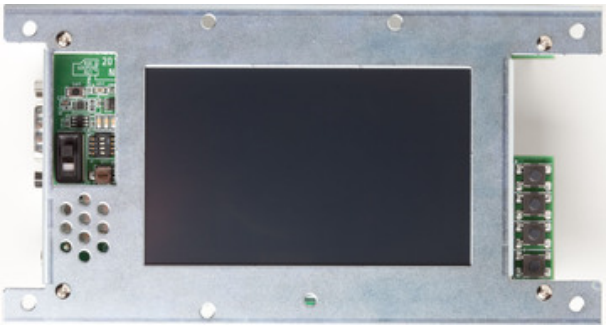
はじめてみよう
リファレンスキット



Version 1.03.01

パーソナルメディア株式会社

目 次

修正履歴

改版	摘要
1.03.01	誤植修正
1.03.00	表題を変更
1.02.00	Cygwin、Eclipse、USB-UART ドライバを更新
1.01.00	USB-UART ドライバを「T-Kernel 2.0 リファレンスキット」の CD に同梱
	物理タイマのサンプル実装を追加
1.00.00	T-Kernel 2.0 リファレンスキット向けに新規作成

はじめに

本書では、はじめてリファレンスキットをご利用になる方を対象に、リファレンス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とおり体験できます。

なお、開発環境としては Linux も利用可能ですが、本書では Windows で Eclipse を用いた開発手順をご紹介します。

1 ターゲット側実行環境のインストール

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。リファレンスキットにおいても、ソフトウェアの開発をクロス開発で行います。(図 1.1)

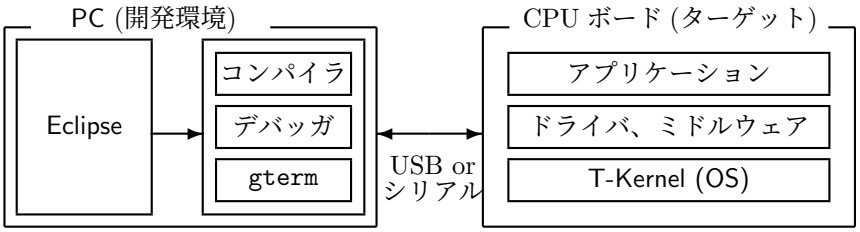


図 1.1 クロス開発

そこでまずこの章では、ターゲット側実行環境 (OS、ドライバ、ミドルウェア等) をリファレンスキット用の microSD にインストールし、次に 2 章でパソコン側の開発環境のインストールを行います。詳細については「T-Kernel 2.0 リファレンスキット」の CD 内の『T-Kernel 2.0 リファレンスキット取扱説明書』の「1 章 はじめに」をあわせてご参照ください。

リファレンスボードの部品配置を図 1.2 に示します。

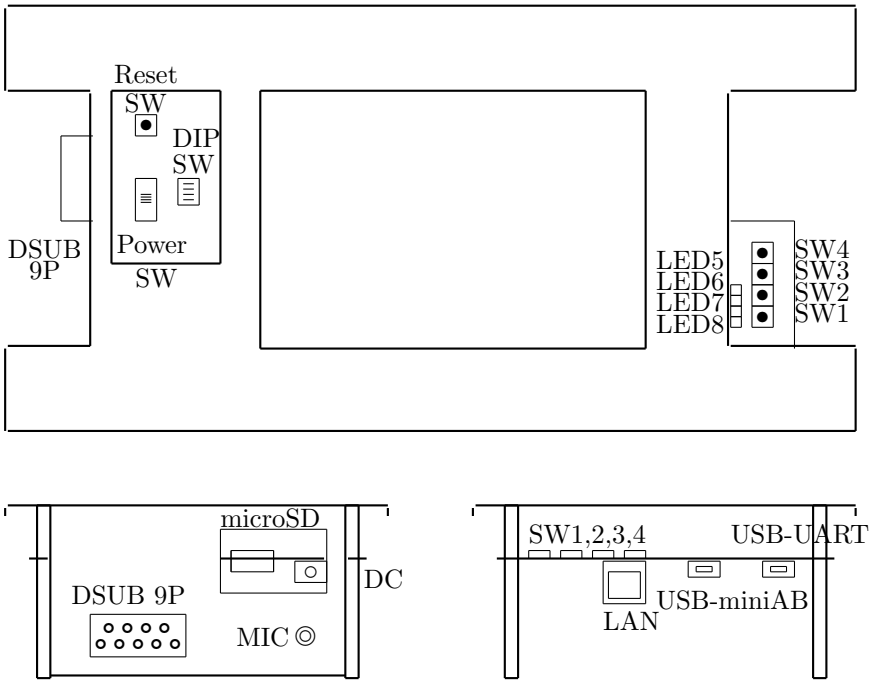


図 1.2 リファレンスキットの CPU ボード

1.1 コンソール接続

リファレンスボードのコンソールは USB とシリアル (RS–232C) のどちらでも接続可能です。

(1) 接続

○ USB 接続の場合

USB 接続の場合、初回は Windows にデバイスドライバをインストールする必要があります。

1. Windows に管理者権限でログオンします。
2. 「T-Kernel 2.0 リファレンスキット」の CD 内の「`common¥soft`」フォルダ以下にある USB-UART (CP210x) ドライバのアーカイブ `CP210x_VCP_Windows.zip` を展開し、次のどちらかを起動してインストールを行います。

Windows 32 ビット版用	`CP210xVCPInstaller_x86.exe`
Windows 64 ビット版用	`CP210xVCPInstaller_x64.exe`

   ▷ 最新版が必要な場合は、Silicon Laboratories 社の以下の URL からダウンロードできます。
   `http://www.silabs.com/Support%20Documents/Software/CP210x_VCP_Windows.zip`

3. AC アダプタからリファレンスボードの DC ジャックに電源を供給し、POWER SW を上にスライドさせて電源を入れます。開発用パソコンの USB ポートとリファレンスボードの USB-UART コネクタを接続します (図 1.3)。

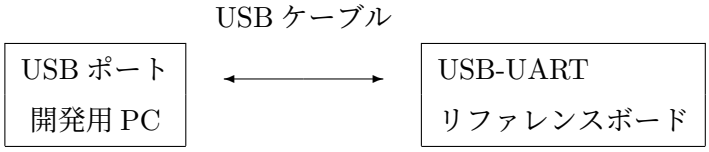


図 1.3 USB 接続

4. Windows がリファレンスボードを検出し、デバイスドライバをインストールします。
5. デバイスマネージャを開きます。
   - Windows 8、7、Vista の場合: コントロールパネル→システムとセキュリティー→システム→デバイスマネージャ
   - Windows XP の場合: コントロールパネル→パフォーマンスとメンテナンス→システム→ハードウェア→デバイスマネージャ

   デバイスマネージャの「ポート (COM と LPT)」の下に「Silicon Labs CP210x USB to UART bridge (COMn)」がインストールされていることを確認します。この「COMn」が COM ポート番号です。

