Tiny4412 ユーザーマニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

更新日 2015/3/31

• 修正履歴

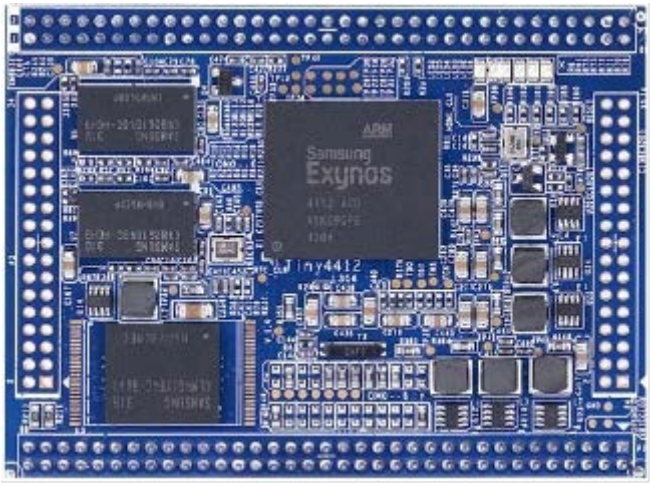
NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2013/08/15
2	Ver1.1	・Android 変更： 1） 自動スリープ状態に入る時間を三週間に延長する； 2） バックライト調整の追加； 3） 電源起動後自動 3G オンラインの追加；メッセージサポートの追加； 4） U-boot-Tiny4412 の追加； 5） Superboot-4412 の追加； 6） HDMI 出力の改良； 7） サムスン（sumsung）からの資料の追加； 8） GPU ドライバーの改善：2D/3D 性能 40%アップ； 9） MiniTools サポートの追加； 10） Android でイーサネットサポートの追加、DHCP 自動 IP 取得サポート； 11） USB カメラサポートの追加； 12） Android4.2.2 にバージョンアップ ・Linux サポートの追加 （Linux-3.5+Qtopia2/Qtopia4/Qt4.7）；	2013/11/10
3	Ver1.2	**・Android 変更：** （1）LCD のちらつく問題の修正 （2）抵抗スクリーンサポートの追加（S70、w101 など） （3）UIRoot 権限サポートの追加 （4）APP プリロードのため、Data パーティション、イメージ書き込みサポートを追加 （5）HDMI 書き込み時、FriendlyARM.ini で LCD-Type パラメータの指定による解像度の変更サポートを追加 （6）SD カードの読み書きプログラム＆読み書き手順の追加 （7）シリアルデバイスの読み書き権限問題の修正 （8）イーサネットのコンフィグインターフェース（固定 IP/DHCP ダイナミック IP のコンフィグをサポート）と起動時の自動接続サポートの追加。 （9）eMMC のパテイーションの最適化。eMMC サイズにより、異なる userdata.img ファイルが作成される。 （10）iTest による COM3 のテストの際、デバイスを /dev/s3c2410_serial3(115200.8.1)に設定した場合、接続不可の問題の修正 **Android4.1.2 追加、主な特徴：** （１）カーネルバージョンは Linux3.0.31	

		（２）赤外線リモコンサポート （３）USB　Wifi インターネットカードで Wifi ネットワークへの接続サポート （４）メディアハードウェア復号化で再生性能の向上 （５）Android ハードウェアアクセスインターフェース（libfriendlyarm-hardware.so）サポートの追加（システムに WatchDog、シリアル、LED、PWM、A/D、IIC、GPIO、SD インターフェース、USB カメラなどのサンプルコードがある） （６）VNC　Server サポート （７）adb デバッグサポート （８）USB カメラサポート （９）MF210　3G モジュールサポート （１０）重力センササポート **・Lunux 更新：** （1）抵抗スクリーンサポートの追加（S70、w101 など） （2）USB　3G サポートの追加 （3）USB カメラサポートの追加 （4）UI 最適化した後の Smplayer サポートの追加（コア：mplayer、複数のデータ書式をサポート） （5）RT8192CU に基づく Mini USB Wifi サポートを追加 （6）Qt バージョンを Qt/E-4.8.5 にアップデートし、回転機能サポートを追加 （7）Qt/E-4.8.5 に基づく WebKit ネットワークブラウザ Arora（豊富な機能で Web ページ内容の正常レンダリングを実現）を追加 （8）統合 python の追加でスクリプトプログライングのサポート、ハードウェアと C プログラムライブラリへのアクセスを実現（IOT 開発（ブザーサンプルが/opt/python/pwm.py に格納されている）に適用） （9）リモートアクセスとデバイスマネージャ、ファイル伝送などに使われる統合 ssh を追加（リモートデバッグ開発と管理（ユーザー名：root　パスワード：fa）に適用） （10）ftp による開発ボードへのファイル伝送時のユーザー名とパスワード問題の修正 **・Superboot と MiniTools 更新：** （1）Android　Data パテーションの書き込みサポートの追加 （2）eMMC サイズの表示が正しくない問題の修正 （3）Android でのデータパティーションにさらに大き利用できるスペースを取らせるため、eMMC のサイズによるスマートパティーションを実現 （4）TrustZone セキュリティモードの起動。このバージョンの Superboot に合わせて利用できるように、カーネルも TrustZone セキュリティモードを起動しなければいけない	

		(5) Android システムを書き込む場合、eMMC のサイズにより、異なる Userdata.ing ファイルを書き込む機能の追加。 **・カーネル更新：** (1) カーネルは TrustZone セキュリティモードを起動 (2) A/D 切替利用不可の問題の修正 **・モバイルインターネット関連製品の開発のため、Tiny4412 拡張ボードを更新：** 下記のインターフェースを追加 (1) Mini PCIe:市販の大部分の 3G モジュールとの接続に使える。(Andoroid で MF210 のドライバを開発しておいた。他の型番の３G モジュールを利用する場合、ドライバを移植必要) (2) RS485 インターフェースを添加	
4	Ver1.3	**Android5.0.2 追加、主な特徴：** （１）カーネルバージョンは Linux3.0.86 （２）赤外線リモコンサポート （３）HDMI 出力（最大 1080P）サポート、LCD と同時に表示可能、解像度設定 GUI もある （４）イーサネット、起動後自動的に接続をサポート、設定画面あり、Static IP 或いは DHCP モードで接続する （５）USB Wifi インターネットカードで Wifi ネットワークへの接続サポート （６）ハードウェア復号化でメディア再生可能 （７）Android ハードウェアアクセスインターフェース（libfriendlyarm-hardware.so）サポートの追加（シリアル、LED、PWM、A/D、IIC、GPIO、SD インターフェース、USB カメラなどのサンプルコードがある） （８）adb デバッグサポート （９）MF210 3G モジュールサポート （１０）重力センササポート	2015/03/31

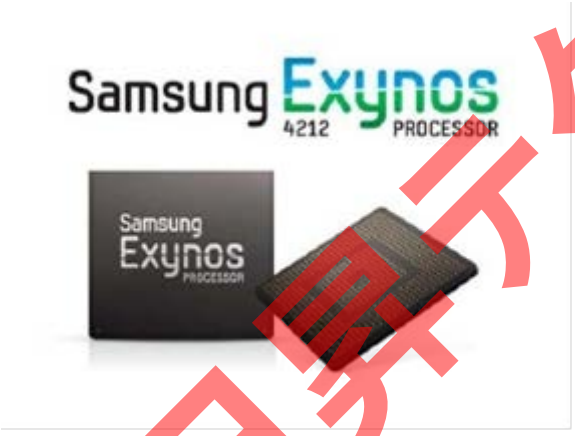
目次

第一章 Tiny4412 開発ボードの概要



(図：Tiny4412 コアボード)

Tiny4412 は高性能の Cortex-A9 コアボード、そしてサムソン(SAMSUNG)社の高性能 Exynos 4412 4コアプロセッサーを採用した。



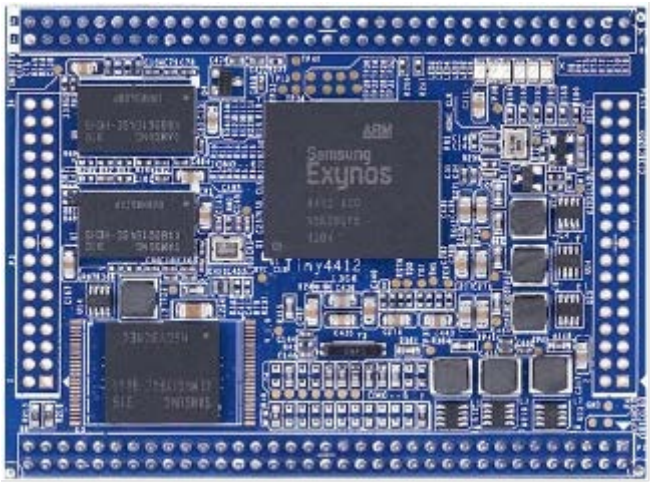
Exynos 4412 内部の GPU は Mali-400 MP の高性能グラフィックスエンジンを統合し、最高 1080P@30fps の HD ヴィデオプレイをサポート、Android などの先進なオペレーティングシステムをスムーズに実行でき；ハイエンドのネットワーク端末 、広告マルチメディア端末、スマートホーム、ハイエンド監視システムなどの開発に非常に適している。

1.1 Tiny4412 コアボード概要

Tiny4412 コアボードは 2.0mm ピッチの 2 列コネクタを採用し(P1、 P2、 P3、 P4)、ほとんどの CPU 機能ピンを引き出し、寸法は(74x55mm)。中に P1 と P2 ピンは標準半田付け、汎用機能を含む；P3 と P4 は空、ユーザー拡張開発に使用する。

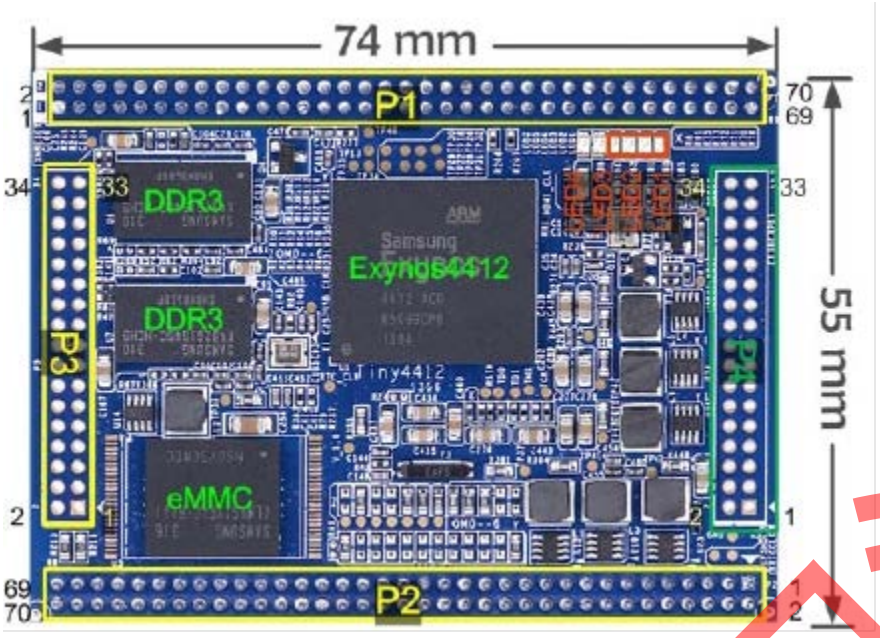
Tiny4412 標準搭載は 1G DDR3 メモリ と 4GB 高速 eMMC フラッシュメモリ。

1.1.1 Tiny4412 コアボードリソース特性

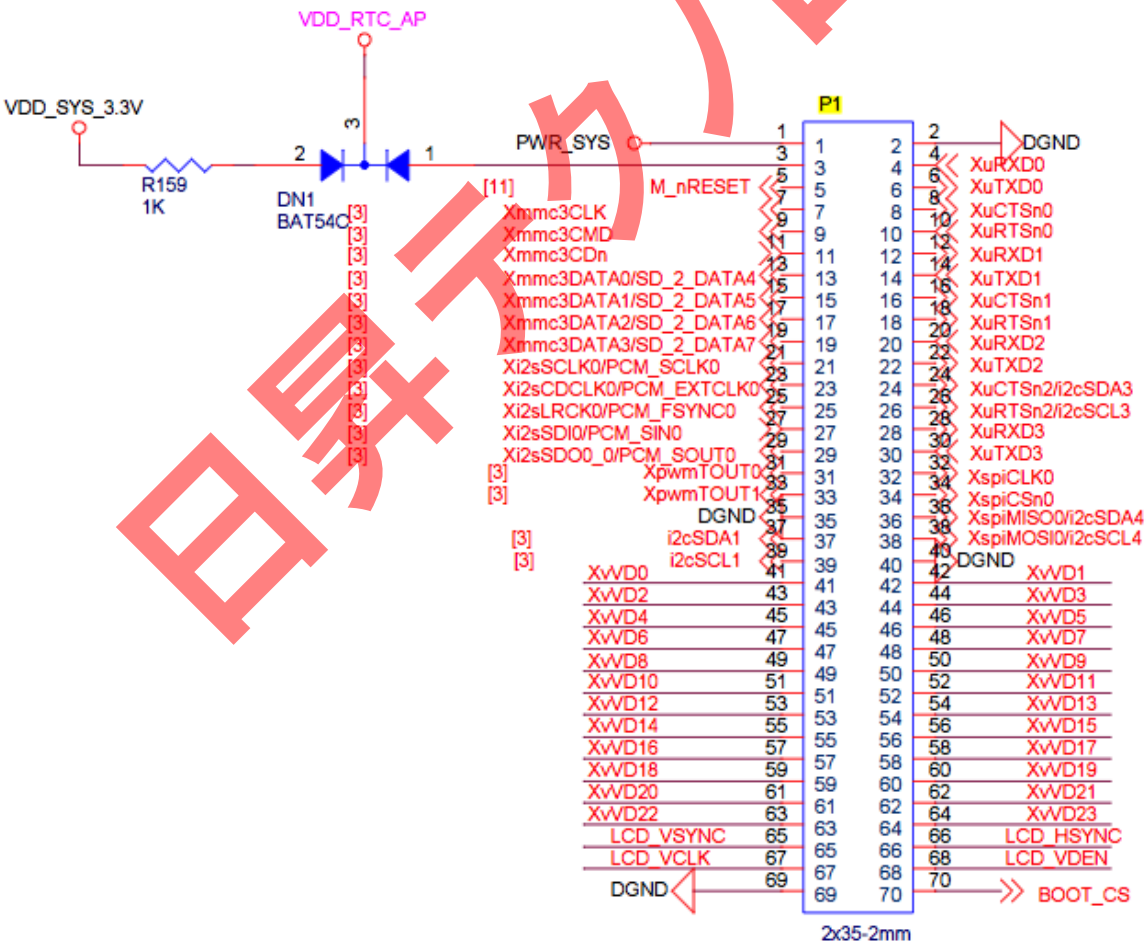


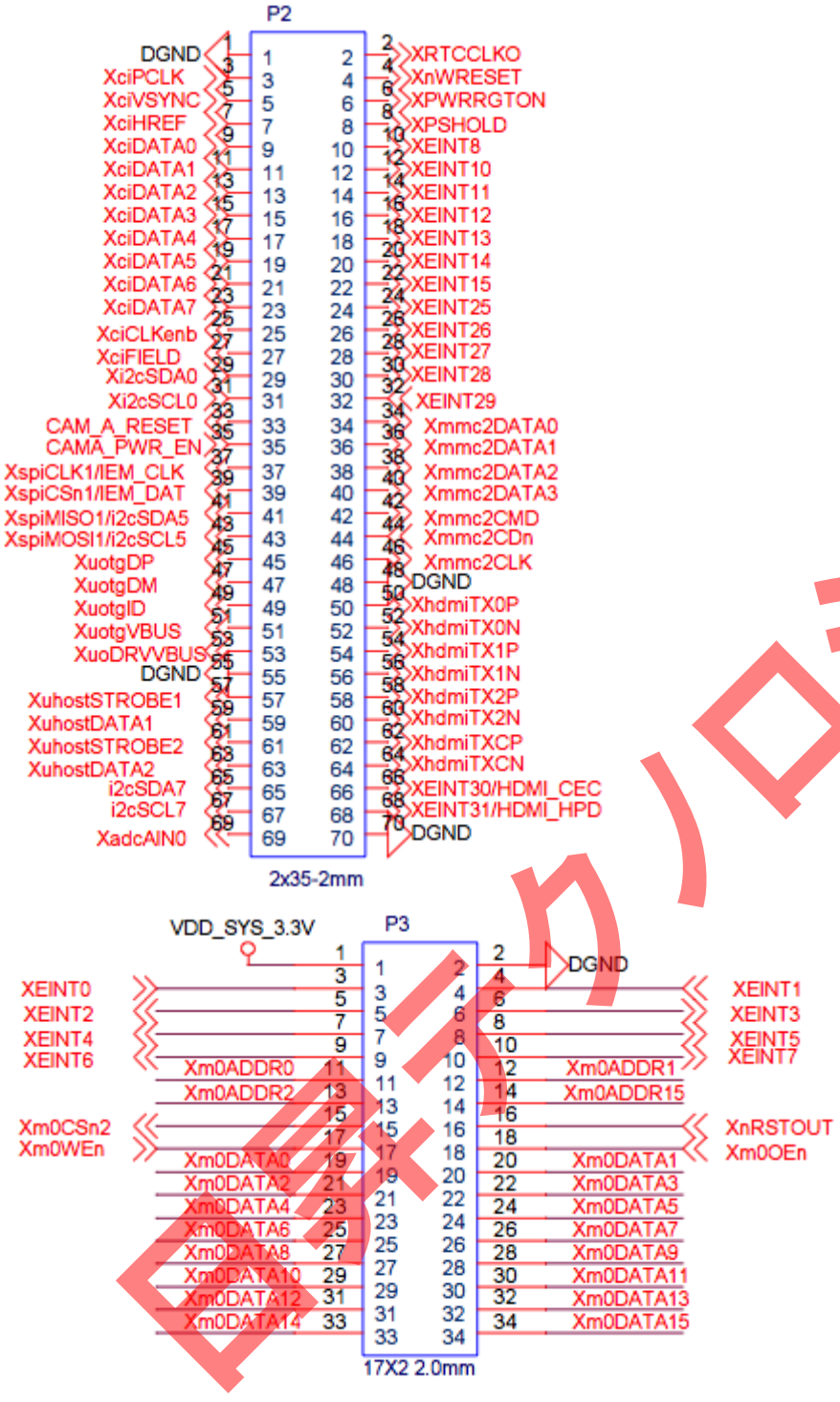
Item	Description
CPU プロセッサー	● Samsung ARM Cortex-A9 クアッドコア Exynos 4412 Quad-core プロセッサー、周波数 1.5GHz ● ARM Mali-400 デュアルコア GPU を内蔵 ● 最高 1080p@30fps のハードウェアデコードヴィデオプレイをサポート、 MPEG4、 H.263、 H.264 などのフォーマットをサポート ● 最高 1080p@30fps のハードウェアエンコード(Mpeg-2/VC1) ビデオ入力をサポート
DDR3 RAM メモリ	● Size: 1G ● 32bit データバス、シングルチャンネル
FLASH メモリ	● 標準配置 4G eMMC
インタフェースリソース	● 2 つの 70 Pin 2.0mm space DIP connector ● 2 つの 34 Pin 2.0mm space DIP connector
搭載リソース	● 4 x User Leds(Green)
供給電源	● Supply Voltage from 2V to 6V(スリープ/ウェイクアップ サポート)
PCB 規格・寸法	● 8 層の高密度回路基板、Immersion gold process ● Size: 74 x 55 x 10 (mm)

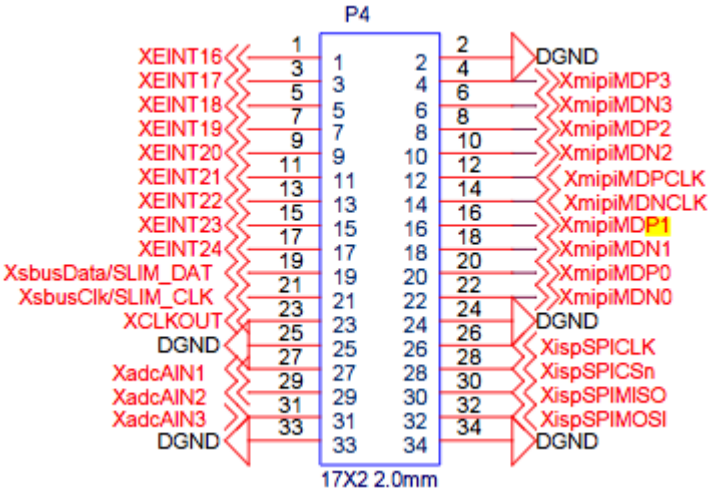
1.1.2 Tiny4412 コアボードピン定義



各インタフェース定義(P1~P4)



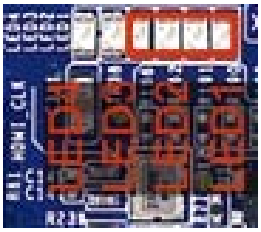

P2
DGND 1 2 XRTCCLKO
XciPCLK 3 4 XnWRESET
XciVSYNC 5 6 XPWRRGTON
XciHREF 7 8 XPSHOLD
XciDATA0 9 10 XEINT8
XciDATA1 11 12 XEINT10
XciDATA2 13 14 XEINT11
XciDATA3 15 16 XEINT12
XciDATA4 17 18 XEINT13
XciDATA5 19 20 XEINT14
XciDATA6 21 22 XEINT15
XciDATA7 23 24 XEINT25
XciCLKenb 25 26 XEINT26
XciFIELD 27 28 XEINT27
Xi2cSDA0 29 30 XEINT28
Xi2cSCL0 31 32 XEINT29
CAM_A_RESET 33 34 Xmmc2DATA0
CAMA_PWR_EN 35 36 Xmmc2DATA1
XspiCLK1/IEM_CLK 37 38 Xmmc2DATA2
XspiCSn1/IEM_DAT 39 40 Xmmc2DATA3
XspiMISO1/i2cSDA5 41 42 Xmmc2CMD
XspiMOSI1/i2cSCL5 43 44 Xmmc2CDn
XuotgDP 45 46 Xmmc2CLK
XuotgDM 47 48 DGND
XuotgID 49 50 XhdmiTX0P
XuotgVBUS 51 52 XhdmiTX0N
XuoDRVVBUS 53 54 XhdmiTX1P
DGND 55 56 XhdmiTX1N
XuhostSTROBE1 57 58 XhdmiTX2P
XuhostDATA1 59 60 XhdmiTX2N
XuhostSTROBE2 61 62 XhdmiTXCP
XuhostDATA2 63 64 XhdmiTXCN
i2cSDA7 65 66 XEINT30/HDMI_CEC
i2cSCL7 67 68 XEINT31/HDMI_HPD
XadcAIN0 69 70 DGND
2x35-2mm
VDD_SYS_3.3V
P3
DGND
XEINT0
XEINT2
XEINT4
XEINT6
XEINT1
XEINT3
XEINT5
XEINT7
Xm0ADDR0
Xm0ADDR2
Xm0ADDR1
Xm0ADDR15
Xm0CSn2
Xm0WEn
XnRSTOUT
Xm0OEn
Xm0DATA0
Xm0DATA2
Xm0DATA4
Xm0DATA6
Xm0DATA8
Xm0DATA10
Xm0DATA12
Xm0DATA14
Xm0DATA1
Xm0DATA3
Xm0DATA5
Xm0DATA7
Xm0DATA9
Xm0DATA11
Xm0DATA13
Xm0DATA15
17X2 2.0mm



1.1.3 Tiny4412 コアボードインタフェース説明

1.1.3.1 ユーザーLED

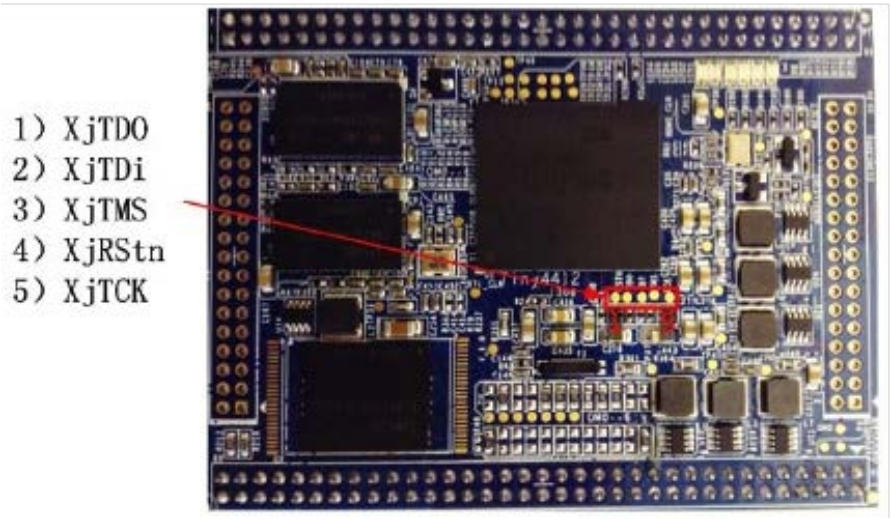
LED は開発中、最も基本に使用されるステータス表示デバイスで、本開発ボードでは 4 つのユーザー書込み LED があり、直接 CPU の GPIO と接続し、ローレベル有効(点灯)、リソース使用の詳細は下記の通り：



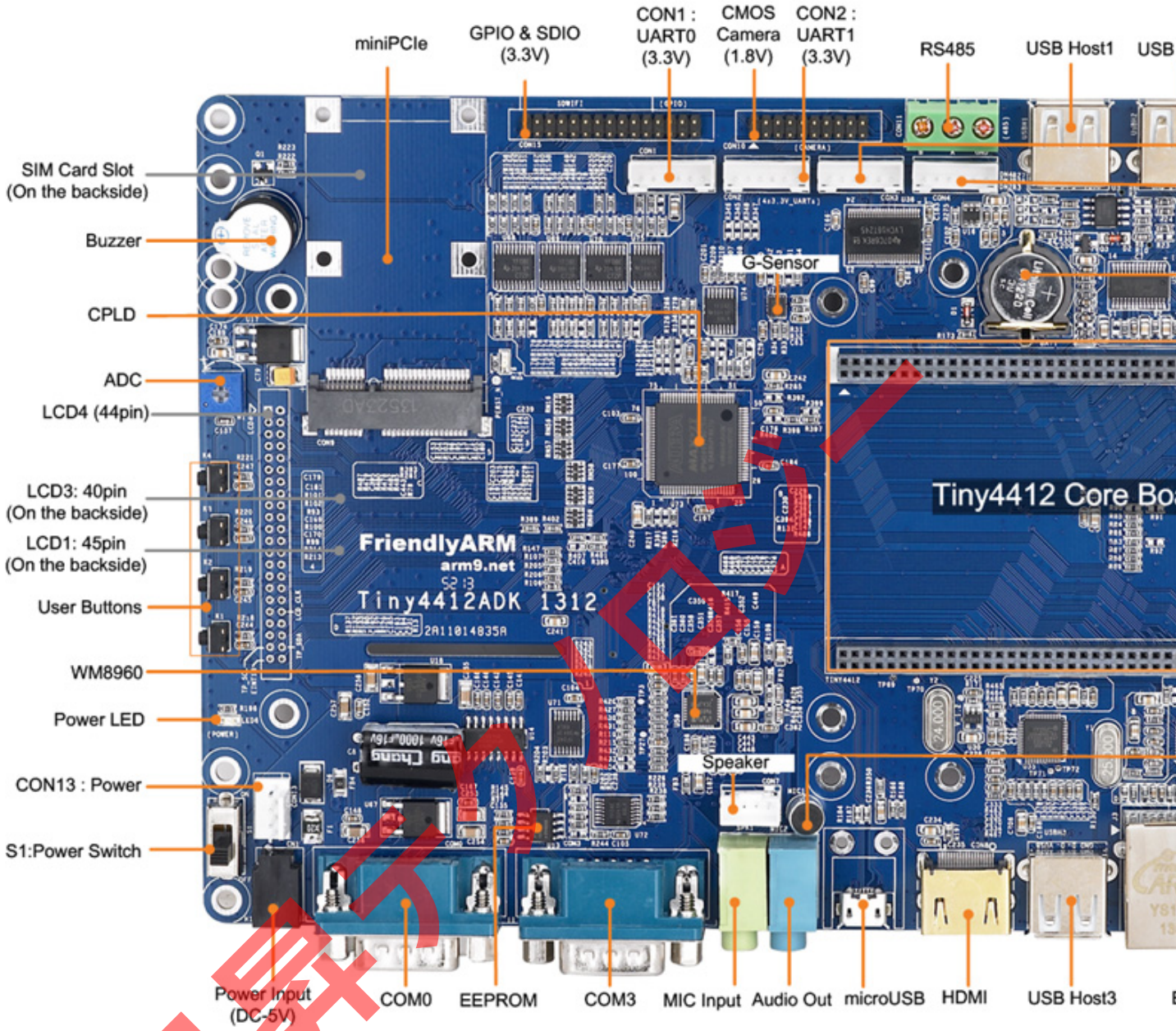
	LED1	LED2	LED3	LED4
GPIO リソース	GPJ_0	GPJ_1	GPJ_2	GPJ_3

1.1.3.2 JTAG インタフェース説明

現在多くのハイエンド CPU は SD カード起動をサポートする。JTAG インタフェースはよく使わない、そして、チップメーカーによりの JTAG インタフェース資料やソフトウェアのサポートも少ないが、依然として、 JTAG でデバッグ・開発に使用ユーザーがいる。開発ボードのスペースの制限で、Tiny4412 には JTAG テストポートを用意していて、ユーザーから引き出して、使用出来る。



1.2 Tiny4412SDK 拡張ボード仕様



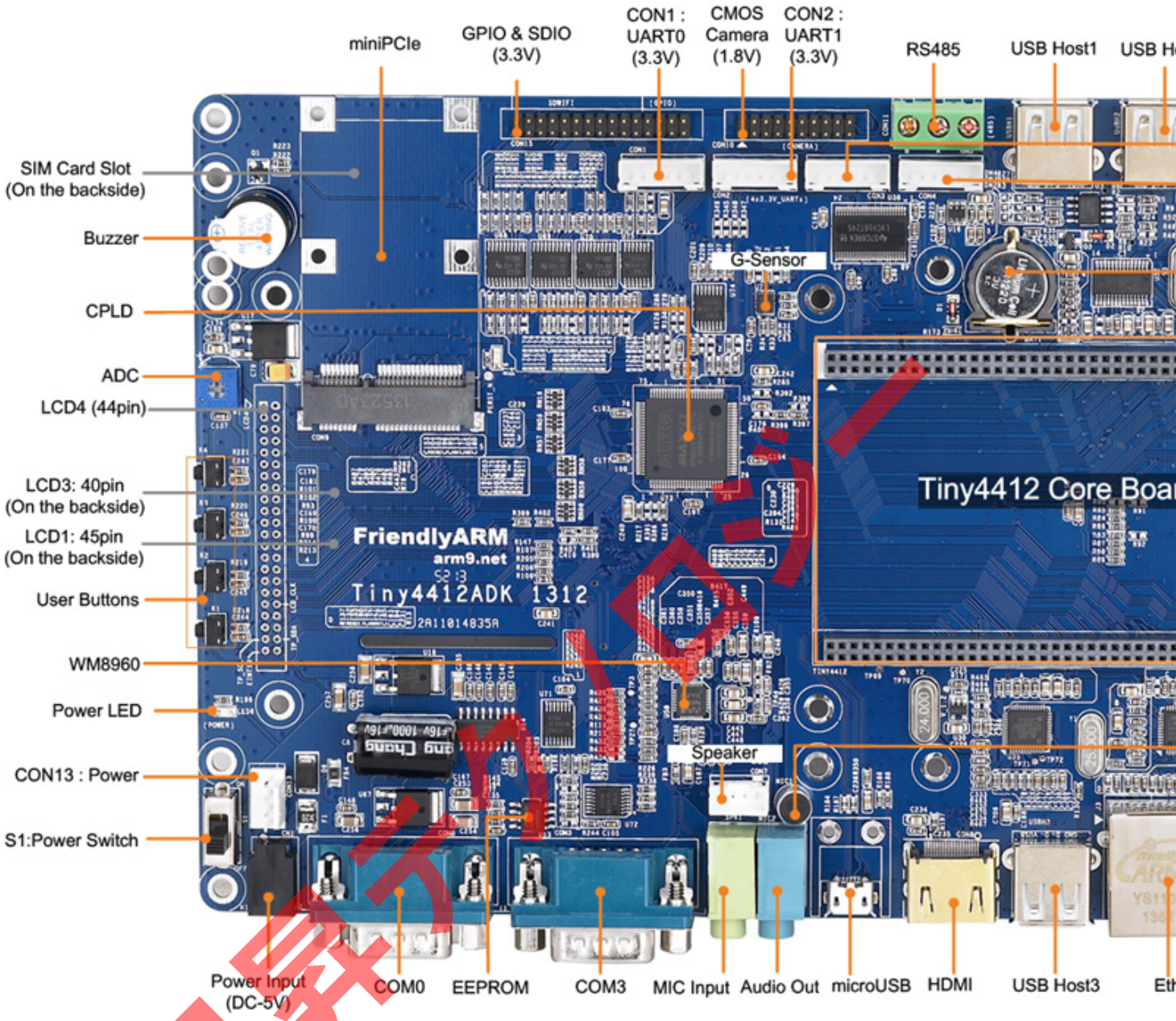
1.2.1 Tiny4412 SDK 拡張ボードハードウェアリソース特性

LCD	● LCD1(背面)： 45Pin、 0.5mm ピッチ、 Mini2440/Mini6410/Mini210 LCD をサポート ● ディスプレイ、ラインプロトコルのバックライト調整と静電容量式タッチをサポート ● LCD3(背面)： 40Pin、 0.5mm ピッチ、Mini2440/Mini6410/Mini210 LCD をサポート ● ディスプレイ、one-wire touch をサポート ● LCD4(前面未引き出し)： 44Pin、 Mini2440/Mini6410/Mini210 LCD をサポート、one-wire touch と静電容量式タッチをサポート、HDMI イ

	ンタフェース(Type A)をサポート ● LCD ・ 3.5゛から 12.1゛までの各種液晶パネル、HD LCD をサポート
ネットワーク	● 1 つの 10/100M イーサネット RJ45 インタフェース(DM9621)
標準インタフェースリソース	● □2 つの DB9 式 RS232 シリアル(他にまた 4 つの TTL レベルシリアルがある) ● 1 つの RS485 インタフェース ● 1 つの MiniPCIe インタフェース ● □1 つの micro USB Slave 2.0 インタフェース ● 1 チャンネル 3.5mm ステレオオーディオ出力インタフェース、1 チャンネルオンボード MIC 入力 ● □1 チャンネル USB Host 2.0 インタフェース ● □1 つの標準 SD カードスロット ● □5V 直流電圧入力:インタフェースソケットモデルは DC-23B
その他搭載リソース	● 1 つの I2C-EEPROM チップ(256byte)、I2C バステストに使用する ● 4 つのユーザーボタン(割り込み式リソースピン) ● 1 つの PWM 制御ブザー ● リアルタイムクロックバックアップバッテリー ● 重力センサーチップ
拡張インタフェースリソース	● 4 つのシリアルソケット ： TTL レベル ● 1 つの GPIO インタフェース (SDIO 含む) ● 1 つの CMOS カメラインタフェース
PCB 規格・寸法	● 層の数 ： 2 ● Size: 180 x 140(mm)
ソフトウェアサポート	● Linux Kernel 3.5 ● Android 4.2.1

1.2.2 Tiny4412 SDK レイアウトとジャンパ

Tiny4412 拡張ボードインタフェースレイアウト下記図の通り：



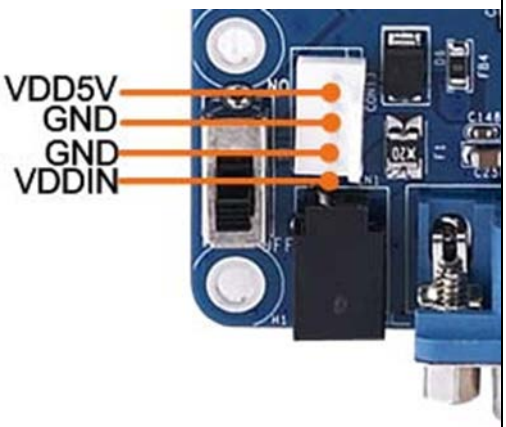
1.3 Tiny4412 開発拡張ボードインタフェース説明

本節は開発ボード上の各インタフェース、モジュールのピン定義と CPU リソースを説明する。もっと詳しい内容は PDF ファイルの回路図をご参照ください。

1.3.1 電源コネクタとソケット

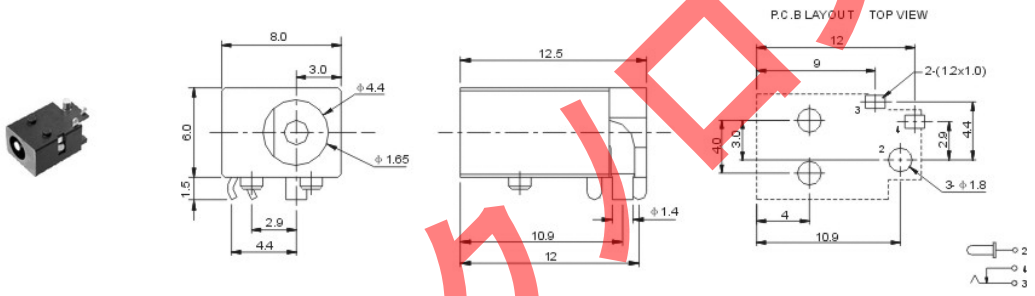
本開発ボードは 5V 直流電源給電で、2 つの電源入力口を提供する。CN1 は 5V 電源アダプターソケット、白の CON5 は 4Pin ソケット、ボードをクローズドシャーシで利用する時便利に電源を接続に使用する。

CON13	NO.	ピン定義
	1	VDD5V

	2	GND
	3	GND
	4	VDDIN
	説明：本接続方法で引き出す場合、電源スイッチ S1 も有効である。	

電源ソケットモデルと寸法：

Type: DC023B

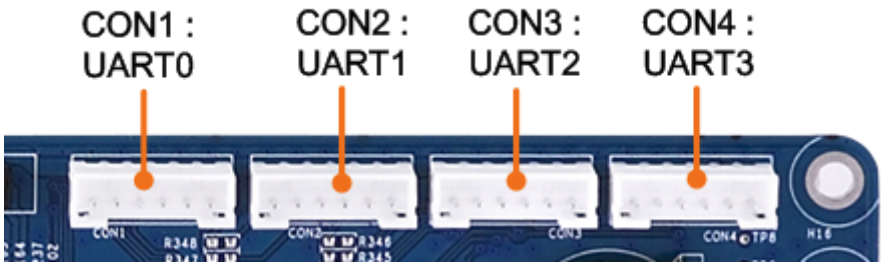


1.3.2 シリアルポート

Exynos4412 には 4 つのシリアルポートがあり、その中 UART1 は 4 ラインの機能シリアル、UART0、2、3 は 2 ラインのシリアルである。

開発ボード上、UART0 と UART3 は既に RS232 レベル変換を行なって、それぞれ COM0 と COM3 と対応し、付属のクロスシリアルケーブルを使用し PC と通信出来る。

CON1、CON2、CON3、 CON4 は開発ボード上の位置と回路図の接続定義対応は下記図の通り：。



CON0 ～CON4	ピン定義(TTL)	COM0	ピン定義(RS232)
1	RTSn	1	NC
2	CTSn	2	RSRXD

3	TXD	3	RSTXD
4	RXD	4	NC
5	5V	5	GND
6	GND	6	NC
		7	NC
		8	NC
		9	NC

COM3	ピン定義(RS232)
1	NC
2	RSRXD
3	RSTXD
4	NC
5	GND
6	NC
7	RSCTSn
8	RSRTSn
9	NC

1.3.3 USB インタフェース

開発ボードで１つの USB Host(2.0)インタフェースを提供する。USB カメラ、USB 無線 LAN 装置、USB マウスとキーボード、U disk などの関連 USB デバイスを使用出来る；

開発ボードでまた１つの microUSB(2.0)インタフェースがあり、主には Android OS の ADB 機能、ソフトウェアのインストールとプログラムデバッグに使用する。

microUSB インタフェース定義は下記の通り：

miniUSB	ピン定義
5	GND
4	OTGID
3	D+
2	D-
1	Vbus

USB Host のインタフェース定義は下記の通り：

USB Host	ピン定義
1	5V
2	D-
3	D+

	4	GND

1.3.4 ネットワークインタフェース

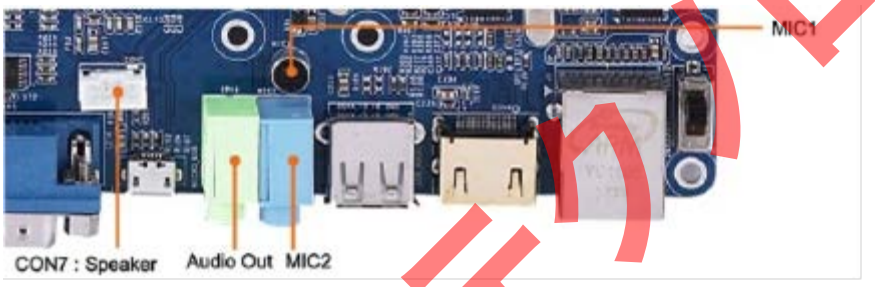
本開発ボードは有線ネットワーク DM9621 チップを使用し、10/100M ネットワークサポート。RJ45 コネクタ内部には、既に結合コイルが含まれているため、一般ネットワークケーブルで本開発ボードとルーターまたはスイッチに直接接続し、ネットワークトランスフォーマーを仲介接続する必要はない。

1.3.5 オーディオインタフェース

Exynos4412 は I2S/PCM/AC97 などのオーディオインタフェースをサポートし、本開発ボードは I2S0 インタフェースを採用し、CODEC デコードチップとして WM8960 を外接する、HDMI オーディオとビデオ同期出力をサポート、WM8960 チップは Tiny4412SDK 拡張ボードに搭載している。

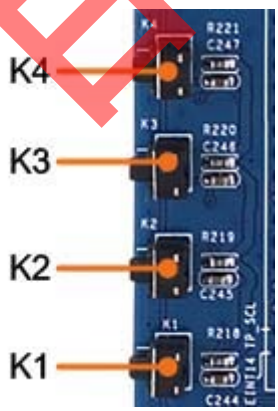
オーディオシステムの出力は開発ボード上の汎用 3.5mm ソケット（緑の audio out)。

便利に開発/使用するため、開発ボードでは MIC を搭載している。オーディオ入力デバイスはテスト用であって、録音専用設備ではないので、録音する時はなるべく音源を MIC に近づいて行う必要。



1.3.6 ユーザーボタン

本開発ボードでは合計 4 つのユーザーテスト用ボタンがある。それを全て CPU 割り込みピンから直接引き出し、ローレベル有効、4 つのボタンの定義は下記の通り：



ボタン	K1	K2	K3	K4
対応割り込み	EINT26	EINT27	EINT28	EINT29
AF	GPX3_2	GPX3_3	GPX3_4	GPX3_5

GPIO				

1.3.7 LCD インタフェース

Tiny4412SDK で 3 つの LCD インタフェースがある、1 つは 45pin (LCD1)、静電容量式タッチスクリーンを接続する。

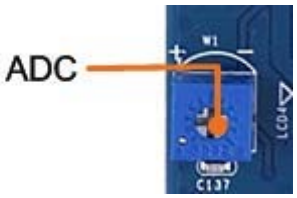
LCD インタフェーススロットには汎用 LCD 使用する制御信号を大部分含めている(フィールド･スキャン、クロックとイネーブル信号など)、と完全な RGB データ信号(RGB 出力は 8:8:8、即最高は 1600 万色の LCD をサポートする)；他に PWM 出力を引き出し、リセット信号(nRESET)、その中 LCD_PWR はバックライト制御信号。

注：one-wire touch を採用するため、LCD1 スロットには CPU 内蔵の 4 ライン式抵抗タッチピンを含まない、代わりに I2C と割り込みピンを追加する(LCD1-41、42、43、44)、このような設計は静電容量式タッチスクリーンを使用する為です。

LCD1	ピン説明	LCD1	ピン説明
1	VDD_5V	2	VDD_5V
3	VD0	4	VD1
5	VD2	6	VD3
7	VD4	8	VD5
9	VD6	10	VD7
11	GND	12	VD8
13	VD9	14	VD10
15	VD11	16	VD12
17	VD13	18	VD14
19	VD15	20	GND
21	VD16	22	VD17
23	VD18	24	VD19
25	VD20	26	VD21
27	VD22	28	VD23
29	GND	30	PWM1/GPD0_1
31	XEINT10/GPH1_2	32	nRSTOUT
33	VDEN	34	VSYNC
35	HSYNC	36	VCLK
37	I2CSCL2	38	XEINT14/GPH1_6
39	I2CSDA2	40	XEINT15/GPH1_7
41	GND	20	
注：Exynos4412 には合計 3 チャンネル I2C がある。ここで使用するのは I2C2。 説明：各ピンと CPU の接続関係、回路図を標準とする、ここでの定義は参照のみ			

1.3.8 ADC 入力

Tiny4412 コアボードには 4 チャンネル ADC 変換チャンネルがあり、その中 AIN0 は開発拡張ボードの変調抵抗 W1 に接続、その他のチャンネルは まだ Tiny4412SDK 拡張ボードでは引き出してない。



1.3.9 PWM 制御ブザー

本開発ボードのブザー Buzzer は PWM0 を通じ制御する。PWM0 は GPD0_0 と対応、そのピンはソフトウェアで PWM 出力に設定し、または一般の GPIO として使用出来る。

1.3.10 I2C-EEPROM

本開発ボードでは１つ直接 CPU の I2C0 信号ピンと接続するの EEPROM チップ AT24C08 があり、容量は 256 byte、ユーザーが I2C バスをテストするために使用する、特定なパラメータを保存してない。

注：Exynos4412 では合計 8 チャンネル I2C があり、ここでは I2C0 を使用する。

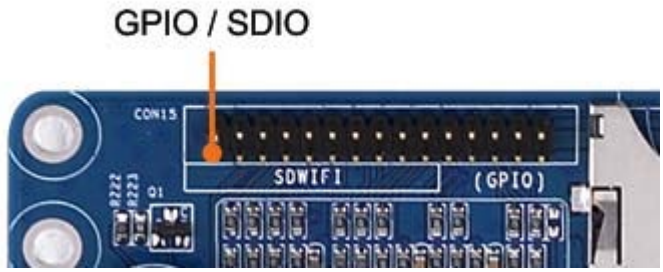
1.3.11 SD カード

Tiny4412 は 2 チャンネル SDIO インタフェースを引き出す。本開発拡張ボードでは、SDIO2 は汎用 SD カ

ードインタフェースに使用する、 SDHC（高速大容量カード）をサポートする。

1.3.12 GPIO/SDIO インタフェース

GPIO は 2.0mm ピッチの 30Pin コネクタ:



その中、前の 20pin は一つの SDIO インタフェース、SD-WiFi モジュールと接続に使用する。

GPIO インタフェースのピン定義は下記の通り：

CON9	ピン定義	CON9	ピン定義
1	VDD3.3V	2	GND
3	TXD2	4	RXD2
5	I2CSCL	6	I2CSDA
7	SPIMOSI0	8	SPIMISO0
9	SPICLK0	10	SPICSn1
11	EINT13	12	EINT12
13	SD3_CLK	14	SD3_CMD
15	SD3_nCD	16	EINT11
17	SD3_DAT0	18	SD3_DAT1
19	SD3_DAT2	20	SD3_DAT3
21	SPIMISO1	22	EINT26
23	SPIMOSI1	24	EINT27
25	SPICLK1	26	EINT28
27	SPICSn1	28	EINT29
29	VDD5V	30	GND
説明：各ピンと CPU の接続関係、回路図を標準とする、ここでの定義は参照のみ			

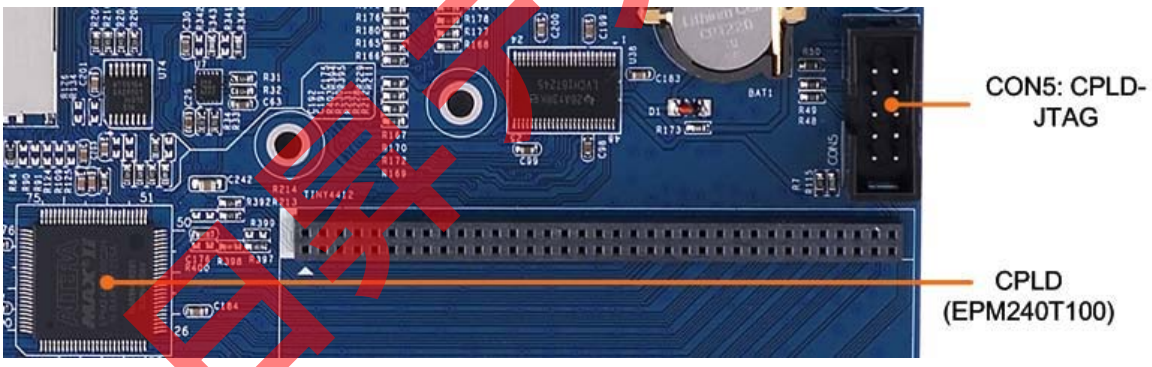
1.3.13 CMOS CAMERA インタフェース

Tiny4412 で１つの CMOS カメライインタフェース（20 ピンの 2.0mm ピッチ）を提供する。CAM130 カメラモジュール（別売、URL: http://www.csun.co.jp/SHOP/2009102501.html ）と直結できる；そして CAM130 カメラモジュール上では回路が無くて、１つのアダプタープレートとして、 ZT130G2 カメラモジュールと接続する。

説明：CAMERA インタフェースは複用ポートで、対応レジスタの設定で GPIO として利用出来る。対応ピンの GPIO は下記の通り：

CMOS CAMERA			
CAMERA	ピン定義	CAMERA	ピン定義
1	I2CSDA0	2	I2CSCL0
3	XciFIELD	4	CAM_RESET/GPJ3_1
5	CAM_CLK	6	CAM_HREF
7	CAM_VSYNC	8	CAM_PCLK
9	CAM_DATA7	10	CAM_DATA6
11	CAM_DATA5	12	CAM_DATA4
13	CAM_DATA3	14	CAM_DATA2
15	CAM_DATA1	16	CAM_DATA0
17	VDD_3.3V	18	VDD_2.45-2.8V
19	VDD_1.8V	20	GND
説明：各ピンと CPU の接続関係、回路図を標準とする、ここでの定義は参照のみ			

1.3.14 CPLD-JTAG インタフェース



Tiny4412SDK 上の CPLD (EPM240T100) ファームウェアを書き込むに使用する。

1.3.15 mini PCIe インターフェース

Tiny 拡張ボードは市販の大部分の 3G モジュールに接続するための MiniPCIe インターフェースを提供した。Android4 で MF210 のドライバを開発。他の型番の 3G モジュールを利用する場合、ドライバーを移植する必要。

1.4 Tiny4412 のソフトウェア特性

1.4.1 Android 4.1.2 システムリソース特性

クロスコンパイラ	arm-linux-gcc-4.5.1-v6-vfp	Mini6410/Mini210 と通用、カーネルをコンパイル時は自動的に ARMv7 命令セットを検索する。ハードウェア浮動小数点演算をサポートする。
Superboot-4412	Superboot と uboot を提供する	
	SD カードオフライン高速書込みをサポートする(1.8M/秒)	
Android カーネル	バージョン: Linux-3.0.31	
	EXT3/YAFFS2/CRAMFS/FAT32 等ファイルシステムをサポート	
	ウォッチドッグドライバー	
	RTC ドライバー	
	4 つの LED ドライバー	
	4 つのユーザーボタンドライバー	
	SPI ドライバー	
	I2C-EEPROM ドライバー	
	PWM 制御ブザードライバー	
	ADC ドライブ(チャンネル: AIN0)	
	CPU 内蔵のタッチスクリーンドライバー	
	静電タッチスクリーンドライバー	
	LCD バックライトドライバー、 127 級変調	
	LCD ドライバー(HD 7〞、一般 7〞)	
	USB Host ドライバー: USB メモリ、Bluetooth 等をサポート	
	USB Device ドライバー: USB ADB/fastboot サポート	
	SD カードドライバー	
	4 つのシリアルポートドライバ__	
	USB WiFi のドライバ	
	オーディオドライバー(WM8960:録音と再生をサポート、ALSA インタフェース、D クラスアンプをサポート)	
	イーサネット(DM9621)	
	FIMC ドライバー	
	JPEG ドライバー	
	MFC マルチメディアドライバー	
	HDMI ドライバー	
	3D 加速	

	2D 加速	
	USB シリアル変換のドライバー	
Android システム	バージョン: Android 4.1.2__r2.1	
アプリ特性	2D/3D 加速サポート	各種 2D/3D ゲームをスムーズに実行出来る
	WiFi サポート	
	3G とメッセージサポート	中興 MF210 3G モジュールサポート
	HDMI オーディオビデオ同期出力サポート	解像度設定可、最大 1080p サポート
	バックライト 127 級変調可能	
	その他	

1.4.2 Android 4.2.2 システムリソース特性

クロスコンパイラ	arm-linux-gcc-4.5.1-v6-vfp	Mini6410/Mini210 と通用、カーネルをコンパイル時は自動的に ARMv7 命令セットを検索する。ハードウェア浮動小数点演算をサポートする。
Superboot-4412	Superboot と uboot を提供する	
	SD カードオフライン高速書込みをサポートする(1.8M/秒)	
Android カーネル	バージョン: Linux-3.5	
	EXT3/YAFFS2/CRAMFS/FAT32 等ファイルシステムをサポート	
	ウォッチドッグドライバー	
	RTC ドライバー	
	4 つの LED ドライバー	
	4 つのユーザーボタンドライバー	
	SPI ドライバー	
	I2C-EEPROM ドライバー	
	PWM 制御ブザードライバー	
	ADC ドライブ(チャンネル: AIN0)	
	CPU 内蔵のタッチスクリーンドライバー	
	静電タッチスクリーンドライバー	
	LCD バックライトドライバー、 127 級変調	
	LCD ドライバー(HD 7〞、一般 7〞)	
	USB Host ドライバー: USB メモリ、Bluetooth 等をサポート	
	USB Device ドライバー: USB ADB/fastboot サポート	
	SD カードドライバー	

	4 つのシリアルポートドライバ__	
	USB WiFi のドライブ、カーネル内蔵、一部のカードに対してのサポートはまだ不十分	
	USB WiFi ドライバー：より多いのモデルの USB WIFI をサポート	
	オーディオドライバー(WM8960:録音と再生をサポート、ALSA インタフェース、D クラスアンプをサポート)	
	イーサネット(DM9621)	
	FIMC ドライバー	
	JPEG ドライバー	
	MFC マルチメディアドライバー	
	HDMI ドライバー	
	3D 加速	
	2D 加速	
	USB シリアル変換のドライバー	
Android システム	バージョン: Android 4.2.2	
アプリ特性	2D/3D 加速サポート	各種 2D/3D ゲームをスムーズに実行出来る
	WiFi サポート	
	3G とメッセージサポート	中興 MF210 3G モジュールサポート
	HDMI オーディオビデオ同期出力サポート	解像度設定可、最大 1080p サポート
	バックライト 127 級変調可能	
	その他	

1.4.3 Linux システムリソース特性

クロスコンパイラ	arm-linux-gcc-4.5.1-v6-vfp	Mini6410/Mini210 と通用、カーネルをコンパイル時は自動的に ARMv7 命令セットを検索する。ハードウェア浮動小数点演算をサポートする。
Superboot-4412	Superboot と uboot を提供する	
	SD カードオフライン高速書込みをサポートする(1.8M/秒)	
Linux カーネル	バージョン: Linux-3.5	完全な BSP
	YAFFS2/CRAMFS/FAT32 等ファイルシステムをサポート	ソースコード、
	ウォッチドッグドライバー	ソースコード
	RTC ドライバー	ソースコード
	4 つの LED ドライバー	ソースコード
	8 つのユーザーボタンドライバー	ソースコード
	SPI ドライバー	ソースコード
	I2C-EEPROM ドライバー	ソースコード
	PWM 制御ブザードライバー	ソースコード
	ADC ドライバー(チャンネル: AIN0)	ソースコード
	CPU 内蔵のタッチスクリーンドライバー	ソースコード
	一線タッチドライバー	ソースコード
	LCD バックライトドライバー、 127 級変調	スースコード
	LCD ドライバー(4.3〞、5〞、7〞等)、回転可能	スースコード
	USB Host ドライバー: USB メモリ、Bluetooth 等をサポート	スースコード
	USB Device ドライバー: USB ADB サポート	スースコード
	SD カードドライバー	スースコード
	4 つのシリアルポートドライバー	ソースコード
	SDWiFi ドライブ (Marvel18686)	ドライブモジュール
	USB WiFi のドライバー、カーネル内蔵、一部のカードに対してのサポートはまだ不十分	ソースコード
	USB WiFi ドライバー：より多いのモデルの USB WIFI をサポート	ドライブモジュール
	オーディオドライバー(WM8960:録音と再生をサポート、ALSA インタフェース、D クラスアンプをサポート)	ドライブモジュール

	イーサネット(DM9000)	ソースコード
	FIMC ドライバー	ソースコード
	JPEG ドライバー	ソースコード
	MFC マルチメディアドライバー	ソースコード
	3D 加速	ソースコード
	2D 加速	ソースコード
	USB シリアル変換のドライバー	ソースコード
	3G ドライバー:実際は USB シリアル変換のドライバー	ソースコード
GUI	Qtopia-2.2.0	ソースコードを提供する(x86、arm バージョン)
	QtEmbedded-4.7.0	ソースコードを提供する(arm バージョン)
	Qt-Extended-4.4.3	携帯版の Qtopia、Qtopia4 も呼べる、ソースコードを提供する
アプリ	ADC 転換テスト	
	LED 制御	
	Buttons ボタンテスト	
	12C-EEPROM テスト	
	LCD テスト	
	Ping テスト	
	レコーダーテスト	
	Web ブラウザ	
	ウォッチドッグテスト	
	インターネット設定（データ保存可能）	
	バックライト制御	
	言語設定：中英語設定可能	
	書く：タッチペンの正確率テスト	
	MMC/SD カードと USB フラッシュメモリの自動的ロードとアンロード	
	Qt4	
	Qtopia4	
	SMPlayer	

第二章 システムインストール

出荷時 Android4.2.1 システムをインストールされている。(/images/Android フォルダにあるバイナリファイル:zImage、ramdisk-u.img、system.img)

事前準備として、本章 1、2 節の内容を説明、参照する。

2.1 開発ボード設定と接続

2.1.1 起動モード選択

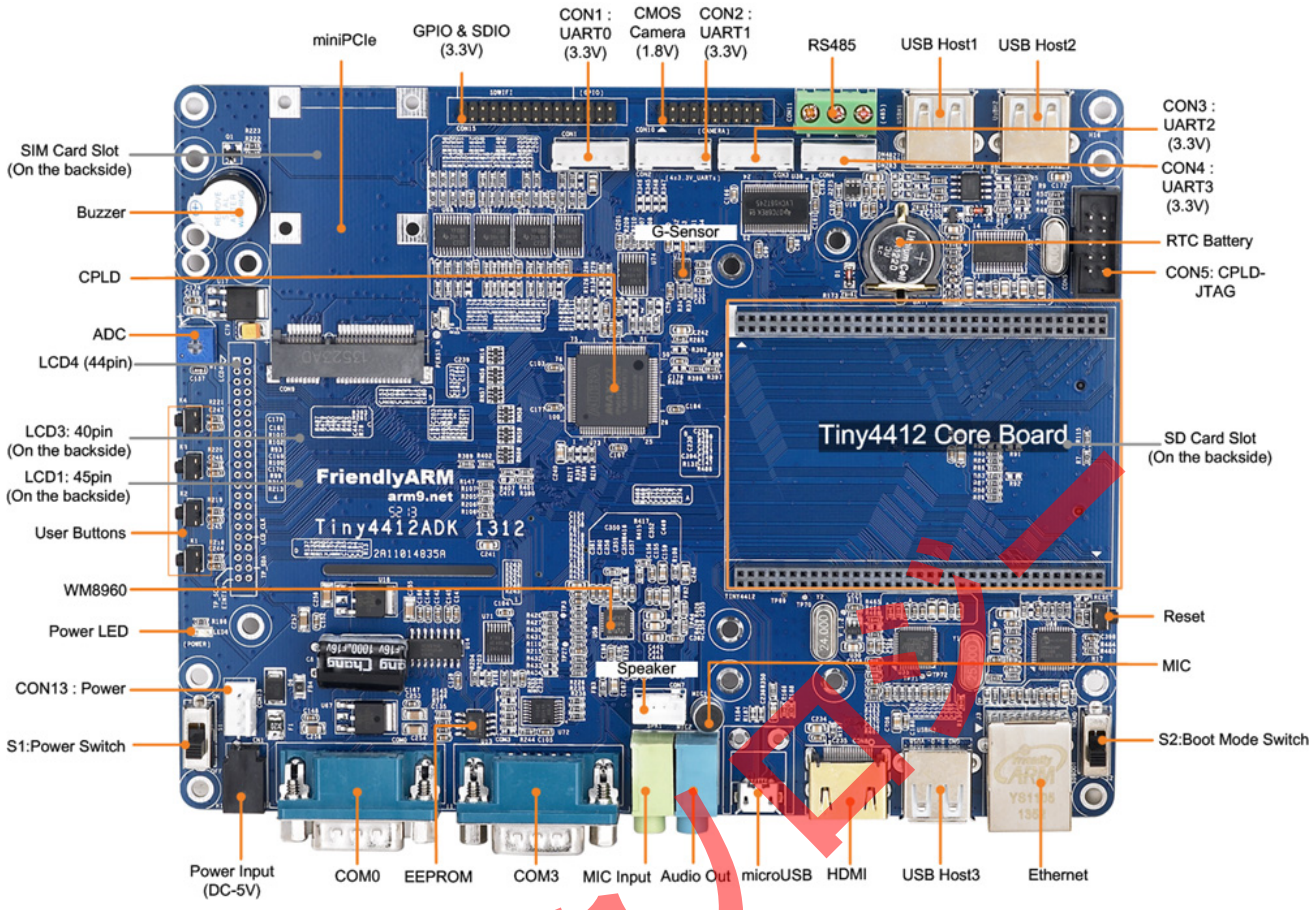
Tiny4412 は SD カードと eMMC の 2 つの起動モードがあり、S2 スイッチで起動モードを切り替える：

アイコン	説明	機能
	S2 を NAND 標識側に切り替えると、システムは eMMC で起動。	システム正常起動
	S2 を SDBOOT 標識側に切り替えると、システムは SD カードで起動。	システム書込み、または SD カード起動に使用する

開発ボードは普段使用する時、S2 は NAND 側に設定する。システムを書込みまたは SD カードで起動する時には SDBOOT 側に切り替える。

2.1.2 外部インタフェース接続

Tiny4412SDK には下記の外部インタフェースがある：



Tiny4412 開発ボードを初回使用する時、下記の手順を参照する：

- ☐ 付属のクロスシリアルケーブルを使用し(青い)開発ボードのシリアル 0 と(図の COM0) PC 側のシリアルと接続
- ☐ 製品提供する 5V 電源アダプターでボードの 5V 入力ソケットと接続(強引に挿し込みは控えください)
- ☐ スピーカーまたはヘッドフォンのソケットをボードの緑のオーディオ出力口に挿し込み
- ☐ LCDがある場合、データケーブルヘーダーの方向で開発ボードの LCD インタフェースと接続（LCD インタフェースは背面にある）

2.1.3 ハイパーターミナル設定

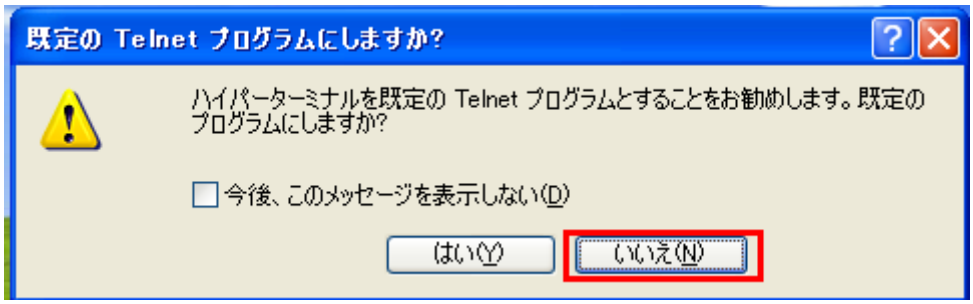
付属のシリアルクロスケーブル以外のシリアルケーブルを使用する場合、マルチメータでクロスケーブルを判別すれば使用出来る。

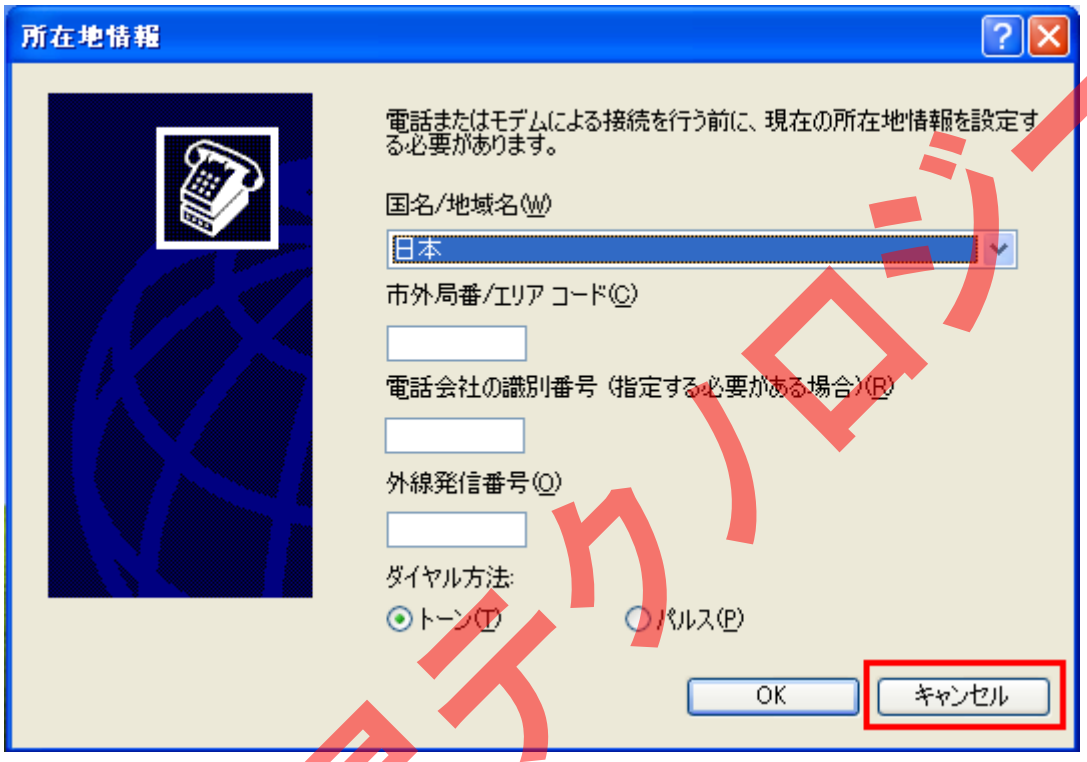
シリアルで開発ボードと接続するには、ターミナルのシミュレーションプログラムが必要とする、大部分のソフトウェアは使用可能で、中に MS-Windows 内蔵のハイパーターミナルは一番使われる、Windows9x OS ではカスタマイズでインストールし；Windows2000 またはその以上のバージョンではデフォルトインストールされてる；Windows7 では putty を使用する。

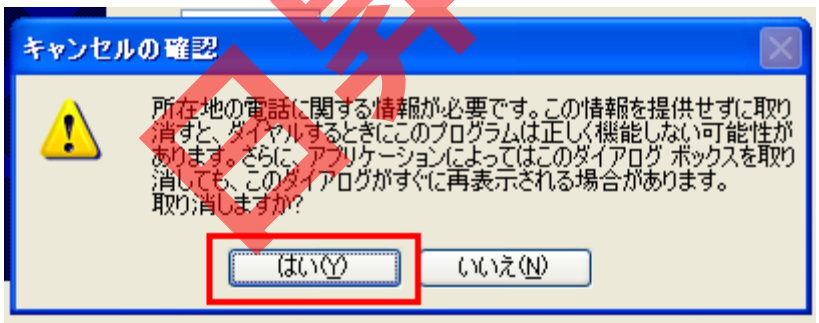
Linux システムにも類似のシリアルターミナルソフトウェア -minicom がある。基本コマンドラインプログラムに基づき使用出来る。

