

Multi-Media ARM9

QQ2440v3 Linux 版

マニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

2009/2/2



copyright@2008-2009



第一章 QQ2440v3 ボードの概要	5
1.1 仕様	5
1.2 使えるデバイス例	
1.3 付属アプリケーション例	9
第二章 QQ2440v3 ボードの初体験	10
2.1 パソコン側のハイパーターミナルの設定	10
2.2 ハイパーターミナルのコンソールで遊ぶ	12
2.2.1 MP3 をディスプレイしよう	12
2.2.2 USBメモリの使用	12
2.2.3 SD/MMCメモリの使用	12
2.2.4 USBカメラで撮影した画像を取る	13
2.2.5 ボタンのテスト	13
2.2.6 シリアルポート 1/2 のテスト	13
2.2.7 ブザーのテスト	13
2.2.8 液晶のバックライトの制御	13
2.2.9 I2Cデバイス(24C04)のテスト	13
2.2.10 RTCの設定	14
2.2.11 液晶(LCD)画面を取ります	14
2.3 QQ2440v3 のネットワーク機能	14
2.3.1 QQ2440v3 のウェブサーバー	14
2.3.2 QQ2440v3 のTelnetとFtp機能	14
2.3.3 DNSとgatewayの設定	14
2.3.4 MACアドレスの設定	14
2.3.5 ネットワーク・ファイルシステム(NFS)のマウント	15
第三章 Linuxのクロス開発環境	16
3.1 クロス開発環境を構築	16
3.2 NFSサーバーを構築	17
3.3 NFSはルートファイルシステムとして起動	17
第四章 Linuxのアプリケーションを開発	
4.1 Hello, World!	
4.2 Hello,Worldをコンパイル	
4.2 Hello,WorldをARM9 ボードで実行	
4.3 他のサンプル	
第五章 Linuxカーネルを再構築	20
5.1 カーネルのソースコードを解凍	



5.2 コノノイク	20
5.3 カーネルをコンパイル	22
5.4 ドライバの場所	22
5.5 デバイスドライバのコンフィグ	23
5.5.1 LCDのコンフィグ	23
5.5.2 タッチパネルのコンフィグ	26
5.5.3 USBマウスとキーボード	
5.5.4 USBメモリ	
5.5.5 USBカメラ	32
5.5.6 CS8900 Ethernet	35
5.5.7 オーディオ	
5.5.8 SD/MMC	42
5.5.9 LED	42
5.5.10 ボタン	43
5.5.11 シリアルポート	44
5.5.12 RTC	45
5.5.13 yaffsファイルシステム	46
5.5.14 EXT2/VFAT/ NFSファイルシステム	
5.6 yaffsルートファイルシステムのイメージを生成	
5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門	
5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード	
5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します	52 53 53 54
5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集	
5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集 5.7.4 ドライバをコンパイルします	
 5.6 yaffsルートファイルシステムのイメージを生成	
 5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集 5.7.4 ドライバをコンパイルします 5.7.5 ARM9 ボードでドライバをインストールします 第六章 生成されたファイルを書き込む 6.1 NOR Flashから起動 6.2 USBドライバのインストール 6.3 NAND Flashのパーティション 6.4 ブートロードの書き込み 	
 5.6 yaffsルートファイルシステムのイメージを生成	525353545656565759595959626366
 5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集 5.7.4 ドライバをコンパイルします 5.7.5 ARM9 ボードでドライバをインストールします 第六章 生成されたファイルを書き込む 6.1 NOR Flashから起動 6.2 USBドライバのインストール 6.3 NAND Flashのパーティション 6.4 ブートロードの書き込み 6.5 Linuxのカーネルの書き込み 6.6 ルート・ファイルシステムの書き込み 	5252535354565656575959595962636366
 5.6 yaffsルートファイルシステムのイメージを生成	5252535354565657595959626263666768
 5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集 5.7.4 ドライバをコンパイルします 5.7.5 ARM9 ボードでドライバをインストールします 第六章 生成されたファイルを書き込む 6.1 NOR Flashから起動 6.2 USBドライバのインストール 6.3 NAND Flashのパーティション 6.4 ブートロードの書き込み 6.5 Linuxのカーネルの書き込み 6.6 ルート・ファイルシステムの書き込み 6.7 NAND Flashのバックアップ 6.8 NAND Flashのリストア 	52525353545656575959595962626366676870
 5.6 yaffsルートファイルシステムのイメージを生成 5.7 Linuxドライバの開発入門 5.7.1 簡単なドライバのソースコード 5.7.2 コンフィグファイルを編集します 5.7.3 Makefileを編集 5.7.4 ドライバをコンパイルします 5.7.5 ARM9 ボードでドライバをインストールします 第六章 生成されたファイルを書き込む 6.1 NOR Flashから起動 6.2 USBドライバのインストール 6.3 NAND Flashのパーティション 6.4 ブートロードの書き込み 6.5 Linuxのカーネルの書き込み 6.7 NAND Flashのパックアップ 6.8 NAND Flashのリストア 6.9 メモリでLinuxカーネルを直接に実行 	5252535354565656575959595962636667687072



7.1 H-JTAGをダウンロードとインストールします	75
7.2 NOR Flashを書き込む	79
付録 1 Qt/Embedded GUIプログラムを作る	
付録2 アプリケーションを移植	90
付録 3 Watchdogの使い方	95
付録 4 ADの使い方	96
付録 5 UVC(USB Device Class)Webカメラを使用	97
付録 6 Mini GUIの移植とプログラム設計	
付録 7 ADS1.2 とH-JTAGでプログラムを開発	
付録 8 U-boot	120

使用されたソースコードは<u>http://www.csun.co.jp/</u>からダ ウンロードできます。

第一章 QQ2440v3 ボードの概要

1.1 仕様



液晶は3.3V/5V二種類がありますQQ2440v3の電圧選択ジャンパー (J2)は必ず正しく設定されなければなりません。

CPU プロセッサー

 ARM920T コアを採用したサムソン(SAMSUNG)社の S3C2440A、周波数 400MHz、 最高周波数 533MHz。



メモリ

- 64MB SDRAM, 32 ビット幅データ・バス, SDRAM の最高周波数 100MHz
- 64MB NAND Flash メモリ
- 2MB NOR Flash メモリ

液晶(LCD)

- 4線式抵抗膜方式のタッチパネルのインターフェース
- 標準の LCD I/F を持って、3.5"から 12.1"までの各種液晶パネル(黒白、STN、TFT、 最高分解能 1024*768)に対応します。

インターフェース

- 10MBase-T Ethernet RJ45(CS8900) x 1
- RS232 シリアルポート x 3
- USB1.1 ホスト×1
- USB1.1 スレーブ x 1
- MMC/SD メモリカードのソケット x 1
- ステレオ・オーディオの出力 x1
- マイクの入力 x 1
- 10 ピンの JTAG(2mm DIP ピッチ)
- ユーザーLED x 4
- ユーザーボタン x 4
- PWM 制御の圧電ブザー x 1
- 可変抵抗、A/D のテストの為に x 1
- I2C バスの AT24C08、I2C バスのテストの為に x 1
- 20 ピン CMOS カメラのインターフェース(2mm DIP ピッチ)
- RTC のバッテリーバックアプ
- 34 ピン GPIO(2mm DIP ピッチ)
- 44 ピンのシステムバス(2mm DIP ピッチ)

搭載した OS

- Linux2.6.13
- WindowsCE.NET 5.0
- uCOSII



外形寸法

• 120×100(mm) 突起物は除く

供給電源

 12VDC 電源、プラグ 2.1mmφ、極性はセンタープラス ⊖ ● ● です。 電源ス イッチと電源指示 LED 付き

ジャンパーJ2 はボードの動作モデルを選択します。一つは Nor Flash から起動です。もう一つは Nand Flash から起動です。この二つの起動モデルでシステムのアドレスが異なります。



デフォルトの設定は Nand Flash から Linux を起動します。



1.2 使えるデバイス例



USB カメラ (SPACXX 又は UVC に対応)



USB 無線 LAN 装置



USB マウスとキーボード



外付けハードディスク



2GBまでの SD/MMCメモリ



USB HUB





USB シリアルポート





液 晶 付属のドライバ以外は、使えない可能性があります。

1.3 付属アプリケーション例





第二章 QQ2440v3 ボードの初体験

• DB9 メス-メス型のストレートケーブルで QQ2240v3 とパソコンを繋ぐ。



- クロス LAN ケーブルで QQ2240v3 とパソコンを繋ぐ。
- QQ2240v3のオーディオ出力とスピーカーを繋ぐ。
- 12V 電源、極性はセンタープラス ⊖-- €-- ⊕ です。電源を入れると:



2.1 パソコン側のハイパーターミナルの設定

パソコンのメニュー : **スタート → すべてのプログラム → アクセサリ → 通信 → ハイ** パーターミナルを選ぶと、次の画面が出てきます。



接続の設定	? 🛛
動しい接続	
名前を入力し、アイコンを選んでください。	
名前(<u>N</u>):	
	>
OK ++>	セル

このハイパーターミナルの名前を入力して、"OK"ボタンを押すと。

接続の設定	? 🛛
🦓 h8	
電話番号の情報を	入力してください。
国/地域番号(<u>C</u>):	日本 (81)
市外局番(E):	03
電話番号(P):	
接続方法(<u>N</u>):	COM1 💌
	OK キャンセル

使用しているシリアルポートを選んでください。



COM3070パティ ?	×
ボートの設定	
ビット/秒(图): 115200	
データ ビット(<u>D</u>): 8	
パリティ(P): なし	
ストップ ビット(S): 1	
フロー制御(F): 歴し	
OK キャンセル 適用(A)	5

シリアル通信速度を 115200bps に設定します。フロー制御はなしです。 "OK"ボタンを押すと、設定が完了します。

2.2 ハイパーターミナルのコンソールで遊ぶ

2.2.1 MP3 をディスプレイしよう

madplay your.mp3 madplay -h は madplay のヘルプを表示します。

2.2.2 USB メモリの使用

USB メモリを USB ホスト又は USB ハブに挿入して、次のコマンド: # mount /dev/sda1 /mnt #ls/mnt

2.2.3 SD/MMC メモリの使用

SD/MMC メモリは SD/MMC スロットに挿入して、次のコマンド: # mount /dev/mmc/blk0/part1 /mnt

ls /mnt

最大 2GB までの SD/MMC メモリが使えます。



2.2.4 USB カメラで撮影した画像を取る

先ず、USB カメラ(中蕊微又は UVC のチップを採用したもの)を USB ホスト又は USB ハ プに挿入して、次のコマンド:

ls /dev/v4l/video0(USB カメラをマウントしたのを確認します。)

/dev/v4l/video0(この情報が見れば、USB カメラのマウントが成功しました。) # spcacat p 100ms N 5(USB カメラの画像を取る。100ms 間 1 つ画像を取って、全ての 5 枚 jpg ファイルを今のフォルダに保存します。)

2.2.5 ボタンのテスト

buttons

ARM9 ボードのユーザボタンを押してください。

2.2.6 シリアルポート 1/2 のテスト

TTL—RS232 変換ボード(別売)でシリアルポート 1/2 とパソコンを繋ぎます。新八イパータ ーミナルを実行して、通信速度 115200、フロー制御なしを設定して、ARM9 のコンソール で次のコマンドを入力します。 # armcomtest -d /dev/tts/1 -o (シリアルポート 1 のテスト)

armcomtest -d /dev/tts/2 -o (シリアルポート 2 のテスト)

2.2.7 ブザーのテスト

pwm_test

2.2.8 液晶のバックライトの制御

QQ2440v3 はこの機能がありません # bl 0 close LCD backlight # bl 1 open LCD backlight

2.2.9 I2C デバイス(24C04)のテスト

I2C メモリ 24C04 に 0~0xFF を書き込みます。 # i2c -w

24C04 を読み出します。

i2c -r



2.2.10 RTC の設定

(1)#date -s 042916352007 #今の時間を設定します: 2007-04-29 16:34
(2)#hwclock -w #今の時間を S3C2440 の RTC に保存します。
(3)#hwclock -s #起動の時、Linux 時間を S3C2440 の RTC から回復します。

hwclock -s コマンドは起動スクリプト(/etc/init.d/rcS)に書き込みま した。起動の時、自動的に実行します。

2.2.11 液晶(LCD)画面を取ります

#snapshot pic.png 液晶(LCD)で表示された画面を pic.png ファイルとして保存します。

2.3 QQ2440v3 のネットワーク機能

2.3.1 QQ2440v3 のウェブサーバー

QQ2440v3 の Linux でウェブサーバー(boa)をインストールしました。パソコンのブラウザ ーで http://192.168.1.230 を入力すると、QQ2440v3 のホームページが見えます。このホー ムページを通じて、QQ2440v3 のユーザーLED と USB カメラをアクセスできます。

2.3.2 QQ2440v3 の Telnet と Ftp 機能

QQ2440v3 の Linux でクライント側とサーバー側の Telnet/Ftp をインストールしました。 ご利用してください。 デフォルトの設定: Telnet のユーザーネームは root です、password がありません。 Ftp のユーザーネームは plg です、password も plg です。

2.3.3 DNSと gateway の設定

DNS の IP アドレスを/etc/resolv.conf ファイルに書き込みます。 gateway の設定:#route add default gw 192.168.1.1

2.3.4 MAC アドレスの設定

ifconfig eth0 down

if config eth0 hw ether 00:11:AA:BB:CC:DD(新 MAC アドレス)

ifconfig eth0 up

新 MAC アドレスが有効のため、これらのコマンドを起動スクリプト/etc/init.d/rcS に書き



込みます。

2.3.5 ネットワーク・ファイルシステム(NFS)のマウント

まず、ネットワーク・ファイルシステムのサーバーを構築します。(3.2 を参照してください) #mount -t nfs -o nolock 192.168.1.111:/root_nfs /mnt

192.168.1.111 はネットワーク・ファイルシステムのサーバーの IP アドレスです。

マウント成功すれば、ARM9 は大きなリモート・ハードディスク(/mnt)を直接にアクセス できます。プログラムを開発する時が便利です。

#umount /mnt #リモート・ハードディスクを ARM9 システムから外します。



第三章 Linux のクロス開発環境

3.1 クロス開発環境を構築

QQ2440v3のクロス開発環境は Redhat9.0 をお勧めします。ほかの Linux の場合は、エラーが出すかもしれません。

Redhat9.0 で ARM のクロス開発ツールをインストールします。

```
# tar xvzf arm-linux-gcc-3.3.2.tgz -C /
```

tar xvzf arm-linux-gcc-2.95.3.tgz -C /

tar xvzf arm-linux-gcc-3.4.1.tgz -C /

使いやすいため、「export PATH=\$PATH:/usr/local/arm/3.4.1/bin」 を.bashrc ファイルに入れます。

再び Redhat9 にログイン、次のコマンド:

arm-linux-gcc -v

💙 root@localhost:~	- - ×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal <u>G</u> o <u>H</u> elp	
root@localhost:~	root@localhost:~
<pre>[root@localhost root]# arm-linux-gcc - Reading specs from /usr/local/arm/3.4 Configured with: /work/crosstool-0.27, 3.2/gcc-3.4.1/configuretarget=arm-1 prefix=/usr/local/arm/3.4.1with- inux/includewith-local-prefix=/usr, e-nlsenable-threads=posixenable- enable-languages=c,c++enable-shar Thread model: posix gcc version 3.4.1 [root@localhost root]# []</pre>	-v .1/lib/gcc/arm-linux/3.4.1/specs /build/arm-linux/gcc-3.4.1-glibc-2. linuxhost=i686-host_pc-linux-gnu -headers=/usr/local/arm/3.4.1/arm-1 /local/arm/3.4.1/arm-linuxdisabl -symvers=gnuenablecxa_atexit redenable-c99enable-long-long
	Y

この画面が出たら、ARM 用のクロス開発環境をインストール成功しました。他のクロス開 発ツールを使用したら、例えば 2.95.3、「export PATH=\$PATH:/usr/local/arm/2.95.3/bin」 を.bashrc ファイルに入れます。



3.2 NFS サーバーを構築

ネットワーク・ファイルシステム(NFS)を使用すれば、ARM9 は大きなリモート・ハードディスクを直接にアクセスできます。プログラムを開発する時が便利です。次は NFS サーバーを構築手順です。

(1) NFS のルートシステムファイルを解凍します。

tar xvzf root_nfs.tgz -C /

(2) /etc/exports ファイルを編集します。

「/root_nfs *(rw,sync,no_root_squash)」を/etc/exports に入れます。

(3) NFS サーバを起動します。

#/etc/init.d/nfs start

(4) NFSf ファイルシステムを確認します。

#mount -t nfs localhost:/root_nfs /mnt/
#ls /mnt

firewallの設定によって、NFS ヘアクセスできない可能性があります。

3.3 NFS はルートファイルシステムとして起動

ARM9ボードが起動、又はリセットの時、ハイパーターミナルにスペースキーを押します。 次のコマンドを入力します。 Supervivi>param linux_cmd_line set "console=ttySAC0 root=/dev/nfs nfsroot=192.168.1.111:/root nfs ip=192.168.1.70:192.168.1.111:192.168.1.111:255.255.255.0:QQ2440v3.arm9.net:eth0:of f" param set linux_cmd_line は linux 起動のパラメーターです。パラメーターの意味は: nfsroot は NFS サーバーの IP アドレス。 "ip="の後ろ: 第一項(192.168.1.70)はARM9ボードのIP; 第二項(192.168.1.111)はホストの IP; 第三項(192.168.1.111)はゲットウェイの IP; 第四項(255.255.255.0)はネットマスク: 第五項はホストのドメイン(自由的に入力でも大丈夫です)



eth0 は LAN デバイスの名前。

Boot コマンドを入力すると Supervivi>boot

ARM9 ボードは NFS からブートします。

第四章 Linux のアプリケーションを開発

4.1 Hello, World!

4.2 Hello,World をコンパイル

#cd examples/hello #arm-linux-gcc -o hello main.c 又は #make 実行できるhelloを生成します。

4.2 Hello,World を ARM9 ボードで実行

生成された実行コードhelloをARM9ボードに入れて、ARM9のコンソールで実行します。 # ./hello hello, FriendlyARM!

