



不可能への挑戦

株式会社日昇テクノロジー

低価格、高品質が不可能？

日昇テクノロジーなら可能にする

Multi-Media ARM9

Linux-2.6.29 版

Eclipse + GCC + open-JTAG

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

2010/3/17



USB で ARM7 と接続

[copyright@2010-2011](http://www.csun.co.jp)



第一章 MINI2440 ボードの概要	7
1.1 仕様	7
1.2 使えるデバイス例	10
1.3 付属アプリケーション例	12
第二章 インターフェースの説明	12
2.1 電源	12
2.2 ユーザーボタン	13
2.3 シリアルポート	13
2.4 液晶 LCD インターフェース	14
2.5 JTAG	14
2.6 GPIO	15
2.7 CMOS CAMERA	16
2.8 システムバス	17
2.9 リセット	18
2.10 AD	18
2.11 PWM ブザー	19
第三章 初体験(GUI)	20
3.1 タッチパネルの校正	20
3.2 日本語の設定	20
3.3 MP3 の再生	22
3.4 ビデオの再生	23
3.5 ピクチャのビューと編集	24
3.6 SD と USB メモリの自動認識	25
3.7 ターミナル	26
3.8 ネットワークの設定	27
3.9 ping	27
3.10 LED テスト	28
3.11 EEPROM テスト	29
3.12 PWM ブザー	30
3.13 音声のレコーダー	31
3.14 USB カメラ	32
3.15 CMOS イメージセンサー	32
3.16 AD テスト	33
3.17 ボタン	33
3.18 手書き	34



3.19 Watch dog.....	35
3.20 回転.....	36
3.21 スタートアップ.....	37
3.22 USB GPS.....	38
第四章 初体験(コンソール).....	39
4.1 パソコン側のハイパーターミナルの設定.....	39
4.2 MP3 の再生.....	41
4.3 USB メモリと外付けハードディスク.....	41
4.4 SD/MMC カード.....	42
4.5 シリアルポートでファイルを ARM9 にダウンロード.....	43
4.6 シリアルポートで ARM9 のファイルを PC に保存.....	45
4.7 LED 制御.....	46
4.8 ボタンのテスト.....	47
4.9 シリアルポートのテスト.....	48
4.10 ブザー(PWM)のテスト.....	49
4.11 LCD のバックライト.....	49
4.12 I2C-EEPROM.....	50
4.13 AD テスト.....	50
4.14 CMOS イメージセンサー.....	51
4.15 ネットワーク機能.....	51
4.15.1 ウェブサーバー.....	51
4.15.2 Telnet と Ftp 機能.....	52
4.15.3 DNS と gateway の設定.....	52
4.15.4 MAC アドレスの設定.....	52
4.15.5 ネットワーク・ファイルシステム(NFS)のマウント.....	52
4.16. RTC の設定.....	52
4.17 液晶(LCD)画面を取ります.....	53
4.18 USB 無線 LAN.....	53
第五章 Linux のクロス開発環境.....	55
5.1 クロス開発環境を構築.....	55
5.2 NFS サーバーを構築.....	56
5.3 NFS はルートファイルシステムとして起動.....	57
第六章 Linux 環境のアプリケーションを開発.....	57
6.1 Hello, World!.....	57
6.2 Hello,World をコンパイル.....	58
6.3 Hello,World を ARM9 ボードで実行.....	58



6.4 ほかのサンプル	58
6.5 Qt/Embedded GUI プログラムを作る	59
第七章 Linux カーネルを再構築	60
7.1 カーネルのソースコードを解凍	60
7.2 Linux を再コンパイル	61
7.3 ドライバの場所	64
7.4 Linux カーネルのコンフィグ	66
7.4.1 LCD 液晶とバックライト	67
7.4.2 タッチパネル	70
7.4.3 USB マウスとキーボード	71
7.4.4 USB メモリ	72
7.4.5 汎用 USB カメラ	74
7.4.6 CMOS イメージセンサー(OV9650)	76
7.4.7 イーサネット	78
7.4.8 USB 無線 LAN	81
7.4.9 オーディオ	84
7.4.10 SD/MMC メモリカード	86
7.4.11 Watchdog	87
7.4.12 LED	88
7.4.13 ボタン	89
7.4.14 PWM ブザー	89
7.4.15 AD	90
7.4.16 シリアルポート	90
7.4.17 リアルタイマーRTC	91
7.4.18 I2C - EEPROM	93
7.4.19 yaffs2 ファイルシステム	94
7.4.20 EXT2/VFAT/ NFS ファイルシステム	98
7.4.21 USB-RS232 シリアルポート	100
7.4.22 ARM7TDMI/LPC2148 との通信	102
7.5 Linux 起動ロゴを作る	105
7.6 yaffs ルートファイルシステムのイメージを生成	107
7.7 Linux ドライバの開発入門	108
7.7.1 簡単なドライバのソースコード	108
7.7.2 コンフィグファイルを編集します	108
7.7.3 Makefile を編集	110
7.7.4 ドライバをコンパイルします	111



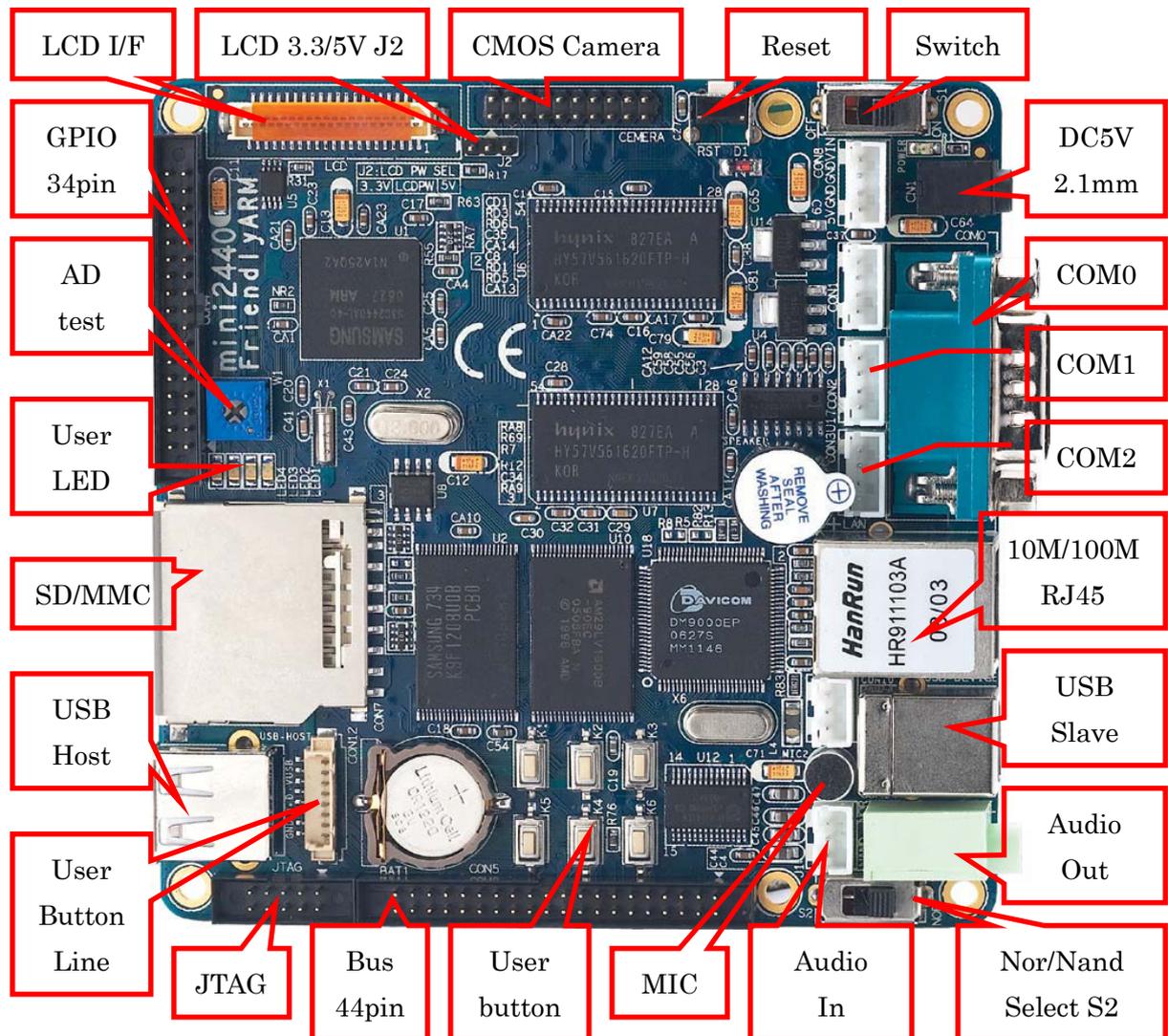
7.7.5 ARM9 ボードでドライバをインストールします	111
第八章 生成されたファイルを書き込む.....	112
8.1 NOR Flash から起動.....	112
8.2 USB ドライバのインストール	112
8.3 NAND Flash のパーティション	115
8.4 ブートロードの書き込み.....	116
8.5 Linux のカーネルの書き込み	119
8.6 ルート・ファイルシステムの書き込み	121
8.7 NAND Flash のバックアップ	122
8.8 NAND Flash のリストア	125
8.9 メモリで Linux カーネルを直接に実行	127
第九章 NOR Flash のブートロードを更新.....	130
9.1 簡易 JTAG で書き込み	130
9.1.1 H-JTAG をダウンロードとインストールします	130
9.1.2 NOR Flash を書き込む.....	135
9.2 Open-JTAG で書き込み	139
9.2.1 ドライバをインストールする	140
9.2.2 書き込み.....	143
第十章 Web カメラストリーミング配信.....	145
10.1 MJPG-streamer のダウンロードとコンパイル	145
10.2 MJPG-streamer を mini/micro2440 ボードにインストール	145
10.3 Web ブラウザで Web カメラを見ましょう	146
第十一章 Eclipse + GCC + Open-JTAG.....	147
11.1 GCC ツールチェーン	147
11.2 Integrated Development Environment(Eclipse)	149
11.3 プロジェクトを作る.....	153
11.4 Eclipse プラグイン(Zylin Embedded CDT)インストール	155
11.5 ビルドの設定	160
11.6 ビルド.....	163
11.7 GDB の設定.....	165
11.8 OpenOCD の設定.....	171
11.9 デバッグ	175
11.10 デバッグ終了	181



- ※ 使用されたソースコードは<http://www.csun.co.jp/>からダウンロードできます。
- ※ この文書の情報は、事前の通知なく変更されることがあります。
- ※ (株)日昇テクノロジーの書面による許可のない複製は、いかなる形態においても厳重に禁じられています。

第一章 MINI2440 ボードの概要

1.1 仕様



※ 液晶は 3.3V/5V 二種類があります。MINI2440 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

- 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

- 10M/100MBase-T Ethernet RJ45(DM9000) x 1
- RS232 シリアルポート x 3
- USB1.1 ホスト x 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 x 1
- 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 6
- 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ップ
- 34 ピン GPIO(2mm DIP ピッチ)
- 44 ピンのシステムバス(2mm DIP ピッチ)

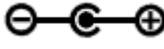
搭載した OS

- Linux2.6.29 + Qtopia2.2.0
- WindowsCE.NET 5.0
- uCOSII

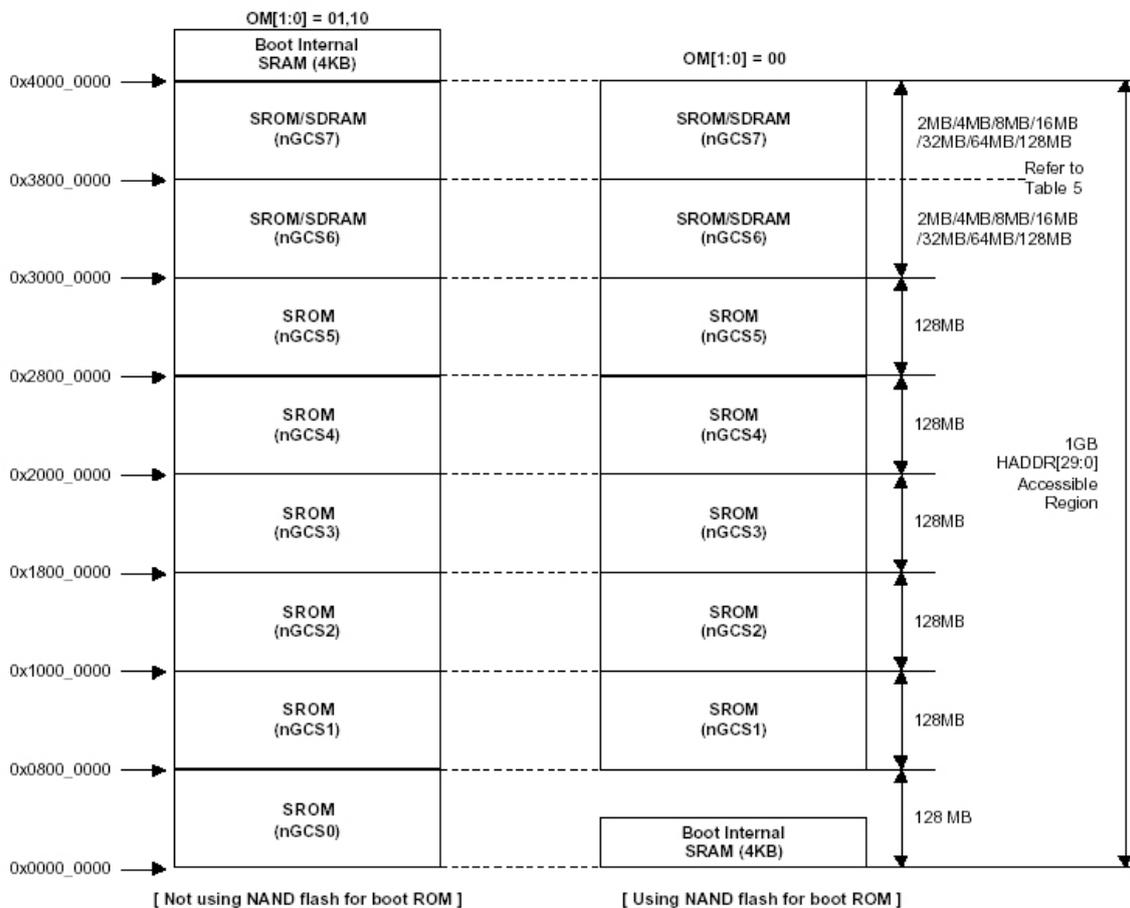
外形寸法

- 100×100(mm) 突起物は除く

供給電源

- 5VDC 電源、プラグ 1.3mmφ、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

スイッチ S2 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

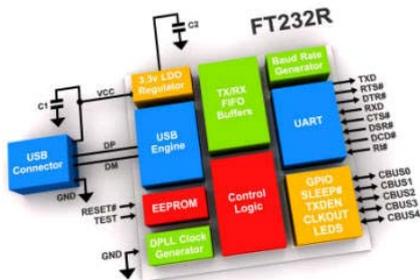


32GB までの
SD/MMC メモリ



USB メモリ

USB HUB



USB シリアルポート



PL2303



不可能への挑戦

株式会社日昇テクノロジー

低価格、高品質が不可能？
日昇テクノロジーなら可能にする



7インチ液晶



1024X768 VGA

※ 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



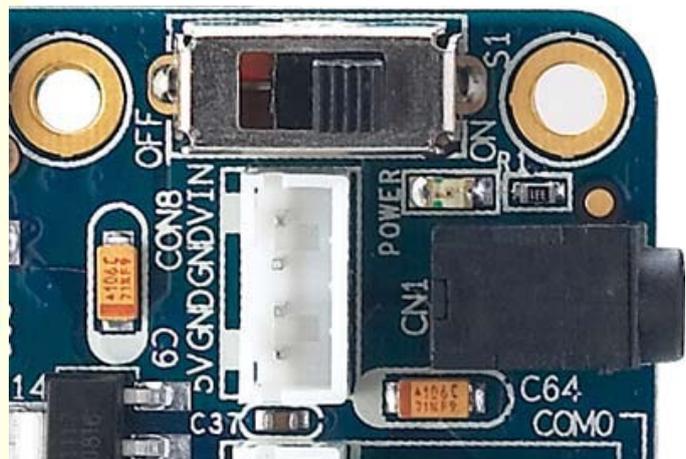
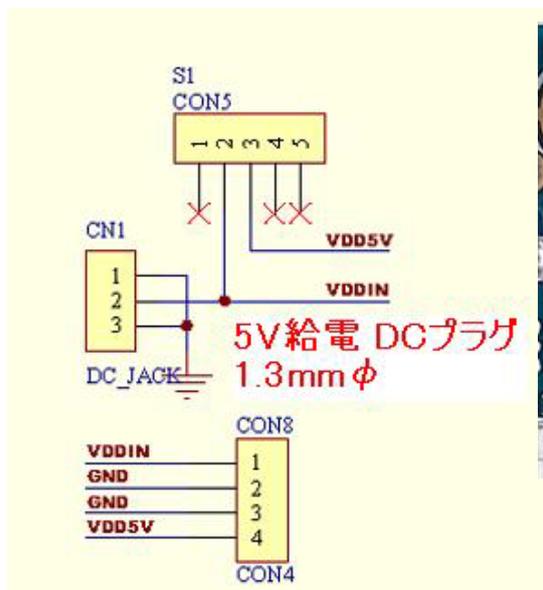
Qtopia デスクトップ

mpeg 映画

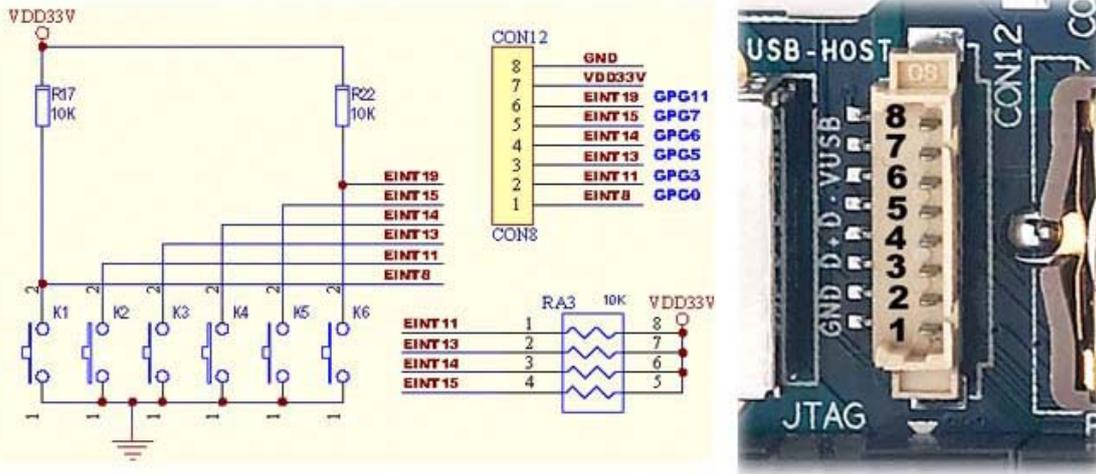
MP3
displayerWeb
ブラウザー

第二章 インターフェースの説明

2.1 電源



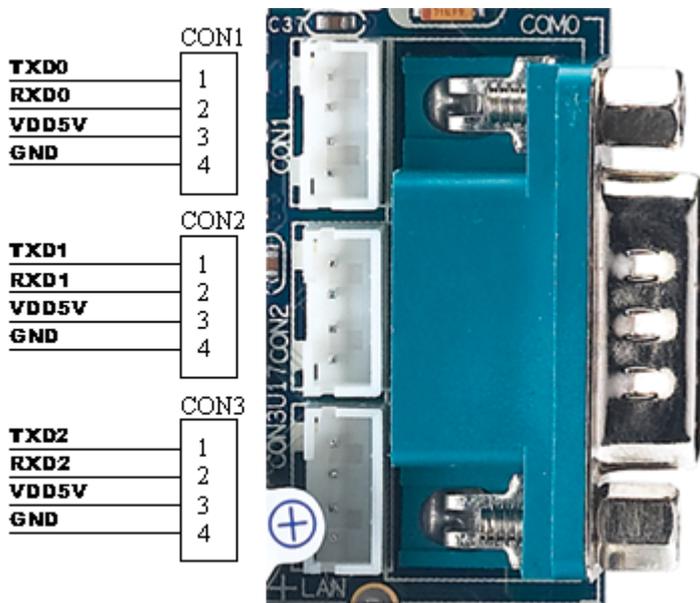
2.2 ユーザーボタン



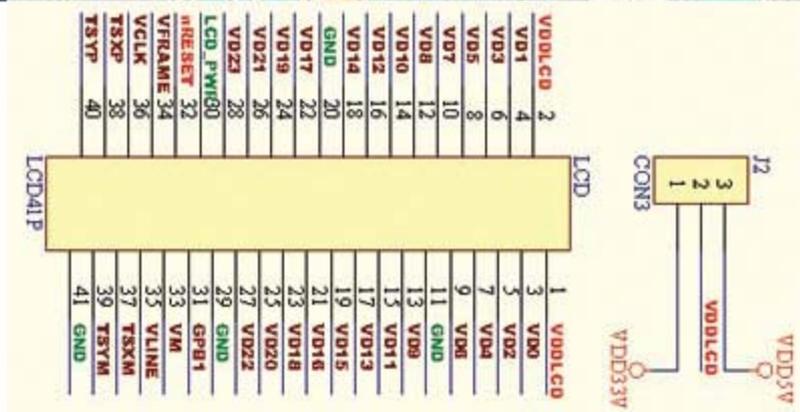
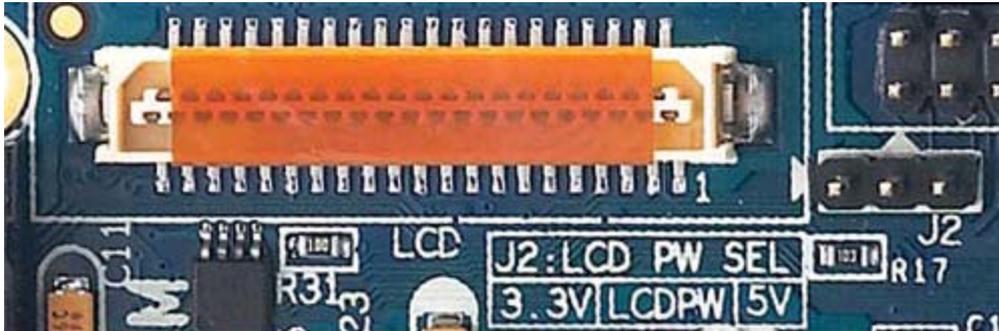
	K1	K2	K3	K3	K4	K5
割り込み	EINT8	EINT11	EINT13	EINT14	EINT15	EINT19
GPIO	GPG0	GPG3	GPG5	GPG6	GPG7	GPG11
他の機能	なし	nSS1	SPIMISO1	SPIMOSI1	SPICLK1	TCLK1
CON12	CON12.1	CON12.2	CON12.3	CON12.4	CON12.5	CON12.6

※ CON12.7 は 3.3V 電源、CON12.8 は GND です。

2.3 シリアルポート

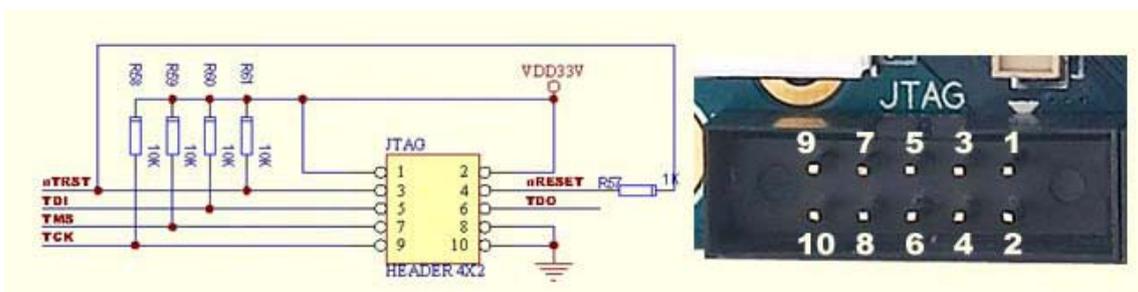


2.4 液晶 LCD インターフェース

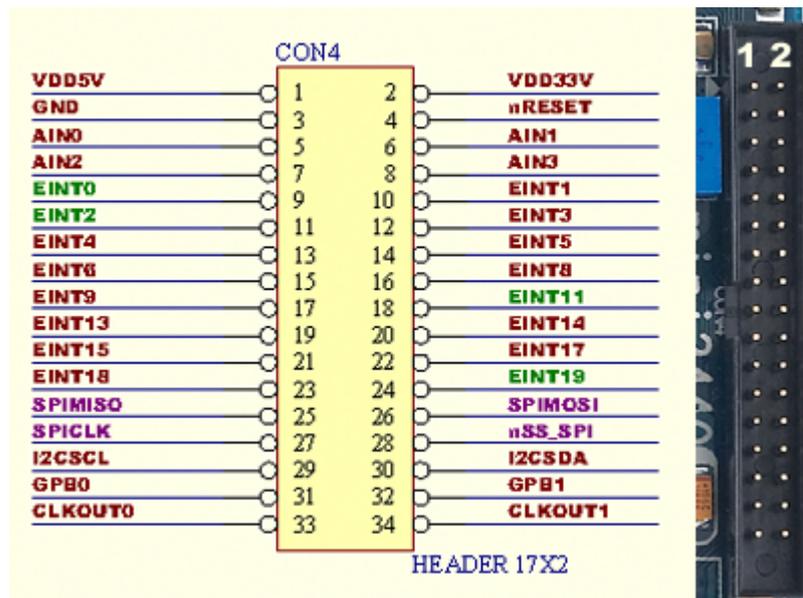


LCD インターフェースは最大 RGB(888)の液晶をサポートします。37,38,39,40 ピンは四線抵抗式のタッチパネルの入力です。J2 は液晶給電の選択、5V 又は 3.3V の液晶に対応します。

2.5 JTAG

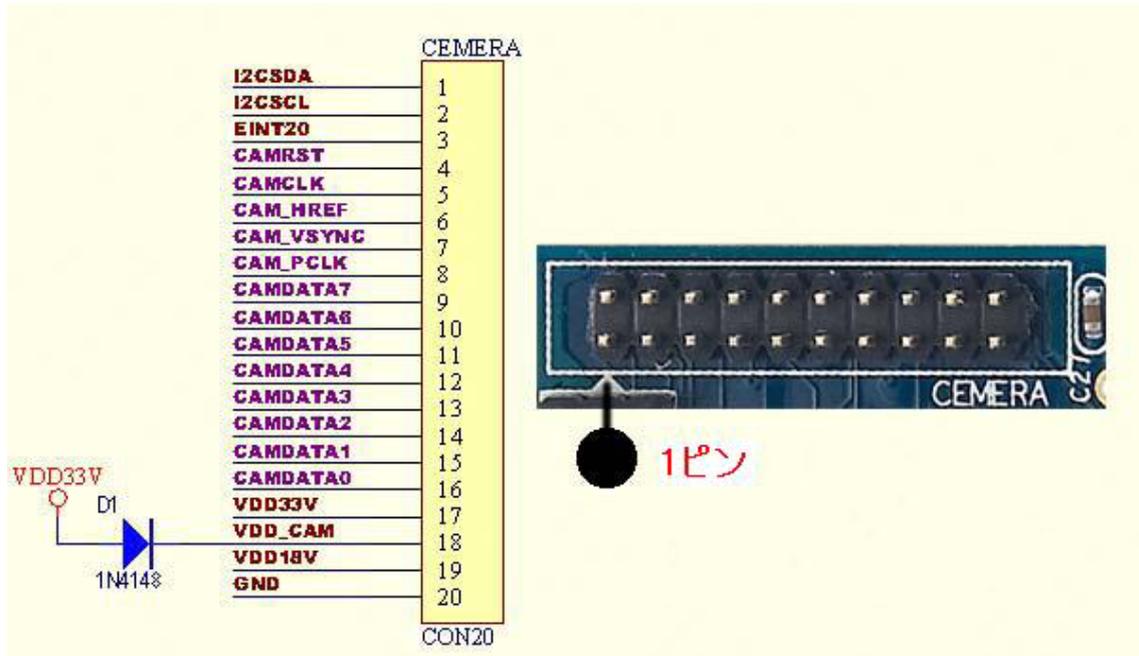


2.6 GPIO



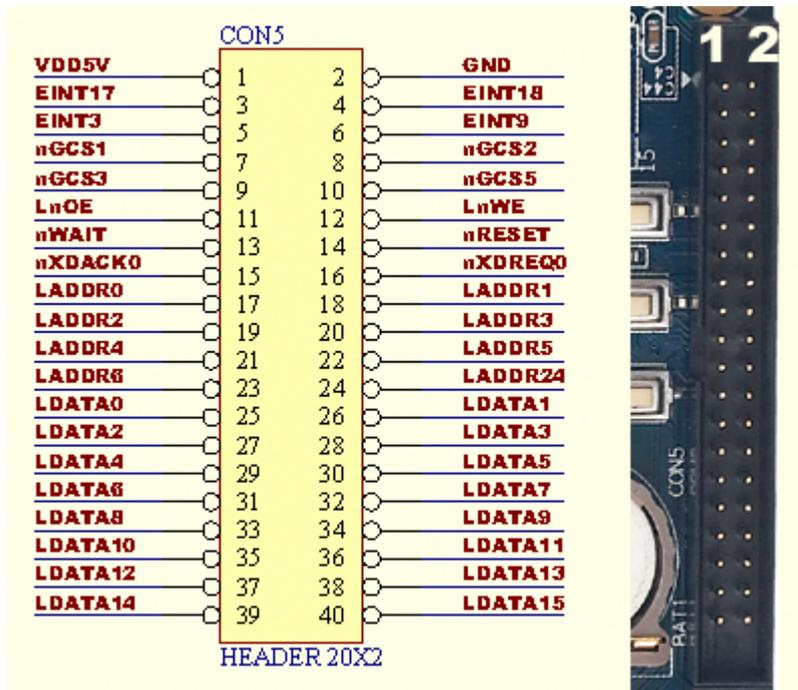
CON4	名前	説明	CON4	名前	説明
1	VDD5V	5V 電源	2	VDD33V	3.3V 出力
3	GND	GND	4	nRESET	リセット出力
5	ANI0	AD ch0	6	ANI1	AD ch1
7	ANI2	AD ch2	8	ANI3	AD ch3
9	EINT0	EINT0/GPF0	10	EINT1	EINT1/GPF1
11	EINT2	EINT2/GPF2	12	EINT3	EINT3/GPF3
13	EINT4	EINT4/GPF4	14	EINT5	EINT5/GPF5
15	EINT6	EINT6/GPF6	16	EINT8	EINT8/GPG0
17	EINT9	EINT9/GPG1	18	EINT11	EINT11/GPG3/nSS1
19	EINT13	EINT13/GPG5/SPIMISO1	20	EINT14	EINT14/GPG6/SPIMOSI1
21	EINT15	EINT15/GPG7/SPICLK1	22	EINT17	EINT17/GPG9/nRST1
23	EINT18	EINT18/GPG10/nCTS1	24	EINT19	EINT19/GPG11
25	SPIMISO	SPIMISO /GPE11	26	SPIMOSI	SPIMOSI /EINT14/GPG6
27	SPICLK	SPICLK /GPE13	28	nSS_SPI	nSS_SPI /EINT10/GPG2
29	I2CSCL	I2CSCL/GPE14	30	I2CSDA	I2CSDA/GPE15
31	GPB0	TOUT0/ GPB0	32	GPB1	TOUT1/ GPB1
33	CLKOUT0	CLKOUT0/GPH9	34	CLKOUT1	CLKOUT1/GPH10

2.7 CMOS CAMERA



CAMERA	名前	他の機能	CAMERA	名前	他の機能
1	I2CSDA	GPE15	2	I2CSCL	GPE14
3	EINT20	GPG12	4	CAMRST	GPJ12
5	CAMCLK	GPJ11	6	CAM_HREF	GPJ10
7	CAM_VSYNC	GPJ9	8	CAM_PCLK	GPJ8
9	CAMDATA7	GPJ7	10	CAMDATA6	GPJ6
11	CAMDATA5	GPJ5	12	CAMDATA4	GPJ4
13	CAMDATA3	GPJ3	14	CAMDATA2	GPJ2
15	CAMDATA1	GPJ1	16	CAMDATA0	GPJ0
17	VDD33V	3.3V 電源	18	VDD_CAM	VDD_CAM
19	VDD18V	1.8V 電源	20	GND	GND

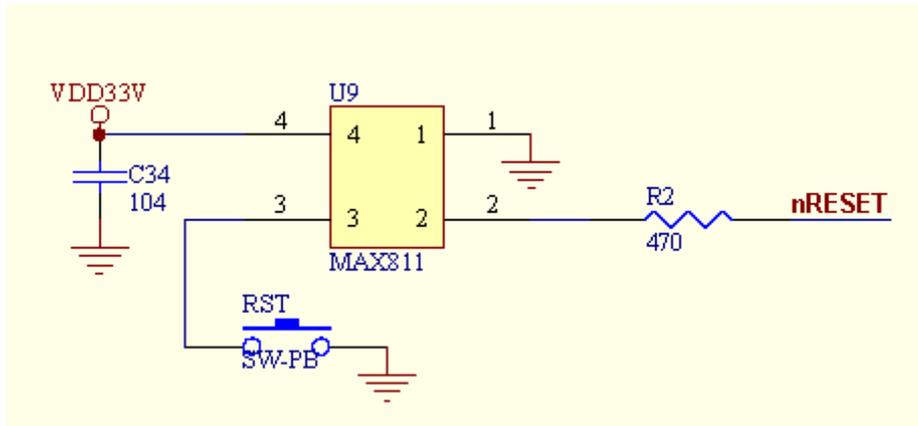
2.8 システムバス



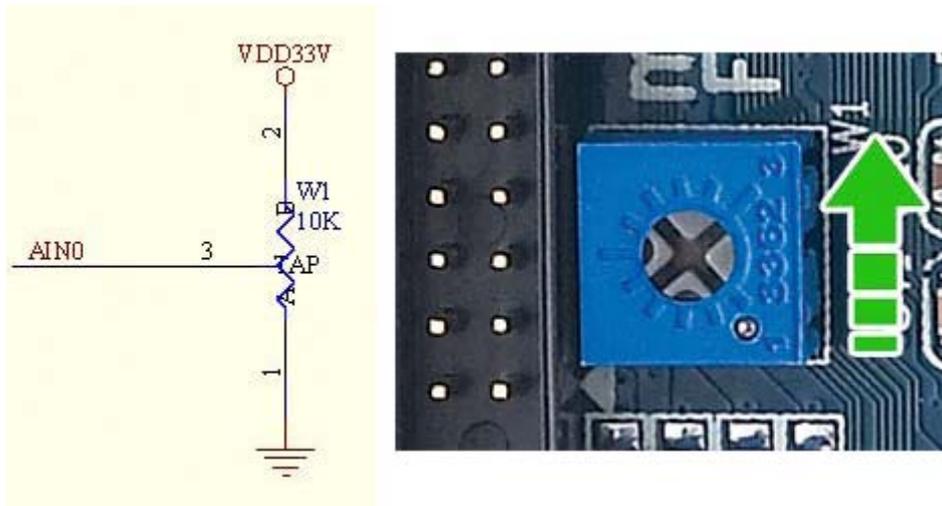
CON5	名前	説明	CON5	名前	説明
1	VDD5V	5V 電源	2	GND	GND
3	EINT17	割り込み 17	4	EINT18	割り込み 18
5	EINT3	割り込み 3	6	EINT9	割り込み 9
7	nGCS1	0x08000000	8	nGCS2	0x10000000
9	nGCS3	0x18000000	10	nGCS5	0x28000000
11	LnOE		12	LnWE	
13	nWAIT		14	nRESET	
15	nXDACK0		16	nXDREQ0	
17	LADDR0		18	LADDR1	
19	LADDR2		20	LADDR3	
21	LADDR4		22	LADDR5	
23	LADDR6		24	LADDR24	
25	LDATA0		26	DATA1	
27	LDATA2		28	DATA3	
29	LDATA4		30	DATA5	
31	LDATA6		32	DATA7	
33	LDATA8		34	DATA9	

35	LDATA10		36	DATA11	
37	LDATA12		38	DATA13	
39	LDATA14		40	DATA115	

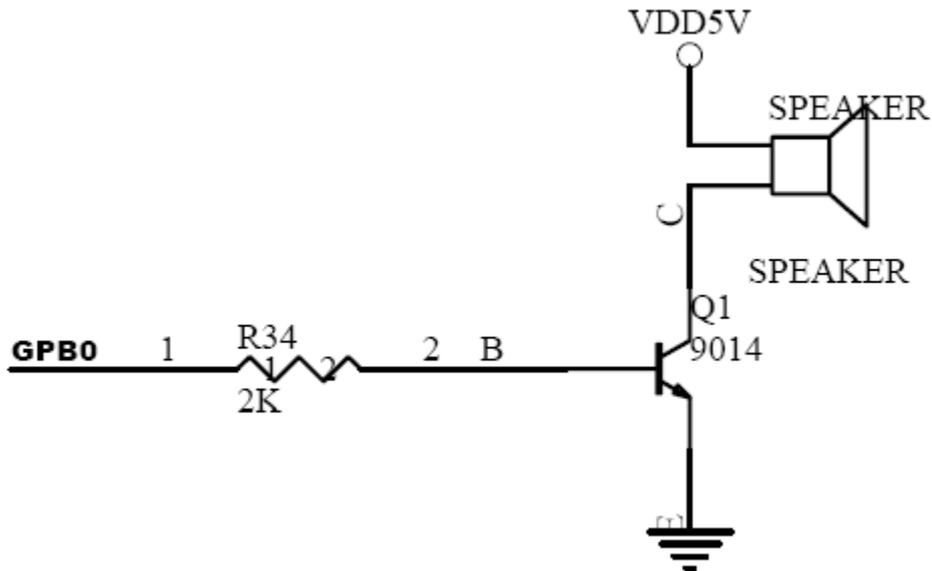
2.9 リセット



2.10 AD

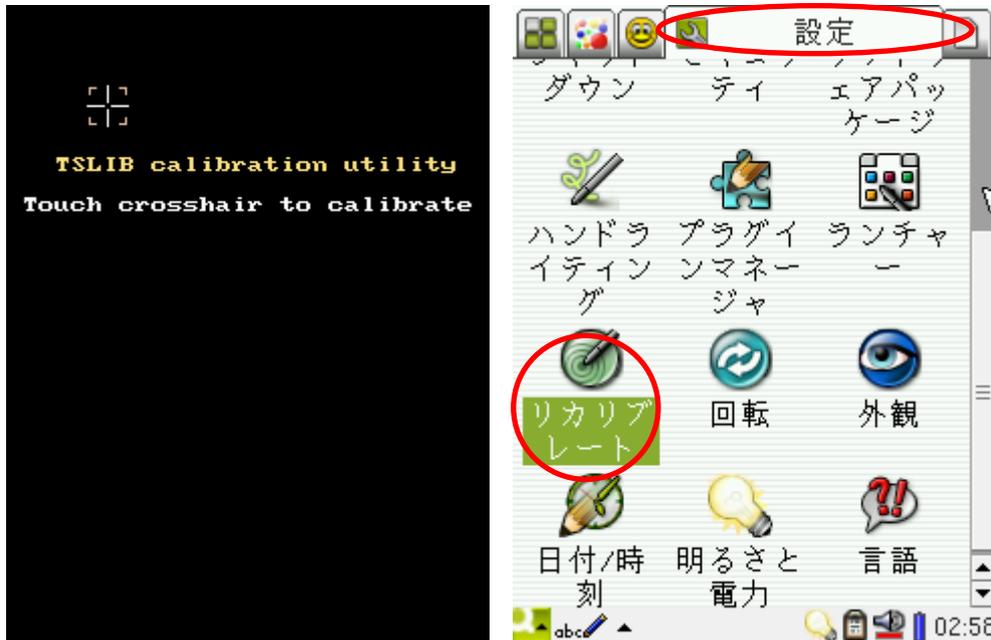


2.11 PWM ブザー



第三章 初体験(GUI)

3.1 タッチパネルの校正



GUI システムを再インストールした後、自動的にタッチパネルの校正画面が出てきます。ペンで“十”字の中心をタッチします。四角と中心、すべて五つの“十”字が順番に出てきます。タッチパネルがずれた場合は、マウスで設定タブのリカリブレートを選択して、校正画面も出てきます。

3.2 日本語の設定



“友善之臂” タブの“语言设置” をクリックして、“Japanese” を選択し、「OK」 ボタンを押します。

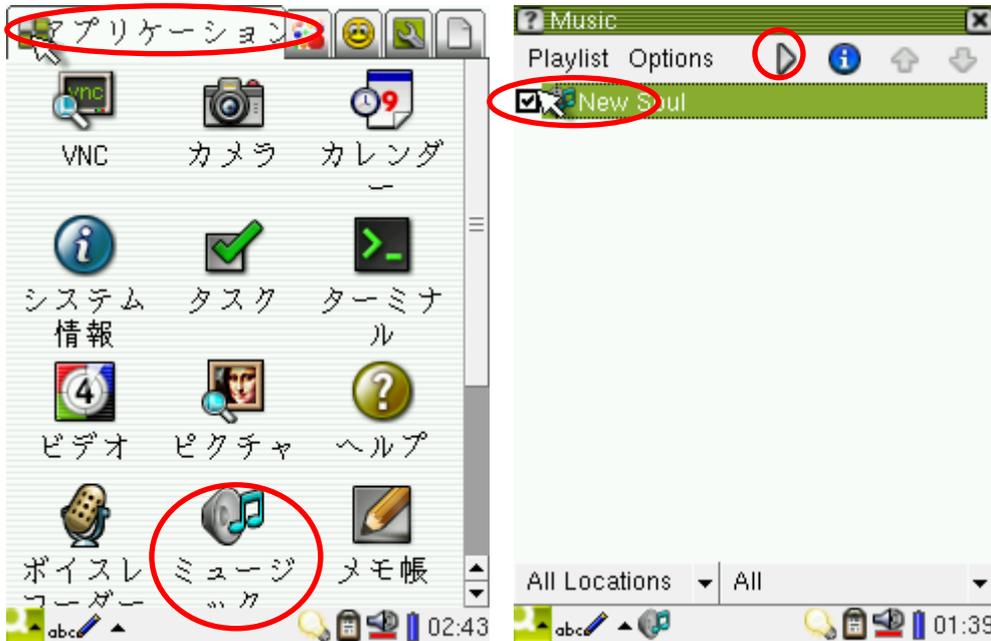


「Yes」 ボタンを押すと、日本語の画面が出てきますが、フォントの原因なので、ある文字が表示できません。“口定” タブの“外口” をクリックして、



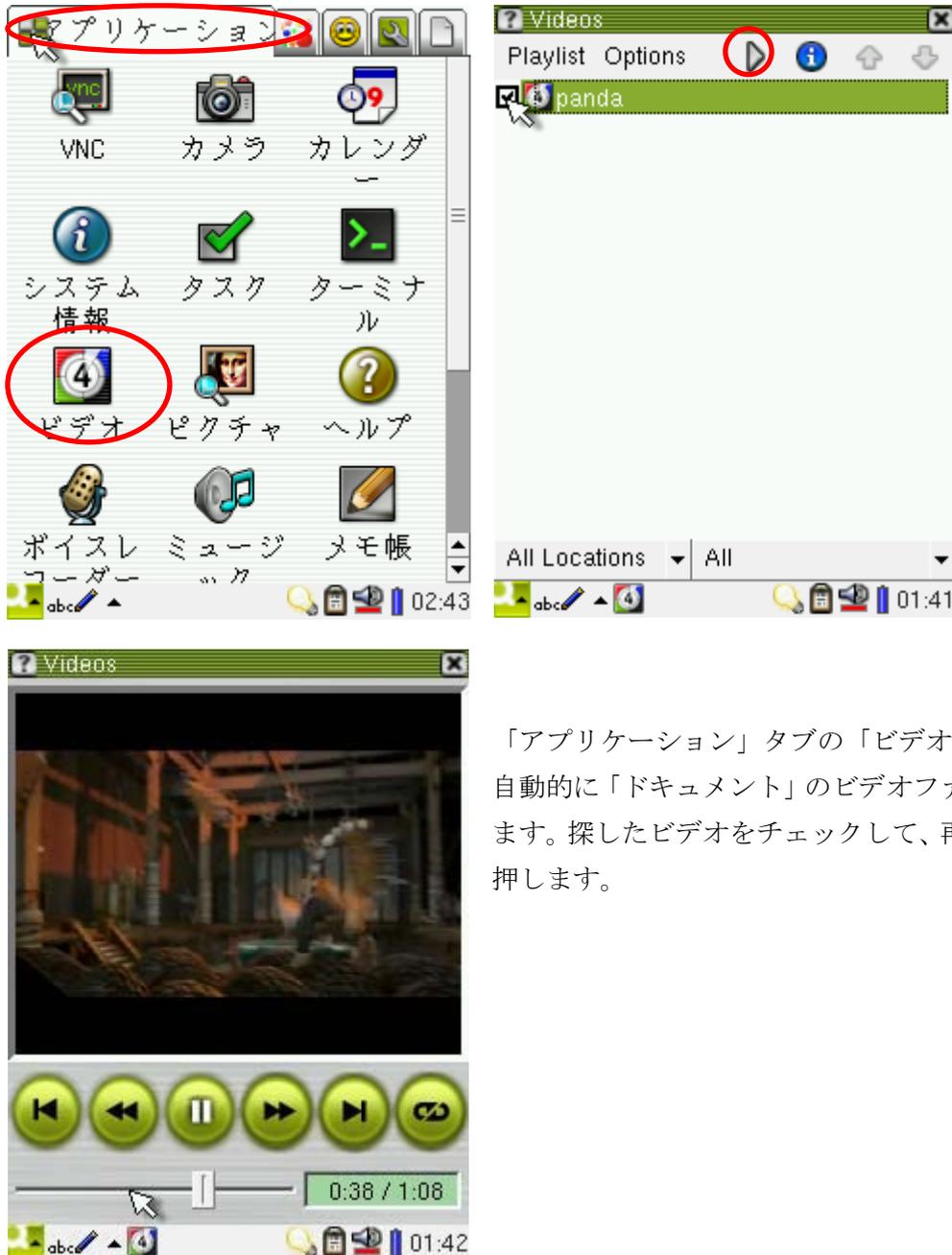
「Font」 タブで「Unifont」 を選択して、右上の「OK」 ボタンを押します。綺麗な日本語の画面が出てきます。

3.3 MP3 の再生



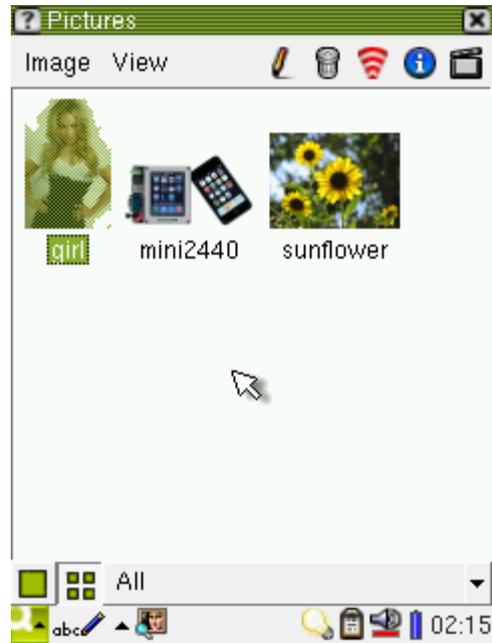
「アプリケーション」タブの「ミュージック」を選択し、自動的に「ドキュメント」のMP3ファイルをさがします。探したMP3をチェックして、再生ボタンを押します。

3.4 ビデオの再生



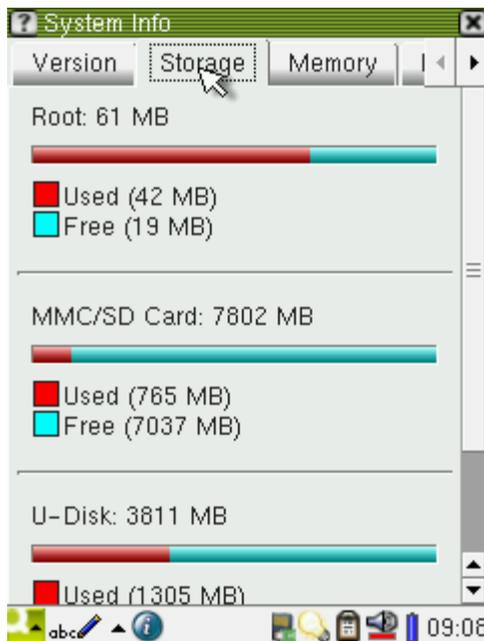
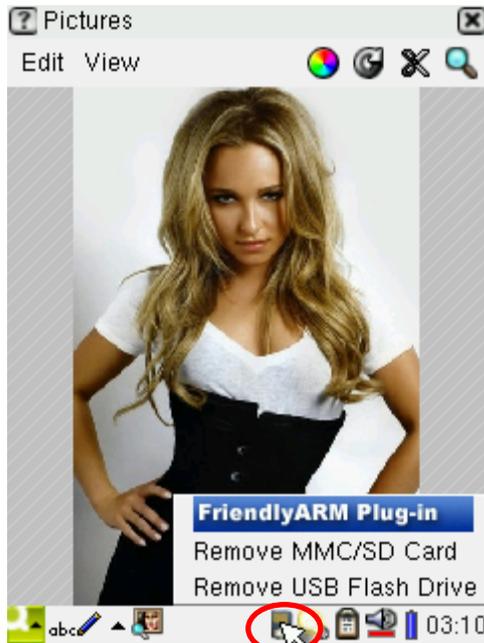
「アプリケーション」タブの「ビデオ」を選択し、自動的に「ドキュメント」のビデオファイルを探します。探したビデオをチェックして、再生ボタンを押します。

3.5 ピクチャのビューと編集



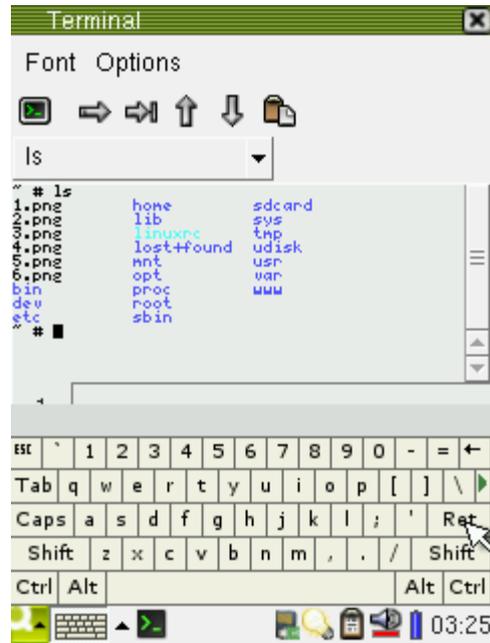
「アプリケーション」タブの「ピクチャ」を選択し、自動的に「ドキュメント」のピクチャを探します。探したビデオをクリックして、ビューと編集できます。

3.6 SD と USB メモリの自動認識



SDカードあるいはUSBメモリをARM9ボードに挿入すると、システムは自動的にSD/USBメモリを認識して、アイコンが出てきます。「アプリケーション」タブの「システム情報」をクリックして、「Storage」タブを選択すると、SD/USBメモリの情報を表示します。

3.7 ターミナル

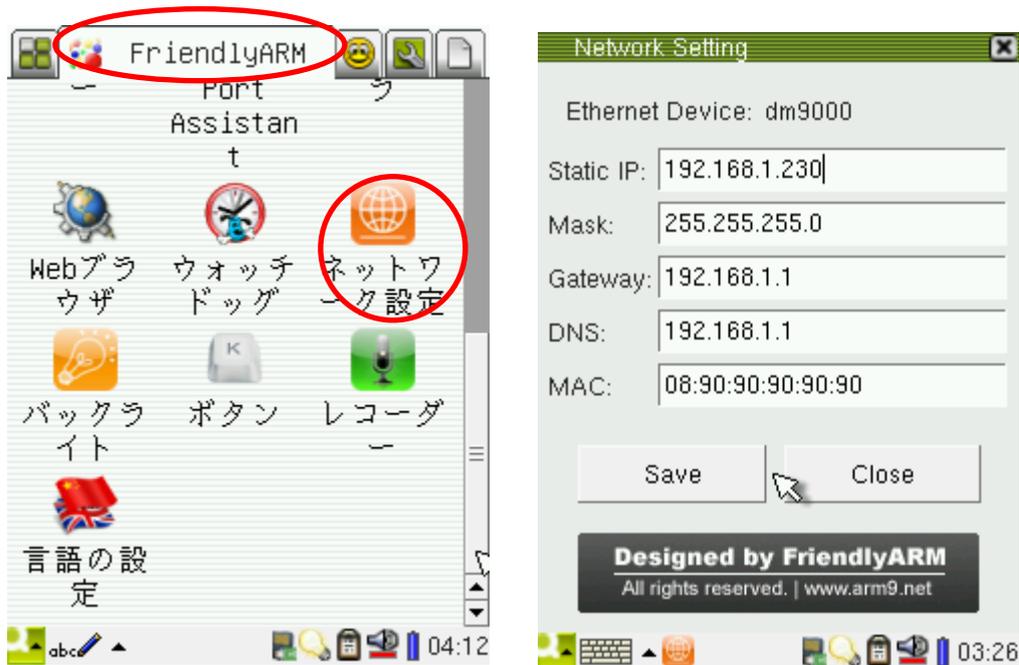


「アプリケーション」タブの「ターミナル」をクリックします。パソコンがなくても、ターミナルでコマンドを入力できます。



ターミナルでほかのシステム(ARM7TDMI/LPC2148)を通信する様子。

3.8 ネットワークの設定



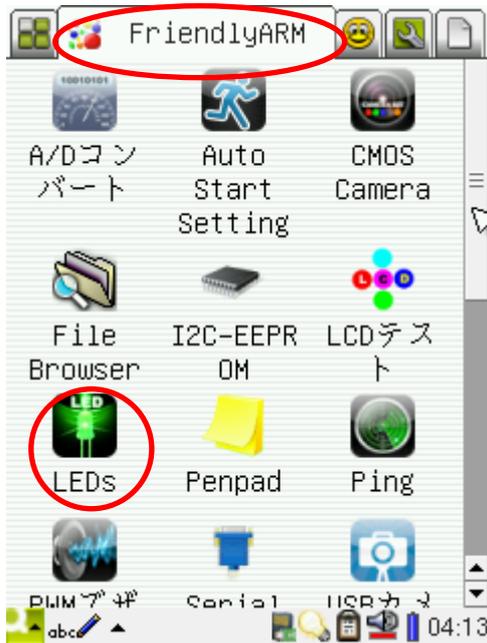
「FriendlyARM」タブの「ネットワーク設定」を選択して、ネットワークを設定します。

3.9 ping



「FriendlyARM」タブの「ネットワーク設定」を選択して、ping をします。

3.10 LED テスト



「FriendlyARM」タブの「LEDs」を選択して、LED の点灯制御ができます。起動した後、「Stop led-player」ボタンを押します。

3.11 EEPROM テスト

FriendlyARM

A/Dコンパート Auto Start Setting CMOS Camera

File Browser I2C-EEPROM LCDテスト

LEDs Penpad Ping

PLIM ア 4# Serial IICカード

04:13

このボタンを押すと、EEPROMに書き込む。

書き込みしたい内容を入力します。

IIC/EEPROM Testing

Write Data below into EEPROM via IIC

Long, long ago there lived a king. He loved horses. One day he asked an artist to draw him a beautiful horse. The artist said...

Status: Ok About... Close

03:52

IIC/EEPROM Testing

21%

書き込み進度。

Read EEPROM Data via IIC

このボタンを押すと、EEPROMからリードする。

Writing... About... Close

03:52

IIC/EEPROM Testing

Write Data below into EEPROM via IIC

Long, long ago there lived a king. He loved horses. One day he asked an artist to draw him a beautiful horse. The artist said...

読み込み進度。

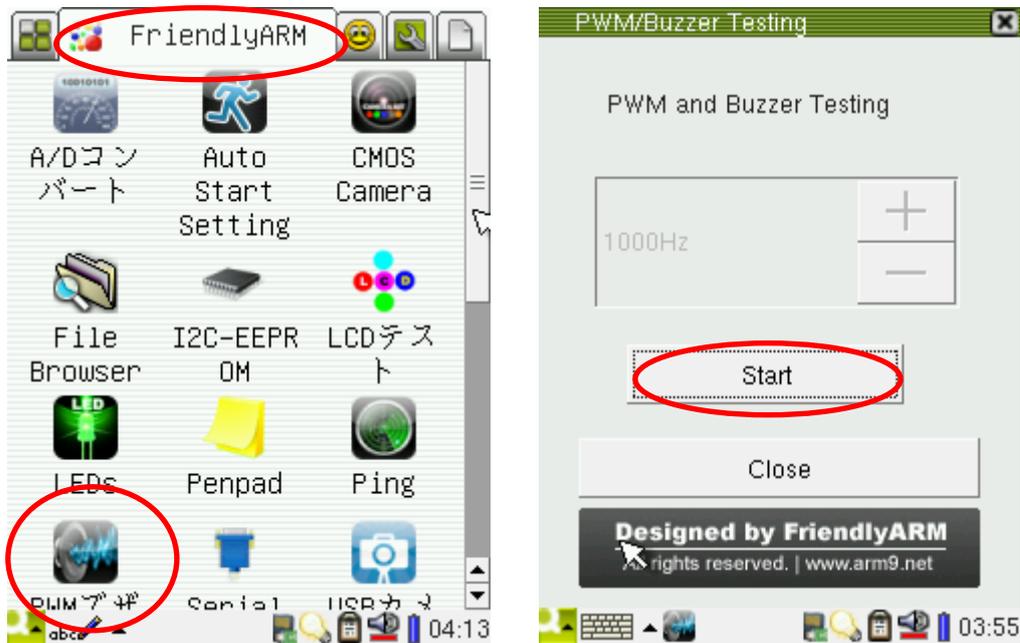
24%

読み込んだ内容

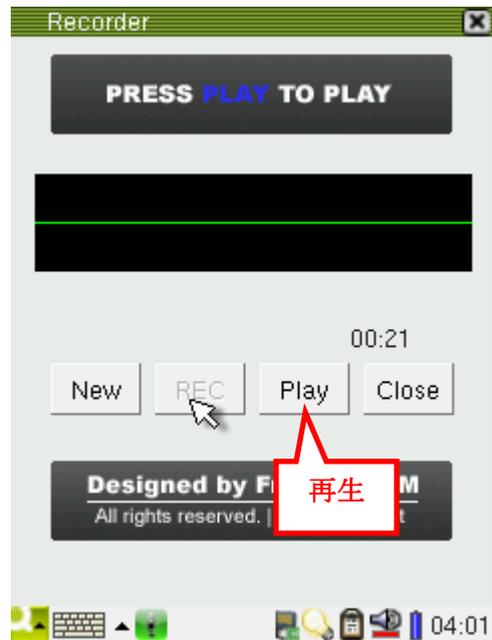
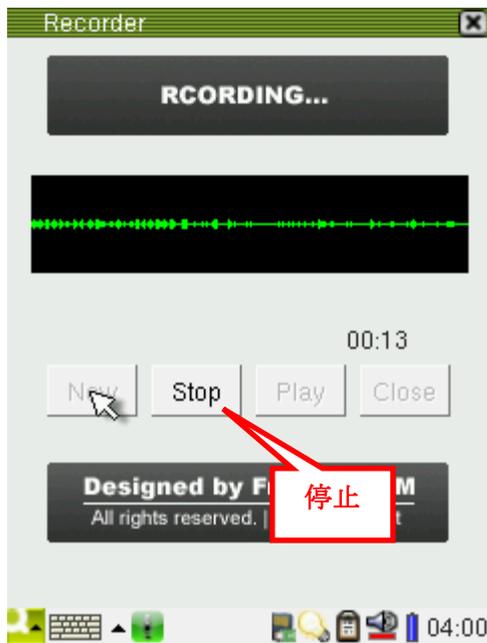
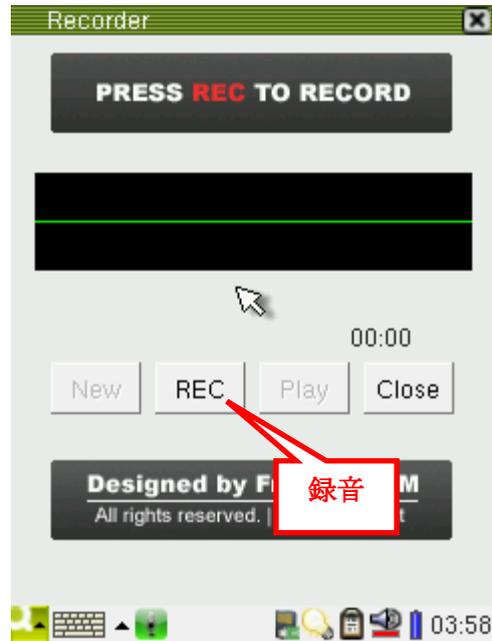
Reading... About... Close

03:52

3.12 PWM ブザー



3.13 音声のレコーダー

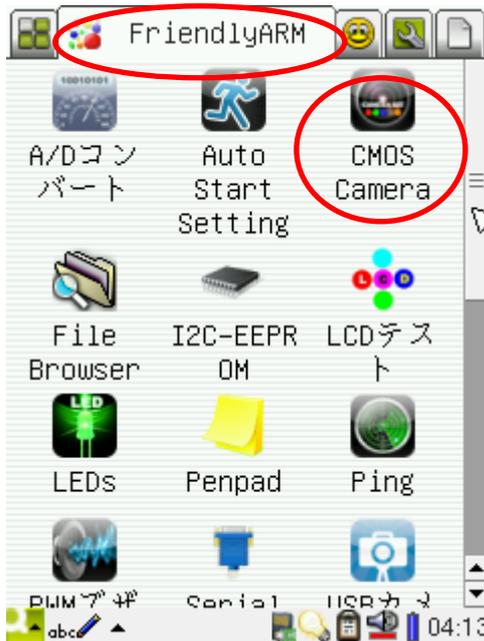


3.14 USB カメラ



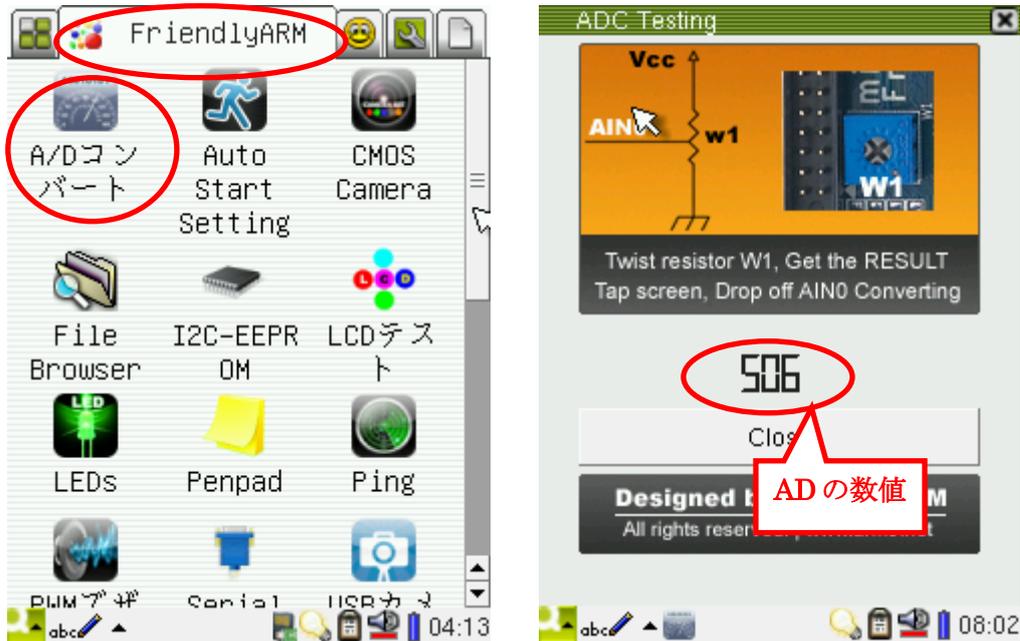
※ Linux-2.6.29はUVCとgspacクラスのUSBカメラをサポートします。

3.15 CMOS イメージセンサー



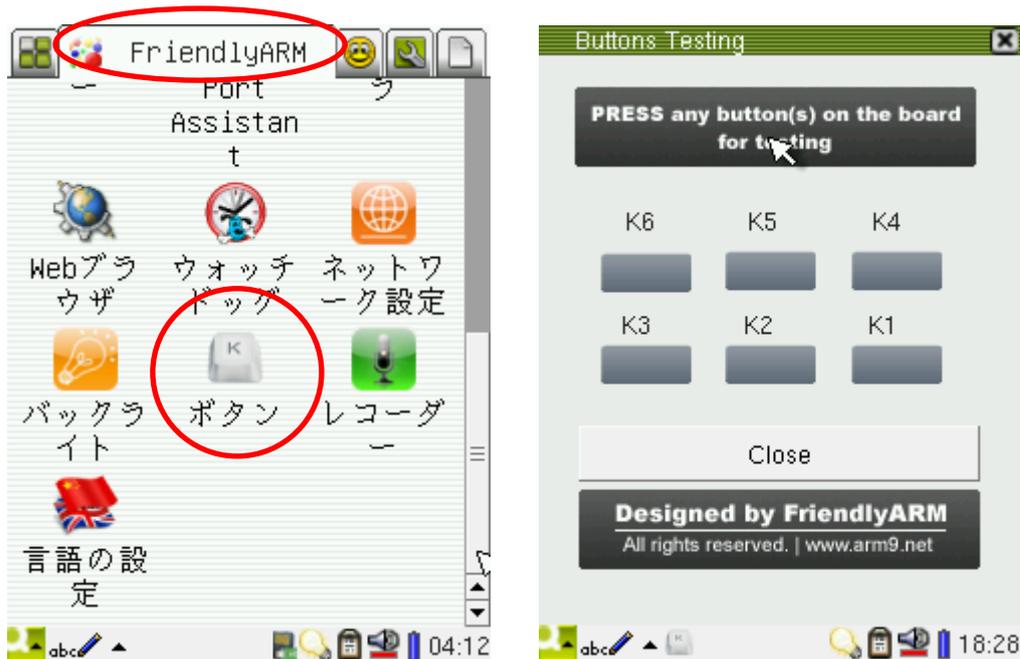
※ OV9650をサポートします。

3.16 AD テスト

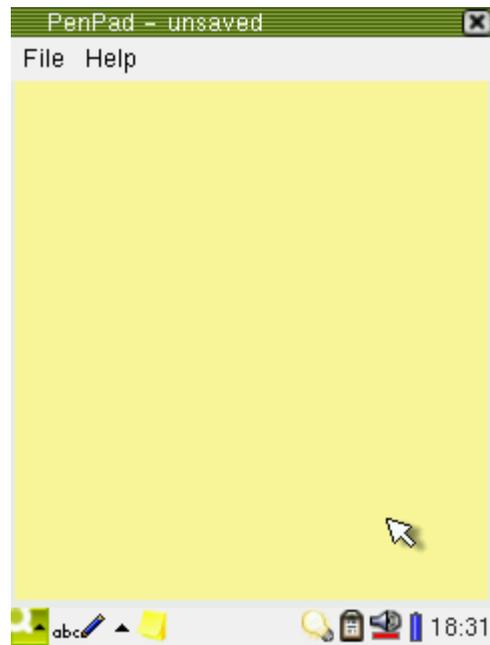


ARM9 ボードの可変抵抗を回ると、AD の数値が変化します。

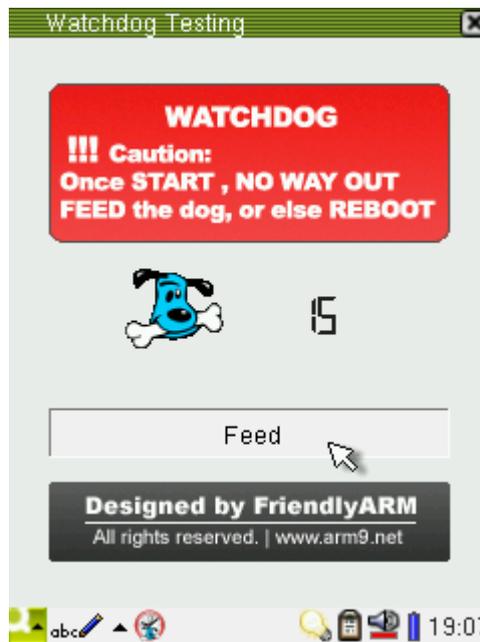
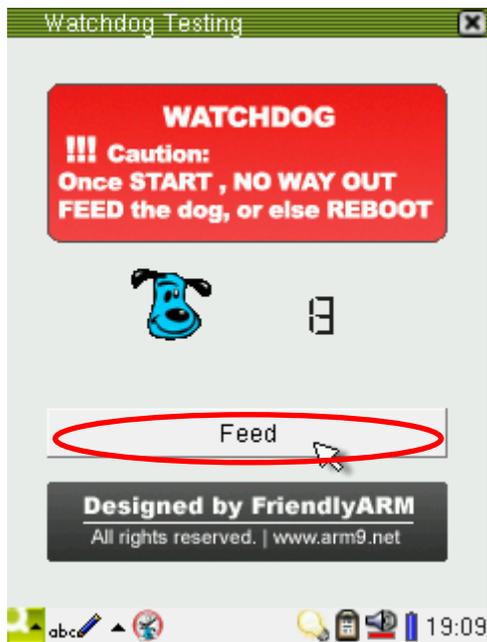
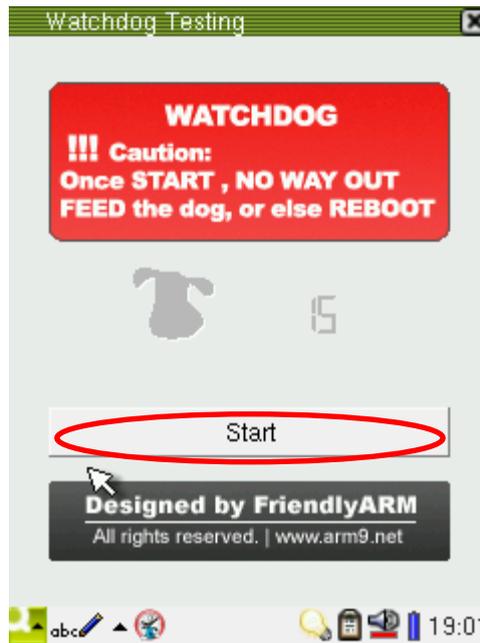
3.17 ボタン