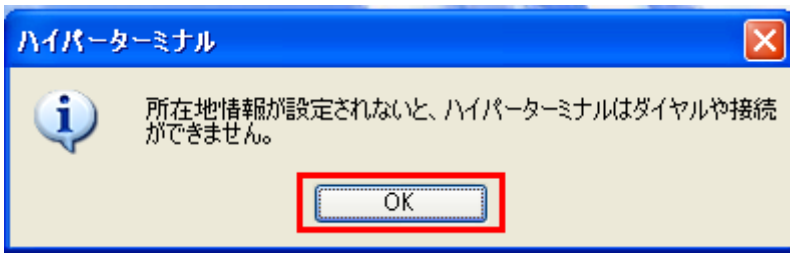
Windows のハイパーターミナルプログラムを例として説明する。

1、［スタート］-［すべてのプログラム］-［アクセサリ］-［通信］-［ハイパータミナル］の順にクリック。

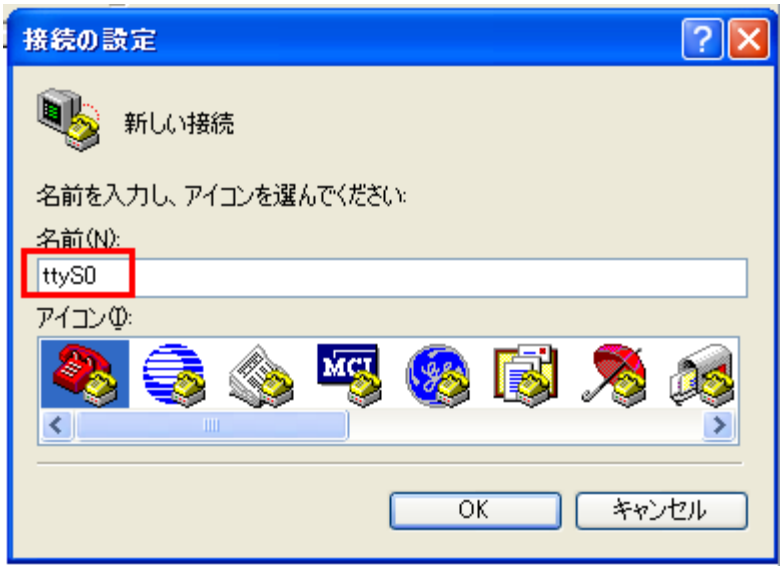




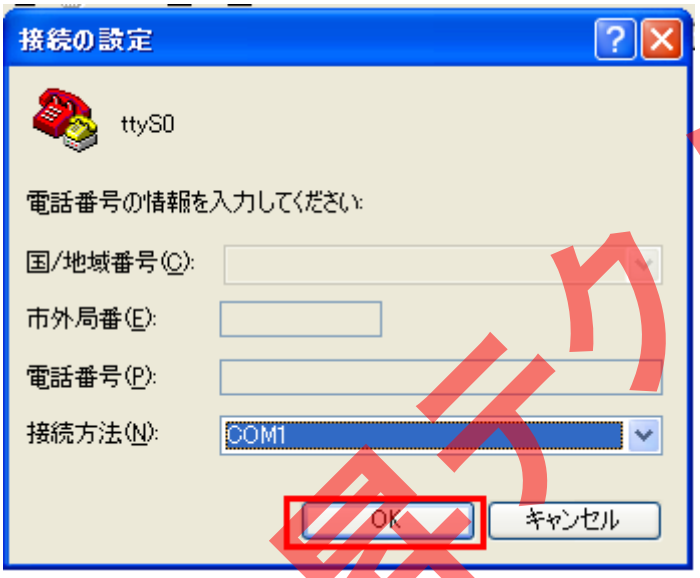




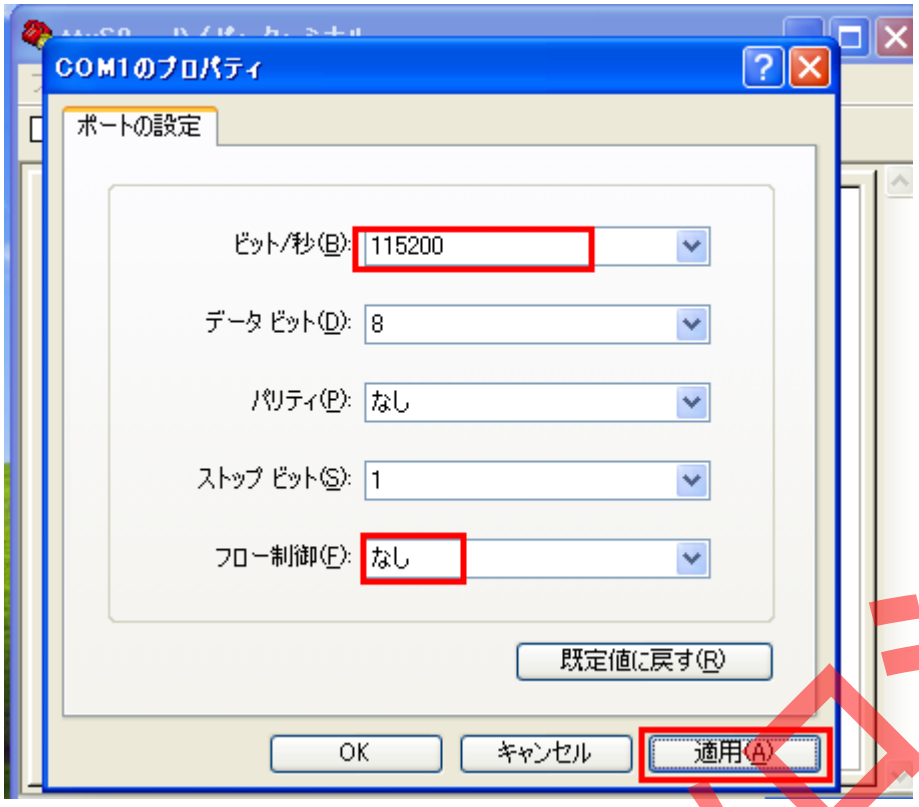
ハイパーターミナルで新しい接続を命名する。Windows システムは｀COM1｀などの命名は禁じられるため、ここでは｀ttyS0｀とする。



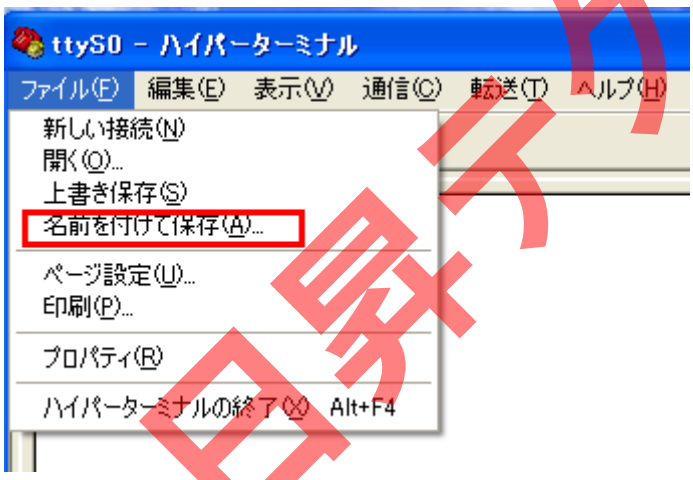
命名完了後、また所在地情報が提示する、キャンセルする。開発ボードのシリアルと接続するため、ここではシリアル 1 を選択：



最後は、シリアル設定、フロー制御はなし、ボーレートは 115200：



全てのパラメータが接続完了後、電源を入れて、ハイパーターミナルでシステム起動画面が表示する。
ハイパーターミナル゛ファイル゛メニューの゛名前を付けて保存゛、設定を保存する。



2.2 システムインストール用の SD カードを作成する

Tiny4412 は出荷時には既に Android4.2.2 OS が書き込み済みで、以後の書込みは SD カードを介し、OS を書き込む。USB または SD カードでオフライン書込みは共に SD カードで開発ボードを起動する必要がある。

プロセス：

Tiny4412 の Bootloader （Superboot4412.bin）を SD カードの１つ目のパーティションに書込み、Tiny4412 は直接 SD カードから起動出来る。Superboot の SD カードオフライン書込み機能を使用し、システムを eMMC に書き込んで、OS のインストールを実現する。

2.2.1 SD-Flasher ツールで Superboot を SD カードに書込む

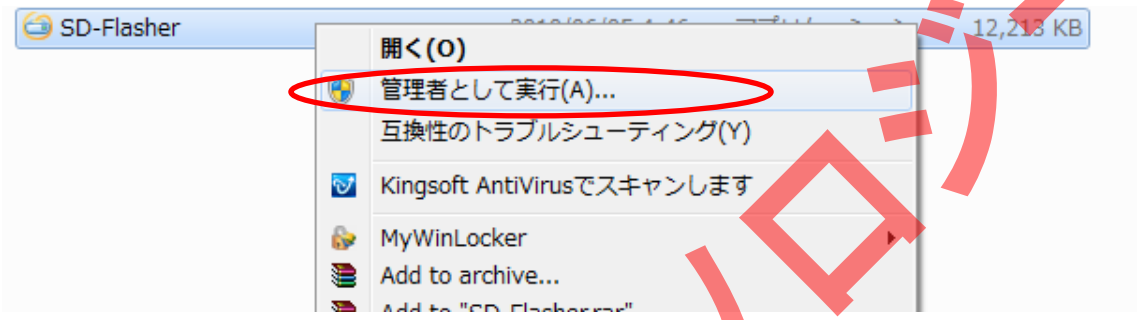
手順は下記の通り：

SD-Flasher.exe は SD カードのパーティションを実行する、１つ目のパーティションは 130M、Superboot 保存に使用する、残りのスペースは FRIENDLYARM パーティションに使用し、システムファイル images を保存する。

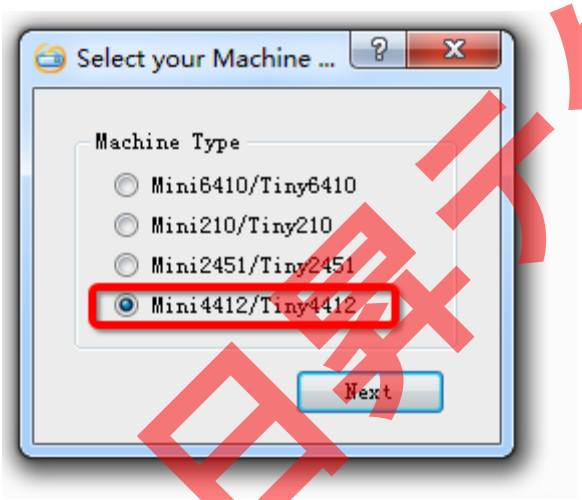
一部の 256M 以下の SD カードは識別できないため、 4G 以上の SD カードを勧む。

Win7 環境のプロセスは下記の通り：

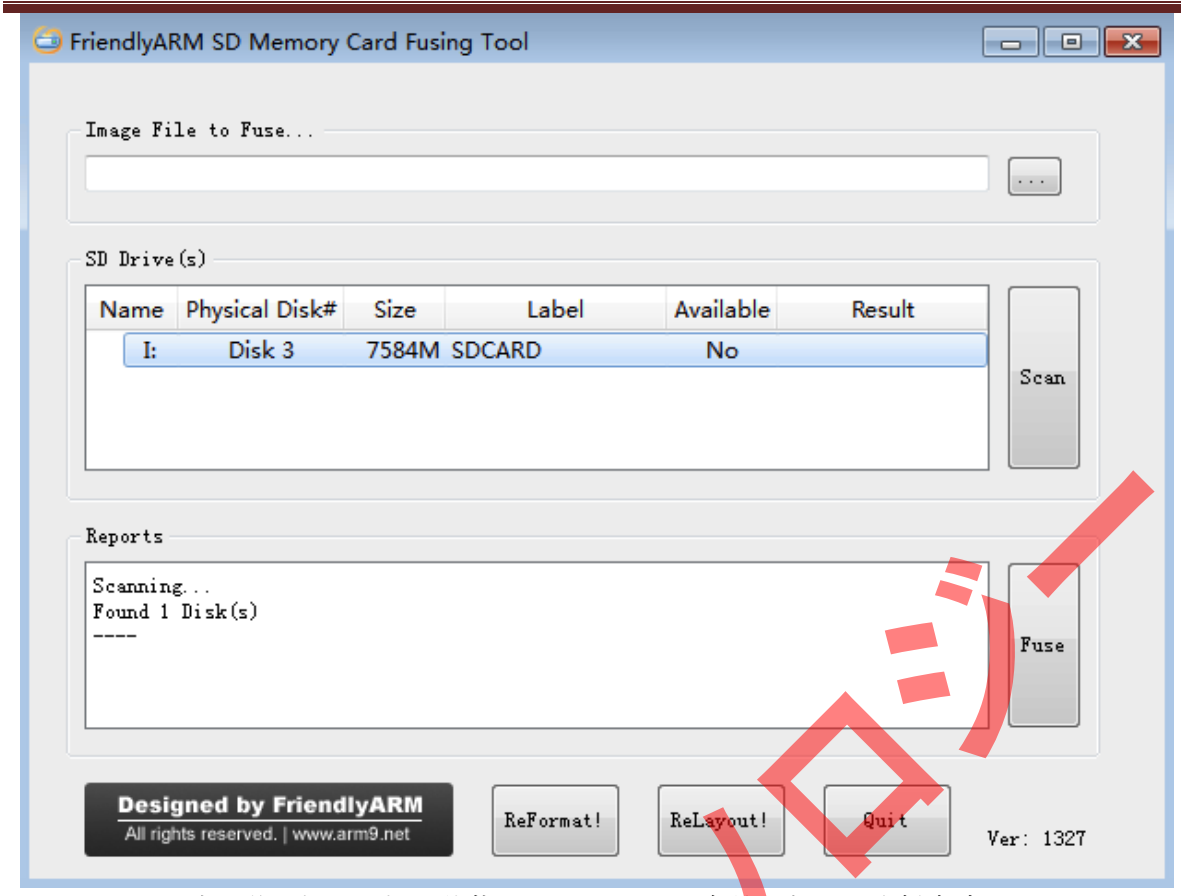
Step1: CD¥tools¥ディレクトリをオープン、 SD-Flasher-1327.7z 解凍し、 SD-Flasher.exe 書込みソフトウェアを起動、admintrator で実行する必要がある。



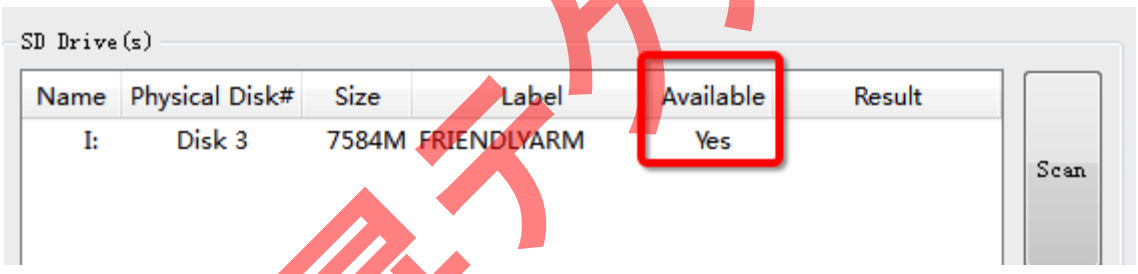
SD-Flasher.exe ソフトウェア起動时、｀Select your Machine…｀で｀Mini4412/Tiny4412｀項を選択：



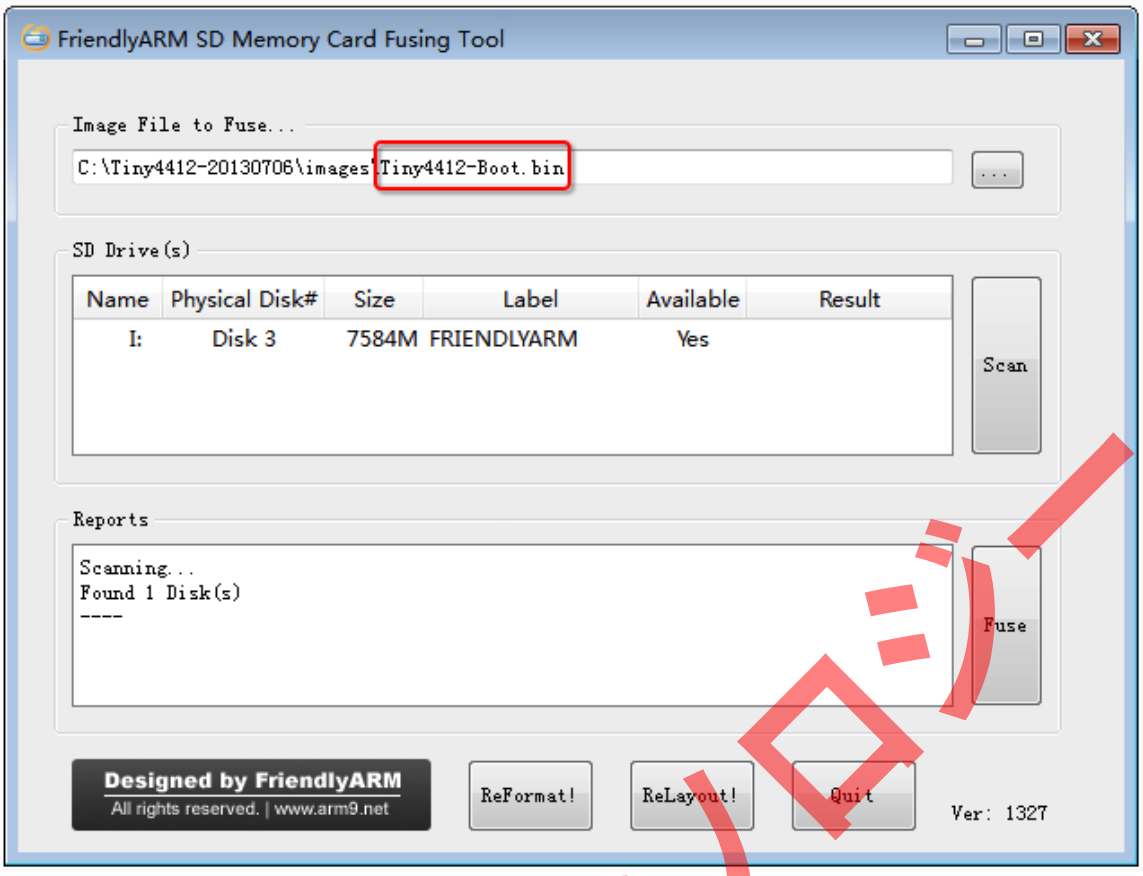
Next、SD-Flasher のメインインタフェースで Scan をクリック、SD カード検索、選定。次は｀ReLayout｀をクリック、SD カードを分割する。＊本動作は SD カードをフォーマットし、/再パーティションする。



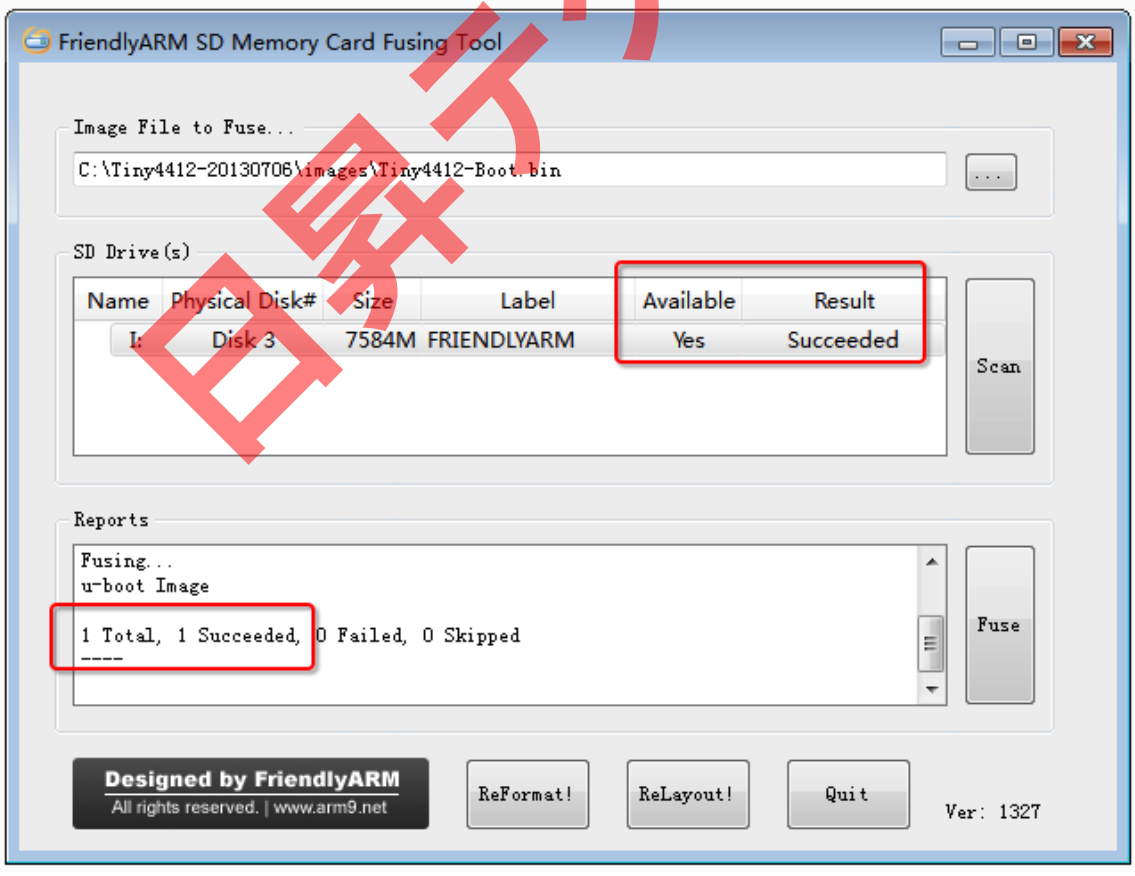
フォーマット完了後、`Scan`、状態で Available は有効となる、分割成功：



Step2:  ボタンをクリック、Superboot4412.bin を選択(デフォルト位置は images/にある)：



Step3:｀Fuse｀クリック、Superboot4412.bin は SD カードのフリーエリアに書込む。以後 SD-Flasher で SD カードを作成時に、 ReLayout を実行する必要はない、FAT32 パーティション中のデータは保存出来る。



Bootloader はSD カードに書き込み後は不可視で、検索方法は：SD カードをシリアル接続した開発ボードに挿し込み、開発ボードの S2 スイッチを｀SDBOOT｀モードに切り替え、通電後、シリアルで情報出力、Superboot4412.bin はSD カードに書き込むと表示する。シリアルに出力が無い場合、書込み失敗と意味する。

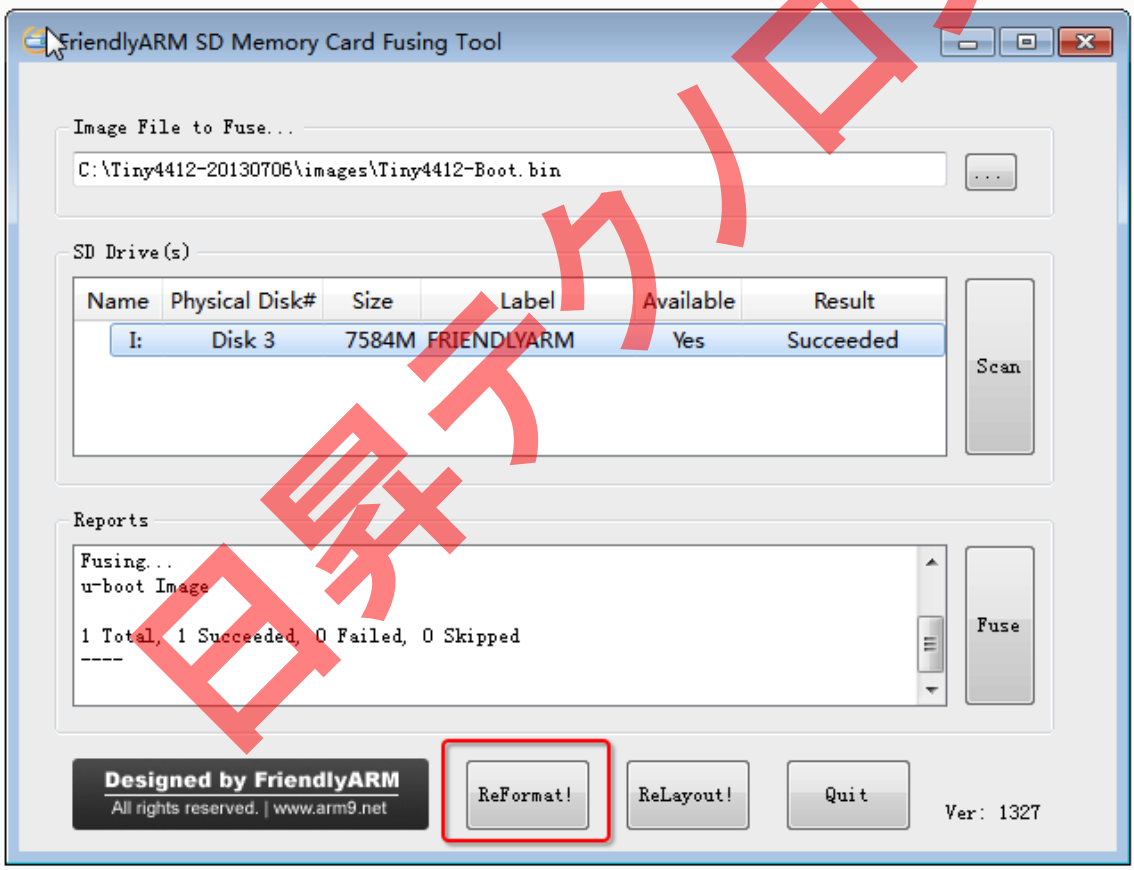
失敗の原因と解決方法は下記の幾つにある：

1. ノートパソコンの内蔵のカードリーダを使用し、識別失敗。外部 usb カードリーダの使用をお勧め。
2. SD カード識別失敗、正規品、4G またその以上の SDHC を使用する
3. microSD カードを使用、接続不良な場合があるので、普通の SD カードの使用をお勧め。
4. 接続不良で失敗する時が多いので、SD カードを抜出して再挿入したり何回試した方が良い。

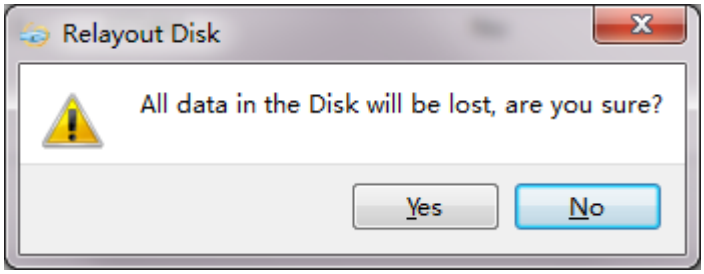
2.2.2 SD カード初期化

Windows7 下で実行する。

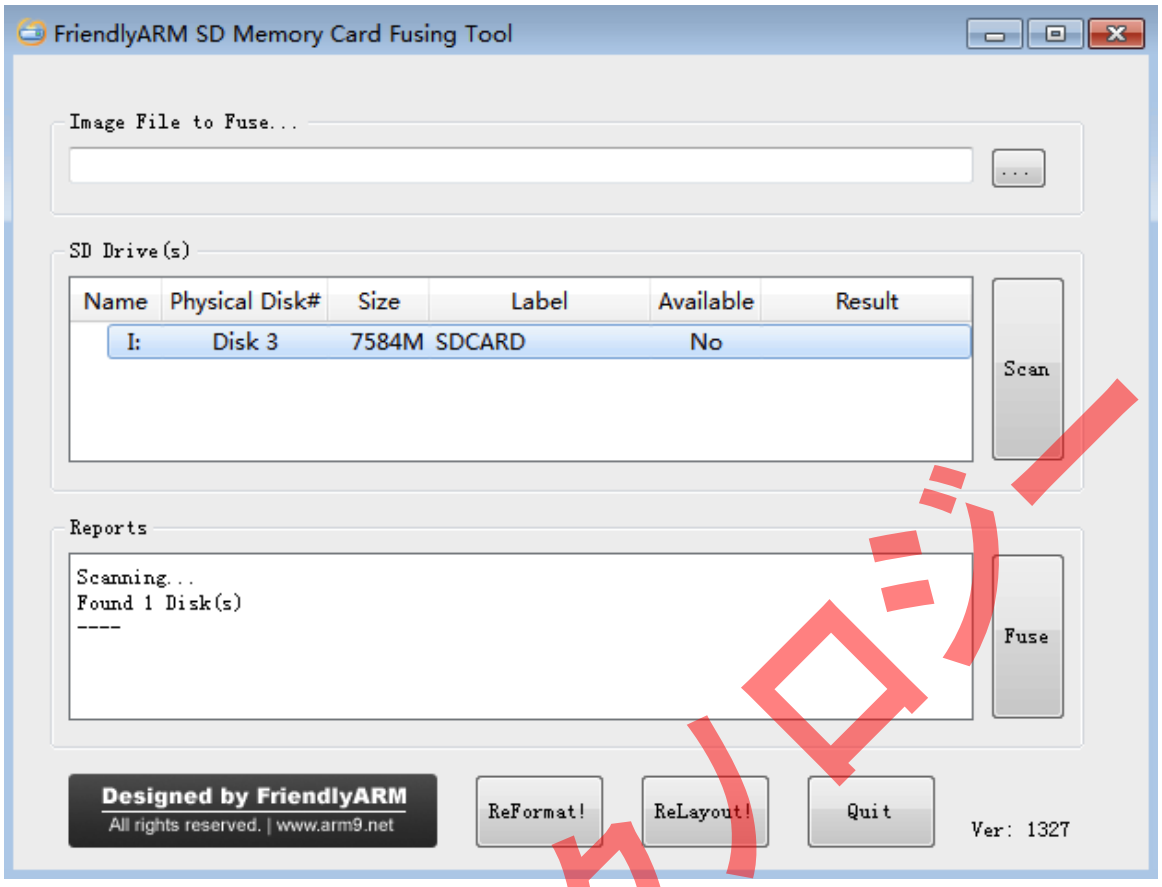
SD-Flasher.exe は予め 130M のスペースを置いて、 Superboot4412.bin の書込みに使用する。従って、SD は開発ボードに使用しない場合、SD カードの容量を回復する時、上記の方法で SD-Flasher.exe を起動し、｀ReFormat!｀機能で SD カードを初期化する：



｀Scan｀でSD カードを検索、｀ReFormat!｀、SD カードデータロストの提示がある：



｀Yes｀、リセット開始、完了後もう一度｀Scan｀をクリック、この時 SD カードは書込み不可の状態となる、即ち初期状態に戻った、下記図の通り：



2.2.3 注意事項

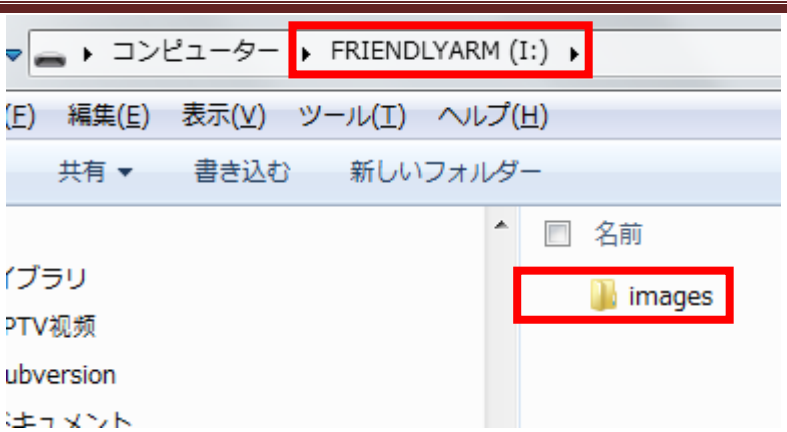
ユーザーは SD カードを使用し、他のデータを保存する場合がある。従って、SD-Flasher ソフトウェアは Vista/Winows7 で実行する時、SD カードを自動のに一般の FAT32 フォーマット（ラベルを自動的に｀FriendlyARM と命名）とノーフォーマットのスペース（130M）の二つの部分に分割する、書込みツールはラベルを判断して、bootloader をノーフォーマットスペースに書き込む。

Vista/Windows7 システムの安全レベルが高いため、直接 Vista/Windows7 システムで強制 SD カードを書き込むのは不可能で、従って、先にスペースを分割する必要がある。

また、Tiny4412-Boot.bin を書込む時、Windows7 で SD-Flasher を実行する。Windows XP の環境では書込みが失敗する場合がある。

2.2.4 images ディレクトリを SD カードにコピーする

SD カード書込み機能を使用する時、付属 DVD の images ディレクトリを SD カードのルートディレクトリにコピーする：



2.3 SD カードでシステム書込み

2.3.1 Android os 書き込み

事前準備： SD カードに Superboot4412.bin を書込み、付属 DVD から対応の書込みファイルを SD カードにコピーされている。images ディレクトリを SD カードのルートディレクトリ下にコピーする。

Step1: SD カードをパソコンに差し込み、` images¥FriendlyARM.ini` ファイルをダブルクリック、FriendlyARM.ini を下記のように変更（デフォルト）：

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)
CheckOneButton=No
Action = Install
OS = Android
LowFormat = No
VerifyNandWrite = No
LCD-Mode = No
CheckCRC32=No
StatusType = Beeper | LED
#################### Android ####################
Android-BootLoader = Superboot4412.bin
Android-Kernel = Android/zImage
Android-CommandLine = console=ttySAC0,115200n8 androidboot.console=ttySAC0
Android-RamDisk =Android/ramdisk-u.img
Android-RootFs-InstallImage = Android/system.img
```

Step2: SD 上に下記のファイルを確認する、ないファイルを DVD から SD カードにコピーする(images ディレクトリ丸めて SD カードのルートディレクトリにコピー)：

ファイル名	説明
images¥Tiny4412-Boot.bin	Bootloader
images¥Android¥zImage	Android カーネル(Linux Kernel 3.5)
images¥Android¥ramdisk-u.img	Android ルートパーティションイメージ

Images¥Android¥system.img	Andorid システムパーティションイメージ
images¥FriendlyARM.ini	システム書込み配置ファイル

Step3: SD カードを開発ボードの SD スロットに 差し込み、 S2 スイッチを SD カード起動に切り替え、電源を入れる、システム書込み開始する時、LCD とシリアルターミナルでスケジュールが表示する。

Step4: システム書込み完成後、開発ボードの S2 スイッチを｀Nand｀起動に切り替え、reboot し、 新しい Android システムが起動する。

2.3.2 Linux システム書き込み

事前準備： SD カードに Superboot4412.bin を書込み、付属 DVD から対応の書込みファイルを SD カードにコピーされている。images ディレクトリを SD カードのルートディレクトリ下にコピーする。

Step1: SD カードをパソコンに差し込み、｀images¥FriendlyARM.ini｀ファイルをダブルクリック、FriendlyARM.ini を下記のように変更（デフォルト）：

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)
CheckOneButton=No
Action = Install
OS = Linux
LowFormat = No
VerifyNandWrite = No
LCD-Mode = No
CheckCRC32=No
StatusType = Beeper | LED
#################### Linux ####################
Linux-BootLoader = Superboot4412.bin
Linux-Kernel = Linux/zImage
Linux-CommandLine = root=/dev/mmcblk0p1 rootfstype=ext4 console=ttySAC0,115200 init=/linuxrc
ctp=2 skipcali=y
Linux-RamDisk = Linux/ramdisk-u.img
Linux-RootFs-InstallImage = Linux/rootfs_qtopia_qt4.img
```

Step2: SD 上に下記のファイルを確認する、ないファイルを DVD から SD カードにコピーする(images ディレクトリ丸めて SD カードのルートディレクトリにコピー):

ファイル名	説明
images¥Tiny4412-Boot.bin	Bootloader。
images¥Android¥zImage	Linux Kernel 3.5)
images¥Android¥ramdisk-u.img	Linux ルートパーティションイメージ
Images¥Android¥system.img	Linux システムパーティションイメージ
images¥FriendlyARM.ini	システム書込み配置ファイル

Step3: SD カードを開発ボードの SD スロットに 差し込み、 S2 スイッチを SD カード起動に切り替え、電源を入れる、システム書込み開始する時、LCD とシリアルターミナルでスケジュールが表示する。

Step4: システム書込み完成後、開発ボードの S2 スイッチを｀Nand｀起動に切り替え、reboot し、 新しい Linux システムが起動する。

2.4 MiniTools でシステム書き込み

MiniTools は新しい USB ダウンロードツールで、携帯を書き込むように開発ボードを書き込む。主な特性は下記の通り：

□**シリアルインタフェースが不要**：MiniToolsは完全にUSBを使用する、シリアルインタフェースが不要。

□**ワンキー書き込み**：MiniToolsはワンキー書き込みを実現させ、書き込みファイルは一つ選択と多数選択ができる。

□**32/64-bitコンピュータサポート**：MiniToolsには32/64-bitに必要なドライブがある、全てのWindowsプラットフォームに通用。

□**クロスプラットフォーム**：MiniToolsはQt4開発を採用し、WindowsとLinux全てサポート。

SDカードでシステム書き込みと違い、Minitoolsを通じて、USBラインでハイスピードにシステムを開発ボードにインストールできる。

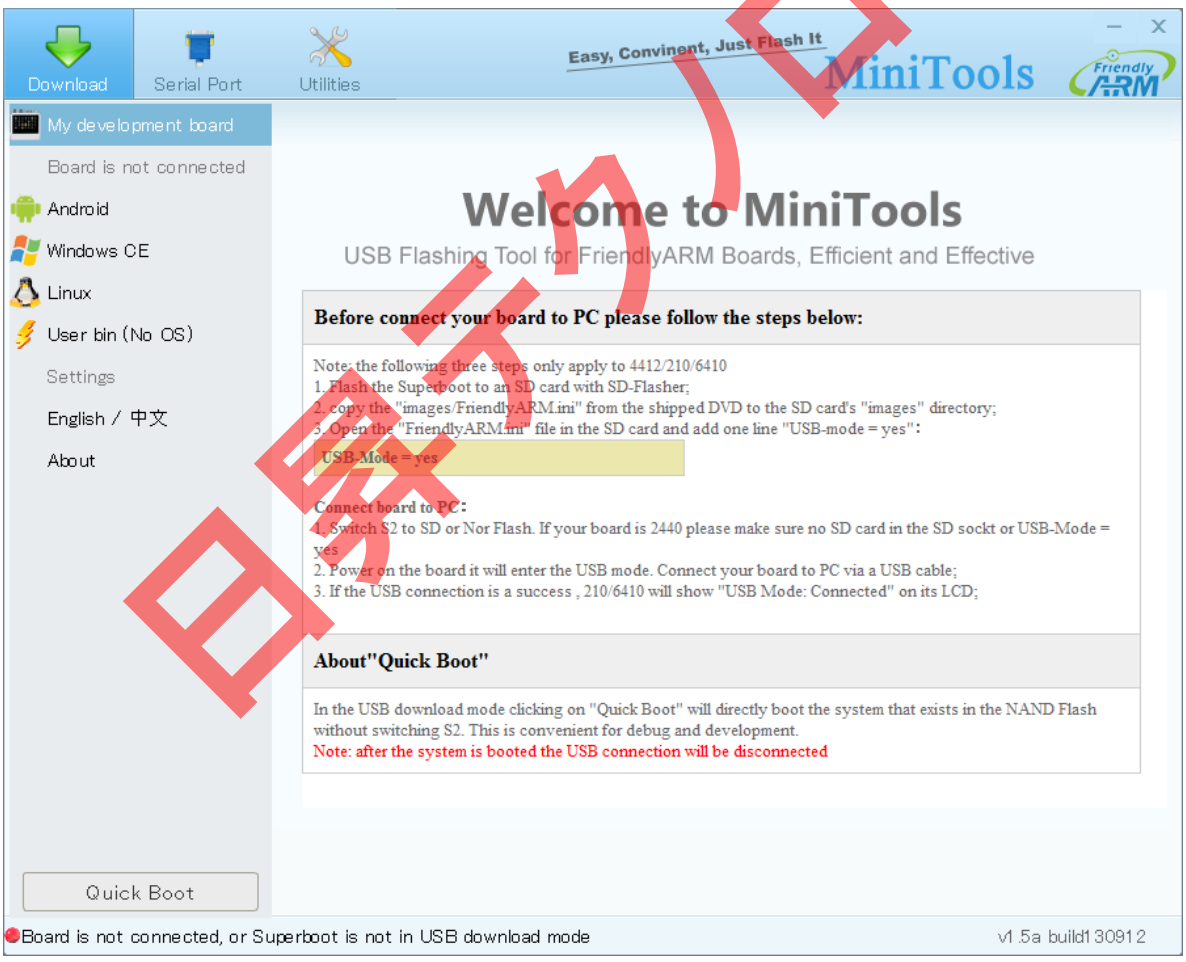
2.4.1 MiniTools インストール

2.4.1.1 Windows システムでのインストール

Tools¥minitools¥ディレクトリの MiniTools-Windows-YYYYMMDD.exe をダブルクリックし、Minitools インストールプログラムが起動する。日本語の環境では一部文字化けがありますが、インストールした後は英語の操作画面になります。必要な USB ダウンロードドライバーが自動的にインストール。ドライバーをインストールするかどうかの提示がポップアップすれば、「インストール」を選択。インストールが完成した後、再び USB ラインを差し込む。Windows のドライバーアップグレードが完成した後、次のステップに入る。

MiniTools インストールしたら、デスクトップに下記のショートカットが出来る。ダブルクリックして MiniTools が起動：

MiniTools の画面は下記の通り：



2.4.1.2 Linux システムでのインストール

Linux システムでは root ユーザーで MiniTools を使用しなければならない。そうでなければ、USB が開発ボードに接続出来ない恐れがある。日常は root ユーザーでおすすめ。

Linux システムで MiniTools のインストールは比較的に簡単である。付属 DVD の tools ディレクトリの MiniTools-Linux-YYYYMMDD.tgz を展開するだけ。MiniTools を起動する時、Minitools ディレクトリに入り、コマンド./start.sh を執行する。Root ユーザーではない場合、root ユーザーに切り替える必要がある。

2.4.2 USB でシステム書き込みの事前準備

1．SD-Flasher で Superboot を SD カードに書き込み、方法は 2.2.1 に参考する。
2. images/FriendlyARM.ini ファイルを SD カードの images にコピーする。
3. SD カードの images/FriendlyARM.ini ファイルに下記の内容を加える：

```
USB-Mode = yes
```

事前準備ができたら、下記の手順で PC と開発ボードを接続する：

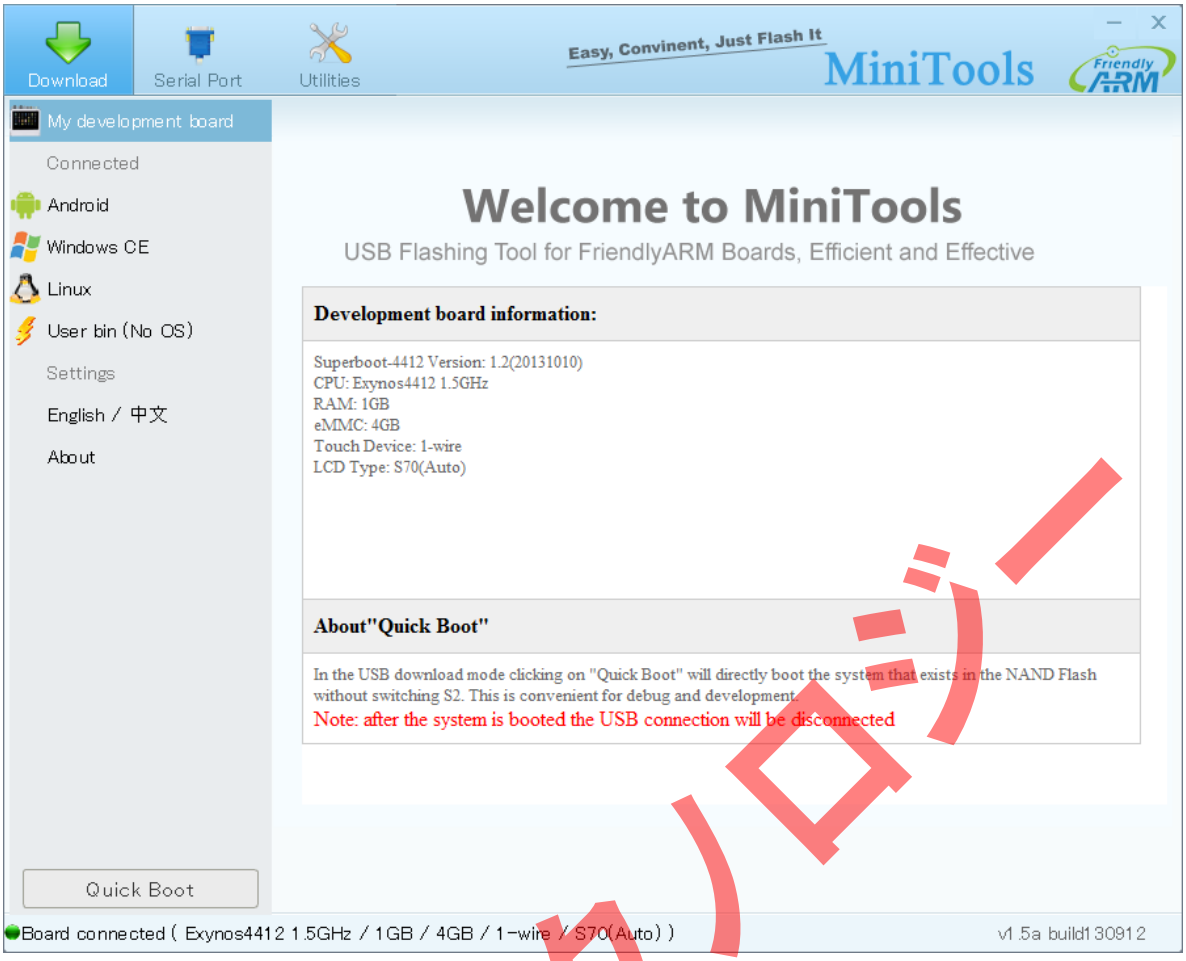
1． S2 スイッチを SD カードモードに切り替える。
2． 電源を入れ、開発ボードは USB ダウンロードモードに入り、LCD に「USB Mode:Waiting」が表示する。
3． USB ラインで PC と開発ボードを接続する。
4． 接続したら、LCD に「USB Mode: Connected」が表示する。

ここまで、Minitools でシステム書き込みができる。

どのように SD カードでシステム書き込みに戻る？
SD カードの images/FriendlyARM.ini ファイルを修正し、USB-Mode = yes を USB-Mode = no に変える。

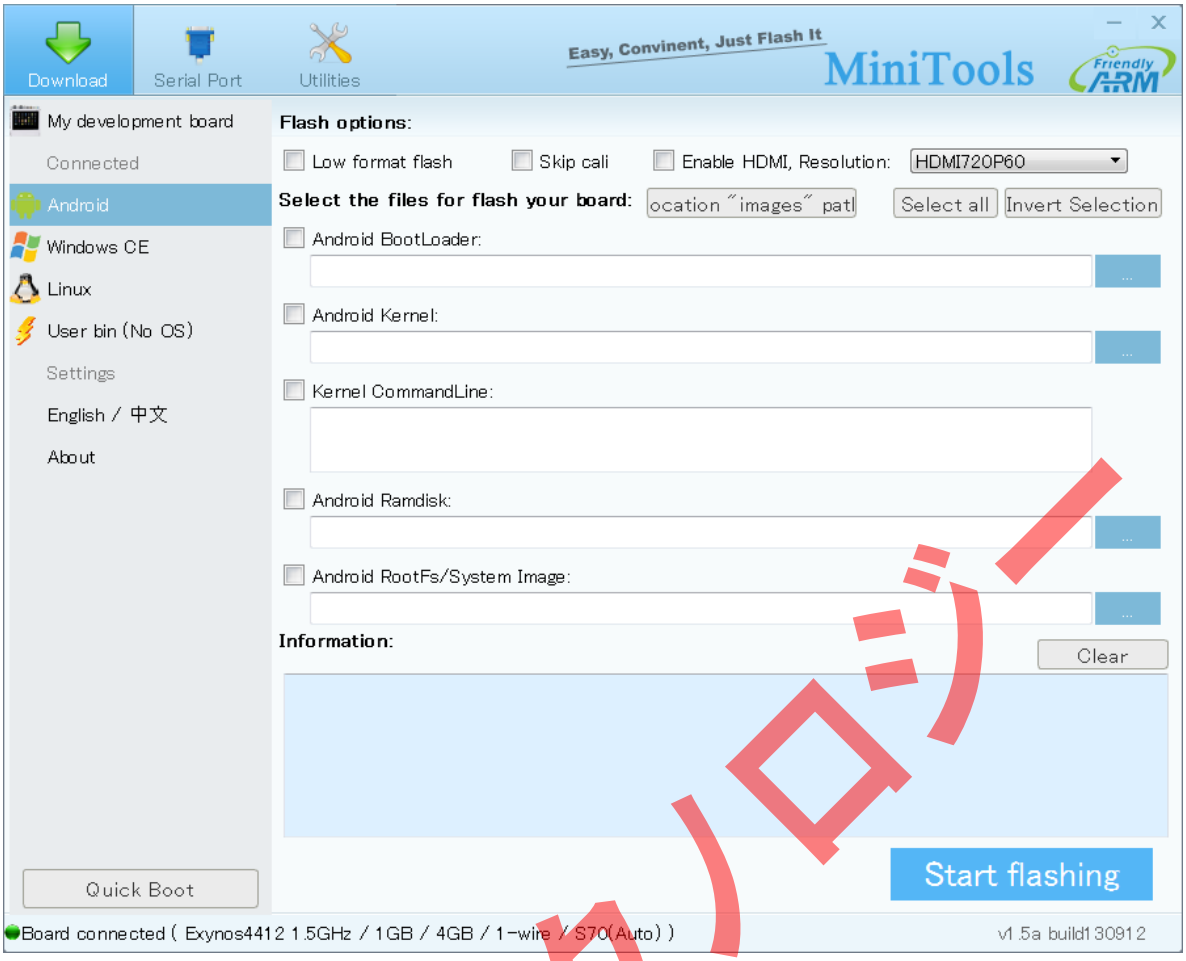
2.4.3 Minitools でシステム書き込み

Superboot を USB ダウンロードモードにし、USB ラインで PC と開発ボードを接続すると、Minitools を起動する時、画面は下記の通り：



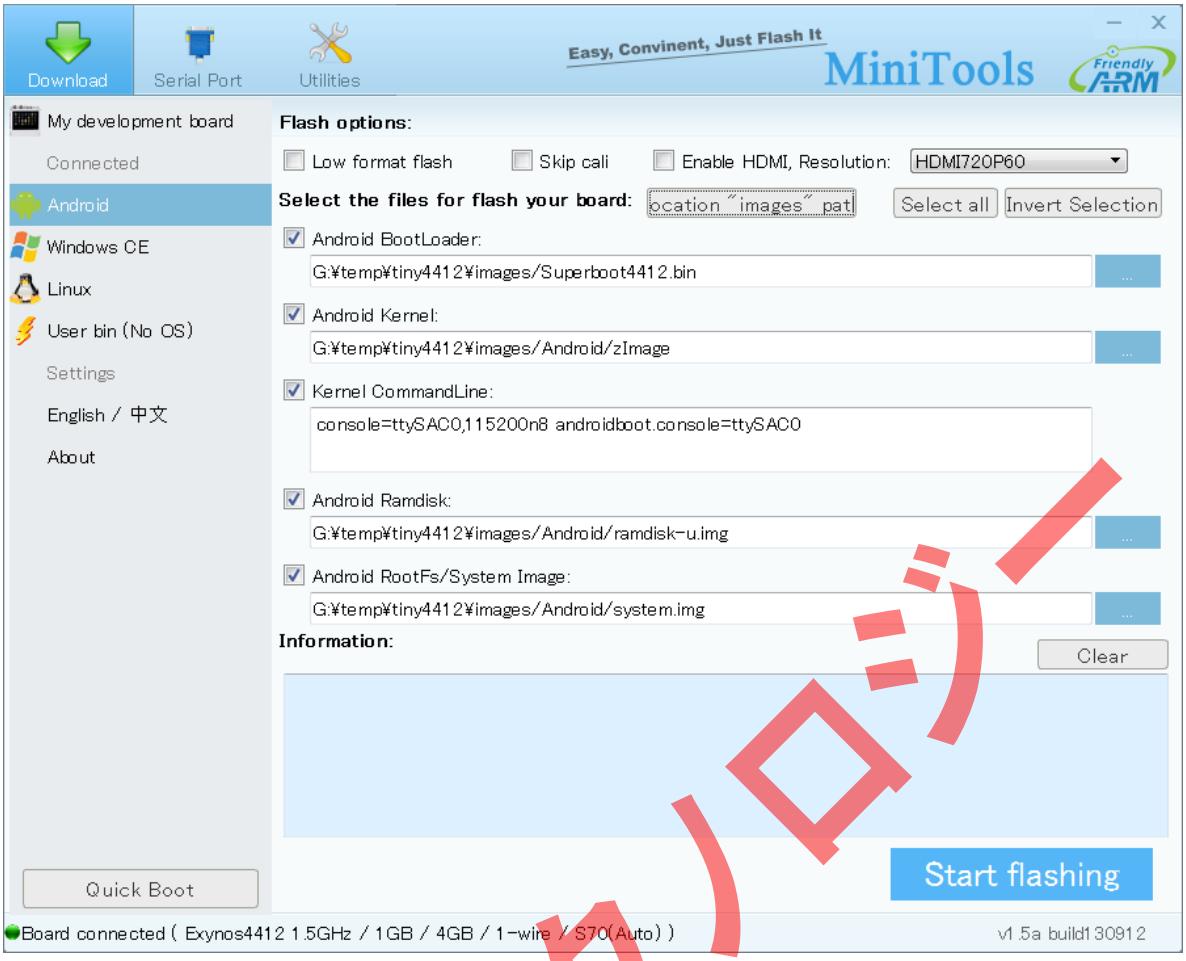
左下に「Quick Boot」ボタンがあり、USB ダウンモード上直接 NAND flash のシステムを起動できる。NAND flash 起動モードに切り替える必要がない。

システムを書き込むには左側に書き込むシステムを選び、例えば Android、該当システムのコンフィグ画面が出る。画面は下記の通り：

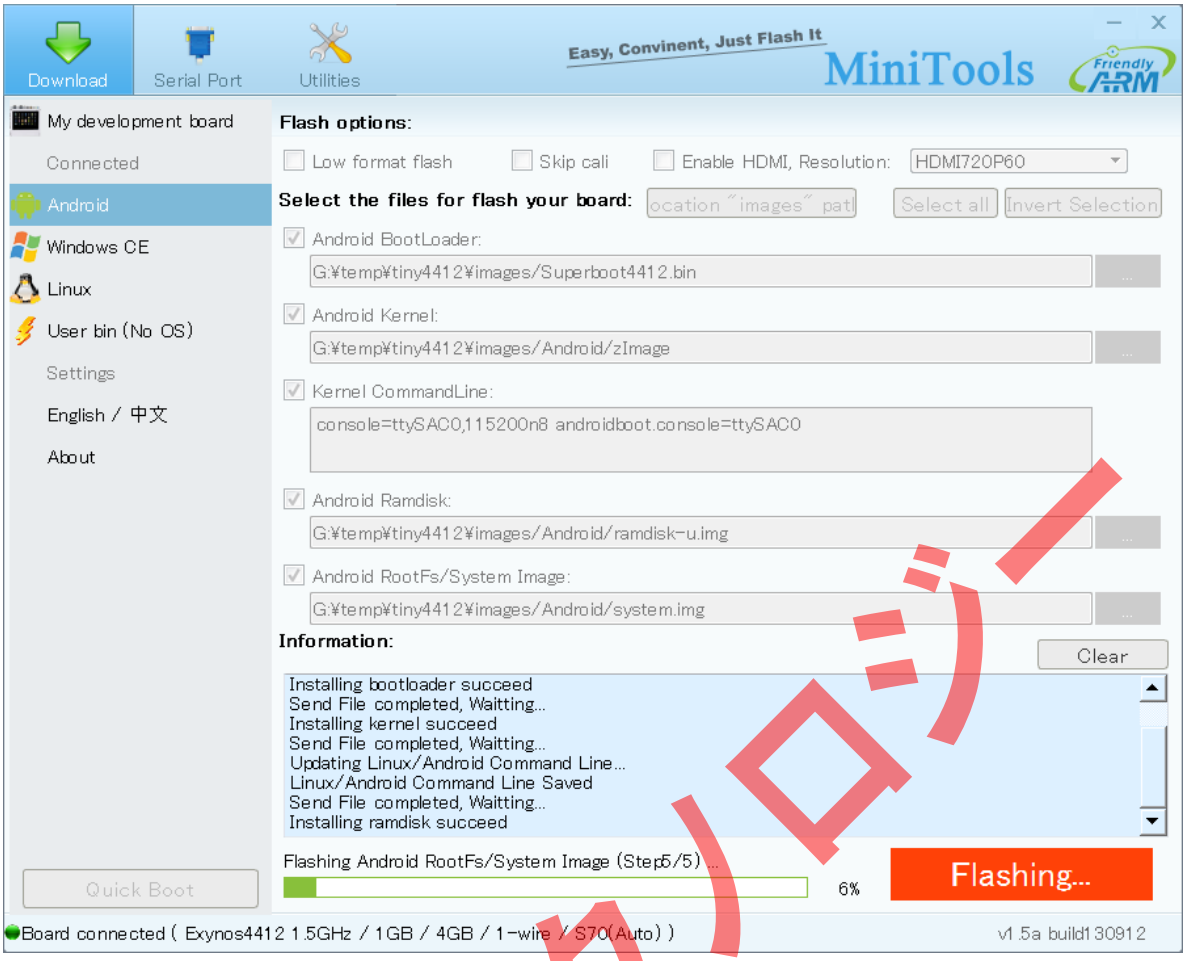


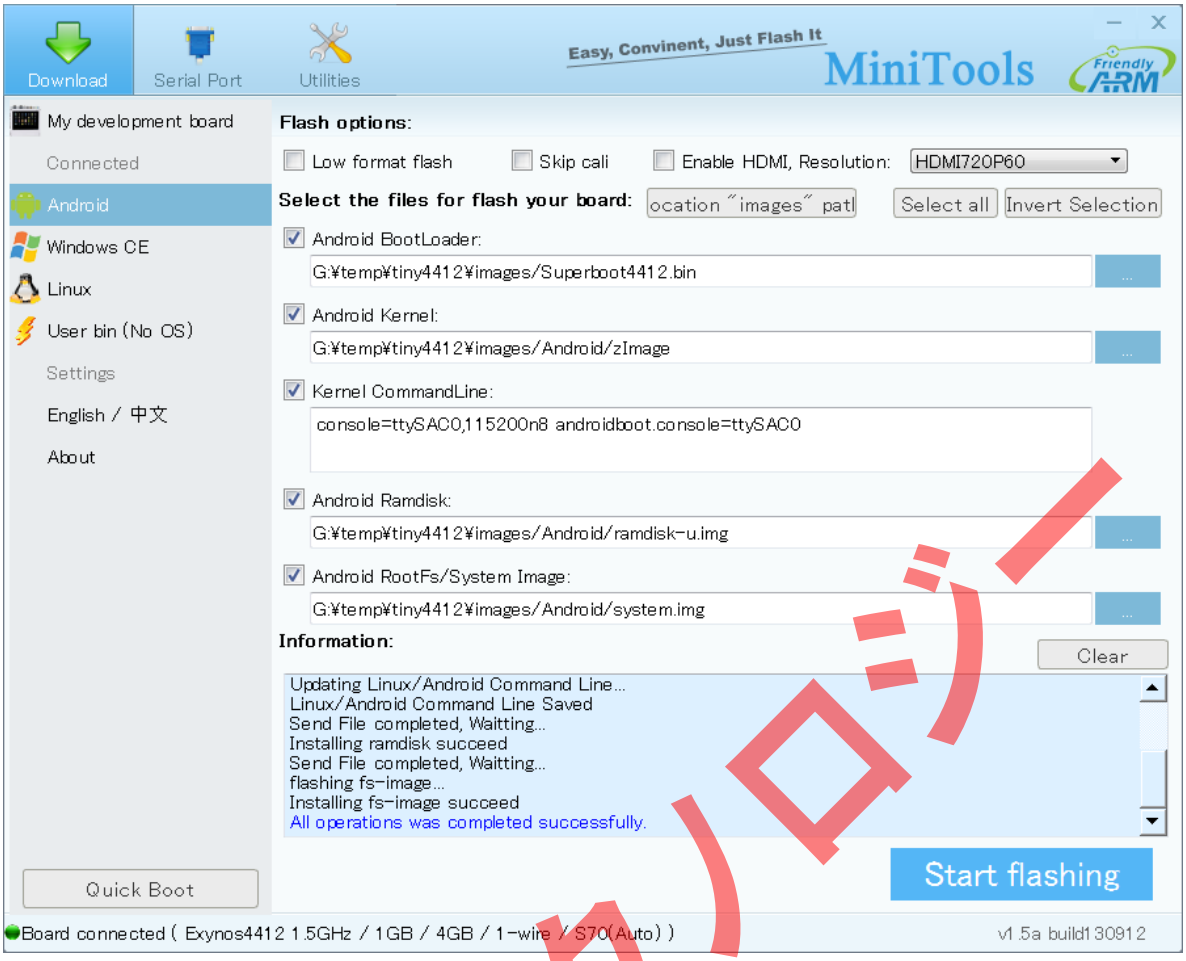
書き込みコンフィグはほとんど FriendlyARM.ini と同じ、FriendlyARM.ini を参考し設定項目を記入できる。更に便利な方法として、「location “images” pat」ボタンをクリックして DVD の images フォルダを指定すると、Minitools が自動的に FriendlyARM.ini の設定内容を画面に記入する。

もう一つの方法は images フォルダを Minitools のインストールディレクトリにコピーすることである。Minitools が起動する度に、自動的に FriendlyARM.ini の内容をロードする。ロード後の画面は下記の通り：



MiniTools を使用し、システム全体の書き込みと一部分の書き込みができる。例えばカーネルだけ或いはファイルシステムだけ書き込む。設定が完成したら、「Start flashing」ボタンをクリック、書き込みの画面は下記の通り：


Download
Serial Port
Utilities
Easy, Convinent, Just Flash It
MiniTools
Friendly ARM
My development board
Connected
Android
Windows CE
Linux
User bin (No OS)
Settings
English / 中文
About
Quick Boot
Flash options:
Low format flash
Skip cali
Enable HDMI, Resolution:
HDMI720P60
Select the files for flash your board:
ocation "images" patl
Select all
Invert Selection
Android BootLoader:
G:¥temp¥tiny4412¥images/Superboot4412.bin
Android Kernel:
G:¥temp¥tiny4412¥images/Android/zImage
Kernel CommandLine:
console=ttySAC0,115200n8 androidboot.console=ttySAC0
Android Ramdisk:
G:¥temp¥tiny4412¥images/Android/ramdisk-u.img
Android RootFs/System Image:
G:¥temp¥tiny4412¥images/Android/system.img
Information:
Clear
Installing bootloader succeed
Send File completed, Waitting...
Installing kernel succeed
Send File completed, Waitting...
Updating Linux/Android Command Line...
Linux/Android Command Line Saved
Send File completed, Waitting...
Installing ramdisk succeed
Flashing Android RootFs/System Image (Step5/5) ...
6%
Flashing...
Board connected (Exynos4412 1.5GHz / 1GB / 4GB / 1-wire / S70(Auto))
v1.5a build130912



書き込み完成後、「Quick Boot」ボタンをクリック、直接に NAND Flash でシステムを起動できる。

開発ボードと接続できない場合：

開発ボードの LCD に USB Mode：Connected が表示し、MiniTools に開発ボードと接続していないと提示する場合、USB ダウンロードドライバーがインストールが失敗した可能性がある。手動で USB ドライバーをインストールすることで解決できる。USB ドライバーは MiniTools インストールディレクトリにある：

第三章 Android 開発マニュアル

Tiny4412 は Android4.2.2 をソフトウェアプラットフォームと使用し（Linux カーネルバージョン 3.5)、そして 6410 と 210 プラットフォームでの Android 向き開発したソフトウェアを Tiny4412 に移植している。

3.1 Android 体験

3.1.1 ボタン

本開発ボードには 4 つのユーザーボタンがある、android システム中の定義は下記の通り：

ボタンナンバー	機能定義
K1	Back（戻る）
K2	Home （ホームに戻る、長押しは、デーモンが表示）
K3	Menu（メニュー、長押しは、スクリーンを回転）
K4	OK（確定ボタン）

（注：Android は設定変換により、ボタンを再定義できる）

3.1.2 Android 汎用コマンド

3.1.2.1 Android システムコマンドラインに入り、root 権限所得

Android システムのコマンドラインに入る。

- USB 接続式でコマンドラインに入る場合、パソコンで Android SDK をインストール/実行、完成後、 adb shell を入力 Android システムコマンドラインに入る。コマンドラインでは root 権限があり、全てのコマンドを実行出来る。
- シリアルターミナルで Andorid コマンドラインに入る場合、 Android SDK をインストール必要はないが、デフォルト状態では root 権限がない。su コマンドを入力し、 root 権限取得する。

3.1.2.2 system パーティション読み取り/書き込み

Android システムはデフォルト設定では、system パーティションは読取専用 mount 、従ってデータの書き込みはできない、ボードで下記のコマンドを入力により、書き込みを実現する：

```
# mount -o remount /dev/block/mmcblk0p2 /system
```

本コマンドはリセット後、無効となる。読み取り専用に戻る。

system パーティションを常に書き込み可能にしようとすると、 Android リソースコードのファイルを編集する：

device/friendly-arm/tiny4412/fstab.tiny4412

ファイルの内容：

```
/dev/block/mmcblk0p2     /system ext4     ro   wait
```

を下記のように変更する。

```
/dev/block/mmcblk0p2    /system ext4    rw  wait
```

3.1.2.3 PC からファイルを開発ボードにアップロード

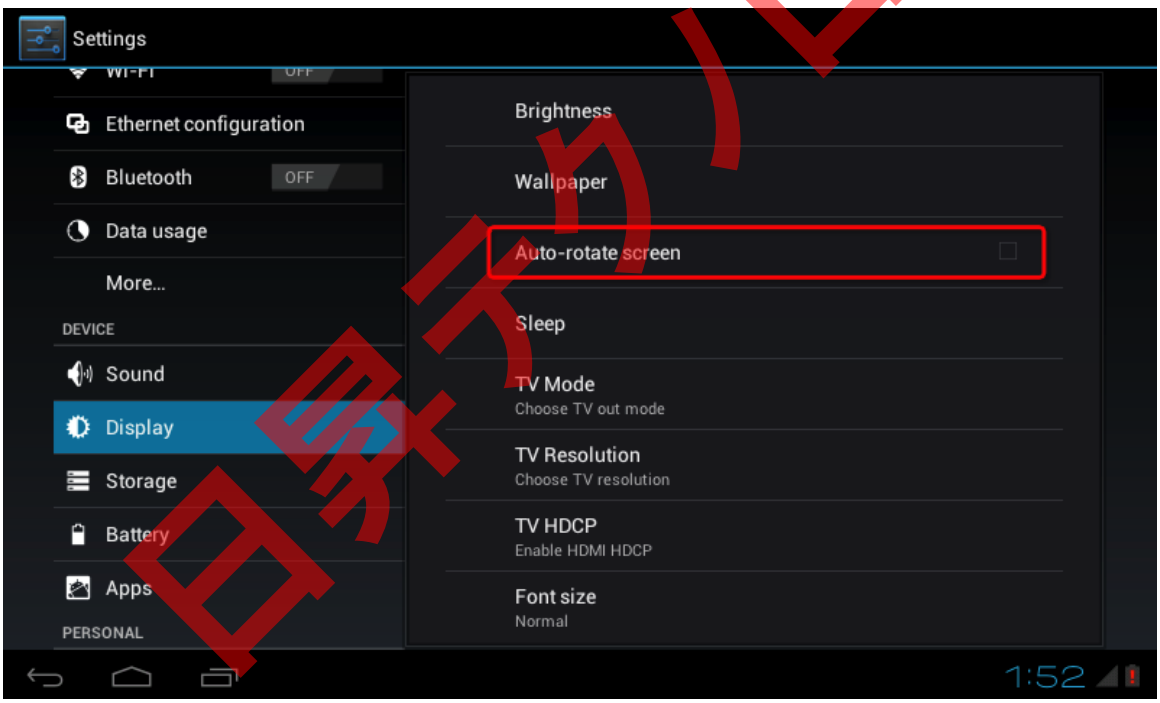
Android SDK 中の adb コマンドでファイルを USB で開発ボードにアップロードする。例えば、sensors.tiny4412.so を開発ボードの/system/lib/hw/ディレクトリ下にコピーする、下記のコマンドを実行する：

```
adb push sensors.tiny4412.so /system/lib/hw/
```

