
はじめてみよう T-Kernel 2/x86



Version 2.0.0



パーソナルメディア株式会社

Copyright © 2008–2011 by Personal Media Corporation

目次	2
目次	
修正履歴	3
はじめに	4
1 ターゲット側実行環境のインストール	5
1.1 実機環境と仮想環境	5
1.2 VMware Player のインストール	6
1.3 VMwareGateway のインストール	7
1.4 T-Kernel 2/x86 のインストール	9
2 開発環境のインストール	18
2.1 Cygwin のインストール	18
2.2 Eclipse のインストール	21
2.3 T-Kernel 開発環境のインストール	22
3 開発手順	28
3.1 ターゲット側と開発環境側の起動	28
3.2 コンソールの利用	29
3.3 プロセスベースと T-Kernel ベース	31
3.4 「Hello, world」プロセスベース編	32
3.5 「Hello, world」T-Kernel ベース編	35
4 実習用サンプルプログラム	39
4.1 ジャグリング (お手玉)	39
4.2 簡易ウェブサーバ	43
4.3 マイクロ秒単位の時間指定サンプル	46
4.4 物理タイマーのサンプル実装	49
索引	52

修正履歴

Version 2.0.0

- T-Kernel 2/x86 評価キット向けに新規作成

はじめに

本書では、はじめて「T-Kernel 2/x86 評価キット」をご利用になる方を対象に、T-Kernel 2/x86 評価キットを使い始めるためのチュートリアルをご提供します。インストールから実習用プログラムの実行まで、この手順に沿って試していただければ、短時間で開発の手順をひとつお体験できます。

1 ターゲット側実行環境のインストール

1.1 実機環境と仮想環境

まずこの章ではターゲット側実行環境 (OS、ドライバ、ミドルウェア等) のインストールを行います。

組込みシステム開発では、プログラムをメイクする開発環境側と、作成したプログラムを実行するターゲット側が、別々のマシンに分かれているのが一般的です。このような開発手法を「クロス開発」と呼びます。

T-Kernel 2/x86 評価キットの特色の一つとして、図 1.1 のように開発用パソコンとは別の実機上にターゲット側実行環境をインストールする方法だけでなく、図 1.2 のように開発用パソコンと同じマシン上の仮想化ソフト (VMware) 上にターゲット側実行環境をインストールする方法も選択できます。

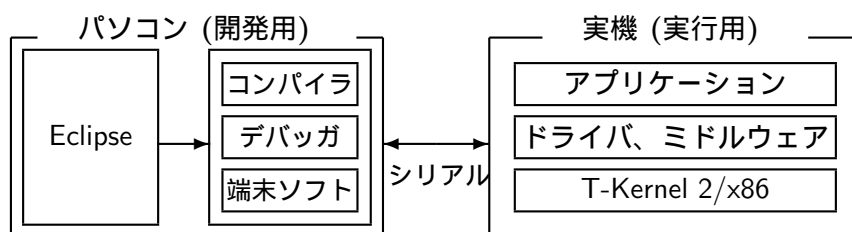


図 1.1 実機環境

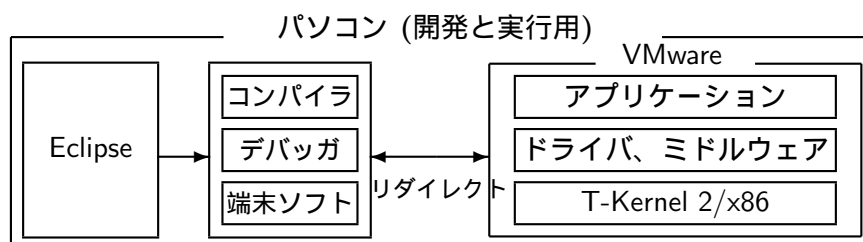


図 1.2 仮想環境

組込みシステムを開発する場合、最終製品は当然実機上になりますが、開発中は仮想環境を利用すると、次のような利点があります。

- パソコンが 1 台あれば開発ができ、他に実機を用意する必要はありません。
- 最終的な実機のハードウェアが完成していない段階でも、ソフトウェアの開発を先行して開始できます。

仮想環境の場合は「1.2 VMware Player のインストール」、「1.3 VMwareGateway のインストール」、「1.4 T-Kernel/x86 のインストール」の順にインストールしてください。実機環境の場合は VMware Player と VMwareGateway はインストール不要ですので、「1.4 T-Kernel/x86 のインストール」に進んでください。

なお仮想環境や実機環境をインストールするマシンに必要なハードウェアの動作条件については、T-Kernel 2/x86 評価キット CD-ROM 内の『評価キット取扱説明書』の「1.2 節動作条件」をご参照ください。

1.2 VMware Player のインストール

T-Kernel 2/x86 を仮想環境上で動作させるために、まず VMware Player をインストールします。

(1) インストーラの起動

T-Kernel 2/x86 評価キット CD-ROM 内の共通ソフトウェアのフォルダ (common¥soft) にある「VMware-player-3.1.3-324285.exe」を起動します。

† ファイル名の中のバージョン番号は上記と異なる場合があります。

† T-Kernel 2/x86 評価キット CD-ROM 内の VMware Player よりも新しいバージョンの方がよろしければ、下記から最新版をダウンロードしてご利用ください。

<http://www.vmware.com/jp/download/player/>

起動してしばらく待つと **次へ (N)** ボタンが有効になるのでクリックします (図 1.3)。

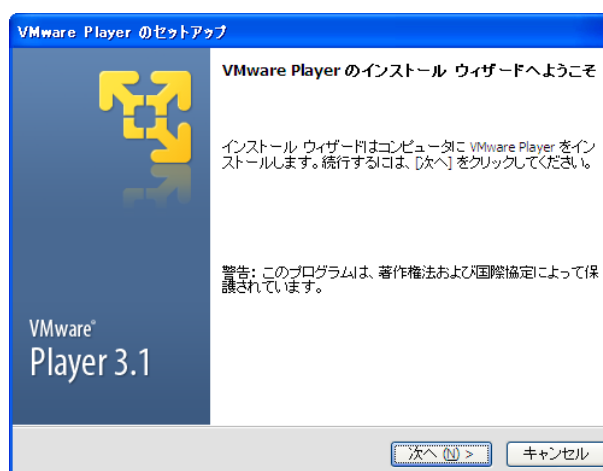


図 1.3 VMware Player のインストーラ起動画面

(2) インストール先フォルダの選択

インストール先フォルダを聞かれます。デフォルト (C:¥Program Files¥VMware¥VMware Player¥) で問題なければそのまま **次へ (N)** をクリックします。

(3) VMware Player の更新確認

VMware Player の起動時に新しいバージョンを自動確認するかどうかを選択して、**次へ (N)** をクリックします。

(4) VMware Player の改善協力

VMware Player の改善のための情報を自動送信するかどうかを選択して、 をクリックします。

(5) 作成するショートカットの確認

作成するショートカットを聞かれますので、特に問題なければそのまま をクリックします。

(6) インストールの開始

をクリックするとインストールが開始します。インストールが完了するまでそのままお待ちください。

(7) インストール完了

インストールが完了したら、Windows の再起動を促されますので、再起動してください。

1.3 VMwareGateway のインストール

T-Kernel 2/x86 と開発環境のコンソール接続は、実機環境の場合はシリアル (RS-232C) ですが、仮想環境では VMwareGateway によるソケット経由の中継で接続します。VMwareGateway は Windows の「サービス」の一つとしてインストールします。

詳細については T-Kernel 2/x86 評価キット CD-ROM 内の『VMware サポートツールインストール方法説明書』をあわせてご参照ください。

(1) インストール先フォルダの作成

「C:¥Program Files¥VMware」の下に「utilities」フォルダを作成します。

(2) 実行ファイルのコピー

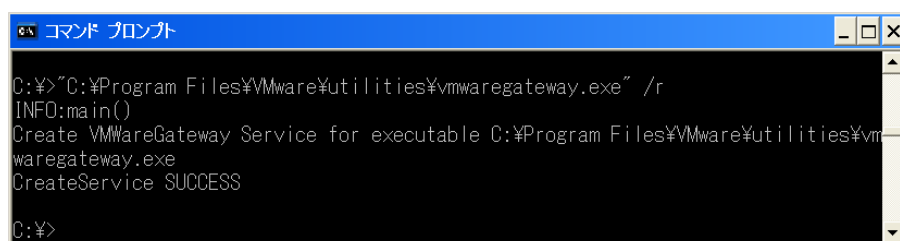
T-Kernel 2/x86 評価キット CD-ROM 内の共通ソフトウェアのフォルダ (common¥soft) にある「vmwaregateway.exe」を「C:¥Program Files¥VMware¥utilities」の下にコピーします。

(3) サービスの登録

コマンドプロンプト (「スタート」「すべてのプログラム」「アクセサリ」「コマンドプロンプト」) を起動してください。

† Windows 7 および Windows Vista の場合は、コマンドプロンプトを左クリックで起動するかわりに、右クリックして「管理者として実行」を選択して起動してください。

「C:¥Program Files¥VMware¥utilities¥vmwaregateway.exe」を「/r」オプションを付けて実行します。パス名に空白が含まれるので、パス名は " で囲ってください (図 1.4)。



```
コマンド プロンプト
C:\>"C:\Program Files\VMware\Utilities\vmwaregateway.exe" /r
INFO:main()
Create VMWareGateway Service for executable C:\Program Files\VMware\Utilities\vmwaregateway.exe
CreateService SUCCESS
C:\>
```

図 1.4 VMWareGateway のサービス登録

(4) サービスの開始

サービス (Windows XP では「コントロールパネル」 「パフォーマンスとメンテナンス」 「管理ツール」 「サービス」、Windows Vista では「コントロールパネル」 「システムとメンテナンス」 「管理ツール」 「サービス」、Windows 7 では「コントロールパネル」 「システムとセキュリティ」 「管理ツール」 「サービス」) を開いて「VMWareGateway」を選択し、右クリックしてメニューから「開始」を実行します (図 1.5)。

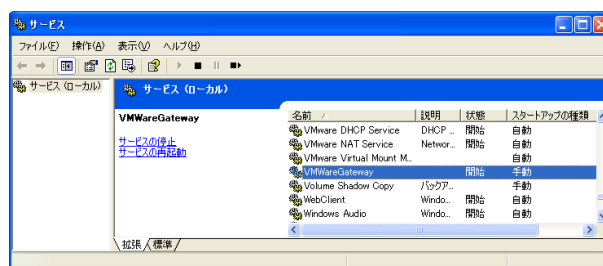


図 1.5 VMWareGateway のサービス開始

(5) サービスの開始の自動化

「VMWareGateway」のプロパティを開き、「スタートアップの種類」を「手動」から「自動」に変更します (図 1.6)。これで次回のコンピュータの起動時からは自動的にサービスが開始されます。

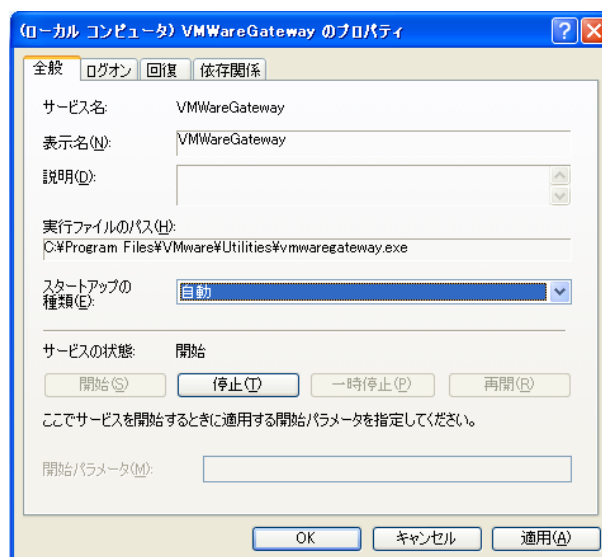


図 1.6 VMWareGateway の開始自動化

1.4 T-Kernel 2/x86 のインストール

仮想環境 (VMware) または実機上に T-Kernel 2/x86 をインストールします。

詳細については T-Kernel 2/x86 評価キット CD-ROM 内の『評価キット取扱説明書』の「2章 ターゲット側ソフトウェアのインストール」をあわせてご参照ください。