ARM9ボードに入れるのは幾つの方法があります。USBメモリ、SDメモリ、FTPな ど。一番便利な方法はNFSです。ARM9ボードは直接にホスト側の実行ファイルを 実行できます。

4.3 他のサンプル

examples.tgzに幾つのサンプルがあります。





examples/led: LEDテスト examples/buttons: ボタンのテスト examples/udptak: UDPサンプル examples/math: 数学処理 examples/pthread: マルチthreadサンプル examples/led-player: pipeサンプル examples/c++: c++のサンプル



第五章 Linux カーネルを再構築

5.1 カーネルのソースコードを解凍

#tar xvzf linux-2.6.13-2440-20080910.tgz

このカーネルはQQ2440v3/MINI2440に両対応する

root@localhost:/opt/FriendlyARM/QQ2440	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal <u>G</u> o <u>H</u> elp	
root@localhost:/opt/FriendlyARM/QQ2440/vivi	root@localhost:/opt/FriendlyARM/QQ2440
kernel-2.6.13/fs/umsdos/	
kernel-2.6.13/fs/umsdos/notes	
kernel-2.6.13/fs/bio.c	
kernel-2.6.13/fs/file_table.c	
kernel-2.6.13/fs/ioprio.c	
kernel-2.6.13/fs/minix/	
kernel-2.6.13/fs/minix/itree_v2.c	
kernel-2.6.13/fs/minix/inode.c	
kernel-2.6.13/fs/minix/Makefile	
kernel-2.6.13/fs/minix/file.c	
kernel-2.6.13/fs/minix/itree_common.c	
kernel-2.6.13/fs/minix/namei.c	
kernel-2.6.13/fs/minix/minix.h	
kernel-2.6.13/fs/minix/itree_v1.c	
kernel-2.6.13/fs/minix/dir.c	
kernel-2.6.13/fs/minix/bitmap.c	
kernel-2.6.13/fs/xattr.c	
kernel-2.6.13/fs/quota.c	
kernel-2.6.13/fs/qnx4/	
kernel-2.6.13/fs/qnx4/inode.c	
kernel-2.6.13/fs/qnx4/Makefile	
kernel-2.6.13/fs/qnx4/file.c	
kernel-2.6.13/fs/qnx4/fsync.c	
	*

5.2 コンフィグ

cd kernel-2.6.13
make menuconfig



<u>-</u> ile <u>E</u>	dit <u>V</u> iew	<u>T</u> ermina	d <u>G</u> o	<u>H</u> elp	
oot@loca	alhost:~				root@localhost:/opt/FriendlyARM/QQ2440/kernel-2.6
Linux	Kernel va	2.6.13 C	onfigu	ration	
			— Lin	ux Kern	el Configuration
Ar: Hij <m: fo:</m: 	row keys ghlighted > modulan r Search	navigat l letter rizes fe . Legen . e . e . e . e . r . i	e the s s are s atures d: [*] rnel h curity yptogr brary s	menu. hotkeys . Press built-: acking options aphic op routines	<pre><enter> selects submenus> Pressing <y> includes, <n> excludes, s <esc><esc> to exit, <?> for Help, in [] excluded <m> module < >> s> ptions> s></m></esc></esc></n></y></enter></pre>
		Lo	ad an	Alterna	te Configuration File
		Sa	ve Con	figurat	ion to an Alternate File

メニューの「Load an Alternate Configuration File」を選択してください。

♥ roo	t@cap	cross:/	opt/Friendly	ARM/r	nini2440/kernel-2.6.13	X
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>T</u> erminal	<u>G</u> 0	<u>H</u> elp	
Linu:	x Keri	nel v2	2.6.13 Con	figu	ration	· · · · · · · · · · · · · · · · · · ·
		E t a c	nter the o load. onfigurat bort. onfig_n35	name Accep ion y	of the configuration file you wish of the name shown to restore the you last retrieved. Leave blank to	
				2	Ok > < Help >	

コンフィグファイル名前「config_n35」を入力して、「ok」ボタンを押します。 Kernel-2.6.13に幾つのコンフィグファイルがあります。名前は 「config_xxx_yyyy」です。xxxは「n35」、又は「a70」です。「n35」 は3.5"液晶に対応する。「a70」は7"液晶に対応する。yyyyは「qq2440」、 又は「mini2440」です。「qq2440」はQQ2440v3ボードに対応する。 「mini2440」はMINI2440ボードに対応する。

5.3 カーネルをコンパイル

カーネルをコンパイルするため、arm-linux-gcc-3.4.1が必要です。

make zImage

✓ root@	root@localhost:/opt/FriendlyARM/QQ2440/kernel				1el-2.6.13 📃 🗖 🎗
<u>F</u> ile <u>E</u>	dit <u>V</u> iev	v <u>T</u> erminal	<u>G</u> 0	<u>H</u> elp	
root@loca	lhost:~				root@localhost:/opt/FriendlyARM/QQ2440/kernel-2.6.13
CHK	incl	ude/asm-arm	n/cons	stants.h	·
UPD	incl	ude/asm-arm	n/cons	stants.h	
Gener	ating in	nclude/asm-	-arm/n	nach-type:	s.h
CC	init	/main.o			
CHK	incl	ude/linux/	compi	le.h	
UPD	incl	ude/linux/	compi	le.h	
CC	init	/version.o			
CC	init	/do_mounts	.0		
CC	init	/do_mounts	devf	5.0	
CC	init	/do_mounts	_rd.o		
LD	init	/mounts.o			
CC	init	/initramfs	. 0		
CC	init	/calibrate	.0		
LD	init	/built-in.	D		
HOSTC	C usr/	gen_init_c	pio		
CHK	usr/	initramfs_	list		
UPD	usr/	initramfs_	list		
CPIO	usr/	initramfs_0	data.	cpio	
GZIP	usr/	initramfs_0	data.	cpio.gz	<u> </u>
AS	usr/	initramfs_0	data.	D	
LD	usr/	built-in.o			
CC	arch	/arm/kerne	l/com	pat.o	
CC	arch	/arm/kerne	l/dma	.0	

コンパイル完了したら、arch/arm/bootしたでカーネルファイルzImageを生成させます。

5.4 ドライバの場所

(1)DM9000 Ethernet devices



kernel-2.6.13/drivers/net/dm9000x.c (2)シリアルポートdev/tts/0,1,2 kernel-2.6.13/drivers/serial/s3c2410.c (3) RTC kernel-2.6.13/drivers/char/s3c2410-rtc.c (4)LED kernel-2.6.13/drivers/char/mini2440_leds.c (5)ボタン kernel-2.6.13/drivers/char/mini2440_buttons.c (6)タッチパネル kernel-2.6.13/drivers/input/touchscreen/s3c2410_ts.c (7)yaffsファイルシステム kernel-2.6.13/fs/yaffs2 (8)USBマウス、キーボード kernel-2.6.13/drivers/ usb/input/hid-input.c (9)SD/MMCドライバ、linux-2.6.13は2GBまでのメモリをアクセスできます。 kernel-2.6.13/drivers/mmc (10)Nand Flash kernel-2.6.13/drivers/mtd/nand (11)UDA1341オーディオ kernel-2.6.13/ sound/oss/uda1341.c kernel-2.6.13/ drivers/13 (12)LCD kernel-2.6.13/drivers/video/s3c2410fb.c (13)USBメモリ kernel-2.6.13/drivers/usb/storage (14)USBカメラ kernel-2.6.13/drivers/usb/media/gspca

5.5 デバイスドライバのコンフィグ

5.5.1 LCD のコンフィグ

メインメニューで「Device Drivers」を選択します。



t@localhost:~		root@localhost:/opt/FriendlyARM/	root@localhost:/opt/FriendlyARM
inux Kernel v2.	6.13 Confi	guration	
		Device Drivers	1
Arrow keys n	avigate th	e menu. <enter> selects sul</enter>	omenus>.
Highlighted	letters ar	e hotkeys. Pressing <y> inc</y>	cludes, <n> excludes,</n>
<m> modulari</m>	zes featur	es. Press <esc><esc> to ext</esc></esc>	it, for Help,
for Search.	Legend: [*] built-in [] excluded <	<m> module < ></m>
	2C supp	ort>	
	H rdware	Monitoring support>	
	13 seria	l bus support>	
	M sc dev	ices>	
	Multimed	ia Capabilities Port drivers	s>
	Miltimed	ia devices>	
	Graphics	support>	
	ound -	>	
	USB supp	ort>	
	MMC/SD C	ard support>	
- E			-
L		the second se	
1	<sel< td=""><td><mark>ect></mark> < Exit > < Help ></td><td>></td></sel<>	<mark>ect></mark> < Exit > < Help >	>

「Graphics support」を選択して、「*」を入れます。

- <*> Support for frame buffer devices
- <*> S3C2410 LCD framebuffer support
- 「LCD select」を選択します。







対応されたLCDドライバを選択してください。

5.5.2 タッチパネルのコンフィグ

「Device Drivers」メニューで「input device support」を選択します。



「Touchscreens」に入ります。次を選択します。

- <*> Touchscreen interface
- <*> Event interface



t@loca	alhost:~		root@loc	alhost:/opt/Frie	ndlyARM/	root@localhost	:/opt/FriendlyARM
inux	Kernel v	2.6.13 Cont	figuratio	n			
4.77	now kove	novigato t	the menu	device sup	electe en	hmonus >	1
AT Hi	ghlighto	lottore	are hotke	ve Prosei	ng (V) in	cludes (NS	eveludes
11	S modula	nizos fost	are notke	ys. riessi	ng (1) In	it 25 for	Holp
<m fo</m 	r Soarch	Logond.	[*] huil	t in [] o	veluded	AN modulo	nerp,
10	r search	. regenu:	["] Dull	t-m []e	xcruded	CM> module	~
	(480)	Hirizont	tal scree	n resolutio	n		
	(272)	Vertical	screen	resolution			
	< >	Toystick i	interface	10001011011			
	<*>>	Touchscree	en interf	ace			
	(240)	Herizont	tal screet	n resolutio	n		
	(320)	Vertical	l screen :	resolution			
	<*>	vent inte	erface				
	< >	vent debu	ugging				
		Input Devi	ice Drive	rs			
	[*]	eyboards	>				
L		18.5					
							्य
		<se< td=""><td>elect></td><td>< Exit ></td><td>< Help</td><td>></td><td></td></se<>	elect>	< Exit >	< Help	>	
					100		



ホームページ:<u>http://www.csun.co.jp</u>



5.5.3 USB マウスとキーボード

「Device Drivers」を選択して、次に「*」を入れます。

<*>USB supoort



次に「*」を入れます。 <*> Support for Host-side USB <*> OHCI HCD support







5.5.4 USB メモリ

^{Γ} Device Drivers \rightarrow ^{Γ} SCSI device support



[*] legacy /proc/scsi support <*> SCSI disk support



(@localnost.~	root@localhost:/opt/FriendlyARM/	root@localhost:/opt/FriendlyARM
nux Kernel v2.6.13 Cor	figuration	, , , ,
	SCSI device support	1
Arrow keys navigate	the menu. <enter> selects sub</enter>	bmenus>.
Highlighted letters	are hotkeys. Pressing <y> in</y>	cludes, <n> excludes,</n>
<m> modularizes feat</m>	tures. Press <esc><esc> to ex:</esc></esc>	it, for Help,
for Search. Legend:	: [*] built-in [] excluded ·	<m> module < ></m>
		1
CSI device su	ipport	
[*] legacy /proc	c/scsi/ support	
SCSI support	t type (disk, tape, CD-ROM)	
<*> SCSI disk su	ipport	
< > CSI tape su	ipport	
< > SCSI OnStream	um SC-xO tape support	
< > SCSI CDROM s	support	
< > SCSI generic	support	
< > SCSI media o	changer support	
Some SCSI de	evices (e.g. CD jukebox) suppor	rt multiple LUNs
		-i
1.000	alacts / Brits / Holm :	>
<	perfects c mart s c merb ;	322

 $\mathcal{F} = \mathcal{F} - \mathcal{F}$ Device Drivers $\downarrow \rightarrow \mathcal{F}$ USB support \downarrow

<*> USB Mass Storage support



	root@localho	st:/opt/FriendlyARM/	root@localhost:/opt/FriendlyARM
inux Kernel v2.6.1	8 Configuration		
		Sector State	
	USE	support	1
Arrow keys navi	ate the menu. <	Enter> selects su	bmenus>.
Highlighted let	ers are hotkeys.	Pressing <y> in</y>	cludes, <n> excludes,</n>
<m> modularizes</m>	features. Press	<esc><esc> to ex:</esc></esc>	it, for Help,
for Search. Le	gend: [*] built-i	n [] excluded ·	<m> module < ></m>
r*(4	200 m 832	
< > SB Print	er support		
NOTE: USB	STORAGE enables	SCSI, and 'SCSI d	isk support' may als
<*> USB Mass	Storage support		
[*] SB Mas	Storage verbose	debug	
[] Datafab	Compact Flash Re	ader support (EXP	ERIMENTAL)
[] reecom	USB/ATAPI Bridge	support	
	1100 /400 D 1 1	unnort	
[] ISD-200	USB/ATA Bridge s	upport	
[] SD-200 [] Microte	h/ZiO! CompactFl	ash/SmartMedia su	pport
[] ISD-200 [] Microte [] SBAT/U	DSB/ATA Bridge s h/ZiO! CompactFl BATO2-based stor	ash/SmartMedia suj age support (EXPE	pport RIMENTAL)
[] ISD-200 [] Microte [] USBAT/U [] SanDisk	USB/ATA Bridge s h/ZiO! CompactFl BBATO2-based stor SDDR-09 (and oth	ash/SmartMedia suj age support (EXPE er SmartMedia) suj	pport RIMENTAL) pport (EXPERIMENTAL)
[] ISD-200 [] Microte [] ISBAT/U [] SanDisk	USB/ATA Bridge s h/ZiO! CompactFl BATO2-based stor SDDR-09 (and oth	ash/SmartMedia suj age support (EXPE er SmartMedia) suj	pport RIMENTAL) pport (EXPERIMENTAL)
[] SD-200 [] M crote [] SBAT/U [] anDisk	USB/AIA Bridge s h/ZiO! CompactFl BATO2-based stor SDDR-09 (and oth	ash/SmartMedia suj age support (EXPE er SmartMedia) suj	pport RIMENTAL) pport (EXPERIMENTAL)
[] ISD-200 [] Microte [] ISBAT/U [] SanDisk	<pre>OSB/ATA Bridge s th/ZiO! CompactFl BATO2-based stor SDDR-09 (and oth </pre> <select> <</select>	ash/SmartMedia suj age support (EXPE er SmartMedia) suj Exit > < Help :	pport RIMENTAL) pport (EXPERIMENTAL)

