# 「μITRON4.0 仕様に完全準拠し拡張を含む
# オープンソースμITRON 仕様 OS の開発」
# ダイナミックローディング機能 取り扱い説明書

2004 年 2 月 23 日 第 1 版

編集 :株式会社エーアイコーポレーション

目次

# 1 章　基本概念及び用語の説明

## 1.1 はじめに

組み込みシステムの多くは保守が困難な場所に隠蔽され設置されています。またこのようなシステムは一旦稼動すると、保守目的で電源をなかなか落とせない状況になります。日本の組み込みシステムにはコードサイズがコンパクトでリアルタイム応答特性に優れた μITRON 仕様 OS が多く採用されています。しかし従来の μITRON 仕様 OS では OS 機能を含めた全てのプログラムが 1 リンクされ、機器に ROM として組み込まれる事を前提としていました。

μITRON 仕様 OS は自身に動的なプログラム更新機能を持っていないため、機器の設置後バグが発見された場合でも、容易にプログラムの差し替えが出来ませんでした。特に設備に隠蔽された機器の場合、ROM 交換によるプログラム交換は大掛かりな保守工事を必要としました。

ダイナミックローディング機構は μITRON4.0 仕様フルセットカーネル上でミドルウェアとして動作し、ネットワークを介してシステムを止めることなくプログラムの更新や新規追加を可能にするものです。従来保守が難しかった設備に隠蔽された機器もこの機能を使用する事により、ネットワークに接続されていれば、μITRON のリアルタイム特性を損なうことなく、汎用 OS のような動的なプログラム更新の利便性を得る事が可能となります。

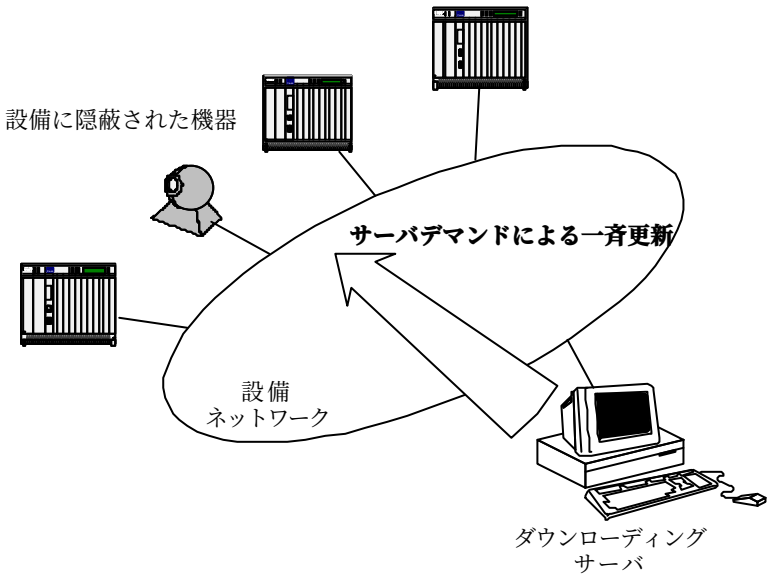



**図 1.1.1　設備に隠蔽された組み込み機器のソフトウェアアップデート**

携帯電話に代表されるように、何らかのユーザ操作インタフェースを持った機器においてもμITRON 仕様 OS が広く使われています。ダイナミックダウンローディング機能は今まで不可能であったユーザ自身の操作による（クライアントデマンド）、機器へのソフトウェアプラグインやソフトウェア不具合の改修を可能とします。

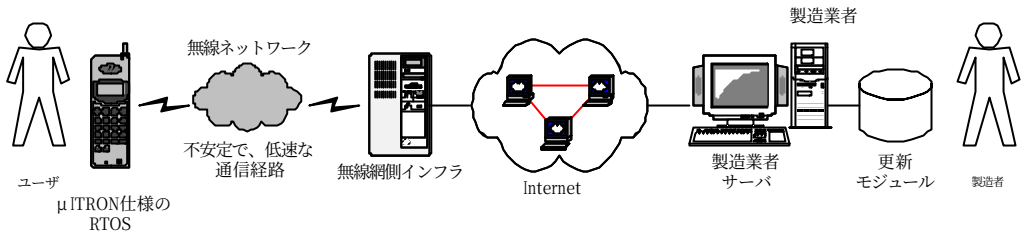



**図 1.1.2　ユーザインタフェース機能を持った機器のソフトウェアアップデート**

昨今、多くの機器が LAN やインターネットなどのネットワーク接続機能を内蔵したものが増えてきました。しかし必要な機能のみを実装する組み込み機器の世界において依然 ネットワーク機能を持たない機器が存在することも事実です。このような機器において FD や CF などのリムーバブルストレージデバイスを使用して修正が必要な不具合ソフトウェアモジュールのみ更新できれば保守の容易性が向上します。なにより全体書き換え更新の場合、システム全体を止める必要がありますが、部分更新方式の場合、更新機能のみの一時的な停止で、カーネル機能を含めその他の機能を動作させたままでソフトウェアの更新が可能となります。ダイナミックダウンローディング機能はネットワークで接続された機器はもとより、このようなネットワーク機能を持たない機器においてもダイナミックダウンロードによるソフトウェア更新を可能とします。

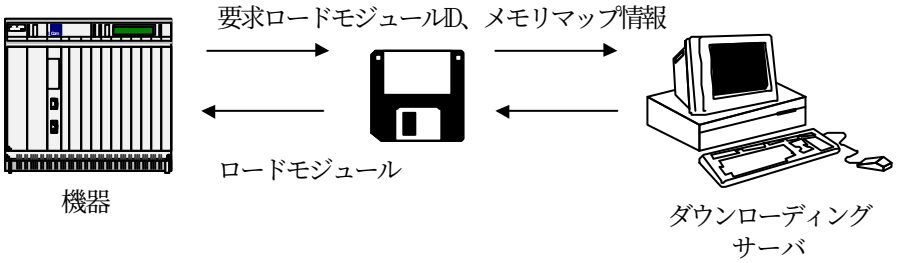



**図 1.1.3　リムーバブルストレージを使った機器のソフトウェア　アップデート**

# 1.2 基本スキーム

パーソナルコンピュータやワークステーションにおいて動作する汎用 OS では、OS 自身が動的なアドレス解決機能を持っています。一方、機器専用のアプリケーションプログラムを動作させる事を目的とした組み込みシステムでは、ランタイムに冗長なプログラムリンカ機能を搭載するのはハードウェアコストの面で許されないのが一般的です。そこでダイナミックダウンロード機構では下図のようにシンボルアドレス解決機能をパーソナルコンピュータシステム等の汎用プラットホームで行わせ、ターゲット機器ではローダ機能のみを搭載し、ネットワーク連携で動的にアドレス解決する機構を採用し実現しています。

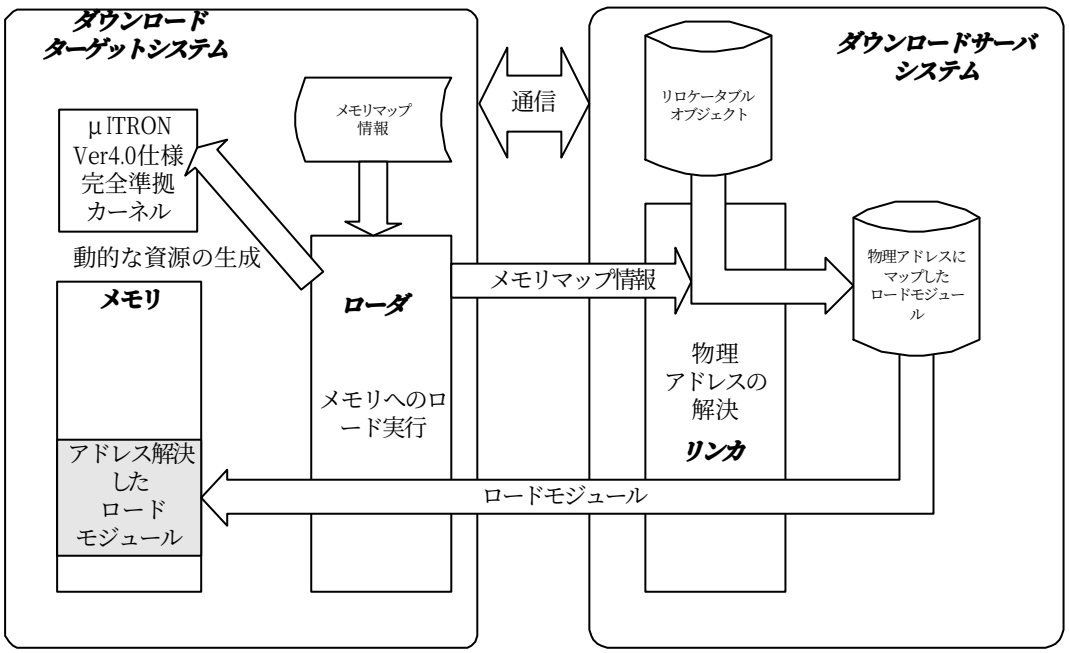



**図 1.2.1　基本スキーム**

# 1.3 ソフトウェア構成の説明

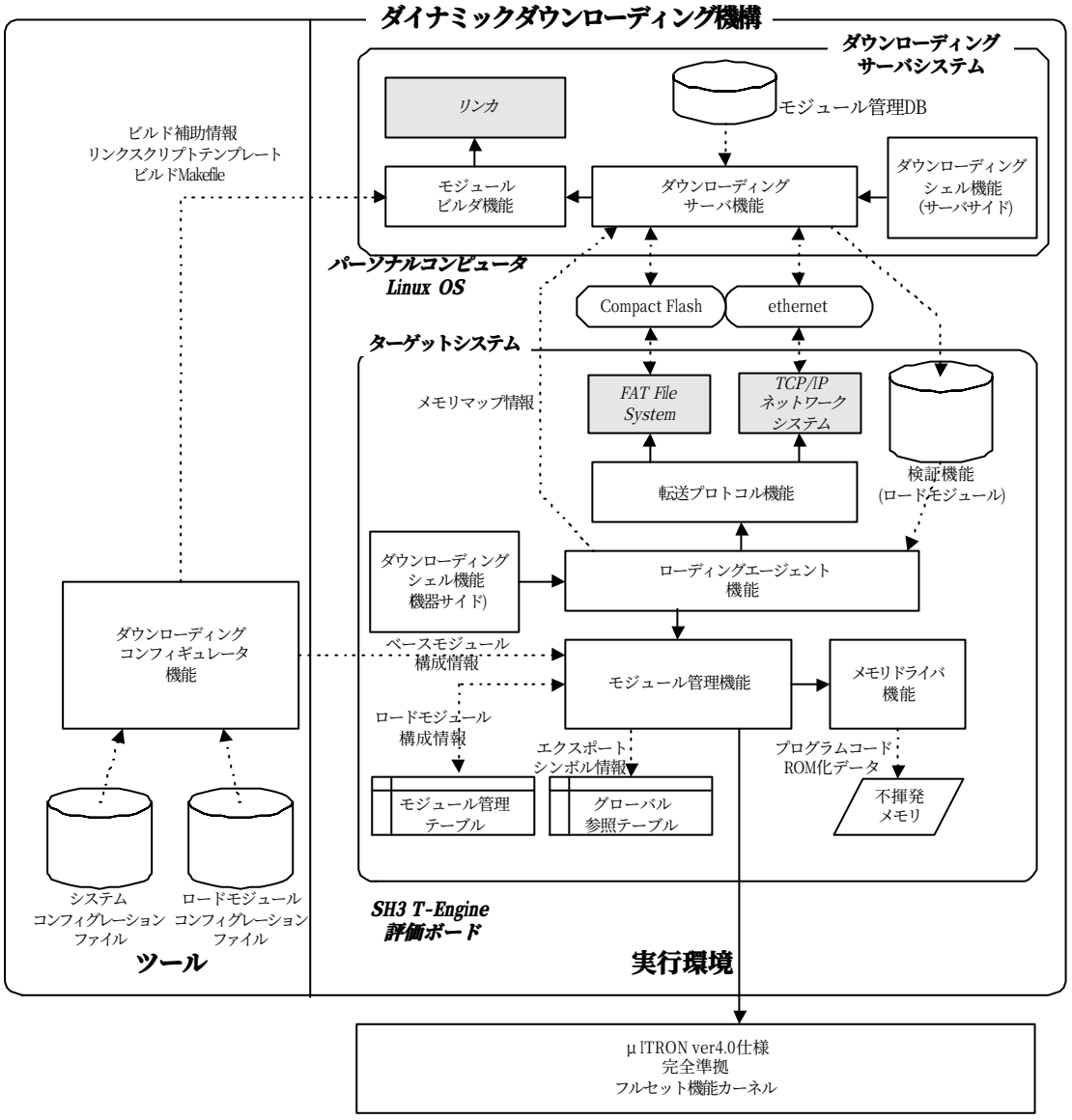



**図 1.3.1 ソフトウェア構成**

ダイナミックローディング機構は以下のソフトウェア部品で構成されます。

### ターゲットシステム サイド

ローダ機能はターゲット組み込み機器 μITRON フルセットカーネル上で動作し、μITRON カーネルにプログラムのダイナミックダウンローディング機能を追加するミドルウェアです。ローダ機能はローディングエージェント機能、モジュール管理機能、転送プロトコル機能で構成されます。また外部のソフトウェアとしてネットワーク機能とC 言語ランタイムライブラリ機能を必要とします。以降それぞれの機能ブロックの役割を説明します。

#### (1) ローディングエージェント機能

ローディングエージェントはダウンロードサーバとのセッションを管理し、トランザクション要求を処理します。

#### (2) モジュール管理機能

ダイナミックローディングされたロードモジュールの管理を行います。ローディングエージェントに対して、ロードモジュールの問合せ、削除、追加のサービスを提供します。

#### (3) 転送プロトコル機能

転送プロトコル機能はネットワークやストレージデバイス等の情報伝達手段を抽象化します。ローディングエージェント機能と情報媒体アクセス機能(TCP/IP)の間に立脚し、ダウンロードトランザクションサービスを提供します。情報媒体アクセス機能として外部プロダクト EBS 社製 TCP/IP SDK RTNET を移植し実現しています。

#### (4) メモリドライバ機能

メモリドライバ機能はフラッシュメモリ等の不揮発性メモリへの書き込み操作を抽象化するドライバ機能です。T-Engine ボードに搭載されるフラッシュメモリ(MBM29DL640)に対して消去、プログラムデータ書き込みを実現します。

#### (5) 検証機能

検証機能はダイナミックダウンローディング機構が実現する動的なプログラム更新機能をデモンストレーション実行するサンプルロードモジュールです。EBS 社製 WEB サーバ SDK を使用し、動的なホームページ閲覧サービス機能のプラグインデモンストレーションを行います。

(6) 外部プロダクト

以下の外部プロダクトを利用します。

・newlib --- C 言語ランタイムライブラリ

・RTNET --- EBS 社製 TCP/IP プロトコルスタック SDK

## ダウンロードサーバ サイド

ダウンロードサーバシステムは Linux OS で動作します。オペレータからのダウンロードデマンドを受付け、ターゲット機器に接続し、動的なプログラムのダウンロードを遂行します。ダウンロードサーバシステムはダウンローディングサーバ機能、モジュールビルダ機能、ダウンローディングシェル機能にて構成されます。また外部のソフトウェアとして RedHat Linux 9 を必要とします。以降それぞれの機能ブロックの役割を説明します。

### (1) ダウンローディングサーバ機能

ダウンローディングサーバ機能は Linux 上のプロセスとして動作し、ダウンローディングシェルプログラムからの処理要求に対してダウンローディングトランザクション処理を行うプログラムです。

### (2) モジュールビルダ機能

モジュールビルダ機能はダウンローディングサーバ機能の最終ロードモジュールを作成する処理を行います。

### (3) ダウンローディングシェル機能

ダイナミックダウンローディング機構におけるユーザインタフェース機能です。Linux 上のプロセスとして動作し、ダウンローディングサーバ機能とは INET ドメインのソケットで接続します。ダウンローディングトランザクションを発生させるコマンドラインインタフェースのユーザインタフェースを提供します。

(4) 外部プロダクト

・RedHat Linux 9 --- 動作 OS

### ツール

ツールシステムは Linux OS で動作し、ダイナミックダウンロード機構におけるロードモジュール作成をバックエンドで支援します。

#### (1) コンフィギュレータ機能

μITRON ver4.0 で定義される静的 API でのモジュールの構成編集を行った設定ファイル(コンフィギュレーションファイル)を解釈し、μITRON カーネル機能、モジュール管理機能へモジュールの初期化情報を出力するツール機能です。