パソコン上にはコンソール接続のための端末ソフト (「ハイパーターミナル」や「Tera Term Pro」など) が必要です。

† Windows 7 および Windows Vista にはハイパーターミナルが付属しません。端末ソフトをお持ちでない場合は「Tera Term Pro (バージョン 2.3)」を下記からダウンロードしてご利用ください。

<http://hp.vector.co.jp/authors/VA002416/>

(1) 仮想マシンのコピー (仮想環境のみ)

仮想環境 (VMware) の場合

T-Kernel 2/x86 評価キット CD-ROM 内の「jp」フォルダの下にある「tkx86_vm」フォルダを、フォルダごとマイドキュメント内の「My Virtual Machines」フォルダ内にコピーします。

† マイドキュメント内に「My Virtual Machines」フォルダがない場合は、フォルダを新規作成してください。

実機環境の場合

不要ですので次へ進んでください。

(2) 端末ソフトの起動

• Tera Term Pro の場合

Tera Term Pro を起動します。

- ハイパーターミナルの場合

Windows XP 上のハイパーターミナルの場合は、「スタート」「すべてのプログラム」「アクセサリ」「通信」「ハイパーターミナル」を起動します。

- † 「既定の Telnet プログラムにしますか?」と聞かれる場合がありますが、「はい」でも「いいえ」でもどちらでも構いません。
また市外局番などを聞かれる場合もありますが、これも実際に電話をかけるわけではありませんので、適当に入力して構いません。

名前とアイコンの選択では、適当な名前を入力して **OK** をクリックします (図 1.7)。



図 1.7 ハイパーターミナルの起動

(3) 接続方法の設定

仮想環境 (VMware) の場合

VMwareGateway 経由で telnet で接続します。接続先は「localhost」、ポート番号は「567」としてください。

- Tera Term Pro の場合

「ファイル(F)」「新規接続(N)」を選択します。新規接続ダイアログで、「TCP/IP」を選択し、ホストに「localhost」と入力し、Telnet にチェックを入れ、TCP ポート# に「567」を入力して **OK** をクリックします (図 1.8)。



図 1.8 仮想環境の場合の Tera Term Pro の設定

- ハイパーターミナルの場合

接続方法に「TCP/IP (Winsock)」を選択します。ホストアドレスに「localhost」、ポート番号は「567」を入力して、**OK** をクリックします (図 1.9)。

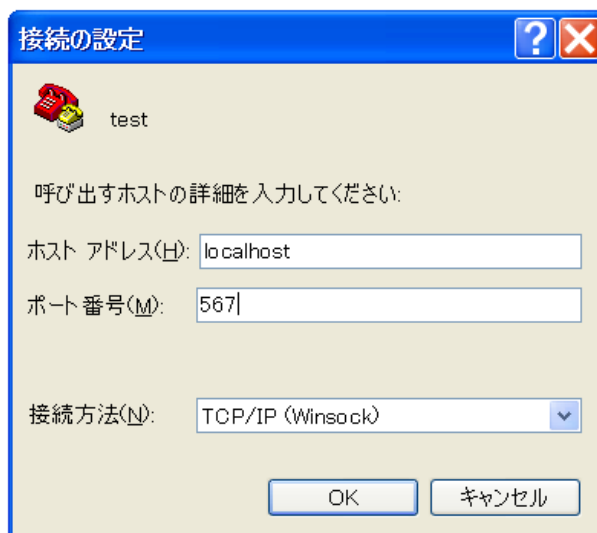


図 1.9 仮想環境の場合のハイパーターミナルの設定

実機環境の場合

開発用パソコンと実機をシリアル (RS-232C) で接続します。開発用パソコンで使用するシリアルポート (「COM1」など) を指定して、ポートの設定は次のように設定します。

ビット/秒	115200 bps
データビット	8 ビット
パリティ	なし
ストップビット	1 ビット
フロー制御	ハードウェア

- Tera Term Pro の場合

「ファイル (F)」 「新規接続 (N)」を選択します。新規接続ダイアログが表示されますので、「シリアル」を選択し、使用するシリアルポート (「COM1」など) を指定して、**OK** をクリックします。

続いて「設定 (S)」 「シリアルポート (P)」を選択します。シリアルポート設定ダイアログが表示されますので、上記のように設定してください。

- ハイパーターミナルの場合

接続方法に使用するシリアルポート (「COM1」など) を選択して**OK** をクリックします。続いてポートの設定ダイアログが表示されますので、上記のように設定してください。

(4) ターゲットの CD-ROM からの起動

仮想環境 (VMware) の場合

T-Kernel 2/x86 評価キット CD-ROM をパソコンの CD-ROM ドライブに入れます。

「My Virtual Machines」フォルダ内の「tkx86_vm」内にある「T-Kernel_x86.vmx」を起動します。

VMware Player のウィンドウが開き、しばらくすると「Operating System not found」と出ますが、これは仮想マシンに CD-ROM ドライブが接続されていないためです。そこで VMware Player のウィンドウ下部の「CD/DVD(IDE)」アイコンをクリックして「接続」を選択します (図 1.10)。

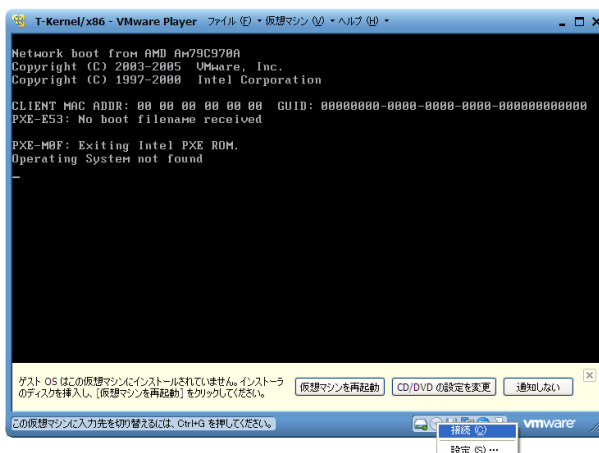


図 1.10 VMware Player の CD-ROM ドライブ接続

† USB 接続の CD/DVD ドライブをご利用の場合でも、「CD/DVD(IDE)」を指定してください。

VMware Player のウィンドウ内をクリックし、矢印のマウスカーソルが消えるのを確認します。

Ctrl + **Alt** + **Insert** キーを押して仮想マシンを再起動します。これで CD-ROM から起動します。

実機環境の場合

T-Kernel 2/x86 評価キット CD-ROM を実機の CD-ROM ドライブに入れて、実機の電源を入れて起動します。

† 実機の BIOS 設定で CD-ROM ブートを行うかどうか選択できる場合は、CD-ROM ブートを行えるように BIOS を設定してください。

(5) 起動の確認

ターゲット側の画面 (仮想環境では VMware Player のウィンドウ上、実機環境では実機に接続されたディスプレイ) には T-Kernel のロゴマークが表示され、それからコンソール画面が表示されます。

一方、開発用パソコンの端末ソフト上では、↵(Enter) キーを入力すると「[/SYS]%'のプロンプトが返ってきます (図 1.11)。

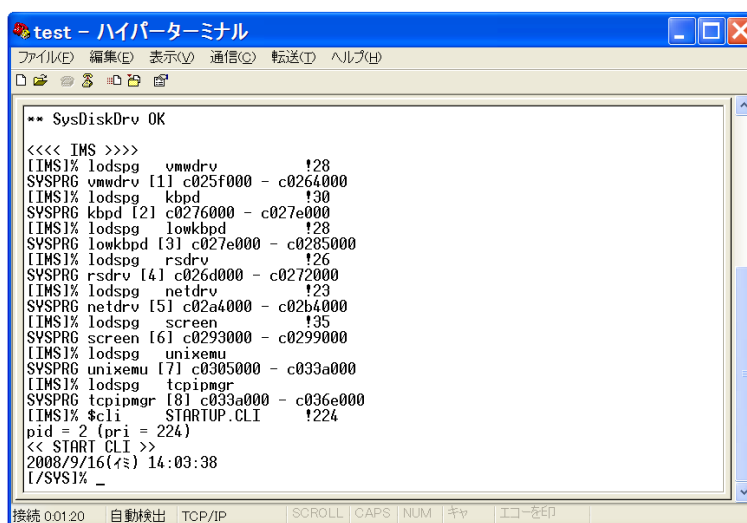


図 1.11 端末ソフト上のプロンプト

- † ターゲット側の画面のコンソールでも操作可能ですが、以下のインストール作業については、ターゲット側の画面のコンソールではなく、開発用パソコンの端末ソフト上で行なってください。

(6) パーティションの設定

端末ソフト上で `hdpart` コマンドを起動して、インストールするディスクのパーティションを設定し、ブート区画を設定します。デバイス名は「`hda`」(ディスク全体)を指定します。以下に例を示します。

```

[/SYS]% hdpart -m1 hda ← — hdpart コマンドの起動
hda [C:1017 H:255 S:63 B:16357785 (7987 MB)]
No System Boot StartCHS EndCHS SecNo SecCnt Size
1 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
2 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
3 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
4 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
** Create/Delete/Boot/Edit/Quit ? c ← — 区画作成
Create PartNo (1-4) ? 1 ← — 最初の区画
Size [GB/MB/KB,All] (<7987MB) ? a ← — 区画サイズ
No System Boot StartCHS EndCHS SecNo SecCnt Size
1 13 BTRON 00 0: 1: 1 1018: 57:24 63 16357722 7987 MB
2 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
3 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
4 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
** Create/Delete/Boot/Edit/Update/Quit ? b ← — ブート区画
Boot PartNo (1-4,Clear) ? 1 ← — 最初の区画
No System Boot StartCHS EndCHS SecNo SecCnt Size
1 13 BTRON 80 0: 1: 1 1018: 57:24 63 16357722 7987 MB
2 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
3 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
4 00 ----- 00 0: 0: 0 0: 0: 0 0 0 0 KB
** Create/Delete/Boot/Edit/Update/Quit ? u ← — 区画更新
** hda: Updated Master Boot Block
[/SYS]%

```

- † 既存の区画に T-Kernel 2/x86 評価キットを上書きインストールする場合は、まず d (delete) で区画を削除してから、c (create) で区画を再作成してください。
- † 旧製品 (T-Kernel/x86 評価キット) からのバージョンアップの場合も、いったん旧製品をインストールした区画を削除してから、新規に区画を作成してインストールしてください。

(7) フォーマット

端末ソフト上で `formatsys` コマンドを使ってディスクをフォーマットします。デバイス名は、ディスクの最初の区画なら「hda0」、2 番目の区画なら「hda1」を指定します。必ず「-b」オプション (ブートコード書込) が必要です。また「-x」オプションも通常指定してください。

```
[/SYS]% formatsys -b -x hda0 SYSTEM ←           — formatsys コマンドの起動
Format pcb0 [STD] SYSTEM
Logical Formatting...
Writing BootCode...
Disk Format Success.
.
ATYPE ATR  NREC  NREF SIZE BLK MTIME           NAME
0003 ----- 1 0    1  32156   9 110121 15:40:12 SBOOT
0003 ----- 1 0    1 335451  83 110121 15:40:13 KERNEL.SYS
0003 ----- 1 0    1   4653   3 110121 15:40:13 SYSCONF
0003 ----- 1 0    1   1448   2 110121 15:40:14 DEVCONF
[/SYS]%
```

- † ここでは `format` コマンドではなく `formatsys` コマンドをご使用ください。

(8) ディスクの接続

前項でフォーマットしたディスクの区画を接続します。

```
[/SYS]% att hda0 HD ←                               — ディスク接続
hda0 -> HD
[/SYS]% cd /HD ←                                     — カレントディレクトリ設定
[/HD]% ls -l ←                                       — ファイル一覧
ATYPE ATR  NREC  NREF SIZE BLK MTIME           NAME
0003 ----- 1 0    1  32156   9 110121 15:40:12 SBOOT
0003 ----- 1 0    1 335451  83 110121 15:40:13 KERNEL.SYS
0003 ----- 1 0    1   4653   3 110121 15:40:13 SYSCONF
0003 ----- 1 0    1   1448   2 110121 15:40:14 DEVCONF
```

【トラブルシューティング】/HD にこれらのファイルがない場合は、正しくインストールされていません。「cd /SYS」でカレントディレクトリを元に戻し、「det hda0」でディスクを切断してから、インストール手順を再度見直してください。