5.5.5 USB カメラ

 $\varkappa = \neg - \Box$ Device Drivers $\neg \neg$ Multimedia devices \Box



@localhost:~		root@localhost:/opt/Friend	llyARM/ ro	oot@localhost:/opt/Friendly/	R
nux Kernel v2.6	.13 Config	guration			
		Device Drivers	24		7
Arrow keys na	vigate the	e menu. <enter> sel</enter>	lects subm	enus>.	
Highlighted 1	etters are es feature	e hotkeys. Pressing	g <y> incl > to evit</y>	udes, <n> excludes,</n>	
for Search.	Legend: [*] built-in [] exc	luded <m< td=""><td>> module < ></td><td></td></m<>	> module < >	
	ð—				
	I2C suppo	ort>			
	H rdware	Monitoring support	>		
	M sc devi	ices>			
	Mltimedi	ia Capabilities Port	t drivers	>	
	Multimedi	ia devices>			
	Craphics	support>			
	Sound	>			
9	SB suppo	ort> Ard support>			
4					-
	Provide State	ets < Exit >	< Help >		
	<sele< td=""><th>a price y</th><td></td><td></td><td></td></sele<>	a price y			

<*> Video For Linux



Ejle	e <u>E</u> dit	host/op View	<u>VFriendly</u> Terminal	ARM/QQ24 <u>G</u> o <u>H</u> e	440/kernel 2.6.1 Ip	3//////////////////////////////////////		_ 🗆 X
root	@localhost	:~		root@l	ocalhost:/opt/Fri	endlyARM/	root@localhost:/opt/Friendly	/ARM/
root	@localhost nux Kern Arrow Highli <m> mo for Se</m>	<pre>keys n ghted dulari arch. </pre>	6.13 Con avigate letters zes feat Legend: *> Video Vid Vid Digit	root@ld figurati the menu are hotk ures. I [*] bui For Lir eo For I io Adapt al Video	ocalhost:/opt/Fri ion Itimedia dev 1. <enter> keys. Press Press <esc>< ilt-in [] hux Linux> ters> b Broadcasti</esc></enter>	endlyARM/ ices selects su ing <y> in Esc> to ex excluded ng Devices < Help</y>	<pre>root@localhost:/opt/Friendly bmenus>. cludes, <n> excludes, it, <?> for Help, <m> module < ></m></n></pre>	

<*> OmniVision Camera Chip support

 $\bigstar = \neg - \ulcorner$ Device Drivers $\urcorner \rightarrow \ulcorner$ USB support \rbrack

<*> USB OV511 Camera support

<*> USB SPCA5XX Sunplus/Vimicro/Sonix jpeg Cameras



@localho	st:~			root@l	ocalhost:/o	pt/Frien	dlyARM/	. roo	t@localh	ost:/op	t/Friend	VARN
nux Ker	nel v2	2.6.13	Confi	gurat	ion							
					ISR on	mort						
Arrow	keys	naviga	te th	emenu	u. <ente< td=""><td>er> se</td><td>lects s</td><td>ubme</td><td>nus</td><td>>.</td><td></td><td></td></ente<>	er> se	lects s	ubme	nus	>.		
Highl	ighted	l lette	rs ar	e hoti	keys. Pi	ressin	g <y> i</y>	nclu	des, <n< td=""><td>> exe</td><td>cludes</td><td></td></n<>	> exe	cludes	
<m> 11</m>	odular	izes f	eatur	es. 1	Press <es< td=""><td>sc><es< td=""><td>c> to e</td><td>xit,</td><td><? > fo</td><td>r He</td><td>lp, <!--2</td--><td>></td></td></es<></td></es<>	sc> <es< td=""><td>c> to e</td><td>xit,</td><td><? > fo</td><td>r He</td><td>lp, <!--2</td--><td>></td></td></es<>	c> to e	xit,	fo	r He	lp, 2</td <td>></td>	>
for S	earch.	Lege	nd: [*] bu:	ilt-in	[] ex	cluded	<m></m>	module	< :	>	
							ana an					7
< >	USB 1	BM (Xi	rlink) C-it	t Camera	suppo	rt					
< >	USB N	onica	webca	m supp	port							
<*>	USB C	W511 C	amera	suppo	ort							
< >	USB S	E401 C	amera	suppo	ort							
< >	USB S	N9C10x	PC C	amera	Controll	ler su	pport					
< >	USB S	TV680	(Penc	am) Ca	amera sup	pport						
< >	USB V	1996[87]CF J	PEG DI	ual Mode	Camer	a suppo	rt				
< >	USB I	hilips	Came	ras								
<*>	USB S	SPCA5XX	Sunp	lus/V:	imicro/So	onix j	peg Can	eras				
IL ST ST	USB N	letwork	Adap	lers	>							
4-(10)-												
			-	_								-1
			<sel< td=""><td>ect></td><td>< Exit</td><td>t ></td><td>< Help</td><td>></td><td></td><td></td><td></td><td></td></sel<>	ect>	< Exit	t >	< Help	>				
				14			100					-

5.5.6 CS8900 Ethernet

メインメニューで「Netwoking」を選択します。



@localhost:~	r	oot@localhost:/opt/	/FriendlyARM/	root@localhost:/opt/Friend	Iyarm
nux Kernel v2.6.1	3 Config	uration			
	14	nux Kernel (on	figuration -		-
Arrow keys navi	gate the	menu. <enter< th=""><th>> selects su</th><th>bmenus>.</th><th></th></enter<>	> selects su	bmenus>.	
Highlighted let	ters are	hotkeys. Pre	ssing <y> in</y>	cludes, <n> excludes</n>	
<m> modularizes</m>	feature	s. Press <esc< td=""><td>><esc> to ex</esc></td><td>it, <? > for Help, <!--</td--><td>></td></td></esc<>	> <esc> to ex</esc>	it, for Help, </td <td>></td>	>
for Search. Le	gend: [*] built-in [] excluded	<m> module < ></m>	
	1				1
	ernel	Features>			
	oot op	tions>			
	loatin	g point emu⊥at	10n>		
1	Serspa	ce binary lorm	als>		
	Network	ing>	0113>		
	Device 1	Drivers>			
	ile sy	stems>			
	rofili	ng support	->		
	Dernel 1	hacking>			
	-				-1
	<sele< td=""><td>ct> < Exit</td><td>> < Help</td><td>></td><td></td></sele<>	ct> < Exit	> < Help	>	

[*] Networking support

「Networking options」に入ります。






$\lambda \equiv \neg \neg$ Device Drivers $\neg \rightarrow \neg$ Network device support \neg







<*> CS8900 support

<*> Generic Media Independent Interface device support



5.5.7 オーディオ

 $\exists \exists \exists \neg \neg$ Device Drivers $\exists \rightarrow \neg L3$ serial bus support \exists



<u>root@lo</u> ∃ile <u>E</u> di	t <u>V</u> iew	<u>T</u> erminal <u>C</u>	M/QQ2440/kernel-2.6.13 <u>G</u> o <u>H</u> elp	
ot@local	host:~		root@localhost:/opt/FriendlyARM/	root@localhost:/opt/FriendlyARM/
Linux K	ernel v2	.6.13 Confi	iguration	
Arr Hig <m> for</m>	ow keys nlighted modular Search.	navigate th letters ar izes featur Legend: [ISDN sub Input de Characte I2C supp H-rdware L3 seria Misc dev Miltimed Graphics	Device Drivers The menu. <enter> selects sub re hotkeys. Pressing <y> ind res. Press <esc><esc> to exist [*] built-in [] excluded of psystem> evice support> er devices> toort> toort> e Monitoring support> rices> tia Capabilities Port drivers support> support></esc></esc></y></enter>	bmenus>. cludes, <n> excludes, it, <? > for Help, <m> module < ></m></n>
		<se]<="" th=""><th>lect> < Exit > < Help :</th><td>></td></se>	lect> < Exit > < Help :	>

「L3 serial bus support」中の内容を全部選択します。

メニュー「Device Drivers」→「Sound→」 Open Sound System - - ->に入ります。





ホームページ:<u>http://www.csun.co.jp</u>

<Select>

< Exit >

41

< Help >



5.5.8 SD/MMC

 $\lambda = -$ ^r Device Drivers \rightarrow ^r MMC/SD Card support - --> \rightarrow



5.5.9 LED

 $\lambda \equiv \neg \neg$ Device Drivers $\neg \rightarrow \neg$ Character devices - - -> \neg



_	<u> </u>	104	<u></u>	20	<u></u> cip		_
(D)	aphily:	/opt/Fri	endlyARM/C	Q2440)/kernel-2.6.13	root@caphily:/opt/FriendlyARM/QQ2440/kernel-2.	6.13
nu	x Ker	nel v2	2.6.13 Co	nfigu	ration		
					— Charact	er devices	1
1	Arrow	keys	navigate	the 1	menu. <en< td=""><td>ter> selects submenus>.</td><td></td></en<>	ter> selects submenus>.	
Ű.	Highl	ighteo	d letters	are l	hotkeys.	Pressing <y> includes, <n> excludes,</n></y>	
- 33	<m> mo</m>	odula	rizes feat	tures	. Press <	Esc> <esc> to exit, <? > for Help, </esc>	
	for Se	earch.	. Legend	: [*]	built-in	[] excluded <m> module < > module</m>	
Г	-		8 			1	
			PMI	->			
			atchdog	Card	s>		
		< >	/ev/nvra	an suj	pport		
		< >	Inhanced	Real	Time Cloc	k Support	
		[*]	S3C2410 I	RTC D	river		
		<>	QQ2440 L1	EDs D	river		
		<m></m>	Q2440 H	ello 1	Module sam	ple	
		<*>	Q2440 Bi	utton	s		
L							
				e 4.5			
				<sele< td=""><td>ct> < E</td><td>xit > < Help ></td><td></td></sele<>	ct> < E	xit > < Help >	

5.5.10 ボタン

 $A \equiv a - f$ Device Drivers $a \rightarrow f$ Character devices - - -> a



e	<u>E</u> dit	<u>V</u> iew	<u>T</u> erminal	<u>G</u> o	<u>H</u> elp		
t@	capbily	:/opt/Frie	endlyARM/Q	Q2440	/kernel-2.6.13	root@capbily:/opt/FriendlyARM/QQ2440/kernel-2.6.1	100
inu	ıx Ker	mel v2	2.6.13 Co	ıfigu	ration		l
					— Charact	er devices ————	ľ
	Arrow	keys	navigate	the r	nenu. <en< td=""><td>ter> selects submenus>.</td><td></td></en<>	ter> selects submenus>.	
	Highl	ighted	l letters	are 1	notkeys.	Pressing <y> includes, <n> excludes,</n></y>	
1	<n> m</n>	odular	izes feat	ures	. Press <	Esc> <esc> to exit, <? > for Help, </esc>	
	for S	earch.	Legend	[*]	built-in	[] excluded <m> nodule < > module</m>	
1	2					1	
			atchdog	Cards	s>		
		< >	/ev/nvra	am suj	pport		
		< >	nhanced	Real	Time Clock	k Support	
		[*]	S3C2410 F	TC Di	river		
		<*>	Q2440 LH	EDs Di	river		
		<n></n>	Q2440 He	2110 1	Module sam	ple	
		<*>	QQ2440 B	itton	S		
		< >	Double Ta	ilk P	C internal	speech card support	
1	L						
				Sele	ct> < E	xit > < Help >	
<u> </u>							

5.5.11 シリアルポート





5.5.12 RTC

 $\checkmark = -$ ^r Device Drivers \rightarrow ^r Character devices - - -> \rightarrow

「Enhanced Real Time Clock Support」を選択しません





5.5.13 yaffs ファイルシステム

 $\lambda \equiv \neg \neg$ File Systems - - -> $\neg \rightarrow$ \neg Miscellaneous filesystems ---> \downarrow







5.5.14 EXT2/VFAT/ NFS ファイルシステム

メニュー「File Systems - - ->」のなかで

1		
Arrow	w keys navigate the menu. <enter> selects submenus>.</enter>	
High	lighted letters are hotkeys. Pressing <y> includes, <n> excludes,</n></y>	
<m> 1</m>	nodularizes features. Press <esc><to <?="" exit,=""> for Help, </to></esc>	
for a	Search. Legend: [*] built-in [] excluded <m> module < ></m>	
	<pre><*> Second extended is support</pre>	
	Ext2 extended attributes	
	[] X12 POSIX ACCESS CONTROL LISTS	
	j xt2 execute in place support	
	<pre>xt3 journalling file system support</pre>	
	<pre>< > Leiserfs support</pre>	
	<pre>< > JFS filesystem support</pre>	
	FS support>	
	<pre>< > Minix fs support</pre>	
L		
5		

自動的なマウントをサポートします。





メニュー「DOS/FAT/NT Filesystems」でFATをサポートします。



Linux Kernel v2.6.13 Configuration DOS/VAT/NT Filesystems Arrow keys navigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc> to exit, <? > for Help, for Search. Legend: [*] built-in [] excluded <m> module < > <</m></esc></m></n></y></enter>	
DOS/FAT/AT Filesystems Arrow keys navigate the menu. <enter> selects submenus>. Highlighted letters are hotkeys. Pressing <y> includes, <n> excludes, <m> modularizes features. Press <esc><esc> to exit, <? > for Help, for Search. Legend: [*] built-in [] excluded <m> module < > <!--</th--><th></th></m></esc></esc></m></n></y></enter>	
(1508859-1) Terault locharset for FAT < > NTFS file system support	
<pre><select> < Exit > < Help ></select></pre>	

メニュー「File System」→「Pseudo filesystems - - ->」 これを選択しないと、yaffsファイルシステムが動けません。





メニュー「File System」→「Network File Systems - - ->」 これを選択しないと、NFSを利用できません。





5.6 yaffs ルートファイルシステムのイメージを生成

1. yaffsイメージを生成するツールを解凍します。

tar xvzf mkyaffsimage.tgz -C /usr/sbin

2. デフォルトルートファイルシステムを解凍します。

tar xvzf root_default.tgz

0	Y	roo	t@cap	cross	/opt/Friendly/	ARM	/mini2440	-	X	
	E	ile	<u>E</u> dit	View	ı <u>T</u> erminal	<u>G</u> o	<u>H</u> elp			
	[r ex ke [r	oot amp rne oot	@capc les 1-2.6 @capc	ross .13 ross	mini2440]# root_defau root_mizi mini2440]#	ls lt	root_qtopia_mouse root_qtopia_tp	ŝ	•	
									*	



3. 開発された実行ファイルをコピーします

cp examples/hello/hello root_default/sbin

4. yaffsイメージを生成します

mkyaffsimage root_default root_default.img

💙 root@capcross:/opt/FriendlyARM/mini2440 📁 🗖	×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal <u>G</u> o <u>H</u> elp	
Object 653, root_default//opt/sample2.ogg is a file, 8 data chunks written	•
Object 654, root_default//opt/sample3.ogg is a file, 10 data chunks written	
Object 655, root_default//opt/data is a directory	
Object 656, root_default//opt/data/README is a file, 1 data chunks written	
Object 657, root_default//opt/data/alien.gif is a file, 1 data chunks written	
Object 658, root_default//opt/data/background.gif is a file, 9 data chunks writt	
en	
Object 659, root_default//opt/data/explode.wav is a file, 11 data chunks written	
Object 660, root_default//opt/data/explosion.gif is a file, 1 data chunks writte	
n	
Object 661, root_default//opt/data/music.it is a file, 217 data chunks written	
Object 662, root_default//opt/data/music.wav is a file, 111 data chunks written	
Object 663, root_default//opt/data/player.gif is a file, 1 data chunks written	
Object 664, root_default//opt/data/shot.gif is a file, 1 data chunks written	
Object 665, root_default//opt/data/shot.wav is a file, 7 data chunks written	
Object 666, root_default//media is a directory	
Object 667, root_default//media/note.jpg is a file, 79 data chunks written	
Object 668, root_default//media/shanghaitan.mp3 is a file, 5973 data chunks writ	
ten	
Object 669, root_default//media/flower.jpg is a file, 252 data chunks written	
Operation complete.	
413 objects in 27 directories	
30573 NAND pages	4
[root@capcross mini2440]# [*