#### (2) 外部プロダクト

以下の外部プロダクトを利用します。

- gnu tool chain --- クロスコンパイラツール
- RedHat Linux 9 --- 動作 OS

# 1.4 用語の説明

**(1) フルセットカーネル**

μITRON ver4.0 仕様で定義された拡張プロファイル機能を全て実装した μITRON 仕様カーネルプログラムの通称。

**(2) モジュール**

リンカによってアドレス解決され、出力されるファイルです。モジュールには動的に装着できないベースモジュールと、装着が可能なロードモジュールが存在します。

**(3) ベースモジュール**

ダイナッミクローディング機能を実現するための機能（フルセットカーネル、ダイナミックダウンローディング機構等の基幹機能）が実装されたリンクモジュール。ダイナッミクローディング機能ではこれらは従来通り工場出荷時機器に組み込まれている事を想定しています。ベースモジュールはダイナミックローディング機能では更新できません。

**(4) ロードモジュール**

動的に機器に組み込み可能なリンクモジュールです。ロードモジュールの属性は基幹属性とオプション属性のモジュールを作成する事が可能です。基幹属性ロードモジュールはシステムにおいて必要不可欠な機能が実装されることを想定していますので、単独でアンロードは出来ません。また、ダウンロード途中で機器の電源を切ってしまった場合も、新旧どちらかのロードモジュールが存在する事を保証しています。一方、オプション属性のロードモジュールはシステムにおいて必須ではない機能を想定しています。単独でアンロードは可能ですが、ダウンロードキャンセルで新旧とも失われてしまう可能性があります。この場合再ダウンロードが必要となります。

### (5) シンボルエクスポート

モジュールはリンク単位です。他のモジュールのシンボル（関数名、変数名）を直接参照することは出来ません。他のモジュールのシンボル参照を可能とする仕組がシンボルエクスポートです。モジュールの開発者ユーザはダイナミックダウンロード機能で拡張された EXPORT 静的 API で、内部のシンボルを外部モジュールに公開する事が可能です。EXPORT 静的 API はベースモジュールのコンフィグレーションとロードモジュールのコンフィグレーションファイルどちらにも記述可能です。ベースモジュールに含まれるカーネルのサービスコールはコンフィギュレータツールがベースモジュールをビルドする際、内部で自動エクスポート処理をしていますので、カーネルサービスコールについて明示的な EXPORT 記述は不要です。

### (6) コンフィグレータ

ユーザ定義した μITRON ver4.0 仕様 静的 API を解釈し、カーネル及びダイナミックローディング機能に構成情報を出力するツールです。コンフィグレータの機能としてベースモジュールをコンフィギュレーションする機能、ロードモジュールをコンフィグレーションする機能、エクスポートシンボルプロセッサ機能を有しています。コンフィグレータは Makefile の一連のモジュールビルド処理の１コマンド処理として記述します。

# 2 章 導入編

本章ではダイナミックダウンロード機能の導入方法を説明します。

## 2.1 準備

### 2.1.1 Linux のインストール

まずダウンローディングサーバ PC に Linux をインストールします。ダウンローディングサーバの実行を確認しているのは RedHat Linux9 配布セットです。ネットワークダウンロードした CD イメージ、購入パッケージ、書籍の付録 CD 等から Linux のインストールを実行して下さい。
CD イメージは http://www.redhat.co.jp からダウンロード可能です。
RedHat Linux の詳細なインストール方法は本書では説明を省略しております。Install 要件は以下の通りです。

**ダイナミックダウンロード(ネットワーク運用)を実行するための Linux システム要件**

- 1つ以上の Ethernet ネットワークインタフェースがインストールされている事
- TFTP サーバサービスがインストールされている事

以下に TFTP サーバ コンフィグレーション(/etc/xinet.d/tftp)例を示します。

**リスト2.1.1.1： tftp サーバ コンフィグレーションファイル**

```
service tftp
{
        socket_type     = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server          = /usr/sbin/in.tftpd
        server_args     = -s /tftpboot
        disable         = no
        per_source      = 11
        cps             = 100 2
        flags           = IPv4
}
```

**ダイナミックダウンロード(オフライン運用)を実行するための Linux システム要件**

- CF または PC ATA フラッシュカードを認識するドライバーが組み込まれていること
- USB カードリードライターを使用する場合、USB ホストドライバーとUSB マスストレージクラスドライバーが組み込まれている事

RedHat9 は通常のインストールを行うと、これらの機能は組み込まれています。
なおダウンロードサーバ オフライン動作の確認は以下の機器、記憶媒体を使用致しました。

USB CF カードリードライタ
　　I-O DATA 製 CF-RW
記憶媒体
　　レキサーメディア社 256M Byte CF

## 2.1.2 GNU ツール(SH3 用クロスコンパイル環境)の導入

### (1) ツールのダウンロード

ダイナミックダウンロード機構はクロスコンパイラとして以下の GNU ツールを使用します。
本節ではこのツールのインストール方法をご説明します。

**表 1 ダイナミックダウンローディングで必要なツール**

| ツール種類 | アーカイブファイル名 | バージョン番号 |
|---|---|---|
| コンパイラ | gcc-core-3.2.tar.gz | 3.2 |
| リンカ、アッセンブラ等 | binutils-2.13.2.1.tar.gz | 2.13.2.1 |
| ライブラリ | newlib-1.11.0.tar.gz | 1.11.0 |

上記ファイルは以下の WEB サイトから入手可能です。(2003 年 11 月現在)

**binutils, gcc-core:**

GNU プロジェクト http://www.gnu.org/
Ring Server http://www.ring.gr.jp/

**newlib:**

Red Hat http://sources.redhat.com/newlib/
または ftp://sources.redhat.com/pub/newlib/

上記表で示したアーカイブファイルをダウンロードし、tar コマンドを使用して解凍します。
解凍例：

**tar zxvf <アーカイブファイル名> [<出力ディレクトリ>]**

ここでは、/home/someone/gnu-src のディレクトリをアーカイブ解凍ディレクトリ、/home/someone/gnu-work を実行ファイルビルド作業ディレクトリと仮定し説明します。
/home/someone/gnu-src で解凍した場合、以下のソースディレクトリがそれぞれ作成されます。

./gnu-src/binutils-2.13.2.1
./gnu-src/gcc-3.2
./ gnu-src/newlib-1.11.0

次に実行ファイルビルド作業ディレクトリを作成します。

```
mkdir   /home/someone/gnu-work/binutils-2.13.2.1
mkdir   /home/someone/gnu-work/gcc-3.2
mkdir   /home/someone/gnu-work/newlib-1.11.0
```

実行ファイルビルド作業ディレクトリでは configure->make->install の順番で作業を行います。
ビルドの順番は binutils、gcc、newlib の順番で行います。

### (2) binutils ビルド作業

作業ディレクトリ.../gnu-work/binutils-2.13.2.1 で以下のコマンドを実行します。

```
$ ./../../gnu-src/binutils-2.13.2.1/configure --target=sh-hitachi-elf
   --prefix=/usr/local --disable-nls
$ make
$ make install
```

上記ではクロス環境ビルドターゲット名として sh-hitachi-elf を、インストール先ディレクトリとして /usr/local 以下を指定しています。make install の実行で/usr/local/bin にビルドした実行ファイルがコピーされます。

注 :make install を実行するには、Install ディレクトリ(/usr/local/以下)への書き込み権限が必要です。

### (3) gcc ビルド作業

作業ディレクトリ.../gnu-work/gcc-3.2 で以下のコマンドを実行します。

```
$ ../../gnu-src/gcc-3.3/configure --target=sh-hitachi-elf --prefix=/usr/local --with-gnu-as --
with-gnu-ld --with-newlib --with-headers=./../../gnu-src/newlib-1.11.0/newlib/libc/include
$ make
$ make install
```

注：configure 時、ローダ機能は newlib ライブラリ機能を一部使用するため、gcc の configure は

newlib との協調ビルドを指定しています。この指定では configure 時 Install 先ディレクトリに newlib のヘッダファイルがコピーされるため、Install 先ディレクトリ（上記では/usr/local 以下）への書き込み権限が必要です。
make install 実行でビルドした実行ファイルがインストールディレクトリにコピーされます。この時も Install 先ディレクトリへの書き込み権限が必要です。

### (4) newlib ビルド作業

作業ディレクトリ.../gnu-work/newlib-1.11.0 で以下のコマンドを実行します。

```
$ ../../gnu-src/newlib-1.11.0/configure --target=sh-hitachi-elf --prefix=/usr/local
$ make
$ make install
```

注：make install を実行するには、Install ディレクトリ(/usr/local/以下)への書き込み権限が必要です。

# 2.2 ダイナミックダウンロード ソフトウェアのインストール

以下にダイナミックダウンローディングの配布セットのディレクトリ構造を示します。

**リスト2.2.1： ダイナミックローディング機構　ディレクトリ階層**

```
/idl-opsw
    ¦-- server
    ¦    ¦-- api
    ¦    ¦-- idl_shell
    ¦    ¦-- idlsrv
    ¦    +- config.sample
    ¦
    +-- target
        ¦-- script.sample
        ¦-- network
        ¦-- idl_kernel
        ¦    ¦-- cfg
        ¦    ¦-- config
        ¦    ¦    +-- sh3
        ¦    ¦         +-- ms7727cp01
        ¦    ¦-- doc
        ¦    ¦-- include
        ¦    ¦-- kernel
        ¦    ¦-- library
        ¦    ¦-- pdic
        ¦    ¦-- systask
        ¦    +-- utils
        ¦-- loader
        +-- modules
             ¦-- base
             +-- load
                  +-- web
                       ¦-- ver2.00
                       +-- ver3.00
```

**表 2.2.1: ダイナミックローディングディレクトリ構成**

| ディレクトリー1 | ディレクトリー2 | 説明 |
|---|---|---|
| server | | サーバソフトウェアディレクトリ |
| | idl_shell | ダウンローディングシェルプログラム |
| | idlsrv | ダウンローディングサーバプログラム |
| | config.data | サンプルサーバコンフィグレーションデータ |
| target | | サーバソフトウェアディレクトリ |
| | idl_kernel | μITRON フルセットカーネル機能 |
| | loader | ダウンローディング機能 |
| | idl_shell | ダウンローディングシェル機能 |
| | network | ネットワーク機能(RTNET) |
| | filesystem | ファイルシステム機能(RTFILES) |
| | module | デモ用モジュール |
| | script.sample | サンプルスクリプト |

idl-opsw 以下を Linux 上の適当なディレクトリに展開します。ここでは/home/someone/idl-opsw として展開したと仮定し説明を行います。別途フルセットカーネルのディレクトリより config,include,kernel,library,systask,utils,pdic ディレクトリを idl_kernel ディレクトリにコピーしフルセセットカーネル機能とダイナミックローディング機能のソースをマージします。

**リスト2.2.2： フルセットカーネルディレクトリ階層**

```
/fi4
    |
    |-- config
    |       +-- sh3
    |             +-- ms7727cp01
    |-- include
    |-- kernel
    |-- library
    |-- pdic
    |-- systask
    +-- utils
```

## 2.3 コンフィギュレータのセットアップ

### 2.3.1 ビルド方法

ソースを展開した際に出来る、コンフィギュレータディレクトリ下 (/home/someone/idl-opsw/target/idl_kernel/cfg)にて make を実行します。これにより、実行ファイル idlcfg が作成されます。次に make install を実行します。make install と実行することで、実行ファイル idlcfg が /usr/local/bin にコピーされます。

注 :make install を実行するには/usr/local/bin への書き込み権限が必要です。

### 2.3.2 起動方法

コンフィグレータの起動は通常 Makefile の一連のビルド処理中に記述します。
idl-opsw/target/sample.script に格納している Makefile を参照してください。
以降にコンフィグレータの起動オプションの説明を行います。

(1) 起動書式

$ idlcfg <オプション>

(2) コンフィグレータ全体のオプション

-v, --verbose : 途中経過などを表示します
-nl, --nologo : 起動時のバナーを表示しません

(3) シンボルローダ

-mod, --module=モジュール名 : ローダブルモジュールからシンボルを抜きます
-nm, --nmfile : gnu-nm の出力ファイルを解析してシンボルを得る

(4) 静的 API パーサ

-s, --source=ファイル名 : 入力ファイル名を指定します
-obj, --dump-object=ファイル名 : 指定したファイルにオブジェクト情報を出力します
-ao, --assign-order=順序 : 自動 ID 割当の割当順序を指定します
割当順序は次の 3 つの組合せで指定します
alphabetic (ABC 順), fcfs (宣言順 [デフォルト]), reverse (逆順)

-opt, --option=モジュール名　　：指定したモジュールを構成する
-ep, --exec-prefix=プレフィクス　：CPP のプレフィクスを指定します

(5) EXPORT ファイル生成 (global_refs.h, global_id.h)
-fg, --force-globalrefs : 更新が無くてもexport_node ファイルを作成します

(6) ベースモジュールコンフィギュレーション
-cfg, --kernel-cfg=[ファイル名]　：カーネル構成ファイルの名前を指定します
-objdir, --object-directory=[場所]　：オブジェクトが格納されている場所を指定します

(7) ローダコンフィギュレーション
-ldrcfg, --loader-cfg=[ファイル名] : ローダ構成ファイルを生成します

(8) ローダブルモジュールコンフィギュレーション
-opt, --option=[モジュール名]　：対象となるモジュール名を設定します
-mod, --module=[ファイル名]　：モジュールを含むオブジェクトファイルを指定します
(パス2)

### 2.3.3 μITRON 仕様 静的 API 機能

ここではダイナミックダウンロードのコンフィグレータ機能が解釈できる静的 API を説明します。
静的 API は μITRON ver4.0 仕様で定義されているものと、ダウンロード用に拡張したものがコンフィグレーションファイルに記述可能です。仕様準拠静的 API としては以下の API を解釈し、カーネル機能及びローダ機能に構成情報を出力します。下記 API の仕様は μITRON ver4.0 仕様書を参照してください。

**表 2.3.2.1: μITRON 仕様 静的 API**

S:スタンダードプロファイル F:拡張プロファイル

| 機能分類 | API 名 | 定義クラス | 内容 | 呼出制約 | |
|---|---|---|---|---|---|
| | | | | ベース | ロード |
| タスク管理 | CRE_TSK | S | タスクの生成 | ○ | ○ |
| タスク例外処理機能 | DEF_TEX | S | タスク例外処理ルーチンの定義 | ○ | ○ |
| 同期通信機能 | CRE_SEM | S | セマフォの生成 | ○ | ○ |
| | CRE_FLG | S | イベントフラグの生成 | ○ | ○ |
| | CRE_DTQ | S | データキューの生成 | ○ | ○ |
| | CRE_MBX | S | メールボックスの生成 | ○ | ○ |
| 拡張同期通信機能 | CRE_MTX | F | ミューテックスの生成 | ○ | ○ |
| | CRE_MBF | F | メッセージバッファの生成 | ○ | ○ |
| | CRE_POR | F | ランデブポートの生成 | ○ | ○ |
| メモリプール管理機能 | CRE_MPF | S | 固定長メモリプールの生成 | ○ | ○ |
| | CRE_MPL | F | 可変帳メモリプールの生成 | ○ | ○ |
| 時間管理機能 | CRE_CYC | S | 周期ハンドラの生成 | ○ | ○ |
| | CRE_ALM | F | アラームハンドラの生成 | ○ | ○ |

| | | | | | |
|---|---|---|---|---|---|
| | DEF_OVR | F | オーバーランハンドラの定義 | ○ | ○ |
| 割込み管理機能 | DEF_INH | S | 割り込みハンドラの定義 | ○ | ○ |
| | ATT_ISR | F | 割り込みサービスルーチンの追加 | ○ | ○ |
| | DEF_EXC | S | CPU 例外ハンドラの定義 | ○ | ○ |
| サービスコール管理機能 | DEF_SVC | S | 拡張サービスコールの定義 | ○ | ○ |
| システム構成管理機能 | ATT_INI | S | 初期化ルーチンの追加 | ○ | × |
| ファイル包含機能 | INCLUDE | S | 外部ファイルのインクルード | ○ | ○ |