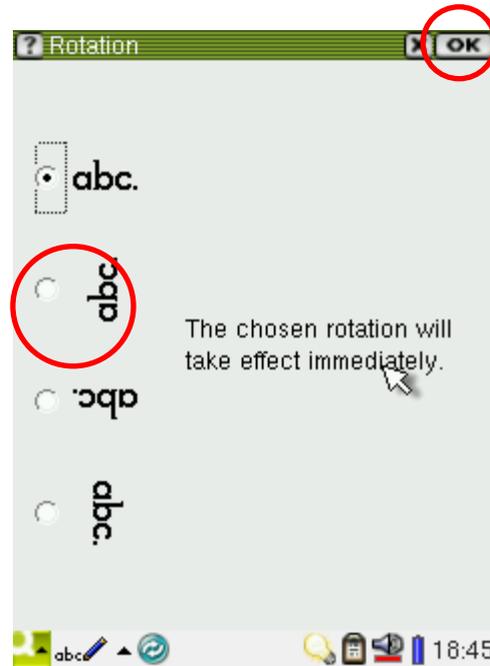
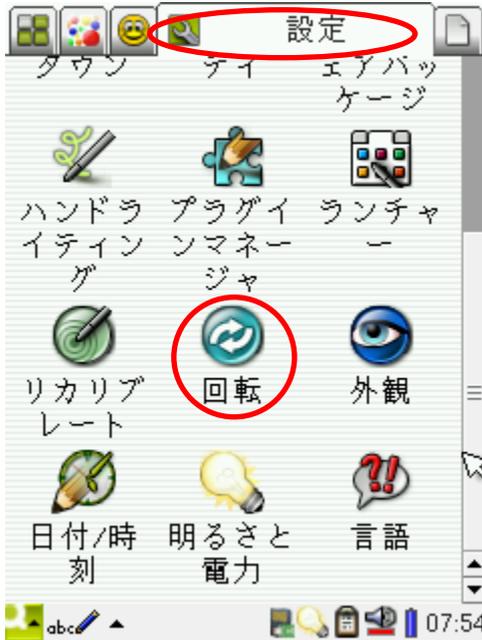
3.18 手書き



3.19 Watch dog



3.20 回転



ほしい方向を選択して、「OK」ボタンを押します。
システムの再起動が必要かもしれません。

3.21 スタートアップ

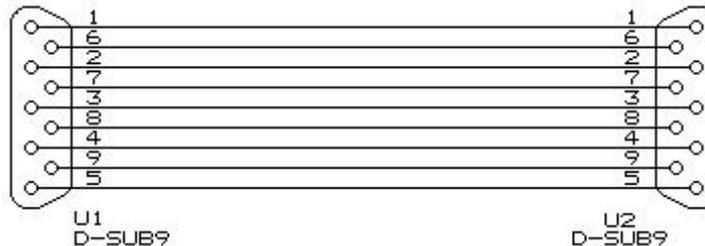


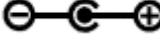
起動の時、アプリケーションを自動的に実行させます。Windows のスタートアップにみたい機能です。

例は起動の時、時計を自動的に実行させます。

第四章 初体験(コンソール)

- DB9 メス-メス型のストレートケーブルで mini2240 とパソコンを繋ぐ。



- クロス LAN ケーブルで mini2240 とパソコンを繋ぐ。
- mini2240 のオーディオ出力とスピーカーを繋ぐ。
- 5V 電源、極性はセンタープラス  です。

4.1 パソコン側のハイパーターミナルの設定

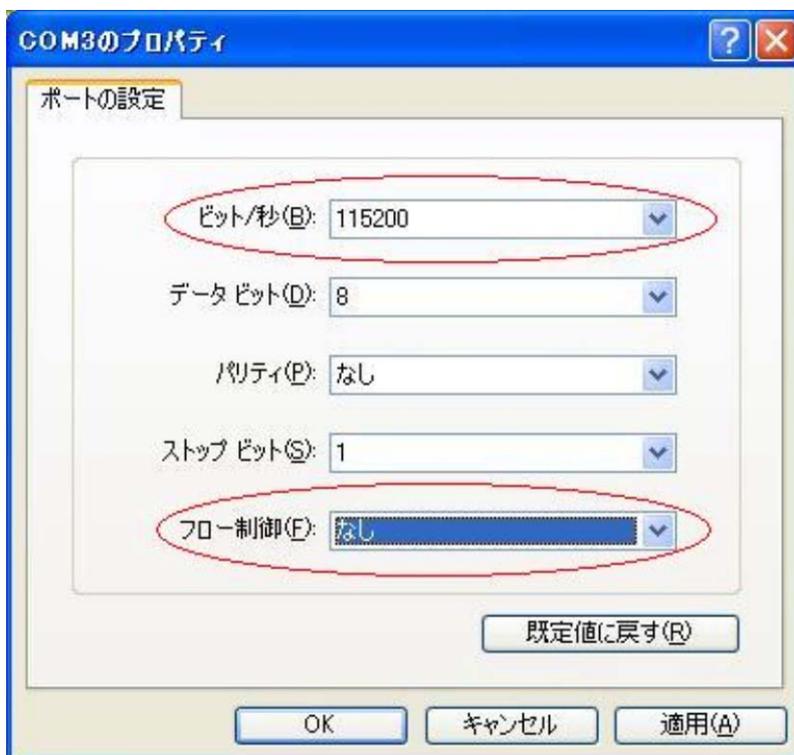
パソコンのメニュー：スタート → すべてのプログラム → アクセサリ → 通信 → ハイパーターミナルを選ぶと、次の画面が出てきます。



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。



使用したいシリアルポートを選んでください。



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。

4.2 MP3 の再生

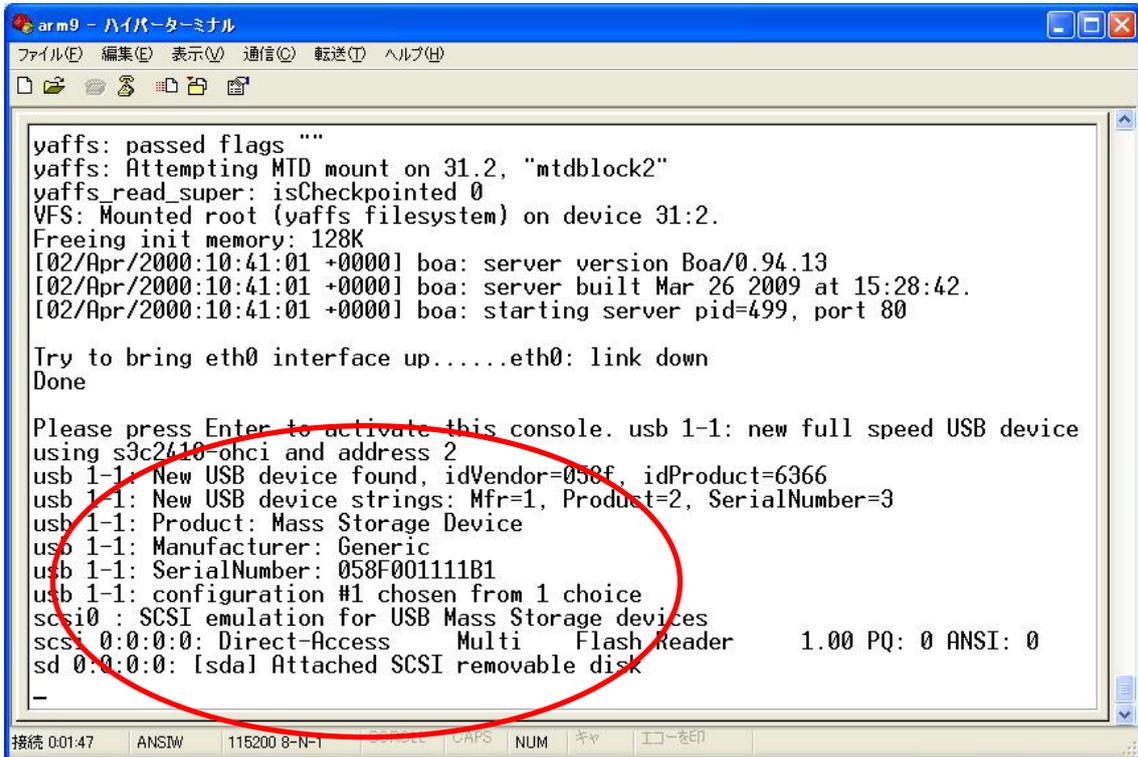
コマンド	madplay
ソースコード	madplay.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI

madplay your.mp3

このコマンドは your.mp3 というファイルを再生します。自分で your.mp3 ファイルを用意してください。「Ctrl+c」で停止させます。

4.3 USB メモリと外付けハードデスク

USB メモリを USB ホスト又は USB ハブに挿入すると



```
arm9 - ハイパーターミナル
ファイル(F) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.2, "mtdblock2"
yaffs_read_super: isCheckpointed 0
VFS: Mounted root (yaffs filesystem) on device 31:2.
Freeing init memory: 128K
[02/Apr/2000:10:41:01 +0000] boa: server version Boa/0.94.13
[02/Apr/2000:10:41:01 +0000] boa: server built Mar 26 2009 at 15:28:42.
[02/Apr/2000:10:41:01 +0000] boa: starting server pid=499, port 80

Try to bring eth0 interface up.....eth0: link down
Done

Please press Enter to activate this console. usb 1-1: new full speed USB device
using s3c2410-ohci and address 2
usb 1-1: New USB device found, idVendor=058f, idProduct=6366
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1: Product: Mass Storage Device
usb 1-1: Manufacturer: Generic
usb 1-1: SerialNumber: 058F001111B1
usb 1-1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
scsi 0:0:0:0: Direct-Access Multi Flash Reader 1.00 PQ: 0 ANSI: 0
sd 0:0:0:0: Isda1 Attached SCSI removable disk
-
```

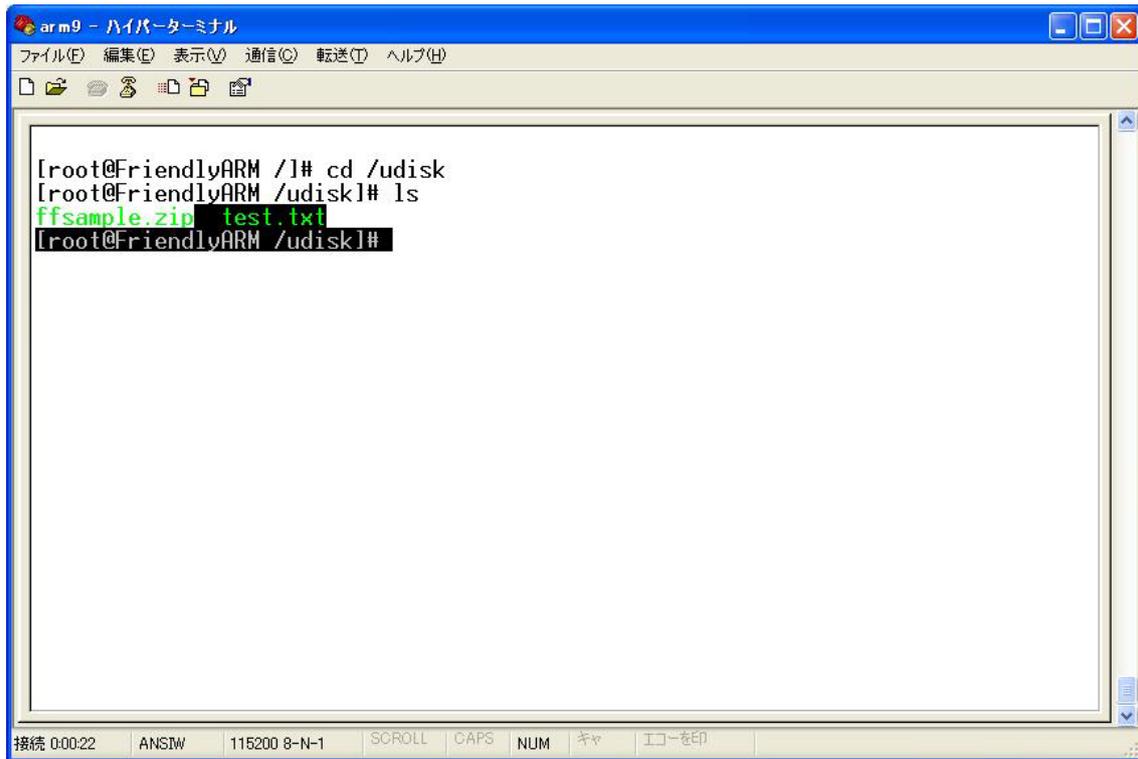
自動的にこのような情報が出てきます。USB メモリのデバイス名は/dev/udisk です。システムは自動的に/dev/udisk にマウントします。

※ **FAT32/VFAT** だけの USB メモリが認識できます。

/dev/udisk に移動して、USB メモリのファイルをリストします。

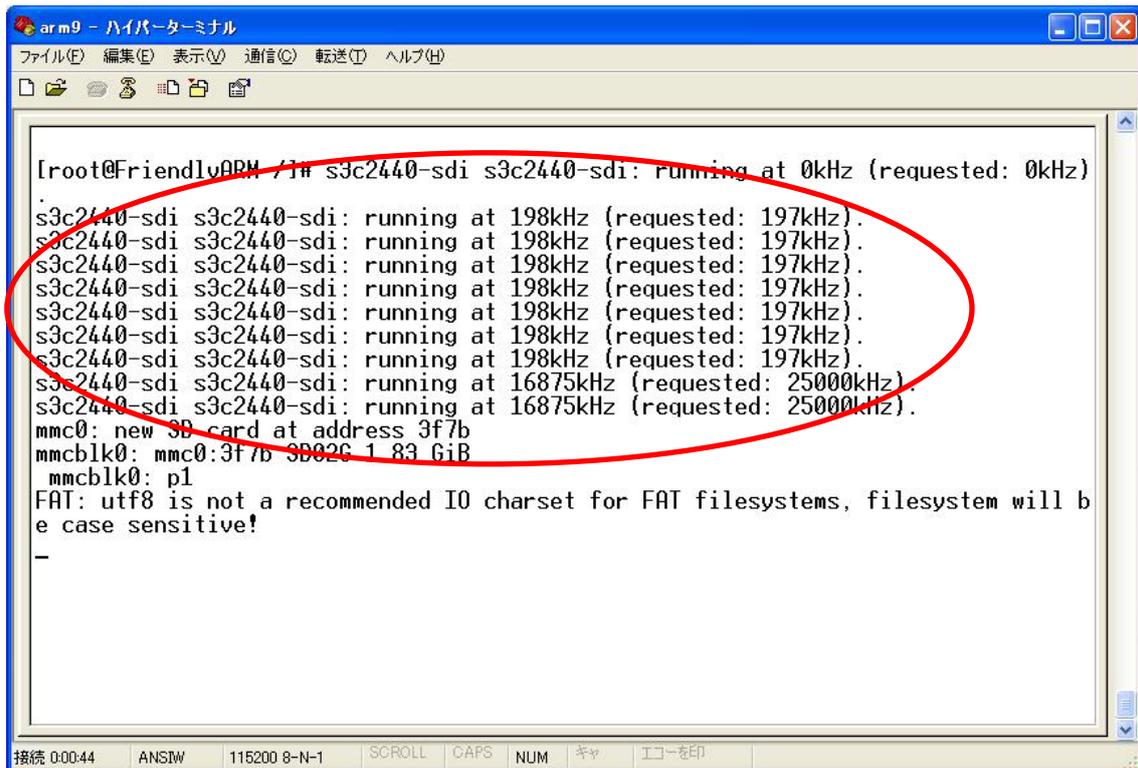
```
# cd /dev/udisk
```

```
# ls
```



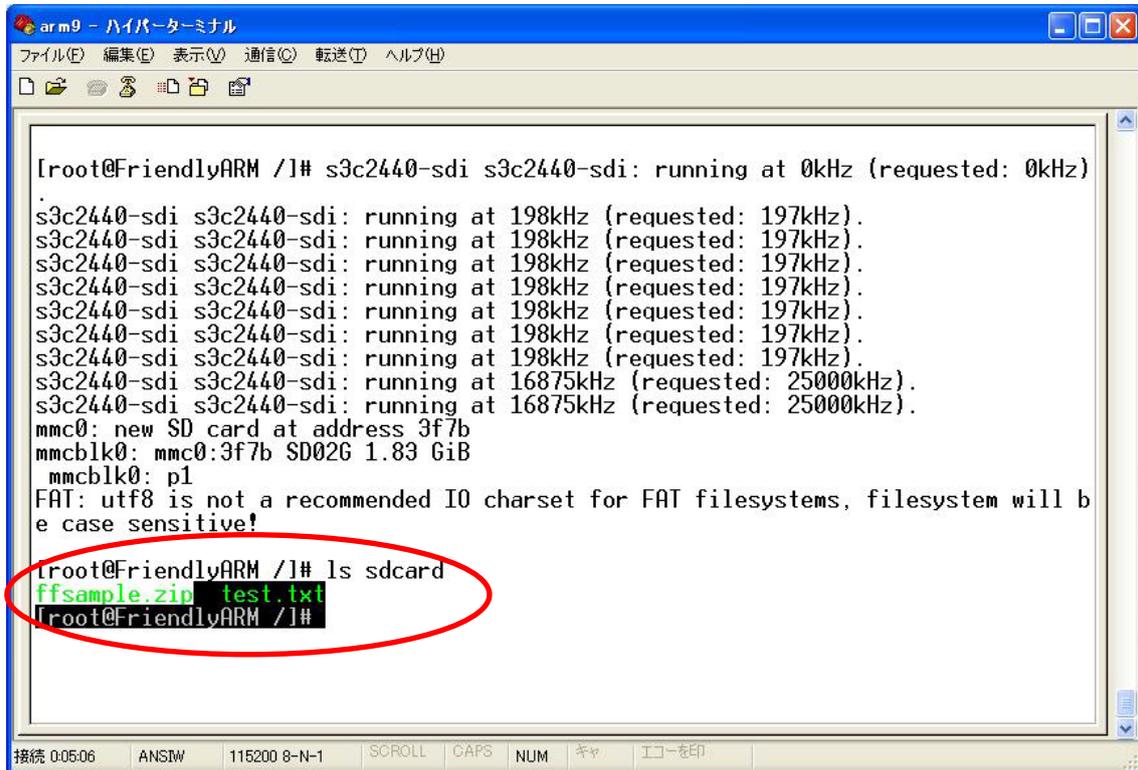
```
ar m9 - ハイパーターミナル
ファイル(F) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)
[root@FriendlyARM /]# cd /udisk
[root@FriendlyARM /udisk]# ls
ffsample.zip  test.txt
[root@FriendlyARM /udisk]#
```

4.4 SD/MMC カード



```
ar m9 - ハイパーターミナル
ファイル(F) 編集(E) 表示(V) 通信(C) 転送(T) ヘルプ(H)
[root@FriendlyARM /]# s3c2440-sdi s3c2440-sdi: running at 0kHz (requested: 0kHz)
.
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
mmc0: new SD card at address 3f7b
mmcblk0: mmc0:3f7b:3B02C 1.83 GiB
mmcblk0: p1
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case sensitive!
-
```

SD/MMC カードを ARM9 に挿入すると、この情報が出てきます。システムは自動的に /sdcard フォルダを生成します。



```
[root@FriendlyARM /]# s3c2440-sdi s3c2440-sdi: running at 0kHz (requested: 0kHz)
.
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
mmc0: new SD card at address 3f7b
mmcblk0: mmc0:3f7b SD02G 1.83 GiB
  mmcblk0: p1
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case sensitive!
[root@FriendlyARM /]# ls /sdcard
ffsample.zip test.txt
[root@FriendlyARM /]#
```

/sdcard フォルダのファイルをリストします。

```
# ls /sdcard
```

4.5 シリアルポートでファイルを ARM9 にダウンロード



「転送」 → 「ファイルの送信」 を選択、



送信したいファイルを選んで、「送信」ボタンを押します。

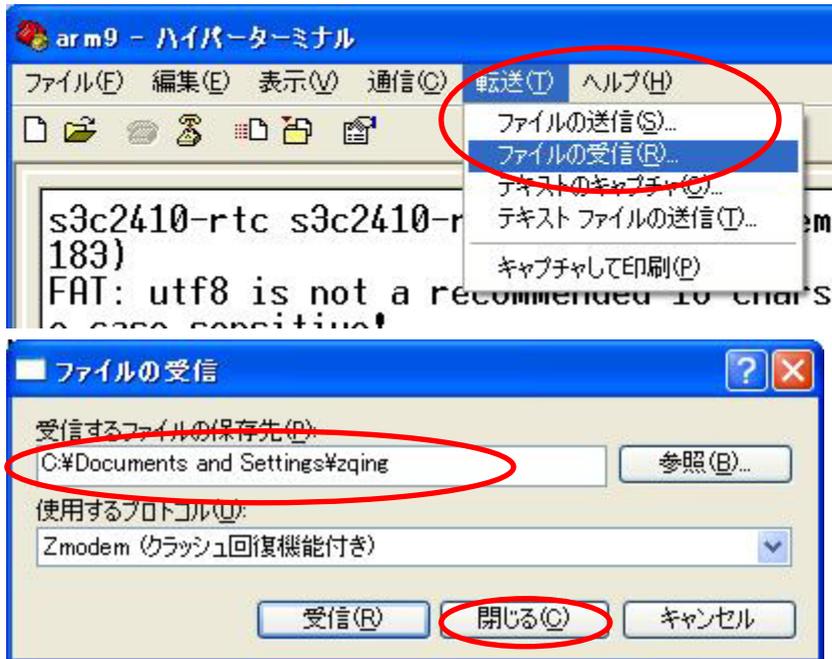


```
[root@FriendlyARM /]# md5sum PRODUCTS.GIF
ae9cf1d3da214a6c985d9de9ece071b9 PRODUCTS.GIF
[root@FriendlyARM /]# _
```

接続 0:03:58 ANSIW 115200 8-N-1 SCROLL CAPS NUM キヤ エコ

転送が正しいかどうか、md5sum コマンドで検証します。

4.6 シリアルポートで ARM9 のファイルを PC に保存



保存先を設定して、「閉じる」ボタンを押します。

sz PRODUCTS.GIF

コマンドで転送が開始します。



4.7 LED 制御

コマンド	led-player leds
ソースコード	led-player.c leds.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/leds
ドライバ	Linux-2.6.29/drivers/char/mini2440_leds.c

1) LED サーバ

システム起動の時、自動的に LED サーバ(**led-player**)を起動させます(**/etc/rc.d/init.d/leds**)。LED を点滅させています。**led-player** を実行した後、**/tmp/led-control** というパイプを生成します。

```
#echo 0 0.2 > /tmp/led-control
```

LED が 0.2 秒周期で流れます。

```
#echo 1 0.2 >/tmp/led-control
```

LED が 0.2 秒周期で累計します。

```
#!/etc/rc.d/init.d/leds stop
```

LED を停止します。

```
#!/etc/rc.d/init.d/leds start
```

LED が点滅をスタートします。

2) 単独 LED 制御

```
#!/etc/rc.d/init.d/leds stop
```

LED サーバをストップさせます。

```
# led
```

Usage: leds led_no 0|1

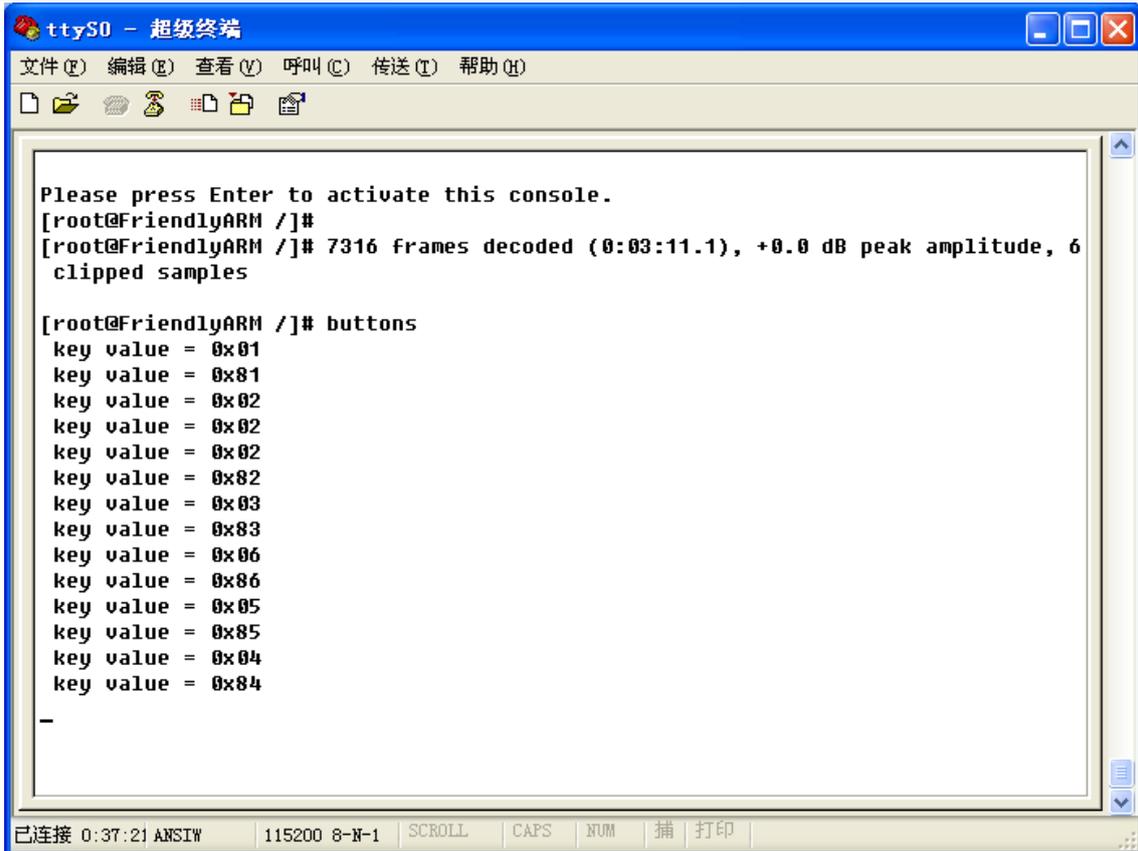
led_no は LED 番号(0,1,2,3)です。

```
#!/led 2 1
```

LED2 を点灯させます。

4.8 ボタンのテスト

コマンド	buttons
ソースコード	buttons_test.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/buttons
ドライバ	Linux-2.6.29/drivers/char/mini2440_buttons.c



```
ttyS0 - 超級终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
Please press Enter to activate this console.
[root@FriendlyARM /]#
[root@FriendlyARM /]# 7316 frames decoded (0:03:11.1), +0.0 dB peak amplitude, 6
clipped samples

[root@FriendlyARM /]# buttons
key value = 0x01
key value = 0x81
key value = 0x02
key value = 0x02
key value = 0x82
key value = 0x03
key value = 0x83
key value = 0x06
key value = 0x86
key value = 0x05
key value = 0x85
key value = 0x04
key value = 0x84
-

已连接 0:37:21 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

#buttons

このコマンドを入力してください。

4.9 シリアルポートのテスト

コマンド	armcomtest
ソースコード	Comtest.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/ttySAC0,1,2 或は /dev/ttyUSB0,1,2 或は /dev/ttyACM0,1,2

- ※ ARM9 は自分の三つのシリアルポートが **/dev/ttySAC0,1,2** です。
- ※ ARM9 は弊社が販売している USB-RS232 変換ケーブルを直接使えます。デバイス名は **/dev/ttyUSB0,1,2** です。
- ※ ARM9 は弊社が販売している ARM7TDMI/LPC2148 を USB で通信できます。ARM7TDMI/LPC2148 のデバイス名は **/dev/ttyACM0,1,2** です。

このコマンドは LPC2148 と通信します。

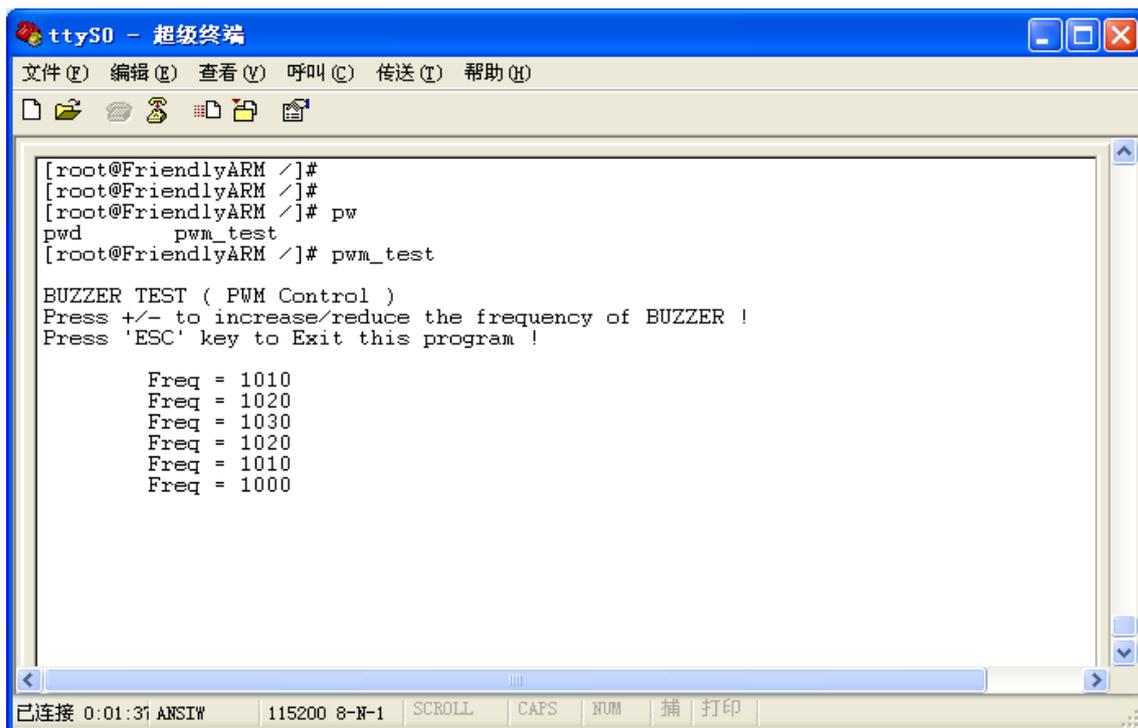
#armcomtest -d /dev/ttyACM0 -o



4.10 ブザー(PWM)のテスト

コマンド	pwm_tset
ソースコード	pwm_tset.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/pwm
ドライバ	Linux-2.6.29/drivers/char/pwm.c

#pwm_test



```
ttyS0 - 超級终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
[ root@FriendlyARM / ]#
[ root@FriendlyARM / ]#
[ root@FriendlyARM / ]# pw
pwd      pwm_test
[ root@FriendlyARM / ]# pwm_test

BUZZER TEST ( PWM Control )
Press +/- to increase/reduce the frequency of BUZZER !
Press 'ESC' key to Exit this program !

      Freq = 1010
      Freq = 1020
      Freq = 1030
      Freq = 1020
      Freq = 1010
      Freq = 1000
```

“+”と“-”キーは音声の周波数を変更させます。”ESC”キーは音声を停止させます。

4.11 LCD のバックライト

#echo 0 > /dev/backlight ;バックライト消灯

#echo 1 > /dev/backlight ;バックライト点灯

4.12 I2C-EEPROM

コマンド	i2c
ソースコード	eeeprom.c 24cXX.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/i2c/0
ドライバ	Linux-2.6.29/drivers/i2c/busses/i2c-s3c2440.c

#i2c -w ;データ(0~255)をボードの 24C08 に書き込む

#i2c -r ;ボードの 24C08 からデータを読み出す

4.13 AD テスト

コマンド	adc-test
ソースコード	adc-test.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/adc
ドライバ	Linux-2.6.29/drivers/char/mini2440_adc.c

#adc-test

ボードの可変抵抗をまわして、AD 数値の変化が見えます。

```
[root@FriendlyARM /]# adc-test
press Ctrl-C to stop
ADC Value: 60
ADC Value: 187
ADC Value: 267
ADC Value: 312
ADC Value: 368
ADC Value: 444
ADC Value: 422
ADC Value: 337
ADC Value: 260
ADC Value: 211
ADC Value: 190
ADC Value: 190
ADC Value: 190
ADC Value: 190
```

4.14 CMOS イメージセンサー

コマンド	camtest
ソースコード	camtest.c
パッケージ	examples.tgz
コンパイル	Arm-linux-gcc-4.3.2 with EABI
デバイス名	/dev/camera
ドライバ	Linux-2.6.29/drivers/media/video/s3c2440camif.c

#camtest



4.15 ネットワーク機能

4.15.1 ウェブサーバー

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で `http://192.168.1.230` を入力すると、mini2440 のホームページが見えます。このホームページを通じて、ユーザーLED と USB カメラ(*)をアクセスできます。

※ **mjpg-streamer** というソフトウェアをインストールする必要があります。

4.15.2 Telnet と Ftp 機能

Linux でクライア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。

4.15.3 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`

4.15.4 MAC アドレスの設定

```
# ifconfig eth0 down
```

```
# ifconfig 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。

4.15.5 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。

```
#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。
```

4.16. 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。

4.17 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 というファイルに保存します。

4.18 USB 無線 LAN

Linux-2.6.29 には rt73 という USB 無線 LAN のドライバを実装しました。このタイプの USB 無線 LAN を ARM9 ボードの USB ホストに挿入すると、次の情報が出てきます。

```
[root@FriendlyARM /]# usb 1-1: new full speed USB device using s3c2410-ohci and address 2
```

```
usb 1-1: New USB device found, idVendor=148f, idProduct=2573
```

```
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
```

```
usb 1-1: Product: 54M.USB.....
```

```
usb 1-1: Manufacturer: Ralink
```

```
usb 1-1: configuration #1 chosen from 1 choice
```

```
wmaster0 (rt73usb): not using net_device_ops yet
```

```
wlan0 (rt73usb): not using net_device_ops yet
```

```
[root@FriendlyARM /]# ifconfig eth0 down
```

有線 LAN をストップ

```
[root@FriendlyARM /]# ifconfig wlan0 up
```

無線 LAN を起動

```
rt73usb 1-1:1.0: firmware: requesting rt73.bin
```

利用可能な無線ネットを探す

```
[root@FriendlyARM /]# iwlist scanning | grep ESSID
```

```
lo Interface doesn't support scanning.
```

```
eth0 Interface doesn't support scanning.
```

```
wmaster0 Interface doesn't support scanning.
```

```
ESSID:"FRIENDLY-ARM"
```

```
ESSID:"NETGEAR"
```

```
ESSID:"TP-LINK"
```

無線ネットの ESSID を入力

```
[root@FriendlyARM /]# iwconfig wlan0 essid "FRIENDLY-ARM"
```

無線ネットのパスワードを入力

```
[root@FriendlyARM /]# iwconfig wlan0 key s:12345
```



無線ネットのルータに接続

```
[root@FriendlyARM /]# iwconfig wlan0 ap auto
```

IP アドレスの設定

```
[root@FriendlyARM /]# ifconfig wlan0 192.168.1.120
```

無線ネットのテスト

```
[root@FriendlyARM /]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=42.804 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=5.020 ms
64 bytes from 192.168.1.1: seq=2 ttl=64 time=5.021 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5.020/17.615/42.804 ms
[root@FriendlyARM /]#
```


再び Fedora のターミナルを開き、次のコマンド：

```
$ arm-linux-gcc -v
```



```
zqing@localhost:~  
ファイル(E) 編集(E) 表示(V) 端末(I) ヘルプ(H)  
[zqing@localhost ~]$ vi ~/.bashrc  
[zqing@localhost ~]$ arm-linux-gcc -v  
Using built-in specs.  
Target: arm-none-linux-gnueabi  
Configured with: /scratch/julian/lite-respin/linux/src/gcc-4.3/configure --build=i686-pc-linux-gnu --host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi --enable-threads --disable-libmudflap --disable-libssp --disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --enable-languages=c,c++ --enable-shared --enable-symvers=gnu --enable-__cxa_atexit --with-pkgversion='Sourcery G++ Lite 2008q3-72' --with-bugurl=https://support.codesourcery.com/GNUToolchain/ --disable-nls --prefix=/opt/codesourcery --with-sysroot=/opt/codesourcery/arm-none-linux-gnueabi/libc --with-build-sysroot=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/libc --with-gmp=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu/usr --with-mpfr=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu/usr --disable-libgomp --enable-poison-system-directories --with-build-time-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin --with-build-time-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin  
Thread model: posix  
gcc version 4.3.2 (Sourcery G++ Lite 2008q3-72)  
[zqing@localhost ~]$ █
```

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。

5.2 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 は大きなホストの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 xvfz root_qtopia.tgz -C /
```

(2) /etc/exports ファイルを編集します。

「/ root_qtopia *(rw, sync, no_root_squash)」を/etc/exports というファイルに入れます。

(3) NFS サーバを起動します。

```
# /etc/init.d/nfs start
```

(4) NFS 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、外部から NFS へアクセスできない可能性があります。

5.3 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 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:MINI2440.arm9.  
net:eth0:off"
```

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。

第六章 Linux 環境のアプリケーションを開発

6.1 Hello, World!

Hello, World のソースコードは examples.tgz にあります。

```
#include <stdio.h>
```

```
int main(void) {  
    printf("hello, FriendlyARM!%n");  
}
```

6.2 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。

6.3 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方法があります。USB・SDメモリ、シリアルポート、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。

6.4 ほかのサンプル

examples.tgzに幾つのサンプルがあります。

adc-test	ADCテスト
buttons	ボタンテスト
c++	C++サンプル
camtest	CMOSイメージセンサーテスト
comtest	シリアルポートテスト
hello	
i2c	i2c EEPROMテスト
led-player	pipeサンプル
leds	LED点灯
math	数学処理
pthread	スレッド

pwm PWMテスト
udptalk UDP通信サンプル

6.5 Qt/Embedded GUI プログラムを作る

詳細の QT 組込開発手順は「[QT-Embedded_DEV_manual.pdf](#)」を参照

```
# tar zxvf arm-qtopia-2.2.0.tar.gz -C /opt/FriendlyARM/mini2440
```

Qtopia-2.2.0 のパッケージをディレクトリ `/opt/FriendlyARM/mini2440` に展開します。

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

```
# ./build-all       長い時間がかかりますので、我慢してください
```

```
# ./mktarget       ターゲットに書き込むファイル target-qtopia-konq.tgz を生成  
サンプル hello をコンパイルします
```

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

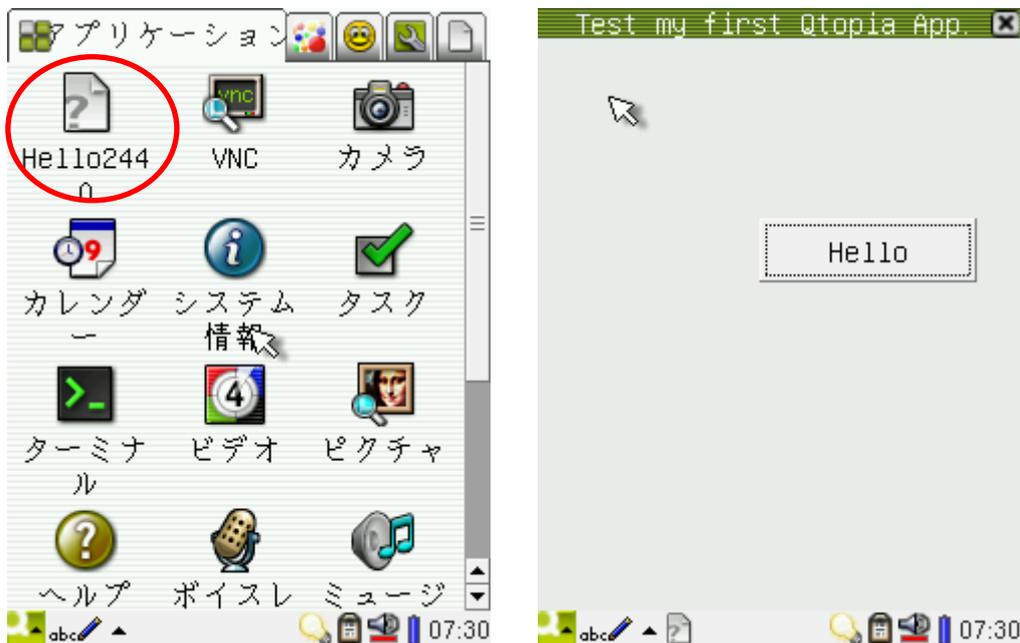
```
# ./build
```

コンパイル完了すれば、ディレクトリ `arm-qtopia/qtopia-2.2.0-FriendlyARM/qtopia/bin` に実行ファイル `hello` を生成させます。実行ファイル `hello` と `arm-qtopia/hello/hello.desktop` を ARM9 ボードにダウンロードして、ARM9 の適当なディレクトリに移転します。

```
#chmod +x hello
```

```
#mv hello /opt/Qtopia/bin
```

```
#mv hello.desktop /opt/Qtopia/apps/Applications
```



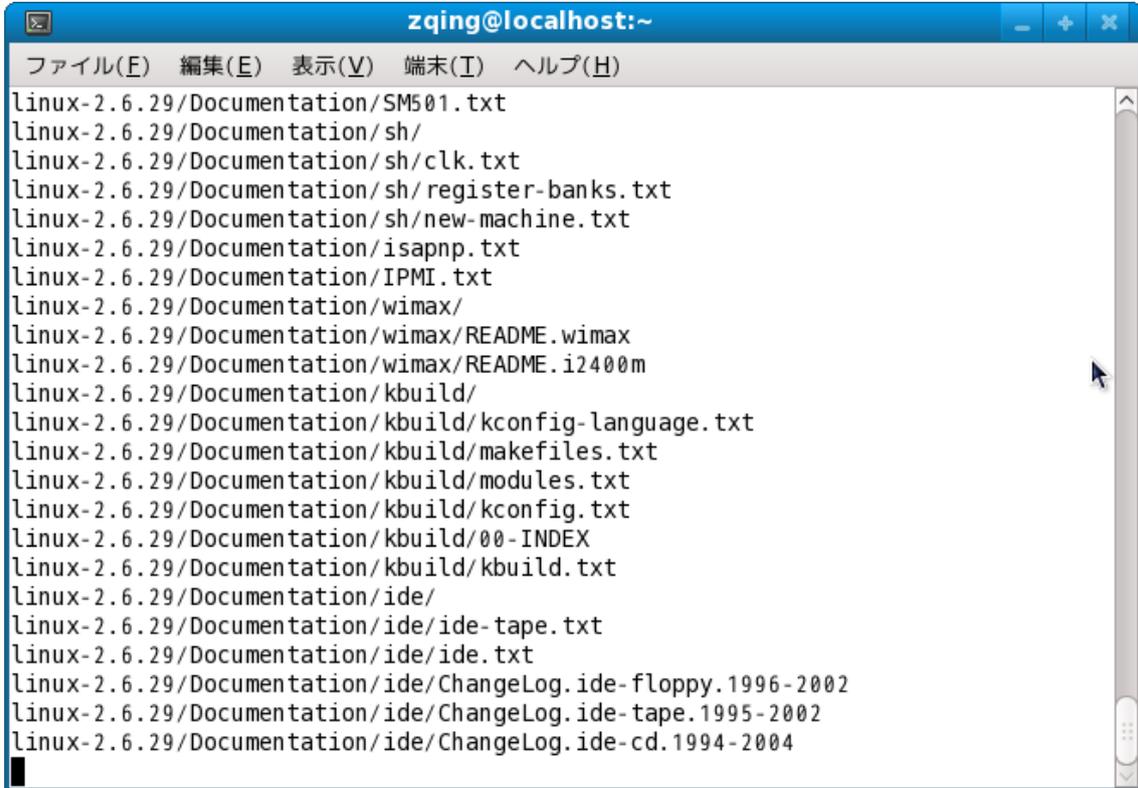
再起動すれば、デスクトップで `hello` のアイコンが見えます。実行させると...

第七章 Linux カーネルを再構築

7.1 カーネルのソースコードを解凍

```
$ tar xvzf linux-2.6.29-mini2440-20090609.tgz
```

※ このカーネルはQQ2440v3/MINI2440に両対応する

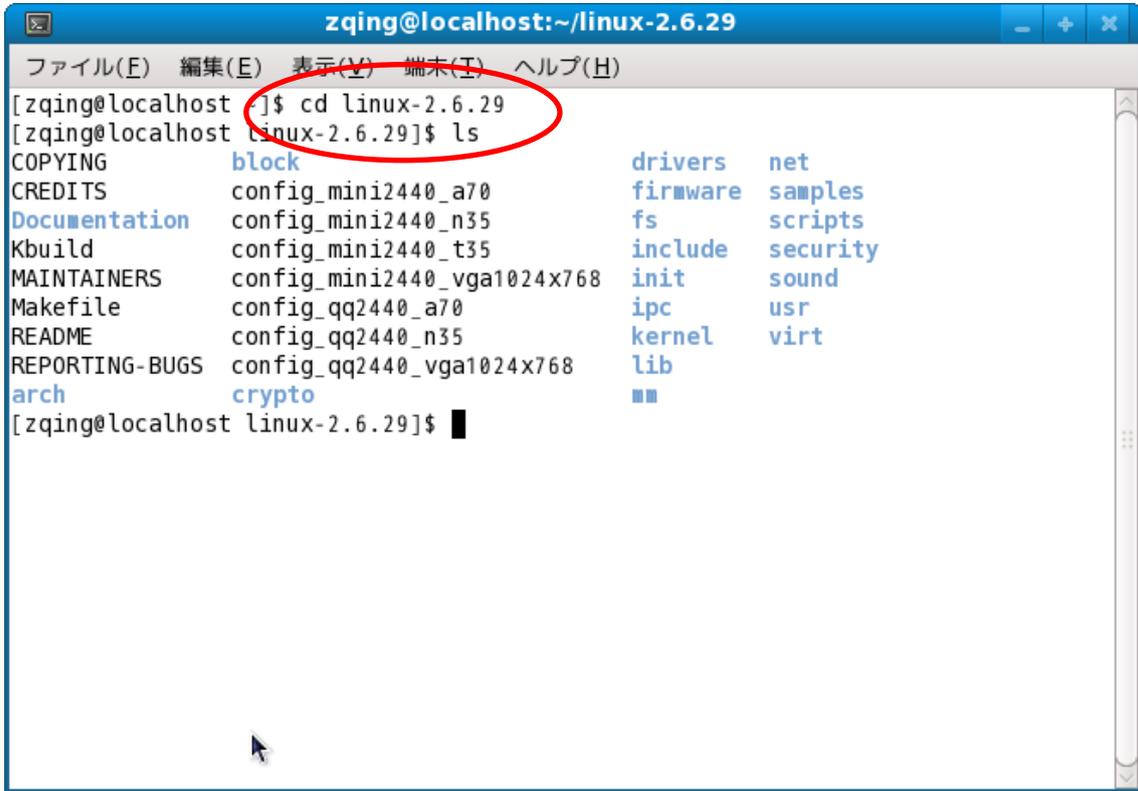


```
zqing@localhost:~  
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)  
linux-2.6.29/Documentation/SM501.txt  
linux-2.6.29/Documentation/sh/  
linux-2.6.29/Documentation/sh/clock.txt  
linux-2.6.29/Documentation/sh/register-banks.txt  
linux-2.6.29/Documentation/sh/new-machine.txt  
linux-2.6.29/Documentation/isapnp.txt  
linux-2.6.29/Documentation/IPMI.txt  
linux-2.6.29/Documentation/wimax/  
linux-2.6.29/Documentation/wimax/README.wimax  
linux-2.6.29/Documentation/wimax/README.i2400m  
linux-2.6.29/Documentation/kbuild/  
linux-2.6.29/Documentation/kbuild/kconfig-language.txt  
linux-2.6.29/Documentation/kbuild/makefiles.txt  
linux-2.6.29/Documentation/kbuild/modules.txt  
linux-2.6.29/Documentation/kbuild/kconfig.txt  
linux-2.6.29/Documentation/kbuild/00-INDEX  
linux-2.6.29/Documentation/kbuild/kbuild.txt  
linux-2.6.29/Documentation/ide/  
linux-2.6.29/Documentation/ide/ide-tape.txt  
linux-2.6.29/Documentation/ide/ide.txt  
linux-2.6.29/Documentation/ide/ChangeLog.ide-floppy.1996-2002  
linux-2.6.29/Documentation/ide/ChangeLog.ide-tape.1995-2002  
linux-2.6.29/Documentation/ide/ChangeLog.ide-cd.1994-2004
```

7.2 Linux を再コンパイル

```
$ cd linux-2.6.29
```

```
$ ls ファイルをリストします
```



```
zqing@localhost:~/linux-2.6.29
ファイル(E) 編集(E) 表示(V) 端末(T) ヘルプ(H)
[zqing@localhost ~]$ cd linux-2.6.29
[zqing@localhost linux-2.6.29]$ ls
COPYING          block            drivers          net
CREDITS          config_mini2440_a70  firmware        samples
Documentation    config_mini2440_n35  fs              scripts
Kbuild           config_mini2440_t35  include         security
MAINTAINERS      config_mini2440_vga1024x768  init           sound
Makefile         config_qq2440_a70    ipc             usr
README          config_qq2440_n35   kernel          virt
REPORTING-BUGS  config_qq2440_vga1024x768  lib
arch             crypto           ■■
```

config_mini2440_xxxはmini/micro2440用コンフィグファイルです。

config_qq2440_xxxはmini2440用コンフィグファイルです。

a70は7インチ用コンフィグファイルです。

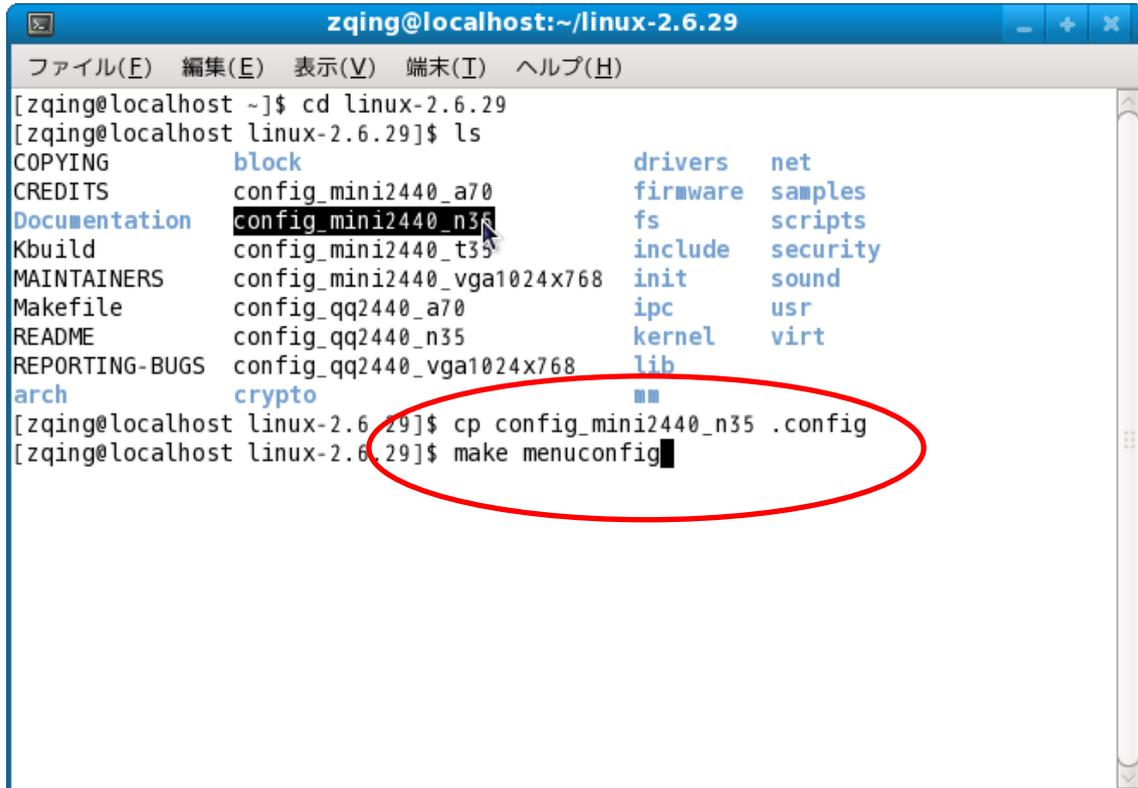
n35は3.5インチ用コンフィグファイルです。

vga1024X768はVGA用コンフィグファイルです。

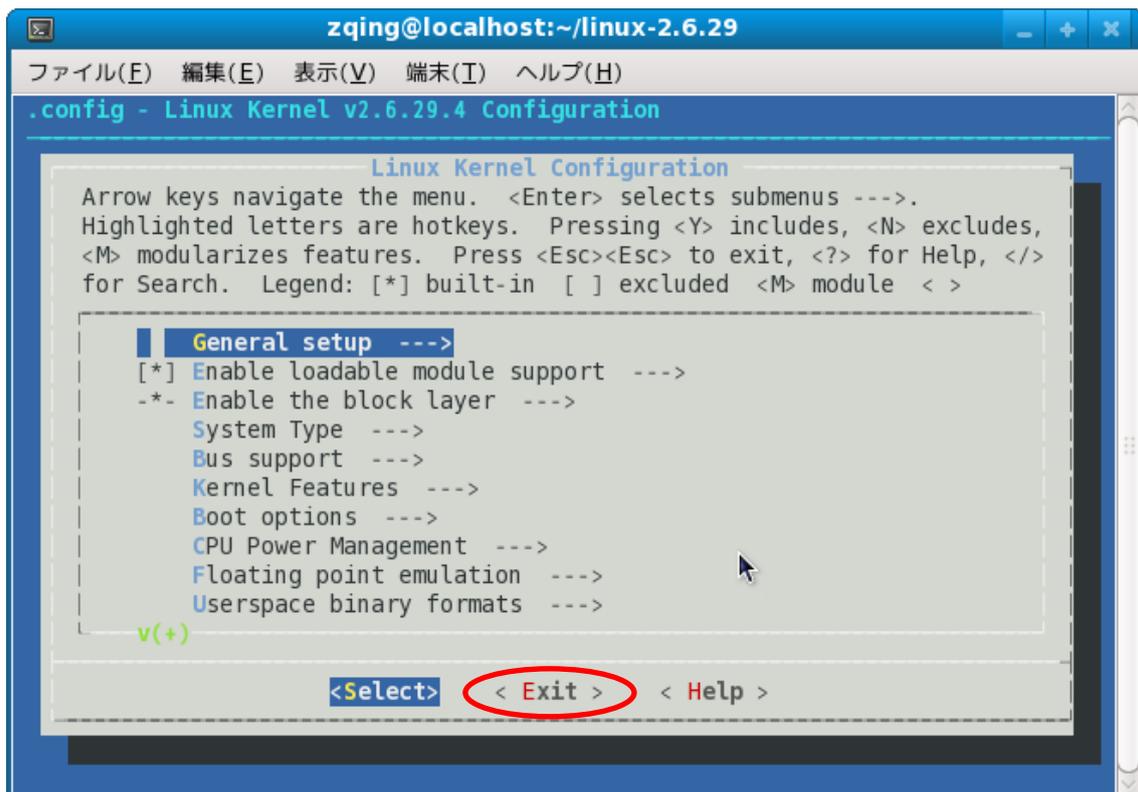
```
$ cp config_mini2440_n35 .config
```

あるコンフィグファイルを選択します

```
$ make menuconfig
```



```
zqing@localhost:~/linux-2.6.29
ファイル(E) 編集(E) 表示(V) 端末(I) ヘルプ(H)
[zqing@localhost ~]$ cd linux-2.6.29
[zqing@localhost linux-2.6.29]$ ls
COPYING          block            drivers          net
CREDITS          config_mini2440_a70  firmware        samples
Documentation    config_mini2440_n35  fs              scripts
Kbuild           config_mini2440_t35  include         security
MAINTAINERS      config_mini2440_vga1024x768  init           sound
Makefile         config_qq2440_a70    ipc            usr
README           config_qq2440_n35    kernel          virt
REPORTING-BUGS  config_qq2440_vga1024x768  lib
arch             crypto           ##
[zqing@localhost linux-2.6.29]$ cp config_mini2440_n35 .config
[zqing@localhost linux-2.6.29]$ make menuconfig
```



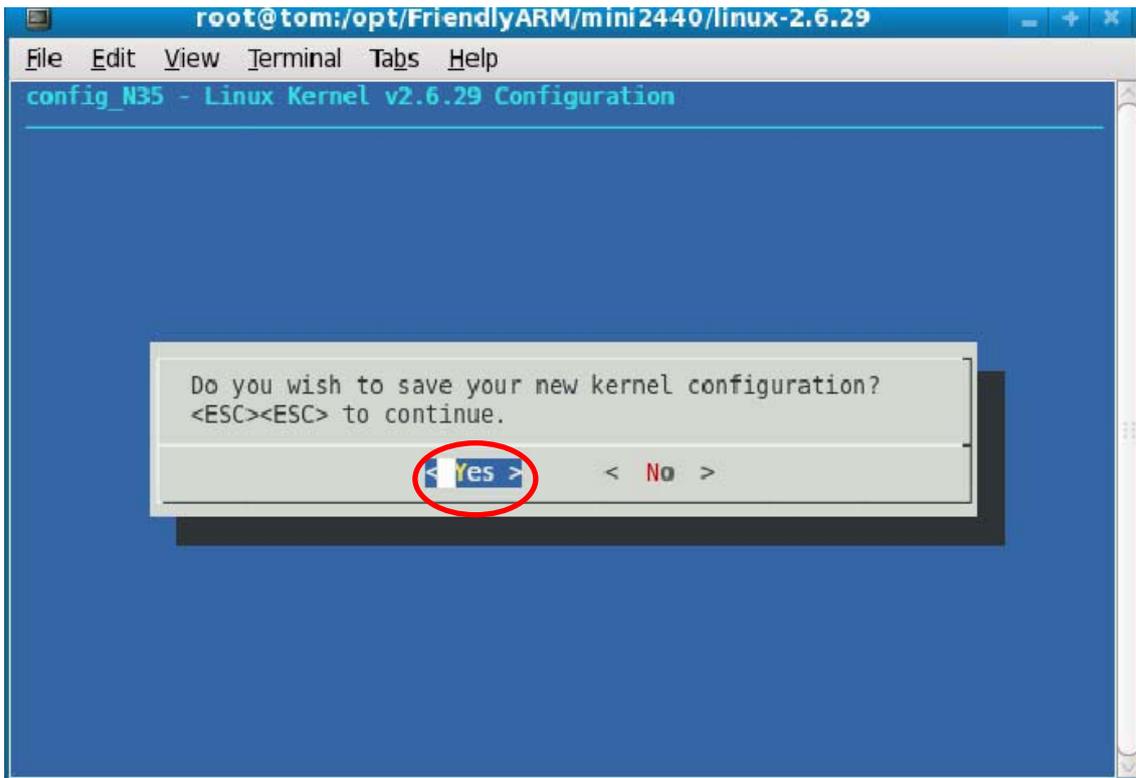
```
zqing@localhost:~/linux-2.6.29
ファイル(E) 編集(E) 表示(V) 端末(I) ヘルプ(H)
.config - Linux Kernel v2.6.29.4 Configuration

Linux Kernel Configuration
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 < >

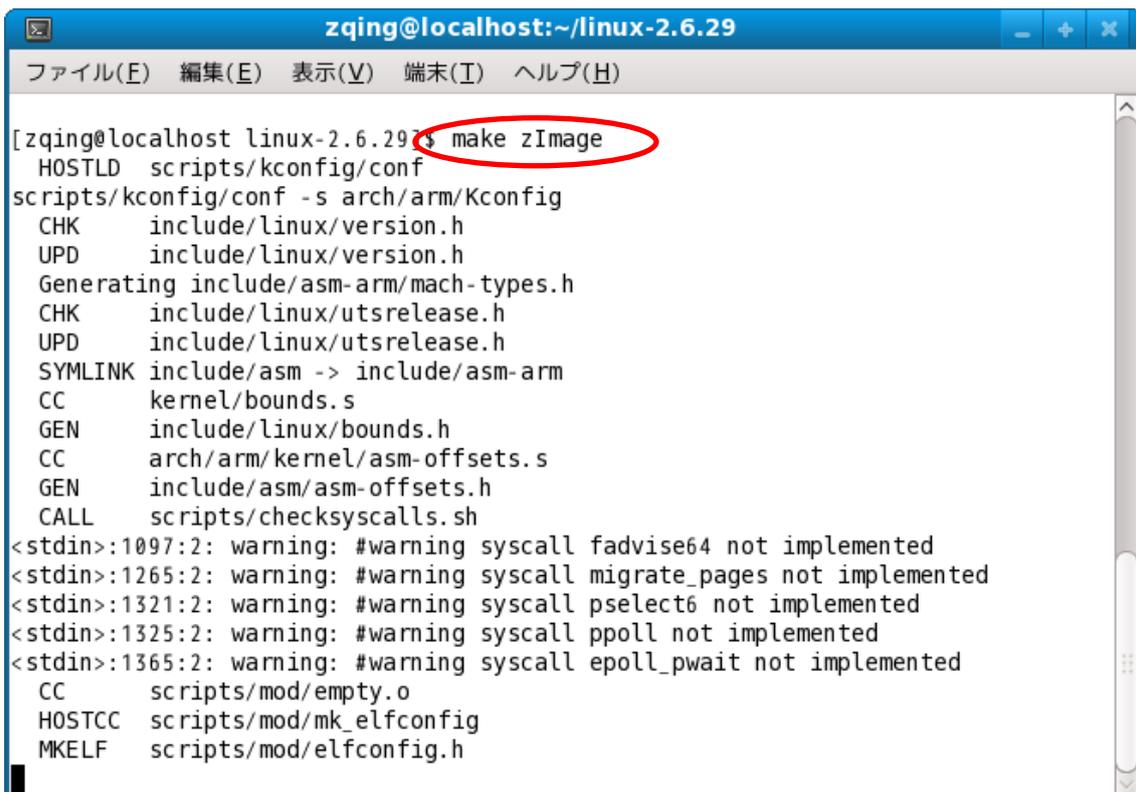
[*] General setup ---
[*] Enable loadable module support ---
[*] Enable the block layer ---
System Type ---
Bus support ---
Kernel Features ---
Boot options ---
CPU Power Management ---
Floating point emulation ---
Userspace binary formats ---

v(+)
```

何も変更しません、「Exit」を選択します。



この画面が出てきたら、「Yes」を押します。**make zImage**でコンパイルします。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
CC      init/version.o
LD      init/built-in.o
LD      .tmp_vmlinux1
KSYM    .tmp_kallsyms1.S
AS      .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM    .tmp_kallsyms2.S
AS      .tmp_kallsyms2.o
LD      vmlinux
SYSMAP  System.map
SYSMAP  .tmp System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP    arch/arm/boot/compressed/piggy.gz
AS      arch/arm/boot/compressed/piggy.o
CC      arch/arm/boot/compressed/misc.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
[root@tom linux-2.6.29]# ls arch/arm/boot/
bootp compressed Image install.sh Makefile zImage
[root@tom linux-2.6.29]#
```

コンパイル完了すると、**arch/arm/boot**フォルダに**Linux**のカーネル**zImage**を生成させます。

7.3 ドライバの場所

(1)DM9000 10/100Mイーサネット

Linux-2.6.29/drivers/net/dm9000.c

(2)シリアルポート(三つのシリアルポート**0,1,2**, デバイス名/**dev/ttySAC0,1,2**)

Linux-2.6.29/drivers/serial/s3c2440.c

(3)リアルタイマー**RTC**

Linux-2.6.29/drivers rtc/rtc-s3c.c

(4)**LED**

Linux-2.6.29/drivers/char/mini2440_leds.c

(5)ボタン

Linux-2.6.29/drivers/char/mini2440_buttons.c

(6)タッチパネル

Linux-2.6.29/drivers/input/touchscreen/s3c2410_ts.c

(7)**yaffs2**ファイルシステム

Linux-2.6.29/fs/yaffs2

(8)**USB**マウス、キーボード

Linux-2.6.29/drivers/usb/hid

**(9)SD/MMCメモリカード(最大32GB)**

Linux-2.6.29/drivers/mmc

(10)Nand Flash

Linux-2.6.29/drivers/mtd/nand

(11)UDA1341オーディオ

Linux-2.6.29/sound/soc/s3c24xx

(12)LCD液晶

Linux-2.6.29/drivers/video/s3c2410fb.c

(13)USBメモリ

Linux-2.6.29/drivers/usb/storage

(14)gspca類USBカメラ

Linux-2.6.29/drivers/media/video/gspca

(15)I2C-EEPROM

linux-2.6.29/drivers/i2c

(16)バックライト

Linux-2.6.29/drivers/video/mini2440_backlight.c

(17)PWMブザー

Linux-2.6.29/drivers/char/mini2440_pwm.c

(18)Watchdog

Linux-2.6.29/drivers/watchdog/s3c2410_wdt.c

(19)ADC

Linux-2.6.29/drivers/char/mini2440_ad.c

(20)CMOSイメージセンサー

Linux-2.6.29/drivers/media/video/s3c2440camif.c

(21)USB無線LAN

Linux-2.6.29/drivers/net/wireless/rt2x00

(22)USB-RS232変換ケーブル

Linux-2.6.29/drivers/usb/serial/pl2302.c

(23)CDC ACM(ARM7TDMI/LPC2148通信)

Linux-2.6.29/drivers/usb/class/

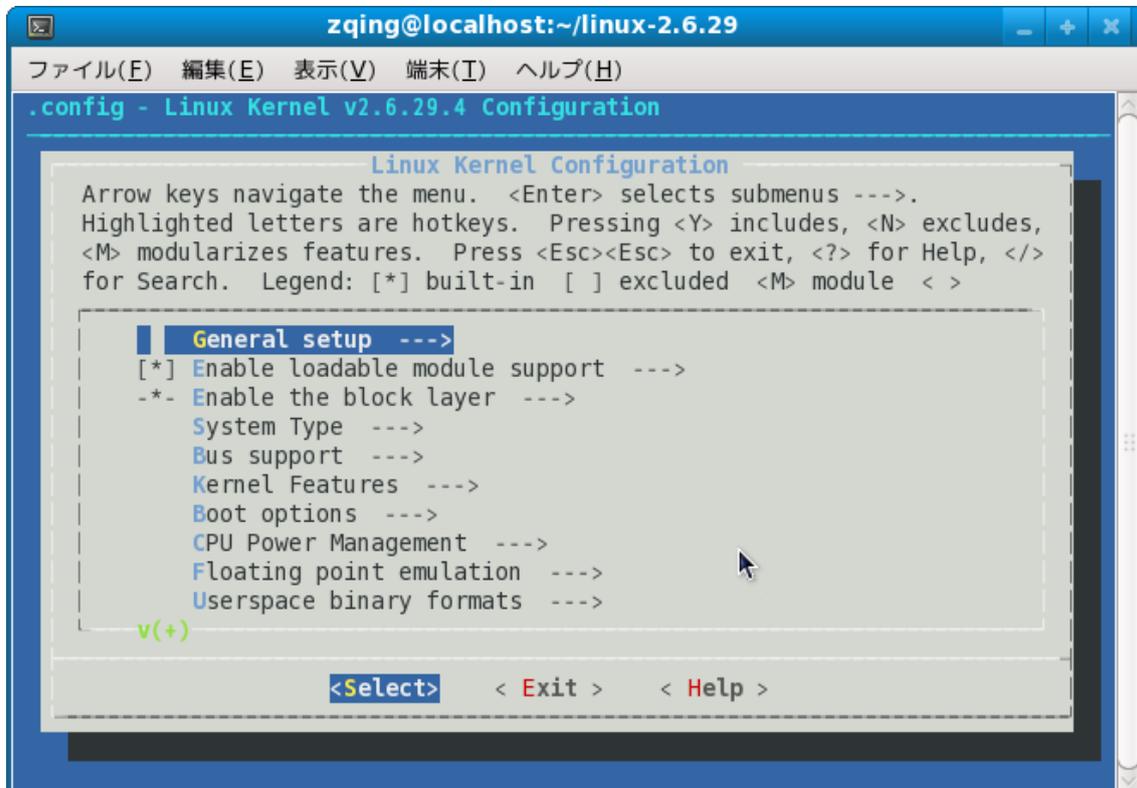
7.4 Linux カーネルのコンフィグ