5.7 Linux ドライバの開発入門

Linuxなどの現代的なOSでは、デバイスに対する入出力はデバイスドライバを通じて行うの が常識です。Linuxは「特権モード」を使い、カーネルモードとユーザーモードを厳密に分 離しています。ユーザーモードからは、物理メモリアドレスやI/Oポートなどへのアクセス はできません。したがって、デバイスに対する入出力は、カーネルモードで動作するドラ イバを通じて行うしかないです。

ある例を通じて、カーネルモードで動作するドライバの設計を紹介します。

5.7.1 簡単なドライバのソースコード

 $\mathcal{V}-\mathcal{A} \exists -\mathcal{F}$: kernel-2.6.13/drivers/char/qq2440_hello_sample.c



```
#include <linux/kernel.h>
#include <linux/module.h>
MODULE_LICENSE("GPL");
static int __init qq2440_hello_module_init(void)
{
        printk("Hello, QQ2440 module is installed !¥n");
        return 0;
}
static void __exit qq2440_hello_module_cleanup(void)
{
        printk("Good-bye, QQ2440 module was removed!¥n");
}
module_init(qq2440_hello_module_init);
module_exit(qq2440_hello_module_cleanup);
```

5.7.2 コンフィグファイルを編集します

kernel-2.6.13/drivers/char/Kconfigを開きます。下の内容を添加します(実は、添加完了しました、確認してみます)。





kernel-2.6.13でmake menuconfigを実行して、メニューDevice Drivers → Character devicesを選んで、



添加されたものが見えます。spaceキーで「M」を選択します。



5.7.3 Makefile を編集

kernel-2.6.13/drivers/char/Makefile を開きます。下の内容を添加します(実は、添加完了 しました、確認してみます)。

🗖 192. 168. 1. 110 – CRT 📃 🗖 🔀
<u>F</u> ile <u>E</u> dit <u>Y</u> iew <u>Options</u> <u>Transfer</u> <u>Script</u> <u>Mindow</u> <u>H</u> elp
13 X 4 X = = = < 5 5 5 6 2 8 2 9
<pre>obj-\$(CONFIG_SONYPI) += sonypi.o obj-\$(CONFIG_RTC) += rtc.o obj-\$(CONFIG_ERTC) += hpet.o obj-\$(CONFIG_GEN_RTC) += genrtc.o obj-\$(CONFIG_SGI_DS1286) += ds1286.o obj-\$(CONFIG_SGI_DS1286) += ds1286.o obj-\$(CONFIG_SGI_IP27_RTC) += ip27-rtc.o obj-\$(CONFIG_SG2410_RTC) += s3c2410-rtc.o obj-\$(CONFIG_S8c2440_LEDS) += sbc2440_leds.o obj-\$(CONFIG_S8c2440_LEDS) += sbc2440_leds.o obj-\$(CONFIG_S8c2440_LEDS) += sbc2440_leds.o obj-\$(CONFIG_GENERIC_NURAM),y) obj-\$(CONFIG_GENERIC_NURAM),y) obj-\$(CONFIG_GENERIC_NURAM),y) obj-\$(CONFIG_NURAM) += generic_nuram.o ndf obj-\$(CONFIG_NURAM) += nuram.o obj-\$(CONFIG_IDSHIBA) += toshiba.o obj-\$(CONFIG_ISK) += i8k.o obj-\$(CONFIG_S1628) += ds1628.o obj-\$(CONFIG_S1628) += ds1628.o obj-\$(CONFIG_S1648) +</pre>
INSERT Ready Telnet 11, 59 23 Rows, 106 Cols Linux

5.7.4 ドライバをコンパイルします

kernel-2.6.13でmake modulesを実行します。kernel-2.6.13/drivers/char/でオブジェクトフ ァイルqq2440_hello_module.koを生成させます。



¥ 100	t@loca	alhost:/	opt/Friendly	ARM/0	2440/kernel-2.6.13	11111111111111111111111111111111111111	///// -	
<u>F</u> ile	<u>E</u> dit	View	<u>T</u> erminal	Go	<u>H</u> elp			
[root	loca	lhost	kernel-2.	6.13]	make modules			*
CHK		includ	le/linux/v	ersic	.h			
make[1	1]: `	arch/a	rm/kernel	/asm-	ffsets.s' is up to	date.		
make[1	1]: `	includ	le/asm-arm	/mach	types.h' is up to d	date.		
CC	[M]	driver	s/char/qq	2440_	ello_module.o			
Bui	lding	modul	es, stage	2.				
MODI	POST							
CC		driver	s/char/qq	2440_	ello_module.mod.o			
LD		driver	s/char/qq	2440_	ello_module.ko			
lroot	aloca	Inost	kerne1-2.	6.13]	is drivers/char/qc	q2440_hello_module.*		
driver	rs/ch	ar/qq2	440_hello	_modu	e.c drivers/cf	nar/qq2440_hello_modul	e.mod.o	
driver	rs/ch	ar/qq2	440_nello	_modu	e.ko arivers/cr	nar/qq2440_ne110_modul	e.o	
Iroot	aloca	ar/qq2	kornol 2	_modu	e.mou.c			
[LOOL(aloca	most	kerner-2.	0.15]				
								1
L								•

5.7.5 ARM9 ボードでドライバをインストールします ドライバをロードします。

#insmod qq2440_hello_module.ko

ドライバを削除します。 #rmmod qq2440_hello_module.ko



🌯 ttyS0 - 超级终端	
文件 (E) 编辑 (E) 查看 (V) 呼叫 (C) 传送 (E) 帮助 (H)	
	~
qq2440 hello module.ko	
[root@FriendlyARM plg]# insmod qq2440_hello_module.ko	
[root@FriendlyARM plg]# rmmod qq2440_hello_module	
Good-bye, QQ2440 module was removed!	
[гоосегтепотонки рід]#	
	>
已连接 0:00:31 ANSIW 115200 8-N-1 SCROLL CAPS NUM 描 打印	.:



6.1 NOR Flash から起動

ARM9ボードのジャンパーJ1をNor Flashに設定して、電源を入れて、ARM9ボードはNor Flashから起動します。



6.2 USB ドライバのインストール

開発されたOSとプログラムをUSB通じてQQ2240v3にダウンロードします。その為、USB ケーブルでQQ2240v3のUSBスレーブポートとパ ソコンのUSBポートを繋ぐことが必要 です。繋ぐと、パソコンは新しいデバイスを発見して、USBドライバをインストールしま す。

















USBドライバをインストール完了あと、パソコンのダウンロード・ツールDNW.exeを実行 して、QQ2240v3とパソコンを繋ぐことが確認できます。



USBドライバはバッグがあります。ARM9ボードが再起動、又はリ セットの時、ホスト側は死んだかもしれません。その原因で、ARM9 ボードが起動完了した後、USBケーブルでホストを繋ぎます。

6.3 NAND Flash のパーティション

メニューの中で、機能号[x]を選択して、NAND Flash のパーティション画面が出てきます。 NAND Flash の中にエラーアリアがあるかもしれません。使用の影 響がありません。



COE1 (1) - CRT		
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>O</u> ptions <u>T</u> ransfer	<u>S</u> cript <u>M</u> indow <u>H</u> elp	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	5 🖨 🖆 💥 💡 🔍	
<pre>[t] Print the TOC struct of [q] Goto shell of vivi Enter your selection: x doing partition size = 0 size = 327680 size = 2424832 check bad block part = 0 end = 327680 part = 1 end = 2424832 part = 2 end = 67108864 part0:</pre>	F wince	
bad_block = 0 part1: offset = 327680 size = 2097152 bad_block = 0 part2:		
offset = 2424832 size = 64667648 bad_block = 0 ###### FriendlyARM BIOS for [x] bon part 0 320k 2368k [v] Download vivi	2440 #####	
Ready	Serial: COM1 27, 23 27 Rows, 73 Cols VT	100 NVM //

6.4 ブートロードの書き込み

メニューの中で、機能号[x]を選択して、NAND Flashのパーティション画面が出てきます。

パソコンで DNW を実行します。



DNW のタイトルに[USB: OK]があれば、パソコンと ARM9 ボードを USB で繋ぎました。



メニューの中で、機能号[v]を選択して、

🎨 ttyS0 –	
<pre>##### FriendlyARM BIOS for 2440 ##### [x] bon part 0 320k 2368k [v] Download vivi [k] Download linux kernel [y] Download root_cramfs image [c] Download root_cramfs image [n] Download Nboot [e] Download Bboot [i] Download WinCE NK.nb0 [w] Download WinCE NK.bin [d] Download & Run [f] Format the nand flash [p] Partition for Linux [b] Boot the system [s] Set the boot parameters [t] Print the TOC struct of wince [q] Goto shell of vivi Enter your selection: v USB host is connected. Waiting a download.</pre>	
0:00:16 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕打印	

DNW を待っています。DNW のメニュー「USB Port」→「Transmit」を選択して、





ブートロード supervivi を選択します。

ファイルを開く						? 🔀
ファイルの場所の:	ince5.0		-	🇢 🖻 💣	.	
よび使ったファイル して デスクトップ マイドキュメント マイニンピュータ	EBOOT_2440.nbl NBOOT.bin NK_A70.bin NK_A70.nb0 NK_N35.bin NK_N35.nb0 readme.txt supervivi_mini24	40				
ر -راوند ای		<u></u>				
	ファイル名(11):	supervivi_mini2440		1	- L	開((())
	ファイルの種類(工):	All Files (*,*)		1	-	キャンセル

ブートロードを書き込み完了すると、自動的にメニューに戻ります。



6.5 Linux のカーネルの書き込み

a. メニューの中で、機能号[k]を選択して、

	_ ^
Erasing done	
Writing done	
Written 117080 bytes	
##### FriendlyARM BIOS for 2440 #####	
[x] hon part 0 320k 2368k	
[V] Download vivi	
[k] Download linux kernel	
[y] Download root yaffs image	
[C] Download root_cramfs image	
[n] Download Nhoot	
[e] Download Eboot	
[i] Download WinCE NK.nb0	
[w] Download WinCE NK.bin	
[d] Download & Run	
[f] Format the nand flash	
[p] Partition for Linux	
[b] Boot the system	
[5] Set the Boot parameters	
[a] Goto shell of vivi	
Enter your selection: k	
USB host is connected. Waiting a download.	
	<u> </u>

カーネルをダウンロードすることを待っています。

b. DNW のメニュー"USB Port → Transmit"を選択して、 カーネルファイル zImage を転送 します

c. 転送完了したら、自動的にメニューに戻ります。



6.6 ルート・ファイルシステムの書き込み

a. メニューの中で、機能号[y]を選択して、

🇞 ttyS0 –	
Erasing done	^
Writing done Written 1556628 bytes	
##### FriendlyARM BIOS for 2440 ##### [x] bon part 0 320k 2368k	
[v] Download vivi [k] Download linux kernel	
[Y] Download root_yarrs image [c] Download root_cramfs image [n] Download Whoot	
[e] Download Eboot [i] Download WinCE NK.nb0	
[w] Download WinCE NK.bin [d] Download & Run	
[f] Format the nand flash [p] Partition for Linux	
<pre>[b] Boot the system [s] Set the boot parameters [t] Print the TOC struct of wince</pre>	
[q] Goto shell of vivi Enter your selection: y USB host is connected. Waiting a download.	
0:04:31 ANSIW 115200 8-N-1 SCROLL CAPS NUM 猜 打印	

ルート・ファイルシステムをダウンロードすることを待っています。

b. DNW のメニュー"USB Port → Transmit"を選択して、ルート・ファイルシステム root_default.imgを転送します

c. 転送完了したら、自動的にメニューに戻ります。

電源を切って、QQ2440v3の起動 J1 を NAND Flash で起動に設定してください。再び電 源を入れて、NAND Flash で書き込み済みの Linux は起動します。



6.7 NAND Flash のバックアップ

新ブートロード supervivi のみ

メニューの中で、機能号[u]を選択して、

🧠 ttysu - 超级终端 📃 🗆 🖌	K	
文件 (E) 编辑 (E) 查看 (V) 呼叫 (C) 传送 (T) 帮助 (H)		
🗅 🖙 🍘 🕉 🗈 🎦 🖆		
[n] Download Nboot	^	
[i] Download WinCE NK.nb0 [w] Download WinCE NK.bin		
[d] Download & Run [f] Format the nand flash		
[b] Boot the system [s] Set the boot parameters		
[t] Print the TOC struct of wince [u] Backup NAND Flash to HOST through USB(upload)		
[r] Restore NAND Flash from HOST through USB [q] Goto shell of vivi Enter your selection: y		
Backup Information:		
Start Hoor : 0x0 End Addr : 0x4000000		
bBackup00B : 1		
bCheckBad : 1 dwDachwyTatallan : Awh2000000		
dwBackuprotailen : 0x4200000 dwReservedBlks : 20		
dwEPInPktSize : 32		
Use dnw.exe to receive backup.		
-		
	~	

DNW のメニュー「Usb Port」→「Backup NandFlash to File」を選択します。



DNV v0.	504 友善	之臂改进版 (增加备份功能)	[CO II:x	
Serial Port	USB Port	Configuration Help		
	Transmi	t/Restore		^
	Backup	NandFlash to File		
	Status			
				~

バックアップのファイルの名前「backup.bin」を入力して

文件(I) 编辑(II) 查看(V) 呼 Seriel Port USB Port Configuration Help	-
Image: Second system Flags : 0xc5 size : 64MB [i] Download WinCE NH erasesize : 16KB [w] Download WinCE NH oobblock : 512	^ ^
[d] Download & Run oobsize : 16 [z] Download zImage i ecctype <td: 0x2<="" td=""> [g] Boot linux from F eccsize <td: 256<="" td=""> [f] Format the nand f Backup Information:</td:></td:>	
[p] Partition for Lin Start Addr : 0x0 [b] Boot the system End Addr : 0x4000000 [s] Set the boot para bBackup00B : 1 [t] Print the TOP str bbackup00B : 1	
[U] Bac Backuping to C:\Documents and Settings\pig\ 800 [r] Res [q] Got Enter y	
Backup Information: times Start Addr End Addr	~
bCheckBad : 1 dwBackupTotalLen : 0x4200000 dwReservedBlks : 20 dwEPInPktSize : 32	
Use dnw.exe to receive backup. Send Flash Backup: [====================================	
▲	>

バックアップ完了したら、次の画面:



📟 DNW ▼0.50A 友善之臂改进版(増加备份功能) [COM:x][🔳 🗖	×
Serial Port USB Port Configuration Help	
USB Backup Start	^
Nand Flash Information:	
type : 0x4	
flags : 0xc5	
size : 64MB	
erasesize : 16KB	
oobblock : 512	
oobsize : 16	
ecctype : 0x2	
eccsize : 256	
Backup Information:	
Start Addr : UXU	Ξ
End Addr : 0x4000000	
bBackupOOB : 1	
bCheckBad : 1	
dwBackupTotalLen : 0x4200000	
dwReservedB1ks : 20	
dwEPInPktSize : 32	
dnw.exe read data 65536 bytes a time, has to read 1056	
times	
======================================	

生成されたバックアップファイルの大きさは 66MB ぐらいです。

6.8 NAND Flash のリストア

新ブートロード supervivi のみ

メニューの中で、機能号[r]を選択して、



🌯 ttyS0 - 超级终端	
文件(E)编辑(E) 查看(V) 呼叫(C) 传送(E) 帮助(H)	
D 🖆 🍘 🐉 🗈 🎦 📸	
<pre>[v] Download vivi [k] Download linux kernel [v] Download root_yaffs image [c] Download root_cramfs image [a] Absolute User Application [n] Download Nboot [e] Download Bboot [i] Download WinCE NK.nb0 [w] Download WinCE NK.nb0 [w] Download & Run [z] Download & Run [z] Download zImage into RAM [g] Boot linux from RAM [f] Format the nand flash [p] Partition for Linux [b] Boot the system [s] Set the boot parameters [t] Print the TOC struct of wince [u] Backup NAND Flash to HOST through USB(upload) [r] Restore NAND Flash from HOST through USB [q] Goto shell of vivi Enter your selection: r USB host is not connected yet. USB host is connected. Waiting a download. -</pre>	
己连接 0:46:04 ANSIW 115200 8-W-1 SCROLL CAPS NUM 捕 打印	