## 2.3.4 拡張した静的 API 機能

次にダウンロード機能で拡張した、μITRONVer4.0 仕様範囲外の静的 API を説明します。

**表 2.3.3.1: ダウンロード機能拡張 静的 API**

| 機能分類 | API 名 | 内容 | 呼出制約 | |
|---|---|---|---|---|
| | | | ベース | ロード |
| ベースモジュール | VATR_BSM | ベースモジュールの属性定義 | ○ | × |
| | VATT_LDM | 装着可能なロードモジュールの定義 | ○ | × |
| ダウンロードモジュール管理機能 | VATR_LDM | ロードモジュールの属性定義 | × | ○ |
| | EXPORT_INI | ロードモジュール初期化処理の定義 | × | ○ |
| | EXPORT_TER | ロードモジュール終了処理の定義 | × | ○ |
| エクスポート機能 | EXPORT | シンボルの EXPORT 宣言 | ○ | ○ |
| システム構成管理機能 | VAID_TSK | 動的生成可能なタスク最大数 | ○ | × |
| | VAID_SEM | 動的生成可能なセマフォ最大数 | ○ | × |
| | VAID_FLG | 動的生成可能なイベントフラグ最大数 | ○ | × |
| | VAID_DTQ | 動的生成可能なデータキュー最大数 | ○ | × |
| | VAID_MBX | 動的生成可能なメールボックス最大数 | ○ | × |
| | VAID_MPF | 動的生成可能な固定長メモリプール最大数 | ○ | × |
| | VAID_MPL | 動的生成可能な可変長メモリプール最大数 | ○ | × |
| | VAID_MBF | 動的生成可能なメッセージバッファ最大数 | ○ | × |
| | VAID_POR | 動的生成可能なランデブポート最大数 | ○ | × |
| | VAID_MTX | 動的生成可能なミューテックス最大数 | ○ | × |
| | VAID_CYC | 動的生成可能な周期ハンドラの最大数 | ○ | × |
| | VAID_ALM | 動的生成可能なアラームハンドラの最大数 | ○ | × |
| | VAID_ISR | 動的生成可能な割り込みサービスルーチンの最大数 | ○ | × |
| | VRID_TSK | タスク ID の予約 | ○ | × |
| | VRID_SEM | セマフォ ID の予約 | ○ | × |

| | | | | |
|---|---|---|---|---|
| | VRID_FLG | イベントフラグ ID の予約 | ○ | × |
| | VRID_DTQ | データキュー ID の予約 | ○ | × |
| | VRID_MBX | メールボックス ID の予約 | ○ | × |
| | VRID_MPF | 固定長メモリプール ID の予約 | ○ | × |
| | VRID_MPL | 可変長メモリプール ID の予約 | ○ | × |
| | VRID_MBF | メッセージバッファ ID の予約 | ○ | × |
| | VRID_POR | ランデブポート ID の予約 | ○ | × |
| | VRID_MTX | ミュテックス ID の予約 | ○ | × |
| | VRID_CYC | 周期ハンドラ ID の予約 | ○ | × |
| | VRID_ALM | アラームハンドラ ID の予約 | ○ | × |
| | VRID_ISR | 割り込みサービスルーチン ID の予約 | ○ | × |
| | VNUM_EXP | グローバルシンボルテーブル最大登録数 | ○ | × |
| | VNUM_LDM | ロードモジュールの装着最大数の定義 | ○ | × |
| | VDEF_KMB | カーネルヒープ領域サイズの定義 | ○ | × |
| | VATT_TER | システム終了ルーチンの追加 | ○ | × |

⑴ VATR_BSM ― ベースモジュール属性の設定

機能：
ベースモジュールのバージョン番号、属性の設定。

書式：
VATR_BSM( module_description, { attribute, version });

module_description：モジュール説明文字列(最大 32 文字)
attribute ： 属性(現状 TA_NULL のみ設定可能)
version ： モジュールのバージョン番号
バージョン番号書式 xx.yy
( xx= major version2 桁 yy= minor version 2 桁)
記述例：
VATR_BSM( "uITRON FullSpec. with loader", { TA_NULL, 01.00 });

⑵ VATR_LDM ― ロードモジュール属性の設定

機能：
ロードモジュールのバージョン番号、属性の設定。

書式：
VATR_LDM( Identifier, module_description, { attribute, version });
Identifier ：ロードモジュール識別子
VATT_LDM でベースモジュールコンフィギュレーションに追加した
ロードモジュールの識別子を指定する。
module_description：モジュール説明文字列(最大 32 文字)
attribute ： 属性
TYPE_STANDARD:基幹属性
TYPE_OPTION :オプション属性
version ： モジュールのバージョン番号
バージョン番号書式 xx.yy
( xx= major version2 桁 yy= minor version 2 桁)
記述例：

VATR_LDM( ldm_test, "Test Load Module", { TYPE_STANDARD, 01.00 } );

⑶）EXPORT ― シンボルのエクスポート宣言

機能：
モジュールの内部シンボルを外部モジュールに公開する。

書式：
EXPORT( "c-symbol-define");
c-symbol-define : C 言語における変数及び関数宣言

記述例：
EXPORT("int  func1( int )" );
EXPORT("int  val1" );
EXPORT("ID  mbxid1" );

⑷）EXPORT_INI ― ロードモジュール初期化処理の登録

機能：
ロードモジュール活性化時に呼び出される、ユーザ初期化ルーチンを登録する。
本定義はロードモジュールのコンフィグレーションファイルに必須である。

書式：
EXPORT_INI( initial_func_name );
initial_func_name： ロードモジュール初期化処理 関数名

記述例：
EXPORT_INI( init_module_func );

5）EXPORT_TER ― ロードモジュール終了処理の登録

機能：
ロードモジュール削除時に呼び出される、ユーザ初期化ルーチンを登録する。
本定義はロードモジュールのコンフィグレーションファイルに必須である。

書式：
EXPORT_TER( unload_func_name );
unload_func_name： ロードモジュール初期化処理 関数名

記述例：
EXPORT_TER( unload_module_func );

6）VATT_LDM ― ロードモジュールのアタッチ

機能：
ベースモジュールコンフィギュレーションにダウンロード可能なロードモジュールを追加する。

書式：

VATT_LDM(Identifier, "load-module-cfg-file" );
Identifier ： ロードモジュール識別子
load-module-cfg-file ： ロードモジュールコンフィグレーションファイル(フルパス指定)

記述例：
VATT_LDM(ldm_test,
"/home/opsw/load/ver1.00/ldm_test.cfg" );

(7) VAID_xxx ― カーネルオブジェクト動的生成 最大数の定義

機能：
自動 ID 採番で動的に生成されるカーネルオブジェクトの最大数を定義する。

書式：
VAID_xxx( max_number );
max_number : カーネルオブジェクトの最大数

記述例：
VAID_TSK( 100 );
VAID_SEM( 30 );
VAID_DTQ( 30 );

(8) VRID_xxx ― カーネルオブジェクトID 予約

機能：
カーネルオブジェクトの ID を予約する。
予約された ID は、静的 API もしくは動的なサービスコールにおいて ID を自動割り当てする際、使用ないことを保証する。

書式：
VRID_xxx( reserved_id );
reserved_id : 予約カーネルオブジェクトID

記述例：
VRID_TSK( 1 );
VRID_SEM( 10 );
VRID_DTQ( 5 );

(9) VNUM_EXP ― シンボルエクスポート最大数の定義

機能：
ロードモジュールがエクスポートするシンボルの最大数を定義する。

書式：
VNUM_EXP( max_export );
max_export:　エクスポートシンボル 登録最大数

記述例：
VNUM_EXP(50);

(10) VNUM_LDM ― ロードモジュールの装着最大数の定義

機能：
ベースモジュールに装着可能なロードモジュールの最大数を定義する。

書式：
VNUM_LDM( max_load_module );
max_load_module:　ロードモジュールの登録最大数

記述例：
VNUM_LDM(50);

(11) VDEF_KMB ― カーネルヒープ領域のサイズ指定

機能：
動的なカーネルオブジェクト生成で使用されるヒープ領域のサイズを指定する。

書式：
VDEF_KMB( heap_size );
heap_size:　カーネルヒープ領域バイトサイズ

記述例：
VNUM_LDM( 40960 );

(12) VATT_TER ― システム終了処理の定義

機能：
システム終了時に呼び出すユーザ定義の終了処理を定義する。

書式：
VATT_TER( terminate_routine );
terminate_routine:　システム終了処理関数名

記述例：
VATT_TER( terminate_system );

(13) コンフィグレータが自動処理する静的 API

下表はロードモジュールがエクスポートしたシンボルを管理するためにコンフィギュレータが内部的に使用する拡張 API です。(ユーザ（開発者）はこの API を使って記述することはありません)
新たなロードモジュールがベースモジュールコンフィグレーションファイルに登録されると、_EXPORT_SYMBOL_定義行がベースモジュールコンフィギュレーションファイル末尾に自動追加され、_EXPORT_NEXTID_定義行が更新されることがあります。

| API 名 | 定義内容 |
|---|---|
| **__EXPORT_NEXTID__** | **グローバル参照シンボル次割り当て ID（自動生成）** |
| **__EXPORT_SYMBOL__** | **グローバル参照シンボル割り当て定義（自動生成）** |

書式
_EXPORT_NEXTID_( next_index );
next_index ：グローバル参照テーブル 次登録インデクス

_EXPORT_SYMBOL_( symbol_name, index, "load_module_identifier");
symbol_name：エクスポートシンボル名
index：登録インデクス
load_module_identifier：ロードモジュール識別子

# 2.4 ダウンローディングサーバのセットアップ

## 2.4.1 ダウンローディングサーバビルド方法

ダイナミックダウンローディングソースを install したディレクトリで以下の作業を行い、ダウンロードサーバ実行ファイルの生成、インストールを行います。

① $ cd <install_dir>/idl-opsw/server/idlsrv ---- ソースディレクトへ移動
② $ make ---- 実行ファイル idlsrv の作成
③ $ make install ---- /usr/local/sbin へ実行ファイルをコピー

③の操作は/usr/local/sbin への書き込み権限が必要です。

## 2.4.2 起動方法

ダウンローディングサーバ (idlsrv) は Linux のプロセスとして動作します。コマンドラインからタイプする事で起動が可能ですが、Linux 起動時に実行されるシェルスクリプトに記述する事でデーモン動作も可能となります。ダウンローディングサーバは実行環境設定ファイル、モジュール管理 DB ファイルの読み出し権限、作業ディレクトリ(ロードモジュール作成ディレクトリ、ログ出力ディレクトリ)への書き込み権限が必要です。通常 root 権限で実行させます。
以降に起動書式、オプション説明をします。

起動書式:
$ idlsrv [options] ---- フォアグランドプロセスとして起動
$ idlsrv [options] & ---- バックグラウンドプロセスとして起動
options 説明
-D : デーモンとして動作します。
-debug : デバッグ用のオプションです。デバッグメッセージをコンソール出力します。
-logging : ダウンローディング実行ログを/var/log/IDL/idlsrv.log ファイルに出力します。

-C <ファイル名> : ダウンローディングサーバコンフィグレーションファイル名を指定する事で実行環境を切り替えます。
(default は/etc/IDL/2.0/idlsrv.conf を参照します。)

Linux 起動時にダウンローディングサーバをデーモンとして初期起動させるには、イニシャルシェルスクリプトの1つである、/etc/rc.d/rc.local に以下の 1行を追加します。

/usr/local/sbin/idlsrv -D

### 2.4.3 ダウンローディングサーバ設定ファイル

idlsrv.conf ファイルはダウンローディングサーバ(idlsrv)の動作を指定するパラメータファイルです。起動引数で格納ディレクトリを指定する事は可能ですが、指定されない場合は/etc/IDL/2.0/のディレクトリを参照します。本書ではデフォルトディレクトリに格納するものとして説明します。

① /etc/IDL/2.0 のディレクトリを作成する。

② ダウンローディング機構ソース配布セット /idl-opsw/server/config.sample/ より、idlsrv.conf.sample を/etc/IDL/2.0/idlsrv.conf にコピーする。

③ 動作環境に合わせてエディタで内容を修正する。

```
1  ##############################################################################
2  #   ダウンロードサーバコンフィギュレーションファイル                         #
3  #                                                                            #
4  ##############################################################################
5
6  # ダウンロードサーバ関連の設定
7  IDLSRV_IP            192.168.0.207  # サーバの IP アドレスを設定
8  IDLSRV_IDL_PORT      49512          # IDL サービスのポート番号を設定
9  IDLSRV_MAX_SESSION   30             # 最大同時セッション数を設定(最大 2147483647)
10 IDLSRV_TIMEOUT       3600           # タイムアウト時間の設定(秒)
11
12 # TFTP サーバ(ファイルサーバ)関連の設定
13 IDLSRV_TFTP_IP       192.168.0.207      # TFTP サーバ(ファイルサーバ)の IP アドレスを設定
14 IDLSRV_TFTP_PORT     69                 # TFTP サービスの UDP ポート番号を設定
15 IDLSRV_TFTP_PATH     /tftpboot          # TFTP サービスのアクセスディレクトリ
16                                  # TFTP サーバの設定(/etc/xinetd.d/tftp)と一致すること
17
18 # モジュールビルダ処理 関連の設定
19 IDLSRV_DATABASE_FILE    /etc/IDL/database.txt    # データベースファイル名
20 IDLSRV_AUXINFO_FILE     build.inf                # 補助情報ファイル名
21 IDLSRV_LINKER_OUTPUT    a.out                    # リンカアウトプットファイル名
22 IDLSRV_MODULE_BINARY    module.bin               # モジュールバイナリ抽出ファイル名
23 IDLSRV_MODULE_FILE      module.idl               # モジュールファイル名
24 IDLSRV_HEADER_SECTION   .module_header           # モジュールヘッダ情報抽出セクション名
25 IDLSRV_HEADER_BINARY    header.bin          # モジュールヘッダ情報抽出ファイル名
26 IDLSRV_HEADER_FILE      header.idl          # モジュールヘッダ情報ファイル名
27
28 #
29 # モジュールビルダ処理で起動する外部実行ファイル
30 IDLSRV_MAKE             /usr/bin/make
31 # マルチターゲットビルドされた objcopy
32 IDLSRV_OBJCOPY          /usr/local/bin/objcopy
33 # end of file
34 # オフラインモードで使用するデバイス情報
35 IDLSRV_MEDIA_FD        /dev/fd0                    # フロッピードライブ指定
36 IDLSRV_MEDIA_HDD       /dev/hda                    # ハードディスク指定
37 IDLSRV_MEDIA_USB       /dev/sda1                   # USB ストレージデバイス指定
38 IDLSRV_MOUNT_POINT     /mnt/IDL                    # デバイスマウントポイント
```

(1) ダウンロードサーバ関連の設定説明

ダウンロードサーバは TCP/IP ソケットにて、アプリケーションであるダウンロードシェルにサービスを提供します。以下はダウンロードサーバ関連の設定項目です。

| 設定項目 | 設定内容 |
|---|---|
| IDLSRV_IP | ダウンローディングサーバが動作するホスト PC の IP アドレスを設定します。<br>(例:192.168.0.207) |
| IDLSRV_IDL_PORT | ダウンローディングサーバがダウンローディングシェルにサービスを提供するポート番号を設定します。<br>(例:49512) |
| IDLSRV_MAX_SESSION | ダウンローディングサーバの最大セッション数(シェル接続数)を設定します。<br>(例:30) |
| IDLSRV_TIMEOUT | シェル無処理タイムアウト監視時間(単位 :秒)を設定します。<br>(例:3600) |

(2) TFTP サーバ関連の設定説明

ダウンロード機構はファイル転送サービスとして Linux TFTP ファイル転送サーバサービスを利用します。以下は Linux TFTP ファイル転送サーバサービスの設定項目です。

| 設定項目 | 設定内容 |
|---|---|
| IDLSRV_TFTP_IP | TFTP サーバの動作するホストPC の IP アドレスを設定します。<br>(例:192.168.0.207) |
| IDLSRV_TFTP_PORT | TFTP サーバがファイル転送サービスを提供するポート番号を設定します。<br>(例:69) |
| IDLSRV_TFTP_PATH | TFTP サービスのルート位置を設定します。<br>(例:/ tftpboot) |

(3) モジュールビルダ処理 関連の設定

| 設定項目 | 設定内容 |
|---|---|
| IDLSRV_DATABASE_FILE | モジュール管理データベースのファイル名情報をフルパスで設定します。<br>(例: /etc/IDL/database.txt) |
| IDLSRV_AUXINFO_FILE | コンフィギュレータが出力するビルド補助情報ファイル名を指定します。本システムでは build.inf と設定します。 |
| IDLSRV_LINKER_OUTPUT | リンカに出力させる、最終リンクオブジェクト名を指定します。(例:a.out) |
| IDLSRV_MODULE_FILE | ロードモジュールファイル名 最終ビルドファイル)を指定します。 |
| IDLSRV_HEADER_SECTION | ロードモジュールヘッダ情報のセクション名を指定します。.module_header と設定して下さい。 |
| IDLSRV_HEADER_BINARY | リンカ出力ファイルからロードモジュールヘッダ情報を切り出し出力する中間ファイル名を指定します。header.bin を特に修正する必要はありません。 |
| IDLSRV_HEADER_FILE | ロードモジュールヘッダ情報ファイル名 最終ビルドファイル)を指定します。header.idl を特に修正する必要はありません。 |

