組込み RTOS 向けアプリケーション開発支援ツール
TLV（トレース ログ ヴィジュアライザー）
フェーズ 4
アプリログ拡張外部仕様書

2009 年 6 月 16 日

改訂履歴

版番	日付	更新内容	更新者
1.0	09/6/2	新規作成	柳澤大祐

目次

1 はじめに 3

1.1 本書の目的 3

1.2 本書の適用範囲 3

1.3 用語の定義/略語の説明 3

1.4 概要 3

2 概要説明 4

2.1 アプリログ拡張 4

2.2 文字列可視化 5

2.3 ユーザ定義状態可視化 7

2.4 ユーザ定義状態可視化（端点付き） 7

1　はじめに

1.1　本書の目的

本書の目的は、文部科学省先導的 IT スペシャリスト育成推進プログラム「OJL による最先端技術適応能力を持つ IT 人材育成拠点の形成」プロジェクトにおける、OJL 科目ソフトウェア工学実践研究の研究テーマである「組込み RTOS 向けアプリケーション開発支援ツールの開発」に対して、その開発するソフトウェア拡張の外部仕様を記述することである。

本書は特に、フェーズ 4 におけるアプリログ拡張の外部仕様に関する記述を行う。

1.2　本書の適用範囲

本書は、組込み MPRTOS 向けアプリケーション開発支援ツールの開発プロジェクト（以下本プロジェクト）のフェーズ 4 におけるアプリログ拡張の外部仕様に関する記述を行う。

1.3　用語の定義/略語の説明

表 1　用語定義

用語・略語	定義・説明
TLV	Trace Log Visualizer
MPRTOS	マルチプロセッサ対応リアルタイムオペレーティングシステム
トレースログファイル	RTOS のトレースログ機能を用いて出力したトレースログや、シミュレータなどが出力するトレースログをファイルにしたもの
標準形式トレースログファイル	本ソフトウェアが扱うことの出来る形式をもつトレースログファイル。各種トレースログファイルは、この共通形式トレースログファイルに変換することにより本ソフトウェアで扱うことが出来るようになる。
変換ルール	トレースログファイルを標準形式トレースログファイルに変換する際に用いられるルール。
可視化ルール	標準形式トレースログファイルを可視化する際に用いられるルール。
TLV ファイル	本ソフトウェアが中間形式として用いるファイル。前述の標準形式トレースログファイルは、この TLV ファイルの一部である。

1.4　概要

本書では、組込み MPRTOS 向けアプリケーション開発支援ツールのソフトウェアの仕様を記述する。本書は特に、フェーズ 4 におけるアプリログ拡張の外部仕様に関する記述を行う。

2　概要説明

これまでに寄せられた TLV への要望として、printf デバッグを行いたいというものがあった。一般的な printf デバッグによる出力を TLV のタイムライン上に表示させることによって、メッセージが出力されたときのタスク状態などが一目でわかるという利点がある。この要望を実現するものがアプリログ拡張である。

図 1　アプリログ拡張イメージ

2.1　アプリログ拡張

今までの TLV では、任意の文字列を可視化することができないので、printf デバッグを行えない。そこで、フェーズ 4 での要求として任意文字列を可視化することが挙げられている。

例えば、`[123456789] printf foobarbaz.` のようなログがあった場合、時刻 123456789 の位置に foobarbaz というメッセージが表示されるような可視化を想定する。

本拡張では、以下の 3 つの可視化方法を提供する。

文字列可視化　任意文字列を TLV 上に表示
ユーザ定義状態可視化　タスクの状態表示のように、ユーザが任意に決めた状態を可視化
ユーザ定義状態可視化（端点付き）　上記の可視化に加え、開始と終了があるもの。関数コールを想定

TOPPERS カーネルでは、ユーザアプリケーションからのログ出力は syslog 関数によって行う。よって、ユーザアプリケーション側からの利用は syslog 関数や、別に用意する API を介して行うものとする。