注：ファイルを system ディレクトリにアップロードするには、前節の system パーティションを読み取り/書き込みに設定する必要がある。

3.1.3 重力センシングモジュールで画面自動回転

Tiny4412SDK には重力センシングモジュールがあり、デフォルトで重力センシングで画面を自動回転出来る。Android の設定->Display->Auto-rotate screen で禁止・起動出来る。



3.1.4 プログラムでの SD カードの読み書き

APP は SD カードに格納される Android/パケット名ディレクトリーにアクセスできる。SD カードのディレクトリは: /storage/sd_external/。APP のパケット名は com.FriendlyARM.sdcarddemo である場合、操作権限なしに以下のパスを介し、SD カードの読み書きができる。

/storage/sd_external/Android/com.FriendlyARM.sdcarddemo

初回でプログラムを起動する場合、SD カードの関連ディレクトリーは存在しないため、下記のコードでビ

ルドできる。

```
File file = new File("/storage/sd_external", "Android/com.FriendlyARM.sdcarddemo");
if (!file.exists()) {
file.mkdirs();
}
```

注意：AndroidManifest.xml ファイルに以下の権限を追記する必要がある。

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_MEDIA_STORAGE" />
```

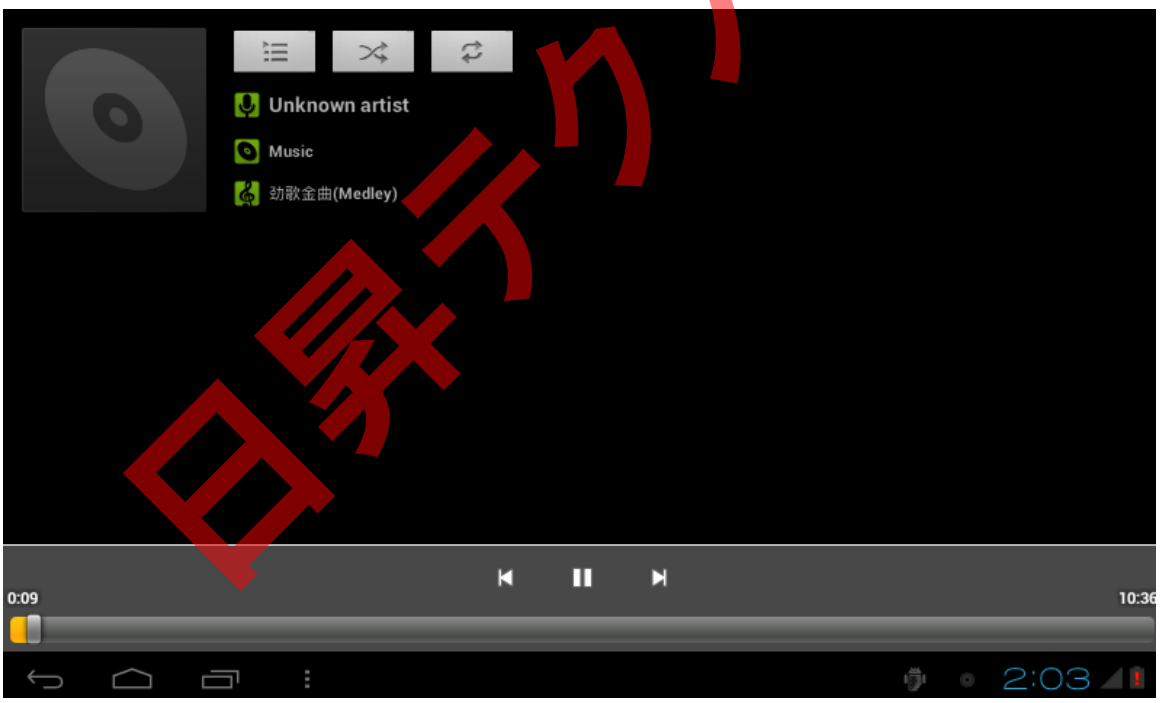
SD カードの読み書きサンプルを提供する。ソースコードは Android4.1.2 のソースコードディレクトリー：device/friendly-arm/tiny4412/SDCardDemo に格納されている。

3.1.5 UI の Root 権限の取得

Root 権限の昇格のため、システムに su と Superuser.apk を統合した。

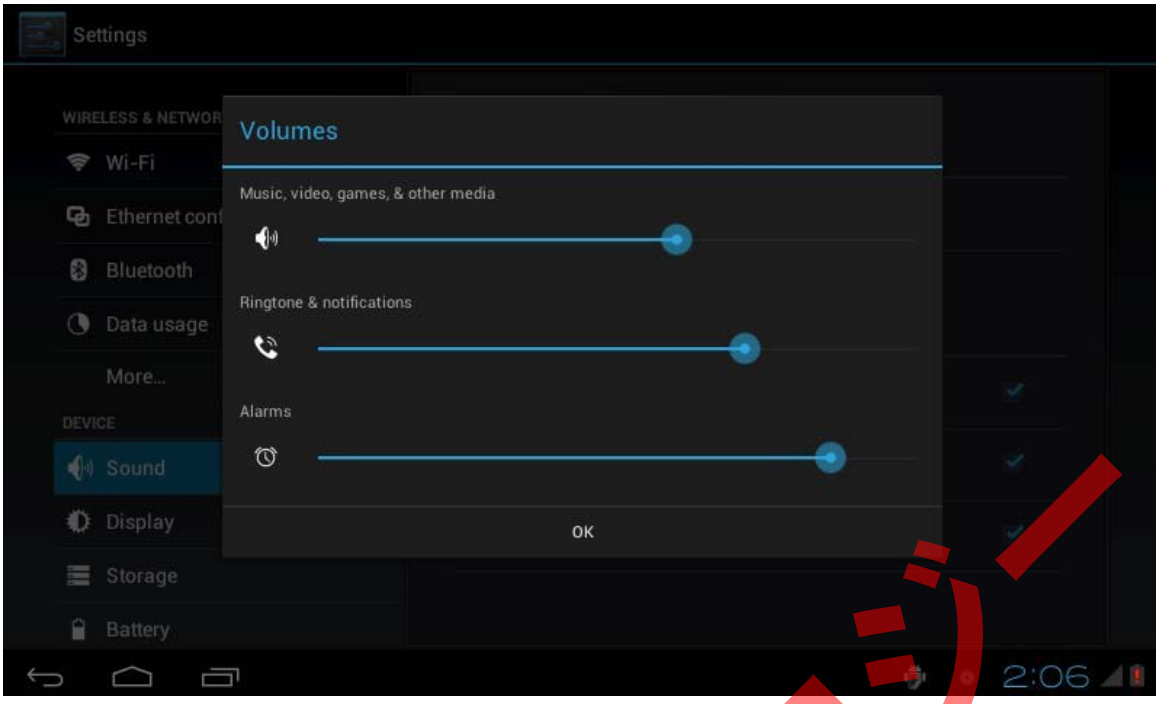
3.1.6 mp3 再生

Android システムは自動的に SD カード中の mp3 ファイルを検索し、下記は mp3 の再生画面である。



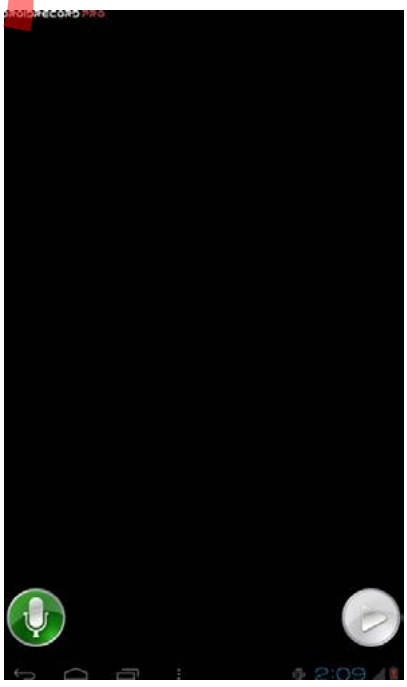
3.1.7 ボリューム調整

Setting -> Sound でボリュームを調整出来る：



3.1.8 録音機能

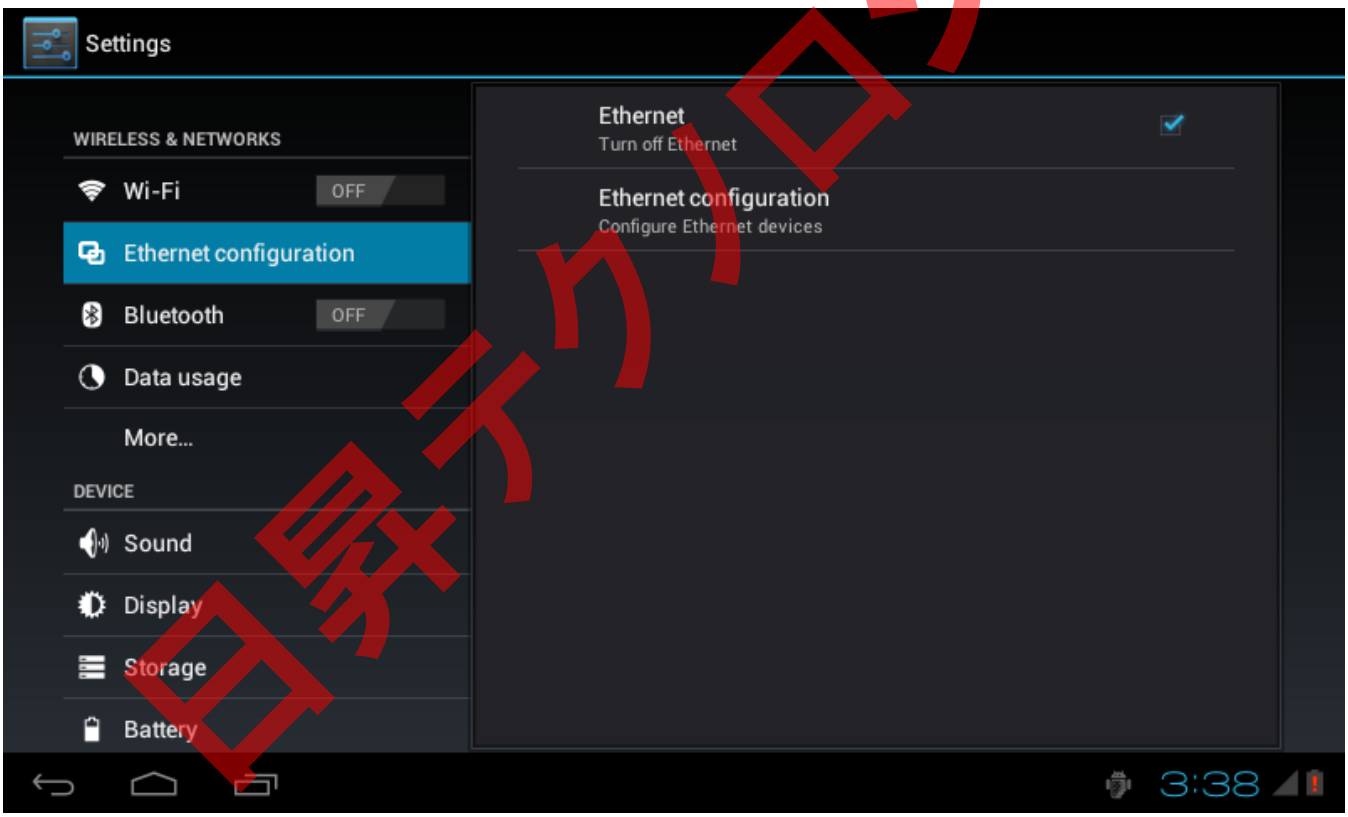
Android に DroidRecord 録音ソフトウェアを統合。録音と再生に使用出来る。プログラムアイコンは下記の通り（右は起動画面）：



録音プロセスは下記の通り：

3.1.9 GUI での有線ネットワークコンフィグ

Android4 の有線ネットワークコンフィグをシステムのネットワークコンフィグに統合した。環境設定のクリックでイーサネットのコンフィグオプションに遷移する。



[Ethernet configuration]をクリックし、以下の操作ができる。

Ethernet:イーサネットの入/切に使われる。イーサネットを改めて接続する場合、切断して再起動するといい。

Ethernetconfigureation:DHCP、固定 IP のコンフィグができる。

イーサネット接続成功後、ステータスバーに ethernet のアイコンが表示される。

3.1.10 イーサネット MAC アドレスの変更

イーサネットカードはメモリ機能を持っていないため、MAC アドレスはドライバによるコンフィグが必要

となる。開発ボード出荷時、MAC アドレスは同じであるため、同一の LAN で複数枚の開発ボードを利用する場合、ネットワーク通信をスムーズに行えるように、MAC アドレスを設定しなければいけない。

MAC アドレスの設定方法は 2 つある。

3.1.10.1　システム書き込み時、FriendlyARM.ini で Mac アドレスの指定

Linux-CommandLine あるいは Android-CommandLine に ethmac パラメータを追記することで、Mac アドレスの設定はできる。

Android-CommandLine = console=ttySAC0,115200n8 androidboot.console=ttySAC0 ethmac=08:90:00:A0:02:99

システムを書き込んだ上、起動する。開発ボードは MAC アドレス：08：90：00：A0：02：99 を利用する。

MAC アドレスに固定書式がある。一般的に後ろの 3 桁を変更する。

3.1.10.2　MiniTools によるシステムパラメータの変更での MAC アドレスの指定

一台のボードの場合、上記の方法による MAC アドレスの変更は問題ないが、多量のボードの場合、効率を考慮し、MiniTools ツールで MAC アドレスを変更したほうがいい。MiniTools のダウンロードアドレス：

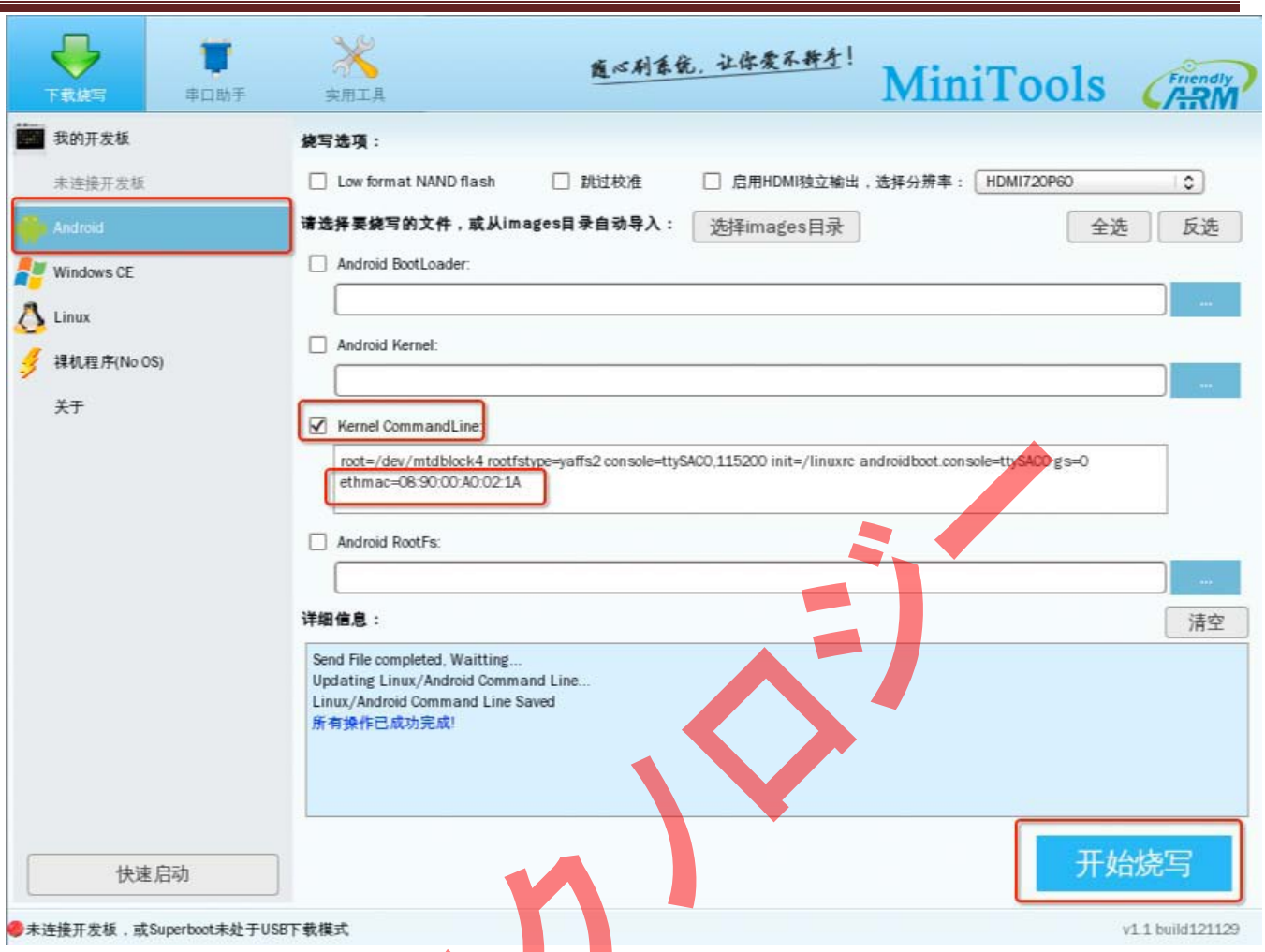
http://www.arm9home.net/read.php?tid-24600.html

MiniTools 起動後、利用するファイルシステム（Android とか）を選択する。UI で”KernelCommandLine”だけにチェックを入れ、以下の KernelCommandLine パラメータを入力する。

console=ttySAC0,115200n8 androidboot.console=ttySAC0 ethmac=08:90:00:A0:02:99

Command Line パラメータの ethmac で Mac アドレスを指定する。

MiniTools の設定効果：



［开始烧写］のクリックで設定を完了する。この操作でシステムを改めて書き込むのではなく、パラメータデータを改めて設定する。

3.1.11 ADB 利用説明

3.1.11.1 USB による ADB の利用

microUSB インターフェースが搭載されている。Android 携帯のように、USB を介して ADB を利用できる。

ADB に接続不可の場合、FriendlyARM.ini の中の Android-CommandLine 設定を確認する。Uhost0=y の設定項目があるなら、削除あるいは uhost0=n に変更する。

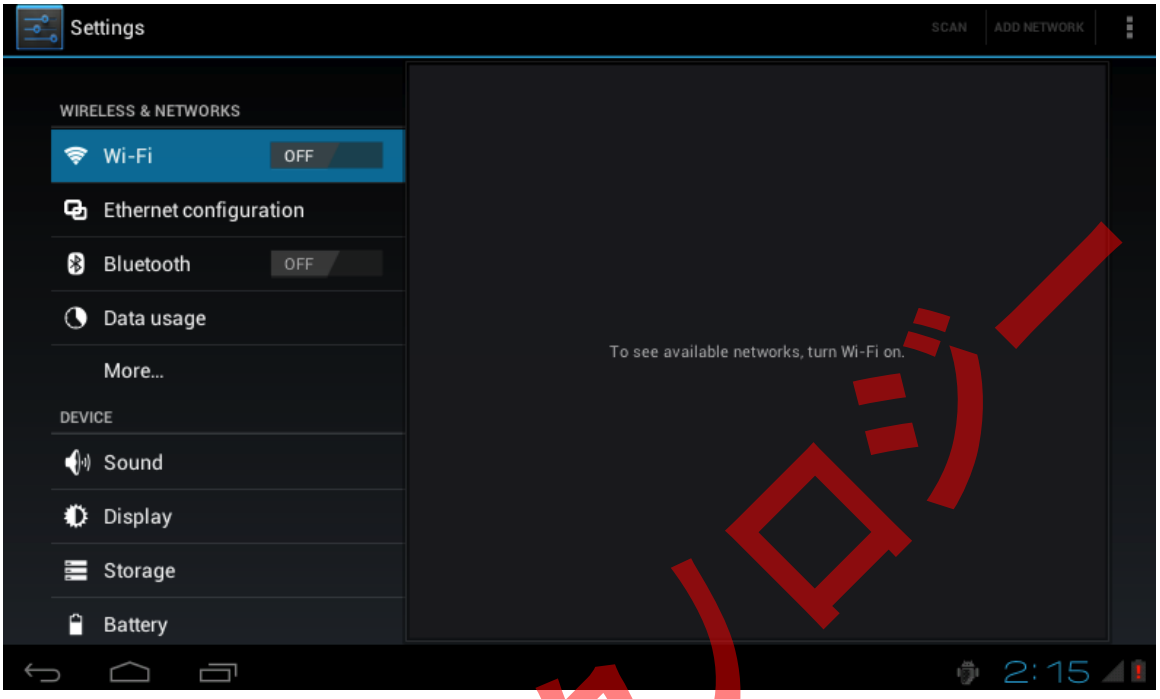
3.1.12 /data/app への APP のプリーロード

プリロードする APP の apk ファイルを Android ソースコードディレクトリーの vendor/friendly-arm/exynos4412/rootdir/system/app/ ディレクトリーに格納する。Android ソースコードのコンパイルが完成後、gen-ing.sh を実行することで、APP を system.img にプレゼンテーションパックする。これをボードに書き込んだら、APP のプリーロードを実現できる。

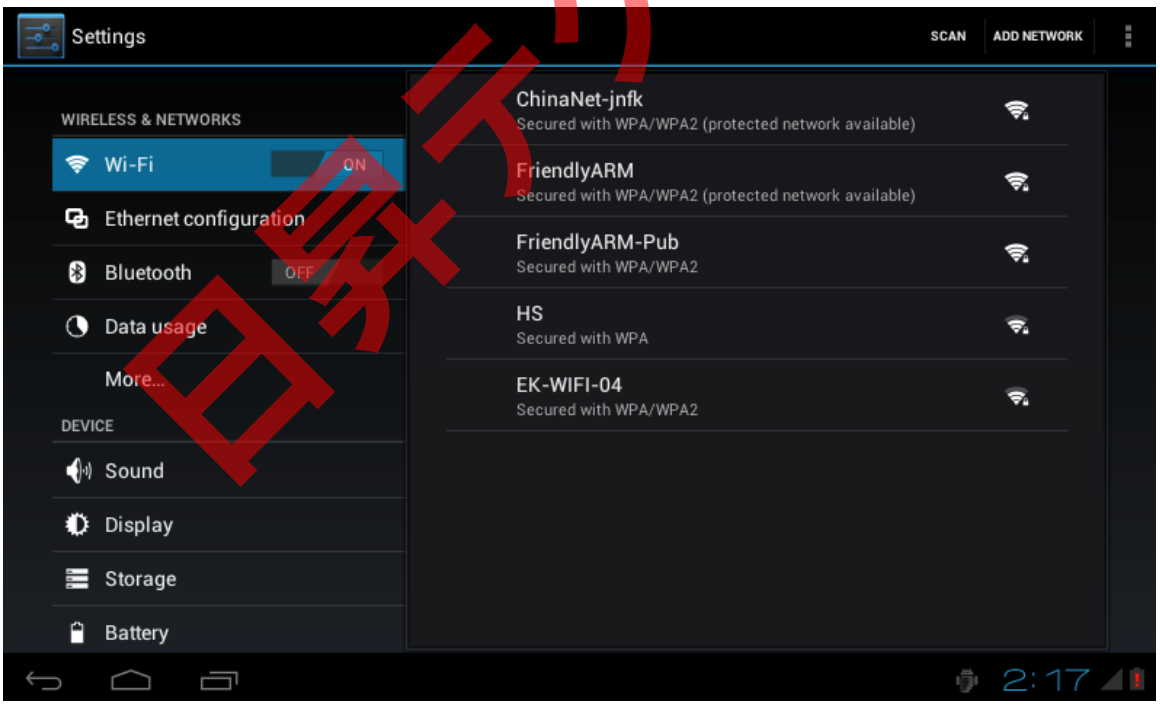
3.1.13　WiFi ネットワークへの接続

本開発ボードは外部 USB WiFi モジュールを接続して無線 WiFi 通信出来る。AndroidOS の WiFi 接続方法は下記の通り：

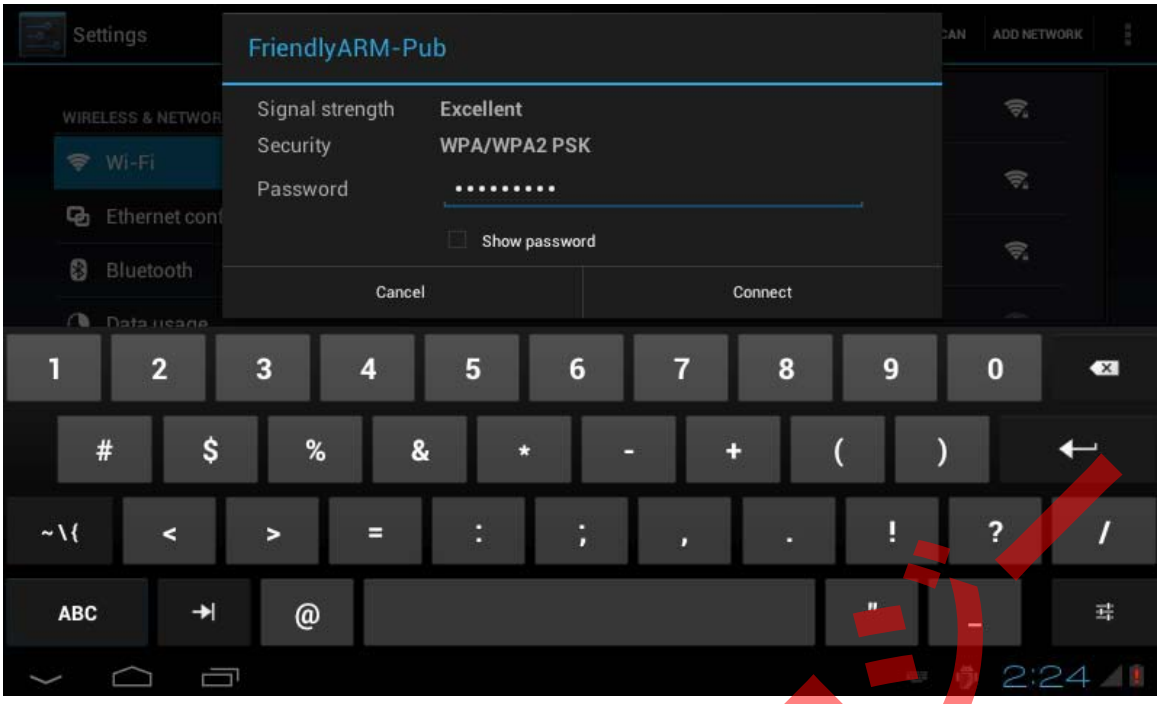
Settings ->Wi-Fi



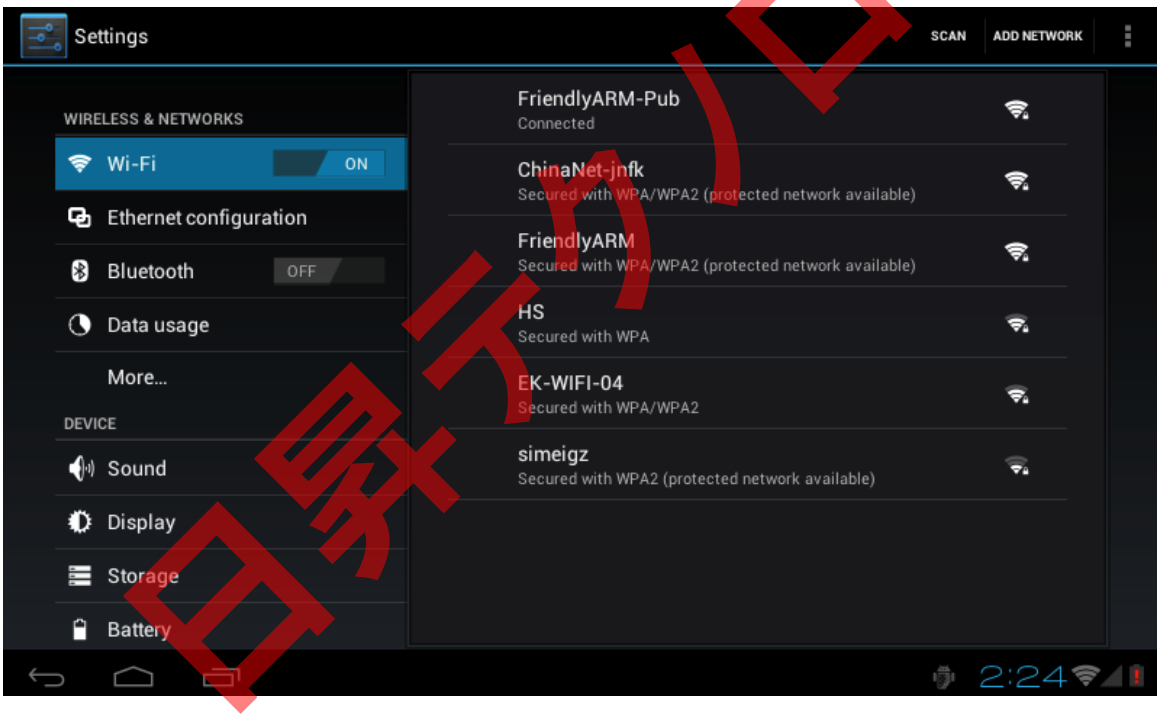
Wi-Fi をクリック、ON、周辺の無線アクセスポイントを自動検索する：



無線アクセスポイントを選択、パスワード入力：



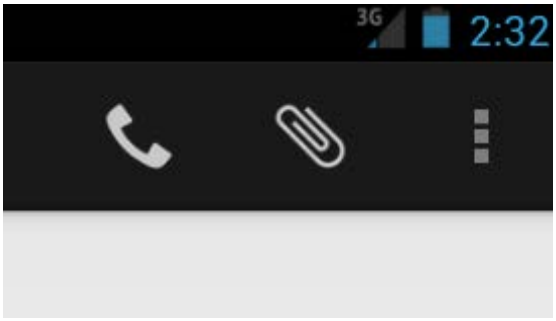
アクセス成功、下記図の通り：



3.1.14 3G 通信及びメッセージ送受信

Tiny4412 は中興 MF210 モジュール（WCDMA）の 3G をサポートし、RIL ドライバーを使用するため、性能が安定している。

Android4 os で MF210 モジュールを差し込んで起動すれば、3G は自動的に接続でき、下図の通り右上に 3G の標示が表示する：



News&Weather アプリを起動し、オンラインでニュースを読むことができる。

メッセージアプリを起動し、メッセージの発信と着信が出来る。

市販の MF210 モジュールも何種類ありますが、テストを行った設備 ID は主に下記の二つである。他の中興 WCDMA モジュールをサポートする可能性もあり、テストが必要となっている。

VID：19d2PID:0117

VID：19d2PID:2003

内部基板にはSIMカードスロット、miniPCIeインタフェースとアンテナがあり、下図の通り：

3.1.15 テレビに HDMI 画像を出力

HDMI ケーブルで本開発ボードとテレビを接続、開発ボードの画面/音声がテレビに同期出力する。

デフォルト解像度は 1080P で、変えたい場合、MiniTools の書き込み UI で解像度の設定をする。SD カードのオフライン書き込みの場合、FriendlyARM.ini ファイルにパラメータ LCD-Type を指定する。システムを書き込む場合も起動する場合も LCD に接続せず、HDMI インターフェース付きのテレビと接続するといい。

LCD－Type は以下の値をサポートする。そのちは関連解像度と更新頻度に対応している。

HDMI1080P60
HDMI1080P60D
HDMI1080I60
HDMI1080I60D
HDMI1080P30
HDMI1080P30D
HDMI720P60
HDMI720P60D
HDMI576P16X9
HDMI576P16X9D
HDMI576P4X3
HDMI576P4X3D
HDMI480P16X9
HDMI480P16X9D
HDMI480P4X3
HDMI480P4X3D

高性能を保つとともに、比例出力を実現するため、最後に”D”が付く LCD-Type 値は Android のネイティブ解像度を HDMI の出力解像度の１／４に調整する。

例：

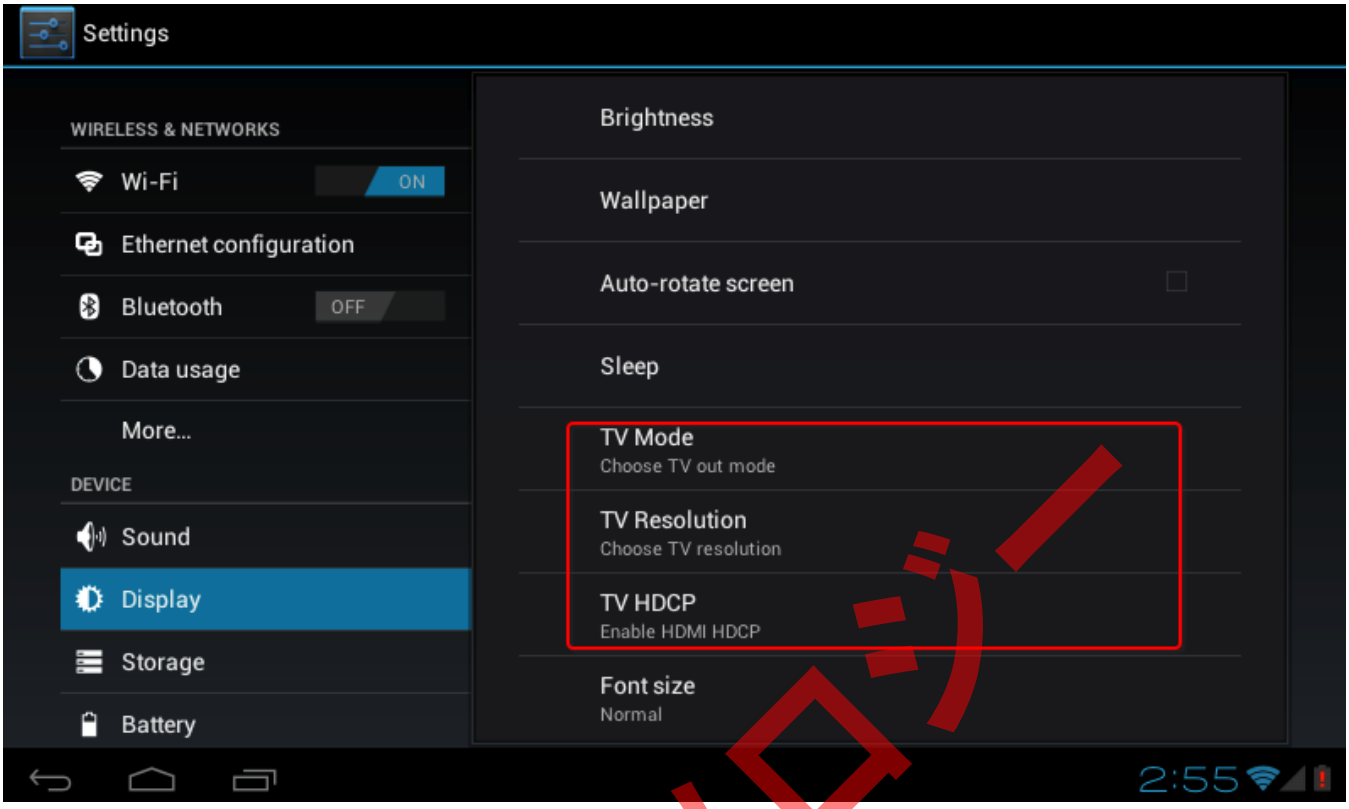
HDDMI1080P60　　Android のネイティブ解像度と HDMI の出力解像度は 1920x1080 である。

HDMI1080P60D　　Android のネイティブ解像度は 960x540 で、HDMI の出力解像度は 1920x1080 である。960*540 の画像を 1920x1080 に補間出力する。

3.1.16 HDMI 出力解像度の変更

HDMI の出力書式を設定する場合、以下のステップがあります。

システムのホームページより S/K[Settings]S/K[Display]を選択する。TV 出力の関連オプションが表示される。

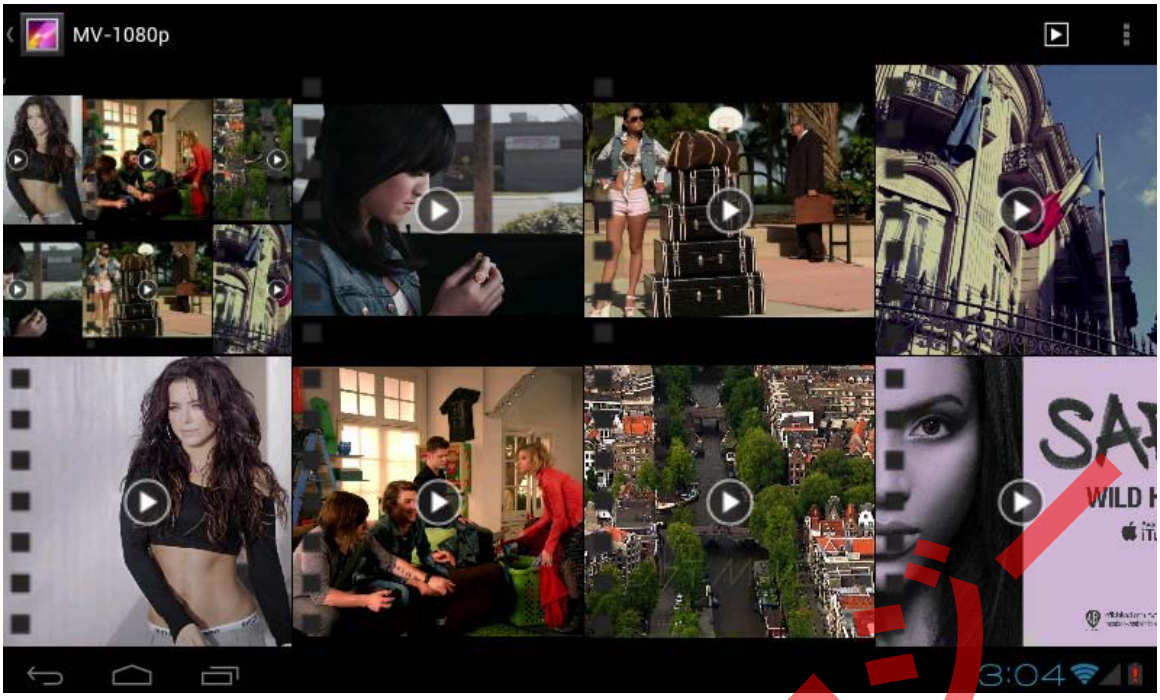


TV Resolution はビデオの解像度（480p/720p/1080p など（TV がサポートする解像度による））の設定に使われる。

3.1.17 HD ビデオの再生

再生するビデオを SD カードにコピーする（注：ビデオフォーマット： mp4 、オーディオ圧縮フォーマット aac 、テストビデオは CD の Test Video ディレクトリ下)、 Gallery アプリをオープン：

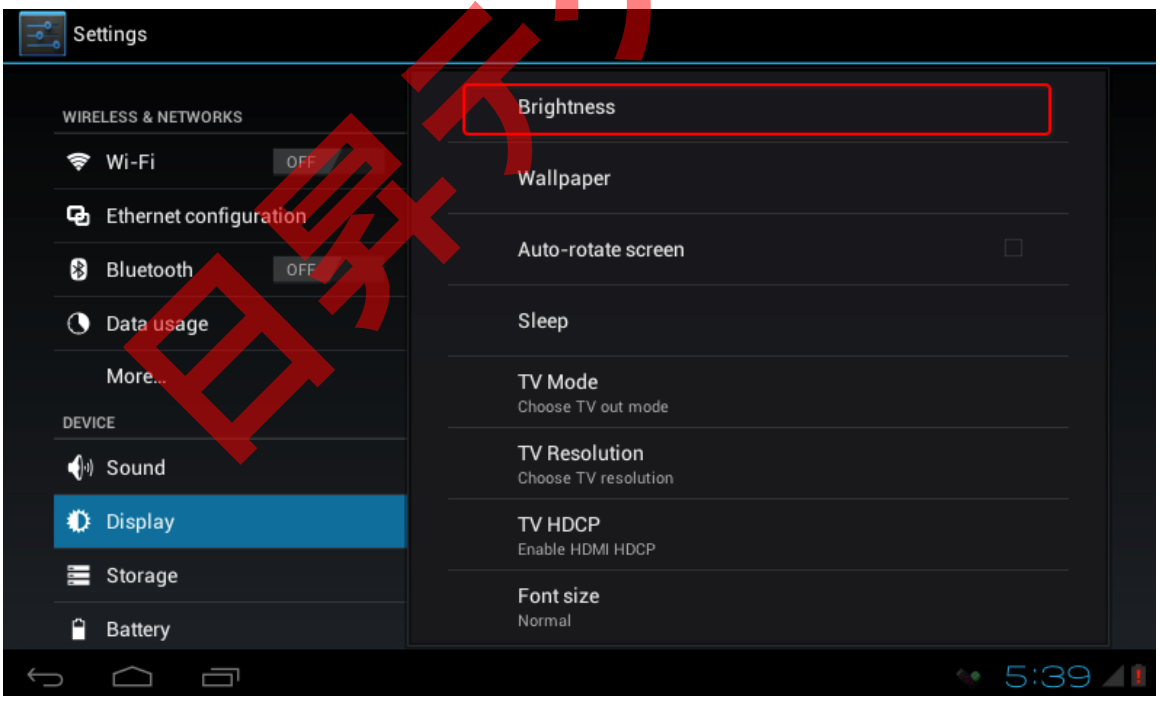
Gallery で検索ビデオを表示する、下記図の通り：



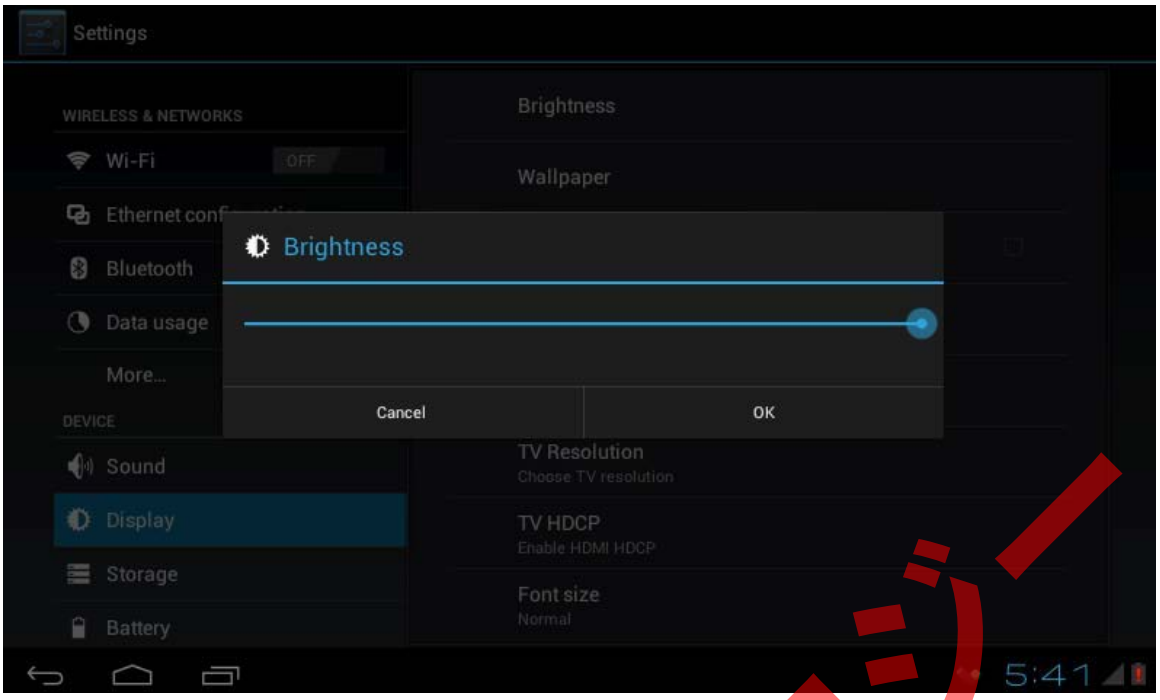
HDMI を接続した場合、画像と音声は同期テレビへ出力する。Exynos4412 は最高 1080p HD をサポートする。

3.1.18 バックライト調整

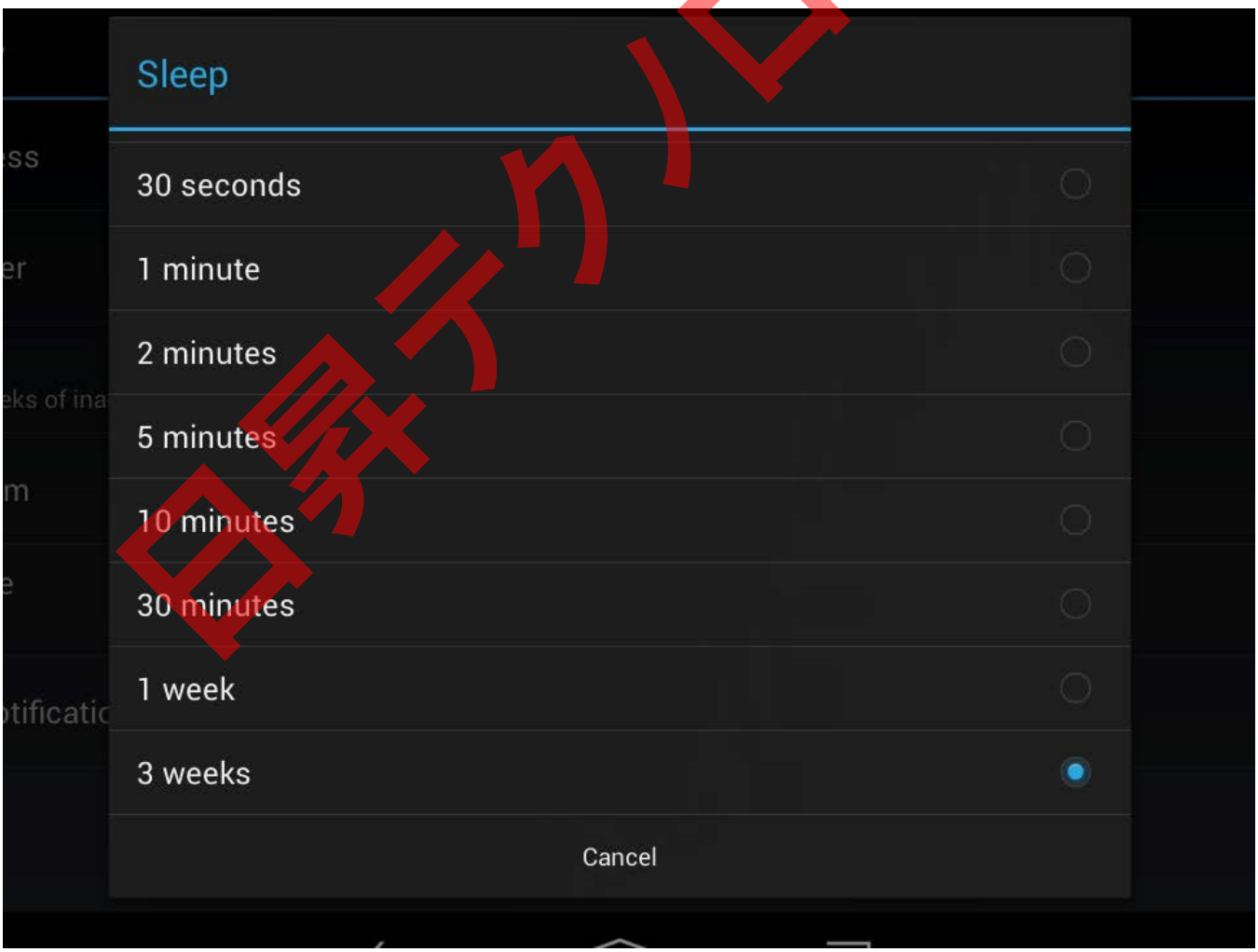
システム設定画面で`Display`->`Brightness`：



`Brightness`バックライトレベル調整できる



Display ->ﾞ Sleepﾞ 、バックライトオフ時間を設定する：



Screen timeout -> Never、 スクリーンは常亮とし、スリープ状態も入らない。

3.1.19 USB カメラ

ボード起動する前に、USB カメラを USB HOST インタフェースに差し込んで、USB Camera アプリをクリックして画像が見える。写真を取ることが出来る、SD カードに保存する。現時点まだ録画をサポートしない。
ソースコードはdevice/friendly-arm/tiny4412/USBCamera をご参照ください。

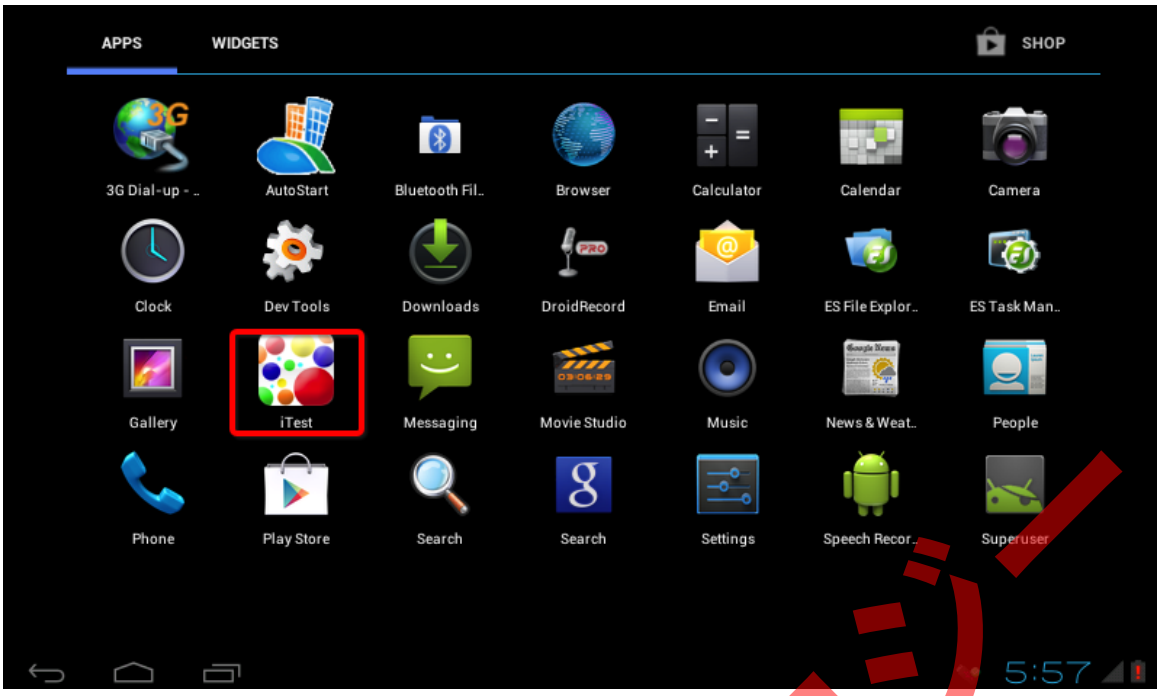
補充説明：
１、USBカメラについて、論理上はYVYV/YUY2フォーマットのカメラであれば全てサポートするはずですが、全て検証したではない。他のフォーマットのカメラであれば、下記変更で対応しますが、性能はYVYV/YUY2フォーマットのカメラより低くなる：
/system/build.propファイル
ro.kernel.android.cam_yuy2=n
注意：SAMSUNGメディアモジュールのため、カメラの出力書式はNV21ではなく、YV12に定義された。
２、USBカメラのデフォルトの解像度は864x480で、/system/build.propファイルのro.kernel.android.cam_def_sizeの変更で指定できる。例えばロジテックのC270は下記解像度をサポートする：1280x720、1184x656、960x720、960x544、864x480、800x448、544x288、352x288、320x176 。
３、USBカメラのサポートする解像度はro.kernel.android.cam_s_sizes の設定で指定できる。
４、/system/build.propファイル中でUSBカメラ関連の設定：

```
#
# USB Camera Preview and Picture Size (for Logitech C270 webcam)
#
ro.kernel.android.cam_def_size=544x288

#
# USB Camera Supported Size (for Logitech C270 webcam)
#
  ro.kernel.android.cam_s_sizes=1280x720,1184x656,960x720,960x544,864x480,800x448,640x480,544x288,352x288,320x176

