0x1062C;bootelf ${loadaddr}'
bfin> saveenv
..done
bfin>
```

eeprom read の第三パラメータは read する filesize を指定していますが、多めの値を書いておいたほうがよいかもしれません。いちいちビルドの度に設定する必要がなくなるからです。ここではジャストの値を書いています。ビルドするバージョンによって変わるので、確認が必要です。

その後リセットすると、自動的に SDK プロジェクトが立ち上がるようになります。

最後に

この文章は Blackfin 空挺団を参考にしながら作成しています。
http://blackfin.s36.coreserver.jp/
特に　”開発ツール”のコンテンツは非常に有益な情報が詰まっております。この HP を作成された酔漢師に末筆ながら感謝とお礼を申し上げます。

【変更履歴】

2010/2/26 ... 新規作成(koban)