SUZAKU
Software Manual

Version 1.1.0

2005 年 1 月 31 日

株式会社アットマークテクノ
http://www.atmark-techno.com/

目次

1. はじめに 1
1.1. マニュアルについて 1
1.2. フォントについて 1
1.3. コマンド入力例の表記について 1
1.4. 注意事項 2
1.4.1. 安全に関する注意事項 2
1.4.2. 取り扱い上の注意事項 2
1.4.3. ソフトウェア使用に関しての注意事項 3
1.5. 謝辞 3
2. SUZAKU の概要 4
2.1. MicroBlaze 4
2.2. uClinux 4
3. 作業の前に 5
3.1. 準備するもの 5
4. まず、動かしてみる 6
4.1. 接続方法 6
4.2. シリアル通信用ソフトウェア 6
4.3. 電源を入れる 8
4.4. ログイン 9
4.5. コマンド入力 9
4.6. ウェブ 10
4.7. 終了方法 10
5. Flash メモリのメモリマップ 11
6. 起動の流れと起動モード 13
6.1. 起動の流れ 13
6.2. 起動モード 14
6.2.1. オートブートモード 14
6.2.2. ブートローダモード 14
6.2.3. モトローラ S 形式ダウンロードモード 15
7. 開発環境の準備 16
7.1. ツールチェーンのインストール 16
7.2. 環境変数の設定 16
7.3. ソースコードの準備 16
7.4. Minicom のインストール 17
7.5. Hermit のインストール 17
8. uClinux の image を作成 18
8.1. 作業の流れ 18
8.2. Menuconfig 20
8.2.1. メインメニュー 20
8.2.2. Vendor/Product メニュー 21
8.2.3. Kernel/Library/Defaults メニュー 21
8.3. ビルド 22
9. uClinux の設定変更 23
9.1. カーネルの設定変更 23
9.2. ユーザランドの設定変更 23
9.3. ZFLAT 対応カーネルを作る 25
10. 自作アプリケーションをイメージに入れる 26
10.1. ディレクトリの作成 26

10.2. Makefile の作成 26
10.3. C ソースコード 27
10.4. コンパイル 28
10.5. ROMFS ディレクトリへのインストール 28
10.6. イメージファイルの生成と実行 28
11. Flat Binary Format 30
11.1. Flat Binary Format の特徴 30
11.2. 実行ファイルの圧縮 30
11.2.1. コンパイル済バイナリファイルを圧縮 30
11.2.2. コンパイル時に圧縮 31
11.3. スタックサイズの指定 32
11.3.1. コンパイル済バイナリファイルスタックサイズを変更 32
11.3.2. コンパイル時にスタックサイズを指定 32
12. Flash メモリの書き換え方法 34
12.1. NetFlash を使用した Flash メモリの書き換えかた 34
12.2. Hermit を使用した Flash メモリの書き換えかた 35
12.2.1. ブートローダモードで起動する 35
12.2.2. ダウンロード 35
12.3. モトローラ S 形式での Flash メモリの書き換えかた 36
13. トラブルシューティング 38
13.1. Hermit 38
13.2. Minicom 38
13.3. uClinux-dist のコンパイル 38

表目次

表 1-1 使用しているフォント 1
表 1-2 表示プロンプトと実行環境の関係 1
表 4-1 シリアル通信設定 7
表 4-2 SUZAKU 初期設定時のユーザとパスワード 9
表 5-1 Flash メモリマップ 11

図目次

図 4-1 SUZAKU のコネクタ配置図 6
図 4-2 Minicom の設定 7
図 4-3 SUZAKU Web Page 10
図 6-1 起動モードの流れ 13
図 6-2 起動モードジャンパ 14
図 8-1 イメージファイル作成の流れ 19
図 8-2 uClinux v3.1.0 Configuration Main Menu 20
図 8-3 AtmarkTechno を選択 21
図 8-4 Kernel/Library/Defaults Selection 22
図 9-1 ユーザランド用 Main Menu 23

例目次

例 4-1 **minicom** の起動 8
例 4-2 uClinux の起動ログ 8
例 4-3 **ifconfig** での表示 9
例 6-1 ブートモード選択画面 14
例 6-2 Hermit の起動 14
例 7-1 Minicom のインストール（Debian で dpkg を使用) 17
例 7-2 Minicom のインストール（Redhat で rpm を使用) 17
例 7-3 Hermit のインストール（Debian) 17
例 7-4 Hermit のインストール（Redhat) 17
例 8-1 make menuconfig の実行 20
例 8-2 ビルドコマンド 22
例 10-1 Hello World 用のディレクトリを用意する 26
例 10-2 Hello World Makefile 27
例 10-3 Hello World ソースコード 28
例 10-4 make の実行 28
例 10-5 Hello World を ROMFS ディレクトリへインストール 28
例 10-6 イメージファイル生成 29
例 10-7 実行 29
例 11-1 通常の Flat Binary Format 31
例 11-2 圧縮された Flat binary Format 31
例 11-3 **FLTFLAGS** による圧縮の指定 32
例 11-4 スタックサイズの変更 32
例 11-5 **FLTFLAGS** によるスタックサイズの指定 33
例 12-1 netflash の起動 34
例 12-2 netflash のヘルプ 34
例 12-3 image.bin の書き換え 35
例 12-4 Hermit コマンド 35
例 12-5 Hermit のヘルプ 36
例 12-6 モトローラ S 形式ダウンロード開始画面 37

1.はじめに

1.1.マニュアルについて

本マニュアルは、SUZAKU 上で MicroBlaze™ソフトプロセッサと uClinux を使用する上で必要な情報のうち、以下の点について記載されています。

- 基本的な使い方
- Flash メモリの書き換え
- カーネルとユーザランドのビルド
- カスタマイズ
- アプリケーション開発

また、特別な表記がないかぎり作業用のコンピュータでは Linux をベースにした OS が動作しているものと仮定します。

SUZAKU の機能を最大限引き出すために、ご活用いただければ幸いです。

1.2.フォントについて

このマニュアルでは以下のようにフォントを使っています。

表 1-1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列

1.3.コマンド入力例の表記について

このマニュアルに記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「/」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「~」で表わします。

表 1-2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC 上の特権ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[SUZAKU /]#	SUZAKU 上の特権ユーザで実行
[SUZAKU /]$	SUZAKU 上の一般ユーザで実行

1.4. 注意事項

1.4.1. 安全に関する注意事項

SUZAKU を安全にご使用いただくために、特に以下の点にご注意くださいますようお願いいたします。

本製品には一般電子機器用（ＯＡ機器･通信機器･計測機器･工作機械等）に製造された半導体部品を使用していますので、その誤作動や故障が直接生命を脅かしたり、身体･財産等に危害を及ぼす恐れのある装置（医療機器･交通機器･燃焼制御･安全装置等）に組み込んで使用したりしないでください。また、半導体部品を使用した製品は、外来ノイズやサージにより誤作動したり故障したりする可能性があります。ご使用になる場合は万一誤作動、故障した場合においても生命･身体･財産等が侵害されることのないよう、装置としての安全設計（リミットスイッチやヒューズ･ブレーカ等の保護回路の設置、装置の多重化等）に万全を期されますようお願い申しあげます。

1.4.2. 取り扱い上の注意事項

劣化、破損、誤動作、発煙、発火の原因となることがあります。取り扱い時には以下のような点にご注意ください。

- 入力電源
  3.3V+5%以上の電圧を入力しないでください。極性を間違わないでください。

- インターフェース
  各インターフェース(外部 I/O、RS232C、Ethernet、JTAG)には規定以外の信号を接続しないでください。信号の極性を間違わないでください。
  信号の入出力方向を間違わないでください。

- 改造
  外部 I/O コネクタ及び JTAG コネクタ(CON2、CON3、CON4、CON5、CON7)にコネクタ等を増設する以外の改造は行わないでください。

- FPGA プログラム
  周辺回路(ボード上の部品も含む)と信号の衝突(同じ信号に 2 つのデバイスから出力する)を起こすような FPGA プログラムを行わないでください。
  FPGA のプログラムを間違わないでください。

- 電源の投入
  本ボードや周辺回路に電源が入っている状態では絶対に FPGA I/O、JTAG 用コネクタの着脱を行わないでください。

- 静電気
  本ボードには CMOS デバイスを使用していますので、ご使用になるまでは帯電防止対策のされている、出荷時のパッケージ等にて保管してください。

- ラッチアップ
  電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等で使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。

- 衝撃、振動
  落下や衝突などの強い衝撃を与えないでください。
  振動部や回転部などへの搭載はしないでください。強い振動や遠心力を与えないでください。

- 高温低温、多湿
  極度に高温や低温になる環境や、湿度が高い環境では使用はしないでください。

- 塵埃
  塵埃の多い環境では使用はしないでください。

1.4.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェア(付属のドキュメント等も含みます)は、現状のまま(AS IS)提供されるものであり、特定の目的に適合することや、その信頼性、正確性を保証するものではありません。また、本製品の使用による結果についてもなんら保証するものではありません。

1.5. 謝辞

SUZAKU で使用しているソフトウェアは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によってなりたっています。この場を借りて感謝の意を示したいと思います。

uClinux は D. Jeff Dionne 氏や Greg Ungere 氏、David McCulloughu 氏、さらに uClinux development list に参加しているすべての人の成果によって支えられています。

uClibc、Busybox は Eric Andersen 氏によって開発･保守されています。

MicroBlaze プロセッサアーキテクチャへの uClinux オリジナルポートは、John Williams 氏(オーストラリア･ブリスベンのクイーンズランド大学エンベデットシステムリサーチグループ所属)によるものです。

2. SUZAKU の概要

SUZAKU は汎用の小型 FPGA ボードです。このマニュアルでは SUZAKU の出荷時に搭載している 32bit ソフトプロセッサ MicroBlaze 上で動く uClinux について説明します。

2.1. MicroBlaze

MicroBlaze はハーバードアーキテクチャを採用した Xilinx 社製の 32 ビットソフトプロセッサです。基本的な RISC 型 CPU で、32bit インストラクションワード長、32 個の汎用レジスタ、IBM の On-chip Peripheral Bus 対応、Fast Simplex Link のサポート、ハードウェアマルチプレクサなどの特徴があります。