#
# USB Camera Using YUY2 ColorSpace (Set to n will support more usb camera model)
#
ro.kernel.android.cam_yuy2=y
```

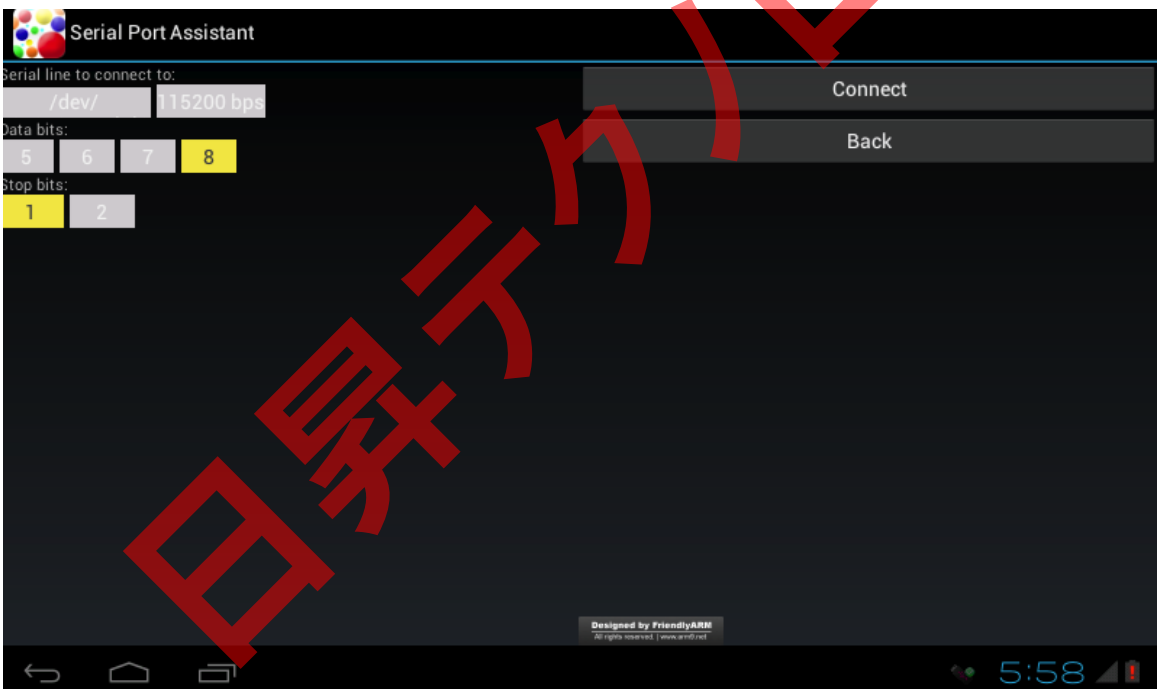
(6) Android ソースコードパケットの device/friendly-arm/tiny4412/USBCamera ディレクトリーに格納されているオーペンソースの DEMO をご参考ください。USB カメラのレビューと撮影をサポートする。

3.1.20 シリアルアシスタント

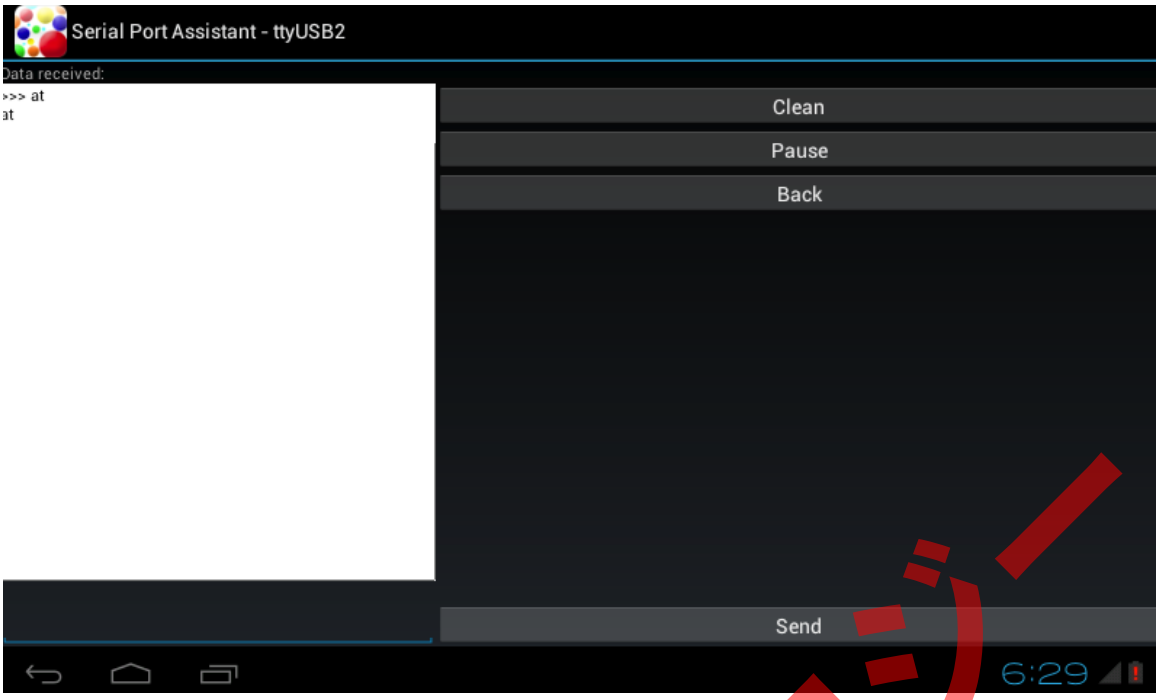
シリアルアシスタント機能、APP で iTest をクリック：



Serial Port Assistant をクリック、アシスタント起動后、左側でシリアルのボーレートなどのパラメータを設定する：



設定完成後、 Connect をクリック、シリアル接続、シリアルテストを行う。下記図の通り：



データをシリアルへ送信する場合、 Send の左側のテキストボックスで入力、send。
Pause はメッセージを一時停止、 Clean は受信したデータをクリア。

注意事項：

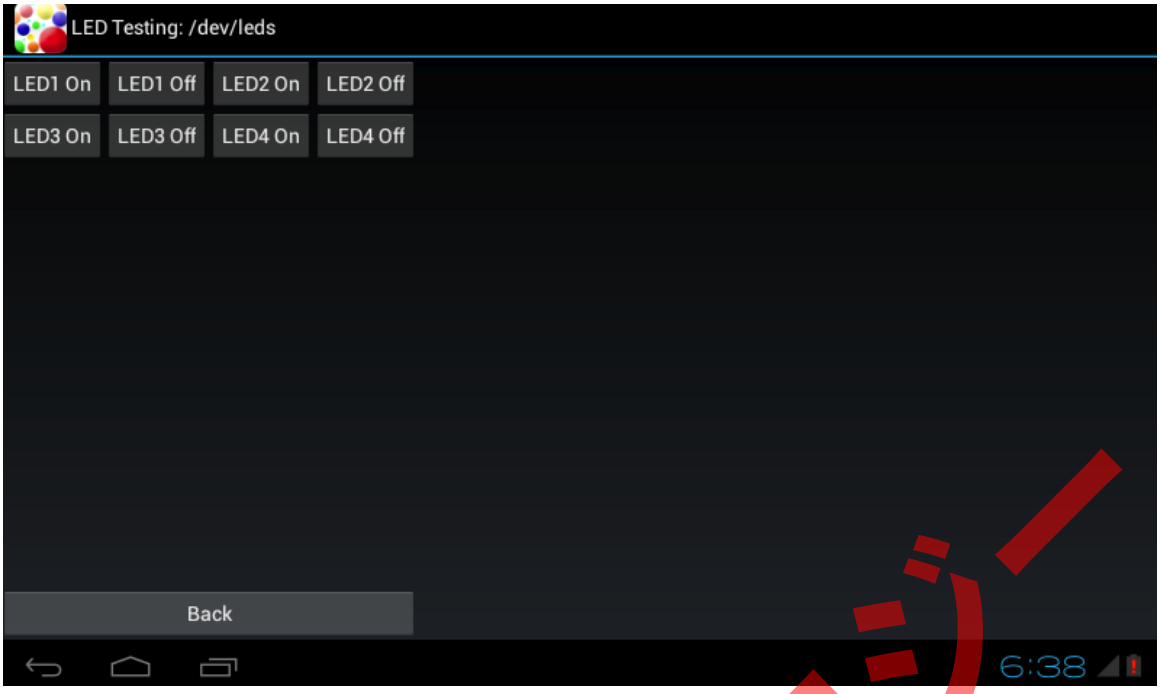
1）シリアルが接続できない場合、コマンドラインでコマンド｀ fuser ファイル名｀ でデバイスは使用中かテストする。

2）他のデバイスに使われてないのにオープンできない場合、｀ ls -l ファイル名｀で権限を確認する。コマンド｀ chmod 777 ファイル名｀でデバイスファイルの権限を変更し、再確認する。

3）s3c2410_serial0 デフォルトでは COM0 デバッグシリアルと接続している。

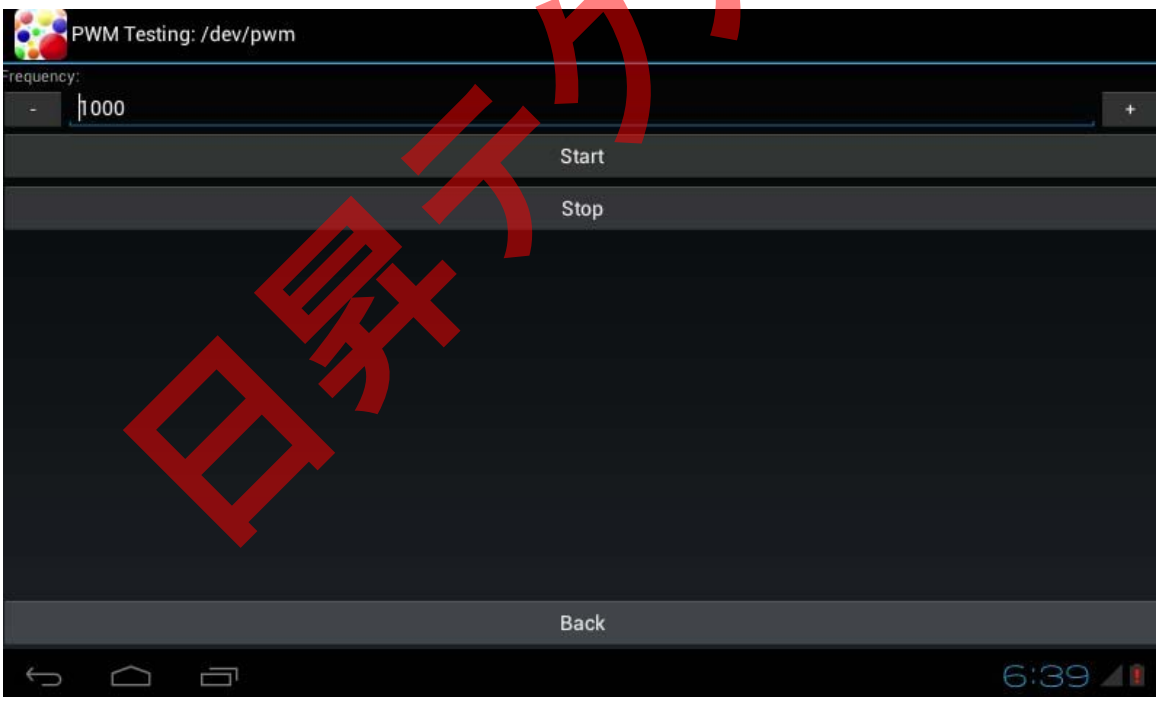
3.1.21 LED テスト

LED テストは iTest -> LED Testing 、LED テスト画面に入る、下記図の通り：



3.1.22 PWM ブザーテスト

PWM テストは iTest -> PWM Testing 、PWM テスト画面に入る、下記図の通り：



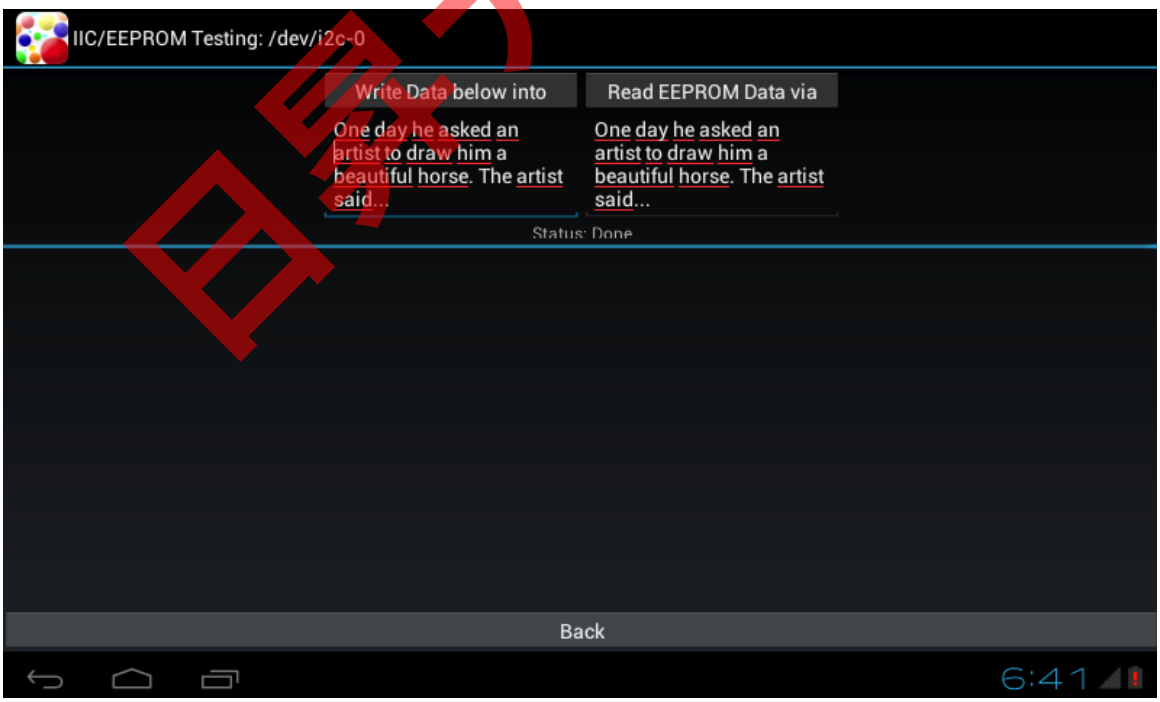
周波数を手動入力できる、Start でブザーを鳴らす、また゛+゛と゛-゛で周波数を調整できる。Stop で終了。

3.1.23 ADC テスト

ADC テスト、即ち A/D 変換結果を確認する時、 iTest -> A/D Convert 、ADC の変換結果が画面で表示する、下記図の通り：

3.1.24 I2C-EEPROM テスト

I2C-EEPROM の読み取り/書き込みテスト、iTest -> IIC/EEPROM Testing 、EEPROM のテスト画面に入る、下記図の通り：



左側の｀Write Data below into EEPROM｀をクリック、左側テキストボックスの文字を EEPROM に書き込み、右側の｀Read EEPROM Data via IIC｀をクリック、 EEPROM 中の文字を読み出し、右のテキストボック

スに保存する。
また、テキストボックスの中の文字を変更できる。

3.2 Android コンパイル環境構築

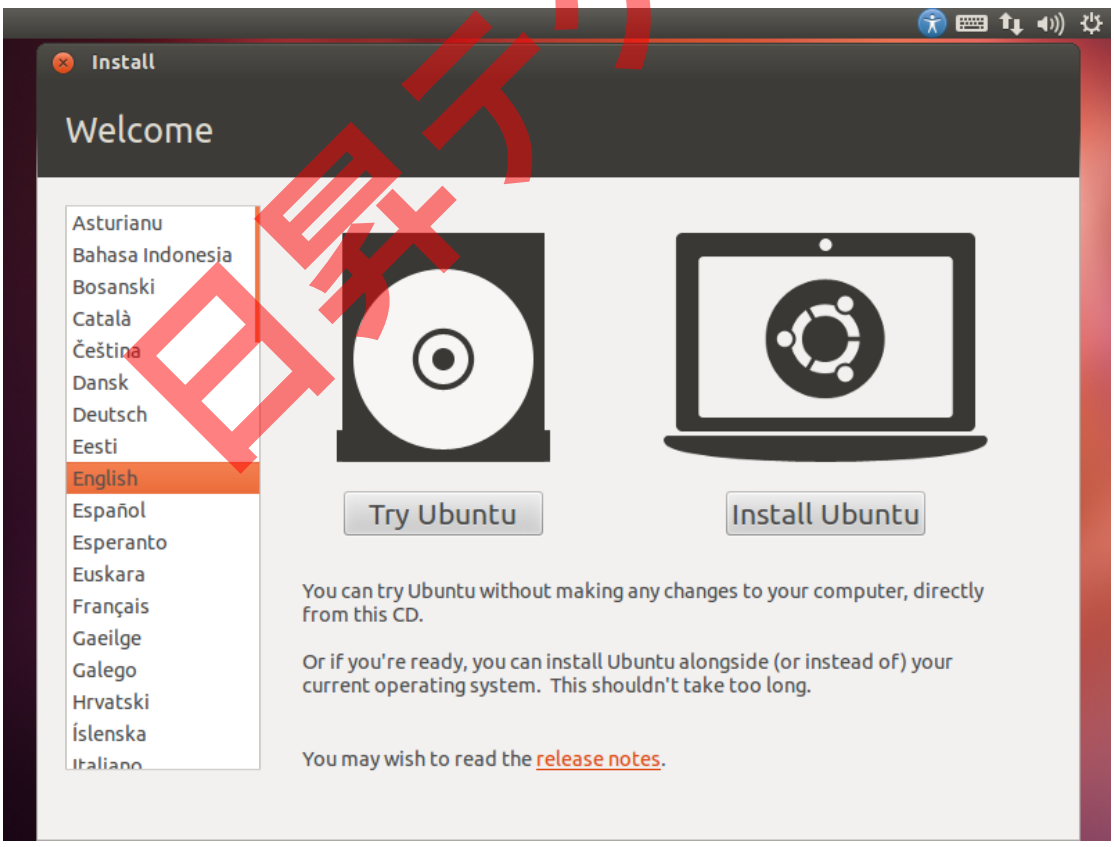
Android カーネル及び基本システム関連の開発、コンパイル環境を構築する。

Step1: Ubuntu12.04.2 64bit インストール、(注、Ubuntu12.04 ではなく、 Ubuntu12.04.2 である)；
Step2: Ubuntu12.04.2 で Android コンパイル環境インストール；
Step3: 付属 DVD から Android 4.2.2 のリソースコード、カーネルソースコード、クロスコンパイラ等をコピー、インストール。カーネルと Android4.2.2 のコンパイルに使用;
Step4: fastboot などの Android 用ツールをインストール。

3.2.1 Ubuntu12.04.2 64bit システムをインストール

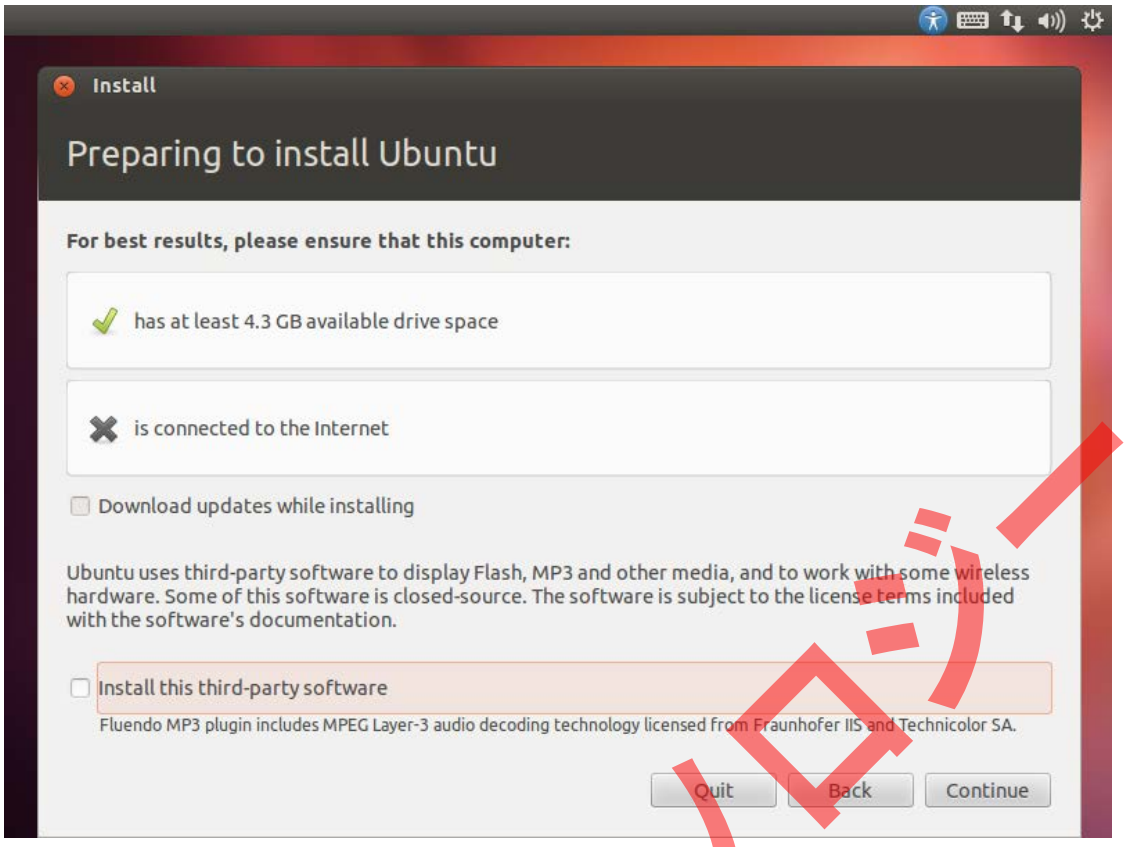
ネットから Ubuntu12.04.2 64bit の DVD イメージをダウンロードする。ダウンロード URL：http://releases.ubuntu.com/precise/、ファイル名：ubuntu-12.04.2-desktop-amd64.iso、CD を作成する。
注：インストールする時はオフラインインストールを薦める。
（例を示すのみで、ユーザーがインストールする時、ID を自由に設定出来る）

□
1) Ubuntu12.04.2 のインストールファイルをオープン：
□
2) 言語選択



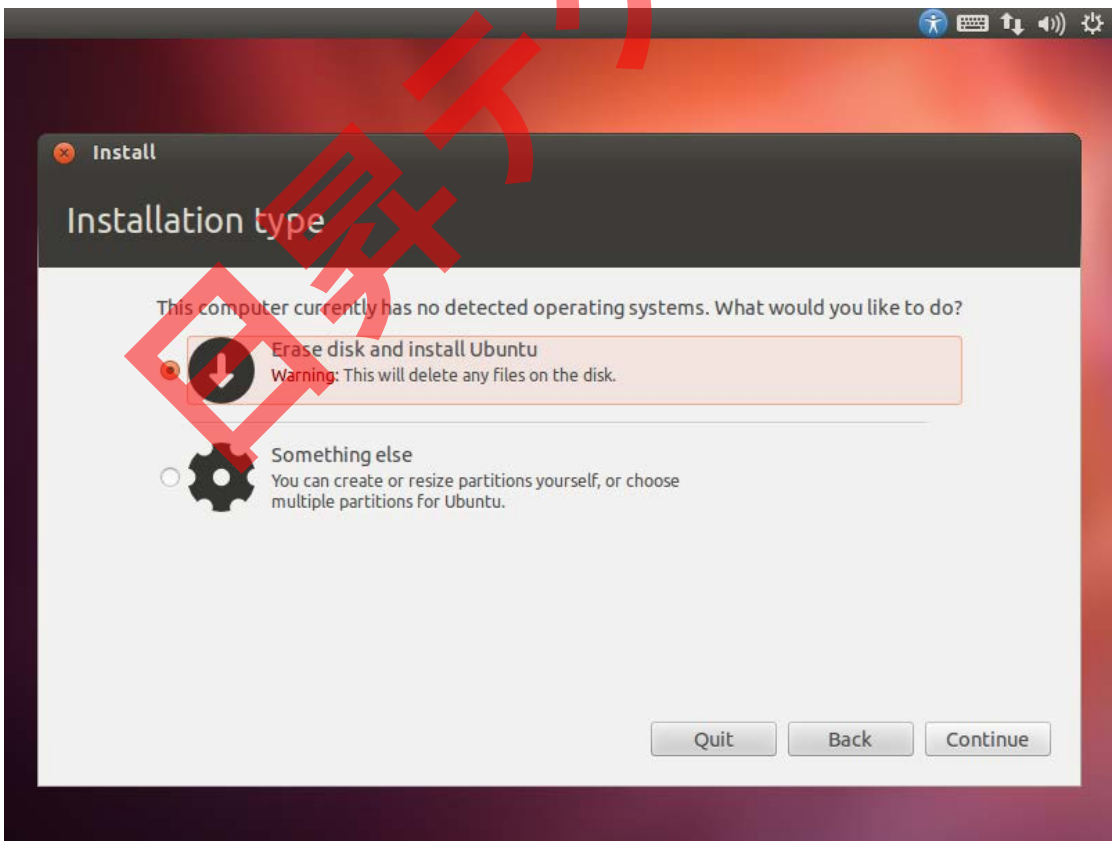
□

3) サードパーティーソフトウェアとアップデートの選択、全部無効とする：

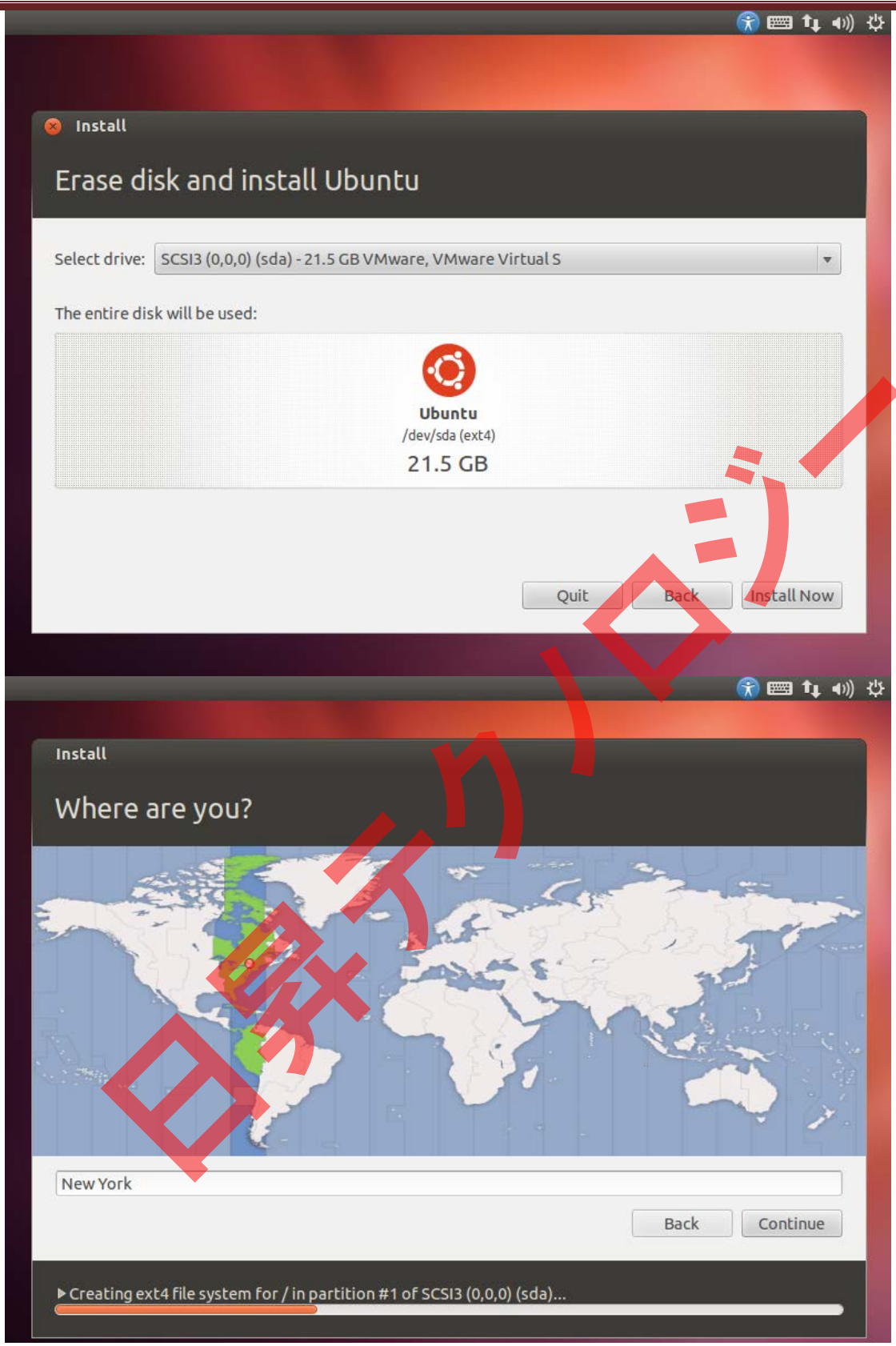


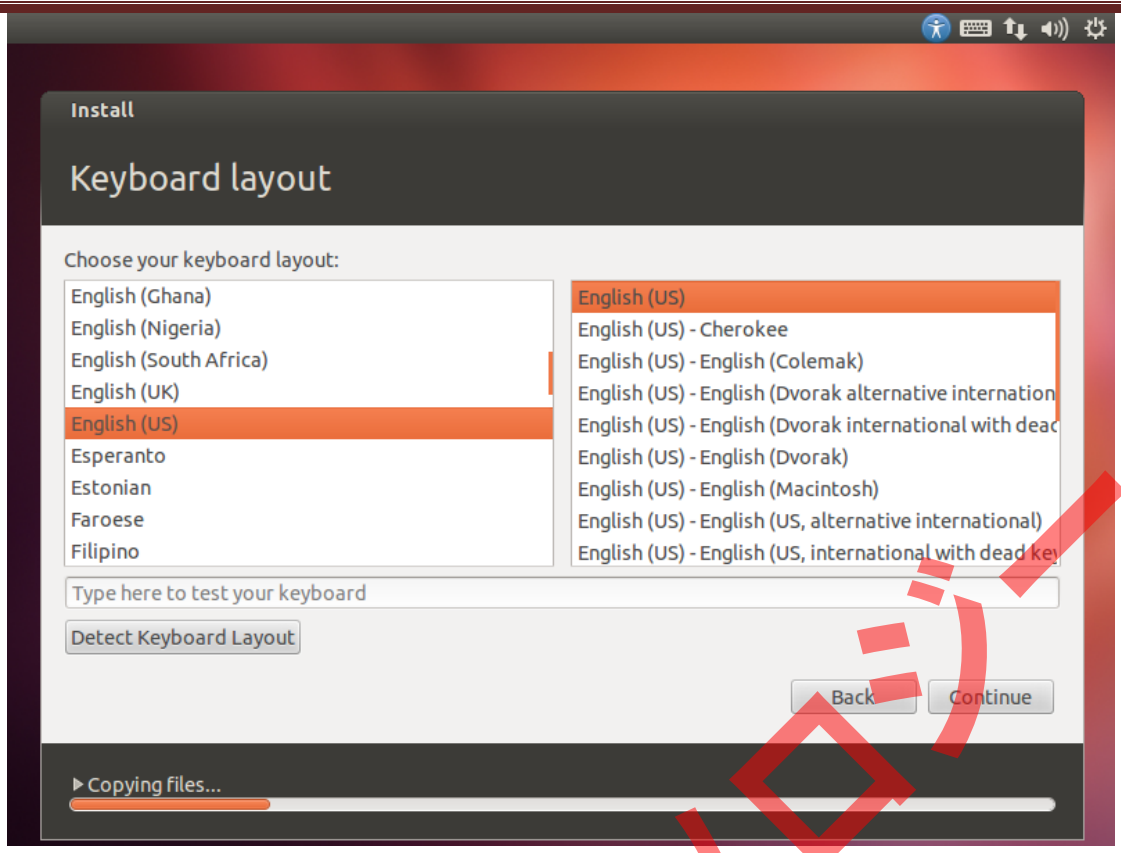
☐

4) パーティション設定、仮想マシン（VM など）でインストールする時は、第 1 項目を選択；他の場合は第 2 項目、別のパーティションを選択：

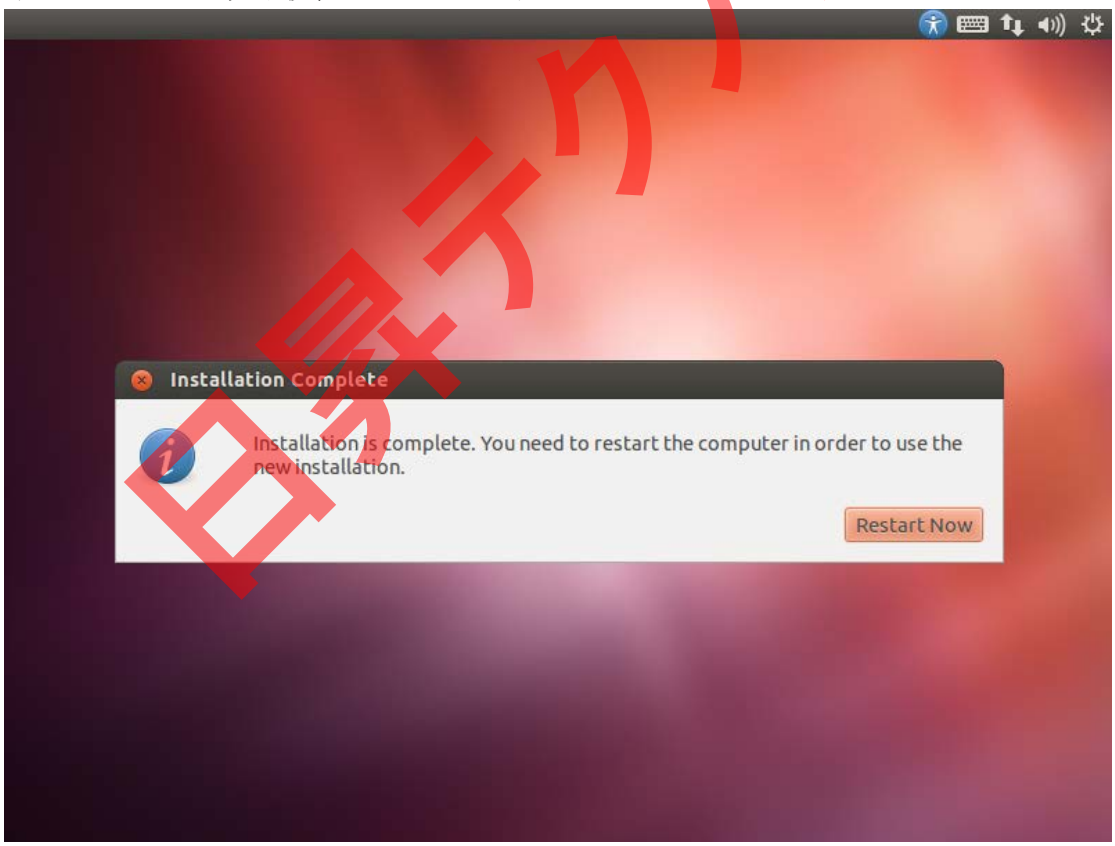


5) 選択後、continue：

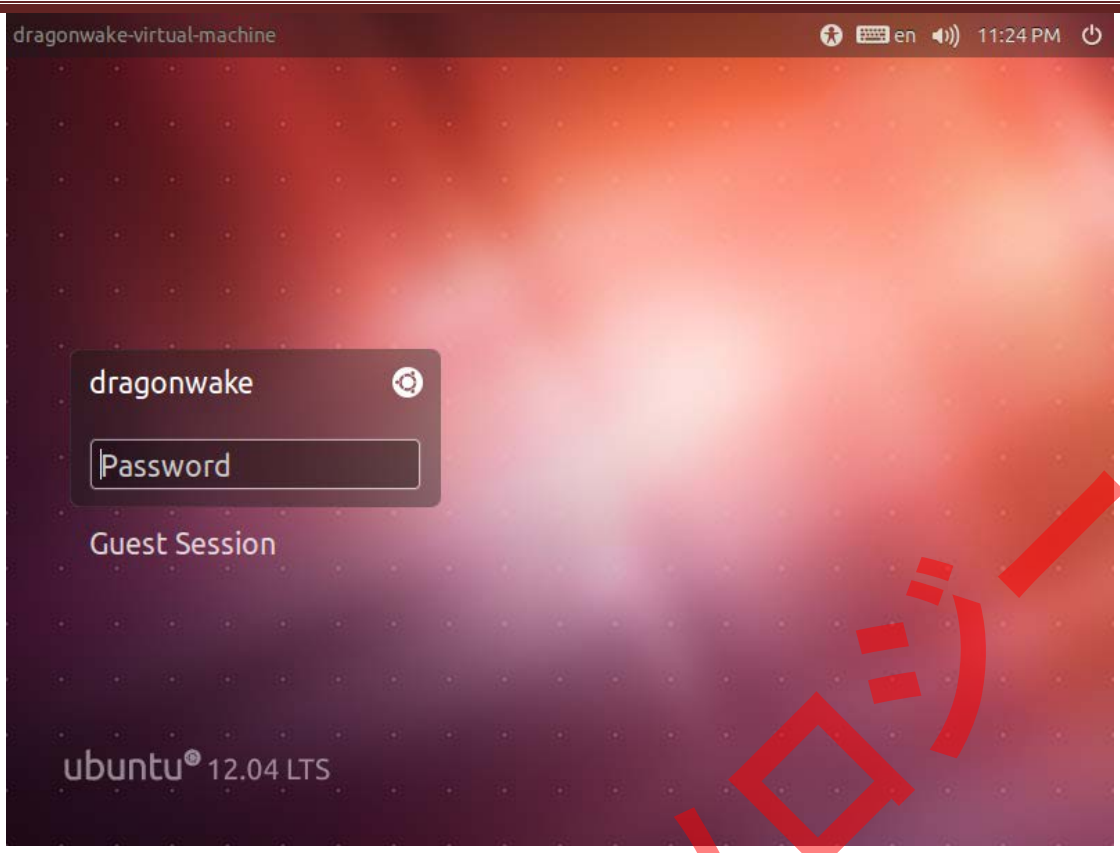

Install
Erase disk and install Ubuntu
Select drive:
SCSI3 (0,0,0) (sda) - 21.5 GB VMware, VMware Virtual S
The entire disk will be used:
Ubuntu
/dev/sda (ext4)
21.5 GB
Quit
Back
Install Now
Install
Where are you?
New York
Back
Continue
Creating ext4 file system for / in partition #1 of SCSI3 (0,0,0) (sda)...



6) インストール完了後、システム再起動：



7) 再起動完了後、ネットワークケーブルを接続、ユーザーID とパスワードを登録する。

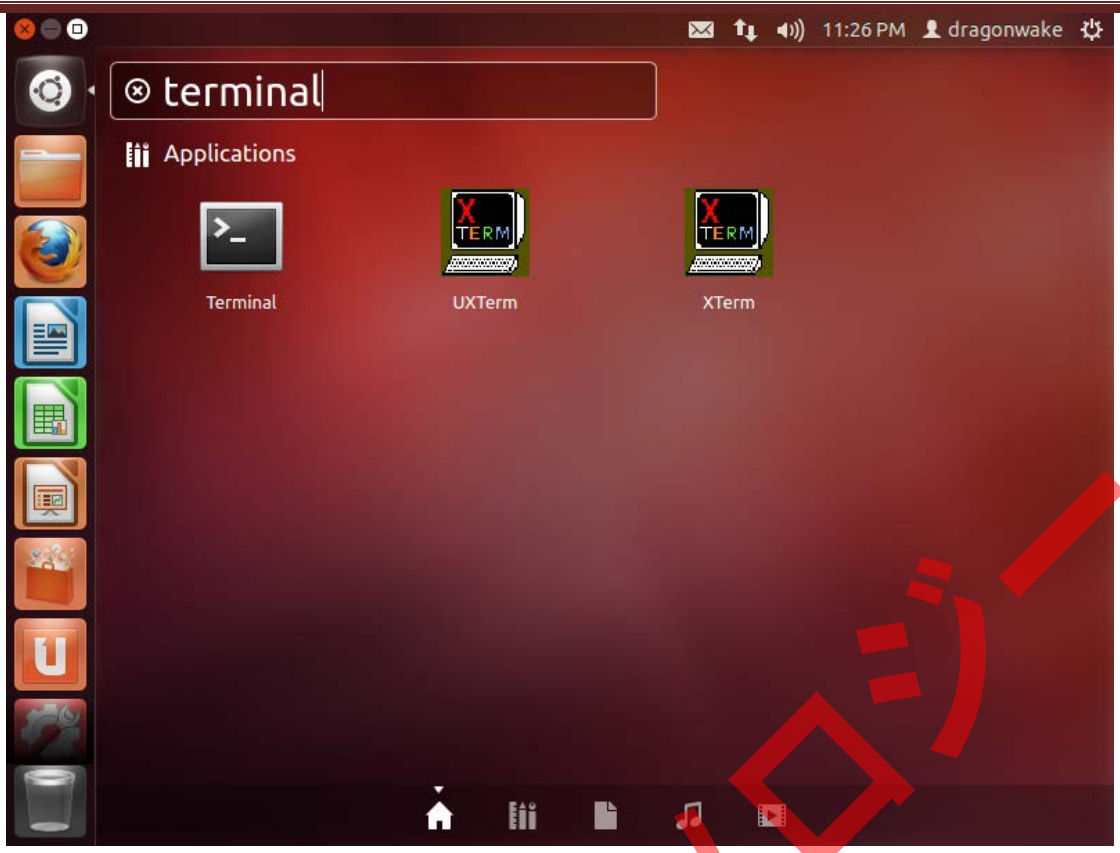


3.2.2 Ubuntu システム設定

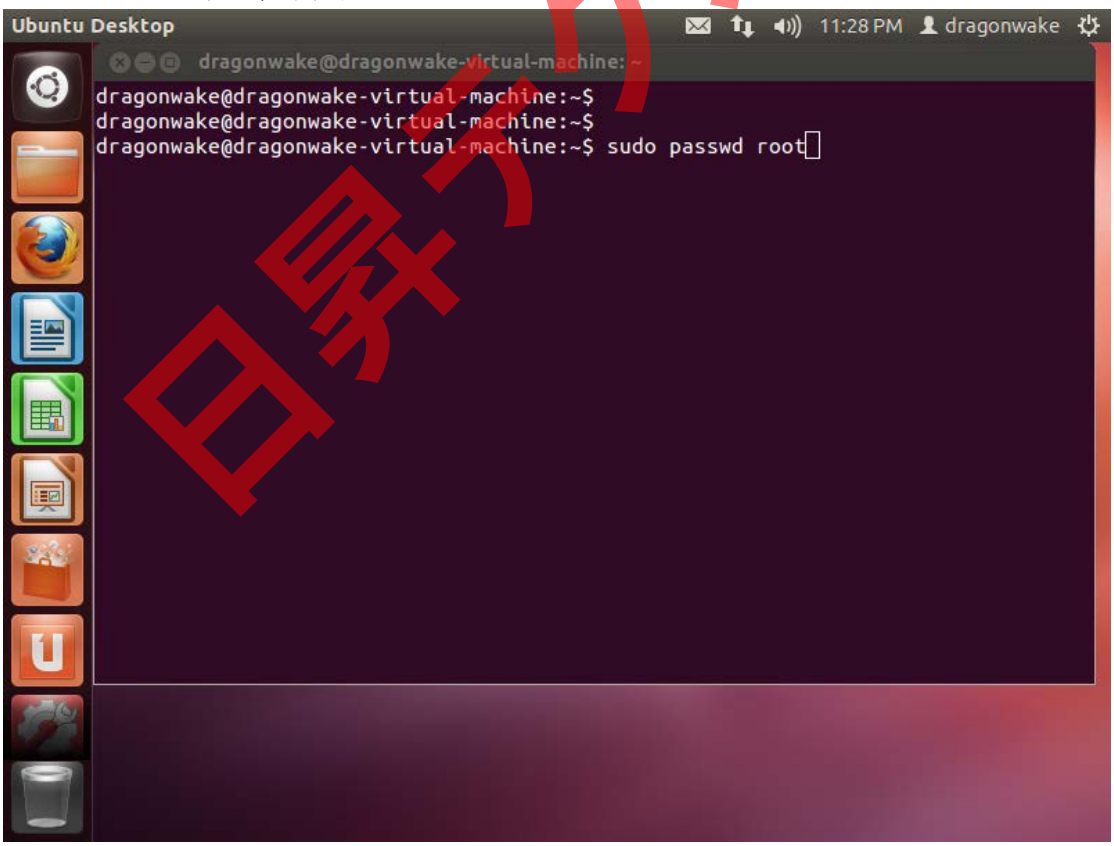
3.2.3 root ユーザーでログイン

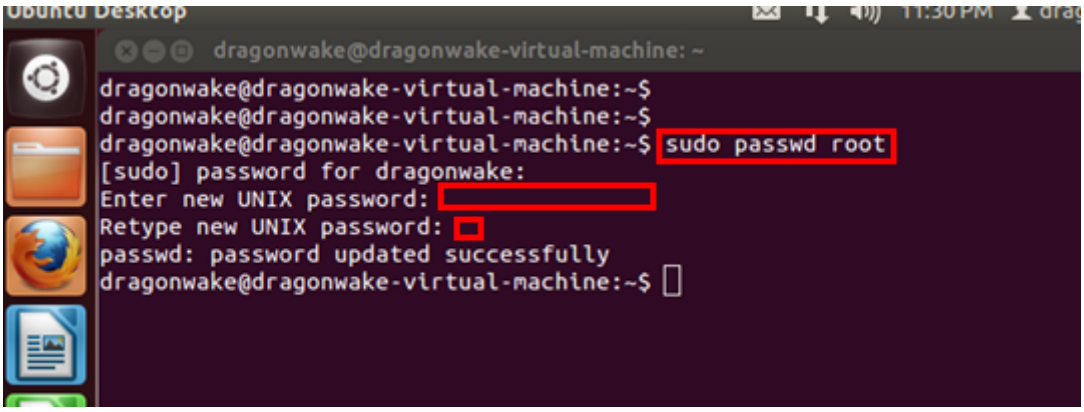
組み込み式開発ではよくシステムツールを使用する、例えば minicom と ifconfig など。従って、便利で開発するため、root ユーザーでログインをお勧め。デフォルトでは Ubuntu は一般ユーザーで登録するため、次の手順で設定する必要がある

コマンドラインで動作する、ターミナルコマンドラインモードに入る、左側上一番目のボタンをクリック、term 入力、検索結果からターミナルツールを選択、〝ターミナル〟起動：



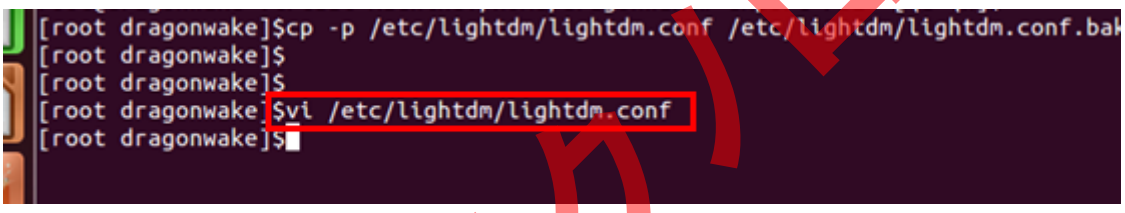
先ず、 root のパスワードを設定する。ターミナルで sudo passwd root を入力、 root ユーザーのパスワードを二回入力する、下記図の通り：



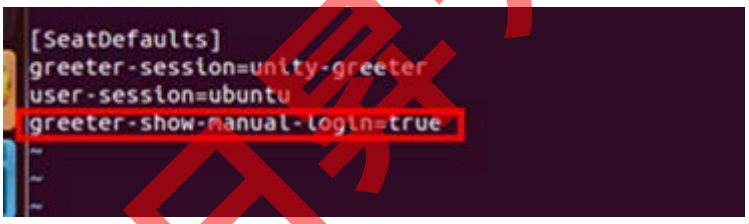


次、su root 入力、root ユーザーでログイン、コマンドプロンプトが長すぎるため、ここで export PS1='[¥u@¥h ¥W]$'を入力、プロンプトを再設定する、次に cp -p /etc/lightdm/lightdm.conf /etc/lightdm/lightdm.conf.bak を入力、 lightgdm 設定をバックアップする：

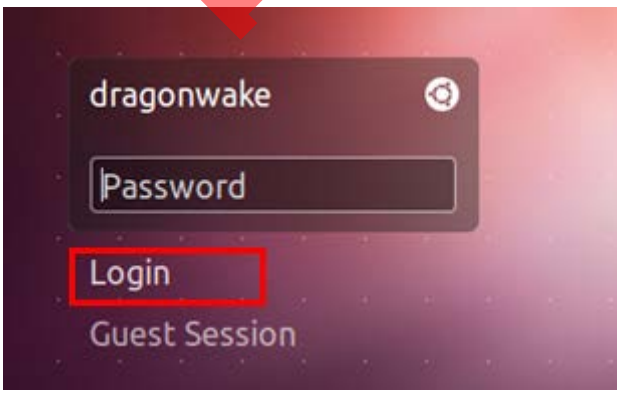
vi /etc/lightdm/lightdm.conf コマンドで /etc/lightdm/lightdm.conf をオープン：



ファイル最後で１ライン追加：
greeter-show-manual-login=true
最終内容は下記図の通り：



保存、再起動、ログイン画面で、｀ログイン｀をクリック：



root 入力、パスワード入力、 root ユーザーでシステムをログインする。

3.2.3.1 Android コンパイル必要なソフトウェアパケージインストール

Ubuntu のデフォルトインストールでは、ソフトウェア開発関連キットはないため、先ず関連ソフトウェア開発キットをオンラインインストール必要がある。

Step1、Tiny4412 付属 DVD から tools ディレクトリ下の ubuntu ディレクトリを tmp ディレクトリにコピー。ネットで iso ファイルをダウンロードする場合では 、下記のコマンドで iso をロードし、コピーする：

```
# mkdir -p /mnt/iso
# mount -o loop Tiny4412-20130707.iso /mnt/iso
# cp /mnt/iso/tools/ubuntu /tmp/ -a
```

cp /mnt/iso/tools/ubuntu /tmp/ -a
(注：#はプロンプト、入力必要がない)

Step2、 jdk6 インストール：

```
# cd /tmp/ubuntu/jdk6/
# chmod 755 install-sun-java6.sh
# ./install-sun-java6.sh
```

Step3、最後、install-devel-packages.sh スクリプト実行、ソフトウェア開発関連ソフトウェアパッケージをオンラインインストール：

```
# cd /tmp/ubuntu/
# chmod 755 install-devel-packages.sh
# ./ install-devel-packages.sh
```

途中でソフトウェアパッケージダウンロードの提示があり、 Y 入力確認する。

3.2.4 クロスコンパイラインストール

arm-linux-gcc-4.5.1（ Mini210 と同じ）を採用、カーネルコンパイルの時は自動的に armv7 コマンドセットを使用し、ハードウェア浮動小数点演算をサポートする。

インストール手順は下記の通り：

Step1：付属 DVD の Android ディレクトリ中の arm-linux-gcc-4.5.1-v6-vfp-YYYYMMDD.tgz を/tmp ディレクトリにコピーし、ディレクトリに入り、解凍コマンドを実行する：

#cd /tmp

#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-YYYYMMDD.tgz -C /

注：C の次にスペースがあり、C は大文字、英語の｀Change｀の頭文字で、ここではディレクトリ変更（チェンジ）の意味とする。

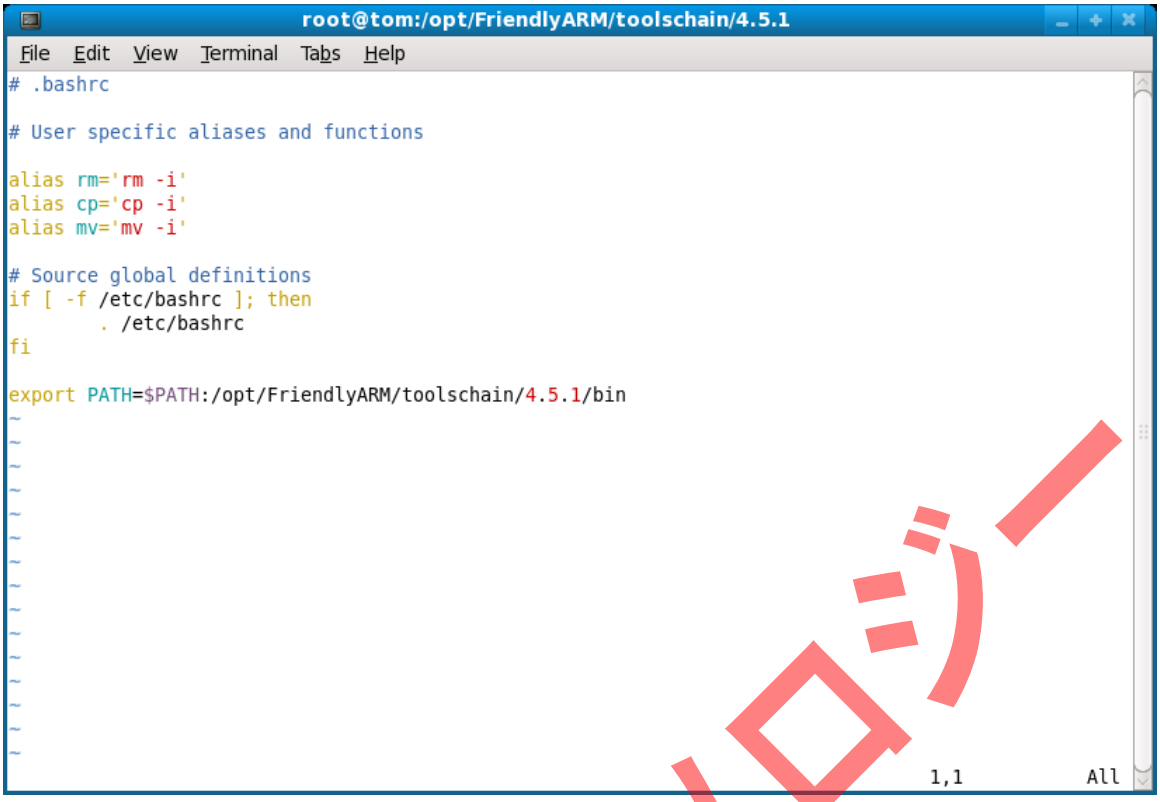
コマンド実行、 arm-linux-gcc を/opt/FriendlyARM/toolschain/4.5.1 ディレクトリ下にインストールする。

Step2：コンパイラのパスをシステム環境変数に追加し、コマンド実行する。

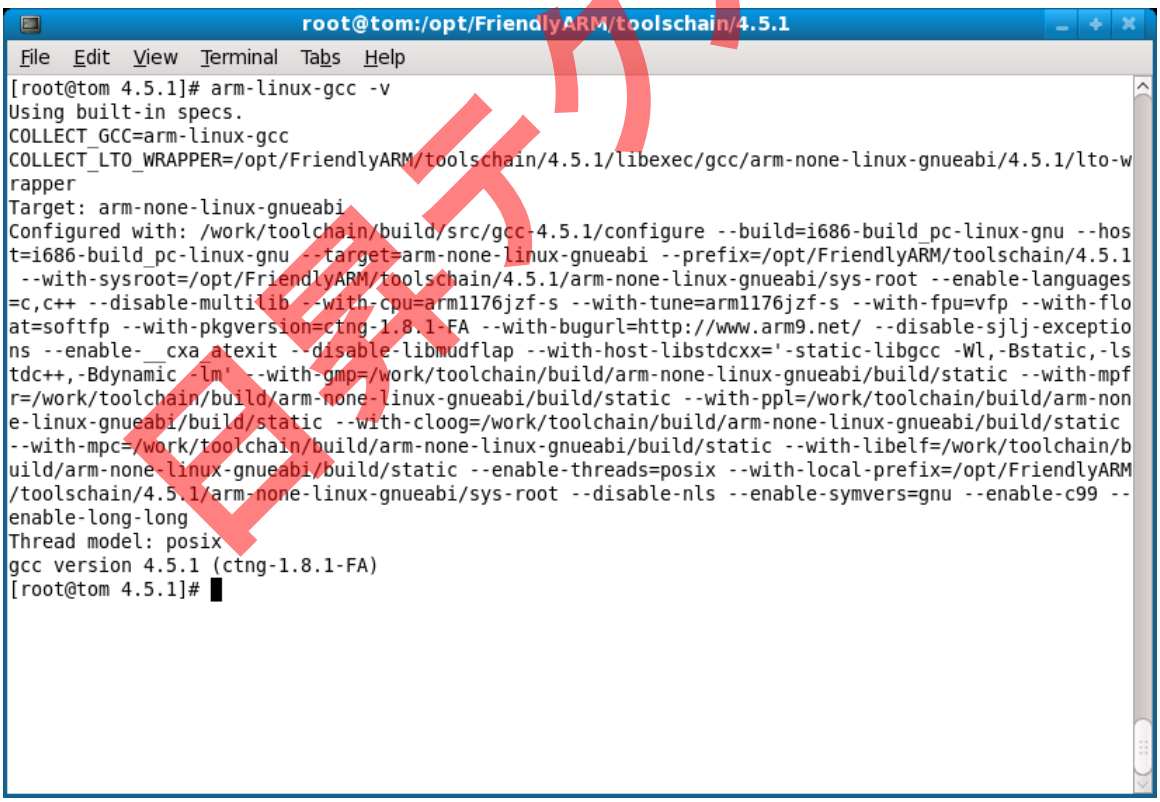
#gedit~/.bashrc

~/.bashrc ファイルコンパイル、 注｀ bashrc｀前では 1 つの｀ .｀があり、 編集後は export

PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin となる。保存、終了。



システムログイン(再起動必要ばない、スタート->logout)、設定有効、コマンドラインで arm-linux-gcc -v を入力、下記の情報が表示し、クロスコンパイル環境がインストール成功と表示する。



3.2.5 Andorid4.2.2 ソースコード解凍・インストール

ディレクトリ/opt/FriendlyARM/tiny4412/android を作成
コマンドラインで 入力
#mkdir -p /opt/FriendlyARM/tiny4412/android
Tiny4412 付属 DVD から Android ディレクトリを tmp ディレクトリ下にコピーする、ネットから iso ファイルをダウンロードする場合、下記のコマンドで iso をロードし、コピーする：

```
# mkdir -p /mnt/iso
# mount -o loop Tiny4412-20130707.iso /mnt/iso
# cp /mnt/iso/Android /tmp/ -a
```

(注：#はプロンプト、入力必要なし)

(1)Android4 カーネルソースコード解凍・インストール
ディレクトリ/opt/FriendlyARM/tiny4412/android で下記のコマンドを実行する：
#cd /opt/FriendlyARM/tiny4412/android
#tar xvzf /tmp/Android/linux-3.5-YYYYMMDD.tgz
生成した linux-3.5 ディレクトリに完全なカーネルソースコードを含む
説明：YYYYMMDD は発行年月。

(2)Android4.2.1 ソースコードパッケージ解凍・インストール
ディレクトリ/opt/FriendlyARM/tiny4412/android で下記のコマンドを実行する：
#cd /opt/FriendlyARM/tiny4412/android
#tar xvzf /tmp/Android/android-4.2.2_r1-fs-YYYYMMDD.tar.gz
android-4.2.2_r1 ディレクトリ生成。
説明：YYYYMMDD は発行年月。

3.3 Linux カーネル設定とコンパイル

Linux3.5 カーネルコンパイル：

```
#cd /opt/FriendlyARM/tiny4412/android/linux-3.5
#cp tiny4412_android_defconfig .config ; config 前に゛.゛付き
```

make menuconfig を実行し、設定を編集する、完了後、 make でコンパイルを開始する：

```
# make
```

終了後 arch/arm/boot ディレクトリで zImage を生成し、SD カード images/Android/下の zImage を書き換えて Tiny4412 に書込む。

3.4 ソースコードから Android 作成

Android ソースコードを便利にコンパイルするため、色んなソースコードパッケージを作成しており、コンパイルと書込みイメージ作成に使用するスクリプトも提供している。スクリプトの機能は下記の通り：

スクリプト	機能	呼び出し例
setenv	Android コンパイル関連環境変数を設定	. setenv ；(注゛.゛の次に 1 つのスペースがある)

gen-img.sh	fastboot と SD カード書込み用のシステムイメージ system.img と ramdisk-u.bin を生成	./gen-img.sh
burn-img.sh	USB 書込み実行：Tiny4412 開発ボードは fastboot ダウンロードモード下の時、USB ケーブル接続後、スクリプトを実行すると、システムイメージファイル system.img と ramdisk-u.bin は Tiny4412 開発ボードにダウンロードする。	./burn-img.sh

Android ソースコードをコンパイルするには、コマンドラインで下記のコマンドを実行する（Android 4.2.2_r1 ソースコード）：

コンパイル：

```
#cd /opt/FriendlyARM/tiny4412/android/ android-4.2.2_r1
#. setenv ;注`.` の次スペースがある
# make
```

（注： make に-j パラメータを追加し、 CPU のマルチコア加速を利用し、コンパイル速度を上げる、例えば 4 コアパソコンで、 make -j4 を入力し、コンパイルする）

Android コンパイルには長時間を掛かるので、バーチャルマシンでコンパイルのを控え、マルチコア CPU と実際の Linux システムを使用しコンパイルする方が速度が速い。

3.5 インストール実行ファイルシステムイメージ生成

コンパイル成功後、下記のコマンドを実行し、システムイメージファイル system.img と ramdisk-u.img を生成する：

```
#./gen-img.sh
```

gen-img.sh 実行、Android ソースコードは現在ディレクトリ下で system.img と ramdisk-u.img を生成する：

system.img と ramdisk-u.img を SD カードの images/Android ディレクトリにコピーし、オフラインで書込み；または burn-img.sh を実行し、 fastboot で USB 書込み出来る。

3.6 Andorid アプリでハードウェア操作

開発ボードのハードウェアリソースを操作する Android アプリを便利に開発するため、関数ライブラリ

(libfriendlyarm-hardware.so)を提供し、Tiny4412 上のハードウェアリソースアクセスに使用する。ハードウェアデバイス：シリアルデバイス、ブザーデバイス、EEPROM、ADC デバイス等をサポートする。

iTest アプリも本関数ライブラリで開発した。Android で iTest を実行し、関数ライブラリの機能を確認できる。

3.6.1 関数ライブラリの使用(libfriendlyarm-hardware.so)

Android に libfriendlyarm-hardware.so が内蔵され、本ライブラリファイルは Android ソースコードディレクトリの下記のパスに置いてる:

```
vendor/friendly-arm/exynos4412/rootdir/system/lib/libfriendlyarm-hardware.so
```

開発ボードでは /system/lib/libfriendlyarm-hardware.so ディレクトリ下に存在する。

Eclipse で Android アプリを開発する場合、下記の方法で libfriendlyarm-hardware.so を利用できる:

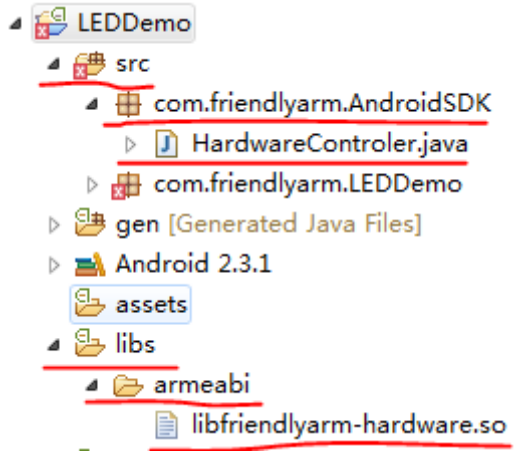
1) Android アプリケーションディレクトリにロケート、アプリケーションディレクトリ下で libs ディレクトリを作成、 libs ディレクトリ下で armeabi ディレクトリを作成、最後に libfriendlyrm-hardware.so ライブラリファイルを armeabi ディレクトリ下にコピーする。

2)アプリケーションディレクトリに戻り、 src ディレクトリ下に入り、com¥friendlyarm¥AndroidSDK 三層のディレクトリを作成、AndroidSDK ディレクトリ下でファイルエディタで 1 つのソースコードファイルを追加し、HardwareControler.java と命名する、次はファイル中に下記のコードを追加する：

```
package com.friendlyarm.AndroidSDK;
import android.util.Log;

publicclass HardwareControler
{
        /* Serial Port */
        staticpublicnativeint openSerialPort( String devName, long baud, int dataBits, int stopBits );

        /* LED */
        staticpublicnativeint setLedState( int ledID, int ledState );

/* PWM */
        staticpublicnativeint PWMPlay(int frequency);
        staticpublicnativeint PWMStop();

/* ADC */
        staticpublicnativeint readADC();
        staticpublicnativeint readADCWithChannel(int channel);
staticpublicnativeint[] readADCWithChannels(int[] channels);

        /* I2C */
        staticpublicnativeint openI2CDevice();
        staticpublicnativeint writeByteDataToI2C(int fd, int pos, byte byteData);
        staticpublicnativeint readByteDataFromI2C(int fd, int pos);

        /* IO */
        staticpublicnativeint write(int fd, byte[] data);
        staticpublicnativeint read(int fd, byte[] buf, int len);
        staticpublicnativeint select(int fd, int sec, int usec);
        staticpublicnativevoid close(int fd);
```

```
/* return 6410 or 210 */
staticpublicnativeint getBoardType();

static {
try {
        System.loadLibrary("friendlyarm-hardware");
        } catch (UnsatisfiedLinkError e) {
Log.d("HardwareControler", "libfriendlyarm-hardware library not found!");
        }
    }
}
```

設定完了後、 Eclipse 起動、 Eclipse の左側で項目リストを右クリックし、` Refresh` 。項目の関連ディレクトリは下記の通り：(赤ライン部分)：



HardwareControler のインタフェースを使用するには、下記のコードを追加、 HardwareControler クラスをインポートする：