(4) モジュールビルダ処理で起動する外部実行ファイル

| 設定項目 | 設定内容 |
|---|---|
| IDLSRV_MAKE | gnu make 実行ファイルをフルパスで指定します。<br>(例: /usr/bin/make) |
| IDLSRV_OBJCOPY | gnu tool binutils に含まれる、objcopy 実行ファイルをフルパスで指定します。objcopy はダウンロードサーバがサポートする全てのターゲットに対して操作可能なマルチターゲットビルドされたものである必要があります。<br>(例: /usr/local/bin/objcopy) |

(5) オフラインモードで使用するデバイス情報

| 設定項目 | 設定内容 |
|---|---|
| IDLSRV_MEDIA_FD | フロッピディスクデバイス(例:/dev/fd0) |
| IDLSRV_MEDIA_HDD | ハードディスクデバイス(PC ATA FLASH カード等)<br>(例:/dev/hda) |
| IDLSRV_MEDIA_USB | USB ストレージデバイス(例:sda1) |
| IDLSRV_MOUNT_POINT | デバイスマウントディレクトリ |
| IDLSRV_MEDIA_DEFAULT | オフライン操作時のデフォルトメディア<br>0： フロッピーディスク<br>1： ハードディスク<br>2：USB ストレージデバイス |

## 2.4.4 サーバ作業ディレクトリの作成

ダウンローディングサーバを動作させる前に以下のディレクトリを手作業で作成しなくてはなりません。実行環境に合わせて以下のディレクトリを作成します。

| 内容 | 本書で対象としている<br>ディレクトリ名・ファイル名 | 特記 |
|---|---|---|
| ダウンローディングサーバ<br>実行環境設定ファイル<br>格納ディレクトリ | /etc/IDL/2.0<br>ファイル名:idlsrv.conf | 起動パラメータで変更可能 |
| モジュール管理データベース<br>格納ディレクトリ | /etc/IDL/2.0<br>ファイル名:database.txt | 実行環境設定ファイルで変更可能 |
| モジュールビルド作業ディレクトリ | /tftpboot/IDL | 実行環境設定ファイルで変更可能。Linux tftp サービスのディレクトリ階層を設定する事。 |
| ログ出力ディレクトリ | /var/log/IDL<br>ファイル名:idlsrv.log | 変更は出来ない |

## 2.4.5 モジュール管理データベースファイル

モジュール管理データベースはダウンロードサーバが接続するターゲットシステムの情報、作成したダウンロードモジュールの情報を登録するテキストファイルです。設定フォーマットはカンマ','を区切り文字とするCSV 形式となっています。(#はコメント記述行を表します。)
本節では、モジュール管理データベースに登録するデータ項目の説明を行います。

(1) ターゲットシステム登録情報

```
1   ##############################################################################
2   #                                                                            #
3   #   database.txt ( ロードモジュール管理データベースファイル)                 #
4   #   2003.09.10   仕様バージョン OPSW-0.1 AIC M.Harada                        #
5   #   2003.09.25   仕様バージョン OPSW-0.2 AIC Y.Nakamura                      #
6   #                                                                            #
7   ##############################################################################
8
9   #
10  # ダウンロードターゲット情報の定義
11  # ターゲット ID
12  # 1: PC Simulator IDL System(uITRON4/PX)
13  # 2: ARM Integrator IDL Demo System(uITRON4/PX)
14  # 3: T-Engine Board Downloding Demo System(uITRON4 Full Spec) ※
15  # エンディアンタイプ
16  # 1:リトルエンディアン
17  # 2:ビッグエンディアン
18  # 書式：
19  # T,<ターゲット ID>,<エンディアンタイプ>,
20  #   <ROM-START>:<ROM-END>,
21  #   <RAM-START>:<RAM-END>,
22  #   <LDMROM-START>:<LDMROM-END>,
23  #   <ROUND-SIZE>,
24  #   <ADDRESS-MASK>,
25  #   <FlashInfoNum>,<Flash-START>:<Flash-END>,<Erase-Block-Size>
26  #
27
    T,3,1,0x00000000:0x007fffff,0x0c000000:0x0dffffff,0x00100000:0x007effff,0x000010
00,0xe0000000,1,0x00000000:0x007fffff,0x00010000
28
29
```

**登録行書式：**
T,<TargetID>,<EndianType>,<ROM-Area-Info>,<RAM-Area-Info>,<LDMROM-Area-Info>,<ROUND-SIZE>,<ADDRESS-MASK>,<FlashBankNum>,<Flash-Bank-Info>

| 設定項目 | 設定内容 |
|---|---|
| TargetID | ターゲットシステムを一意に識別する任意の ID 番号を設定します。 |
| EndianType | ターゲットシステムのエンディアンタイプ (バイト順)を設定します。<br>設定値　1:リトル 2:ビッグ |
| ROM-Area-Info | ROM 全体のアドレス範囲を<開始アドレス>:<終了アドレス>の書式で設定します。 |
| RAM-Area-Info | RAM 全体のアドレス範囲を<開始アドレス>:<終了アドレス>の書式で設定します。 |
| LDROM-Area-Info | ROM-Area-Info の中でロードモジュールを割付可能なアドレス範囲を<開始アドレス>:<終了アドレス>の書式で設定します。 |
| ROUND_SIZE | ロードモジュールの領域を割り付ける際の、アドレス境界整合値を指定します。 |
| ADDRESS_MASK | CPU によっては、論理的なアドレス空間を識別する情報を論理アドレスに持つものが存在します。ここでは論理アドレスを物理アドレス情報へ変換するためのマスク情報を設定します。論理アドレス＝物理アドレスとなる CPU ではオール 0 を設定します。<br>SH3 設定例 : 0xe0000000<br>M68K 設定例: 0x0 |
| FlashBankNum | LDROM-Area-Info 中の Flash 消去バンク定義情報の情報数(Flash-Bank-Info 数)を設定します。 |
| Flash-Bank-Info | Flash 消去バンク定義情報を<開始アドレス>:<終了アドレス>,<消去ブロックサイズ>の書式で設定します。 |

(2) ロードモジュール登録情報

```
30   #
31   # ロードモジュールの定義
32   # 定義行書式:
33   # M,<module-id>,<target-id>,<base-version>,<module-
      ver>,<attr>,<description>,<makefile>,<base-dir>,<module-dir>
34   #
35   # 定義項目説明:
36   # M             : モジュール定義行キーワードタグ
37   # module-id     : モジュールID
38   # device-id     :  使用可能機器ID(=1),
39   # base-version  : ロード可能ベースモジュールバージョン範囲(編集禁止)
40   # module-ver    : モジュールバージョン
41   # attr          : モジュール属性(1=optional,2=standard)
42   # description   : モジュール説明
43   # makefile      : ロードモジュールMakefile Template
44   # base-dir      : システムコンフィグレーション ディレクトリ
45   # module-dir    : モジュールコンフィグレーション ディレクトリ
46
64   # WEBロードモジュール(standard module)
65   M,17,3,0x00000200,0x200,1,IPA web server (OPTION),/home/opsw/idl-
opsw/target/modules/load/IPA_insp/test-15/web/ver2.00/build.mak,/home/opsw/idl-
opsw/target/modules/base,/home/opsw/idl-opsw/target/modules/load/IPA_insp/test-
15/web/ver2.00
66   M,17,3,0x00000200,0x300,1,IPA web server (OPTION),/home/opsw/idl-
opsw/target/modules/load/IPA_insp/test-15/web/ver3.00/build.mak,/home/opsw/idl-
opsw/target/modules/base,/home/opsw/idl-opsw/target/modules/load/IPA_insp/test-
15/web/ver3.00
```

**登録行書式：**
M,<module-id>,<target-id>,<base-version>,<module-ver>,<attr>,<description>,<makefile>,<base-dir>,<module-dir>

| 設定項目 | 設定内容 |
|---|---|
| module-id | コンフィギュレータが割り付けたロードモジュール ID 番号を設定します。 |
| target-id | 登録するロードモジュールをダウンロード可能なシステムをターゲットシステム登録情報で設定したターゲット ID 値で指定します。 |
| base-version | 登録するロードモジュールが動作可能なベースモジュールバージョン番号範囲を以下の 16 進 8 桁の数値書式で設定します。<br>xxxxyyyy：xxxx 開始バージョン　yyyy 終了バージョン |
| module-ver | 登録するロードモジュールのバージョン番号を設定します。<br>本バージョン番号はロードモジュールコンフィギュレーションファイル VATR_LDM で設定したバージョン番号と一致させる必要があります。 |
| LDROM-Area-Info | ROM-Area-Info の中でロードモジュールを割付可能なアドレス範囲を<開始アドレス>:<終了アドレス>の書式で設定します。 |
| ROUND_SIZE | ロードモジュールの領域を割り付ける際の、アドレス境界整合値を指定します。 |
| ADDRESS_MASK | CPU によっては、論理的なアドレス空間を識別する情報を論理アドレスに持つものが存在します。ここでは論理アドレスを物理アドレス情報へ変換するためのマスク情報を設定します。論理アドレス＝物理アドレスとなる CPU ではオール 0 を設定します。<br>SH3 設定例：0xe0000000 M68K 設定例: 0x0 |
| FlashBankNum | LDROM-Area-Info 中の Flash 消去バンク定義情報の情報数(Flash-Bank-Info 数)を設定します。 |
| Flash-Bank-Info | Flash 消去バンク定義情報を<開始アドレス>:<終了アドレス>,<消去ブロックサイズ>の書式で設定します。 |

(3) ターゲット機器 ネットワークノード登録情報

```
71  #  ターゲット機器 ネットワークノード登録情報
72  #  ターゲット機器のネットワーク接続情報を登録する。
73  #   書式：
74  #    N,<Node-name>,<TargetID>,<IP-Address>,<Port-Number>
75
76  N,TE1,3,192.168.0.33,49512
77  N,TE4,3,192.168.0.250,49512
78  N,TE5,3,192.168.0.251,49512
79
```

```
登録 行書式：
N,<Node-name>,<TargetID>,<IP-Address>,<Port-Number>
```

| 設定項目 | 設定内容 |
|---|---|
| Node-name | ターゲットノードの識別名称を記述します。 |
| TargetID | ノードのシステム種別をターゲットID で設定します。 |
| IP-Address | ターゲットノードの IP アドレス情報を設定します。 |
| Port-Number | ターゲットで動作するローダ機能のポート番号を指定します。 |

(4) ターゲット機器 ネットワークノードグループ登録情報

```
80   # ターゲット機器 グループ登録情報
81   #     同じシステムタイプを持ったターゲット機器の総称を登録する・
82   #     書式：
83   #     G,<group-name>,<node-name1>,<node-name2>,……
84   G,DL-DemoSystem,TE1,TE2
85   G,DL-DemoSystem2,TE2
86   G,DL-DemoSystem3,TE1
87   G,DL-DemoSystem4,TE1
```

```
登録 行書式：
G,<group-name>,<node-name1>,<node-name2>,……
```

| 設定項目 | 設定内容 |
|---|---|
| group-name | グループ識別名称を記述します。 |
| node-name | グループに所属するノード名リストを記述します。 |

# 2.5 サーバダウンロードシェル

## 2.5.1 ダウンロードシェルプログラムのビルド方法

ダイナミックダウンローディングソースを install したディレクトリで以下の作業を行い、ダウンロードシェル実行ファイルの生成、インストールを行います。

① $ cd <install_dir>/idl-opsw/server/idl_shell ---- ソースディレクトへ移動
② $ make ---- 実行ファイル idl_shell の作成
③ $ make install ---- /usr/local/bin へ実行ファイルをコピー

③の操作は/usr/local/bin への書き込み権限が必要です。

## 2.5.2 起動方法

**$idl_shell [起動パラメータ]**

シュルプログラム起動時にダウンローディングサーバに接続します。起動パラメータは下表の通りです。-sオプションを省略した場合はローカルホスト指定になります。（ポートは 49512 固定）ダウンローディングサーバ接続後は、プロンプト（"> "）が表示されてコマンド入力待ちになりますので、コマンド名と必要なパラメータを入力し、「Enter」キーを押下すればそのコマンドが実行されます。（コマンド名は大文字で指定することも出来ます）

**>[コマンド] [パラメータ]**

| | | | |
|---|---|---|---|
| 起動パラメータ | -j | メッセージの日本語表示 | |
| | -h | ヘルプ（コマンド一覧を表示） | |
| | -s[IP:PORT] | ダウンローディングサーバへのアクセスポイント | |
| 使用例 | **$ ./idl_shell -s localhost:49512 -j**<br>**$ ./idl_shell -s192.168.0.205:49512**<br>**$ ./idl_shell -j -h** | | |
| メッセージ<br>接続成功： | Connect Server IP:127.0.0.1 port:49512 | | |
| 接続失敗： | Server connect error | サーバー接続エラー | |
| 備考 | 接続に失敗した場合はシュルプログラムは終了します。 | | |

### 2.5.3 シェルコマンド一覧

| コマンド名 | 省略形 | コマンド説明 |
|---|---|---|
| connect | con | ターゲットへ接続 |
| mdlist | md | サーバに登録されているロードモジュールの一覧表示 |
| status | sta | ターゲットに実装されているロードモジュールの一覧表示 |
| unload | uld | ターゲットに実装されているロードモジュールの削除 |
| load | ld | ターゲットにモジュールをロードする |
| target | ta | 接続中のターゲット一覧表示 |
| group | gr | データベースに登録されているグループの一覧表示 |
| member | mem | グループに所属しているターゲットの一覧表示 |
| tglist | tg | 接続可能ターゲットの一覧表示 |
| disconnect | disc | ターゲットの切断 |
| reload | rel | ターゲットにモジュールを再ロードする |
| help | h | コマンドの使用法の説明 |
| quit | q | シェルプログラム終了 |
| build | b | オフライン運用時のモジュールビルド指定コマンド |

### 2.5.4 シェルコマンドリファレンス

## 1. connect :ターゲットの接続

ダウンローディングサーバに対して指定したターゲットへの接続を指示します。接続先としてグループ名を指定することで、そのグループに所属しているターゲット全てに対して一括接続を行います。グループ名を指定する場合は－ gオプションを指定してください。

| | | |
|---|---|---|
| コマンド名（省略形） | Connect（con） | |
| コマンド書式 | Connect [パラメータ] | |
| パラメータ | モジュール管理 DB に登録したターゲットノード名またはグループ名を指定します。ターゲットの IPアドレスとポート番号を直接指定することも可能です。 | |
| 使用例：IP指定<br>ターゲット名指定<br>グループ名指定 | >connect TE5<br>>connect -gDemoSystem<br>>connect 192.168.0.205:9876 | |
| 正常時メッセージ | Connect Target name [ T-Engine ] IP [ 192.168.100.2:10002 ] | |
| エラーメッセージ<br>（起動オプションで-jが指定された時は日本語表示） | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Group member list error<br>グループ所属ターゲット取得エラー | 指定グループ名がデータベースに登録されていない |
| | No Target List<br>ターゲットリストなし | 指定グループのターゲットリストが見つからなかった。 |
| | Unknown target<br>ターゲット名不正 | 指定ターゲット名がデータベースに登録されていない |
| | Already connect target<br>接続済みターゲット | 指定したターゲット名または、IPのターゲットと既に接続されている。 |
| 備考 | 接続が成功した場合、プロンプトが “> “ → “con> “ になる。 | |

## 2 mdlist :モジュールリストの取得

モジュール管理 DB に登録されているロードモジュールのリストを取得し、一覧表示します。

| | | |
|---|---|---|
| コマンド名（省略形） | mdlist（md） | |
| コマンド書式 | Mdlist | |
| パラメータ | なし | |
| 使用例 | >mdlist | |
| 正常時表示例 | [ Module list ]<br>Module ID:0 version:1.00 attribute:optional [ test modul1 ]<br>Module ID:1 version:2.00 attribute:optional [ test modul2 ]<br>Module ID:2 version:1.00 attribute:standard [ ]<br>Module ID:2 version:1.10 attribute:standard [ ]<br>Module ID:2 version:1.20 attribute:standard [ test ] | |
| エラーメッセージ（起動オプションで -j が指定された時は日本語表示） | Module list error<br>モジュールリスト取得エラー | モジュールリスト取得が失敗した |
| | Server fatal error<br>サーバ障害 | サーバ内で致命的な障害発生（サーバを直ちに切断する） |
| 備考 | 正常時に表示されるモジュール内容は以下の通りです。<br>モジュール ID モジュールバージョン 属性 モジュール名 | |