```
$ cd linux-2.6.29
```

```
$ cp config_mini2440_n35 .config
```

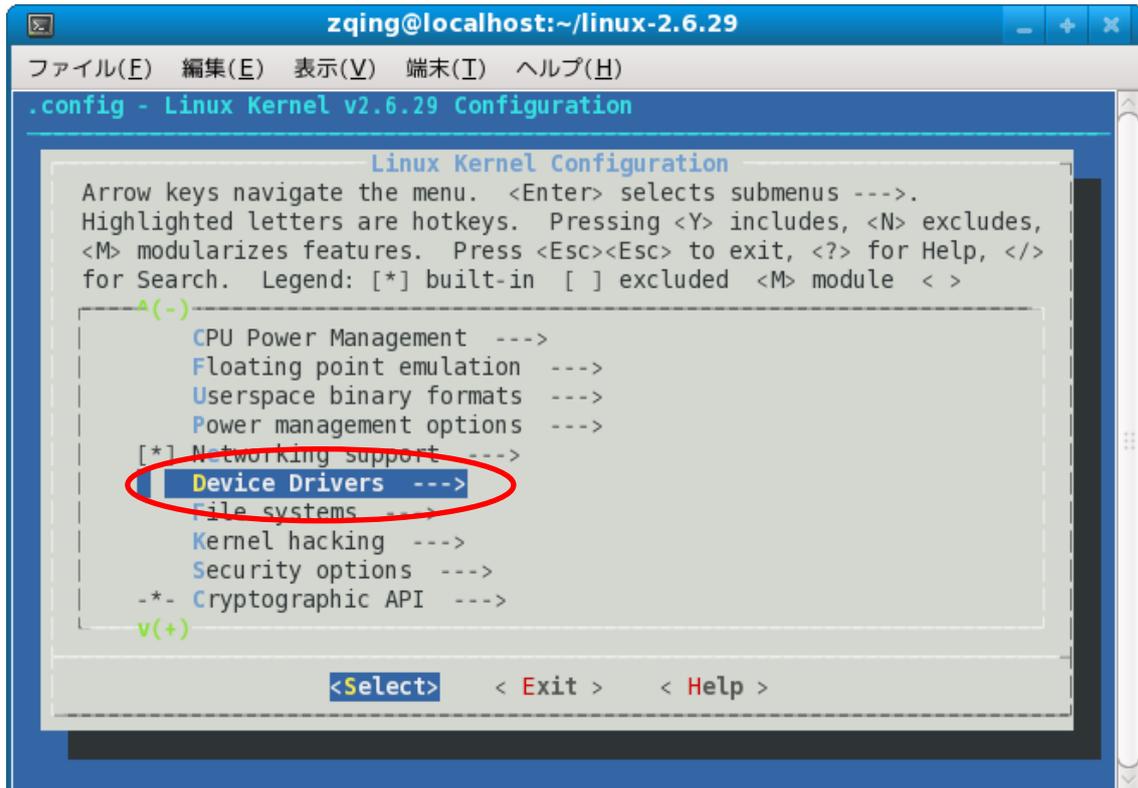
あるコンフィグファイルを選択します

```
$ make menuconfig
```

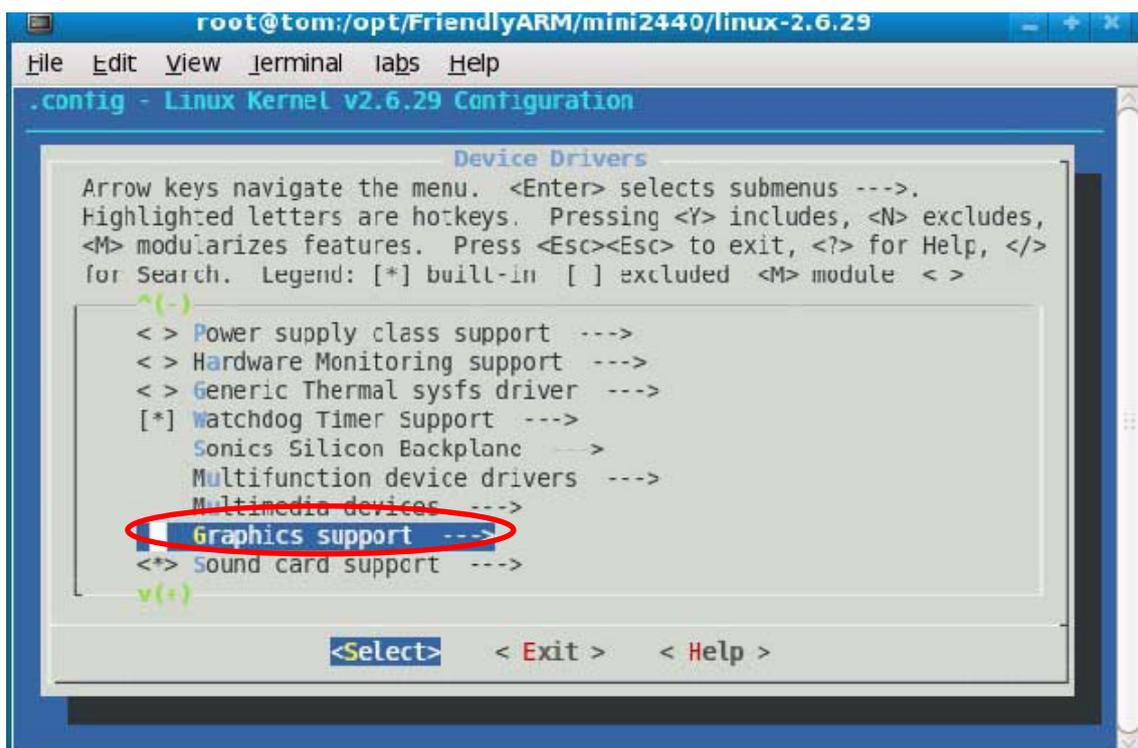


初のコンフィグ画面です。

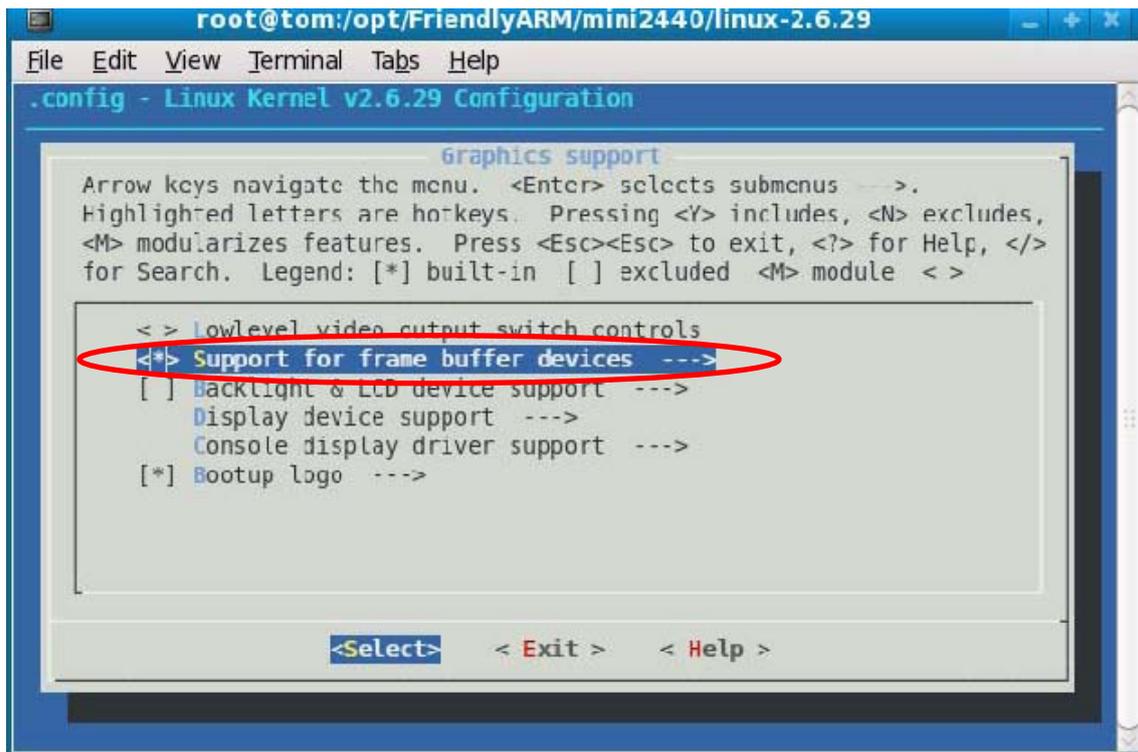
7.4.1 LCD 液晶とバックライト



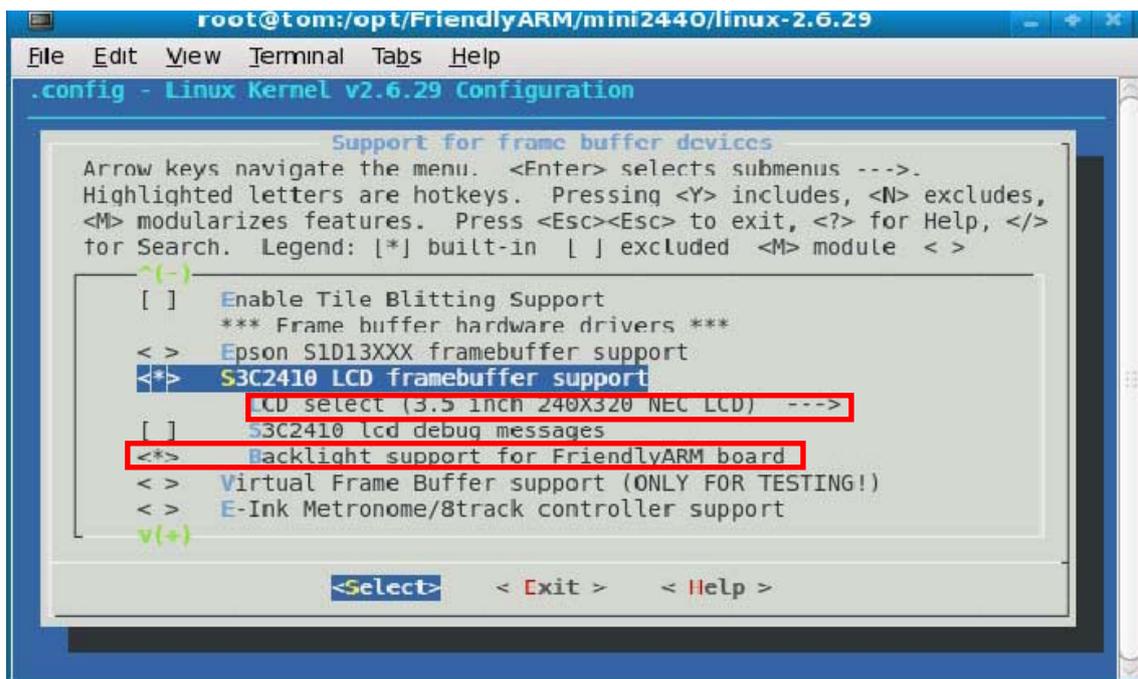
初の画面で「Device Drivers」を選択、



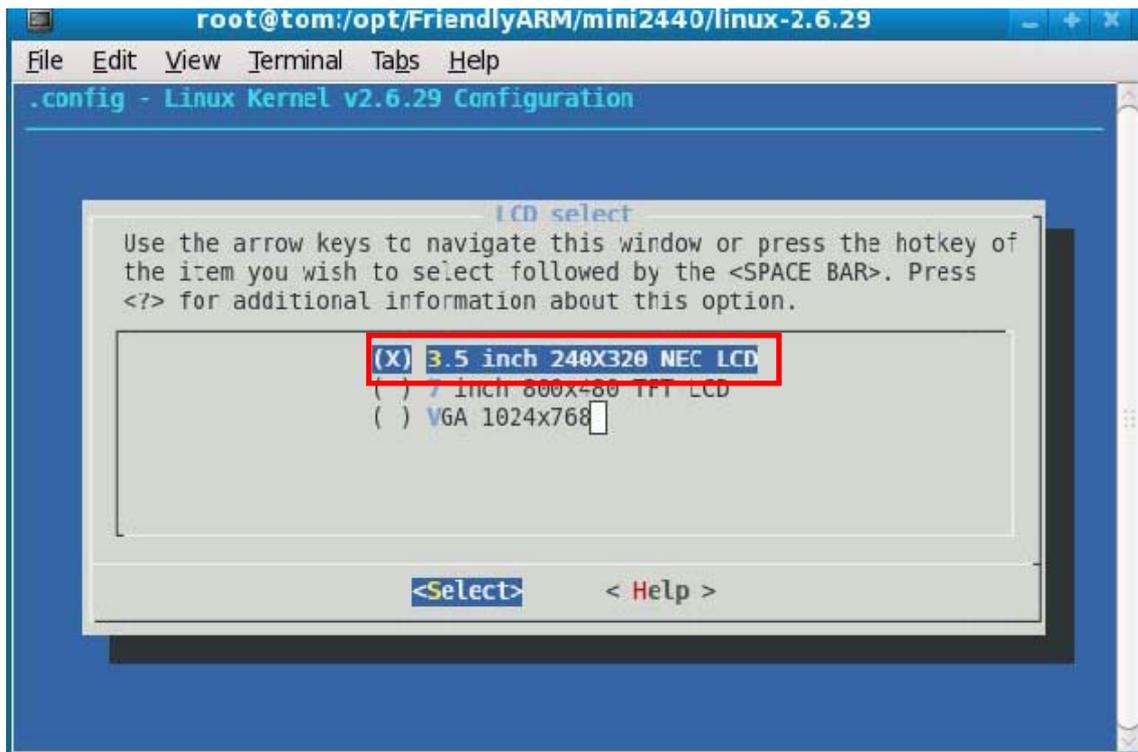
「Enter」キーで「Graphics support」に入ります。



「support for frame buffer devices」に入ります。



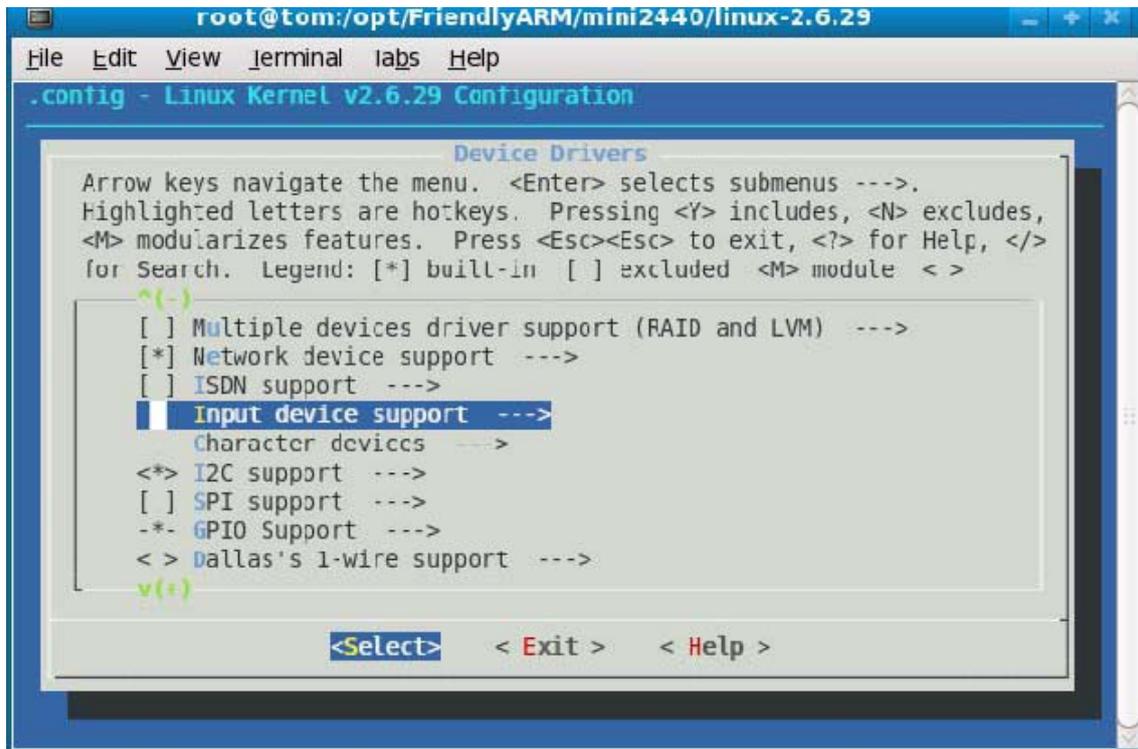
まず、スペースキーで「Backlight support for FriendlyARM board」に<*>を入れます。「LCD select」に入ります。



3.5インチのLCDを選択します。コンフィグ完了したら、「Exit」で「Device Drivers」メニューに戻ります。

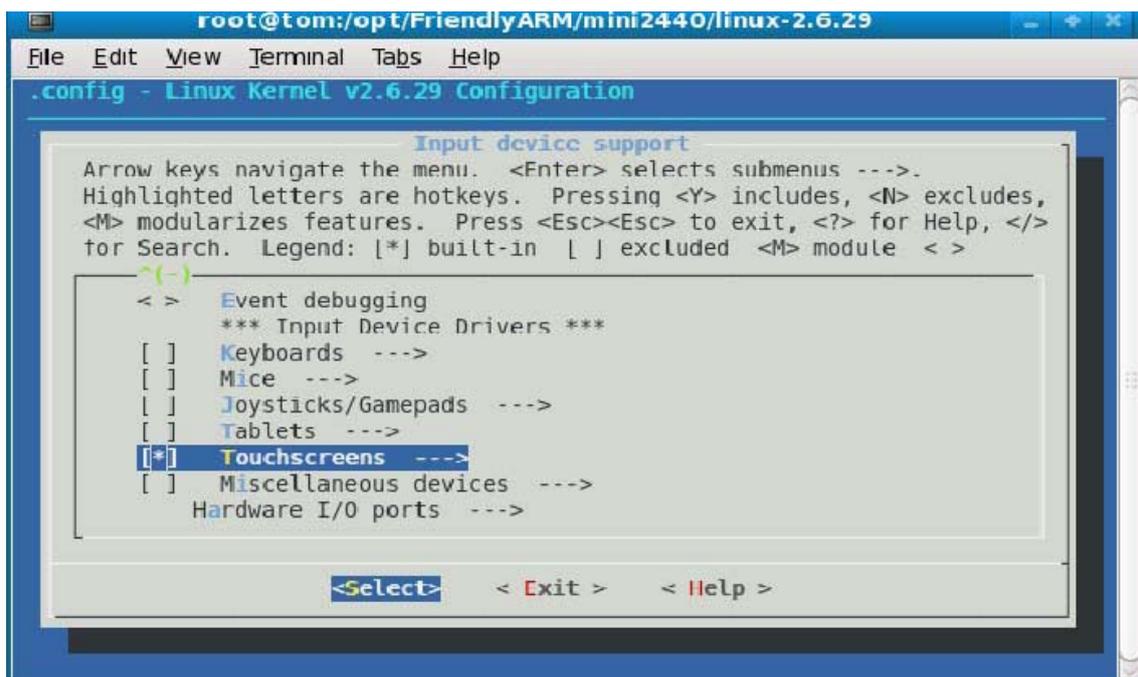
7.4.2 タッチパネル

「Device Drivers」メニューの「Input device support」に入ります。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

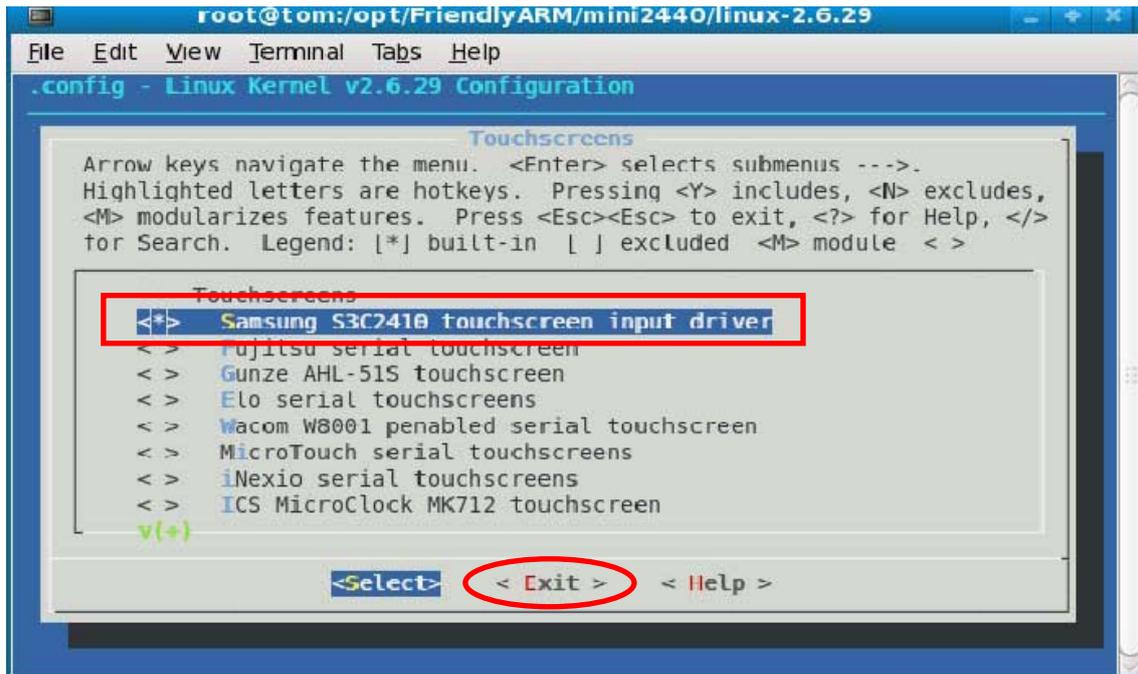
Device Drivers
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
^(-)
[ ] Multiple devices driver support (RAID and LVM) --->
[*] Network device support --->
[ ] ISDN support --->
| Input device support --->
| Character devices --->
<*> I2C support --->
[ ] SPI support --->
-* GPIO Support --->
< > Dallas's 1-wire support --->
v(+)
<Select> < Exit > < Help >
```



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Input device support
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 < >
^(-)
< > Event debugging
*** Input Device Drivers ***
[ ] Keyboards --->
[ ] Mice --->
[ ] Joysticks/Gamepads --->
[ ] Tablets --->
| [*] Touchscreens --->
| Miscellaneous devices --->
Hardware I/O ports --->
<Select> < Exit > < Help >
```

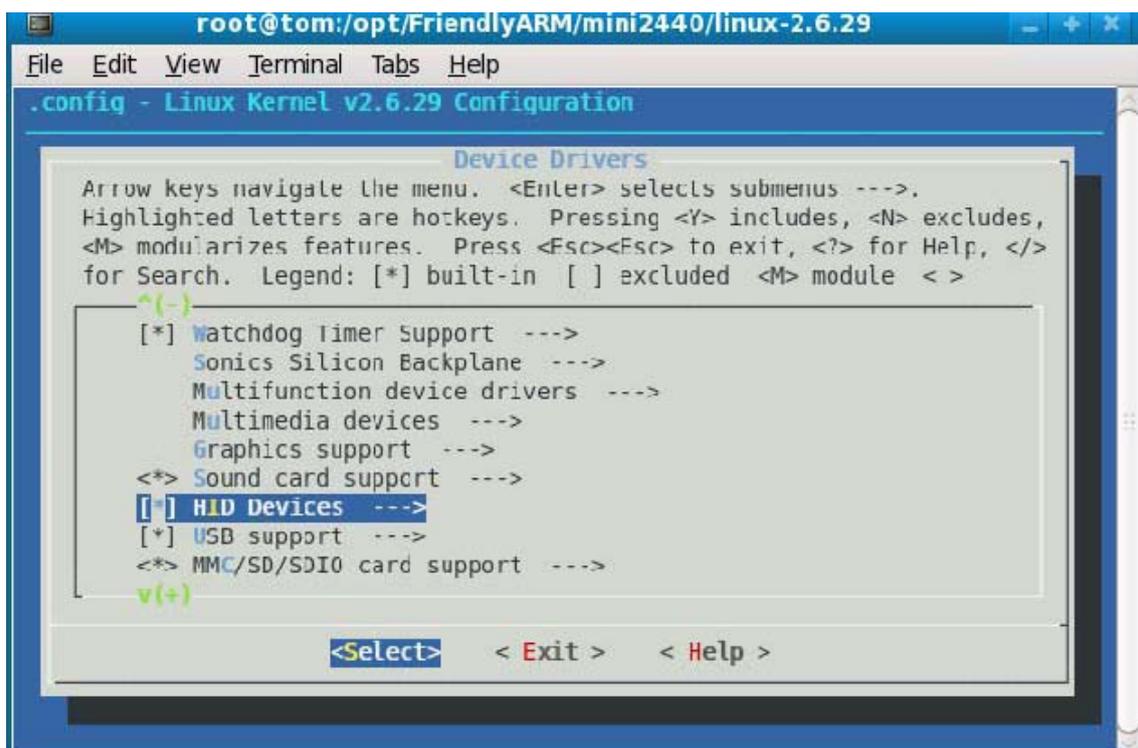
「Touchscreens」に入ります。

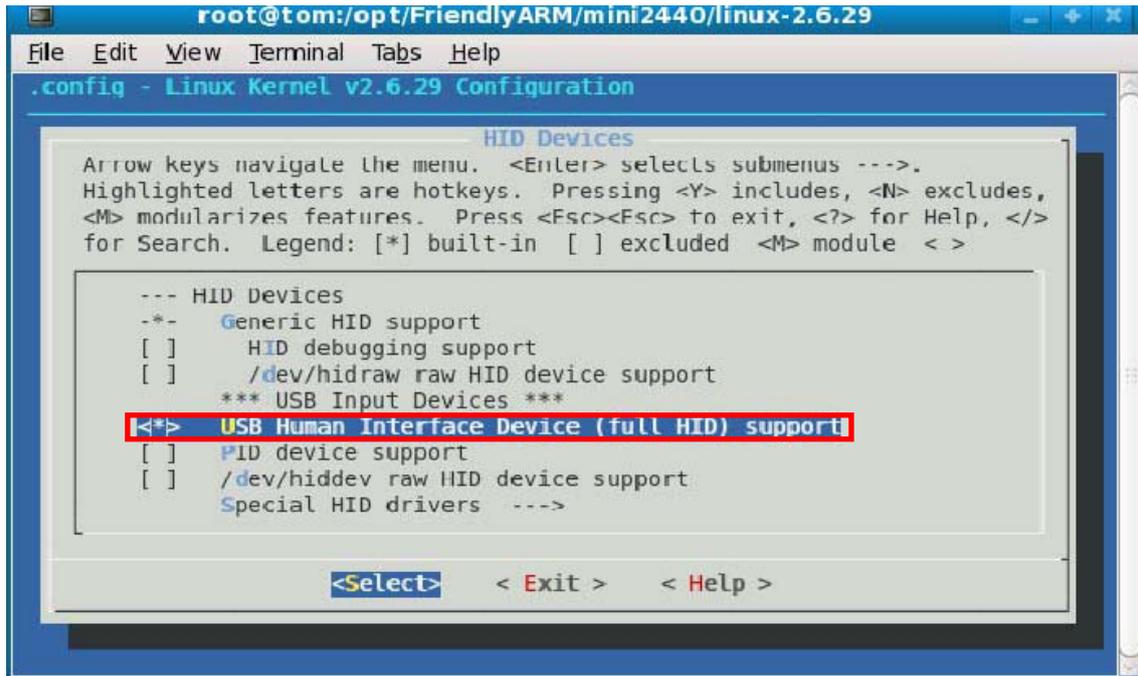


画面のように選択します。「Exit」で「Device Drivers」メニューに戻ります。

7.4.3 USB マウスとキーボード

「Device Drivers」メニューの「HID Devices」に入ります。

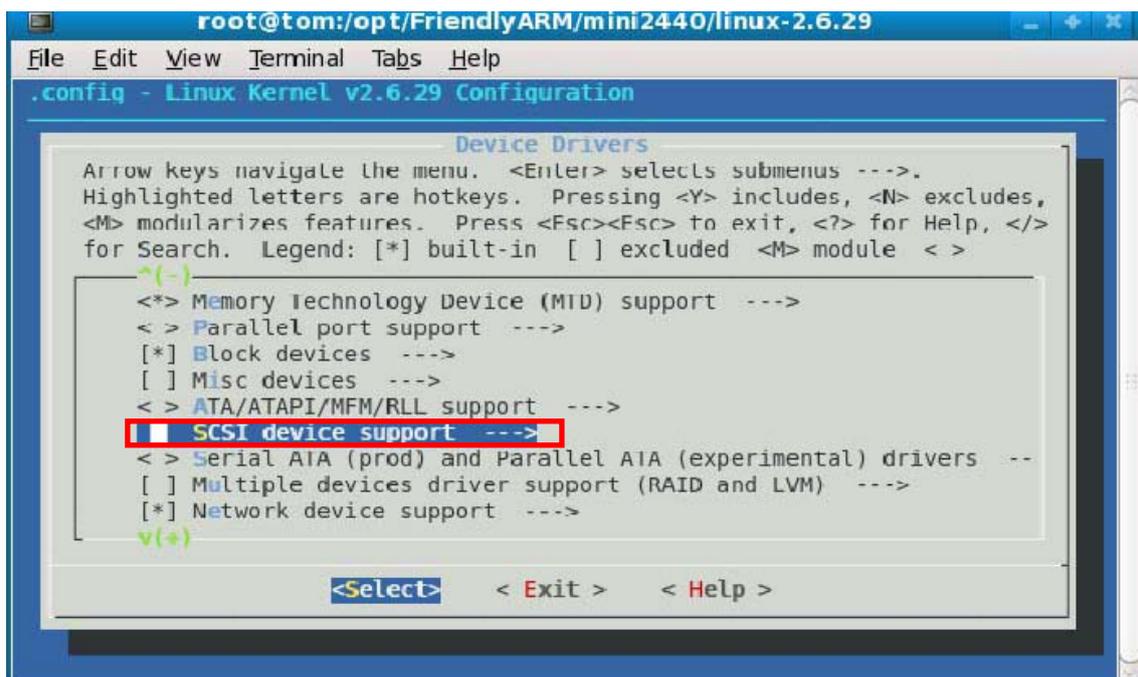


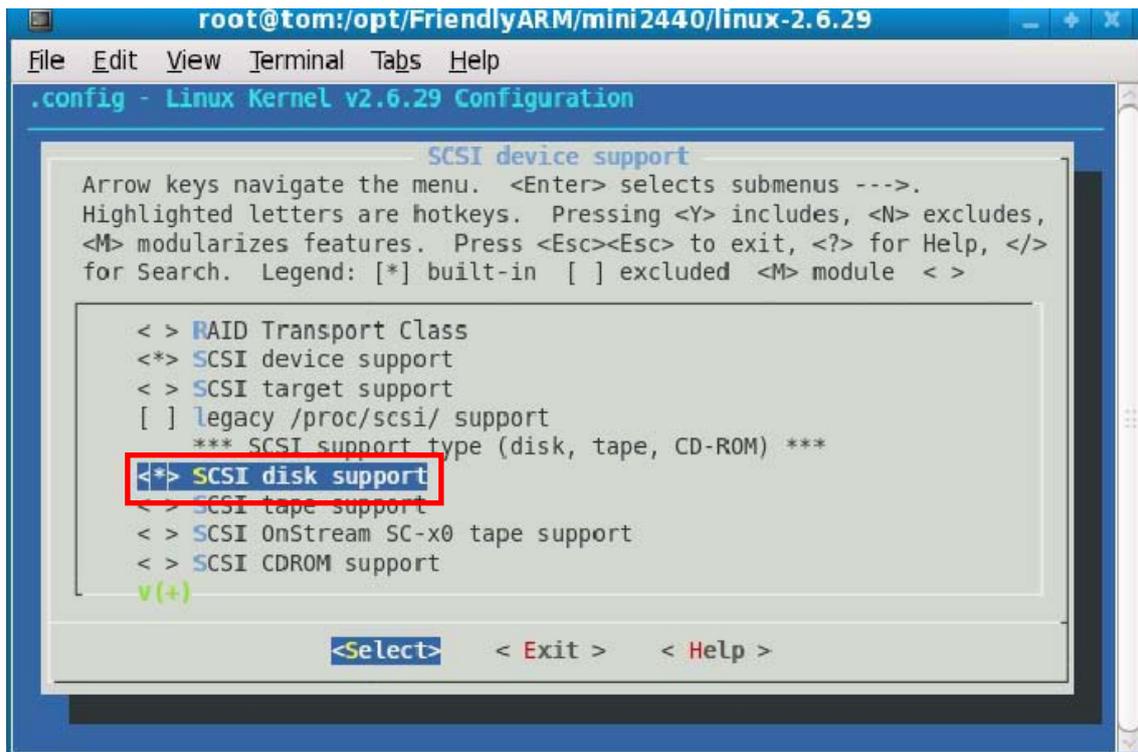


画面のように選択して、「Exit」で「Device Drivers」メニューに戻ります。

7.4.4 USB メモリ

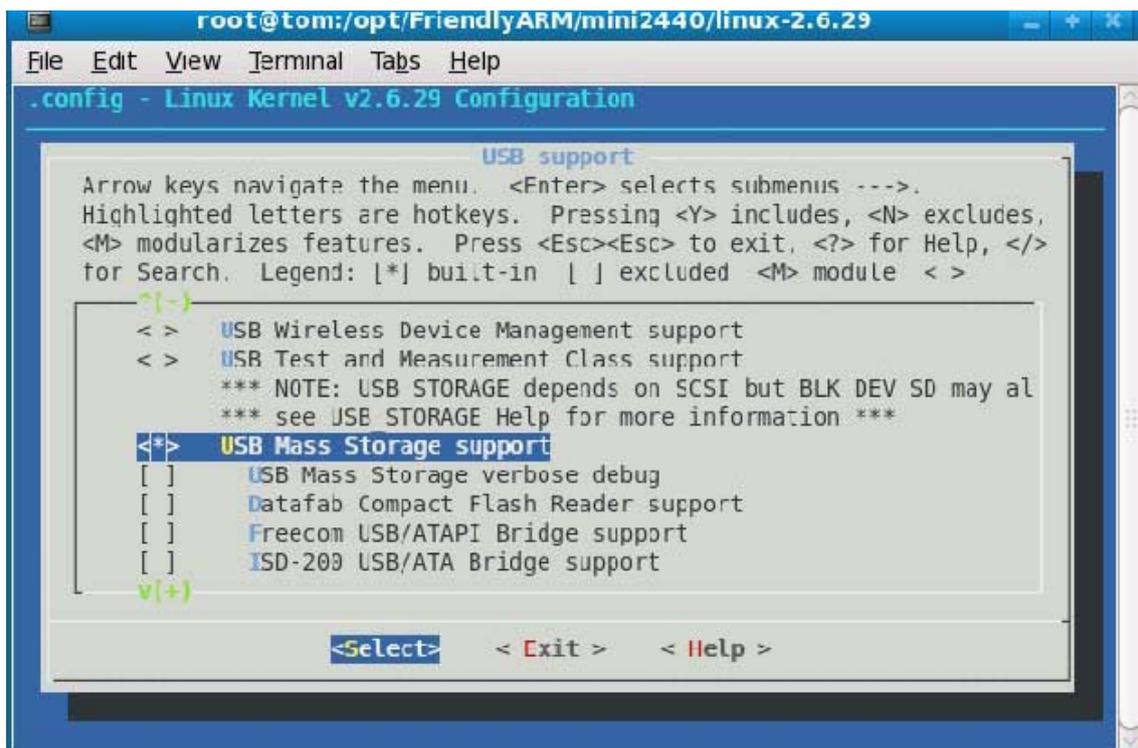
「Device Drivers」メニューの「SCSI device support」に入ります。





画面のように選択して、「Exit」で「Device Drivers」メニューに戻ります。

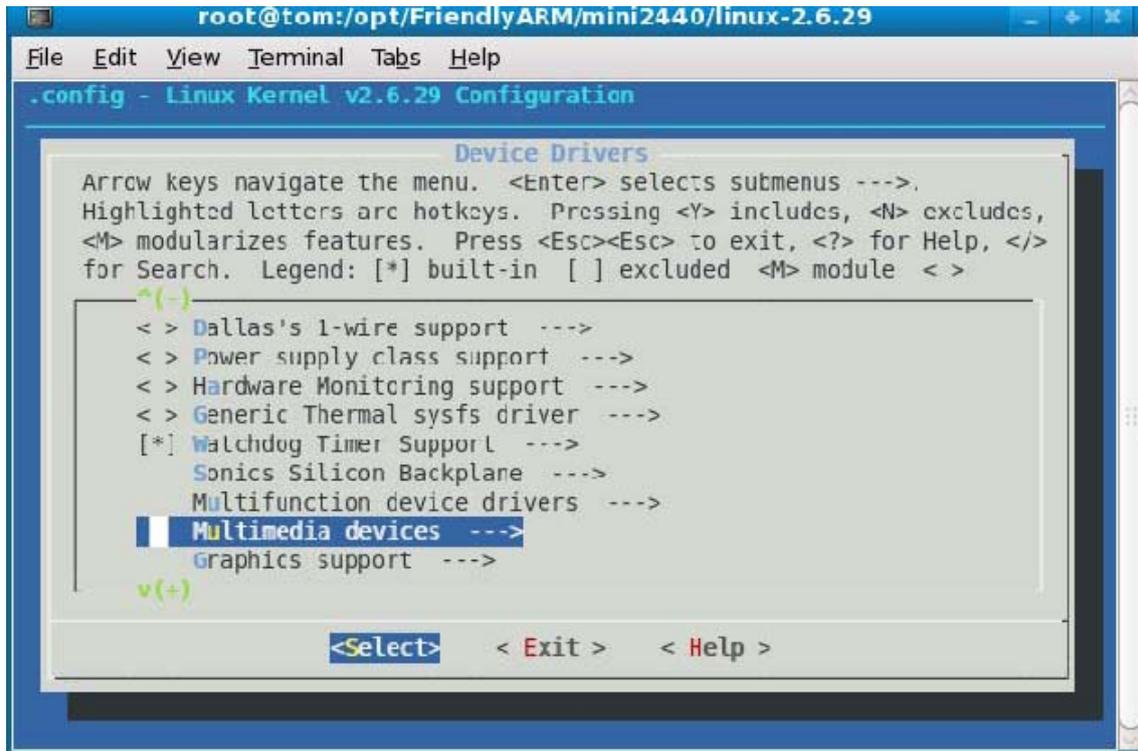
「Device Drivers」メニューの「USB support」に入ります。



「USB Mass Storage support」を選択して、「Exit」で「Device Drivers」メニューに戻ります。

7.4.5 汎用 USB カメラ

「Device Drivers」メニューの「Multimedia devices」に入ります。



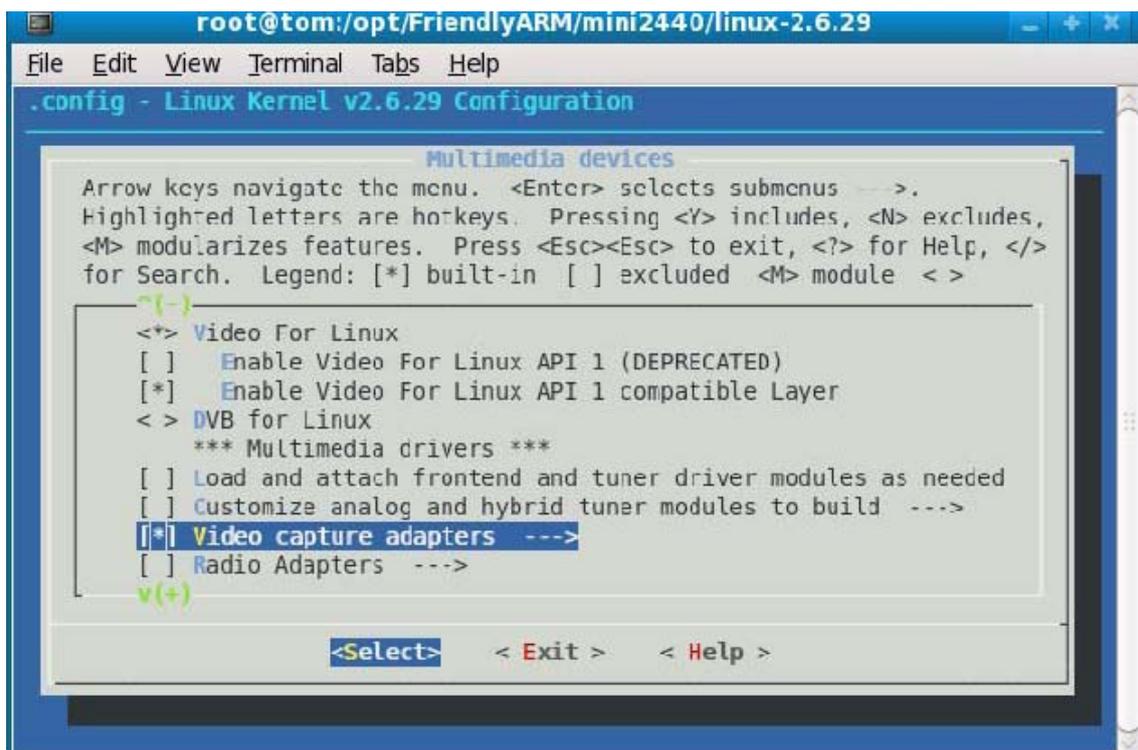
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Device Drivers
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 < >

^(-)
< > Dallas's 1-wire support --->
< > Power supply class support --->
< > Hardware Monitoring support --->
< > Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
    Sonics Silicon Backplane --->
    Multifunction device drivers --->
    Multimedia devices --->
    Graphics support --->
v(+)

<Select> < Exit > < Help >
```

「video capture adapters」に入ります。

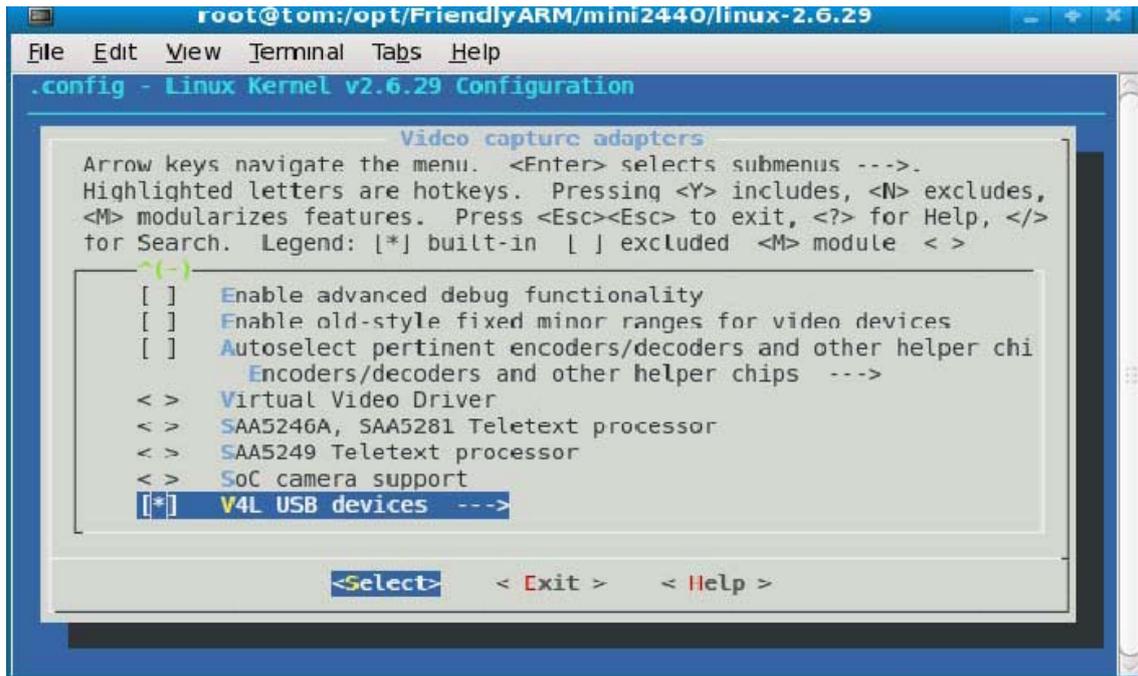


```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

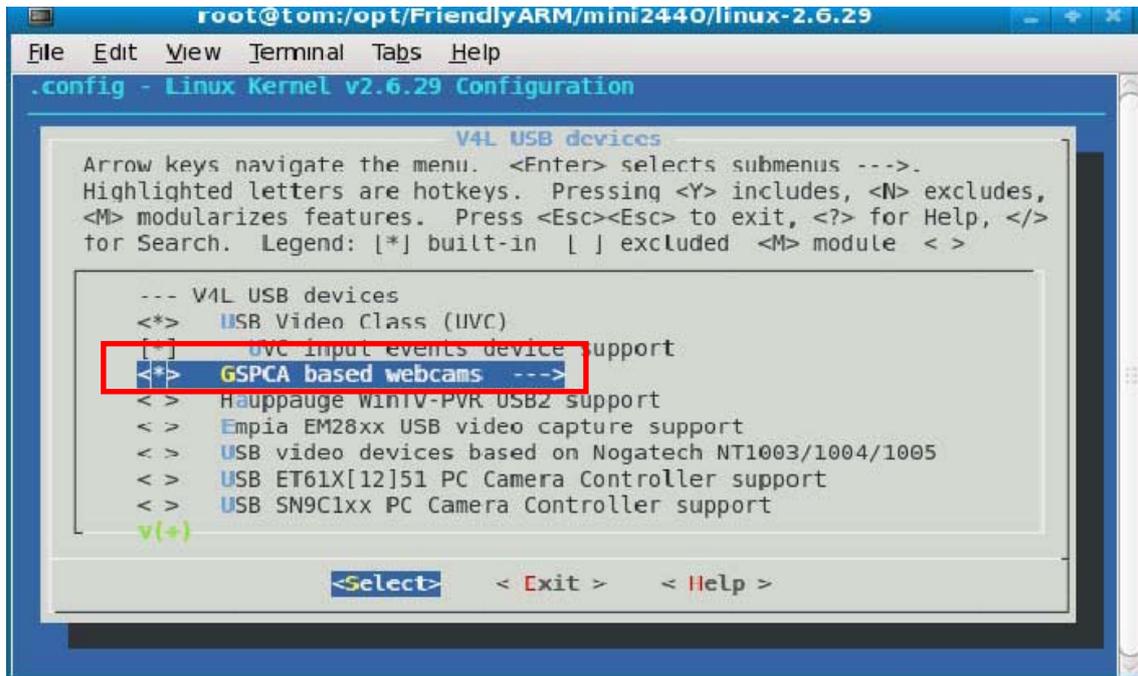
Multimedia devices
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

^(-)
<*> Video For Linux
[ ] Enable Video For Linux API 1 (DEPRECATED)
[*] Enable Video For Linux API 1 compatible Layer
< > DVB for Linux
    *** Multimedia drivers ***
[ ] Load and attach frontend and tuner driver modules as needed
[ ] Customize analog and hybrid tuner modules to build --->
[*] Video capture adapters --->
[ ] Radio Adapters --->
v(+)

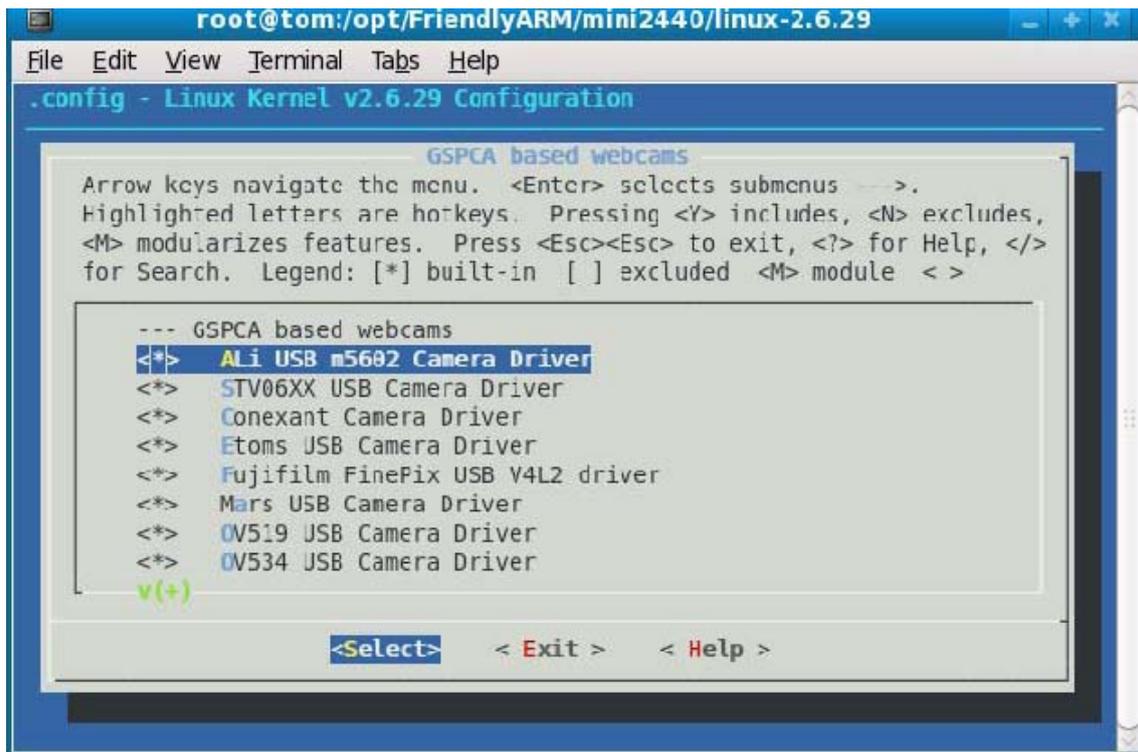
<Select> < Exit > < Help >
```



「V4L USB devices」に入ります。



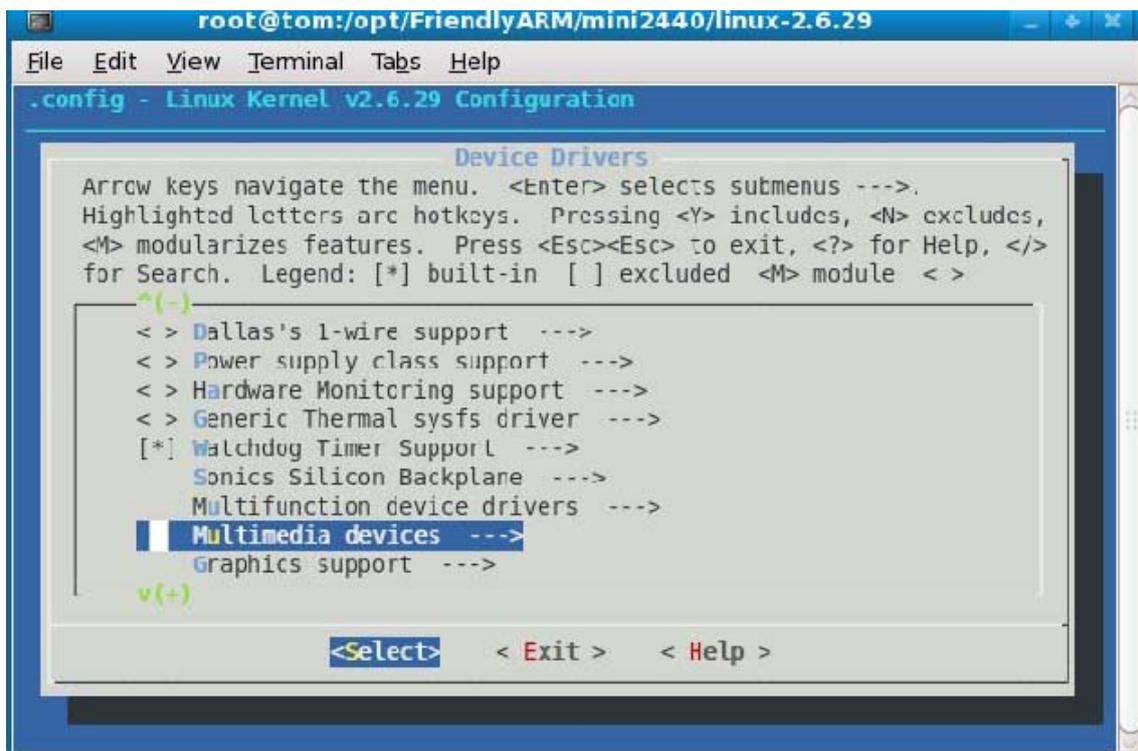
「GSPCA based webcams」に入ります。GSPCAはあるフランス人によって作られた汎用USBカメラドライバです。たくさん種類のUSBカメラをサポートしますが、USBカメラは微妙な差がありますので、アプリケーションは区別処理しなければなりません。

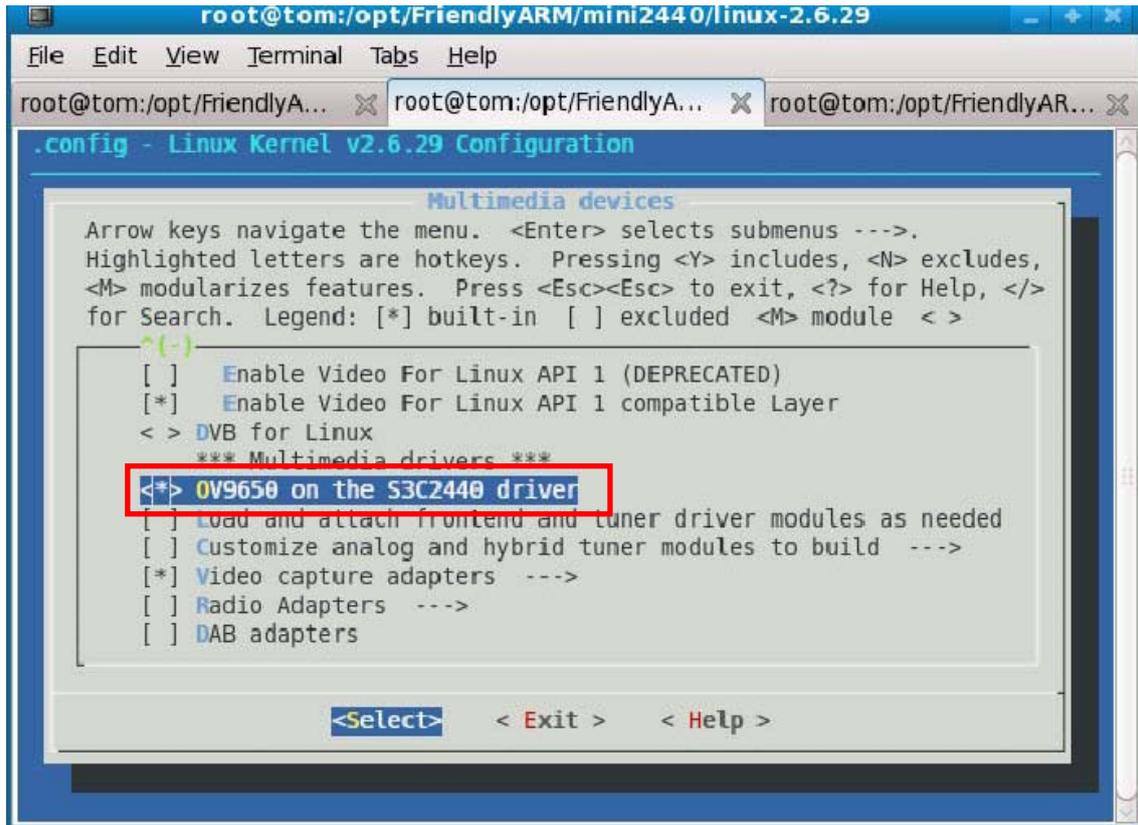


すべてのUSBカメラを選択して、「Exit」で「Device Drivers」メニューに戻ります。

7.4.6 CMOS イメージセンサー(OV9650)

「Device Drivers」メニューの「Multimedia devices」に入ります。



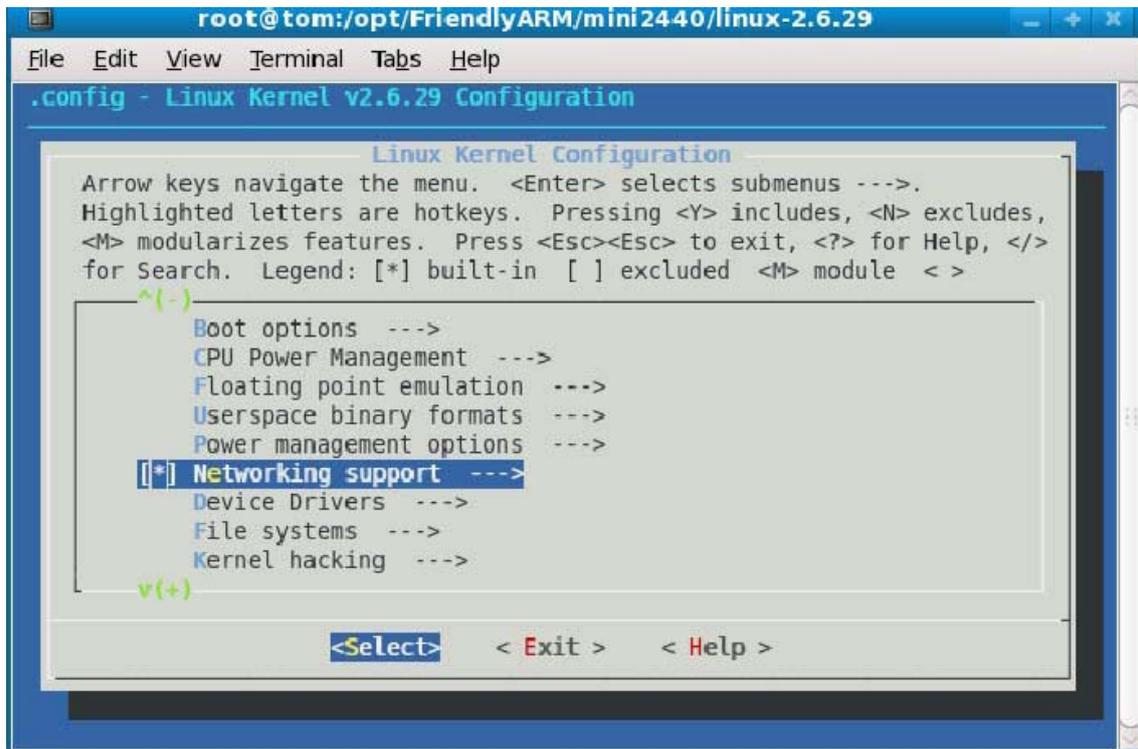


「OV9650 on the S3C2440 driver」を選択して、「Exit」で「Device Drivers」メニューに戻ります。

※ このドライバはV4L/V4L2ドライバではありません。普通のキャラクタ・ドライバです。

7.4.7 イーサネット

「Device Drivers」メニューの「Networking support」に入ります。

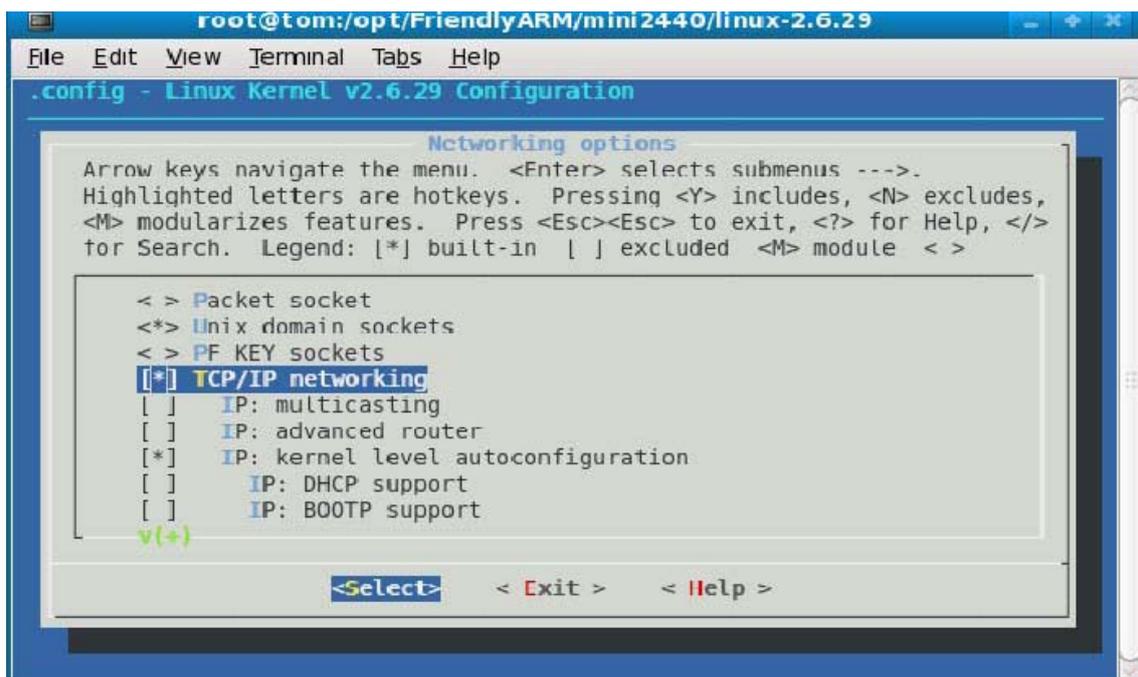


```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Linux Kernel Configuration
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 < >

~(-)
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
  [*] Networking support --->
  Device Drivers --->
  File systems --->
  Kernel hacking --->

v(+)
  <Select> < Exit > < Help >
```



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

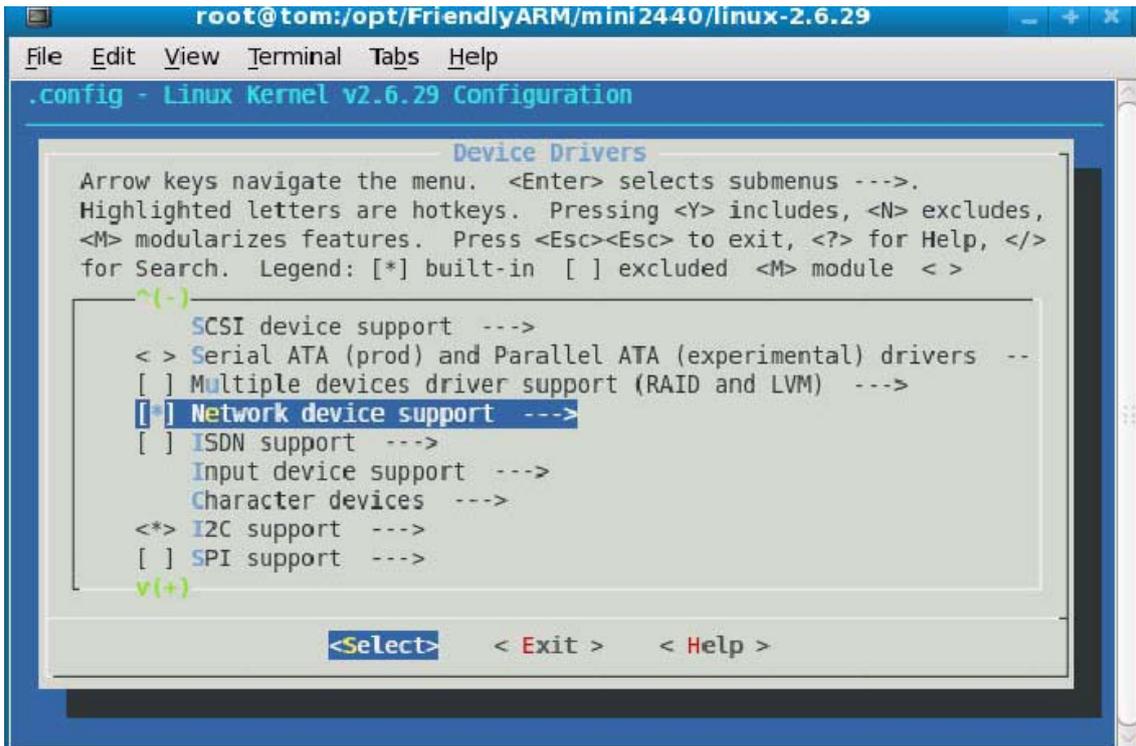
Networking options
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

