T-Engine フォーラム

Java WG

T-JV 入門

Version 1.01

2005 年 9 月

はじめに

本書は、T-JV の入門書の位置づけとして、T-Engine フォーラム Java WG においてまとめられたものである。

今後、Ported Extension のひとつとして T-JV が活用され、T-Engine が世の中に深く浸透することを期待するものである。

編纂に当たり、Java WG メンバに多大なご協力を頂いたことに感謝したい。

T-Engine フォーラム

改訂履歴

版数	改版日	内容	備考
0.90	2005/3/22	新規作成	
1.00	2005/7/21	誤字脱字の修正等	
1.01	2005/9/13	細部修正･公開	

注意事項

・個人の私的使用に限り、講習テキストを自由に利用することができます。
・原則として、講習テキストの再配布を行うことはできません。
・講習テキストの著作権は T-Engine フォーラムに帰属します。
・講習テキストに記されている内容は、将来予告無く変更されることがあります。
・T-Engine フォーラムおよび講習テキスト執筆協力者は、講習テキストに記載されている内容について、明示的にも黙示的にも、また法律で定められているか否か問わず、一切の保証をいたしません。
・T-Engine フォーラムおよび講習テキスト執筆協力者は、講習テキストに関連して発生したいかなる損害もしくは不利益についても一切責任を負いません。

・TRON は、"The Real-time Operating system Nucleus"の略称です。
・ITRON は、"Industrial TRON"の略称です。
・μITRON は、"Micro Industrial TRON"の略称です。
・TRON、BTRON、ITRON、μITRON、eTRON、T-Engine、μT-Engine、T-Monitor、T-Kernel、T-JV、T-Dist はコンピュータの仕様に対する名称であり、特定の商品を指すものではありません。
・Sun、SunMicrosystems、Java および Java に関連する商標は、SunMicrosytems Inc.の米国およびその他の国における商標または登録商標です。
・JBlend 並びに Aplix または JBlend に関連する商標は、株式会社アプリックスの日本およびその他の国における商標または登録商標です。
・KASAGO は、株式会社エルミックシステムの商標です。
・Linux は、Linus Torvalds の米国およびその他の国における商標または登録商標です。
・UNIX は、TheOpenGroup の米国およびその他の国における商標または登録商標です。
・Red Hat は、Red Hat,Inc.の米国およびその他の国における商標または登録商標です。
・Microsoft、Windows は、Microsoft Corporation の米国およびその他の国における商標または登録商標です。
・その他の本書に記載されている会社名、商品名などは、各社の商標または登録商標です。
本書では、『™』、『®』マークを明記しておりません。

本書の構成

組込み Java 入門

Copyright (C) T-Engine Forum 2005 All rights reserved.

本章の概要および前提知識

- 本章の概要
  - Javaの概要
- 本章を読むために必要な前提知識
  - 特になし

Copyright (C) T-Engine Forum 2005 All rights reserved.

はじめに

- 組込みソフトウェア開発の問題点
  - ターゲット毎の開発が必要
  - プログラムの複雑化、開発規模の増大
  - ネットワーク化とセキュリティ
  - 限られたリソース (CPU性能、メモリ容量)
  - ユーザーによるアプリケーション開発の困難さ

  ⇒ オブジェクト指向、オープンな仕様の必要性

Copyright (C) T-Engine Forum 2005 All rights reserved.

Javaとは

- Javaは、米 Sun Microsystems Inc.社が1995年に開発した
  - オブジェクト指向プログラミング言語、および、
  - その実行環境

Copyright (C) T-Engine Forum 2005 All rights reserved.

Java の特徴

- プログラミング言語としての Java
  - オブジェクト指向
    ⇒ ソフトウェアの部品化、再利用性の向上
  - ガーベッジコレクション
    ⇒ メモリ破壊防止･不正アクセスの禁止、開発の容易化
  - コンパイルにより Java バイトコード生成、バイトコードは Java バーチャルマシンが実行
    ⇒ ターゲット非依存、”Write Once, Run Any Where”

- プラットフォームとしての Java
  - Java バーチャルマシン (Java Virtual Machine)
  - 仕様化されたAPIセット
  - 優れたネットワーク機能とセキュリティ

Copyright (C) T-Engine Forum 2005 All rights reserved.

Java プラットフォーム

- Java 2
  - バージョン1.2以降、Java 2
- エディション
  - Java 2 は目的別にエディション (Edition) に分類
    - J2EE (Java 2 Platform, Enterprise Edition)
      - サーバーサイド、エンタープライズ向け
    - J2SE (Java 2 Platform, Standard Edition)
      - デスクトップアプリケーション、開発向け
    - J2ME (Java 2 Platform, Micro Edition)
      - 組込み分野向け
- コンフィギュレーション と プロファイル
  - 組込み機器：特殊なデバイス, 限られたリソース…etc.

  ⇒ J2ME ではターゲットを想定し、コンフィギュレーション (Configuration) と プロファイル (Profile) を規定

Copyright (C) T-Engine Forum 2005 All rights reserved.

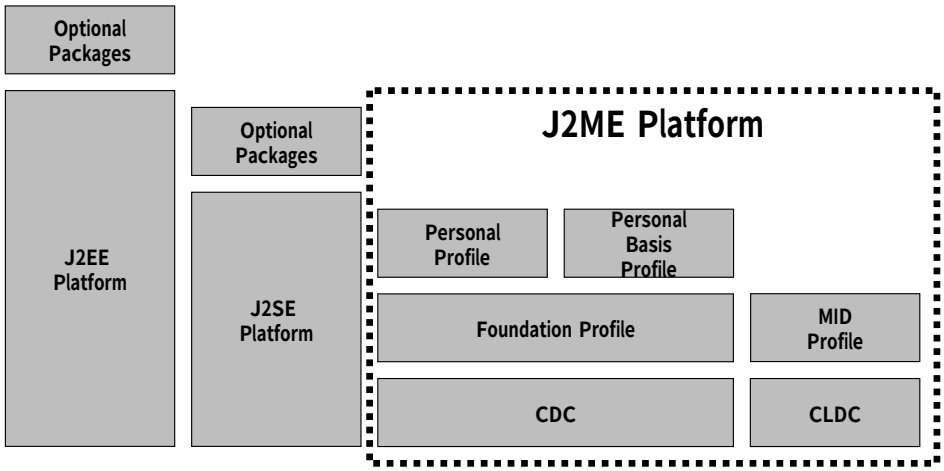
コンフィギュレーションとプロファイル

- コンフィギュレーション
  - 実行環境の基礎要件の定義
    - CDC (Connected Device Configuration)
      - PDA、カーナビ、セットトップボックスなど
      - 32bit CPU, 数MB以上のメモリ
      - バーチャルマシンは CVM
    - CLDC (Connected Limited Device Configuration)
      - 携帯電話、家電、小規模PDAなど
      - 16/32bit CPU, 数百KB以上のメモリ
      - バーチャルマシンは KVM （サブセット）
- プロファイル
  - 実行環境における、使用目的に対するAPIセットの定義
    - CDC 向けプロファイル：
      - Foundation Profile
      - Personal Basis Profile
      - Personal Profile
    - CLDC 向けプロファイル：
      - MIDP (Mobile Information Device Profile)
      - DoCoMo Profile (DoJa), KDDI Profile, …etc.

Copyright (C) T-Engine Forum 2005 All rights reserved.

Java アーキテクチャ概要

■ Java アーキテクチャ概要



Copyright (C) T-Engine Forum 2005 All rights reserved.

組込み Java の現状

- 広がる組込み Java の世界
  - 携帯電話、PDA、デジタルテレビ、カーナビゲーションなど、既に多くの製品に組込まれている
  - オプションAPIの充実・最新技術への対応
- 実行環境の性能・応答性向上
  - ハードウェアよるバイトコード実行技術
  - JIT、Hot Spot 等によるソフトウェア手法
  - リアルタイム Java

Copyright (C) T-Engine Forum 2005 All rights reserved.

・オプションパッケージ (API) は、業界の動向に合わせて最新のものが次々に仕様化されています。

・従来 Java の最大の弱みであった ”実行性能” も現在では克服されており、組込み Java においてもネイティブ環境と同等の実行性能をもっています。

組込み Java を利用するには

- Java プラットフォームの選定
  - JCP (Java Community Process) により策定、JSR (Java Specification Request) として仕様を公開
    - CDC (JSR 36 CDC, JSR 218 CDC1.1)
    - Foundation Profile (JSR 46 FP, JSR 219 FP1.1)
    - Personal Basis Profile (JSR 129 PBP, JSR 217 PBP1.1)
    - Personal Profile (JSR 62 PP, JSR 216 PP1.1)
    - CLDC (JSR 30 CLDC, JSR 139 CLDC1.1)
    - MIDP (JSR 37 MIDP, JSR 118 MIDP2.0)
    - …and so on.
- Java プラットフォームの実装
  - Java 言語で開発したソフトウェアはターゲット非依存であるが、Java プラットフォームはターゲットに依存

  ⇒ Java プラットフォームをターゲットに実装することで、Java の資産およびその利便性を利用できる

Copyright (C) T-Engine Forum 2005 All rights reserved.

・JCPのURL
- http://www.jcp.org/

本章のまとめ

- Javaは、米Sun Microsystems Inc.社が1995年に開発したオブジェクト指向プログラミング言語、および、その実行環境である
- Javaプラットフォームは3つのエディションからなり、組み込み分野向けにJ2MEが規定されている
- Javaプラットフォーム上のソフトウェアはターゲット非依存であり、ターゲットが変わってもJavaの資産の活用が可能である

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV概要

Copyright (C) T-Engine Forum 2005 All rights reserved.