詳しくはXilinx 社のサイトをご覧ください。

2.2. uClinux

uClinux は Linux からの派生[1]カーネルです。Linux から MMU への依存を無くすることにより、MMU を持たない CPU 上でも Linux が動作します。D. Jeff Dionne 氏が MMU を持たない Motorola 製の 68000 に Linux を移植[2]したことから始まったプロジェクトです。現在はuClinux.orgで開発が進められています。

uClinux.org では、カーネルの他にツールチェーンや uClinux-dist と呼ばれるユーザランドアプリケーションの配布も行なっています。このマニュアルでは uClinux-dist の使用を前提とします。

MicroBlaze uClinux プロジェクトは John Williams 氏のプロジェクトで、MicroBlaze への uClinux の移植、Linux 用のツールチェーンの配布などを行なっています。

[1] 2.5 系の開発版で uClinux の成果が本家 Linux の方にもマージされました。
[2] D. Jeff Dionne: "Linux port to straight 68000 (no MMU)" http://lkml.org/lkml/1998/2/12/31

3. 作業の前に

3.1. 準備するもの

SUZAKU を使用する前に、次のものを準備して下さい。

- 作業用 PC
  ソフトウェア開発用として、Linux が動作し、シリアルポート(1 ポート)を持つ PC が必要です。

- D-Sub9 ピンクロスケーブル
  D-Sub9 ピン(メス−メス)の「クロス接続用」のケーブルです。

- D-Sub9 ピン-10 ピン変換ケーブル
  D-Sub9 ピンと本ボードのピンヘッダ(10 ピン)を接続するための、D-Sub9 ピン-10 ピン変換ケーブルを用意してください。

- 開発キット付属 CDROM(以降、「付属 CDROM」)
  SUZAKU に関する各種マニュアルやソースコードが収納されています。

- シリアル通信ソフト
  このマニュアルでは Minicom を使用します。

- DC3.3V 電源
  DC3.3V 出力のものを使用してください。

4. まず、動かしてみる

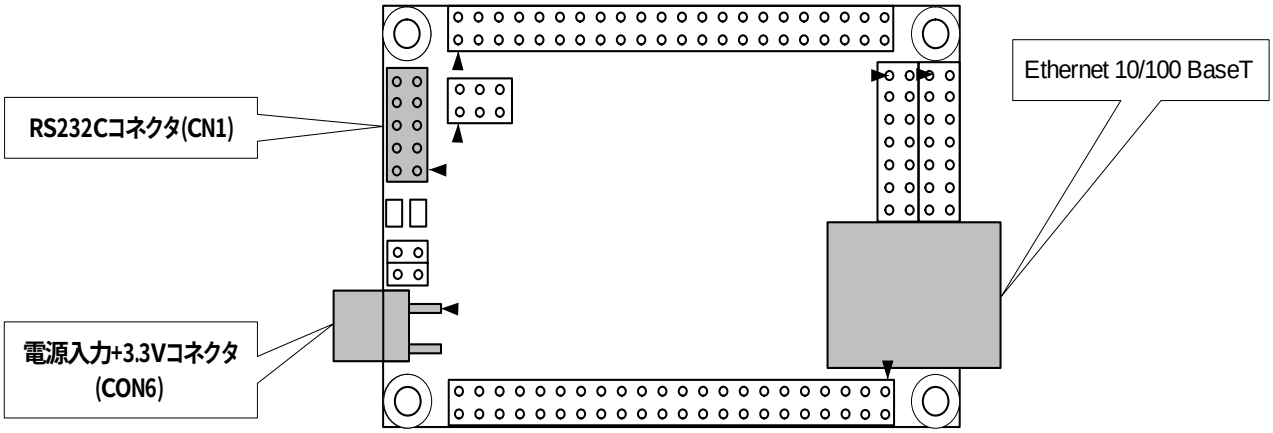
この章では、出荷状態の SUZAKU に各種ケーブルをつなぎ、SUZAKU を動かしてみます。出荷状態の SUZAKU にはソフトプロセッサの MicroBlaze、さらに MicroBlaze 用の uClinux が Flash メモリに入っています。出荷状態の SUZAKU は DHCP で IP を取得するように設定されています。お使いの環境に DHCP サーバがない場合は準備してください。

以下では SUZAKU にケーブルを接続する方法から、ウェブによるアクセスまでを順番に説明していきます。

4.1. 接続方法

必要なケーブルは、電源ケーブル、D-Sub9 ピン-10 ピン変換ケーブル、LAN ケーブルです。「図 4-1 SUZAKU のコネクタ配置図」を参照して、適切なコネクタに接続してください。SUZAKU 本体には電源スイッチはありません。他の準備が終るまで、電源のスイッチを切っておいてください。

図 4-1 SUZAKU のコネクタ配置図



注意

RS232C コネクタ(CON1)に D-Sub9 ピン-10 品変換ケーブルを接続する時、コネクタの白い三角マークと SUZAKU 基板上の白い三角マークを合わせるように接続します。コネクタの向きを反対に接続すると、機器を破損する恐れがありますので十分にご注意ください。

4.2. シリアル通信用ソフトウェア

SUZAKU はシリアルポートをコンソールとして使用します。SUZAKU のコンソールから出力される情報を読みとったり、SUZAKU のコンソールに情報を送ったりするには、シリアル通信用のソフトウェアが必要です。このマニュアルでは GPL で配布されている Minicom を使用します。付属 CDROM には Debian と Redhat 用のパッケージとソースコードを収録しています。付属 CDROM の **tools** ディレクトリを参照してください。インストールの方法は「7.4.Minicom のインストール」で詳しく説明します。Minicom がインストールされていない場合はそちらを参照してください。

インストールが完了したら Minicom の設定を行なってください。SUZAKU とのシリアル通信には以下の値を使用します。

表 4-1 シリアル通信設定

項目	設定
転送レート	115,200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

Minicom に**-s** オプションを付けて起動します。**-s** を指定することで設定画面に移行します。シリアルポートの項目を表 4-1 シリアル通信設定にそって変更します。

図 4-2 Minicom の設定

変更を保存して終了後、再度 Minicom を起動します。

TIPS

Minicom の初期設定では、起動時にモデムの初期化を行なうようになっていることが多いようです。設定で初期化用の AT コマンドを外すか、Minicom に**-o** オプションを付けて起動することでモデム初期化コマンドを省略することができます。

また、使用するシリアルポートの読み込みと書き込み権限が無い場合、Minicom の起動に失敗します。使用するシリアルポートの権限を確認してください。詳しくは Minicom のマニュアルまたはご使用の OS のマニュアルをご覧ください。

例 4-1 minicom の起動

```
[PC ~]$ minicom -o

Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Nov 12 2003, 19:21:57.

Press CTRL-A Z for help on special keys
```

4.3. 電源を入れる

起動モード選択ジャンパがオープンになっていることを確認します。出荷時の SUZAKU には複数の起動モードがあります。この起動モードを選択するのが起動モード選択ジャンパです。詳しくは「図 6-2 起動モードジャンパ」を参照してください。Minicom が起動されていることを確認してから、SUZAKU の電源を入れます。

SUZAKU の電源を入れると Minicom の画面に「例 4-2 uClinux の起動ログ」が表示されます。表示されない場合は「13.トラブルシューティング」を参照してください。

例 4-2 uClinux の起動ログ

```
Switching to 2nd stage bootloader...
Copying kernel...done.
Linux version 2.4.27-uc0-suzaku0 (atmark@build) (gcc version 2.95.3-4 Xilinx EDK 6.1 Build
EDK_G.11) #1 Sat Jan 29 16:23:31 JST 2005
On node 0 totalpages: 4096
zone(0): 4096 pages.
zone(1): 0 pages.
zone(2): 0 pages.
CPU: MICROBLAZE
Console: xmbserial on UARTLite
Kernel command line:
Calibrating delay loop... 25.44 BogoMIPS
Memory: 16MB = 16MB total
Memory: 13336KB available (1037K code, 1824K data, 44K init)
Dentry cache hash table entries: 2048 (order: 2, 16384 bytes)
Inode cache hash table entries: 1024 (order: 1, 8192 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 4096 (order: 2, 16384 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Microblaze UARTlite serial driver version 1.00
ttyS0 at 0xffff2000 (irq = 1) is a Microblaze UARTlite
Starting kswapd
xgpio #0 at 0xFFFF5000 mapped to 0xFFFF5000
Xilinx GPIO registered
SMSC LAN91C111 Driver (v2.0), (Linux Kernel 2.4 + Support for Odd Byte) 09/24/01
 -      by Pramod Bhardwaj (pramod.bhardwaj@smsc.com)
eth0: SMC91C11xFD(rev:1) at 0xffe00300 IRQ:2 MEMSIZE:8192b NOWAIT:0 ADDR: 00:11:0c:00:15:55
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
Suzaku MTD mappings:
  Flash 0x800000 at 0xff000000
flash: Found an alias at 0x400000 for the chip at 0x0
 Amd/Fujitsu Extended Query Table v1.1 at 0x0040
```

```
flash: Swapping erase regions for broken CFI table.
number of CFI chips: 1
cfi_cmdset_0002: Disabling fast programming due to code brokenness.
Creating 7 MTD partitions on "flash":
0x00000000-0x00400000 : "Flash/All"
0x00000000-0x00080000 : "Flash/FPGA"
0x00080000-0x000a0000 : "Flash/Bootloader"
0x003f0000-0x00400000 : "Flash/Config"
0x000a0000-0x003f0000 : "Flash/Image"
0x000a0000-0x00210000 : "Flash/Kernel"
0x00210000-0x003f0000 : "Flash/User"
FLASH partition type: auto 4MiB
uclinux[mtd]: RAM probe address=0x8013cb88 size=0x18f000
uclinux[mtd]: root filesystem index=7
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 1024)
VFS: Mounted root (romfs filesystem) readonly.
Freeing init memory: 44K
Mounting proc:
Mounting var:
Populating /var:
Running local start scripts.
Mounting /etc/config:
Populating /etc/config:
flatfsd: Created 4 configuration files (150 bytes)
Setting hostname:
Setting up interface lo:
Starting DHCP client:
Starting inetd:
Starting thttpd:
eth0:PHY remote fault detected

SUZAKU login:
```