< > Packet socket
< * > Unix domain sockets
< > PF KEY sockets
[*] TCP/IP networking
  [ ] IP: multicasting
  [ ] IP: advanced router
  [*] IP: kernel level autoconfiguration
  [ ] IP: DHCP support
  [ ] IP: BOOTP support

v(+)
  <Select> < Exit > < Help >
```

TCP/IPプロトコルを選択して、「Exit」で「Device Drivers」メニューに戻ります。

「Device Drivers」メニューの「Network device support」に入ります。

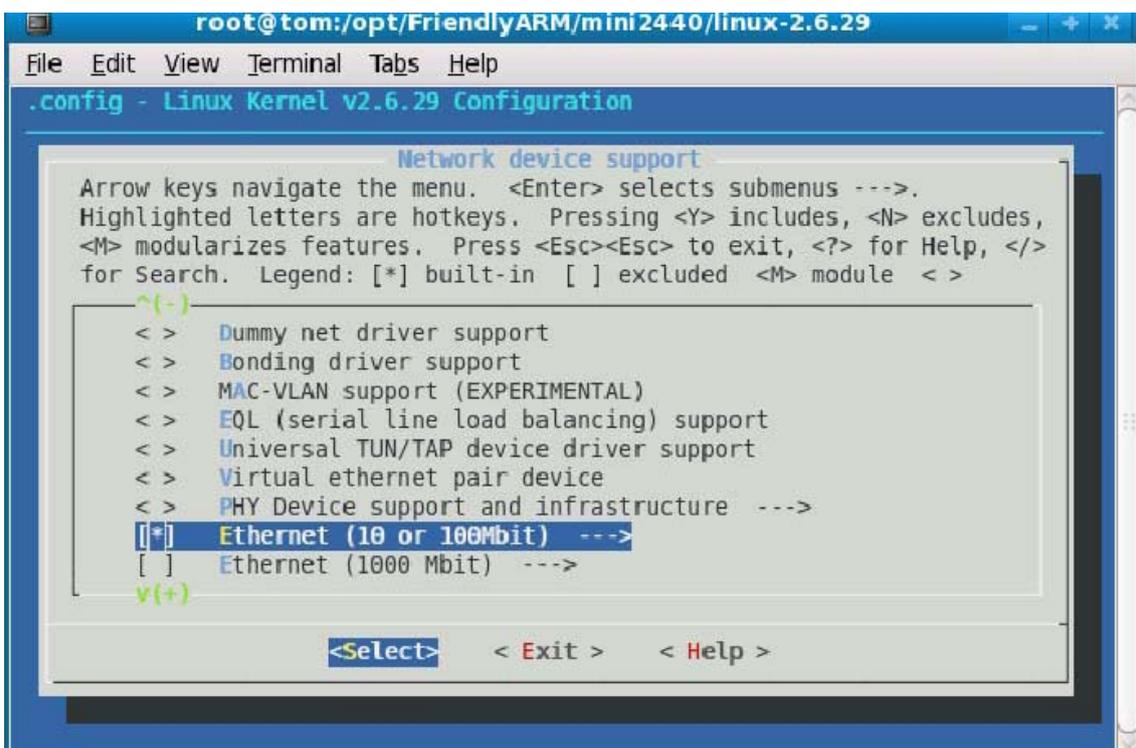


```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Device Drivers
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 <>

^(-)
  <> SCSI device support --->
  <> Serial ATA (prod) and Parallel ATA (experimental) drivers --
  [ ] Multiple devices driver support (RAID and LVM) --->
  [*] Network device support --->
  [ ] ISDN support --->
  Input device support --->
  Character devices --->
  <*> I2C support --->
  [ ] SPI support --->
v(+)

<Select> < Exit > < Help >
```



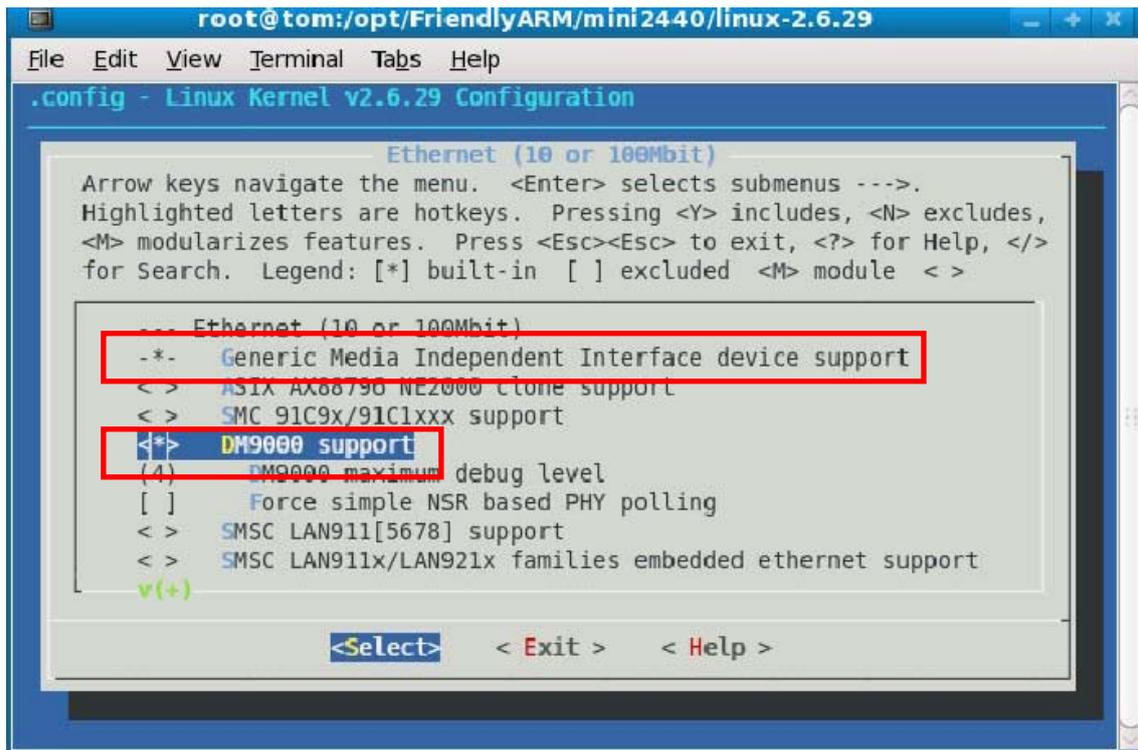
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Network device support
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 <>

^(-)
  <> Dummy net driver support
  <> Bonding driver support
  <> MAC-VLAN support (EXPERIMENTAL)
  <> EQL (serial line load balancing) support
  <> Universal TUN/TAP device driver support
  <> Virtual ethernet pair device
  <> PHY Device support and infrastructure --->
  [*] Ethernet (10 or 100Mbit) --->
  [ ] Ethernet (1000 Mbit) --->
v(+)

<Select> < Exit > < Help >
```

「Ethernet(10 or 100Mbit)」に入ります。



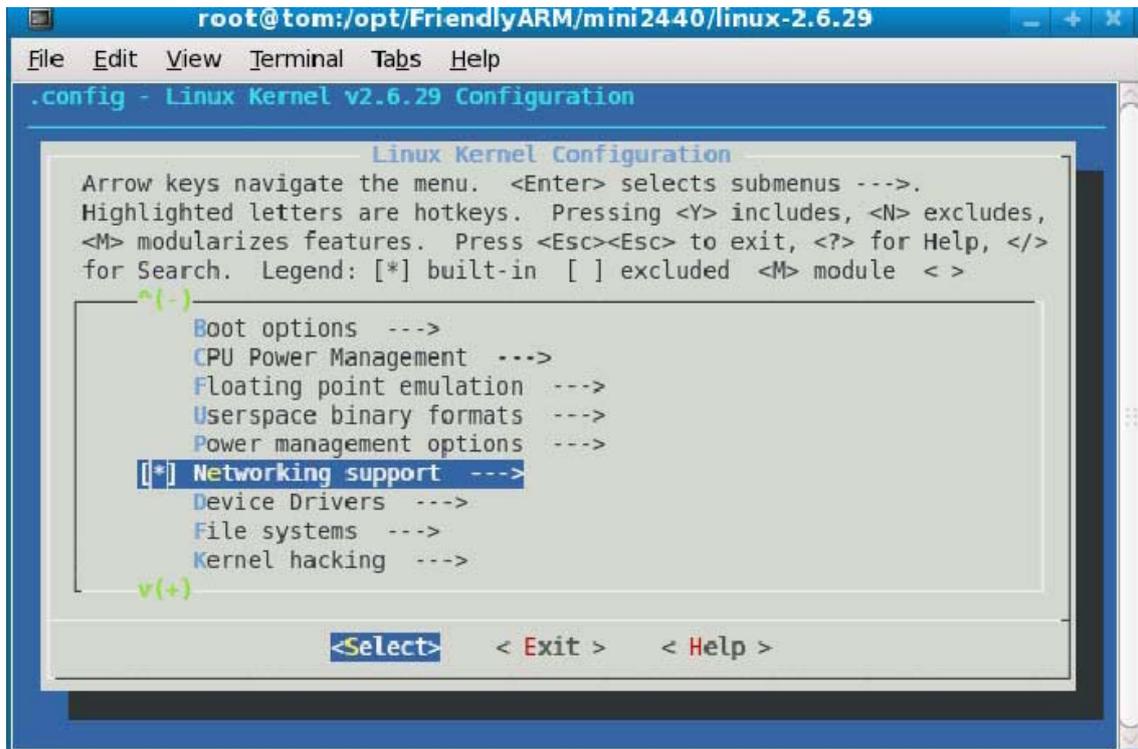
< * > Generic Media Independent Interface device support

< * > DM9000 support

「Exit」で「Device Drivers」メニューに戻ります。

7.4.8 USB 無線 LAN

「Device Drivers」メニューの「Networking support」に入ります。



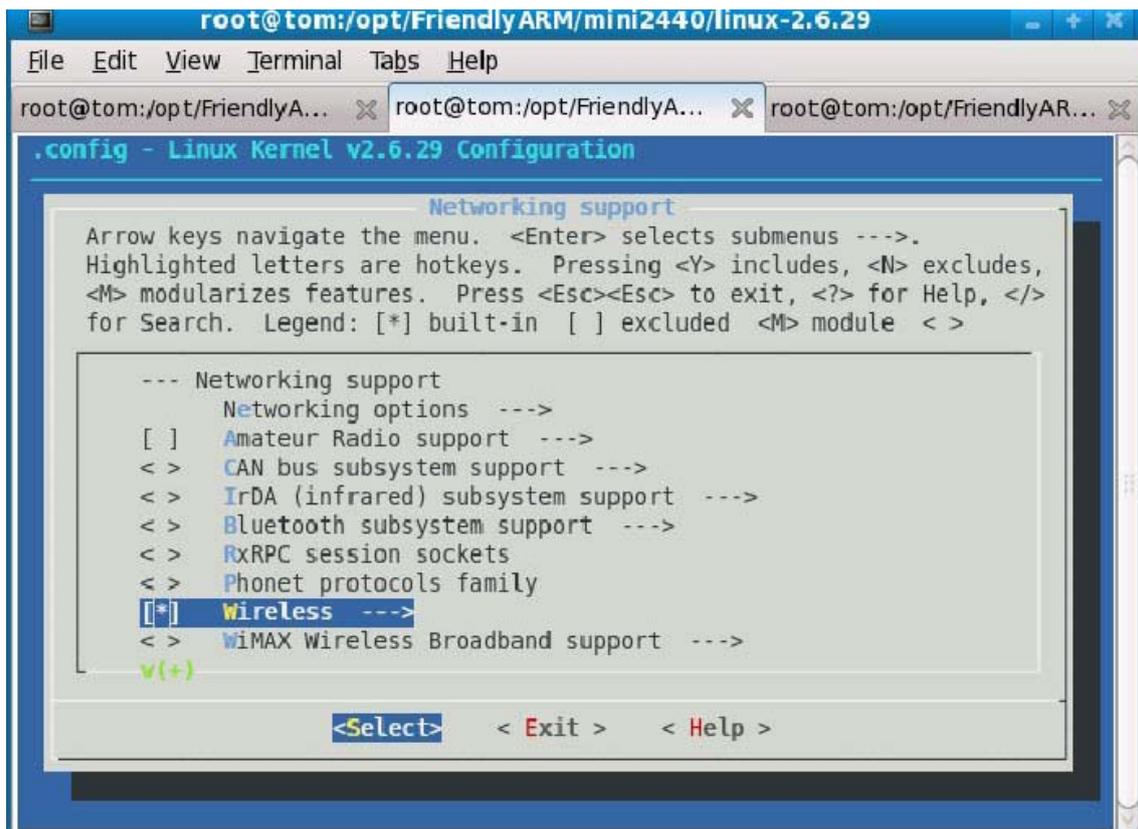
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Linux Kernel Configuration
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 < >

^(-)
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
  [*] Networking support --->
  Device Drivers --->
  File systems --->
  Kernel hacking --->

v(+)

<Select> < Exit > < Help >
```



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
root@tom:/opt/FriendlyA... x root@tom:/opt/FriendlyA... x root@tom:/opt/FriendlyAR... x
.config - Linux Kernel v2.6.29 Configuration

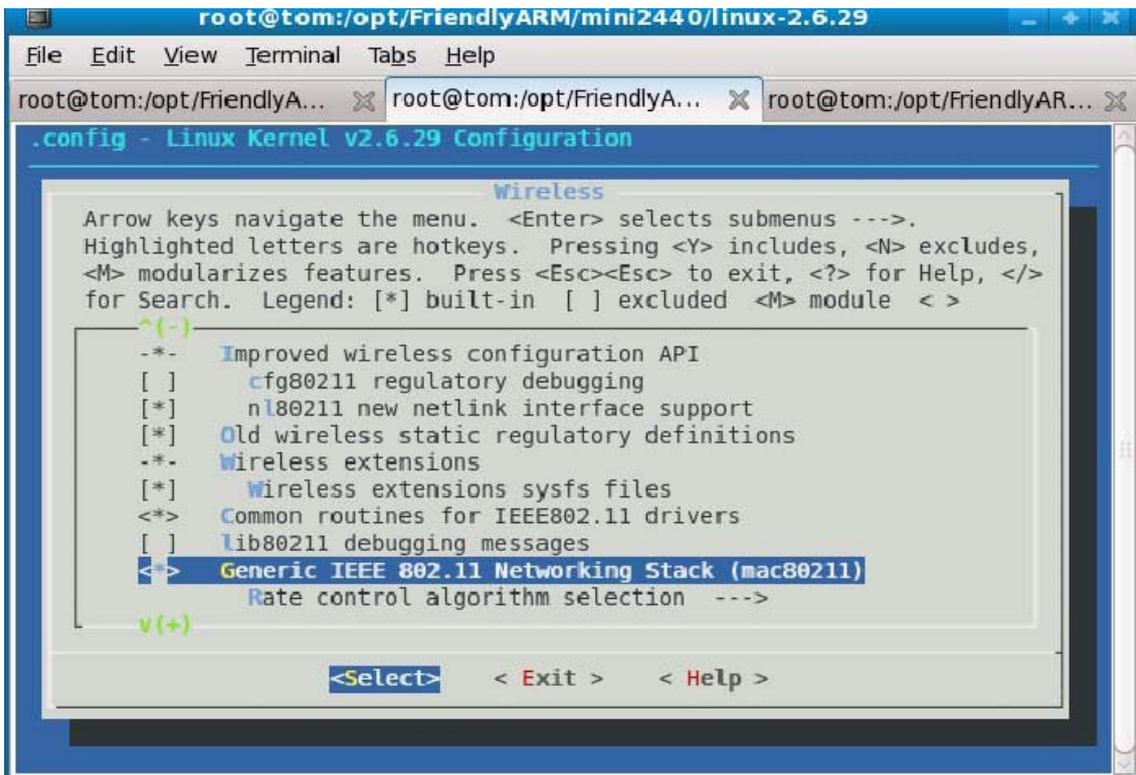
Networking support
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

--- Networking support
  Networking options --->
  [ ] Amateur Radio support --->
  < > CAN bus subsystem support --->
  < > IrDA (infrared) subsystem support --->
  < > Bluetooth subsystem support --->
  < > RxRPC session sockets
  < > Phonet protocols family
  [*] Wireless --->
  < > WiMAX Wireless Broadband support --->

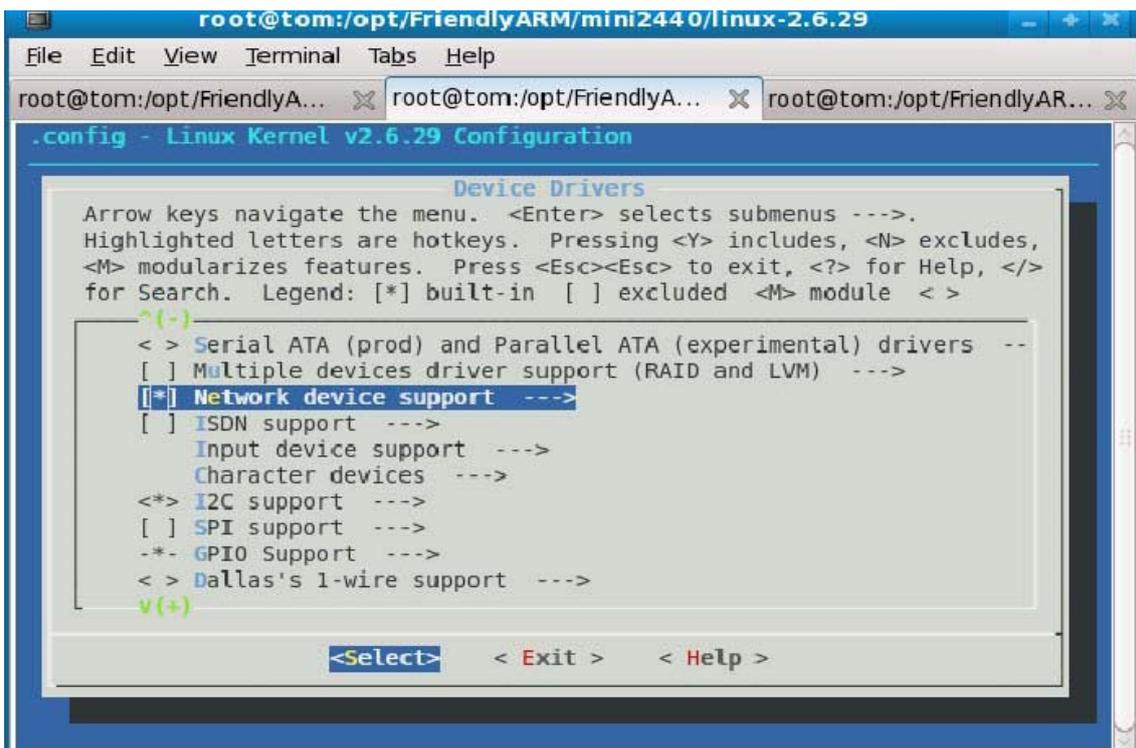
v(+)

<Select> < Exit > < Help >
```

「wireless」に入ります。



画面のように「*」を選択して、「Exit」で「Device Drivers」メニューに戻ります。



「Device Drivers」メニューの「Network device support」に入ります。

```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
root@tom:/opt/FriendlyA... root@tom:/opt/FriendlyA... root@tom:/opt/FriendlyAR...
.config - Linux Kernel v2.6.29 Configuration

Network device support
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 <>

^(-)
<> PHY Device support and infrastructure --->
[*] Ethernet (10 or 100Mbit) --->
[ ] Ethernet (1000 Mbit) --->
[ ] Ethernet (10000 Mbit) --->
| Wireless LAN --->
  *** Enable WiMAX (Networking options) to see the WiMAX driv
  USB Network Adapters --->
[ ] Wan interfaces support --->
<> PPP (point-to-point protocol) support
<> SLIP (serial line) support
v(+)
```

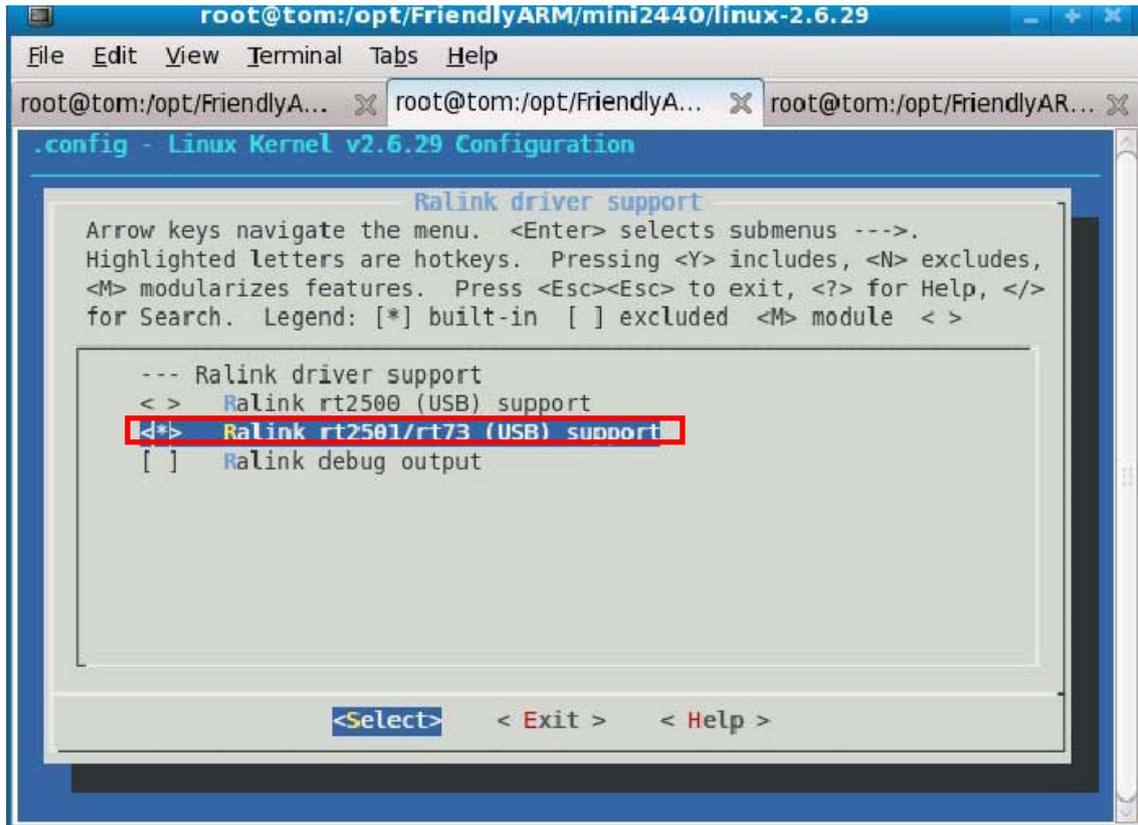
「Wireless LAN」に入ります。

```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
root@tom:/opt/FriendlyA... root@tom:/opt/FriendlyA... root@tom:/opt/FriendlyAR...
.config - Linux Kernel v2.6.29 Configuration

Wireless LAN
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 <>

^(-)
<> USB ZD1201 based Wireless device support
<> Wireless RNDIS USB support
<> Realtek 8187 and 8187B USB support
<> Simulated radio testing tool for mac80211
<> Softmac Prism54 support
<> IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
<> Broadcom 43xx wireless support (mac80211 stack)
<> Broadcom 43xx-legacy wireless support (mac80211 stack)
<> ZydAS ZD1211/ZD1211B USB-wireless support
<M> Ralink driver support --->
```

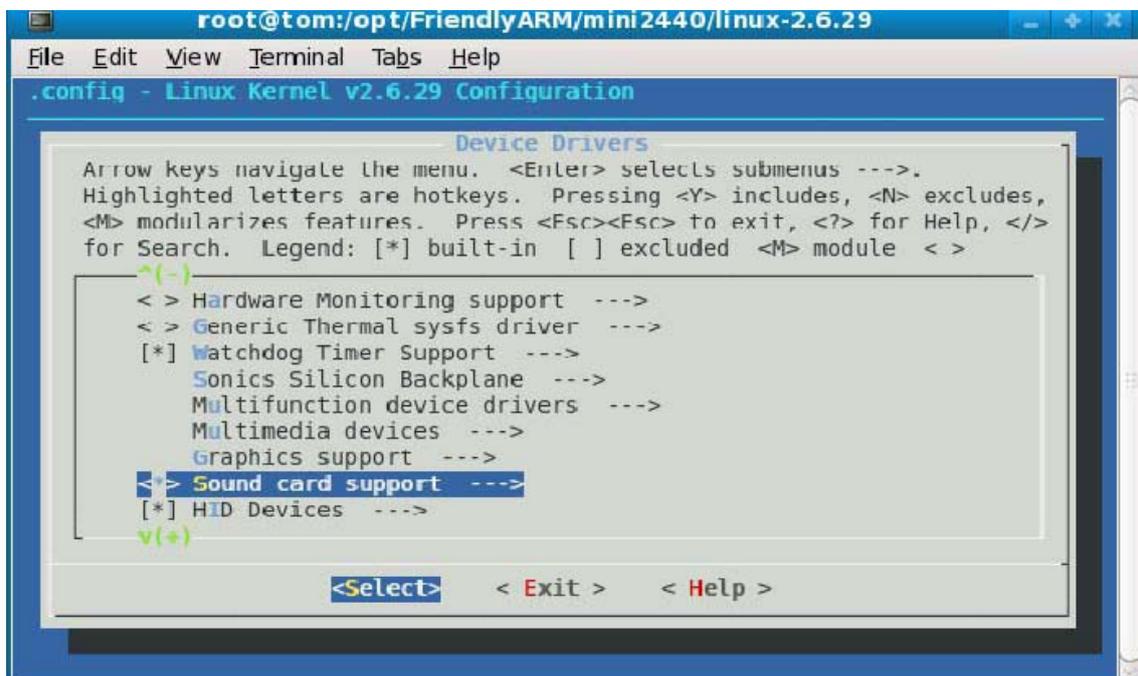
「Ralink driver support」に入ります。

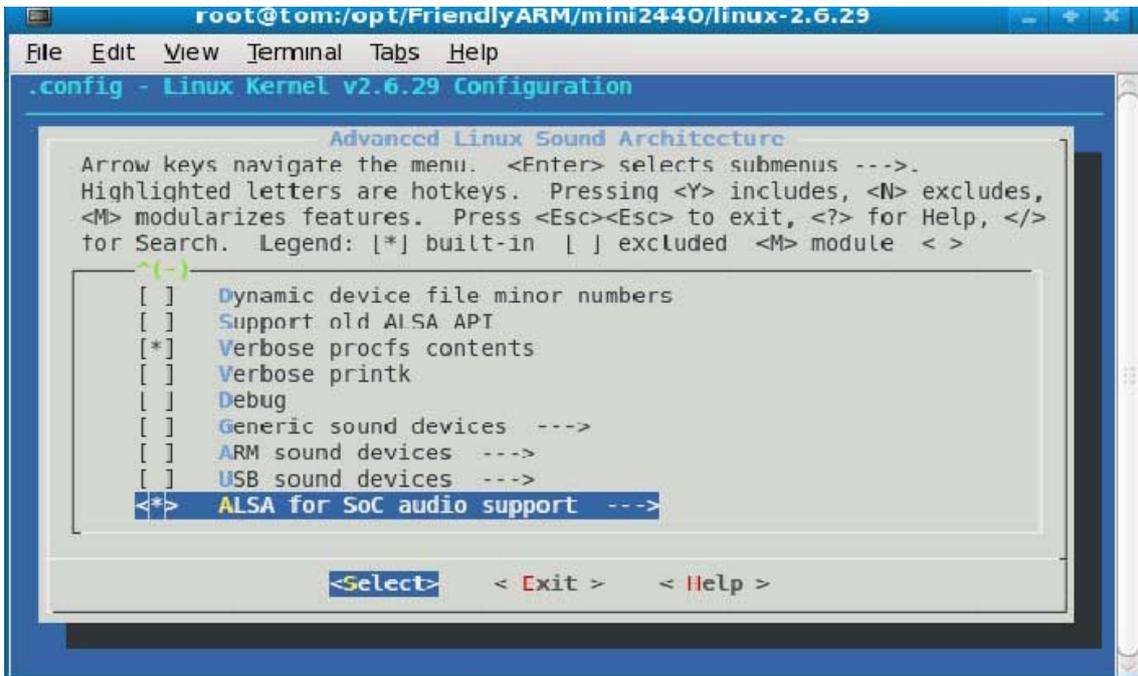
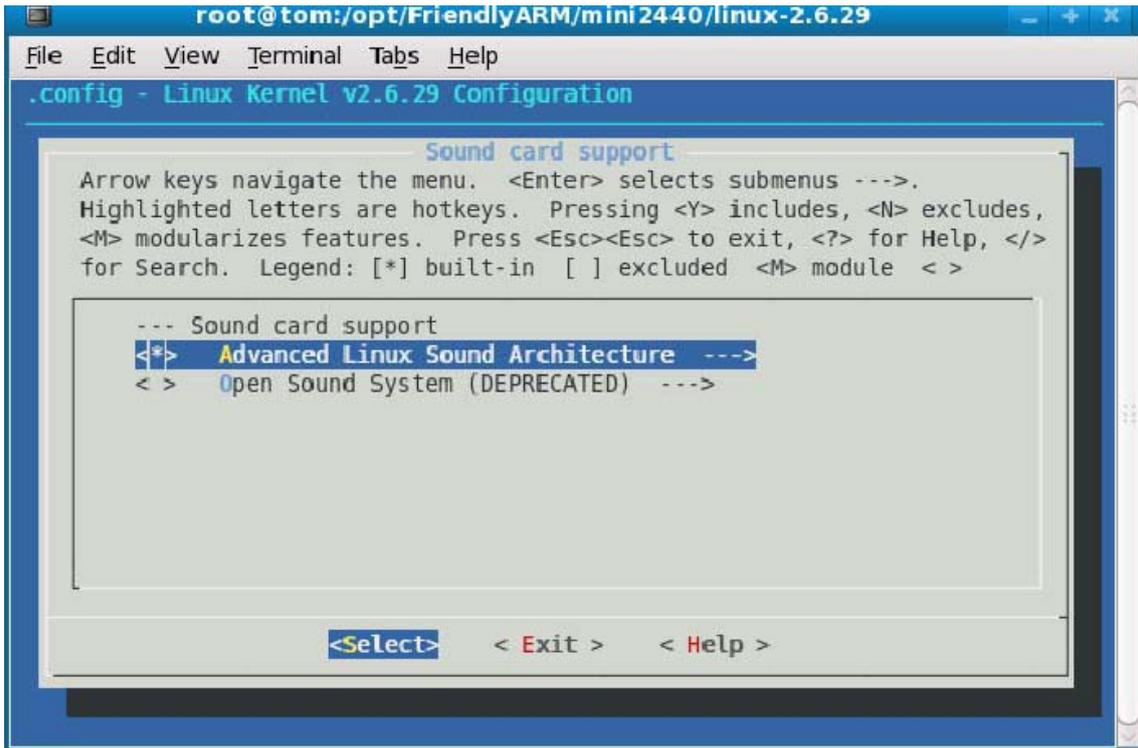


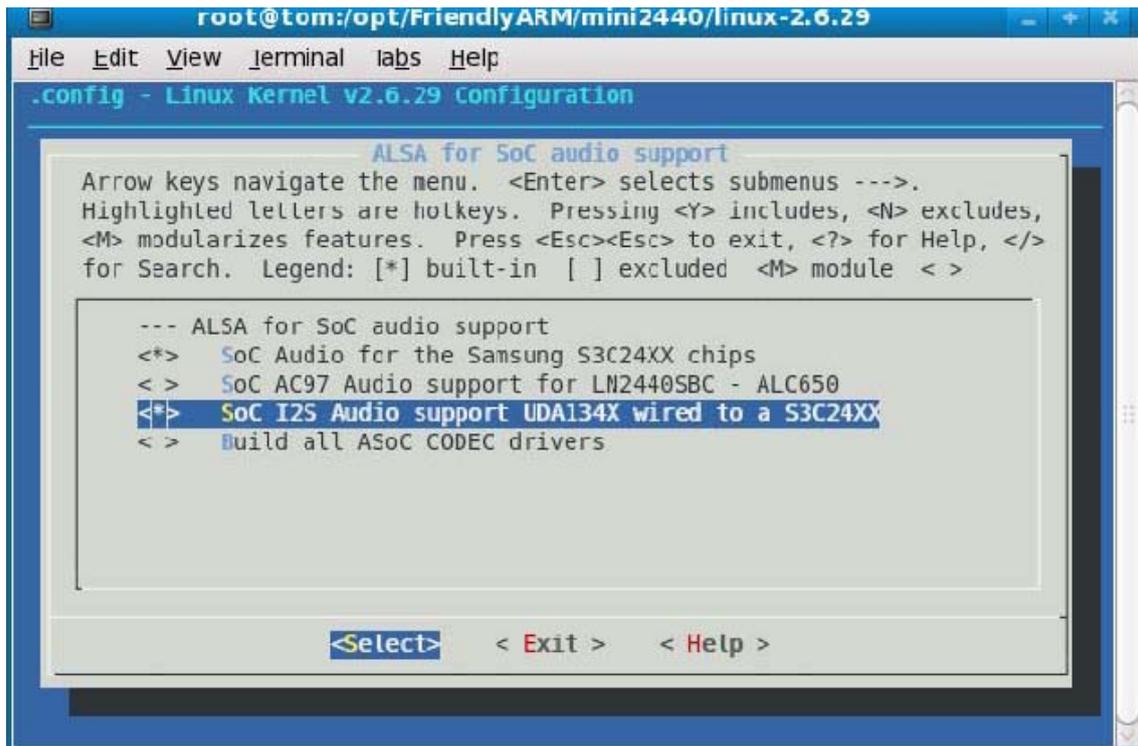
「Exit」で「Device Drivers」メニューに戻ります。

7.4.9 オーディオ

「Device Drivers」メニューの「Sound card support」に入ります。



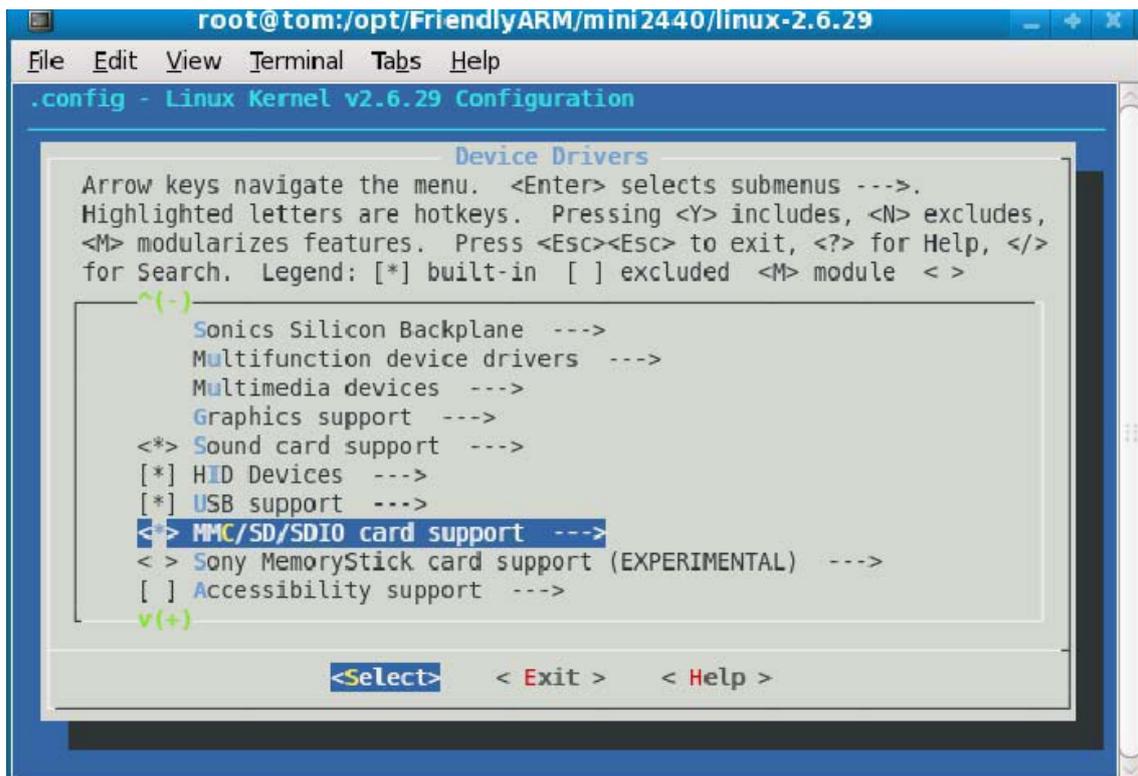


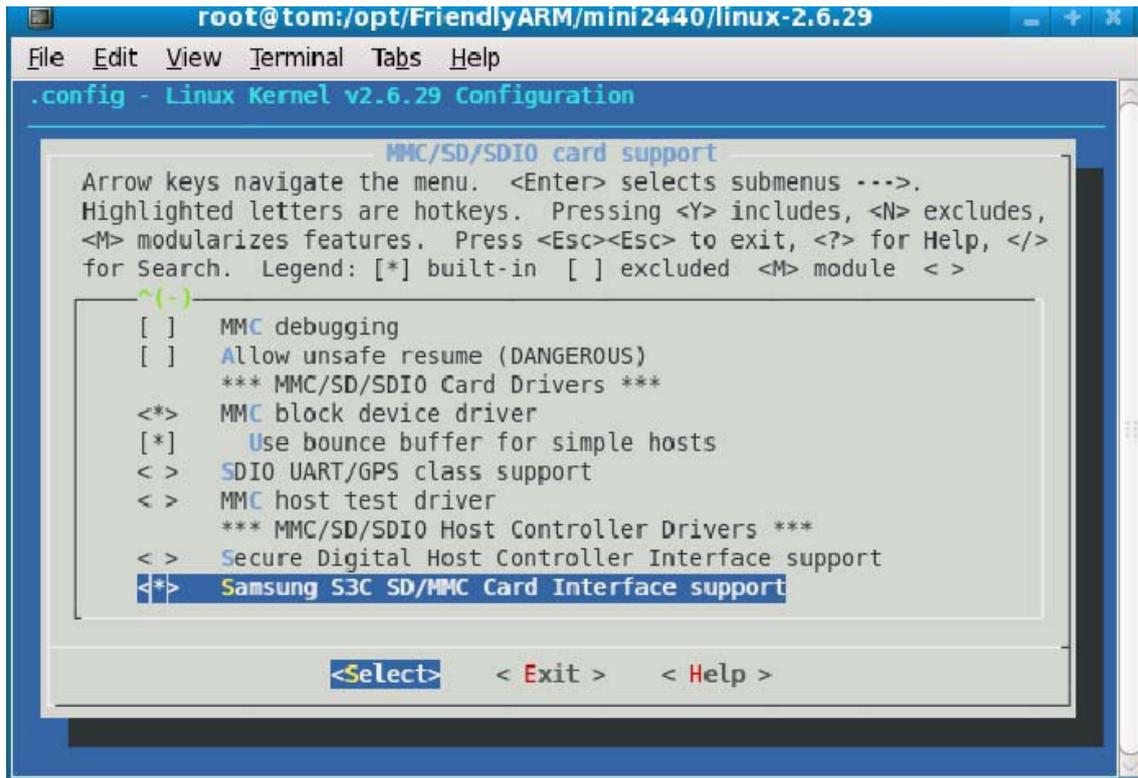


「Exit」で「Device Drivers」メニューに戻ります。

7.4.10 SD/MMC メモリカード

「Device Drivers」メニューの「MMC/SD/SDIO card support」に入ります。

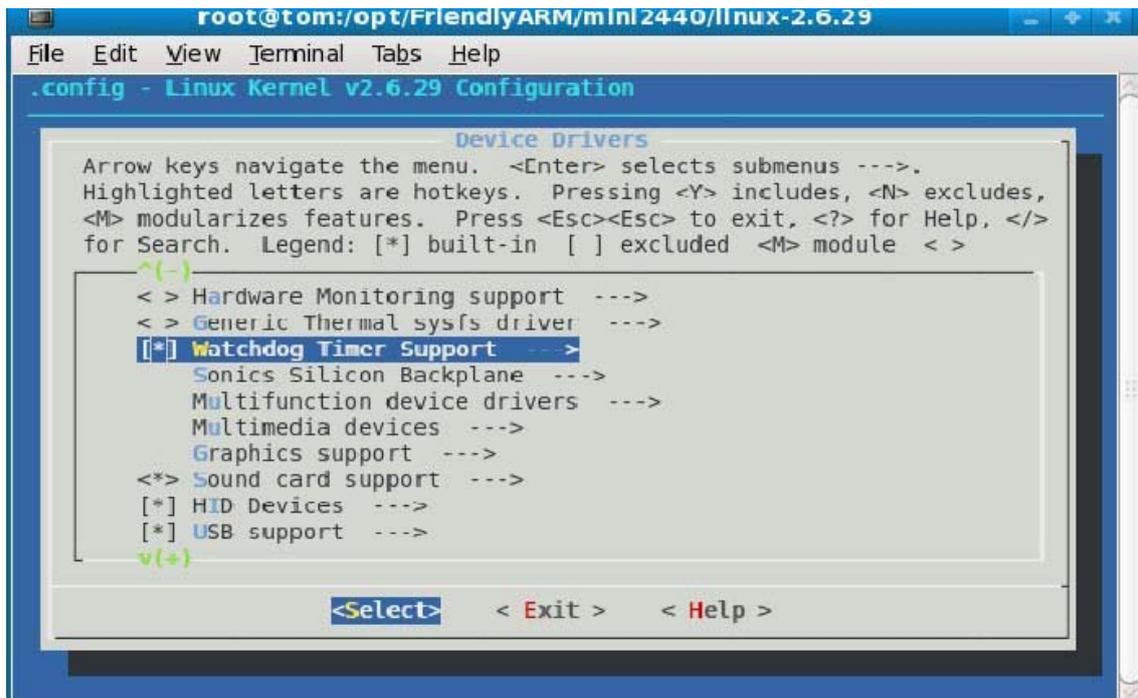


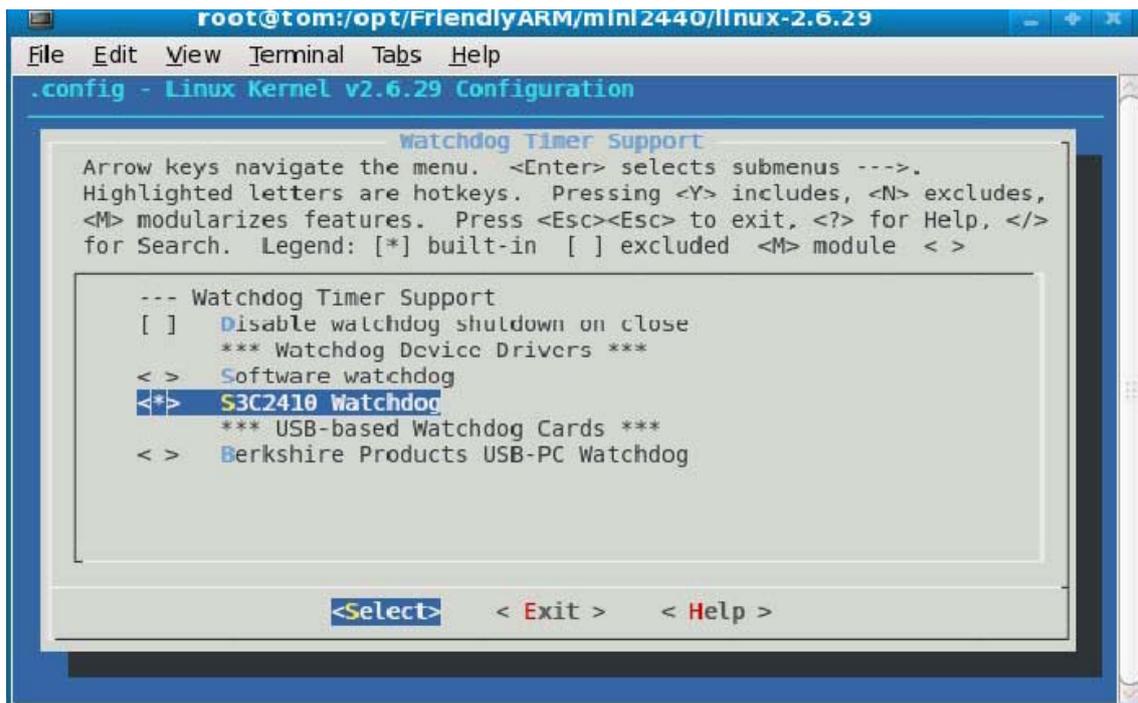


画面のように「*」を選択して、「Exit」で「Device Drivers」メニューに戻ります。

7.4.11 Watchdog

「Device Drivers」メニューの「Watchdog Timer support」に入ります。

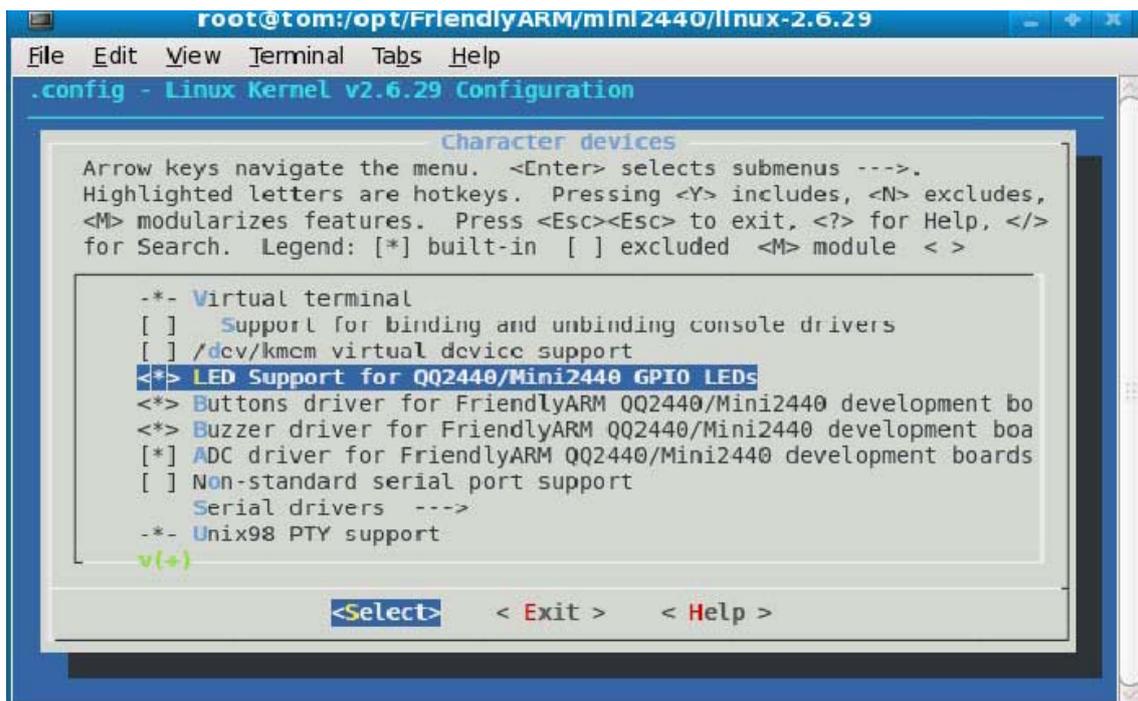




「S3C2410 Watchdog」を選択して、「Exit」で「Device Drivers」メニューに戻ります。

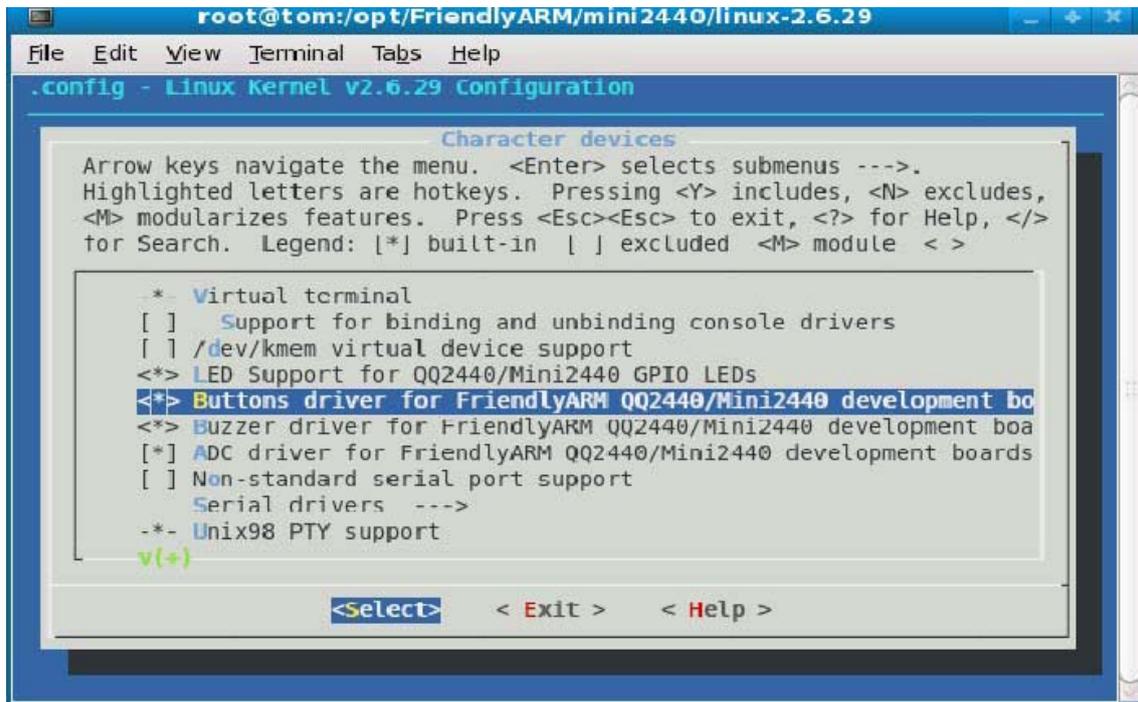
7.4.12 LED

「Device Drivers」メニューの「Character devices - ->」に入ります。



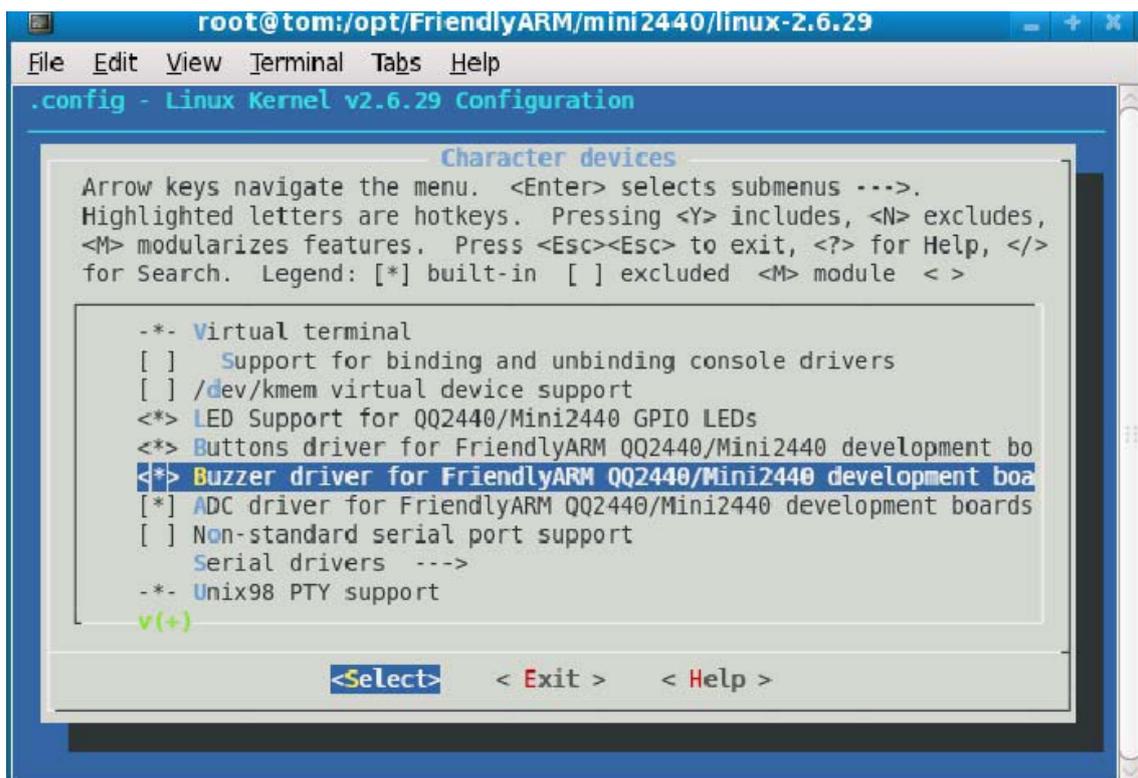
7.4.13 ボタン

「Device Drivers」メニューの「Character devices - - ->」に入ります。



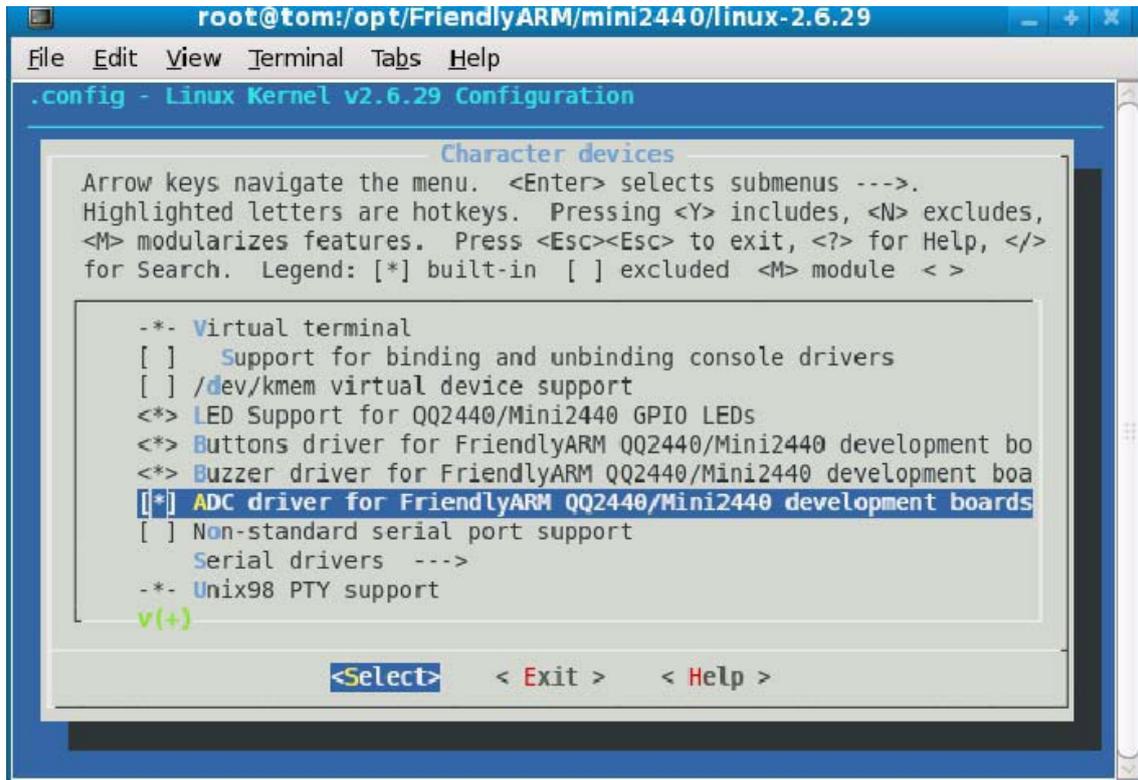
7.4.14 PWM ブザー

「Device Drivers」メニューの「Character devices - - ->」に入ります。



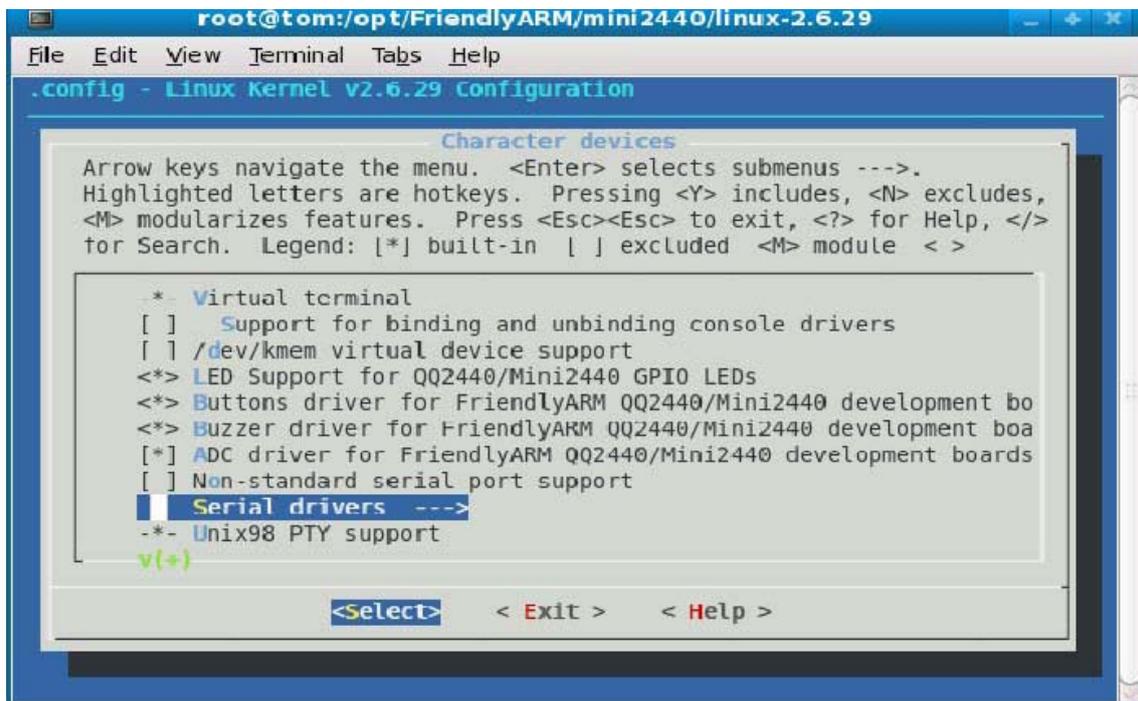
7.4.15 AD

「Device Drivers」メニューの「Character devices - - ->」に入ります。

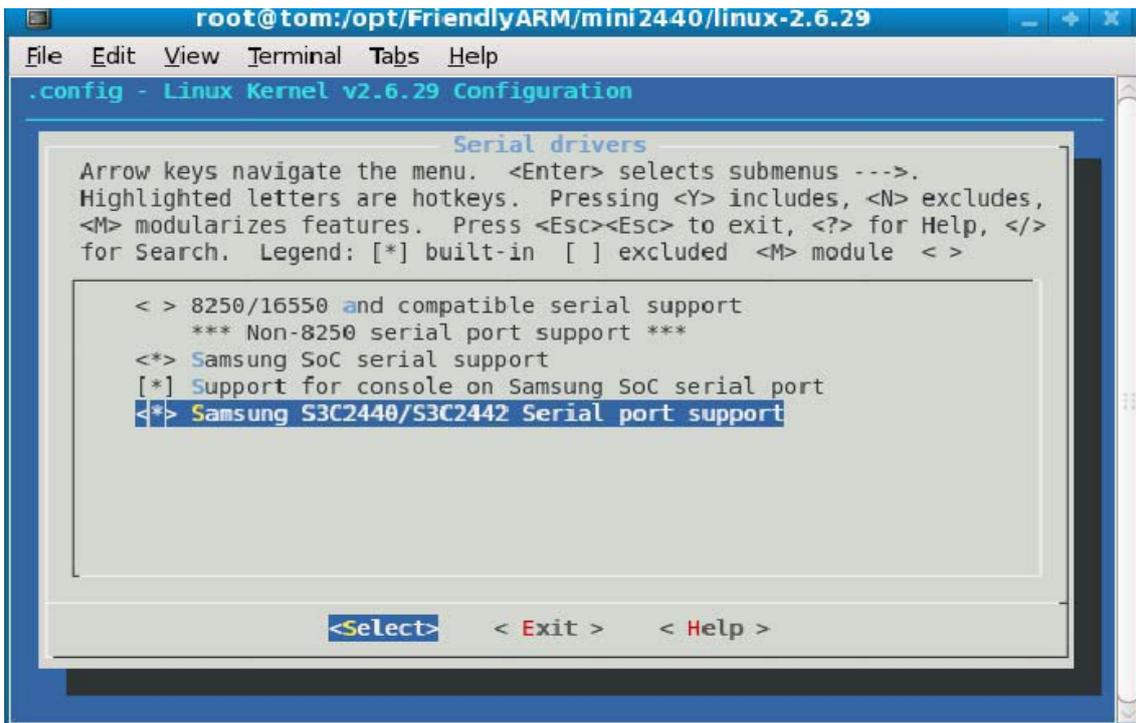


7.4.16 シリアルポート

「Device Drivers」メニューの「Character devices - - ->」に入ります。

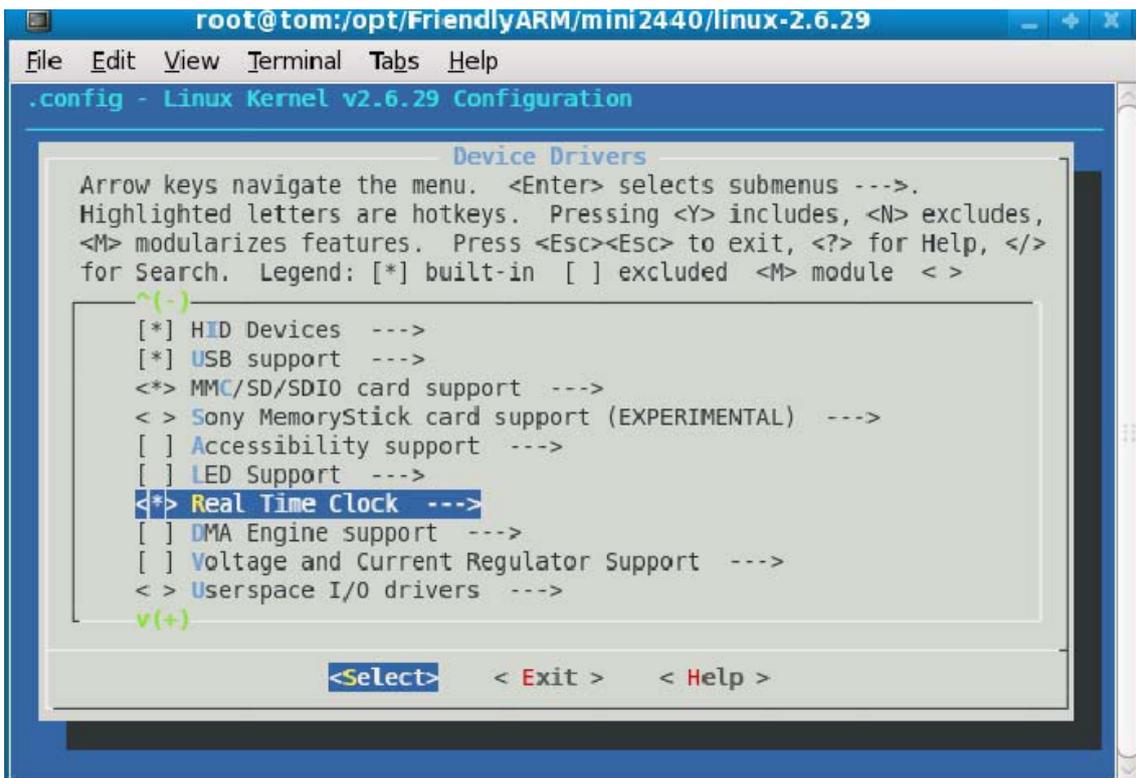


「serial driver」に入ります。



7.4.17 リアルタイマーRTC

「Device Drivers」メニューの「Character devices - ->」に入ります。



「Real Time Clock」に入ります。

```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Real Time Clock
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 <>

--- Real Time Clock
[*] Set system time from RTC on startup and resume
(rtc0) RTC used to set the system time
[ ] RTC debug support
*** RTC interfaces ***
[*] /sys/class/rtc/rtcN (sysfs)
[*] /proc/driver/rtc (procfs for rtc0)
[*] /dev/rtcN (character devices)
[ ] RTC UIE emulation on dev interface
<> Test driver/device
v(+)
```

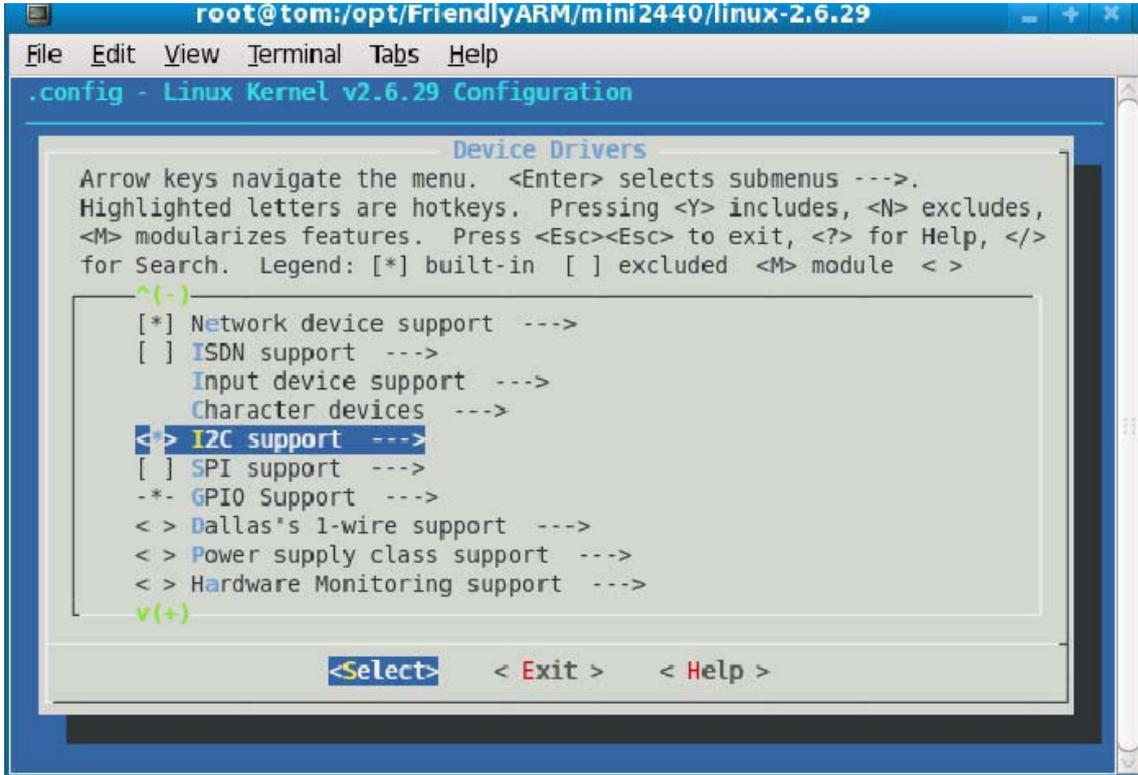
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Real Time Clock
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 <>

^(-)
<> Maxim/Dallas DS1553
<> Maxim/Dallas DS1742/1743
<> Simtek STK17TA8
<> ST M48T86/Dallas DS12887
<> ST M48T35
<> ST M48T59/M48T08/M48T02
<> TI BQ4802
<> EM Microelectronic V3020
*** on-CPU RTC drivers ***
<+> Samsung S3C series SoC RTC
```

7.4.18 I2C - EEPROM

「Device Drivers」メニューの「I2C support」に入ります。

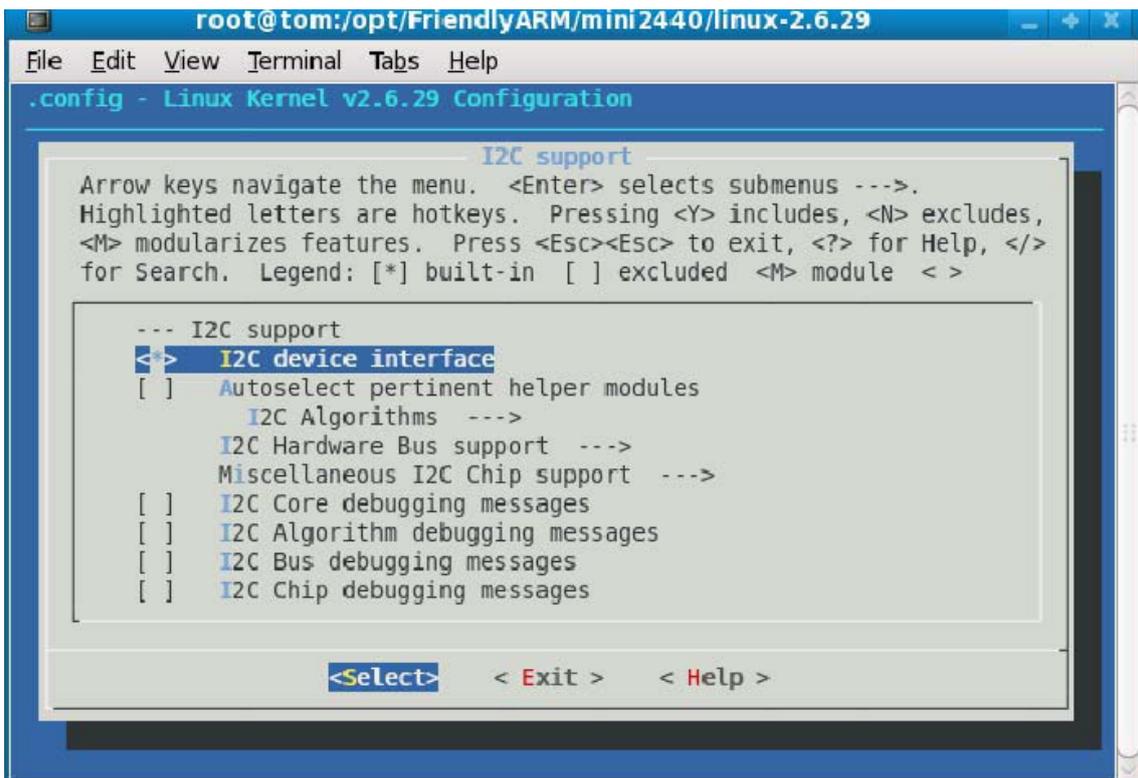