USB 接続の場合の注意事項

- PC と リファレンスボードは直接 USB ケーブルで接続してください。USB ハブを介して接続すると問題が生じる場合があります。
- USB 接続での起動、再起動、リセットは、SW2 を押しながら行ってください。出荷時の設定では、SW2 を押さない場合はシリアル接続となります。なお、9 ページの手順で USB 接続用の ROM 情報をフラッシュROM に書き込めば、SW2 を押さずに USB 接続が可能になります。
- 再起動後に端末ソフトとの通信が切れる場合は、端末ソフトをいったん終了して再起動してください。

○ シリアル接続 (RS–232C) の場合

開発用パソコンの COM ポートとリファレンスボードの DSUB 9P コネクタをシリアル (RS–232C) クロスケーブルで接続します (図 1.4)。

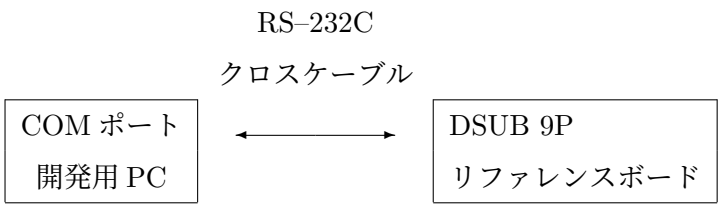


図 1.4　シリアル接続

(2) リファレンスボードの起動

AC アダプタからリファレンスボードの DC ジャックに電源を供給し、POWER SW を上にスライドさせて電源を入れます。

USB 接続の場合は、SW2 を押しながら電源を入れてください。

(3) 端末ソフトの起動

パソコン上には端末ソフト (「Tera Term」や「ハイパーターミナル」など) が必要です。

▷ Windows 8、7、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」を下記からダウンロード、インストールしてください。

`http://ttssh2.sourceforge.jp/`

▷ この章では最初のセットアップ時には Tera Term などの端末ソフトを使いますが、次章以降で Eclipse の開発環境をインストールすれば、端末ソフトを使うかわりに Eclipse 上の `gterm` を使ってコンソール操作ができます。ただし、最初のセットアップ時には、接続がうまくいかない場合の原因切り分けのためにも、まずは Tera Term などの端末ソフトで接続を確認することをお勧めします。

- **Tera Term の場合**

  Tera Term 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。

  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。

(4) 通信条件の設定

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、ポートの通信条件の設定を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	XON/XOFF

▷ ハードウェアフロー制御 (RTS/CTS) はサポートしていません。必ず XON/XOFF を指定してください。

- **Tera Term の場合**

  Tera Term の起動時に (もしくは起動後にメニューバーの「ファイル」→「新規接続」を選択すると) 新規接続ダイアログが表示されます。「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。

  続いてメニューバーの「設定」→「シリアルポート」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、図 1.5 のように設定してください。

図 1.5 Tera Term の通信設定

- **ハイパーターミナルの場合**
  「接続方法」の欄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。

(5) 通信の確認

端末ソフト上で ↩ (Enter) キーを何回か押してみて、CLI のプロンプト (`[/SYS]%`) が表示されれば成功です (図 1.6)。

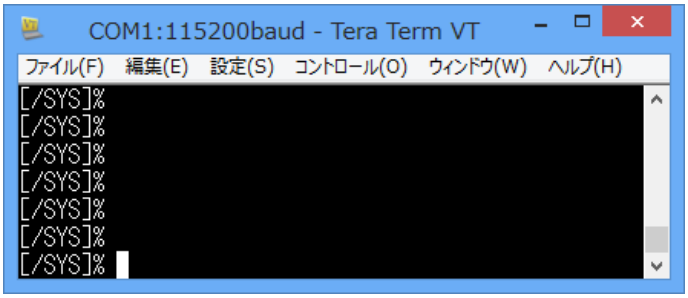


図 1.6 端末ソフト上に表示された CLI のプロンプト

(6) ROM 情報の書き込み (USB 接続時のみ)

「T-Kernel 2.0 リファレンスキット」の CD 内の「`jp¥soft`」フォルダ以下にある次のファイルは、USB 接続とシリアル接続を切り替える設定をフラッシュROM に書き込むものです。

`rominfo.mot`	シリアル接続 (SW2 押下起動では USB 接続)	出荷時設定
`rominfo-usb.mot`	USB 接続 (SW2 押下起動ではシリアル接続)	

出荷時の設定では、SW2 を押さずに起動するとシリアル接続になります。次のように USB 接続用の ROM 情報をフラッシュROM に書き込めば、SW2 を押さずに USB 接続が可能になります。

まず SW1 と SW2 を両方押しながらリセットスイッチを押して、T-Monitor を起動します。端末ソフト上で ↩ (Enter) キーを何回か押すと、T-Monitor のプロンプト (`TM>`) が表示されます。

T-Monitor の `FlashLoad` コマンドを実行します。

```
TM> FlashLoad↩
Copy Flash ROM Image to RAM Area
> Load S-Format Data of Flash ROM
```

しばらくして「`> Load S-Format Data of Flash ROM`」が表示されたら、「T-Kernel 2.0 リファレンスキット」の CD 内のファイル「`rominfo-usb.mot`」を送信します。

- **Tera Term の場合**
  メニューバーの「ファイル」→「ファイル送信」を選択して、送信するファイル (rominfo-usb.mot) を指定します。
- **ハイパーターミナルの場合**
  メニューバーの「転送」→「テキストファイルの送信」を選択して、送信するファイル (rominfo-usb.mot) を指定します。拡張子は.txt ではないので*.*を選択します。

```
Loaded:  70020000 -> 7002007F
Writing Flash ROM at 70020000 [1 blks] ...  wait
TM>
```

フラッシュROM への書き込みが終了して T-Monitor のプロンプト (TM>) が表示されれば成功です。次回の起動時から、SW2 を押さなくても USB 接続になります。

1.2 起動ディスク作成

(1) microSD カードの挿入

空の (または内容を消去しても構わない) microSD カードを用意して、リファレンスボードに挿入します。

▷ microSD のかわりに eMMC または USB ストレージも使用可能です。詳細は『T-Kernel 2.0 リファレンスキット取扱説明書』の「起動ディスクの作成」をご参照ください。

(2) パーティションの分割

端末ソフト上で「hdpart」コマンドを起動します。その際のデバイス名は「pcb」(microSD 全体) を指定します。以下に例を示します。

```
[/SYS]% hdpart pcb↩                                  — hdpart コマンドの起動
pcb   [C:485 H:64 S:63 B:1961984 (958 MB)]
No System    Boot  StartCHS     EndCHS    SecNo  SecCnt    Size
1  0b DOS     00    0:  1: 1  486: 38:38     63 1961921  957 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/Quit ? d↩                    — 区画削除
Delete PartNo (1-4) ? 1↩                               — 第 1 区画
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↩                               — 第 1 区画
Size [GB/MB/KB,All] (<957MB) ? a↩                      — 区画サイズ
No System    Boot  StartCHS     EndCHS    SecNo  SecCnt    Size
1  13 BTRON   00    0:  1: 1  486: 38:38     63 1961921  957 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↩                          — 第 1 区画
No System    Boot  StartCHS     EndCHS    SecNo  SecCnt    Size
1  13 BTRON   80    0:  1: 1  486: 38:38     63 1961921  957 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
** pcb: Updated Master Boot Block
[/SYS]%
```