4.4. ログイン

Minicom に表示されている SUZAKU のログインプロンプトから root ユーザでログインします。パスワードの初期設定は「root」です。

表 4-2 SUZAKU 初期設定時のユーザとパスワード

ユーザ名	パスワード
root	root

4.5. コマンド入力

コマンド入力例として、**ifconfig** コマンドでネットワーク設定を表示してみます。SUZAKU の初期ネットワーク設定では dhcp を使用するようになっています。

例 4-3 ifconfig での表示

```
[SUZAKU ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:11:0C:00:10:00
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:347 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          Interrupt:2 Base address:0x300

[SUZAKU ~]#
```

4.6. ウェブ

初期設定時の SUZAKU では、thttpdという小さな HTTP サーバが起動しています。「例 4-3 **ifconfig での表示**」で表示された IP アドレス(例では 192.168.1.10)にお使いのウェブブラウザでアクセスすることで、動作確認ができます。

図 4-3 SUZAKU Web Page

4.7. 終了方法

SUZAKUではルートファイルシステムを読み込み専用にすることで、電源即断時のルートファイルシステムの破損を回避しています。このため通常 SUZAKU を終了する場合は電源を切るだけで終了することができます。ただし、SUZAKUが設定ファイルを保存するために採用している Flat Filesystemでは、情報を書き出している間の電源断には対応していません。電源を切ることによって Flat Filesystem 上のデータを失う可能性があります。

詳しくは Flat Filesystem や Flat Filesystem Daemon のマニュアルをご覧ください。

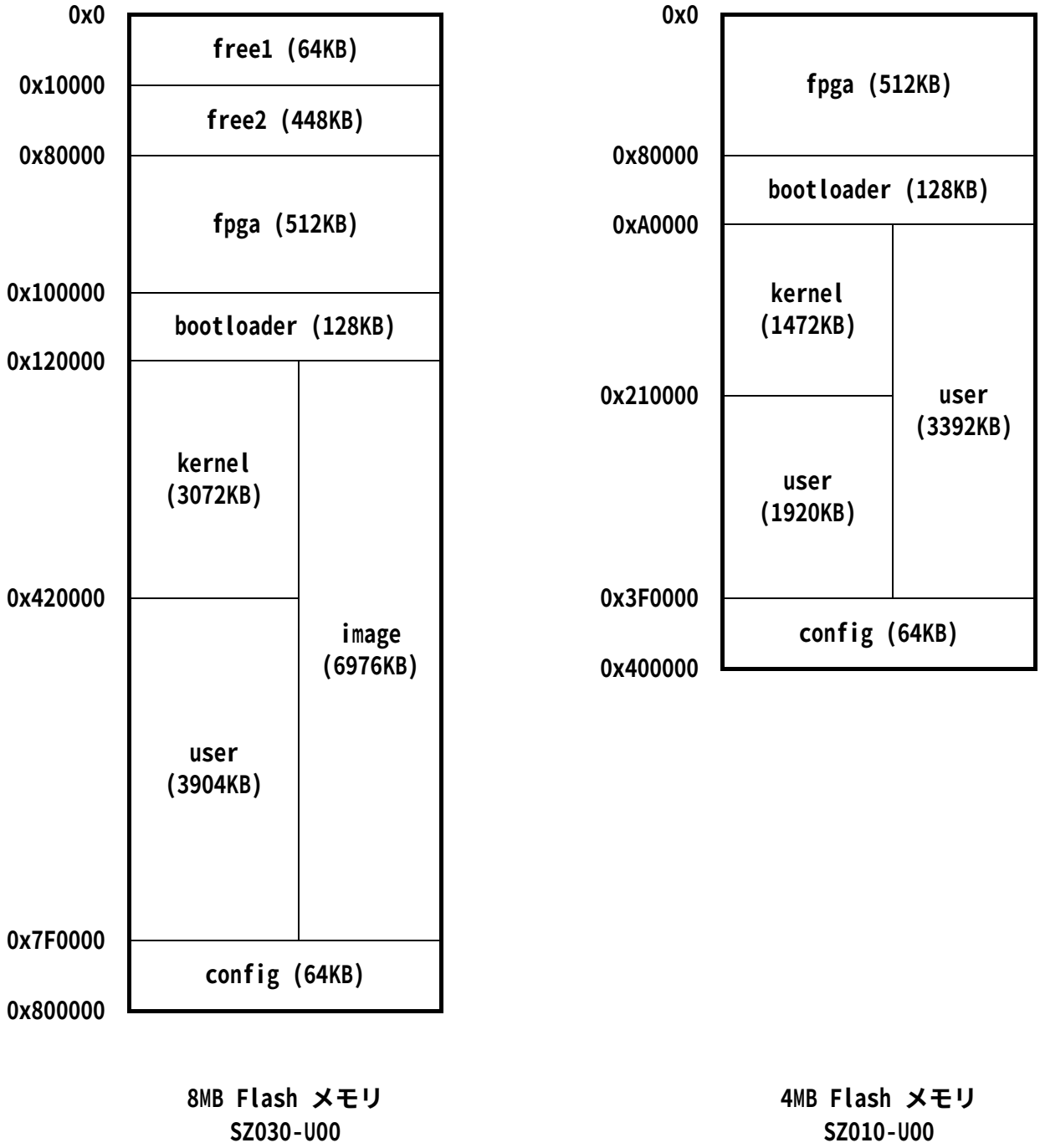
注意

SUZAKU をカスタマイズすることでルートファイルシステムを出荷時の ROMFS から JFFS2 に変更することが可能ですが、この場合電源を切ることによって保存したはずのデータを失う可能性があります。詳しくは JFFS2 のマニュアルをご覧ください。

5. Flash メモリのメモリマップ

SUZAKU の Flash メモリは「表 5-1 Flash メモリマップ」のようにリージョンと呼ばれる複数の領域に分割されています。搭載されている Flash メモリのサイズによって領域の区分が異なります。SUZAKU 上でソフトウェアを使うために便宜的に分割しています。

表 5-1 Flash メモリマップ



危険

fpga のリージョンには、FPGA のコンフィグレーションデータが入っています。この領域に不適切なデータを書き込んだ場合、SUZAKU の異常動作により SUZAKU 及び周辺機器が発熱、劣化、破損する可能性があります。正常に動作させるためには、FPGA コンフィグレーションデータの再プログラミングが必要になります。詳しくは Hardware Manual を参照してください。

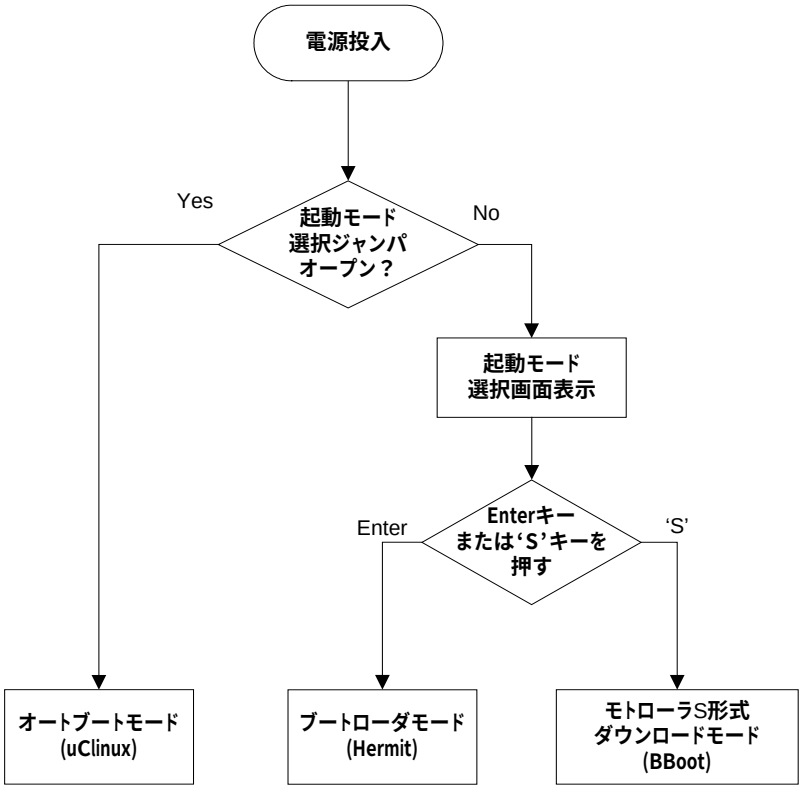
6. 起動の流れと起動モード

初期設定時の SUZAKU では 3 つの起動モードを持っています。また、SUZAKU 上で uClinux が起動するまでには、2 つのブートローダが動いています。この章ではまず簡単に起動の流れを説明します。その後、3 つの起動モードについて説明します。

6.1. 起動の流れ

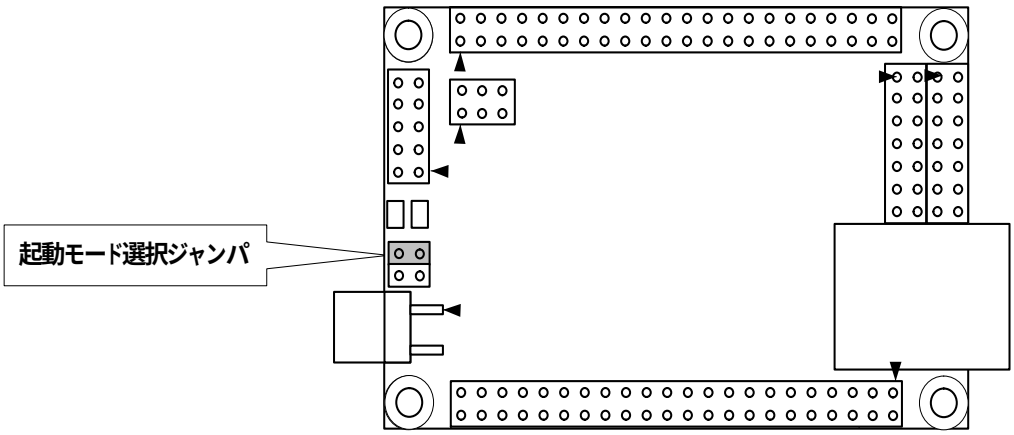
まず起動の流れを「図 6-1 起動モードの流れ」に示します。

図 6-1 起動モードの流れ



起動の流れを決めているのは、SUZAKU 上にある起動モードジャンパです。起動モードジャンパの位置は「図 6-2 起動モードジャンパ」を参照してください。このジャンパがオープンの場合は uClinux が起動するオートブートモードになっています。このジャンパがショートの場合はブートモード選択画面に移行します。