## 3. status :モジュール状態問い合わせ

ターゲットに装着されているロードモジュールのうち、指定したモジュール IDのロードモジュール情報を取得し、表示します。モジュール IDを省略した場合、または、ワイルドカード(*)を指定した場合は装着されている全てのロードモジュール情報を一覧表示します。
このコマンドは接続されている全てのターゲットに対して、ロードモージュールの状態を問い合わせします。

| | | |
|---|---|---|
| コマンド名（省略形） | status（sta） | |
| コマンド書式 | Status [パラメータ] | |
| パラメータ | モジュール ID<br>ワイルドカード（"* "、"all "） | |
| 使用例<br>ワイルドカード指定 | >status 1<br>>status * | |
| 正常時表示例 | [ Module status ]<br>1: Target Name [ T-Engine ] IP [192.168.0.10:4000]<br>Module ID:1 version:1.02 attribute:optional [ Telephone ]<br>Module ID:2 version:2.00 attribute:standard [ Mobile ]<br>Module ID:3 version:5.00 attribute:optional [ driver ]<br>2: Target Name [ ARM9 ] IP [192.168.0.20:5000]<br>Module ID:0 version:1.02 attribute:optional [ Telephone ]<br>Module ID:1 version:3.02 attribute:basic [ LAN ] | |
| エラーメッセージ<br>（起動オプションで-j が指定された時は日本語表示） | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Group member list error<br>グループ所属ターゲット取得エラー | 指定グループ名がデータベースに登録されていない |
| | No Target List<br>ターゲットリストなし | 指定グループのターゲットリストが見つからなかった。 |
| | Unknown target<br>ターゲット名不正 | 指定ターゲット名がデータベースに登録されていない |
| | Already connect target<br>接続済みターゲット | 指定したターゲット名または、IP のターゲットと既に接続されている。 |
| 備考 | 正常時に表示されるモジュール内容は以下の通りです。<br>モジュール ID モジュールバージョン 属性 モジュール名 | |

## 4. unload :モジュールの削除

ターゲットシステムに装着されているロードモジュールのうち、指定したモジュール IDのロードモジュール削除(unload)します。このコマンドは接続されている全てのターゲットに対して、指定 IDのロードモジュールを削除します。

| | | |
|---|---|---|
| コマンド名（省略形） | unload（uld） | |
| コマンド書式 | Unload | |
| パラメータ | モジュール ID | |
| 使用例 | >unload 1 | |
| 正常時表示例 | [ Module unload ]<br>1: Target Name [ T-Engine ] IP [192.168.0.205:49512]<br>Delete complete | |
| 正常時メッセージ | Delete complete<br>モジュール削除成功 | 指定した IDのモジュールを削除した。 |
| エラーメッセージ<br>（起動オプションで-j が指定された時は日本語表示） | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Unconnected target<br>ターゲット未接続 | ターゲットが接続されていない |
| | Target error<br>ターゲットエラー | ターゲットが応答しない（タイムアウト） |
| | Unload error<br>アンロードエラー | 削除失敗 |
| 備考 | 本コマンドを実行する前に、ターゲットと接続しておく必要があります。 | |

## 5. load :モジュール追加、更新

指定した IDとバージョンのロードモジュールを、ターゲットシステムのメモリマップに合わせて再ビルド (リンク) し、ターゲットシステムにロードします。ロード中は ctrl＋c によるキャンセルが可能です。このコマンドは接続されている全てのターゲットに対して、指定 IDのロードモジュールをロードします。

| | | |
|---|---|---|
| コマンド名 (省略形) | load (ld) | |
| コマンド書式 | load [パラメータ1] [パラメータ2] | |
| パラメータ1<br>パラメータ2 | モジュール ID<br>モジュールバージョン | |
| 使用例 | >load 3 1.23<br>>load 2 1 | |
| 正常時表示例 | [ Module loading ]<br>1: Target Name [ T-Engine ] IP [192.168.0.205:49512]<br>Please wait a moment to load...<br>Load complete<br>(18) | |
| 正常時メッセージ | Load complete<br>モジュールローディング | モジュールロード成功 |
| エラーメッセージ<br>(起動オプションで-j が指定された時は日本語表示) | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Unconnected target<br>ターゲット未接続 | ターゲットが接続されていない |
| | Target error<br>ターゲットエラー | ターゲットが応答しない (タイムアウト) |
| | Loading failure<br>ローディングエラー | モジュールロード失敗 |
| | Loading cancel<br>ローディングキャンセル | モジュールロードを ctrl＋c で中断した |
| | Build error<br>ビルドエラー | モジュールのビルドが失敗した |
| 備考 | 本コマンドを実行する前に、ターゲットと接続しておく必要があります。 | |

## 6. reload :モジュールの再ロード (デバッグ支援コマンド)

指定したセッションIDのターゲットシステムへロードしたビルド済みロードモジュールを、現在接続されている全てのターゲットシステムへ再ロードします。

**注:**

**本コマンドはロードモジュールデバッグする際、再ビルドを行いたくないケースを支援するコマンドです。ターゲットに再ダウンロードして正常に動作するかはオペレータ責任となります。実運用時の使用は避けて下さい。**

| | | |
|---|---|---|
| コマンド名 (省略形) | reload (rld) | |
| コマンド書式 | reload [パラメータ1] [パラメータ2] [パラメータ3] | |
| パラメータ1<br>パラメータ2<br>パラメータ3 | セッション ID<br>モジュール ID<br>モジュールバージョン | |
| 使用例 | >reload 100 3 1.23<br>>reload 100 2 1 | |
| 正常時表示例 | [ Module reloading ]<br>1: Target Name [ T-Engine ] IP [192.168.0.205:49512]<br>Please wait a moment to load...<br>Load complete<br>(18) | |
| 正常時メッセージ | Load complete<br>モジュールローディング | モジュールロード成功 |
| エラーメッセージ<br>(起動オプションで-j が指定された時は日本語表示) | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Unconnected target<br>ターゲット未接続 | ターゲットが接続されていない |
| | Target error<br>ターゲットエラー | ターゲットが応答しない (タイムアウト) |
| | Loading failure<br>ローディングエラー | モジュールロード失敗 |
| | Loading cancel<br>ローディングキャンセル | モジュールロードを ctrl+c で中断した |
| 備考 | 本コマンドを実行する前に、ターゲットを接続しておく必要があります。 | |

## 7. target :接続中のターゲットを表示

現在接続されているターゲット(IPアドレス)の一覧を表示します。

| | |
|---|---|
| コマンド名 (省略形) | target (ta) |
| コマンド書式 | Target |
| パラメータ | なし |
| 使用例 | >target |
| 正常時表示例 | [ Connect Target list ]<br>1: Target Name [ T-Engine ] IP [192.168.0.10:4000]<br>2: Target Name [ ARM9 Board ] IP [192.168.0.20:5000] |
| 出力メッセージ | なし |
| 備考 | 本コマンドはサーバへの問い合わせは行いません。 |

## 8. tglist :接続可能ターゲット表示

ローディングサーバ内のデータベースから接続可能なターゲットシステムのリストを取得し、一覧を示します。

| | |
|---|---|
| コマンド名 (省略形) | tglist (tg) |
| コマンド書式 | Tglist |
| パラメータ | なし |
| 使用例 | >tglist |
| 正常時表示例 | [ Target list ]<br>1: Target Name [ T-Engine ] IP [192.168.0.10:10001]<br>GROUP DemoSystem1<br>GROUP DemoSystem2<br>2: Target Name [ ARM9 Board ] IP [192.168.0.20:10002]<br>3: Target Name [ SH3 ] IP [192.168.0.30:10003]<br>GROUP DemoSystem1 |
| 出力メッセージ | なし |
| 備考 | |

## 9. group :グループのリスト表示

ローディングサーバ内のデータベースに登録されているグループのリストを取得し、一覧を表示します。

| | |
|---|---|
| コマンド名（省略形） | group（gr） |
| コマンド書式 | Group |
| パラメータ | なし |
| 使用例 | >group |
| 正常時表示例 | [ Group list ]<br>1: DemoSystem1<br>2: DemoSystem2<br>3: DemoSystem3 |
| 出力メッセージ | なし |
| 備考 | |

## 10. member :グループに所属するターゲットのリスト表示

指定したグループがローディングサーバ内のデータベースに登録されているか検索し、登録されていた場合は、そのグループに所属しているターゲット（IPアドレス）のリストを取得し、一覧表を示します。

| | | |
|---|---|---|
| コマンド名（省略形） | member（mem） | |
| コマンド書式 | member [パラメータ] | |
| パラメータ | グループ名 | |
| 使用例 | >mem DemoSystem | |
| 正常時表示例 | [ Group member list ]<br>1: Target Name [ T-Engine ] IP [ 192.168.0.10:6000 ] ID:0<br>2: Target Name [ ARM9 Board ] IP [ 192.168.0.20:7000 ] ID:0<br>3: Target Name [ SH3 ] IP [ 192.168.0.30:6000 ] ID:1<br>4: Target Name [ ARM7 Board ] IP [ 192.168.0.40:7000 ] ID:1 | |
| エラーメッセージ<br>（起動オプションで-jが指定された時は日本語表示） | Parameter error<br>パラメータが不正 | パラメータが指定されていない |
| | Group member list error<br>グループ所属ターゲット取得エラー | ターゲットリスト取得失敗 |
| 備考 | | |

## 11. disconnect :ターゲットの切断

指定したターゲット名または、IPのターゲットを切断します。またグループ名を指定することで、そのグループに所属しているターゲット全てに対して一括切断を行います。グループ名を指定する場合は－ gオプションを指定してください。

| | |
|---|---|
| コマンド名（省略形） | disconnect（con） |
| コマンド書式 | disconnect [パラメータ] |
| パラメータ | ターゲットの IPアドレスとポート番号<br>ターゲット名<br>グループ名 |
| 使用例：IP指定<br>ターゲット名指定<br>グループ名指定<br>接続中の全ターゲット指定<br>（省略可） | >disconnect 192.168.0.205:9876<br>>disconnect TE5<br>>disconnect ?gDemoSystem<br>>disconnect *<br>>disc |
| 出力メッセージ | Disconnect Target |
| 備考 | 全てのターゲットを切断した場合は、プロンプトが “con>“ → “>“ になります。 |

## 12. quit :サーバーの切断（終了）

シェルプログラムを終了します。

| | |
|---|---|
| コマンド名（省略形） | quit（q） |
| パラメータ | なし |
| 出力メッセージ | なし |
| 使用例 | >q |

## 13. build :オフライン運用時のモジュールビルド指定

ビルド要求パケットファイル（BUILDREQ.LDM）が存在する、ダウンローディングサーバのデフォルトデバイス（FD、HDD、USBストレージデバイス）を指定し、ビルド要求パケットファイルに記述されているモジュール IDとモジュールバージョンのロードモジュールをビルドします。ビルド実行後、ビルド応答（結果）を応答パケットファイル（BUILDRES.LDM）に出力し、生成したロードモジュールを MDLHEAD.LDM（モジュールヘッダパケット）、MDLBODY.LDM（モジュール本体パケット）ファイルとして、メディアに書き込みます。

| | | |
|---|---|---|
| コマンド名（省略形） | build（b） | |
| コマンド書式 | build [パラメータ] | |
| パラメータ | パケットファイル格納メディア<br>“fd”、“hdd”、“usb” | |
| 使用例 | >build fd | |
| 正常時表示例 | [ Offline Build ]<br>Module complete | |
| エラーメッセージ<br>（起動オプションで-j が指定された時は日本語表示） | Media error<br>マウントエラー（不正メディア） | 指定メディアのマウントが出来ない |
| | File create error<br>ファイル作成失敗 | パケットファイルの作成が出来ない |
| | File open error<br>ファイルオープンエラー | 要求パケットが存在しない、または<br>ファイルリードエラーが発生 |
| | Invalid build request file<br>不正なビルド要求 | 要求パケットの書式が不正 |
| 備考 | build コマンドはオフライン運用専用のコマンドです。 | |

# 2.6 ターゲットダウンロードシェル

## 2.6.1 起動方法

ネットワーククライアントデマンド運用またはオフライン運用でダウンロード機能を実行するには機器側にダウンロードシェル機能が必要となります。ベースモジュールにクライアントデマンドもしくはオフライン運用を選択しビルドした場合、以下のようにダウンロードシェル機能が自動起動されます。

```
FI4 Kernel Release 1.4 (patchlevel = 0) for MS7727CP01(SH7727 T-Engine) (Feb 18 2004, 16:27:21)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
                                Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2003- by Monami software, LP


ITRON with Dynamic Loading System(network operation(client demand)) (Feb 18 2004, 16:26:46)
Copyright (C) 2002,2003 by AI Corporation, Inc., JAPAN
Copyright (C) 2002,2003 by Information-technology Promotion Agency, JAPAN
Base-Module Description[uITRON FullSpec. with loader] Version[00.01]

**** TOPPERS/IDL TARGET SHELL ****
Version: $Revision: 1.14 $
>
```

## 2.6.2 シェルコマンドリファレンス

### 1). ダウンロードサーバーの接続

| | |
|---|---|
| コマンド説明 | ローディングサーバに接続を行う。 |
| コマンド名<br>省略形 | connect<br>con |
| パラメータ | ローディングサーバの IPアドレスとポート番号<br>パラメータを省略した場合はローカルサーバへの接続を行う |
| 書式 | connect△[ IP:ポート] |
| 使用例 | >connect 192.168.0.205:9876 |
| オフライン運用時 | 無効 |
| 備考 | 接続後のプロンプト表示を変える<br>［> ］→ ［con> ］<br>**本コマンドは、オフライン運用では使用しない。**<br>**（ターゲットの起動後、サーバと接 続している状態になる）** |

■メッセージ出力

エラー：

「接続中エラー」

指定した IPのサーバと既に接続されている

「文法エラー」

指定した IP のサーバが存在しない

「サーバ接続エラー」

サーバとの接続に失敗した

接続成功：

「Connect Server IP ：xx.xx.xx.xx:xx」

## 2) モジュールリストの取得

| | |
|---|---|
| コマンド説明 | ローディングサーバに登録されているロードモジュールのリストを取得し、一覧表示する。 |
| コマンド名<br>省略形 | mdlist<br>md |
| パラメータ | なし |
| 使用例 | con>mdlist |
| オフライン運用時 | 無効 |

■メッセージ出力

エラー：

「モジュールリスト取得エラー」

取得成功：モジュール一覧表示

```
[ Module list ]
  1: Module ID:1   version:1.05   attribute:optional    [ mem dump ]
  2: Module ID:2   version:1.50   attribute:standard   [ iosim ]
  3: Module ID:5   version:1.01   attribute:base       [ test md ]
                    ・
                    ・
                    ・
```

### 3) モジュール状態問い合わせ

| | |
|---|---|
| コマンド説明 | ターゲットに実装されているロードモジュールのうち、指定したモジュール IDのロードモジュール情報を表示する。モジュール IDに ”* ”または ”all ”を指定した場合、実装されている全てのロードモジュール情報を一覧表示する。 |
| コマンド名<br>省略形 | atatus<br>sta |
| パラメータ | モジュール ID |
| 書式 | status△[モジュール ID] |
| 使用例 | con>status *<br>con>sta all |
| オフライン運用時 | 有効 |

機能概略：

ターゲットのモジュールの状態情報を返す。

引数としてモジュール IDを指定した場合、そのロードモジュールの状態を表示する。

■メッセージ出力

エラー：

「状態取得エラー」

「モジュール IDが無効」

問い合わせ成功 :状態表示

```
[ Module status ]
   Module ID:2   version:1.50   attribute:optional        [ iosim ]
   Module ID:0   version:1.01   attribute:standard
   Module ID:5   version:1.01   attribute:base module   [ test md ]
                      ・
                      ・
                      ・
```

### 4) モジュールの削除

| | |
|---|---|
| コマンド説明 | ターゲットに実装されているロードモジュールのうち、指定したモジュール IDのロードモジュール削除する。 |
| コマンド名<br>省略形 | unload<br>uld |
| パラメータ | モジュール ID |
| 書式 | Unload△[モジュール ID] |
| 使用例 | con>unload 1 |
| オフライン運用時 | 有効 |