(3) フォーマット

端末ソフト上で「format」コマンドを使って第 1 区画をフォーマットします。デバイス名は「pcb0」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 2GB 以上の区画の場合は「-x」オプションも指定してください。

```
[/SYS]% format -b pcb0 SYSTEM↩          — 第 1 区画のフォーマット
Format pcb0 [STD] SYSTEM
Logical Formatting...
Writing BootCode...
Disk Format Success.
[/SYS]%
```

(4) ROM ディスクのファイルコピー

端末ソフト上で ROM ディスクの全ファイルを microSD の第 1 区画にコピーします。

```
[/SYS]% att pcb0 A↩                      — 第 1 区画の接続
pcb0 -> A
[/SYS]% rcp -r /SYS /A/=↩          — ROM ディスクのファイルコピー
[/SYS]% det pcb0↩                        — 第 1 区画の切断
[/SYS]%
```

(5) リファレンスボードの再起動

「exit」コマンドと「exit -1」コマンドでリファレンスボードを再起動させます。

```
[/SYS]% exit↩
<< EXIT cli >>
[IMS]% exit     1
[IMS]% exit -1↩
<< SYSTEM RESET >>
   ⋮
[/SYS]%
```

端末ソフト上には、T-Kernel の起動メッセージが続いた後、CLI プロンプト ([/SYS]%) が表示されます。

(6) microSD からの起動確認

再起動後、CLI の「df」コマンドで、システムディスク (/SYS) が microSD カードの第 1 区画を表す「pcb0」になっていることを確認します。もし ROM ディスクを表す「rda」になっている場合は、うまくいっていませんので、手順を再度見直してください。

```
[/SYS]% df↩
PATH  DEV     TOTAL     FREE USED  UNIT MAXFILE NAME
/SYS  pcb0   980960   977108   0%  4096   65535 SYSTEM
[/SYS]%
```

(7) PMC T-Shell インストールパッケージのコピー

FAT フォーマットされた USB ストレージ (USB メモリなど) を用意します。PC 上で、「T-Kernel 2.0 リファレンスキット」の CD 内の「jp¥soft」フォルダ以下にある次のファイルを、USB ストレージのルート直下にコピーします。

ファイル名	内容
install_tshell.bz	PMC T-Shell インストールパッケージ

▷ USB ストレージを経由するかわりに、USB/シリアル転送 (XMODEM プロトコル)、もしくは LAN(FTP) によるファイル転送も可能です。詳細は『T-Kernel 2.0 リファレンスキット取扱説明書』の「PMC T-Shell の追加」をご参照ください。

(8) PMC T-Shell のインストール

USB ストレージをリファレンスボードの USB miniAB コネクタに接続します。(USB ストレージ側のコネクタを USB miniAB コネクタに変換するケーブルが必要です)

端末ソフト上で次のように操作します。

```
[/SYS]% att -m uda0 uda0↩              — USB ストレージの接続 (FAT 区画)
uda0 -> uda0
[/SYS]% expf -U /uda0/install_tshell.bz↩   — パッケージの展開
[/SYS]% install_tshell /SYS↩               — インストール
インストールが完了しました。再起動してください。
[/SYS]% rm -r install_tshell↩          — インストーラの削除
[/SYS]% det -u uda0↩                   — USB ストレージの切断
[/SYS]%
```

▷ 端末ソフトの文字コードが日本語 EUC でない場合は「インストールが完了しました。再起動してください」が文字化けしますが正常です。

「det」コマンドによる切断後、USB ストレージは抜いて構いません。

(9) PMC T-Shell の確認

「exit」コマンドと「exit -1」コマンドでリファレンスボードを再起動させます。


```
[/SYS]% exit↩
<< EXIT cli >>
[IMS]% exit     1
[IMS]% exit -1↩
<< SYSTEM RESET >>
    ⋮
[/SYS]%
```

端末ソフト上には、T-Kernel の起動メッセージが続いた後、CLI プロンプト ([/SYS]%) が表示されます。

一方、リファレンスボードの LCD 画面上には、図 1.7 のような PMC T-Shell のウィンドウ画面が表示されます。

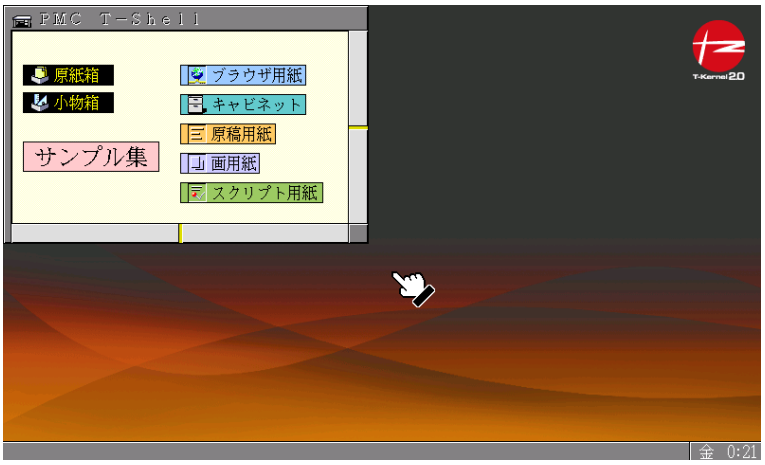


図 1.7　PMC T-Shell の初期ウィンドウ

画面下のシステムメッセージパネルをタッチするとメニューが表示されます。メニューの「小物」→「タッチパネル調整」で、タッチパネルの座標位置を調整できます。調整した結果は再起動後に反映されます。

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 2.0 リファレンスキットの 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。管理者以外のアカウントの場合、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 2.0 リファレンスキットの 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」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。

Cygwin システムパッケージ	`cygwin_d-X.X.X.zip`

▷ `X.X.X` にはファイルのバージョン番号が入ります。

(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-x86.exe` を起動します (図 2.1)。

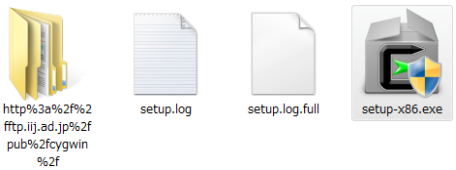


図 2.1 Cygwin インストーラの起動

インストーラが起動したら **次へ** 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 **次へ** 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)。

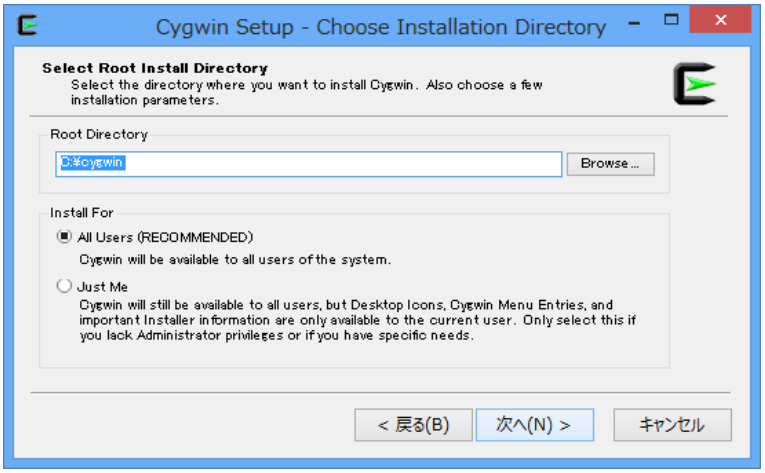


図 2.2　インストールディレクトリの選択

「Install For」は「All Users」を選択してください。これはデフォルトです。

次へ 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、**次へ** 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。