```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Device Drivers
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 <>

^(-)
[*] Network device support --->
[ ] ISDN support --->
  Input device support --->
  Character devices --->
<M> I2C support --->
[ ] SPI support --->
-* GPIO Support --->
<> Dallas's 1-wire support --->
<> Power supply class support --->
<> Hardware Monitoring support --->
v(+)

<Select> <Exit> <Help>
```

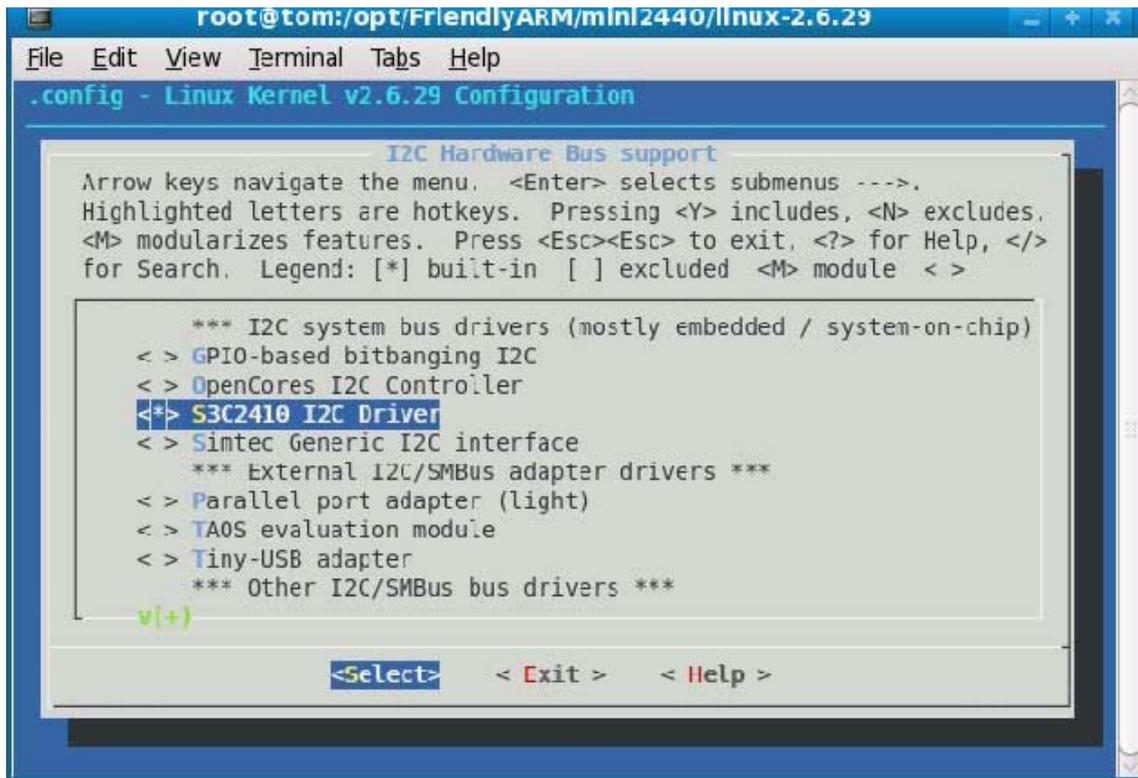


```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

I2C support
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 <>

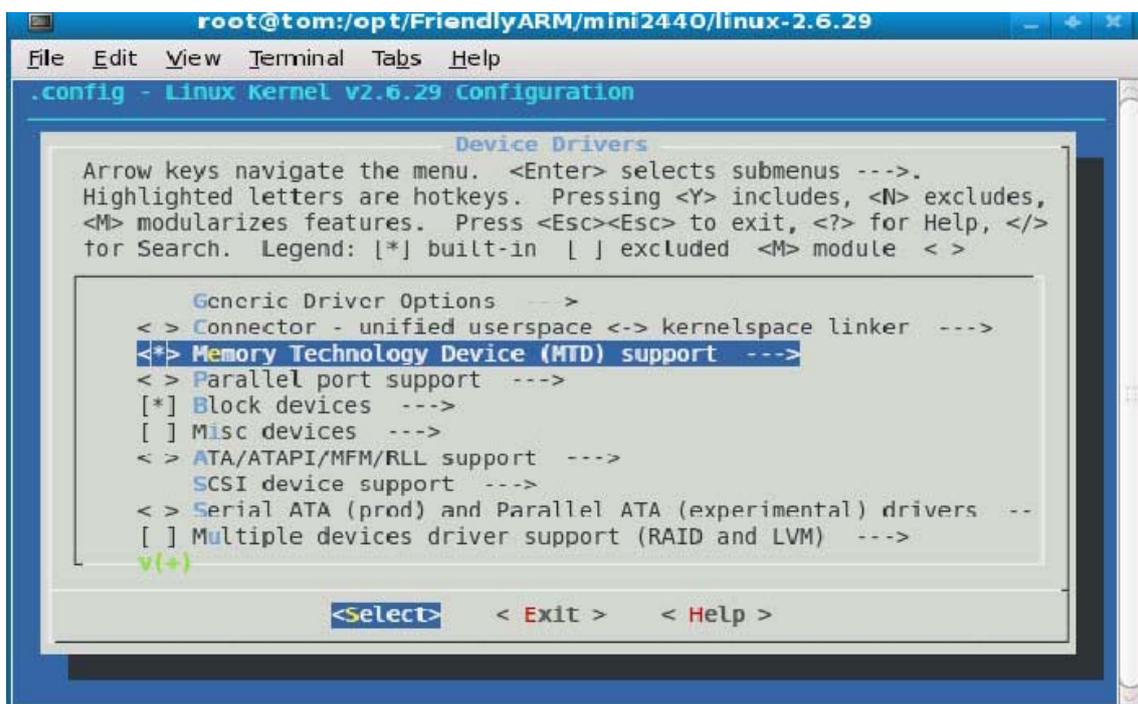
--- I2C support
<M> I2C device interface
[ ] Autoselect pertinent helper modules
  I2C Algorithms --->
  I2C Hardware Bus support --->
  Miscellaneous I2C Chip support --->
[ ] I2C Core debugging messages
[ ] I2C Algorithm debugging messages
[ ] I2C Bus debugging messages
[ ] I2C Chip debugging messages

<Select> <Exit> <Help>
```



7.4.19 yaff2s ファイルシステム

「Device Drivers」メニューの「Memory Technology Device (MTD) support」に入ります。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Memory Technology Device (MTD) support
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 < >
^(-)
< > MTD concatenating support
[*] MTD partitioning support
< > MTD tests support
< > RedBoot partition table parsing
[ ] Command line partition table parsing
< > ARM Firmware Suite partition parsing
< > TI AR7 partitioning support
*** User Modules And Translation Layers ***
<*> Direct char device access to MTD devices
-* Common interface to block layer for MTD *translation layers
v(+)

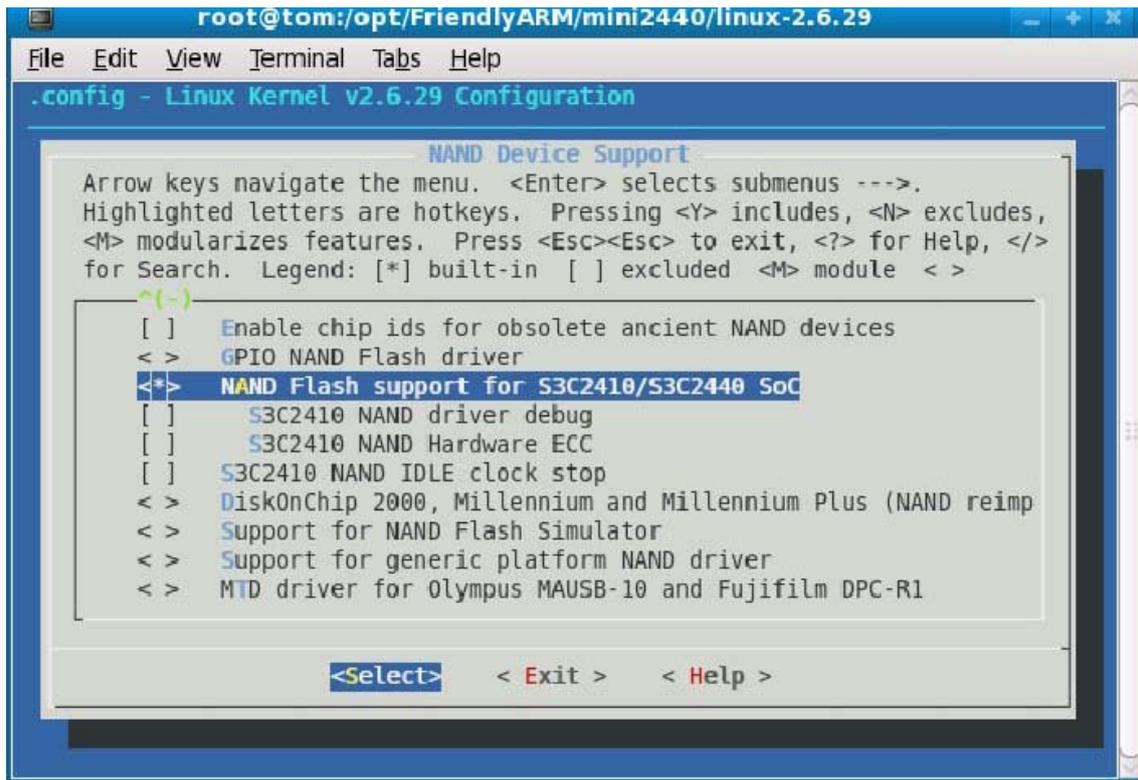
<Select> < Exit > < Help >
```

画面のように「*」を選択します。「NAND Device Support」を探して、入ります。

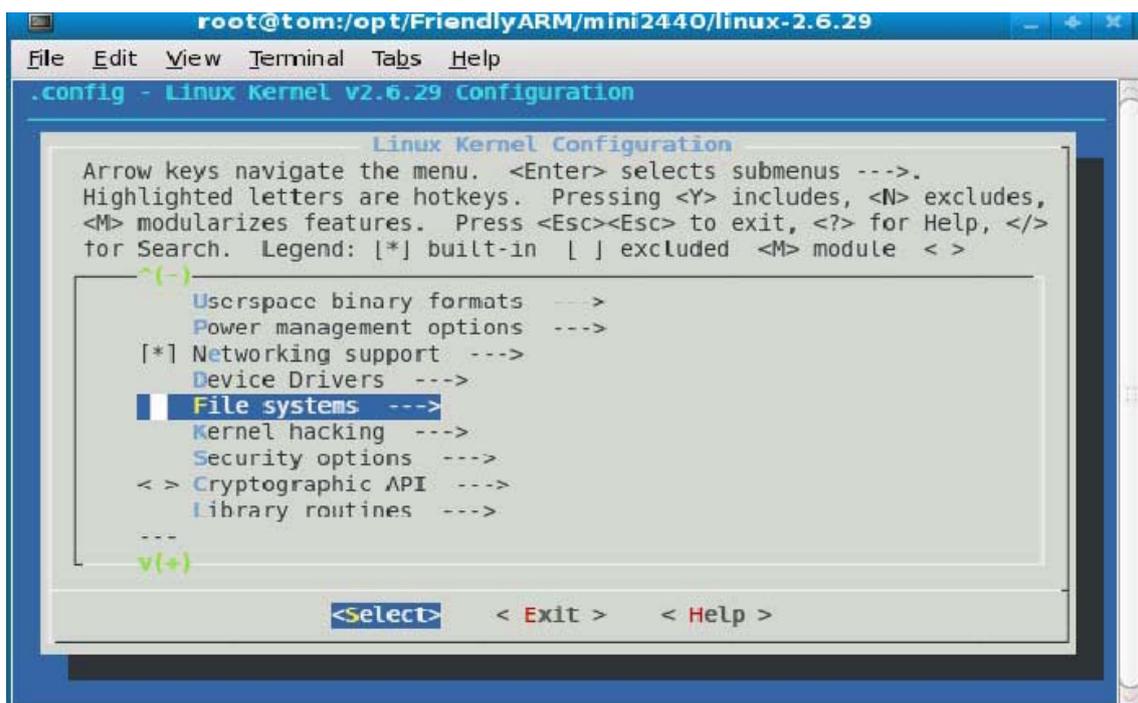
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Memory Technology Device (MTD) support
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 < >
^(-)
< > NAND (NAND Flash Translation Layer) support
< > INFTL (Inverse NAND Flash Translation Layer) support
< > Resident Flash Disk (Flash Translation Layer) support
< > NAND SFFDC (SmartMedia) read only translation layer
< > Log panic/oops to an MTD buffer
RAM/ROM/Flash chip drivers --->
Mapping drivers for chip access --->
Self-contained MTD device drivers --->
<*> NAND Device Support --->
< > OneNAND Device Support --->
v(+)

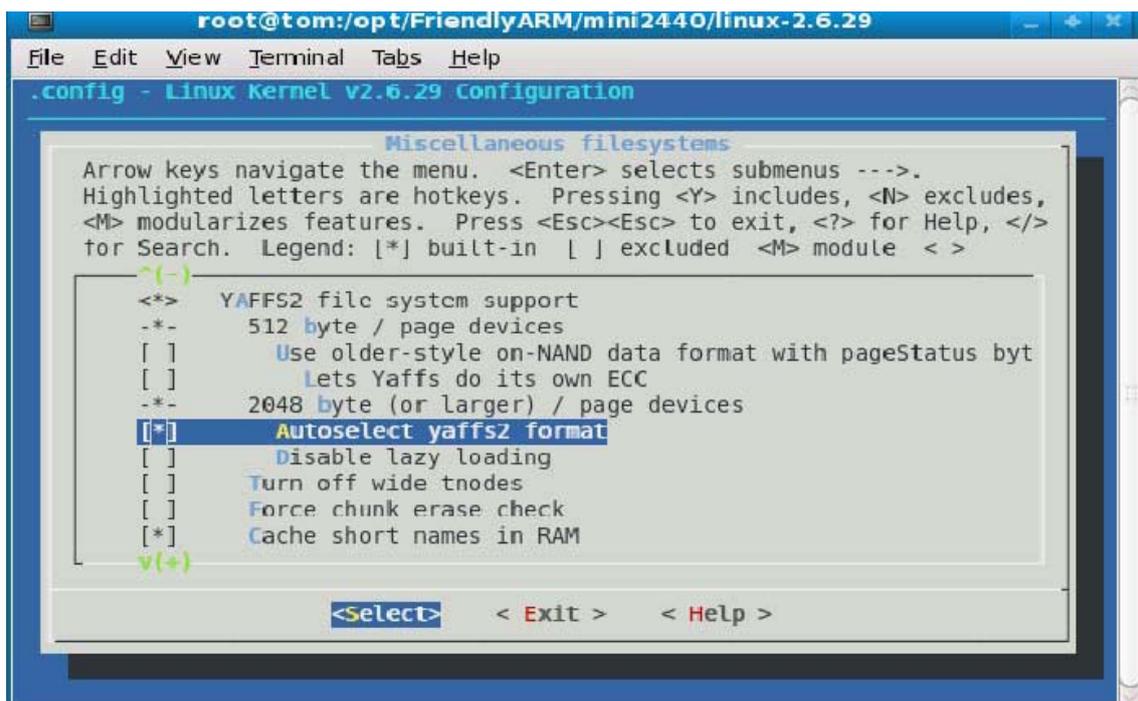
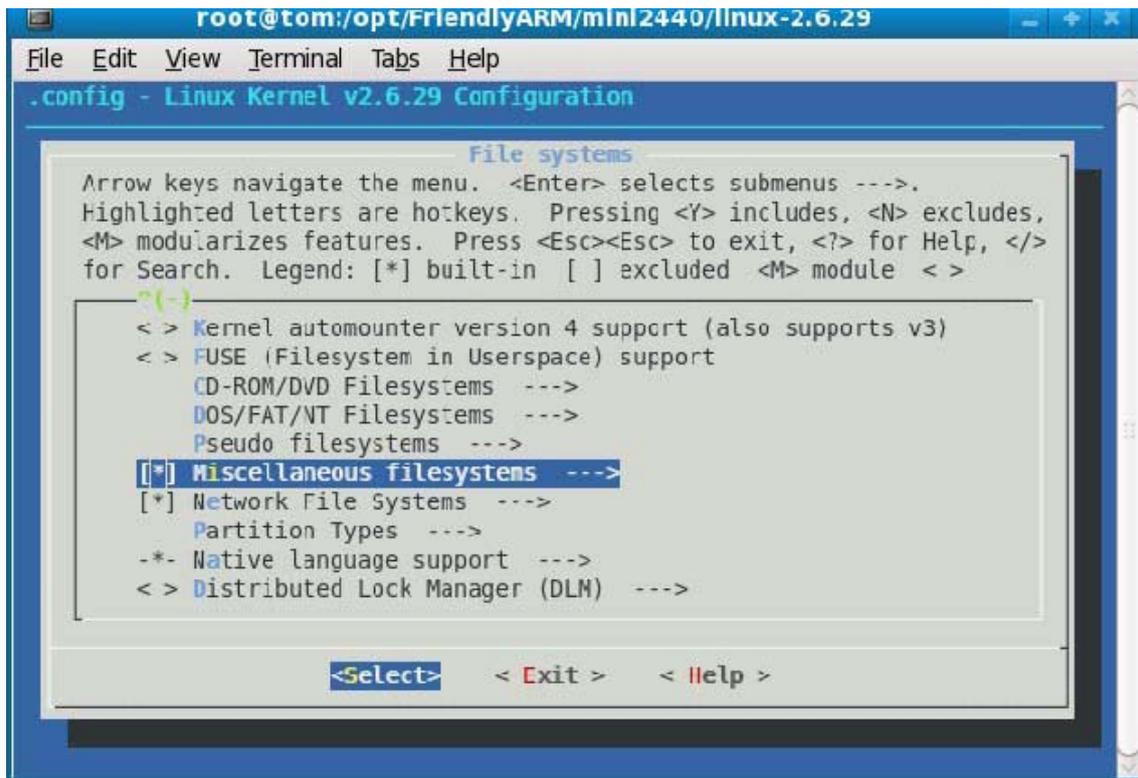
<Select> < Exit > < Help >
```



選択して、メインメニューに戻ります。メインメニューで「File systems」を探して、入ります。



「Miscellaneous filesystems」に入ります。

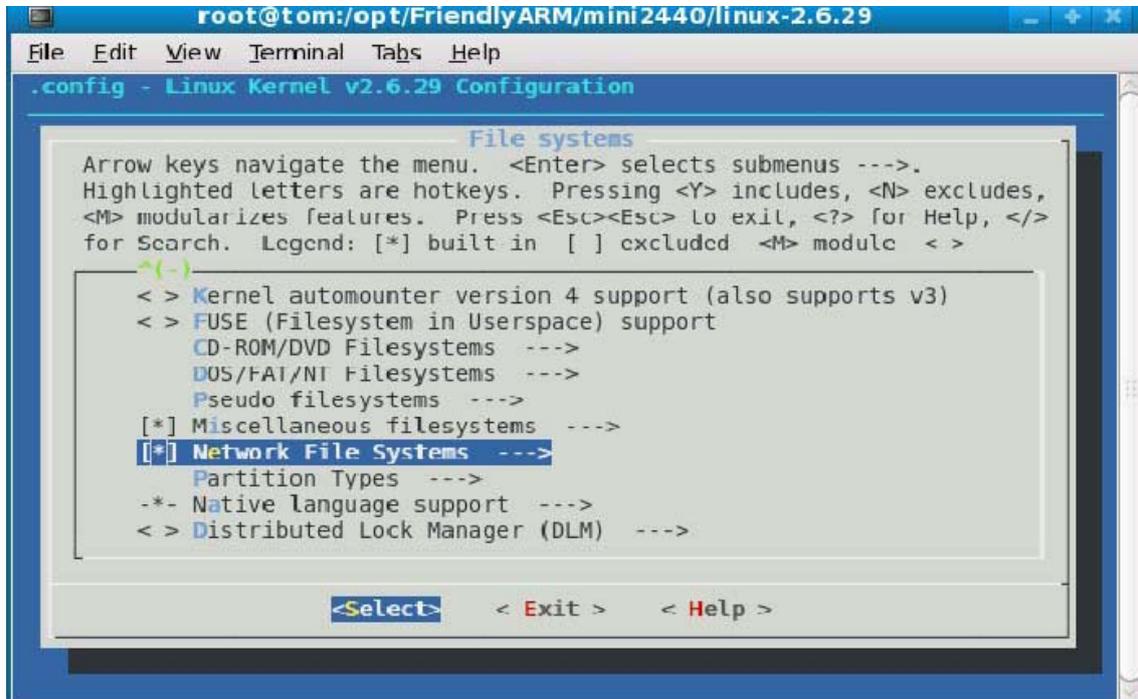


「YAFFS2 file system support」を選択します。「Exit」で「Device Drivers」メニューに戻ります。

7.4.20 EXT2/VFAT/ NFS ファイルシステム

NFS :

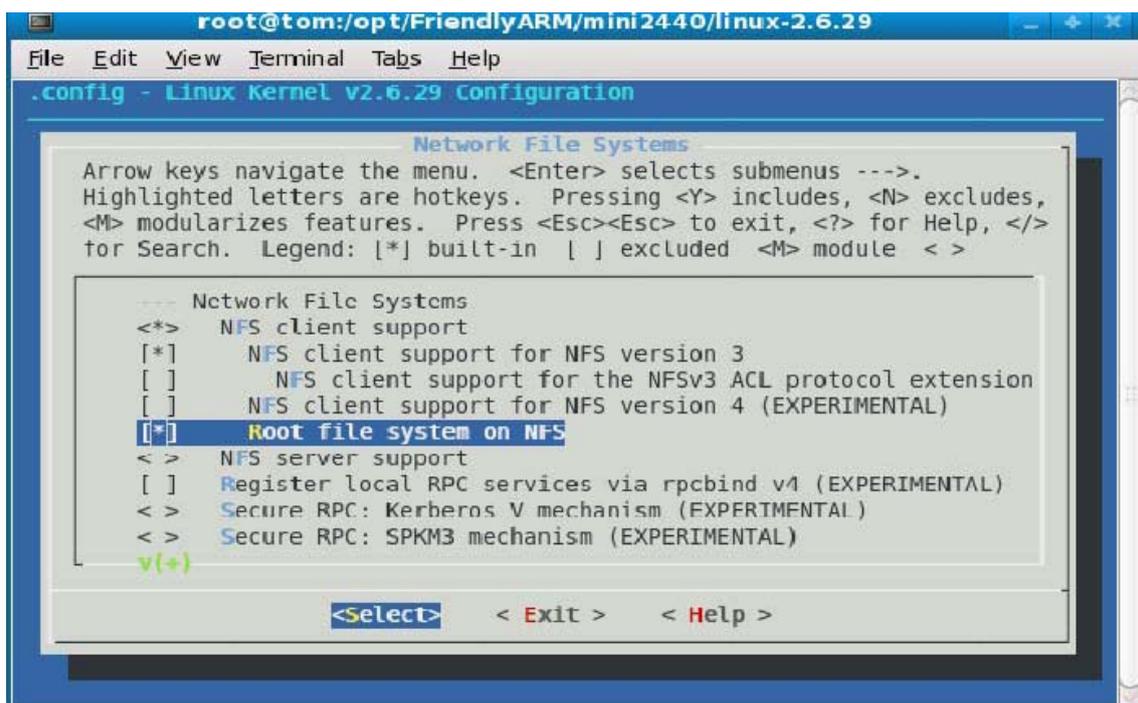
「File systems」メニューのNetwork File Systemsに入ります。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

File systems
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 < >
v(-)
< > Kernel automounter version 4 support (also supports v3)
< > FUSE (Filesystem in Userspace) support
CD-ROM/DVD Filesystems --->
DOS/FAT/NT Filesystems --->
Pseudo filesystems --->
[*] Miscellaneous filesystems --->
[*] Network File Systems --->
Partition Types --->
-*- Native language support --->
< > Distributed Lock Manager (DLM) --->

<Select> < Exit > < Help >
```



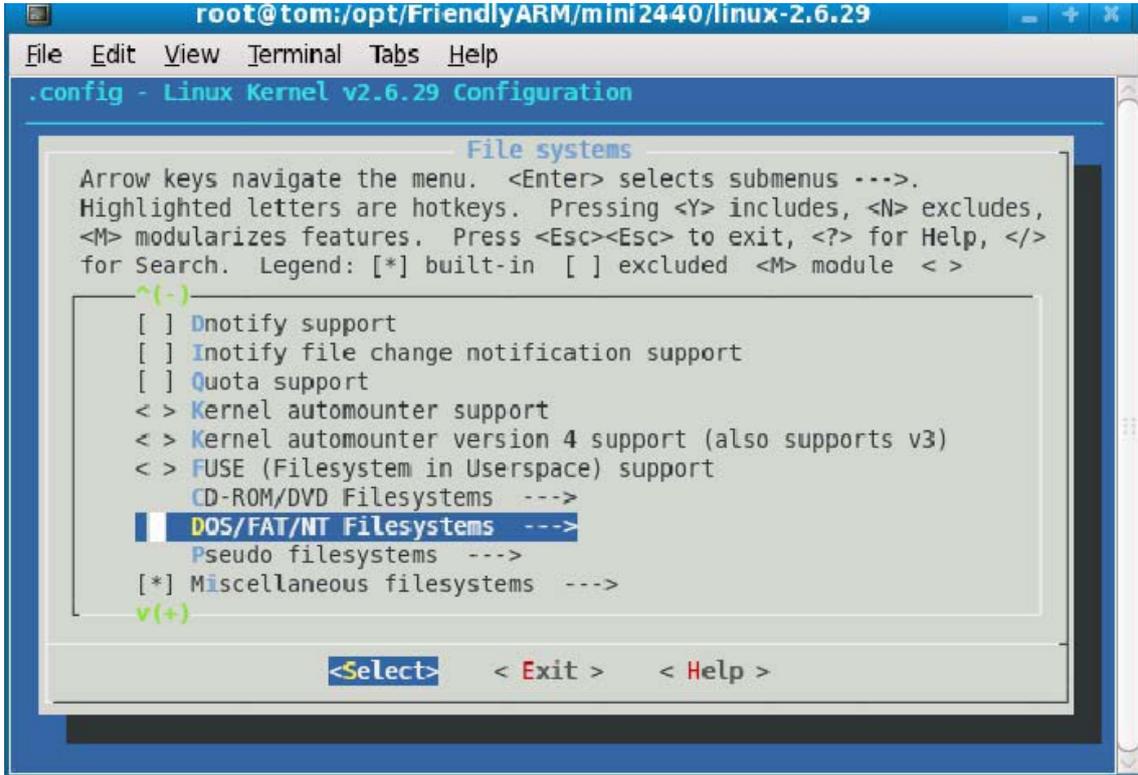
```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

Network File Systems
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 < >
v(+)
--- Network File Systems
<*> NFS client support
[*] NFS client support for NFS version 3
[ ] NFS client support for the NFSv3 ACL protocol extension
[ ] NFS client support for NFS version 4 (EXPERIMENTAL)
[*] Root file system on NFS
< > NFS server support
[ ] Register local RPC services via rpcbind v4 (EXPERIMENTAL)
< > Secure RPC: Kerberos V mechanism (EXPERIMENTAL)
< > Secure RPC: SPKM3 mechanism (EXPERIMENTAL)

<Select> < Exit > < Help >
```

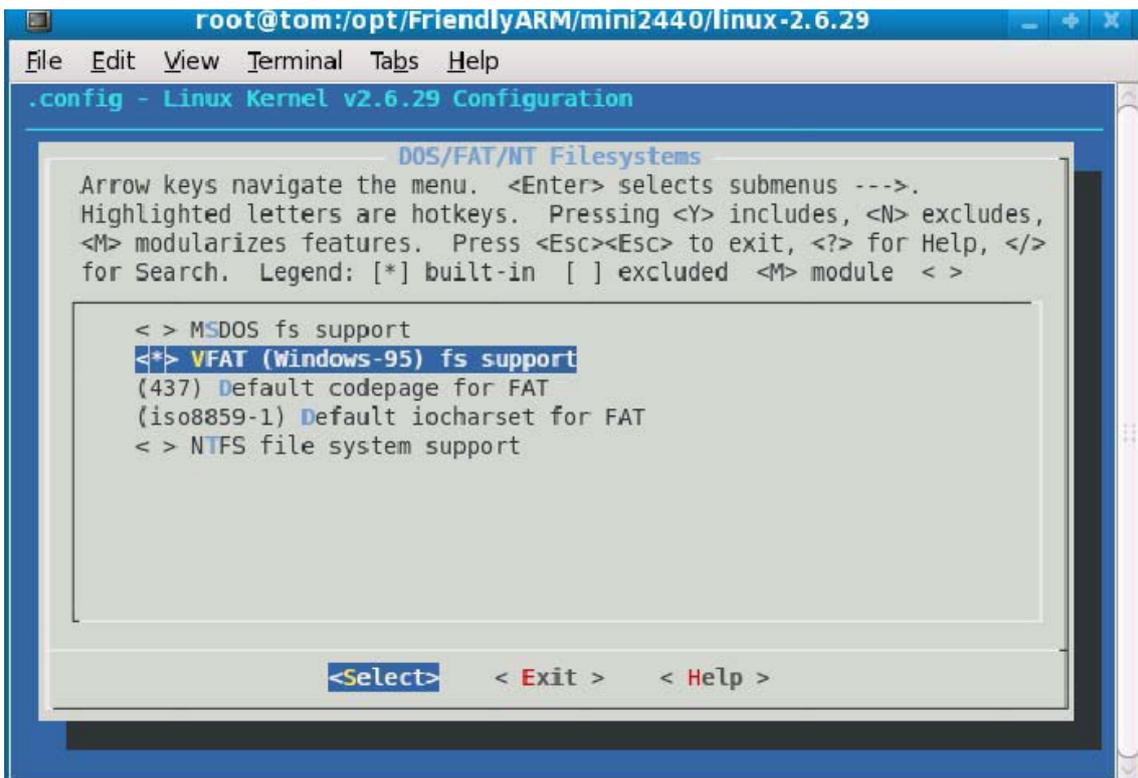
FAT :

「File systems」メニューの「DOS/FAT/NT Filesystems」に入ります。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

File systems
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 < >
^(-)
[ ] Dnotify support
[ ] Inotify file change notification support
[ ] Quota support
< > Kernel automounter support
< > Kernel automounter version 4 support (also supports v3)
< > FUSE (Filesystem in Userspace) support
CD-ROM/DVD Filesystems --->
DOS/FAT/NT Filesystems --->
Pseudo filesystems --->
[*] Miscellaneous filesystems --->
v(+)
```



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
.config - Linux Kernel v2.6.29 Configuration

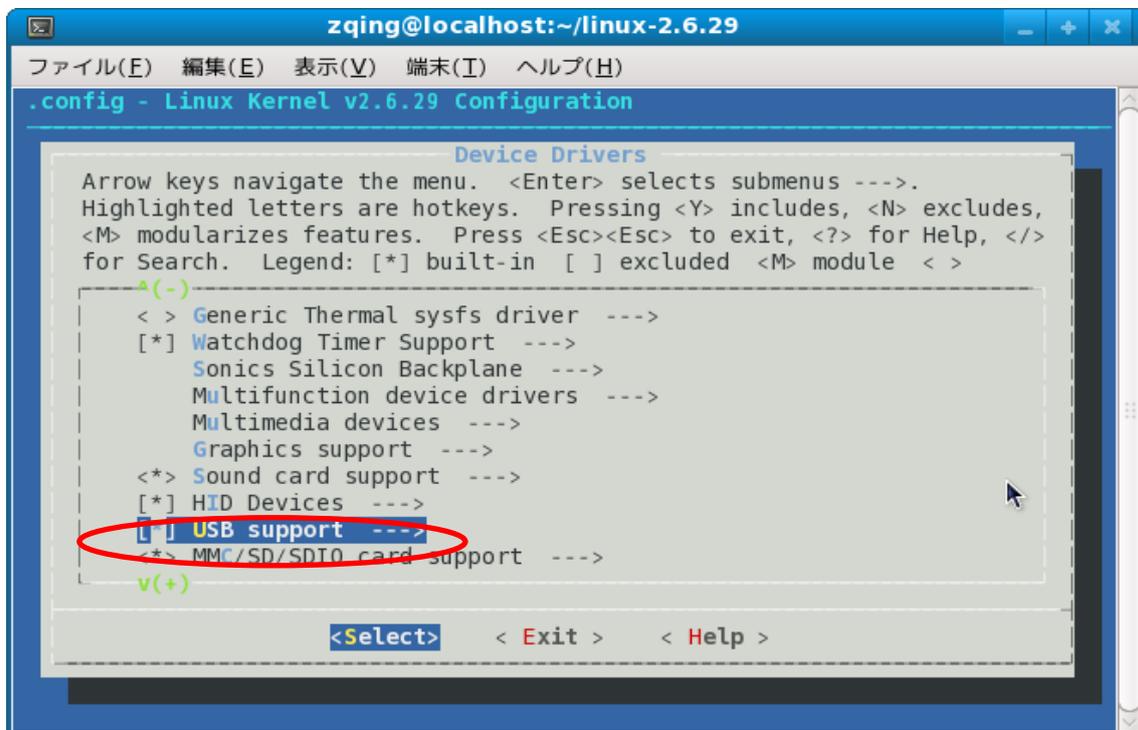
DOS/FAT/NT Filesystems
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >
< > MSDOS fs support
<*> VFAT (Windows-95) fs support
(437) Default codepage for FAT
(iso8859-1) Default iocharset for FAT
< > NTFS file system support
```

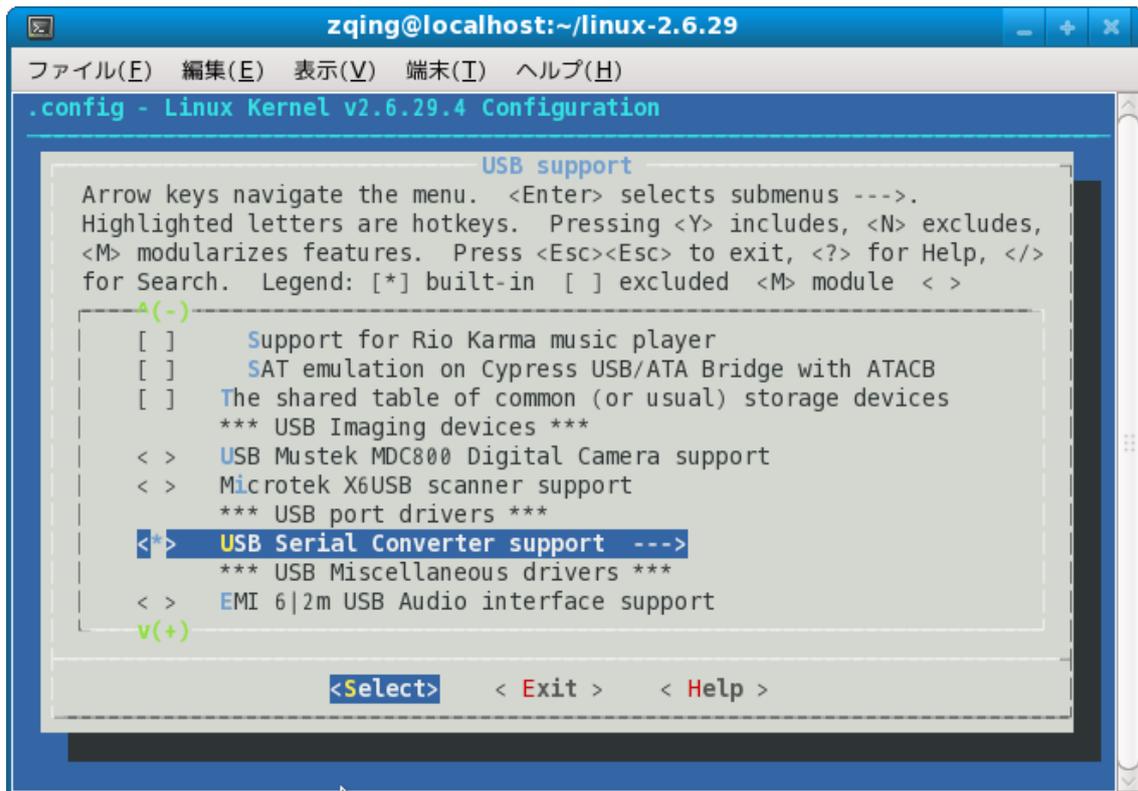
7.4.21 USB-RS232 シリアルポート



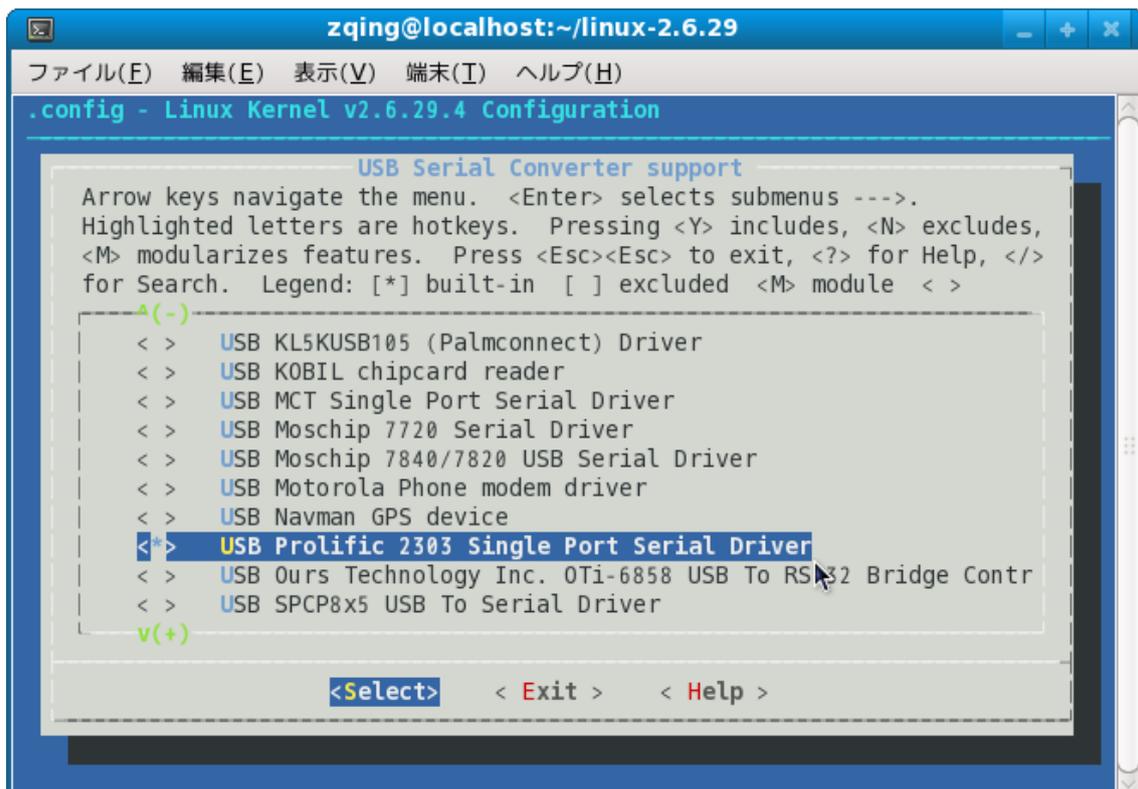
弊社が販売している USB-RS232 変換ケーブル

「Device Drivers」メニューの「USB support」に入ります。





「USB Serial Converter support」に入ります。



Prolific 社の PL2303 のドライバを選択します。

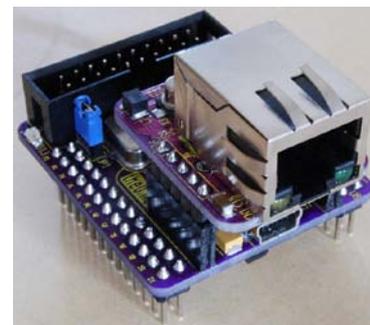
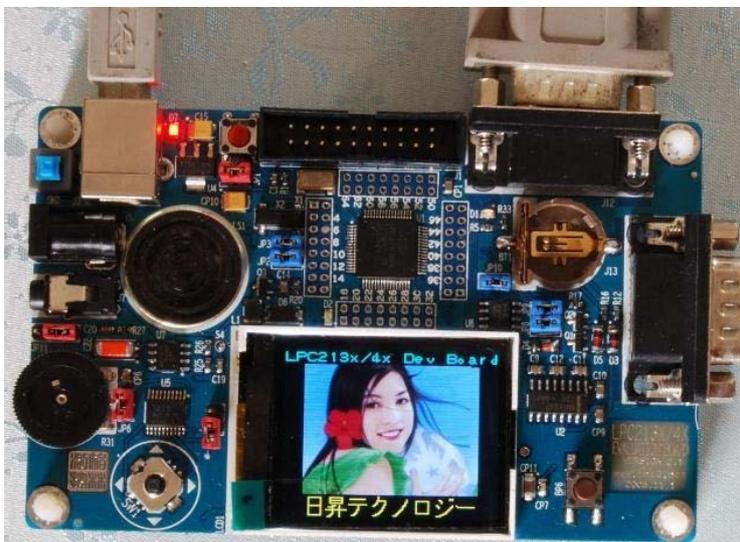
7.4.22 ARM7TDMI/LPC2148 との通信

ARM9 は標準 OS に Linux を採用します。Linux には、信頼性が高いネットワークスタックが実装され、利用できます。従って、ネットワークに接続する信頼性の高い遠隔制御機器が、容易に作成できる利点があります。Linux にも USB スタックが実装され、多種類の USB デバイスを利用できます。例えば、USB プリンター、USB 無線 LAN、USB メモリ、SD カードなど。パソコンの Linux 上のアプリケーションが ARM9 上で利用できます。ゼロから開発せずに、例えば Web サーバーなどが組み込み用機器で利用できるわけで、これは非常に大きな利点といえます。

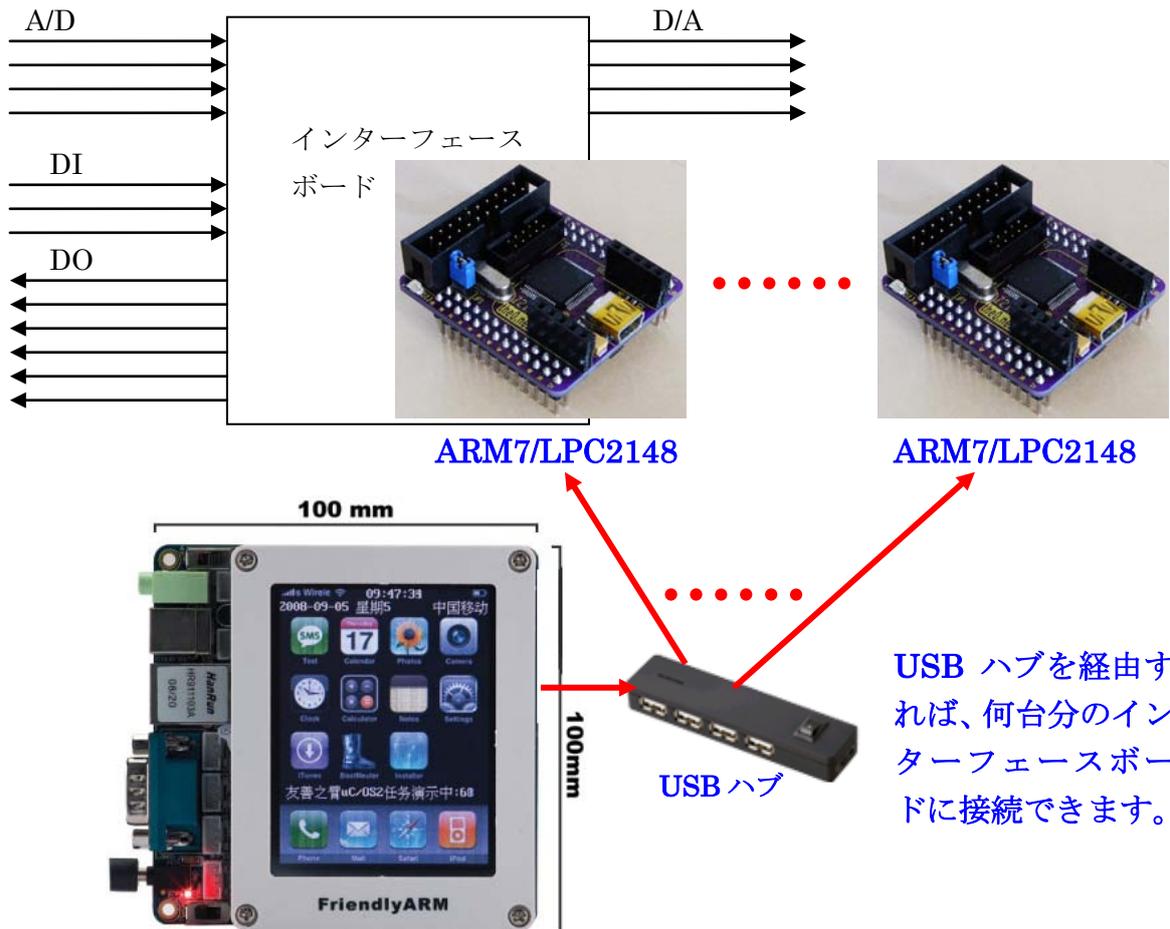
Linux の便利さの反面、複雑、重い、反応速度が遅いです。反応速度は大体数十 ms ぐらいです。この反応速度は人間との会話に満足できますが、機械制御のリアルタイム性に足りないかもしれません。

ARM7 シリーズはリアルタイム制御向けのマイコンです。OS なしあるいは簡単な RTOS を搭載します。1 μ s~1ms 以上の反応速度が実現できます。LPC2148 は NXP 社によって開発された ARM7 シリーズのマイコンです。CPU の周波数 60MHz、512KB Flash、42KB RAM。14 チャンネル 10 ビット AD、1 チャンネル 10 ビット DA、6 チャンネルの PWM。

その上、LPC2148 には USB ターゲットポートを持ちます。最大通信速度 12Mbps。LPC2148 は USB デバイスとして使えます。ARM9 は USB ハブを経由すれば、何台分の LPC2148 にも接続できます。システムは ARM9/MINI2440 と ARM7/LPC2148 を同時に採用すれば、Linux の便利な機能と ARM7 のリアルタイム性を組み合わせ、高度複雑なアプリケーションとリアルタイム制御が両立できるシステムを作れます。

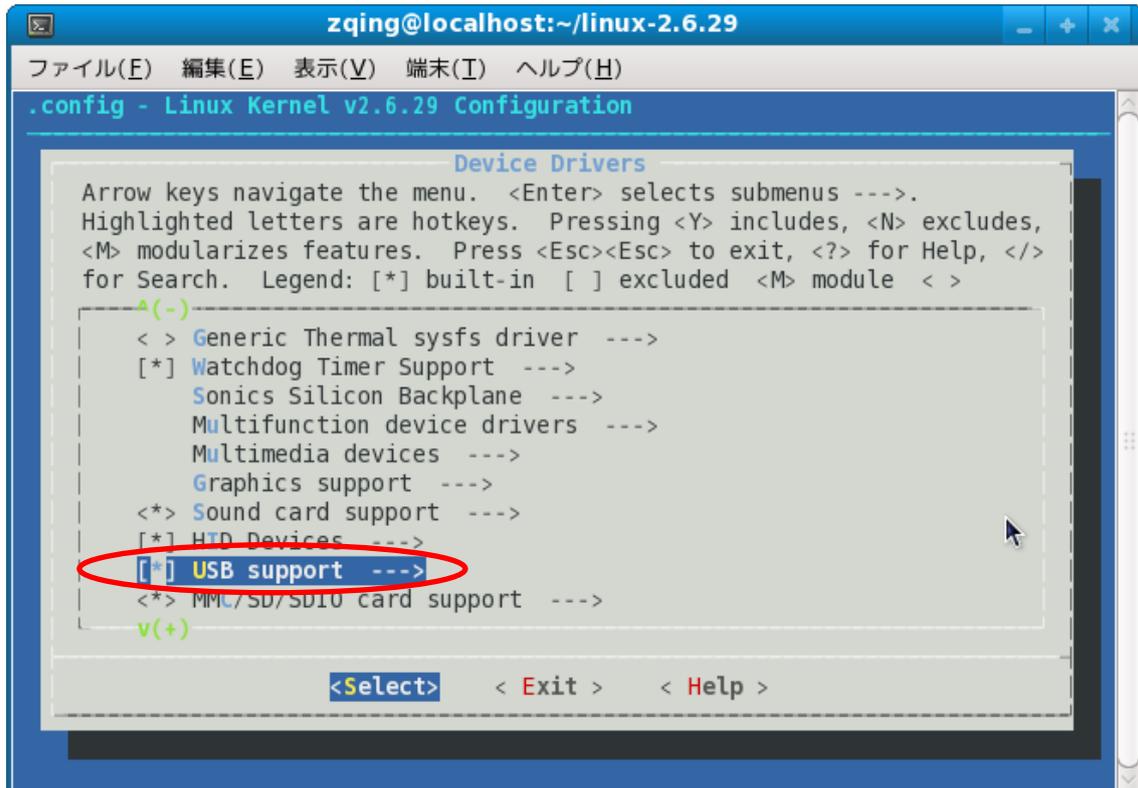


弊社が販売している LPC2148 開発キットとモジュール

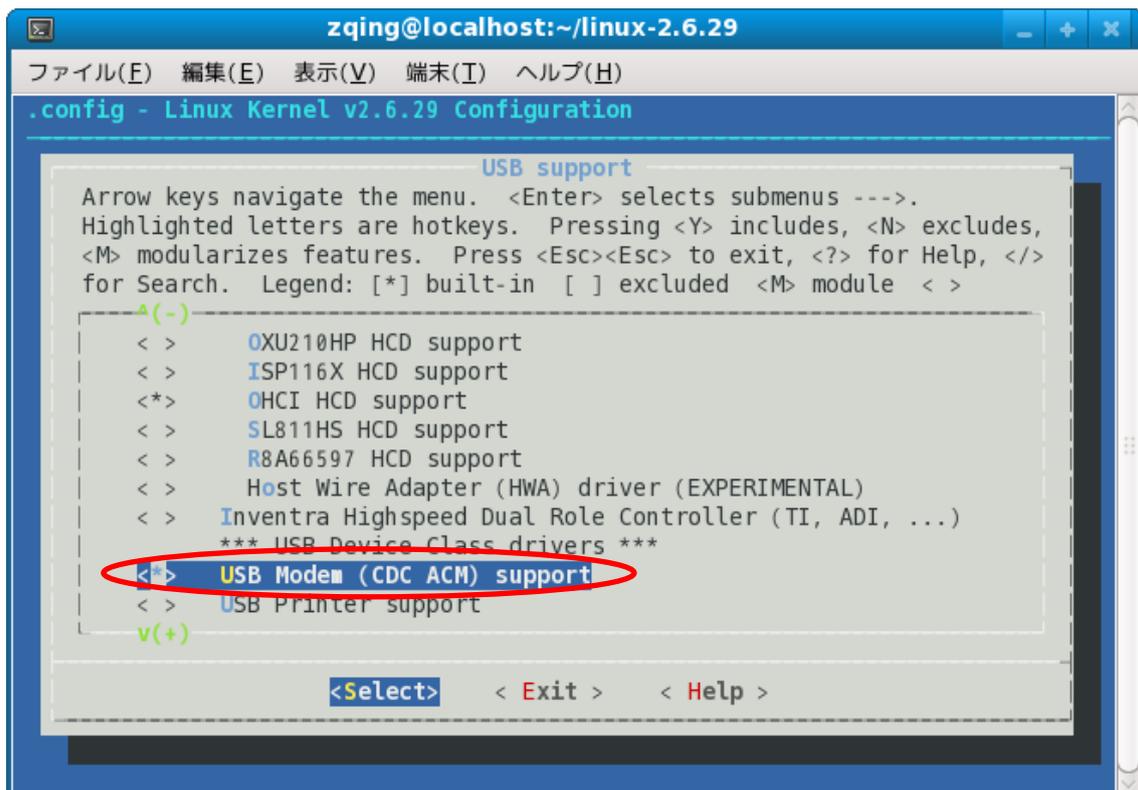


ARM9がARM7TDMI/LPC2148をUSBで通信する様子

「Device Drivers」メニューの「USB support」に入ります。



「USB Modem(CDC ACM) support」を選択します。“Exit” & “Save” します。



7.5 Linux 起動ロゴを作る



Linux が起動の時、このようなロゴが出てきます。自分で好きな画像に変換することができます。

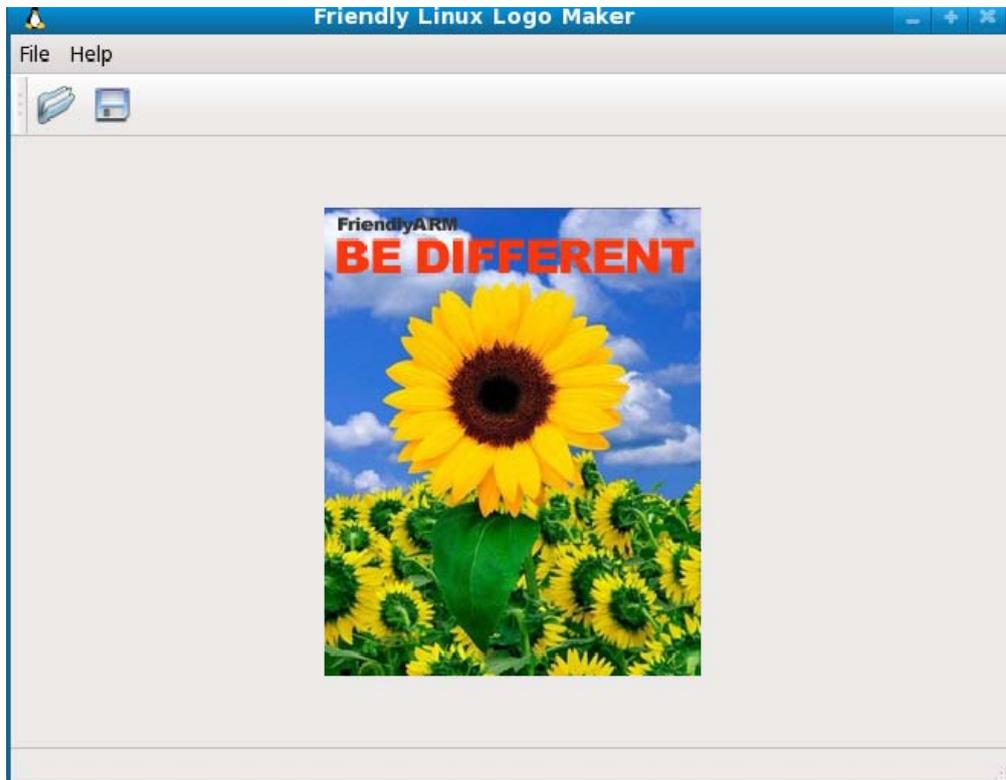
ロゴを作るツールを解凍します。

```
$ su スーパーユーザーに切り替え
```

```
# tar zxvf logomaker.tgz -C /
```

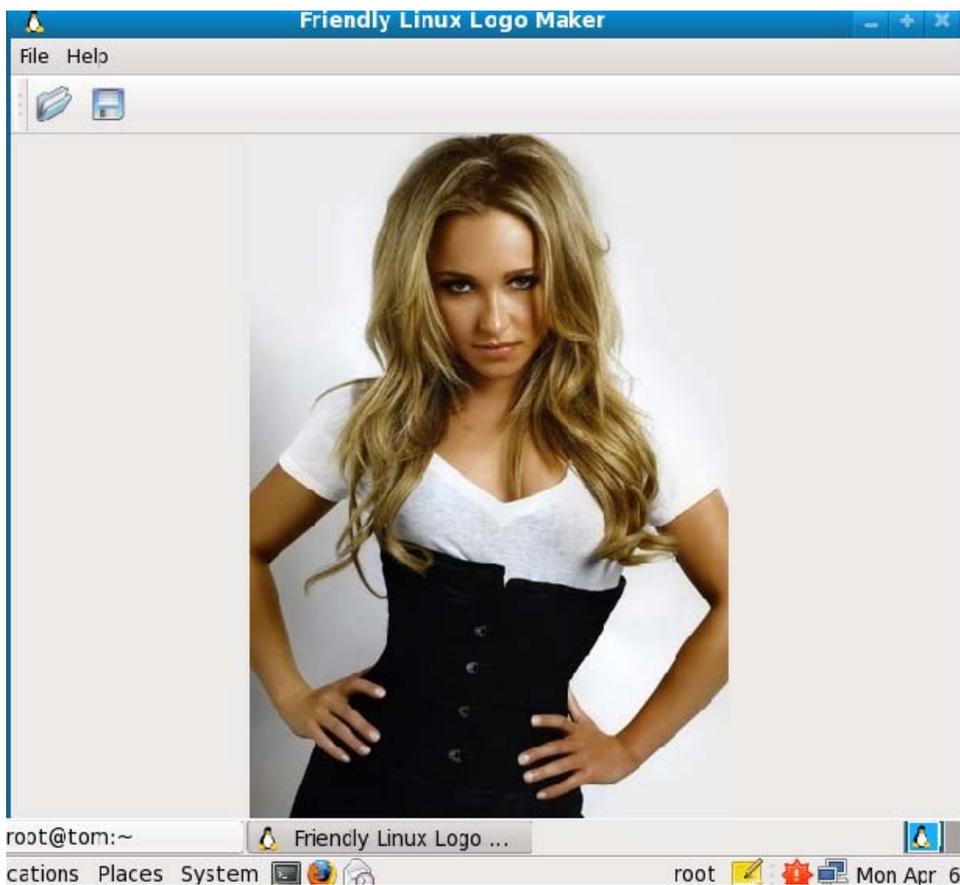
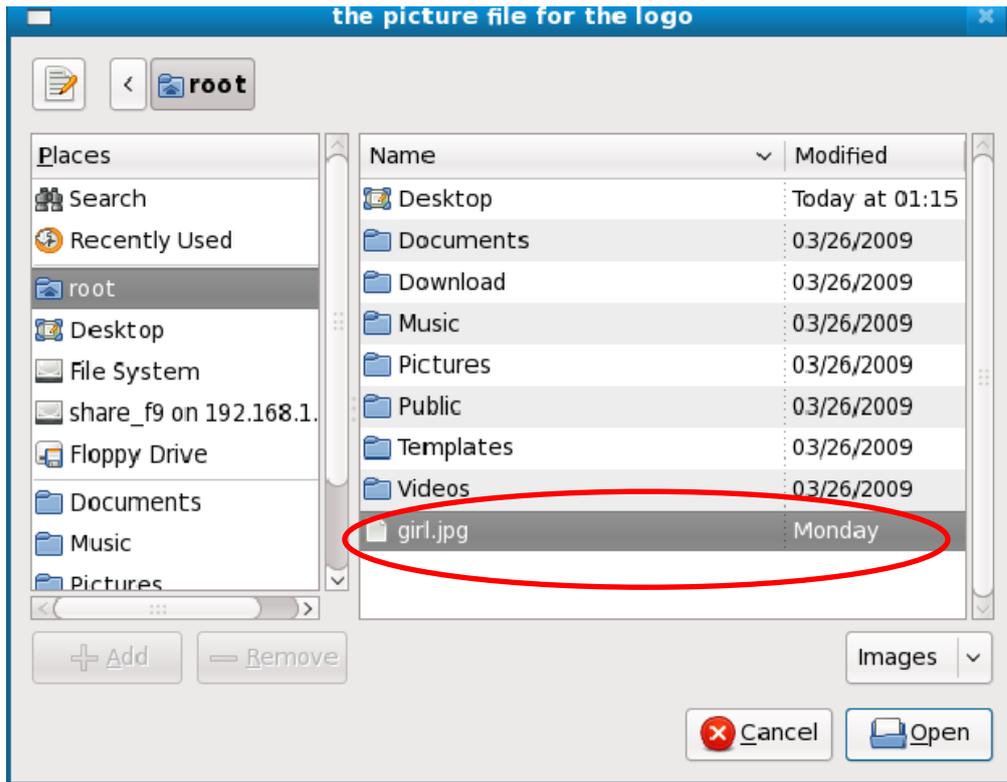
コンソールで

```
$ logomaker
```



Logomaker の初の画面です。

File → Open a picture file...で好きなピクチャを選択します。





選択したピクチャが表示されます。File → Convert the picture to a Linux Logo File で Linux logo に変換します。linux_logo_clut224.ppm というロゴファイルを生成します。このファイルを linux-2.6.29/drivers/video/logo にコピーすれば、新ロゴを生成します。

7.6 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。

```
$ su スーパーユーザーに切り替え
```

```
# tar xvzf mkyaffsimage.tgz -C /
```

```
# mkyaffsimage root_qtopia root_qtopia.img
```

```
root@tom:/opt/FriendlyARM/mini2440
File Edit View Terminal Tabs Help
Object 3094, root_qtopia/etc/init.d/rcS is a file, 3 data chunks written
Object 3095, root_qtopia/etc/init.d/ifconfig-eth0 is a file, 2 data chunks written
Object 3096, root_qtopia/etc/issue.net is a file, 1 data chunks written
Object 3097, root_qtopia/etc/boa is a directory
Object 3098, root_qtopia/etc/boa/boa.conf is a file, 1 data chunks written
Object 3099, root_qtopia/etc/ftpchroot is a file, 1 data chunks written
Object 3100, root_qtopia/etc/profile is a file, 1 data chunks written
Object 3101, root_qtopia/etc/eth0-setting is a file, 1 data chunks written
Object 3102, root_qtopia/etc/services is a file, 2 data chunks written
Object 3103, root_qtopia/etc/localtime is a file, 1 data chunks written
Object 3104, root_qtopia/etc/mtab is a symlink to "/proc/mounts"
Object 3105, root_qtopia/etc/passwd is a file, 1 data chunks written
Object 3106, root_qtopia/etc/mime.types is a file, 25 data chunks written
Object 3107, root_qtopia/etc/mdev.conf is a file, 2 data chunks written
Object 3108, root_qtopia/etc/resolv.conf is a file, 1 data chunks written
Object 3109, root_qtopia/etc/login.defs is a file, 12 data chunks written
Object 3110, root_qtopia/etc/hosts is a file, 1 data chunks written
Object 3111, root_qtopia/proc is a directory
Operation complete.
2855 objects in 247 directories
86739 NAND pages
FriendlyARM Computer Technology Inc.
[root@tom mini2440]#
```

7.7 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ありません。

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。

7.7.1 簡単なドライバのソースコード

ソースコード：[linux-2.6.29/drivers/char/mini2440_hello_module.c](#)

```
#include <linux/kernel.h>
#include <linux/module.h>
static int __init mini2440_hello_module_init(void)
{
    printk("Hello, Mini2440 module is installed !%n");
    return 0;
}
static void __exit mini2440_hello_module_cleanup(void)
{
    printk("Good-bye, Mini2440 module was removed !%n");
}
module_init(mini2440_hello_module_init);
module_exit(mini2440_hello_module_cleanup);
MODULE_LICENSE("GPL");
```

7.7.2 コンフィグファイルを編集します

[linux-2.6.29/drivers/char/Kconfig](#)を開きます。下の内容を追加します(実は、追加完了しました、確認してみます)。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29/drivers/char
File Edit View Terminal Tabs Help

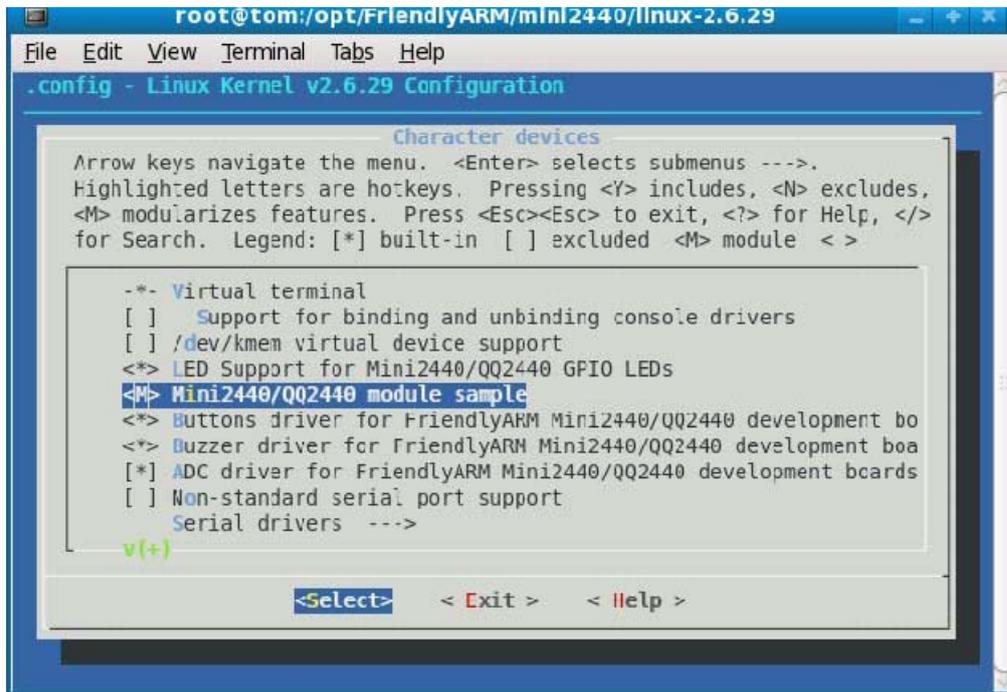
/dev/kmem device is rarely used, but can be used for certain
kind of kernel debugging operations.
When in doubt, say "N".

config LEDS_MINI2440
tristate "LED Support for Mini2440/QQ2440 GPIO LEDs"
depends on ARCH_S3C2410
help
This option enables support for LEDs connected to GPIO lines
on Mini2440/QQ2440 boards.

config MINI2440_HELLO_MODULE
tristate "Mini2440/QQ2440 module sample"
depends on ARCH_S3C2410
default m if MACH_FRIENDLY_ARM_MINI2440
help
Mini2440/QQ2440 module sample.