機能概略：

指定した IDのロードモジュールをターゲットから削除する。

■メッセージ出力

エラー：

「アンロードエラー」

「モジュール IDが無効」

削除成功：

「モジュールアンロード」

### 5) モジュールビルド（オフライン運用専用コマンド）

| | |
|---|---|
| コマンド説明 | ロードモジュールビルド指示情報をCF に出力する。 |
| コマンド名<br>省略形 | build<br>bl |
| パラメータ | モジュール ID<br>モジュールバージョン |
| 書式 | Build△[モジュール ID]△[モジュールバージョン] |
| 使用例 | >build 3 1.23 |
| オフライン運用時 | オフライン運用時のみ有効 |
| 備考 | 事前にＣＦをターゲットに挿入しておく必要がある |

■メッセージ出力

エラー：ＣＦカードがセットされていないなど。

「ファイルオープンエラー」

ファイル作成成功：

「BUILDREQ.LDM ファイル作成」

### 6) ロードモジュール追加、更新

| | |
|---|---|
| コマンド説明 | ロードモジュールをターゲットシステムにローディングする。<br>（オフライン運用時には、CF に収められたモジュールをロードする） |
| コマンド名<br>省略形 | load<br>ld |
| パラメータ | モジュール ID<br>バージョン |
| 書式 | load　（オフライン運用時は引数なし）<br>load <module_id> <version no.> |
| 使用例 | con>load 3 1.23 |
| オフライン運用時 | 有効 |

■メッセージ出力

エラー：

「ロードエラー」

ロード成功：

「モジュールロード」

## 7). サーバーの切断 終了）

| コマンド説明 | サーバーを切断します |
|---|---|
| コマンド名<br>省略形 | disconnect<br>disc |
| パラメータ | なし |
| 使用例 | >disc |
| オフライン運用時 | 無効 (オフライン運用時には、サーバとの通信を行なわないので、本コマンドは使用しない) |

■メッセージ出力

「Disconnect Server」

# 2.7 ダウンローディングターゲットのセットアップ

## 2.7.1 ターゲットシステム(ms7727cp01) H/W のセットアップ

ダウンローディング機構のターゲットハードウェアとして SH7709 T-Engine ボード(ms7727cp01)と LAN 拡張ボード(MSTLANEX01)が必要となります。T-Engine ボード開発セットに同梱されている取り扱い説明書に従って、ハードウェアを組み立てます。
(T-Engine ボード開発セットに含まれるデバッグボードを実装して、JTAG ICE を使用することも可能です。Computex 製 JTAG ICE PALMiCE HUD1360 を使用して動作確認をしております。)

・MS7727CP01 CPU ボード DIP-SW (SW5)設定
ms7727cp01 CPU ボードは 8bit DIP-SW が実装されています。各ビットの意味を下表に示します。設定位置に合せて変更します。

| 番号 | 設定内容 | SW 利用機能 | 出荷時状態 | 設定位置 |
|---|---|---|---|---|
| 1 | アプリケーションブート条件<br>ON:手動 OFF:自動 | T-Monitor | 自動 | 自動 |
| 2 | T-Monitor 通信速度 *1<br>ON:115K OFF:38.4K | T-Monitor | 38.4Kbps | 115Kbps |
| 3 | 未使用 | | | |
| 4 | CPU クロック<br>ON:114MHz OFF:96MHz | T-Monitor | 96MHz | 96MHz |
| 5 | 未使用 | | | |
| 6 | ロードモジュール登録初期化 *2<br>ON:初期化 OFF :未初期化 | ダイナミックローディング機構 | 未初期化 | 未初期化 |
| 7 | 電源 ON 条件 *3<br>ON:自動 OFF:手動 | H/W | 手動 | 自動(単体)<br>手動(ICE) |
| 8 | MD5 端子<br>ON:ビッグ OFF:リトル | H/W | リトル | リトル |

*1:シリアル通信速度 115Kbps は T-Monitor、ダイナミックダウンローディング機構 syslog 出力の通信速度を合せるためです。

*2 SW 番号 6 のみダイナミックローディング機構で使用しています。ON にして電源を ON にするとロードモジュール登録情報を初期化 (全て登録されていない状態)にフォーマットします。

*3
本 DIP-SW を ON に設定すると、AC 電源プラグを基版に挿入したことにより電源が全体に供給さ

れます。OFF 設定では、電源 SW を押下する事により電源コントローラによって電源が全体に供給されます。ICE 使用時は ON で設定します。

・デバッグボード

デバッグボードを装着することにより ICE を使用してのロード実行やフラッシュデバイスへの書き込みが可能となります。以下に ICE を使用する際のデバッグボードジャンパ設定を記述します。

| ジャンパピン | 設定内容 | 設定位置 |
|---|---|---|
| J1 | ブートアドレスへの配置デバイス<br>オープン:フラッシュROM<br>ショート:デバッグボードEEPROM | オープン |
| J2 | ASEMD 端子レベル<br>オープン:通常モード<br>ショート: H-UDI J-TAG ICE モード | ショート |

・LAN 拡張ボード（MSTLANEX01）設定

LAN 拡張ボードもジャンパピンによってハードウェア設定が変更可能です。この設定は出荷時デフォルト設定のままで変更は不要です。

デフォルト設定内容

LAN ボード割付エリア　　：　J4 オープン（CS4 選択）

LAN コントローラ割り込み信号：　JP45-6 ショート(IRQ2 選択）

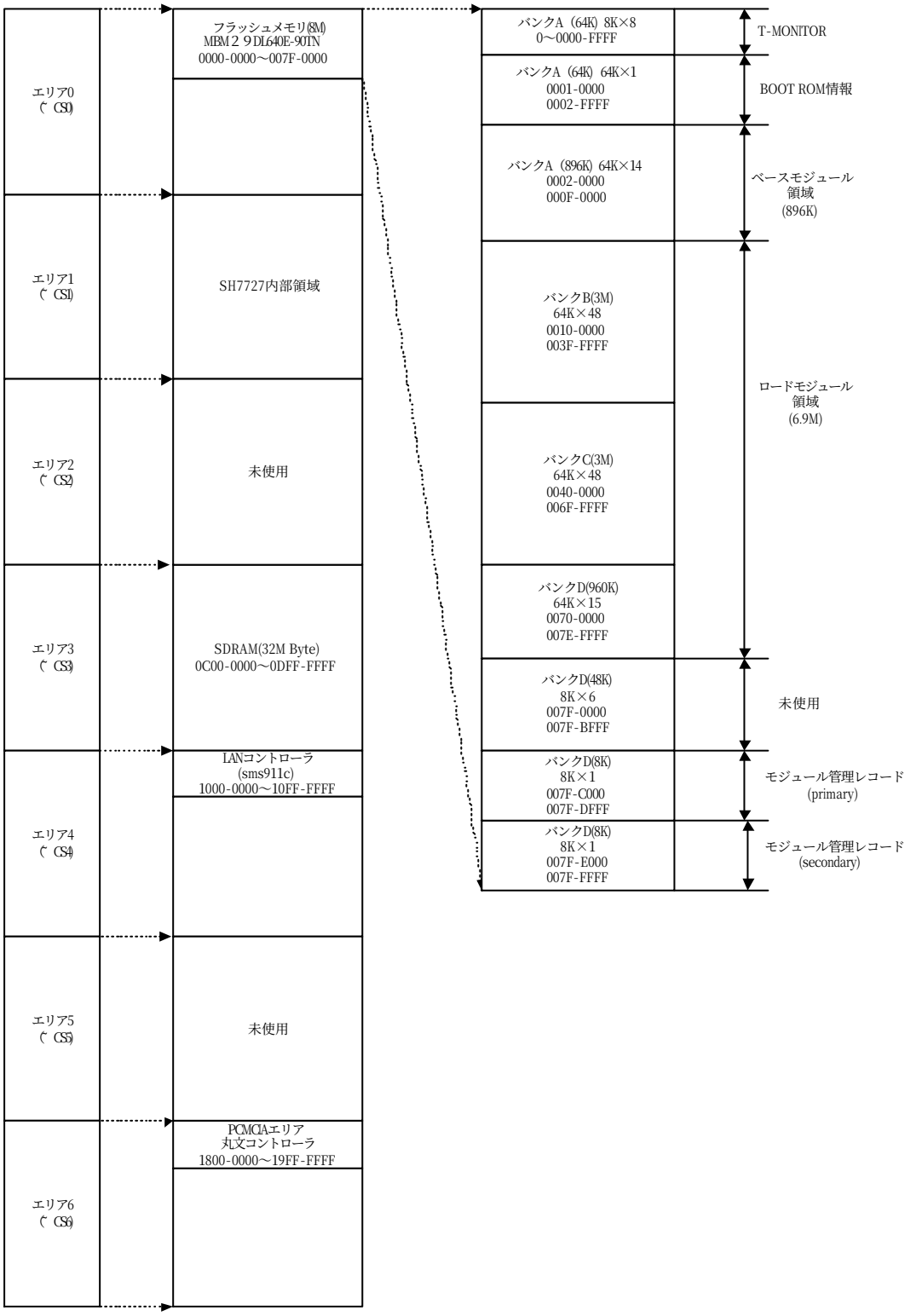



**図 2.7.1.1: T-Engine ボードメモリマップ**

## 2.7.2 ターゲットソフトウェアの構成

配布セットを展開したディレクトリ<inst_dir>/idl-opsw/target 以下がターゲットソフトウェアのソースディレクトリです。<inst_dir>/idl-opsw/target/idl_kernel 以下は μITRON カーネル機能を展開するディレクトリです。μITRONver4.0 完全準拠フルセットカーネルの配布セットから以下のディレクトリをコピーしてマージします。

カーネル本体 ./kernel
システムタスク ./systask
ライブラリ ./library
ビルドユーテリティ ./utils
CPU・システム依存部 ./config

## 2.7.3 ローダ機能のコンフィグレーション説明

ターゲットで動作するローダ機能をネットワーク環境に合せて以下の変更をする必要があります。

(1) ダウンロード運用モード
ファイル名: target/modules/base/Makefile

| 設定項目 | マクロ名 | 設定例 |
|---|---|---|
| 運用モード設定 | LOAD_OPERATION | SERVER_DEMAND : ネットワークサーバデマンド<br>CLIENT_DEMAND : ネットワーククライアントデマンド<br>OFFLINE : オフライン運用 |

(2) ネットワークパラメータの設定
ファイル名: target/loader/loader_param.h

| 設定項目 | マクロ名 | 設定例 |
|---|---|---|
| ターゲットIP アドレス | IDLTGT_IP | {192,168,0,250} |
| ダウンロード接続ポート | IDLTGT_PORT | 49512 |

本設定はオフライン運用を選択した場合、無効となる情報です。

# 3 章　チュートリアル編

## 3.1 ダウンローディングターゲットシステムの準備

### 3.1.1 ms7727cp01 のハードウェアセットアップ

ネットワーク操作を行う場合、ターゲット機器とダウンローディングサーバ PC をネットワークケーブルで接続します。オフライン操作を行う場合、CF 媒体を介して機器とサーバはインタフェースをします。本節では接続方法の一例を示します。

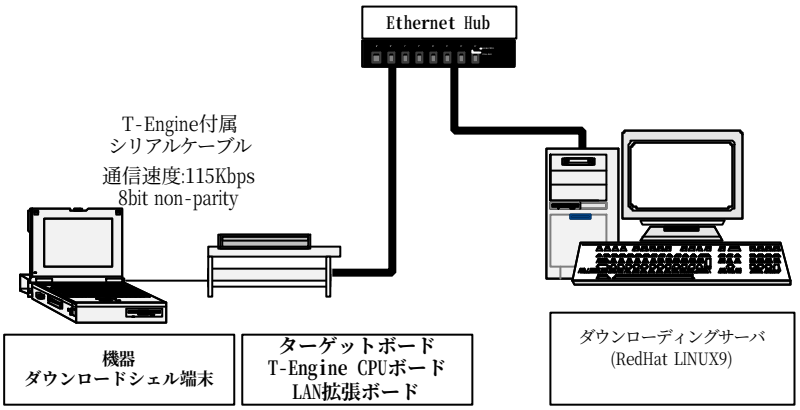



**図 3.1.1.1： T-Engine ボードを単体で使用する場合の接続方法（ネットワーク操作）**

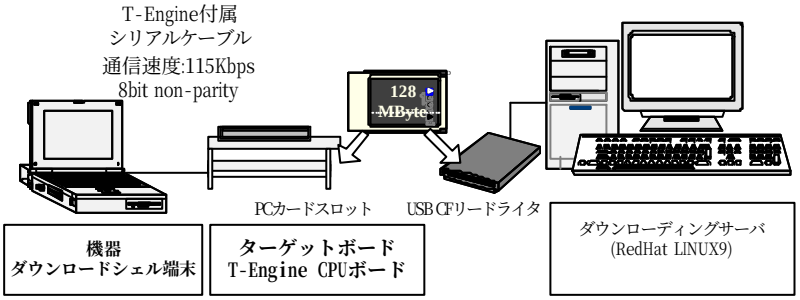



**図 3.1.1.2: T-Engine ボードを単体で使用する場合の接続方法（オフライン操作）**

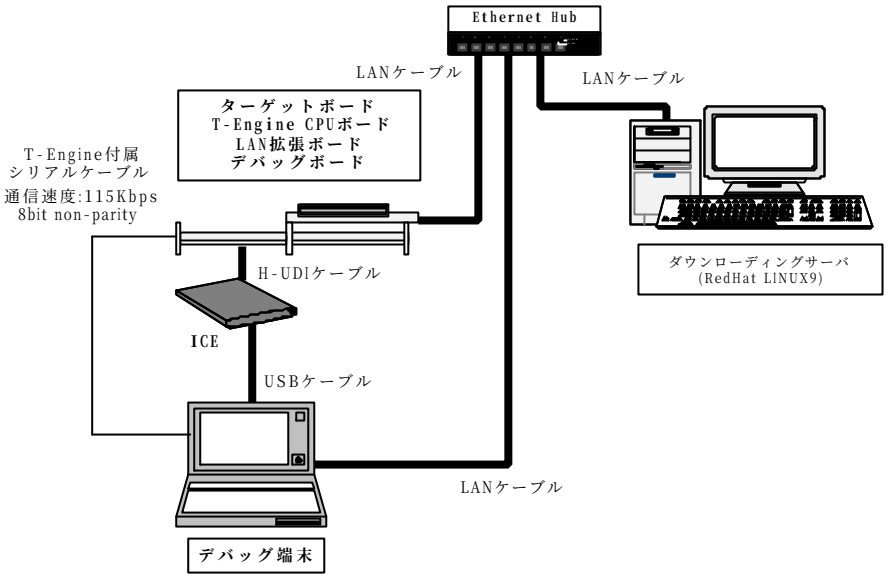
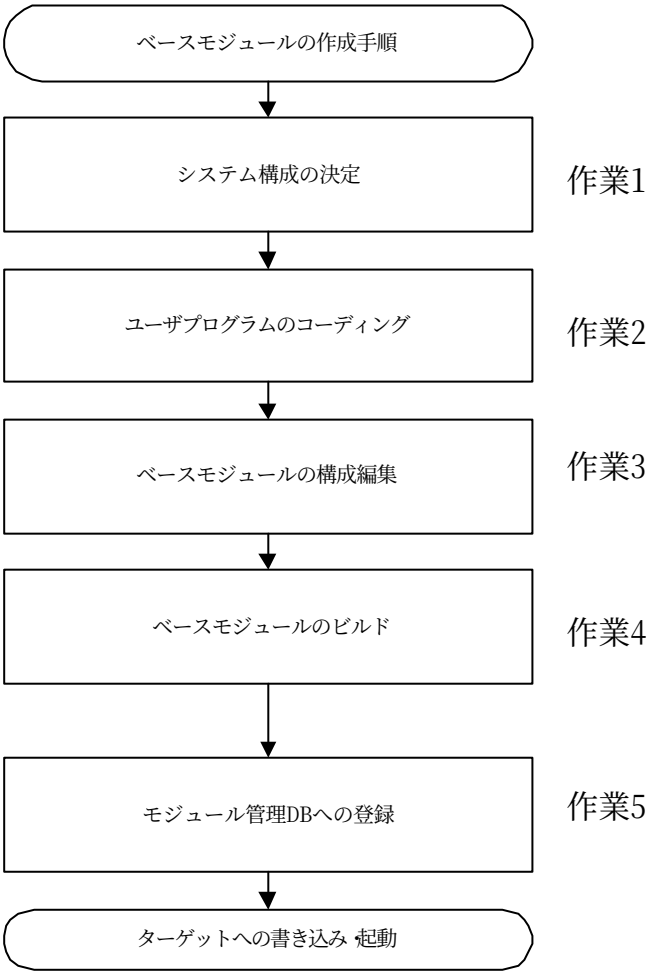