(9) CD-ROM の接続

まず「df」コマンドで CD-ROM のデバイス名を確認します。/SYS に対するデバイス名が例えば「hdc2」であれば、インストールデータは数字を 0 に変えた「hdc0」に格納されています。その区画を接続します。

```
[/HD]% df ← — ディスク使用状況の表示
PATH DEV TOTAL FREE USED UNIT MAXFILE NAME
/SYS hdc2 3100 22 99% 1024 256 PM:TK2-x86
/HD hda0 8178860 8176304 0% 4096 65535 SYSTEM
[/HD]% att -c hdc0 CD ← — CD-ROM 接続
hdc0 -> CD (ReadOnly)
```

(10) 基本システムのインストール

基本システムのインストーラを展開して実行します。

```
[/HD]% expf -U /CD/JP/SOFT/INSTALL.BZ ← — アーカイブ展開
[/HD]% install /HD ← — インストーラ実行
基本システムのインストールが完了しました。
[/HD]% rm -r install ← — インストーラ削除
```

(11) PMC T-Shell のインストール

T-Shell のインストーラを展開して実行します。

```
[/HD]% expf -U /CD/JP/SOFT/TSHELL.BZ ← — アーカイブ展開
[/HD]% install_tshell /HD ← — インストーラ実行
インストールが完了しました。再起動してください。
[/HD]% rm -r install_tshell ← — インストーラ削除
```

(12) T-Kernel 2/x86 の再起動

exit コマンドを使って終了させます。

```
[/HD]% exit ← — CLI 終了
<< EXIT cli >>
[IMS]% exit
<< SYSTEM SHUTDOWN >>
```

そのあと T-Kernel 2/x86 評価キット CD-ROM をドライブから取り出して、再起動させます。

仮想環境 (VMware) の場合

「My Virtual Machines」フォルダ内の「tkx86_vm」内にある「T-Kernel_x86.vmx」を起動します。ショートカットを作成しておくともよいでしょう。

実機環境の場合

実機の電源を入れて起動します。

ターゲット側の画面 (仮想環境では VMware Player のウィンドウ上、実機環境では実機に接続されたディスプレイ) に図 1.12 のような PMC T-Shell のウィンドウ画面が表示されます。

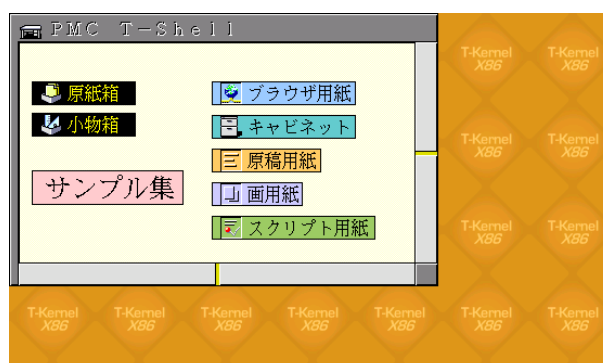


図 1.12 PMC T-Shell の初期ウィンドウ

一方端末ソフト上には、T-Kernel 2/x86 の起動メッセージが続いた後、「[/SYS]%'と表示されます。

df コマンドを使うと起動ディスク (/SYS) のデバイス名が分かります。

```
[/SYS]% df ← — ディスクの確認
PATH DEV TOTAL FREE USED UNIT MAXFILE NAME
/SYS hda0 265040 239752 9% 4096 22528 SYSTEM
```

(13) 画面サイズと色数の設定

画面はデフォルトでは 1024 × 768、65536 色になっています。これを変更したい場合は次のようにデバイスコンフィグレーションの VIDEOMODE を設定します。

```
[/SYS]% devconf VIDEOMODE 2 ← — 640 × 480、65536 色の場合
+ 0: VIDEOMODE 2
[/SYS]% exit ← — CLI 終了
<< EXIT cli >>
[IMS]% exit 1
[IMS]% exit -3 ← — 再起動
<< SYSTEM REBOOT >>
```

† サポートしている表示モードと VIDEOMODE の数値の関係については、T-Kernel 2/x86 評価キット CD-ROM 内の『T-Kernel 2/x86 実装仕様書』の「4.6 スクリーン (ディスプレイ) ドライバ」をご参照ください。

† devconf コマンドのかわりに、PMC T-Shell の画面で右クリックして「システム環境設定」「画面サイズ色数」で設定することも可能です。設定後に再起動が必要です。

(14) ターゲットシステムの終了

ターゲット側のシステムを終了するには、マウスの右ボタンをクリックするとメニューが表示されますので、「終了」を選択してください。「システムを終了しますか?」と確認ダイアログが出ますので、「終了」を選択してください。

デバッグモードの場合は、さらに端末ソフト上から exit コマンドを 2 回使って終了させてください。

```
[/SYS]% exit ← — CLI 終了
<< EXIT cli >>
[IMS]% $logon E
pid = 13 (pri = -1)
[IMS]% exit 1
[IMS]% exit ← — システム終了
<< SYSTEM SHUTDOWN >>
```

(15) 端末ソフトの終了

端末ソフト (Tera Term Pro やハイパーターミナルなど) を終了します。

- † この章ではターゲット側のインストールのためにハイパーターミナルなどの端末ソフトを使用しましたが、次の章で開発環境として Eclipse をインストールすれば、コンソールは Eclipse 上の gterm から使う形になります。

2 開発環境のインストール

2.1 Cygwin のインストール

Windows パソコン上で開発環境を使うには、まず Cygwin をインストールする必要があります。詳細については T-Kernel 2/x86 評価キット CD-ROM 内の『Cygwin インストール方法説明書』をあわせてご参照ください。

- † Windows のログイン時のアカウントは、管理者権限としてください。制限されたアカウントの場合、ソフトウェアのインストールがうまくいきません。
また Windows のログイン時のアカウント名は、半角英数字のみとしてください。アカウント名に全角文字や半角空白などが入ると、Cygwin で不都合が生じる場合があります。
- † 既に Cygwin がインストールされている場合でも、バージョンが古かったり、必要なモジュールが足りない場合も考えられますので、『Cygwin インストール方法説明書』をご一読の上、必要に応じて再インストールをお願いいたします。

(1) パッケージの展開

T-Kernel 2/x86 評価キット CD-ROM 付属 CD-ROM 内の共通ソフトウェアのフォルダ (common\soft) にある Cygwin システムパッケージ「cygwin.1.7.7-1.zip」を、いったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先はハードディスクのどこでも構いません。

- † ファイル名の中のバージョン番号は上記と異なる場合があります。

(2) インストーラの起動

展開したフォルダ内にあるインストーラ setup.exe を起動します (図 2.1)。

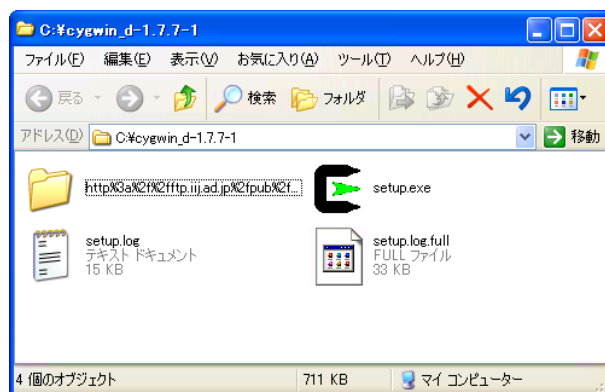


図 2.1 Cygwin インストーラの起動

インストーラが起動したら をクリックします。

(3) インストールタイプの選択

「Install from Local Directory」を選択して をクリックします。

(4) インストールディレクトリの選択

Cygwin をインストールするディレクトリを選択します。デフォルトでは「C:\cygwin」になっています。支障がなければデフォルトのままにしておきます (図 2.2)。

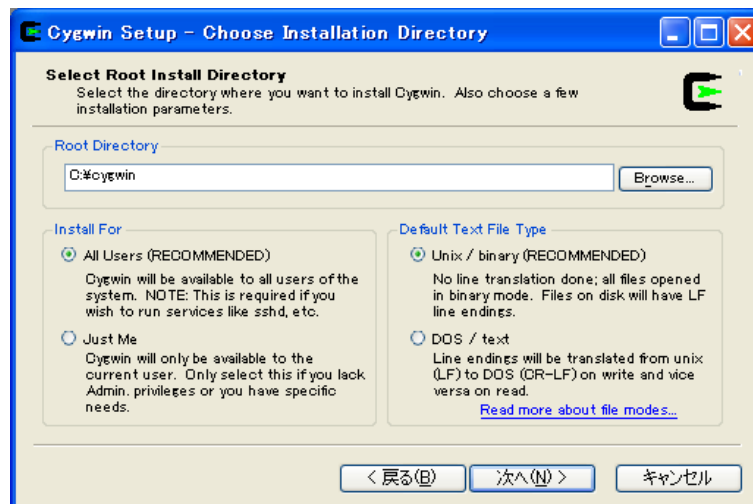


図 2.2 インストールディレクトリの選択

また「Install For」は「All Users」、 「Default Text File Type」は「Unix」を選択してください。いずれもデフォルトです。

Next > をクリックします。

(5) ローカルパッケージディレクトリの選択

(1) でインストールパッケージを展開したフォルダを指定して、 **Next >** をクリックします。

(6) インストールするパッケージの選択

ファイルのチェックが行われた後、インストールパッケージの選択画面になります。「All」の右の「Default」をクリックして「Install」の状態にしてから **Next >** をクリックします (図 2.3)。

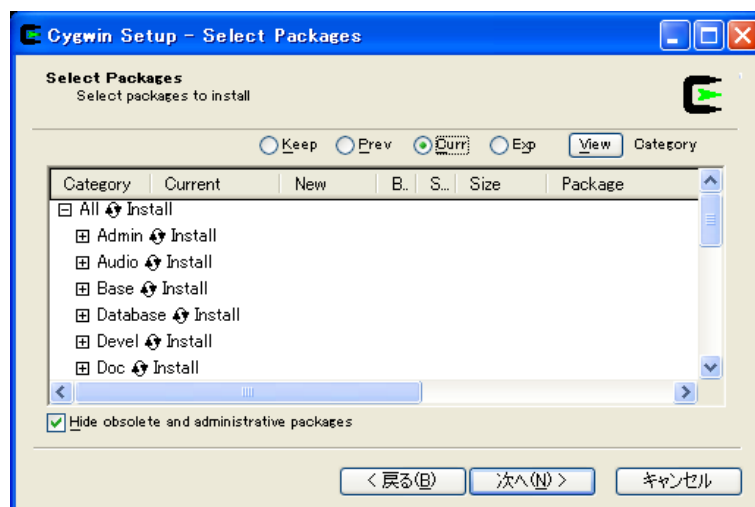


図 2.3 インストールするパッケージの選択

(7) インストール

インストールが開始されます。時間がかかりますので、進度が 100% になるまでしばらくお待ちください。

(8) アイコンの作成

「Create icon on Desktop」を選択して **完了** をクリックします

(9) インストール完了

OK をクリックします。

(10) ファイル名の大文字と小文字の区別の設定

Cygwin 起動用バッチファイル「C:¥cygwin¥cygwin.bat」をテキストエディタで開き、「bash --login -i」という行の前に次の行を追加します。

```
set CYGWIN=nowinsymlinks check_case:strict
```

(11) home ディレクトリの初期化

デスクトップ上の「Cygwin」アイコンをダブルクリックして Cygwin を起動します。最初の起動時に home ディレクトリが初期化されます。

(12) gmake へのリンクの作成

gmake で make が起動するように、Cygwin 上で次のようにシンボリックリンクを作成します。

```
$ cd /usr/bin ↵
$ ln -s make gmake ↵
$
```

(13) Perl へのリンクの作成

Perl がパス名 /usr/local/bin/perl で起動するように、Cygwin 上で次のようにシンボリックリンクを作成します。

```
$ cd /usr/local/bin ←
$ ln -s /usr/bin/perl ←
$
```

終わったら Cygwin のウィンドウは exit コマンドで閉じて構いません。

2.2 Eclipse のインストール

統合開発環境の Eclipse をインストールします。

詳細については T-Kernel 2/x86 評価キット CD-ROM 内の『Eclipse インストール方法説明書』をあわせてご参照ください。

(1) Java 実行環境のインストール

Eclipse の実行には Java 実行環境が必要です。まだインストールされていない場合、またはバージョンが古い場合は、<http://www.java.com/ja/> からダウンロードしてインストールしてください。