本章の概要および前提知識

- 本章の概要
  - T-JVの概要説明
- 本章を読むために必要な前提知識
  - Javaに関する一般的な知識
  - T-Engineに関する一般的な知識

Copyright (C) T-Engine Forum 2005 All rights reserved.

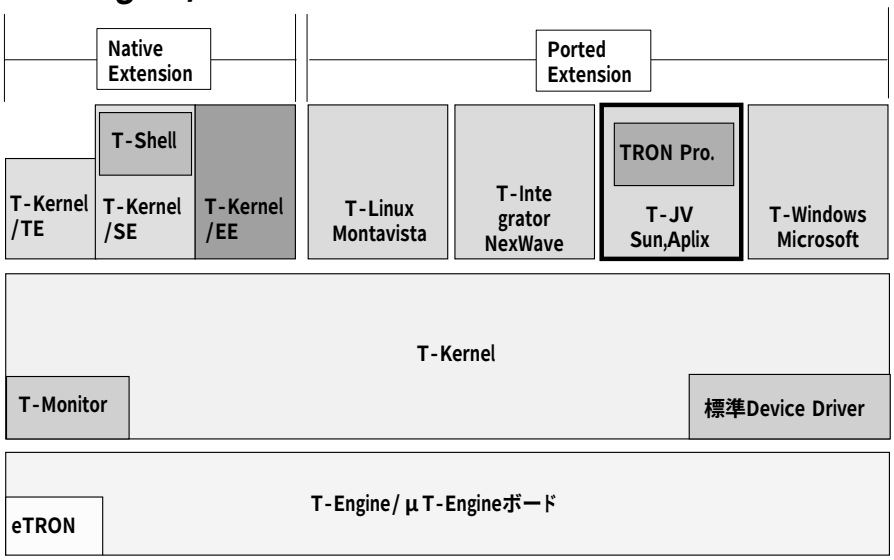
T-JV仕様策定の背景

- 2002年6月
  - T-Enigneフォーラムが設立
- 2002年8月
  - T-Engine Forum A会員の中で、特にJavaに関する仕様策定を行なうJava WGの第一回が開催される
- 2004年5月
  - JavaWGにおいて策定がすすめられているT-JV仕様に基づいたT-Engine上のJava実行環境であるT-JVリファレンス実装ソースコードをA会員向けにリリース
- 2004年8月
  - T-Enigneフォーラムより、T-JVリファレンス実装ソースコードをB会員及び学術会員向けにリリース

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVの位置づけ

- T-Engine/PEのひとつ



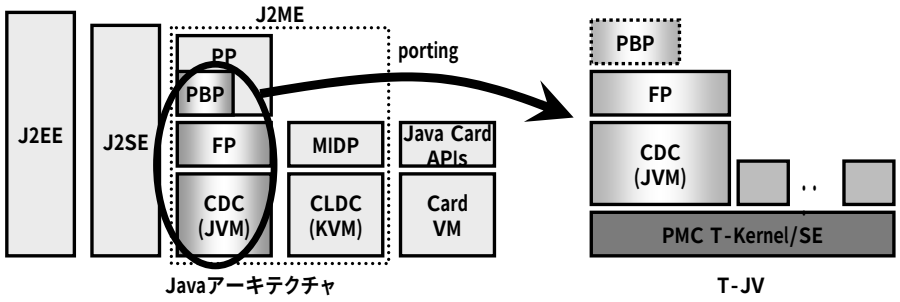
Copyright (C) T-Engine Forum 2005 All rights reserved.

T-Engine/PE(T-Engine/Ported Extension)は、既存のミドルウェアやカーネルをT-Kernel上に移植したものです。

T-JVのアーキテクチャ仕様

- T-JVのVMとして、CDCを採用
  - T-Engine Forum Java WG にて T-JV 仕様を策定
    - ◆T-Engineは、多くのインタフェースを持つ
    - ◆T-Engineは、 CPU能力も組み込み機器の中では比較的高い
    - ◆T-Engineは、 LCDを持ち合わせている

      → 比較的リッチなコンフィギュレーションを選択することが可能



Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVアーキテクチャ仕様の特長

- Javaは、プラットフォーム非依存（VMがハードウェアやOSの違いを吸収する）ため、ソフトウェアの移植性が高い
- オブジェクト指向言語であるため、モジュールの再利用性が高まる
- ネットワーク周りのAPIがサポートされている
- T-JVはCDC(JVM)をベースとしているため、サポートするAPIの範囲であればJava 2 Standard Development Kitを使っての開発が可能
- ハードウェアとソフトウェア（Javaアプリケーション）の並行開発が可能となり開発期間の短縮が可能

Copyright (C) T-Engine Forum 2005 All rights reserved.

基本的に、Javaの特徴をそのまま受け継いでいます。

T-JVのリファレンス実装

- T-Engine Forumより2つのT-JVリファレンス実装が公開
  - T-JV (アプリックス + UNL)
    - 株式会社アプリックスとYRPユビキタスネットワーキング研究所により共同構築したT-JVリファレンス実装
      - Connected Device Configuration 1.0
      - Foundation Profile 1.0
      - Personal Basis Profile 1.0
  - T-JV (サンマイクロシステムズ + UNL)
    - サン・マイクロシステムズ株式会社とYRPユビキタスネットワーキング研究所により共同構築したT-JVリファレンス実装
      - Connected Device Configuration 1.0
      - Foundation Profile 1.0
      - TRON Code Profile

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 18

T-Engineフォーラムより公開されているT-JVのリファレンス実装ソースコードは2種類あります。
双方のリファレンス実装とも、T-Engine/SH7727開発キット向けのものとなっております。

T-JVリファレンス実装入手条件

- J2ME CDC/FP及びPBPのSCSLをSun社と締結していること
  - Sun社のWebサイト上で締結可能
- T-JVリファレンス実装の利用条件に同意できること
  - T-Engine Forum A会員、B会員、学術会員が対象

※Sun Community Source License (SCSL)

・Sun Microsystems Inc.社がプログラミング言語(およびその開発・実行環境)「Java 2」に適用したライセンス体系
・ソースコードの使用と修正は基本的に無償
・自社製品にJava 2(またはその派生物)を組み込んで販売し、利益をあげるような場合や、社内用に利用する場合には、利益に応じたライセンス料を支払う必要がある

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVのリファレンス実装を入手するためには、2つの手続きが必要になります。

また、研究や評価目的への利用は可能ですが、社内システムや商用システムには利用できませんので、注意が必要です。商用利用や社内システムとして利用する場合は、別途契約をする必要があります。

リファレンス実装の配付内容

- 配付内容
  - T-JV (アプリックス＋UNL)
    - マニュアル
    - VMライブラリのビルド環境
    - 実行イメージのビルド環境
  - T-JV (サンマイクロシステムズ＋UNL)
    - マニュアル
    - T-JV本体 (CDC/FP + TRON Code Profile) のソースコード
    - コンパイル済みのJava実行環境
    - マイクロスクリプトを用いたデモプログラム
    - T-JV専用ビルド環境
    - 専用 unix エミュレータ

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 20

リファレンス実装には、マニュアルやビルド環境等が含まれております。

リファレンス実装の特徴

- GUI
  - アプリックス＋UNL版
    - PBPを実装済み
  - サンマイクロシステムズ＋UNL版
    - T-Shellマイクロスクリプトとの連携サンプルアプリケーションを添付
- ネットワーク
  - アプリックス＋UNL版
    - 株式会社エルミックシステム製のKASAGO for T-Engine のTCP/IPドライバを利用したTCP/IP通信が可能
  - サンマイクロシステムズ＋UNL版
    - パーソナルメディア株式会社製のT-Shellに付属するTCP/IPドライバを利用してTCP/IP通信が可能

Copyright (C) T-Engine Forum 2005 All rights reserved.

ソースが公開されており、自由にカスタマイズすることも可能となっております。

T-JV展望(1)

- プロファイルの充実
  - 今後、T-Engine Forum Java WG にて下記プロファイルを策定していく予定
    - UC（ユビキタスコミュニケータ） Profile
      - YRPユビキタス・ネットワーキング研究所にて開発されたUCの入出力インタフェースや、デバイスに対応したUC向けプロファイル
    - eTRON Profile
      - YRPユビキタス・ネットワーキング研究所にて開発されたeTRONデバイスに対するプロファイル
    - ucode Profile
      - ucode解決サーバとの通信プロファイル
      - アプリケーションサービスサーバとの通信プロファイル
    - 家電制御プロファイル
      - 家電制御のためのプロファイル

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

現在、T-Engine ForumのJavaワーキンググループでは、この仕様策定作業を実施しており、各プロファイルの拡張の検討を実施しております。

T-JV展望(2)

- リファレンス実装の移植
  - T-Engine Forum Java WG では、T-JVリファレンス実装をUC（ユビキタスコミュニケータ）にも対応させていく予定
  - なお、T-JVリファレンス実装の利用ライセンスの範囲内であれば、T-Engine Forumより入手したT-JVリファレンス実装のソースを利用して、他ターゲットへ自由にportingすることも可能

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVリファレンス実装のライセンス範囲内であれば、ソースを改造して、他ターゲットへportingすることも可能となっています。

本章のまとめ

- T-JVは、T-Engine上のJava実行環境仕様であり、Ported Extensionのひとつである
- T-JVは、CDCを採用している
- T-JVには、2種類のリファレンス実装が公開されており、それぞれライセンス契約を行うことにより利用可能となる

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVリファレンス実装の紹介

Copyright (C) T-Engine Forum 2005 All rights reserved.

