TOPPERS/ASP for LPCXpresso ユーザーズ・マニュアル

LPC1700 依存部 1.7.~~0~~1 対応版 ~~17-Jul~~09-Sep-2011

TOPPERS/ASP for LPC プロジェクト

TOPPERS/ASP for LPCXpresso ユーザーズ・マニュアル

1 はじめに

TOPPER/ASP は、TOPPERS プロジェクトがリリースしているフリーの uITRON4.0 準拠 RTOS です。

RTOS としては別段とんがっているわけではありませんが、uITRON はそこそこ関連文書が入手しやすい上に仕様が枯れているので安心して使う事が出来ます。TPPERS/ASP カーネルは、TOPPERS プロジェクトがリリースしている最新のカーネルで、TOPPERS/JSP カーネルの後継としてアップデートが行われています。今後、TOPPERS プロジェクトは多くの成果物を ASP カーネルを土台として開発すると宣言しています。

この文書は TOPPERS/ASP カーネルの LPCXpresso 176x 依存部について、LPCXpresso IDE による利用の方法を説明します。このプロジェクトの成果物の最新版は TOPPERS/ASP for LPC プロジェクトから入手できます。

説明に使用する開発環境の OS は Ubuntu 10.04 LTS です。

1.1 TOPPERS/ASP for LPC について

TOPPERS/ASP for LPCXpresso は、TOPPERS/ASP for LPC プロジェクトの成果物です。

TOPPERS/ASP for LPC は TOPPERS/ASP を NXP 社の CORTEX-M3 マイコン、LPC176x に移植したもので、TOPPERS プロジェクトによらない非公式の配布です。ライセンスは TOPPERS ライセンスですので商用、非商用の区別無く自由につかえ、GPL2 コードと同時に使う事もできます。

LPC1768 依存部は 2010 年 6 月に最初の検討が始まり、ターゲット、開発環境、ツールチェーン、ベースとなるソースコードの検討を経て同 9 月に実機上で最初の動作を開始しました。このときは、TOPPERS/ASP 1.3.0 STM32F103 依存部を元にして移植を行い、その後、1.6.0 に対応しました。

その後、ターゲット非依存部 1.7.0 に続いて、TOPPERS プロジェクトより CORTEX-M3 依存部 1.7.0 が公開されました。当プロジェクトでは LPC1768 依存部 1.7.1 より、この公式 CORTEX-M3 依存部を取り入れて使用しています。

最初に対応したハードウェアは日新テクニカ社の LPC1768 評価ボードです。その後、この依存部は、lpc1768_generic_gcc と名前を変えて LPC1768 汎用依存部としてリリースされています。そしてさらに、LPCXpresso 176x を意識した lpcxpresso1768_gcc が追加されました。

開発環境は GCC です。TOPPERS/ASP　for LPC プロジェクトは Codesourcery G++ Lite の arm-none-eabi-gcc をコンパイラとして使用しています。また、LPCXpresso を使用する場合には IDE が持っている arm-none-eabi-gcc を使用します。

1.2 TOPPERS/JSP for LPC プロジェクトについて

プロジェクト・ページは Sourceforge によってホスティングされています。

- http://sourceforge.jp/projects/toppersasp4lpc

このプロジェクトではリリース毎に TOPPERS/ASP for LPC のソースコードをパッケージ化してアップロードしています。また、最新の非リリース・コードを取得したければ、SVN から入手することも出来ます。

掲示板もあるので、質問があればそちらで問い合わせることが可能です。

1.3 ライセンス及び責任の放棄

TOPPERS/ASP for LPC カーネルは TOPPERS ライセンスに従って配布されています。同ライセンスについては TOPPERS プロジェクトを参照してください。基本的に商用、非商用を問わず、利用料やソースコード開示を求めな

いライセンスです。

また、このプロジェクトでは一部 ARM/NXP 社の CMSIS ライブラリを使用しています。CMSIS の利用および再配布については、ARM/NXP のライセンスに従ってください。

TOPPERS/ASP for LPC プロジェクトの成果物を使った結果については、プロジェクト関係者あるいはその他の誰も、金銭的、道義的責任を負いません。またサポート義務も負いません。いかなる損害に対しても保証、謝罪をいたしません。

2 開発環境のインストール

2.1 解説する環境

この文書では

- Ubuntu 10.04
- LPCXpresso for Linux v3.8.2 Build 129 (CodeRed)
- TOPPERS/ASP ターゲット非依存部 1.7.0
- TOPPERS/ASP LPCXpresso 依存部 1.7.1~~0~~

を利用した開発環境の構築を説明します。ホストコンピュータは物理マシンの他 VMware Workstation 7.0 上の仮想マシンでも LPC-Link によるターゲットのデバッグが可能であることを確認しています。

2.2 ターゲット・ハードウェア

ターゲット・ハードウェアは LPCXpresso 1769(写真 1)です。

このボードは、2010 年に販売されていた LPCXpresso 1768 の後継基板で、細かい改良がなされているものの、マイコンが LPC1768 から LPC1769 に変更された以外は同じです。LPC1769 は LPC1768 の高速版であり、周波数が 100MHz から 120MHz に向上したことを除けば、メモリ構成やペリフェラル構成、ピン配置まで一緒です。結局、LPCXpresso1769 は LPCXpresso1768 の高速版として使えます。

LPCXpresso 1769 はほぼ中央で二分割して使う事も出来ます。この場合 USB コネクタが接続されている側は、LPC-Link と呼ばれるデバッガ・ボードとして働きます。残りの部分がターゲットとなるマイコンボードです。