(2) Eclipse システムパッケージの展開

T-Kernel 2/x86 評価キット CD-ROM 内の共通ソフトウェアのフォルダ (common¥soft) にある以下の 4 つのファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先は C ドライブの直下 (C:¥) を指定してください。

Eclipse 本体	eclipse-platform-3.2.2-win32.zip
Eclipse 本体用日本語化パック	NLpack1-eclipse-platform-3.2.1-win32.zip
CDT プラグイン	org.eclipse.cdt-3.1.2-win32.x86.zip
CDT 日本語化パック	CDT_NL_3.1.1.zip

† ファイル名の中のバージョン番号は上記と異なる場合があります。

† 展開先として C:¥ を指定すると、実際には C:¥eclipse というフォルダが作成され、その下に展開されます。

† CDT プラグインの展開時に、上書きされるかどうか 2 回聞かれますが、ライセンス表示の html ファイルですので、上書きで問題ありません。

(3) ショートカットの作成

「C:¥eclipse¥eclipse.exe」のショートカットをデスクトップ上に作成します。

(4) 起動の確認

Eclipse のショートカットをダブルクリックして Eclipse を起動します。図 2.4 のような起動画面が表示され、それからワークスペースの選択ダイアログが出れば正常です。いったん をクリックして終了してください。



図 2.4 Eclipse の起動画面

【トラブルシューティング】「Java Runtime Environment がない」と表示される場合は、(1) の Java 実行環境のインストールをご確認ください。

2.3 T-Kernel 開発環境のインストール

Eclipse に T-Kernel 開発用プラグインを追加インストールします。

詳細については T-Kernel 2/x86 評価キット CD-ROM 内の『Eclipse 用 T-Kernel 開発環境インストール方法説明書』と『GNU 開発環境 (Eclipse 版) 説明書』をあわせてご参照ください。

(1) T-Kernel 開発用 Eclipse プラグインの展開

T-Kernel 2/x86 評価キット CD-ROM 内の日本語版ソフトウェアのページ (jp¥soft) にある以下の 2 つのファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先は C:\eclipse を指定してください。

プラットフォーム共通部分	com.t.engine4u.te.1.1.0.zip
各機種対応部分	com.t.engine4u.tl.pcat.1.1.0.zip

† zip アーカイブの展開は、Windows 標準の zip 展開機能で展開してください。他のソフトを使って展開すると、ソフトによっては正しく展開できない場合があります。

† ファイル名の中のバージョン番号は上記と異なる場合があります。

(2) T-Shell 開発用追加モジュールの展開

T-Kernel 2/x86 評価キット CD-ROM 内の日本語版ソフトウェアのページ (jp¥soft) にある以下のファイルをいったんデスクトップなどに保存した上で、Windows のエクスプローラでマウスの右ボタンをクリックして「すべて展開 (A)...」を実行してください。展開先

は「C:¥eclipse¥plugins¥com.t_engine4u.tl.pcat.1.1.0_1.1.0¥te」を指定してください。

T-Shell 開発用追加モジュール	com.t_engine4u.tsh.pcat.1.1.0.zip
--------------------	-----------------------------------

† ファイル名の中のバージョン番号は上記と異なる場合があります。

(3) Eclipse の起動

Eclipse のショートカットをダブルクリックして Eclipse を起動します。

(4) ワークスペースの選択

起動時にワークスペースの選択を聞かれます。ここではプロセスベースのプログラムを作成するための標準的なワークスペースである「C:¥te¥bappl」としてください。

† ワークスペースとは、プロジェクトを保管するフォルダのことです。通常、プロセスベースのプログラムを作成する場合は「C:¥te¥bappl」、T-Kernel ベースのプログラムを作成する場合は「C:¥te¥kappl」、デバイスドライバを作成する場合は「C:¥te¥driver」をワークスペースとするのが標準的ですが、これとは違う場所にワークスペースを作成しても構いません。

(5) ワークベンチにジャンプ

「ようこそ」ビューが表示されますので、ワークベンチをクリックします (図 2.5)。

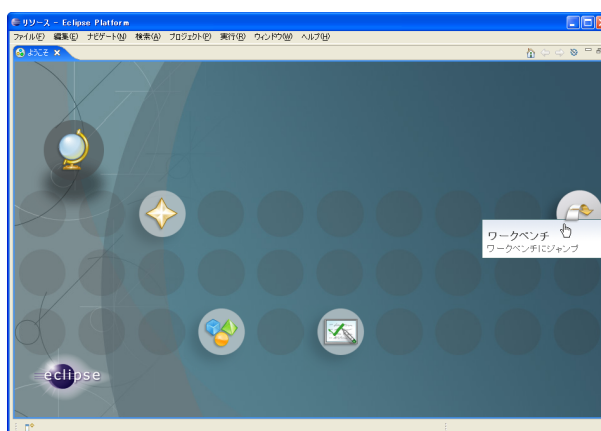


図 2.5 Eclipse の「ようこそ」ビュー

(6) T-Engine 開発パースペクティブを開く

ツールバーの「ウィンドウ」 「パースペクティブを開く」 「T-Engine 開発」 を選択します (図 2.6)。

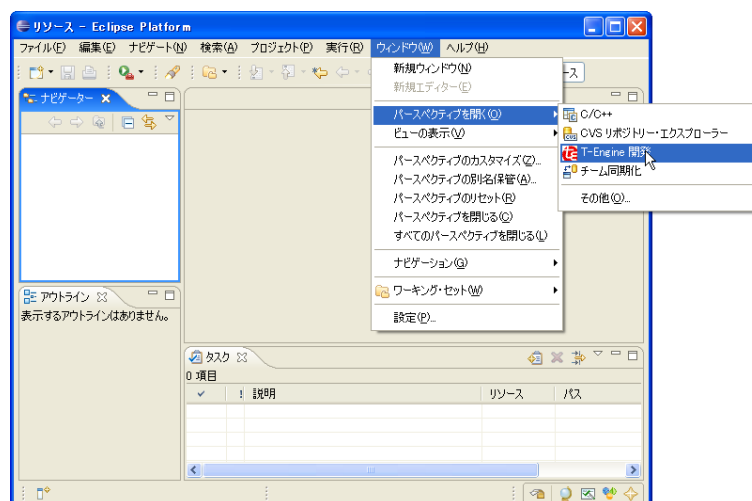


図 2.6 T-Engine 開発パースペクティブの選択

(7) T-Engine 開発環境の設定

ツールバーの「ウィンドウ」「設定」で設定ダイアログを開きます。

設定ダイアログの左側の中から「T-Engine 開発環境」を選択して内容を確認します。

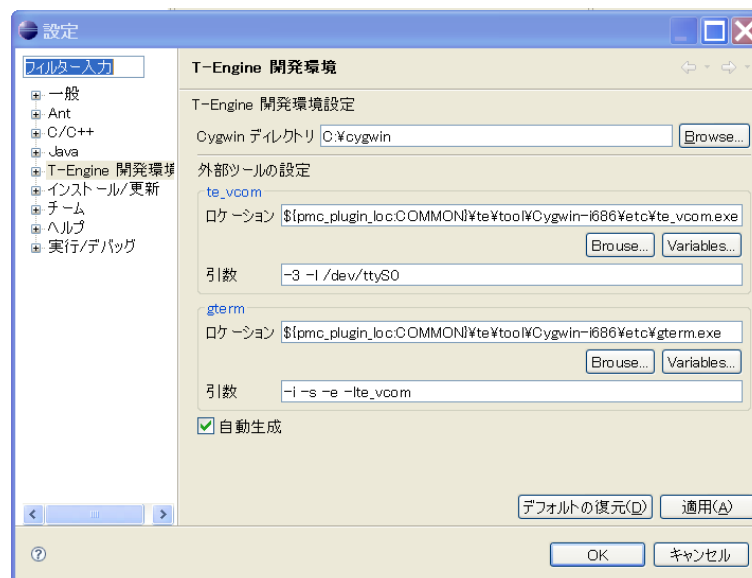


図 2.7 T-Engine 開発環境の設定

仮想環境 (VMware) の場合

te_vcom の引数として「-l localhost」を指定します。

† -l はマイナス・エルです。

実機環境の場合

シリアルポートとして COM1 を通信速度 115200 bps で使う場合は、te_vcom の引数として「-B -l /dev/ttyS0」を指定します。

† COM ポート番号から 1 を引いた値がデバイス名になります。例えば COM2 であれば「-B -1 /dev/ttyS1」を設定してください。

(8) ワークスペースの設定

設定ダイアログの左側の中から「一般」の「ワークスペース」を選択して、次の設定を行います (図 2.8)。

- 「自動的にビルド」のチェックを外します。
- 「テキスト・ファイル・エンコード」の「その他」をチェックして、「EUC-JP」を選択します。
- 「新規テキスト・ファイルの行区切り文字」の「その他」をチェックして、「Unix」を選択します。

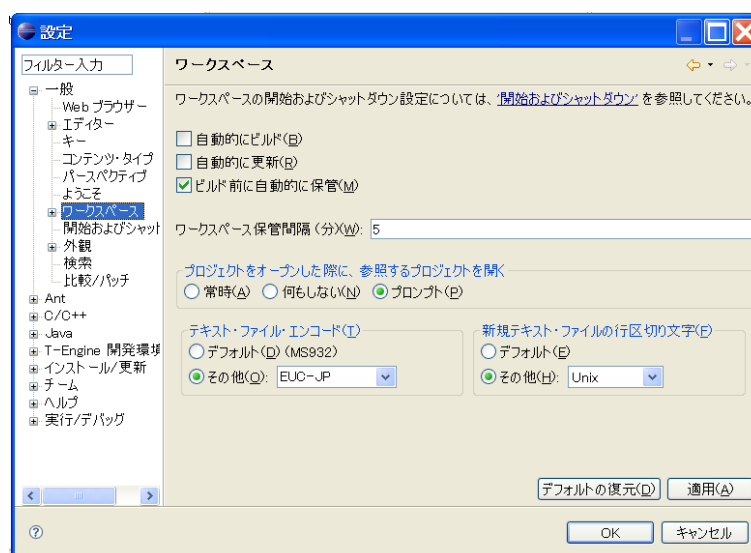


図 2.8 ワークスペースの設定

なお環境設定はワークスペースごとに行う必要があります。

プロセスベースのワークスペース「C:¥te¥bapp1」での設定が終わったら、ツールバーの「ファイル」 「ワークスペースの切り替え」でワークスペースを「C:¥te¥kapp1」(T-Kernel ベースのプログラム用のワークスペース) に切り替えて、同様の設定を行ってください。さらにワークスペースを「C:¥te¥driver」(デバイスドライバ用のワークスペース) に切り替えて、同様の設定を行ってください。

(9) ターゲット側との接続の確認

ターゲットと通信を行うため、中継プログラム (te_vcom) とターミナルエミュレータ (gterm) を起動します。ツールバーの外部ツールの をクリックし、「外部ツール (E)...」をクリックします (図 2.9)。

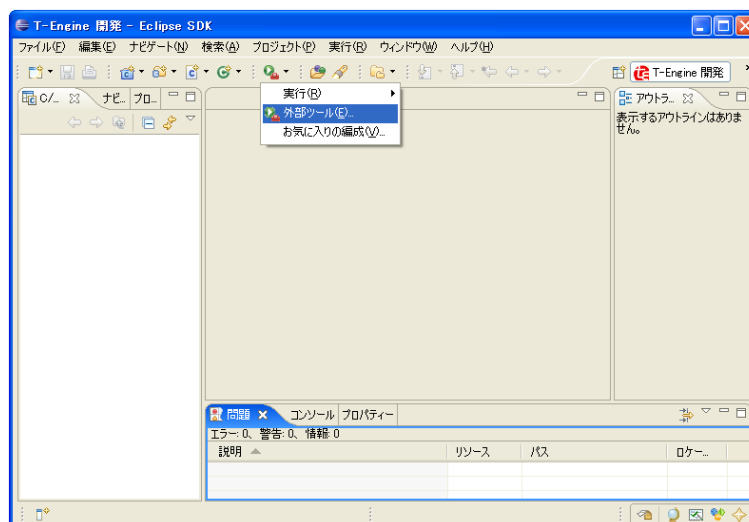


図 2.9 外部ツールの起動

外部ツールのダイアログが表示されたら、左側の「プログラム」の左の+をクリックすると「te_vcom」と「gterm」の設定が表示されます。設定内容は通常変更する必要はありません。まずは「te_vcom」を選択し、「実行」してください(図 2.10)。