DNW のメニュー「Usb Port」→「Transmit/Restore」を選択します。

DNV v0.	50A 友善之臂改进版(增加备份功能) [COII:x][📃	
Serial Port	USB Port Configuration Help	
=======	Transmit/Restore =============	^
Nand Flast	Backup NandFlash to File	
	Status	
Backup Inf dnw.exe re times ====================================	size : 64MB erasesize : 16KB oobblock : 512 oobsize : 16 ecctype : 0x2 eccsize : 256 formation: Start Addr : 0x0 End Addr : 0x4000000 bBackup00B : 1 bCheckBad : 1 dwBackupTotalLen : 0x4200000 dwReservedBlks : 20 dwEPInPktSize : 32 ead data 65536 bytes a time, has to read 1056	
		*

バックアップされたファイル「backup.bin」を選択します。



🗞 ttySO - 超级终端	
文件(12) 编辑(12) 查看(12)	🚥 DHW 👓.50A 友善之替改进版(增加备份功能) [COM:x][🛄 🗖 🔀
C 📽 🍘 🐉 🗠 🎦	Serial Port USB Port Configuration Help
<pre>[i] Download WinGl [w] Download WinGl [d] Download & Run [z] Download & Run [z] Download zImay [g] Boot linux fro [f] Format the nam [p] Partition for [b] Boot the syste [s] Set the boot p [t] Print the TOC [u] Backup NAND F: [r] Restore NAND F [r] Restore NAND F [r] Goto shell of Enter your select: USB host is conner Now, Downloading Flash params: oobs 67108864 Erasing and programs Block erasing(add)</pre>	Nand Flash Information: type : 0x4 flags : 0xc5 size : 64MB erasesize : 16KB oobblock : 512 oobsize : 16 ecctype : 0x2 Backup Inf Downloading C:\Documents and Settings\plg\ Backup Inf DBackupTotalLen : 0x4200000 dwReservedBlks : 20 dwEPInPktSize : 32 dnw.exe read data 65536 bytes a time, has to read 1056 times
0x 003a 0000/ 0	0233 0×00000000/00000 00232/04096=05%
*	
已连接 0:47:31 自动检测	115200 8-N-1 SUBULL CAPS NUM 3# 3150

6.9 メモリで Linux カーネルを直接に実行

一般的に Linux のカーネルを NAND Flash に書き込み、実行させます。毎回 Linux カーネルを更新すれば、NAND Flash も更新することが必要です。デバッグの時、不便です。
 ブートロード Suppervivi は Linux カーネルをメモリにロードして、直接に実行します。

- 1. 電源を切って、QQ2440v3の起動 J1 を Nor Flash で起動に設定してください。再び電源を入れて、Nor Flash で起動します。
- 2. Supervivi のメニューの中で、機能号[z]を選択して、


文件 ① 编辑 ② 查看 ② 呼叫 ② 传送 ① 帮助 ④	🎨 ttyS0 - 超级终端	
Image:	文件 (E) 编辑 (E) 查看 (Y) 呼叫 (C) 传送 (E) 帮助 (H)	
<pre>##### FriendlyARM BIOS for 2440 ##### [x] bon part 0 320k 2368k [v] Download vivi [k] Download vivi [k] Download root_vaffs image [c] Download root_cramfs image [a] Absolute User Application [n] Download Eboot [e] Download Eboot [i] Download WinCE NK.nb0 [w] Download WinCE NK.nb0 [w] Download & Run [z] Download & Run [z] Download zImage into RAM [g] Boot linux from RAM [f] Format the nand flash [p] Partition for Linux [b] Boot the system [s] Set the boot parameters [t] Print the TOC struct of wince [u] Backup NAND Flash to HOST through USB(upload) [r] Restore NAND Flash from HOST through USB [[q] Goto shell of vivi Enter your selection:</pre>	다 🖆 🍘 🐉 🗈 🖰 📸	
<pre>##### FriendlyARM BIOS for 2440 ##### [x] bon part 0 320k 2368k [v] Download vivi [k] Download linux kernel [y] Download root_yaffs image [c] Download root_cramfs image [a] Absolute User Application [n] Download Nhoot [e] Download WinCE NK.nb0 [v] Download WinCE NK.bin [d] Download WinCE NK.bin [d] Download & Run [z] Download zImage into RAM [g] Boot linux from RAM [f] Format the nand flash [p] Partition for Linux [b] Boot the system [s] Set the boot parameters [t] Print the TOC struct of wince [u] Backup NAND Flash to HOST through USB(upload) [r] Restore NAND Flash from HOST through USB [q] Goto shell of vivi Enter your selection:</pre>		
	<pre>##### FriendlyARM BIOS for 2440 ##### [x] bon part 0 320k 2368k [v] Download vivi [k] Download linux kernel [y] Download root_yaffs image [c] Download root_cramfs image [a] Absolute User Application [n] Download Nboot [e] Download Whoet [e] Download WinCE NK.nb0 [v] Download WinCE NK.nb0 [v] Download WinCE NK.bin [d] Download & Run [z] Download zImage into RAM [g] Boot linux from RAM [f] Format the nand flash [p] Partition for Linux [b] Boot the system [s] Set the boot parameters [t] Frint the TOC struct of wince [u] Backup NAND Flash to HOST through USB(upload) [r] Restore NAND Flash from HOST through USB [g] Goto shell of vivi Enter your selection:</pre>	

- 3. DNW のメニュー"USB Port → Transmit"を選択して、カーネルファイル zImage を転送します
- 4. 転送完了したら、自動的に Supervivi のメニューに戻ります。機能号[g]を選択して、 linux カーネルを実行させます。この画面が出たら:

🏶 ttyS0 - 超级终端	×
文件 (2) 编辑 (2) 查看 (Y) 呼叫 (C) 传送 (2) 帮助 (H)	
S3c2410-sdi s3c2410-sdi: running at 198kHz (requested: 197kHz). s3c2410-sdi s3c2410-sdi: s3cmci_request: no card s3c2410-	
yaffs: Aftempting MID mount on 31.2, "mtdblock2" VFS: Mounted root (yaffs filesystem). mount_devfs_fs(): unable to mount devfs, err: -2 Freeing init memory: 144K Warning: unable to open an initial console. Kernel panic - not syncing: No init found. Try passing init= option to kernel. -	~
已连接 0:18:25 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印	.:



ルートファイルシステムが見つかりませんでした!

Superviviのメニューの中で、機能号[y]を選択して、NAND Flash に root_default.img を 書き込みます。ルートファイルシステムを作ります。又は、NFS をルートファイルシス テムとして指定します。

Linux カーネルを実行させる前に、NFS を指定します。Supervivi のメニューの中で、機能号[q]を選択して、次のコマンドを入力してください。

Supervivi>param set linux_cmd_line ''console=ttySAC0 root=/dev/nfs nfsroot=192.168.1.111:/root_nfs

ip=192.168.1.70:192.168.1.111:192.168.1.111:255.255.255.0:QQ2440.arm9.net:eth0: off'' (NFSの設定)

Supervivi> boot ram (メモリでカーネルを起動させます)



第七章 NOR Flash のブートロードを更新

一般的に NOR Flash のブートロードを更新することが必要ないです。

NOR Flash は H-JTAG というツールで更新されます。

H-JTAGはARMの為のJTAGエミュレータです。AXD又はkeilをサポートします。デバッグ のスピードも速いです。詳しい情報はこちらです。 <u>http://www.hjtag.com</u>

弊社はH-JTAGのハードウェアを提供しております。パソコンはLTPが必要です。

7.1 H-JTAG をダウンロードとインストールします

ホームページ<u>http://www.hjtag.com</u>から最新版をダウンロードできます。

H-JTAGの特性:

- a. RDI 1.5.0 & 1.5.1 をサポートします;
- b. ARM7 & ARM9 (ARM9E-SとARM9EJ-Sを含む);
- c. thumb & arm 命令;
- d. little-endian & big-endian;
- e. semihosting;
- f. 実行環境WINDOWS 9.X/NT/2000/XP;
- g. flashの書き込み



弊社は H-JTAG のハードウェアを提供しております。パソコンは LTP が必要です。

インストール完了すれば、デスクトップで H-JTAG と H-Flasher を生成します。H-JTAG を実行すると、このエラーメッセージが出てきます。





設定しないから。"Ok"ボタンを押すと、初の画面が出てきます。



H-JTAG 𝔅𝔆=𝔅=𝔅=𝔅: Setting → Jtag Settings



Jtag Selection	User Defined Pin Assignme
C Wiggler (Predefined)	TMS Pin4 D2
C Sdt Jtag (Predefined)	TCK Pin2 D0 💌
User Defined	TDI Pin3 D1 💌
Reset Signal Output	TDO Pin11 Busy 💌
nTRST output inverted	nTRST NO TAP RST
nSRST output inverted	nSRST NO SYS RST -

この様な設定して、"Ok"ボタンを押します。

H-JTAG のメニュー : Script \rightarrow Init Script

ldx	Cmd	Width	Address	Value	
					1

[&]quot;Load"ボタンを押します。



ファイルを開く					? 🔀
ファイルの場所型:	H-JTAG	•	🗢 🔁	r 🔁	
Friendly ARM24	140 his				
ファイル名(N):	FriendlyARM2440.his			開	\odot
ファイルの種類(工):	H-Jtag Init Script (*.his)		•	キャン	セル

FriendlyARM2440.his というファイルを選択します。

次の画面が出	てき	ます	0
--------	----	----	---

ldx	Cmd	Width	Address	Value	
1	Setmern	32-Bit	0x53000000	0x00000000	
2	Setmem	32-Bit	0x4A000008	DxFFFFFFF	
3	Setmern	32-Bit	0x4A00001C	0x000007FF	
4	Setmern	32-Bit	0x53000000	0x00000000	
5	Setmem	32-Bit	0x56000050	0x000055AA	
6	Setmem	32-Bit	0x4C000014	0x0000007	
7	Setmem	32-Bit	0x4C000000	0x00FFFFFF	
8	Setmem	32-Bit	0x4C000004	0x00061012	
9	Setmem	32-Bit	0x4C000008	0x00040042	
10	Setmem	32-Bit	0x48000000	0x22111120	
11	Setmem	32-Bit	0x48000004	0x00002F50	-

"Ok"ボタンを押します。ご注意: "Enable Auto Init"をチェックしない。

パソコンと ARM9 ボードを H-JTAG で繋ぎます。ARM9 ボードの電源を入れます。 H-JTAG のメニュー: Operations → Detect Target を選択すると



H-JTAG Server	
File Operations Flasher Script Settings Options Hel	P
*	
ARM920T 0x0032409D	
Ready	

H-JTAG はターゲット ARM ボードを認識しました。

7.2 NOR Flash を書き込む

ARM9 ボードが NOR Flash から起動することを確認してください。

H-JTAG のメインメニュー「Flasher」→「Start H-Flasher」で H-Flasher を実行します。



🖻 H-Flasher		
New Load Save Save A:	s Options Exit About	
Program Wizard 1 Flash Selection 2 Memory Config 3 Init Script 4 Programming 7 H-Flasher Help	 Flash Selection AMD ATMEL ATMEL2 FUJITSU HYNIX PHILIPS SAMSUNG SST WINBOND 	Vendor: AMD PartNo: Type: Sector: Size: ID: Width:

H-Flasherのメインメニュー「Load」、H-Flasher_qq2440.hfcというファイルを開きます。

🔁 H-Flasher - H-Fl	asher_QQ2440.hfc	
New Load Save Save	As Options Exit About	
Program Wizard	>> Flash Selection	
• 22 1 C 1		

H-Flasherの左側の「4 Programming」を選択します。



🖬 H-Flasher - H-Flasher_QQ2440.hfc				
New Load Save Save.	As Options Exit About			
Program Wizard	>> Programming - AM29LV160DB			
1 Flash Selection 2 Memory Config 3 Init Script	Flash: Unchecked Target: Unchecked	Check		
 Programming H-Flasher Help 	Type: Auto Flash Download	Program		
	Dst Addr:			
	From: Entire Chip	Erase		
		DIGHK		

「Check」ボタンを押すと、QQ2440v3が使用したNor Flash(AM29LV160DB)を発 見します。

🖬 H-Flasher - H-Flasher_QQ2440.hfc					
New Load Save Save	As Options Ex	it About			
Program Wizard	>> Program	ning - AM29LV160DB			
1 Flash Selection 2 Memory Config	Flash: AM: Target: ARł	29LV160DB 0x22490001 4920T Little-Endian	Check		
3 Init Script					
4 Programming	Type: Au	o Flash Download 📃 💌	Program		
🤁 H-Flasher Help	Src File:				
	From: En	ire Chip	Erase		
	To: En	ire Chip	Blank		



「Type」の「Plain Binary Format」を選択します。

🔁 H-Flasher - H-Fl	asher_QQ2440.hfc	
New Load Save Save	As Options Exit About	
Program Wizard	>> Programming - AM29LV160DB	
 Flash Selection Memory Config Init Script 	Flash: Unchecked Target: Unchecked	Check
 Programming H-Flasher Help 	Type: Auto Flash Download Auto Flash Download Src File: Intel Hex Format Plain Binary Format Dst Addr:	Program
	From: Entire Chip	Erase Blank

書き込みのファイルsuperviviを選択します。「Dst Addr」で0を入力します。「Program」 ボタンを押すと、Nor Flashに書き込みます。



w Load Save Sav	e As Optior	ns Exit About	
rogram Wizard	>> Pro	gramming - AM29LV160DB	
Flash Selection	Flash:	Unchecked	Check
Memory Cor H-Fla	sher		
Init Scrip			
Programmir	supervivi		Program
H-Flasher	00.01.10.1		
	00:01:10 1:	5% 22 KB/S 512e = 123.2 KB	
		Stop	
i.e.	From:	Entire Chip	Erase

書き込み完了	の画面:
--------	------

H-Fla	sher	
	Programmed and verified succ	essfully.
	00:05:40 100% 23 KB/s	Size = 123.2 KB
		Close

付録 1 Qt/Embedded GUI プログラムを作る

ARM9ボードのGUIはQt/Embeddedを使用しています。他のGUIシステムも使えます。例 えばminiGUI, ucGUIなど

Qt/Embeddedは、ひとつのソースコード(シングルソース)から複数のプラットフォームで 稼働するアプリケーションを開発することができるC++ GUI ツールキットであり、Unix、 Linux、Windows、Mac OS X、組み込みLinux 用のGUI アプリケーションを構築するこ



とができます。

その優れた開発コンセプトは、広く世界中のソフトウェア技術者に評価されており、今日 では、 GPL 版、商用版合わせて、世界中に15 万人の開発者がいると言われています。ま た、Qt の採用実績は、大学教育、宇宙開発、特種映像技術、医療、科学技術シュミレショ ンと多岐にわたり、すでに数千にも及ぶ商用アプリケーションが開発されて います。

Qt Embedded は、組み込みLinux に特化して提供されるQt で、メモリやCPU などのマ シンリソースが少ない環境でも使用することができます。デスクトップ版のQt は描画の基 本的な部分にX サーバーを使用しますが、X はある程度のマシンリソースを必要とします ので、可能な限りハードウェアの実装を小さくしたり、消費電力を抑える必要がある製品 には向きません。Qt Embedded はLinux カーネルのAPI を使用し、フレームバッファへ 直接描画しますので、X を必要としません。また、使用する機能だけを使ってプログラム 構築することができますので、実行プログラムのフットプリントを最適化することもでき ます。

詳しい情報はこちらです: <u>http://trolltech.com/lang/japanese/qt-embedded</u>

三つのパッケージのソースコードが同じです。buildスクリプトだけが異なります。

1. ソースコードを解凍します:

#tar xvzf x86-qtopia.tgz -C /opt/FriendlyARM/mmini2440 #tar xvzf arm-qtopia.tgz -C /opt/FriendlyARM/mini2440 #tar xvzf ipaq-qtopia.tgz -C /opt/FriendlyARM/mini2440

パソコンで仮想実行するため、/etc/ld.so.confファイルを直します。次の内容に変更します。 /opt/FriendlyARM/mini2440/x86-qtopia/qt/lib /opt/FriendlyARM/mini2440/x86-qtopia/qtopia/lib /usr/kerberos/lib /usr/X11R6/lib



/usr/lib/sane /usr/lib/mysql

2. X86プラットフォームのQtopiaをコンパイルする

#cd /opt/FriendlyARM/mini2440/x86-qtopia #./build-all (長い時間がかかります、我慢してください) #./konq_to_qtopia (生成されたブラウザをQtopiaに入ります) #ldconfig (生成されたqtとqtopiaライブラリを有効します)