ピンは 2.54mm ピッチ一列のものが二カ所からでており、ブレッドボードに刺して使う事が出来ます。

LPCXpresso1769 は比較的入手が容易です。秋葉原のショップで販売されているほか通販で購入する事も出来ます。価格はだいたい 3000 円ほどです。

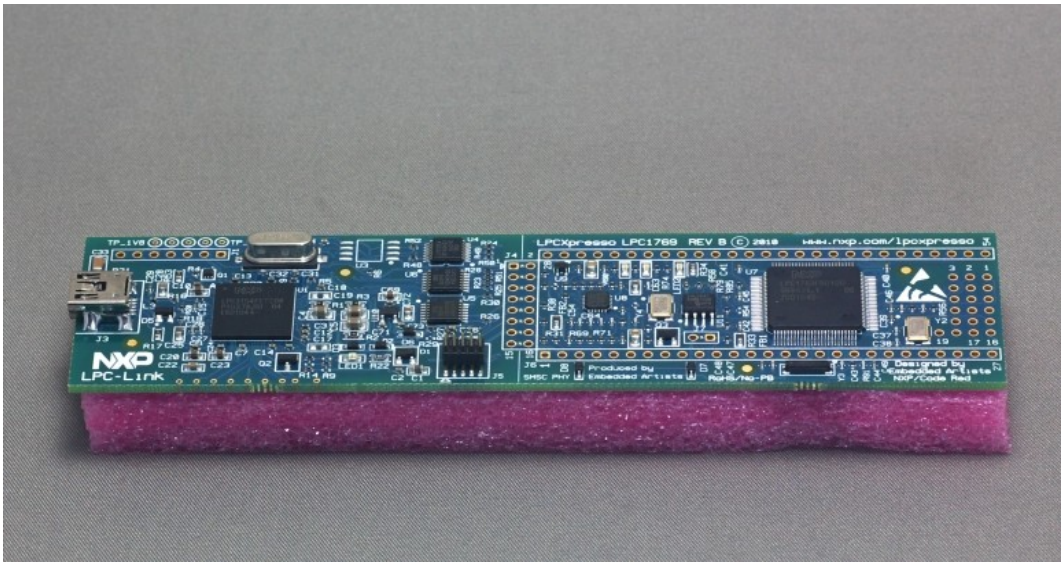

写真 1 LPCXpresso 1769 基板

2.3 Linux の取得と設定

開発環境には Linux のディストリビューションの一つ、Ubuntu 10.04 を使います。

Ubuntu 10.04 はこのドキュメントの執筆段階でリリースからすでに 1 年が経過しており、最新とは言えません。しかし、この版は Ubuntu プロジェクトが LTS(Long Term Support)と呼ぶ長期補償版であり、リリースから 3 年間セキュリティ・アップデートなどを行う事が約束されています。特段の理由が無い限り、ソフトウェア開発には LTS 版を使う方がいいでしょう。

Ubuntu 10.04 のインストール CD イメージは、以下のページから取得出来ます。英語版ですが、取得した Ubuntu はインストール時に指定すれば日本語環境として使用できます。

CD イメージを入手したら CD に焼きます。この CD からブートすることで実マシンへ Ubuntu をインストールすることができます。

- http://releases.ubuntu.com/jaunty/

日本語 Remix 版の方がいいという方は、そちらを試してみるのも面白いでしょう。

- http://www.ubuntulinux.jp/products/JA-Localized/download

実マシンに Ubuntu を入れることに躊躇するならば、VMware Player をお奨めします。VMware 上でも USB ICE を使えることを確認しています[1]。VMware Player 3.0 からは仮想マシンを作れるようになりましたので、インストールは簡単です。

- http://www.vmware.com/jp/products/player/

VMware Player 上に Ubuntu をインストールする方法は、ネットに沢山解説がありますのでそちらを参照してください。一つの例として以下のページを紹介しておきます。

- http://d.hatena.ne.jp/suikan+embedded/20100620/1277043924

なお、VMware Player にインストールするのであれば、CD イメージを CD に焼く必要はありません。VMware Player はイメージ・ファイルを CD としてマウントすることが出来ますので、ダウンロードしたイメージから直接仮想マシンを起動出来ます。

2.4 開発環境の構築

TOPPERS/ASP は、アプリケーションのビルドの前に、TOPPERS/ASP 自身のコンフィギュレータのビルドを必要とします。コンフィギュレータは今回の開発環境では、Ubuntu 10.04 上で動作するコマンド・ライン・アプリケーションとなります。

Ubuntu 10.04 は、標準インストール状態ではコンフィギュレータをビルドできません。そこで、ビルドの準備を整えておきます。ビルドの準備は以下のコマンドを適当なファイルにコピーして、一度だけコマンド・ラインから実行すれば完了です。パスワードを求められたら入力してください。

[1] 確認したのは VMWare Player ではなく、VMware Workstation 7.0

```
sudo apt-get install doxygen g++ ckermit libboost-all-dev lv
echo set line /dev/ttyUSB0 > ~/.kermrc
echo set speed 57600 >> ~/.kermrc
echo set parity none >> ~/.kermrc
echo set flow-control none >> ~/.kermrc
echo set carrier-watch off >> ~/.kermrc
```

上記のスクリプトを実行すると、コンフィギュレータのビルドに必要なツールに加えて、kermit の実行に必要な設定も追加されます。

なお、この kermit の設定は /dev/ttyUSB0 に接続されたポートを使用するためのものです。別のポートを使用する場合は、手作業で ~/.kermrc を編集し直してください。

2.5 Libftd2xx の取得とインストール

LPCXpresso IDE は、2010 年 5 月時点で Linux 用の libftdi ライブラリと衝突し、使用中の USB シリアル変換アダプタを強制切断する問題を持っています。

この問題は FTDI 社のチップを利用したシリアル変換アダプタでのみ発生します。LPCXpresso IDE の製造もとである CodeRed 社は、これを libftdi ドライバの問題であるとしています。