**図 3.1.1.3：ICE を使用する場合の接続方法**

### 3.1.2 ベースモジュールのコンフィグレーション及びビルド

ベースモジュールの作成は概ね以下の作業手順になります。以降サンプルベースモジュールを引用し各作業を説明します。



作業 1:システム構成の決定

ベースモジュールに含むサブシステムの構成を決定します。

最低限必要なものはカーネル機能、ローダ機能、ネットワークプロトコル機能です。必要に応じてサブシステムを追加して行きます。

作業 2：ユーザプログラムのコーディング

必要に応じてベースモジュールに含めるユーザプログラムをコーディングします。

作業 3：ベースモジュールの構成編集

ベースモジュールコンフィギュレーションファイルを編集し、ベースモジュールに必要なカーネルオブジェクトを追加していきます。また、今後動的に追加されるカーネル資源の最大数やロードモジュールの装着最大数を見積り、コンフィギュレーションファイルに設定します。サンプルでは本ファイルは basemdl.cfg というファイル名で格納されています。

```
43 VDEF_KMB(3000); /* カーネルヒープ領域サイズ */
44
45 INCLUDE("\"bsmcfg.h\"");
46 INCLUDE("\"itron_px.h\"");
47
48 /* ベースモジュールの属性設定 */
49 VATR_BSM( "dynamic-loader with FI40", { TA_NULL, 01.00 });
50
51 /* system timer */
52 #include "../systask/timer.cfg"
53 /* serial driver */
54 #include "../systask/serial.cfg"
55 /* loader */
56 #include "../../loader/loader.cfg"
57
58 /* 資源最大数の定義 */
59 VNUM_LDM( 50 );
60 VNUM_EXP( 100 );
61 VAID_TSK( 100 );
62 VAID_SEM( 30 );
63 VAID_DTQ( 30 );
64 VAID_FLG( 30 );
65 VAID_MBX( 100 );
66 VAID_MPF( 30 );
67 VAID_CYC( 30 );
68 VAID_MPL( 10 );
70 VAID_MTX( 30 );
71 VAID_ALM( 30 );
72 VAID_MBF( 30 );
73 VAID_POR( 30 );
```

カーネルサービスコールはコンフィギュレータ機能によって自動エクスポートされます。この事により

ユーザはベースモジュールに含まれるサービスコールをロードモジュールサイドから別ロード単位である事を意識することなく呼び出す事が可能です。その他ベースモジュールに含まれるシンボルをエクスポートしたい場合、EXPORT 静的 API を使用して明示的に公開する必要があります。

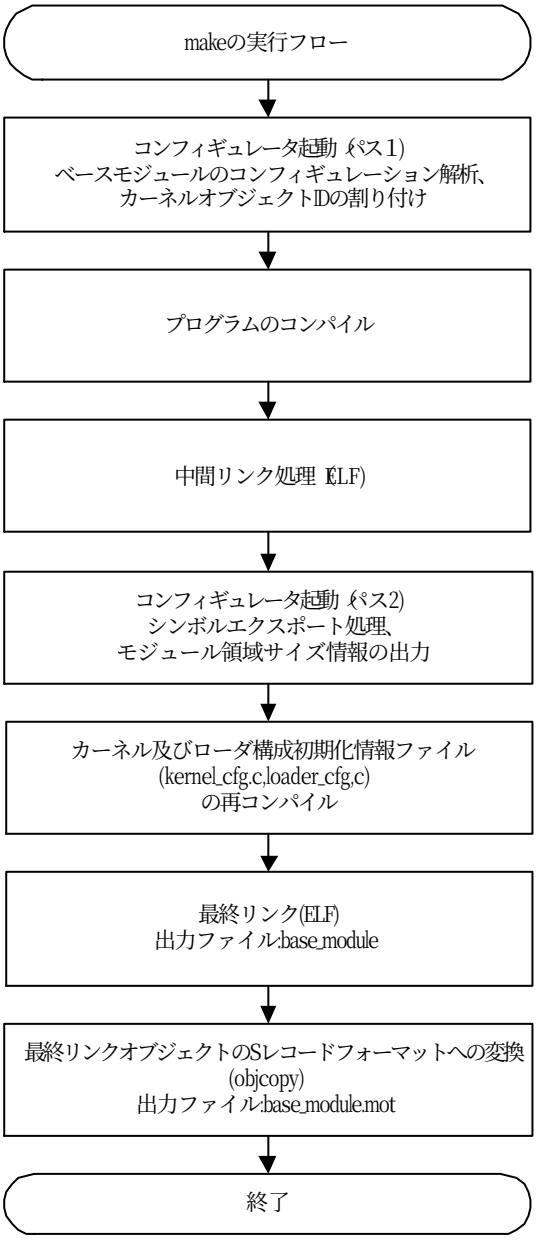
```
102 /* ベースモジュールからのエクスポートシンボル定義 */
103 EXPORT( "ER syslog(UINT prio, const char *format, ...)" );
104 EXPORT( "ER query_mod(ID mod_id, UH *mod_ver, W *mod_type, W *all_module_num)" );
105
106 /* RTNET 関連の EXPORT 定義 */
107 EXPORT("int  accept( )");
108 EXPORT("int  bind( )");
109 EXPORT("int  closesocket( )");
110 EXPORT("int  ioctlsocket( )");
111 EXPORT("int  listen ( )");
112 EXPORT("int  recv ( )");
113 EXPORT("int  select( )");
114 EXPORT("int  sendto ( )");
115 EXPORT("int  setsockopt ( )");
116 EXPORT("int  socket ( )");
117 EXPORT("void  proc_fd_zero( )");
118 EXPORT( "UB bsm_ver[]");
```

作業 4：ベースモジュールのビルド

次にベースモジュールのビルドを行います。サンプルベースモジュールのフォルダ target/modules/base にて以下のコマンドを実行します。

```
$ make 'ROM=true'
```

下図のように make 実行によりビルド処理が動作し、最終ロードモジュール base_module.mot が作成されます。



作業 5：モジュール管理データベースへの登録

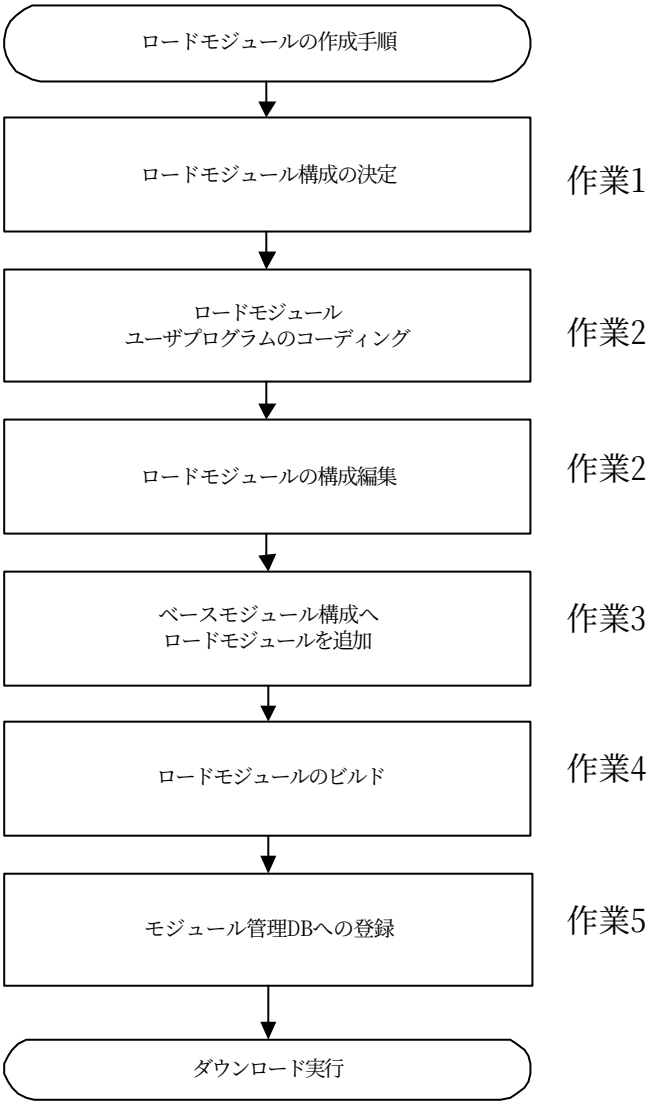
ベースモジュール書き込み動作させるターゲット機器のネットワークノード情報をモジュール管理 DB に追加します。

```
#    ターゲット機器 ノード登録情報
#     ターゲット機器の接続情報を登録する。
#     書式：
#     N,<ターゲット名>,<ターゲット ID>,<IP-Address>,<Port-Number>
N,TE4,3,192.168.0.250,49512
```

作業 6：ベースモジュールのターゲット機器への書き込み

ターゲットに電源を投入し、最終ロードモジュール base_module.mot を T-Engine ボードのフラッシュ ROM に書き込みます。フラッシュROM へのプログラム書き込み方法として、T-Monitor を利用する方法とICE を使用する方法があります。書き込み方法の詳細は T-Monitor 又は ICE のマニュアルをご覧下さい。

### 3.1.3　ロードモジュールの作成



作業 1 ロードモジュール構成の決定

ロードモジュールに含むタスク構成やライブラリ構成を決定します。また、ここでロードモジュールが外部に公開するサービスも決定します。本書では ebs 社製 WEB サーバをロードモジュールとして動作させる方法を例として説明します。

作業 2 ユーザプログラムのコーディング

作業 1 で決定した構成に基づき、ロードモジュールに含めるソースコードのコーディングを行います。ロードモジュール初期化処理、終了処理は必須ですので合せてコーディングを行います。

ここでは/home/someone /idl-opsw/target/modules/load/web/ver2.00 のディレクトリにソースファイルを格納するものとします。

作業 3 ロードモジュールの構成編集

ロードモジュールコンフィグレーションファイルを作成し、ロードモジュールに含めるカーネルオブジェクトを静的 API で記述します。ldm_websvr.cfg には WEB サーバが使用するカーネルリソースを設定しています。

```
INCLUDE("\"websvr.h\"");
INCLUDE("\"ebs.h\"");

VATR_LDM( ldm_websvr, "web server", { TYPE_OPTION, 0x200} );
CRE_TSK( ID_RTIP_TSK_MAINAPP,
        { TA_HLNG|TA_ACT, 1, tsk_websvr, 12, 4096, NULL } );
/* WEBCFG_WEB_MAX_SPAWN */
CRE_TSK(ID_RTIP_TSK_WEBS_0, {TA_HLNG, 0, ks_task_entry, 10, 16384, NULL });
CRE_TSK(ID_RTIP_TSK_WEBS_1, {TA_HLNG, 0, ks_task_entry, 10, 16384, NULL });
CRE_TSK(ID_RTIP_TSK_WEBS_2, {TA_HLNG, 0, ks_task_entry, 10, 16384, NULL });
CRE_TSK(ID_RTIP_TSK_WEBS_3, {TA_HLNG, 0, ks_task_entry, 10, 16384, NULL });
CRE_TSK(ID_RTIP_TSK_WEBS_4, {TA_HLNG, 0, ks_task_entry, 10, 16384, NULL });
/* Signal Semaphore */
CRE_SEM( ID_WEB_SEM0, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM1, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM2, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM3, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM4, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM5, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM6, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM7, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM8, { TA_TFIFO, 1, 1 } );  /*  */
CRE_SEM( ID_WEB_SEM9, { TA_TFIFO, 1, 1 } );  /*  */
/*  */
EXPORT_INI( web_init_mdl );
EXPORT_TER( web_exit_mdl );
```

作業４ ベースモジュール構成へのロードモジュー ル登録
次に VATT_LDM 静的 API でベースモジュールコンフィグレーションファイルに先程作成したロードモジュールコンフィグレーション情報を登録します。

```
VATT_LDM(ldm_websvr,
"/home/someone/idl-opsw/target/modules/load/web/ver2.00/ ldm_websvr" );
```

登録後、ベースモジュールディレクトリで make update と実行することでコンフィグレータを動作させ、ベースモジュール情報にロードモジュールを追加します。（ここではベースモジュールのコンパイル・リンク動作は不要です。）この操作を行うことでモジュール ID が自動採番され、その ID 値は kernel_id.h に ldm_websvr マクロとして出力されます。ここではモジュール ID が 1 として採番出力されたと仮定し説明します。

作業５ ロードモジュールのビルド
次に先程のロードモジュール作業ディレクトリ(/home/someone /idl-opsw/target/modules/load /web/ver2.00)にて make を実行します。この実行でロードモジュール中間ファイル load_module.o とモジュールビルダへの補助情報ファイル build.inf が作成されます。

作業６
ロードモジュールの準備として最後のステップである、モジュール管理データベースへ の作成ロードモジュール登録を行います。ここではターゲット ID に実装できるロードモジュール ID=1 として登録します。

```
M,1,3,0x00000200,0x200,1,IPAweb server (OPTION),
/home/opsw/idl-opsw/target/modules/load//web/ver2.00/build.mak,
/home/opsw/idl-opsw/target/modules/base,/
home/opsw/idl-opsw/target/modules/load//web/ver2.00
```

## 3.1.4 ダウンロードの操作

### (1) ターゲットの起動

ターゲットボードの電源を入れ、フラッシュに書き込んだベースモジュールを動作させます。
正常に起動すると FI4 カーネルとダイナミックダウンローディング機能のバナーが出力される事を確認することができます。

```
FI4 Kernel Release 1.4 (patchlevel = 0) for MS7727CP01(SH7727 T-Engine) (Feb 18 2004, 16:27:21)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
                            Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2003- by Monami software, LP

ITRON with Dynamic Loading System(network operation(client demand)) (Feb 18 2004, 16:26:46)
Copyright (C) 2002,2003 by AI Corporation, Inc., JAPAN
Copyright (C) 2002,2003 by Information-technology Promotion Agency, JAPAN
Base-Module Description[uITRON FullSpec. with loader] Version[00.01]
```

### (2) ダウンロードサーバの起動

ダウンロードサーバを起動します。Linux システム起動時、デーモンプロセスとして起動する事も可能ですが。ここでは root ユーザでフォアグランドプロセスとして起動しています。

```
$ su
  Password: xxxx --- 管理者パスワード
[root]#idlsrv -debug
IDL Server Version 02.00
[2003-11-24 13:43:06] START
*** IP address : 192.168.0.207 port number 49512
```

### (3) ネットワーク サーバデマンド操作によるダウンロード実行

機器自体が設備に隠蔽され、機器ユーザとのインタフェースを持たない場合、サーバより機器に対して指示を行い、プログラムをプラグインしたり、バージョンアップする運用形態が考えられます。
本節ではサーバデマンド操作でのダウンローディングの実際の操作を説明します。

まず、Linux サーバのシェルプロンプトにより idl_shell と入力し、ダウンロードシェルプログラムを起動します。

```
$ idl_shell
Connect Server IP:127.0.0.1 port:49512
>
```

ダウンロードシェルから接続可能なターゲットシステム一覧を表示し、ターゲット接続を行います。
以下では TE5 という名称が設定されている機器に接続を行っています。接続が正常に終了すると **CON>** プロンプトに変わります。プロンプトが接続状態(CON>)に変わると、ターゲットへの操作（状態取得、ロードモジュールの削除、ダウンロード）が可能となります

```
>tglist
[ Target list ]
  1: Target IP [192.168.0.33:49512]  TE1
  2: Target IP [192.168.0.250:49512]  TE4
  3: Target IP [192.168.0.220:49512]  TE5
  GROUP DL-DemoSystem
  GROUP DL-DemoSystem3
  GROUP DL-DemoSystem4
>con TE5
Connect Target IP:192.168.0.220 port:49512
CON>
```

なお、接続するターゲットはグループ化が可能です。connect(con)コマンドでグループ名を指定した場合、複数の機器に同時に接続を行い、ロードモジュールの一斉ダウンロードが行えます。
いよいよダウンロードの実行ですが、以下ではダウンロード前に機器のモジュール状態を確認しています。機器の状態表示からベースモジュール以外何もロードされていないことが確認できます。