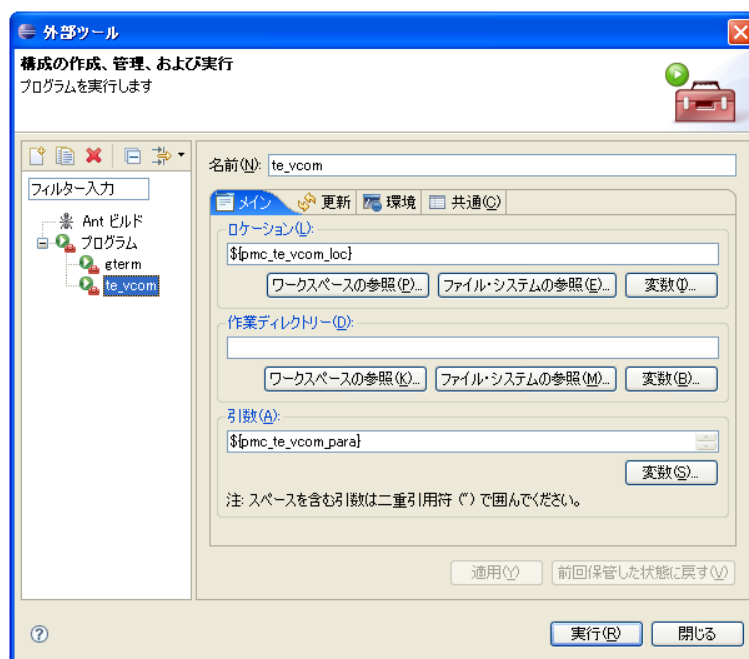


図 2.10 te_vcom の起動

続いて同様の手順で「gterm」を起動してください。コンソールに「gterm」の起動メッセージが表示されます。

ターゲット側の T-Kernel 2/x86 が起動した状態で、gterm のコンソール上で ↵(Enter) キーを入力すると、ターゲット側から返されるプロンプト「[/SYS]%」が表示されます(図 2.11)。

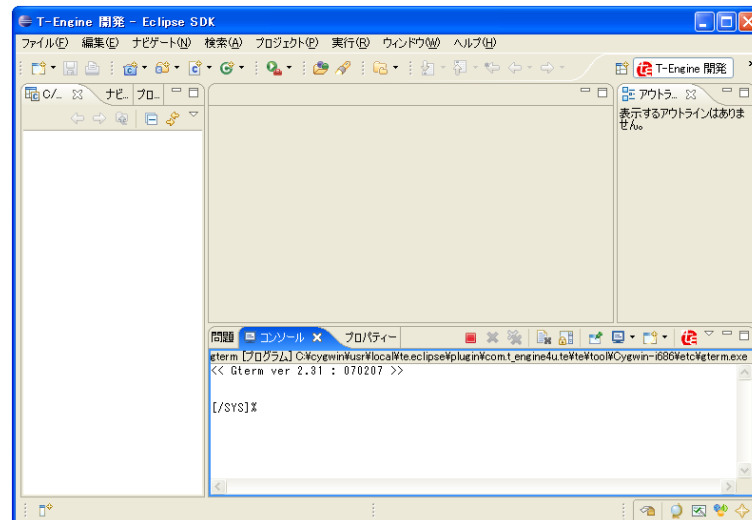


図 2.11 gterm 上に表示されたターゲット側プロンプト

これらの外部ツール `te_vcom` と `gterm` は、一度起動すると次回からはツールバーの外部ツールのプルダウンリストにショートカットが表示されますので、ショートカットを使って起動できます。

3 開発手順

3.1 ターゲット側と開発環境側の起動

開発を始めるには、まずターゲット側と開発環境側をそれぞれ次の手順で起動します。

(1) ターゲット側の起動

ターゲット側で T-Kernel 2/x86 を起動します。

仮想環境 (VMware) の場合

「My Virtual Machines」フォルダ内の「tkx86_vm」内にある「T-Kernel_x86.vmx」を起動します。ショートカットを作成しておくとい良いでしょう。

VMware のウィンドウ内に T-Shell の初期ウィンドウ (1 章の図 1.12) が表示されれば正常です。

【トラブルシューティング】前回の終了状況によっては、そうならない場合もあります。その場合は VMware Player の上辺にある「Player」「Troubleshoot」「Reset」でリセットをかけてください。

実機環境の場合

実機の電源を入れて起動します。

実機のディスプレイ上に T-Shell の初期ウィンドウが表示されれば正常です。

(2) Eclipse の起動

Eclipse を起動します。

最初にワークスペースの選択を聞かれますので、プロセスベースの場合は「C:¥te¥bapp1」とします。

(3) te_vcom と gterm の起動

T-Kernel 2/x86 と通信を行うには、ツールバーの外部ツールの をクリックして、中継プログラム (te_vcom) と端末エミュレータ (gterm) を起動しておく必要があります。

† Eclipse 上の開発では、通常はコンソールからコマンドを手で入力しなくても、Eclipse のダイアログからプログラムの転送や実行ができるようになっています。しかしこの場合でも Eclipse と T-Kernel 2/x86 の間の通信はコンソールを使いますので、通信が確立されていないとプログラムの転送や実行もできません。

したがって te_vcom や gterm を必ず起動しておく必要があります。とくに Eclipse を起動した直後や、ワークスペースを切り替えた直後は、te_vcom や gterm は起動されていないので、毎回起動する必要があります。

Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト「[/SYS]%'が表示されることを確認します。

† 複数の端末ソフトを同時に T-Kernel 2/x86 に接続しないでください。具体的には、Eclipse 上で te_vcom や gterm を起動する際は、ハイパーターミナルや TeraTerm などの他の端末ソフトは接続しない状態にしてください。また te_vcom や gterm を 2 個以上重ねて起動しないでください。

【トラブルシューティング】CLI のプロンプトが出ていない場合 (T-Monitor のプロンプト「TM>」が出ているなど) は、ターゲット側にリセットをかけて再起動してください。仮想環境の場合は VMware Player の上辺にある「Player」「Troubleshoot」「Reset」でリセットがかかります。

3.2 コンソールの利用

T-Kernel 2/x86 は開発に便利な各種ツール類が付属しています。プログラムの転送や実行については、gterm のコンソールからコマンドを手で入力しなくても、Eclipse のダイアログからプログラムの転送や実行ができるようになっています。しかしそれ以外のさまざまな用途のために、必要に応じて gterm のコンソールからこれらのツール類をご利用ください。

CLI (Command Line Interpreter)

プロセススペースのコマンドラインインタプリタです。開発に必要な、特にファイル関連操作などの強力な機能を提供します。プロンプトは標準では「[/SYS]%」です。詳しくは T-Kernel 2/x86 評価キット CD-ROM 内の『開発ツール説明書』の「3. CLI 説明書」をご覧ください。

† CLI のプロンプトは、cd コマンドでチェンジディレクトリすると、その変更先ディレクトリ名になります。

CLI 上で動作するユーティリティ

hdpart (区画作成)、format (フォーマット) などの各種ユーティリティーが CLI 上で使える外部コマンドとしてシステムディスク上に格納されています。詳しくは『開発ツール説明書』の「4. ユーティリティー」、「5. UNIX(ファイル)エミュレータコマンドツール」、「6. ネットワークユーティリティー」をご覧ください。

IMS (Initial Monitor System)

T-Kernel ベースのモニタです。プロンプトは標準では「[IMS]%」です。詳しくは『開発ツール説明書』の「2. IMS」をご覧ください。

T-Monitor

最も基本的なモニタです。ハードウェアレベルの操作などに威力を発揮します。プロンプトは「TM>」です。

T-Monitor のコマンド一覧は『T-Monitor 仕様書』の「3.3. コマンド一覧」をご覧ください。さらに T-Kernel 2/x86 で追加された機能もあります。追加されたコマンドは『実装仕様書』の「2. T-Monitor 実装仕様」をご覧ください。

以下に CLI と T-Monitor の簡単な実習例を示します。

(1) CLI のプロンプト確認

Eclipse のコンソールウィンドウの gterm 上で ↵(Enter) キーを何回か押して、CLI のプロンプト「[/SYS]%」が表示されることを確認します。

```
[/SYS]%
```

(2) CLI コマンド一覧

CLI の “?” コマンドを入力して、CLI のコマンド一覧を表示させます。

```
[/SYS]% ? ←
                CLI コマンド一覧
att      det      eject   cd      ls      fs      ...(後略)...
```

(3) CLI コマンド詳細

CLI の “?” コマンドを使って “ls” コマンドの詳細ヘルプを表示させます。

```
[/SYS]% ? ls ←
ls [-f] [-F] [-l|-t] [<パス名>..]
<パス名>のファイルの直下にあるファイル一覧を表示する
...(後略) ...
```

(4) ファイル一覧

CLI の “ls” コマンドを使ってファイルの一覧を表示させます。

```
[/SYS]% ls ←
SBOOT          KERNEL.SYS     SYSCONF       ... (後略) ...
```

(5) T-Monitor への移行

CLI の 「#」 コマンドを使って T-Monitor へ移行します。

```
[/SYS]% # ←
TM>
```

(6) T-Monitor コマンド一覧

T-Monitor の “?” コマンドを使って T-Monitor のコマンド一覧を表示させます。

```
TM> ? ←
--- Command List :   "? command" for details ---
DumpByte/Half/Word(D/DB/DH/DW) ... (後略) ...
```

(7) レジスタ表示

T-Monitor の “R”(Register) コマンドを使って CPU レジスタの状態を表示させます。

```
TM> r ←
EIP : 001117D7  EFLAGS: 00001246  IF:1 ... (後略) ...
```

(8) T-Monitor からの復帰

T-Monitor の “G” (Go) コマンドを使って CLI へ復帰します。

```
TM> g ←
[/SYS]%
```

3.3 プロセスベースと T-Kernel ベース

T-Kernel 2/x86 上のプログラムは、大きく分けると「プロセスベースのプログラム」と「T-Kernel ベースのプログラム」の二つに分類されます。(このほか「モニタベースのプログラム」もありますが、本書では扱いません)

プロセスベースのプログラム

T-Kernel Extension 上で動作するプログラムです。

一般のアプリケーションを作成するのに向いています。

メモリ保護が効くので、プロセスに不具合があってもシステム全体に悪影響を及ぼす可能性は少なくなります。

メモリ空間としては、一つのプロセスに対して一つの独立したローカル空間が割り当てられ、そのローカル空間上で動作します。ただしプロセス内にサブタスクを生成することができ、その場合は一つのプロセス内の複数のタスクが同じ空間を共有します。

プロセスの実行では、まず `main()` 関数が実行され、`main()` 関数が終了するとプロセスも終了します。

プロセスベースのプログラムから使える API については、T-Kernel 2/x86 評価キット CD-ROM 内の以下のドキュメントをご参照ください。

- 『T-Kernel Extension 説明書』
プロセスやタスクなどの基本機能、ファイル、イベントなどの仕様書です。
- 『PMC T-Shell 説明書』、『PMC T-Shell プログラミング解説書』
画面描画、フォント、GUI、ネットワーク (TCP/IP) などの仕様書と解説書です。
† T-Shell の機能は基本的にはプロセスベース専用です。
- 『ライブラリ説明書』
C 言語標準ライブラリなどの説明書です。`fopen()` などのファイル操作ライブラリ (`stdio.h`) も含んでいます。
- 『デバイスドライバ説明書』
デバイスドライバを呼び出す場合に必要となる仕様書です。

T-Kernel ベースのプログラム

T-Kernel の機能を直接使うプログラムです。

ハードウェア制御や割込みなどを扱うことができるため、デバイスドライバなどを作成するのに向いています。

メモリ保護は効きません。

メモリ空間は全体で一つの空間を共有します。システムの共有空間内に複数の T-Kernel ベースのプログラムがロードされ常駐する形です。

T-Kernel ベースのプログラムは `lodspg` コマンドで共有空間上にロードします。この時に `main()` 関数が呼ばれ、第一引数 `ac` は渡される引数の個数を示します。通常この中でタスクやハンドラの生成などの初期化処理を行います。

また `unlspg` コマンドでアンロードすることができます。この時にも `main()` 関数が呼ばれ、第一引数 `ac` はマイナスの値です。通常この中で、最初に生成したタスクやハンドラの削除などの後処理を行います。