config MINI2440_BUTTONS
tristate "Buttons driver for FriendlyARM Mini2440/QQ2440 development boards"
depends on MACH_FRIENDLY_ARM_MINI2440
default y if MACH_FRIENDLY_ARM_MINI2440
```

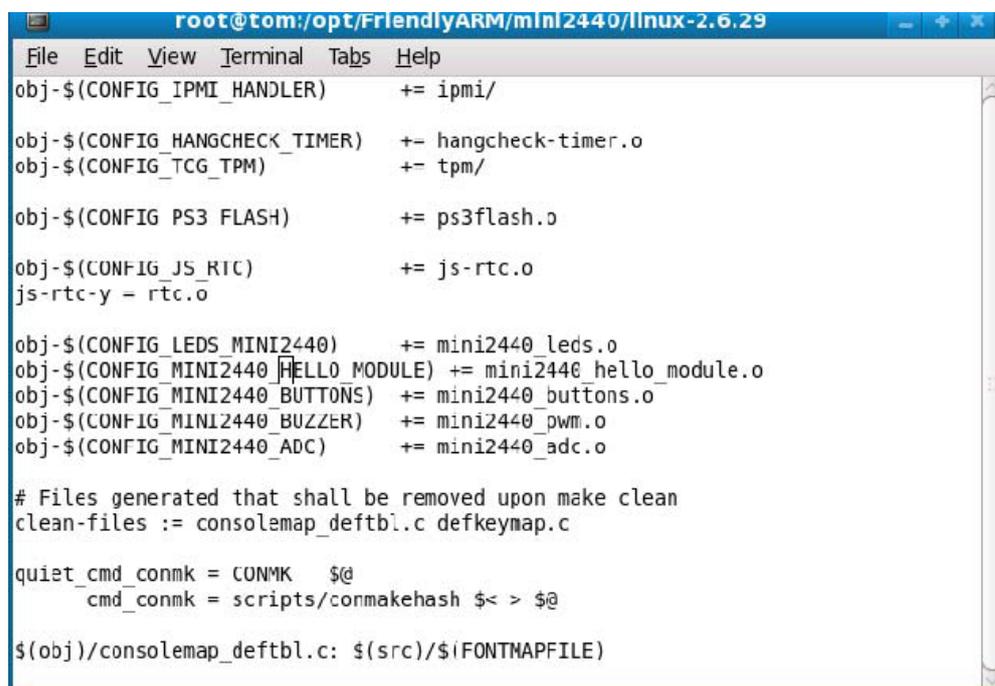
Linux-2.6.29でmake menuconfigを実行して、メニューDevice Drivers → Character devicesを選んで、



追加されたものが見えます。spaceキーで「M」を選択します。

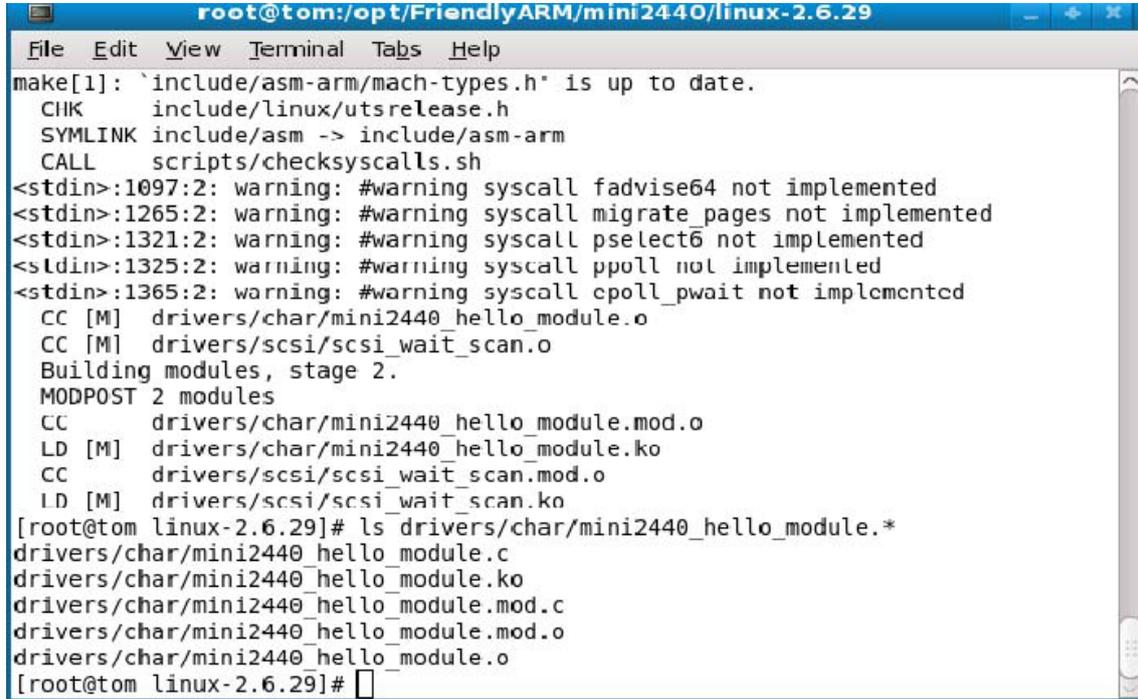
7.7.3 Makefile を編集

linux-2.6.29/drivers/char/Makefileを開きます。下の内容を追加します(実は、追加完了しました、確認してみます)。



7.7.4 ドライバをコンパイルします

linux-2.6.29 で make modules を実行します。linux-2.6.29/drivers/char/でオブジェクトファイル mini2440_hello_module.ko を生成させます。



```
root@tom:/opt/FriendlyARM/mini2440/linux-2.6.29
File Edit View Terminal Tabs Help
make[1]: `include/asm-arm/mach-types.h' is up to date.
CHK include/linux/utsrelease.h
SYMLINK include/asm -> include/asm-arm
CALL scripts/checksyscalls.sh
<stdin>:1097:2: warning: #warning syscall fadvise64 not implemented
<stdin>:1265:2: warning: #warning syscall migrate_pages not implemented
<stdin>:1321:2: warning: #warning syscall pselect6 not implemented
<stdin>:1325:2: warning: #warning syscall ppoll not implemented
<stdin>:1365:2: warning: #warning syscall cpoll_pwait not implemented
CC [M] drivers/char/mini2440_hello_module.o
CC [M] drivers/scsi/scsi_wait_scan.o
Building modules, stage 2.
MODPOST 2 modules
CC drivers/char/mini2440_hello_module.mod.o
LD [M] drivers/char/mini2440_hello_module.ko
CC drivers/scsi/scsi_wait_scan.mod.o
LD [M] drivers/scsi/scsi_wait_scan.ko
[root@tom linux-2.6.29]# ls drivers/char/mini2440_hello_module.*
drivers/char/mini2440_hello_module.c
drivers/char/mini2440_hello_module.ko
drivers/char/mini2440_hello_module.mod.c
drivers/char/mini2440_hello_module.mod.o
drivers/char/mini2440_hello_module.o
[root@tom linux-2.6.29]#
```

7.7.5 ARM9 ボードでドライバをインストールします

mini2440_hello_module.koをARM9にダウンロードロードします。

```
#insmod mini2440_hello_module.ko
```

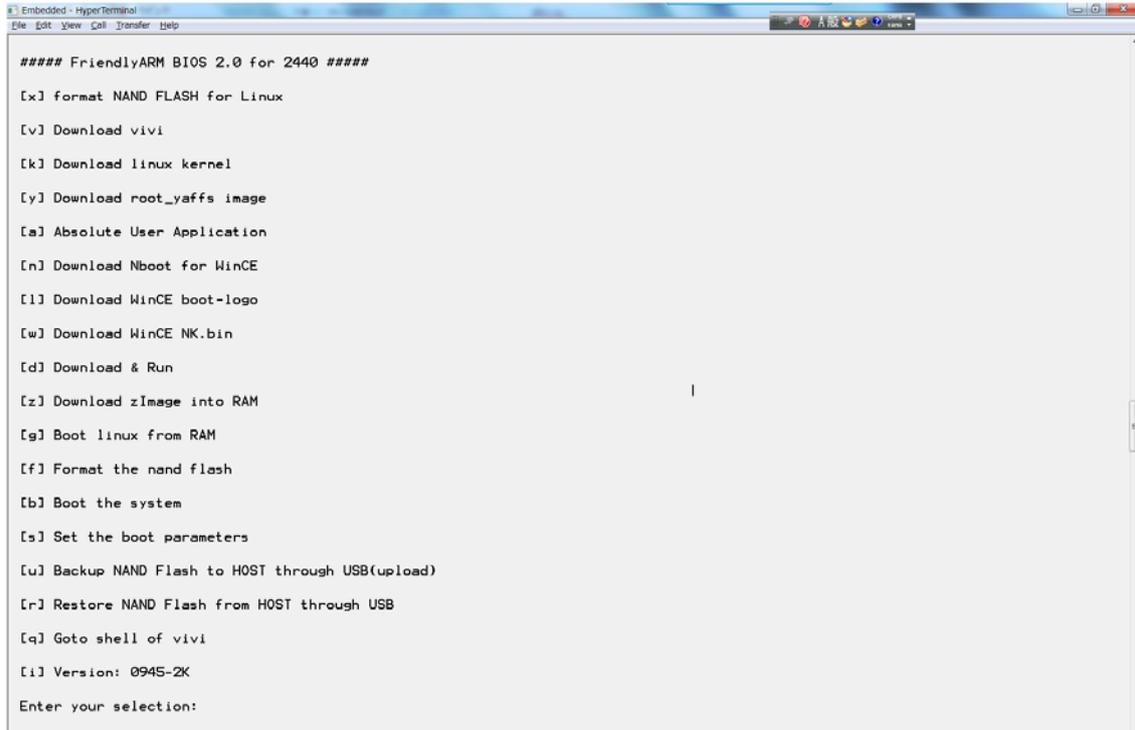
ドライバを削除します。

```
#rmmod mini2440_hello_module.ko
```

第八章 生成されたファイルを書き込む

8.1 NOR Flash から起動

ARM9ボードのS2スイッチを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。



```
##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

8.2 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mini2240にダウンロードします。その為、USBケーブルでmini2240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。

新しいハードウェアの検出ウィザード



新しいハードウェアの検索ウィザードの開始

お使いのコンピュータ、ハードウェアのインストール CD または Windows Update の Web サイトを検索して (ユーザーの了解のもとに) 現在のソフトウェアおよび更新されたソフトウェアを検索します。
[プライバシー ポリシー](#)を表示します。

ソフトウェア検索のため、Windows Update に接続しますか？

- はい、今回のみ接続します (Y)
- はい、今すぐおよびデバイスの接続時には毎回接続します (E)
- いいえ、今回は接続しません (N)

続行するには、[次へ] をクリックしてください。

< 戻る (B)

次へ (N) >

キャンセル

新しいハードウェアの検出ウィザード



このウィザードでは、次のハードウェアに必要なソフトウェアをインストールします。

SEC S3C2410X Test B/D



ハードウェアに付属のインストール CD またはフロッピー ディスクがある場合は、挿入してください。

インストール方法を選んでください。

- ソフトウェアを自動的にインストールする (推奨) (A)
- 一覧または特定の場所からインストールする (詳細) (S)

続行するには、[次へ] をクリックしてください。

< 戻る (B)

次へ (N) >

キャンセル

新しいハードウェアの検出ウィザード

検索とインストールのオプションを選んでください。

 次の場所で最適なドライバを検索する(S)

下のチェック ボックスを使って、リムーバブル メディアやローカル パスから検索できます。検索された最適なドライバがインストールされます。

 リムーバブル メディア (フロッピー、CD-ROM など) を検索(M) 次の場所を含める(Q):

参照(R)

 検索しないで、インストールするドライバを選択する(D)

一覧からドライバを選択するには、このオプションを選びます。選択されたドライバは、ハードウェアに最適なものとは限りません。

< 戻る(B)

次へ(N) >

キャンセル

新しいハードウェアの検出ウィザード

ソフトウェアをインストールしています。お待ちください...



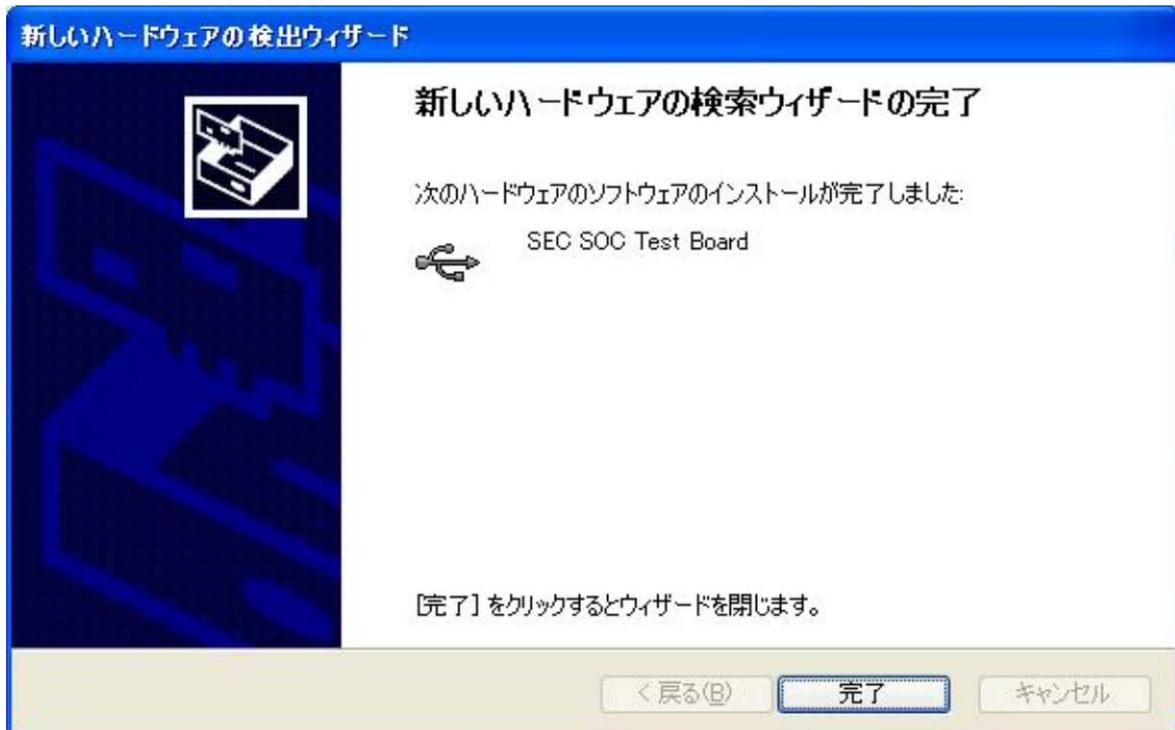
SEC SOC Test Board



< 戻る(B)

次へ(N) >

キャンセル



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、mini2240とパソコンを繋ぐことが確認できます。

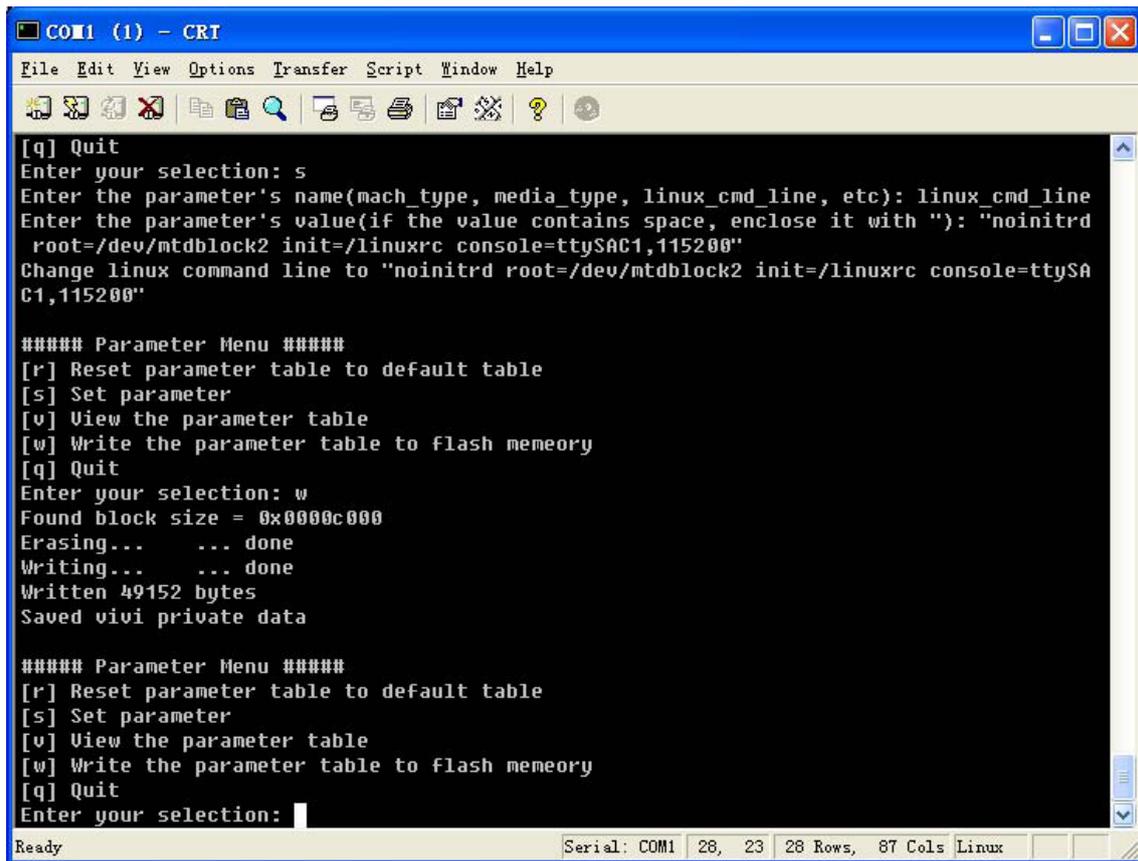


※ USBドライバはバグがあります。ARM9ボードが再起動、又はリセットの時、ホスト側は死んだかもしれません。その原因で、ARM9ボードが起動完了した後、USBケーブルでホストを繋ぎます。

8.3 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 の中にエラーエリアがあるかもしれません。使用の影響がありません。



```
COM1 (1) - CRT
File Edit View Options Transfer Script Window Help
[q] Quit
Enter your selection: s
Enter the parameter's name(mach_type, media_type, linux_cmd_line, etc): linux_cmd_line
Enter the parameter's value(if the value contains space, enclose it with "): "noinitrd
root=/dev/mtdblock2 init=/linuxrc console=ttySAC1,115200"
Change linux command line to "noinitrd root=/dev/mtdblock2 init=/linuxrc console=ttySA
C1,115200"

##### Parameter Menu #####
[r] Reset parameter table to default table
[s] Set parameter
[v] View the parameter table
[w] Write the parameter table to flash memory
[q] Quit
Enter your selection: w
Found block size = 0x0000c000
Erasing... .. done
Writing... .. done
Written 49152 bytes
Saved vivi private data

##### Parameter Menu #####
[r] Reset parameter table to default table
[s] Set parameter
[v] View the parameter table
[w] Write the parameter table to flash memory
[q] Quit
Enter your selection: 
```

8.4 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。



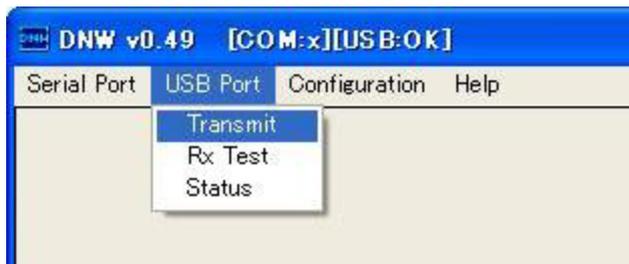
DNW のタイトルに[USB: OK]があれば、パソコンと ARM9 ボードを USB で繋ぎました。



メニューの中で、機能号[v]を選択して、

```
##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

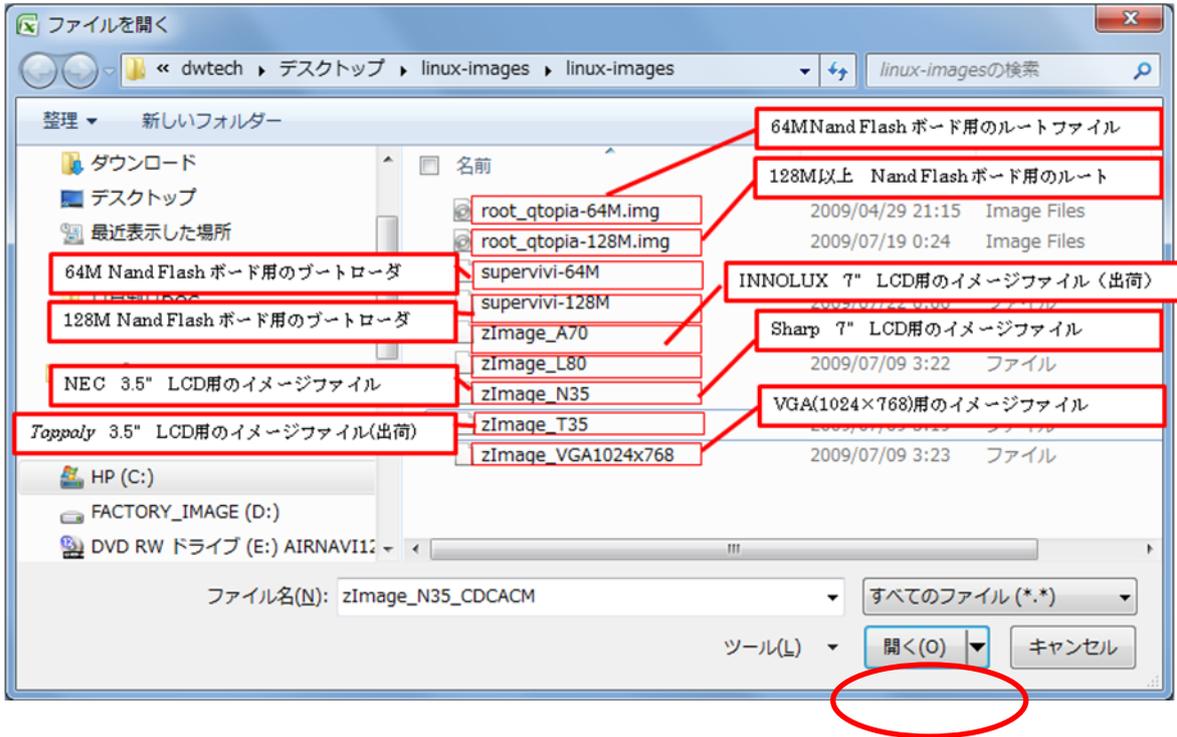
DNW を待っています。DNW のメニュー「USB Port」→「Transmit」を選択して、



ブートロード `supervivi_mini2440` を選択して、「開く」を押します。

*Mini2440 セットの場合：LCD 出荷状態：Toppoly 3.5" LCD

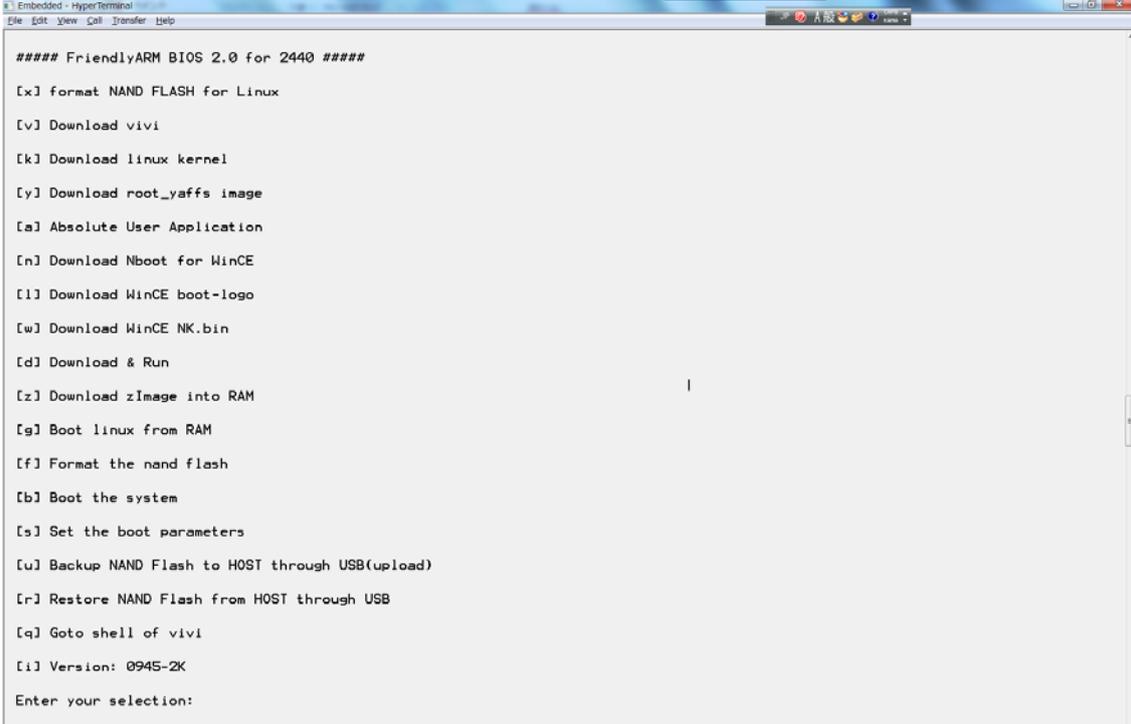
*Micro2440 セットの場合：LCD 出荷状態：INNOLUX 7" LCD



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。

8.5 Linux のカーネルの書き込み

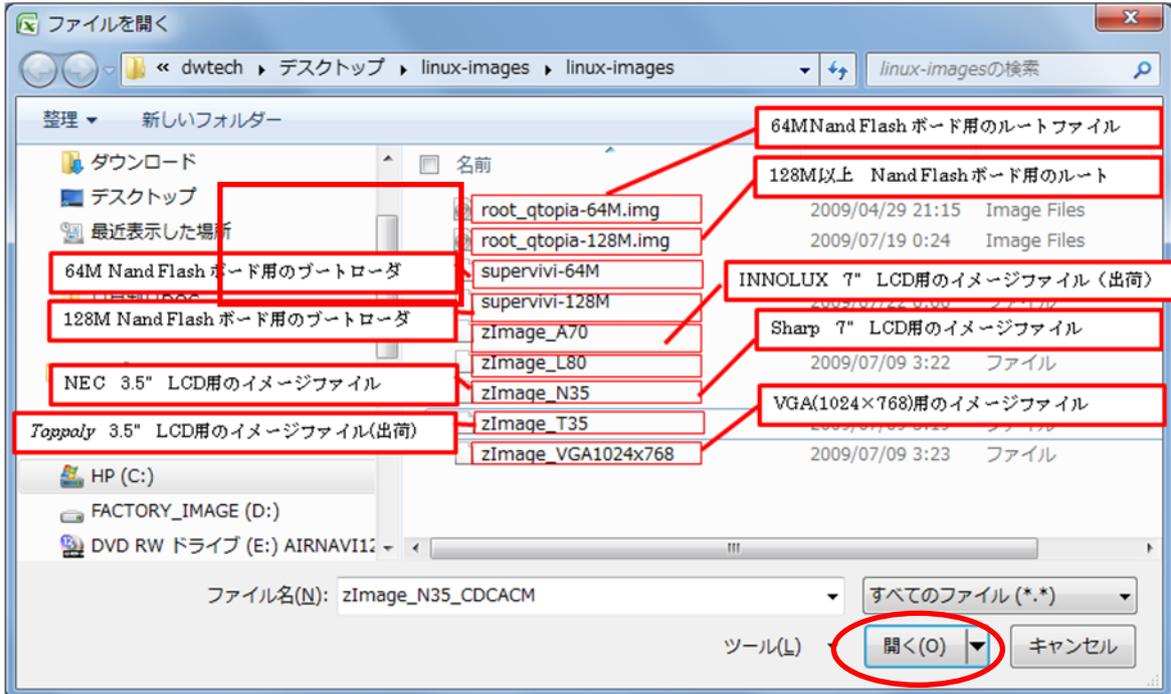
a. メニューの中で、機能号[k]を選択して、



```
##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

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、Linux カーネルファイル zImage_XXX を選択して、「開く」を押します。

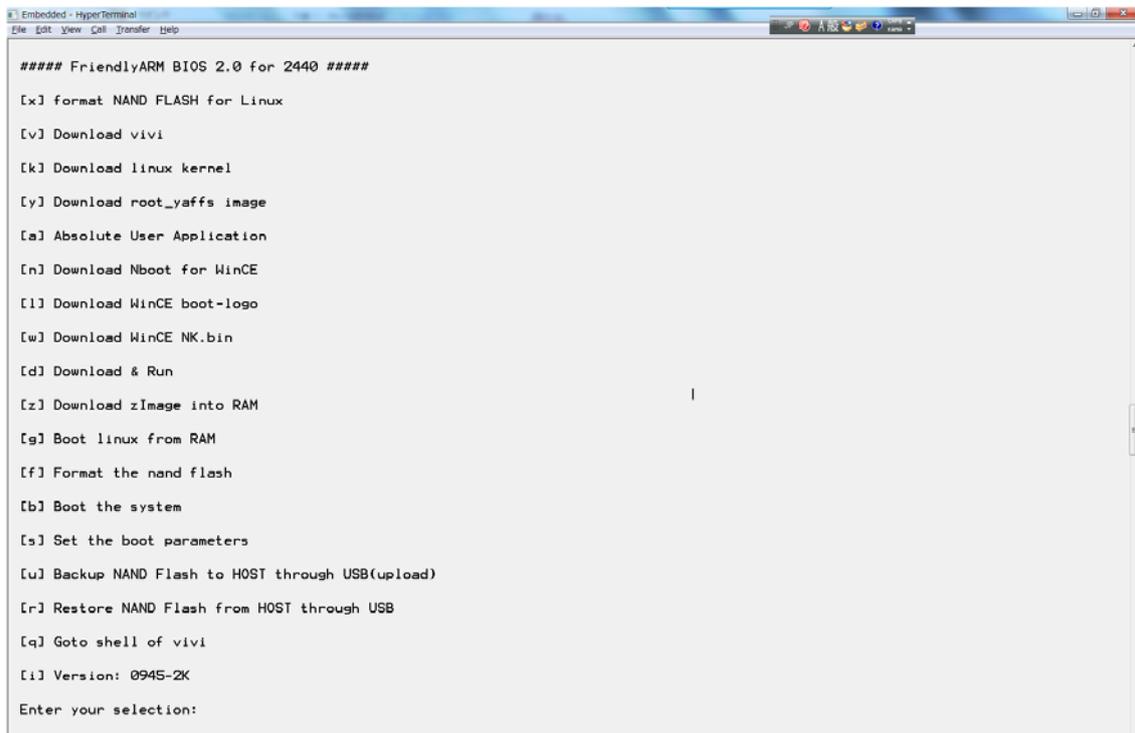


zImage_A70 7インチ液晶カーネル
zImage_T35 3.5インチ液晶カーネル
zImage_VGA1024X768 VGA カーネル

c. 転送完了したら、自動的にメニューに戻ります。

8.6 ルート・ファイルシステムの書き込み

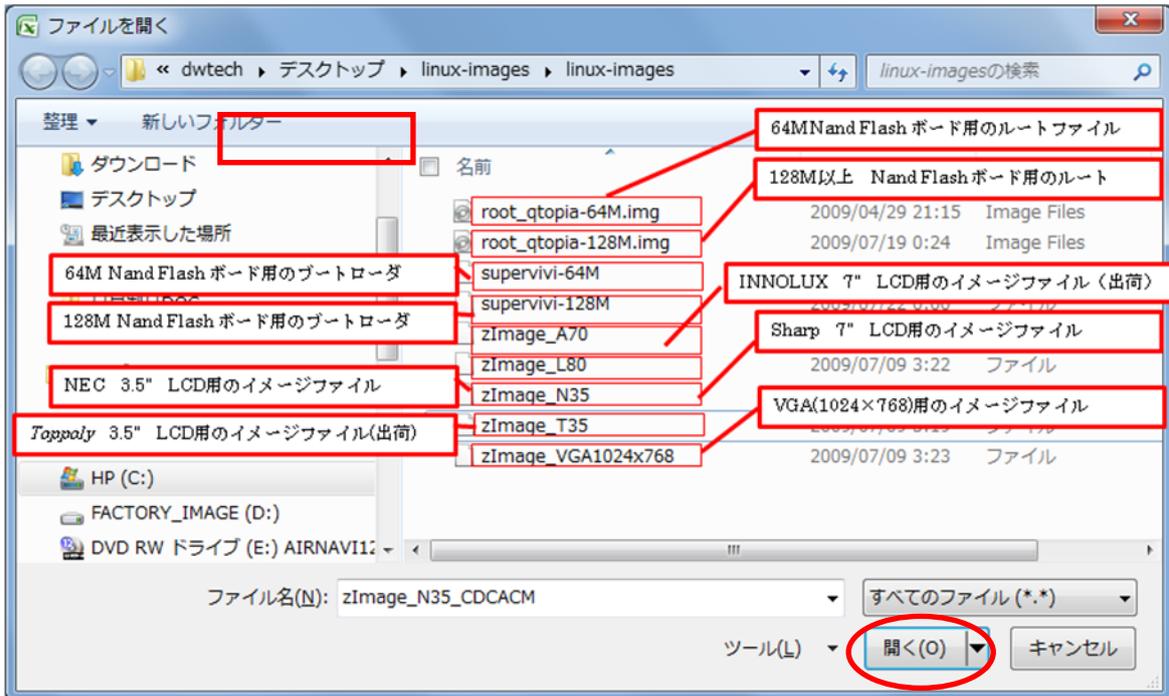
a. メニューの中で、機能号[y]を選択して、



```
##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

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_qtopia.img を選択して、「開く」を押します。



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、mini2440 の起動 S2 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。

8.7 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、

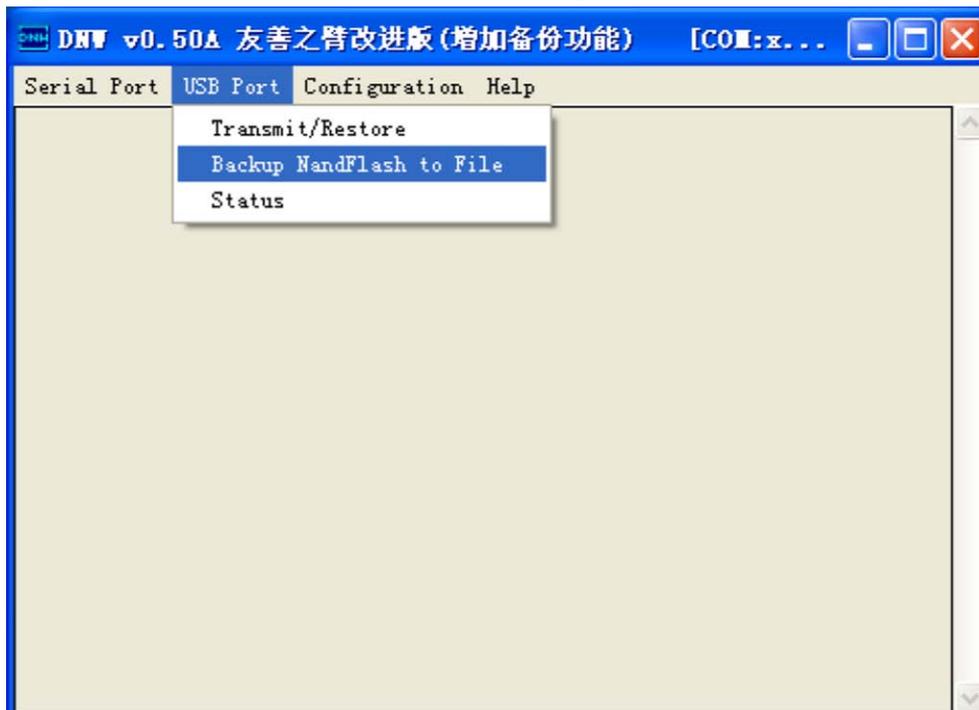


```
Embedded - HyperTerminal
File Edit View Call Transfer Help

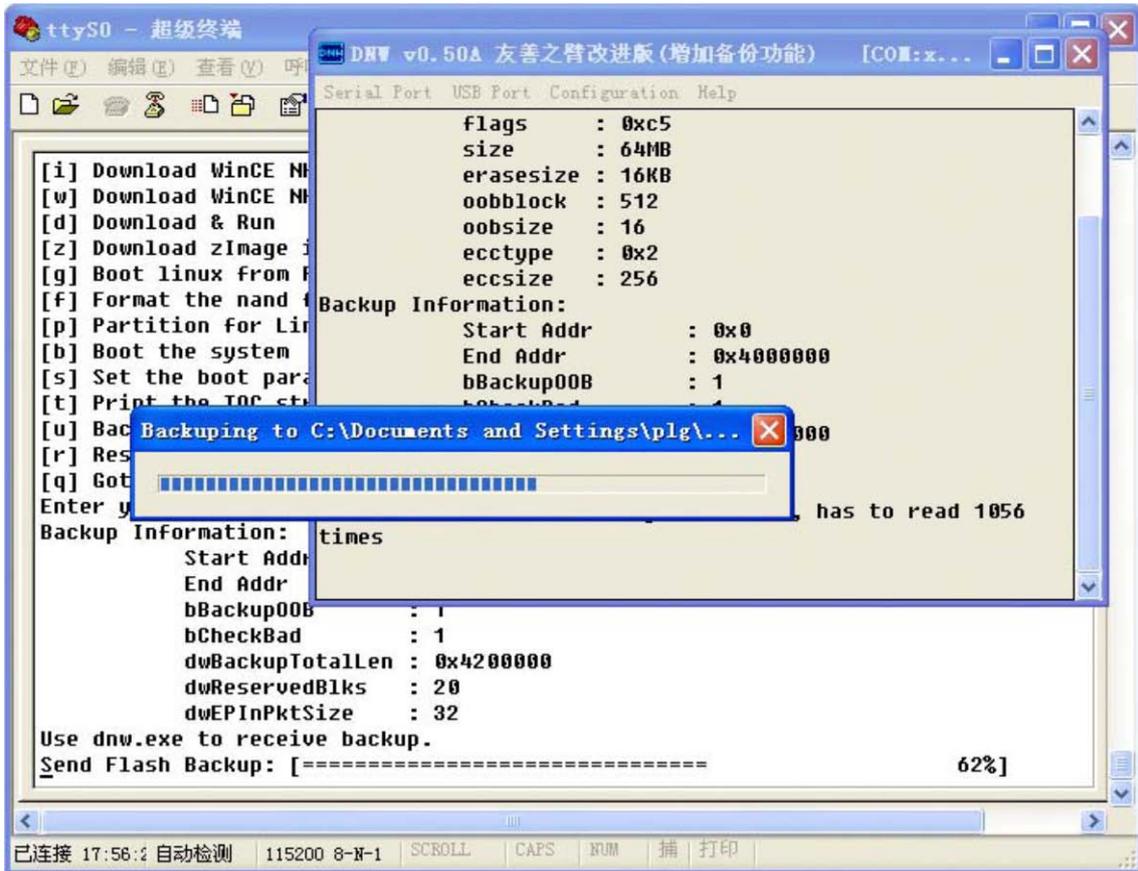
##### FriendlyARM BIOS 2.0 for 2440 #####

[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

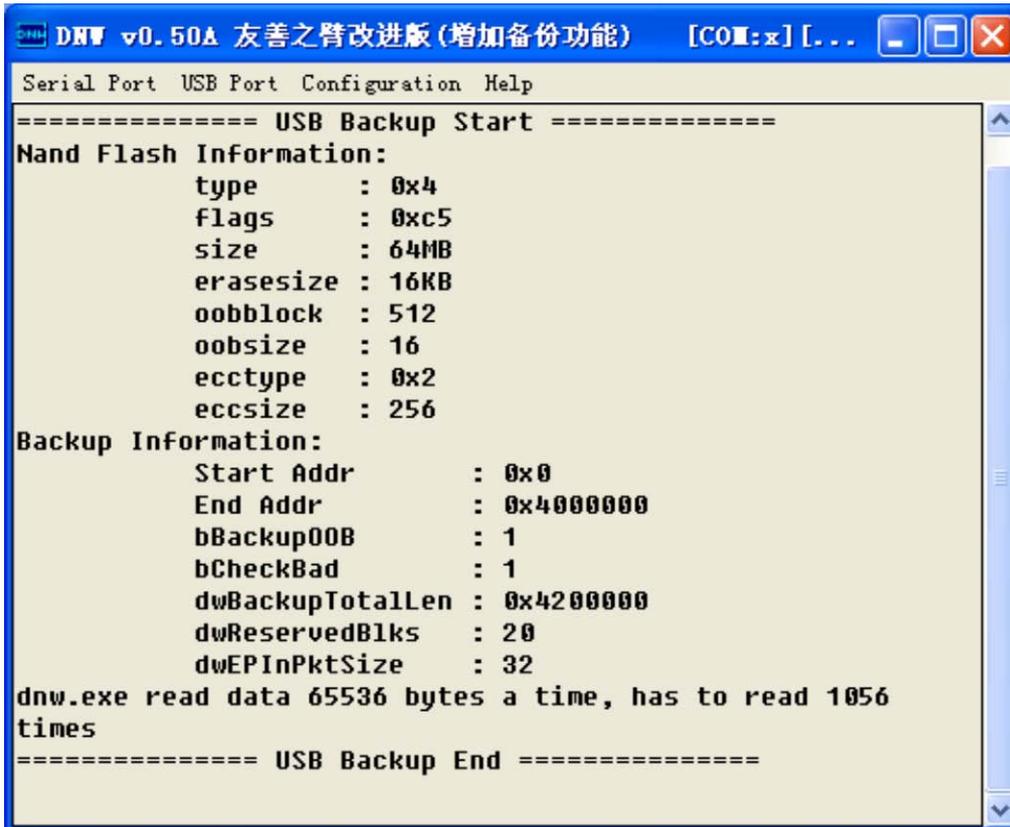
DNW のメニュー「Usb Port」 → 「Backup NandFlash to File」 を選択します。



バックアップのファイルの名前「backup.bin」を入力して



バックアップ完了したら、次の画面：



```
DNW v0.50A 友善之臂改进版(增加备份功能) [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      : 0x0  
  End Addr        : 0x4000000  
  bBackup00B     : 1  
  bCheckBad      : 1  
  dwBackupTotalLen : 0x4200000  
  dwReservedBlks : 20  
  dwEPInPktSize  : 32  
dnw.exe read data 65536 bytes a time, has to read 1056  
times  
===== USB Backup End =====
```

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。

8.8 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、



```
Embedded - HyperTerminal
File Edit View Call Transfer Help

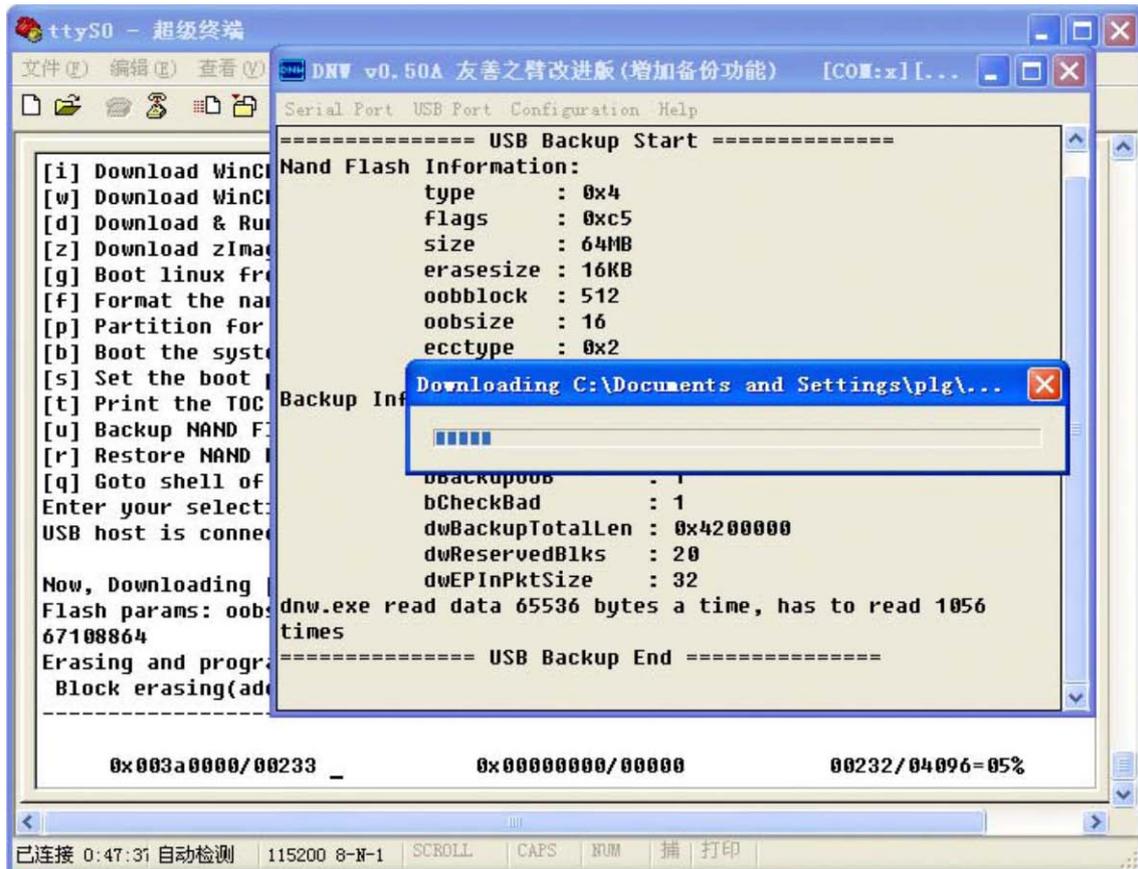
##### FriendlyARM BIOS 2.0 for 2440 #####

[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K
Enter your selection:
```

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

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

バックアップされたファイル「backup.bin」を選択します。



8.9 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 Supervivi 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、mini2440 の起動 S2 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、

```
Embedded - HyperTerminal
File Edit View Call Transfer Help

##### FriendlyARM BIOS 2.0 for 2440 #####

[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
[i] Version: 0945-2K

Enter your selection:
```

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 - 超級终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)

s3c2410-sdi s3c2410-sdi: running at 198kHz (requested: 197kHz).
s3c2410-sdi s3c2410-sdi: s3cmci_request: no card
s3c2410-sdi s3c2410-sdi: powered down.
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 4096 (order: 3, 32768 bytes)
TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP reno registered
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
yaffs: dev is 32505858 name is "mtdblock2"
yaffs: Attempting MTD 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 0:18:24 ANSII 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:MINI2440.arm9.net:et
h0:off"    (NFSの設定)
```

```
Supervivi> boot ram    (メモリでカーネルを起動させます)
```

第九章 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。

9.1 簡易 JTAG で書き込み

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。

<http://www.hitag.com>

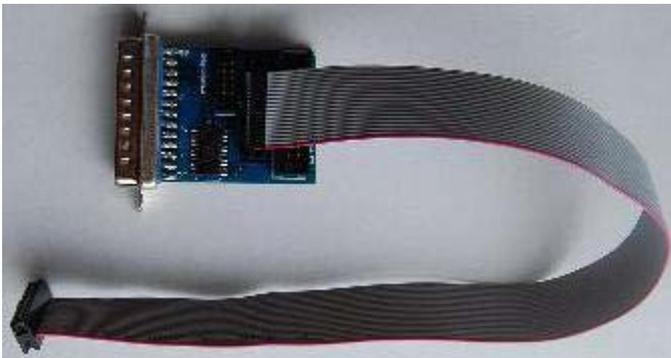
弊社は H-JTAG のハードウェアを提供しております。パソコンは LTP が必要です。

9.1.1 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 <http://www.hitag.com> 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

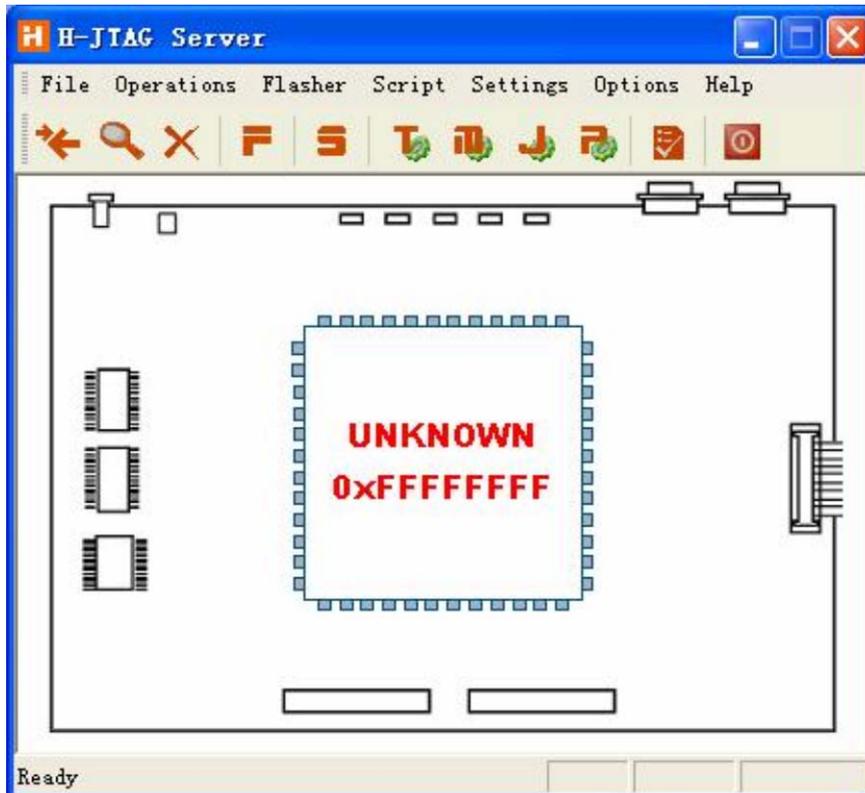


弊社は H-JTAG のハードウェアを提供しております。パソコンは LTP が必要です。

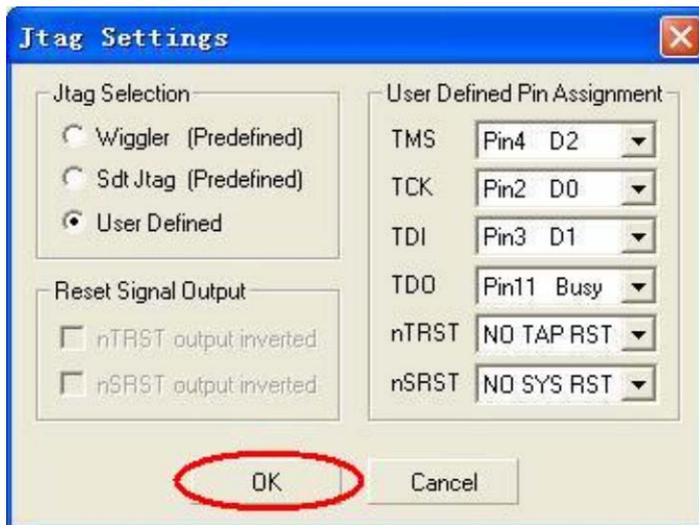
インストール完了すれば、デスクトップで H-JTAG と H-Flasher を生成します。H-JTAG を実行すると、このエラーメッセージが出てきます。



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。

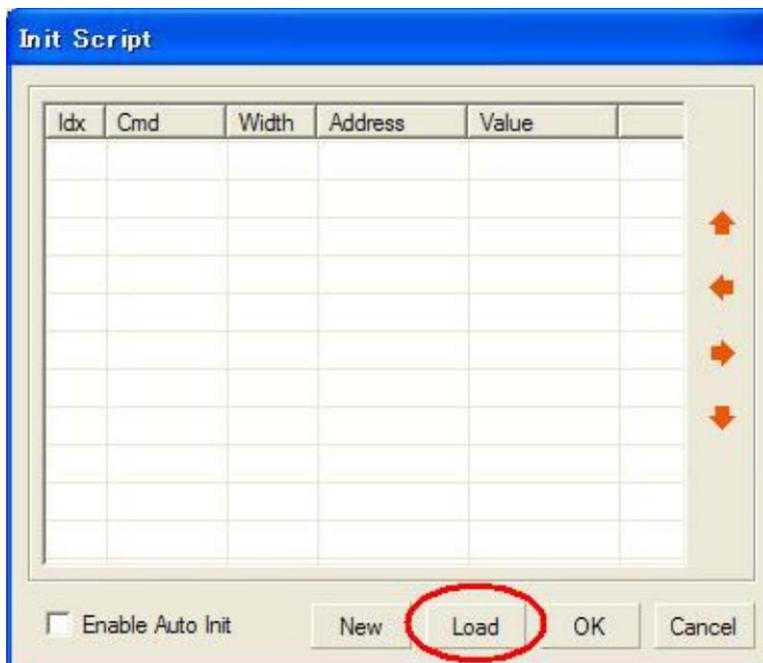


H-JTAG のメニュー : Setting → Jtag Settings

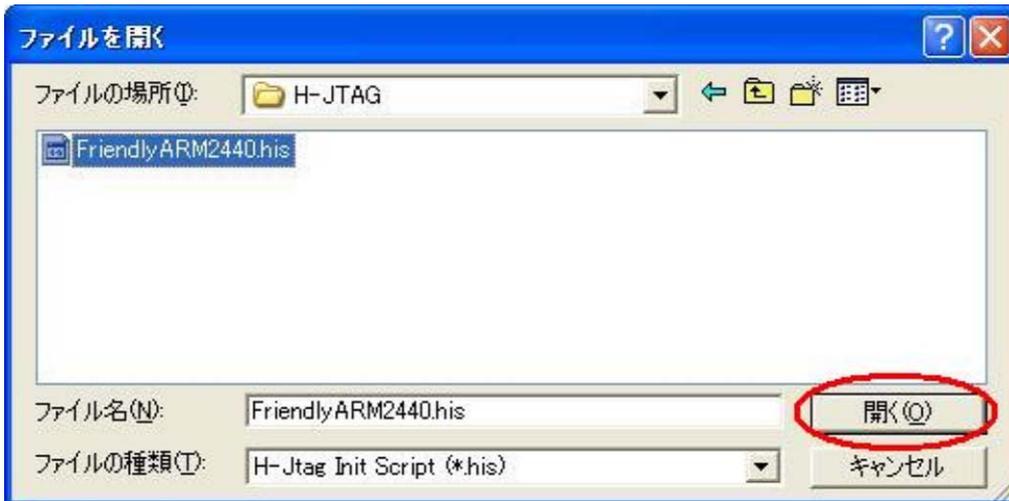


この様な設定して、"Ok"ボタンを押します。

H-JTAG のメニュー : Script → Init Script

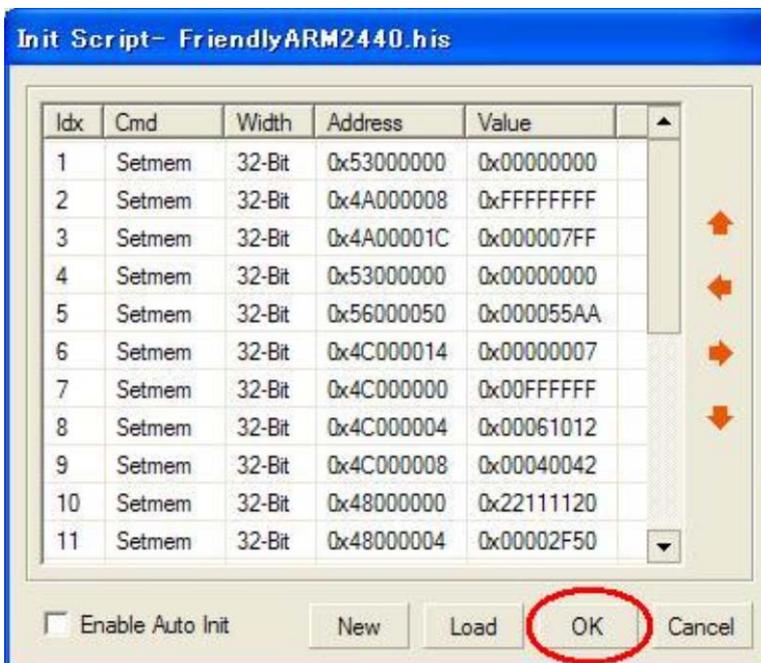


"Load"ボタンを押します。



FriendlyARM2440.his というファイルを選択します。

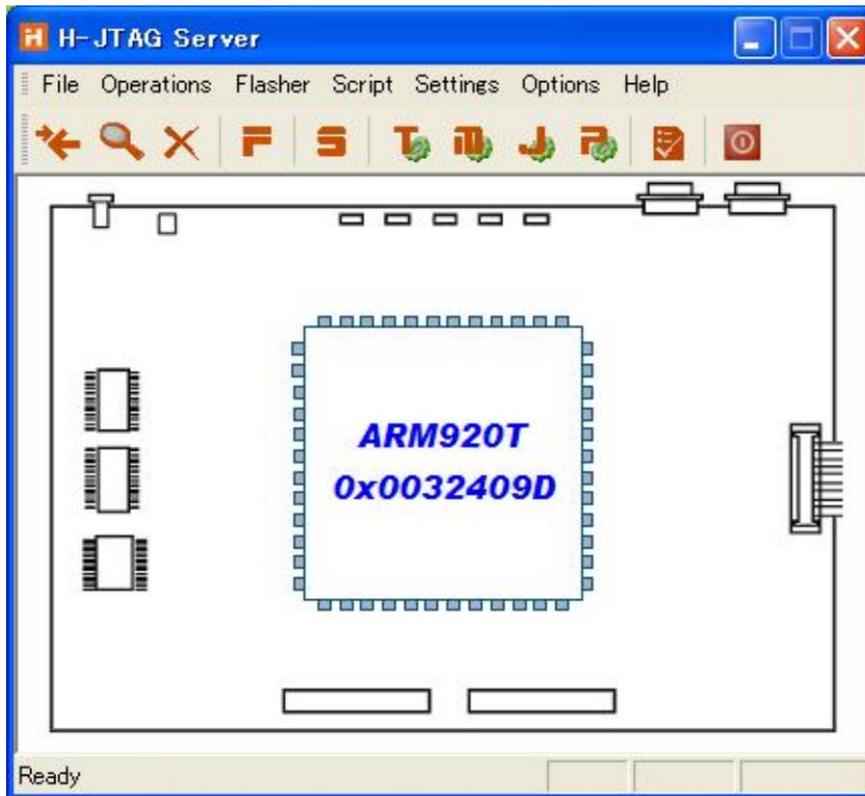
次の画面が出てきます。



"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 はターゲット ARM ボードを認識しました。

9.1.2 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。

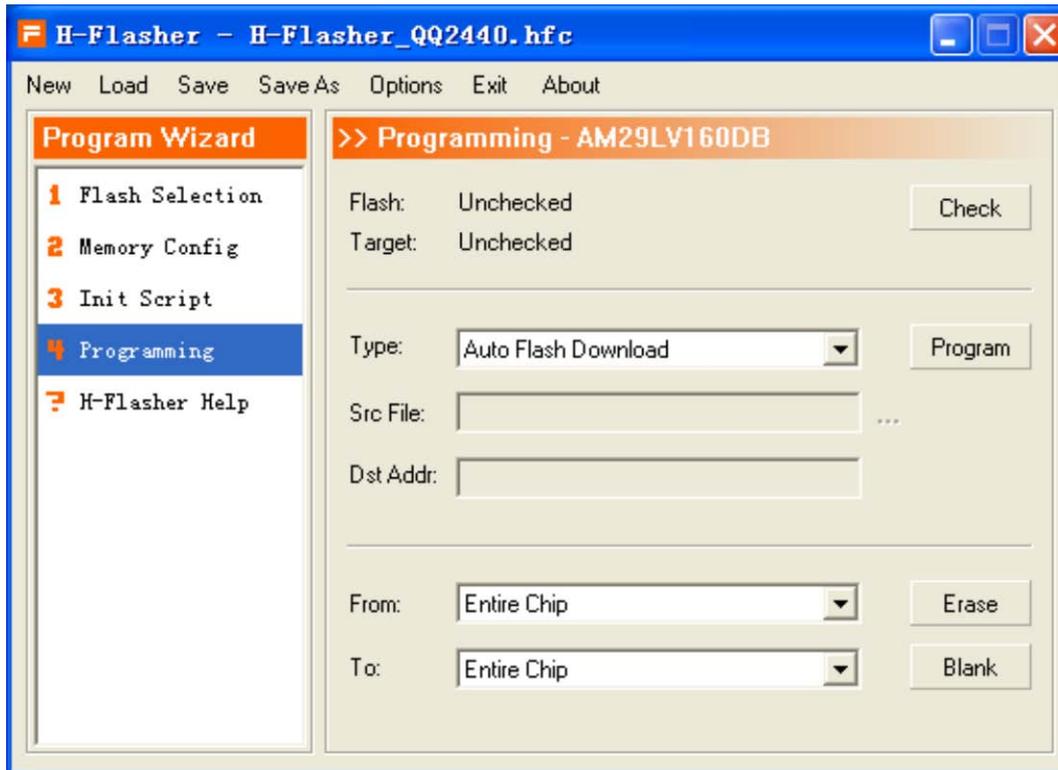
H-JTAG のメインメニュー「Flasher」 → 「Start H-Flasher」で H-Flasher を実行します。



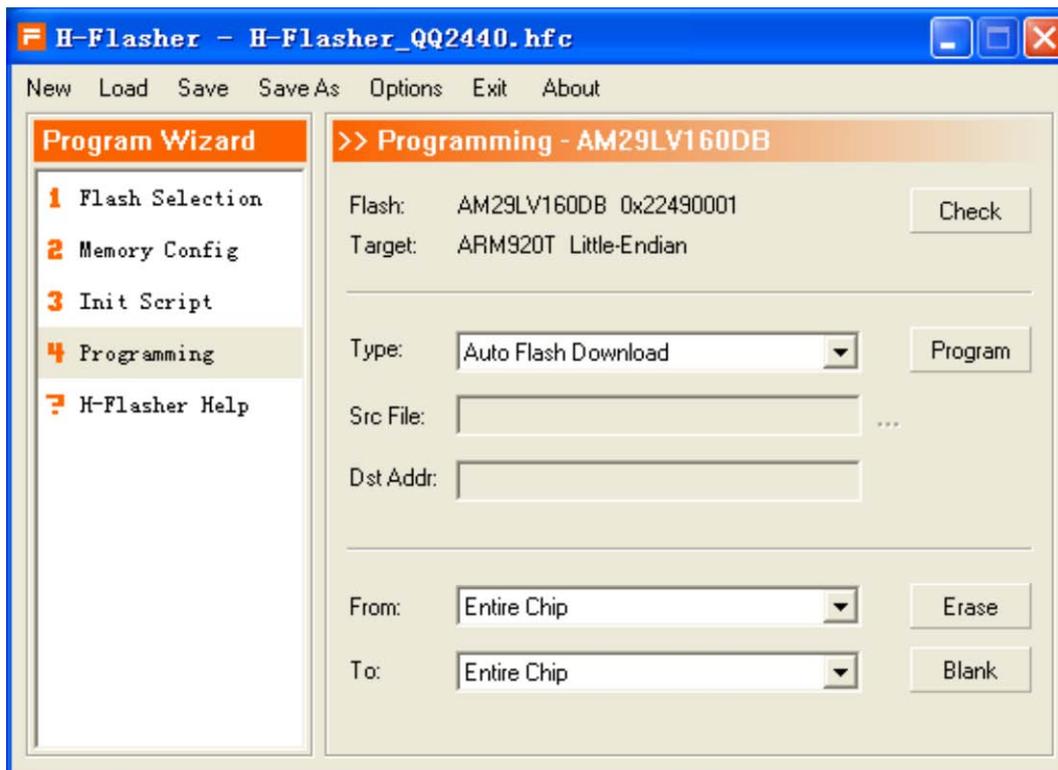
H-Flasherのメインメニュー「Load」、H-Flasher_mini2440.hfcというファイルを開きます。



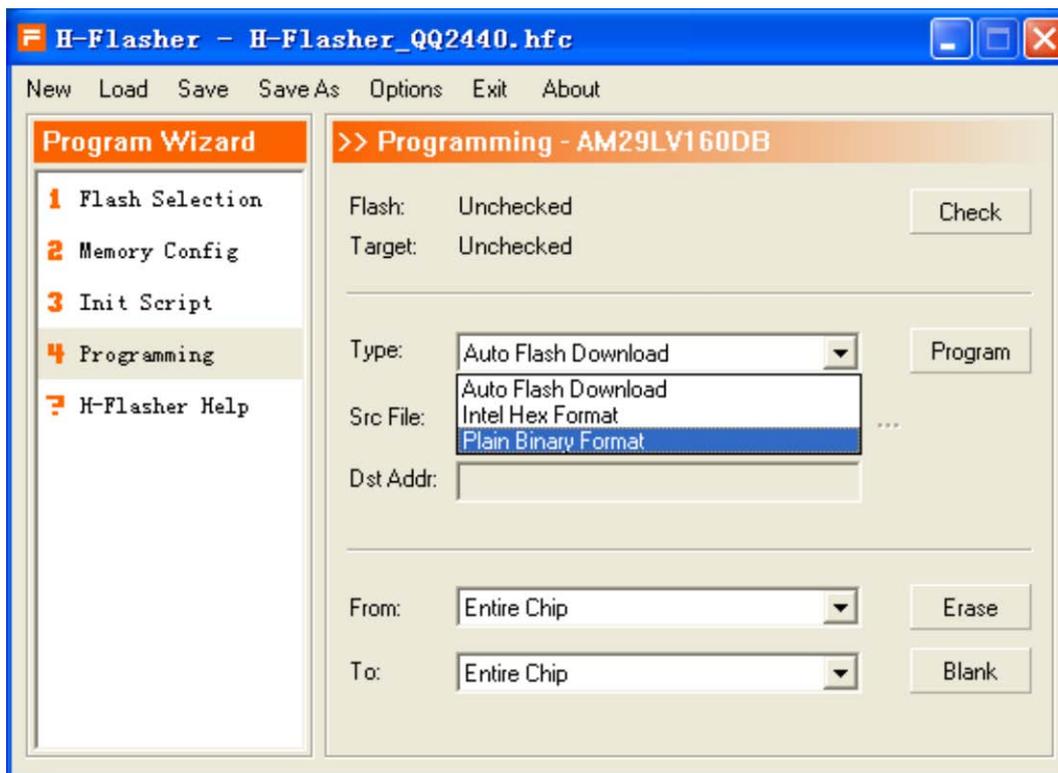
H-Flasherの左側の「4 Programming」を選択します。



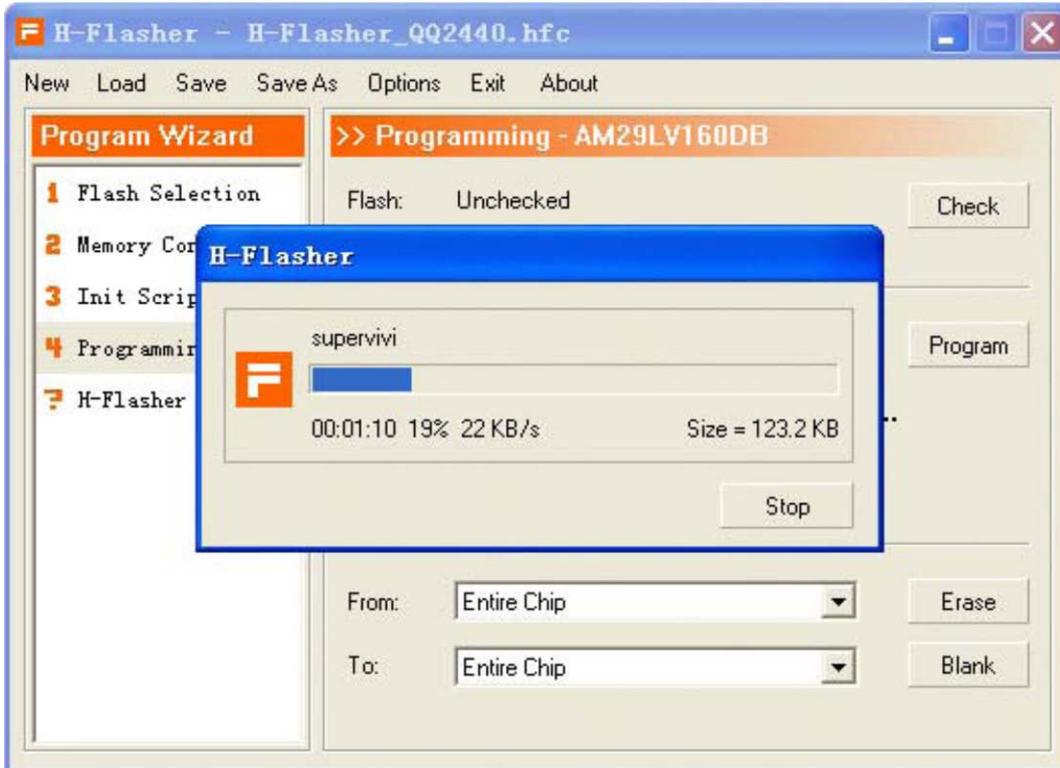
「Check」ボタンを押すと、mini2440が使用したNor Flash(AM29LV160DB)を発見します。



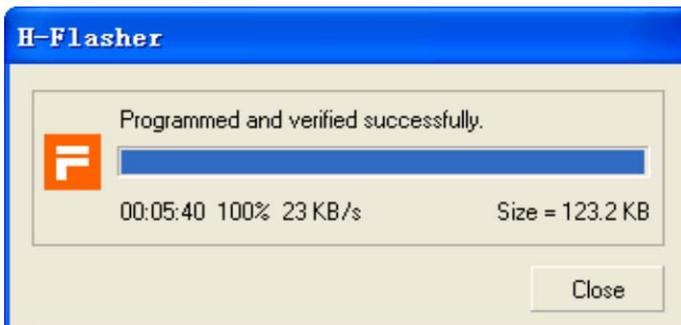
「Type」の「Plain Binary Format」を選択します。



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。



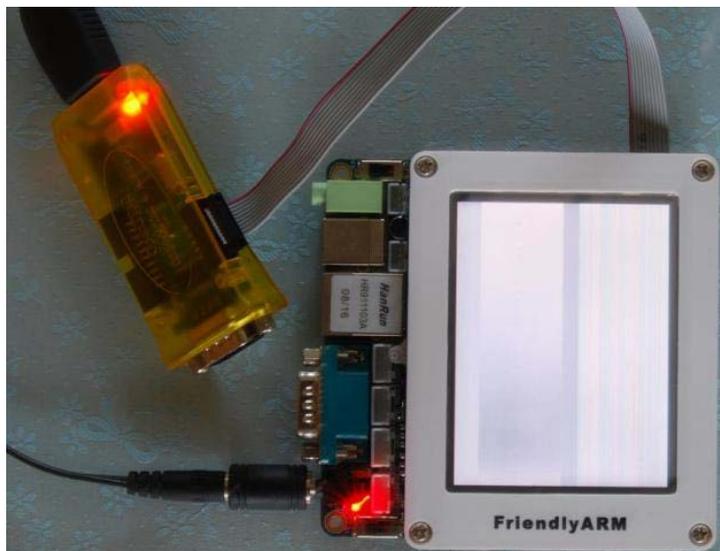
書き込み完了の画面：



9.2 Open-JTAG で書き込み



弊社が販売している Open-JTAG は ARM 用の USB-JTAG エミュレータです。ARM7、ARM9、Cortex-M3、XSCALE に対応、OpenOCD をサポートします。USB-RS232 機能もあります。COM と LPT ポートがないノートパソコンに最適。



Opne-JTAGで書き込み様子

9.2.1 ドライバをインストールする

※Windows 2000、X P用ドライバダウンロード URL :

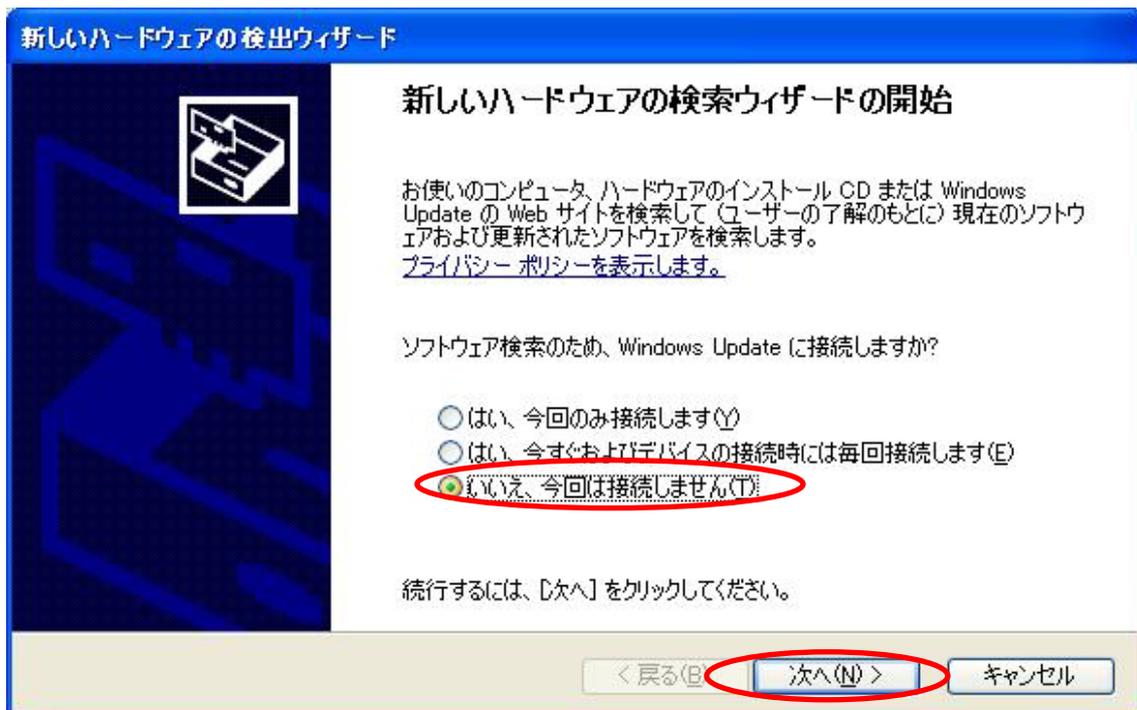
<http://www.dragonwake.com/download/open-jtag/open-jtag-driver.zip>

※Windows7 用ドライバダウンロード URL :

http://www.dragonwake.com/download/open-jtag/open-jtag-driver_win7.zip

「usb-drivers」フォルダーにある。

OpenJTAG をパソコンの USB ポートに挿入して、下の通りにドライバをインストールしてください。



新しいハードウェアの検出ウィザード



このウィザードでは、次のハードウェアに必要なソフトウェアをインストールします。

USB<=>JTAG&RS232



ハードウェアに付属のインストール CD またはフロッピー ディスクがある場合は、挿入してください。

インストール方法を選んでください。

ソフトウェアを自動的にインストールする (推奨) (A)

二覧または特定の場所からインストールする (詳細) (S)

続行するには、[次へ] をクリックしてください。

< 戻る (B)

次へ (N) >

キャンセル

新しいハードウェアの検出ウィザード

検索とインストールのオプションを選んでください。



次の場所で最適なドライバを検索する (S)

下のチェック ボックスを使って、リムーバブル メディアやローカル パスから検索できます。検索された最適なドライバがインストールされます。

リムーバブル メディア (フロッピー、CD-ROM など) を検索 (M)

次の場所を含める (Q):

C:\%OpenJTAG#\usb-driver

参照 (R)

検索しないで、インストールするドライバを選択する (D)

一覧からドライバを選択するには、このオプションを選びます。選択されたドライバは、ハードウェアに最適なものとは限りません。

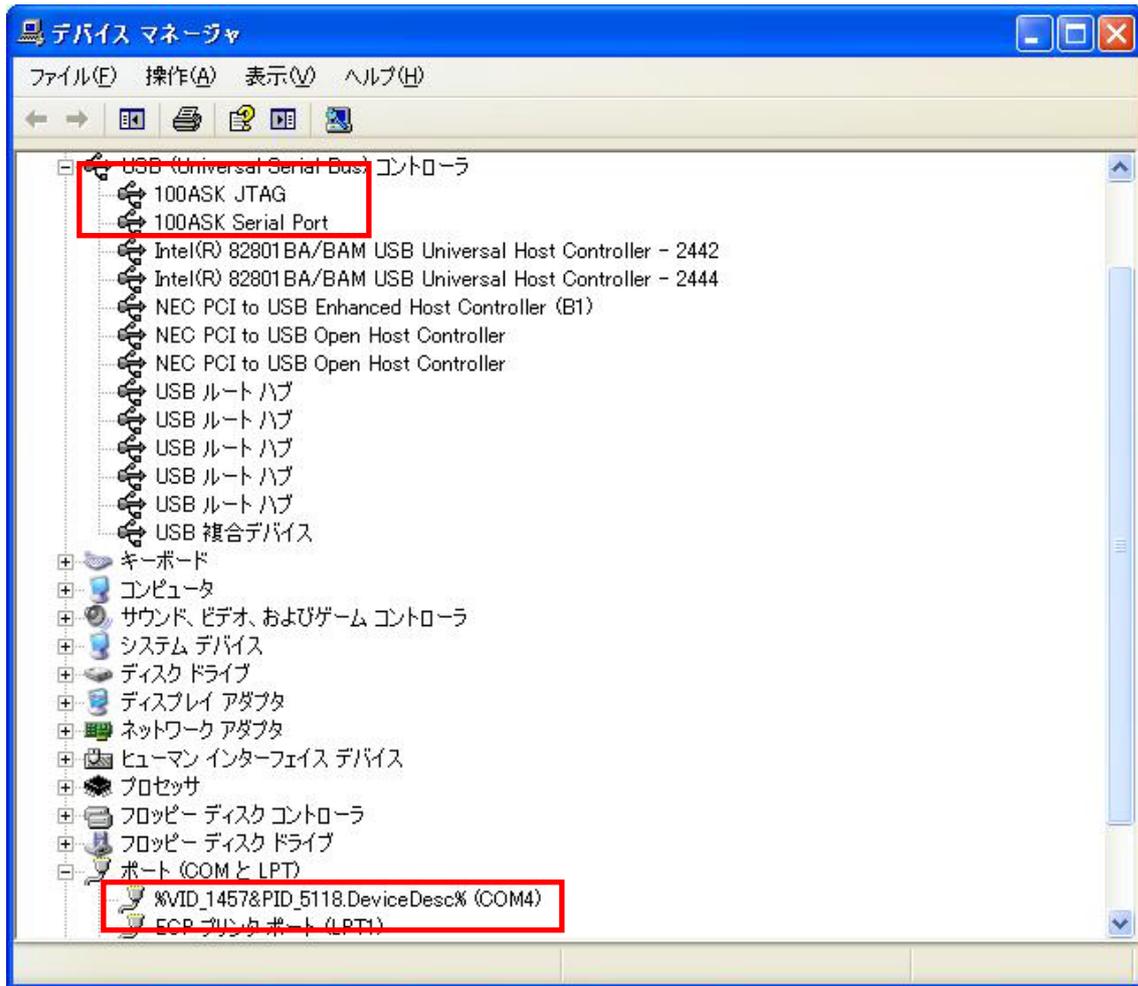
< 戻る (B)

次へ (N) >

キャンセル



USB ドライバのインストールは3回があります。インストール完了すると、デバイスマネージャで三つのデバイスが見えます。



※ OpenJTAG は USB シリアルポートとして使えます。

9.2.2 書き込み

※書き込みツールダウンロード URL :

上記 USB ドライバダウンロード URL と同じ

※Windows 2000、X P 用ドライバダウンロード URL :

<http://www.dragonwake.com/download/open-jtag/open-jtag-driver.zip>

※Windows7 用ドライバダウンロード URL :

http://www.dragonwake.com/download/open-jtag/open-jtag-driver_win7.zip

「tools」フォルダーにある。

※ブートロードファイルダウンロード URL :

<http://www.dragonwake.com/download/arm9-download/linux-2.6.32.2/linux-images.tgz>

解凍後、色んな種類のボード用のブートロードファイルがありますが、ボードの種類により、正しいファイルを選んでください。(ファイルの説明は中身の「Readme.txt」を参照)



C:\¥openJTAG¥open-jtag>**sjf24x0_ft2232.exe supervivi_mini2440**