この問題を回避するには、FTDI 社が配布している Linux 用の libftd2xx ドライバを使用します。Libftd2xx ドライバの入手とインストールに関しては、FTDI 社の情報を参照してください。

- http://www.ftdichip.com/Drivers/D2XX.htm

libftd2xx ドライバを使う場合も、USB 変換アダプタは /dev/ttyUSBx (x は整数) と名前が付けられます。最初の変換アダプタの名前は /dev/ttyUSB0 になります。

2.6 LPCXpresso IDE の取得とインストール

統合開発環境である LPCXpresso IDE は、CodeRed の WEB サイトからダウンロードします。この文書で解説しているのは LPCXpresso for Linux v3.8.2 Build 129 です。

- http://lpcxpresso.code-red-tech.com/LPCXpresso/

ダウンロードには、アカウント作成が必須です。ダウンロード後の使い方に関しては、NXP のドキュメントを参照してください。

- http://ics.nxp.com/support/documents/microcontrollers/pdf/lpcxpresso.getting.started.pdf

2.7 Eclox のインストール

Eclox は、Doxygen を Eclipse から使用するためのプラグインです。

インストールすると、Doxygen コメントのコンパイルの他、設定ファイルのウィザード、エラーメッセージからエラー箇所へのジャンプなどの機能が追加されます。

CMSIS ライブラリや TOPPERS/ASP for LPC は Doxygen 文書を活用しているため、Eclox を LPCXpresso IDE にインストールしておくことをおすすめします。

インストールは以下の手順で行います。

まず IDE のメインメニューから、Help->Install New software… を実行します。次に現れたダイアログの Work

with フィールドに http://home.gna.org/eclox/#update-site を入力してエンター・キーを押します。うまくいかないようなら Add ボタン押して入力します（図 1）。

しばらくすると、Name 欄の下が Pending…から Eclox に変化しますので、変化したら、チェックボックスをチェックして、インストールを進めます。あとは、指示に従って作業すれば Eclox のインストールは終了します。

Doxygen や Eclox の使い方に関しては書籍やネット上の情報を参考にしてください。

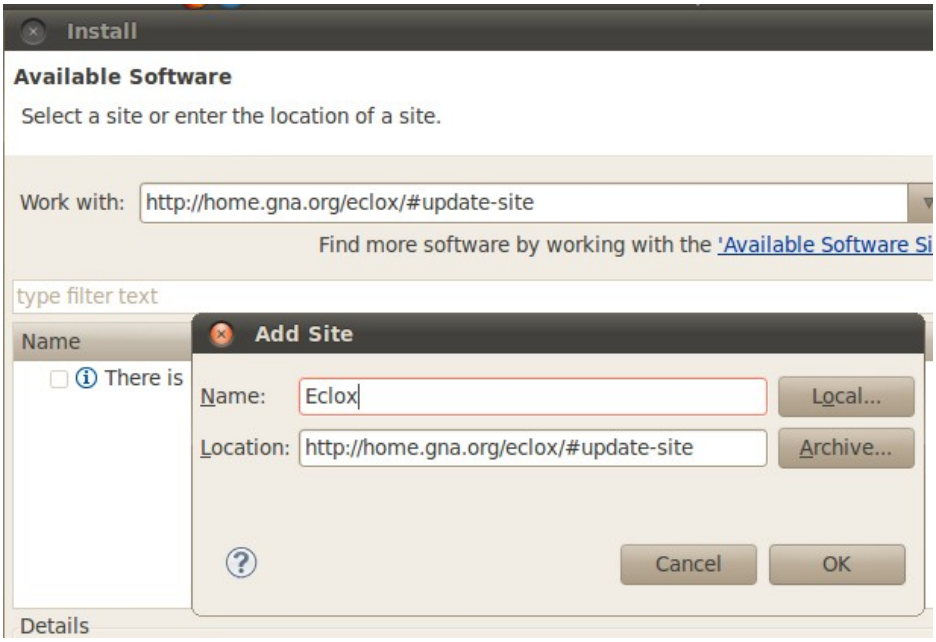

図 1 Eclox のインストール

2.8 Subversive のインストール

SVN を使いたい方は、Subversive をインストールするとよいでしょう。

Subversive は、Eclipse 用の SVN クライアントで、SVN サーバー上のプロジェクトとの同期のほか、SVN サーバーのブラウズ、バージョン間の比較などの機能が提供されます。

インストールは以下の手順で行います。

まず IDE のメインメニューから、Help->Install New software… を実行します。次に現れたダイアログの Work with フィールドに http://download.eclipse.org/releases/galileo/ を入力してエンター・キーを押します。うまくいかないようなら Add ボタン押して入力します。

しばらくすると、Name 欄の下が Pending…からソフトウェアの一覧に変化しますので、Collaboration を展開して、”Subversive SVN Team Provider” を選択してインストールします（図 2）。