T-Kernel ベースのプログラムから使える API については、T-Kernel 2/x86 評価キット CD-ROM 内の以下のドキュメントをご参照ください。

- 『T-Kernel 仕様書』
タスクやセマフォ、割込みハンドラなどのリアルタイム OS としての基本機能 (T-Kernel/OS) と、デバイスドライバ管理などの機能 (T-Kernel/SM) などの仕様書です。
- 『ライブラリ説明書』
C 言語標準ライブラリなどの説明書です。
 - † ライブラリ説明書に書かれたライブラリは、一部にプロセス専用、T-Kernel ベース専用のものもありますが、それ以外はどちらからも利用可能です。
- 『デバイスドライバ説明書』
デバイスドライバを作成したりデバイスドライバを呼び出す場合に必要となる仕様書です。
 - † デバイスドライバを呼び出すのはプロセス専用でも T-Kernel ベースでもどちらでも可能ですが、デバイスドライバを作成できるのは T-Kernel ベースだけです。

3.4 「Hello, world」プロセス専用編

ここではまず有名な「Hello, world」プログラムを、プロセス専用で作成してみましょう。詳細については『GNU 開発環境 (Eclipse 版) 説明書』をあわせてご参照ください。

(1) ワークスペースの指定

プロセス専用のプログラムですので、Eclipse 起動時にワークスペースとして「`C:\te\bapp1`」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、ツールバーの「ファイル」「ワークスペースの切り替え」でワークスペースを「`C:\te\bapp1`」に切り替えてください。

- † ワークスペースを切り替えると Eclipse がいったん終了して `te_vcom` と `gterm` も終了しますので、再度 `te_vcom` と `gterm` を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの `gterm` で、`↵`(Enter) キーを何回か押して、CLI のプロンプト「`[/SYS]%`」が表示されることを確認しておきます。

(2) プロジェクトの新規作成

テンプレートを使ってプロジェクトを新規作成します。

「C/C++ プロジェクト」ビュー内で右クリックして (またはツールバーの「ファイル」から)、「新規」「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクト作成ダイアログで以下のように入力します (図 3.1)。

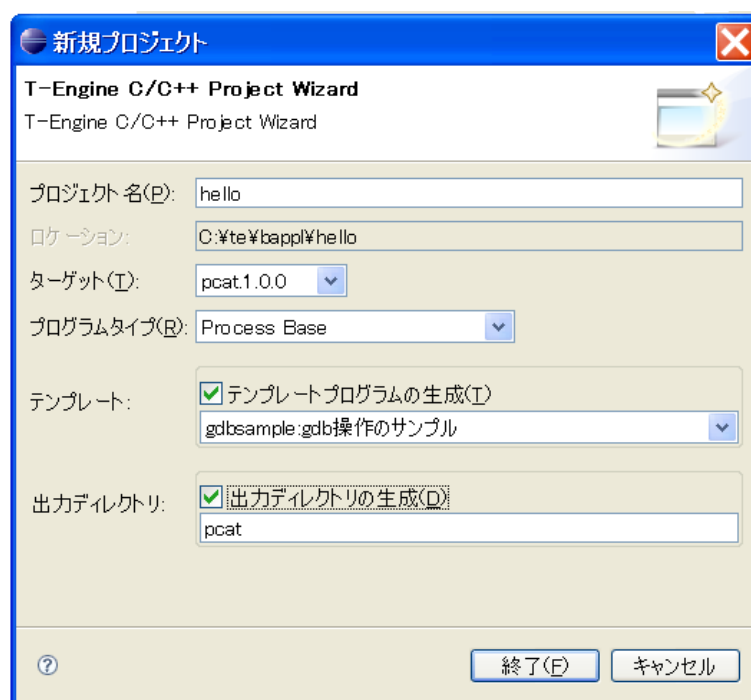


図 3.1 新規プロジェクト作成ダイアログ

- プロジェクト名:
プロジェクト名は自由につけることができますが、ここでは例えば「hello」とします。
- ターゲット:
複数の機種のパラグインをインストールしている場合は、ここでターゲットの機種を選択します。
- プログラムタイプ:
「Process Base」を指定します。ワークスペースを「C:%te%bappl」としていれば、「ワークスペース名による自動決定」のままでも自動的にプロセスベースになります。
- テンプレート:
チェックを入れて、テンプレートをここでは「gdbsample:gdb 操作のサンプル」を選択します。
- 出力ディレクトリの生成:
チェックを入れます。

最後に **終了(F)** をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

「C/C++ プロジェクト」ビュー内に「hello」プロジェクトが生成されますので、ダブルクリックして開くとソースプログラムなどが参照できます。

ソースを次のように修正して、「Hello, world」を作成しましょう。

- hello/src/main.c :

たとえば以下のように修正してください。

```
/* Hello, world (プロセススペース) */
#include <basic.h>      /* 基本共通ヘッダ */
#include <stdio.h>     /* printf() など */
W main( W ac, TC *av[] )
{
    printf( "Hello, world\n" );
    return 0;
}
```

修正したら、ツールバーの「ファイル」 「保管」で修正を保存します。

† W や TC などは、T-Kernel 仕様書で定義された型です。ここではプロセスにコマンドライン引数を渡す意味で使っています。また return 0 としているのは、プロセス終了時に終了コードとして値 0 を返すためです。

詳細は『T-Kernel Extension 説明書』などをご参照ください。

- hello/src/Makefile :

作成対象は「TARGET = hello」として、作成対象(メイクしてできる実行ファイル名)を hello に変更します。

ソースファイルは「SRC = main.c」とします。(sub.c は削除します)

修正したら、ツールバーの「ファイル」 「保管」で修正を保存します。

- hello/src/sub.c, hello/src/sub.h :

今回は不要なので削除します。右クリックメニューの中の「削除」で削除できます。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「hello/pcat/Makefile」を選択した上で、ツールバーの「プロジェクト」 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「hello/pcat/」の下に「hello」という名前で実行ファイルが生成されます(人が走っている図形で表示されます)。

【トラブルシューティング】「ビルダー起動中のエラー」が発生する場合は、「hello/pcat/Makefile」を選択しているか、いまいちどご確認ください。

【トラブルシューティング】「makedeps: Command not found」と出る場合は、/usr/local/bin/perl に Perl が正しくインストールされていない可能性があります。Cygwin のインストールおよびインストール後の設定をご確認ください。

(5) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「hello/pcat/hello」を選択した上で、右クリックメニューの「実行」 「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「hello」の転送や実行方法などの設定が自動設定されます(図 3.2)。

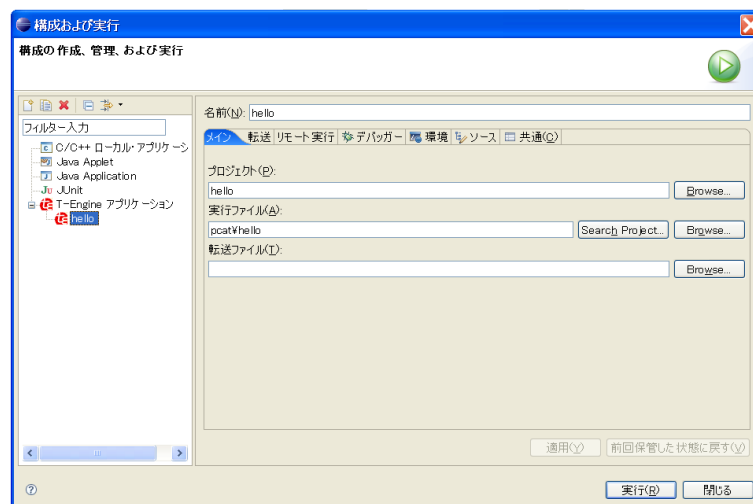


図 3.2 「構成および実行」ダイアログ

実行(R) をクリックすると、自動的に転送 (recv コマンド) と実行が行われ、「Hello, world」が表示されます(図 3.3)。

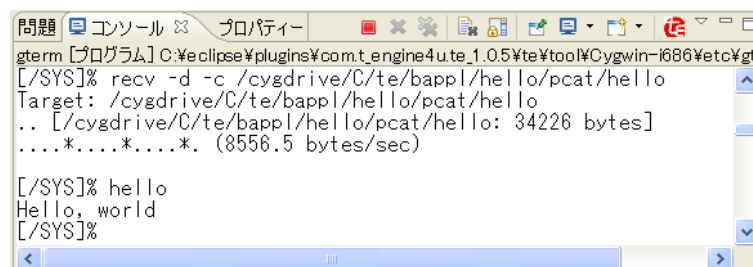


図 3.3 「Hello, world」の実行

【トラブルシューティング】ファイルの転送と実行は CLI を使っていますので、CLI との通信ができないとうまくいきません。

このため Eclipse を起動したら、最初にツールバーの外部ツールから te_vcom と gterm を順番に起動しておく必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [~/SYS]% が表示されることを確認しておきます。

【トラブルシューティング】「Can't Create (-1966080)」と表示される場合：このエラーは、起動ディスクがリードオンリーで書き込みができないことを示しています。書き込み不可能なディスクから起動しているるとこのような現象になります。

3.5 「Hello, world」T-Kernel ベース編

今度は「Hello, world」プログラムを T-Kernel ベースで作成してみましょう。ロード時に「Hello, world」と表示し、アンロード時に「See you again」と表示するようにします。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「C:¥te¥kapp1」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、ツールバーの「ファイル」「ワークスペースの切り替え」でワークスペースを「C:¥te¥kapp1」に切り替えてください。

- † ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認しておきます。

(2) プロジェクトの新規作成

テンプレートを使ってプロジェクトを新規作成します。

「C/C++ プロジェクト」ビュー内で右クリックして (またはツールバーの「ファイル」から)、「新規」「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
プロジェクト名は自由につけることができますが、ここでは例えば「hello2」とします。
- ターゲット:
複数の機種プラグインをインストールしている場合は、ここでターゲットの機種を選択します。
- プログラムタイプ:
「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kapp1」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。
- テンプレート:
チェックを入れて、テンプレートをここでは「sample:文字列出力サンプル」を選択します。
- 出力ディレクトリの生成:
チェックを入れます。

最後に 終了 (F) をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

「C/C++ プロジェクト」ビュー内に「hello2」プロジェクトが生成されますので、ダブルクリックして開くとソースプログラムなどが参照できます。

ソースを次のように修正して、「Hello, world」を作成しましょう。

- hello2/src/main.c :
たとえば以下のように修正してください。

```
/* Hello, world (T-Kernel ベース) */
#include <basic.h>      /* 基本共通ヘッダ */
#include <tk/tkernel.h> /* T-Kernel ヘッダ */
#include <stdio.h>      /* printf() など */

ER main( INT ac, UB *av[] )
{
    if (ac >= 0) {
        printf("Hello, world\n");
    } else {
        printf("See you again\n");
    }
    return E_OK;
}
```

修正したら、ツールバーの「ファイル」 「保管」で修正を保存します。

- hello2/src/Makefile :

作成対象は「TARGET = hello2」として、作成対象 (メイクしてできる実行ファイル名) を hello2 に変更します。

修正したら、ツールバーの「ファイル」 「保管」で修正を保存します。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「hello2/pcat/Makefile」を選択した上で、ツールバーの「プロジェクト」 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「hello2/pcat/」の下に「hello2」という名前で、リロケータブル形式の実行ファイルが生成されます (人が走っている図形では表示されませんが正常です)。このほか絶対アドレス形式の hello2.abs と、それを strip してサイズを小さくした hello2.trg も作成されます。

(5) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「hello2/pcat/hello2」を選択した上で、右クリックメニューの「実行」 「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「hello2」の転送や実行方法などの設定が自動設定されます。

実行 (R) をクリックすると、自動的に転送 (recv コマンド) と共有空間上にロード (lodspg コマンド) が行われ、「Hello, world」が表示されます。

(6) 共有空間の表示

Eclipse のコンソールウィンドウの gterm で、共有空間上にロードされているプログラムの一覧を「ref spg」コマンドで見て、新たに hello2 がロードされていることを確認します。

```
[/SYS]% ref spg ↵  
[ 1] - 0xc0261000 - 5  vmwdrv  
... (中略) ...  
[14] - 0xc056b000 - 4  hello2  
[/SYS]%
```

† システムプログラム ID やアドレスは条件によって異なります。

(7) アンロード

前項で表示された hello2 の システムプログラム ID (上の例では 14) を指定して unlspg コマンドで hello2 をアンロードします。「See you again」と表示されます。

```
[/SYS]% unlspg 14 ↵  
See you again  
[/SYS]%
```