```
+-----+
|   Flash Programmer for OpenJTAG of www.100ask.net   |
|   OpenJTAG is a USB to JTAG & RS232 tool based FT2232 |
|   This programmer supports both of S3C2410X & S3C2440 |
|   Author: Email/MSN(thisway.diy@163.com), QQ(17653039) |
+-----+
```

Usage: sjf24x0_ft2232.exe [filename]

Select the CPU:

- 1. S3C2410X
- 2. S3C2440X**

Enter the number: 2

S3C24X0 detected, cpuID = 0x0032409d

[Main Menu]

0:Nand Flash prog **1:Nor Flash prog** 2:Memory Rd/Wr 3:Exit

Select the function to test: 1

Detect Nor Flash ...

SST 39VF1601

Size: 2 MB

Image Size: 0x1f314

～略～

Erasing done

write ...

100% done

第十章 Web カメラストリーミング配信

10.1 MJPG-streamerのダウンロードとコンパイル

mjpg-streamerの最新バージョンをダウンロードします。

```
$ svn co https://mjpg-streamer.svn.sourceforge.net/svnroot/mjpg-streamer
mjpg-streamer
```

```
$ cd mjpg-streamer/mjpg-streamer
```

```
$ make CC=arm-linux-gcc
```

成功すれば実行ファイルmjpg_streamerとライブラリ・ファイル*.soを生成します。

10.2 MJPG-streamer を mini/micro2440 ボードにインストール

生成されたファイル(mjpg_streamer、*.so)とmjpg-streamerのwwwディレクトリをmini/micro2440ボードにダウンロードして、特定のディレクトリにコピーしてください。

```
# mkdir -p /usr/lib/
```

```
# cp *.so /usr/lib/
```

```
# cp mjpg_streamer /usr/bin/
```

```
# mv www /www/uvc-www
```



弊社が販売している UVC(USB Device Class)に対応した Web カメラを mini/micro2440 の USB ホストに接続して、mini/micro2440 のコンソールで次のコマンドで MJPG-streamer を起動させます。

```
# mjpg_streamer --input "input_uvc.so" --device /dev/video0 --fps 5 --resolution 640x480 --yuv" --output "output_http.so" --port 8080 --www /www/uvc-www"
```

```
MJPEG Streamer Version.: 2.0
```

```
i: Using V4L2 device.: /dev/video0
```

```
i: Desired Resolution: 640 x 480
```

```
i: Frames Per Second.: 5
```

```
i: Format.....: YUV
i: JPEG Quality.....: 80
format asked unavailable get width 352 height 288
o: www-folder-path....: /www/uvc-www/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: enabled
```

- ※ UVC カメラによって、“-yuv”というパラメータは不要の可能性もあるかもしれません
- ※ UVC 以外のカメラは [input_gspcav1.so](#) を使います。

10.3 Web ブラウザで Web カメラを見ましょう

Web ブラウザで、「<http://mini/micro2440> ボードの IP アドレス:8080/」にアクセスすると、MJPEG-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 Explorer 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:



第十一章 Eclipse + GCC + Open-JTAG

11.1 GCC ツールチェーン

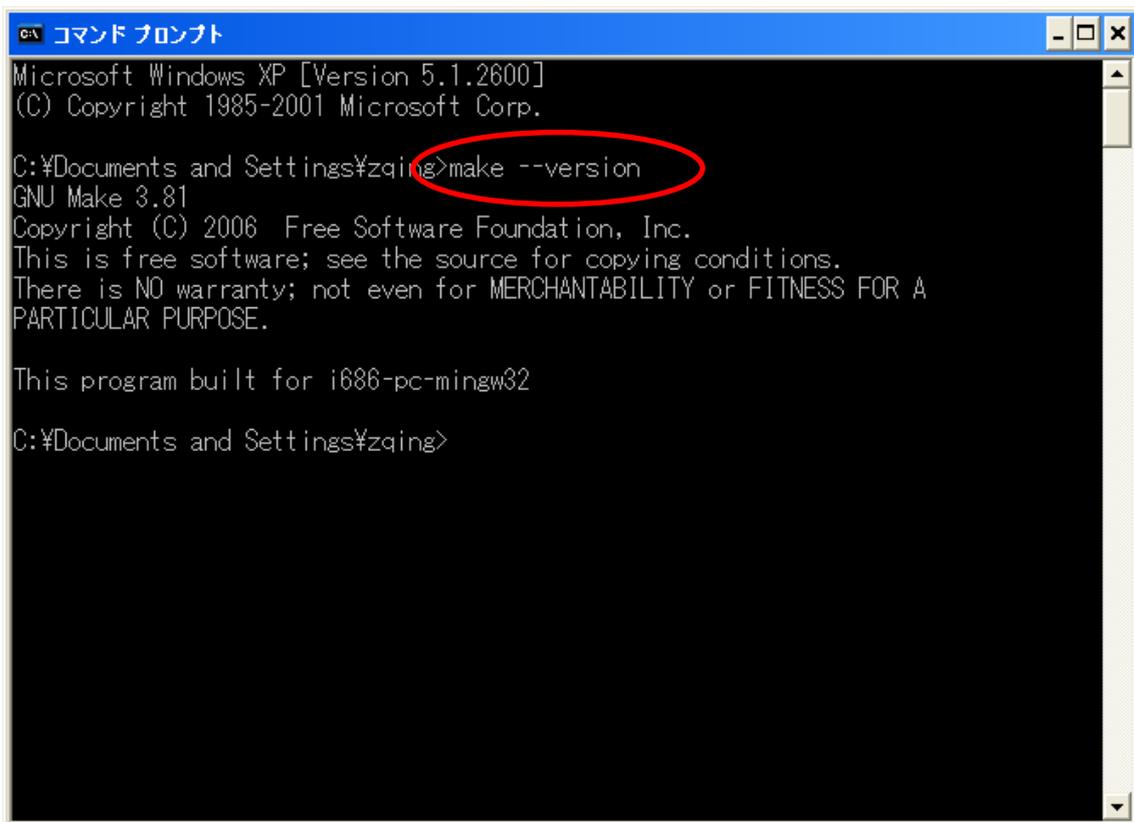
各種ユーティリティ :

<http://www.yagarto.de/download/yagarto/yagarto-tools-20070303-setup.exe>

GCC ツールチェーン

http://sourceforge.net/projects/yagarto/files/YAGARTO%20for%20Windows/yagarto-build-1.19.1_gcc-4.3.3-c-c%2B%2B-1.17.0_gi-6.8.50_20090329.exe/download

インストールが出来たら `make` の確認をするためコマンドプロンプトを起動し、右記のコマンドを入力します(`make --version`)。画面に下記のメッセージが出てくればOKです。



```
コマンド プロンプト
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

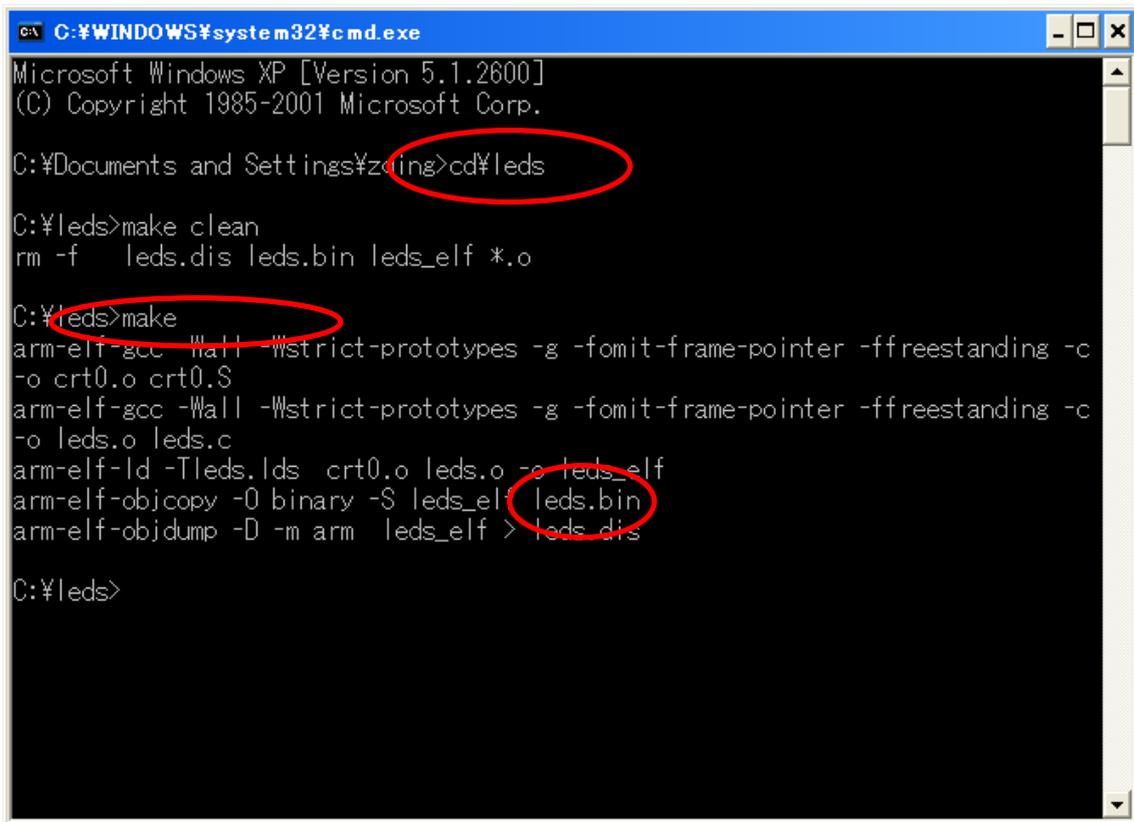
C:\Documents and Settings\%username%>make --version
GNU Make 3.81
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i686-pc-mingw32

C:\Documents and Settings\%username%>
```

サンプルのコンパイル：

1. コマンドプロンプトでディレクトリを移動
(cd¥leds)
2. 下記のコマンドを入力します
(make)



```
C:\¥WINDOWS¥system32¥cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\¥Documents and Settings¥zoring>cd¥leds

C:\¥leds>make clean
rm -f leds.dis leds.bin leds_elf *.o

C:\¥leds>make
arm-elf-gcc -Wall -Wstrict-prototypes -g -fomit-frame-pointer -ffreestanding -c
-o crt0.o crt0.S
arm-elf-gcc -Wall -Wstrict-prototypes -g -fomit-frame-pointer -ffreestanding -c
-o leds.o leds.c
arm-elf-ld -Tleds.lds crt0.o leds.o -o leds_elf
arm-elf-objcopy -O binary -S leds_elf leds.bin
arm-elf-objdump -D -m arm leds_elf > leds.dis

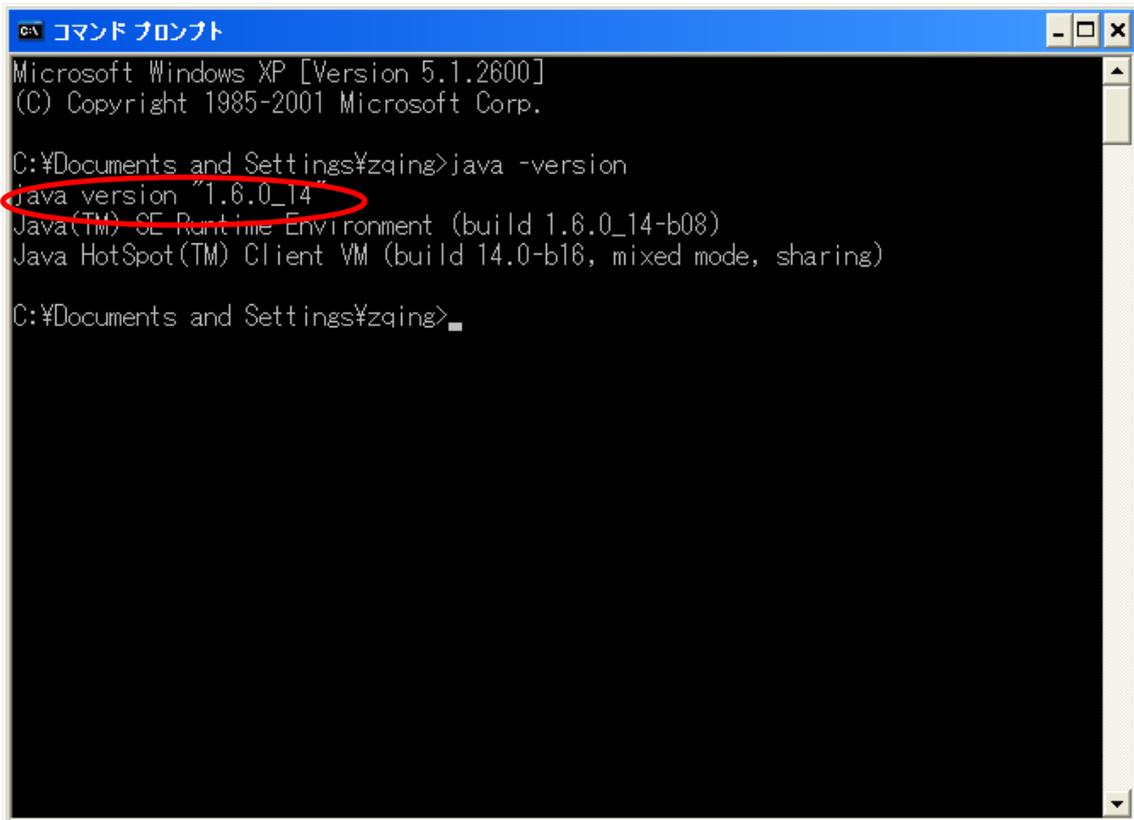
C:\¥leds>
```

コンパイル成功したら、*.bin ファイルを生成させます。

11.2 Integrated Development Environment(Eclipse)

JRE バージョン確認：

確認コマンド： `java -version`



```
コマンド プロンプト
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\yzqing>java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)

C:\Documents and Settings\yzqing>
```

JRE がなければ、あるいは 1.4.2 以下なら、JRE のインストールが必要です。

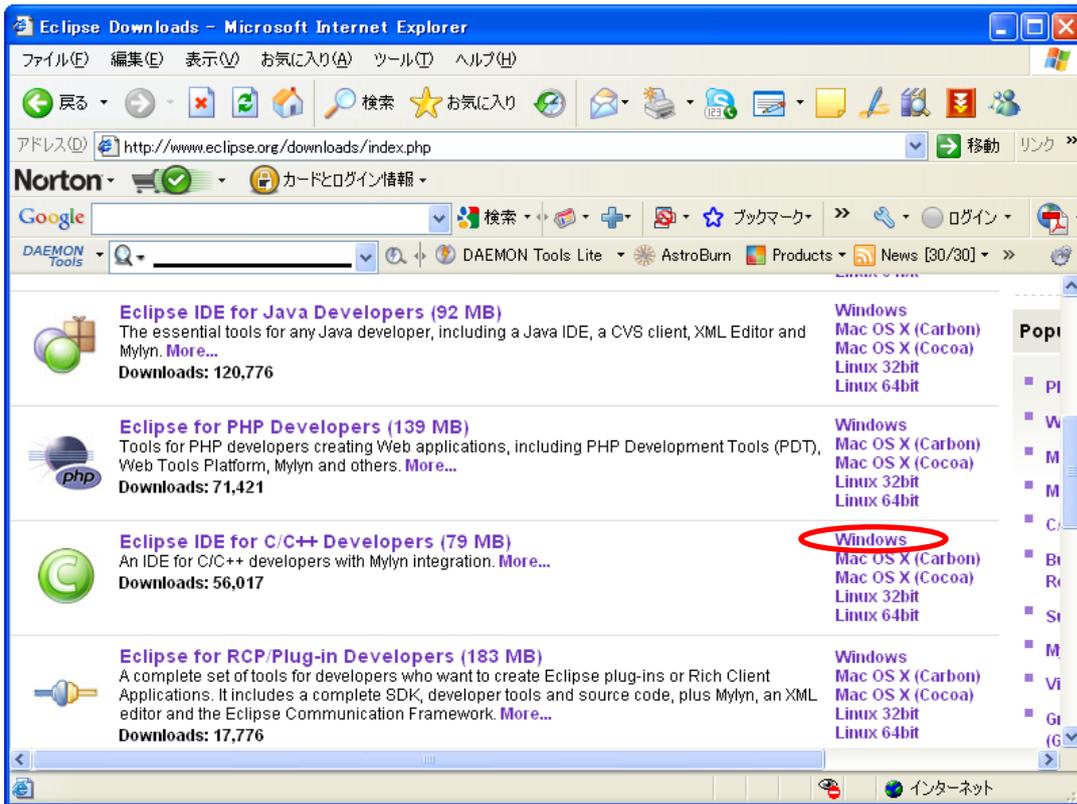
<http://java.sun.com/javase/downloads/index.jsp>

Eclipse のインストール：

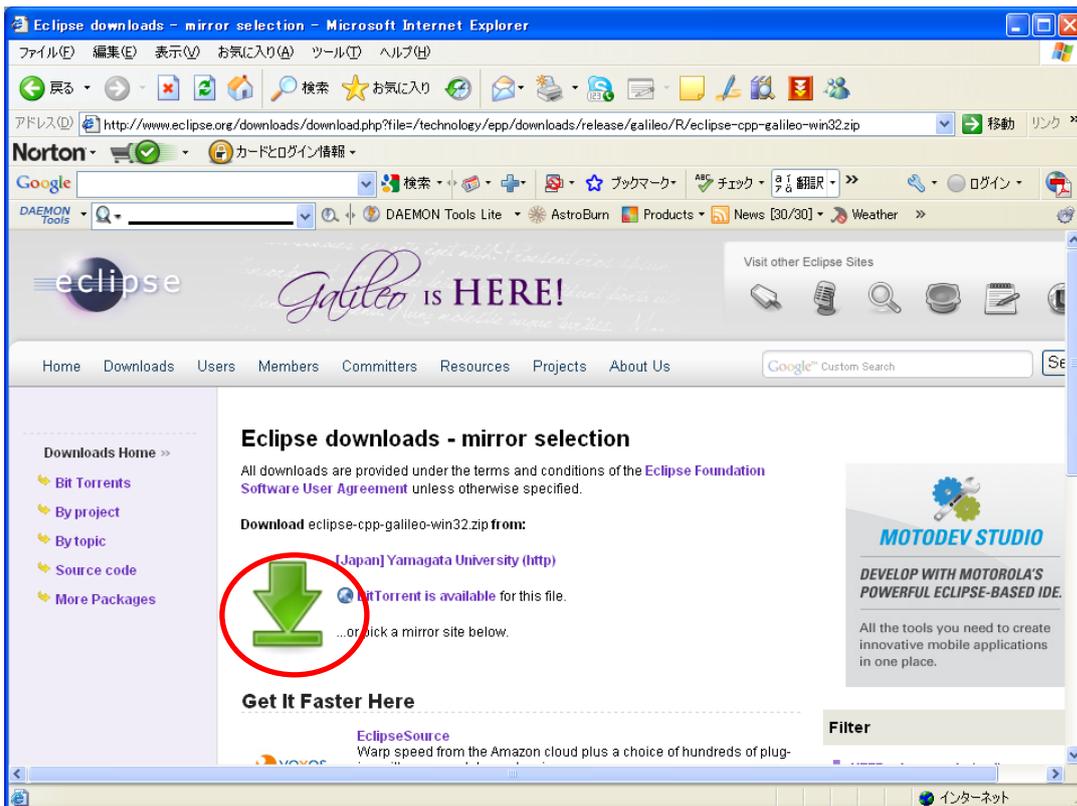
1) 下記のリンクをクリック

<http://www.eclipse.org/downloads/index.php>

2) Eclipse IDE for C/C++ Developers(79MB)の Windows をクリック

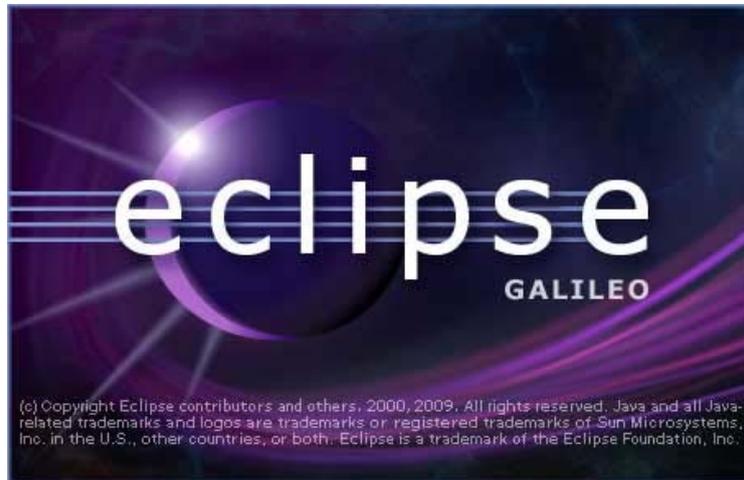


3) 画面の下矢印をクリックしダウンロード

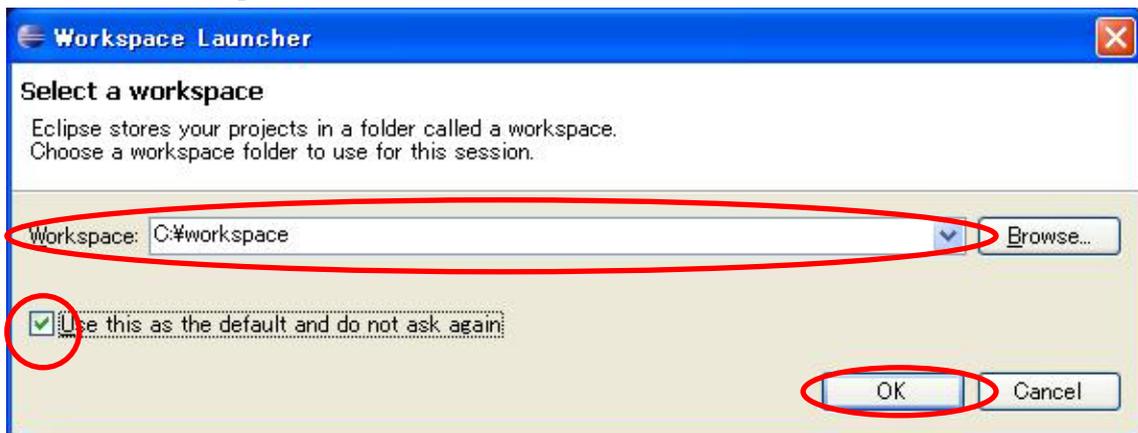


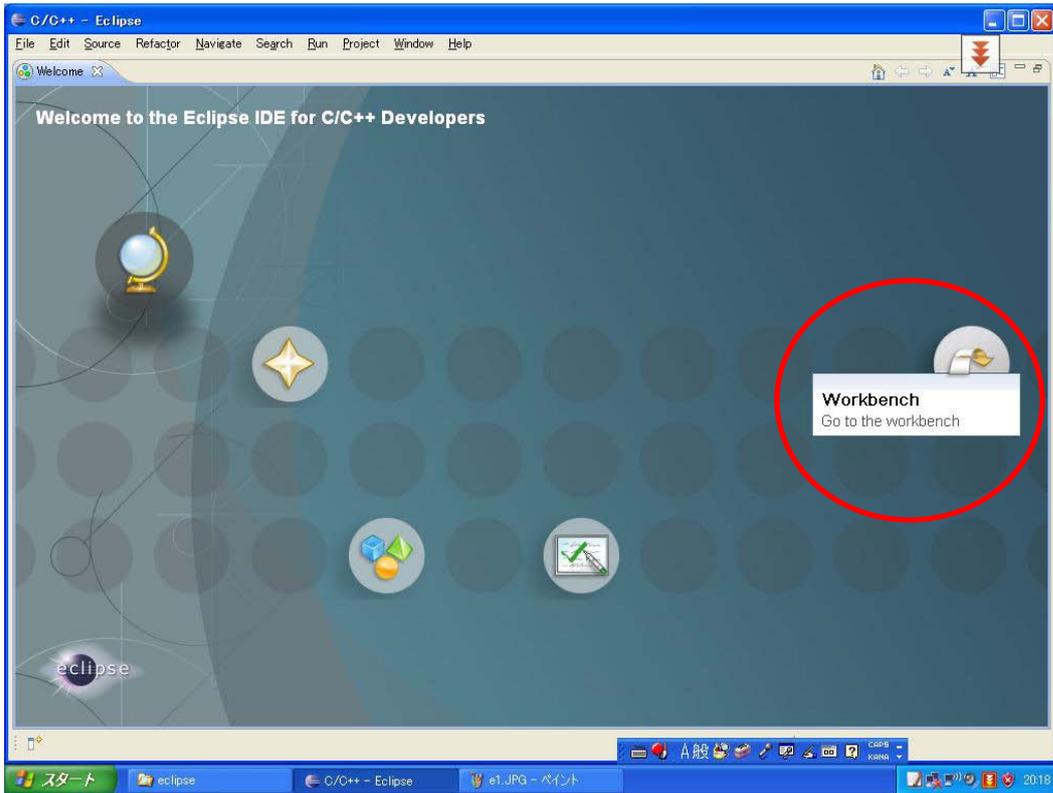
4) ダウンロードしたファイル"eclipse-cpp-galileo-win32.zip"を解凍し、そのなかの"eclipse"フォルダを適当な場所(C:\eclipse)へ移動する。

5) Eclipse を起動する。

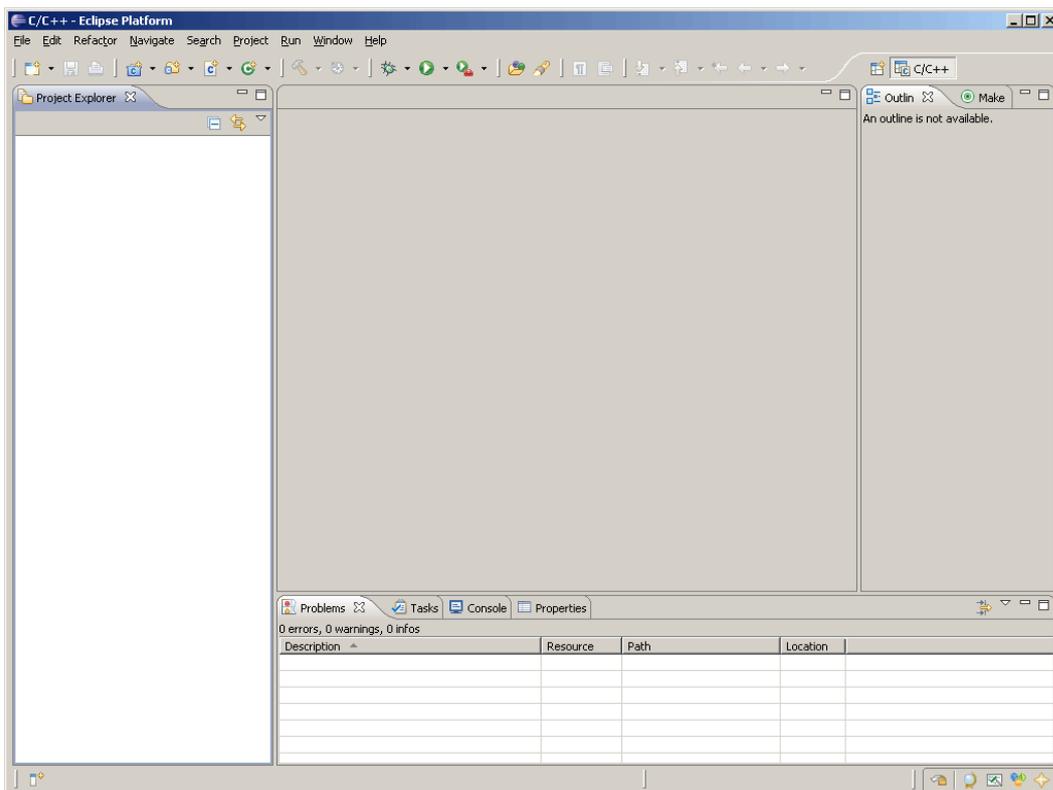


6) 最初に Workspace の場所を聞いてきます。適当なフォルダに変更してください。



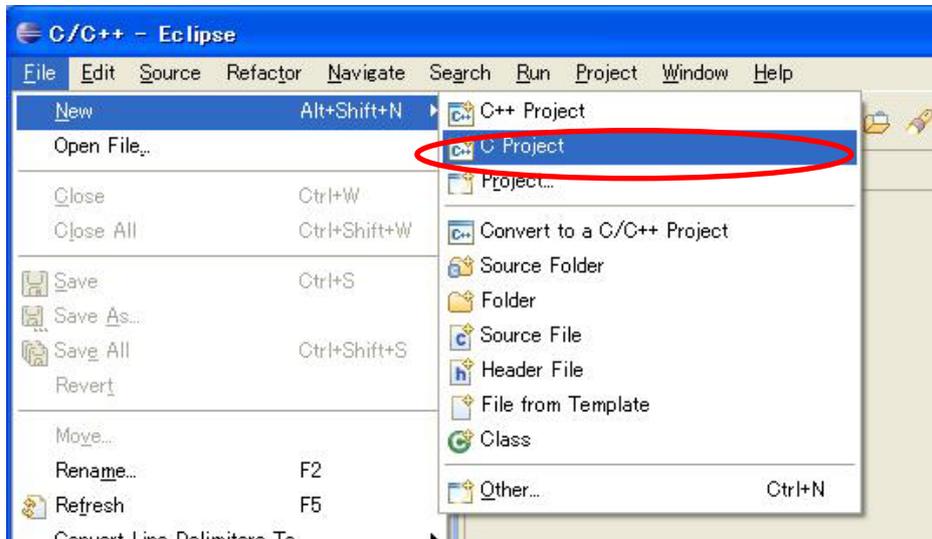


画面の Workbench をクリックします。

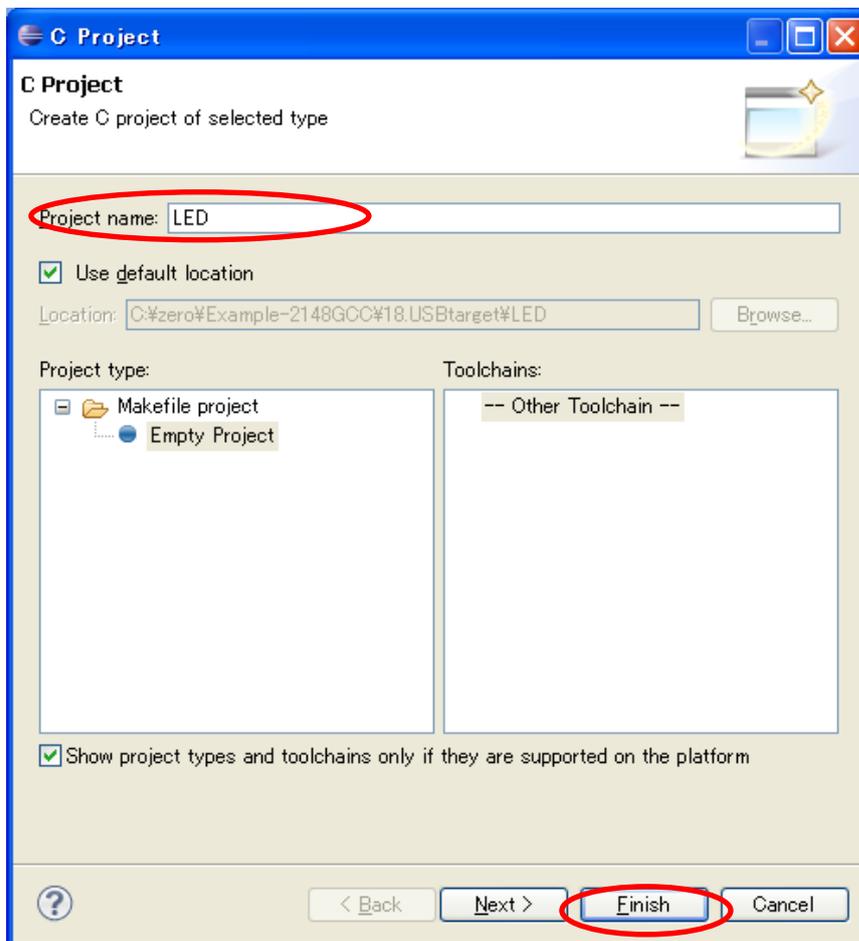


11.3 プロジェクトを作る

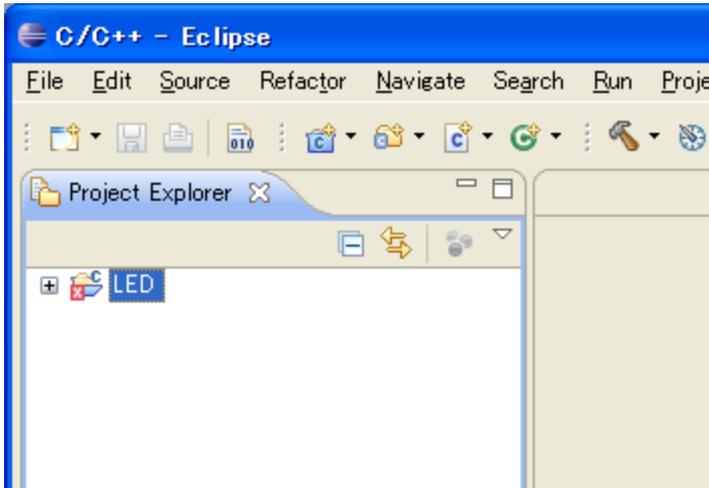
新規プロジェクトを作成するため"File"→"New"→"C Project"を選択します



プロジェクト名を聞かれるので適当な名前(LED)を入力し Finish ボタンを押します。

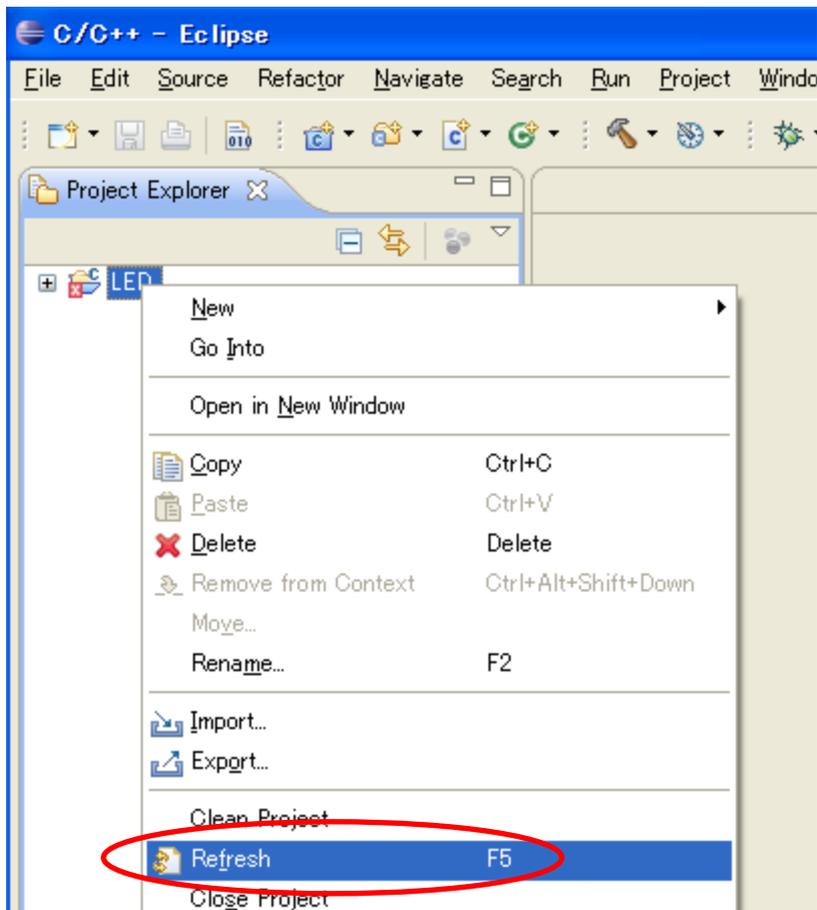


Project Explorer にプロジェクト LED が追加されましたが中身が何もないので、"×"がついています。

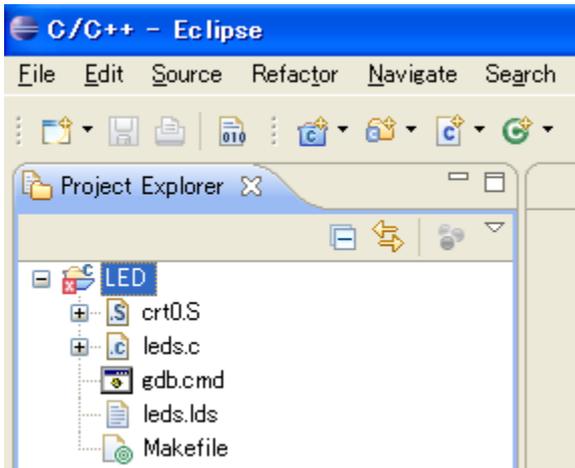


サンプル LEDs.zip のなかのファイルを"C:¥workspace¥LED"にコピーしてください。

Eclipse の” File”→”Refresh”を選択します。

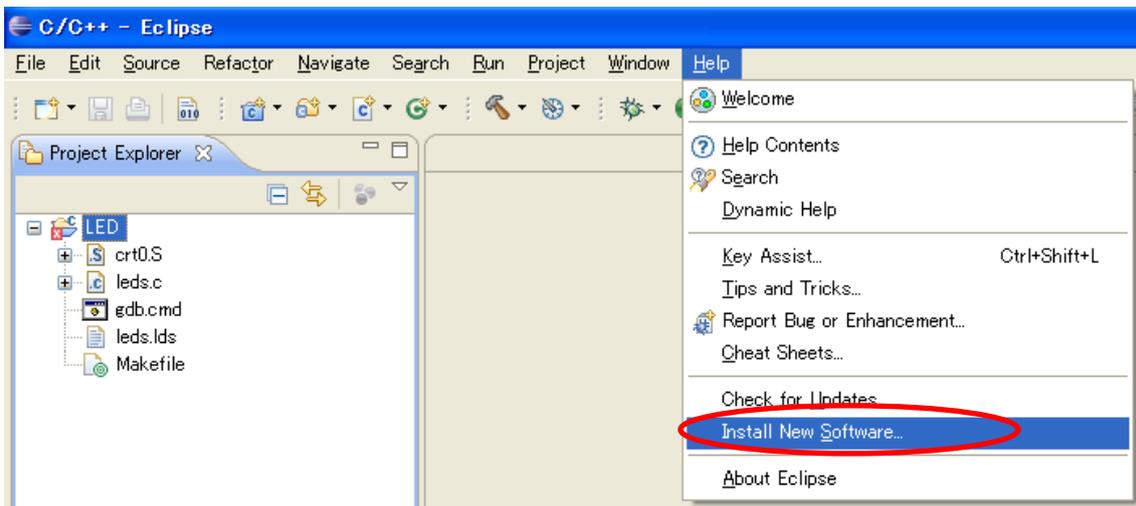


Project Explorer の"LED"プロジェクトの左にある+をクリックするとファイルの一覧が表示されます。

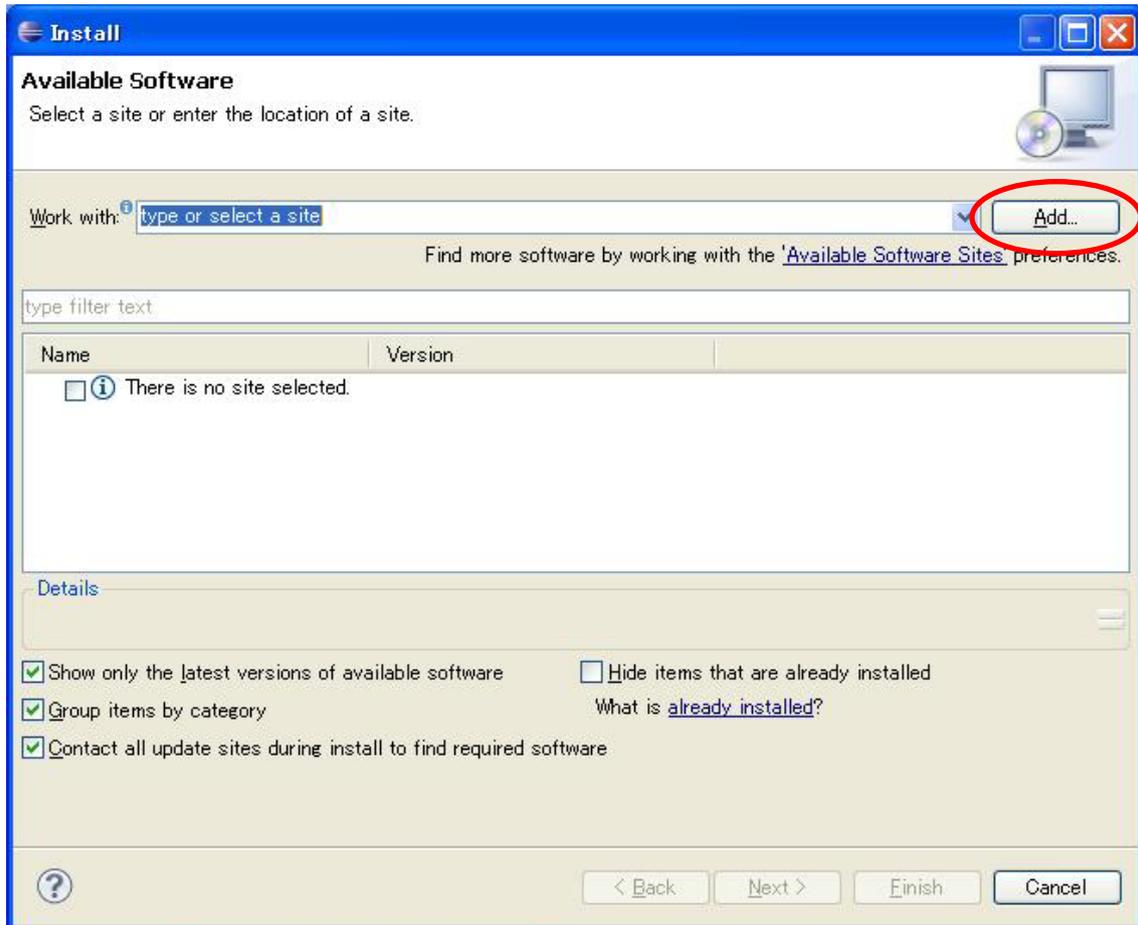


11.4 Eclipse プラグイン(Zylin Embedded CDT)インストール

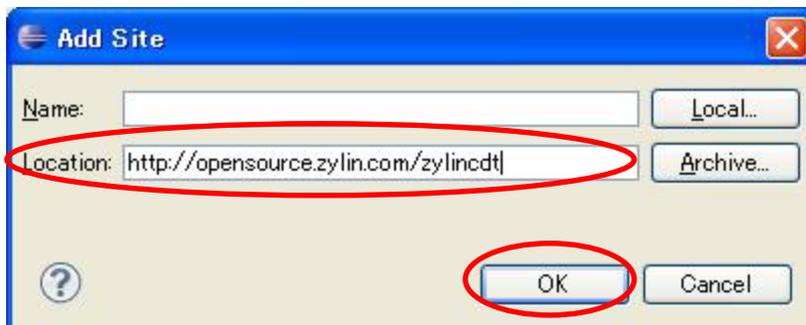
Eclipse の"Help"→"Install New Software"を選択します



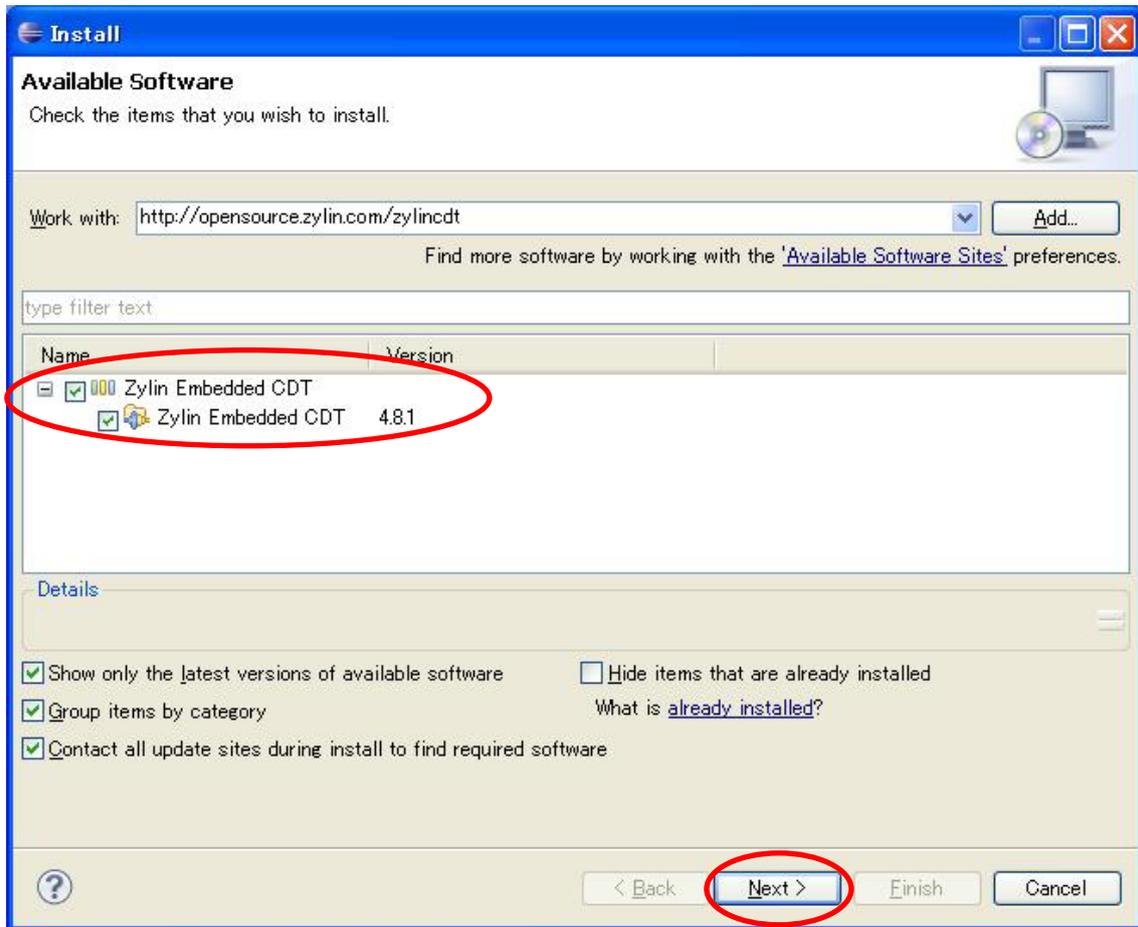
Add ボタンを押します。

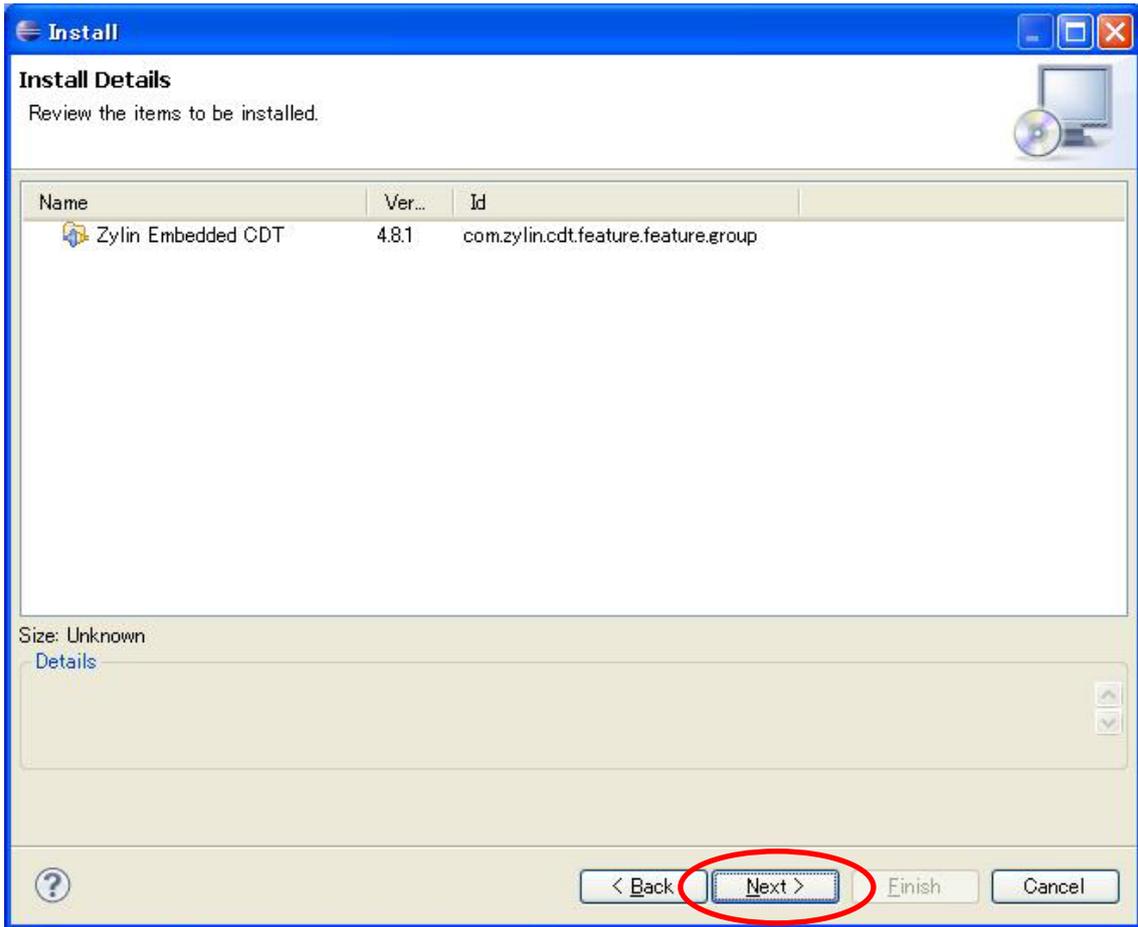


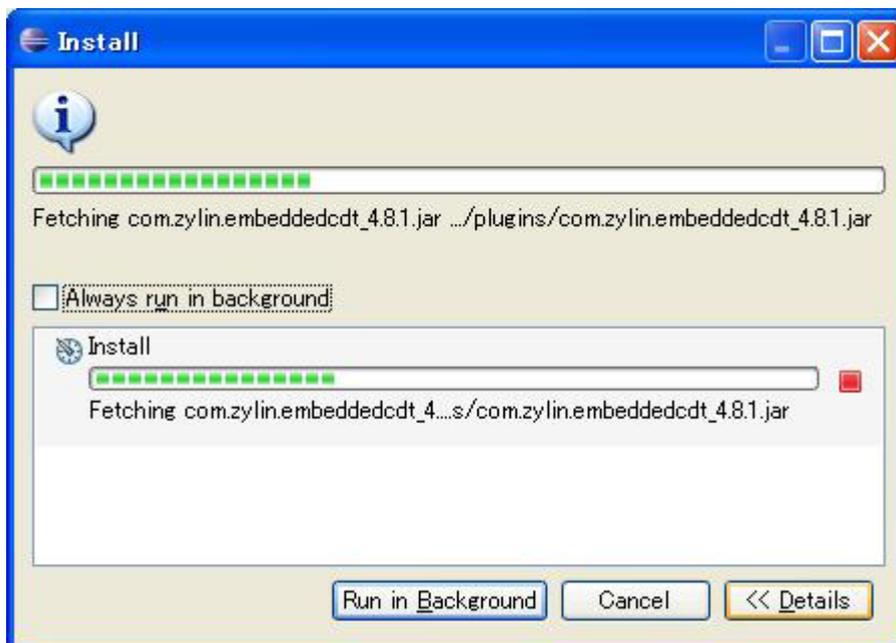
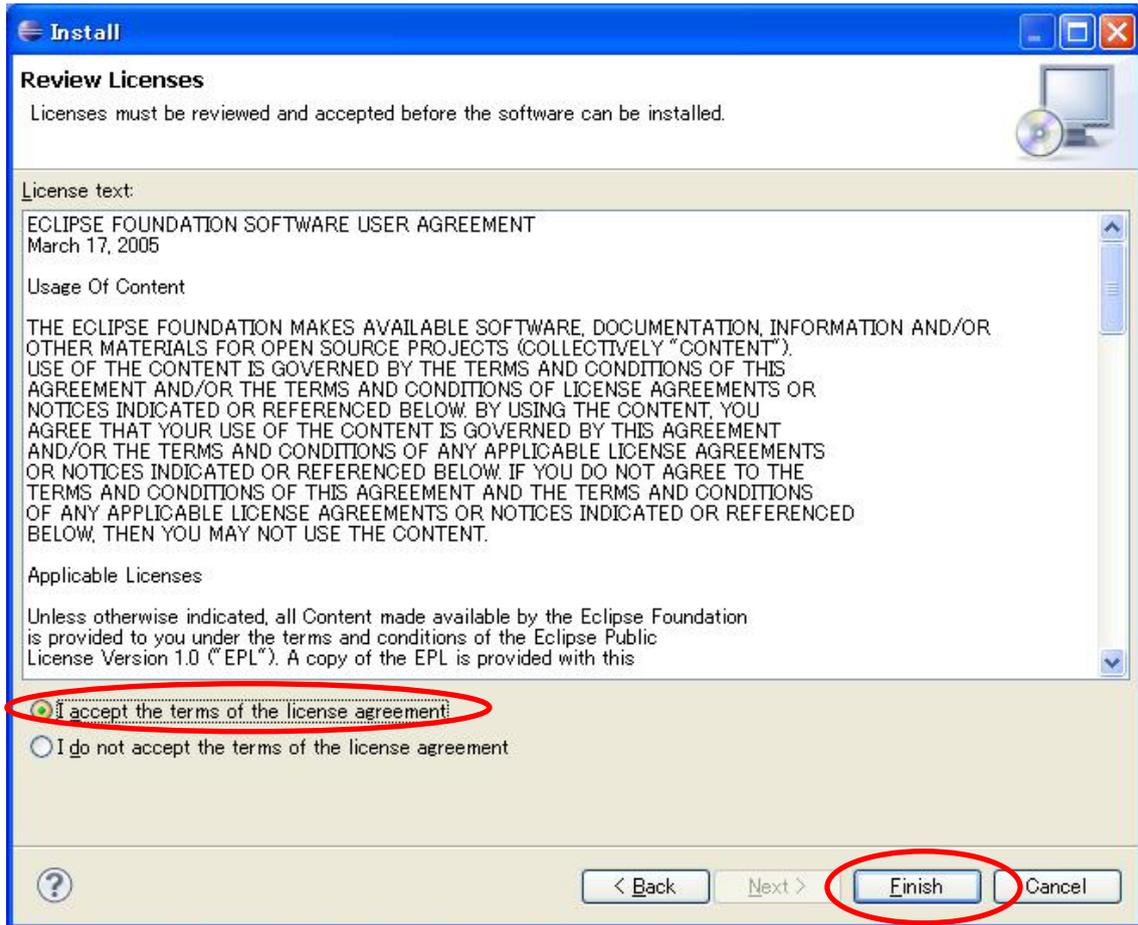
Add Site の"Location"に"http://opensource.zylin.com/zylincdt"と入力し OK ボタンを押す。



Install に "http://opensource.zylin.com/zylincdt" が追加されるのでチェックボックスをクリックしチェックを入れて Next ボタンを押す。





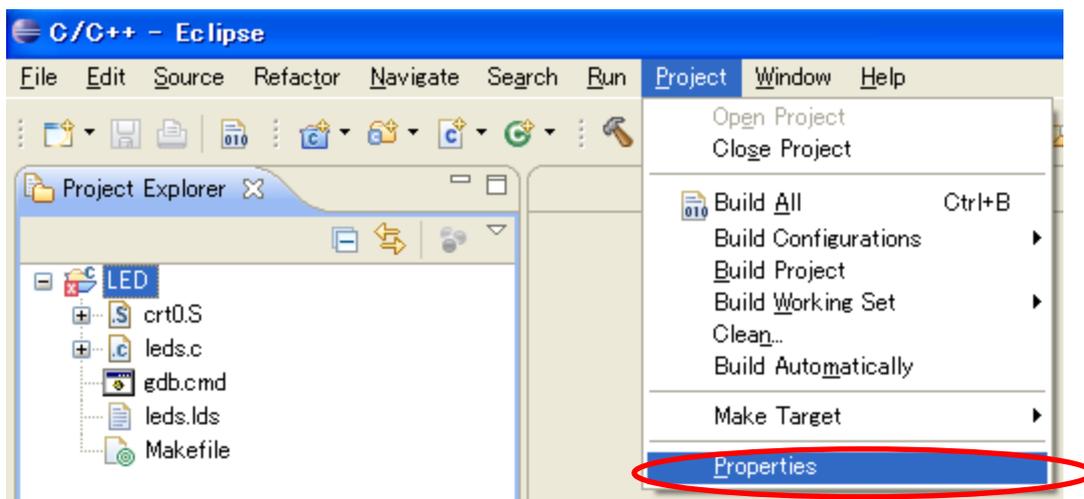




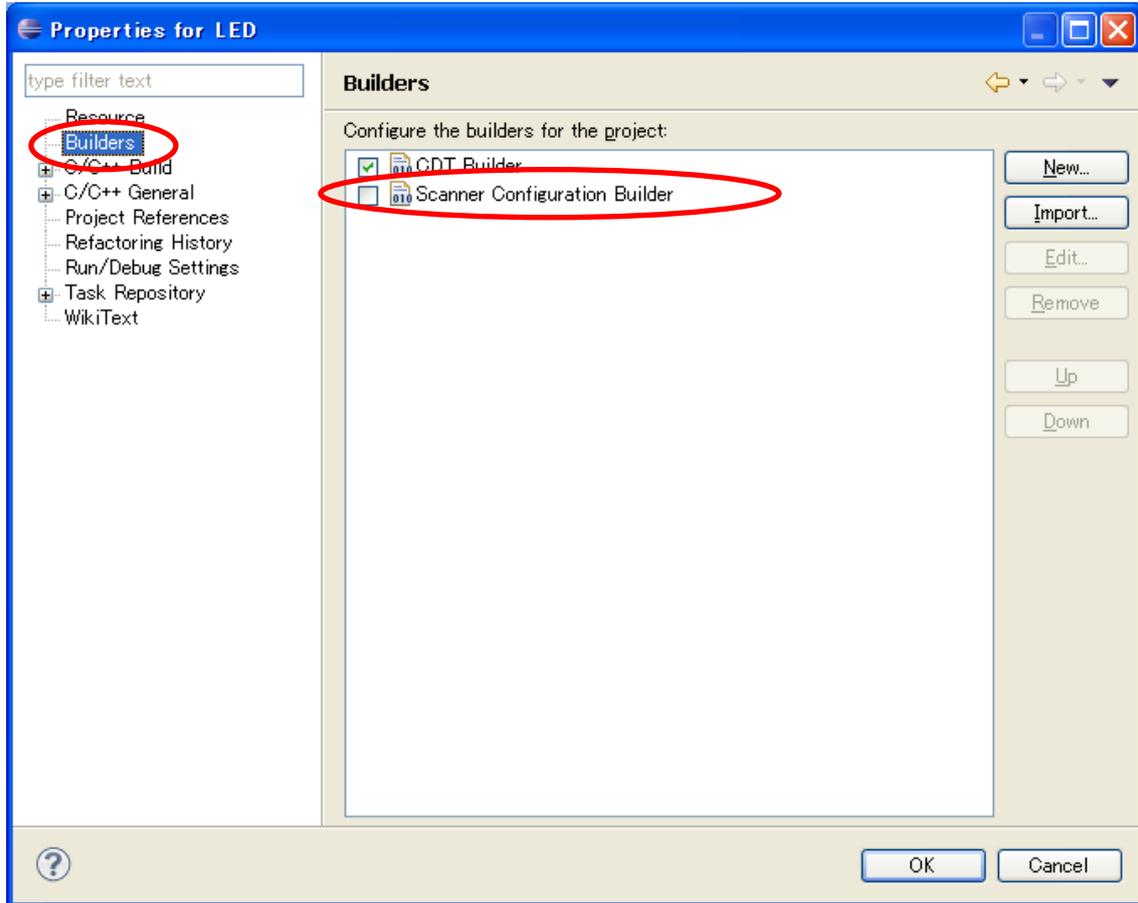
インストール完了したら、Yes ボタンを押して、Eclipse を再起動させます。

11.5 ビルドの設定

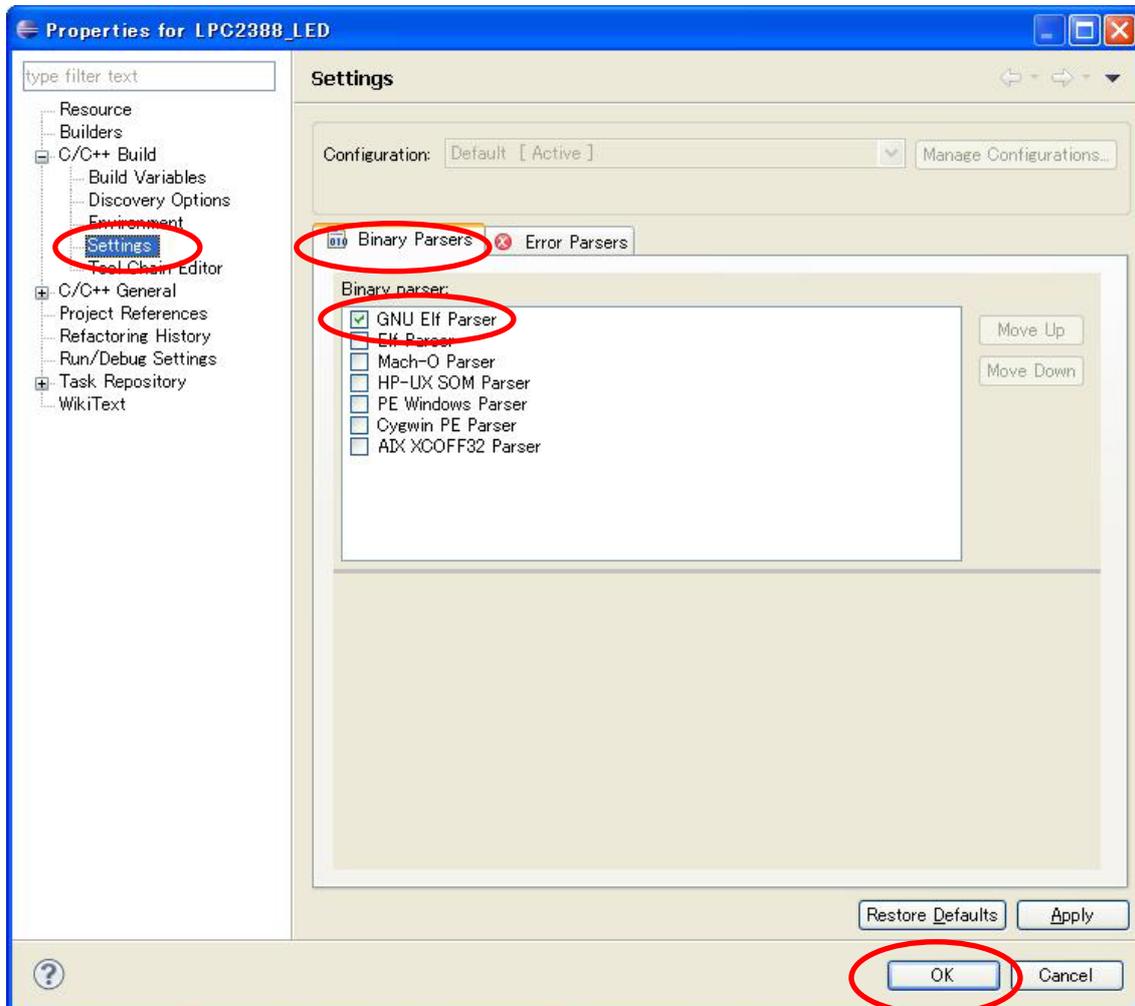
Eclipse の"Project"→"Preferences"を選択する。



Preferences の"Build"を選択し"Scanner Configuration Builder"のチェックマークを外して

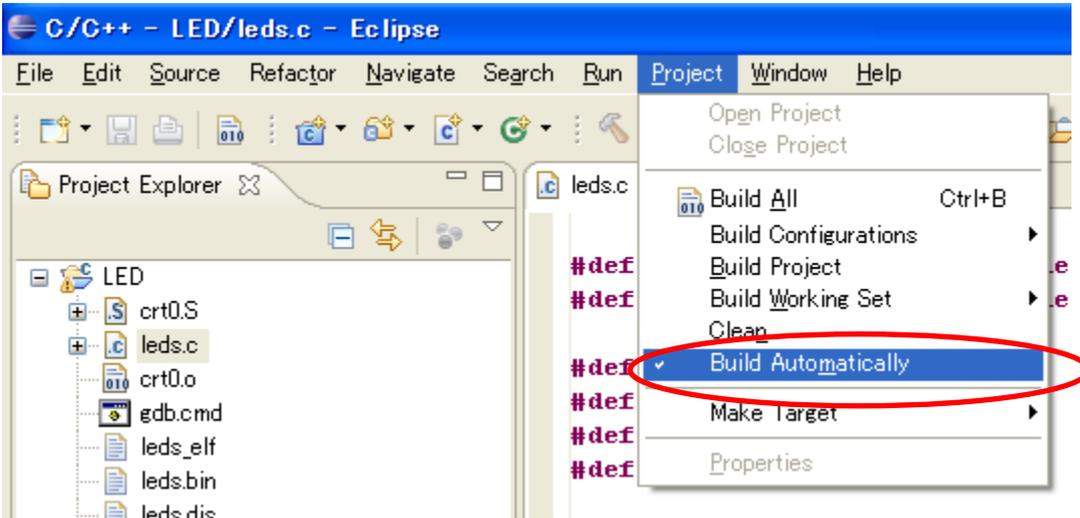


Preferences の "C/C++ Build" → "Settings" を選択し "Binary Parsers" タブの "GNU Elf Parser" にチェックを入れて OK ボタンを押します

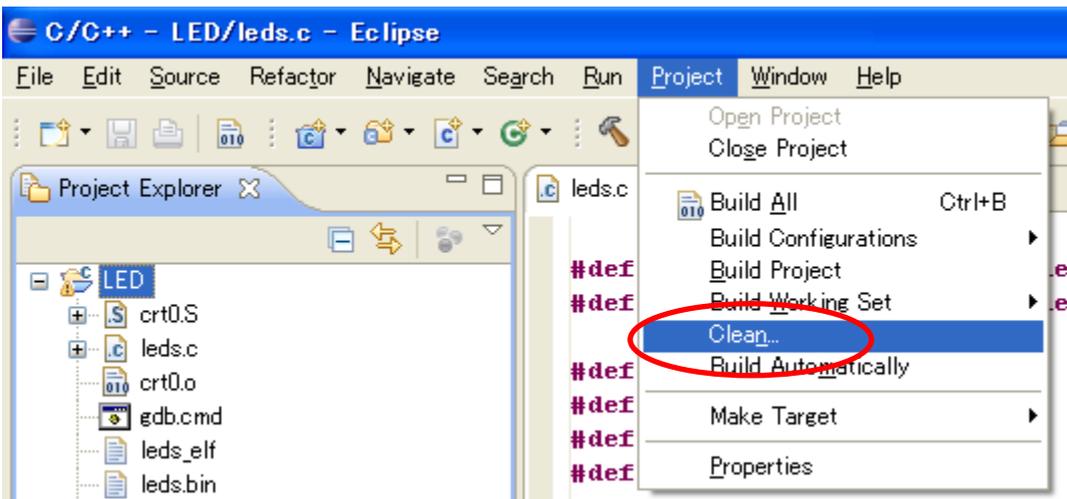


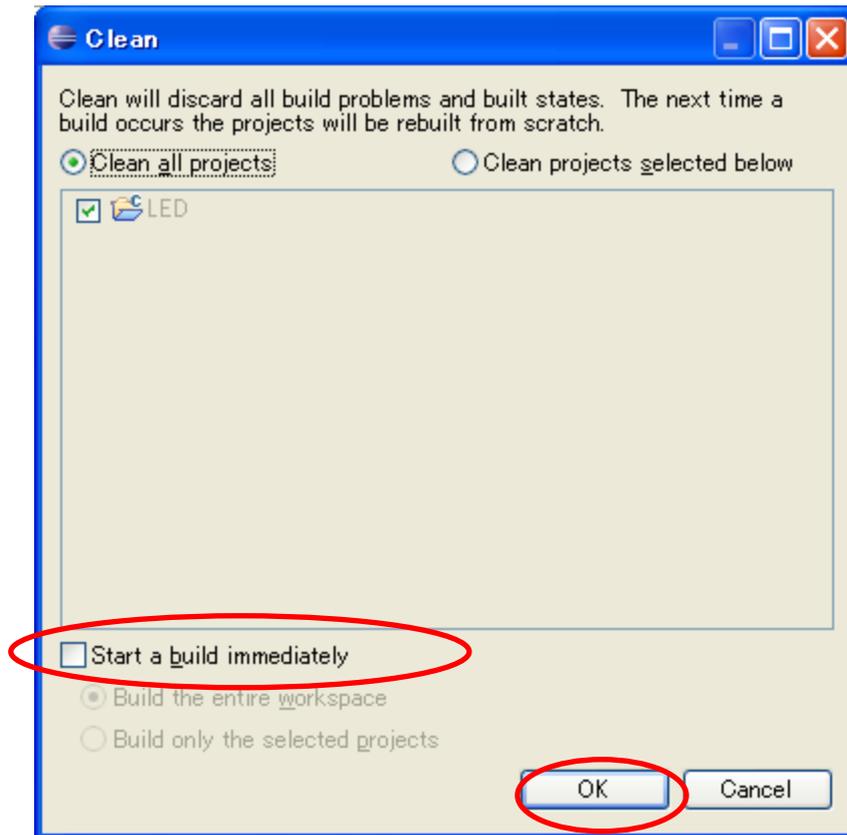
11.6 ビルド

Eclipse の"Project"→"Build Automatically"のチェックを外してください。



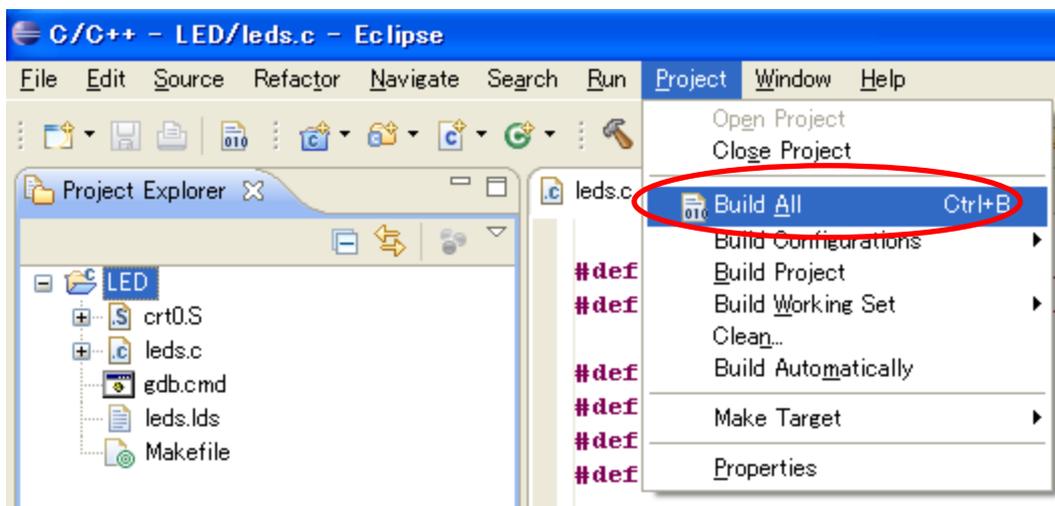
"Project"→"Clean"を選択するクリアが行われます。

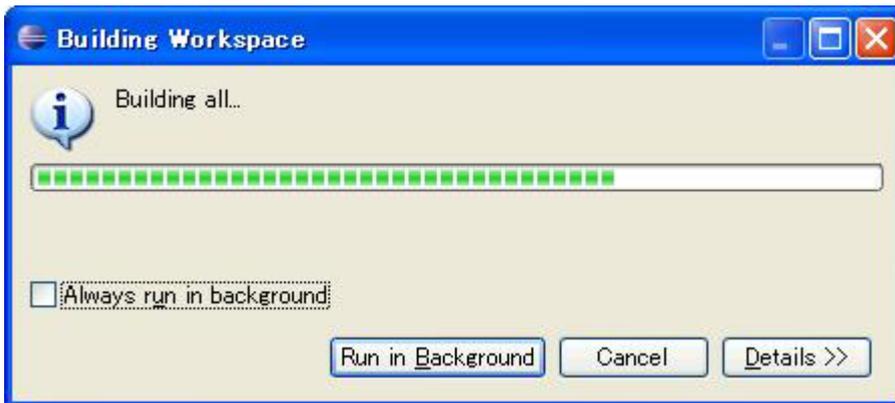




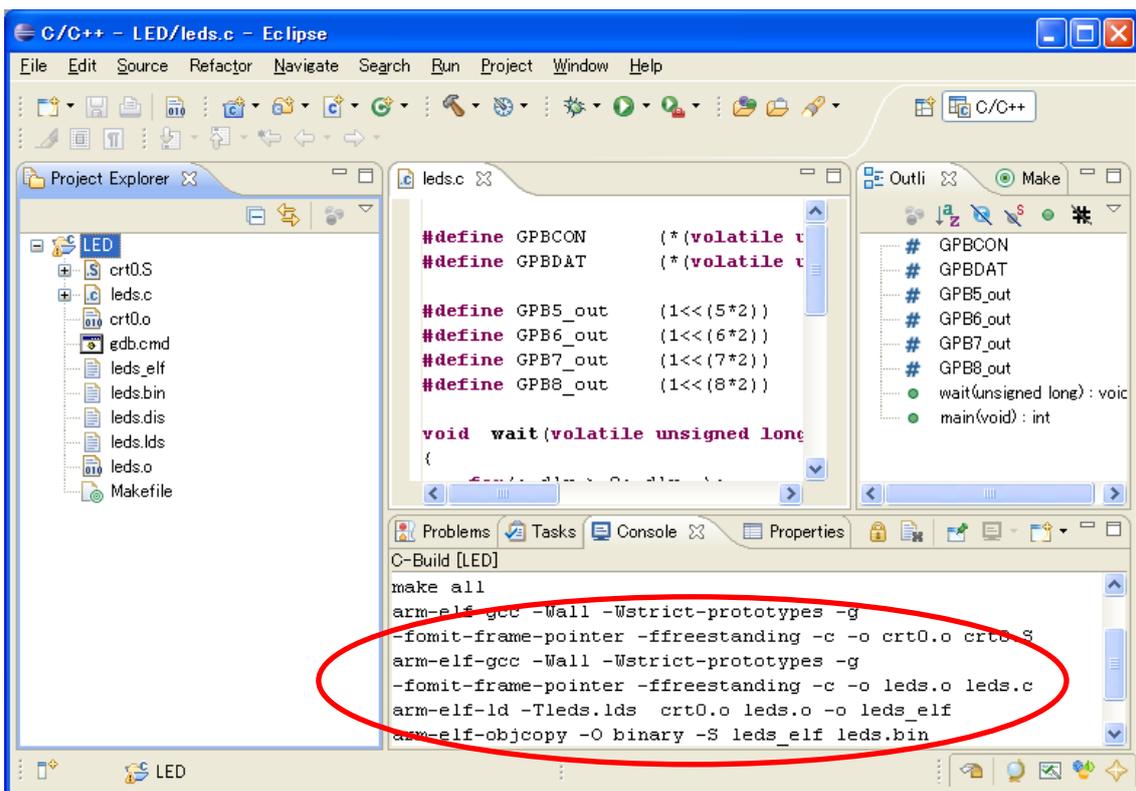
「Start a build immediately」のチェックマークを外して、「Ok」を押します。

"Project"→"Build All"を選択するとビルドが行われます。





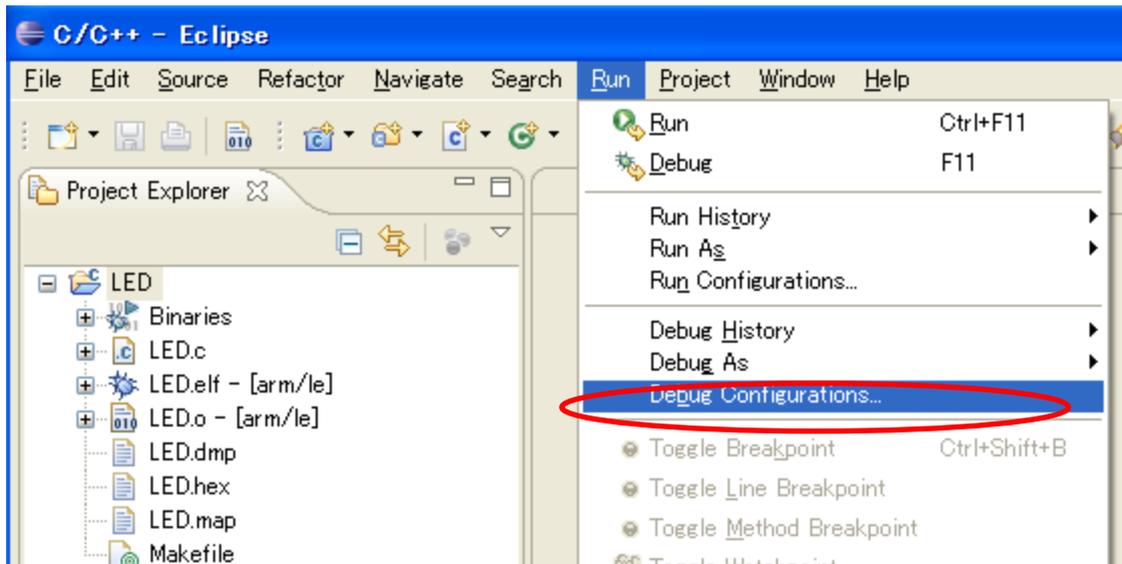
コンパイル中です。



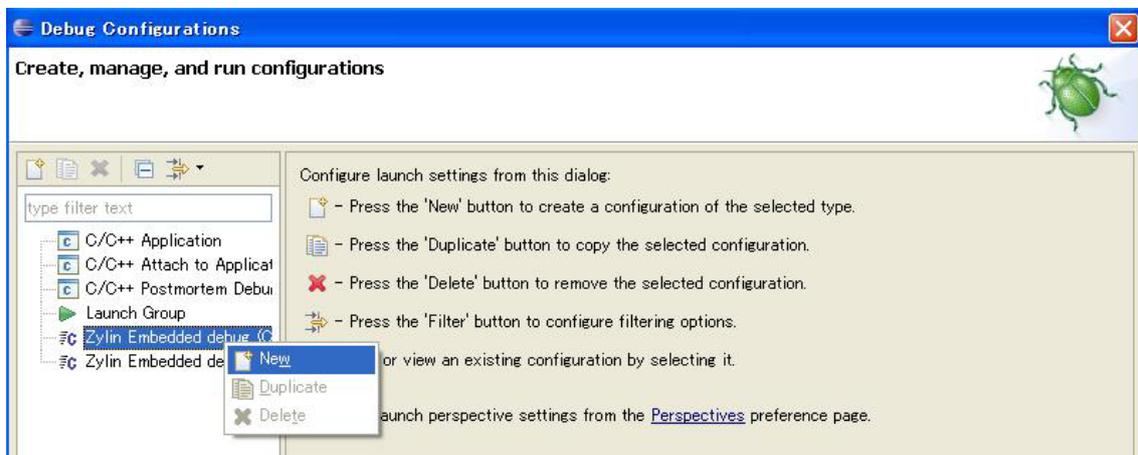
コンパイルが成功すれば、実行ファイル led_elf.elf と leds.bin を生成されます。

11.7 GDB の設定

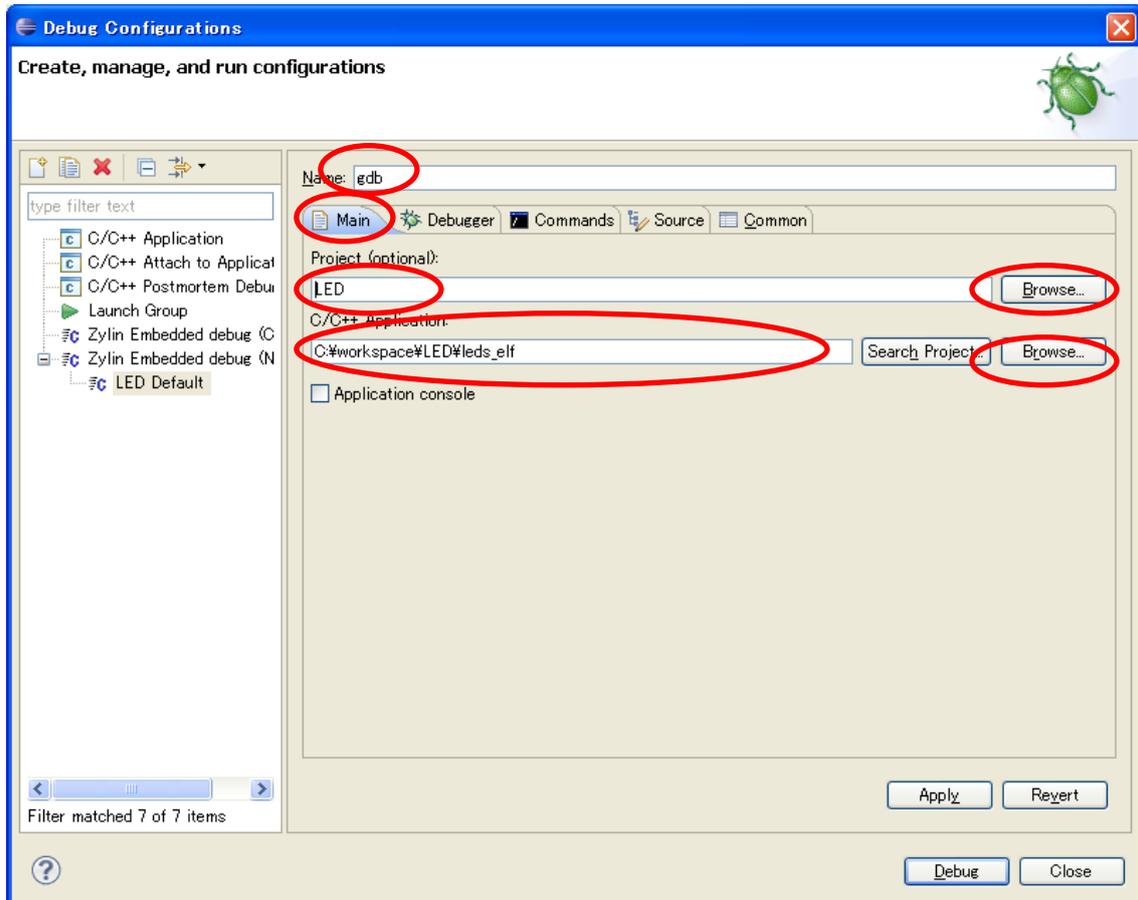
Eclipse の "Run" → "Debug Configurations..." を選択します。



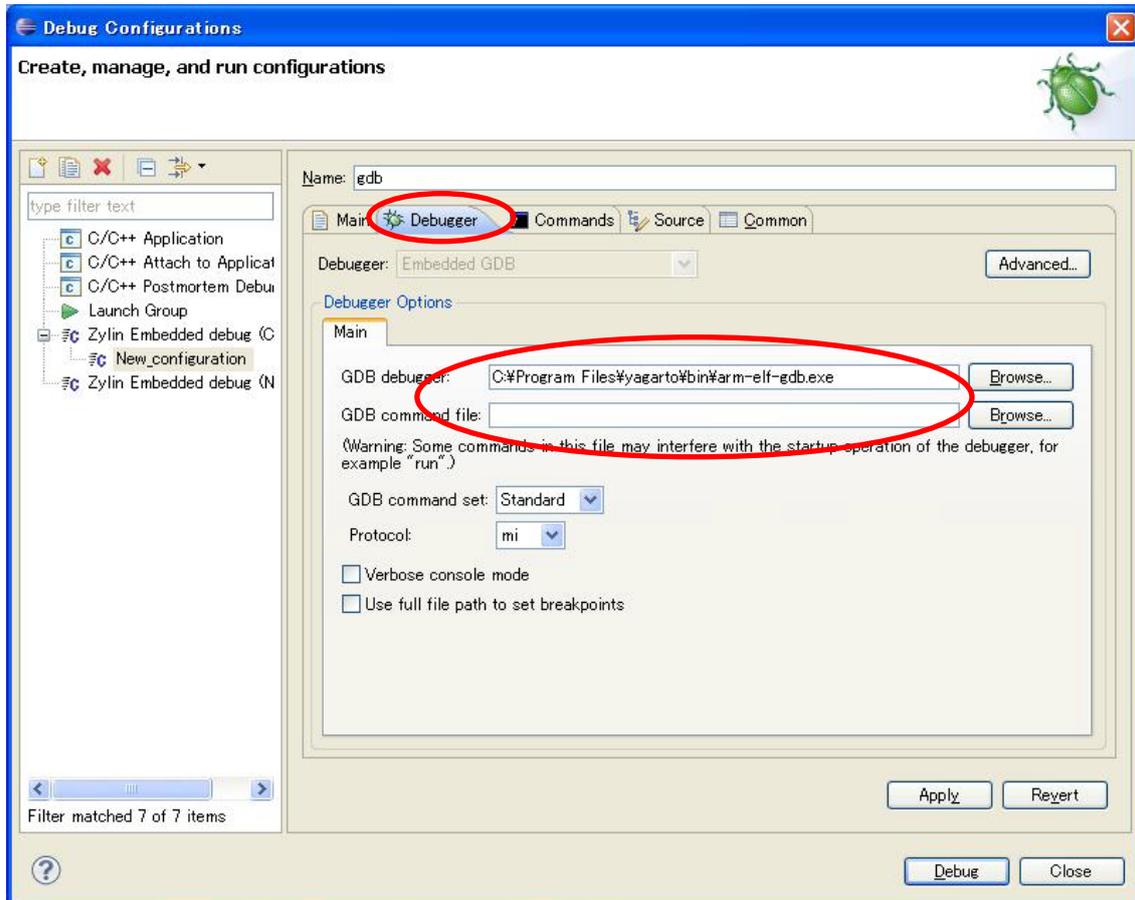
Debug Configurations の"Zylin Embedded debug(Native)"を右クリックし"New"を選択する。



Name に適当な名前を入れる。例、"gdb"と入れます。Main タブの"Project"に"LED"、"C/C++ Application:"に"C:\workspace\LED\leds_elf"と入力します。



Debugger タブの"GDB debugger:"に"arm-elf-gdb"、"GDB command file:"に何も入力しません。



Commands タブの“Initialize' commands”に下記の画面の様に入力します