図 6-2 起動モードジャンパ



SUZAKU は起動モードジャンパがオープン、つまりオートブートモードで出荷されています。

6.2.起動モード

SUZAKU には以下の 3 つの起動モードがあります。以下では各モードについて説明します。

6.2.1. オートブートモード

このモードでは SUZAKU の起動時に uClinux が自動的に起動されます。

6.2.2. ブートローダモード

このモードはブートローダのみを起動する場合に使用します。出荷時の SUZAKU には Hermit と呼ばれる高機能なブートローダ兼ダウンローダが Flash メモリ内に入っています。このモードでは Hermit がユーザからの入力待ち状態になります。Hermit にコマンドを与えることにより、Memory 内部の情報を読み出したり、uClinux のイメージを Flash メモリに書き込むことができます。

ブートローダモードに移行するには、起動モードジャンパをショートにして、「例 6-1 ブートモード選択画面」にあるように、ブートモード選択画面でエンターキーを入力します。SUZAKU のデフォルトブートローダである Hermit が起動すると「例 6-2 Hermit の起動」のような画面になります。

例 6-1 ブートモード選択画面

```
Welcome to BBoot v1.1 (microblaze)
SUZAKU's first stage bootloader and S-Record downloader

hit Enter key to activate second stage bootloader or
hit 'S' key to download S-Record
```

例 6-2 Hermit の起動

```
Hermit v1.3-armadillo-8 compiled at 15:39:32, Jan 29 2005
hermit>
```

Hermit を使って Flash メモリにデータを書き込む場合には一度 SUZAKU をこのモードで起動しておかなければなりません。詳しい Flash メモリの書き換え方法は「12.2Hermit を使用した Flash メモリの書き換えかた」をご覧ください。

6.2.3. モトローラ S 形式ダウンロードモード

このモードは Flash メモリ上のブートローダが何らかの理由で動作しなくなった場合に、ブートローダを再書き込みするために使用します。このモードへの移行は、起動モード選択ジャンパをショートにして、「例 6-1 ブートモード選択画面」に移動した後キーボードから「S」を入力します。

出荷時の SUZAKU には BBoot と呼ばれるモトローラ S 形式ダウンローダが入っています。この BBoot は Flash メモリではなく FPGA の Block RAM と呼ばれる領域に入っていますので、誤って uClinux や Hermit を破壊してしまっても MicroBlaze が動いている状態ならば使用できます。また、BBoot は SUZAKU の起動時に最初に動くプログラムのため、ファーストステージブートローダとも呼ばれます。

BBoot を使ったモトローラ S 形式の詳しい書き込み方法については「12.3モトローラ S 形式での Flash メモリの書き換えかた」をご覧ください。

7. 開発環境の準備

この章ではMicroBlaze uClinuxの開発環境を準備します。**/usr/local/microblaze-elf-tools/**にツールチェーンを展開、PATH 環境変数の設定、ソースコードの準備を行ないます。

7.1. ツールチェーンのインストール

ツールチェーンは付属 CDROM の **cross-dev** ディレクトリに入っています。また MicroBlaze uClinux プロジェクトのホームページでも配布しています。

ツールチェーンは gzip で圧縮された tar アーカーブ形式になっています。**/usr/local/microblaze-elf-tools/**に展開してください。

ここでは付属 CDROM からインストールする場合の一連のコマンドを示します。

```
[PC ~]$ su -
[PC ~]# mkdir -p /usr/local/microblaze-elf-tools/
[PC ~]# cd /usr/local/microblaze-elf-tools/
[PC microblaze-elf-tools]# mount /cdrom
[PC microblaze-elf-tools]# tar -xvzf /cdrom/cross-dev/microblaze-elf-tools-20040315.tar.gz
[PC microblaze-elf-tools]# ls
bin  include  info  lib  man  microblaze  share
[PC microblaze-elf-tools]# exit
[PC ~]$
```

7.2. 環境変数の設定

MicroBlaze 用のツールチェーンを使いやすくするために、ツールチェーンの実行ファイルが入っているディレクトリを **PATH** 環境変数に追加します。シェルによって設定方法が異なりますので、詳しくはお使いのシェルのマニュアルを参照してください。

ここでは **bash** の設定方法を例に取ります。

```
[PC ~]$ export PATH=$PATH:/usr/local/microblaze-elf-tools/bin
[PC ~]$
```

7.3. ソースコードの準備

uClinux.org ではカーネル、ライブラリ、アプリケーションのソースコードをまとめて Full Source Distribution として配布しています。uClinux.org で配布している Full Source Distribution のファイル名は **uClinux-dist-YYYYMMDD.tar.gz** となっています。SUZAKU 用にはこの uClinux-dist にいくつかの修正を加えたものを作成しています。SUZAKU 用の uClinux-dist は CDROM の **source** ディレクトリに **uClinux-dist.tar.gz** というファイル名で入っています。また、日々更新されているソースコードの追従を容易にするため、SUZAKU 用 uClinux-dist には uClinux.org の CVS 情報を残しています。

uClinux-dist.tar.gz はどこに展開しても問題はありません。作業のしやすい場所に展開してください。このマニュアルでは**~/**に展開するものとして説明します。

```
[PC ~]$ su -
[PC ~]# mount /cdrom
[PC ~]# exit
[PC ~]$ tar -xvzf /cdrom/source/uClinux-dist-YYYYMMDD-suzakuX.tar.gz
    :
    :
[PC ~]$ ls
uClinux-dist
[PC ~]$
```

7.4. Minicom のインストール

以下に Debian と Redhat でのインストール手順を例として示します。
詳しくはお使いの OS のマニュアルを参照してください。

例 7-1 Minicom のインストール (Debian で dpkg を使用)

```
[PC /cdrom/tools]# dpkg -i minicom_2.1-4.woody.1_i386.deb
[PC /cdrom/tools]#
```

例 7-2 Minicom のインストール (Redhat で rpm を使用)

```
[PC /cdrom/tools]# rpm -i minicom_2.1-1-rh7.3.i386.rpm
[PC /cdrom/tools]#
```

7.5. Hermit のインストール

作業用 PC に Hermit をインストールします。Hermit は SUZAKU 上で動作するターゲット側と、作業用 PC 上で動作し SUZAKU 上のプログラムと通信するホスト側の二つのプログラムがあります。ターゲット側の Hermit は uClinux を起動するブートローダとしても動作する重要なプログラムです。ホスト側の Hermit は SUZAKU の Flash メモリの書き換え時に使用します。

付属 CD の **tools** ディレクトリに Hermit のパッケージが入っています。お使いの OS にあわせてインストールしてください。以下に Debian と Redhat の例を示します。

例 7-3 Hermit のインストール (Debian)

```
[PC /cdrom/tools]# dpkg -i hermit_1.3-armadillo-8_i386.deb
[PC /cdrom/tools]#
```

例 7-4 Hermit のインストール (Redhat)

```
[PC /cdrom/tools]# rpm -i hermit-1.3_armadillo-8.i386.rpm
[PC /cdrom/tools]#
```

8.uClinux の image を作成

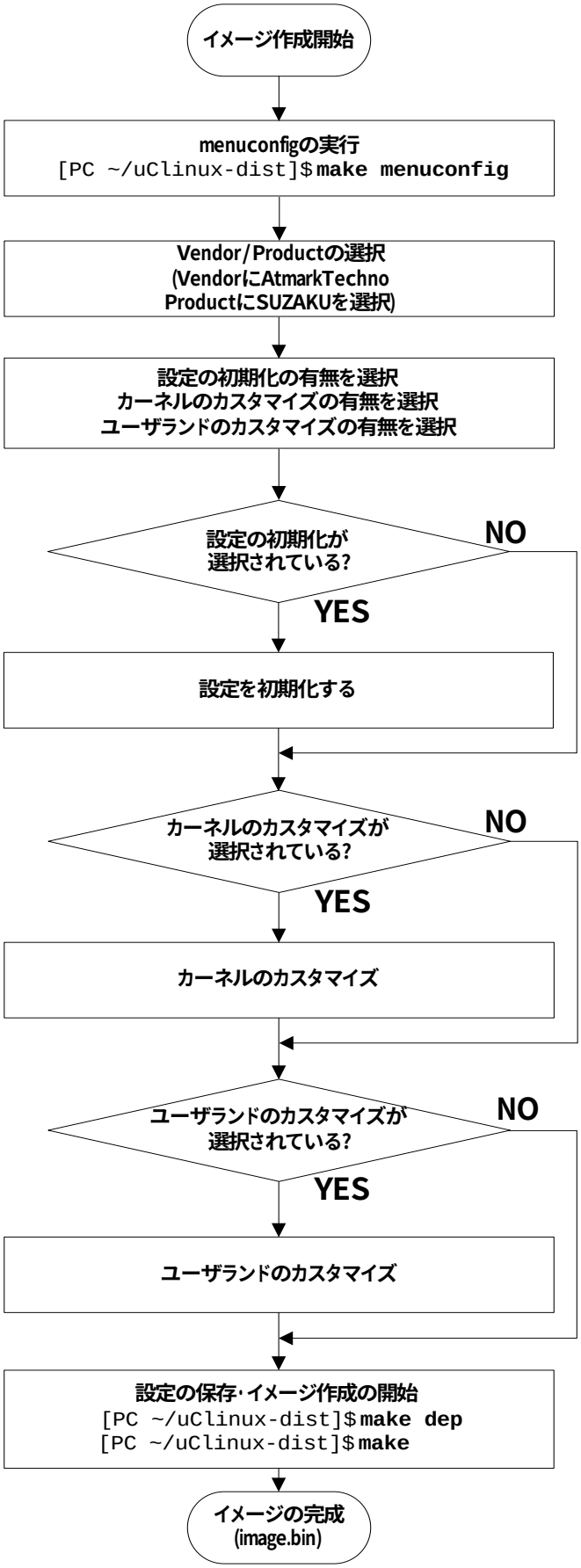
この章では実際にインストールしたツールチェーンとソースコードを使って Flash メモリに書き込むためのイメージファイルを作成してみます。uClinux-dist では Linux Kernel のビルドシステムと同じ方法でカスタマイズできるようになっています。Linux Kernel を自分でコンパイルしている人にとっては扱いやすいと思います。このマニュアルでは、menuconfig を例にとって説明していきます。