```
import com.friendlyarm.AndroidSDK.HardwareControler;
```

HardwareControler クラスのインタフェースを呼び出す。次節で HardwareControler クラスの関数インタフェースを説明する。

3.6.2 関数ライブラリ(libfriendlyarm-hardware.so)インタフェース説明

アプリ層で、前節の HardwareControler クラスで libfriendlyarm-hardware.so ライブラリ中のインタフェースを呼び出せる。 HardwareControler クラス中のインタフェースの定義は下記の通り、インタフェースはクラス方法のため、HardwareControler オブジェクトインスタンスを作成する必要がない。

3.6.2.1 シリアル通信のインタフェース説明

シリアル関連インタフェースは下記の通り：

インタフェースネーム	パラメータ及び戻り値説明	機能説明
int openSerialPortEx(String devName, long baud, int dataBits, int stopBits, String parityBit, String flowCtrl)	パラメータ説明： devName: シリアルデバイスファイル名、選択可能の値： /dev/s3c2410_serial1 /dev/s3c2410_serial2 /dev/s3c2410_serial3 /dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 baud: ボーレート dataBits: データビット （範囲 5~8、通常 8 ）	指定シリアルデバイスをオープン、ファイル記述子を戻る。

	stopBits: ストップビット （範囲 1~2、通常 1 ） parityBit: O は ODD、E は EVEN、N はなし flowCtrl :H はハードウェアフローコントロール、S はソフトウェアフローコントロール、N はなし 戻り値説明： シリアルオープン成功、戻るのシリアルファイル記述子を使用し、read、write と select などのを実現できる；オープン失敗の場合、-1 を戻る。	
int write(int fd、byte[] data)	パラメータ説明： fd: 書き込みデータのファイル記述子 data: 書き込みデータ 戻り値説明： 成功 - 書き込みデータのバイト数を戻る、 失敗 - -1 を戻る。	オープンしたデバイスまたはファイルでデータ書き込み。
int read(int fd、byte[] buf、int len)	パラメータ説明： fd: 読み取りデータのファイル記述子 buf: 保存データのバッファ len: 読み取りデータのバイト数 戻り値説明： 成功 - 読み取りのバイト数を戻る； 失敗- -1 戻る。 read を呼び出す前で既にファイル語尾に到達する場合、今回の read は 0 戻る。	オープンしたデバイスまたはファイルからデータ読み取り。
int select(int fd、 int sec、 int usec)	パラメータ説明： fd: 問い合わせのファイル記述子 sen: ブロックのデータ待ち時間（単位：秒） usec: ブロックのデータ待ち時間 （単位：ナノ秒、1 ミリ秒= 1000 ナノ秒 ns） 戻り値説明： fd に読み取りデータがある場合、1 戻る； 無い場合、0 戻る； 失敗-1 戻る。	オープンしたデバイスまたはファイルから読み取りデータ検索。
void close(int fd)	パラメータ説明： fd: クローズするのファイル記述子 戻り値説明： 無し	指定ファイル記述子をクローズ

インタフェースの使用説明：

先ず openSerialPort でシリアルデバイスをオープン、スレッドでまたは timer を介し、select を呼び出し、 インタフェースポーリングし、シリアルデバイスのデータ状況を判断する。データある場合、 read を呼び出し、インタフェースからデータを読み取り。

シリアルにデータ書き込む場合、write インタフェースを呼び出す。

シリアル使用完了後、close でシリアルをクローズ。

3.6.2.2 LED ON/OFF のインタフェース説明

LED 動作のインタフェースは下記の通り：

インタフェースネーム	パラメータ与戻り値説明	機能説明
int setLedState(int ledID, int ledState)	パラメータ説明： ledID：ON/OFF LED ID(範囲 0~3) ledState: 1 点灯、0 滅 戻り値説明： 成功ー0、失敗ー1 を戻る	LED ON/OFF に設定するインタフェース。

3.6.2.3 PWM ブザー鳴らす/停止のインタフェース説明

ブザー動作のインタフェースは下記の通り：

インタフェースネーム	パラメータと戻り値説明	機能説明
int PWMPlay(**int** frequency);	パラメータ説明： frequency: 鳴らしの周波数 戻り値説明： 成功ー0、失敗ー1 を戻る	指定の周波数でブザーを鳴らす
int PWMStop();	パラメータ説明： 無し 戻り値説明： 成功ー0、失敗ー1 を戻る	ブザー停止

3.6.2.4 ADC の変換結果読み取りのインタフェース説明

ADC 動作のインタフェースは下記の通り：

インタフェースネーム	パラメータと戻り値説明	機能説明
int readADC()	パラメータ説明： 無し 戻り値説明： 成功 - ADC 変換結果を戻る 失敗 - -1 戻る	ADC 変換結果読み取り
int readADCWithChannel(intchannel)	パラメータ説明： channel: 指定チャンネル ADC の値を読み取り、選択可能のパラメータは 0、1、4、5	指定チャンネルの ADC 変換結果読み取り

	戻り値説明： 成功 － ADC 変換結果を戻る 失敗 － -1 戻る	
int[] readADCWithChannels(int[] channels);	パラメータ説明： channels: チャンネルする ADC チャンネルのアレー 戻り値説明： 成功 -複数 ADC 結果戻る（アレー)、 失敗 － 空き戻る	1 回で複数チャンネルの結果を読み取り、 性能は良い

3.6.2.5 EEPROM データの書き込み/読み取りのインタフェース説明

EEPROM 動作のインタフェースは下記の通り：

インタフェースネーム	パラメータと戻り値説明	機能説明
int openI2CDevice();	パラメータ説明： 无 戻り値説明： IIC デバイスオープン、 成功 - IIC デバイスのファイル記述子戻る 失敗 - -1 戻る。	IIC デバイスオープン、ファイル記述子戻る。 デバイスオープン後、writeByteDataToI2C と readByteDataFromI2C 関数を使用し EEPROM に読み取り/書き込みを行う。
intwriteByteDataToI2C(**int** fd、 **int** pos、 **byte** byteData);	パラメータ説明： fd: openI2CDevice から戻るのファイル記述子 pos: データが EEPROM の位置を指定する（0~255) byteData: 書き込みのデータ 戻り値説明： 成功 - 書き込みのバイト数戻る、 失敗 - -1 戻る。	EEPROM にデータ書き込み(1 回は 1 つのバイトを書き込む)。 注、当関数は時間消費の関数(約 10 ミリ秒)、ワーカースレッドで呼び出す。
int readByteDataFromI2C(**int** fd、 **int** pos);	パラメータ説明： fd: openI2CDevice から戻るのファイル記述子 pos: データが EEPROM の位置を指定する(0~255) 戻り値説明： 成功 - 読み取りのデータ戻る（可强、 失敗 - -1 戻る、 read 呼び出し前既にファイルの語尾に到達する場合、今回の read は 0 戻る。 戻り値のタイプは int で、byte に変換する必要がある。	オープンしたデバイスまたはファイルからデータ読み取り。 注、当関数は時間消費の関数(約 10 ミリ秒)、ワーカースレッドで呼び出す。

void close(int fd)	パラメータ説明： fd: クローズするのファイル記述子 戻り値説明： 無し	指定ファイル記述子をクローズ

インタフェースの使用説明：

openI2CDevice を呼び出し IIC デバイスをオープン、次に新規スレッドを作成、writeByteDataToI2C を呼び出し、EEPROM にデータを書き込む；または readByteDataFromI2C を呼び出し、 EEPROM からデータ読み取る。 writeByteDataToI2C と readByteDataFromI2C 関数は読み取り/書き込み時、10 ミリ秒の遅延がある、元の GUI スレッドで直接呼び出すには、画面一時反応なしな場合がある（ショートブロック）。

EEPROM では 256 バイトのデータを保存出来る、書き込み/読み取りの指定位置の範囲は 0~255、1 回は 1 バイト読み取り/書き込み。

EEPROM 動作完了後、close を呼び出し、ファイル記述子をクローズする。

3.6.3 サンプル

付属DVDの゛Android゛ディレクトリ下に1つのデモプログラムLEDDemoがあり、Windows環境ではEclipseでプロジェクトをオープンしてlibfriendlyarm-hardware.soの使い方を確認できる。

miniUSB ケーブルで開発ボードと接続して、 Eclipse で直接デモ・プログラムを開発ボードにダウンロードし、実行しテストできる。

第四章 Linux マニュアル

4.1 Linux の GUI

Tiny4412 の Linux には Qtopia2.2.0、QtE4.7 と Qt Extended 4.4.3、三つの GUI（グラフィカルユーザーインターフェース）システムがプリインストールされている。三つの GUI システムを自由に切り替えることができ、非常に便利である。その中、Qtopia2.2.0 は起動時デフォルト GUI システムである。

Qtopia2.2.0 は TROLLTECH 会社が Qt/Embedded 2.3 を基に開発した PDA バージョン（最終バージョン）の GUI システムである。Qtopia2.2.0 以後、TROLLTECH 会社は PDA バージョンの GUI を提供したことがない。最新バージョンの Qtopia は携帯バージョン(Qt Extended 4.4.3)だけ、TROLLTECH 会社は 2009.3 から Qtopia PDA バージョンと携帯バージョンの GUI システムの授権を停止したが、Qt/Embedded（略称は QtE）の開発を続けている。

QtE の最新バージョンは http://qt.nokia.com/からご確認ください。本開発ボードで移植したバージョンは QtE-4.7.0 である。

Qt Extended 4.4.3 は TROLLTECH が開発した携帯デスクトップの最新バージョンであり、該当シリーズの最終バージョンである。Qtopia4 と呼ぶ。

4.1.1 メイン画面

Qtopia システムメイン画面には五つのアイコンがあり、5 種類のアプリ/ファイルを表示している。アイコンをクリックすれば、対応のサブクラス画面に入る。

メイン画面の左下の「start」アイコンをクリックすれば、五つのサブメニューが出る。これらはメイン画面の上のアイコンと対応している。

4.1.2 Mp3 の再生

「アプリケーション」タブの「ミュージック」をクリックし、プレーヤー画面が出る。「Audio」リストに MP3 ファイルをチェックし、再生ボタンをクリック。

説明：「Audio」リストは「Documents」の全ての Mp3 ファイルを対応している。

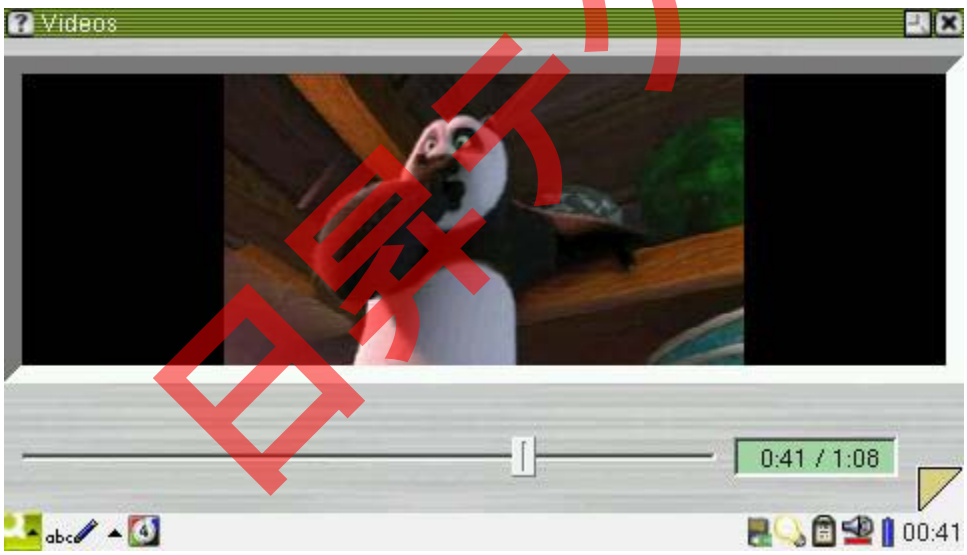
提示：直接に「Documents」の Mp3 ファイルをクリックし再生できる。

4.1.3 ビデオの再生

「アプリケーション」タブの「ビデオ」をクリックし、プレーヤー画面が出る。「Video」リストにビデオファイルをチェックし、再生ボタンをクリック。このプレーヤーは Qtopia にプリインストールされたもので、H.264/H.263/Mpeg4 などのビデオをスムーズに再生できる。

説明：「Video」リストは「Documents」の全てのビデオファイルを対応している。

提示：直接に「Documents」のビデオファイルをクリックし再生できる。



4.1.4 SMPlayer

MPlayer はオーペンソースのクロスプラットフォーム Player である。各種のオーペンソースの AV デコードライブラリを介し、各種書式のビデオファイルを再生できるし、各種のディスプレイデバイス（X11, Framebuffer, SDL, DFB など）への出力をサポートする。ここのバージョンは Framebuffer に基づく物である。

Mplayer はグラフィカルインターフェースをサポートしないが、多数のフロントインターフェイス

（SMPlayer,KMPlayer,KPlayer など）を利用できる。ここでは Qt4.8.5 ライブラリに基づく SMPlayer を利用する。詳細的な内容は下記の公式サイトにご参照ください。

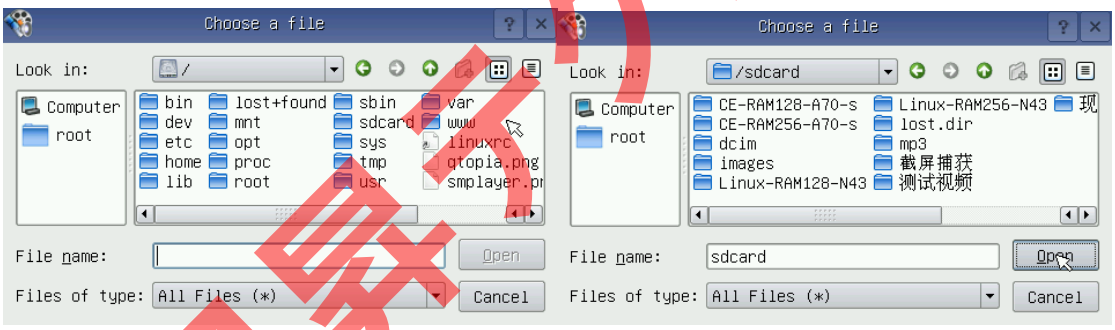
MPlayer の公式サイト：http://www.mplayerhq.hu

SMPlayer の公式サイト：http://smplayer.sourceforge.net/

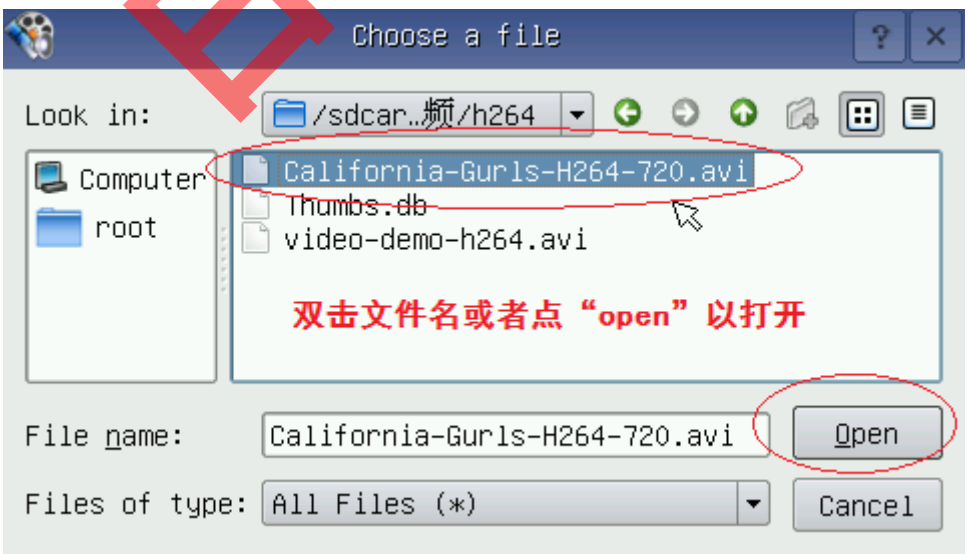
4.1.4.1SMPlayer によるビデオの再生

”FriendlyARM”プログラムグループで「SMPlayer」アイコンをクリックして Player を起動する。

S/K[Open]をクリックし、再生ファイルを選択する。”sdcard”ディレクトリーを開く。

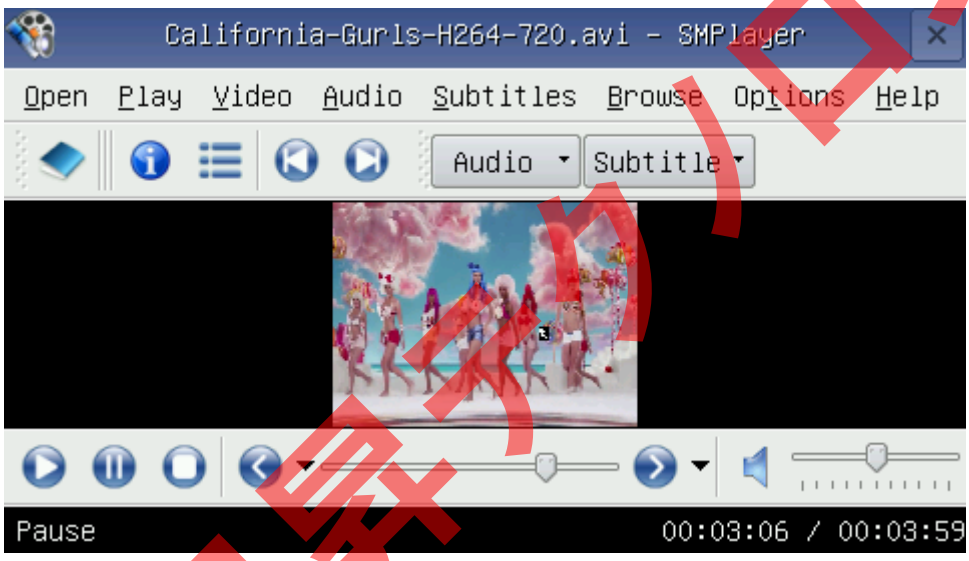


再生ファイルを選択し、S/K[Open]をクリックする。



フルスクリーン再生が始まる。

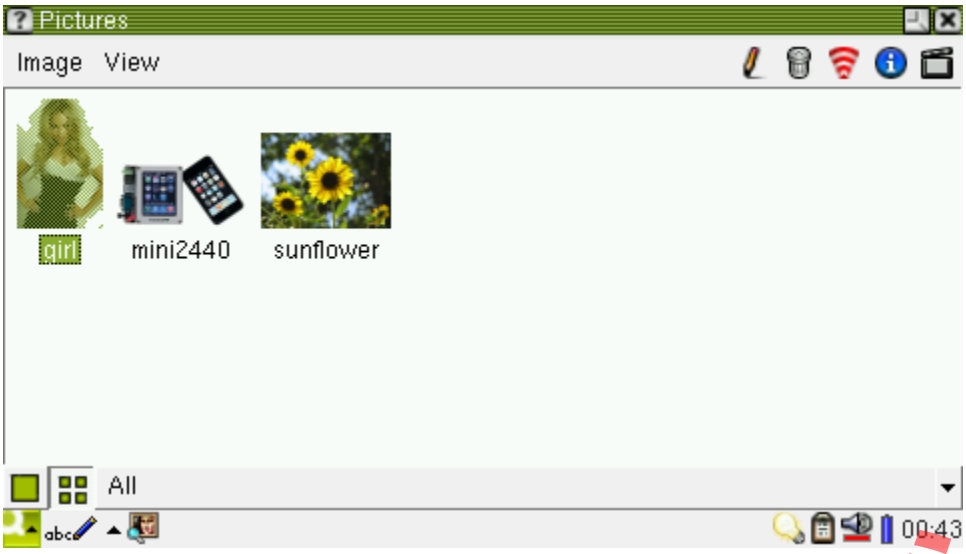
ビデオ再生中に、スクリーンをクリックすると、再生が一時停止になり、再生画面が縮小した Player ベース画面に遷移する。



その時、ボリュームの調節、早送り、早戻しなどの操作できる。

4.1.5 ピクチャのビュー

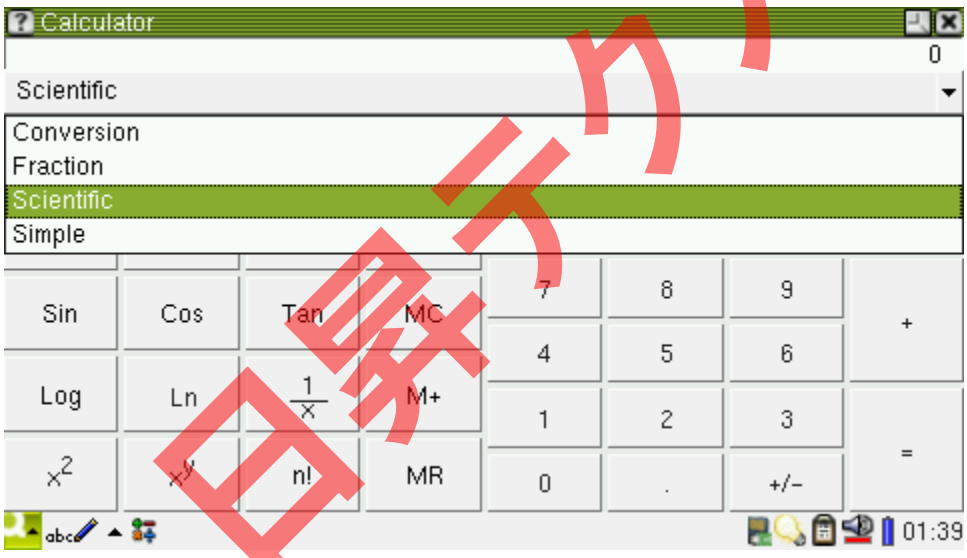
アプリケーション」タブの「ピクチャ」をクリックし、「Documents」のピクチャのサムネイルが出る。ピクチャのある SD カード或いは USB フラッシュメモリを差し込めば、その中のピクチャのサムネイルも同時に出る。



Otopia2.2.0 システムのピクチャーブラウザーは以前の Atopia1.7.0 と比べて、かなり改善している。ピクチャの簡単な編集が可能、手入れも便利である。

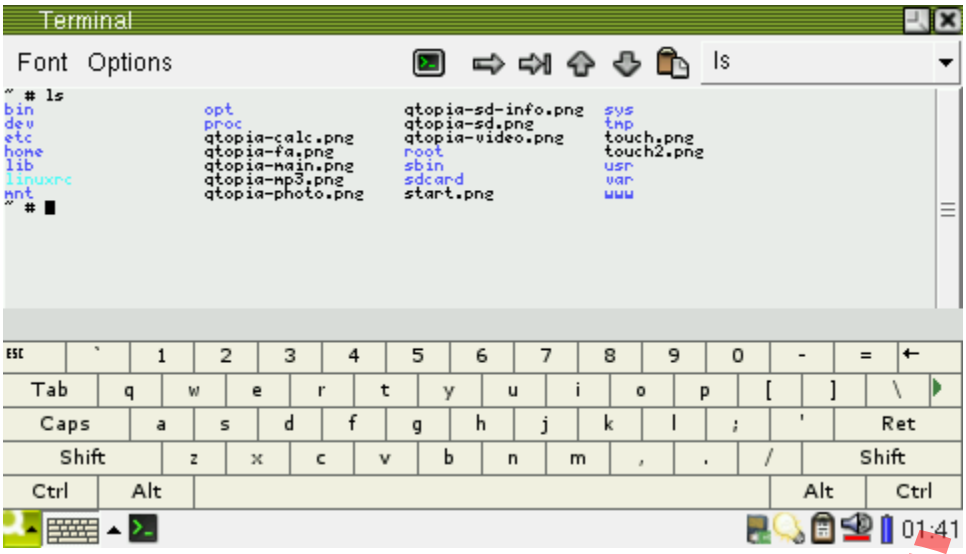
4.1.6 電卓

「アプリケーション」タブの「電卓」をクリックし、電卓の画面が出てくる。選択リストから Simple, Fraction, Scientific, Conversion 等のタイプの電卓を選択できる。画面は下図の通り：



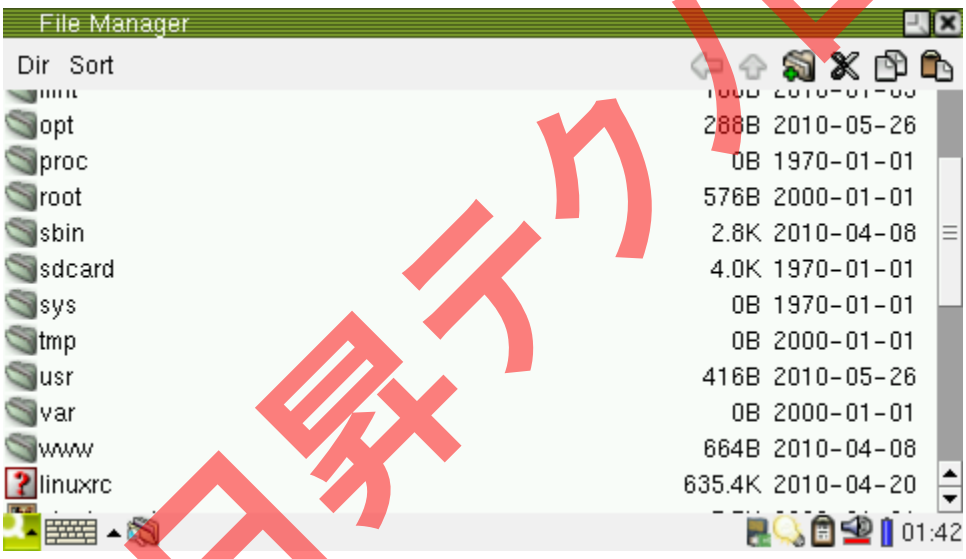
4.1.7 ターミナル

「アプリケーション」タブの「ターミナル」をクリックし、ターミナル画面が表示される。USB キーボードを接続する（ターミナルを起動する前 USB キーボードを接続したら使えなくなる）あるいはスクリーンの下のソフトキーボードを使って Linux コマンドを入力する。なお、表示モードを変更する場合、Option メニューに設定を行える。画面は下図の通り：



4.1.8 ファイルブラウザー

「FriendlyARM」タブの「File Browser」をクリックし,下図の画面が出てくる：

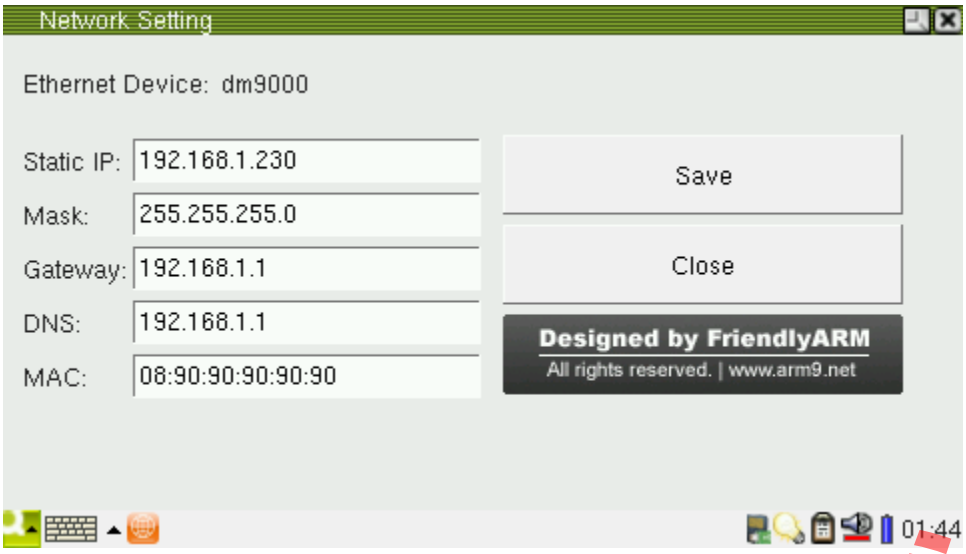


ファイルブラウザーを利用し、開発ボードの目次とファイルのビューと管理ができる。

説明：Qtopia-2.2.0 のオリジナルのバージョンにはファイルブラウザーがないため、Qtopia-1.7.0 のファイルブラウザーを移植した。

4.1.9 イーサネットの設定

「FriendlyARM」タブの「ネットワーク設定」をクリックし、ネットワークを設定できる。画面は下図の通り：



ここでパラメーターの設定ができる：

- ☐ スタティックIPアドレス-デフォルト192.168.1.230
- ☐ サブネットマスク-デフォルト255.255.255.0
- ☐ ゲートウェイ-デフォルト192.168.1.1
- ☐ DNS-デフォルト192.168.1.1、ゲートウェイと同じ
- ☐ MAC-ドライバーがソフトウェアで設定し、変更出来る。本開発ボードはデフォルト08:90:90:90:90:90

「Save」ボタンをクリックすれば、修正内容を保存し、直ちに有効になる。再起動しても、修正内容は残される。この設定と関連するパラメーター設定ファイルは/etc/eth0-setting。

説明：/etc/eth0-setting のパラメーターファイルはシステムインストール後は存在しないが、「Save」をクリックし自動的に作成される。また、ターミナルから ifconfig で IP アドレスを変更した場合はこの設定ファイルには影響がない。

4.1.10 WiFi 無線 LAN 設定

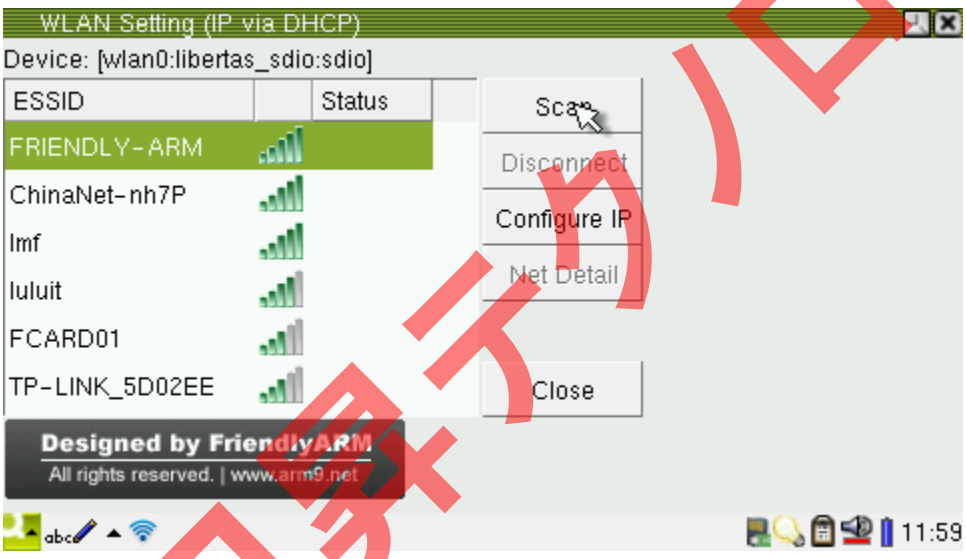
Linux での WiFi 無線 LAN の設定を説明する。Tiny4412 は市販の多くの USB WiFi 無線 LAN をサポートする。

4.1.10.1 WiFi 無線 LAN 設定アプリを起動

「FriendlyARM」タブの「Wireless Lan Setting」をクリックする：

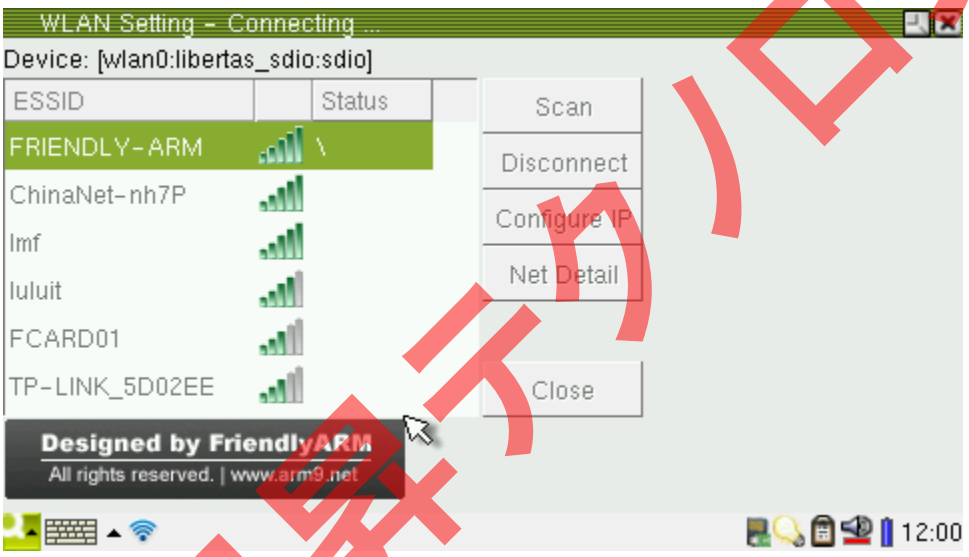
4.1.10.2 無線 AP 検索及び接続

無線 LAN 設定アプリを起動時、無線 AP を自動的に検索し、以下の図のように AP の SSID 及び電波信号レベルが表示される。無線 AP が見つからない場合、「Scan」をクリックし再検索する。

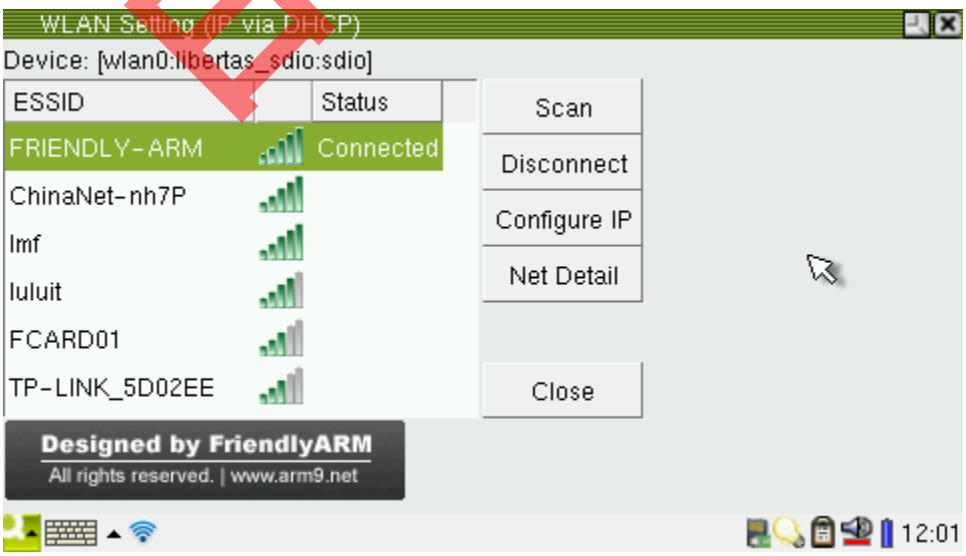


無線 AP が見つかれば、WiFi を接続する場合、リストに出る該当 ESSID をクリックし以下の画面が出てきて、無線 AP のパスワード入力が要求される：

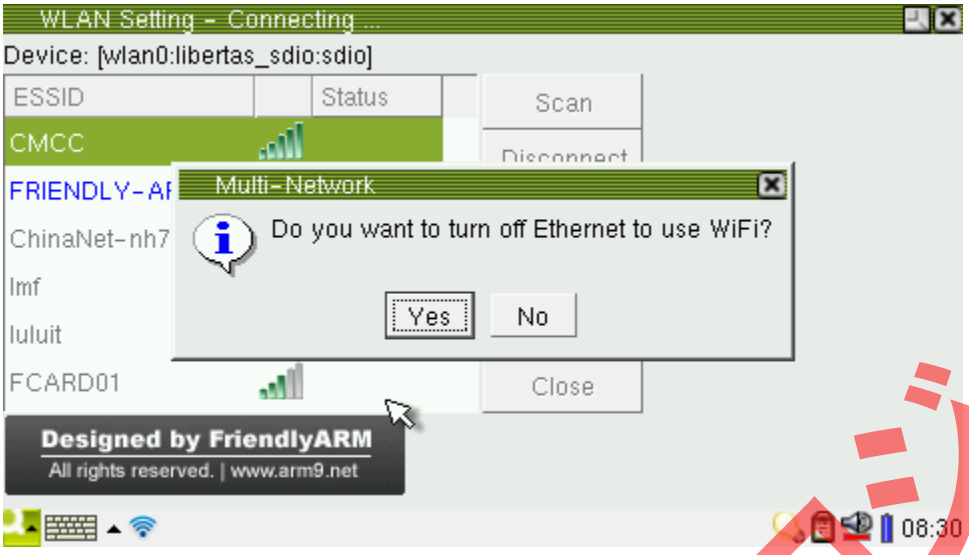
パスワードを入力してから（パスワードのない場合は空きのまま）「Connect」をクリックし、下図の画面に表示される：



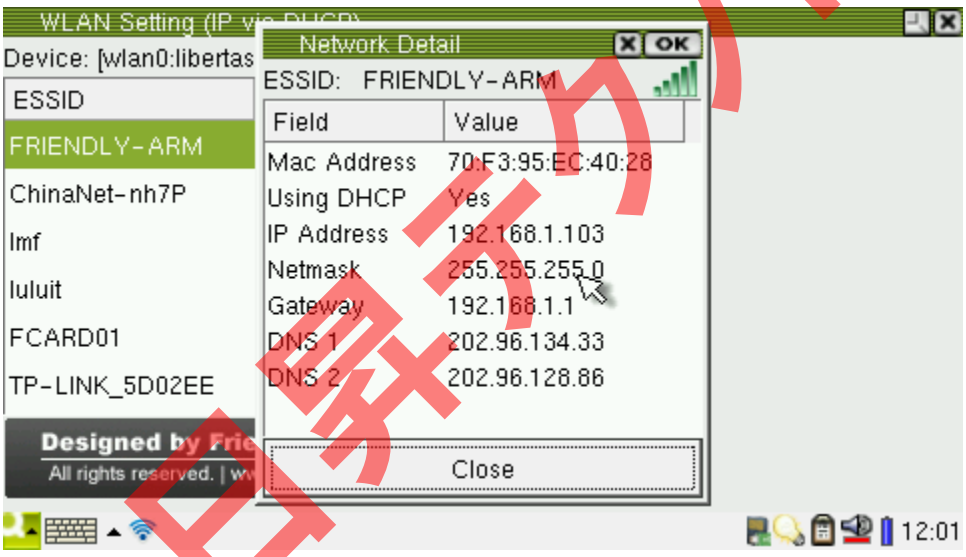
接続が完成したら、接続した無線 AP の後ろに Connected が出てくる：



イーサネットを起動している場合、イーサネットを切るかどうかを確認する画面が出てくる。「Yes」をクリックすると、イーサネットを切り WiFi 無線 LAN に接続する（「No」をクリックする場合、イーサネットが優先可能性がある）。イーサネットに戻る場合、ネットワーク設定アプリを起動し設定して「Save」をクリックする或いはターミナルに ifconfig eth0 up を入力して起動する。



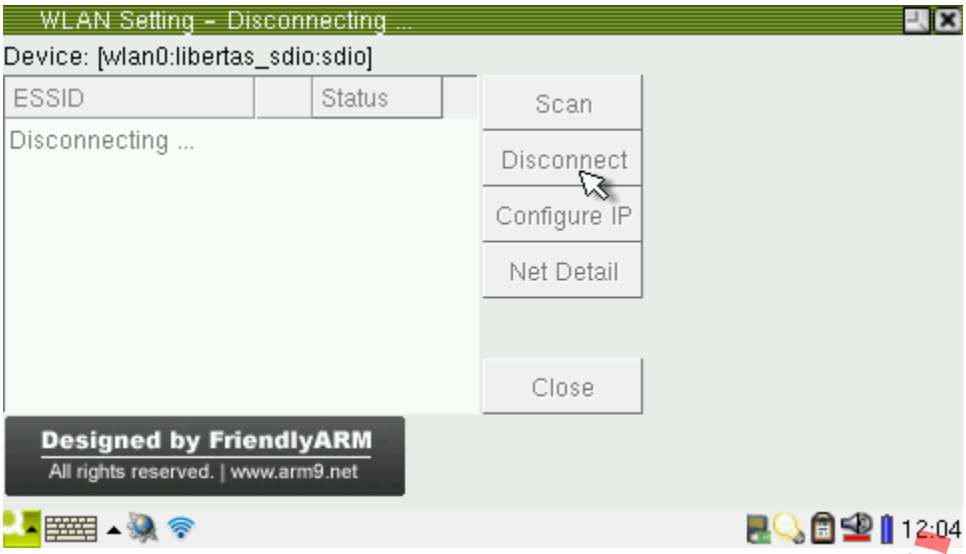
「Net Detail」をクリックすれば、ネットワークの詳しい情報（IP アドレス、DNS など）が見られる：



接続が成功したら、無線 LAN 設定アプリメイン画面に「Close」をクリックすればから下図のように最小化に出来る。WiFi アイコンをクリックすると、無線 LAN 設定アプリメイン画面に戻れる。

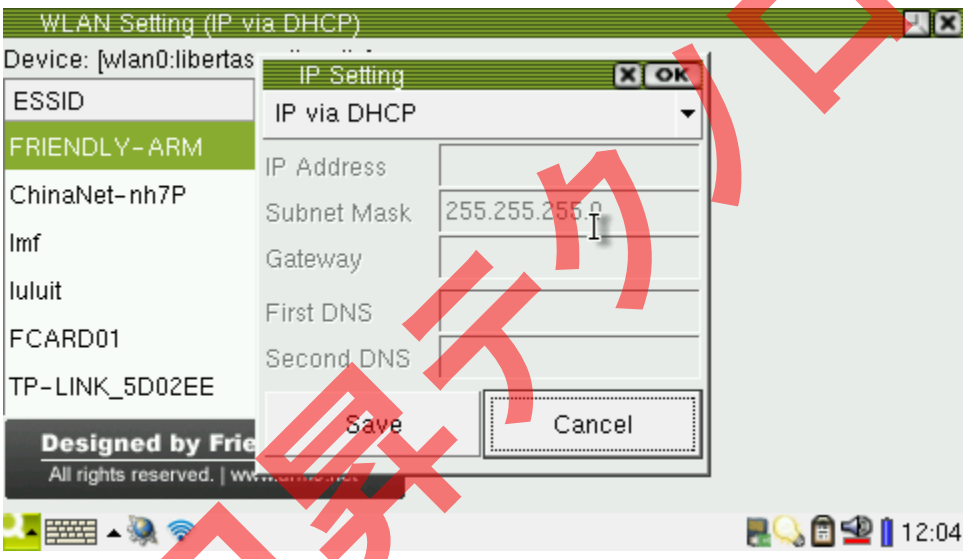
4.1.10.3 無線 LAN を切る

無線 LAN 設定アプリメイン画面に「Disconnect」をクリックすらば、下図のように無線 LAN を切ることができる：

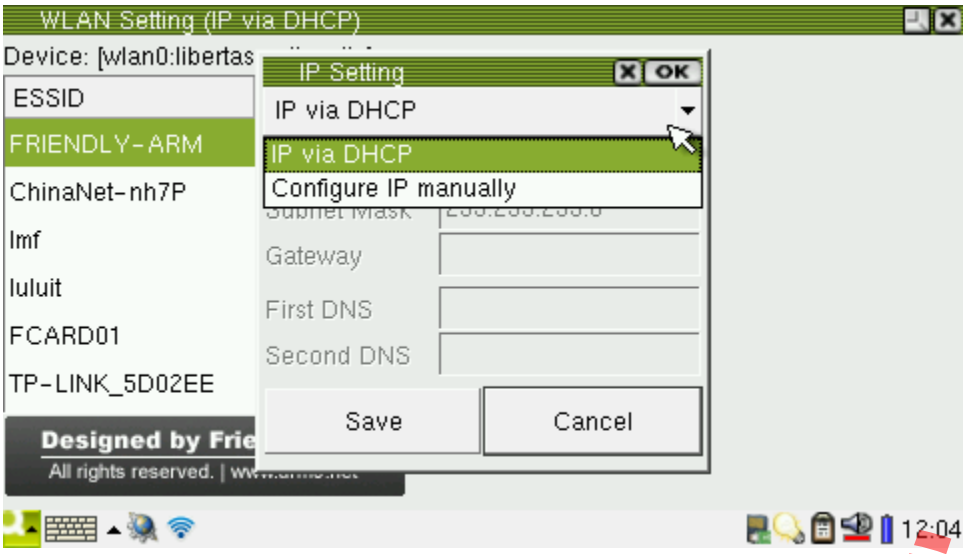


4.1.10.4 IP アドレスの設定

無線 LAN 設定アプリメイン画面に「Configure IP」をクリックすれば、下図のように IP 設定画面に入る：



画面のプルダウンリストをクリックし、DHCP（IP via DHCP）または手動 IP アドレス設定(Configure IP manually)という二つ方法が選択できる：



設定完了後、「Save」をクリックし設定結果を保存する。

4.1.11 WiFi AP の使用

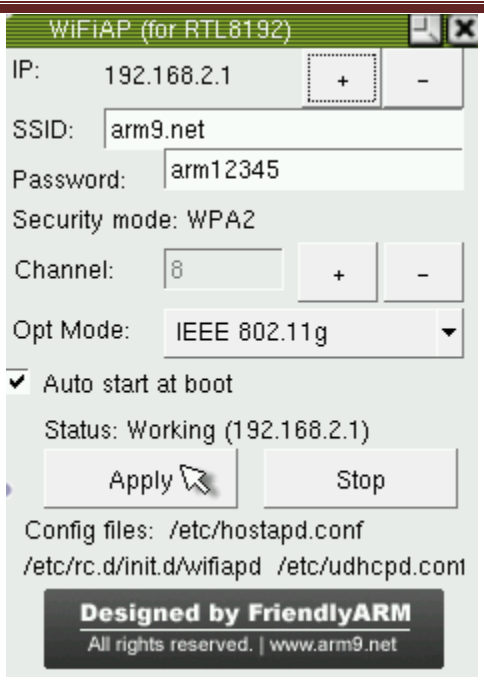
本開発ボードの WiFi AP は開発ボードに無線 LAN 接続ポイント（Access Point）を提供する機能である。他の設備（例えば携帯、ノートパソコン）は WiFi で開発ボードに接続する（他のルーターを使用する必要がない）。複数の設備の同時接続をサポートする。設備の間にローカルの LAN が形成され、相互通信することができる。

該当性能を実現するには RT8192 WiFi モジュールが必要。

注意事項：まだルーター性能が実現されていないため、インターネットにアクセスできない。

WiFi AP の設定と起動を行う場合、「FriendlyARM」に下図の WiFi AP アイコンが見つかる：

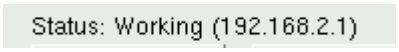
クリックすれば、WiFi AP 設定アプリを起動できる。画面は下図の通り：



設定できる項目が下記のグラフに定義する：

設定できる項目	性能
IP	IP アドレスのセグメントを設定する、範囲は 192.168.2.1～192.168.254.1 である。設備が開発ボードに接続した場合、自動的にこのセグメントで IP アドレスを配る。例えば、もし開発ボードの IP アドレスは 192.168.2.1 であれば、携帯が開発ボードに接続する場合、携帯の IP アドレスは 192.168.2.2 である。 IP アドレスは LAN の他のルーターと重複しないように注意してください。
SSID	無線 LAN 接続ポイントの名称。
Password	パスワード、最少 8 桁、WPA2 を使用。
Channel	デフォルト 8、もし所属する区域にはルーターがあれば、重複しないように注意してください。
Opt Mode	WiFi プロトコル、デフォルト 802.11 g。
Auto start at boot	次回起動する時自動的に WiFiAP モードを入るかどうかを選択する。

設定が完成後、「Apply」をクリックし設定を保存し、「Start」をクリックし WiFiAP モードを起動する。下図の画面の状態は「Working (192.168.2.1)」で、WiFiAP モードに入ったと表示し、開発ボードの IP アドレスは 192.168.2.1：



WiFi で開発ボードが見つかり、SSID を arm9.net と設定したため、WiFi リストの arm9.net をクリックし、パスワードを入力してから接続できる（接続できない場合は接続し直してください）：

接続ができたら、携帯が獲得した IP アドレスなどは下図のとおりである：

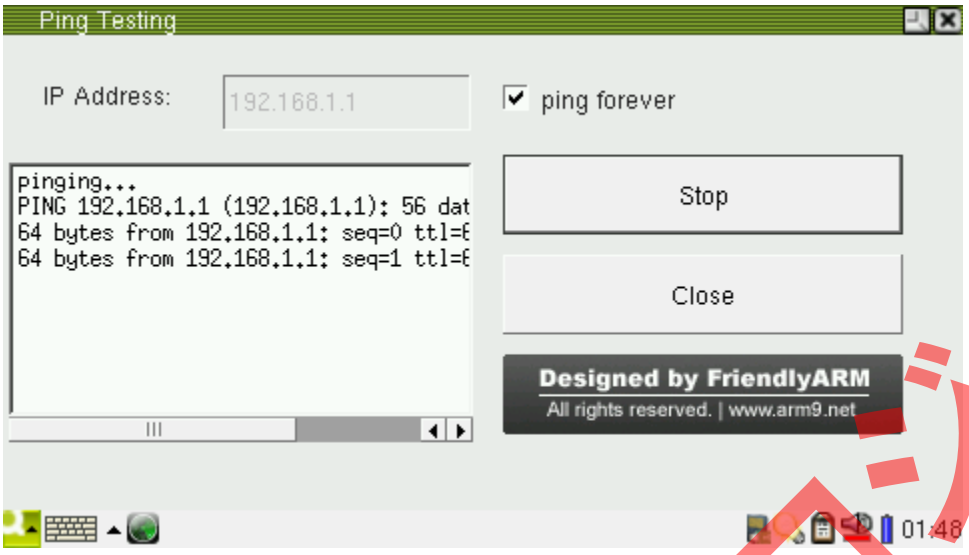
開発ボードの IP アドレスは 192.168.2.1 で、携帯ブラウザーで http://192.168.2.1 を入力し、下図のように開発ボードの Web Server が発表したネットサイトにアクセスできる：

WiFiAP Setting 設定アプリは設定のインフォメーションを下記の設定ファイルに保存し、手動で変更することができる：

設定ファイル	作用
/etc/hostapd.conf	WiFi 無線 LAN の関連パラメーター設定
/etc/udhcpd.conf	DHCP のパラメーター設定、他の設備に配る IP アドレス、DNS などのインフォメーションを決める
/etc/rc.d/init.d/wifiapd	WiPiAP の開閉、開発ボードの IP アドレス設定できる
/etc/init.d/rcS	携帯が起動すると自動的に WiFiAP モードに入る場合、当ファイルに /etc/rc.d/init.wifiapd start を入力する

4.1.12 Ping テスト

LAN ケーブルを接続し、有効な Gateway、DNS など設定した後 Ping テストでネットが有効かどうかテストできる。「FriendlyARM」タブの「Ping テスト」アイコンをクリックし、下図のような画面が出てくる：



注意事項：ドメイン名を Ping テストする場合、有効なゲートウェイと DNS の設定が必要となっている。更に、ネットがインターネットに接続できるように確認してください。

Ping テスト画面を閉じるには、先ずは「Stop」をクリックし Ping テストを中止する。

4.1.13 KonquerorWeb ブラウザー

「FriendlyARM」タブの「Web ブラウザー」をクリックし、画面の下のソフトキーボードをタッチし、アドレス欄に URL を入力して「Ret」をクリックすると、ウェブサイトにアクセスできる。

説明：本開発ボードの Web ブラウザーは Konqueror/Embedded で、ソースコードを公開するブラウザーである。

4.1.14 WebKit に基づく Qt4Web ブラウザーArora

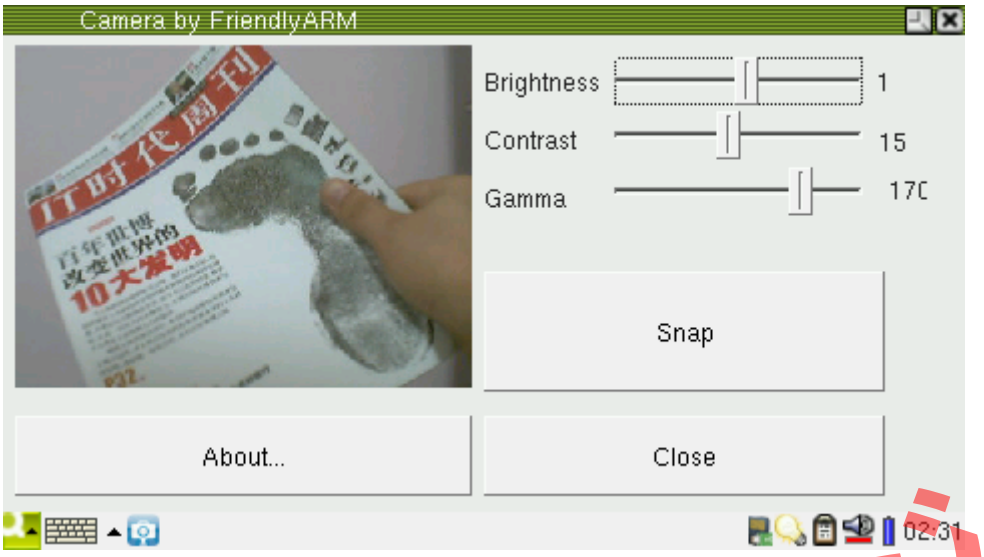
Arora はオーペンソースの WebKit である。開発ボードに移植し、WebKit ブラウザーアイコンのクリックで起動できる。項目アドレス：https://code.google.com/p/arora/。

Arora のデフォルトホームページはユーザーの設定により変更できる（/root/qt4 ディレクトリーのバックアップを介する）。

4.1.15　USB カメラによる撮影

カメラを開発ボードの USB　Host ポートに挿入し、FriendlyARM プログラムグルップより USB カメラを起動する。プレービューが表示される。[snap]のクリックで撮影できる。写真は”ドキュメントグループ”に保存される。

プログラムは明るさ、コントラスト、ガンマ値の設定ができる。



※全ての USB カメラをサポートする保証はできません。

4.1.16　3G ネットワークの利用

省略

4.1.16.1　3G ネットワークのオートダイヤル設定

省略

4.1.16.2　USB 3G カードの型番一覧

開発ボードがサポートする USB 3G カードの型番一覧：

Huawei E169 (CDMA2000)

Huawei E261 (WCDMA)
Huawei E1750/E1550 (WCDMA)
ZTE AC581 (CDMA2000)
ZTE AC8710 (CDMA2000)
ZTE MU351 (TD-SCDMA)

ZTE 6535-Z
ZTE AC2710 (EVDO)
ZTE AC2726
ZTE K3520-Z
ZTE K3565
ZTE MF110 (Variant)
ZTE MF112
ZTE MF620 (aka "Onda MH600HS")
ZTE MF622 (aka "Onda MDC502HS")
ZTE MF628
ZTE MF638 (aka "Onda MDC525UP")
ZTE WCDMA Stick from BNSL
HuaXing E600 (NXP Semiconductors "Dragonfly")
Huawei E1612
Huawei E1690
Huawei E180
Huawei E270+ (HSPA+ modem)
Huawei E630
Huawei EC168C (from Zantel)
Huawei K3765
Huawei K4505
Huawei R201
Huawei U7510 / U7517
Huawei U8110 (Android smartphone)
Onda MW833UP

A-Link 3GU
AT&T USBConnect Quicksilver (made by Option, HSO driver)
AVM Fritz!Wlan USB Stick N
Alcatel One Touch X020 (aka OT-X020, aka MBD-100HU, aka Nuton 3.5G), works with Emobile D11LC
Alcatel X200/X060S
Alcatel X220L, X215S
AnyDATA ADU-500A, ADU-510A, ADU-510L, ADU-520A

Atheros Wireless / Netgear WNDA3200
BSNL Capitel
BandLuxe C120
BandRich BandLuxe C170, BandLuxe C270
Beceem BCSM250
C-motech CGU-628 (aka "Franklin Wireless CGU-628A" aka "4G Systems XS Stick W12")
C-motech CHU-629S
C-motech D-50 (aka "CDU-680")
Cricket A600
EpiValley SEC-7089 (featured by Alegro and Starcomms / iZAP)
Franklin Wireless U210
Hummer DTM5731
InfoCert Business Key (SmartCard/Reader emulation)
Kyocera W06K CDMA modem
LG HDM-2100 (EVDO Rev.A USB modem)
LG L-05A
LG LDU-1900D EV-DO (Rev. A)
LG LUU-2100TI (aka AT&T USBConnect Turbo)
Motorola 802.11 bg WLAN (TER/GUSB3-E)
MyWave SW006 Sport Phone/Modem Combination
Nokia CS-10
Nokia CS-15
Novatel MC990D
Novatel U727 USB modem
Novatel U760 USB modem
Novatel Wireless Ovation MC950D HSUPA
ONDA MT505UP (most likely a ZTE model)
Olivetti Olicard 100 and others
Olivetti Olicard 145
Option GlobeSurfer Icon 7.2
Option GlobeSurfer Icon 7.2, new firmware (HSO driver)
Option GlobeTrotter EXPRESS 7.2 (aka "T-Mobile wnw Express II")

Option GlobeTrotter GT MAX 3.6 (aka "T-Mobile Web'n'walk Card Compact II")
Option GlobeTrotter HSUPA Modem (aka "T-Mobile Web'n'walk Card Compact III')
Option iCON 210
Option iCON 225 HSDPA
Philips TalkTalk (NXP Semiconductors "Dragonfly")
Rogers Rocket Stick (a Sony Ericsson device)
Royaltek Q110 - UNCONFIRMED!
ST Mobile Connect HSUPA USB Modem

Sagem F@ST 9520-35-GLR
Samsung GT-B3730
Samsung SGH-Z810 USB (with microSD card)
Samsung U209
Sierra Wireless AirCard 881U (most likely 880U too)
Sierra Wireless Compass 597
Siptune LM-75 ("LinuxModem")
Solomon S3Gm-660
Sony Ericsson MD300
Sony Ericsson MD400
Toshiba G450
UTStarcom UM175 (distributor "Alltel")
UTStarcom UM185E (distributor "Alltel")
Vertex Wireless 100 Series
Vodafone (Huawei) K4605
Vodafone (ZTE) K3805-Z
Vodafone MD950 (Wisue Technology)
Zydas ZD1211RW WLAN USB, Sphairon HomeLink 1202 (Variant 1)
Zydas ZD1211RW WLAN USB, Sphairon HomeLink 1202 (Variant 2)

4.1.17 LED テスト

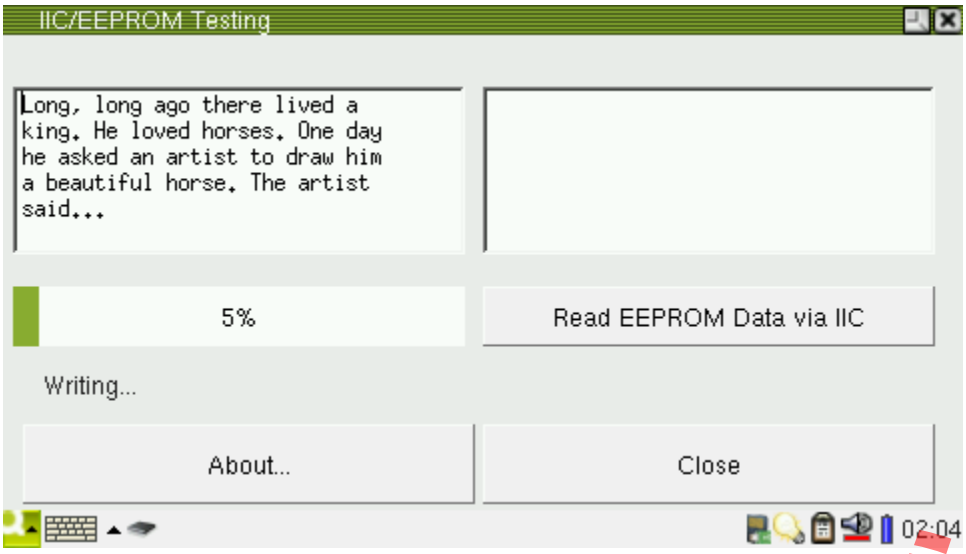
「FriendlyARM」タブの「LED テスト」をクリックし、下図の画面が出てくる：

起動した後、「Stop led-player」ボタンだけが有効となっている。システムが起動する時、デフォルトで led-player プログラムを働かせるためである。単独に一つの LED を制御する場合はこのプログラムを止める必要がある。

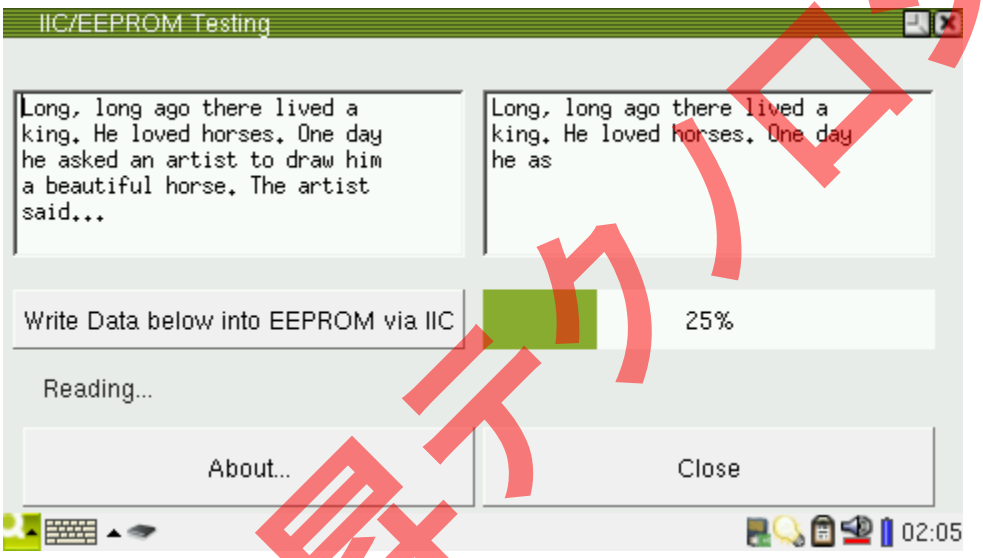
「Stop led-player」をクリックすると、下図のように他のボタンが有効になる。「LED テスト」画面を閉じると、また led-player プログラムが起動する。

4.1.18 EEPROM Write/Read テスト

「FriendlyARM」タブの「I2C-EEPROM」をクリックし、「ソフトキーボード」をタッチし書き込むスペースに何か入力してから、「Write Data below into EEPROM via IIC」ボタンをクリックするとボタンが進捗バーに変化し、下図のように書き込み進捗を表示する：

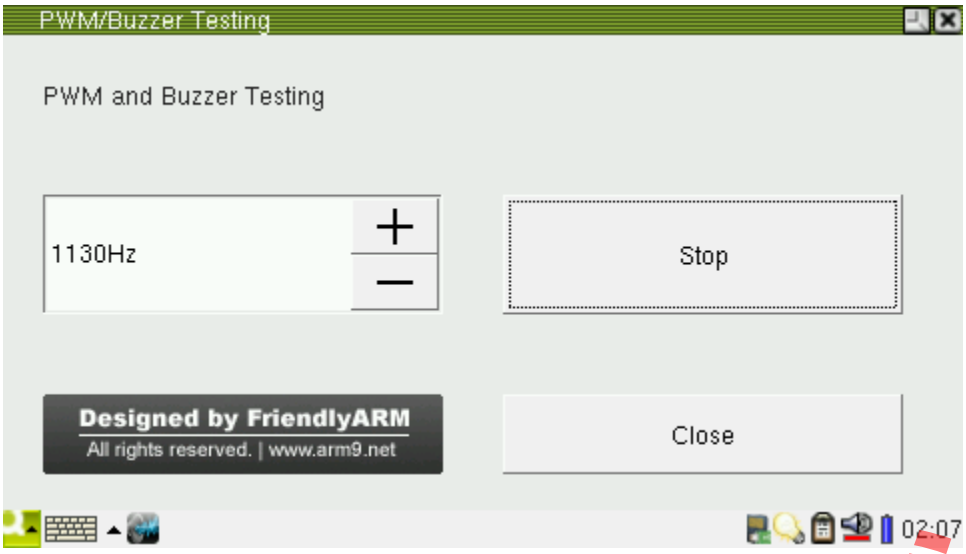


「Read EEPROM Data via IIC」をクリックすると、ボタンが進捗バーに変化し、下図のように読み出し進捗を表示する：



4.1.19 PWM ブザー

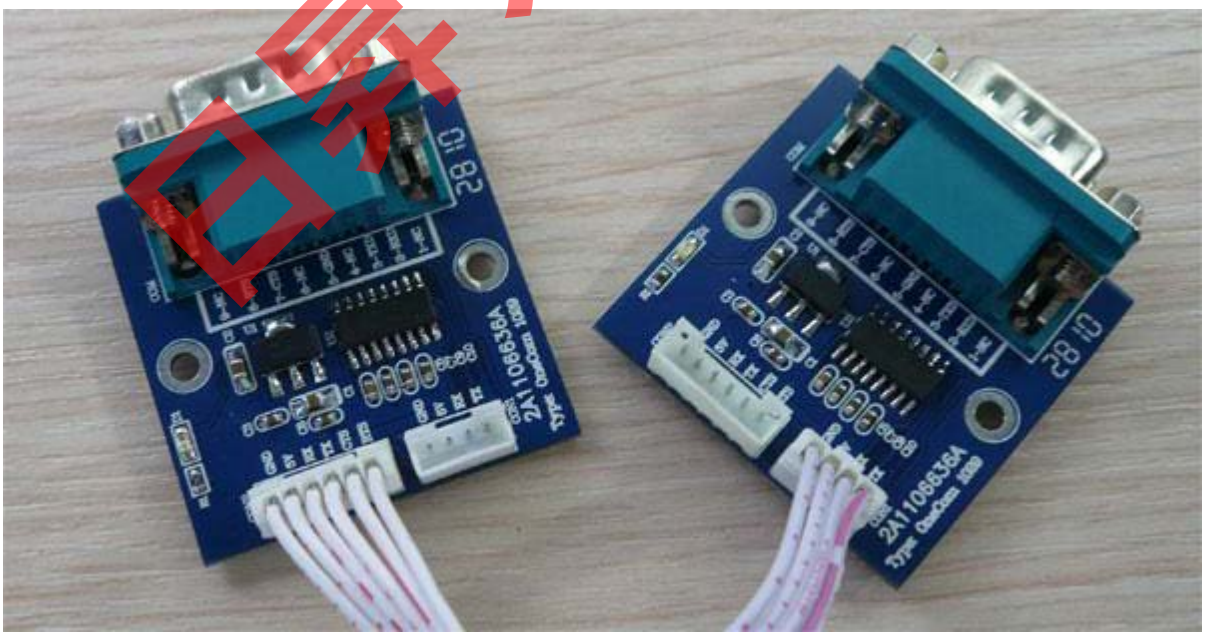
「FriendlyARM」タブの「PWM ブザー」をクリックしアプリを起動する。デフォルトの PWM 出力周波数は 1000Hz である。「Start」をクリックするとブザーから音が出てきて、“＋”或いは“－”で PWM 出力周波数を変更し、、ブザーの音が変化する。「Stop」クリックし PWM 出力を中止する。



4.1.20 シリアルポートアシスタント

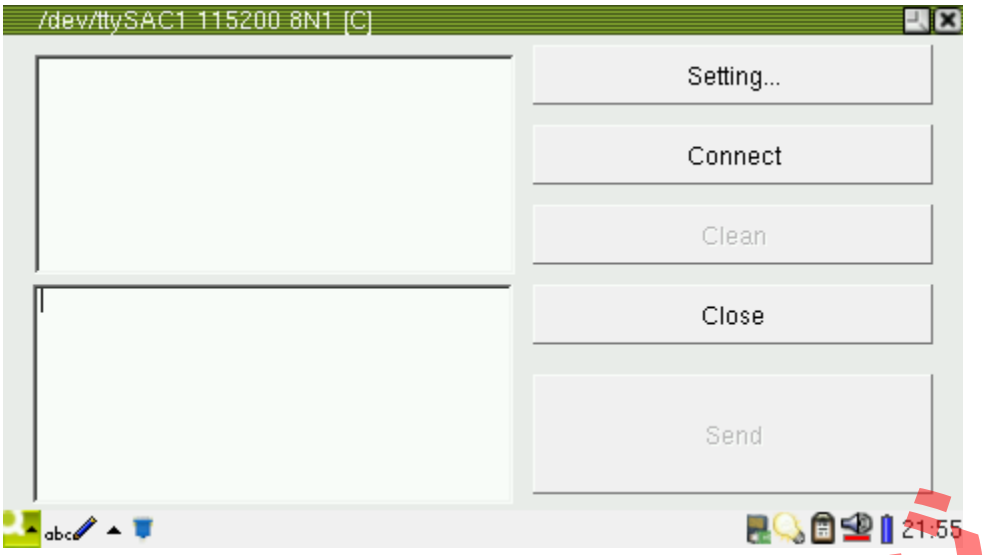
提示：アプリを起動する前にケーブルを接続してください。

- 4412にはUART0,1,2,3四つのシリアルポートがあり、UART0,1は4線式シリアルポートで、UART2,3は2線式シリアルポートである。
- 本開発ボードのUART0はRS232により転換され、COM0と対応している。付属されたクロスシリアルケーブルでPCと接続する（弊社は下図のように「OneCom」RS232転換モジュールを提供している、URL: http://www.csun.co.jp/SHOP/2011070702.html ）。PCに接続する時、ご使用になっているシリアルケーブル（クロスか直連か）に注意してください。
- 本アプリはUSB/RS232転換ケーブルもサポートする。USB/RS232変換ケーブルをボードのUSB Host口に挿し込み、開発ボードのシリアルポートを拡張できる。対応する設備名称は/dev/ttyUSB0,1,2,3などである。USB Hubを利用し、複数のUSB/RS232変換ケーブルを接続できる。



RS232 拡張ボードを開発ボードの CON2/3/4 に繋がり、クロスケーブルで PC と接続する。

「FriendlyARM」タブの「Serial Port Assistant」をクリックし、下図の画面が出てくる：

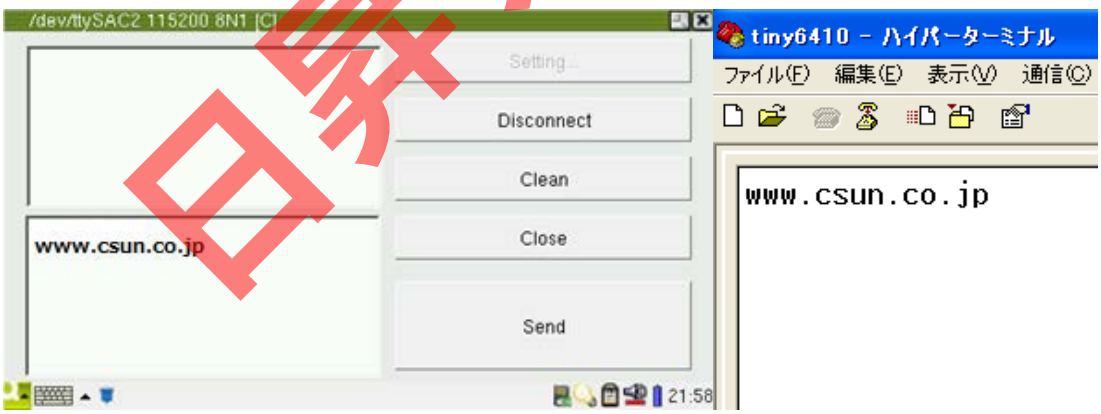


タイトルからデフォルトの設定は「ttySAC1 115200 8N1 [C]」と分かる。下記の設定を表している：

- シリアルポート設備：/dev/ttySAC1、CPU の UART1 に対応する
- ボーレート：115200
- データビット：8
- フロー制御：なし
- ストップビット：1
- [C]：キャラクターモード、[H]の場合は 16 進モード

上図には二つのエディタエリアがあり、上のエディタエリアは受信したデータを表示し、編集することができない。下のエディタエリアは USB キーボード或いは Qtopia のソフトキーボードでデータを入力できる。

「Connect」ボタンをクリックし、/dev/ttySAC1 をオープンする。下のエディタエリアに入力して「Send」をクリックし、接続しているシリアルポートにデータの送信ができる。下図は Windows ハイパーターミナルが受信したデータの画面である（ハイパーターミナルに対応するシリアルポートも 115200 8N1 に設定する必要がある）。



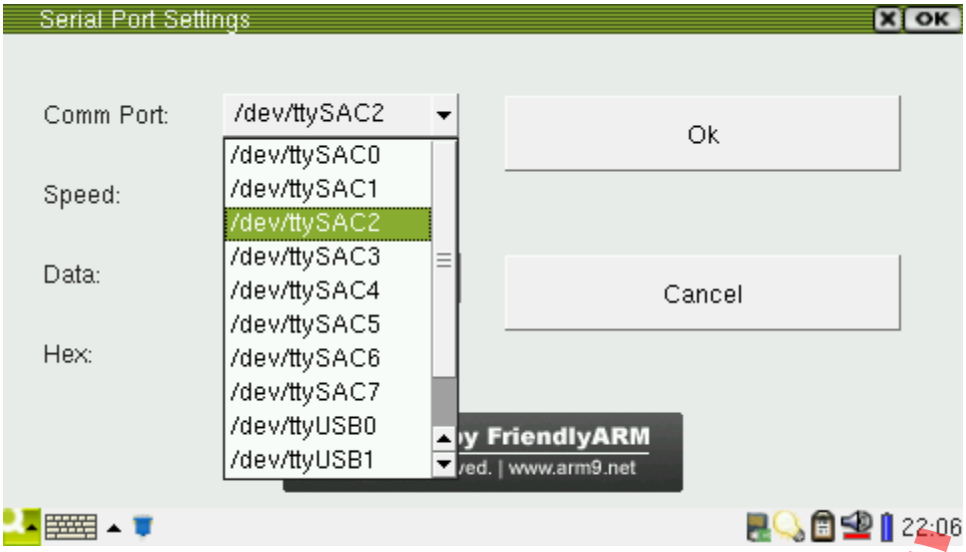
「Disconnect」クリックし接続を切る。「Setting」をクリックし、設定画面に入る。ここでは通常に使われているシリアルポート設定パラメーターをリストしている。

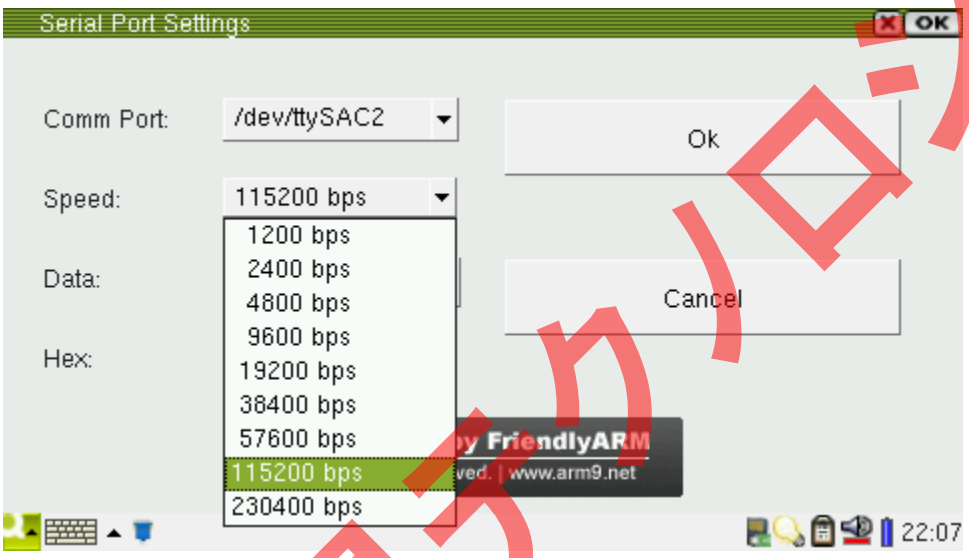
Comm Port：/dev/ttySAC0,1,2,3 あるいは USB 変換ケーブル利用時の/dev/ttyUSB0,1,2,3 が選択できる。

Speed：ボーレートが選択できる。

Data：8 ビット/7 ビットが選択でき、通常は 8 ビット。

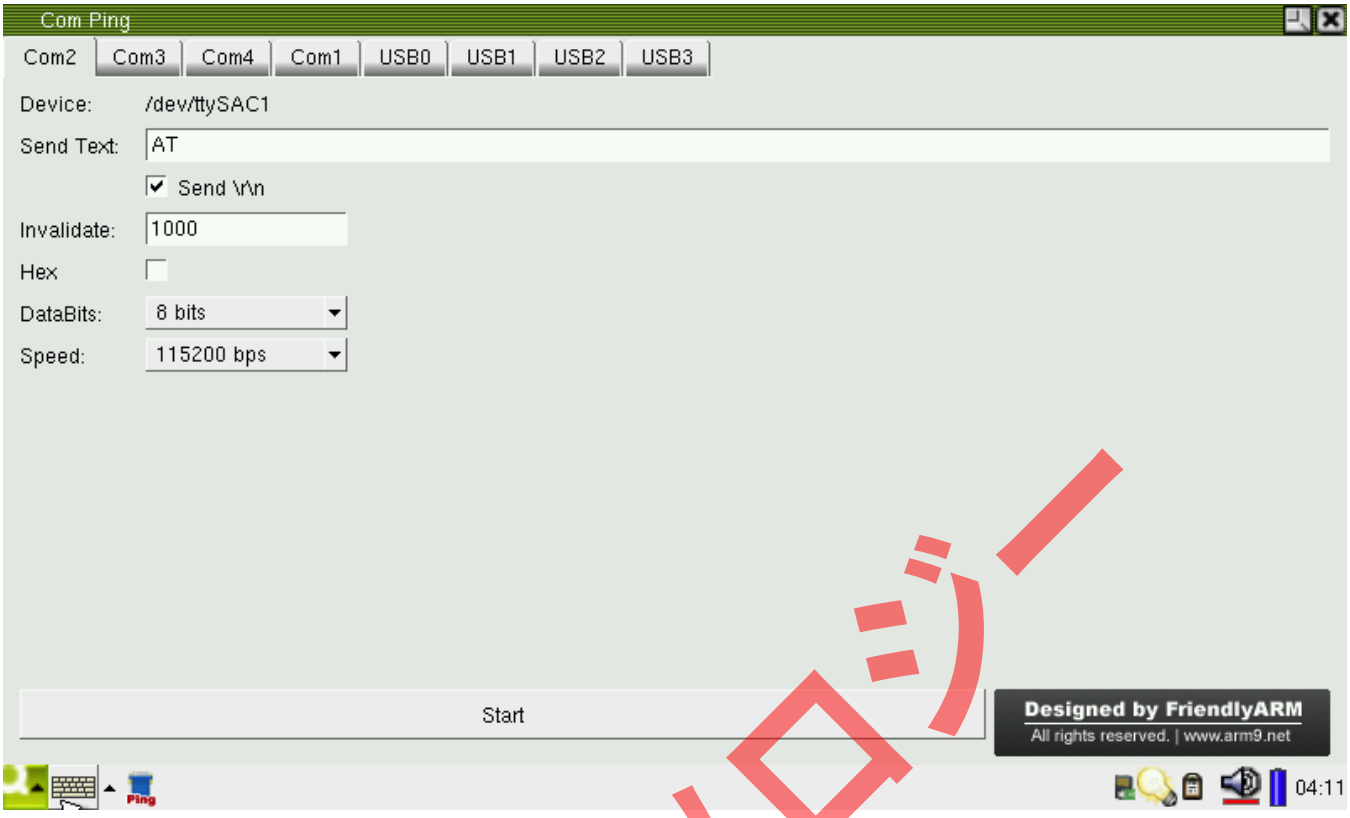
Hex: 16 進入力或いはデータを表示する。





4.1.21 Com Ping テスト

Com Ping テストは同時に複数のシリアルポートをテストでき、開発ボードの四つのシリアルポートと USB 転換ケーブルをサポートしている。



Com Ping テストの使用方法はシリアルポートアシスタントとは殆ど同じ、違う所は下記の通り：

!)「Send Text」で送信データを指定

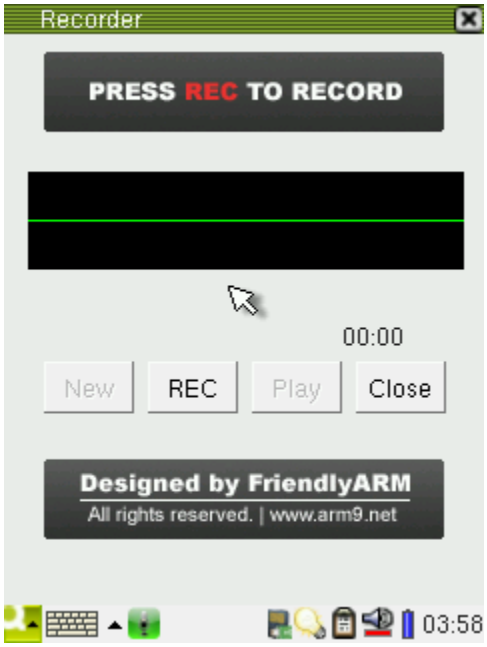
2)「Inwalidate」は時間の間隔を示している、この時間間隔毎に Send Text の内容をシリアルポートに送信する。

3) シリアルポートアシスタントと同じ、16 進で送信できる。

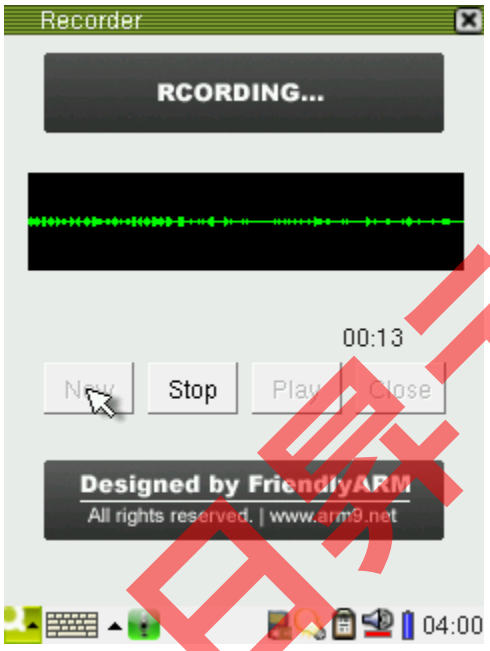
設定完成後、「Start」をクリックし自動的に通信が始まり、複数のシリアルポートの同時通信ができる。なお、設定パラメーターは「Start」をクリックする時自動的に保存され、次回アプリを起動する時設定する必要がない。

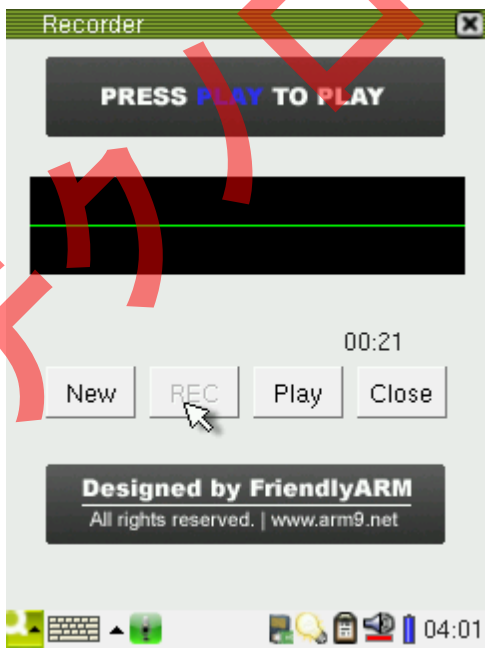
4.1.22 レコーダー

「FriendlyARM」タブの「Recorder」をクリックし、下図の画面が出てくる：

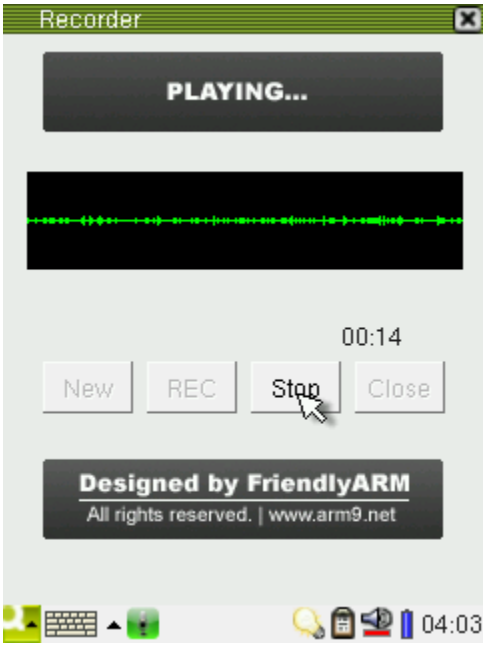


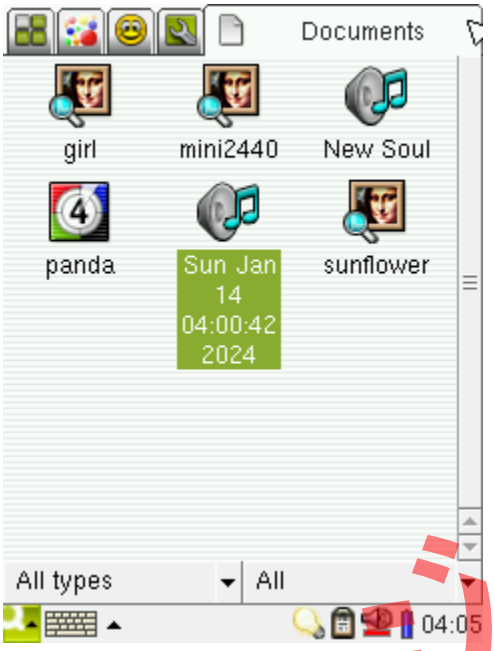
画面提示により「REC」をクリックすると録音が開始する。マイクを通じて話すと録音の波形がみられる。「STOP」をクリックし録音を中止する。画面は下図の通り：





Play をクリックし録音されたファイルを再生できる。録音されたファイルは自動的に“WAV”フォーマットで「Documents」フォルダに保存される。





4.1.23 LCD テスト

本アプリは LCD に不良ピクセルがあるかどうかをテストする。最多三つの不良ビクセルが許可される。「FriendlyARM」タブの「LCD テスト」をクリックし、画面は下図の通り：

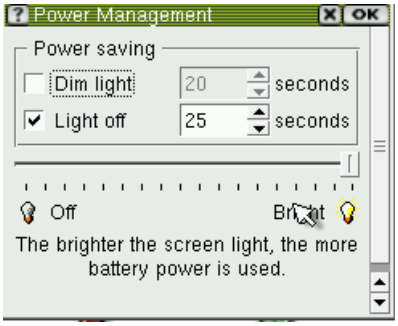
本アプリはオートモードとマニュアルモードがある：

Auto-loop はオートモードで、実行すると赤、黄、白、青、ブルー、緑、ピンク、黒、８色順次にフルスクリーンで表示する。途中にスクリーンをタッチするとテスト画面に戻る。

Manual-control はマニュアルモードで、実行するとスクリーンをタッチする度に色が変更し、赤、黄、白、青、ブルー、緑、ピンク、黒、８色順次に一回表示したらテスト画面に戻る。

4.1.24 バックライト調節

「設定」タブの「電源管理」をクリックし、下図の画面が出てくる：



デフォルトの消灯時間は 25 秒、右側の調節ボタンをクリックし、他の間隔を選択できる。「Light off」をチェックしない場合、バックライトはオン状態を続ける。

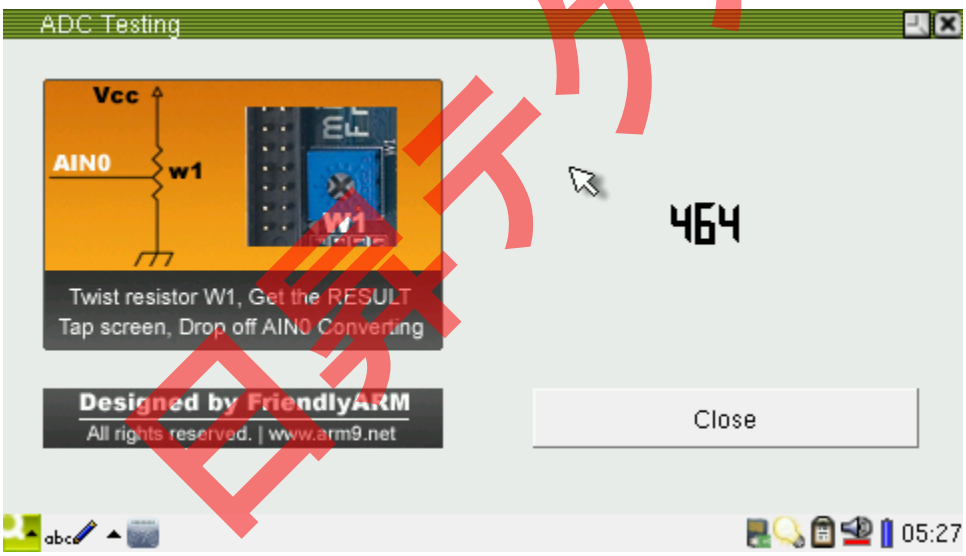
またバックライト調節性能があり、下のスライドアイコンを移動すると、バックライトの明るさが調節できる。

4.1.25 A/D 変換

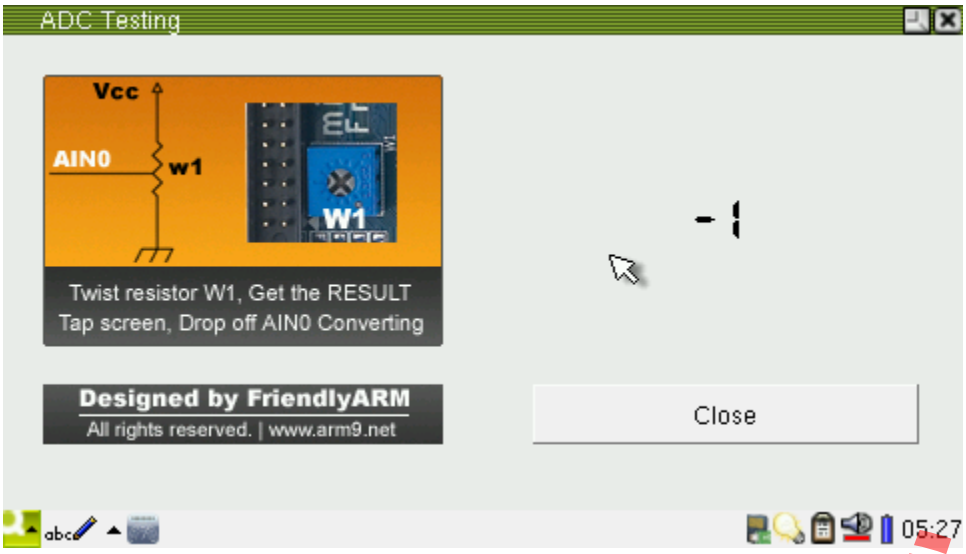
CPU 内部には 8 チャネルの A/D 転換を搭載しているが、コンバータは一つしかない。一般的に AIN4、AIN5、AIN6、AIN7 はタッチパネルの YM、YP、XM、XP チャネルとして利用される。本開発ボードは AIN1-3 を CON6 に引き出し、AIN0 は可変抵抗 W1 と繋っている。

「FriendlyARM」タブの「A/D Convert」をクリックする。

可変抵抗 W1 を調整すると、下図のように転換結果が 0～1024 の間に変化する：

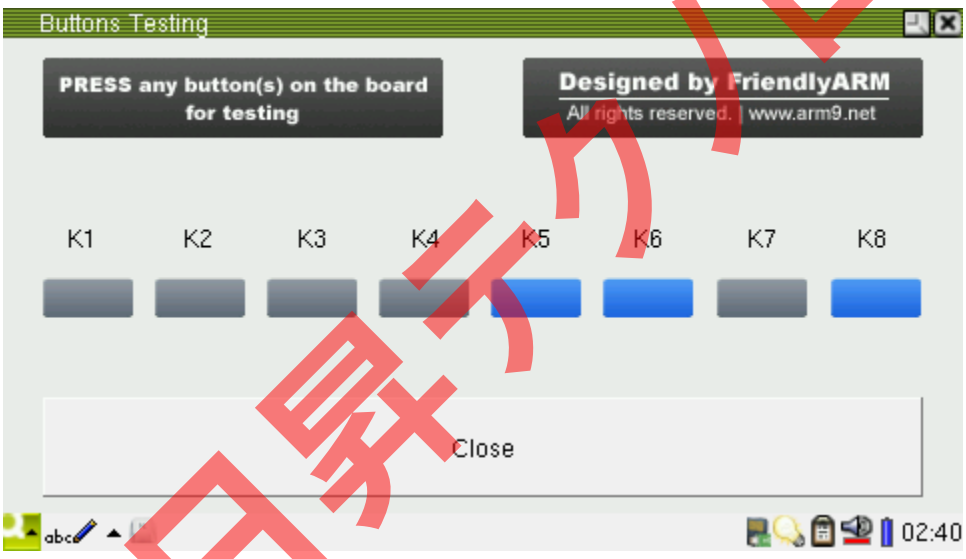


タッチパネルをタッチすると A/D 転換器はタッチパネルのチャネルを選択し転換結果は－1 となる。タッチペンが離れると転換器はまた AIN0 チャネルを選択する。



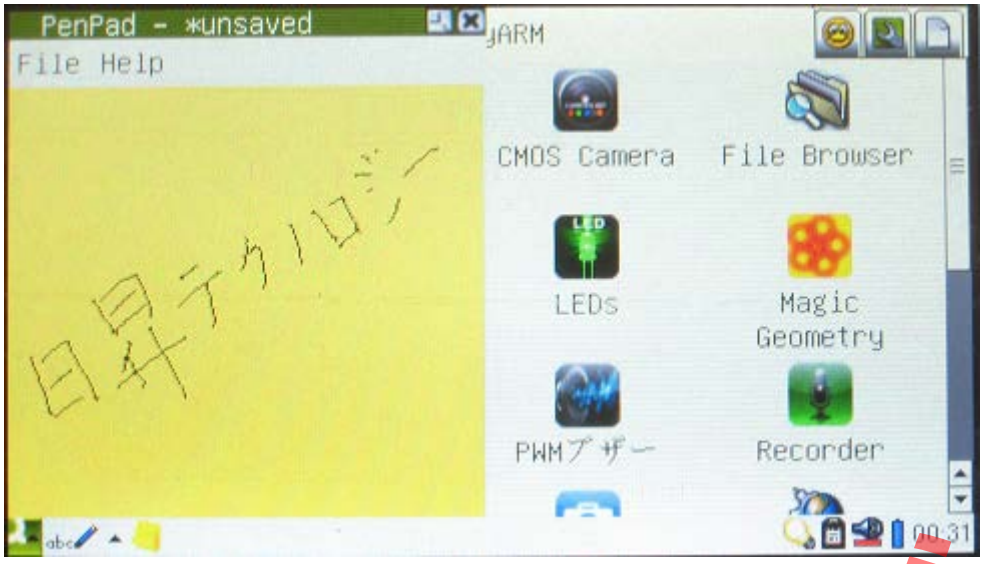
4.1.26 ボタンテスト

「FriendlyARM」タブの「ボタン」をクリックし、開発ボードの任意のボタンをタッチする（複数も OK）と、下図のように対応しているアイコンが青色に変化し、放すと灰色に復帰する。



4.1.27 タッチペンテスト

「FriendlyARM」タブの「Penpad」をクリックし、下図のように黄色のパッドが出てくる。ここにタッチペンで任意に書ける（ペンは黒色、1 pixel）。File->Save をクリックし png フォーマットで書いた内容が保存できる（保存位置は：/Documents/image/png/）。ファイル名は 001 から 999 まで 999 つのファイルが保存できる。

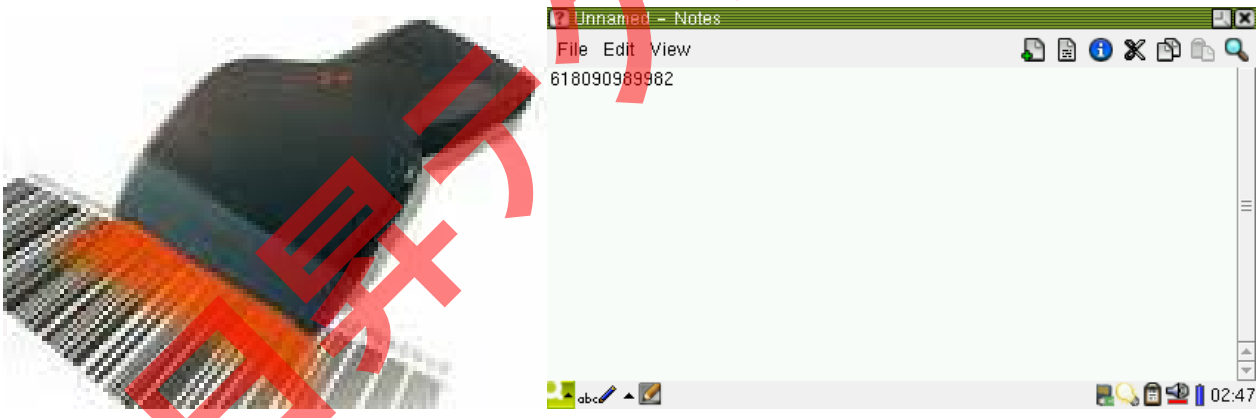


4.1.28 Barcode Scanner

本開発ボードは USB Barcode Scanner をサポートする。USB Barcode Scanner は実際に HID 設備で、USB キーボードに等しい。

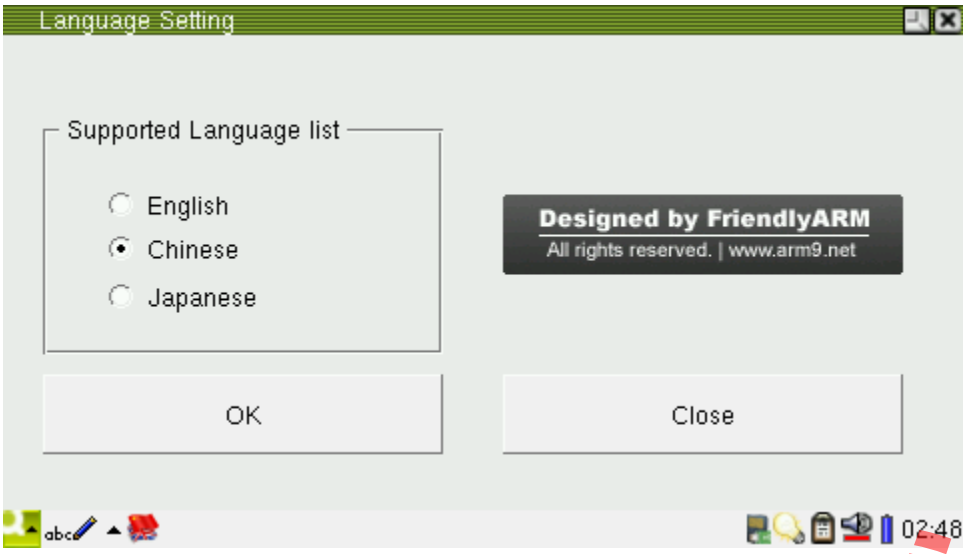
GUI 画面起動後 USB Barcode Scanner を挿入してください。ボードの電源を入れる前にに挿入したら使用できなくなる。

USB Barcode Scanner を挿し込み、「アプリケーション」タブの「エディター」をクリックし、バーコードをスキャンするとエディターにバーコードの数字が表示する。

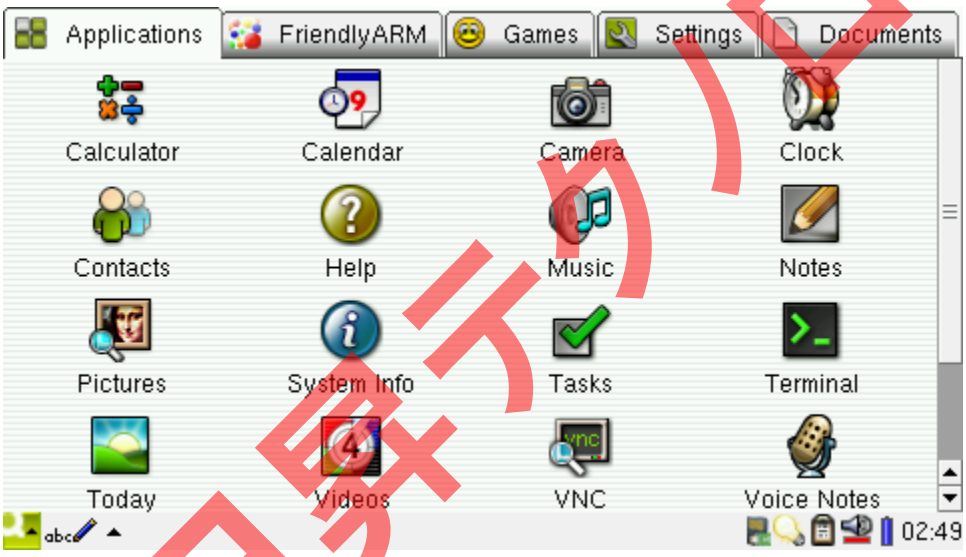


4.1.29 言語設定

「FriendlyARM」タブの「Language Setting」をクリックし、下図の画面が出てくる：



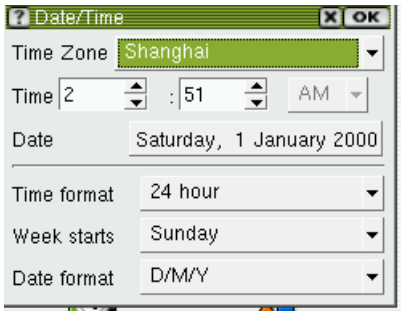
現在は英語、中国語、日本語三つの言語が設定できる。英語を設定する場合、「English」をチェックし、「OK」をクリックすると、提示が出てくる。「Yes」をクリックしたら、システムが再起動し、言語が英語になる。「No」をクリックすると、元の画面に戻る（中国語と日本語のバージョンはアプリの名前を翻訳するだけ)。言語変更後の画面は下図の通り：



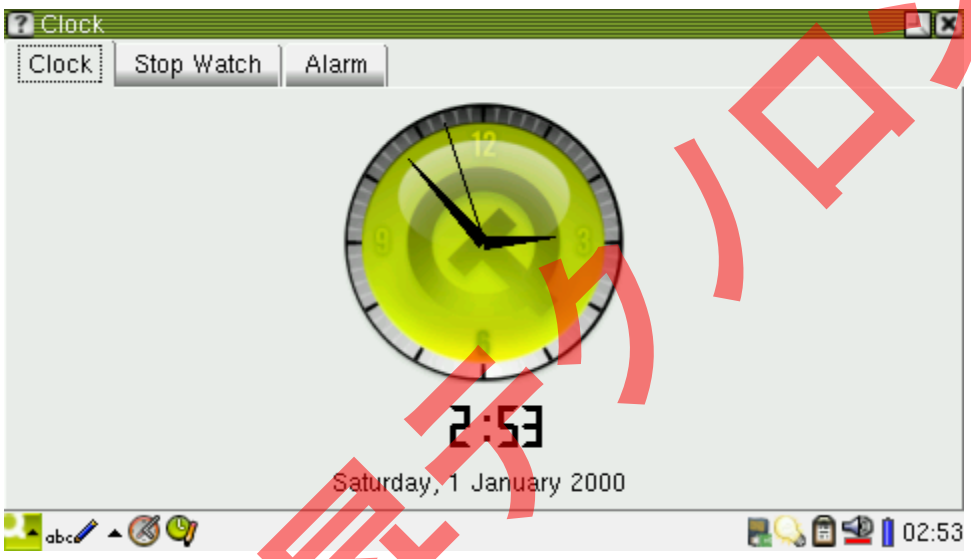
4.1.30 タイムゾーン、日付、時間、アラームの設定

出荷時の日付は正確ではないため、ユーザーから調整できる。また、RTC を提供しているので、調整した結果が保存できる。調整の手順は下記の通り：

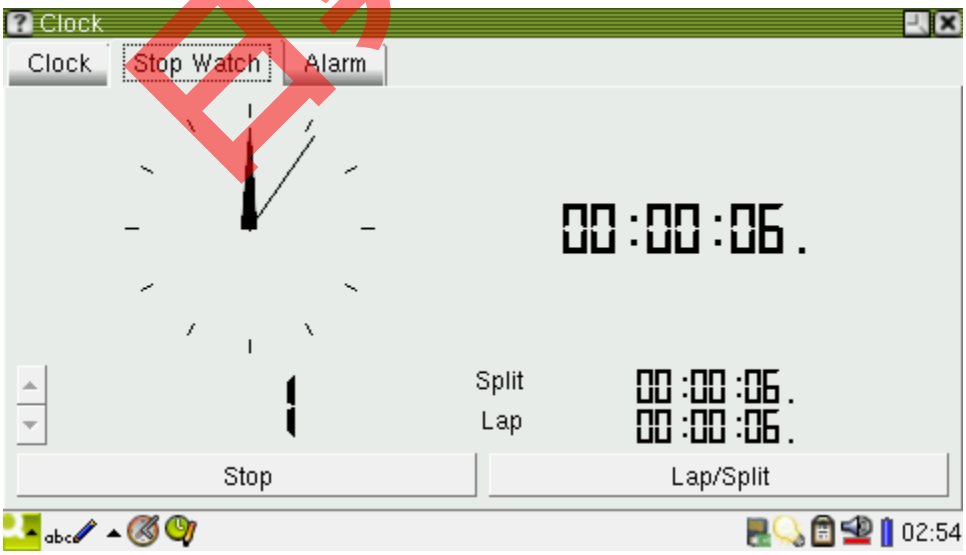
画面右下にあるタスクバーの時間の所をタッチすると設定メニューが出てくる。「Set time…」をクリックしてタイム設定画面が表示される。下図のようにタイムゾーン、日付、時間が設定できる。



メニューに「Clock…」をクリックすると、クロック画面が出てくる。



「Stop Watch」タブをクリックすると、ストップウォッチプログラムに入る。

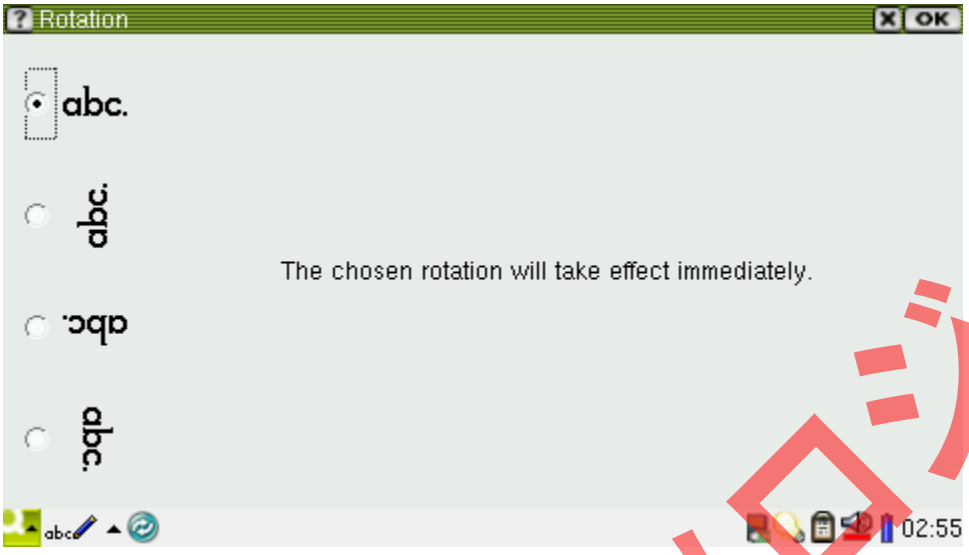


「Alarm」タブをクリックし、アラームを設定できる。設定時間になると、一分間音が鳴る。「OK」をク

リックし音が停止する。

4.1.31 スクリーンの回転

「設定」タブの「回転」をクリックし、相応の画面に入る。下図のように、四つの回転方向が選択できる：



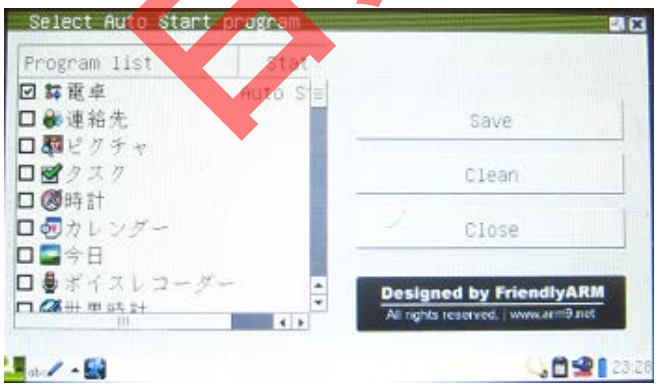
回転方向を選択し、「OK」をクリックすると効果が見られる。
Qtopia を再起動する必要の場合もある。
回転の実現は Qtopia の機能を利用しており、LCD のドライバーと関連していない。

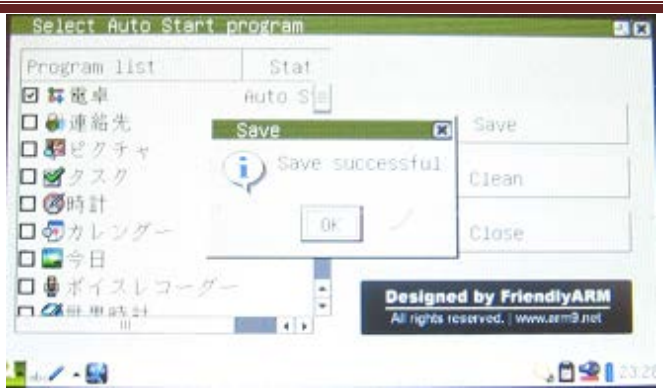
4.1.32 自動起動アプリの設定

Qtopia のアプリ或いはユーザーが開発したアプリをシステム起動時に自動的に動くように設定する機能である。

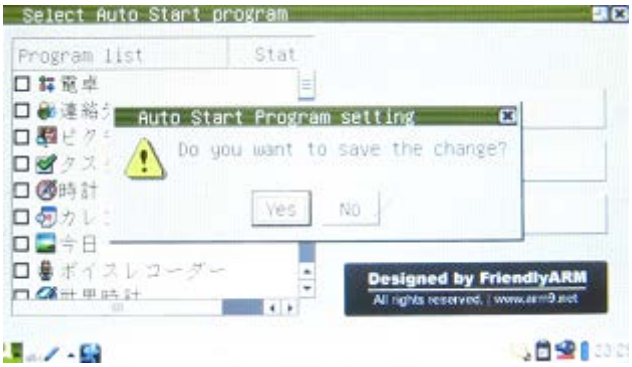
「FriendlyARM」タブの「Auto Start Setting」をクリックする

例えば「Program list」から電卓を選択すると、Status が Auto Start に変化する。「Save」クリックすると成功の提示が出てくる。このプログラムを閉じ、システムを再起動すると、電卓プログラムが起動される。





自動起動に設定したアプリを取り消す場合は「Clean」をクリックし、「Close」を選択し、確認画面が出たら、「Yes」をクリックすれば設定できる。



4.1.33 シャットダウンについて

「設定」タブの「シャットダウン」をクリックし、下図のように四つの選択肢がある：

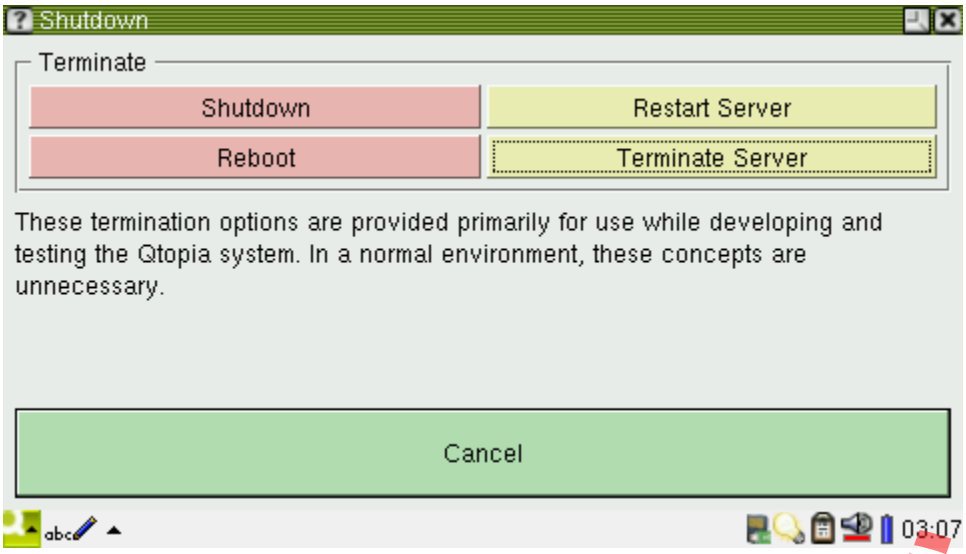
Shutdown：次々に全てのプログラムを閉じる。CPU は完全に動作しないため、システムの消費電力が最も低い状態である。

Reboot：“Hot”再起動。NOR FLASH モードであれば、次々に全てのプログラムを閉じ、再起動した後は Superboot のメニューモードになる。NAND FLASH モードであれば、次々に全てのプログラムを閉じ、再起動した後は Qtopia システムに入る。

注意事項：Reboot は Watchdog と完全に違う機能で、Watchdog は“Cold”スタートで次々に各プログラムを閉じるのではなく、直接にリセットする。

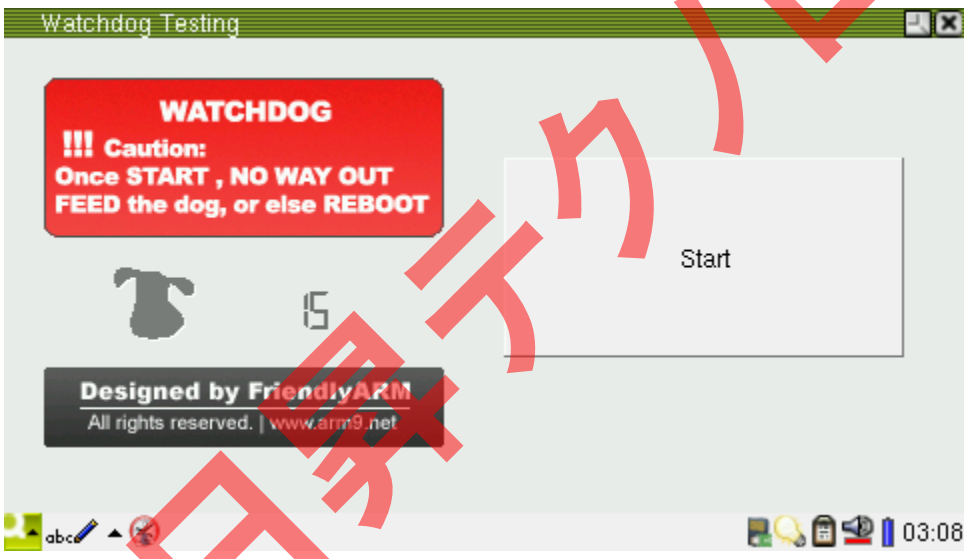
Restart Server：Qtopia GUI システムだけ再起動し、Linux システムに影響がない。

Terminates Server：Qtopia GUI システムを中止する。クリックすると、画面に表示しているのはメモリに残っているデータで、実際の画面ではない。

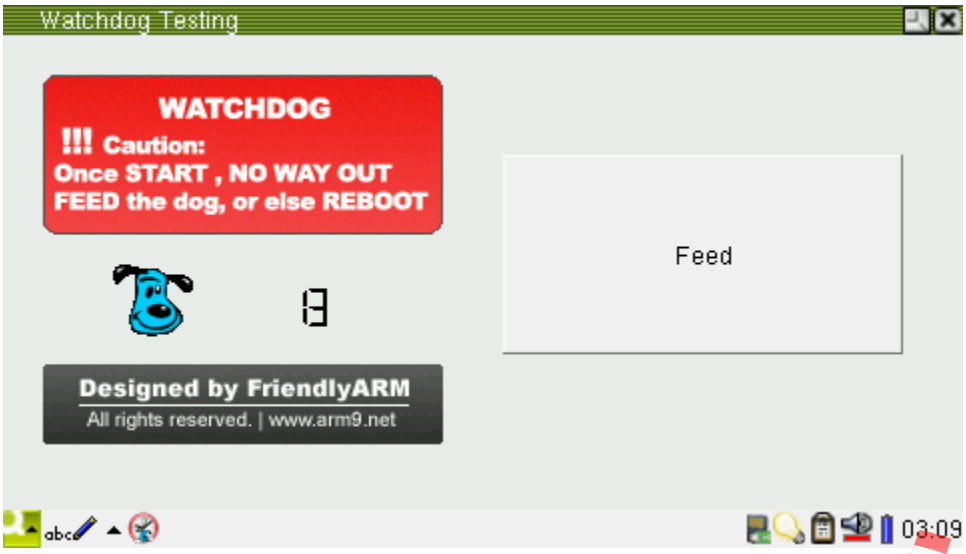


4.1.34 ウォッチドッグ

「FriendlyARM」タブの「ウォッチドッグ」をクリックし、下図の画面が出てくる：



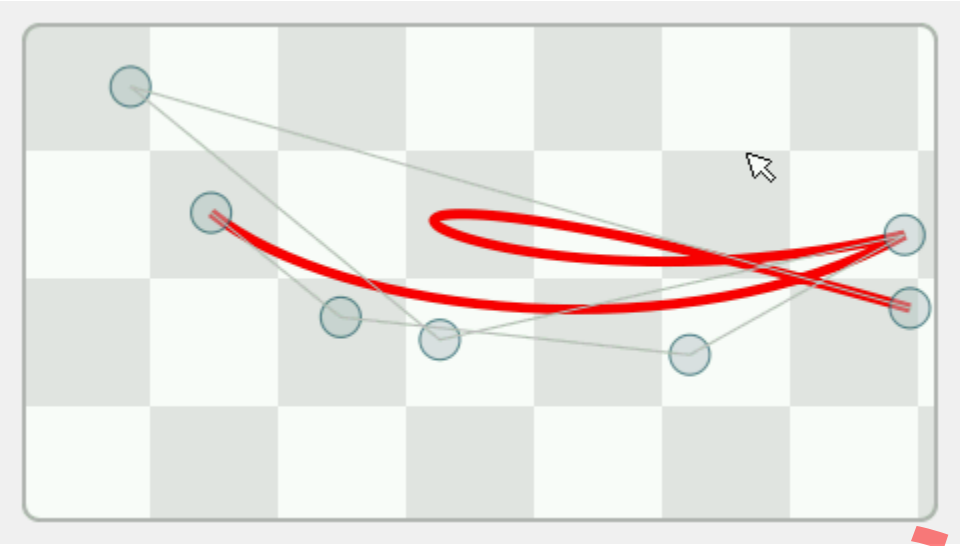
「Start」ボタンを押すと、停止できない。15 秒過ぎるとシステムが Reboot する。15 秒以内に「Feed」クリックすれば、下図のようにタイマーが再設定される。



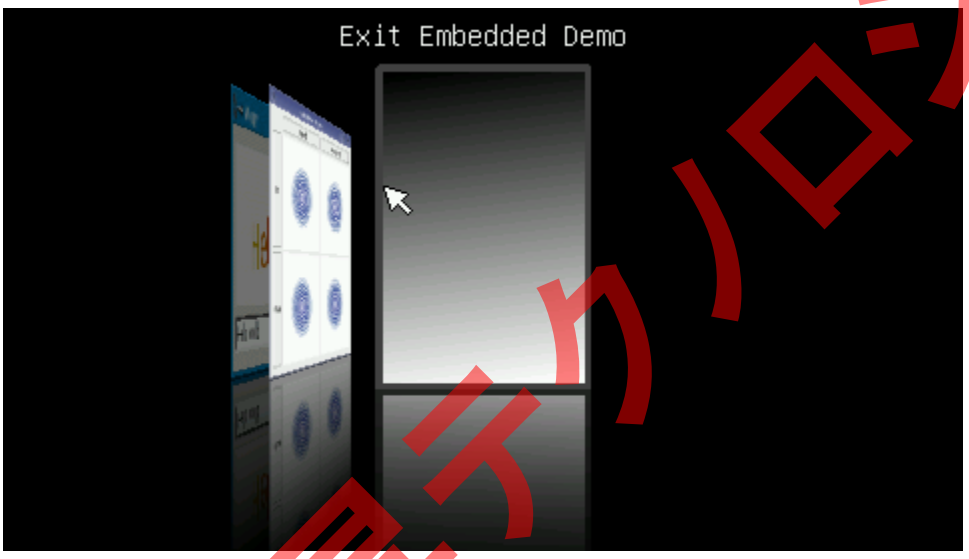
4.1.35 QtE-4.8.5 の起動

「FriendlyARM」タブの「Start Qt4.8.5」をクリックし QtE-4.8.5 を起動する。プログラムを閉じると、Qtopia-2.2.0 に戻る。

QtE-4.7.0を起動すると下図のようにCoverFlow効果の画面が出てくる。タッチペンで左右ドラッグでき、Cover をクリックし起動する：



「Exit Embedded Demo」をクリックすると QtE-4.7.0 を閉じ、Qtopia-2.2.0 に戻る。



4.1.36 Python によるハードウエアへのアクセス

開発ボードに搭載されたのは Python2.7.2 で、/opt/python ディレクトリーに起動できるテストプログラム/opt/python/pwm.py が格納されている。

Python は強い機能を持っている。コンパイルを要らず、システムレベルの API を統合しているから、ハードウエアへのアクセスなど簡単にできる。

4.1.36.1 python によるブザーのコントロール

vi で下記の内容をファイルに保存し、pwm.py と名称を付ける。