本章の概要および前提知識

- 本章の概要
  - T-JVリファレンス実装を利用したJavaアプリケーションの実行
    - T-JV(アプリックス+UNL)版
    - T-JV(サン・マイクロシステムズ+UNL)版
- 前提知識
  - Javaに関する一般的な知識
  - T-Engineに関する一般的な知識
  - T-JVに関する一般的な知識

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVリファレンス実装のライセンス範囲内であれば、ソースを改造して、他ターゲットへportingすることも可能となっています。

T-JVリファレンス実装の紹介
～T-JV（アプリックス + UNL）版～

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV(アプリックス+UNL)の概要

- J2MEのCDCに基づく Java 実行環境を提供
  - J2SE ベースのクラスライブラリ
  - FP (Foundation Profile)、PBP (Personal Basis Profile) をサポート
  - CLDC1.0 と互換性のあるライブラリ
- Java 実行環境のビルド環境を提供

Copyright (C) T-Engine Forum 2005 All rights reserved.

前提条件(1/2)

- T-JV(アプリックス+UNL) の実行に必要なもの
  - T-Engine/SH7727 開発キット T-Engine ボード
  - 上記ハードウェアにシリアル接続したターミナル (PCなど)
  - T-Engine ボード用起動ディスク
    - ◆コンパクトフラッシュやUSB メモリ
  - T-Engine Board LAN 拡張ボード
    - ◆ネットワーク使用時のみ
  - エルミックシステム Kasago for T-Engine
    - ◆ネットワーク使用時のみ

Copyright (C) T-Engine Forum 2005 All rights reserved.

前提条件(2/2)

- T-JV（アプリックス+UNL）のビルドに必要なもの
  - T-Engine/SH7727 開発キットGNU 開発環境
  - Java 2 SDK, Standard Edition, v1.3.1
    - http://java.sun.com/j2se/1.3/
  - エルミックシステム Kasago for T-Engine
  - 上記開発環境がインストールされた Linux マシン

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV(アプリックス+UNL)のビルドと実行

ビルド環境は2つに分かれ、互いに独立している

VM ライブラリのビルド環境	jcnlib_te_sh7727.tar.gz	ライブラリの生成のみを行う
実行イメージのビルド環境	jbstart_te_sh7727.tar.gz	ターゲットデバイス依存部分を定義しており、これとVMライブラリをリンクして実行イメージを生成する

これらの 2 つのファイルを、開発環境がインストールされている Linux マシンに展開してビルド作業を行う

```
$ tar zxvf jcnlib_te_sh7727.tar.gz
$ tar zxvf jbstart_te_sh7727.tar.gz
```

2つのディレクトリ、jcnlib_te_sh7727、jbstart_te_sh7727 が作成される。ネイティブメソッドの組み込みなど、カスタマイズの多くは実行イメージのビルド環境のみの変更で行える

Copyright (C) T-Engine Forum 2005 All rights reserved.

ライブラリのビルド(1/2)

1) jcnlib_te_sh7727/cvm.sh の以下の部分を，Java 2 SDK, Standard Edition, v1.3.1 をインストールしているディレクトリに修正
(例：/usr/local/jdk1.3.1_11 にインストールしている場合)

ファイル名：jcnlib_te_sh7727/cvm.sh

```
export JDK_HOME=/tools/java/jdk1.3.1
```

↓ 変更後

```
export JDK_HOME=/usr/local/jdk1.3.1_11
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

ライブラリのビルド(2/2)

2) ディレクトリ jcnlib_te_sh7727/ でシェルスクリプト cvm.sh を実行する

```
$ cd jcnlib_te_sh7727/
$ sh cvm.sh
```

ディレクトリ jcnlib_te_sh7727/build/TE_SH7727/bin にVM ライブラリ libjcn.a が生成される。

3) 生成された VM ライブラリ libjcn.a を，ディレクトリ jbstart_te_sh7727/jcn/lib にコピー

```
$ cp jcnlib_te_sh7727/build/TE_SH7727/bin/libjcn.a jbstart_te_sh7727/jcn/lib
```

この後，実行イメージのビルドを行うことによって，ビルドしたVM ライブラリが実行イメージに反映される。

Copyright (C) T-Engine Forum 2005 All rights reserved.

Java実行環境の設定(1/2)

jbstart_te_sh7727/h/jbstart.h 中の以下の定義を任意の値に変更し，実行イメージをビルドすることで設定が反映される。

設定値	初期値	説明
DEFAULT_CLASSPATH	“/SYS/classes”	Java クラスパスを指定。ディレクトリの他にjar ファイルやzip ファイルも指定できる。複数のパスを指定する場合はコロン (”:”) で区切って並べる。デフォルト値のままの値を使用する場合，/SYS/classes ディレクトリを作成し，起動するクラスファイルをその中に配置。
DEFAULT_VMARGS	“”	VM の起動オプションを指定。
DEFAULT_APPNAME	“aplix.pbp.TestLauncher”	起動するアプリケーションのメインクラス名を指定。
DEFAULT_APPARGS	“”	アプリケーションの引数を指定。
DEFAULT_HOMEPATH	“/SYS/home”	ホームディレクトリを指定。ここで指定されたディレクトリが，Java プログラム中での相対パス指定時のカレントディレクトリとなる。また，この文字列は java.lang.System.getProperty() でkey 値を“user.home”, “user.dir” とした場合に取得できるディレクトリ名となる。

Copyright (C) T-Engine Forum 2005 All rights reserved.

Java実行環境の設定(2/2)

jbstart_te_sh7727/h/jbstart.h 中の以下のネットワーク設定の定義は、Kasago サブシステムを T-Engine にインストールした場合に有効になる

設定値	初期値	説明
NETWORK_IPADDR	“192.168.10.10”	T-Engine ボードのIP アドレスを設定
NETWORK_NETMASK	“255.255.255.0”	ネットマスクを設定
NETWORK_GATEWAY	“192.168.10.0”	デフォルトゲートウェイのIP アドレスを指定
NETWORK_1STDNS	“192.168.10.1”	プライマリDNS サーバのアドレスを指定
NETWORK_2NDDNS	“192.168.10.2”	セカンダリDNS サーバのアドレスを指定(セカンダリDNS サーバを指定しない場合は、NETWORK_2NDDNS を未定義にする)
NETWORK_HOSTNAME	“host.domain.com”	T-Engine ボードのホスト名を設定

Copyright (C) T-Engine Forum 2005 All rights reserved.

実行イメージのビルド

1) ディレクトリを移動

```
$ cd jbstart_te_sh7727/sh7727/
```

2) VM ライブラリを作り変えた場合などは、make clean を実行してビルド済の実行イメージを削除

```
$ make clean
```

3) make を実行して、Java 実行環境の実行イメージをビルド

```
$ make
```

ディレクトリ jbstart_te_sh7727/sh7727/ に、実行イメージ jbstartが生成される

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV（アプリックス＋UNL）の実行(1/2)

作成した実行イメージ jbstart をT-Engine ボードに転送

```
$ cd jbstart_te_sh7727/sh7727/
$ /usr/local/te/etc/gterm -B -l/dev/ttyS0
```

```
[/SYS] recv jbstart
```

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

ここでは説明を簡潔にするため、recv コマンドを例にしていますが、実行イメージ (jbstart) のサイズは5 MB ほどあり、recv コマンドでは転送に時間がかかります。

そのため、実際に転送する場合は、別の手段を取るほうが良いでしょう。

（一度FATパーティションの CF や USB にコピーして、mscnv コマンドでT-Engine 上に読み取るなど...）

T-JV (アプリックス＋UNL) の実行(2/2)

3) T-JV (アプリックス+UNL) の起動

- 転送した実行イメージ jbstart を実行し、T-JV (アプリックス+UNL)を起動

```
[/SYS] lodspg jbstart
```

この実行は，アプリケーションのクラスが存在しないために
　java.lang.NoClassDefFoundError:aplix.pbp.TestLauncher
と表示され終了する。
なお，一番最後に出るメッセージ
　jbstart:Can't Loaded System Program -327680
は、jbstart をシステムに常駐させないようにしているためで，
jbstart の実行が成功してもこのメッセージは表示される。

Copyright (C) T-Engine Forum 2005 All rights reserved.

サンプルプログラム

- 2 種類のサンプルプログラムの実行までの手順
  - Hello World!
  - デバッグボードのLED の操作
    - ネイティブメソッドを使用

Copyright (C) T-Engine Forum 2005 All rights reserved.

Hello World (1/5)

“Hello World!” と文字列を出力するプログラム

以下の Java ソースファイル "HelloWorld.java" を作成

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println( "Hello World!" );
    }
}
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

Hello World (2/5)

■ コンパイル方法

ソースファイルをコンパイルする

```
$ javac HelloWorld.java
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

Hello World (3/5)

jbstart.h 内で定義している起動クラスを HelloWorld に変更

ファイル：jbstart_te_sh7727/h/jbstart.h

```
#define DEFAULT_CLASSPATH "/SYS/classes"
#define DEFAULT_VMARGS ""
#define DEFAULT_APPNAME "aplix.pbp.TestLauncher"
#define DEFAULT_APPARGS ""
#define DEFAULT_HOMEPATH "/SYS/home"
```

↓変更後

```
#define DEFAULT_CLASSPATH "/SYS/classes"
#define DEFAULT_VMARGS ""
#define DEFAULT_APPNAME "HelloWorld"
#define DEFAULT_APPARGS ""
#define DEFAULT_HOMEPATH "/SYS/home"
```