3. 仮想スクリーン

qvfbはX86プラットフォームの仮想スクリーンです。生成されたQtはこの仮想スクリーンで 動きます。 qvfbのインストールの方法: \$ tar zxvf qvfb⁻1.1.tar.gz \$ cd qvfb⁻1.1 \$./configure \$ make \$ su -c 'make install'

3. X86プラットフォームで仮想実行する

#. set-env ("."と"set-env"の間、スペースが必要です。) # qvfb & # qpe





qvfbのメニュー「File」→「Configure」で仮想スクリーンの分解能を設定できます。

4. Hello,Worldをコンパイルする

#cd /opt/FriendlyARM/mini2440/x86-qtopia/hello

#make

x86-qtopia/qtopia/binフォルダに実行ファイルhelloを生成します。



5. 単独にHello,Worldを実行する

#qvfb -width 640 -height 480 &

#hello -qws

Virtual framebuffer 640x480 32bpp Display :0	•	- 🗆 X
ile ⊻iew		Help
Test my first Otopia Application		
	1.5	
[]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]		
Hello, SBC-2410X		
<u>kanan ana ana ana ana ana ana ana ana an</u>		
Hello, Gtopia world!		
	Virtual framebuffer 640x480 32bpp Display :0 ile ⊻lew Test my first Otopia Application Hello, SBC-2410× Hello, Gtopia world!	Virtual framebulfer 640x480 32 bpp Display x0



6. Qtopiaの環境でHello,Worldを実行する

#cd hello

#cp hello.desktop /opt/FriendlyARM/mini2440/x86-qtopia/qtopia/apps/Applications
#qvfb -width 640 -height 480 &

#qpe



7. ARM9プラットフォームのQtopiaをコンパイルする

開発環境はRedhat9をお勧めします。他のLinuxではエラーがあるかもしれません。

クロス開発ツールはarm-linux-gcc-3.3.2です。他のgccはエラーがあるかもしれません。

#cd /opt/FriendlyARM/mini2440/arm-qtopia

- #./build-all(長い時間がかかります、我慢してください)
- #./mktarget_qtopia (target_qtopia.tgzを生成する)
- #./mktarget_konq(target_konq.tgzを生成する)

8. Hello,Worldをコンパイルする

#cd /opt/FriendlyARM/mini2440/arm-qtopia

#. set-env · .'と' set-env'の間に必ずスペースを入ります。



#cd hello

#make arm-qtopia/qtopia/binフォルダに実行ファイルhelloを生成します。

9. ARM9でhelloを実行します:

ARM9のコンソール:

#mv /mnt/opt/FriendlyARM/QQ2440/ipaq-qtopia/qtopia/bin/hello /opt/qtopia/bin #mv /mnt/opt/FriendlyARM/QQ2440/ipaq-qtopia/hello/hello.desktop /opt/qtopia/apps/Applications /mntはARM9とパソコンの共有フォルダです。

ARM9を再起動すれば、helloのアイコンが見えます。実行すると:

Test my first	Otopia Application	8
		~
	Hello	13
He	ello, Qtopia world!	
🚱 ab c 🖉 🔺 🔛		🕒 🗐 🐏 🗍 17:54

10. 単独にARM9でhelloを実行します:

ARM9のコンソール: export set HOME=/root export set QTDIR=/opt/qt export set QPEDIR=/opt/qtopia export set QWS_KEYBOARD="USB:/dev/input/event1" export set QWS_MOUSE_PROTO="USB:/dev/input/mouse0" export set PATH=\$QPEDIR/bin:\$PATH export set LD_LIBRARY_PATH=\$QTDIR/lib:\$QPEDIR/lib \$QPEDIR/bin/hello -qws

11. 自分で生成されたQtopiaをファイルシステムに組み込む

#cd /opt/FriendlyARM/mini2440/root_qtopia_mouse



#tar xvzf ../arm-qtopia/target_qtopia.tgz (Qtopiaを更新する)
#tar xvzf ../arm-qtopia/target_konq.tgz (ブラウザを更新する)
#mkyaffsimage root_qtopia_mouse my_qtopia.img
生成されたmy_qtopia.imgファイルをARM9ボードに書き込みます。

付録2 アプリケーションを移植

Linux環境にたくさんのオープンソースのアプリケーションがありますが、これらは殆んど パソコン(x86)用のものです。どうやってパソコン用のものをARM9に移植しますか。ある 例を通じて、紹介します。

ARM9ボードの起動の時、自動的にあるMP3をディスプレイします。このツールはmadplay です。 madplayのウェブサイトはこちらです。 <u>http://www.underbit.com/products/mad/</u>

このウェブサイトの紹介によって、madplayはライブラリlibmadとlibid3tagが必要です。 ウェブサイトからこの三つのソースファイルをダウンロードします。 madplay-0.15.2b.tar.gz libmad-0.15.1b.tar.gz libid3tag-0.15.1b.tar.gz **多分他のソースファイルも必要するかもしれません。移植中に問題があれば、だんだん添 加します。**

先ず、パソコンでコンパイルしましょう。

step1: ソースファイルをsrc-x86に解凍します mkdir src-x86 mkdir target-x86 tar xvzf libid3tag-0.15.1b.tar.gz -C src-x86 tar xvzf libmad-0.15.1b.tar.gz -C src-x86 tar xvzf madplay-0.15.2b.tar.gz -C src-x86

step2: ライブラリをコンパイルします cd ../src-x86/libid3tag-0.15.1b

./configure --prefix=/madplay-source/target-x86



make make install

cd ../ libmad-0.15.1b ./configure --prefix=/madplay-source/target-x86 make make install

コンパイル成功したら、target-x86でincludeとlibを生成します。

step3: madplayをコンフィグします cd madplay-0.15.2b ./configure --prefix=/madplay-source/target-x86 そうすると、エラーが出てきます。 configure: error: mad.h was not found **** You must first install libmad before you can build this package. **** If libmad is already installed, you may need to use the CPPFLAGS *** environment variable to specify its installed location, e.g. -I<dir>.

エラー情報によって、madplayをコンフィグする前にlibmadをインストールすることが必 要です。step2でlibmadのincludeを生成しましたから、CPPFLAGS環境変数を設定してみ ます。

./configure --prefix=/madplay-source/target-x86 CPPFLAGS=-I/madplay-source/target-x86/include

もうエラーが出てきます。

configure: error: libmad was not found
*** You must first install libmad before you can build this package.
*** If libmad is already installed, you may need to use the LDFLAGS
*** environment variable to specify its installed location, e.g. -L<dir>.

エラー情報によって、ライブラリの環境変数LDFLAGSの設定も必要です。もう一度コンフ ィグしてみます。

./configure --prefix=/madplay-source/target-x86 CPPFLAGS=-I/madplay-source/target-x86/include LDFLAGS=-L/madplay-source/target-x86/lib



今回コンフィグ成功しました。

step4: madplayをコンパイルします コンフィグ成功したら、コンパイルできます。 make make install

成功したら、target-x86/binで実行ファイルmadplayが見えます。

step5: madplayをテストします あるMP3ファイルをmadplayディレクトリにコピーします。 ./madplay test.mp3 このコマンドすると、MP3が聞こえます。

これまで、madplayをパソコンに移植完了しました。今までの操作ステップとコマンドも 覚えますか。忘れないように、これらコマンドをあるスクリプトに書きます。

#!/bin/sh

MADPLAY_DIR=\$PWD SRC_DIR=src-x86 TARGET_DIR=\$MADPLAY_DIR/target-x86

tar xvzf ./tarball/libid3tag-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/libmad-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/madplay-0.15.2b.tar.gz -C \$SRC_DIR

cd \$SRC_DIR/libid3tag-0.15.1b ./configure --prefix=\$TARGET_DIR make;make install cd ../../

cd \$SRC_DIR/libmad-0.15.1b ./configure --prefix=\$TARGET_DIR make;make install



cd ../..

cd \$SRC_DIR/madplay-0.15.2b ./configure --prefix=\$TARGET_DIR CPPFLAGS=-I\$TARGET_DIR/include LDFLAGS=-L\$TARGET_DIR/lib make;make install cd ../../

このスクリプトに基づいて、ARM9に移植してみよう。

ARMの移植なら、ARMのコンパイルツールを指定することが必要です。デフォルトコンパ イルツールはgccです。更新されたスクリプト(赤い文字はARMの為の更新):

#!/bin/sh

MADPLAY_DIR=\$PWD SRC_DIR=**src-arm** TARGET_DIR=\$MADPLAY_DIR/target-arm

tar xvzf ./tarball/libid3tag-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/libmad-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/madplay-0.15.2b.tar.gz -C \$SRC_DIR

export CC=arm-linux-gcc

cd \$SRC_DIR/libid3tag-0.15.1b ./configure --host=arm-linux --prefix=\$TARGET_DIR CPPFLAGS=-I\$TARGET_DIR/include LDFLAGS=-L\$TARGET_DIR/lib make;make install cd ../../

cd \$SRC_DIR/libmad-0.15.1b ./configure --host=arm-linux --prefix=\$TARGET_DIR make;make install cd ../..

cd $SRC_DIR/madplay-0.15.2b$



./configure --host=arm-linux --prefix=\$TARGET_DIR CPPFLAGS=-I\$TARGET_DIR/include LDFLAGS=-L\$TARGET_DIR/lib make;make install cd ../../

こ のスクリプトを実行すると、エラーも出てきます。原因はlibid3tagをコンパイル失敗し ました。libid3tagのコンパイルはライブラリ zlibに依頼します。x86の環境でzlibがあるの で、別途必要ないです。ARMの環境にはありません。ウェブサイトでzlibをダウンロード してみ よう。もう一度スクリプトを更新します(青い文字はzlibに関連します)。

#!/bin/sh

MADPLAY_DIR=\$PWD SRC_DIR=src-arm TARGET_DIR=\$MADPLAY_DIR/target-arm

tar xvzf ./tarball/libid3tag-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/libmad-0.15.1b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/madplay-0.15.2b.tar.gz -C \$SRC_DIR tar xvzf ./tarball/zlib-1.2.3.tar.gz -C \$SRC_DIR

export CC=arm-linux-gcc

cd \$SRC_DIR/zlib-1.2.3 ./configure --prefix=\$TARGET_DIR make && make install cd ../..

cd \$SRC_DIR/libid3tag-0.15.1b ./configure --host=arm-linux --prefix=\$TARGET_DIR CPPFLAGS=-I\$TARGET_DIR/include LDFLAGS=-L\$TARGET_DIR/lib make;make install cd ../../

cd \$SRC_DIR/libmad-0.15.1b ./configure --host=arm-linux --prefix=\$TARGET_DIR



make;make install cd ../..

cd \$SRC_DIR/madplay-0.15.2b ./configure --host=arm-linux --prefix=\$TARGET_DIR CPPFLAGS=-I\$TARGET_DIR/include LDFLAGS=-L\$TARGET_DIR/lib make;make install cd ../../

今回ARM環境の実行ファイルを生成しました。target-arm/binで実行ファイルmadplayが 見えます。生成されたmadplayとライブラリをFTPでARM9ボードの/usr/bin/と/usr/lib/に ダウンロードします。

ARM9ボードでmadplayをテストしてみよう!

付録 3 Watchdog の使い方

ウォッチドッグ タイマ (Watchdog Timer) は組み込みシステムの大切な機能です。システ ムフリーズ状態になった時点で再起動 (reboot)を実行させることができます。OSなし環 境でウォッチドッ グ タイマをよく使いますが、Linux環境でどうやって使用しますか。

step1:

Linuxのコンフィグ kernel-2.6.13でmake menuconfig メニューDevice Drivers → Character devices → Watchdog Cards → Watchdog Timer Support → S3C2410 Watchdogを"M"に設定します。 *'M' の設定はLinuxカーネルを更新しなくでも、ドライバをダイナミックロードすること できます。'*'に設定すると、ドライバはLinuxカーネルの一部 としてコンパイルします。 LinuxカーネルをARM9ボードのFlashに再び書き込みしなければなりません。

step2:

kernel-2.6.13でmake modulesコマンドでドライバをコンパイルします。 成功すれば、オブジェクトファイルkernel-2.6.13/drivers/char/watchdog/s3c2410_wdt.ko を生成します。

step3:



ARM9ボードでドライバをダイナミックロードします。 insmod s3c2410_wdt.ko mknod /dev/watchdog c 10 130

ls /dev/コマンドでwatchdogが見えるはずです。

step4:

ARM9ボードでwatchdogを起動します。 echo "@" > /dev/watchdog

長い間echo "@" > /dev/watchdogをしないと、watchdogが動作して、ARM9ボードを再起動 させます。

詳しいWatchdogの使い方、こちらに参照してください。 Linux カーネルのドキュメント: kernel-2.6.13/Documentation/watchdog/watchdog.txt

付録 4 AD の使い方

ソース:readadc.c 実行ファイル:readadc

ADCのドライバ ソース:sbc2440_adc_sample.c 実行ファイル:sbc2440_adc_sample.ko

ドライバのコンパイル: sbc2440_adc_sample.cをlinux/drivers/char/にコピーしてください。 linux/drivers/char/Kconfigファイルに次の内容を入力してください。

config SBC2440_ADC tristate "SBC2440 ADCs Driver" depends on ARCH_S3C2410 help



SBC2440 ADC.

linux/drivers/char/Makefileファイルに次の内容を入力してください。

obj-\$(CONFIG_SBC2440_ADC) += sbc2440_adc_sample.o

make menuconfigでADCのドライバが見えます。[M]を選択した、makeで再コンパ イルすれば、linux/drivers/char/にsbc2440_adc_sample.koを生成させます。

ARM9ボードでADCドライバのインストール: # insmod sbc2440_adc_sample.ko # mknod /dev/sbc2440_adc_sample0 c 236 0

ADCドライバのテスト #./readadc

付録 5 UVC(USB Device Class)Web カメラを使用

USB ビデオクラスとは

読み方∶ユーエスビービデオクラス 別名∶USB Video Device Class 【英】USB video class

USB ビデオクラスとは、USB 2.0 の規格が拡張されたもので、家庭用ビデオカメラ、またはライブカメラなどに用いられる Web カメラなどの、映像を扱う機器を USB でパソコンに接続するための規格のことである。

USB 規格を策定する組織である USB Implementers Forum によって策定された。 Motion-JPEG、DV、MPEG-2ないしはMPEG-4などの圧縮データを、あるいは非圧縮 のデータを、USB を通じて転送することが可能となる。

UVC カメラを ARM9 で使用できるソフト

tar zxvf arm-linux-uvc.tgz



arm-linux-uvc の中に次のファイルがあります。

uvcvideo-r104.tar.gz: uvcvideo ドライバのソースファイル。 mjpg_streamer_rev35.tgz: ストリーミングの配信用のソースファイル。

libjpeg62: MJPG-streamer をコンパイルするためのライブラリと.h ファイル

arm9-bin: クロスコンパイルされる実行ファイルです。 uvcvideo.ko: uvcvideo ドライバ *.so: MJPG-streamer のライブラリ mjpg_streamer: MJPG-streamer の実行ファイル

www: MJPG-streamer のホームページ

ソフトを ARM9 ボードにインストールする手順

これらのファイルを ARM9 ボードに FTP でアップロードします。 UVC(USB Device Class)に対応した Web カメラを ARM9 の USB ポートに接続して、 ARM9 ボードの環境で次のコマンドを入力します。

先ず、特定なディレクトリにファイルをコピーします。 # cp arm9-bin/*.so /usr/lib/ # cp arm9-bin/mjpg-streamer /usr/bin/

次に、uvccideo.koをカーネルに組み込みます。 # insmod arm9-bin/uvcvideo.ko uvcvideo: Found UVC 1.00 device USB 2.0 Camera (0c45:62c0) usbcore: registered new driver uvcvideo USB Video Class driver (v0.1.0)

最後に、以下のコマンドで MJPG-streamer を起動します。 # mjpg_streamer --input "input_uvc.so --device /dev/video0 --fps 5 --resolution 640x480 --yuv" --output "output_http.so --port 8080 --www /mjpg-streamer/www"