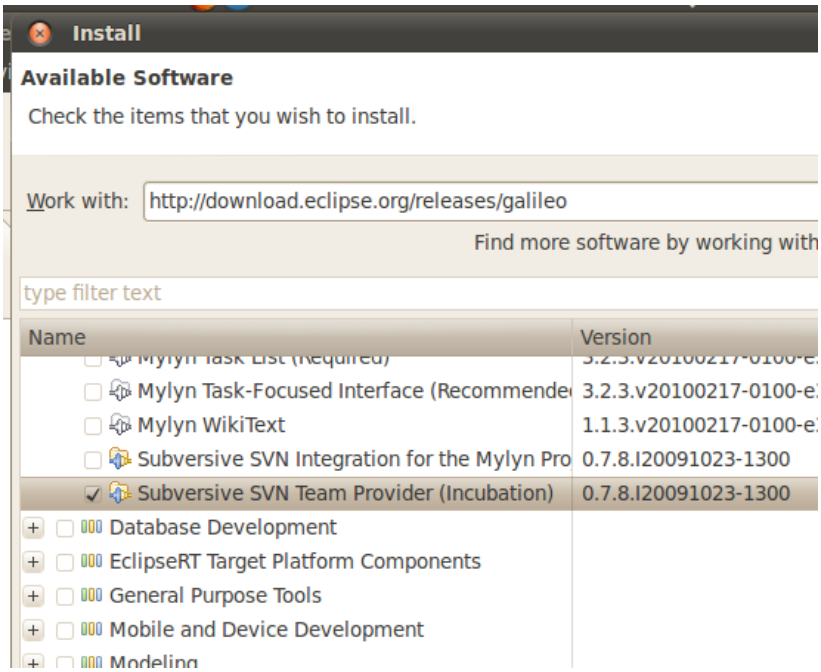

図 2 Subversive のインストール

あとは手続きに従ってインストールします。

さて、Subversive はプラグインの他に、Connector と呼ばれるソフトウェアをインストールするまでは使えません。Connector をインストールするには IDE のメニューから Window -> Open Perspective -> Others…を実行します。そうすると、Eclipse のパースペクティブの一覧があらわれますので、Show all チェックボックスをチェックします。そして現れたリストから”SVN Repository Exploring” を選択します（図 3）。

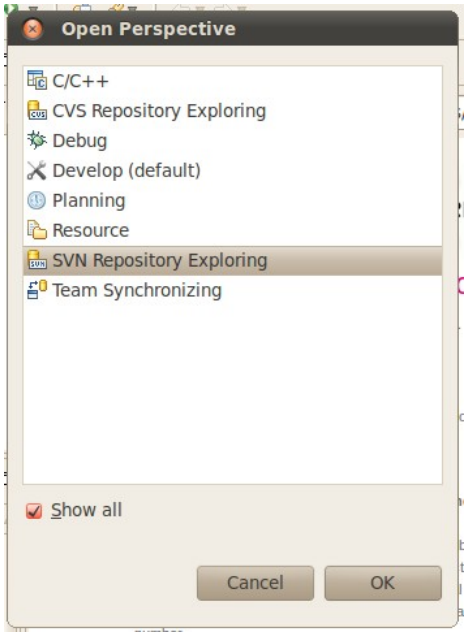

図 3 SVN Repository Exploring

OK を押すと、SVN Repository Explorer が開き、ただちに、Connector のインストール・ダイアログが開くので SVN KIT 1.3.0 を選んでインストールします（図 3）。