ディレクトリ jbstart_te_sh7727/sh7727/ で make を行い実行イメージを再構築

```
$ make
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

Hello World (4/5)

■ 実行方法

作成したHelloWorld.class をT-Engine に転送する。Java 実行環境の設定どおり，/SYS/classesの下に置く

```
[/SYS] cd classes
[/SYS/classes] recv HelloWorld.class
```

再構築したjbstart を転送

```
[/SYS] recv -d jbstart
```

(-d オプションをつけると，既にjbstart があったら上書きする)
jbstart を起動する

```
[/SYS] lodspg jbstart
```

実行結果は次ページのとおり。下から3 行目に「Hello World!」と表示されている

Copyright (C) T-Engine Forum 2005 All rights reserved.

Hello World (5/5)


```
[/SYS]% lodspg jbstart
JBlend(TM) - JCN (Java Common Nucleus) by Aplix Corporation
JCN[notice] _getconf("tracelog") returns null.
JCN[notice] log level init by default.
JCN[info] build date :2003-12-17_12:40
JCN[info] log level :0x0003ff0f (verbose 0x0003ff00)
JCN[info]thread: Java thread:
JCN[info]thread: threads = 48
JCN[info]thread: min task id = 0
JCN[info]thread: stack size = 0x0000c000
JCN[info]thread: Priority:
JCN[info]thread: vmhigh1, vmhigh2 = 105,106
JCN[info]thread: java [max,min] = 107,108,109,110,111,112,113,114,115,116
JCN[info]thread: Other resources:
JCN[info]memory: native heap = <managed by external functions>
JCN[info]memory: Java heap = 0x40678008 - 0x40d78007, 0x00700000(bytes)
JCN[notice] _getconf("io") returns null.
JCN[notice]fs: getconf("io") files=0, use default setting(16).
JCN[notice] _getconf("op.consio") returns null.
JCN[notice]fs: Standard file I/O is used as console I/O.
JCN[info]fs: (/ROM) mount success.
JCN[notice] _getconf("fs") returns null.
JCN[notice]net: hostname=host.domain.com
JCN[info] argc=1
JCN[info] argv: HelloWorld
Hello World!
JCN[info] javavm_start: returns 0
jbstart: Can't Loaded System Program -327680
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(1/9)

- デバッグボードの8 bit LED を操作するプログラム
- ネイティブメソッドを使用
  - C 言語 (など) で記述した Java クラスのメソッド
  - Java からは制御できないハードウェアの機能が利用可能

Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(2/9)

T-Kernel は動的なライブラリのロードに対応していないため，ネイティブメソッドを利用する場合は実行イメージ内にネイティブメソッドを定義

以下の Java ソースファイル "LedTest.java" を作成

```
public class LedTest {
    static {
        System.loadLibrary("LedTest");
    }
    public static void main(String[] args) {
        while (true) {
            for (int i = 1; i < 0x100; i = i << 1) {
                controlLED((byte) i);
                try {
                    Thread.sleep(500); /* wait 500ms */
                } catch (InterruptedException e) {}
            }
        }
    }
    private static native void controlLED(byte i);
}
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(3/9)

LedTest クラスに定義しているcontrolLED(byte i)がネイティブメソッド。ネイティブメソッドcontroldLED() は渡されたバイト値を，デバッグボードの8 bit LED ポートにそのまま出力することとする。

Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(4/9)

■ コンパイル方法

クラスファイルとネイティブメソッドのヘッダファイルを作成する。

```
$ javac LedTest.java
$ javah LedTest
```

LedTest.class とLedTest.h が生成される。LedTest.class はT-Engine に転送し，LedTest.h は Java 実行環境のビルド環境のヘッダファイルのディレクトリ(jbstart_te_sh7727/h/) にコピーする。Java 2 SDK,Standard Edition, v1.3.1 に付属しているjni.h もコピーする。

```
$ cp LedTest.h $WORK/jbstart_te_sh7727/h/
$ cp /usr/local/jdk1.3.1_11/include/jni.h $WORK/jbstart_te_sh7727/h/
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(5/9)

ネイティブメソッドLedTest.controlLED() に対応するC 言語の関数を以下のように記述する。

ファイル名：LedTest.c

```
#include "LedTest.h"
#define DEBUGBOARD_8BIT_LED_ADDR 0xA1600000
JNIEXPORT void JNICALL Java_LedTest_controlLED(JNIEnv *env, jclass cs, jbyte i)
{
*((unsigned char*) DEBUGBOARD_8BIT_LED_ADDR) = i;
}
```

作成したLedTest.c はjbstart_te_sh7727/src/ に置く。

Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(6/9)

ネイティブメソッドをJava 実行環境に登録する

- ネイティブメソッドの登録はJava のネイティブメソッド LedTest.controlLED() と，C 言語の関数 Java_LedTest_controlLED() を関連付けることで行う。
- ファイル resource_conf_jni.c に，Java_LedTest_controlLED() を宣言しているヘッダファイル LedTest.h のインクルードと，usrJniSymtab[] への Java_LedTest_controlLED() の情報の追加を行う。

  ファイル名：jbstart_te_sh7727/src/resource_conf_jni.c

```
#include "LedTest.h"
static const struct jcn_jnisymtab usrJniSymtab[] = {
{"Java_LedTest_controlLED", Java_LedTest_controlLED},
{NULL, NULL}
};
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(7/9)

jbstart.h 内で定義している起動クラスを LedTest に変更

ファイル名：jbstart_te_sh7727/h/jbstart.h

```
#define DEFAULT_CLASSPATH "/SYS/classes"
#define DEFAULT_VMARGS ""
#define DEFAULT_APPNAME "aplix.pbp.TestLauncher"
#define DEFAULT_APPARGS ""
#define DEFAULT_HOMEPATH "/SYS/home"
```

↓変更後

```
#define DEFAULT_CLASSPATH "/SYS/classes"
#define DEFAULT_VMARGS ""
#define DEFAULT_APPNAME "LedTest"
#define DEFAULT_APPARGS ""
#define DEFAULT_HOMEPATH "/SYS/home"
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(8/9)

Makefile のSRC にLedTest.c を追加

jbstart_te_sh7727/sh7727/GNUmakefile

```
SRC     = jbstart.c \
          resource_conf_font.c \
          resource_conf_jni.c \
          resource_op_io.c \
          resource_op_key.c \
          resource_op_mouse.c \
          resource_op_net.c \
          resource_op_netdb.c \
          resource_op_time.c \
          resource_op_video_misc.c \
          resource_conf_video.c \
          device_kbpd.c \
          device_screen.c \
          device_net.c \
          builtinFont6x12.c \
          builtinFont12x12.c \
          builtinFont8x16.c \
          builtinFont16x16.c \
          LedTest.c
```

最後にjbstart_te_sh7727/sh7727/ でmake を実行してjbstart を再構築する。

Copyright (C) T-Engine Forum 2005 All rights reserved.

LED の操作(9/9)

■ 実行方法

コンパイル作業でできたファイルのうち，次の2つをT-Engineに転送する

- LedTest.class
- jbstart

```
[/SYS]% cd classes
[/SYS/classes]% recv LedTest.class
[/SYS/classes]% cd ..
[/SYS]% recv -d jbstart
```

jbstart を起動。

```
[/SYS] lodspg jbstart
```

起動後，デバッグボードのLED が点滅する。

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVリファレンス実装の紹介
～T-JV（サンマイクロシステムズ + UNL）版～

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV(Sun+UNL)とは

- UNLとSun MicrosystemsによるT-Engine向けのJava実行環境のリファレンス実装
- Sun RI
  - 様々なプラットフォームへの移植性を考慮にいれたSunのJ2ME CDC + Foundation Profileのリファレンス実装
  - J2ME CDC(Connected Device Configuration) 1.0.1 + Foundation Profile

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 55

JCP(Java Community Process)は、Javaに関する仕様を作成するためのプロセスを規定しています。

JCPでは、仕様を定義するとともに、それが実現可能か否かを実証するために、リファレンス実装を求めています。

T-JVのベースとなる仕様は、JCPのJSR-36(The J2ME Connected Device Configuration)で定義されているJava仮想マシン仕様上に、JSR-46(Foundation Profile)に定義されるクラスライブラリ(API)です。

Sun では、このFoundation Profile についてのリファレンス実装を公開しており、その実装をベースにT-Engine(SH-7727)上に移植したものが、T-JV(UNL+Sun)です。

Sunのリファレンス実装は、組み込み向けのJava仮想マシンのリファレンス実装ということで、さまざまな組む込み向けのハードウェアに移植しやすいように設計されており、基本的にはハードウェア側に標準Cライブラリやグラフィックライブラリが存在しておれば、移植が容易となっています。

ただ、T-Engine上には、標準Cライブラリが存在せず、T-JVをT-Engineに移植するに際しては、Cライブラリの一部をT-Engine上に実装することから行っています。

T-JV実行環境

- Java
  - J2ME CDC + FP
- T-Engine
  - SH7727
  - T-Shell
- ライブラリ
  - UNIX Emulator

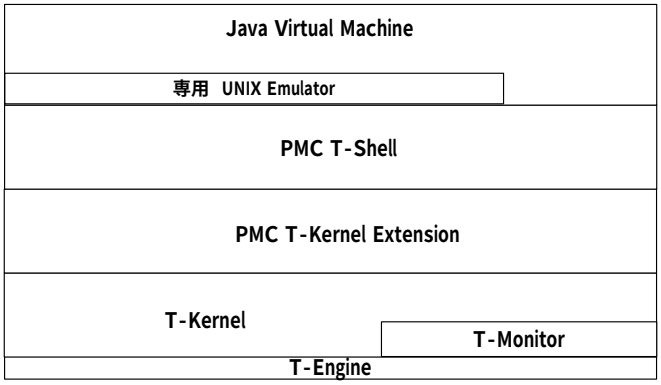
Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVに必要なハードウェア、ソフトウェアは次のとおりです。

ターゲット側：
T-Engine/SH7727開発キット + 拡張LANボード
PMC T-Shell開発キット
Cライブラリ（UNIXエミュレータ）
ホスト側：
Linuxが稼動できるＰＣ
PSは、RedHat Linux 7.3

UNIXエミュレータは、パーソナルメディアより販売されている超漢字に実装中のUNIXエミュレータの一部を流用しています。
UNIXエミュレータは、Ｔ－ＪＶと共に配布されるものを利用します。
PMCから提供されているUNIXエミュレータとの互換性はないため、混同しないように注意する必要があります。

T-JV(Sun+UNL)アーキテクチャ



Copyright (C) T-Engine Forum 2005 All rights reserved.