4 実習用サンプルプログラム

T-Kernel 2/x86 評価キット付属 CD-ROM には以下の実習用サンプルプログラムのソースが付属しています。

- プロセスアプリケーションのサンプル

- (1) ジャグリング (お手玉)

画面上にジャグリング (お手玉) のアニメーションを表示するプログラムです。

- (2) 簡易ウェブサーバ

簡単なウェブサーバです。T-Kernel 2/x86 上のファイルをネットワーク経由で配信して、他のマシン上のブラウザから見ることができます。

- T-Kernel 2.0 の新機能を使った T-Kernel ベースのサンプル

T-Kernel 2.0 では、従来の T-Kernel に比べて、新たに次の機能が標準化されています:

- ・ マイクロ秒単位の時間指定
- ・ 物理タイマー
- ・ 大容量デバイス対応
- ・ キャッシュ操作、メモリアクセス権操作
- ・ 高速ロック

このうち T-Kernel 2/x86 評価キット付属 CD-ROM には次のサンプルが付属します。

- (3) マイクロ秒単位の時間指定サンプル

`tk_get_otm_u` (マイクロ秒単位の稼働時刻参照) と `tk_cre_cyc_u` (マイクロ秒単位の周期ハンドラ生成) を使ったサンプルプログラムです。

- (4) 物理タイマーのサンプル実装

物理タイマーを使って、タスク起動 (`tk_sta_tsk`) にかかる時間を測定するサンプルプログラムです。

物理タイマーはハードウェアや用途に応じてさまざまな方法や実装が考えられるため、T-Kernel 2/x86 では物理タイマーを標準実装とはせずに、実装例の一つとしてソースプログラムをサンプル提供します。このサンプルの物理タイマーの仕様や制限については、ソース先頭のコメントをご参照ください。

4.1 ジャグリング (お手玉)

画面上にジャグリング (お手玉) のアニメーションを表示する、プロセスベースのプログラムです。表示には PMC T-Shell の描画機能を使っています。

ブレークポイントをかけて止めながら少しずつ動かしたり、変数の値を変更するといった、デバッグのサンプルとしても使えます。

(1) ワークスペースの指定

プロセススペースのプログラムですので、Eclipse 起動時にワークスペースとして「C:¥te¥bapp1」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、ツールバーの「ファイル」「ワークスペースの切り替え」でワークスペースを「C:¥te¥bapp1」に切り替えてください。

- † ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認しておきます。

(2) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして(またはツールバーの「ファイル」から)、「新規」「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
ここでは「juggling」とします。
- ターゲット:
複数の機種のパラグインをインストールしている場合は、ここでターゲットの機種を選択します。
- プログラムタイプ:
「Process Base」を指定します。ワークスペースを「C:¥te¥bapp1」としていれば、「ワークスペース名による自動決定」のままでも自動的にプロセススペースになります。
- テンプレート:
チェックを入れません。
- 出力ディレクトリの生成:
チェックを入れます。

最後に 終了 (F) をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

Windows 標準の zip アーカイブ展開機能を使って、T-Kernel 2/x86 評価キット CD-ROM 内の実習用プログラム「juggling.zip」を「C:¥te¥bapp1」以下に展開します。

Eclipse の「C/C++ プロジェクト」ビュー内で右クリックして(またはツールバーの「ファイル」から)、「更新」を選択すると、展開されたソースディレクトリ「src」が表示されます。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「juggling/pcat/Makefile」を選択した上で、ツールバーの「プロジェクト」「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「juggling/pcat/」の下に「juggling」という名前で実行ファイルが生成されます (人が走っている図形で表示されます)。

(5) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「juggling/pcat/juggling」を選択した上で、右クリックメニューの「実行」「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「juggling」の転送や実行方法などの設定が自動設定されます。

実行 (R) をクリックすると、自動的に転送 (recv コマンド) と実行が行われ、ジャグリングのアニメーションが表示されます (図 4.1)。

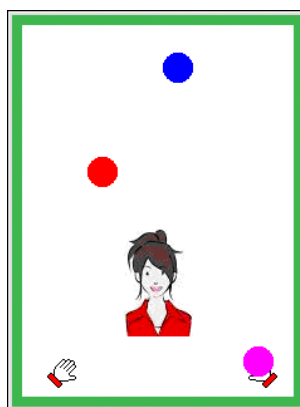


図 4.1 ジャグリングの実行画面

(6) デバッグ用のメイク

デバッグを行うためには、デバッグ情報付きの実行ファイルをメイクする必要があります。「pcat.debug」という名前のフォルダ上でメイクすれば、デバッグオプションがついて、デバッグ情報つきでメイクされます。

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「juggling/pcat.debug/Makefile」を選択した上で、ツールバーの「プロジェクト」「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「juggling/pcat.debug/」の下に「juggling」という名前で、デバッグ情報付きの実行ファイルが生成されます (虫の図形で表示されます)。

(7) デバッグの開始

デバッグするデバッグ情報付き実行ファイルとして「C/C++ プロジェクト」ビュー内の「juggling/pcat.debug/juggling」を選択した上で、右クリックメニューの「デバッグ」「構成およびデバッグ」を選択します。

「構成およびデバッグ」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「juggling」の転送やデバッグ方法などの設定が自動設定されます。

† (5) で作成した実行のための設定も残っていますが、今回はデバッグを行うので、必ず新規作成してください。

デバッグ (D) をクリックすると、転送とデバッグが開始され、デバッグパースペクティブに切り替わります。切り替えの確認ダイアログが出る場合は **はい (Y)** で切り替えてください。

(8) ブレークポイントの設定

現在 main() 関数の先頭で停止した状態です。

ここで main() 関数の最後の方の「if (i > 0) sleep(i);」という行にブレークポイントをかけましょう。この行の左端でマウスを右クリックしてメニューを出し、「ブレークポイントの切り替え (B)」でブレークポイントを設定します (図 4.2)。

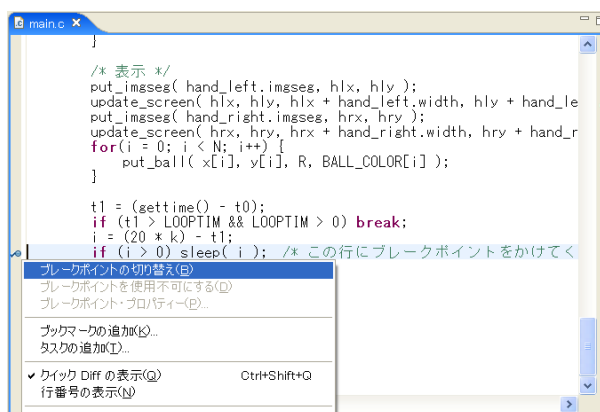


図 4.2 ブレークポイントの設定

(9) 実行再開

デバッグウィンドウ内の再開ボタン (緑の右の三角) をクリックして実行再開すると、ブレークポイントで停止するまで実行します (図 4.3)。

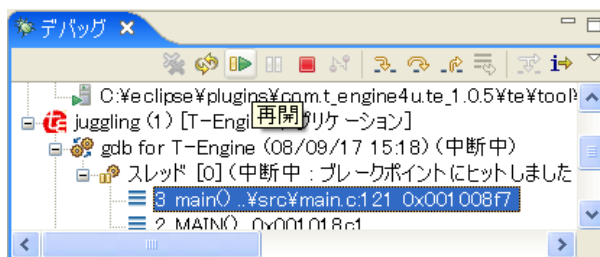


図 4.3 実行再開

再開を繰り返すと、ターゲット側の画面上では 1 ステップずつボールが動いていくのが分かります。

(10) 変数の値の変更

ブレークポイントで停止した状態で、変数ウィンドウ内の変数 N の値 (ボールの個数) を 3 から 6 に増やしてみましょう (図 4.4)。

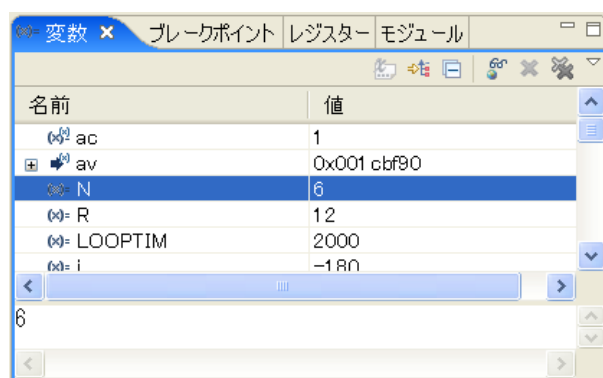


図 4.4 変数の値の変更

実行再開すると、ターゲット側の画面上ではボールの個数が 6 個に増えます。

同様に変数 R (ボールの半径) を変えて実行再開すると、ターゲット側の画面上のボールの大きさが変わります。

(11) デバッグの終了

デバッグウィンドウ内の終了ボタン (赤い四角) をクリックします。

さらにデバッグウィンドウ内の「終了したすべての起動を除去」のボタン (灰色の 2 重 ×) をクリックします。

最後に Eclipse のウィンドウの右上にある「》」を選択して「T-Engine 開発」を選択すれば、元の T-Engine 開発パースペクティブに戻ります。

4.2 簡易ウェブサーバ

簡単なウェブサーバです。T-Kernel 2/x86 上のファイルをネットワーク経由で配信して、他のマシン上のブラウザから見るができます。

プロセススペースのプログラムです。ネットワーク通信には PMC T-Shell の TCP/IP 機能を使っています。

(1) ワークスペースの指定

プロセススペースのプログラムですので、Eclipse 起動時にワークスペースとして「C:%te%bapp1」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、ツールバーの「ファイル」→「ワークスペースの切り替え」でワークスペースを「C:%te%bapp1」に切り替えてください。

- † ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認しておきます。

(2) ネットワークの設定

Eclipse のコンソールウィンドウの gterm 上で ↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認したのち、「netconf」コマンドを使ってネットワークの設定を行います。

ご使用のネットワーク環境が DHCP による IP アドレスの自動割当てを行う場合には標準設定の 0.0.0.0 のままで問題ありません。そうでない場合には、「netconf c」コマンドで T-Kernel 2/x86 に IP アドレスを割り当てて下さい。

```

[/SYS]% netconf c ↵
hostname   = ? tkernel ↵           — 自ホスト名
host ip    = 0.0.0.0 ? 192.168.0.70 ↵ — 自 IP アドレス
dns1name   = ? ↵                   — DNS サーバ 1
dns1 ip    = 0.0.0.0 ? ↵
dns2name   = ? ↵                   — DNS サーバ 2
dns2 ip    = 0.0.0.0 ? ↵
domain     = ? ↵                   — ドメイン名
gateway ip = 0.0.0.0 ? ↵           — ゲートウェイ
subnetmask = 255.255.255.0 ? ↵     — サブネットマスク
wlan       = none (n/a/i)? ↵       — 有線 LAN
[/SYS]%

```

- † netconf コマンドのかわりに、T-Shell の画面で右クリックして「ネットワーク設定」で設定することも可能です。

(3) ネットワークの動作確認

Eclipse のコンソールウィンドウの gterm 上で、ping コマンドでネットワークの接続を確認します。

```

[/SYS]% ping localhost ↵           — 自分への ping
localhost is alive <192.168.0.70> : 0 ms
[/SYS]% ping 192.168.0.1 ↵       — 他のマシンへの ping
192.168.0.1 is alive <192.168.0.1> : 0 ms

```

- † この例では T-Kernel 2/x86 の IP アドレスは 192.168.0.70、他のマシンの IP アドレスは 192.168.0.1 ですが、それぞれご使用のネットワーク環境に合わせて適宜読み替えて下さい。

- † 仮想環境 (VMware) の場合は、VMware Player のウィンドウ下辺にある「ネットワークアダプタ」アイコンをクリックして、「接続 (C)」を選択してください。さらに「設定 (S)...」を選択すると、ネットワーク接続方法として「ブリッジ」「NAT」「ホストオンリー」が選択できますので、ご利用のネットワーク環境に応じて適切に設定してください。例えば仮想マシン (VMware 上の T-Kernel 2/x86) も他の物理マシンと同様に LAN に直接接続する場合は、「ホストオンリー」を選択してください。

† 実機環境の場合は、ネットワークのハードウェアに応じた設定が必要です。PMC T-Shell の画面で右クリックして「システム環境設定」、「ネットワークアダプタ」で設定できます。