```
#!/usr/bin/python
import fcntl
fd = open('/dev/pwm', 'r')
```

fcntl.ioctl(fd, 1, 100)

Chmod755./pwm.py で実行可能権限を与えて、コマンドラインに./pwm.py を入力したら、ブザーが鳴る。このプログラムは出荷時、/opt/python ディレクトリーにプリセットしておいたから、直接起動できる。

ブザーを止める場合、プログラムの fcntl.ioctl(fd, 1, 100) を fcntl.ioctl(fd, 0, 100) に変更した上、再起動するといい。

4.1.36. 2 python と c/c++の混用

まず、A/Linux ディレクトリーの下の python-friendlyarm.tgz を PC Linux システムで解凍する。
コマンド：

```
cd /opt/
mkdir python-arm
cd python-arm
tar xvzf ~/python-friendlyarm.tgz
```

次は c++ファイル api.cpp を書く。

```
#include <Python.h>
class MyClass {
public:
int add(int x,int y) { return x+y; }
};
extern "C" int add(int x,int y)
{
MyClass obj;
return obj.add(x,y);
}
```

上記のプログラムを動的ライブラリーにコンパイルする。

```
arm-linux-g++ -fPIC api.cpp -o api.so -shared -I/opt/python-arm/include/python2.7
-I/opt/python-arm/lib/python2.7/config
```

作成された api.so を SD カードにコピーし、python スクリプトでコールできるようになる。

```
#!/usr/bin/python
import ctypes
plib = ctypes.CDLL('/sdcard/api.so')
print "result: %d" %(plib.add(1,2))
```

4.1.37 ssh による開発ボードへのリモートアクセス

開発ボードの Linux システムに ssh を統合した。PC でネットワークを介し、開発ボードの文字端末に入る手順：

（1）開発ボードでイーサネット（USB　WIFI でも可）に接続する。それからシリアル端末で ifconfig コマンドで開発ボードの IP アドレスを確認する。シリアル端末と接続していない場合、LCD のネットワーク設定で IP アドレス（例：192.168.1.230）を設定する。

（2）PC　Linux コマンドラインに ssh root@192.168.1.230（開発ボードの IP アドレス）を入力する。パスワード fa を入力したら、開発ボードの文字端末に入れる。PC は Windows システムの場合、putty ソフトウエアで登録できる。

開発ボード root のデフォルトパスワードは fa で、passwd.root コマンドでパスワードを変更できる。

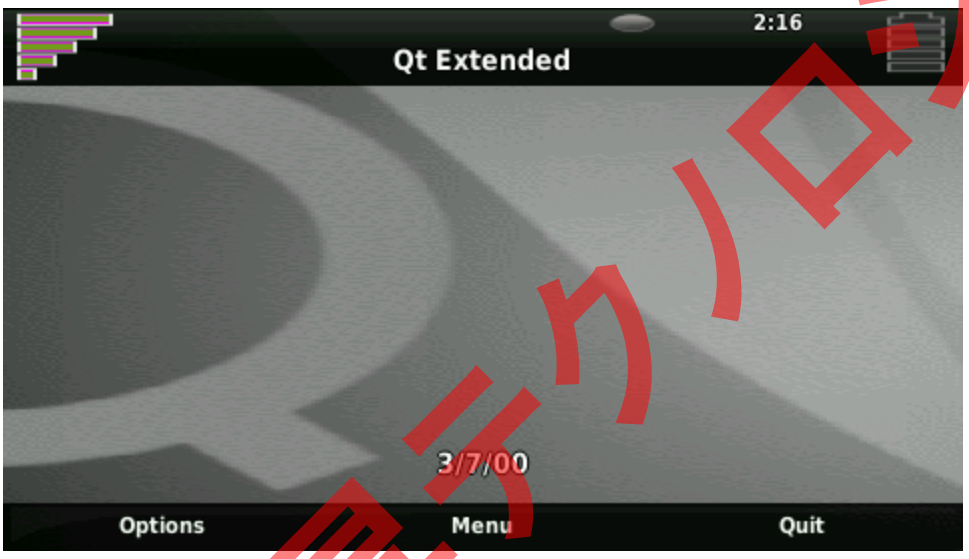
4.1.38 Qtopia4 の起動

「FriendlyARM」タブの「Start QtExtended4.4.3」をクリックし Qtopia4 を起動する。プログラムを閉じると、Qtopia-2.2.0 に戻る：

初回起動する時下図の画面が表示される：

画面提示によってスクリーンをタッチすると、設定画面が出てくる：

「Finish」をクリックし、Qtopia4 に入る：



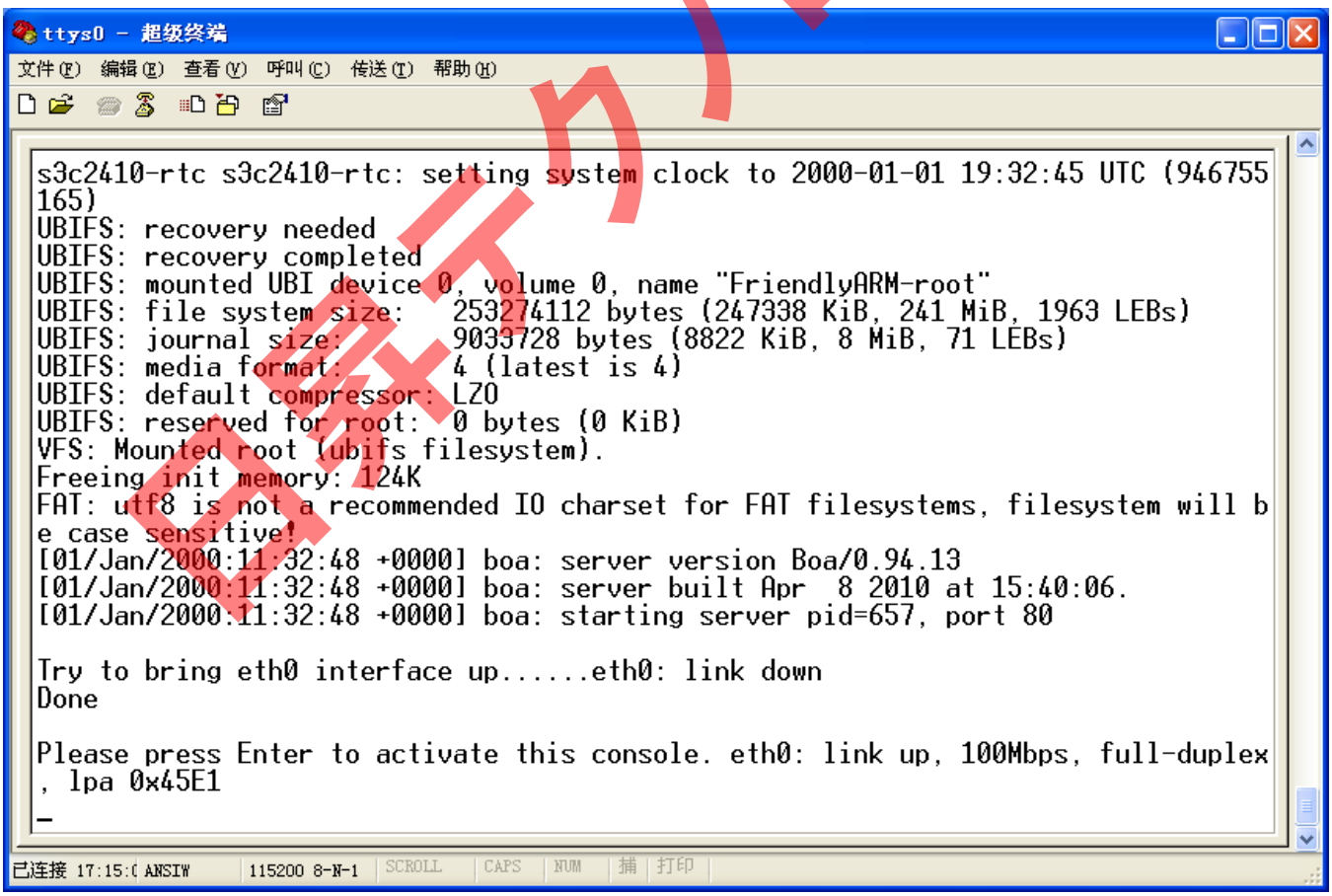
「Menu」をクリックし、ファンクション画面に入る：

「Back」をクリックしメイン画面に戻り、「Quit」をクリックすると Qtopia2 に戻る：

4.2 シリアルポート端末でボードの制御

本節の操作を行う前にインストールマニュアルの 2.1.3 をご参照してハイパーターミナルを設定してください。

Linux システムを起動して、ハイパーターミナルの端末で Enter をクリックすると、コンソールモードで下記テキストが出てくる：

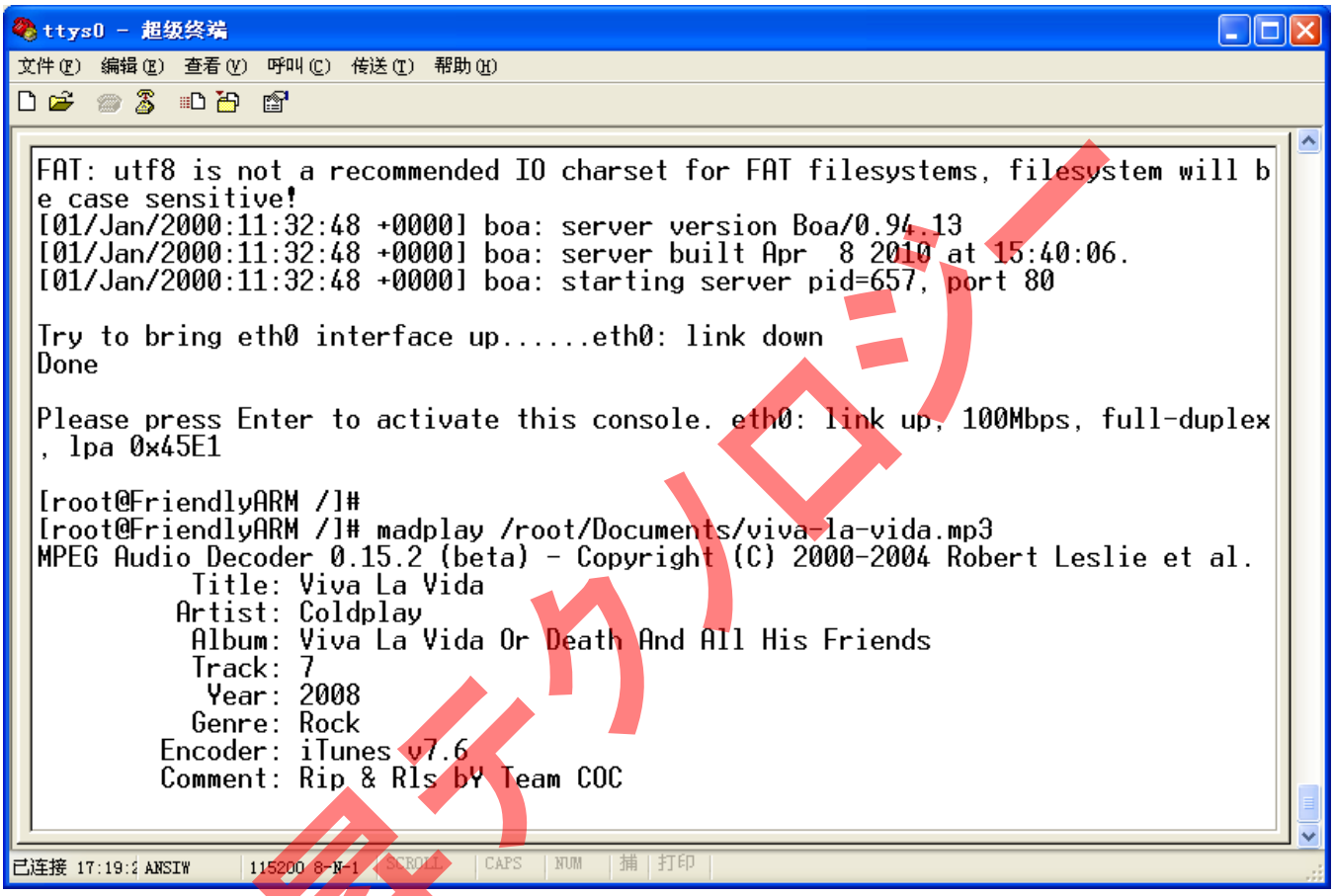


4.2.1 Mp3 の再生

Madplay は端末を基に移植した mp3 プレーヤーで、最も簡単な使用方法は下記の通り：

 # madplay your.mp3

このコマンドはデフォルトモードで your.mp3 というファイルを再生する（ボードには your.mp3 ファイルがない、ここでは例を挙げるだけ)。「madplay -h」コマンドでヘルプ画面が表示される。下図の画面はボードにプリインストールされた mp3 である：



4.2.2 アプリの中止

アプリを中止するため、ハイパーターミナルで同時に Ctrl と c を押す（先に Ctrl を押し、放さないまま c キーを押せば OK)。

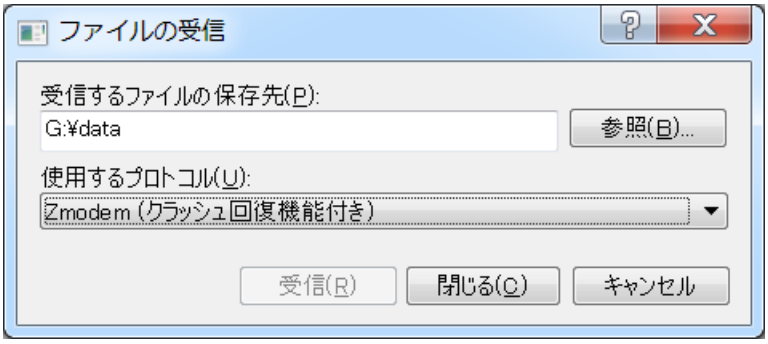
アプリはバックグランドで実行する場合、「kill」コマンドで該当プロセスをクロースできる。

4.2.3 シリアルポートで PC と相互ファイルの転送

注意事項：USB 転換ケーブルを使用する場合、順調にできない可能性もある。

1） sz コマンドで PC にファイルを発送する

ハイパーターミナルの画面にマウスの右ボタンをクリックし、「ファイルの受信」を選択し、下図のようにフォルダとプロトコルを設定する：

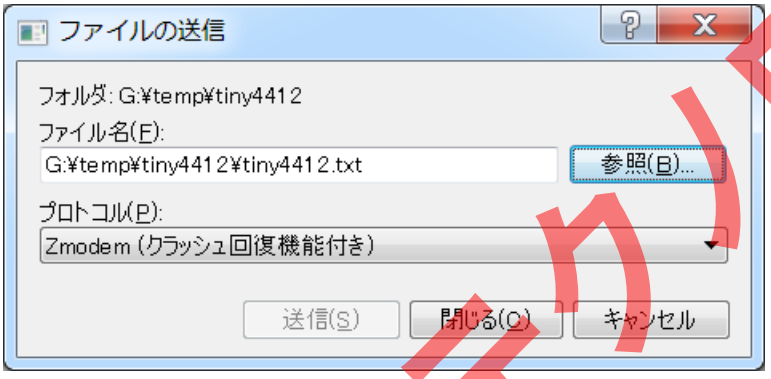


コマンドラインに“sz /root/Documents/viva-la-vida.mp3”を入力すると“/root/Documents/”フォルダにある viva-la-vida.mp3 ファイルを PC に転送し始める。該当ファイルのサイズは大きいため、何分かかるはずで、送信完了後、ファイルが設定したフォルダに保存される。

2） rz コマンドで PC からファイルをダウンロードする

rz コマンドを入力し、PC からのファイルを受信する。

ハイパーターミナルの画面にマウスの右ボタンをクリックし、「ファイルの送信」を選択し、下図のようにフォルダとプロトコルを設定する：



「送信」をクリックし、ボードがファイルを受信し始める。

受信完了後、md5sum コマンドで該当ファイルが送信したファイルかどうかを検証できる。

4.2.4 LED 制御

1) LED サーバー

システム起動の時、自動的に LED サーバー(led-player)を起動させる(/etc/rc.d/init.d/leds)。led-player が実行した後、/tmp/led-control というパイプファイルを生成し、該当パイプにパラメーターを送信し、LED を点滅させる：

#echo 0 0.2 > /tmp/led-control

4 つの LED が 0.2 秒周期で流れる。

#echo 1 0.2 >/tmp/led-control

4 つの LED が 0.2 秒周期で累計する

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

4 つの LED が停止する。

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

4 つの LED が点滅をリスタートする。。

2) 単独 LED 制御

/bin/leds は個別の LED を制御するプログラムである。使用する前に led-player をストップする必要がある。

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

LED サーバーをストップさせる。LED の使用方法は下記の通り：

[root@fa/]# led

Usage: leds led_no 0|1

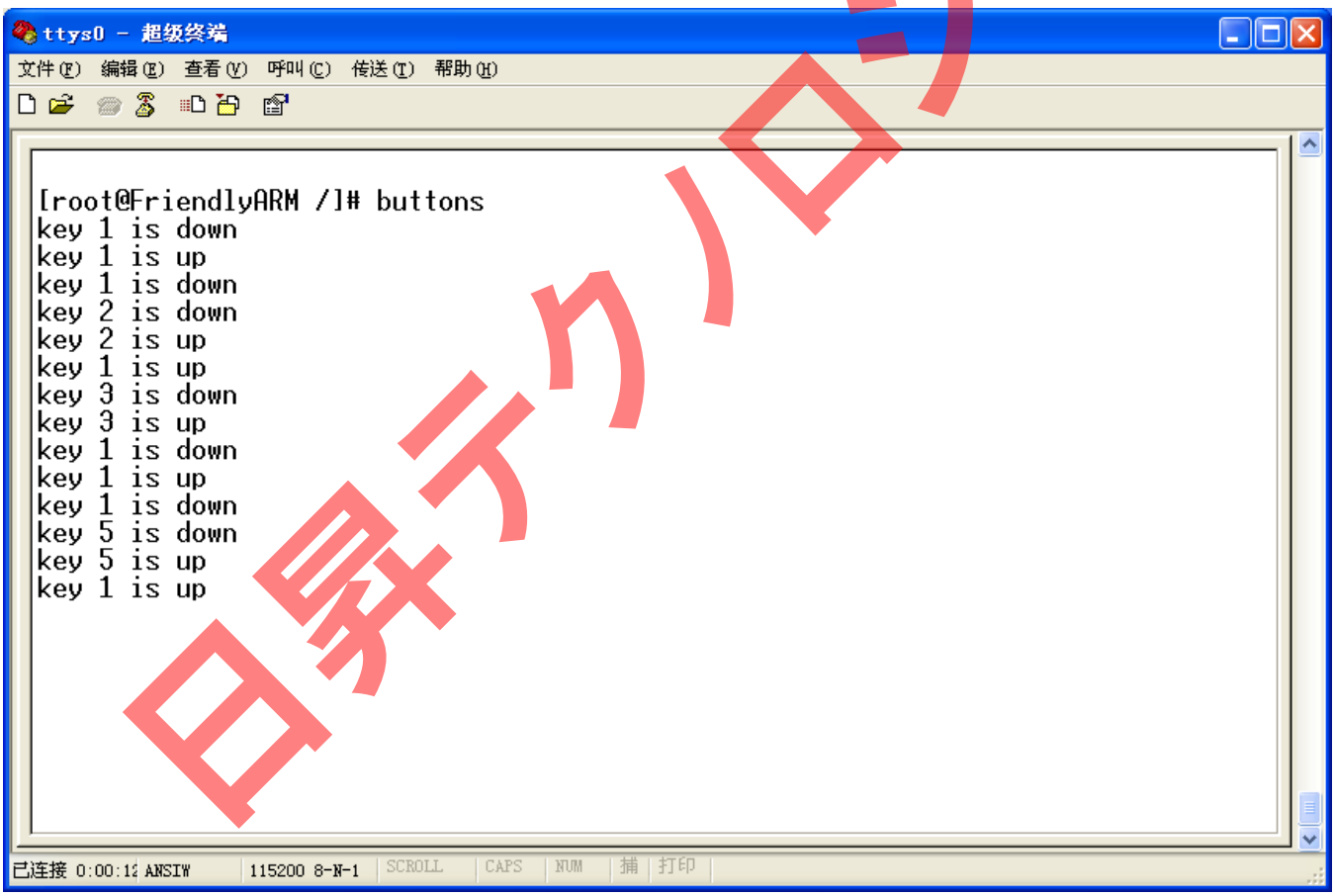
led_no は LED 番号(0,1,2,3)で、0|1 は消灯｜点灯に対応する。

#led 2 1

LED3 を点灯させる。

4.2.5 ボタンのテスト

「buttons」コマンドを入力し、ボードのボタンを押すと、下図のように対応する情報が出てくる：



4.2.6 シリアルポートのテスト

Linux システム起動後。シリアルポート 0,1,2 は/dev/ttySAC0,1,2,3 に対応する。

シリアルポート 2 をテストする時は COM2 をシリアルポートのある PC に接続する。ご使用になる PC のボーレートは前述の通り 115200 に設定する。

コマンドラインに下記コマンドを入力する：

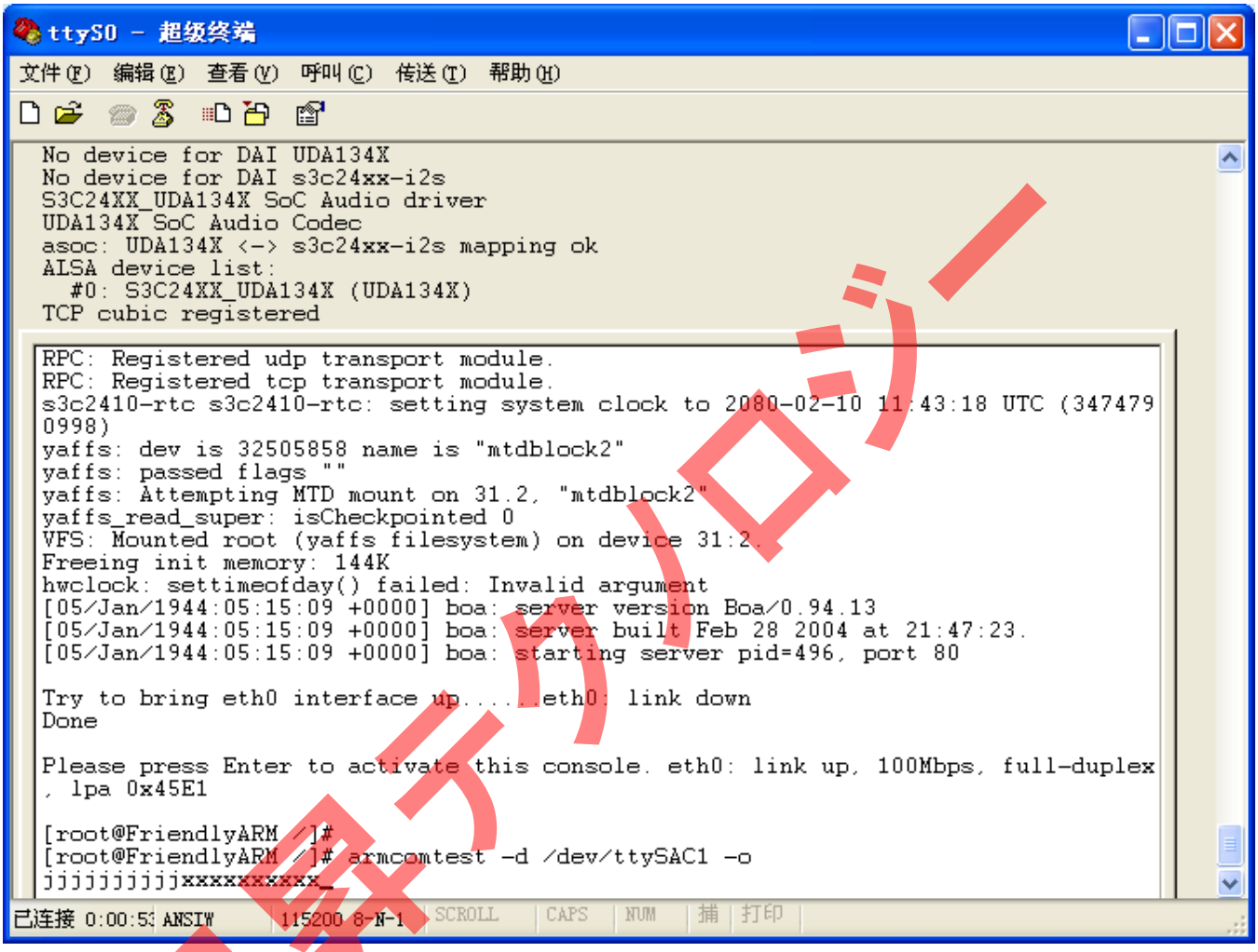
#armcomtest -d /dev/ttySAC1 -o

キャラクターを入力すると、PC のハイパーターミナルに表示する。

シリアルポート 3 をテストする場合は COM3 をシリアルポートのある PC に接続しコマンドラインに下記コマンドを入力する：

#armcomtest -d /dev/ttySAC2 -o

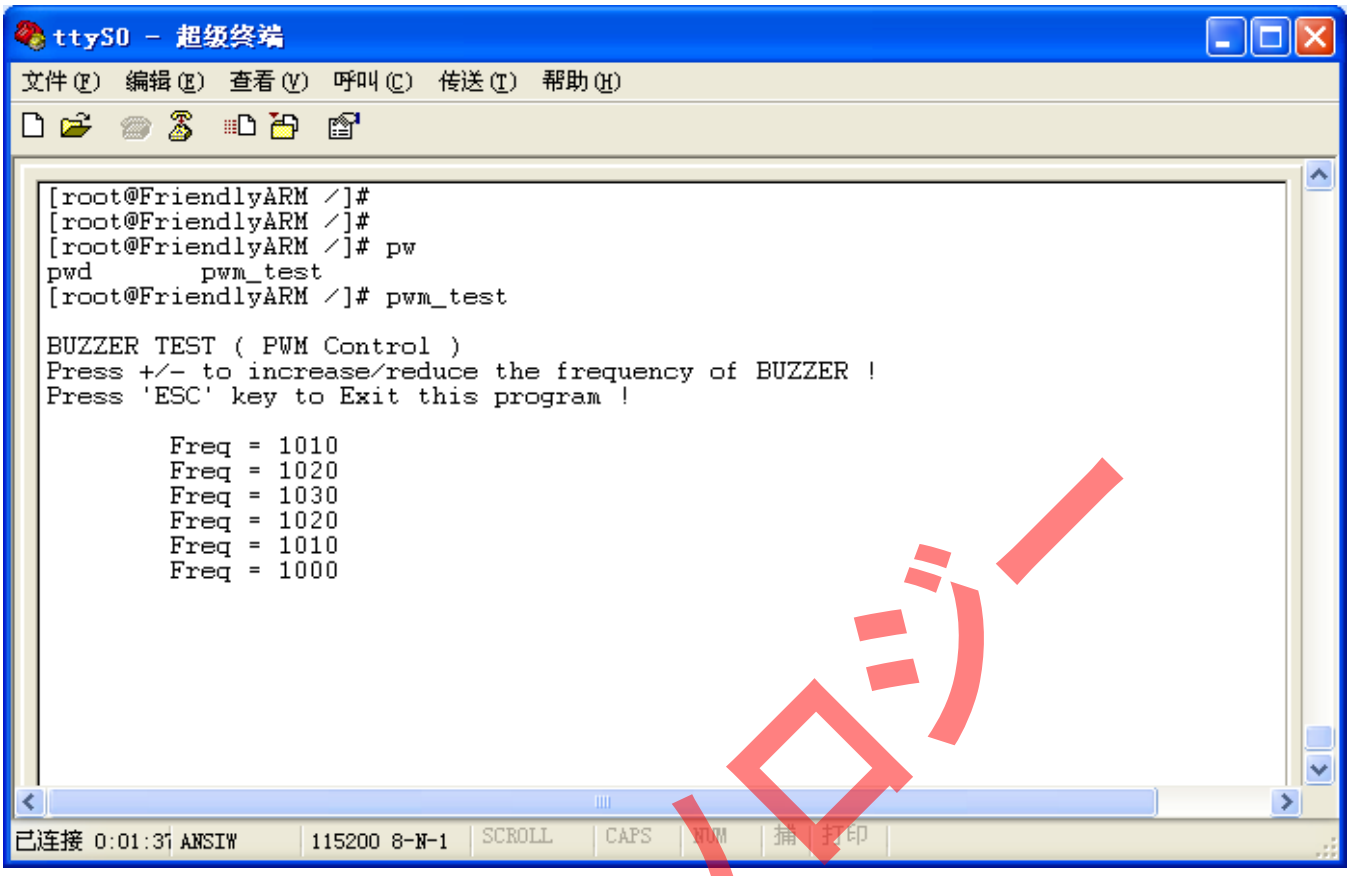
テストの画面は下図の通り：



4.2.7 ブザーテスト

コマンドラインに“pwm_test”を入力し、ブザーの音が出る。“+”或いは“-”で下図のように出力 PWM の周波数を変化させる。

ESC キーでテストを中止させる。



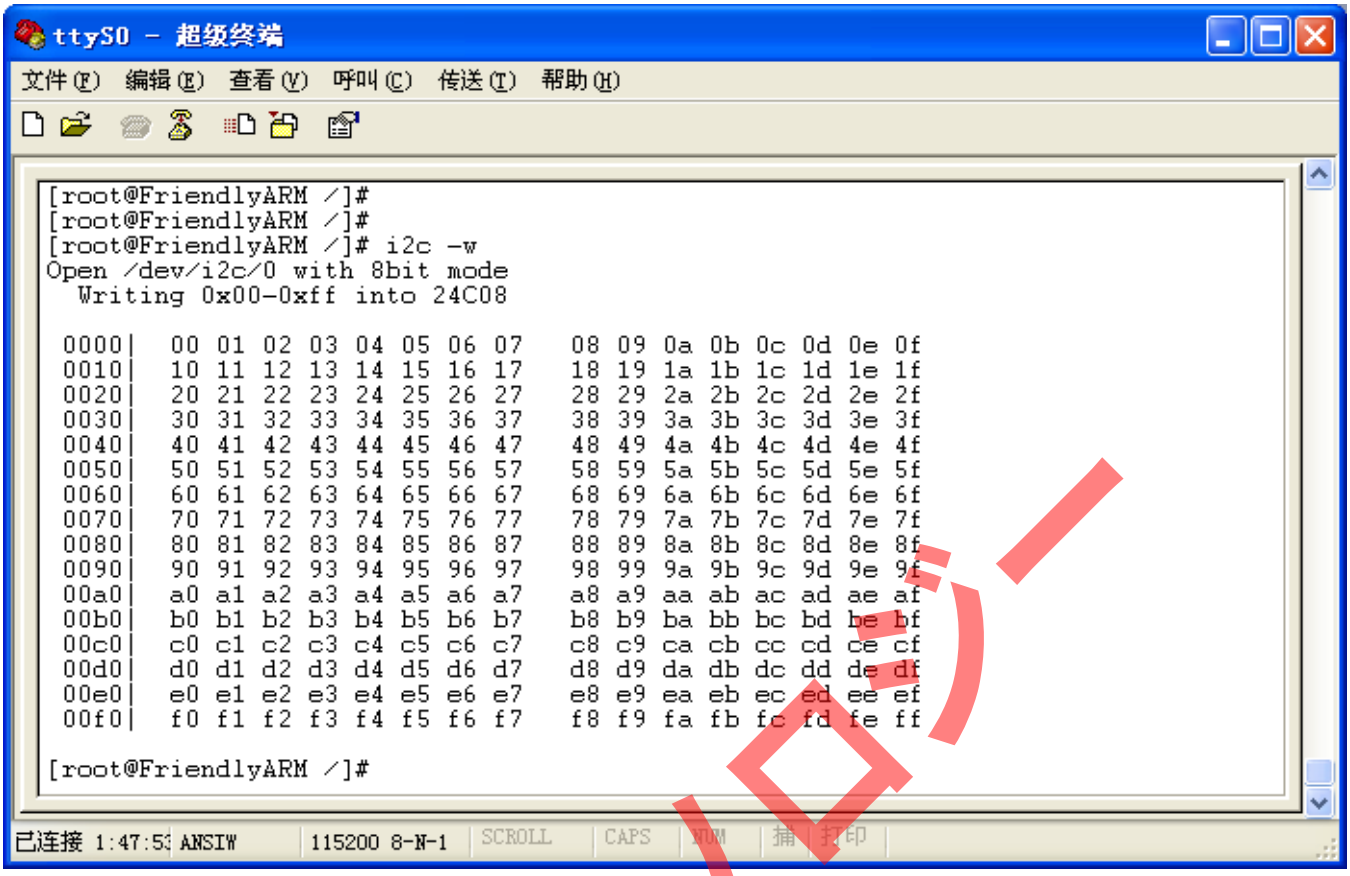
4.2.8 LCD バックライト制御

LCD デバイスファイル：/dev/backlight-1wire

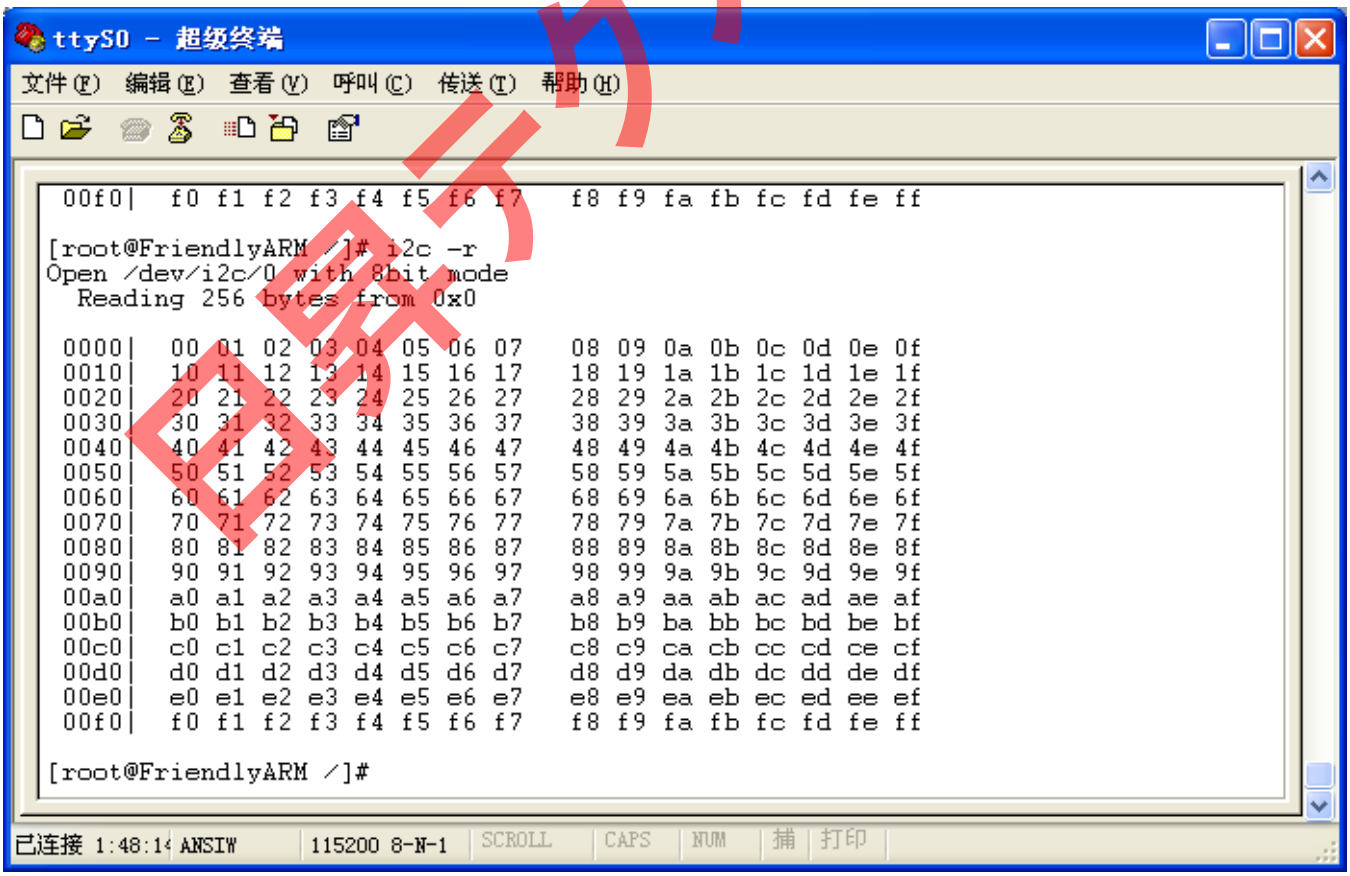
#echo 0 > /dev/backlight でバックライトを消灯させる。
バックライトデバイスファイルに 1-127 を送信すれば、バックライトの明るさを調節できる。127 は最も明るい状態である。
#echo 15 > /dev/backlight を入力すると、微かなバックライトが見える。

4.2.9 I2C-EEPROM テスト

コマンドラインに I2c-w を入力し、ボードの 24C08 にデータ（0x00-0xff）を書き込む。

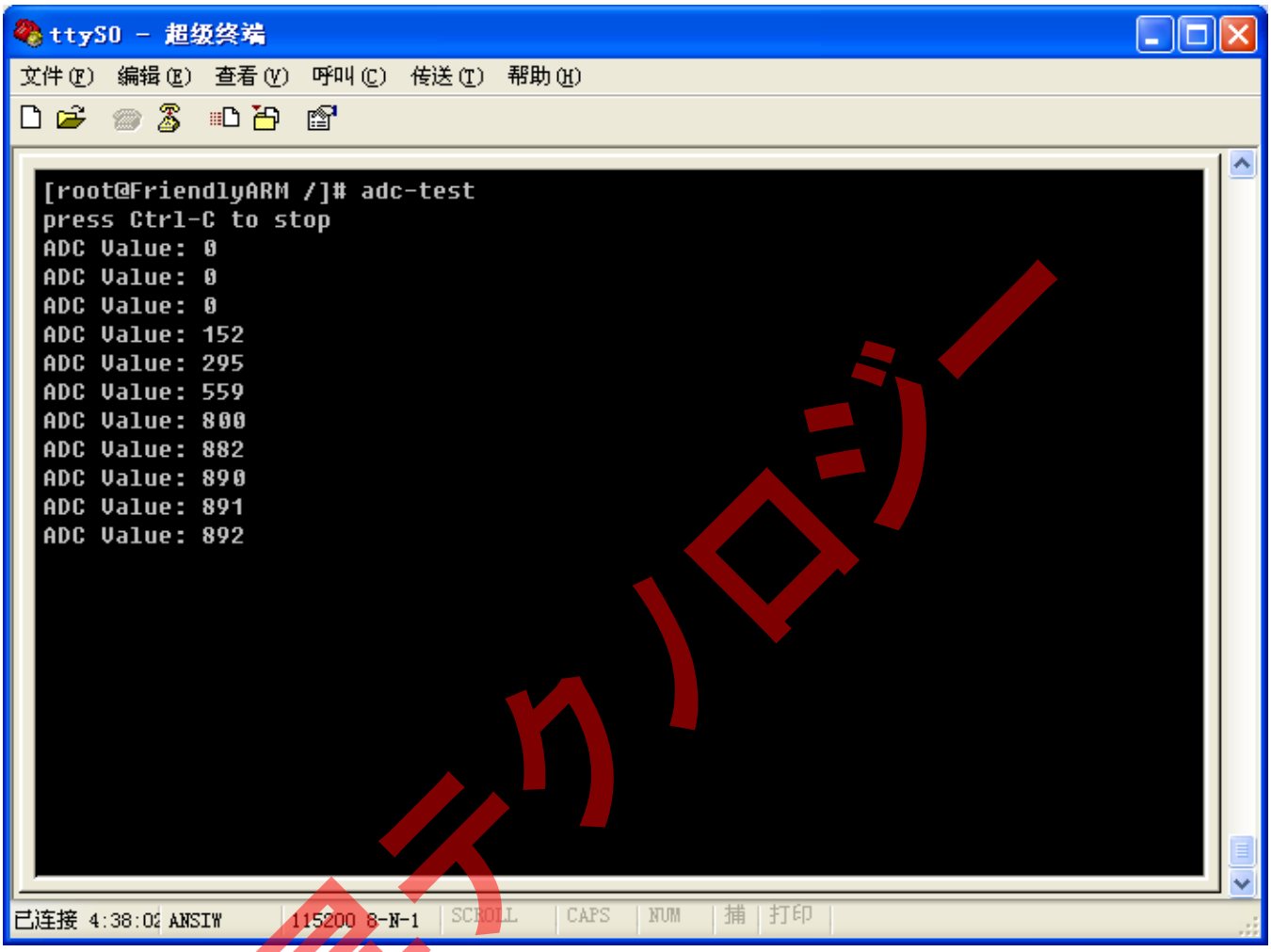


コマンドラインに I2c-r を入力し、ボードの 24C08 からデータ（0x00-0xff）を読み出す。



4.2.10 AD テスト

Adc-test コマンドを入力し、AD 転換のテストを行う。開発ボード上の可変抵抗 W1 を変更すれば、ハイパーターミナルから転換結果が見られる。



4.2.11 WiFi 無線 LAN の設定

多種の USB 無線 LAN のドライバーを実装しています。
弊社販売している USB 無線 LAN(802.11bg)で設定手順を説明します。(SD WiFi の設定も同じ)
製品紹介 HP：http://www.csun.co.jp/SHOP/200906118.html

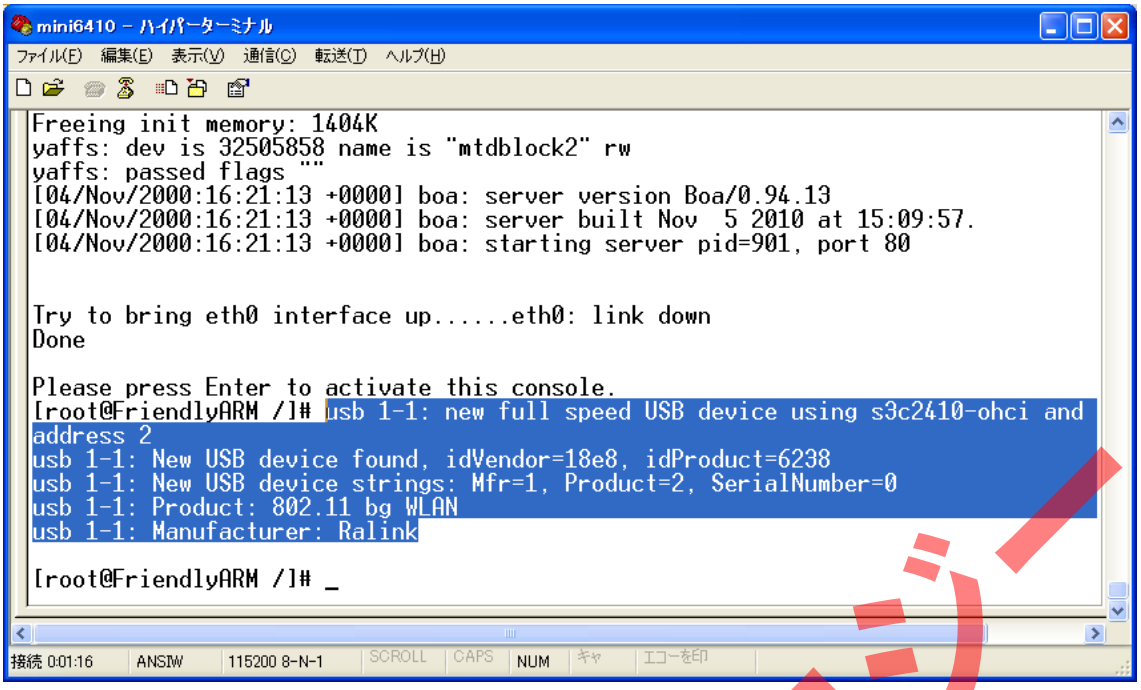
下記三つのプログラムでで実現している：

- scan-wifi - 無線LANをスキャンする
- start-wifi - 無線LANを起動する
- stop-wifi - 無線LANを停止する

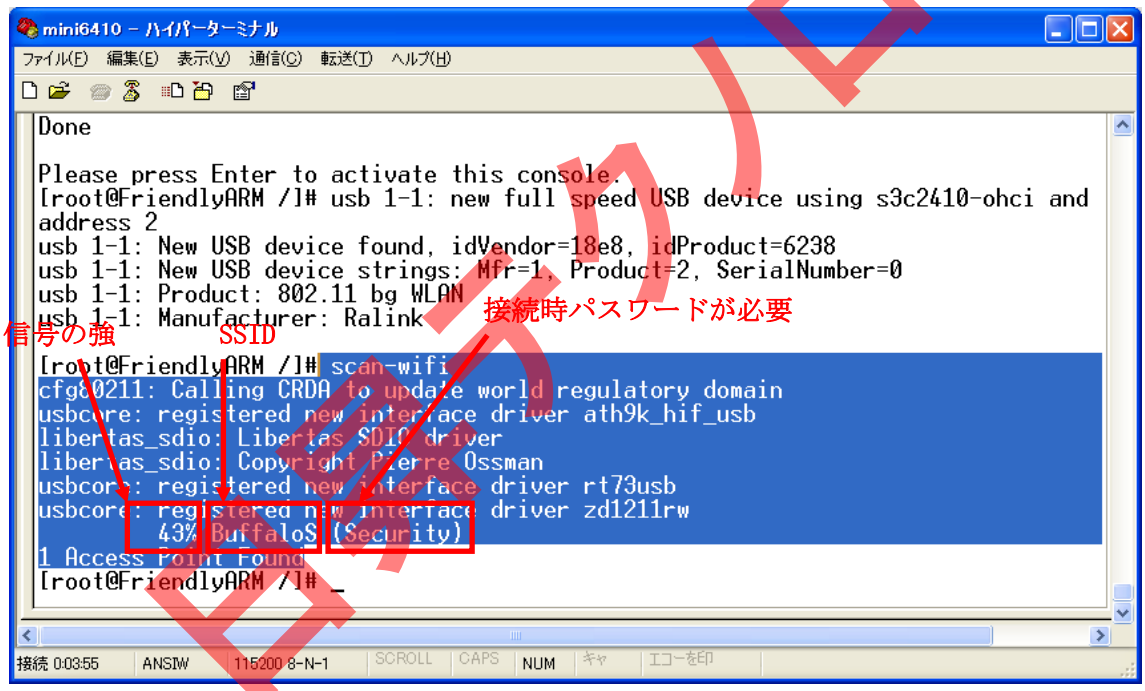
三つのプログラムは/usr/sbin ファルダにインストールされている。

1． 無線 LAN をスキャンする

USB 無線 LAN モジュールをボードに差込む。ハイパーターミナルに下記の様な情報が表示します：



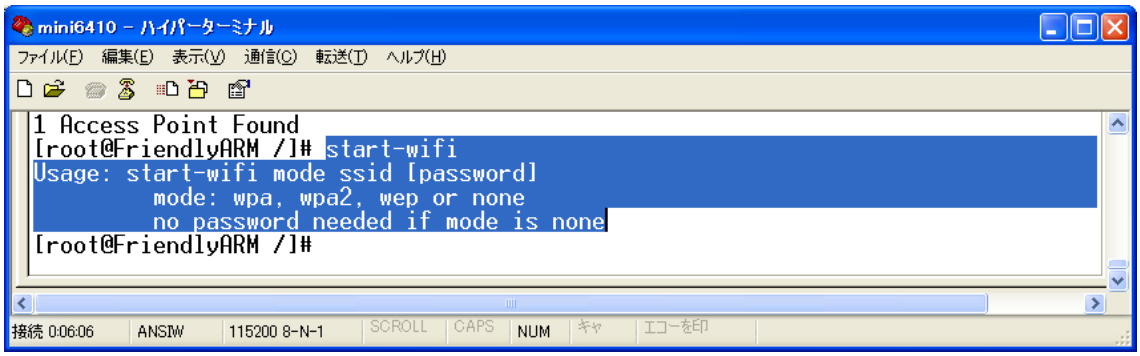
下図のように#scan-wifi コマンドを実行し、無線 LAN を検索する：



無線 LAN 信号の強さ、名称を表示し、パスワードのある LAN は「Security」と表示する。

2. 無線 LAN に接続する

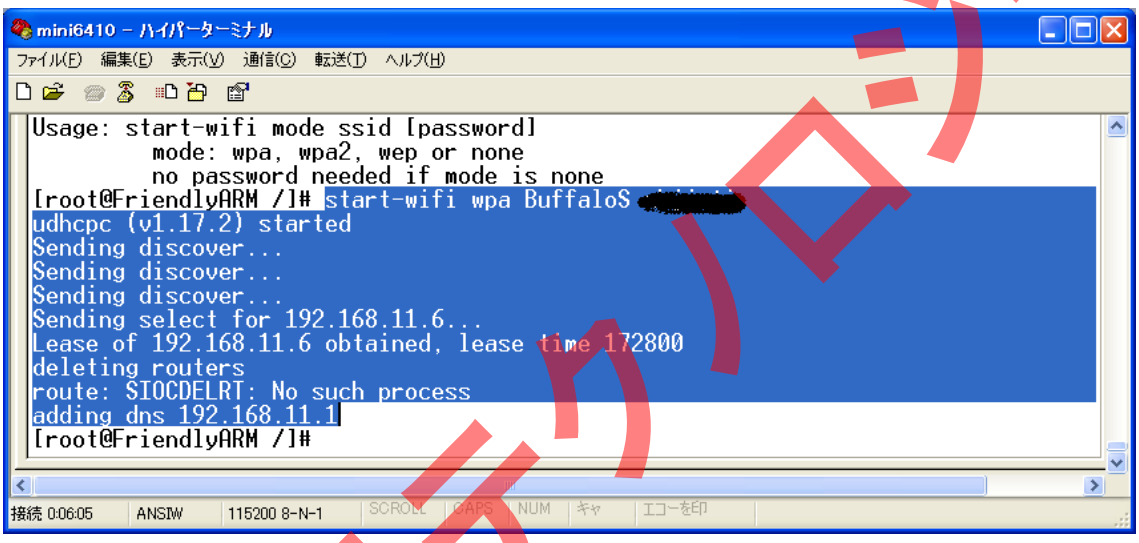
各種の無線 LAN によって、接続時必要なパラメーターが違う。Start-wifi コマンドで下図のように提示情報が出てくる：



mode -セキュリティーモード、“wpa”、“wpa2”、“wep”或いは“none”になる。“none”はパスワード不要。

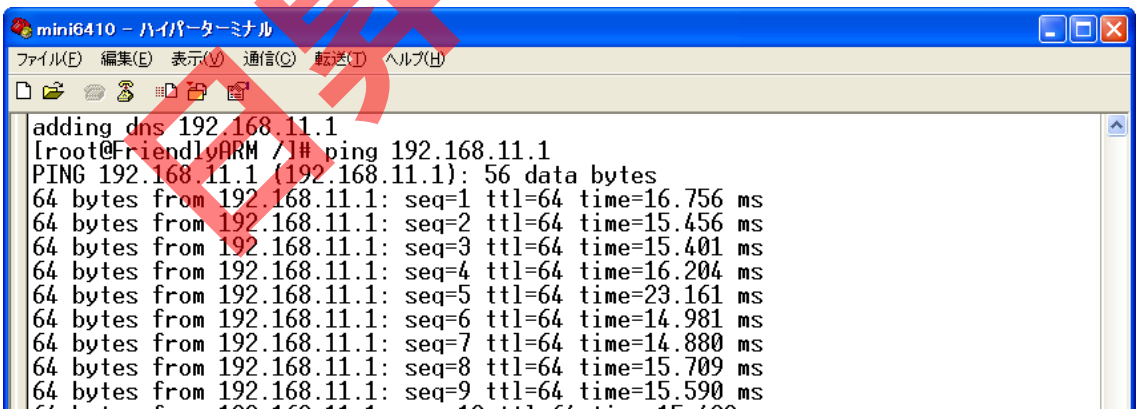
ssid -接続しようとする無線 LAN の名称。

password -無線 LAN に接続時のパスワード。



暫くすると自動的に IP アドレス 192.168.11.6 が配れます。

Ping コマンドで接続されたか確認して見ます。



※パスワードない無線 LAN に接続する場合は start-wifi none ssid コマンドで実現できます。

3. 無線 LAN を切る

stop-wifi コマンドで実現で無線 LAN を切る。

4.2.12 ネットワークの設定

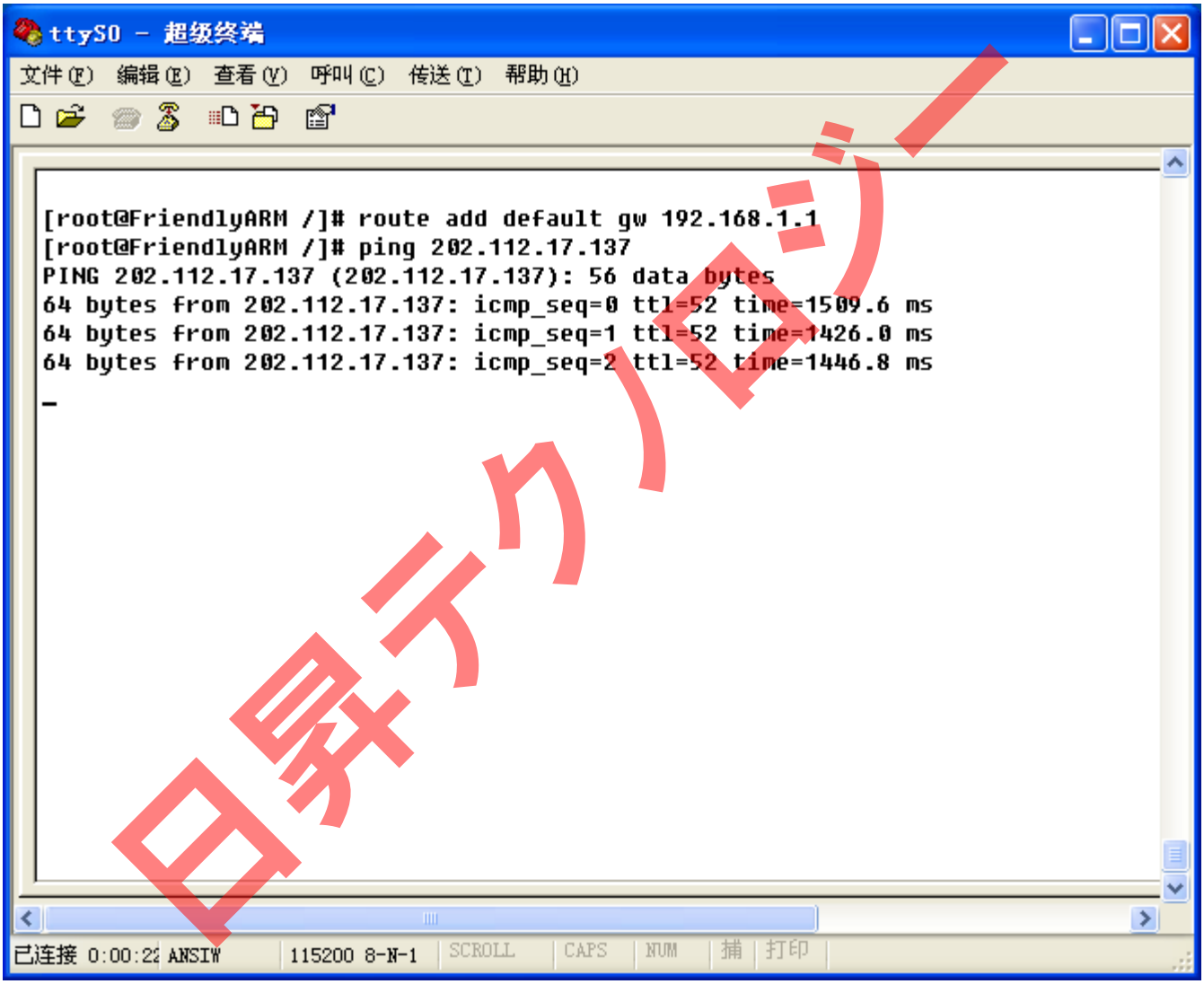
先ずはネットワークがインターネットにアクセスできるを確保ください。ゲートウェイ IP アドレスをメモし（例えばここは 192.168.1.1)、route コマンドで設定を行う：