▷ Cygwin 1.7 系列の最初のインストール時には「Setup Alert」ダイアログが表示されます。内容をよく読んでから作業を行ってください。旧バージョンの Cygwin を使用している場合、何らかの影響が出る可能性があります。

「All」の右の「Default」をクリックして「Install」の状態にしてから **次へ** をクリックします (図 2.3)。

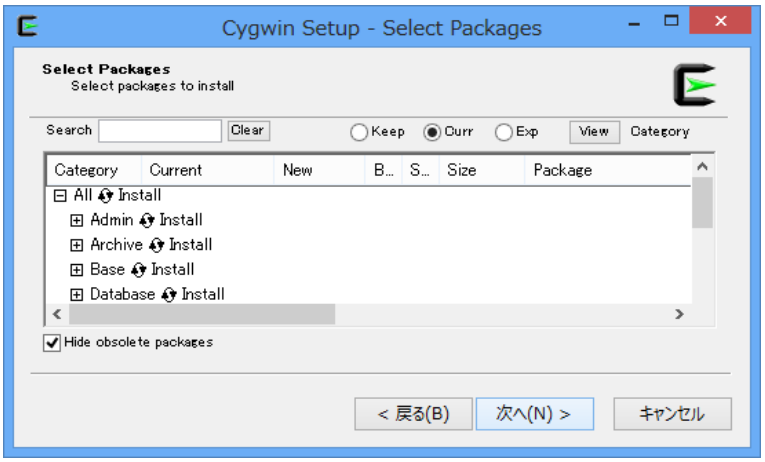


図 2.3　インストールするパッケージの選択

(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。

「プログラム互換性アシスタント」が表示される場合は、「このプログラムは正しくインストールされました」をクリックします。

(9) ホームディレクトリの初期化

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

(10) 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 ↩
```

(11) 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 ↩
```

(12) 日本語 EUC の設定

もし開発環境を実際に使うアカウントが今までの管理者アカウントと異なる場合は、exit コマンドで Cygwin のウィンドウを閉じ、実際に使うアカウントで Windows にログオンし直してください。デスクトップ上の「Cygwin Terminal」アイコンをダブルクリックして Cygwin を起動します。最初の起動時にホームディレクトリが初期化されます。

ホームディレクトリ上の .bashrc に「export LANG=ja_JP.eucJP」という行を追加します。
例えば次のように操作します。

```
$ cd ↩
$ echo export LANG=ja_JP.eucJP >> .bashrc ↩
```

また、Cygwin のウィンドウ上でマウスの右ボタンをクリックしてメニューを開き、「Options」を選択します。「Text」を選択し、「Locale」を「ja_JP」に、「Character set」を「eucJP」に設定して、OK をクリックします。

exit コマンドで Cygwin のウィンドウを閉じます。

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 2.0 リファレンスキット 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/` からダウンロードしてインストールしてください。

▷ Windows が 64 ビット版の場合でも、32 ビット用の Java 実行環境が必要です。
32 ビット用の Java 実行環境を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 2.0 リファレンスキット 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」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+CDT	`eclipse-cpp-kepler-R-win32.zip`

▷ 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。

(3) Eclipse の日本語化

T-Kernel 2.0 リファレンスキットの CD-ROM 内の日本語版ソフトウェアのフォルダ (`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」を実行してください。展開先は「`C:¥eclipse`」を指定してください。

日本語化プラグイン	`pleiades_1.4.0.zip`

続いて「`C:¥eclipse¥eclipse.ini`」をテキストエディタで開き、最後に次の 1 行を追加します。

```
-javaagent:plugins/jp.sourceforge.mergedoc.pleiades/pleiades.jar
```

▷ 改行コードが LF のため、メモ帳以外のテキストエディタをご使用ください。

(4) 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。

(5) 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 **キャンセル** 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 2.0 リファレンスキット」の CD 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 2.0 リファレンスキット」の CD 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」を実行してください。展開先は `C:¥eclipse` を指定してください。

プラットホーム共通部分	`com.t_engine4u.te.X.X.X.zip`
各機種対応部分	`com.t_engine4u.tl.em1d512_te.X.X.X.zip`

▷ X.X.X にはファイルのバージョン番号が入ります。

▷ 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。

(2) PMC T-Shell 開発用追加モジュールの展開

「T-Kernel 2.0 リファレンスキット」の CD 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てください。

PMC T-Shell 開発用追加モジュール	com.t_engine4u.tsh.em1d512_te.X.X.X.zip

▷ X.X.X にはファイルのバージョン番号が入ります。

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」を実行してください。展開先は
C:¥eclipse¥plugins¥com.t_engine4u.tl.em1d512_te.X.X.X_X.X.X¥te
を指定してください。

▷ 展開時にファイルの置き換え (上書き) の確認が出ますが、問題ありませんので置き換え (上書き) を選択してください。

(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」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)。

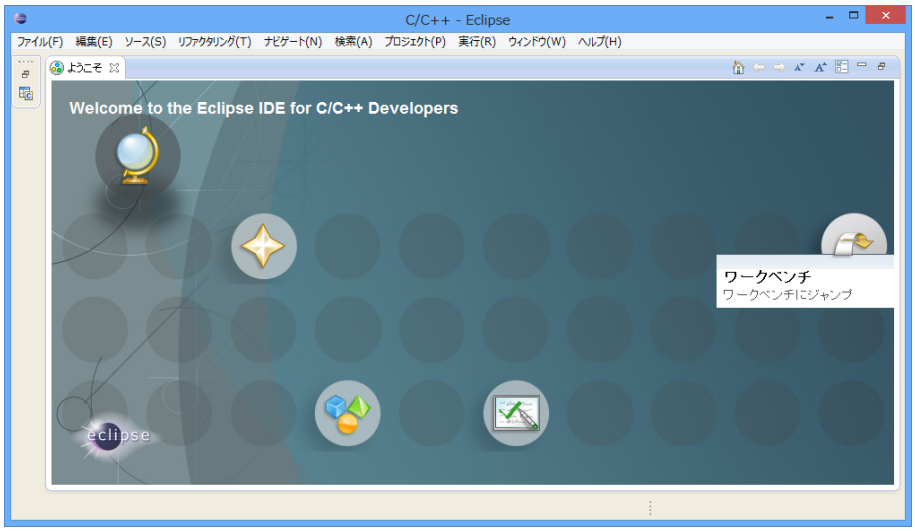


図 2.5 Eclipse の「ようこそ」ビュー

(6) T-Engine 開発パースペクティブを開く

メニューバーの「ウィンドウ」→「パースペクティブを開く」→「その他」 を選択します。ダイアログが表示されますので、その中の「T-Engine 開発」を選択して OK をクリックします。(図 2.6)