(4) 画像ファイルの転送

ウェブサーバで配信するコンテンツとなる画像ファイルを、ターゲット側に転送しておきます。

Eclipse のコンソールウィンドウの gterm 上で、次のように `recv` コマンドを使って、開発環境側のパソコン上にある画像ファイル (対応している形式は拡張子が `.jpg` または `.png` であるもの) を転送します。

例えば「`C:%tmp%abc.jpg`」を転送する場合は次のようにします。

```
[/SYS]% recv -c -d /cygdrive/c/tmp/abc.jpg ←
```

同様に画像ファイルを数枚転送しておきます。

(5) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして (またはツールバーの「ファイル」から)、「新規」、「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
ここでは「`webserv`」とします。
- ターゲット:
複数の機種プラグインをインストールしている場合は、ここでターゲットの機種を選択します。
- プログラムタイプ:
「Process Base」を指定します。ワークスペースを「`C:%te%bapp1`」としていれば、「ワークスペース名による自動決定」のままでも自動的にプロセススペースになります。
- テンプレート:
チェックを入れません。
- 出力ディレクトリの生成:
チェックを入れます。

最後に **終了 (F)** をクリックするとプロジェクトが自動生成されます。

(6) ソースの作成

Windows 標準の zip アーカイブ展開機能を使って、T-Kernel 2/x86 評価キット CD-ROM 内の実習用プログラム「`webserv.zip`」を「`C:%te%bapp1`」以下に展開します。

Eclipse の「C/C++ プロジェクト」ビュー内で右クリックして (またはツールバーの「ファイル」から)、「更新」を選択すると、展開されたソースディレクトリ「`src`」が表示されます。

(7) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「webserv/pcat/Makefile」を選択した上で、ツールバーの「プロジェクト」 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「webserv/pcat/」の下に「webserv」という名前で実行ファイルが生成されます (人が走っている図形で表示されます)。

(8) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「webserv/pcat/webserv」を選択した上で、右クリックメニューの「実行」 「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「webserv」の転送や実行方法などの設定が自動設定されます。

実行 (R) をクリックすると、自動的に転送 (recv コマンド) と実行が行われます。

(9) ブラウザからの閲覧

ネットワーク上のパソコンのブラウザから、T-Kernel 2/x86 の IP アドレスを指定して、ブラウザ上に画像一覧が配信されることを確認します。例えば T-Kernel 2/x86 の IP アドレスが 192.168.0.70 であれば、ブラウザから次のように URL を入力して下さい:

「http://192.168.0.70/index.html」

† index.html を指定すると、画像ファイルの一覧画面が自動作成されて配信されます。

(10) プロセス終了

最後にこのウェブサーバを終了させたい場合は、コンソールから ^C (Ctrl-C) を送ることでプロセスを終了させてください。Eclipse のコンソールウィンドウの上辺の右端付近の TE ログマークのボタンを使って ^C を送信することができます (図 4.5)。



図 4.5 Eclipse での ^C の送信

4.3 マイクロ秒単位の時間指定サンプル

tk_get_otm_u (マイクロ秒単位の稼働時刻参照) と tk_cre_cyc_u (マイクロ秒単位の周期ハンドラ生成) を使った T-Kernel ベースのサンプルプログラムです。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「C:¥te¥kapp1」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、メニューバーの「ファイル」「ワークスペースの切り替え」でワークスペースを「C:¥te¥kapp1」に切り替えてください。

- † ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認しておきます。

(2) システムタイマー割込み間隔 (TTimPeriod) の変更

T-Kernel 2.0 ではマイクロ秒単位の時間指定が可能ですが、時間の分解能はシステムタイマー割込み間隔 (TTimPeriod) となります。標準では 10 ミリ秒、つまり 10000 マイクロ秒です。ここでは「sysconf TTimPeriod ミリ秒 マイクロ秒」コマンドで 100 マイクロ秒に変更してみましょう。変更は再起動後に有効になります。

```
[/SYS]% sysconf TTimPeriod 0 100 ↵
+ 0: TTimPeriod 0 100
[/SYS]% exit ↵
<< EXIT cli >>
[IMS]% exit 1
[IMS]% exit -3 ↵
<< SYSTEM REBOOT >>
```

(3) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして (またはメニューバーの「ファイル」から)、「新規」「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
ここでは「usec」とします。
- ターゲット:
ターゲットの機種 (pcat) を選択します。
- プログラムタイプ:
「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kapp1」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。
- テンプレート:
チェックを入れません。
- 出力ディレクトリの生成:
チェックを入れます。

最後に **終了 (F)** をクリックするとプロジェクトが自動生成されます。

(4) ソースの作成

Windows 標準の zip アーカイブ展開機能を使って、T-Kernel 2/x86 評価キット CD-ROM の「jp\tutorial」フォルダにある実習用プログラム「usec.zip」を「C:\te\kapp1」以下に展開します。

Eclipse の「C/C++ プロジェクト」ビュー内で右クリックして(またはメニューバーの「ファイル」から)、「更新」を選択すると、展開されたソースディレクトリ「src」が表示されます。

(5) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「usec/pcat/Makefile」を選択した上で、メニューバーの「プロジェクト」 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「usec/pcat/」の下に「usec」という名前で、リロケータブル形式の実行ファイルが生成されます。

(6) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「usec/pcat/usec」を選択した上で、右クリックメニューの「実行」 「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「usec」の転送や実行方法などの設定が自動設定されます。

実行 (R) をクリックすると、自動的に転送 (recv コマンド) と共有空間上にロード (lodspg コマンド) が行われます。

```
[/SYS]% lodspg usec
SYSPRG usec [14] c0676000 - c067a000
TTimPeriod = 100[us]
#1: 8783629700[us]
#2: 8783629800[us]
#3: 8783629900[us]
#4: 8783630000[us]
#5: 8783630100[us]
[/SYS]%
```

(7) プログラムのアンロード

前項で表示された usec のシステムプログラム ID (上の例では 14) を指定して unlspg コマンドで usec をアンロードします。

```
[/SYS]% unlspg 14 ←
[/SYS]%
```

(8) TTimPeriod の復旧

システムタイマー割込み間隔 (TTimPeriod) を標準の 10 ミリ秒に戻しておきます。

```
[/SYS]% sysconf TTimPeriod 10 ←  
+ 0: TTimPeriod 10  
[/SYS]% exit ←  
<< EXIT cli >>  
[IMS]% exit 1  
[IMS]% exit -3 ←  
<< SYSTEM REBOOT >>
```

4.4 物理タイマーのサンプル実装

物理タイマーを使って、タスク起動 (tk_sta_tsk) にかかる時間を測定する T-Kernel ベースのサンプルプログラムです。物理タイマーの分解能はシステムタイマー割込み間隔 (TTimPeriod) とは無関係です。

(1) ワークスペースの指定

T-Kernel ベースのプログラムですので、Eclipse 起動時にワークスペースとして「C:¥te¥kapp1」を指定してください。

既に別のワークスペースで Eclipse が起動している場合は、メニューバーの「ファイル」「ワークスペースの切り替え」でワークスペースを「C:¥te¥kapp1」に切り替えてください。

† ワークスペースを切り替えると Eclipse がいったん終了して te_vcom と gterm も終了しますので、再度 te_vcom と gterm を順番に起動する必要があります。その上で Eclipse のコンソールウィンドウの gterm で、↵(Enter) キーを何回か押して、CLI のプロンプト [/SYS]% が表示されることを確認しておきます。

(2) プロジェクトの新規作成

プロジェクトを新規作成します。今回はテンプレートは使用しません。

「C/C++ プロジェクト」ビュー内で右クリックして (またはメニューバーの「ファイル」から)、「新規」「T-Engine C/C++ プロジェクト」を選択して、新規プロジェクトダイアログで以下のように入力します。

- プロジェクト名:
ここでは「ptimer」とします。
- ターゲット:
ターゲットの機種 (pcat) を選択します。
- プログラムタイプ:
「T-Kernel Base」を指定します。ワークスペースを「C:¥te¥kapp1」としていれば、「ワークスペース名による自動決定」のままでも自動的に T-Kernel ベースになります。

- テンプレート:
チェックを入れません。
- 出力ディレクトリの生成:
チェックを入れます。

最後に **終了 (F)** をクリックするとプロジェクトが自動生成されます。

(3) ソースの作成

Windows 標準の zip アーカイブ展開機能を使って、T-Kernel 2/x86 評価キット CD-ROM の「jp\tutorial」フォルダにある実習用プログラム「ptimer.zip」を「C:\te\kapl」以下に展開します。

Eclipse の「C/C++ プロジェクト」ビュー内で右クリックして(またはメニューバーの「ファイル」から)、「更新」を選択すると、展開されたソースディレクトリ「src」が表示されます。

(4) メイク

メイクするターゲットとして「C/C++ プロジェクト」ビュー内の「ptimer/pcat/Makefile」を選択した上で、メニューバーの「プロジェクト」 「T-Engine Target の Make all」でメイクします。

メイクが成功すると、「ptimer/pcat/」の下に「ptimer」という名前で、リロケータブル形式の実行ファイルが生成されます。

(5) 実行ファイルの転送と実行

転送する実行ファイルとして「C/C++ プロジェクト」ビュー内の「ptimer/pcat/ptimer」を選択した上で、右クリックメニューの「実行」 「構成および実行」を選択します。

「構成および実行」ダイアログが表示されますので、「T-Engine アプリケーション」で右クリックして「新規」を選択します。「ptimer」の転送や実行方法などの設定が自動設定されます。

実行 (R) をクリックすると、自動的に転送 (recv コマンド) と共有空間上にロード (lodspg コマンド) が行われます。

```
[/SYS]% lodspg ptimer
SYSPRG ptimer [14] c0676000 - c067a000
Physical timer resolution: 167.8[MHz] = 6.0[ns]
GetPhysicalTimerCount: 30.2[count] = 179.7[ns]
GetPhysicalTimerCount + tk_sta_tsk: 142.8[count] = 850.9[ns]
tk_sta_tsk: 112.6[count] = 671.2[ns]
[/SYS]%
```

上記の例ではタスク起動 (tk_sta_tsk) に約 670 ナノ秒 (0.67 マイクロ秒) かかっています。物理タイマーによる時刻測定 (GetPhysicalTimerCount) そのものにも時間がかかるため、その時間を引いた値です。

† 上記の結果はハードウェアやソフトウェアの条件に依存します。例えば仮想環境 (VMware) では一般に上記に比べて遅くなります。

(6) プログラムのアンロード

前項で表示された `ptimer` のシステムプログラム ID (上の例では 14) を指定して `unlspg` コマンドで `ptimer` をアンロードします。

```
[/SYS]% unlspg 14 ←  
[/SYS]%
```

索引

	C		
CLI		29	
Cygwin		18	
	D		
df コマンド		15	
	E		
Eclipse		21	
	F		
formatsys コマンド		14	
	G		
gterm		28	
	H		
hdpart コマンド		13	
	I		
IMS		29	
	J		
Java 実行環境		21	
	L		
lodspg コマンド		32	
ls コマンド		30	
	N		
netconf コマンド		44	
	P		
Perl		21	
ping コマンド		44	
	R		
recv コマンド		45	
	T		
T-Kernel ベース		31	
T-Monitor		29	
TCP/IP		43	
			te_vcom
			28
			Tera Term Pro
			9
			T-Kernel 2.0
			39
			TTimPeriod
			47
	U		
			unlspg コマンド
			32
	V		
			VMware Player
			6
			VMwareGateway
			7
	あ		
			ウェブサーバ
			43
	か		
			仮想環境
			5
			画面サイズ
			16
			クロス開発
			5
			コンソール
			29
	さ		
			サービス
			7
			実機環境
			5
			実行再開
			42
			ジャグリング
			39
	た		
			デバッグ情報
			41
	は		
			ハイパーターミナル
			9
			物理タイマー
			39
			ブレークポイント
			42
			プロジェクト
			32
			プロセスベース
			31
	ま		
			マイクロ秒
			39
			メモリ保護
			31
	ら		
			リセット
			29

わ
ワークスペース 23

はじめてみよう T-Kernel 2/x86

Version 2.0.0

パーソナルメディア株式会社

Web: <http://www.t-engine4u.com/>

E-Mail: te-sales@personal-media.co.jp

Copyright © 2008–2011 by Personal Media Corporation