Web ブラウザで ARM9 ボードを見ましょう

Web ブラウザで、「http://ARM9 の IP アドレス):8080/」にアクセスすると、 MJPG-Streamer Demo Pages が表示されます。静止画、動画、および Pan/Tilt/LED の On/Off 等の制御をすることができます。 (Internet Exploler 6 及び 7 では、MJPEG によるストリーム(動画)を閲覧することができません。しかし、Javascript を使用したス トリーム(動画)は、 Internet Explorer でも閲覧することができます。)

Web ブラウザで見る様子:



付録 6 Mini GUI の移植とプログラム設計

1. MiniGUI 概要

MiniGUI は、複数のオペレーティング・システム対応の、組み込みデバイス向けグラフィ カル・ユーザ・インタフェース開発システムおよびミドルウェアとして代表的な ものです。 MiniGUI は、アジアの代表的なテレコムやエンターテインメント機器メーカーの多くに採 用されており、携帯デバイス(携帯電話、ポータブル・メディア・プレイヤー、 PDA) セットトップ・ボックス、医療機器、産業用制御システムおよび軍事システムにも幅広く



利用されています。

MiniGUI は、高速で安定性がある軽量 GUI フレームワークおよび開発システムで、リソ ースやリアルタイム・パフォーマンスに厳しい制約のあるアプリケーション向けに最適化 されています。 MiniGUI は成熟したマルチウィンドウイング / メッセージング・メカニ ズムを備え、LCD や YUV など様々なデバイスと連携し、拡張 GDI API (すなわち、ラ スター、複雑な描画 / 作図、2D グラフィックス)をサポートし、フラッシュ GUI 構築用 のスキン API を提供します。

MiniGUI は、static labels, ボタン, edit boxes(単一またはマルチライン), list and combo boxes, progress bars, property sheets, toolbars, track bars, tree, list, grid, icon views、月 例カレンダー, アニメーションといった、各種コントロール機能(ウィジェット)の包括的 なスイートを提供します。MiniGUI は、dialog boxes, message boxes, menus, acceleration keys, carets, timers を備えています。また、広く普及している画像ファイル・タイプ(す なわち、GIF、JPEG、BMP、PNG)、Windows リソースファイル(すなわち、Windows ビットマップ、アイコン、カーソル)、複数の文字セット、フォント・タイプ、キーボード・ レイアウトをサポートします。

詳しい情報はこちらです。 <u>http://www.minigui.org</u>

2. X86 プラットフォームで実行してみよう

MiniGUI は他のライブラリを使用していますので、コンパイルする前に、PNG/JPEG/TTF などライブラリをインストールすることが必要です。

2.1 PNG ライプラリをコンパイルとインストール

\$ tar zxvf libpng_src.tgz
\$ cd libpng
\$ cp scripts/makefile.linux Makefile
\$ make
\$ su -c 'make install'

2.2 JPEG ライブラリをコンパイルとインストール

\$ tar zxvf jpegsrc.v6b.tar.gz
\$ cd jpeg-6b





\$./configure\$ make\$ su -c 'make install'

2.3 TrueTypeFont ライブラリをコンパイルとインストール

\$ tar zxvf freetype-1.3.1.tar.gz
\$ mkdir -p libttf/extend
\$ cp freetype-1.3.1/lib/* freetype-1.3.1/lib/arch/ansi/* libttf/
\$ cp freetype-1.3.1/lib/extend/* libttf/extend/
\$ cp freetype-1.3.1/lib/* freetype-1.3.1/lib/arch/ansi/* libttf/
\$ cd libttf

\$ gcc -c -fPIC -O2 freetype.c
\$ gcc -c -fPIC -O2 -I./ extend/*.c
\$ gcc --shared -o libttf.so *.o

\$ su -c ' cp libttf.so /usr/local/arm/2.95.3/arm-linux/lib'

2.4 MiniGUI をコンパイルとインストール

\$ tar zxvf libminigui-1.6.10.tar.gz
\$ cd libminigui-1.6.10
\$./configure
\$ make
\$ su -c 'make install'

2.5 MiniGUI のソースをインストール

\$ tar zxvf minigui-res-1.6.10.tar.gz
\$ cd minigui-res-1.6.10
\$ su -c 'make install'

/etc/ld.so.conf ファイルを編集して、MiniGUI のインストール先/usr/local/lib を入ります。ル

ートアカウントで次のコマンドでシステムのライブラリを更新します。

Idconfig



2.6 仮想スクリーン

MiniGUIはX86プラットフォームで実行すれば、仮想スクリーンが必要です。 \$ tar zxvf qvfb-1.1.tar.gz \$ cd qvfb-1.1 \$./configure \$ make \$ su -c 'make install'

\$ qvfb &

qvfb のメニュー「File」→「Configure」で仮想スクリーンの分解能を設定できます。

2.7 MiniGUI のデモ

MiniGUI はデモ、ゲーム、サンプルがあります。 デモ:mde-1.6.10.tar.gz ゲーム:games-1.6.10.tar.gz

\$ tar zxvf mde-1.6.10.tar.gz
\$ cd mde-1.6.10
\$./configure
\$ make

フォルダの中で幾つの例を生成します。一つの例を実行してみよう。

\$ cd bomb

\$./bomb



4	Vi	rtu	al f	ran	neb	uff	er (640	x48	30	32 b	pp	Di	spl	aý	:0														-	
E	ile	$\underline{\vee}$	<u>(</u> iev	v																											<u>H</u> elp
Bc	շահ	<u>с</u>	lan	ne	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	×		
0	tart		aval	~	bou	ŀ																							_		
3	an	L.0	sver	~	Dou														_												
												6	2					-	91												
۲	۲											۰									•	۲		•	۰						
					•			۲	۰				۲							•							•				
	۲	۰						۰			1		۲	2					۲		•								٠		
1	2	2	1	1	۲		۲				•					۰		•	۲		ļ				•		ļ				
	<u> </u>		_	2	Ļ		•	•		Ļ		•					Ļ	Ļ	Ļ	Ļ	Ļ	Ļ		<u> </u>	Ļ	Ļ	Ļ				
\parallel	1	2	3	3		2	3	3		Ļ	2				•	H	<u> </u>		┡	Ļ	┞	┝		<u> </u>	┞						
	1				2	1	1		3	Ļ		_	_		-	-	┝	ļ.	┢	┞	┞	┢	ļ*		┝		⊢	I¥.			
1	1	3	4	3	1	-	1	1	3	h					-	-	┝	┝	┢	┝	┝		╘		H	F	┢	┢			
	1	1	1	1	-	-	-	1	3	ð							┢─	┢	 •	┝	┢	F	h	┢─	┢		┝				
1	1	1	1	1	1	-	-	1	1		۲				•		┢╴	i-		H	•	•	b	Ł	H	F	i-	•			
F	-	1	2	1	2	2	2	2	3	۲	Ė	Ē			İ	•		i	i i	Ē	i i	Ė		1	i	i	•	Ċ	Г		
	\square	1	1	2		•	۰												•			•			•	۲	۲	٠			
1	1	1	2															•			•		•								
۲	X			•				٠	۲	۲	۲			۲	۲						•			•			•				
		•			•	۲									•	۰	۲	۰		•											
_																															

Windows の画面みたいです。他の例をやってみましょう。

2.8 MiniGUI 環境のプログラム設計

MiniGUIのウェブサイトで<u>http://www.minigui.org/docs.shtml</u> 参考資料を提供しています。ご覧ください。 サンプル:mg-samples-1.6.10.tar.gz

3. ARM9 プラットフォームに移植する

インストール手順は X86 と大体同じです。ARM9 で幾つのライブラリがないので、添加することが必要です。MiniGUI のクロス開発ツールは 2.95.3 を使用します。

3.1 Z ライブラリをコンパイルとインストール

Z ライブラリは他のライブラリの基礎です。X86 のデフォルト設定は Z ライブラリがあり



ますが、ARM9 がありません。先ず、Z ライブラリをインストールします。 \$ tar zxvf zlib⁻1.2.3.tar.gz \$ cd zlib⁻1.2.3 \$ export CC=arm⁻linux⁻gcc #コンパイルツールを arm に変換する \$./configure --prefix=/usr/local/arm/2.95.3/arm⁻linux/ --shared \$ make \$ su ⁻c 'make install'

3.2 PNG ライブラリをコンパイルとインストール

\$ tar zxvf libpng_src.tgz
\$ cd libpng
\$ cp scripts/makefile.linux Makefile

Makefile ファイルを編集して、次の内容に変更してください CC=arm-linux-gcc prefix=/usr/local/arm/2.95.3/arm-linux ZLIBLIB = /usr/local/arm/2.95.3/arm-linux/lib ZLIBINC = /usr/local/arm/2.95.3/arm-linux/include

\$ make \$ su -c 'make install'

3.3 JPEG ライブラリをインストール

JPEG のソースパッケージの configure のなかにバッグがあるので、コンパイルするのは大 変です。そして、直接に arm-linux-gcc-3.3.2 の JPEG ライブラリをコピーします。 su -c 'cp /usr/local/arm/3.3.2/arm-linux/lib/libjpeg*.* /usr/local/arm/2.95.3/arm-linux/lib'

3.4 TrueTypeFont ライブラリをコンパイルとインストール

tar zxvf freetype-1.3.1.tar.gz mkdir -p libttf/extend cp freetype-1.3.1/lib/* freetype-1.3.1/lib/arch/ansi/* libttf/ cp freetype-1.3.1/lib/extend/* libttf/extend/ cp freetype-1.3.1/lib/* freetype-1.3.1/lib/arch/ansi/* libttf/ cd libttf arm-linux-gcc -c -fPIC -O2 freetype.c arm-linux-gcc -c -fPIC -O2 -I./ extend/*.c



arm-linux-gcc --shared -o libttf.so *.o su -c 'cp libttf.so /usr/local/arm/2.95.3/arm-linux/lib'

3.5 MiniGUI をコンパイルとインストール

\$ tar zxvf libminigui-1.6.10.tar.gz \$ cd libminigui-1.6.10

configure ファイルを編集して、次の内容を先頭に追加してください CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc CPP=/usr/local/arm/2.95.3/bin/arm-linux-ld LD=/usr/local/arm/2.95.3/bin/arm-linux-ar RANLIB=/usr/local/arm/2.95.3/bin/arm-linux-ranlib STRIP=/usr/local/arm/2.95.3/bin/arm-linux-strip

\$./configure --host=arm-linux --enable-jpgsupport=yes --enable-pngsupport=yes
--enable-gifsupport=yes --disable-lite --prefix=/usr/local/arm/2.95.3/arm-linux
\$ make
\$ su -c 'make install'

3.6 MiniGUI のソースをインストール

\$ tar zxvf minigui-res-1.6.10.tar.gz \$ cd minigui-res-1.6.10

config.linux ファイルを編集して、次のような変更します。 TOPDIR=/minigui/miniguitmp

\$ su -c 'make install'

3.7 popt ライブラリをコンパイルとインストール

poptは MiniGUI デモ用のライブラリです。

\$ tar zxvf popt-1.7.tar.gz

d = 1.7

\$./configure --prefix=/usr/local/arm/2.95.3/arm-linux/ --host=arm-linux --enable-shared --enable-static

\$ make



\$ su -c 'make install'

3.8 生成されたファイル ARM9 のフォルダにコピー

仮定 ROOT_NFS は ARM9 の NFS ルートファイルシステムです。

次のファイルを/usr/local/arm/2.95.3/arm-linux/lib から ROOT_NFS/usr/lib にコピーして

ください。

libjpeg.la	libminigui.so	libpopt.so.0
libjpeg.so	libm.so	libpopt.so.0.0.0
libjpeg.so.62	libm.so.6	libttf.so
libjpeg.so.62.0.0	libpng.so	libz.so
libm-2.2.3.so	libpng.so.2	libz.so.1
libminigui-1.6.so.10	libpng.so.2.1.0.12	libz.so.1.2.3
libminigui-1.6.so.10.0.0	libpopt.la	
libminigui.la	libpopt.so	

MiniGUI 用ソースをコピーします。

\$ cp --archive /minigui/miniguitmp/usr ROOT_NFS/

MiniGUI 用コンフィグファイル MiniGUI.cfg をコピーします。

 $cp/usr/local/arm/2.95.3/arm-linux/etc/MiniGUI.cfg ROOT_NFS/usr/local/etc$

MiniGUI.cfg を編集して、次のような変更します。

[system]										
# GAL engine and default option	ons									
gal_engine=fbcon										
# IAL engine										
ial_engine=dummy	#デモ用から、入力デバイスを指定しません									
mdev=none										
mtype=none										
[fbcon]										
defaultmode=240x320-16bpp	#3.5 インチの LCD、7 インチなら 800x480									

MiniGUI が動く環境は MiniGUI.cfg に設定されます。MiniGUI はあるファイルを見つからないと、MiniGUI.cfg を直してください。



3.9 デモのインストール

\$ tar zxvf mde-1.6.10.tar.gz
\$ cd mde-1.6.10
\$./configure --prefix=/\$ROOT_NFS/usr/local --host=arm-linux
\$ make

\$ mkdir -p /\$ROOT_NFS/usr/local/lib/shared/miniguiapps

\$ cp -r -a mde-1.6.10/* /\$ROOT_NFS/usr/local/lib/shared/miniguiapps 実は実行ファイルと res ファイルだけをコピーするのは十分です。

3.10 ARM9 で実行してみょう

ARM9 ボードは NFS で起動します。ホストの ROOT_NFS は ARM9 のルートファイルシ ステムです。

ARM9のコンソール:

 $\# \ cd \ /usr/local/lib/shared/miniguiapps$

cd bomb

./bomb

BO	Nomb Game														
Start Level About															
	<mark></mark>														
Ц		4													
		4													
Ц	_	Ļ													
Ц	_	4													
Ц	_	4													
Ц	_	4													
Ц	4	Ļ													
Ļ	4	Ļ										Ļ			
Ц	4	Ļ													
Ц	4	Ļ													
H	4	4										4			

ARM9ボードで同じ画面が出てきます。

若し次の故障があれば

GAL fbcon engine: Can't open /dev/tty0: No such file or directory



GAL: Init GAL engine failure: fbcon. GDI: Can not initialize graphics engine!