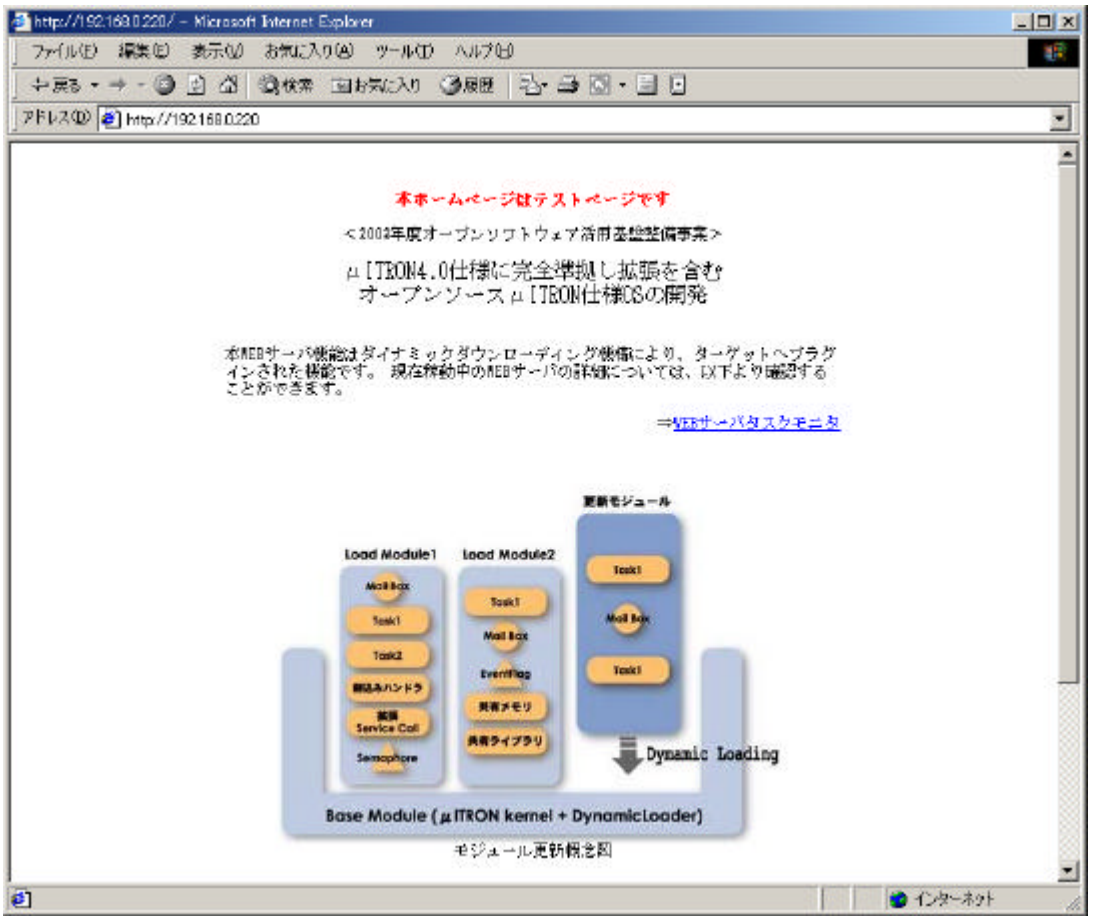
```
con>sta *
[ Module status ]
  1: Target IP [192.168.0.220:49512]
   Module ID:0   version:1.00  attribute:basic      [ BASE_MODULE ]
```

以下は接続ターゲット(TE5)に対してモジュール ID 1 ver2.00 のロードモジュールをダウンロード実行し、機器のロード状態を確認する操作したところです。新たに WEB サーバロードモジュールが装着された事が確認できます。

```
>mdlist
[ Module list ]
 Module ID:1  version:2.00  attribute:optional   [ IPA web server (OPTION) ]
 Module ID:1  version:3.00  attribute:optional   [ IPA web server (OPTION) ]


con>ld 1 2.00
[ Module loading ]
  1: Target IP [192.168.0.220:49512]
Please wait a moment to load...
 Load complete
 (10295)
con>sta *
[ Module status ]
  1: Target IP [192.168.0.220:49512]
   Module ID:0   version:1.00  attribute:basic      [ BASE_MODULE ]
   Module ID:1   version:2.00  attribute:optional   [ IPA web server (OPTION) ]
```

以下はダウンロードした WEB サーバモジュール機能によって返却されるホームページドキュメントを PC の WEB ブラウザより閲覧した画面です。



モジュール更新概念図

上記にてダウンロードしたロードモジュールはオプション属性のため、アンロードすることが可能です。以下では、アンロード実行後、再度モジュールの状態を確認し ID=1 のロードモジュールが削除されたことを確認しています。

```
con>uld 17
[ Module unload ]
  1: Target IP [192.168.0.220:49512]
 Delete complete

con>sta *
[ Module status ]
  1: Target IP [192.168.0.220:49512]
   Module ID:0   version:1.00  attribute:basic      [ BASE_MODULE ]
```

シェルコマンドを終了するには,q とタイプします。このコマンドでターゲット接続を終了し、ダウンロードサーバとのセッションを終了します。

```
con>q
  1: Target IP [192.168.0.220:49512]
Disconnect Server
```

## (4) ネットワーク クライアントデマンド操作によるダウンロード実行

機器自体が機器ユーザとのインタフェース機能を持つ場合、機器ユーザの都合で新たなプログラムのプラグインやバージョンアップ操作を行う運用形態が考えられます。
本節ではクライアント機器）デマンド操作でのダウンローディングの実際の操作をご説明います。

ベースモジュールをクライアントデマンドコンフィグレーションを行ってビルドした場合、以下のようにシステム起動時　ダウンロードシェルが起動されます。
（機器で動作するダウンロードシェルは T-Engine ボード シリアルケーブルをパソコン COM ポートに接続し、汎用ターミナルソフトを起動することで操作することができます。）
以下では help コマンドにて実装されているコマンド一覧を表示し、hostname コマンドで機器のアドレス IP アドレス）を確認しています。

```
FI4 Kernel Release 1.4 (patchlevel = 0) for MS7727CP01(SH7727 T-Engine) (Feb 18 2004, 16:27:21)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
                                Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2003- by Monami software, LP


ITRON with Dynamic Loading System(network operation(client demand)) (Feb 18 2004, 16:26:46)
Copyright (C) 2002,2003 by AI Corporation, Inc., JAPAN
Copyright (C) 2002,2003 by Information-technology Promotion Agency, JAPAN
Base-Module Description[uITRON FullSpec. with loader] Version[00.01]

**** TOPPERS/IDL TARGET SHELL ****
Version: $Revision: 1.14 $
> help
help (h)            help
connect (con)    connnect [IP]:[ポート]
disconnect (disc)          disconnect
mdlist (md)        mdlist
status (sta)      status ([ID]|*|all)
unload (uld)      unload [ID]
load (ld)          load [ID] [モジュールバージョン]
hostname (host) hostname ([IP])
bdump (bd)         bdump [スタートアドレス] [サイズ]
hdump (hd)         hdump [スタートアドレス] [サイズ]
wdump (wd)         wdump [スタートアドレス] [サイズ]

> hostname
host: 192.168.0.223
```

機器よりダウンロード操作を行う場合は connect(con)コマンドでダウンロードサーバに接続します。接続が正常終了するとプロンプトが CON>に変わります。

```
> con 192.168.0.207:49512
Connect Server IP: 192.168.0.207:49512
CON>
```

サーバに接続完了後、ロード可能なモジュール一覧を取得し、モジュールロードの操作を行います。以下ではモジュール ID＝１ バージョン番号 1.00 のロード操作を行っています。ロード実行後 status(sta)コマンドでロードモジュールの装着状態を確認すると、モジュール ID＝１が追加されたことが確認できます。

```
CON> mdlist
[Module list]
Module ID: 1 version: 01.00 attribute: optional IPA test 3(1) (OPTION)
Module ID: 2 version: 01.00 attribute: optional IPA test 3(2) (OPTION)
Module ID: 3 version: 01.00 attribute: optional IPA test 4(1) (OPTION)
Module ID: 4 version: 01.00 attribute: optional IPA test 4(2) (OPTION)
Module ID: 5 version: 01.00 attribute: optional IPA test 5 (OPTION)
Module ID: 6 version: 01.00 attribute: optional IPA test 6 (OPTION)

CON> load 1 1.00
callback init_mdl        ------ ダウンロードしたロードモジュールが出力しているデバッグ
MODULE3-1 : Test Task Start     メッセージ
モジュールロード

CON> status -m *
[Module status]
Module ID:0  version: 00.01 attribute: base     ulTRON FullSpec. with loader
   Memory object block  1: start A0020000  end A0055578
   Memory object block  2: start AC100000  end AC1CA29C
Module ID:1  version: 01.00 attribute: optionalTest Loa Memory object b8B
   Memory object block  2: start AC000000  end AC00114B
   Memory object block  3: start 00100190  end 0010033B
```

ダウンロードシェルのコマンドを使用して、機器自身のロードモジュールを削除することが可能です。
以下は先ほどダウンロードしたロードモジュールを削除しています。

```
CON> unload 1
unload module
```

ダウンロード操作終了時は、disconnect(discon)コマンドにてダウンロードサーバとの切断を行います。

```
CON> discon
Disconnect server
>
```

(5) オフライン操作によるダウンロード実行

昨今の組み込み機器はネットワーク機能を搭載したものが増えてきてきましたが、ネットワーク機能をもたない機器も依然存在しています。このような機器ではフロッピーディスクや CF のようなリムーバブルストレージ媒体を使用してソフトウェアのバージョンアップする運用が考えられます。
本節ではオフライン操作でのダウンローディング操作を説明します。
機器側は T-Engine ボードの PC カードスロット、Linux サーバ側は USB カードリードライタを使い、レキサーメデ ィア社製 CF ストレージカードを使ってダウンロード実行を行っています。）

オフライン運用コンフィグレーションした機器は電源を投入すると以下のように機器で動作するダウンローディングシェルが起動され、コマンドプロンプトが表示されます。

```
ITRON with Dynamic Loading System(offline operation) (Feb 16 2004, 18:19:43)
Copyright (C) 2002,2003 by AI Corporation, Inc., JAPAN
Copyright (C) 2002,2003 by Information-technology Promotion Agency, JAPAN
Base-Module Description[uITRON FullSpec. with loader] Version[00.01]

**** TOPPERS/IDL TARGET SHELL ****
CON> help
help (h)p          help
status (sta)     status ([ID]|*|all)
unload (uld)     unload [ID]
build (bl)       build [ID] [モジュールバージョン]
load (ld)          load
bdump (bd)       bdump [スタートアドレス] [サイズ]
hdump (hd)       hdump [スタートアドレス] [サイズ]
wdump (wd)       wdump [スタートアドレス] [サイズ]
```

T-Engine ボード CF スロットに CF カードを挿入し、build コマンドをタイプすることででロードモジュールビルド指示データを CF に作成します。

```
CON> build 1 1.00
BUILDREQ.LDM ファイル作成
```

次に機器より CF カードを抜き、Linux に接続されている CF リードライタに先程書き込んだ CF カードを挿入します。
サーバで動作するダウンロードシェルを起動し、build コマンドでロードモジュールの作成を完了させます。

```
$ idl_shell
Connect Server IP:127.0.0.1 port:49512
>
>build
[ Offline Build ]
 Module complete
```

再度、サーバ USB CF カードリードライタからCF カードを抜き、機器の PC カードスロットに挿入し、ダウンロード操作を完了させます。
以下では load コマンドでビルドしたモジュール ID=1 バージョン番号=1 のロードモジュールを機器にダウンロードし、status コマンドでロードモジュールの装着状態を確認しています。

```
CON> load
MODULE3-1 : Test Task Start
モジュールロード

CON> status -m *
[Module status]
Module ID:0  version: 02.00 attribute: base    uITRON FullSpec. with loader
   Memory object block  1: start A0020000  end A00638EC
   Memory object block  2: start AC100000  end AC1D304C
Module ID:1  version: 01.00 attribute: optionalTest Load Module 30
   Memory object block  1: start A0100000  end A010018B
   Memory object block  2: start AC000000  end AC00114B
   Memory object block  3: start 00100190  end 0010033B
```

# 4 章 補足資料編

## 4.1 他のシステムへの移植ガイド

### 4.1.1 TOPPERS フル機能/カーネルサービス

ダイナミックダウンロード機能はフルセットカーネルの機能に依存しています。現状フルセットカーネル対応のプロセッサは SH3 のみですが、TOPPERS プロジェクトでは対応 CPU を増やしていく予定です。またユーザ自身でもフルセットカーネルの CPU・H/W システム依存部の構築ガイドを参照することにより対応することが可能です。詳細はフルセットカーネルの移植ガイドを参照してください。

### 4.1.2 ネットワークサービス

ダイナミックローディングを T-Engine ボード上で動作させるためには、TCP/IP ネットワークへのアクセス機能が必要となります。ダイナミックローディングは現状以下の EBS 社製 TCP/IP プロトコルスタックを利用して確認をしておりますが、ソケット API の利用可能な他社 TCP/IP プロトコルスタックを実装することで容易にダイナミックローディング機能を移植する事が可能です。
以下に移植の際見直す関数を示します。

ファイル名：loader/idl_comm_transport.c
ネットワークワーク機能の初期化
　int init_comm( IDL_SAP *server_sap, IDL_SAP *target_sap )
ネットワークワーク機能の終了
　int terminate_comm( VP param )
トランスポートオープン
　ER_ID comm_open_init( void )
トランスポート解放
　ER_ID comm_open_term( ID so_id )
ダウンロードセッションのオープン
　ER_ID comm_open_req( ID conn_so_id )
送信処理
　int comm_send_req( ID so_id, UB *data, UH len )

受信処理

int comm_read_req( ID so_id, UB *buf, UH buf_len )

セッションクローズ

int comm_close_req( int so_id )

ファイル転送要求

ER_ID comm_getfile_req( IDL_SAP *pserver_sap, char *filepath )

ファイルデータの読み出し

ER_UINT comm_fileread_req( ID tftp_soid, UB *readbuf, UH buflen, UH *pseqno, UW *peof_flg,

ファイルデータの確認応答

ER comm_filedata_cnf( ID tftp_soid, IDL_SAP *p_ackto, UH seqno )

ファイル転送の中止

ER comm_trabort_req( ID tftp_soid, IDL_SAP *p_errto )

ファイル転送の終了

ER comm_file_close( ID tftp_soid )

受信データ発生確認

UH poll_valid_data( ID so_id )

### 4.1.3　フラッシュメモリドライバ

配布ソースは T-Engine ターゲットボード MS7727CP01 に搭載されたフラッシュメモリチップ MBM29DL640E に対する、メモリドライバを実装しています。他のターゲットシステムに移植する際搭載されているメモリチップに適合したメモリドライバを移植する必要があります。補足説明として他のメモリチップに移植する際、変更が必要なファイル、関数を以下に示します。

修正ファイル名：target/loader/flash_drv.c

修正対象関数名：
・フラッシュ消去機能(ER　Drv_Flash_Erase( ))
・フラッシュ書き込み機能(ER　Drv_Flash_Write( ))
・フラッシュバッファ書き込み機能(ER　Drv_Flash_Buff_Write( ))
・フラッシュ保護機能(ER　Drv_Flash_Lock( ))
・フラッシュ保護解除機能(ER　Drv_Flash_Unlock( ))
・フラッシュ初期化関数(ER　Drv_Flash_Init( ))

# 4.2 ICE を使用したダウンロードモジュールのデバッグ方法の紹介

ダウンロードサーバを'-debug'オプション付で起動した場合、ダイナミックローディングで作成した最終リンクファイル(ELF)を削除しません。この最終リンクファイルに存在するデバッグ情報のみをダウンロード後デバッガに読み込ませる事で、ロードモジュールの C 言語ソースデバッグが可能となります。

最終ロードモジュールが残っているディレクトリ:
ダウンローディングサーバ設定ファイル(idlsrv.conf) で設定した IDLSRV_TFTP_PATH ディレクトの以下ディレクトリ
<IDLSRV_TFTP_PATH ディレクトリ>/IDL/<ダウンローディングセッション番号>

なお、<ダウンローディングセッション番号>はロード実行完了メッセージで確認する事が可能です。

```
con>ld 1 2.00
[ Module loading ]
  1: Target IP [192.168.0.251:49512]
Please wait a moment to load...
 Load complete
(10295) ―― セッション番号
```

最終リンカアウトプットファイル名:
ダウンローディングサーバ設定ファイル(idlsrv.conf) で設定した IDLSRV_LINKER_OUTPUT で設定したファイル(デフォルトファイル名は a.out)。

補足:
gcc ツールチェインをサポートするデバッグシステムは、シームレスに gcc オブジェクトファイルを読み込ませる事が可能なものと、デバッガが認識できるファイルフォーマットへのコンバータを提供しているものが存在します。
今回使用した Computex 社製デバッガシステムは後者の方で dw22ctx.exe というWindows で実行可能なオブジェクトコンバータが付属します。今回の開発ではこの dw22ctx.exe を使用しシンボルデバッグを行いました。