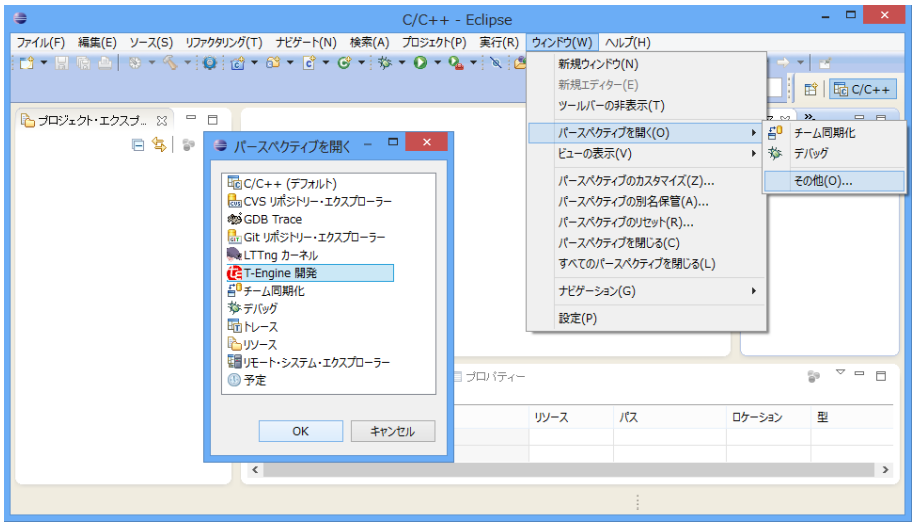


図 2.6 パースペクティブを開く

(7) T-Engine 開発環境の設定

メニューバーの「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)。

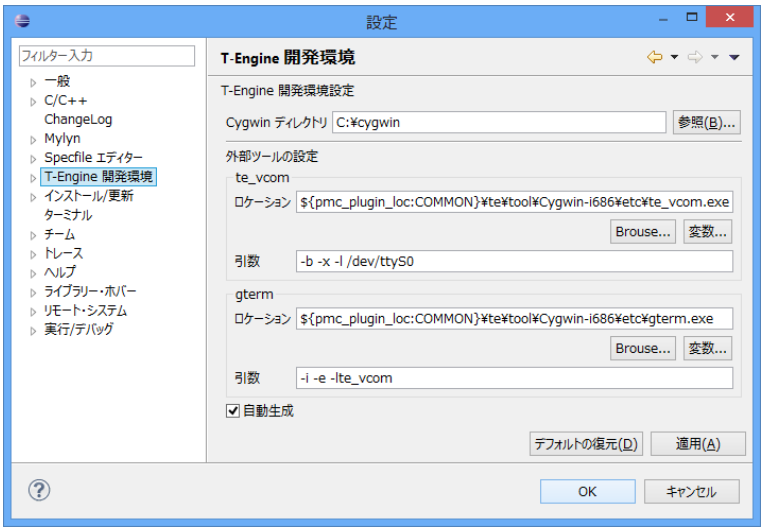


図 2.7 T-Engine 開発環境の設定

COM ポート番号として COMn を使う場合、`te_vcom` の引数として「`-b -x -l /dev/ttyS`$(n-1)$」を指定します。

- `-b` は通信速度 115200bps の指定です。
- `-x` はハードウェアフロー制御 (RTS/CTS) を行わない指定です。
- `-l` はマイナス・エルで、ラインデバイス名を指定します。COM ポート番号から 1 を引いた値がラインデバイス名になります。例えば COM1 であれば「`-b -x -l /dev/ttyS0`」、COM8 であれば「`-b -x -l /dev/ttyS7`」を設定してください。

▷ USB 接続の場合:
デバッグ時に問題が生じる可能性があります。
その場合は `te_vcom` のロケーションを次のように変更してください。

(変更前) `$pmc_plugin_loc:COMMON¥te¥tool¥Cygwin-i686¥etc¥te_vcom.exe`
↓
(変更後) `$pmc_plugin_loc:COMMON¥te¥tool¥Cygwin-i686¥etc¥te_vcom.1_13.exe`

▷ シリアル接続 (RS–232C) で市販の USB シリアルアダプタを使用する場合:
通信速度が低下する可能性があります。
その場合は `te_vcom` の引数に次のように追加してください。

(変更前) `-b -x -l /dev/ttyS0`
↓
(変更後) `-b -x -l /dev/ttyS0 -e c1024`

(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」はチェックを入れた状態にします。

- 「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。選択肢にない場合は直接「EUC-JP」とキー入力します。
- 「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。

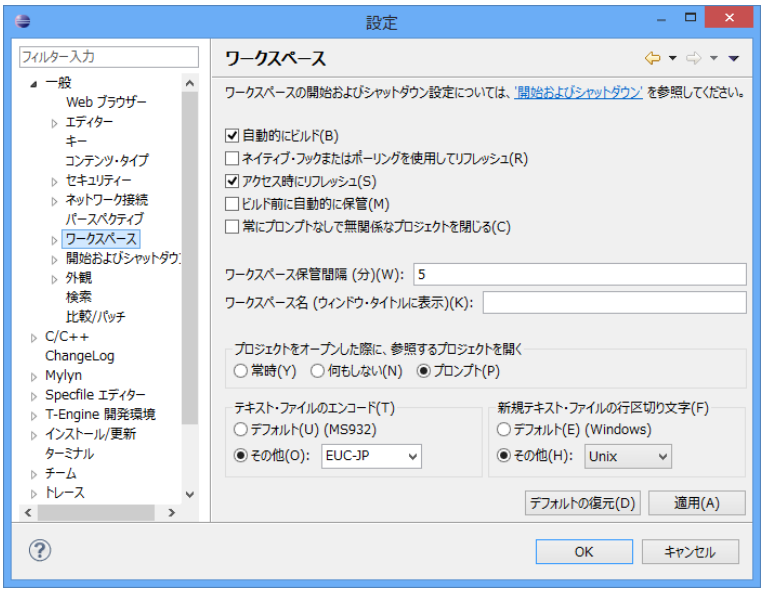


図 2.8　ワークスペースの設定

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¥bappl」での設定が終わったら、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¥kappl」(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。

(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。

メニューバーの「実行」→「外部ツール」→「外部ツールの構成」を選択します。外部ツール構成のダイアログが表示されたら、左側の「プログラム」を開くと te_vcom と gterm の設定が表示されます。