本機能をユーザアプリケーションから利用する場合は、以下の手順で行う。

1. TLV のリソースファイルに追記
2. アプリケーションにログ出力関数コールを追加（syslog 関数または本拡張で提供する API）
3. アプリケーションを実行
4. TLV を実行

また、今回の拡張は TLV 本体には手を入れず、可視化ルールなどの記述のみで実現することとした。

2.2 文字列可視化

文字列の可視化には、タスクに関連するものとそうでないものがある。以降、タスクに関連する文字列可視化を**タスク文字列可視化**、タスクに関連しない文字列可視化を**文字列可視化**と呼ぶ。

タスク文字列可視化は、タスクの属性として定義することで、現在のタスク状態表示などの上に重ねて表示する。文字列可視化は、それ専用の行で表示する。

2.2.1 入力

斜体で表示されている箇所は、アプリケーション開発者が望む値に置換する場所であることを示している。

■ログ書式 TLV へ入力するログの書式を定義する。

文字列可視化のログ書式

■syslog syslog(“applog id %s %s.”,*id*,*str*);

■API void applog_str(const char *_id_, const char *_str_)

■ログ出力 [*time*] applog id *id* *str*.

time 時刻 `[0-9a-zA-Z]+`
id 表示行 ID `[^\. ]+`
str 出力文字列 `[^\.]*`

タスク文字列可視化のログ書式

■syslog syslog(“applog task %s %s.”,*tid*,*str*);

■API void applog_tsk(const char *_tid_, const char *_str_)

■ログ出力 [*time*] applog task *tid* *str*.

time 時刻 `[0-9a-zA-Z]+`
tid タスク ID `[0-9]+`
str 出力文字列 `[^\.]*`

■リソースファイル書式 以下の記述は、文字列可視化を行う際に、ユーザがリソースファイルに追加するものである。

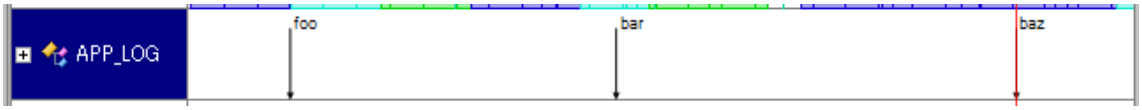

図 2　文字列可視化

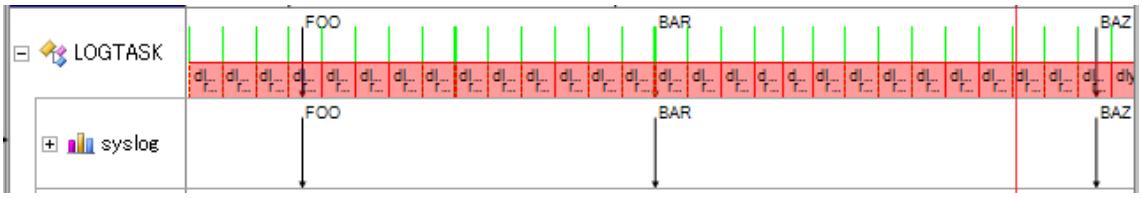

図 3　タスク文字列の可視化

文字列可視化リソースファイル書式

```
"rowname":{
    "Type":"ApplogStr",
    "Attributes":{
        "id":"",
        "str":""
    }
}
```

rowname　文字列可視化行の名前 `[^"]+`

2.2.2　出力

以下では、文字列の可視化方法を定義する。

■文字列可視化　まず、文字列可視化を示す。

```
syslog("applog id 1 foo.");
```

または

```
applog_str("1", "foo");
```

このようなログが出力された場合、図 2 のように可視化を行う。

■タスク文字列可視化　次に、タスクに関連する文字列の可視化を示す。

```
syslog("applog task 1 FOO.");
```

または

```
applog_tsk("1", "FOO");
```

このようなログが出力された場合、図 3 のように可視化を行う。タスクの状態やシステムコールと重複して見づらい場合は、ユーザが不要な可視化項目を可視化しないようにすることで見やすいよう設定することとする。

2.3　ユーザ定義状態可視化

2.3.1　入力

2.3.2　出力

2.4　ユーザ定義状態可視化（端点付き）

2.4.1　入力

2.4.2　出力

参考文献