最初は、出荷時に SUZAKU の Flash メモリに書かれているデフォルトのイメージファイルを作成します。

8.1.作業の流れ

まず、「図 8-1 イメージファイル作成の流れ」にイメージファイル作成の流れを示します。

図 8-1 イメージファイル作成の流れ



作業は大きく初期化とカーネル、ユーザランドの 3 つに分られます。

注意

初期化を選択するとカーネルとユーザランドの両方が初期化されてしまいます。また、初期化を選ぶ前までの作業は破棄されてしまいますので、注意が必要です。

8.2. Menuconfig

Linux カーネルと同じく uClinux-dist のビルドシステムでは config、menuconfig、xconfig が使用できます。ここでは menuconfig を使って uClinux-dist のビルドを行ないます

まず、「7.3.ソースコードの準備」で展開した uClinux-dist ディレクトリに移動して **make** menuconfig とコマンドを入力してください。

例 8-1 make menuconfig の実行

```
[PC ~]$ cd uClinux-dist
[PC ~/uClinux-dist]$ make menuconfig
```

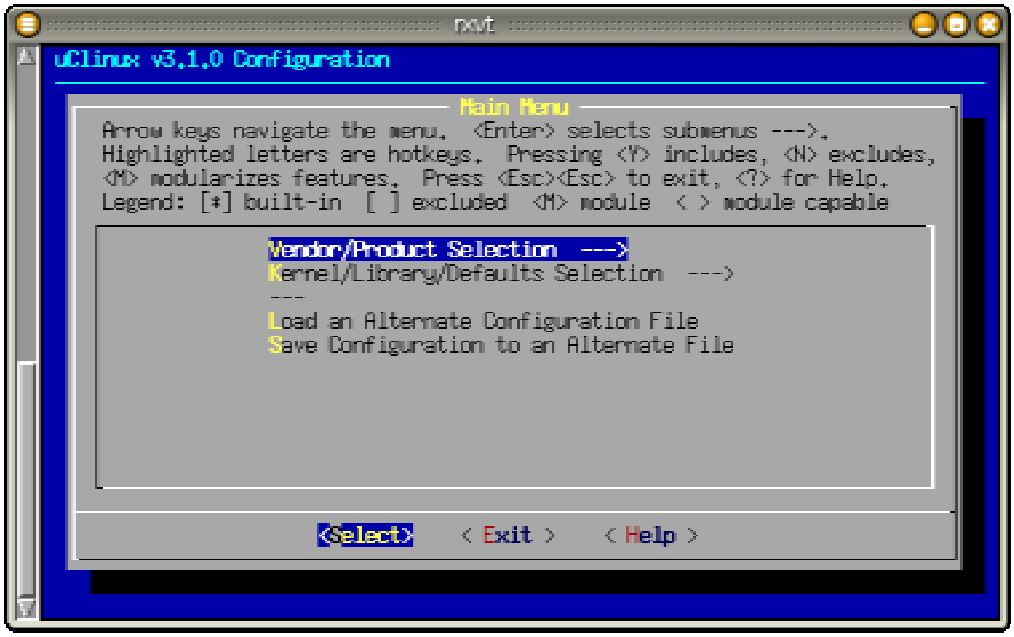
注意

make menuconfig では最初の実行時に ncurses ライブラリを必要とする画面制御プログラムをコンパイルします。このため ncurses ライブラリがインストールされている必要があります。OS によっては ncurses ライブラリパッケージだけではなく、同ライブラリの開発用パッケージをインストールする必要があります。

8.2.1. メインメニュー

コマンドを入力した画面が「図 8-2 uClinux v3.1.0 Configuration Main Menu」のようになります。この画面が uClinux-dist のメインメニューです。

図 8-2 uClinux v3.1.0 Configuration Main Menu



menuconfig ではキーボードのカーソルキーを使って選択メニューの移動を行ないます。選択メニューの右側が「**--->**」となっている場所はサブメニューへ移動できることを表わします。メニュー選択時には画面下の「**<select>**」がハイライトされていることを確認してからエンターキーを押します。

この画面では、Vendor/Product の選択メニューへの移動とカーネル、ライブラリ、初期値選択メニューへの移動が可能です。

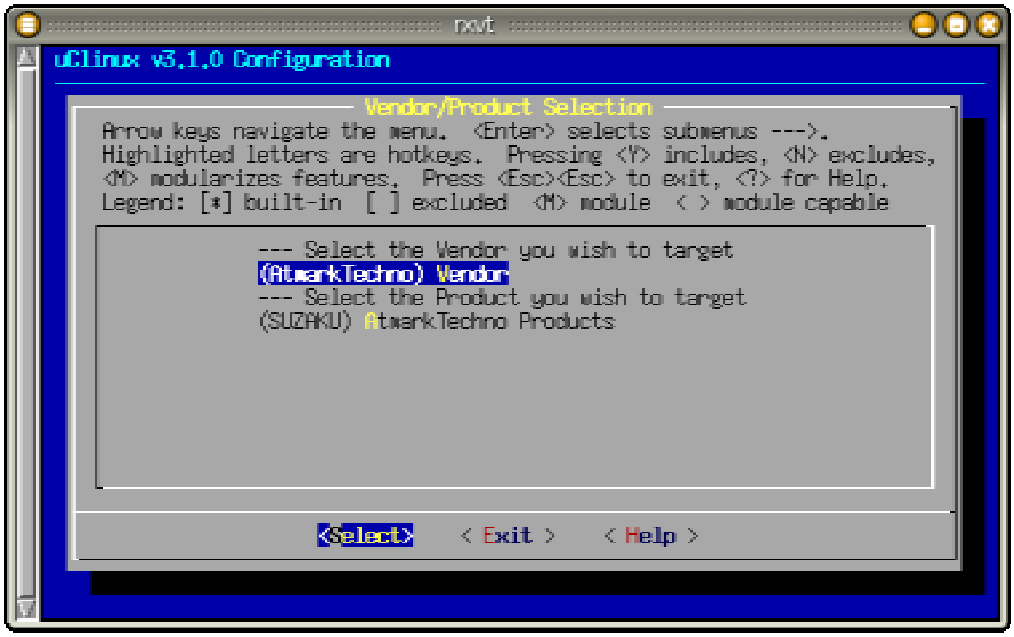
8.2.2. Vendor/Product メニュー

メインメニューから Vendor/Product を選択して、サブメニュー画面へ移動します。

デフォルトでは「**SnapGear**」が選択されています。「**SnapGear (Vendor)**」にカーソルをあわせてエンターキーを押すことで他のベンダーを選択できます。リストの中から **AtmarkTechno** を選択してください。

「図 8-3 AtmarkTechno を選択」のように **AtmarkTechno** を選択することで、**Product** で **SUZAKU** が選択されます。

図 8-3 AtmarkTechno を選択



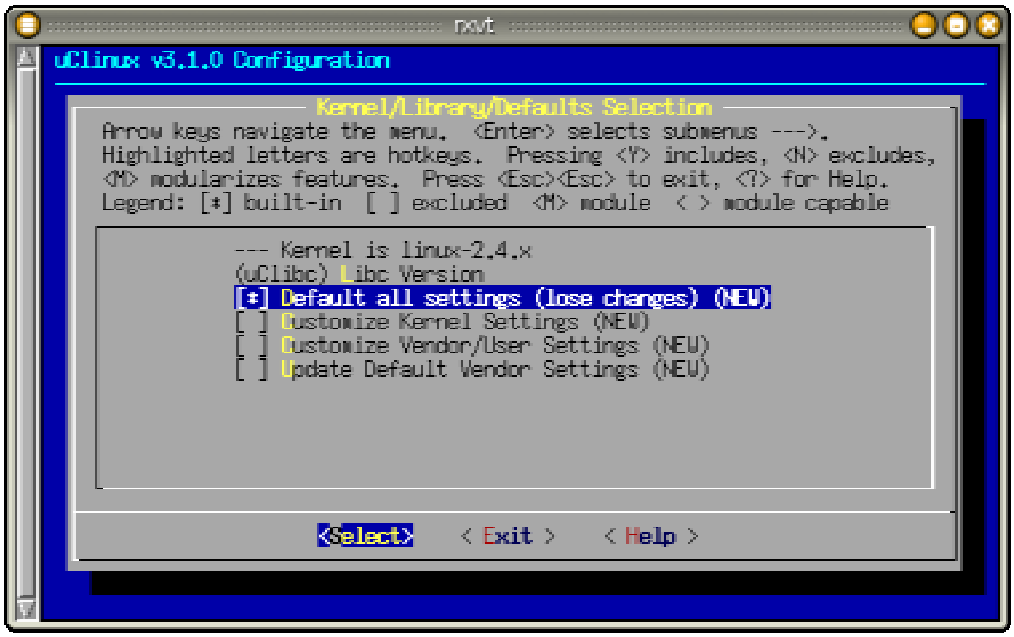
カーソルキー使って **Exit** を選択して、メインメニューに戻ります

8.2.3. Kernel/Library/Defaults メニュー

メインメニューから **Kernel/Library/Defaults Selection** を選択し、サブメニューに移動します。

メニューの左側にある[ ]は選択状態を表します。今回は「**Default all settings (lose changes) (NEW)**」を選択してください。選択するにはカーソルを上下キーで動かし、スペースバーで選択します。選択すると[ ]の中に「*」が表示されます。