```
target remote localhost:3333
```

```
monitor halt //ボードの実行を停止させる
```

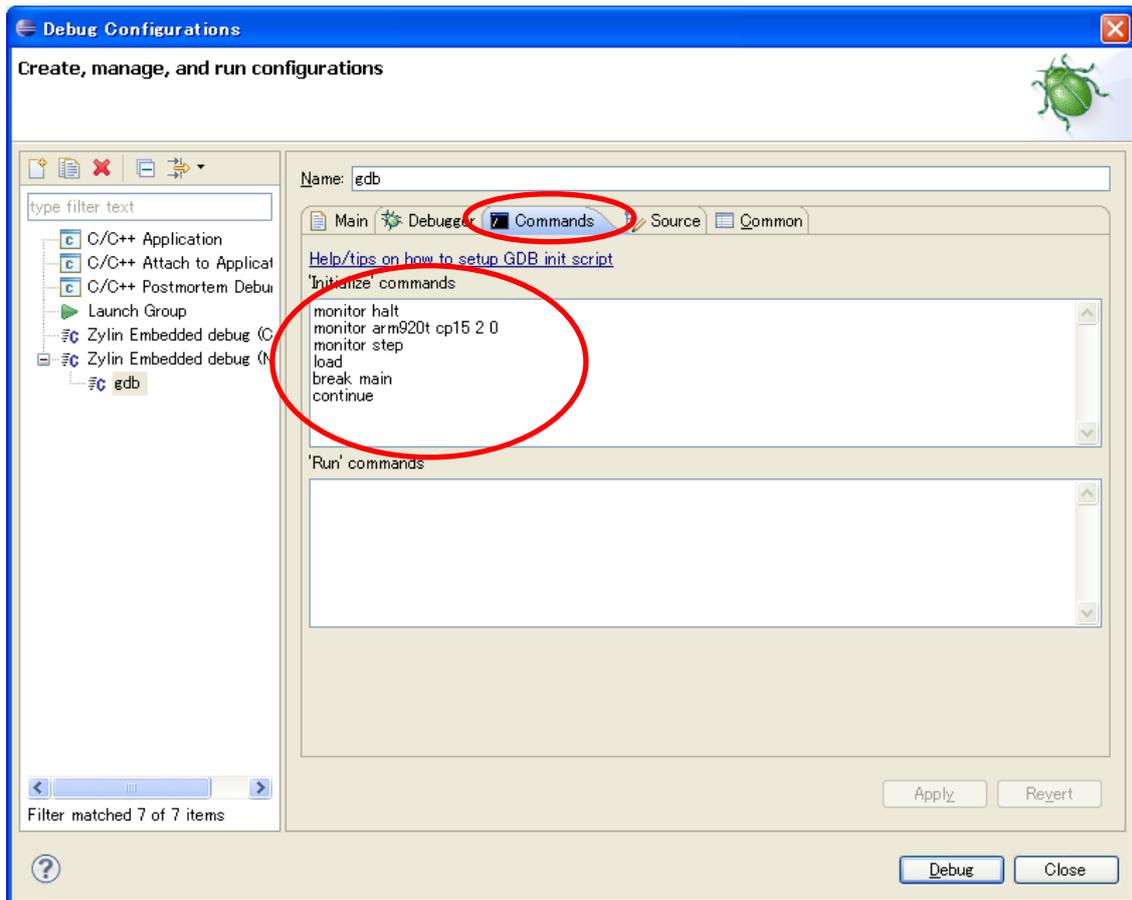
```
monitor arm920t cp15 2 0 // MMU機能をクローズ
```

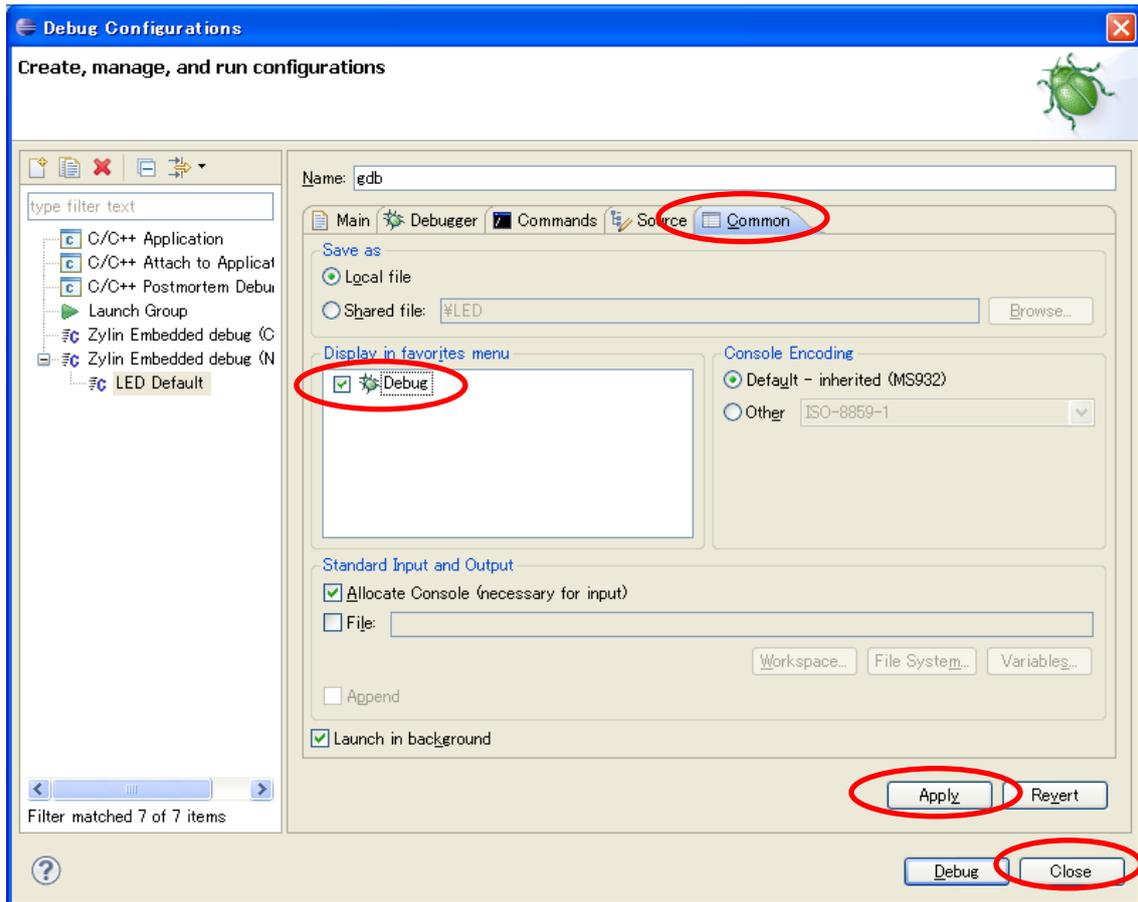
```
monitor step //ステップで実行するように
```

```
load //leds_elfをロード
```

```
break main //「main」関数にブレークポイントを設定
```

```
continue //プログラムを実行させて、「main」にストップ
```

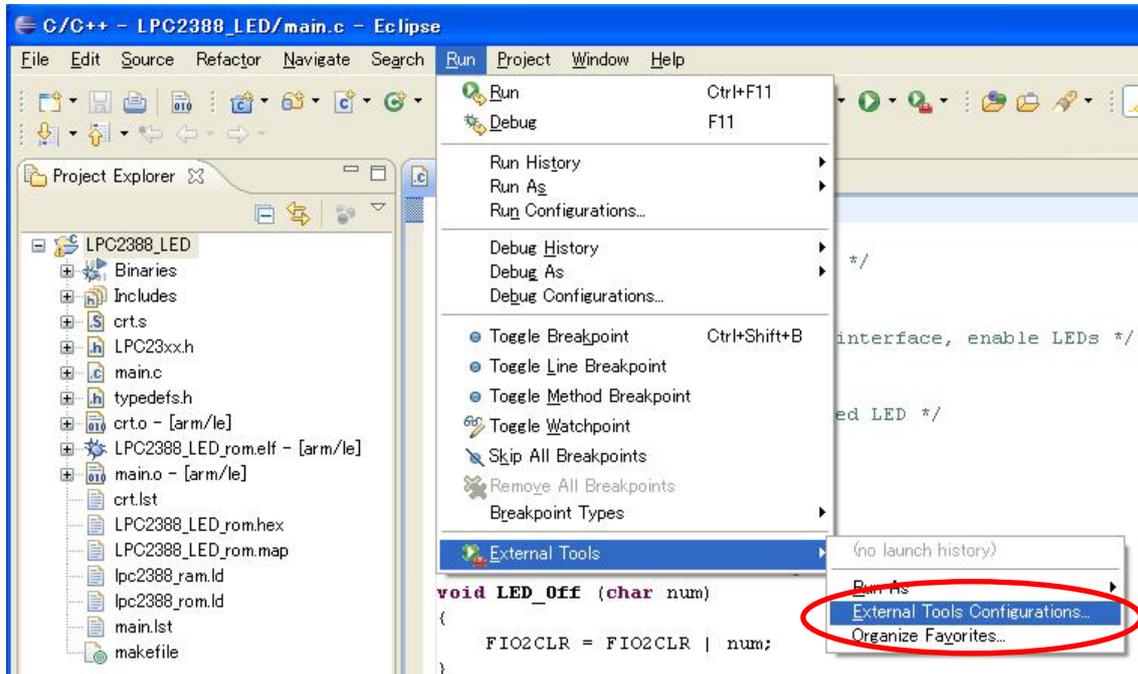




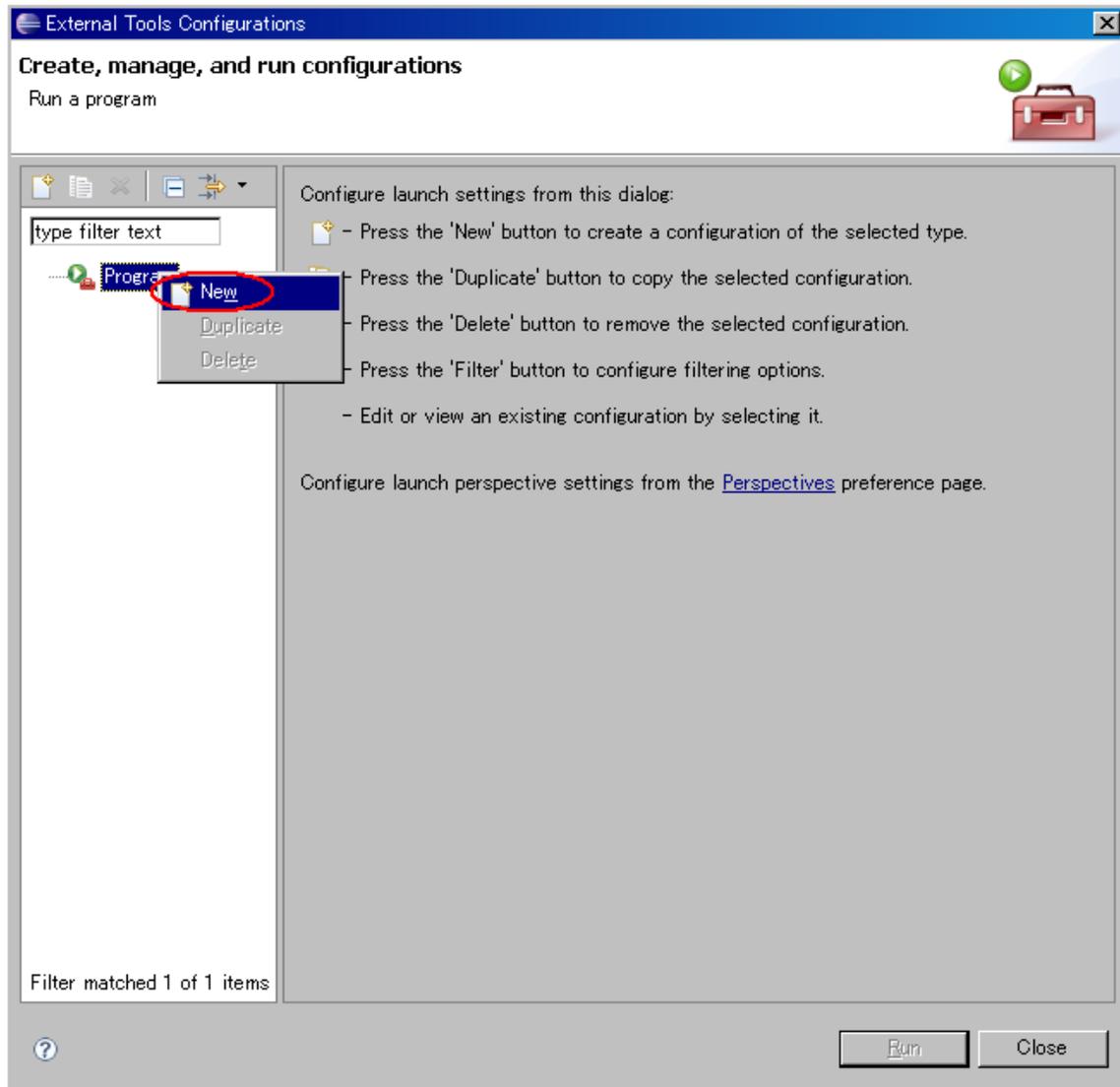
Common タブの "Display in favorites menu" の Debug にチェックを入れます。全てを入力し終わったら "Apply" ボタンを押し、"Close" ボタンを押します。

11.8 OpenOCD の設定

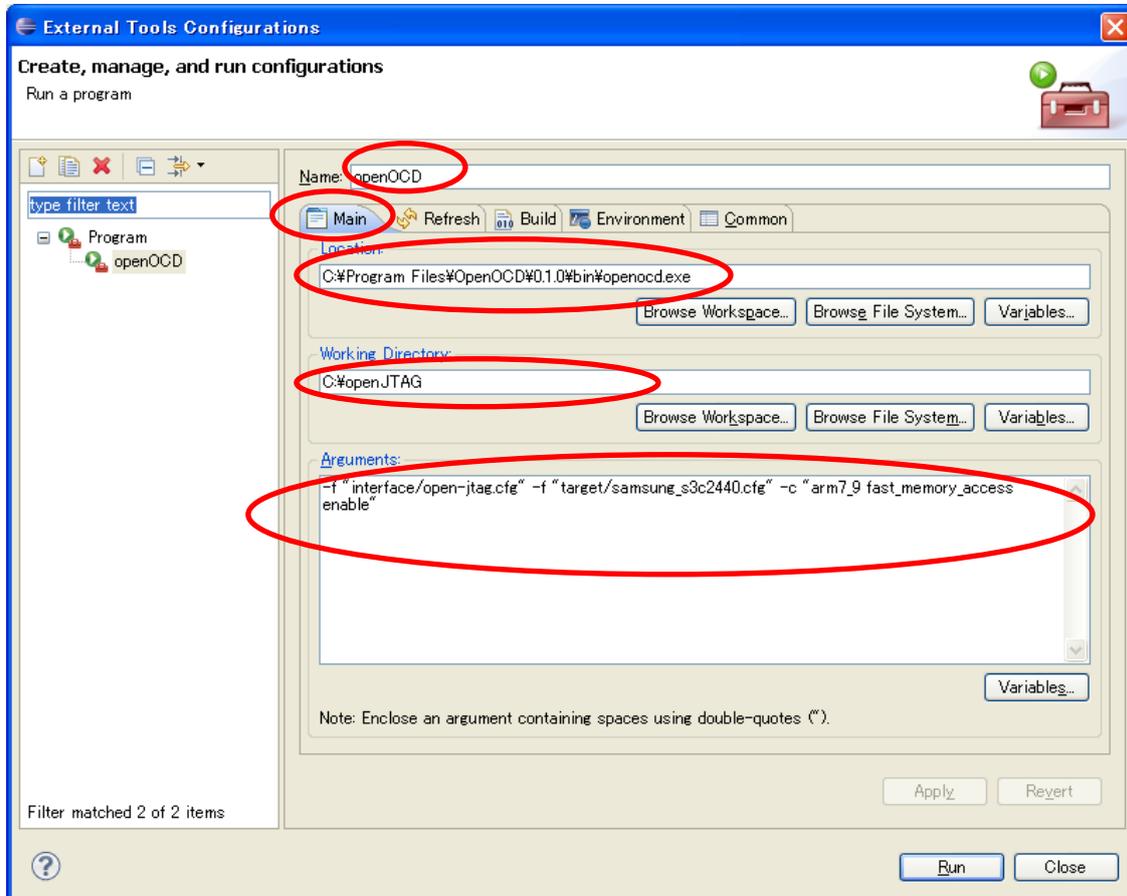
Eclipse の"Run"→"External Tools."→"External Tools Configurations..."を選択します。

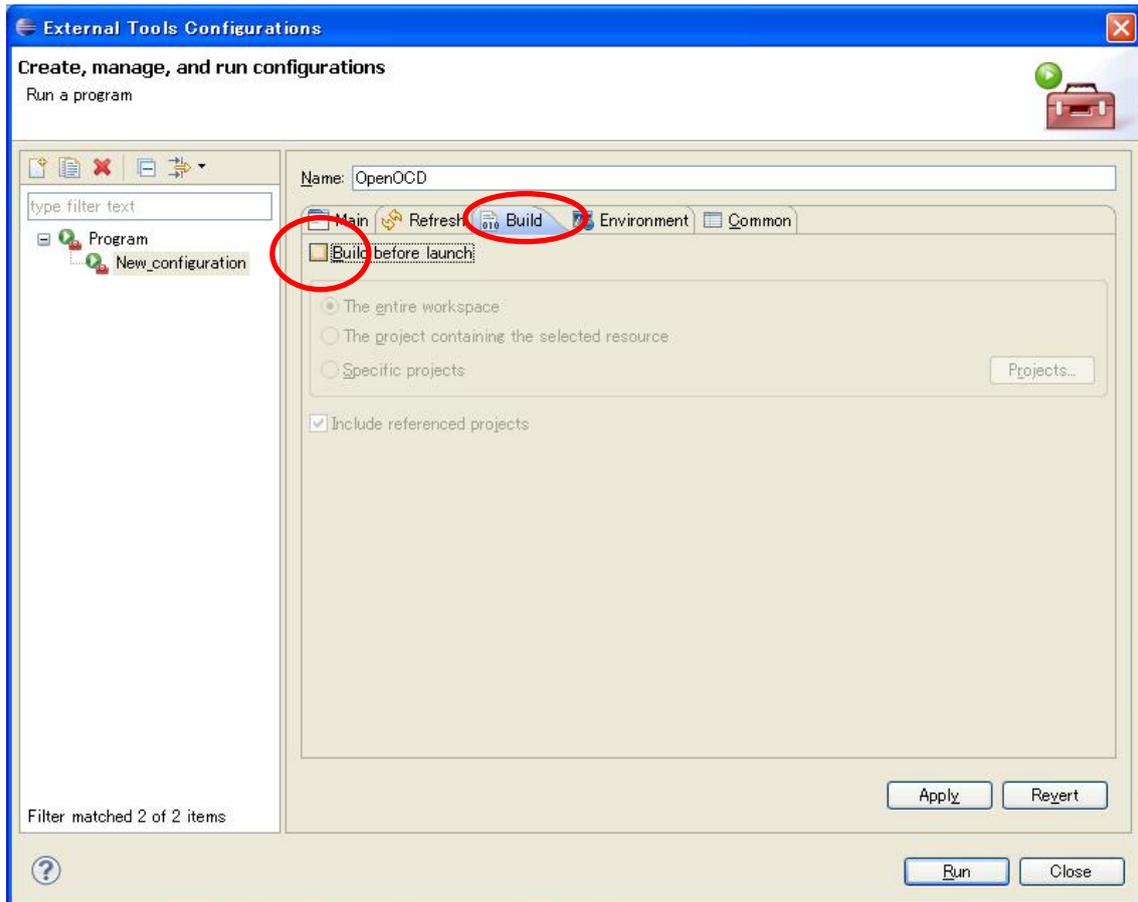


External Tools Configurations の"Program"を右クリックし、"New"を選択します。

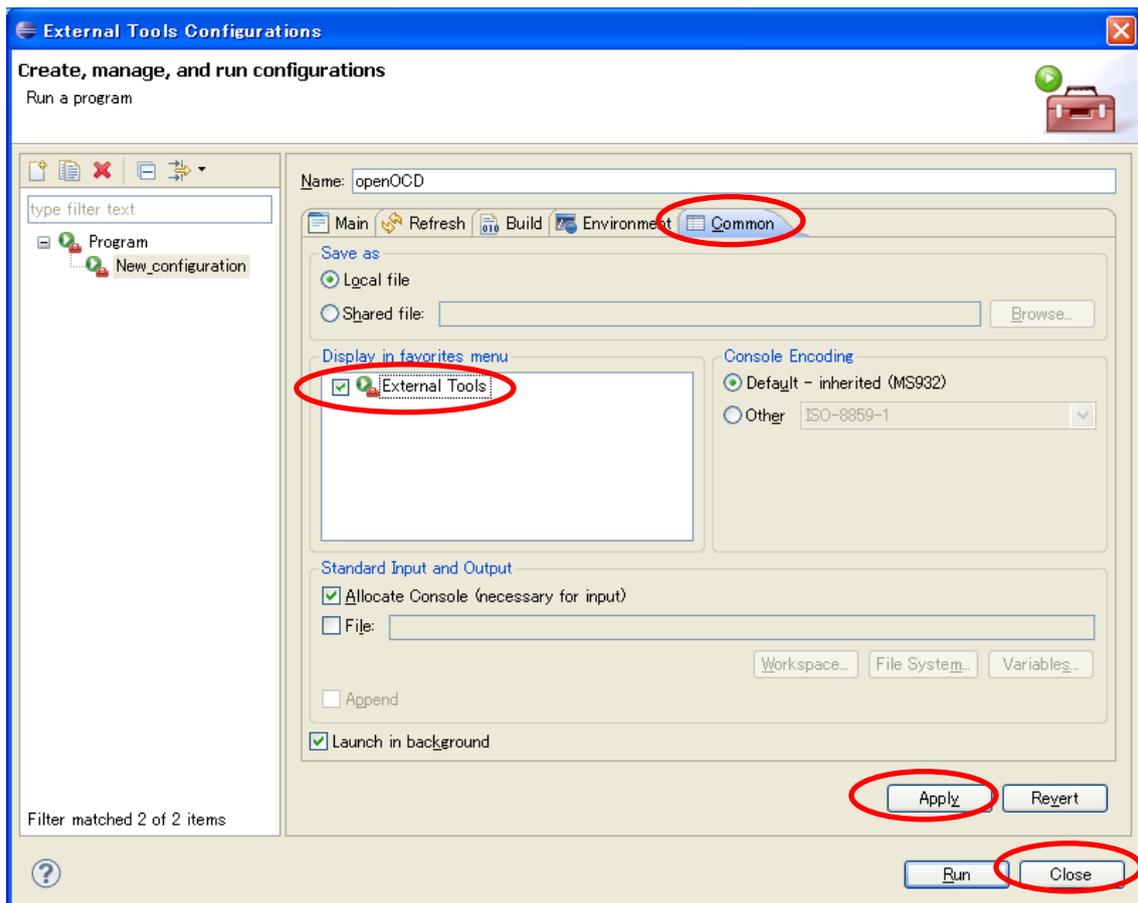


Main タブの"Name"に適切な名前を入力してください。私は” OpenOCD”と入れました。
"Location:"に"C:¥Program Files¥OpenOCD¥0.1.0¥bin¥openocd.exe"、
"Working Directory:"に"C:¥openJTAG"、
"Arguments:"に"-f "interface/open-jtag.cfg" -f "target/samsung_s3c2440.cfg" -c "arm7_9 fast_memory_access enable"と入力します。





Build タブをクリックし"Build before launch"にチェックを外れます。



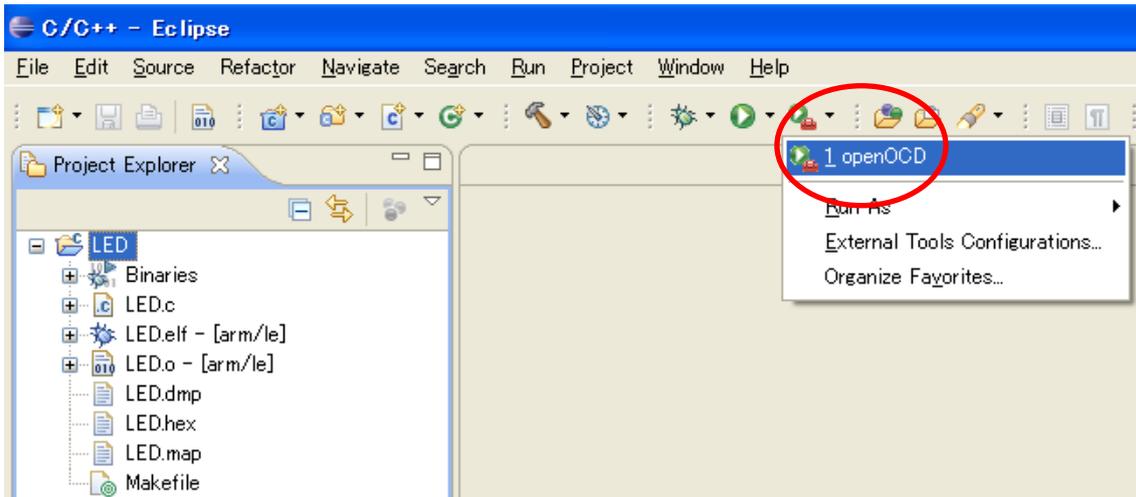
Common タブをクリックし"Display in favorites menu"の"External Tools"にチェックを入れます。全てを入力し終わったら"Apply"ボタンを押し、"Close"ボタンを押します。

11.9 デバッグ

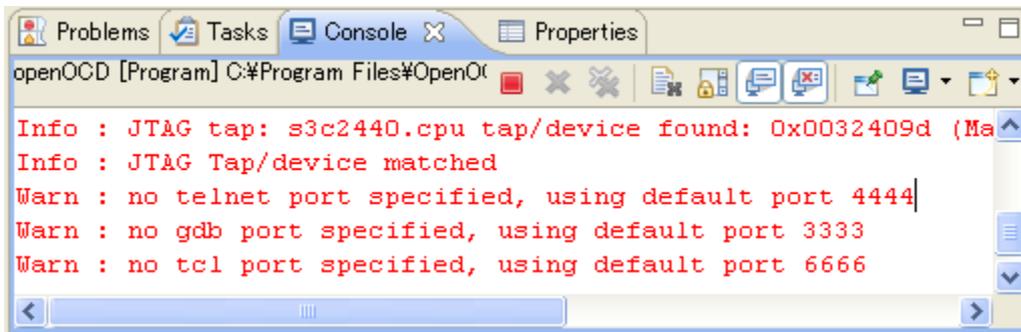
電源投入

1. OpenJTAG をターゲット(MINI2440 ボード)とパソコンに接続
2. ターゲットに電源を入れます

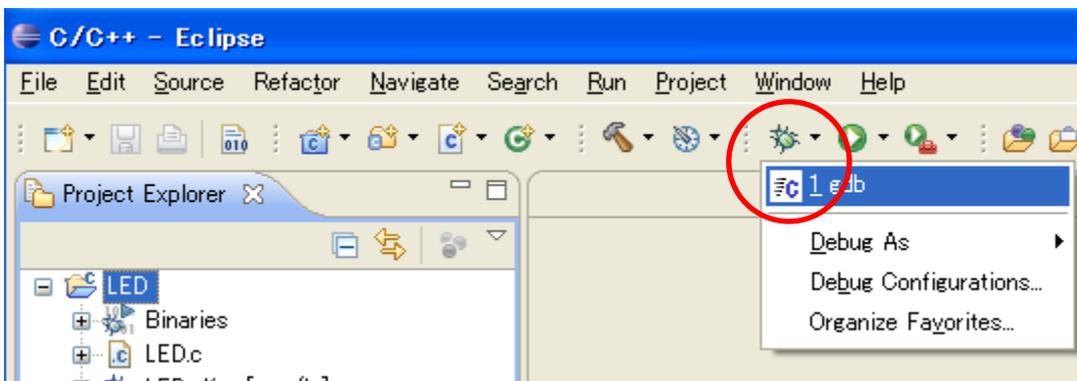
External Tools の▼ボタンをクリックし、OpenOCD を選択

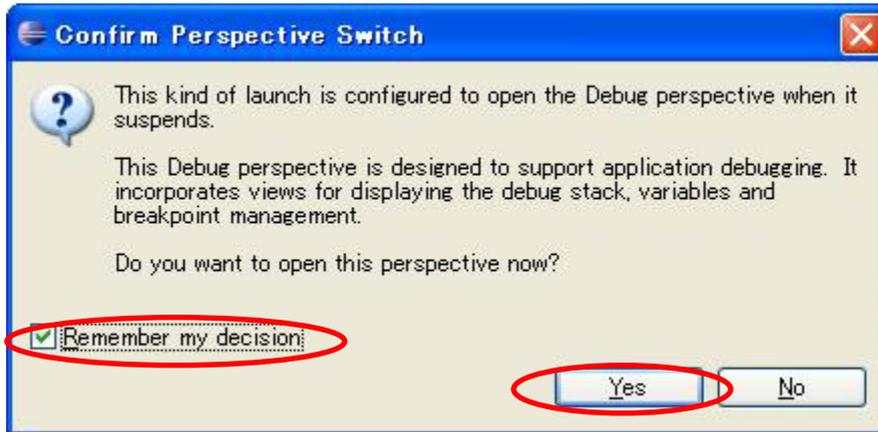


Console ウィンドに下記のメッセージが出力

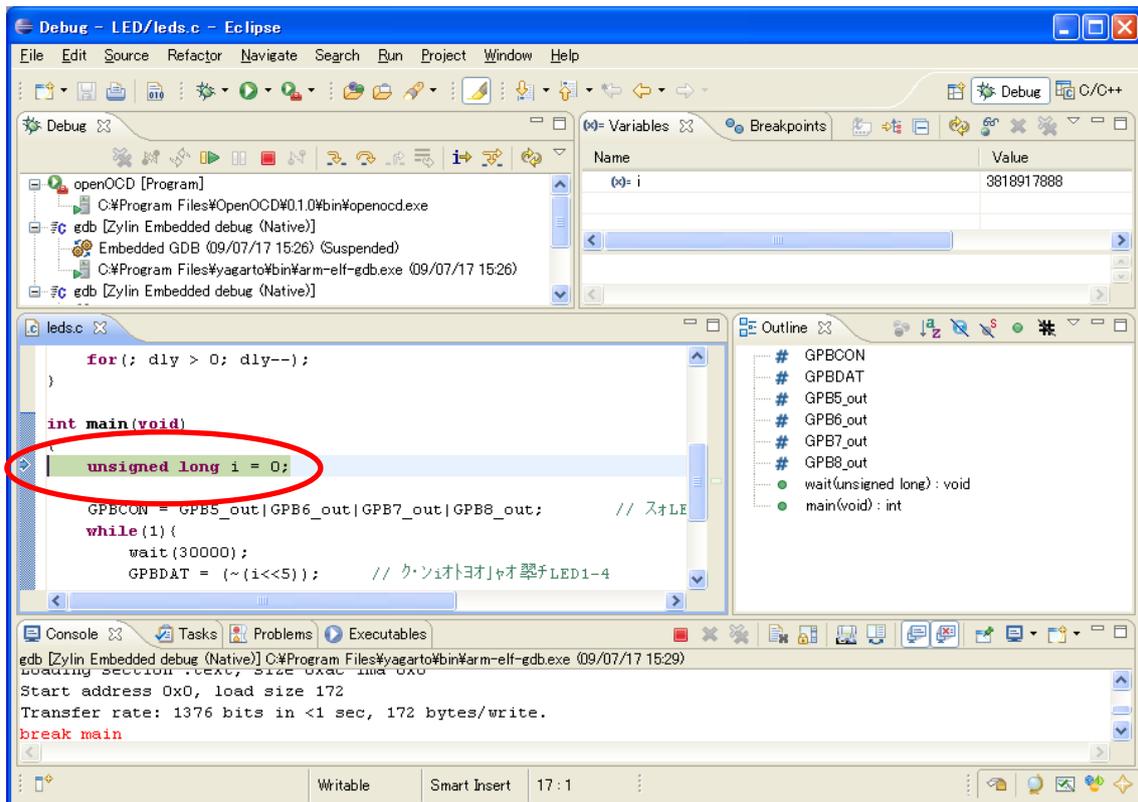


Debug の▼ボタンをクリックし、"gdb"を選択。

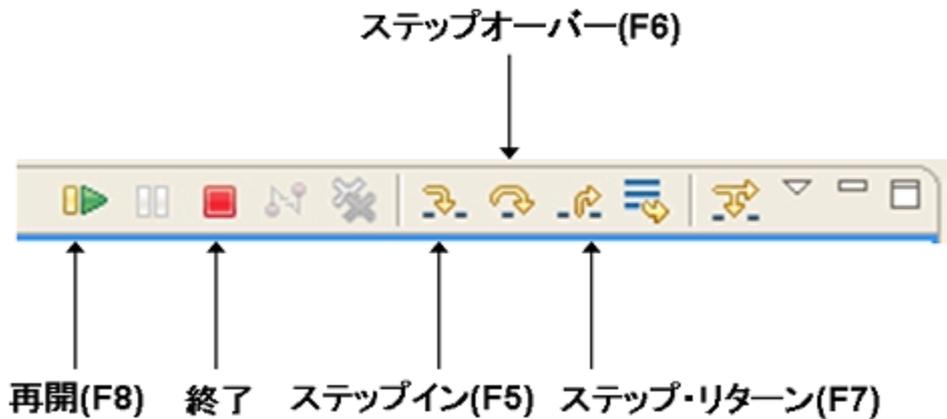




Yes ボタンを押して、デバッグが開始します。



Eclipse に Debug 用のコマンドあるいはショートカット一覧
詳しくは Eclipse のドキュメントを参照



ステップ実行において良く使われる操作の一覧を以下に示します。

操作名	ショートカットキー
再開	F8
ステップイン	F5
ステップオーバー	F6
ステップ・リターン	F7

ステップ実行とは関係ありませんが、前回起動したクラスを再度実行したデバッグする場合は、以下のショートカットキーが便利です。

操作名	ショートカットキー
前回の起動を実行	Ctrl + F11
前回の起動をデバッグ	F11

ブレークポイントでプログラムが中断した状態から、次のブレークポイントまで実行させたり、1行ずつ実行させたりできます。コード「GPBDAT = (~i<<5)」を繰り返して実行することにより、LEDランプが1つずつ点滅



The screenshot shows the Eclipse IDE interface for debugging a C program. The main window displays the source code for `leds.c`. A red circle highlights the line `GPBDA1 = ~(i<<5);` in the `while(1)` loop, with a comment `// iの値により、LED1-4を点滅させる`. The `Variables` window shows the variable `i` with a value of `3`. The `Disassembly` window shows the assembly instructions for the current line, including `mov r2, #1442840576`. The `Console` window shows a breakpoint hit at `leds.c:17` and a warning message: `Warning: /cygdrive/D/embedded/eclipse/workspace/mini2440/MINI2440_LED: No such file or directory.`

Debug 途中の ARM9 ボードの様子①(一番右の LED ランプが点灯)



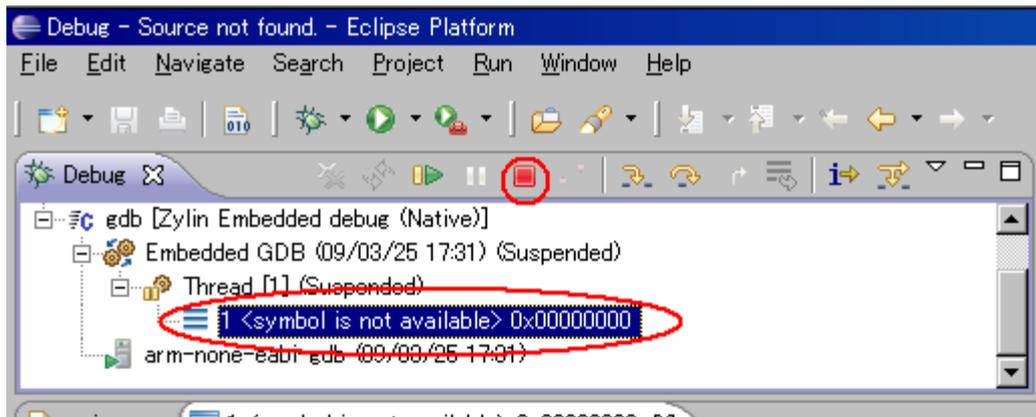
Debug 途中の ARM9 ボードの様子(2(右から2番目の LED ランプが点灯))



11.10 デバッグ終了

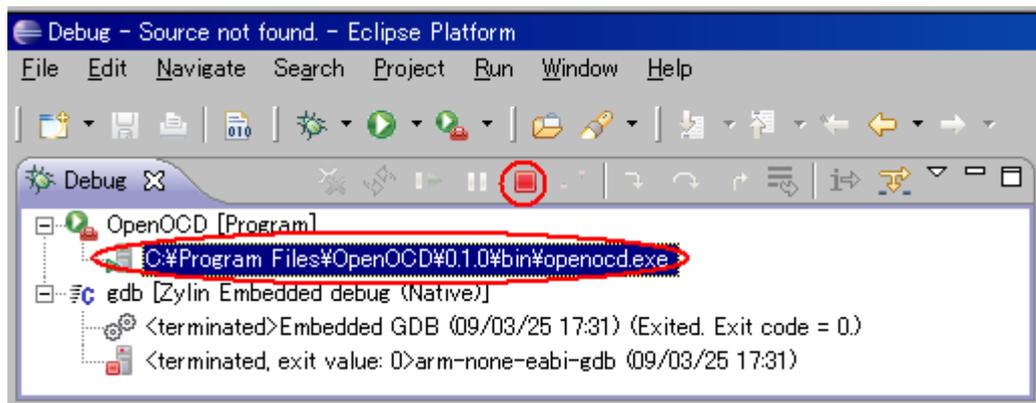
1) gdb の停止

Debug ウィンドウの gdb の Thread を選択し、停止ボタンと押します



2) OpenOCD の停止

Debug ウィンドウの OpenOCD の Thread を選択し、停止ボタンを押します



3) 電源停止

ターゲットの電源を停止

4) OpenJTAG をターゲットから取り外す

5) 上記が面倒であれば Eclipse を終了しターゲットの電源停止、open-JTAG を取り外しても OK です。