T-Engine Forum
Specification

TEF040-S104-1.00.01/ja
2006-06-06

T-Format (4): バイナリコード形式

T-Format (4): Binary Code Format

Number:	TEF040-S104-1.00.01/ja
Title:	T-Format (4): バイナリコード形式
	T-Format (4): Binary Code Format
Status:	[ ] Working Draft, [ ] Final Draft for Voting, [X] Technical Report
Date:	2003/02/28 First Edited
	2003/08/01 Updated to 1.00.00
	2006/06/06 Updated to 1.00.01

目 次

はじめに (Foreword)

あらゆるものにコンピュータが入りネットワークでつながれるユビキタス・コンピューティング環境の構築を目指した、オープンなリアルタイムシステム標準開発環境を提供するため、T-Engine プロジェクトが発足した。 T-Engine は携帯情報機器やネットワーク接続型の家電機器などを効率良く短期間で開発するのに最適な開発環境の提供を目指している。T-Engine は eTRON と呼ばれるプロジェクトのネットワークセキュリティアーキテクチャに対応し、安全でない通信路を介した通信においても、盗聴、改竄、なりすましを防御して安全に目的の相手に電子情報を送る機構を備える。

効率のよい開発をサポートするために、規格化されたハードウェア（T-Engine ボード）、標準リアルタイムカーネル（T-Kernel）を定め、ミドルウェアを流通させることに特に力を入れている。 また、T-Engine は半導体メーカー、ハードウェアメーカー、ソフトウェアメーカー、システムメーカーの連携を円滑にし、相互のビジネスを活発化し、開発期間や開発コストの低減により付加価値の高い製品を短期間で提供することを狙っている。更に、T-Engine は高度な半導体技術や実装技術、ソフトウェア技術を採用し、 他に追随を許さない先進的な応用製品の開発を行う。

規定範囲 (Scope)

T-Format は T-Engine、T-Kernel 上で動作するミドルウェアやアプリケーションソフトウェアのコード形式を規定する。T-Format には、以下の 3 種類の規定が含まれる。

1. ソースコードスタイルガイドライン
   T-Engine、T-Kernel 上で動作するミドルウェアやアプリケーションソフトウェアのソースコードのスタイル形式。異なるベンダーが作成したソースコードを組み合わせてコンパイル、リンクできるための規定である。

2. 標準バイナリ形式
   T-Engine、T-Kernel 上で動作するミドルウェアやアプリケーションソフトウェアがバイナリコードで配布される際の標準実行形式。実行コード形式とデバッガシンボル形式を規定する。

3. 標準ドキュメント形式
   流通するミドルウェアやアプリケーションソフトウェアに添付するドキュメントの種類と形式に関する規定。

本仕様は上記の T-Format の一部分を構成し、配布するバイナリコードのスタイルガイドラインを定めるものである。また、デバッガシンボル形式などのデバッグ情報については、ソースコードでの配布が前提となるため、本仕様では規定しない。

参照規定 (Normative References)

[1] T-Engine Forum. "T-Format (3) : Global Symbol Naming Rule in C Language", TEF-040-S103, 2002.

[2] Free Software Fundation. "GCC, the GNU Compiler Collection". http://gcc.gnu.org/

[3] Free Software Fundation. GNU. http://gnu.org/

[4] Tool Interface Standard. "Executable and Linkable Format (ELF)". Specification Version 1.1

用語定義 (Terms and Definitions)

ミドルウェア (Middleware) “T-Format(3)[1] 用語定義” を参照。

1. T-Engine ミドルウェアモデル

“T-Format(3)[1] T-Engine ミドルウェアモデル” を参照。

2. 配布されるバイナリ形式

T-Format バイナリ形式は、以下の二つの条件を満たす形式である。

1. ELF（Executable and Linking Format）[4] 仕様を満たす（必要条件）。
2. CPU はコンパイラ等に依存する部分で、ELF 仕様では規定できない部分に関しては、T-Engine Forum が別途定める GNU[3] ベースのリファレンス開発環境による実装に従う。

以下に述べる技術的理由により、実装に依存せず、且つ現在の T-Engine に実装されているあらゆる CPU において必ず動作を保証することができる有効なバイナリ形式の仕様、さらにそれを扱うことができる実装は存在しなかった。そのため、現実的なバイナリ形式の規定は、リファレンス開発環境の GCC（GNU Compiler Collection）[2] でコンパイルされてバイナリコードを標準バイナリコード形式とすることであるという結論に達した。

2.1 コンパイラによる相違の問題

標準バイナリ形式として、多くのコンパイラでサポートされ、また、GNU 開発環境の主要なバイナリ形式である、ELF（Executableand Linking Format）が考えられる。しかし、ELF はコンパイラ依存の情報をもつ可能性があり、コンパイラが異なると同じバージョンの ELF 形式のオブジェクトファイルであっても、動作は保証することはできない。従って、ELF 仕様だけをもってして、標準バイナリ形式仕様として、オブジェクトファイルの形式に規定するだけでは、T-Format が求めるソフトウェアのバイナリ流通性の要求を満たすことはできない。

2.2 GCC の ABI(Application Binary Interface) の問題

GCC の ABI は、GCC のバージョンが異なっても基本的には不変ではあるが、必ず同じであるとは限らず、バージョンによって部分的に異なる場合がある。例えば、GCC3.0 と GCC3.1 の間では、C++の ABI について変更が行われている。また、GCC はソースコードで配布されるので、ユーザが各自の環境においてビルドを行う。その際のビルドオプションによっては、ABI が異なる場合があるため、同じバージョンの GCC でコンパイルを行ったバイナリコードであっても、ABI が必ず同じであるとは限らない。例えば、オプション例として、C 言語のグローバルシンボルに「_」を付加する/付加しないといった違いがある。

以上の理由から、GCC のリリースバージョンを規定することによっても、T-Format が求めるソフトウェアのバイナリ流通性の要求を満たすバイナリ形式を規定したことにはならない。