基本的には、JVMは、パーソナルメディアから販売される「PMC T-Shell」、および「PMC T-Kernel Extension」上で稼動します。また、標準Cライブラリを使用するために、一部UNIXエミュレータも利用しています。
従って、ハードウェア(T-Engine) 上でＪＶＭを実行させるためには、T-Kernel Extension、T-Shell が必要となります。
この構成からわかるように、基本的には、T-Shell、T-Kernel Extension までは、T-Engineの各プラットフォームに対応しているため、UNIXエミュレータの機能を各プラットフォームに移植すれば、SH-7727以外のプラットフォームにも対応は可能となります。

制限事項

- ダイナミックリンク機能
- 絶対パス指定
- JNIのアーギュメント数の制限
- スレッド(suspend/resume)

Copyright (C) T-Engine Forum 2005 All rights reserved.

ダイナミックリンクをサポートしていないため、クラスファイルのロードは、JVM起動時にすべてロードさせておく必要があります。

ロード方法は、

JVM起動時に –Xbootclasspath/a オプションを指定します。

また、相対パスが使用できないため、アプリケーション起動も含めて、すべて絶対パス指定でファイル等と指定する必要があります。

たとえば、

cvm –Djava.classpath=/SYS/WORK/java/test.jar Hello

というように、パスを使用するときには絶対パスで指定します。

JNIを利用する場合に、JNIで使用できるアーギュメント数が2つに制限されています。

JNIが問題となるのは、一般のJavaアプリではなく、JVM自身のネイティブメソッドに組み込む場合に限られます。

(JNIはCVM上で実行されるJavaアプリケーションからはサポートされていないため)

スレッドのsuspend/resumeはサポートされていません。

T-JV の構成

- t-jv_sun.unl_readme_v010000.pdf
  - マニュアル
- cdcfoundation.tar.gz
  - CDC/FP + TRONCodeプロファイルのソースコード
- cvm_bin.tar.gz
  - コンパイル済みのJava実行環境
- CVM_DEMO.BPK
  - マイクロスクリプトを用いたデモプログラム
- t-jv_buildenv.tar.gz
  - Java実行環境構築のためのT-JV専用ビルド環境
- **unixemu**
  - T-JV専用unixエミュレータ

Copyright (C) T-Engine Forum 2005 All rights reserved.

開発環境の条件

- 開発ホストマシン
  - シリアルポートを持つPC/AT互換機
  - make, gcc等の標準開発環境を持つLinux
    - RedHat Linux 7.3
  - Java実行環境構築のための専用ビルド環境
    - JDK1.3.1
  - Linux用GNU開発環境
- 開発ターゲットマシン
  - T-Engine SH7727 + 拡張LANボード
  - コンパクトフラッシュカード + PCカードアダプタ
  - T-Shell

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 60

T-JVでは、アプリケーションの開発等は、開発ホストマシン上で行うため、T-Engineとは別に、Linuxの稼動するホストマシンが必要です。

ホストマシンに要求仕様は、

・RedHat7.3

・JDK1.3.1

が稼動することです。

T-JVをビルドさせるためには、gcc, make等のLinux用のGNU開発環境が必要となります。

また、T-Engineと接続させるために、シリアルポート、あるいはネットワーク接続環境が必要となります。

T-JVを稼動させるターゲットマシンは、T-Engine SH-7727に拡張LANボードを搭載したものです。

また、T-Shellのインストールを行うためにコンパクトフラッシュおよびT-Engineに接続させるPCカードアダプタが必要です。

T-JV環境構築の流れ

1. T-JV開発環境のインストール(ホスト側)
2. ホストとターゲットマシンの通信環境の設定
3. T-Kernelのアップデート
4. 起動ディスクの作成
    1. パーティション設定、フォーマット
    2. T-Shellインストール
    3. T-Kernelパラメータの設定
5. Java実行環境(CVM)のインストール(ターゲット)
6. Unixエミュレータのロード
7. Javaアプリケーションの実行

Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV ビルド環境のインストール(1)

- T-JV専用ビルド開発環境(t-jv_buildenv.tar.gz)のインストール
  - 既存のクロス開発環境と置き換える必要がある
    既存の開発環境が上書きされる点に注意すること
    展開先は /usr/local/te
  - T-JV構築用の環境として、他の開発環境と分離することが望ましい
- 環境変数の設定

```
#tar zxvf t-jv_buildenv.tar.gz
```

```
$export BD=/usr/local/te
$export GNUs=/usr
$export GNU_BD=$BD/tool/Linux-i686
$export GNUsh=$GNU_BD/sh-unknown-tkernel
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

ホストPC側のビルド環境のインストール手順です。

T-Engineフォーラムよりダウンロードしたアーカイブを展開したときに含まれるt-jv_buildenv.tar.gz を「/usr/local/te」ディレクトリにて展開します。

この展開されるディレクトリは、通常のT-Engine開発環境と同じであるため、他のT-Engine開発環境がすでに含まれている場合には、上書きされてしまいます。したがって、T-JV用のビルド環境をインストールするホストPCは、他のT-Engine開発マシンとは分離するのが良いです。

PMCより公開されているT-Engine　GNU開発環境とディレクトリ構成は類似していますが、内容についてはT-JV用にカスタマイズされているため、同一ではないため、通常のT-Engineの開発環境に不具合が発生する可能性があるためです。

また、各環境変数は、t-jv_buildenv.tar.gz を展開したディレクトリに対応させる必要があります。

ただし、DBについては、/usr/local/te に展開することを前提条件とします。つまり、上記の値そのままの設定を利用することとなります。

T-JV ビルド環境のインストール(2)

■ 通信設定

- gterm(開発環境に付属するターミナルソフト)を使用

```
$BD/etc/gterm -l/dev/ttyS0 -B
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV ビルド環境のインストール(3)

- T-Kernelのアップデート
  - バージョン1.1.02
  - http://www.personal-media.co.jp/te/support/
  - T-Monitor上でロムイメージの更新を行う
    ※ DIP SW-1 はONにしておく

```
TM> .flload /tmp/romimage-1.1.02.mot
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

次に、T-Kernelのアップデートを行います。

T-JVを実行させるために必要なカーネルのバージョンは、1.1.02 です。

カーネルのアップデートの方法については、パーソナルメディアより配布されているドキュメントを参照してください。

T-JV ビルド環境のインストール(4)

■ 起動ディスクの作成

- コンパクトフラッシュカードのパーティション設定

```
[/SYS]% hdpart pca
```

- ディスクのフォーマット

```
[/SYS]% format -b pca WORK
```

- フラッシュメモリの内容をコピー

```
[/SYS]% att pca A
[/SYS]% cp -b -r -v * /A
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV ビルド環境のインストール(5)

■ T-Shellのインストール

- PMC T-Shell/SH7727開発キット取扱説明書参照のこと
- インストール例(CD-ROMドライブからのインストール)

CD-ROMドライブをマウント

```
[/SYS]% att uda2 uda2
```

CFにT-Shellをインストール

```
[/SYS]% cd /uda2
[/uda2]% install SYS
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JV ビルド環境のインストール(6)

- JDK1.3.1のインストール
  - http://java.sun.com/j2se/1.3/ja/download.html
- 環境編集の設定

```
$export PATH=/usr/java/jdk1.3.1_11/bin:$PATH
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

CVMのインストール

- コンパイル済みCVMの展開(開発マシン上)
  - cvm_bin.tar.gz
    - bin/cvm
    - bin/cvm.map
    - lib/foundation.jar
    - lib/security/java.policy
    - lib/security/java.security

Copyright (C) T-Engine Forum 2005 All rights reserved.

CVMのインストール(2)

■ ターゲットへの転送

```
$/usr/local/te/etc/gterm -l/dev/ttyS0 -B
```

```
[/SYS/WORK]% mkf -t6 java
[/SYS/WORK]% cd java
 ・・・
[/SYS/WORK/java/bin]% recv -d bin/cvm
```

■ インストール先のディレクトリ構成

/SYS/WORK/java/bin/cvm
/lib/foundation.jar
/lib/security/java.policy
/lib/security/java.security
(※) 転送先は固定ディレクトリ

Copyright (C) T-Engine Forum 2005 All rights reserved.

前ページにて、展開されたファイルをT-Engine側に転送します。
T-Engine側のディレクトリ構成は、上記スライドにあるように固定です。
該当するディレクトリがない場合には、mkfコマンドにてディレクトリを作成後、ホストPC側から転送します。

T-JV T-Kernelパラメータの設定

■ /SYS/SYSCONFの編集

```
TmaxSemID                500
TmaxMaxID                500
```

※edコマンド等でSYSCONFファイル編集

Copyright (C) T-Engine Forum 2005 All rights reserved.