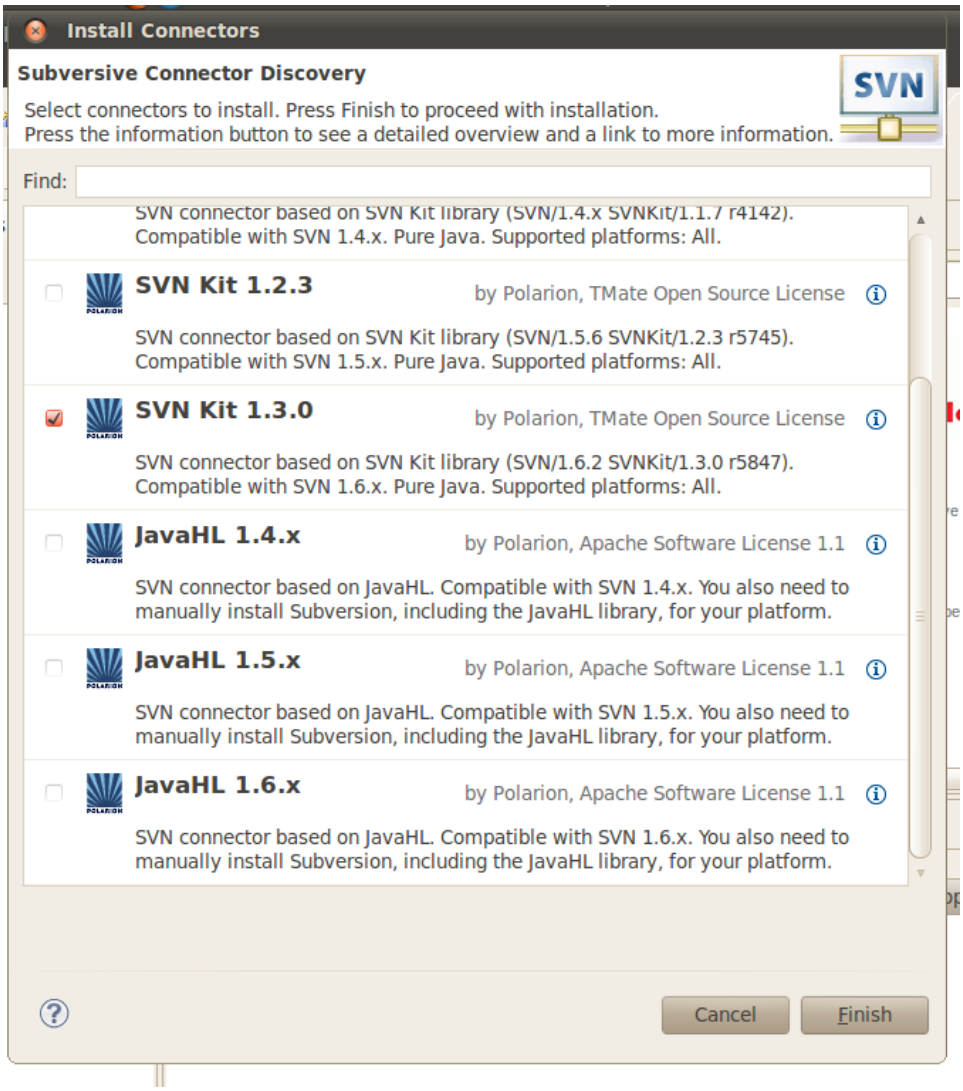

図 4 SVN KIT 1.3.0 を選ぶ

コネクタをインストールすると、Subversive を使って、IDE 上からサーバーのプロジェクトを利用できるようになります。

Subversive や SVN の実際の使い方は Eclipse 関連の書籍やネット上の解説を参考にしてください。

3 TOPPERS/ASP for LPCXpresso の取得

TOPPERS/ASP for LPCXpresso を取得する一番いい方法は、リリース済みのサンプル・プログラムをダウンロードすることです。

- http://sourceforge.jp/projects/toppersasp4lpc/releases/53188~~http://sourceforge.jp/projects/toppersasp4lpc/releases/?package_id=11898~~

この文書の執筆時点で最新のサンプルは Sample_LPCXpresso1768_2011~~0507~~0910.zip です。ダウンロードすると、zip ファイル形式であることがわかりますが、このままにして展開はしません。

このサンプルは以下のような要素からなっています。

- CMSIS 2.0
- TOPPERS/ASP ターゲット非依存部 1.7.0
- TOPPERS/ASP CORTEX-M3 依存部 1.7.0
- ~~+~~LPCXpresso 1768 依存部 1.7.1~~0~~
- TOPPERS/ASP 1.7.0 用 CFG
- ペリフェラル・サブシステム
- ASP アプリケーション・スケルトン
- ASP アプリケーション・サンプル

CMSIS は ARM 社が開発した CORTEX-Mx 用のライブラリで、レジスタへのアクセスを簡易化するマクロや DSP 関連関数がそろっています。NXP はこのライブラリに独自のペリフェラル・アクセス・マクロおよび操作関数と例題をパッケージして配布しています。

TOPPERS/ASP for LPCXpresso の CMSIS は、NXP が配布している CMSIS を LPCXpresso IDE 用にビルドしなおしたものです。一部バグを修正していますが大本のコードは NXP および ARM が開発したものです。再配布する場合や、製品化の際は NXP および ARM のライセンスに従ってください。

TOPPERS/ASP カーネルは、TOPPERS プロジェクトによる STM32F103 へのポーティングを元に、ターゲット依存部を LPC1768/1769 用に移植し直したものです。利用および再配布は TOPPERS プロジェクトのライセンスに従ってください。

ペリフェラル・サブシステムは、CMSIS を使用した TOPPERS/ASP 用のライブラリで、現在の所 I2C 用のライブラリが用意されています。

これらを使ったアプリケーションの構成を図 5 に示します。

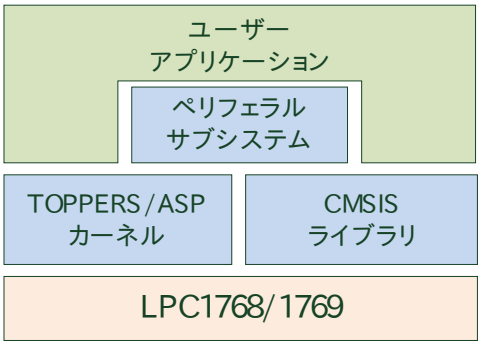

図 5 プログラムの構成

さて、サンプル・プログラムの zip ファイルをダウンロードしたら、IDE からインポートします。

インポートは IDE のメニューから、File→Import を実行します。すると、Import ダイアログが開きますので、General→Existing Projects into Workspace を選んで Next を押します（図 6）。選択するのは Archive File ではなく、Existing Projects into Workspace であることに注意してください。