図 8-4 Kernel/Library/Defaults Selection



Default all settings (lose changes) を選択後、**Exit** を選択しメインメニューに戻ります。メインメニューで **Exit** を選択すると、設定を保存するか聞かれますので **Yes** を選択して保存します。

画面上に設定のログが流れた後、プロンプトに戻ります

8.3. ビルド

実際にコンパイルを行なったり、イメージファイルを生成することをビルドすると言います。ビルドには **make** を使用します。「例 8-2 ビルドコマンド」のようにプロンプトで入力します

例 8-2 ビルドコマンド

```
[PC ~/uClinux-dist]$ make dep; make
```

最初の **make dep** は Linux Kernel の依存関係を解決するためのコマンドです。2.4 系までの Linux Kernel コンパイル経験者には見なれたコマンドだと思います。

次の **make** コマンドは全てのビルドプロセスを行ない、最終的に Flash メモリに書き込むことのできるイメージファイルを生成してくれます。ビルドプロセスが問題なく終了すると、**uClinux-dist/images/**ディレクトリに **image.bin** というファイルが生成されます。

実際に生成されたイメージファイルを SUZAKU に書き込む方法は「12.Flash メモリの書き換え方法」を参照してください。

[3] ここでのイメージファイルとは、uClinux カーネルとユーザランドが一つにまとまっているバイナリファイルを意味します。このバイナリファイルを Hermit を使って Flash メモリに書き込むことにより、SUZAKU 上で uClinux が起動します。

9.uClinux の設定変更

この章では uClinux のイメージのカスタマイズを行ないます。uClinux-dist のメニューを使ったカスタマイズが中心です。

9.1.カーネルの設定変更

カーネルのカスタマイズを行なう場合は、uClinux Configuration の Kernel/Library/Defaults Selection メニューで **Customize Kernel Settings** を選択します。このメニューを選択し uClinux v3.1.0 Configuration のメインメニューを終了させると、自動的に Linux Kernel Configuration のメニュー画面が表示されます。Linux Kernel Configuration 自体は本家 Linux と同じものです。

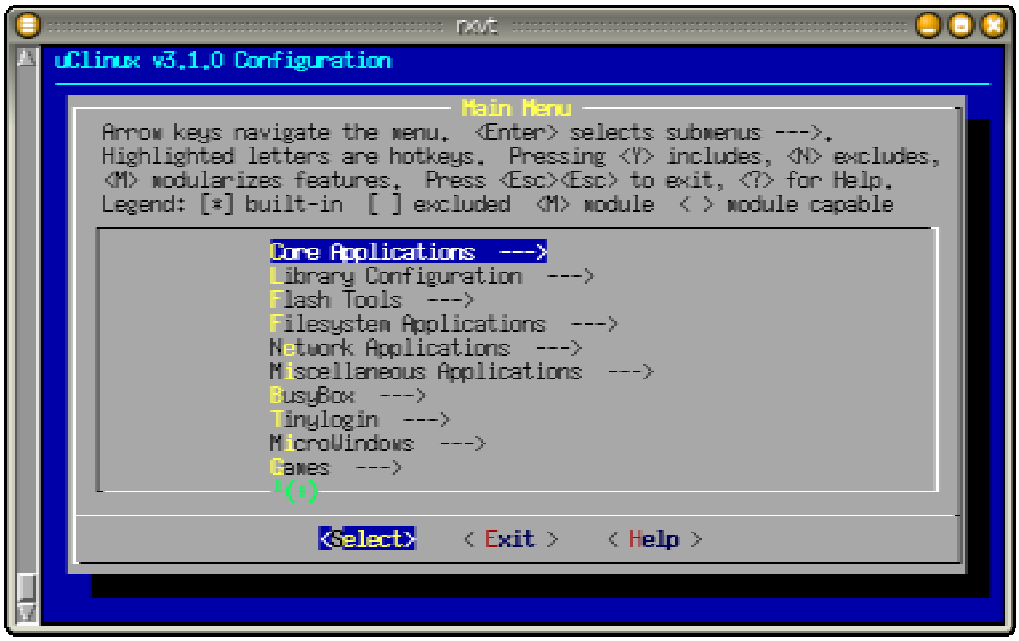
カーネルをカスタマイズ後、**Exit** を選んでください。最後に設定を保存するか聞かれますので **Yes** を選択してください。

ユーザランドのカスタマイズが無い場合は、ビルドを行ないます。ビルドについては「8.3.ビルド」を参照してください。

9.2.ユーザランドの設定変更

ユーザランドのカスタマイズを行なう場合は、uClinux Configuration の Kernel/Library/Defaults Selection メニューで Customize Vendor/UserSettings を選択します。このメニューを選択し uClinux v3.1.0Configuration のメインメニューを終了させるとユーザランド用の Main Menu になります。

図 9-1 ユーザランド用 Main Menu



ユーザランドのカスタマイズ画面では、uClinux Full Source Distribution で配布されているアプリケーションを生成するイメージに追加するか否か、または追加するアプリケーションのカスタマイズを行ないます。さらに root ユーザのパスワードなどもここで変更することができます。

uClinux-dist には 150 以上ものアプリケーションが集められています。このため、uClinux Configuration のメニューではアプリケーションをいくつかのカテゴリに分類しています。

- Core Application

  システムとして動作するために必要な基本的なアプリケーションが入っています。システムの初期化を行なう init やユーザ認証の login などがこのセクションで選択できます。

- Library Configuration

  アプリケーションが必要とするライブラリの選択ができます。

- Flash Tools

  Flash メモリに関係のあるアプリケーションが選択できます。SUZAKU では、netflash と呼ばれるネットワークアップデート用アプリケーションがここで選択されています。

- Filesystem Applications

  ファイルシステムに関係のあるアプリケーションが選択できます。SUZAKU では flatfsd を選択しています。その他、mount、fdisk、ext2 ファイルシステム、reiser ファイルシステム、Samba などが含まれます。

- Network Applications

  ネットワークに関係のあるアプリケーションが選択できます。SUZAKU で使用している dhcpcd-new、ftpd、ifconfig、inetd、thttpd の他にも ppp やワイヤレスネットワークのユーティリティなども含まれます。

- Miscellaneous Applications

  上記のカテゴリに属さないアプリケーションが収められています。Unix の一般的なコマンド(cp、ls、rm 等)やエディタ、オーディオ関連、スクリプト言語インタプリタなどが含まれます。

- Busybox

  Busybox のカスタマイズを行ないます。Busybox は複数のコマンド機能をもった単一コマンドで多くの組み込み Linux での実績を持っています。Busybox はとても多くカスタマイズできるため、別セクションになっています。

- Tinylogin

  Tinylogin も複数コマンドの機能をもつアプケーションです。login や passwd、getty など認証に関係のある機能を提供します。多くのカスタマイズが可能なため別セクションになっています。

- MicroWindows

  MicroWindows は組み込み機器をターゲットにしたグラフィカルウインド環境です。LCD などを持つ機器を開発する場合に使えます。

- Game

  ゲームです。説明はいらないでしょう。

- Miscellaneous Configuration

  いろいろな設定がまとめられています。SUZAKU の root ユーザのパスワードもここで変更できます。

- Debug Builds

  デバッグ用のオプションがまとめられています。開発中にアプリケーションのデバッグを行なうときに選択します。

すべてのアプリケーションがすべてのアーキテクチャで動作することは保証されていませんが、多くの場合ほんのわずかな変更(ソースコードレベルであったり、Makefile の変更だったり)で動作させることが可能です。

多くの開発者が変更などの議論を uClinux.org のメーリングリストで行っています。興味のあるかたは是非参加してください。

9.3. ZFLAT 対応カーネルを作る

実際に ZFLAT に対応したカーネルを作ってみます。ZFLAT とは圧縮された FlatBinary Format の実行ファイルです。Flat Binary Format については、「11.Flat Binary Format」を参照してください。

uClinux-dist のディレクトリで **make menuconfig** を実行し、Kernel/Library/Defaults Selection メニューから Customize Kernel Settings を選択します。メインメニューから Exit を選択します。設定を変更するか聞かれますので **Yes** を選択します。

Linux Kernel Configuration のメインメニューの中から、**General setup** に移動します。**Kernel support for flat binaries** の下にある **Enable ZFLAT support** を選択します。メインメニューから **Exit** を選択し、設定を保存し終了します。

make clean dep all を実行し、**image.bin** を生成します。生成されたイメージのカーネルでは ZFLAT ファイルを実行することができます。

Flat Binary Format 実行ファイルの圧縮方法は「11.2.実行ファイルの圧縮」を参照してください。

10. 自作アプリケーションをイメージに入れる

この章では Hello World を作って SUZAKU 上で動かしてみます。コンパイルは uClinux-dist のパッケージの外(これを OTC: Out of Tree Compile と呼びます)で行ないます。サンプルコードは付属 CDROM の **sample** に入っています。

uClinux-dist の外でコンパイルする場合でも、uClinux-dist で提供されているライブラリや Makefile を使用しますので、一度 SUZAKU 用にビルドした uClinux-dist ディレクトリが必要です。まだ一度もビルドしていない場合は「8.uClinux の image を作成」を参照しビルドしてください。

10.1. ディレクトリの作成

実際に作業を行なう場所はどこでもかまいません。ここでは**~/hello** を使用します。

例 10-1 Hello World 用のディレクトリを用意する

```
[PC ~]$ mkdir hello
[PC ~]$
```

10.2. Makefile の作成

Makefile のサンプルテンプレートが付属 CDROM の **sample/hello/Makefile** にありますので、このファイルをコピーします。

例 10-2 Hello World Makefile

```
ifndef ROOTDIR
ROOTDIR=../uClinux-dist                                  ①
endif
ROMFSDIR  = $(ROOTDIR)/romfs
ROMFSINST = romfs-inst.sh
PATH      := $(PATH):$(ROOTDIR)/tools

UCLINUX_BUILD_USER = 1                                   ②
include $(ROOTDIR)/.config
LIBCDIR = $(CONFIG_LIBCDIR)
include $(ROOTDIR)/config.arch


EXEC = hello                                             ③
OBJS = hello.o                                           ④


all: $(EXEC)

$(EXEC): $(OBJS)
          $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
          -rm -f $(EXEC) *.elf *.gdb *.o

romfs:
          $(ROMFSINST) /bin/$(EXEC)

%.o: %.c
          $(CC) -c $(CFLAGS) -o $@ $<                    ⑤
```