JVMを起動させるために、セマフォ数の上限値を変更します。

この設定値は、/SYS/SYSCONF にて指定されているので、エディタでその数を500に変更します。

(デフォルト値は１００となっています)

このファイルの設定値が不正だと、T-Engine自体が起動できなくなるため、設定ファイルの変更は慎重に行う必要があります。

設定ファイルの編集は、gterm 上で ラインエディタ ed を使用するか、一度ファイルをホストPC側にfget (簡易ftpコマンド) で取得し、PC側で変更した後、fputで T-Engine側に戻すなどします。

UNIXエミュレータのインストール

- CVMを実行するためにはUNIXエミュレータのロードが必要
  - CVMの実装は標準Cライブラリを利用しているため
- ${BD}/lib/sh3l/unixemu
  → /SYS/WORK/java/unixemu へコピー
- 起動時にUNIXエミュレータをロードする

```
[SYS/WORK/java]% loadspg /SYS/WORK/java/unixemu
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

次に、UNIXエミュレータをT-Engine側に配置します。
UNIXエミュレータは t-jv_buildenv.tar.gzに含まれています。

- /usr/local/te/lib/sh3l/unixemu

このファイルを

- /SYS/WORK/java

にコピーします。

CVMの起動

- CVMの実行(クラスファイル)

```
[SYS/WORK/java]% cvm -Djava.class.path=/SYS/WORK/java/test HelloWorld
```

- CVMの実行　　(アーカイブからの実行)

```
[SYS/WORK/java]% cvm -Xbootclasspath/a=/SYS/WORK/java/test/testclasses.zip HelloWorld
```

2005.09.13 version 1.01　Copyright (C) T-Engine Forum 2005 All rights reserved.

JVMは、/SYS/WORK/java/cvm コマンドにより実行されます。
上記例にあるように、クラスパスを-Dオプションにて指定します。

アプリケーションの実行方法には、クラスファイルを直接実行させる方法と、zipアーカイブにクラスファイルをまとめたものから起動させる方法があります。
前者の場合、クラスパスを –Djava.class.path にて指定します。
後者の場合、JVM起動時にクラスファイルをロードさせるために、-Xbootclasspath/a にてアーカイブを指定する必要があります。これは、T-JV(UNL+Sun)がダイナミックロードをサポートしていないため、JVM起動時にクラスファイル全てをロードさせる必要があるためです。

T-JV ビルド環境

- Makefileの設定
  - ${CVM_HOME}/cdcfoundation/build/btron3/defs.mk

```
JDK_HOME = /micro/tools/java/jdk1.3
```

- makeの実行
  - ビルドディレクトリ
    - ${CVM_HOME}/cdcfoundation/build/btron3-sh/
  - ビルドパラメータ
    - CVM_DEBUG=true
    - J2ME_CLASSLIB=foundation

```
make CVM_DEBUG=true J2ME_CLASSLIB=foundation
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

本ページでは、T-JVをビルドするための方法について説明します。

まず、ホストPC側で、コンパイル用に設定を行う必要があります。

次に、gnu make にて、ビルドを行います。

Makefileは、${CVM_HOME}/cdcfoundation/build/btron-sh/ にある Makefile を指定します。

この際に、CVM_DEBUG および、J2ME_CLASSLIB の値をそれぞれ上記スライドの値に指定してmakeします。

T-JV生成物

- ${CVM_HOME}/cdcfoundation/build/btron3-sh3/

bin/cvm	JavaVM
bin/cvm -map	
lib/foundation.jar	FP クラスライブラリ
lib/security/java.policy	
lib/security/java.security	

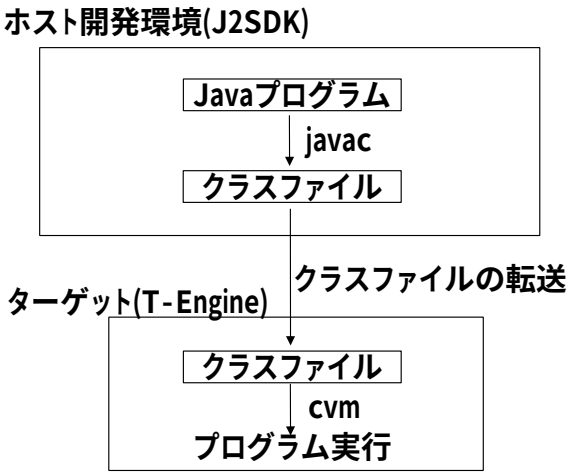
Copyright (C) T-Engine Forum 2005 All rights reserved.

make が完了すると、CVMのバイナリ、クラスファイルが作成されます。
作成されるファイル等は、

${CVM_HOME}/cdcfoundation/build/btron3-sh3/

以下に作成されます。
このファイルをT-Engine側に転送することで、CVMをT-Engine上で使用することが可能となります。

Javaプログラム開発の流れ



2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 75

Javaアプリケーションは、ホスト開発環境にて作成します。Javaプログラムを作成した後、javaコンパイラ(javac) にてコンパイルを行い、クラスファイルを作成します。

コンパイラは、J2SE向けのクラスファイルを作成しますが、本アプリケーションはJ2MECDC上で稼動するため、CDCおよびFoundationプロファイルに定義されていないAPIは使用できない点に注意する必要があります。

クラスファイルが作成されたら、転送ソフトウェア等を利用してターゲットとなるT-Engineに転送します。

T-Engine側に転送されたアプリケーションは、gterm上のコンソールから cvm を起動させるか、マイクロスクリプトで起動用のスクリプトを作成して、T-EngineのLCD上から起動することとなります。

開発の流れ ～ ホスト側

- 基本的には、通常のJavaプログラムの作成と同じ
  - JDK1.3
- J2ME特有のクラスを利用するときには、クラスライブラリーを含める
  - javax.microedition.io
  - org.tron.io

```
javac -classpath ${CVM_HOM}/lib/foundation.jar MyApp.java
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

T-JVは、CDCベースのJava実行環境であるため、javax.micoedition.io など J2ME固有のAPIも利用できますが、コンパイルする際には、これらのクラスファイルもCLASSPATHに含めておく必要があります。

開発の流れ ～ ターゲット側

- **cvmの実行**
  /SYS/WORK/java/bin/cvm
- **起動時オプション**
  -Djava.class.path
  -Xbootclasspass/a