```
# route add default gw 192.168.1.1
```

これで IP アドレスで直接にインターネットにアクセスできる。例えば 202.112.17.137 アドレスを Ping する場合：

```
#ping 202.112.17.137
```

下図の画面が出てくる：



ドメイン名で Ping したい場合は DNS サーバーIP アドレスを確認する必要がある。

例えば DNS サーバーIP アドレスを“202.96.128.86”とし、下記の設定を行う：

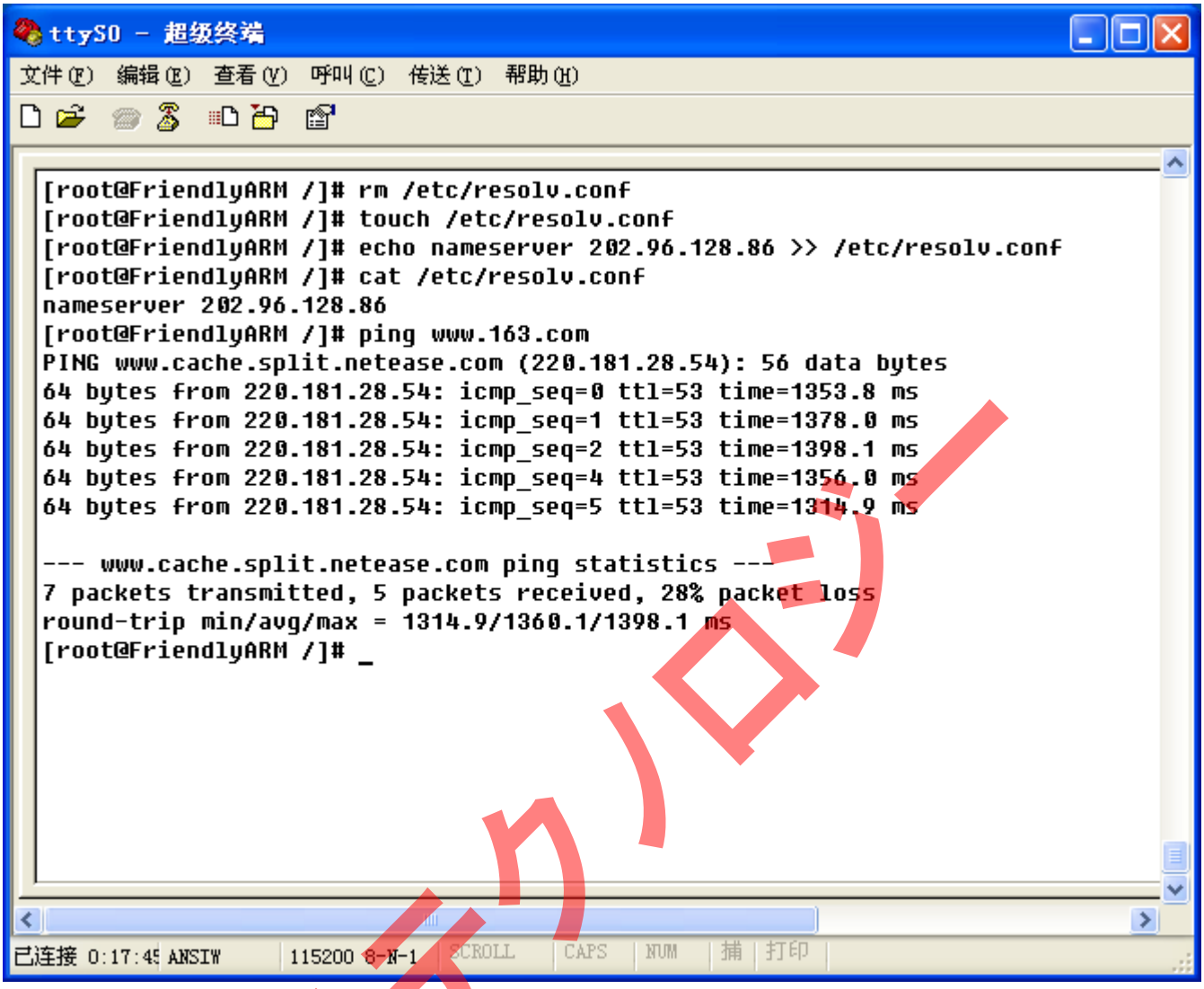
#rm /etc/resolv.conf：元の設定ファイルを削除する

#touch /etc/resolv.conf：新しい設定ファイルを作成する

#echo nameserver 202.96.128.86 >> /etc/resolv.conf：DNS サーバーIP アドレスを設定ファイルに書き込む。

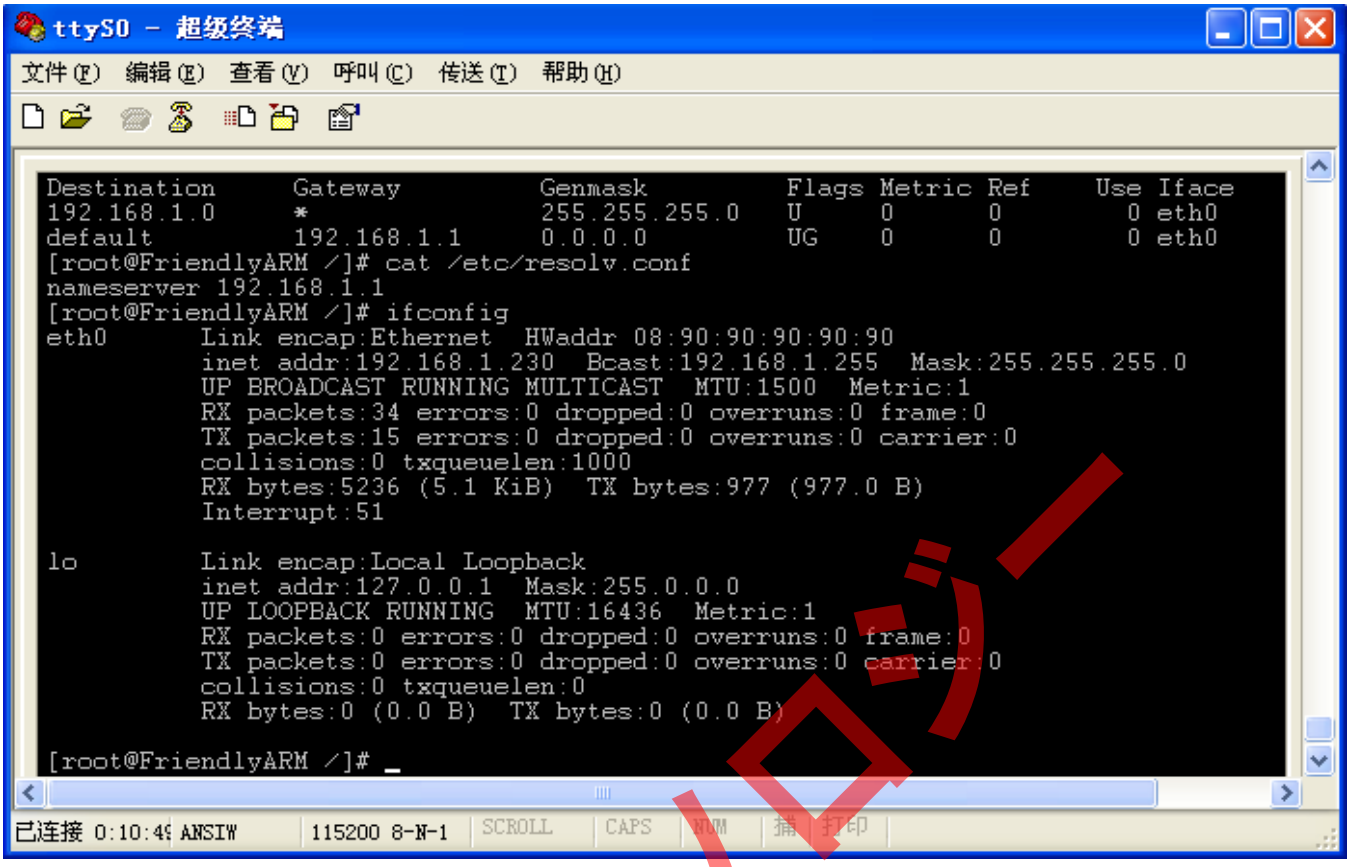
ここでは主に/etc/resolv.conf ファイルを修正する。vi コマンドでも修正できる。

プロセスは下図の通り：



4.2.13 MAC アドレスの設定

#ifconfig コマンドでカレントの MAC アドレスを確認する：



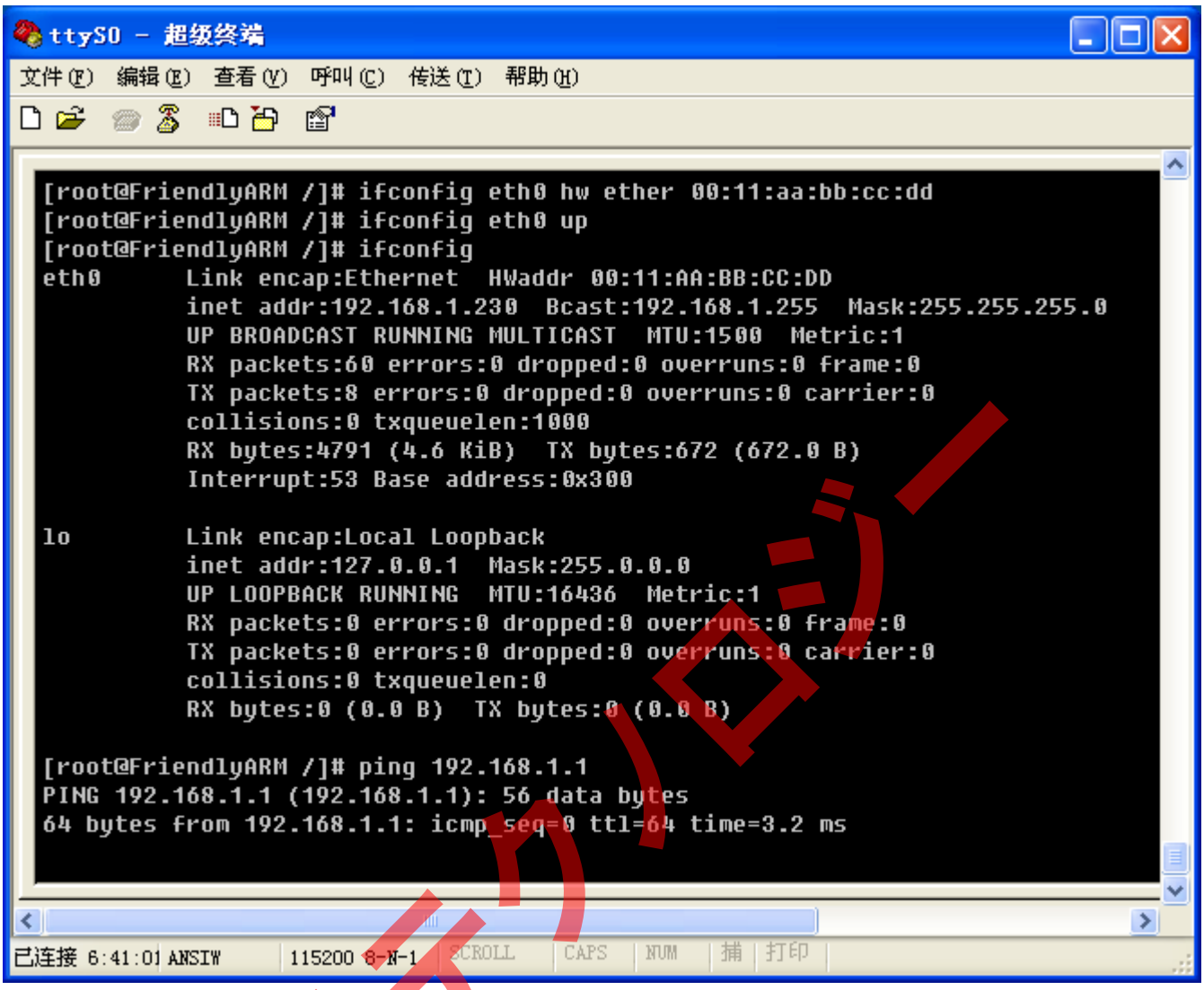
上図のようにカレントの MAC アドレスは“08：90：90：90：90：90”。これはデフォルトの MAC アドレスである。

MAC アドレスを変更する場合は先ずネットワークをクロースし、下記のように ifconfig で新しい MAC アドレスを設定する：

```
#ifconfig eth0 down
#ifconfig eth0 hw ether 00:11:AA:BB:CC:DD
```

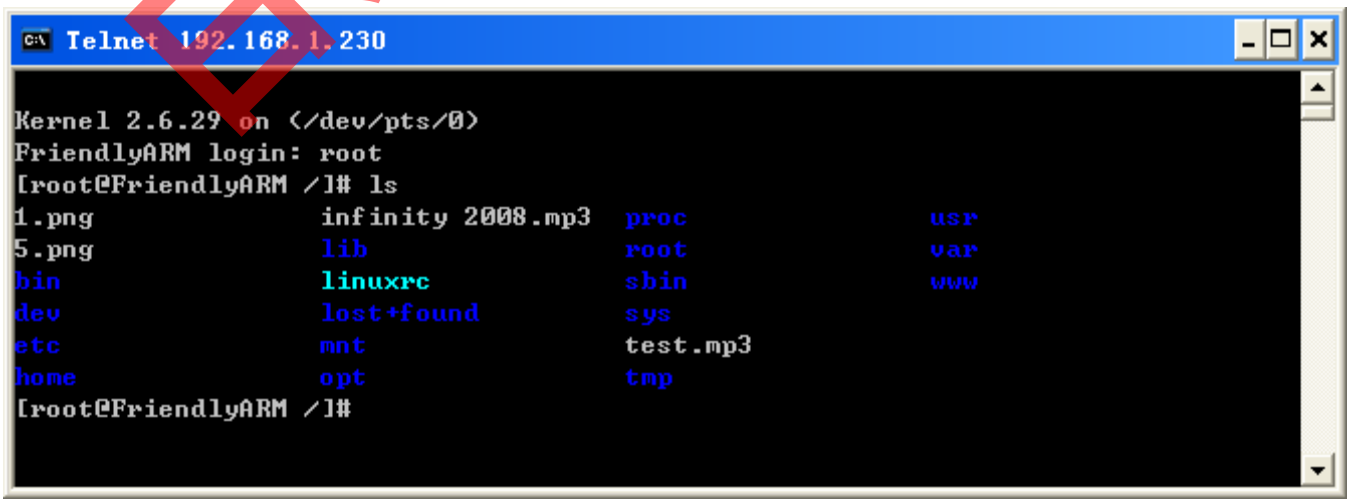
ネットワークを起動し、変更後の MAC アドレスが見られる。下記のように Ping コマンドでネットワーク接続正常か確認できる：

```
#ifconfig eth0 up
#ifconfig
#ping 192.168.1.1
```



4.2.14 Telnet で開発ボードにログオン

Windows の CMD から「telnet 192.168.1.230」を入力し、下図の画面が出てくる。「root」を入力し（パスワード不要）、システムに入る。



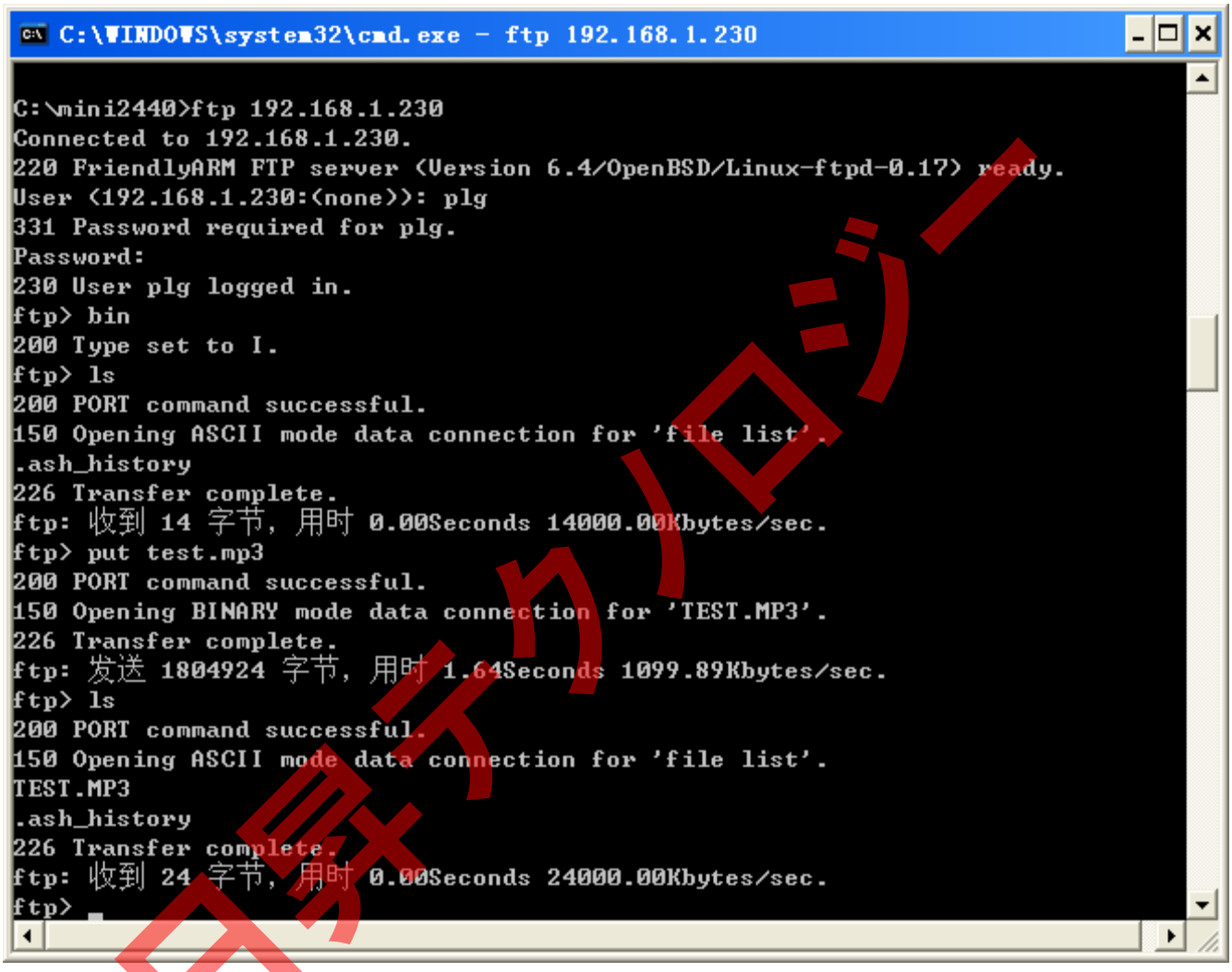
4.2.15 FTP 機能

本ボードには ftp コマンドと ftp サーバーがインストールされた。テストを手軽にするため。PC の CMD からボードにログオンし、ファイルを送信する。

注意事項：フォルダにファイルがあることを確保ください。ここは test.mp3 になる。

ftp のユーザー名は **plg** で、password も **plg** である。

送信完了後、/home/plg フォルダに test.mp3 ファイルが出てくる。



4.2.16 WEB からボード上の LED の制御

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、ボード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 をアクセスできます。

「LED テスト」は CGI でボード上の LED を制御し、二つの表示モードと三つの表示スピードが選択できる。

ウェブサーバーを停止する場合、下記のコマンドを入力する：