① **ROOTDIR** が指定されていない場合、現在のディレクトリと並列に uClinux-dist ディレクトリがあると仮定します。他の場所の場合は **ROOTDIR** を指定してください。
② **UCLINUX_BUILD_USER** を定義します。この変数を定義することで、ユーザランド用のコンパイラオプションが選ばれます。
③ 生成される実行ファイル名を変数 **EXEC** に設定します。ここでは **hello** とします。
④ 生成する実行ファイルに使用するオブジェクトファイルを指定します。複数ある場合はスペースで区切って羅列します。
⑤ C ソースコードを同名のオブジェクトコードに変換するパターンルールです。詳しくは Make のマニュアルをご覧ください。

10.3. C ソースコード

Hello World の C コードを「例 10-3 Hello World ソースコード」に示します。

例 10-3 Hello World ソースコード

```
#include <stdio.h>

int main(int argc, char * argv[])
{
        printf("Hello SUZAKU\n");
        return 0;
}
```

どの C の教科書にでも載っている、普通の Hello World です。これを、**hello.c** として保存します。Makefile に従い **hello.o** にコンパイルされます。

10.4. コンパイル

以下のように **make** します。**make** に成功すると **hello.o**, **hello.gdb**, **hello** の 3 つのファイルが生成されます。**hello** が実行ファイルです。

例 10-4 make の実行

```
[PC ~/hello]$ make
[PC ~/hello]$ ls
Makefile  hello  hello.c  hello.gdb  hello.o
[PC ~/hello]$
```

10.5. ROMFS ディレクトリへのインストール

イメージファイルのユーザ領域は、**uClinux-dist/romfs** ディレクトリから作成されます。そのため自作のアプリケーションを Flash メモリに書き込むイメージに含めるためには、アプリケーションを **romfs** ディレクト内にコピーする必要があります。

今回のように生成される実行ファイルが一つであればコピーするのも手間ではありませんが、複数の実行ファイルを生成したり、設定ファイルやデータファイルもコピーするときは大変です。このため、Makefile のテンプレートでは romfs ターゲットを持っています。

例 10-5 Hello World を ROMFS ディレクトリへインストール

```
[PC ~/hello]$ make romfs
romfs-inst.sh /bin/hello
[PC ~/hello]$ ls ../uClinux-dist/romfs/bin/hello
../uClinux-dist/romfs/bin/hello
[PC ~/hello]$
```

10.6. イメージファイルの生成と実行

最後に uClinux-dist のディレクトリに移動して、イメージファイルを生成します。

make コマンドで image ターゲットを指定すると、romfs ディレクトリを元にユーザランドのイメージファイル(**romfs.img**)を作成した後、カーネルのイメージファイル(**linux.bin**)と結合して Flash メモリに書き込むイメージファイル(**image.bin**)を生成します。

例 10-6 イメージファイル生成

```
[PC ~/hello]$ cd ../uClinux-dist
[PC ~/uClinux-dist]$ make image
    :
    :
    :
[PC ~/uClinux-dist]$ ls images
image.bin  linux.bin  romfs.img
[PC ~/uClinux-dist]$
```

生成されたイメージファイルを SUZAKU にダウンロードし、**hello** を実行します。ダウンロードの方法については「12.Flash メモリの書き換え方法」を参照してください。

例 10-7 実行

```
[SUZAKU ~]# hello
Hello SUZAKU World!
[SUZAKU ~]#
```

11. Flat Binary Format

この章では uClinux の特徴の一つである Flat Binary Format について説明します。uClinux が対象としている組み込み機器では実行バイナリーの大きさは大きな問題です。一般の Linux が採用している ELF は柔軟性に富んだフォーマットですがサイズが大きすぎる問題がありました。そこで多くの uClinux では昔ながらの a.out フォーマットに似た新たなバイナリーフォーマットを採用しています。

はじめに Flat Binary Format の特徴を説明した後、実行ファイルの圧縮方法とスタックサイズの変更方法を説明します。

11.1. Flat Binary Format の特徴

Flat Binary Format には以下のような特徴があります。

- シンプル
  シンプルな設計はバイナリファイルの実行速度と大きさに貢献しています。もちろん ELF のような柔軟性は減りますが、組み込み機器にとっては必要なトレードオフと言えます。

- 圧縮可能
  Flat Binary Format は圧縮可能なフォーマットです。圧縮には 2 種類あり、ファイル全体の圧縮とデータ領域のみの圧縮が可能です。実行ファイルはロードされる時に伸長されるため、起動速度は非圧縮の実行ファイルに比べ遅くなります。もちろん起動してしまえば非圧縮の実行ファイルとの差はなくなってしまいますので、常駐プロセスのように起動停止を繰り返さないプログラムには適しています。

- スタックサイズフィールド
  再コンパイルせずに変更可能なスタックサイズのフィールドを持っています。MMU を持たない CPU では動的にスタック領域を拡張することが難しいため、固定サイズのスタック領域を持ちます。このフィールドは **flthdr** と呼ばれるツールで変更することが可能です。また、コンパイル時にサイズを指定することも可能です。

- XIP (eXecute In Place)対応
  Flat Binary Format は XIP にも対応しています。XIP とは eXecute In Place(その場実行)の略で、一般的には RAM に実行バイナリーをコピーすることなく、保存されている ROM 上で実行する機能のことです。

11.2. 実行ファイルの圧縮

例として前章で作成した Hello World プログラムを圧縮します。後半ではコンパイル時に圧縮を指定する方法を説明します。

注意

デフォルトのカーネルでは、圧縮された Flat Binary Format(ZFLAT)のファイルを実行できません。カーネルを ZFLAT 対応にするには「9.3ZFLAT 対応カーネルを作る」を参照してください。

11.2.1. コンパイル済バイナリファイルを圧縮

flthdr を使ってコンパイル済のバイナリーファイルを圧縮します。**flthdr** は Flat Binary Format のファイルを編集･表示するプログラムです。MicroBlaze ツールチェーンには **mb-flthdr** という名前で入っているので、以降 **mb-flthdr** として説明します。

通常のコンパイルで生成された実行ファイルでは以下のようになります。

例 11-1 通常の Flat Binary Format

```
[PC ~/hello]$ make
[PC ~/hello]$ mb-flthdr hello
hello
    Magic:        bFLT
    Rev:          4
    Entry:        0x50
    Data Start:   0x3e60
    Data End:     0x4bf0
    BSS End:      0x6bf0
    Stack Size:   0x1000
    Reloc Start:  0x4bf0
    Reloc Count:  0x53
    Flags:        0x1 ( Load-to-Ram )
[PC ~/hello]$
```

次に **mb-flthdr** で圧縮してみます。

例 11-2 圧縮された Flat binary Format

```
[PC ~/hello]$ mb-flthdr -z hello ------------------------------------------------------  ①
zflat hello --> hello
[PC ~/hello]$ mb-flthdr hello ---------------------------------------------------------  ②
hello
    Magic:        bFLT
    Rev:          4
    Entry:        0x50
    Data Start:   0x3e60
    Data End:     0x4bf0
    BSS End:      0x6bf0
    Stack Size:   0x1000
    Reloc Start:  0x4bf0
    Reloc Count:  0x53
    Flags:        0x5 ( Load-to-Ram Gzip-Compressed ) ----------------------------  ③
[PC ~/hello]$
```

① **mb-flthdr** に圧縮オプション'-z'を渡す
② **mb-flthdr** コマンドで生成された実行ファイルのヘッダ部を表示させる
③ Gzip-Compressed フラグが確認できる

11.2.2. コンパイル時に圧縮

コンパイル時に圧縮するには、**FLTFLAGS** 環境変数を使います。以下に Hello World での例を示します。

例 11-3 FLTFLAGS による圧縮の指定

```
[PC ~/hello]$ make FLTFLAGS=-z  ------------------------------------------------  ①
[PC ~/hello]$ mb-flthdr hello  -------------------------------------------------  ②
hello
    Magic:        bFLT
    Rev:          4
    Entry:        0x50
    Data Start:   0x3e60
    Data End:     0x4bf0
    BSS End:      0x6bf0
    Stack Size:   0x1000
    Reloc Start:  0x4bf0
    Reloc Count:  0x53
    Flags:        0x5 ( Load-to-Ram Gzip-Compressed ) ----------------------------  ③
[PC ~/hello]$
```

① **FLTFLAGS** 環境変数に**'-z'**をセットして **make** を実行する
② **mb-flthdr** コマンドで生成された実行ファイルのヘッダ部を表示させる
③ Gzip-Compressed フラグが確認できる

11.3. スタックサイズの指定

スタックサイズを指定する方法を 2 種類紹介します。

11.3.1. コンパイル済バイナリファイルスタックサイズを変更

mb-flthdr の**-s** でスタックサイズを指定します。アーキテクチャによりますが、スタックサイズのデフォルト値は 4096 が多いです。

例 11-4 スタックサイズの変更

```
[PC ~/hello]$ mb-flthdr -s 8192 ------------------------------------------------  ①
[PC ~/hello]$ mb-flthdr hello ---------------------------------------------------  ②
hello
    Magic:        bFLT
    Rev:          4
    Entry:        0x50
    Data Start:   0x3e60
    Data End:     0x4bf0
    BSS End:      0x6bf0
    Stack Size:   0x2000 ------------------------------------------------------  ③
    Reloc Start:  0x4bf0
    Reloc Count:  0x53
    Flags:        0x1 ( Load-to-Ram )
[PC ~/hello]$
```

① **mb-flthdr** にスタックサイズ変更オプション**'-s'**とスタックサイズを渡す
② **mb-flthdr** コマンドで生成された実行ファイルのヘッダ部を表示させる
③ 8192byte に変更されているのが確認できる