次のコマンドを実行して、再びデモを実行します。

mknod /dev/tty0 c 4 0

4. MiniGUI O IAL(Input Abstract Layer)

MiniGUI は異なる入力デバイス、例えばマウス、キーボード、ボタンなどに対応するため、 IAL(Input Abstract Layer)があります。IAL はこれら入力信号を内部統一のメッセージに 変換しますので、MiniGUI のアプリケーションはプラットフォームを関連しません。異な るハードウェアでも、異なる OS でも動けます。

自分の IAL プログラムを作る手順:

仮定新 IAL プルグラムの名前は_NAME_IAL です。IAL のフォルダは libminigui/src/ial です。

(1) ial.c ファイルに次の内容を添加します。

//input の配列の中に

#ifdef _NAME _IAL

{"NAME ", InitNAMEInput, TermNAMEInput},

#endif

(2) 新 IAL プログラムを Makefile.am に入れます。

(3) MiniGUI をコンフィグする時、使用したい IAL を指定します。例:

\$./configure --host=arm-linux --enable-jpgsupport=yes --enable-pngsupport=yes --enable-gifsupport=yes --disable-lite --prefix=/usr/local/arm/2.95.3/arm-linux --enable-_NAME_IAL=yes

(4) IAL プログラムの例

smdk2410.tarはmini2440用IALです。3.5インチタッチパネルと6個ボタンのIALです。 参考してください。詳しい作り方はMiniGUIのドキュメントを参照してください。


付録 7 ADS1.2 と H-JTAG でプログラムを開発

ADS1.2 ARM Developer Suite)は ARM 社によって開発された A 開発ツールです。ADS1.2 で開発されたプログラムは直接に ARM9 にロードできるし、実行できる。ADS1.2 で OS なし環境でプログラムを開発するのは便利です。

ADS1.2 でプログラムを開発することを紹介いたします。

1. 新プロジェクトを作る

 $\operatorname{ADS} \mathfrak{OSID} \twoheadrightarrow \operatorname{File} \mathfrak{I} \mathfrak{S}$

New	
Project File Object	Project name: myled Location: C:¥work¥myled Set. Add to Project: Project:
	OK キャンセル

"Project name"と"Location"を入力してください。入力完了すれば、"Ok"ボタンを押して、 次のウィンドが出てきます。



li m	yled.mcp					X
	DebugRel	- 🔝 😣	/ 🧭 💺	• [
Files	Link Order Tar	rgets				
					1000000000000	
*	File		Code	Data	10.	à
*	File	A	dd Files	Data	10. 1	

マウスの右ボタンをクリックして、"Add Files"を選択してください。

ソースフォルダ myled/lib 中のファイルを選択してください。

Select files to a	add	? 🛛
ファイルの場所① 2440addr.inc 2440init.s 2440lib.c 2440lib.c 2440slib.s Memofg.inc Option.inc	ib	
オブジェクト名(N): オブジェクトの種類	<pre>"2440slib.s" "2440init.s" "2440lib.c" All Files (*.*)</pre>	開((<u>(</u>)) キャンセル

"開く"ボタンを押すと、次のウィンドが出てきます。"Ok"ボタンを押してください。

Add Files	
Add files to targets:	
Targets	
🔽 DebugRel	-
🔽 Release	_
🔽 Debug	
0.22	
	Ψ.
OK Can	cel 1



ソースフォルダ myled 中の main.c も選択してください。

Select files to a	add	? 🛛
ファイルの場所型:	myled	
inc ib myled_Data inain.c myled.mcp		
オブジェクト名(<u>N</u>): オブジェクトの種類	Main.c	■ 開(@) ▼ キャンセル

プロジェクトで全てファイルを添加した様子:

I• •	nyled.mcp					×
-0	DebugRel 🗾 🔢 💐	¥ 🚳 🕵 I				
File	es Link Order Targets					
*	File	Code	Data	0	ŧ	4
4	2440slib.s	0	0	•	• •	1
4	1 2440init.s	0	0		• 2	1
4	1 2440lib.c	0	0	•	• 1	1
*	main.c	0	0	٠	• 2	١
						-
	4 files	0	0			1



2. コンパイルの環境の設定

コンパイルの環境の設定:メニューEdit → DebugRel Setting...を選択してください。左側の「Target Settings」を選択して、右側の「Post linker」で「ARM fromELF」を選択します。

DebugRel Settings	
Target Settings Panels □ Target Settings □ Access Paths □ Build Extras □ Runtime Settings □ File Mappings □ Source Trees □ ARM Target □ Language Settings □ ARM C Compiler □ Thumb C Compiler □ Thumb C++ Compiler □ Thumb C++ Compiler □ ARM Linker □ ARM fromELF □ Editor ▼	Target Settings Target Name: DebugRel Linker: ARM Linker Pre-linker: None Post-linker: ARM fromELF Output Directory: Browse {Project} Clear Save project entries using relative paths
	Factory Settings Revert Import Panel Export Panel
	OK Cancel Apply



左側の「ARM Assembler」を選択して、アセンブラのコンパイラーを ARM920T に設定してください。

DebugRel Settings	? 🔀
Narget Settings Panels → Target → Target Settings → Access Paths → Build Extras → Runtime Settings → File Mappings → Source Trees → ARM Target ⊖ Language Settings → ARM C Compiler → ARM C Compiler → Thumb C Compiler → Thumb C++ Compil → Linker	ARM Assembler Target ATPCS Options Predefines Listing Control Extras Architecture or Processor Eloating Point ARM920T Imitial State No floating point Byte Order Initial State ARM Big Endian Equivalent Command Line -keep -g -cpu ARM920T -fpu None
ARM fromELF	Factory Settings Revert Import Panel Export Panel OK Cancel Apply



左側の「ARM C Compiler」を選択して、C コンパイラーを ARM920T に設定してください。

DebugRel Settings				? 🛛
Target Settings Panels □ Target Settings □ Target Settings □ Access Paths □ Build Extras □ Runtime Settings □ Source Trees □ ARM Target □ ARM Assembler □ ARM Compiler □ ARM Compiler □ Thumb C Compiler □ Thumb C++ Compiler □ Thumb C++ Compiler □ ARM Linker □ ARM Linker □ ARM fromELF	ARM C Compiler Target and Source A Architecture or Proc ARM920T Byte Order © Little Endian © Big Endian Equivalent Comman -01 -g+ -cpu ARM	TPCS Warnings Errors cessor Source Langua ANSI/ISO S d Line 920T	Debug/Opt F	reprocessor Cod t n softfp
Editor 🗸	<			>
	Factory Settings	Revert	mport Panel	Export Panel
		ОК	Cancel	Apply

左側の「ARM Linker」を選択して、プログラムの実行アドレスを設定します。ARM9 ボードの場合は 0x30000000 です。

💼 DebugRel Settings				? 🛛
Target Settings Panels □ Target Settings □ Target Settings □ Access Paths □ Build Extras □ Runtime Settings □ File Mappings □ Source Trees □ ARM Target □ Language Settings □ ARM C Compiler □ ARM C Compiler □ Thumb C Compiler □ Thumb C++ Compil. □ Linker □ ARM Linker □ ARM Linker	ARM Linker Output Options Layout Linktype C Partial C Scattered Scatter description file Symbol definitions file Symbol gditing file Equivalent Command Lin -info totals -ro-base 0x	Listings Extras	R <u>W</u> Base	☐ Bopi ☐ Re ☐ Rwp <u>i</u> ☐ Split I <u>m</u> age ☐ Cl ☐ Cl
Editor	<			>
	Factory Settings	vert	Import Panel	Export Panel
		ОК	Cancel	Apply



左側の「ARM fromELF」を選択して、出力ファイルの名前を入力してください。

💼 DebugRel Settings		? 🛛
Target Settings Panels	ARM fromELF	
 ☐- Target △ Target Settings △ Access Paths → Build Extras → Runtime Settings 	Options Include debug sections in output	Text format flags └└ Verbose └ Disassemble code └ Print contents of data
File Mappings Source Trees ARM Target Language Settings	Output format Plain binary	Print debug tables Print gelocation inform Print symbol table
ARM Assembler ARM C Compiler ARM C++ Compiler	Output file name myled.bin Choose	Print string table
Thumb C Compiler Thumb C++ Compil. Linker ARM Linker	-c -output myled.bin -bin	
Editor	Factory Settings Revert Impor	rt Panel Evoort Panel
	OK	Cancel Apply

ADS メニュー: Project → make を選択して、実行できる bin ファイルを生成します。





エラーがなければ、myled/myled_Data/DebugRel で myled.bin を生成します。

3. デバッグ環境の設定

ADS1.2のAXD Debuggerを実行します。

🐼 AXD		
<u>F</u> ile <u>S</u> earch <u>P</u> rocessor Views System Views E <u>x</u>	ecute Op <u>t</u> ions <u>W</u> indow <u>H</u>	elp
r R 🕑 🖌 I I 🖉 📓 🦓	r 🖪 V 🕵 🔳 🖻 Q	
Target Image Fi		
System Output Monitor		
RDI Log Debug Log		
Log file:		
IntCtrl, Tracer, RDI Codesequences ABM RDI 1.5.1 -> ASYNC RDI Protocol Converter ADS v1	2 [Build number 805]. Convrinht	(c) ABM
For Help, press F1	(No Pos) ARMUL ARM7TDMI	(No Imag

AXD Debugger $\mathcal{O} \times \exists \exists - : Options \rightarrow Configuer Target...$

C	hoose Targe	t i		? 🔀
Ť	Target Environ	iments		
	Target	RDI File	Version	Add
	ARMUL	1.5.1 C¥PROGRA 1¥ARM¥ADSVI_2¥Bin¥ARMulate.dll	1.2.0.805	Remove
				Rename
				Save As
				Configure
	Please the list be use	select a target environment from the above list or add a target . Note that a target environment has to be configured at least o d.	: environment to once before it can	
		ОК	Cancel	Help



"Add"ボタンを押して、H-JTAG をインストールしたフォルダで H-JTAG.dll ファイルを添加します。

ファイルを開く		? 🛛
ファイルの場所型:	🔁 H-JTAG	
FConfig FDevice HConfig Target Temp		
ファイル名(<u>N</u>): ファイルの種類(T):	H-JTAG.dll	

H-JTAG.dll を添加した様子。

Carget	BDI	File	Version	Add
ADP	1.5.1	C:#PROGRA*1#ARM#ADSv1_2#Bin#Remote_A.dll	1.2.0.805	
ARMUL	1.5.1	C:¥PROGRA [~] 1¥ARM¥ADSv1_2¥Bin¥ARMulate.dll	1.2.0.805	Remove
H-JTAG	1.5.1	C:¥Program.Files¥H-JTAG¥H-JTAG.dll	V0.4.4	
				Rename
				-
				Save As
				Configure
JTAG Debu	ig Interface	e for ARM In-Circuit Emulation.		



4. H-JTAG でプログラムをデバッグします

ADS1.2 の開発環境で File → Open...

2440test フォルダの 2440test.mcp を選択します。

Leap Code 1956 436 4668	Image: Second	•			
Code 1956 436 4668	Data 1 0 0				
Code 1956 436	Data 1	• •			
Code 1956 436 4668	Data 4 0 0	¢	► 100		
Code 1956 436 4668	Data 4 0 0	¢ •	• [j]		
Code 1956 436 4668	Data 4 0 0	¢ • 2	10		
Code 1956 436 4668	Data a	¢ • z			
Code 1956 436 4668	Data 4 0 0	نې د د	-10		
1956 436 4668	0				
436	Ő	12			
4668	č –				
	8				
2284	404	17			
672		17			
1652	32	17			
492	8	1.5			
812	õ	1.5			
364	ō	1.5	i		
5756	875	. 7			
2828	68	ъ 🗑			
860	12	. 5			
4984	186	4 😨	- I		
1816	237K	. 7	1		
7660	52				
3100	560	. 3	1		
0	150K	• 1	-		
44K	1.70M		7		
	2284 672 1652 492 364 5756 2828 860 4984 1816 7660 3100 0 44K	2284 404 672 0 1652 32 492 8 812 0 364 0 5756 875 2828 68 860 12 4984 186 1816 237K 7660 52 3100 560 0 150K 44K 1.70M	2284 404 • • • • • • • • • • • • • • • • •	2284 404 • x 672 0 • x 1652 32 • x 492 8 • x 812 0 • x 364 0 • x 5756 875 • x 2828 68 • x 860 12 • x 4984 186 • x 1816 237K • x 7660 52 • x 3100 560 • x 0 150K • x ▼	2284 404 • ▼ 672 0 • ▼ 1652 32 • ▼ 492 8 • ▼ 812 0 • ▼ 364 0 • ▼ 5756 875 • ▼ 2828 68 • ▼ 860 12 • ▼ 4984 186 • ▼ 1816 237K • ▼ 7660 52 • ▼ 3100 560 • ▼ 0 150K • ▼ ▼ 44K 1.70M

ADS1.2 𝔅为===: Project → Debug

ADS1.2 は 2440test プロジェクットをコンパイルして、自動的に AXD Debugger を実行し て、生成された 2440test.axf ファイルを H-JTAG で ARM9 ボードのメモリでダウンロード して、デバッグします。



🛞 AXD		
Eile Search Processor Views System	Views Execute Options Mindow	/ Help
Target Image Fi	ading Image	
System Output Monitor RDI Log Debug Log Log file: HJTAG VO.4.4 (BUILD 20070401) Copyright	C) Twentvone 2004-2007, All Rights Res	verved.
) 	
For Help, press F1	Loading Image	No Pos> H-JTAG ARM920T (No Image Name)

2440test.axf が大きいから、ダウンロードの時間がかかります。

🙆 AXD		🔳 🗖 📃
File Search Processor Views	System Views E <u>x</u>	ecute Options Window Help F 🔍 🎔 呢 🕒 🔂 📗 🖭 🖛 🖂 📰 🔛 🖿 💷
Target Image Fi ◀ ► ■# ARM920T	(A 1171)	
	108 109 110 111 → 112 113 114 115 115	<pre>- D: Vort 22200 est \sic 22400 init.s [ENTRY_BUS_WIDTH=8 streq r0,[r0,-r10,ror #1] ;DCD 0x070000ea] b ResetHandler] b HandlerUndef ;handler for Undefined mode b HandlerSWI ;handler for SWI interrupt b HendlerBebert ;handler for SWI interrupt </pre>
System Output Monitor RDI Log Debug Log Log file: HJTAG VO.4.4 (BUILD 20070401) Cop	oyright (C) Twentyon	e 2004-2007. All Rights Reserved.
For Help, press F1		Line 112, Col O H-JTAG ARM920T 2440test.axf

ダウンロード完了の様子。



ADS1.2 のメニュー: Execute → Go プログラムは 2440test の Main 関数に入ります。

🙆 AXD		
<u>F</u> ile <u>S</u> earch <u>P</u> rocessor Views Sy	zstem Views Execute Op <u>t</u> ions <u>W</u> indow <u>H</u> elp	
m ni 🕑 🚘 主 💵 💵		
Target Image Fi 4 >		
ARM920T		
	ARE9201 - D:\work\2440test\src\2440init.s	
	🐼 ARN920T - D:\work\2440test\src\Nain.c	
	162 };	_
	163	
	165 void Main(void)	
	• 166	
	168 int i;	
	169 U8 key;	
	120 H22 mpl1 wel = 0 ·)
Senter Outent Beritan		-19-
RDI Log Debug Log		
Log file:		1
H-JTAG V0.4.4 (BUILD 20070401) Copy	right (C) Twentyone 2004-2007. All Rights Reserved.	<u></u>
<		
For Help, press F1	Line 166, Col O H-JTAG ARM920T	2440test.axf 🏑

デバッグが始まります。

付録 8 U-boot

Das U-Boot は、複数のアーキテクチャに対応したブートローダである。サポートするアー キテクチャは PPC、ARM、AVR32、MIPS、x86、68k、Nios embedded processor、MicroBlaze である。GPL のライセンスの元でリリースされている。GNU toolchain(例えば crosstool、 the Embedded Linux Development Kit (ELDK)、OSELAS.Toolchain など)を使うことに より、x86 アーキテクチャのいかなるコンピュータ上でもビルドすることが出来る。 http://www.denx.de/wiki/U-Boot/WebHome

u-boot-1.1.6-FA24x0.tar.gz は S3C2440 用 U-boot です。以下の機能をサポートしています。

- 1. xmodem プロトコル
- 2. USB ダウンロード、パソコン側の DNW と一緒に動けます。
- 3. Ethernet CS8900/DM9000
- 4. NAND Flash の読み書き
- 5. Nor 又は NAND からブートできます



6. yaffs ファイルシステムの書き込み

U-boot のインストール手順:

1. U-boot を解凍 #tar xvzf u-boot-1.1.6-FA24x0.tar.gz

2. CS8900/DM9000 の選択

include/configs/100ask24x0.hを直す #if 0 #define CONFIG_DRIVER_CS8900 1 /* we have a CS8900 on-board */ #define CS8900_BASE 0x19000300 #define CS8900_BUS16 1 /* the Linux driver does accesses as shorts */ #endif #if !defined(CONFIG_DRIVER_CS8900) #define CONFIG_DRIVER_DM9000 1 #define CONFIG_DRIVER_DM9000 1 #define CONFIG_DM9000_USE_16BIT 1 #define CONFIG_DM9000_BASE 0x20000000 #define DM9000_IO 0x20000000 #define DM9000_DATA 0x2000004 #endif

3. コンフィグとコンパイル

#make open24x0_config #make 成功したら、u-boot.bin を生成します。

4. U-boot の書き込み

ARM9を NOR Flash から起動させます。



🍓 ttyS0 - 超级终端	×
文件(E) 编辑(E) 查看(V) 呼叫(C) 传送(C) 帮助(E)	
D 😅 📾 🌋 🗈 🎦 🗳	
	~
USB host is connected. Waiting a download.	-
##### ErjendlueRH BIOS for 2008 #####	
[x] bon part 0 320k 2368k	
[v] Download vivi	
[k] Download linux kernel	
[y] Download root_yaffs image	
[c] Download root_cramfs image	
[a] HOSOIUCE USER HPPIICACION	
[e] Download Hoot	
[1] Download WinCE NK.nbØ	
[v] Download WinGE NK.bin	
[d] Download & Run	
[f] Format the nand flash	
[p] Partition for Linux	
[b] Boot the system	
[5] Set the boot parameters	
[u] Backun NAND Flash to HOST through USB(unload)	
[r] Restore NAND Flash from HOST through USB	
[q] Goto shell of vivi	
Enter your selection: _	B
	*

メニューの中で、機能号[a]を選択して、DNWを実行します。DNWのメニュー「USBPORT」

→ 「Transmit/Restore」で生成された u-boot.bin を選択して、書き込みます。