```
/SYS/WORK/java/bin/cvm -Djava.class.path=/SYS/WORK/java/test MyApp
```


Copyright (C) T-Engine Forum 2005 All rights reserved.

ホストPC上で作成されたクラスファイルを、T-Engine上に持ってきた後、上記の起動オプションを指定して、JavaVMを起動させることで、アプリケーションを実行することが可能となります。

TRON Code Profile

Copyright (C) T-Engine Forum 2005 All rights reserved.

本章の概要および前提知識

- 本章の概要
  - TRONの特長である多漢字機能の概要とその意義の説明
  - T-JV上で多漢字機能を扱うためのAPIセットである、TRON Code Profileについての解説
- 本章を読むために必要な前提知識
  - JavaやT-JVについての一般的な知識

Copyright (C) T-Engine Forum 2005 All rights reserved.

・本章の前半ではTRONの特長である多漢字機能の概要とその意義について説明し、本章の後半では、それをT-JV上で実現する TRON Code Profile について解説します。

本章でご紹介する内容

- TRONコードとは

  多漢字問題とTRONコードの意義
  - 多漢字･多文字の応用例
  - トロンプロジェクトとTRONコード

  TRONコードの実際
  - 最大150万字で拡張可能、現在17万字
  - Unicodeを包含、韓国、中国、日本の字体も区別
  - 原規格との対応
  - TRONコードの仕組み
  - 無償で利用可能な多漢字･多書体フォント
- T-JVとTRONコード
- TRON Code Profileの実際
- 本章のまとめ

Copyright (C) T-Engine Forum 2005 All rights reserved.

TRONコードとは

多漢字問題とTRONコードの意義

- 多漢字･多文字問題とは
  コンピュータ上で、より多種類の漢字を扱う要求と、それが簡単に実現できない状況を表現した語

- 過去の文献の電子化やアーカイブにはもちろん、人名、地名などの固有名詞を扱う際には不可避
  JISやUnicodeでは致命的に不足
  外字はインターネットや電子メールで使えない
  住基ネット実現の際も問題に

  → 「統一文字」という規格で部分的解決を図る

【例】参議院議員の氏名に使われる漢字の不備
http://www.sangiin.go.jp/japanese/joho1/seiji.htm

Copyright (C) T-Engine Forum 2005 All rights reserved.

・【例】で示したサイトは参議院のページであり、JIS第1･第2水準で表記できない議員の氏名を画像で表示しています。

多漢字･多文字の応用例

- 多数の人名を扱う名簿管理や電子政府端末
  - 戸籍管理、顧客管理(DMなど)、人名データベース
  - 個人レベルでは年賀状の宛名など
- 電子ブック、電子辞書
- 国文学や歴史学、文化的な視点から
  - 図書館の蔵書検索システム
  - 過去の史料や文学作品の正確な翻刻、アーカイブ
  - 国文や漢文の研究者の間での情報流通(メール等)

Copyright (C) T-Engine Forum 2005 All rights reserved.

・多漢字機能の応用例としては、多数の正確な人名の表記が必要な名簿管理システムや電子政府関連システムが重要です。

・また、国文学や歴史に関係する文化的な分野でも、多漢字機能の需要が高いです。従来は外字で対応していましたが、外字では情報流通に問題が残ります。TRONコードなら、外字に頼らずすべての文字にコードを割り当てることができるので、情報流通やアーカイブにも安心して利用できます。

多漢字･多文字の応用例: 三国志

Copyright (C) T-Engine Forum 2005 All rights reserved.

･例として、三国志に出てくる人名の漢字も、TRONコードを使えばこのように正確に表記できます。

トロンプロジェクトとTRONコード

- トロンプロジェクトの開始当初の1980年中頃より、国際的な多言語システムを研究
- 1990年代の中頃から後半にかけて、多漢字を実装
- 1999年に最大150万字処理可能なOS「超漢字」を発売
- TRONで扱うネイティブの文字コードがTRONコード

【TRONコードの設計方針】

(a) 同時に混在して扱える文字数に制限を設けない

(b) 複数の文字セット(原規格という意味で)を包含

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved. 84

・TRONプロジェクトとTRONコードの歴史や経緯について説明します。

・TRONコードは、JISやUnicodeなど複数の文字セットを統合せずに並行して取り入れるための「器(うつわ)」です。

TRONコードの実際

最大150万字でさらに拡張可能、現在17万字

JIS X 0208:1997
JIS X 0213:2000
JIS X 0212-1990
GT Font
Dai Kanwa Jiten
KS X 1001:1992
GB 2312-80
CNS 11643-1986
i-mode Pictogram
Braille
Symbols, etc.
Unicode

2005.09.13 version 1.01
Copyright (C) T-Engine Forum 2005 All rights reserved.

・ここに例示した文字は、T-Engine 上でもすべて扱うことができます。

・右の画面は、TRONの文字検索データベースを使って、「新」「韓」という部品を含む漢字を検索したところです。

Unicodeを包含、韓国、中国、日本の字体も区別

	Unic	TR	日	中	韓	台湾	GT	J	大漢
729	50D8	儉	儉				儉		儉
730	50D9	儒					儒儒	儀	儀
731	50DA	僚	僚	僚	僚	僚	僚		僚
732	50DB	僧				僧	僧		僧
733	50DC	僭	僭				僭		僭
734	50DD	僵	僵			僵	僵		僵
735	50DE	僞	僞		僞		僞僞		僞
736	50DF	儀	儀				儀		儀
737	50E0	儒				儒	儒		儒
738	50E1	德					德	德	德
739	50E2	儛	儛				儛儛		儛
740	50E3	僧	僧			僧	僧		僧
741	50E4	僤	僤			僤	僤		僤
742	50E5	僥	僥		僥	僥	僥		僥
743	50E6	僕	僕	僕		僕	僕	僕	僕
744	50E7	僧	僧	僧	僧	僧	僧僧	僧	僧僧
745	50E8	僖	僖			僖	僖僖		僖
746	50E9	僊	僊			僊	僊僊	僊	僊
747	50EA	僑				僑	僑		僑
748	50EB	僮					僮		僮
749	50EC	僬		僬		僬	僬		僬
750	50ED	僧	僧	僧	僧	僧	僧僧		僧
751	50EE	僮	僮	僮		僮	僮		僮
752	50EF	僑	僑			僑	僑僑		僑
753	50F0	雙				雙	雙		雙
754	50F1	僱	僱			僱	僱僱		僱
755	50F2	僔	僔				僔僔	僔	僔
756	50F3	僕		僕		僕	僕	僕	
757	50F4	僩					僩		
758	50F5	僵	僵	僵		僵	僵		僵
759	50F6	僶	僶			僶	僶		僶

Copyright (C) T-Engine Forum 2005 All rights reserved.

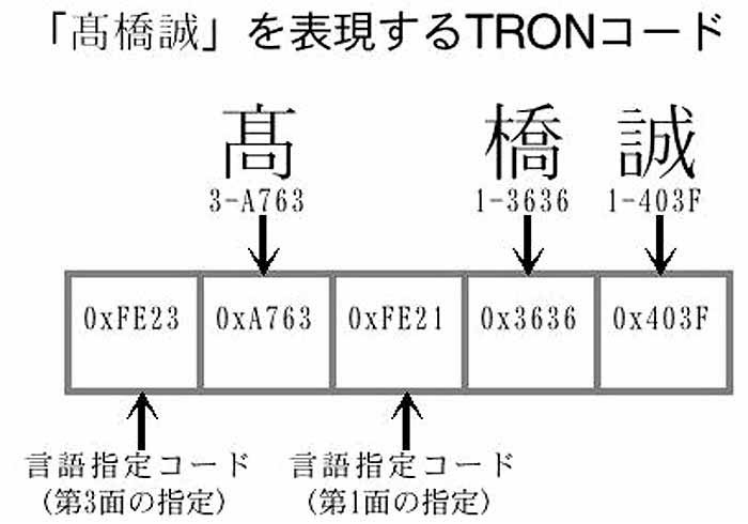
• 「僧」という漢字は、日本、中国、韓国で細かい字形が違っています。この違いは、日本、中国、韓国の「僧」に共通の文字コードを与える Unicode では統合されてしまい、区別できなくなってしまいます。しかし、TRON コードを使えば、日本の「僧」、中国の「僧」、韓国の「僧」をきちんと区別して扱うことができます。

原規格との対応

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

・TRONコードでは、JISや各国の文字セットを、Unicodeのように統合することなく、TRONコードの各面にそのまま配置しています。

TRONコードの仕組み



Copyright (C) T-Engine Forum 2005 All rights reserved.

・TRONコードの具体的なコーディング方法について説明しています。TRONコードでは、16ビットの容量を持つ言語面(スクリプト)を複数切り替えることにより、150万の文字を利用できます。異なる言語面への切り替えが発生する際には、この図のように、言語指定コードを挿入します。

無償で利用可能な多漢字･多書体フォント

- SLフォント(仮称)の3書体を準備中
- 定められた条件の下では、商用利用も含めてライセンス費用不要(予定)

コラム: 研究用には甲骨文字なども準備中

Copyright (C) T-Engine Forum 2005 All rights reserved.

･TRONコードに対応した多漢字の文字フォントは、既に開発済みのGT書体のほか、ゴシック体、明朝体、楷書体などのフォントの開発が進んでおり、組み込みシステムにおいても無償で利用できるようになる予定です。

T-JVとTRONコード

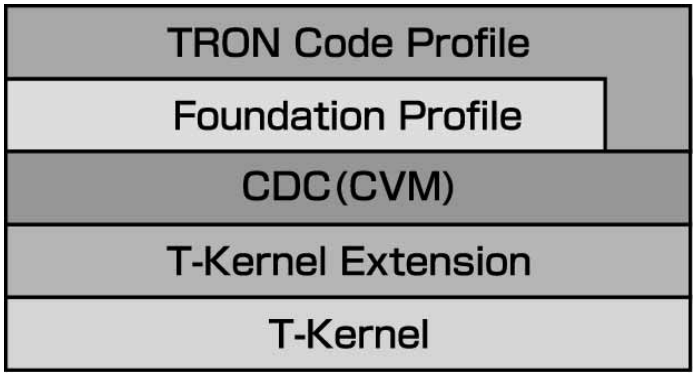
- Java VMには手を加えずプロファイルレベルで実装
  TRONコードを扱うための各クラスを追加
  - TRONコードでの文字列操作
  - TRONコードでのストリーム入出力、ファイルI/O
  - TRONコード とUnicodeの間の変換も可能

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

•Java には Unicode用の Char クラス、String クラスがありますが、これはT-JVでもそのまま利用できます。T-JVの TRON Code Profile では、これに加えて、TRON コードに対応した TChar クラス、 TString クラスの機能を提供します。

•UnicodeとTRONコードの間の変換も可能です。(ただし Unicode 表に存在しない文字は変換できません)

ソフトウェア構成図



Copyright (C) T-Engine Forum 2005 All rights reserved.

・図は、TRON Code Profile に関連したソフトウェア構成を示すものです。T-Kernel + Extension の上に J2ME (Java2 Platform, Macro Edition) の Connected Device Configuration (CDC) / Foundation Profile を載せ、その上に TRON Code Profile を載せた構成になっています。

TRON Code Profileの実際

- TCharクラス

  TRONコード1文字を扱う
  (標準のCharクラスのTRONコード版)

  - コンストラクタ

    TRONコード、&T形式、Unicode 1文字などを指定して
    TRONコード1文字の生成が可能

  - メソッド

    文字比較　　compareTo()
    長さ1のTStringへの変換 toTString()など

2005.09.13 version 1.01　Copyright (C) T-Engine Forum 2005 All rights reserved.

・TChar クラスは TRONコードの1文字を扱うクラスです。

・実装上の内部表現としては、言語指定コードとその言語面の中の16ビット文字コードを合わせて、1文字を32ビットで扱います。

TRON Code Profile API(1)

- **TStringクラス**
  TRONコード文字列を扱う
  (標準のStringクラスのTRONコード版)

- **コンストラクタ**
  TRONコード配列、Unicode文字列などを指定して
  対応するTRONコード文字列の生成が可能

- **メソッド**

文字列比較	compareTo()
文字列連結	concat()
部分文字列抜き出し	substring()、