▷ 「te_vcom」とは別に「te_vcom(telnet)」も表示される場合がありますが、今回は使用しません。

まずは te_vcom を選択し、**実行** をクリックします (図 2.9)。

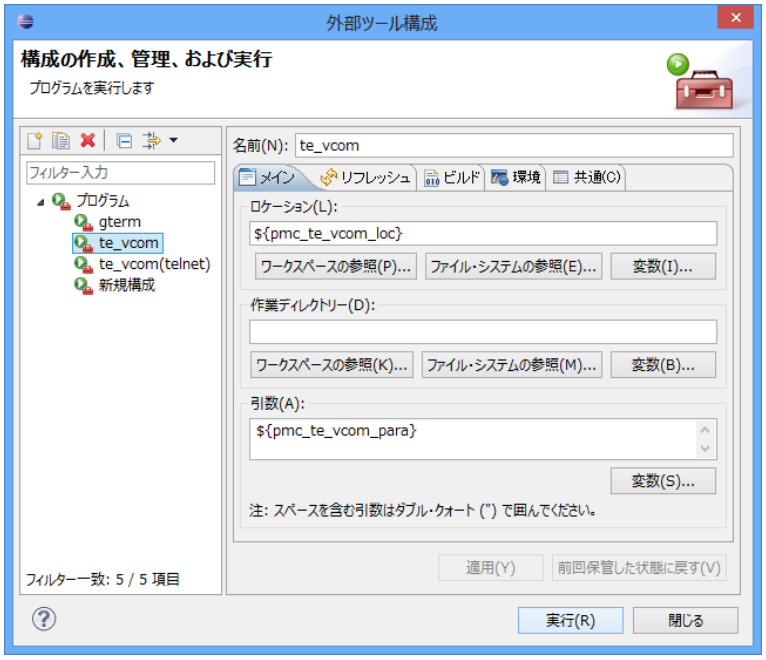


図 2.9　te_vcom の起動

続いて同様の手順で外部ツール構成のダイアログから 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。

ターゲット側のリファレンスボードが起動した状態で、コンソールビューの gterm 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 2.10)。

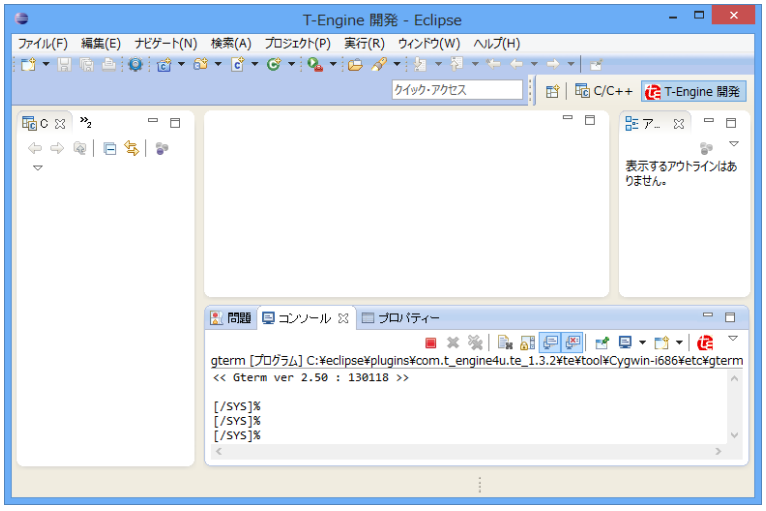


図 2.10　gterm 上に表示されたターゲット側プロンプト

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 (リファレンスボード) の起動

AC アダプタからリファレンスボードの DC ジャックに電源を供給し、POWER SW を上にスライドさせて電源を入れます。

リファレンスボードの LCD 画面に PMC 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¥bappl`」とします。

(3) `te_vcom` と `gterm` の起動

リファレンスボードと通信を行うには、メニューバーの「実行」→「外部ツール」から中継プログラム (`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 とリファレンスボード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` は起動されていませんので、毎回起動する必要があります。

Eclipse のコンソールビューの `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。

▷ 複数の端末ソフトを同時にリファレンスボード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、Tera Term やハイパーターミナル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、リファレンスボードのリセットスイッチを押してターゲット側を再起動してください。

3.2 コンソールの利用

リファレンスキットに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)

プロセスベ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 2.0 リファレンスキット」の CD 内の『開発ツール説明書』の「CLI 説明書」の章をご覧下さい。

- ▷ 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「ユーティリティ」、「UNIX(ファイル) エミュレータコマンドツール」「ネットワークユーティリティ」の各章をご覧下さい。

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「IMS」の章をご覧下さい。

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「コマンド一覧」の章をご覧下さい。さらに リファレンスキット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「T-Monitor 実装仕様」の章をご覧下さい。

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 のコンソールビューの 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) T-Monitor コマンド詳細

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

```
TM> ?  iw↩
InputWord(IW) port : Input Word from I/O port
```

(8) I/O 読み込み

T-Monitor の “iw” コマンドを使って 0xc0050010 番地 (GIO[31:0] 入力データ リード レジスタ) から 32 ビットの値を読み込んで、プッシュスイッチ SW1,2,3,4 の状態を確認します。


```
TM> iw c0050010↩    — SW1 押下
Port C0050010:W --> 00000101
TM> iw c0050010↩    — SW2 押下
Port C0050010:W --> 00000081
```

SW1	bit8	0x00000100
SW2	bit7	0x00000080
SW3	bit6	0x00000040
SW4	bit4	0x00000010

(9) 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

リファレンスボード上のプログラムは、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。

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 2.0 リファレンスキット」の CD 内の以下のドキュメントをご参照ください。

- **『PMC 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。
  - ▷ PMC 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。

- **『T-Engine 開発キット デバイスドライバ共通説明書』**
  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 2.0 リファレンスキット」の CD 内の以下のドキュメントをご参照ください。

- **『T-Kernel 2.0 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、一部にプロセスベース専用、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。
- **『T-Engine 開発キット デバイスドライバ共通説明書』**
  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ベースでも 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」プロセスベース編

ここではまず有名な「Hello, world」プログラムを、プロセスベース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¥bappl」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¥bappl」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 のコンソールビューの 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: 「hello」と入力します。
- ターゲット: ターゲットの機種 (em1d512_te) 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、「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 **完了** をクリックするとプロジェクト「hello」が自動生成されます。(図 3.1)

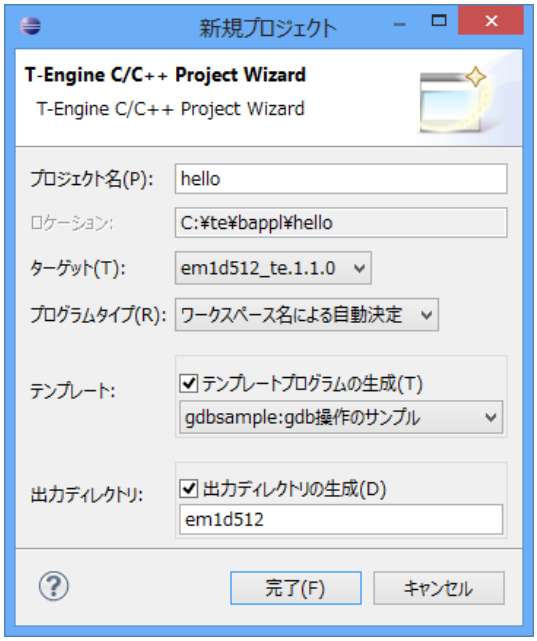


図 3.1 新規プロジェクト作成ダイアログ

(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 (プロセスベース) */
#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、メニューバーの「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、メニューバーの「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/em1d512.te/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、hello/em1d512.te/ の下に「hello」という名前で、実行ファイルが生成されます。

▷ プロセスの実行ファイルは、実行図形または虫図形で表示されます。リンクするライブラリの一部にデバッグ情報が付いている場合は虫図形で表示されますが、正常です。

【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/em1d512.te/Makefile」を選択しているか、いまいちどご確認ください。
また、「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/em1d512.te/hello」を選択した上で、右クリックメニューの「実行」→「実行の構成」を選択します。