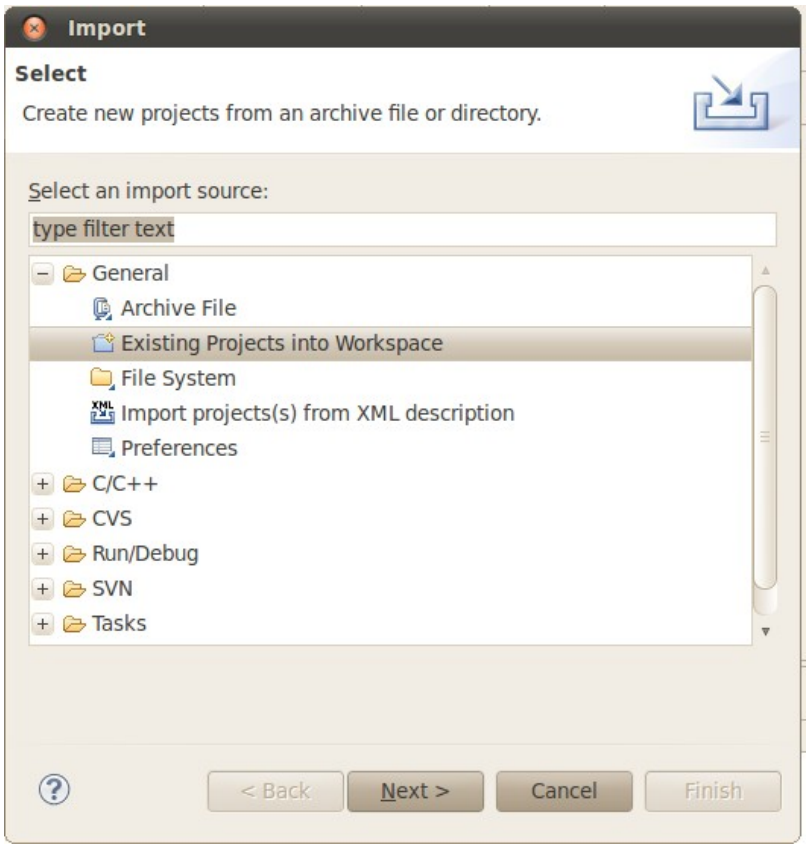

図 6 Import ダイアログ

次のダイアログで読み込む tar アーカイブを指定してやると、ワークスペースへの読み込みが完了します（図 7）。

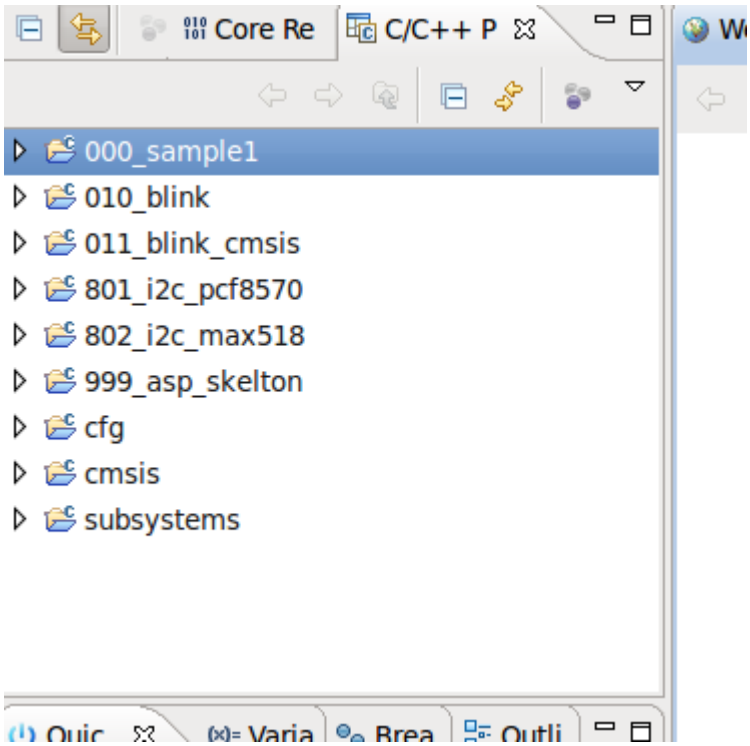

図 7 プロジェクト読み込み完了

4 blink のビルドと実行

4.1 blink について

通常、TOPPERS/ASP アプリケーションを試しに使ってみるときには、Sample1 と呼ばれるアプリケーションを使います。しかしながら、このアプリケーションにはシリアル・ポートが必要であり、一方、買ったままの LPCXpresso1768 には PC と接続するためのシリアル・ポートがありません（UART 端子は出ているが、PC と接続するにはレベル変換か USB シリアル変換が必要）。

そこで、最初に LPCXpersso1768 の LED を点滅させるアプリケーション、blink をビルドしてみましょう。Blink は 010_blink プロジェクトとしてサンプル・プログラムに収録されています。このプロジェクトは CMSIS もサブシステムも使わず、TOPPERS/ASP の機能だけで動作します。

4.2 ディレクトリ構成

アプリケーションのディレクトリ構成は以下のようになっています。

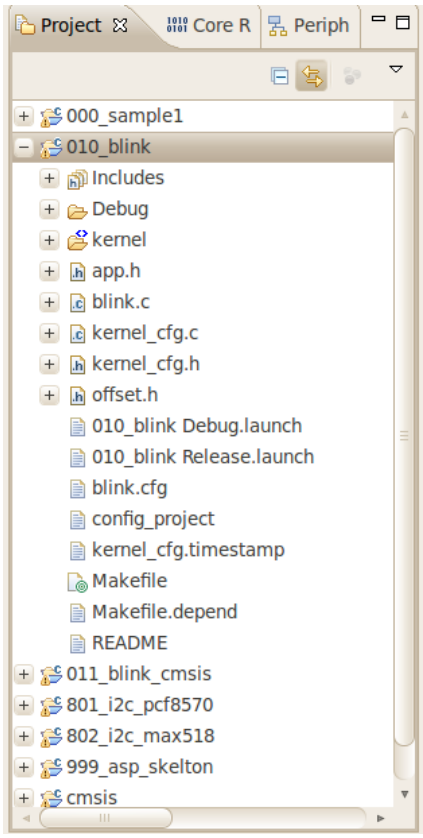

図 8 ディレクトリ構成

サブディレクトリ kernel には ASP カーネルがおいてあります。

ターゲット依存部は lpcxpresso1768_gcc で、LPCXpresso IDE で動かすためにカスタマイズしたものです。このターゲット依存部は内部の RC 発振器を使って 100MHz で動作します。LPC1769 は 120MHz で動作しますが、LPC1768 と互換性をとるために 100MHz に制限してあります。

アプリケーションのソースコードは blink.c と app.h です。そのほかに kernel_cfg.h と kernel_cfg.c があります

が、これは TOPPERS のコンフィギュレータが自動生成したものであり、アプリケーション・プログラマが作ったものではありません。同じく offset.h もコンフィギュレータが生成したものです。

コンフィギュレータのソースである blink.cfg は、タスクやセマフォを宣言するファイルで、これはアプリケーション・プログラマが用意するものです。コンフィギュレータはこのファイルを参照して Makefile を作り出します。

4.3 アプリケーションのビルド

アプリケーションのビルドは簡単です。

プロジェクトを選択し、プロジェクト一覧の下にある”build 010_blink”をクリックすればビルドが始まります（図 9）。ツールのバージョン等のミスマッチがなければ、ビルドは問題なく終了するはずです。