Unicode文字列への変換	toString() など

Copyright (C) T-Engine Forum 2005 All rights reserved.

・TString クラスは、TRONコード文字列を扱うクラスです。

TRON Code Profile API(2)

- TInputStreamReaderクラス
  - バイトストリームからTRONコードストリームへの変換クラス
- TOutputStreamWriterクラス
  - TRONコードストリームからバイトストリームへの変換クラス
- TFileReaderクラス
  - ファイルからTRONコードを読み込むクラス
- TFileWriterクラス
  - ファイルへTRONコードを書き込むクラス

Copyright (C) T-Engine Forum 2005 All rights reserved.

・このように、TRONコードのバイトストリームやファイルを扱うクラスも用意されています。

本章のまとめ

- コンピュータ上で正確な文字を扱うためには、TRONコードによるソリューションが有効
- TRON Code Profileにより、Java上でTRONコードを扱うことが可能

2005.09.13 version 1.01 Copyright (C) T-Engine Forum 2005 All rights reserved.

・本章のまとめと参考文献を示します。

参考URL・参考文献

Copyright (C) T-Engine Forum 2005 All rights reserved.

参考URL

- T-Engineフォーラム
  - http://www.t-engine.org/
- トロンプロジェクト
  - http://www.tron.org/
- 超漢字ウェブサイト
  - http://www.chokanji.com/
- TFTSのプレス発表
  - http://www.uidcenter.org/japanese/press/TEP040310.pdf
- Java Technology
  - http://java.sun.com/

Copyright (C) T-Engine Forum 2005 All rights reserved.

参考文献

- T-Kernel標準ハンドブック
  - 監修 坂村健　編著 T-Engineフォーラム
- 技術情報誌『TRONWARE』
- 他、T-Engine/TRON関係書籍
  - パーソナルメディア株式会社
    - http://www.personal-media.co.jp/book/

Copyright (C) T-Engine Forum 2005 All rights reserved.

用語集

用語	欧文表記	意味
CDC	Connected Device Configuration	J2MEの中でも、カーナビやPDAなどの比較的ハイエンドな組み込み機器向けのConfiguration。なお、CDCの仕様は、JCPによってJSR-000036として規定されている。
CLDC	Connected Limited Device Configuration	J2MEの中でも、携帯電話などの比較的ローエンドな組み込み機器向けのConfiguration。なお、CLDCの仕様は、JCPによってJSR-000030として規定されている
CLI	Command Line Interpreter	パーソナルメディア株式会社より発売されている「T-Engine開発キット」に付属する開発用ツール。T-Engine上のアプリケーションのひとつであり、下記機能を有する。 ・コマンドによるファイルを中心とした各種操作 ・システムプログラム（サブシステム）のロード／アンロード ・アプリケーションプログラムの実行 ・コマンドファイルの実行
Configuration	Configuration	Java用語。J2SEから機能を絞り込んだVMを含むサブセットで、携帯電話や家電など比較的ローエンドな組み込み機器向けの「CLDC」と、PDAやカーナビなど比較的ハイエンドな組み込み機器向けの「CDC」の2種類がある。
FP	Foundation Profile	J2MEの中でも、CDCの上位に提供されるAPI群のProfile。
GNU	GNU is Not Unix.	FSF(Free Software Foundation)が進めているUNIX互換ソフトウェア群の開発プロジェクトの総称。 http://www.gnu.org/
gterm	gterm	T-Engine開発環境に付属するターミナルソフト。
Hot Spot	Java HotSpot	Javaバイトコード内で繰り返し処理などの部分を、実行前にあらかじめネイティブコードに変換しておくことにより高速実行を実現するJVMに内蔵された機構。
J2EE	Java2 Enterprise Edition	Javaプラットフォームの中でも、エンタープライズシステムのための共通プラットフォーム
J2ME	Java2 Micro Edition	Javaプラットフォームの中でも、組み込み機器などリソースが限られた環境向けの共通プラットフォーム。様々なデバイスの要求に応えるため、Configuration、Profile、Optional Packages等が各種定義されている。
J2SDK	Java2 Standard Development Kit	Java2アプリケーションを開発するための環境および開発キット。
J2SE	Java2 Standard Edition	Javaプラットフォームの中でも、主にデスクトップシステムを対象としたプラットフォーム。Java2プラットフォームの中核に位置する。

用語	欧文表記	意味
Java	Java	SunMicrosystems Inc.社が1995年に開発したオブジェクト指向プログラミング言語、および、その実行環境
JCP	Java Community Process	SunMicrosystems Inc.社が中心となって運営する、Javaに関する標準仕様の決定機構。様々な企業が本機構に参加しており、参加メンバは各種仕様の変更リクエストを自由に提出することが可能となっている。
JITコンパイラ	Just-In-Time Compiler	Javaのクラスをロードした後にネイティブコードに変換するコンパイラ。
JNI	Java Native Interface	JVMとネイティブ機能をつなぐインタフェース。
JRE	Java Runtime Environment	Java2アプリケーションを実行するための環境。
JSR	Java Specification Requests	Javaの新しい仕様または、仕様改訂のための提案。JCPのメンバによりリクエストされる。
JVM	Java Virtual Machine	Javaバイトコードをそのプラットフォームのネイティブコードに変換して実行するソフトウェア。Java仮想マシン。
Kasago for T-Engine	Kasago for T-Engine	組み込みシステム用TCP/IPプロトコルスタックKASAGO IPv6をT-Engineの各種仕様に対応させた株式会社得るミックシステムの製品。
KVM	Kilo Virtual Machine	SunMicrosystems Inc.社の開発した、組み込み機器向けのJava仮想マシン。サイズがキロバイト程度のため、Kilo VMと呼ばれている。
MIDP	Mobile Information Device Profile	J2ME/CLDC用のプロファイルのひとつで、携帯電話などの携帯端末向けに定義されたJava実行環境の仕様。
PBP	Personal Basis Profile	J2MEの中でもFPの上に提供される、カーナビゲーションの操作画面など、GUIを提供するための比較的低レベルのグラフィックス処理を実装するためのAPIが提供されているProfile。
PP	Personal Profile	J2MEの中でもFPおよびPBPの上に提供される、awtをベースとした高レベルユーザインタフェース構築のための高レベルAPI Profile。
Profile	Profile	Java用語。J2MEにおけるConfigurationの上位に置かれる、各組み込み機器向けに特化したAPIセット。
RS-232C	Recommended Standard 232 version C	米国電子工業会により標準化されたシリアル通信規格。開発ホストのPCとターゲットであるT-Engine間をシリアル通信で接続する際に利用する。

用語	欧文表記	意味
SCSL	Sun Community Source License	SunMicrosystemsInc.社が同社のプログラミング言語、およびその開発･実行環境である「Java2」のソースコードに対して適用したライセンス体系。
T-Engineフォーラム	T-Engine Forum	組み込み機器向けのリアルタイムOS「μITRON」をベースにした開発プラットフォーム「T-Engine」に関連する技術の研究開発、標準化、普及に取り組む業界団体。
T-Engineフォーラム Java WG	T-Engine Forum Java WG	T-Engineフォーラムのワーキンググループのひとつ。Javaに特化したテーマで議論を行うソフトウェア関連企業中心の会議体。
T-JV	T-JV	T-Engine/T-Kernel上のJava実行環境。J2MEのCDCが採用されている。
T-Kernel	T-Kernel	T-Engineの標準リアルタイムカーネル。μITRONと似た機能やAPIを持つが、ミドルウェアの流通性を高める機能が追加されている。ソースが公開されており、利用条件(T-License)に同意すれば、誰でも無償で利用できる。
T-Kernel Extension	T-Kernel Extension	T-Kernelで共通に利用される、基本的･汎用的なミドルウェア。プロセス管理や仮想記憶、ハイパーテキスト型のファイル管理などを提供するStandard Extensionのほか、Tiny Extension、Enterprise Extensionなどがある。
TRON	The Realtime Operating system Nucleus	理想的なコンピュータアーキテクチャの構築を目指して、1984年に東京大学の坂村氏が始めたプロジェクト。
T-Shell	T-Shell	パーソナルメディア株式会社より発売されている多漢字GUIシステムを実現するT-Engine用ミドルウェア。
UC	Ubiquitous Communicator	YRPユビキタス･ネットワーキング研究所が開発した、PDA型のユビキタスコンピューティング環境とのコミュニケーションツール。
UNIXエミュレータ	UNIX Emulator	T-Engine上でUNIXの機能を実現する機能。T-JV（Sun）で提供されるUNIXエミュレータとPMCから提供されるUNIXエミュレータには互換性がない点に注意する。
カーネル	Kernel	OSの基本機能を実装したソフトウェア。
クロス開発環境	Cross-Development Environment	プログラムのコンパイルを行うマシン（ホスト）とコンパイルされたプログラムを実行するマシン（ターゲット）が異なる開発環境。
ミドルウェア	Middleware	OS上で動作し、アプリケーションソフトウェアに対してOSよりも高度で具体的な機能を提供するソフトウェア。OSとアプリケーションソフトウェアの間の中間的な性格を持っている。ソフトウェアとほぼ同義。

T-JV 入門　執筆協力者（順不同・敬称略）

岡田　弘一	株式会社アプリックス
金子　久美子	株式会社アプリックス
吉岡　学	株式会社アプリックス
上原　富士雄	株式会社アプリックス
神戸　博之	サン・マイクロシステムズ株式会社
向井　真也	富士通株式会社
松為　彰	パーソナルメディア株式会社
中村　大真	パーソナルメディア株式会社
水本　米喜	東芝ソリューション株式会社
越塚　登	YRP ユビキタス・ネットワーキング研究所
田口　剛生	YRP ユビキタス・ネットワーキング研究所
河合　泰浩	沖電気工業株式会社

T-Engine フォーラム　Java WG

T-JV 入門

2005 年 9 月 13 日　発行

発行所

T-Engine フォーラム

〒141-0031 東京都品川区西五反田 2-20-1 第 28 興和ビル

URL : http://www.t-engine.org/

TEL : 03-5437-0572 (窓口)　　FAX : 03-5437-2399 (窓口)

Copyright (c) T-Engine Forum 2005 All rights reserved.