11.3.2. コンパイル時にスタックサイズを指定

コンパイル時にスタックサイズを指定するには、**FLTFLAGS** 環境変数を使います。以下に Hello World での例を示します。

例 11-5 FLTFLAGS によるスタックサイズの指定

```
[PC ~/hello]$ make FLTFLAGS='-s 8192' ---------------------------------------------  ①
[PC ~/hello]$ mb-flthdr hello ------------------------------------------------------  ②
hello
    Magic:        bFLT
    Rev:          4
    Entry:        0x50
    Data Start:   0x3e60
    Data End:     0x4bf0
    BSS End:      0x6bf0
    Stack Size:   0x2000 ---------------------------------------------------------  ③
    Reloc Start:  0x4bf0
    Reloc Count:  0x53
    Flags:        0x1 ( Load-to-Ram )
[PC ~/hello]$
```

① **FLTFLAGS** 環境変数に**'-s 8192'**をセットして **make** を実行する
② **mb-flthdr** コマンドで生成された実行ファイルのヘッダ部を表示させる
③ 8192byte に変更されたスタックが確認できる

12. Flash メモリの書き換え方法

SUZAKU の Flash メモリを書き換える方法は 3 つあります。この章ではこの 3 つの方法について説明します。NetFlash は uClinux が動作しているときに、Hermit はブートローダモードで SUZAKU を起動したときに使用できます。

12.1. NetFlash を使用した Flash メモリの書き換えかた

uClinux が起動している状態から、Flash メモリを書き換えます。

NetFlash というコマンドを使って Flash メモリを書き換えます。NetFlash はネットワーク経由でイメージファイルをダウンロードし、Flash メモリに書き込みます。NetFlash は http、ftp、tftp に対応しています。NetFlash を使用するには、いずれかに対応したアプリケーションが動作しているサーバが必要となります。お使いの環境にこのようなサーバがない場合はネットワーク管理者に相談してください。

「例 12-1 netflash の起動」のように SUZAKU 上でコマンドを発行します。

例 12-1 netflash の起動

```
[SUZAKU ~]# netflash http://local.server.name/suzaku_images/image.bin
```

local.server.name/suzaku_images/ はお使いの環境にあわせて変更してください。

Flash メモリの書き込みに成功すると、システムが再起動されます。

NetFlash にはさまざまなオプションが用意されています。参考までに以下に NetFlash のヘルプを掲載します。

例 12-2 netflash のヘルプ

```
[SUZAKU ~]# netflash -?
usage: netflash [-bCfFhijklntuv?] [-c <console-device>] [-d <delay>] [-o <offset>]
[-r <flash-device>] [<net-server>] <file-name>

        -b      don't reboot hardware when done
        -C      check that image was written correctly
        -f      use FTP as load protocol
        -F      force overwrite (do not preserve special regions)
        -h      print help
        -i      ignore any version information
        -H      ignore hardware type information
        -j      image is a JFFS2 filesystem
        -k      don't kill other processes (or delays kill until
                after downloading when root filesystem is inside flash)
        -K      only kill unnecessary processes (or delays kill until
                after downloading when root filesystem is inside flash)
        -l      lock flash segments when done
        -n      file with no checksum at end (implies no version information)
        -p      preserve portions of flash segments not actually written.
        -s      stop erasing/programming at end of input data
        -t      check the image and then throw it away
        -u      unlock flash segments before programming
        -v      display version number
[SUZAKU ~]#
```

12.2. Hermit を使用した Flash メモリの書き換えかた

Hermit を使った Flash メモリの書き換え手順を説明します。PC から対象となる機器(この場合は SUZAKU)にデータを転送することから、Flash メモリを書き換えることをダウンロードとも言います。Hermit を使って書き換えることができるものは、Hermit 自身と uClinux です。Hermit は指定された Flash メモリのリージョンまたはアドレスにシリアルポート経由でイメージファイルを書き込みます。Flash メモリマップやリージョンについては「5.Flash メモリのメモリマップを」参照してください。

注意

Hermit がインストールされていることを確認してください。インストールされてない場合は「7.5.Hermit のインストール」」を参照してください。

12.2.1. ブートローダモードで起動する

最初に SUZAKU をブートローダモードで起動します。起動選択ジャンパをショートにし、起動モード選択画面でエンターキーを押すことでブートローダモードになります。起動モードについて詳しくは「6.2.起動モード」を参照してください。

12.2.2. ダウンロード

PC から以下のコマンドを入力し、ダウンロードを始めます。

注意

Minicom がシリアルポートを使用している状態では、Hermit がシリアルポートを使用できないためダウンロードに失敗します。必ず Mincom は終了してから Hermit を起動してください。

uClinux のイメージファイル(**image.bin**)を書き換える場合は「例 12-3 image.bin の書き換え」のようなコマンドになります。

例 12-3 image.bin の書き換え

```
[PC ~]$ hermit download -i ~/uCliunux-dist/images/image.bin -r image
```

Hermit のイメージファイル(**/usr/lib/hermit/loader-suzaku.bin**)を書き換える場合は「例 12-4 Hermit コマンド」のようなコマンドになります。

例 12-4 Hermit コマンド

```
[PC ~]$ hermit download -i /usr/lib/hermit/loader-suzaku.bin -r bootloader --force-locked
```

各パラメータを説明します。

hermit

Hermit のコマンド名

download

ダウンロードコマンド

-i ファイル名

転送用ファイルを指定します。**loader-suzaku.bin** は Hermit と一緒にインストールされます

-r リージョン

転送先のリージョンを指定します。リージョンについては「5.Flash メモリのメモリマップ」を参照してください

--force-locked

ロックされているリージョンに強制的に書き込みます

参考までに Hermit のヘルプを掲載します。詳しい内容は Hermit の man page を参照してください。

例 12-5 Hermit のヘルプ

```
[PC ~]$ hermit
Usage: hermit [options] command [command options]
Available commands: download, help, go, map, terminal, upload
Armadillo-J command: firmupdate
Multiple commands may be given.
General options (defaults) [environment]:
        -e, --ethernet
        -i, --input-file <path>
        --netif <ifname> (eth0) [HERMIT_NETIF]
        --memory-map <path>
        --port <dev> (/dev/ttyS0) [HERMIT_PORT]
        -o, --output-file <path>
        --remote-mac <MAC address>
        -v, --verbose
        -V, --version
Download options:
        -a, --address <addr>
        --force-locked
        -r, --region <region name>
Memory map options:
        --anonymous-regions
[PC ~]$
```

12.3. モトローラ S 形式での Flash メモリの書き換えかた

モトローラ S 形式でのダウンロードはソフトウェアで SUZAKU の Flash メモリを更新する最終手段です。uClinux が起動しなくても、Hermit が起動しなくても使えます。ここでは、Minicom を使ってモトローラ S 形式の Hermit を SUZAKU にダウンロードする方法を説明します。

モトローラ S 形式の Hermit は付属 CDROM の tools/s-record/ディレクトリに **hermit.srec** と **hermit-8MB.srec** いう名前で入っています。それぞれ 4MB、8MB の Flash メモリ用です。このファイルがホームディレクトリにコピーされていると仮定します。以降、4MB の Flash メモリを前提に話を進めます。8MB Flash メモリをご使用の場合は適時読み替えてください。

注意

Flash メモリのサイズによって、Hermit が書き込まれるアドレスが異なります。モトローラ S 形式ではファイル内に書き込みアドレスが書かれているために、このように二つのファイルに分けてあります。両方のファイルは同じバイナリファイルから生成されています。

まず、シリアルクロスケーブルで SUZAKU と PC を接続し、PC 上で Minicom を起動します。次に SUZAKU の起動モードジャンパをショートして SUZAKU の電源を入れます。これで Minicom の画面に起動モード選択画面が表示されます。

「S」を起動モード選択画面で入力すると以下のようなメッセージが表示されます。

例 12-6 モトローラ S 形式ダウンロード開始画面

```
Erasing Flash...Done
Start sending S-Record!!
```

このメッセージの後、SUZAKU はモトローラ S 形式ファイルを待ちます。

Minicom のメニュー画面から「Send files」を選択します。アップロードサブメニューの中から ASCII を選択し、ファイル選択画面に移動します。ファイル選択画面で **hermit.srec** を選択して転送を開始します。転送中は Minicom の中に小さな画面が表示されます。転送が終了すると

```
READY: press any key to continue...
```

と表示されるので SUZAKU をリブートしてください。リブートした SUZAKU でブートローダモードを選択すると Hermit が起動されます。

13. トラブルシューティング

13.1. Hermit

Q. hermit コマンドが見つかりません

A. Hermit をインストールしてください ⇒ 7.5Hermit のインストール

Q. 「hermit: panic: can't connect to target」とエラーになります。

A. SUZAKU の電源は入っていますか？
シリアルクロスケーブルは正しく接続されていますか？ ⇒ 4.1接続方法
SUZAKU はダウンローダモードになっていますか？ ⇒ 6.2.2ブートローダモード

13.2. Minicom

Q. minicom を起動しようとしても、「command not found」とエラーになります

A. Minicom をインストールしてください ⇒ 7.4Minicom のインストール

Q. minicom を起動して、SUZAKU の電源を入れても何も画面に表示されません

A. SUZAKU が接続されているシリアルポートと Minicom が使っているポートは同じですか？
Minicom のシリアルポート設定は適切ですか？ ⇒ 4.2シリアル通信用ソフトウェア

13.3. uClinux-dist のコンパイル

Q. mb-gcc が見つからずエラーになります

A. ツールチェーンをインストールしてください ⇒ 7.1ツールチェーンのインストール

Q. ツールチェーンをインストールしたのに、まだ見つからないとエラーになります

A. 環境変数 PATH が正しく設定されていますか？ ⇒ 7.2環境変数の設定

改訂履歴

Ver.	年月日	改訂内容
1.0	2004/04/29	初版作成
1.0.1	2004/06/04	Minicom のインストールを apt から dpkg に変更 ツールチェーン名の誤字を修正 OTC の Makefile を汎用化
1.0.2	2004/12/15	会社住所変更
1.1.0	2005/1/31	8MB Flash の記述を追加

SUZAKU Software Manual **2005 年 1 月 31 日** version 1.1.0

株式会社アットマークテクノ
004-0062 札幌市中央区北 5 条東 2 丁目 AFT ビル 6F 011-207-6550 FAX: 011-207-6570