ビルドが完成したら、ターゲットにロードして動かしてみてください。ターゲットへのロード方法やデバッグ方法は、LPCXpresso IDE のドキュメントを参考にしてください。

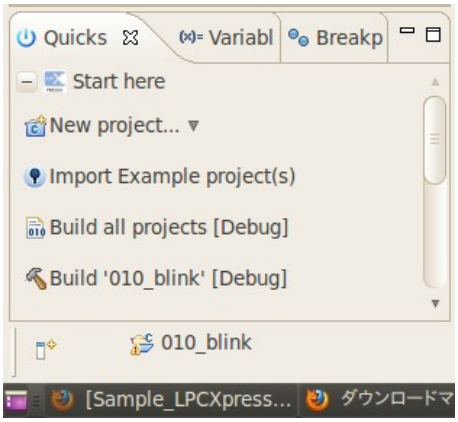

図 9 ビルドの方法

4.4 コンフィギュレータのビルドとコンフィギュレーション

本来 TOPPERS/ASP で一番面倒なのはコンフィギュレーションです。これさえ乗り越えてしまえば、RTOS のおいしさを存分味わうことができます。

しかし、コンフィギュレーションで躓いて飽きてしまってはもったいないので、このサンプル集ではコンフィギュレーション済みのプロジェクトを提供しています。一方で、ユーザー・アプリケーションを自分で開発する際にはコンフィギュレーションを避けて通ることはできません。

コンフィギュレーションはすでに述べたように面倒な作業です。パラメータを覚えるのも面倒ですので、必要に応じて後々 TOPPERS プロジェクトのドキュメントで勉強していただくとして、サンプル集では config_project というスクリプト・ファイルにコンフィギュレーションの手続きを記述しています。

コンフィギュレーションに当たっては TOPPERS のコンフィギュレータが必要になりますが、これは cfg プロジェクトとして独立のプロジェクト扱いをしています。こうすることで、各プロジェクトが独立したコンフィギュレータをビルドして使用する無駄を省いています。~~無い場合にはビルドするようにスクリプトを書いています。~~

一旦 config_project が成功したならば、その後 LPCXpresso が扱えるようにプロジェクトを整えるには、コンフィギュレーション後に次のコマンドを実行~~します~~するだけです。

```
make depend
```

これで LPCXpresso IDE でのビルドが可能になります。

なお、プロジェクトによっては make コマンドで最後までビルドすることはできません。TOPPERS/ASP for LPCXpresso は IDE でのビルドを前提としているため、コマンド・ラインでの環境設定を最小限にしているからです。どうしてもコマンド・ラインでビルドを行いたい場合には、Makefile.target の中でインクルード・ファイルのパスや追加ライブラリを調整してみてください。

5 Kermit による動作確認

「開発環境の構築」で解説したスクリプトは、Kermit の設定とインストールも行います。シリアル・ポートを用意してやれば、Kermit を使って sample1 の動作確認を行うことができます。

最初に、ここで説明する Kermit には一つ制限があることに注意してください。伝統的にシリアル通信ポートの名前は固定であったため、Kermit の設定ファイルも決まった名前のシリアル・デバイスを参照します。ところが、最近主流の USB シリアル・アダプタの場合、いろいろな条件でデバイスの名前が変わります。

名前を決める重要な要素は、シリアル・デバイスとして何番目にシステムに認識されたかと言うことです。先のインストール・スクリプトでは、シリアル・ポートの名前は”/dev/ttyUSB0”に固定して使っています。USB シリアル変換アダプタを複数使う場合には、シリアル・ポートとして使うものを最初にシステムに挿すようにしてください。

Shell から

```
kermit
```

と、入力することで kermit を起動できます。起動直後はユーザーコマンドを待っていますので”c”コマンドで端末モードに移行してください。端末モードから戻るには、エスケープ・コマンド”CTRL-¥ C”を入力します。

図４にシリアル通信の様子を示します（使用しているのは別のプロセッサ用のカーネルです）。

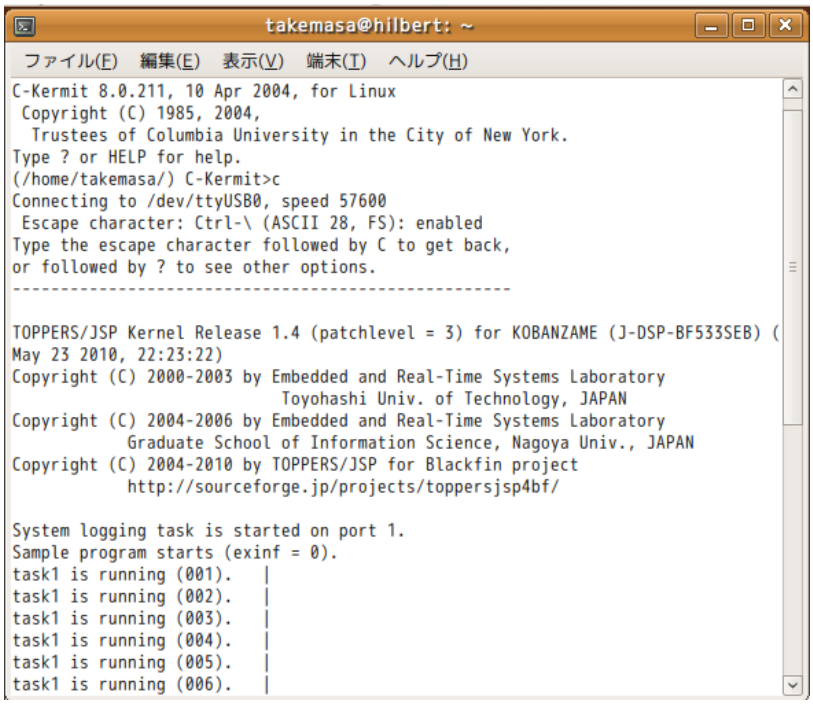

図 10 kermit でシリアル通信を行っているところ

6 CMSIS と SUBSYSTEMS を使ったアプリのビルド

4 章では TOPPERS/ASP だけを使ったアプリのビルドを行いました。この章では、CMSIS および SUBSYSTEMS を使ったアプリのビルドを行います。