実行構成ダイアログが表示されますので、「T-Engine アプリケーション (T-Monitor)」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)。

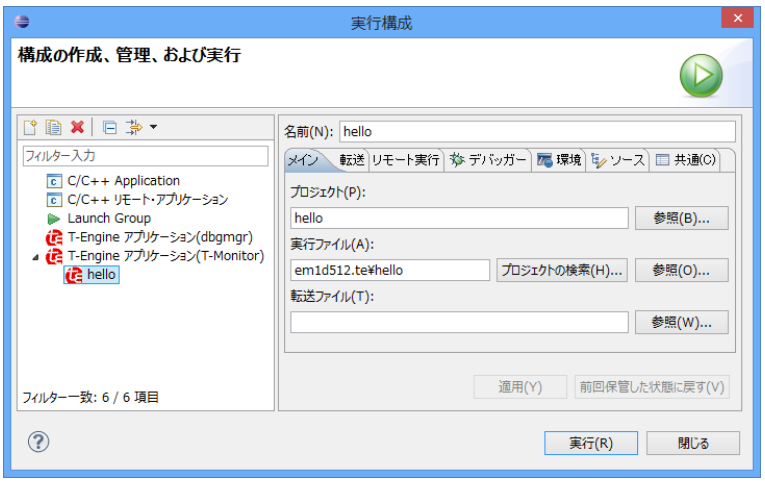


図 3.2 実行構成ダイアログ

▷ 「T-Engine アプリケーション (T-Monitor)」とは別に「T-Engine アプリケーション (dbgmgr)」も表示される場合がありますが、今回は使用しません。

実行 をクリックすると、自動的に転送 (recv コマンド) とプロセス実行が行われ、コンソール上に「Hello, world」が表示されます (図 3.3)。

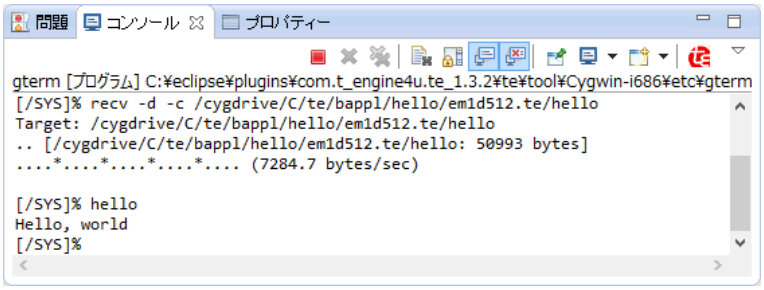


図 3.3 「Hello, world」の実行

【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、最初にメニュー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 のコンソールビューの 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。書込み不可能なディスクから起動してい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¥kappl」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¥kappl」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 のコンソールビューの 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

「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: 「hello2」と入力します。
- ターゲット: ターゲットの機種 (em1d512_te) 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¥kappl」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、「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 **完了** をクリックするとプロジェクト「hello2」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、メニューバーの「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、メニューバーの「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/em1d512.te/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、hello2/em1d512.te/ 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。

▷ リロケータブル形式の実行ファイルは、Eclipse ではバイナリファイルの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
「hello2/em1d512.te/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 (T-Monitor)」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が自動的に設定されます。
実行 をクリックすると、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。

コンソール上に「Hello, world」が表示されます。

(6) 共有空間の表示

Eclipse のコンソールビューの 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] - 0xd018b000 -  6  screen
… (中略) …
[13] - 0xd049d000 -  5  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 13) を指定して unlspg コマンドで hello2 をアンロードします。コンソール上に「See you again」と表示されます。

```
[/SYS]% unlspg 13↩
See you again
[/SYS]%
```


4 実習用サンプルプログラム

「T-Kernel 2.0 リファレンスキット」の CD 内のチュートリアルのページ (`jp¥tutorial`) 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ます。

- **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。

- **簡易ウェブサーバ**

  簡単なウェブサーバです。リファレンスボード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。

- **LED と割込み**

  LED の点滅と、スイッチによる割込みの検出を行う、T-Kernel ベースのプログラムです。

- **物理タイマのサンプル実装**

  物理タイマを使う、T-Kernel ベースのプログラムです。

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¥bappl`」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¥bappl`」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 のコンソールビューの `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

「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: 「`juggling`」と入力します。

- ターゲット: ターゲットの機種 (em1d512_te) 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 **完了** をクリックするとプロジェクト「juggling」が自動生成されます。

(3) ソースのインポート

Eclipse の「C/C++ プロジェクト」ビュー内に作成されたプロジェクト「juggling」を選択します。右クリックしてメニューを開き (またはメニューバーの「ファイル」から)、「インポート」を選択して「インポート」ダイアログを開きます。
「一般」の下にある「アーカイブファイル」を選択して **次へ** をクリックします。
ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、「T-Kernel 2.0 リファレンスキット」の CD 内の実習用プログラム「juggling.zip」を選択して **開く** をクリックします。**完了** をクリックします。
以上の操作により、ソースが juggling/src 以下にインポートされます。

(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/em1d512.te/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、juggling/em1d512.te/ の下に「juggling」という名前で、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
「juggling/em1d512.te/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 (T-Monitor)」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が自動的に設定されます。
実行 をクリックすると、転送 (recv コマンド) とプロセス実行が行われます。 リファレンスボードの画面上に、ジャグリングのアニメーションが表示されます (図 4.1)。

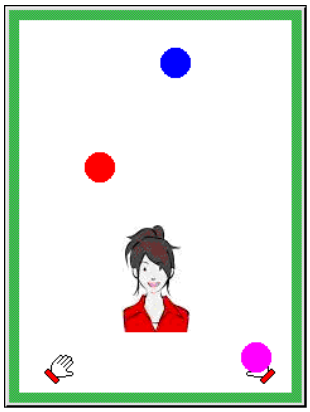


図 4.1 ジャグリングの実行画面

(6) デバッグ用のメイク

デバッグを行うためには、デバッグ情報つきの実行ファイルをメイクする必要があります。「em1d512.te.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/em1d512.te.debug/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、juggling/em1d512.te.debug/の下に「juggling」という名前で、デバッグ情報つきの実行ファイルが生成されます。

(7) デバッグの開始

デバッグするデバッグ情報つき実行ファイルとして「C/C++ プロジェクト」ビュー内の「juggling/em1d512.te.debug/juggling」を選択した上で、右クリックメニューの「デバッグ」→ 「デバッグの構成」を選択します。

デバッグ構成ダイアログが表示されますので、「T-Engine アプリケーション (T-Monitor)」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 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** 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」でブレークポイントを設定します (図 4.2)。

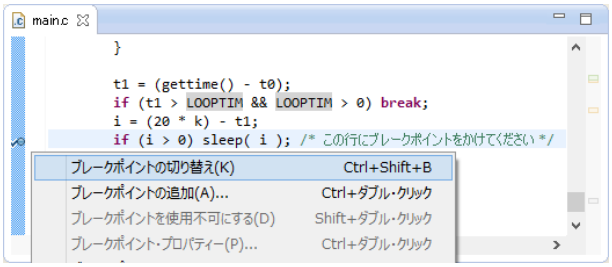


図 4.2　ブレークポイントの設定

(9) 実行再開

ツールバーの再開ボタン (またはメニューバーの「実行」→「再開」) 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)。