6.1 ビルド手順

手順そのものは簡単で、~~アプリのビルドの前に、cmsis プロジェクトをビルドし、subsystems プロジェクトをビルドしておくだけです。この二つのプロジェクトはいずれも Static Archive であり、ビルドすると、それぞれ libcmsis.a と libsubsystems.a を生成します。最後に~~単にアプリをビルドするだけです。~~して終わりです。~~配布されているサンプルは、必要に応じて cmsis と subsystems を参照するプロジェクトとして宣言しているため、アプリケーションのビルド時には、必要に応じてそれらのライブラリがビルドされます。

アプリケーションから見ると、libcmsis.a も、libsubsystems.a も、単なるアーカイブです。アプリケーションは、プロパティを使ってこの二つのアーカイブを使用することを宣言しています。また、アーカイブに関連するインクルード・ファイルを読み込むためのパスも追加しています。

CMSIS と SUBSYSTEMS を使ったアプリの例としては、プロジェクト 801_i2c_pcf8570 があります。

6.2 801_i2c_pcf8570 のコンフィギュレーション

プロジェクト 801_i2c_pcf8570 のコンフィギュレーション・ファイル i2c_pcf8570.cfg は、サブシステムの i2c マスター機能を使用するため、i2c0_m.cfg を読み込んでいます。

読み込まれている i2c0_m.cfg は subsystems プロジェクトが提供するコンフィギュレーション・ファイルで、I2C0 ペリフェラルをマスターとして使用するための初期化ルーチン、および割り込みハンドラを提供しています。コンフィギュレーション・ファイルがあらかたの設定を行ってしまうため、アプリケーション側では、I2C0 用のピン設定さえしてしまえばあとは読み書き関数を使うだけになっています。

ピン設定をサブシステム提供のイニシャライザにさせずにアプリケーションで行っているのは、I2C ペリフェラルによっては、ピンの割り当て方法が複数あるからです。これをイニシャライザで決め打ちしてしまうとユーザーの自由度が下がります。

サブシステム提供のコンフィギュレーション・ファイルはすでに紹介したものの他に i2c1_m.cfg および i2c2_m.cfg があり、それぞれ I2C1、I2C2 ペリフェラルをマスターとして使用する際に使います。

サブシステム提供の関数に関しては、サブシステムのソースコードに記述された Doxygen コメントか、あるいは Doxygen 文書を参照してください。

7 独自アプリケーションを作る

TOPPERS/ASP for LPCXpresso のアプリケーション開発は、基本的には他の TOPPERS/ASP アプリケーション開発と同じです。

ただし、IDE の設定は非常に面倒です。そこで、この章では既存のアプリケーションをテンプレートとして、独自アプリケーションを作る方法を説明します。

7.1 テンプレートとなるアプリケーション

最初にテンプレートとなる、アプリケーションを選びます。

テンプレートにするアプリケーションは、元々そのために用意されている 999_asp_skelton を使うのが順当ですが、代わりに他のアプリケーションを使ってもかまいません。ただし、使用するアーカイブの設定やインクルード・ファイルの設定を考えると、999_asp_skelton か、i2c のサンプル・プログラムから使用する方がいいでしょう。

7.2 テンプレートのリネーム

テンプレートにするプロジェクトを決めたら、次にそのプロジェクトを新しいプロジェクト名にリネームします。

この際、コピーではなくリネームを行うよう気をつけてください。プロジェクトをリネームする場合、IDE はリネームに応じてプロジェクト内部のパスなどを変更して一貫性を保とうとします。しかし、コピーをした場合にはこれが行われません。結果的にインクルード・パスなどの再設定が必要な事から、プロジェクトのコピーは割に合わない作業です。

7.3 アプリケーション・ファイルのリネーム

TOPPERS/ASP のコンフィギュレータは、-A パラメータでアプリケーション名を指定しますが、このとき、アプリケーション名.c、およびアプリケーション名.cfg と言う名前のファイルが両方とも揃っている必要があります。たとえば、

```
Kernel/configure –T lpcxpresso1768_gcc –A FOO -g ../cfg/cfg/cfg
```

と指定した場合、FOO.c および FOO.cfg を用意しなければなりません[1]。

このため、アプリケーション関連ファイルを解明する場合には、このルールに従う必要があります。具体的には

- コンフィギュレーション・ファイル(.cfg ファイル)の名前をアプリケーション名に合わせる
- config_project ファイル内部の configurator スクリプトの-A 引数をアプリケーション名にする

この二つの作業を行った後、以下のコマンドをプロジェクト・ディレクトリ内部で実行します。

```
make clean
./config_project
```

これで新しいプロジェクト用に名前を変えることが出来ました。

コンフィギュレーションがうまくいったら、IDE からビルド出来るはずです。

[1] 詳しくは ASP カーネルの doc 配下のテキストを参照してください。

8 おわりに

8.1 履歴

- 2011 年 9 月 10 日 LPCXpersso1768 依存部 1.7.1 対応。
- 2011 年 7 月 17 日 LPCXpresso1768 依存部 1.7.0 対応。
- 2011 年 5 月 29 日 LPCXpresso1768 依存部 1.6.2 対応。
- 2011 年 5 月 8 日 LPCXpresso1768 依存部 1.6.1 対応。