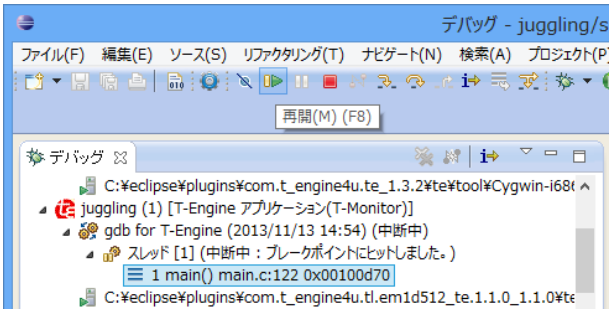


図 4.3　実行再開

再開を繰り返すと、リファレンスボードの画面上では 1 ステップずつボールが動きます。

(10) 変数の値の変更

ブレークポイントで停止した状態で、変数ビュー内の変数 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)。

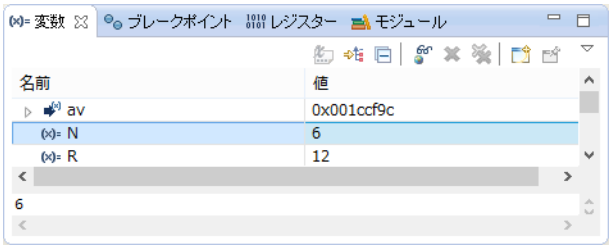


図 4.4　変数の値の変更

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

ツールバーの終了ボタン (■) (またはメニューバーの「実行」→「終了」) をクリックします。

さらにデバッグビュー内の「終了したすべての起動を除去」ボタン (✖) をクリックします。

最後に Eclipse のウィンドウの右上付近にある「T-Engine 開発」を選択すれば、元の T-Engin 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。リファレンスボード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。

プロセスベースのプログラムです。ネットワーク通信には 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¥bappl」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¥bappl」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 のコンソールビューの 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 のコンソールビューの 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 リファレンスボードに 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、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 のコンソールビューの 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 リファレンスボードの 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。

(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 のコンソールビューの 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

「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: 「webserv」と入力します。
- ターゲット: ターゲットの機種 (em1d512_te) 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 **完了** をクリックするとプロジェクト「webserv」が自動生成されます。

(6) ソースのインポート

Eclipse の「C/C++ プロジェクト」ビュー内に作成されたプロジェクト「webserv」を選択します。右クリックしてメニューを開き (またはメニューバーの「ファイル」から)、「インポート」を選択して「インポート」ダイアログを開きます。
「一般」の下にある「アーカイブファイル」を選択して **次へ** をクリックします。
ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、「T-Kernel 2.0 リファレンスキット」の CD 内の実習用プログラム「webserv.zip」を選択して **開く** をクリックします。**完了** をクリックします。
以上の操作により、ソースが webserv/src 以下にインポートされます。

(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/em1d512.te/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、webserv/em1d512.te/ の下に「webserv」という名前で、実行ファイルが生成されます。

(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/em1d512.te/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 (T-Monitor)」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が自動的に設定されます。
実行 をクリックすると、転送 (recv コマンド) とプロセス実行が行われます。

(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、リファレンスボードの 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。例えばリファレンスボードの 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 のコンソールビューの上辺の右端付近の TE ロゴ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)。

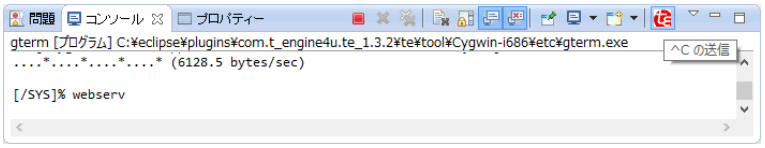


図 4.5　Eclipse での ^C の送信

4.3　LED と割込み

LED の点滅と、スイッチによる割込みの検出を行う、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¥kappl`」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¥kappl`」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 のコンソールビューの `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

「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: 「`ledint`」と入力します。
- ターゲット: ターゲットの機種 (`em1d512_te`) 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¥kappl`」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 **完了** をクリックするとプロジェクト「`ledint`」が自動生成されます。

(3) ソースのインポート

Eclipse の「C/C++ プロジェクト」ビュー内に作成されたプロジェクト「`ledint`」を選択します。右クリックしてメニューを開き (またはメニューバーの「ファイル」から)、「インポート」を選択して「インポート」ダイアログを開きます。
「一般」の下にある「アーカイブファイル」を選択して **次へ** をクリックします。

ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、「T-Kernel 2.0 リファレンスキット」の CD 内の実習用プログラム「ledint.zip」を選択して **開く** をクリックします。**完了** をクリックします。
以上の操作により、ソースが ledint/src 以下にインポートされます。

(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「ledint/em1d512.te/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、ledint/em1d512.te/ の下に「ledint」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「ledint/em1d512.te/ledint」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 (T-Monitor)」で右クリックして「新規」を選択します。ledint の転送や実行方法などが自動的に設定されます。
実行 をクリックすると、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。

このプログラムにより、LED5、LED6、LED7 または LED8 が点滅を開始します。SW4 を押すと点滅パターンが変わります。

4.4 物理タイマのサンプル実装

物理タイマを使った T-Kernel ベースのサンプルプログラムです。

物理タイマ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 2.0 リファレンスキットでは物理タイマ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の仕様については、ソース先頭のコメントをご参照ください。

(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¥kappl」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¥kappl」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 のコンソールビューの 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

「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: 「ptimer」と入力します。
- ターゲット: ターゲットの機種 (em1d512_te) 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¥kappl」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 **完了** をクリックするとプロジェクト「ptimer」が自動生成されます。

(3) ソースのインポート

Eclipse の「C/C++ プロジェクト」ビュー内に作成されたプロジェクト「ptimer」を選択します。右クリックしてメニューを開き (またはメニューバーの「ファイル」から)、「インポート」を選択して「インポート」ダイアログを開きます。
「一般」の下にある「アーカイブファイル」を選択して **次へ** をクリックします。
ソースアーカイブファイル欄の右の **参照** をクリックして「アーカイブファイルからインポート」ダイアログを開き、「T-Kernel 2.0 リファレンスキット」の CD 内の実習用プログラム「ptimer.zip」を選択して **開く** をクリックします。**完了** をクリックします。
以上の操作により、ソースが ptimer/src 以下にインポートされます。

(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/em1d512.te/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/em1d512.te/ 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/em1d512.te/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

(T-Monitor)」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が自動的に設定されます。
実行 をクリックすると、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
Physical timer number=1, clock=28672000[Hz], limit=2866
SYSPRG ptimer [14] d04a1000 - d04a7000
[/SYS]% Hello 10000
Hello 20000
Hello 30000
  ⋮
```

100 マイクロ秒周期で物理タイマハンドラを起動し、10000 回 (1 秒) ごとに printf で表示を行います。

索 引

はじめてみようリファレンスキット

Version 1.03.01

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