```
#/etc/rc.d/init.d/httpd stop
```

再起動する場、下記のコマンドを入力する：

```
#/etc/rc.d/init.d/httpd start
```

4.2.17 RTC の設定

Linux には通常 date コマンドで時間変更し、hwclock コマンドで S3C2440 のクロックを Linux のクロックと一致させる。使用方法は下記の通り：

(1)**#date -s 042916352007** #現在の時間を 2007-04-29 16:34 に設定する

(2)**#hwclock -w** #設定した時間を S3C2440 の RTC に保存する。

(3)起動時**#hwclock -s** #で Linux クロックを RTC に回復できる

注意事項：hwclock -s コマンドは既に起動スクリプト(/etc/init.d/rcS)に書き込み、起動時自動的に実行する。

4.2.18 パワーダウン時フラッシュにデータの保存

本開発ボード OS は読み書き可能なファイルシステム yaffs2（組み込みシステムに専用の Flash 管理のファイルシステム）をさいようしたため、便利に動的にデータを保存でき、パワーダウン後にもデータを失うことがない。ハイパーターミナルに下記コマンド入力：

#cp /shanghaitan.mp3 /home/plg

このコマンドは「/home/plg」フォルダに同じファイルをコピーした。シャットダウンし、システムをリスタートした後、「/home/plg」フォルダに shanghaitan.mp3 が見られる。

4.2.19 自動起動アプリの設定

起動スクリプトで自動起動アプリを設定できる。Windows システムの Autobat と類似する機能である。起動スクリプトは/etc/init.d/rcS ファイルにある。

4.2.20 画面コピー

snapshot コマンドで LCD の表示を画面コピーし、png フォーマットで保存する。

#snapshot pic.png を実行すると、LCD の表示をコピーし、pic.png ファイルとして保存する。

4.2.21 メモリのチェック

本ボードは 512M DDR RAM を提供しており、「cat /proc/meminfo」コマンドでチェックできる。2D/3D 加速とマルチメディアドライバが一部分のメモリを使っているため、表示するメモリは 512M より少ない。

4.3 Fedora9.0 のインストールと設定

本節はバーチャルマシン/PC で Fedora9.0 のインストールをはじめ、Linux 開発環境を詳しく説明する。他のプラットフォームでテストしたことがないが、Linux に十分詳しいなら、エラーを解決できると思われる。多くのエラーは対象のプラットフォームにあるライブラリ或いはツールが足りないことによるものである。初心者に対しては、弊社と同じプラットフォーム（Fedora9.0）のご使用をお勧め。

該当会社のホームページからダウンロードできる。

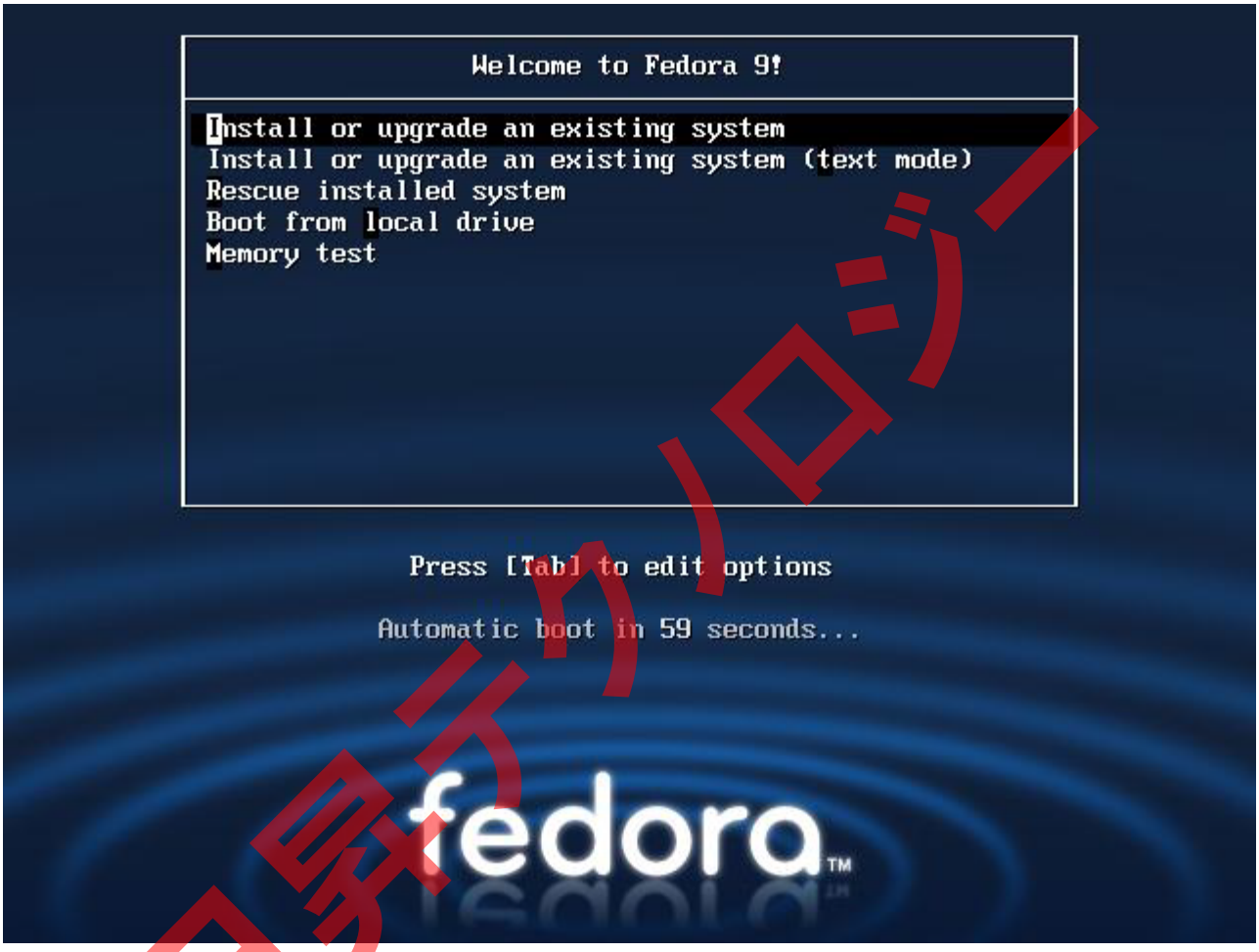
ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso

インストールする時、コンポーネントを漏らさないように、マニュアルに従って実施してください。

Linux ディストリビューションは多いため、全てのインストール手順を作成するのは不可能であることをご了承ください。

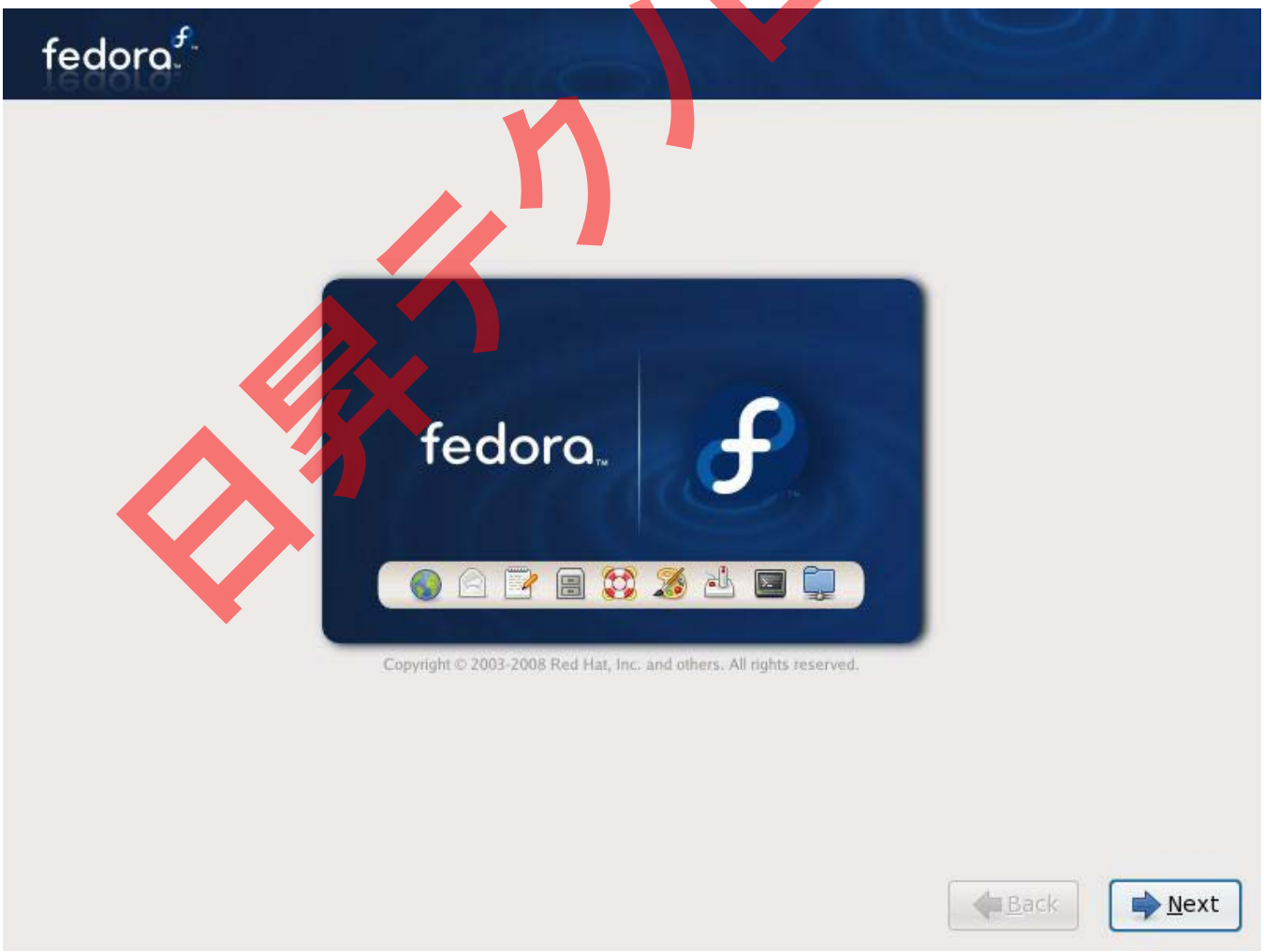
4.3.1 Fedora9.0 のインストール

Step1：インストールディスクを DVD ドライバに入れ。BIOS をディスクから起動と設定し、システム起動後下図の画面が出てきて、Enter キーで次に進む：

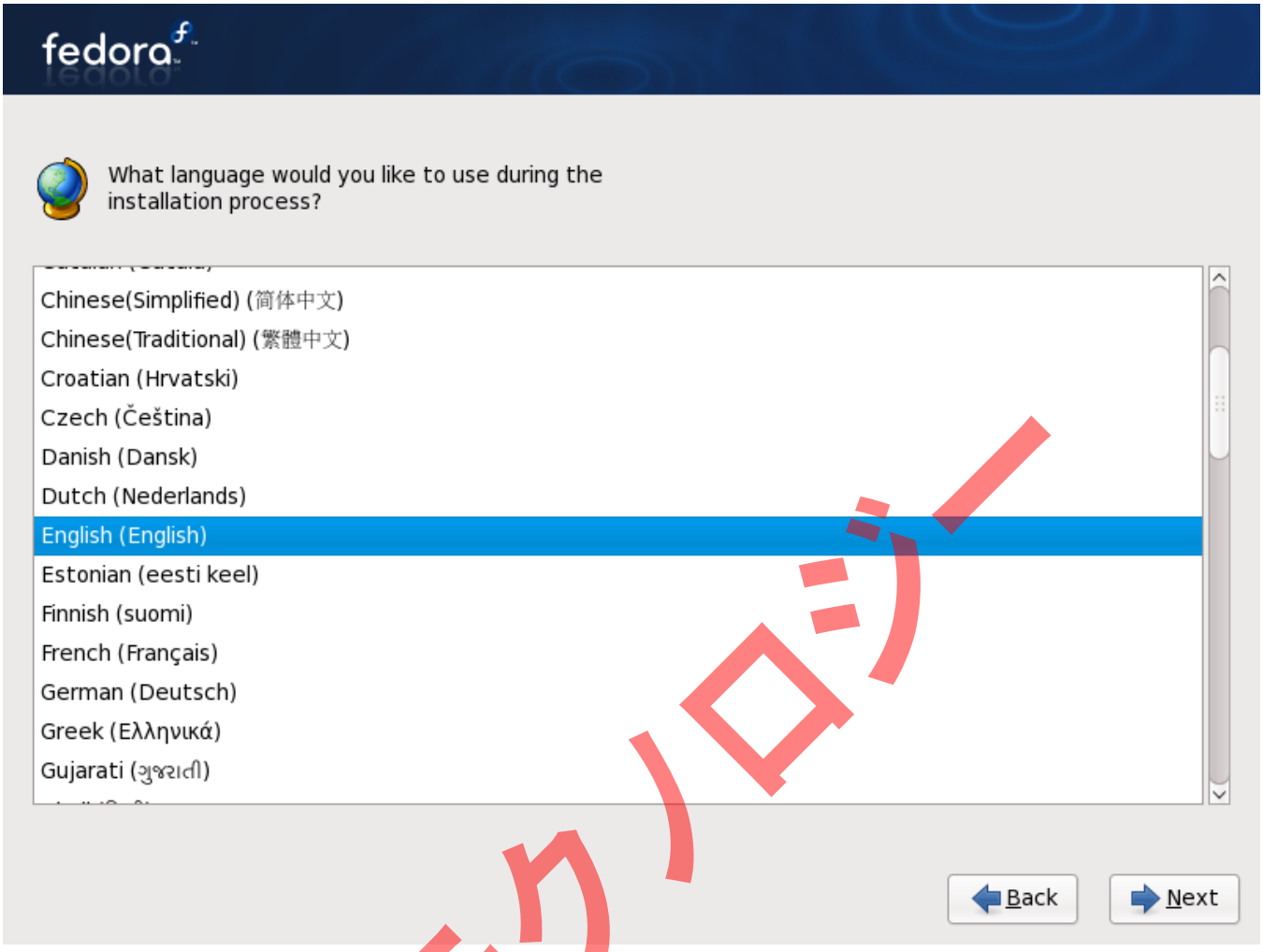


Step2：インストールディスクのチェックプロセスに入る。通常はチェックが必要ないため。Skip を選択する：

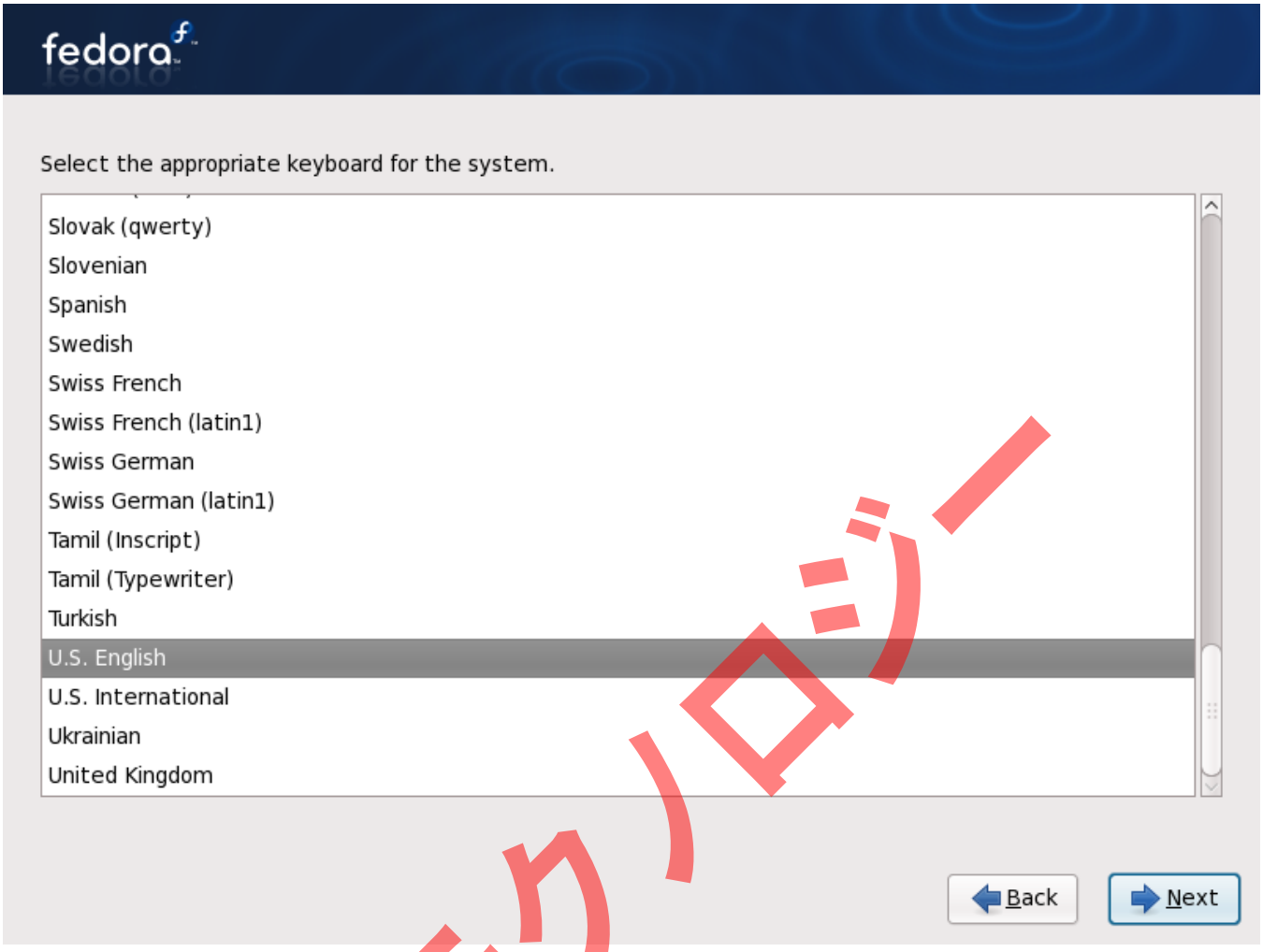
Step3：暫くするとインストール画面に入り、Next をクリックする：



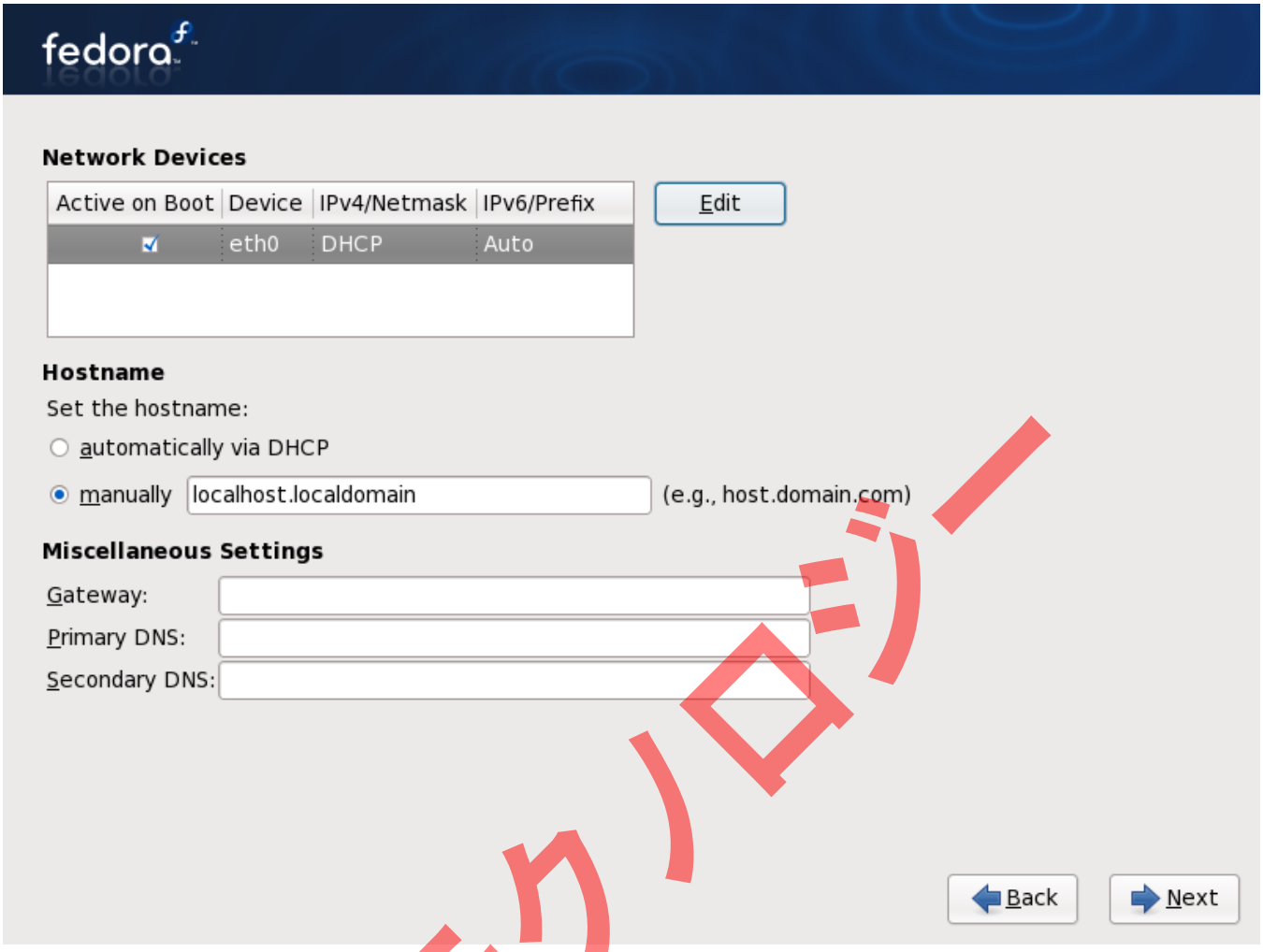
Step4：インストール言語を選択する、ここは英語：



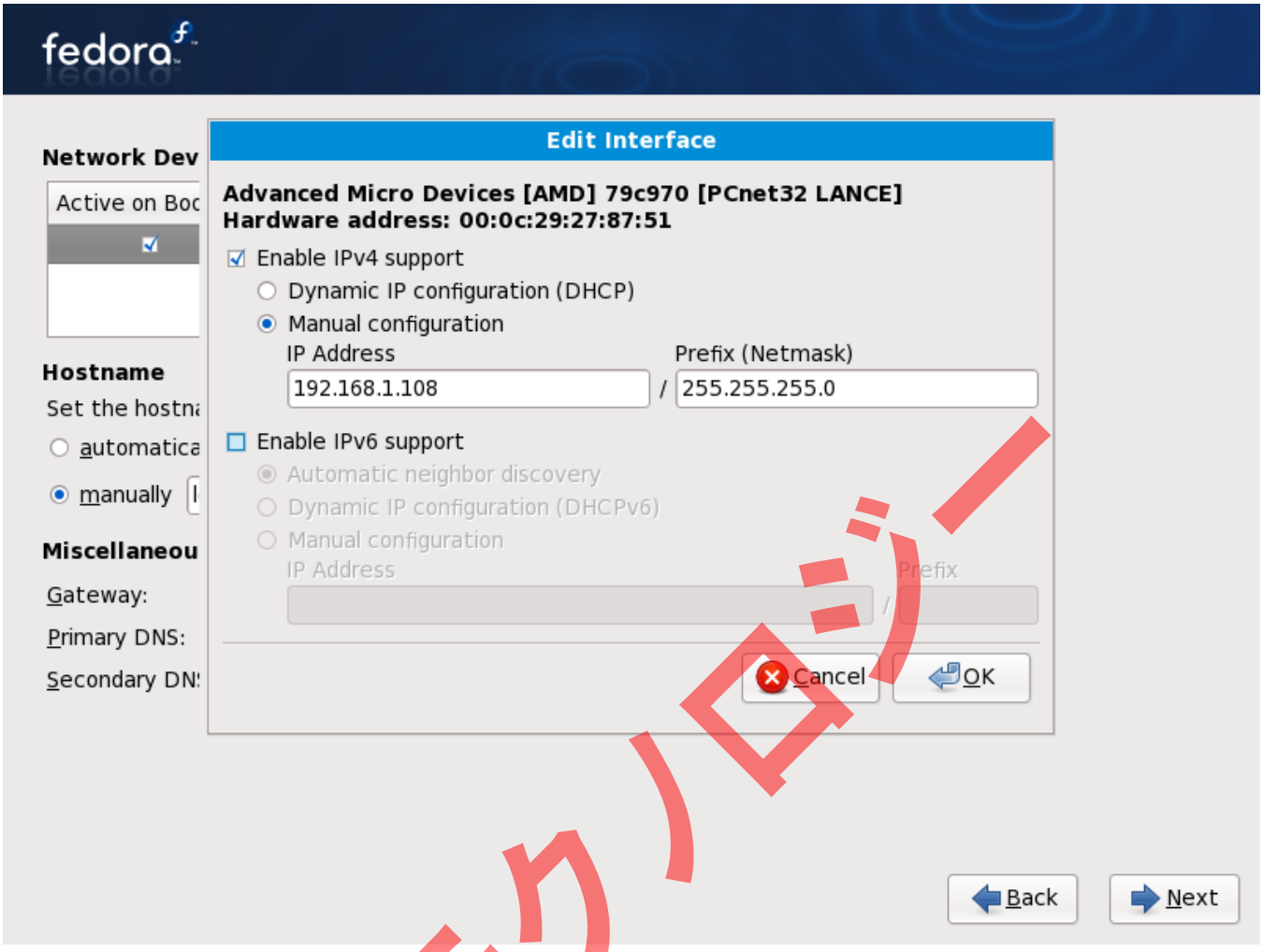
Step5：キーボードを選択する、ここはアメリカ式キーボード：



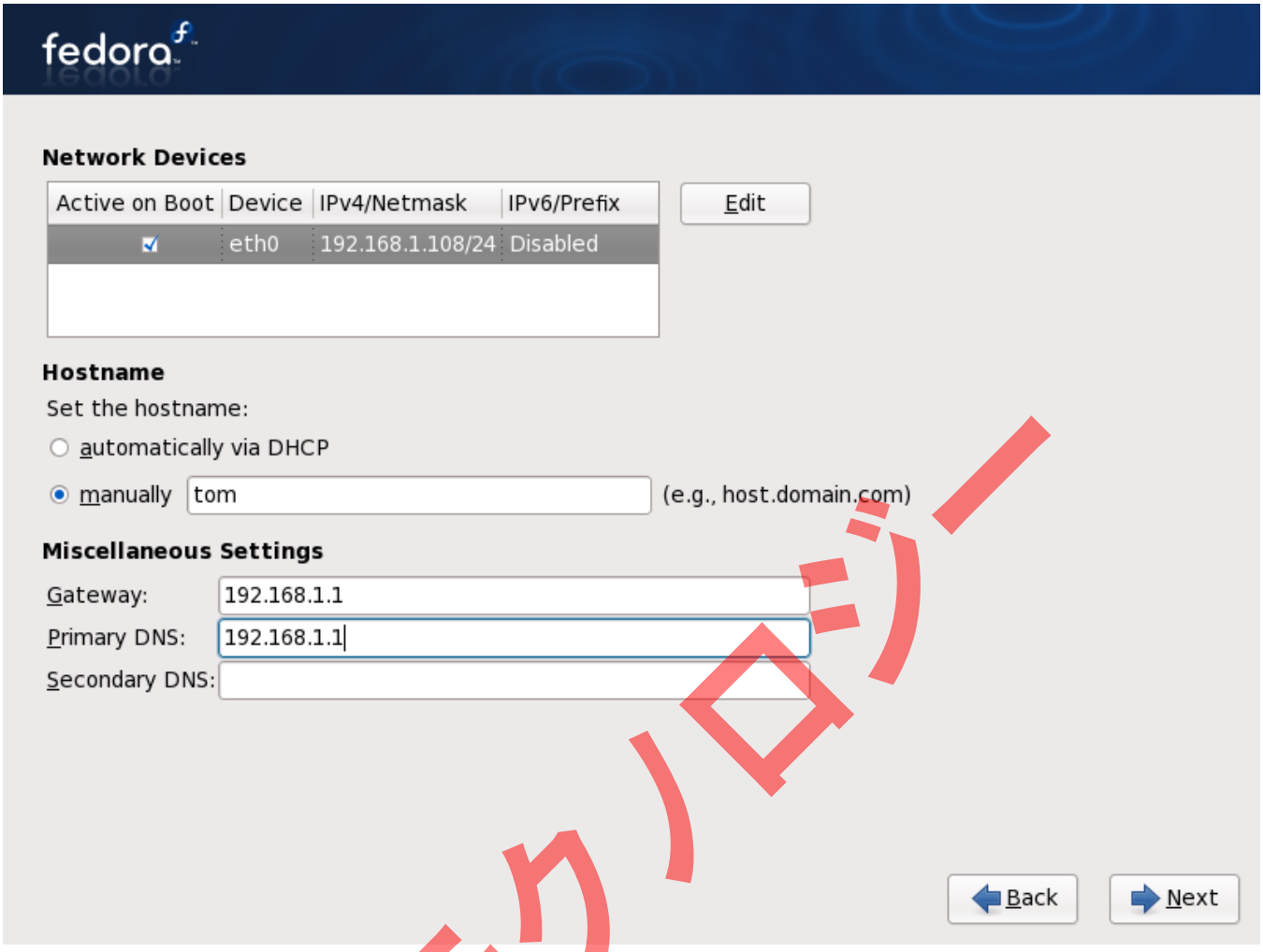
Step6:ネットワークの設定



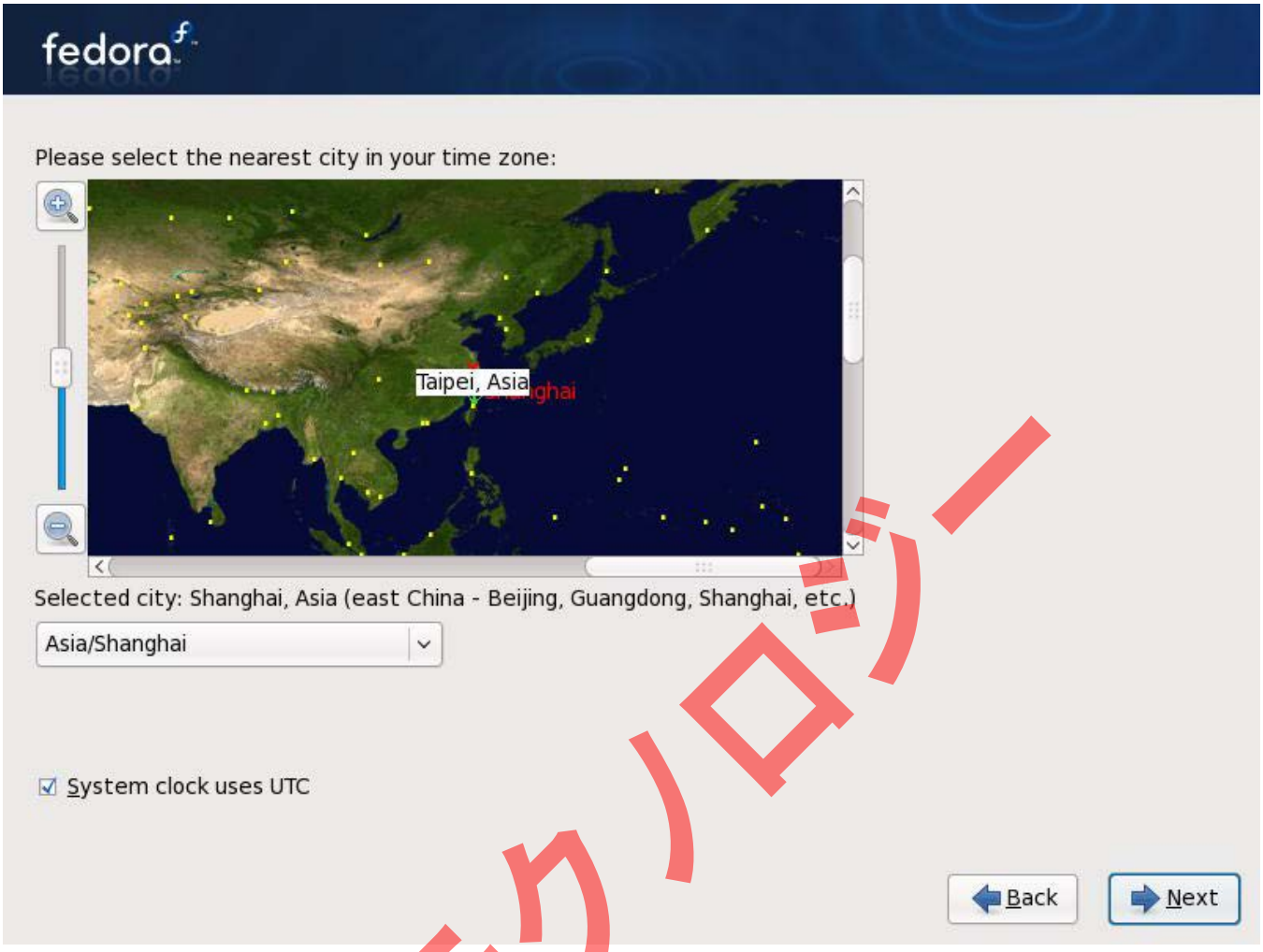
「Edit」をクリックし、DHCP に設定しないでください。通常はスタティック IP アドレスを使用し、下図を参照して IP アドレスとサブネットマスクを入力する：



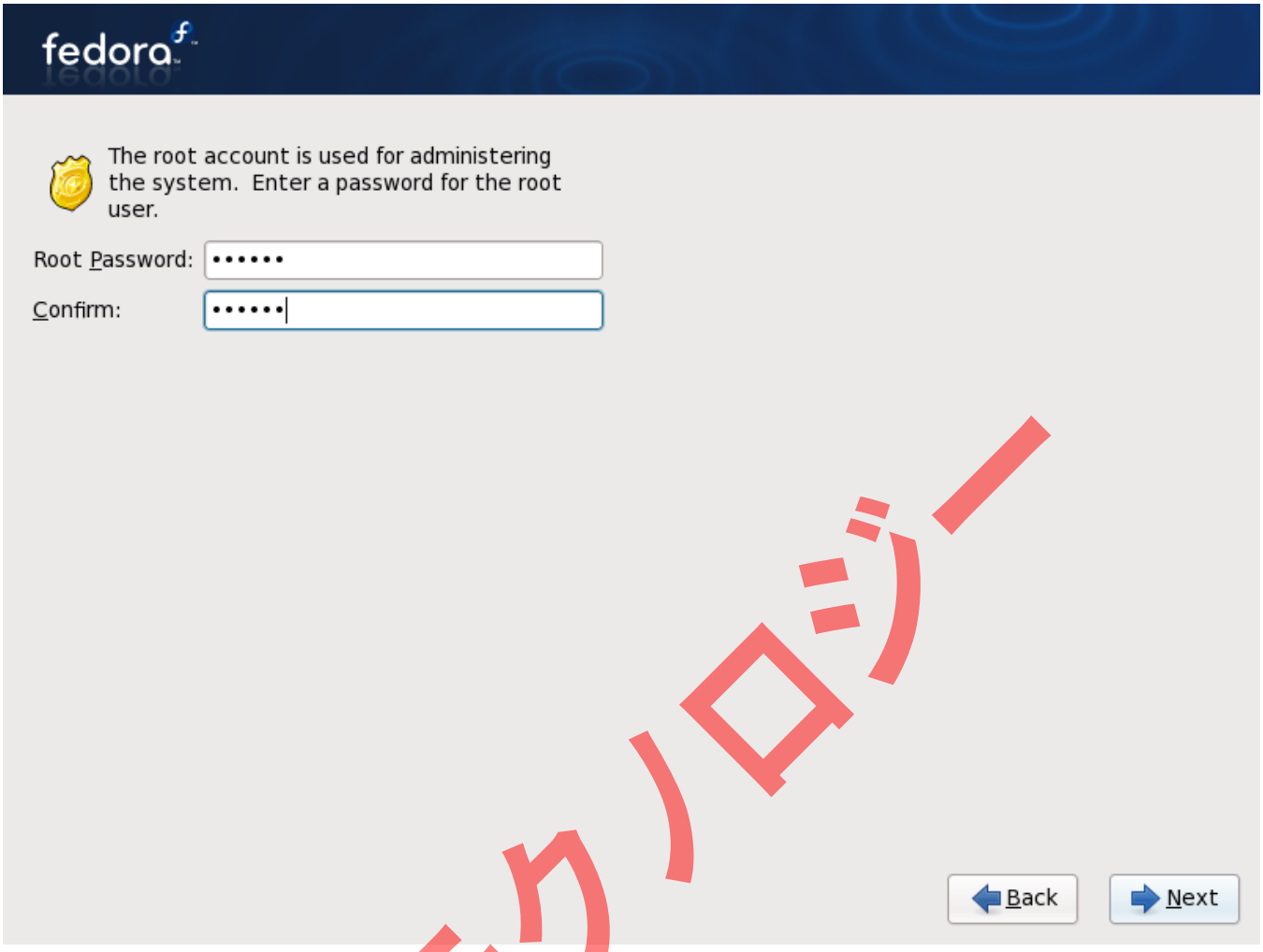
「OK」をクリックし、ゲートウェイと DNS などを設定する：



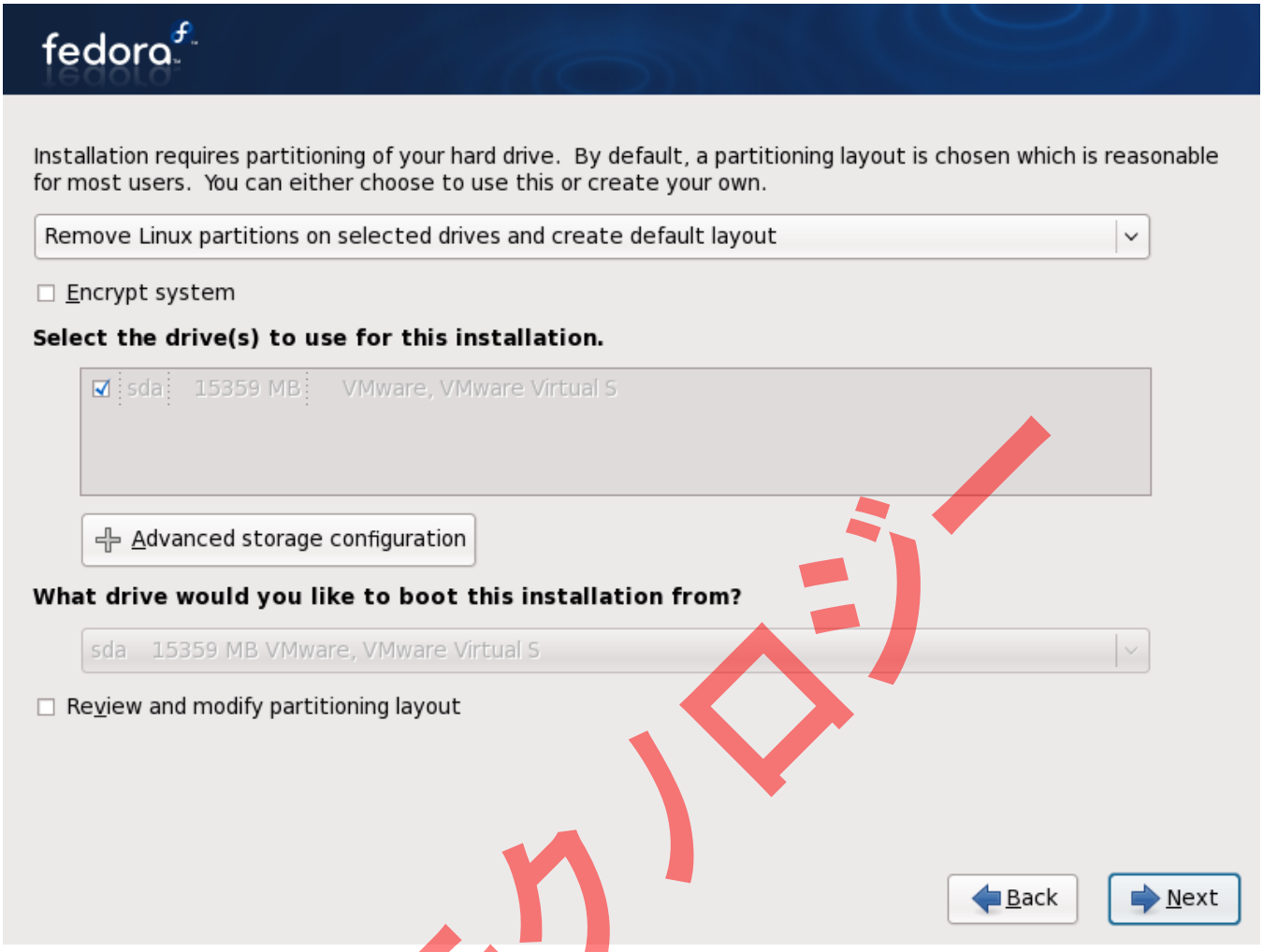
Step7：タイムゾーンを設定する。バーチャルマシンを使用しない場合、「System clock uses UTC」をチェックしなくてもいい：



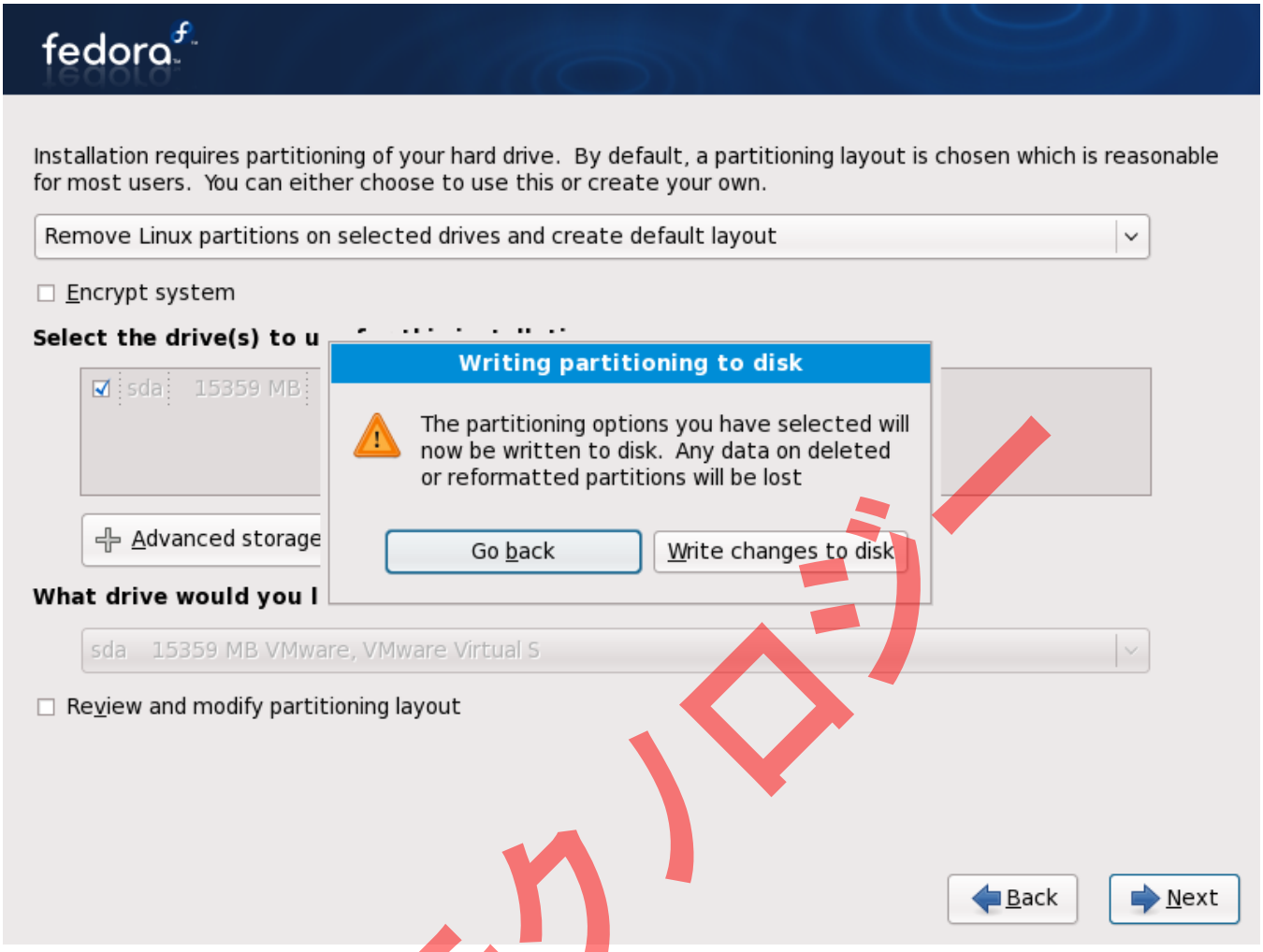
Step8：root ユーザーパスワードを設定し、６キャラクター以上に設定する必要がある；



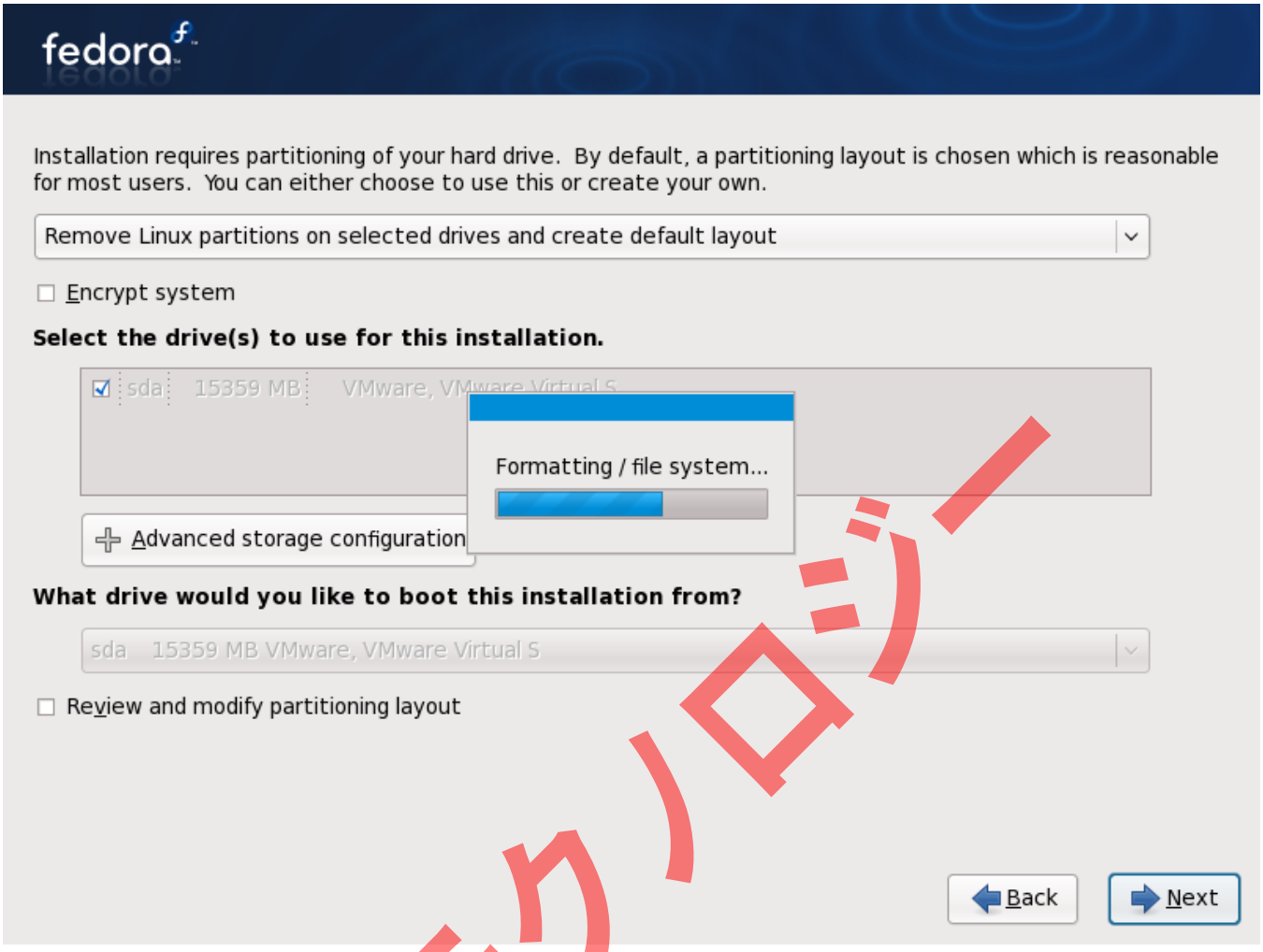
Step9：パーティションを設定する、通常はデフォルトを選択する。ハードディスクのデータをバックアップしてください。



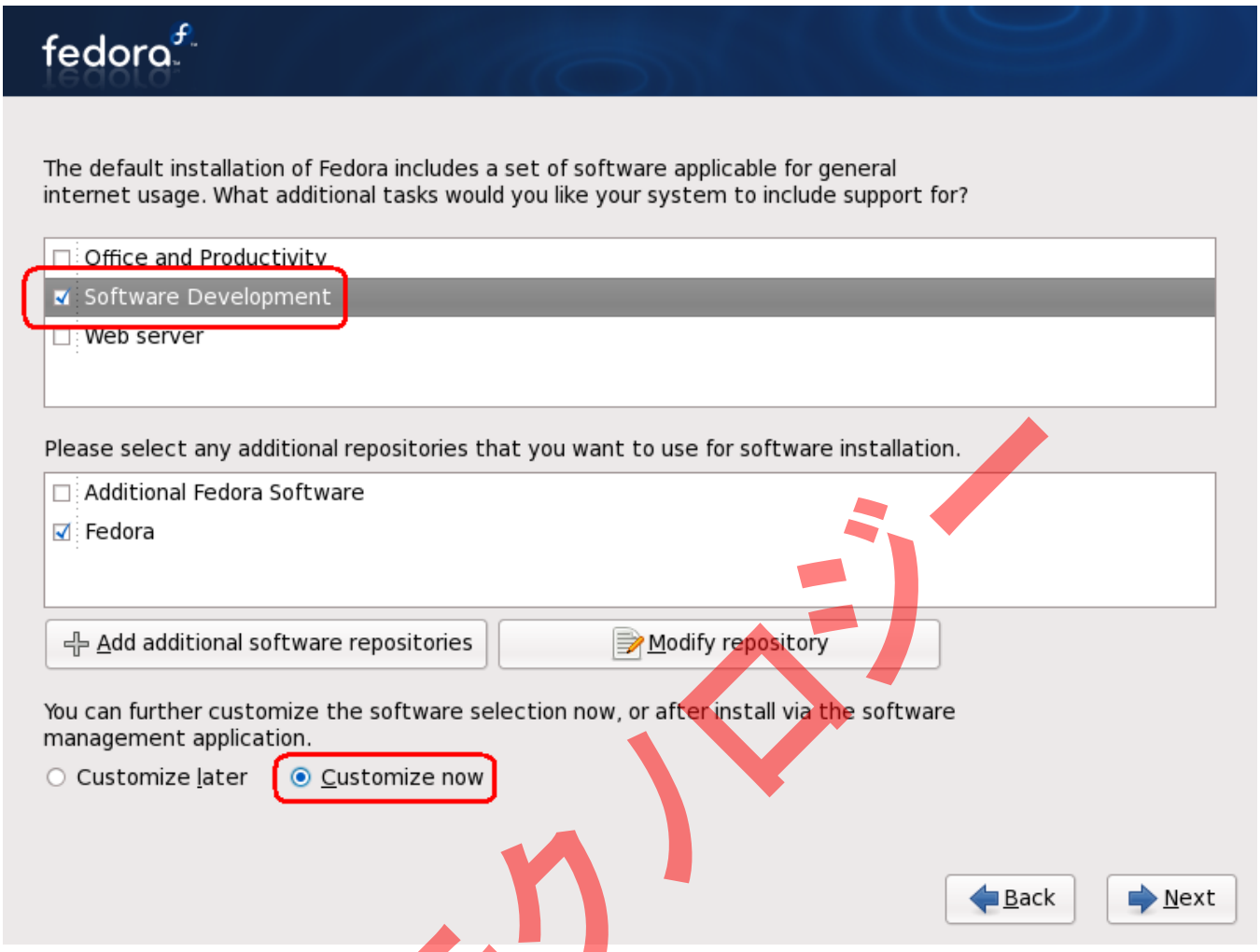
「Next」をクリックすると、フォーマットの警告が出てくる。通常は Vmware バーチャルマシンを使用するため、「Write changes to disk」を選択し、フォーマットする。



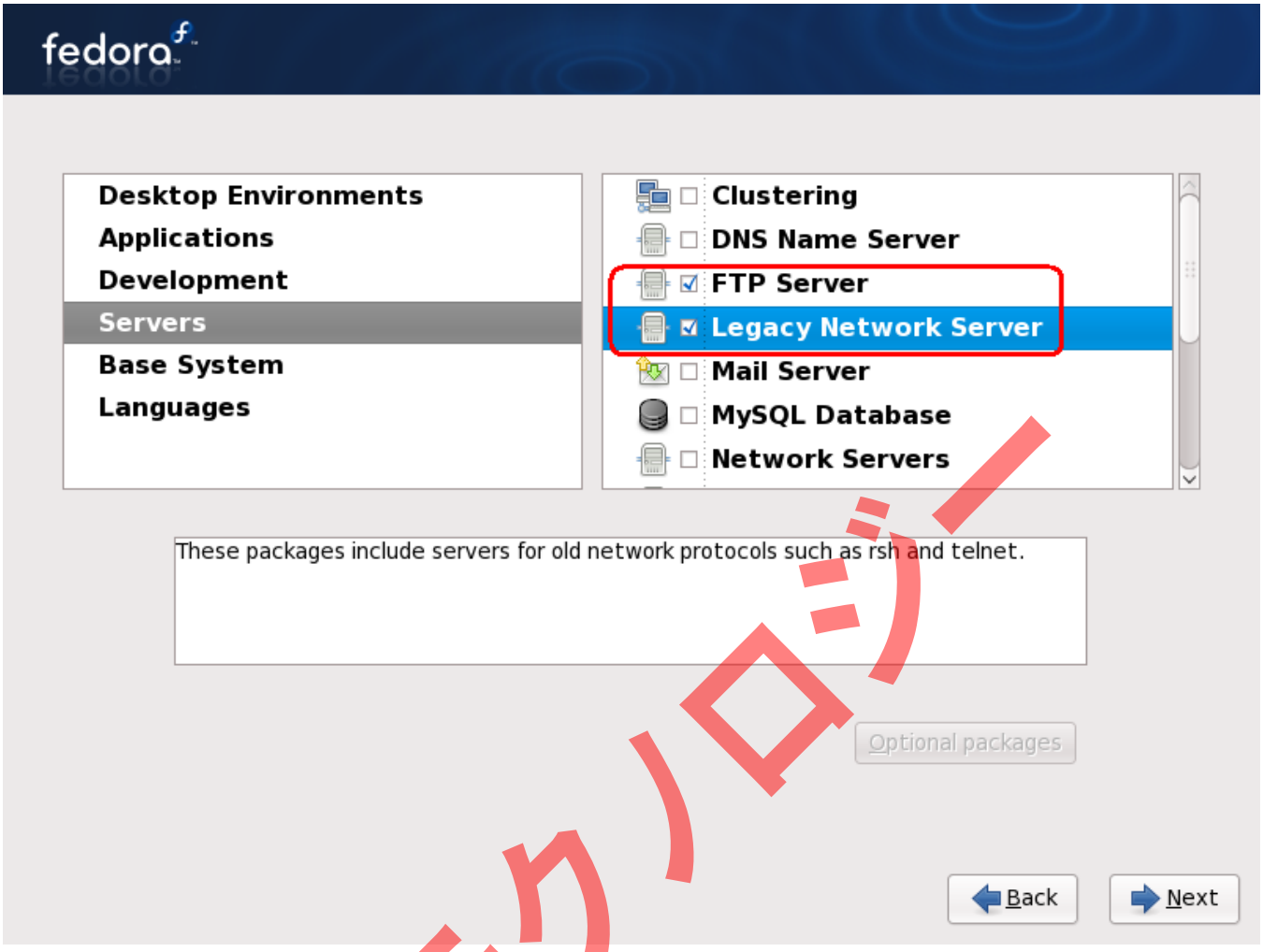
フォーマット中：



Step11：下図のようにインストールのモードを選択し、「Next」をクリックする：

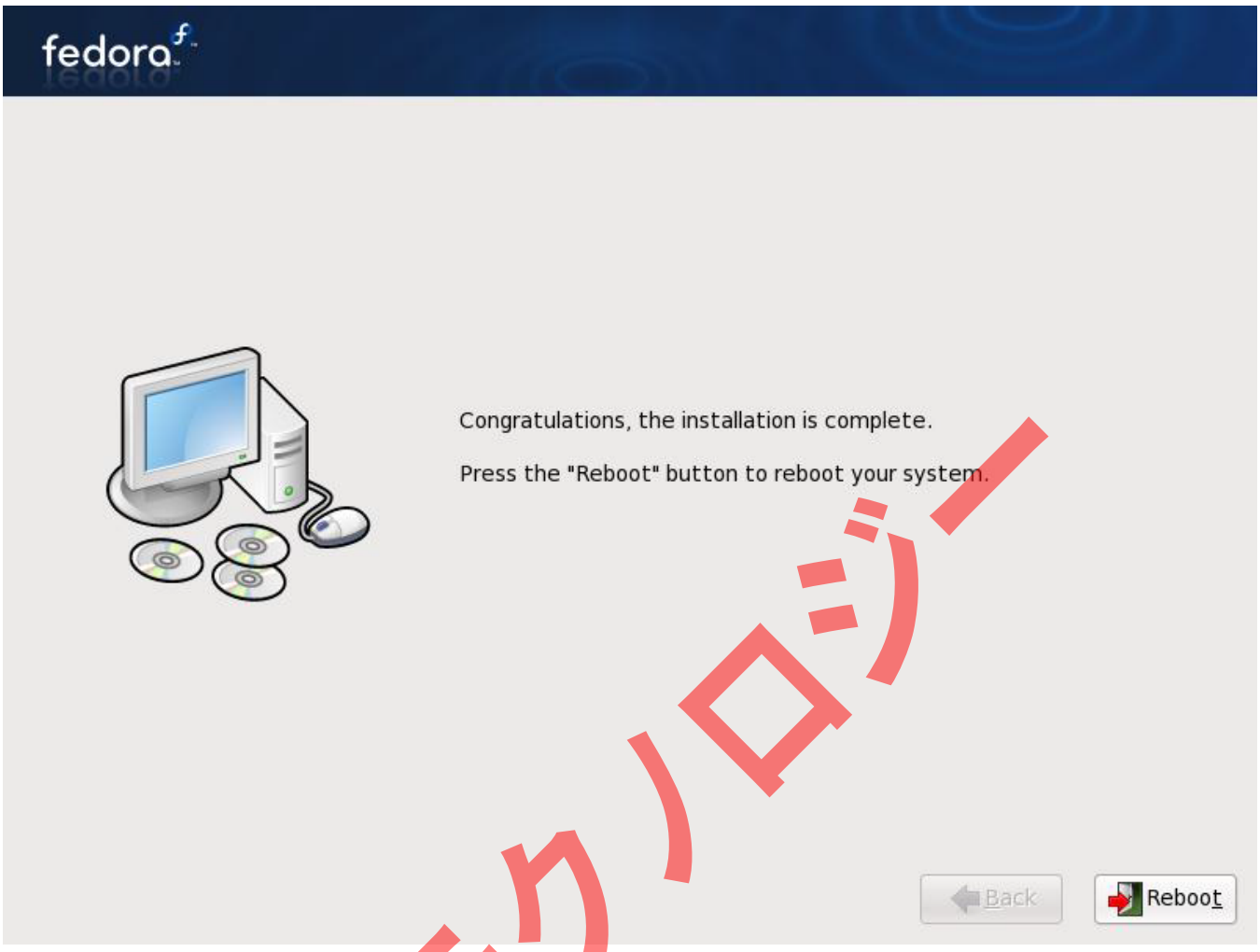


Step12：Servers の選択肢は下図の通り：



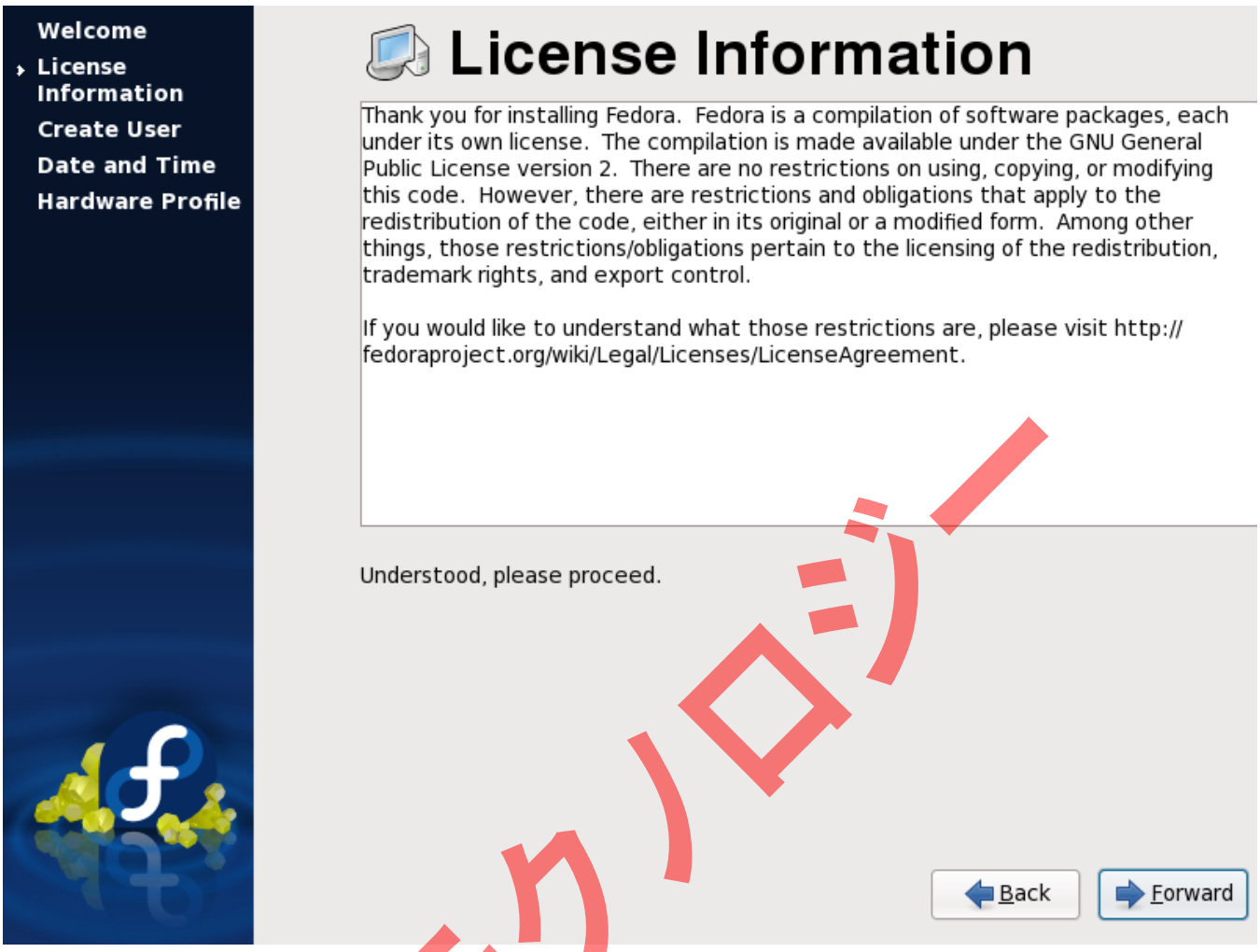
Step13：インストールが始まり、かなりお時間がかかる。

Step14：インストール完了の画面は下図の通り：

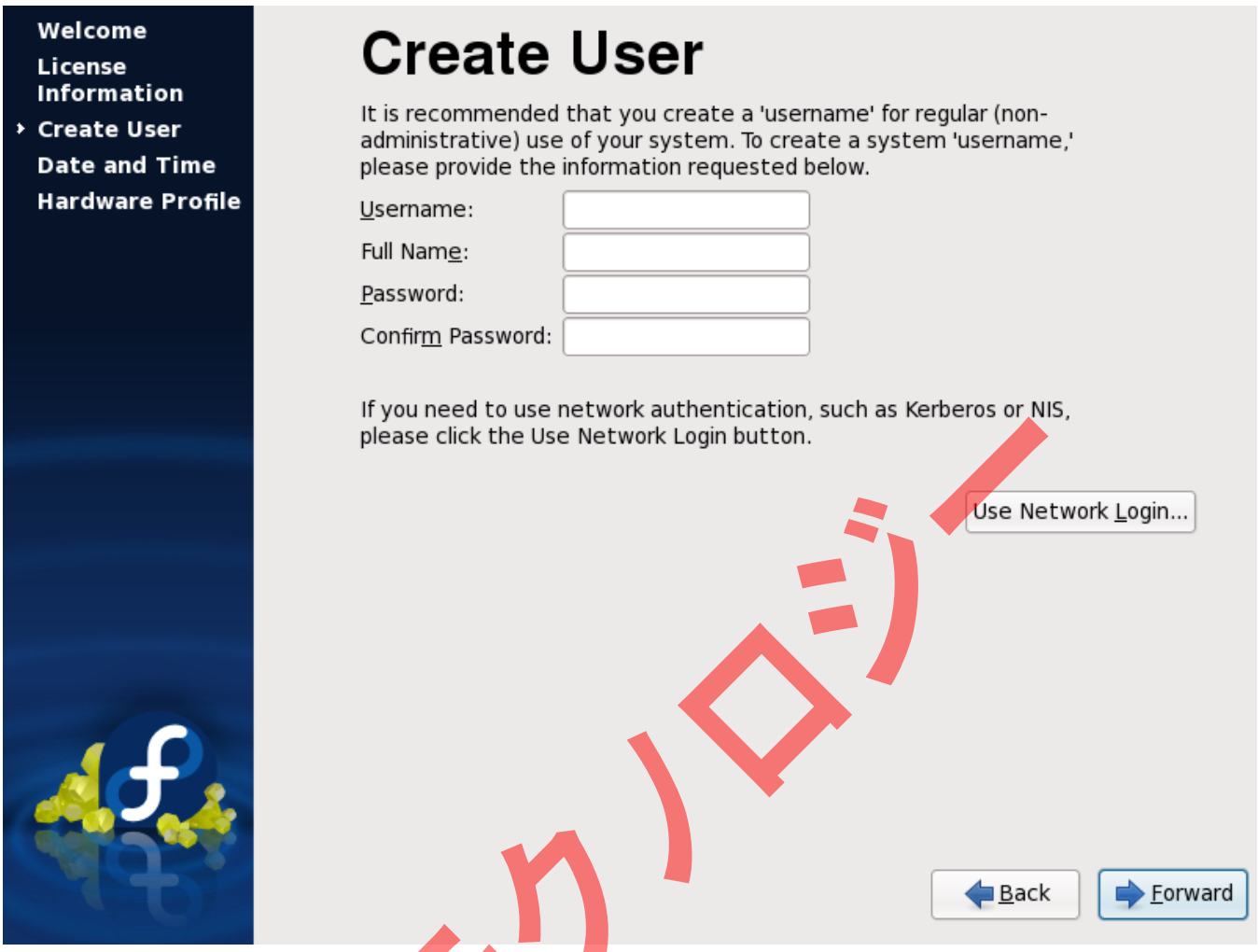


Step15：「Reboot」をクリックしてシステムをリスタートし、下図のように初期化の画面が出てくる：

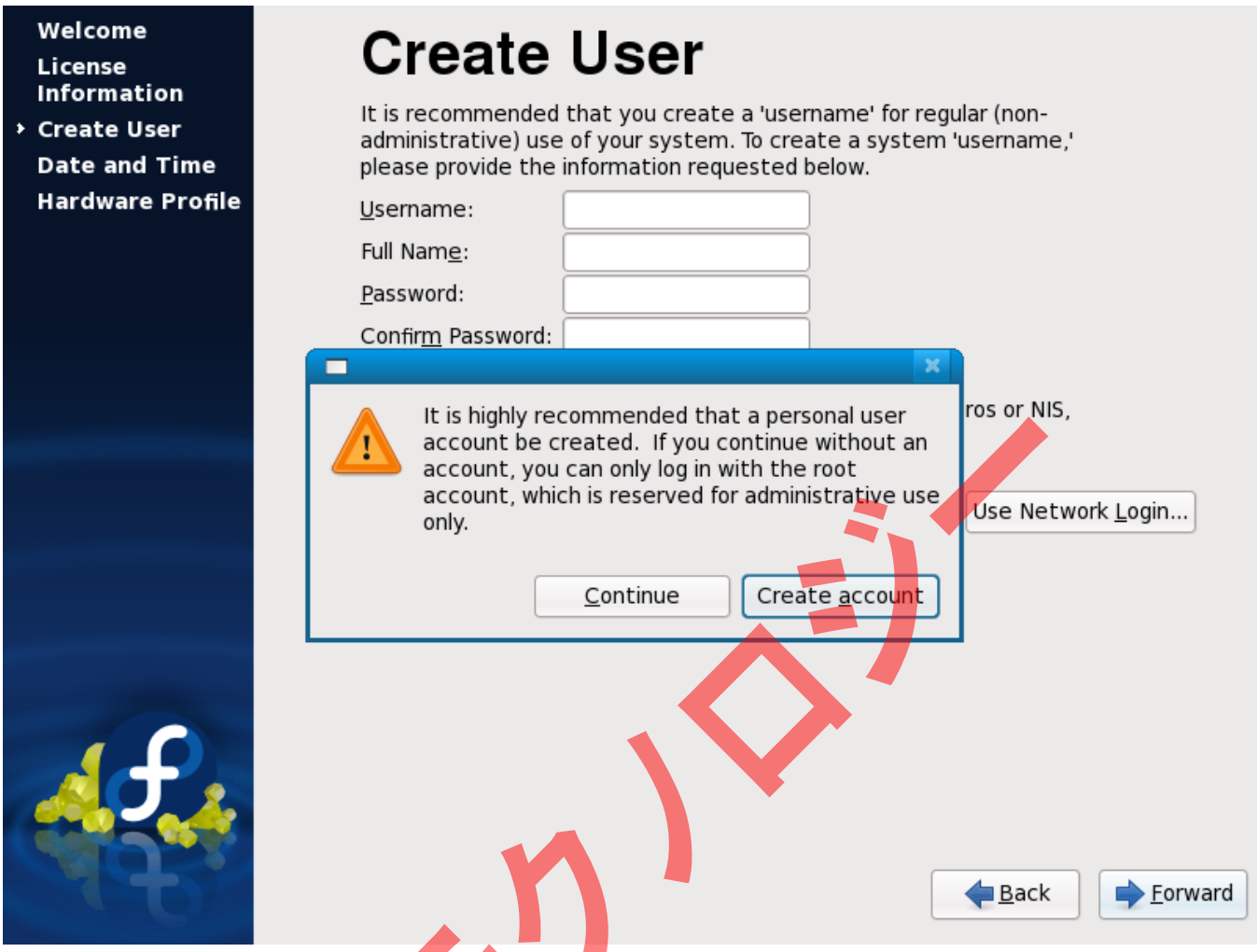
Step16：ライセンス情報が出てきて、そのまま次へ進む：



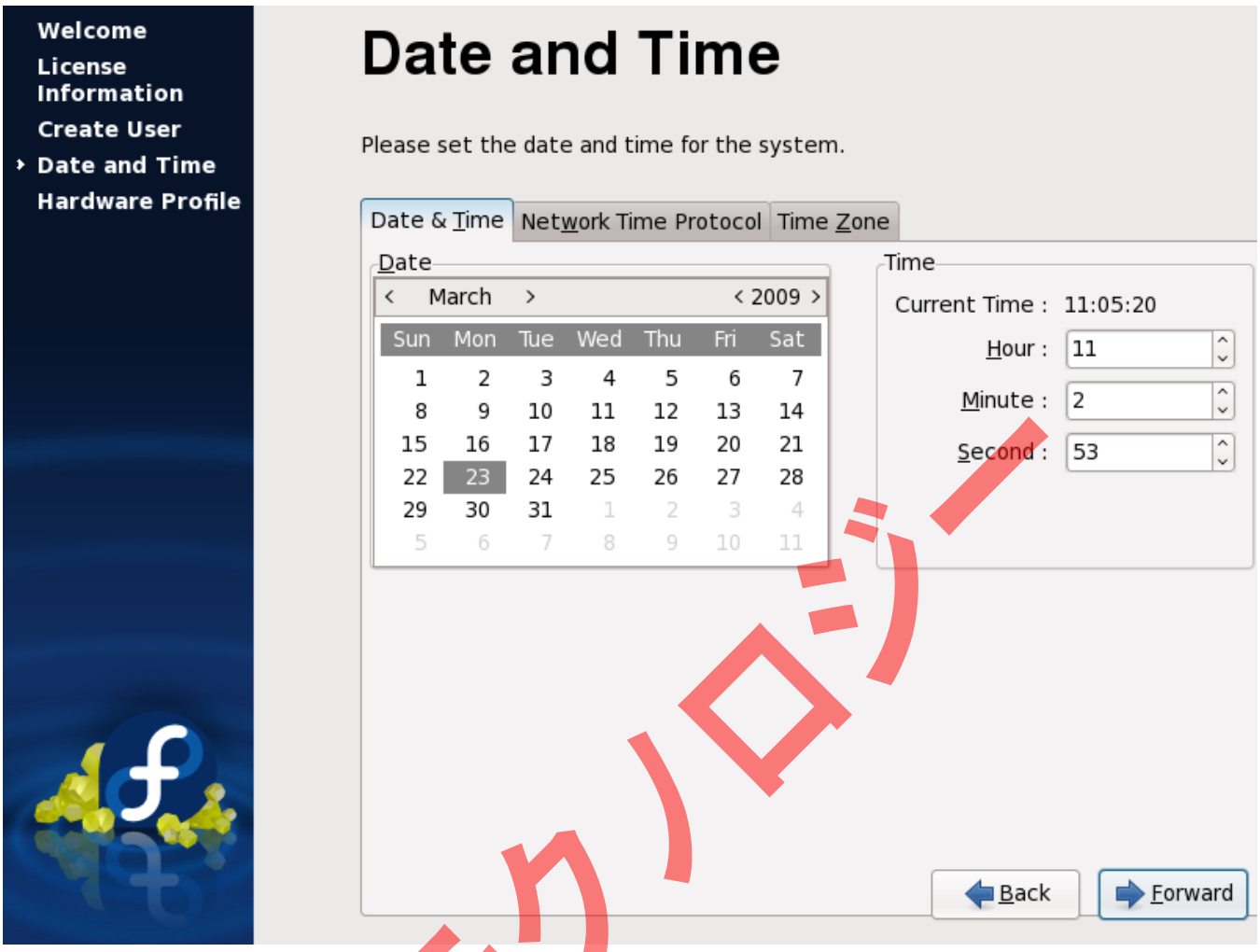
Step17：ユーザーを作成する。新しく作らないので、「Forward」をクリックし次へ進む：



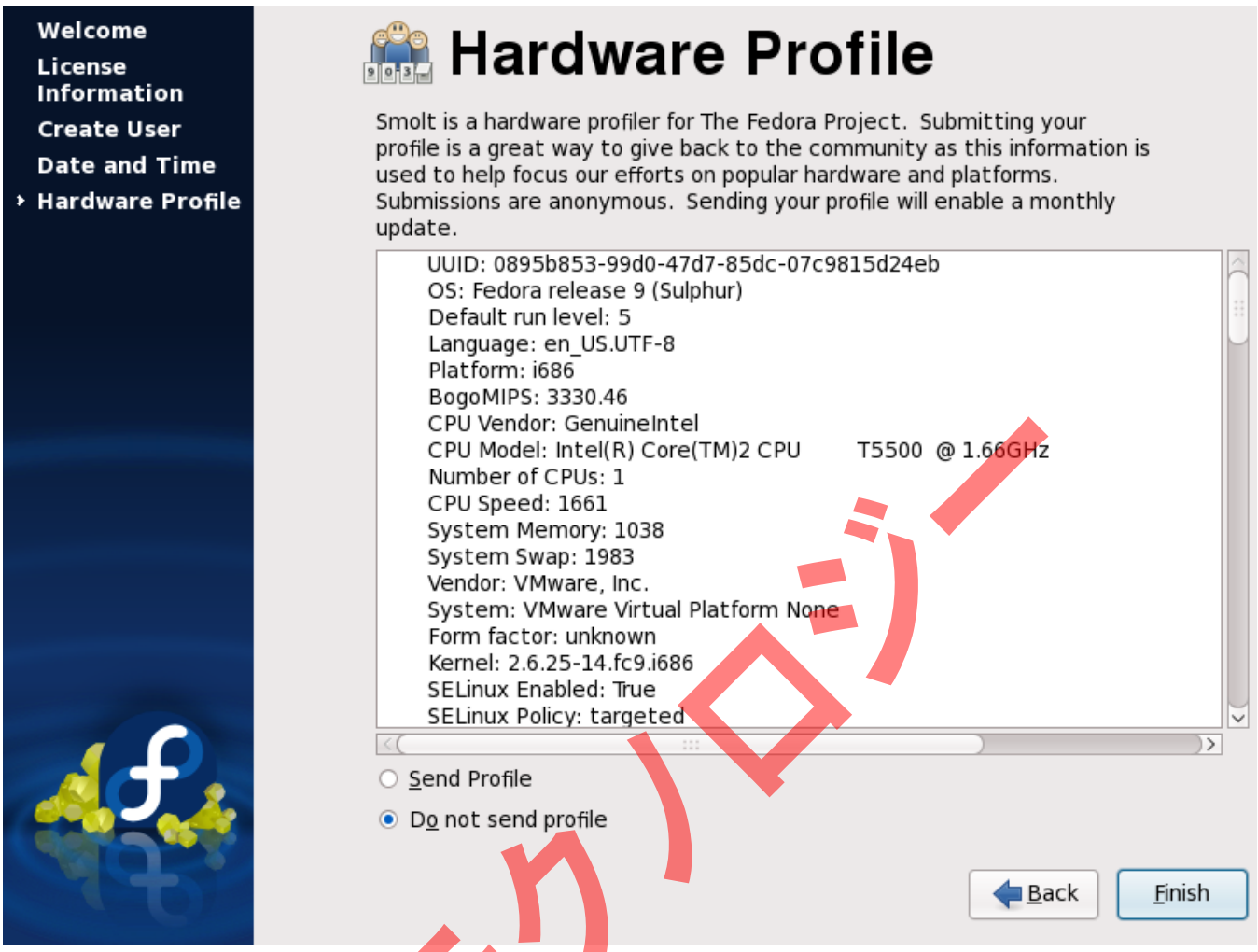
確認画面が出てきて、「Continue」をクリックし次へ進む：



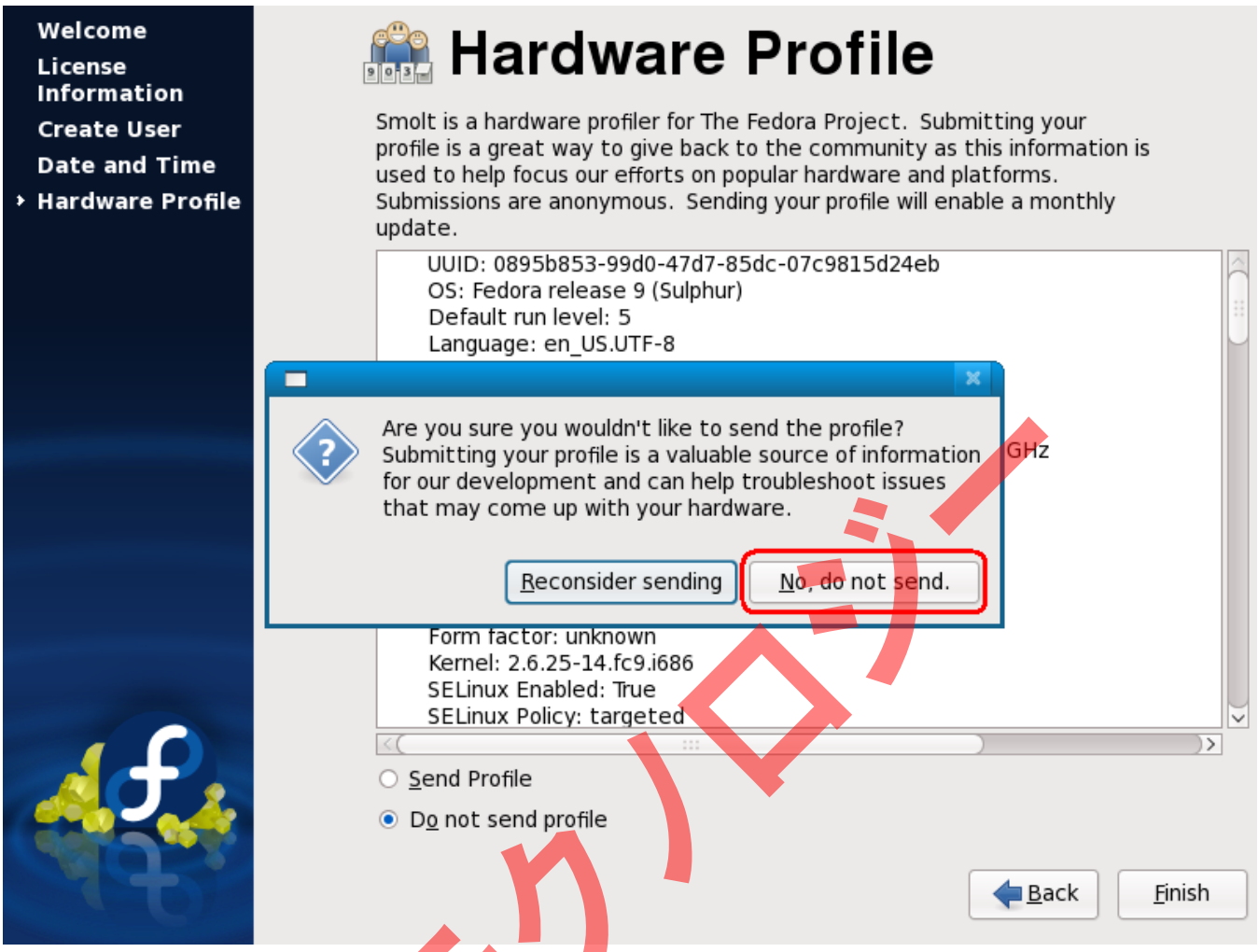
Step18：日付と時刻を設定画面、何もしない、次へ進む：



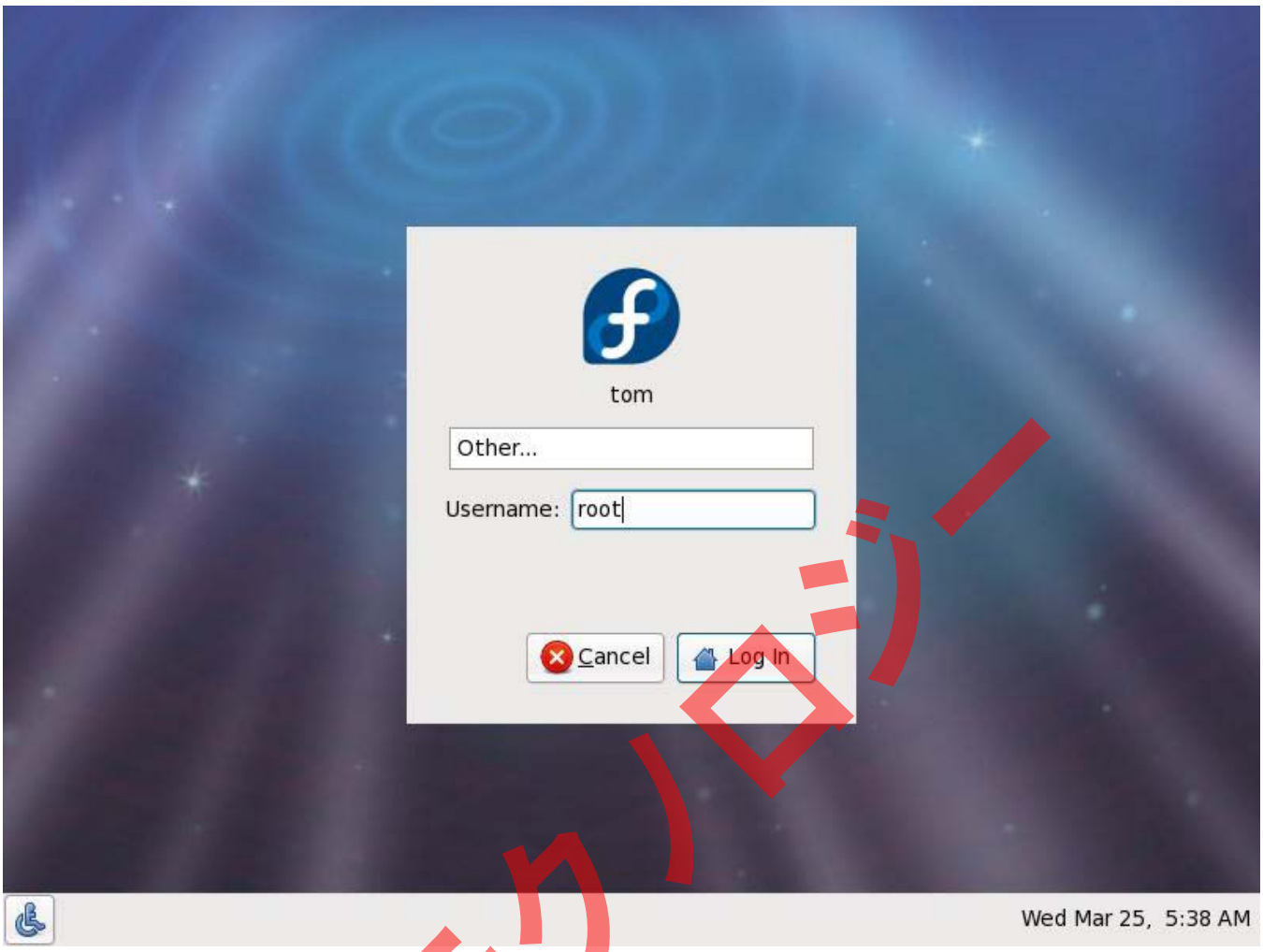
Step19：ハードウェアの情報が出てきて、そのまま「Finish」をクリックする：



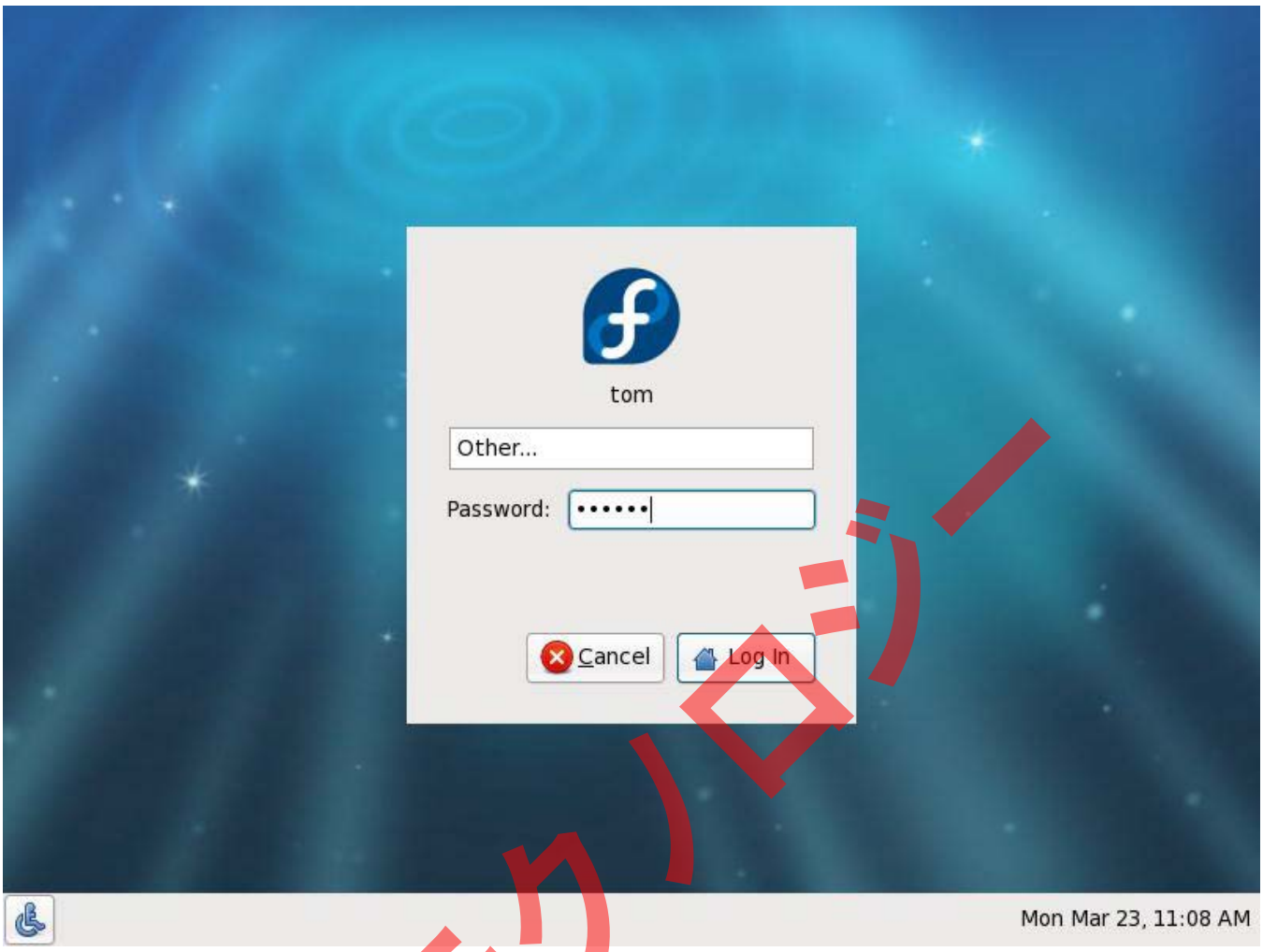
提示画面が出てきて、下図のように選択し、次へ進む：



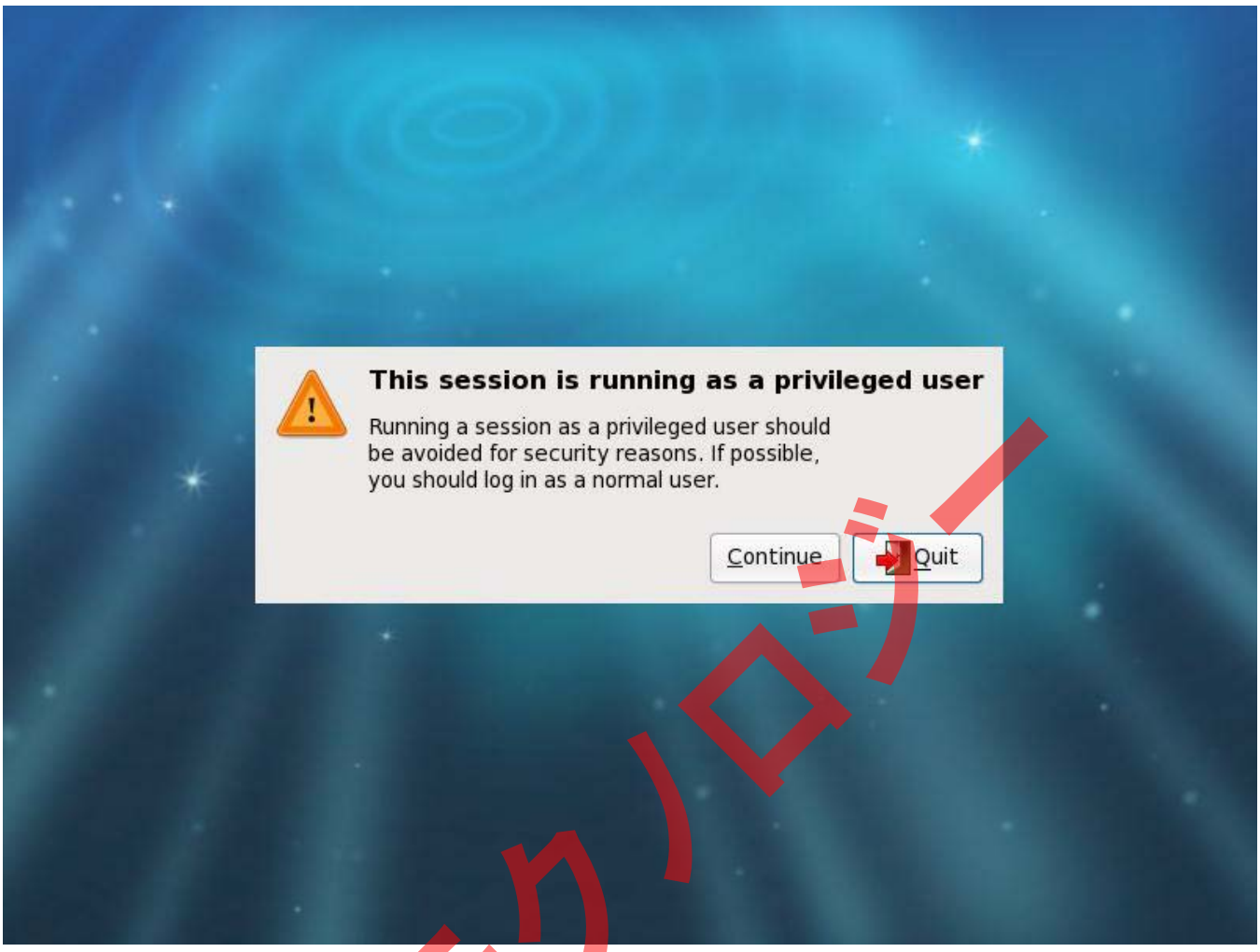
Step20：ログイン画面が出てきて、先ずは root を入力する：



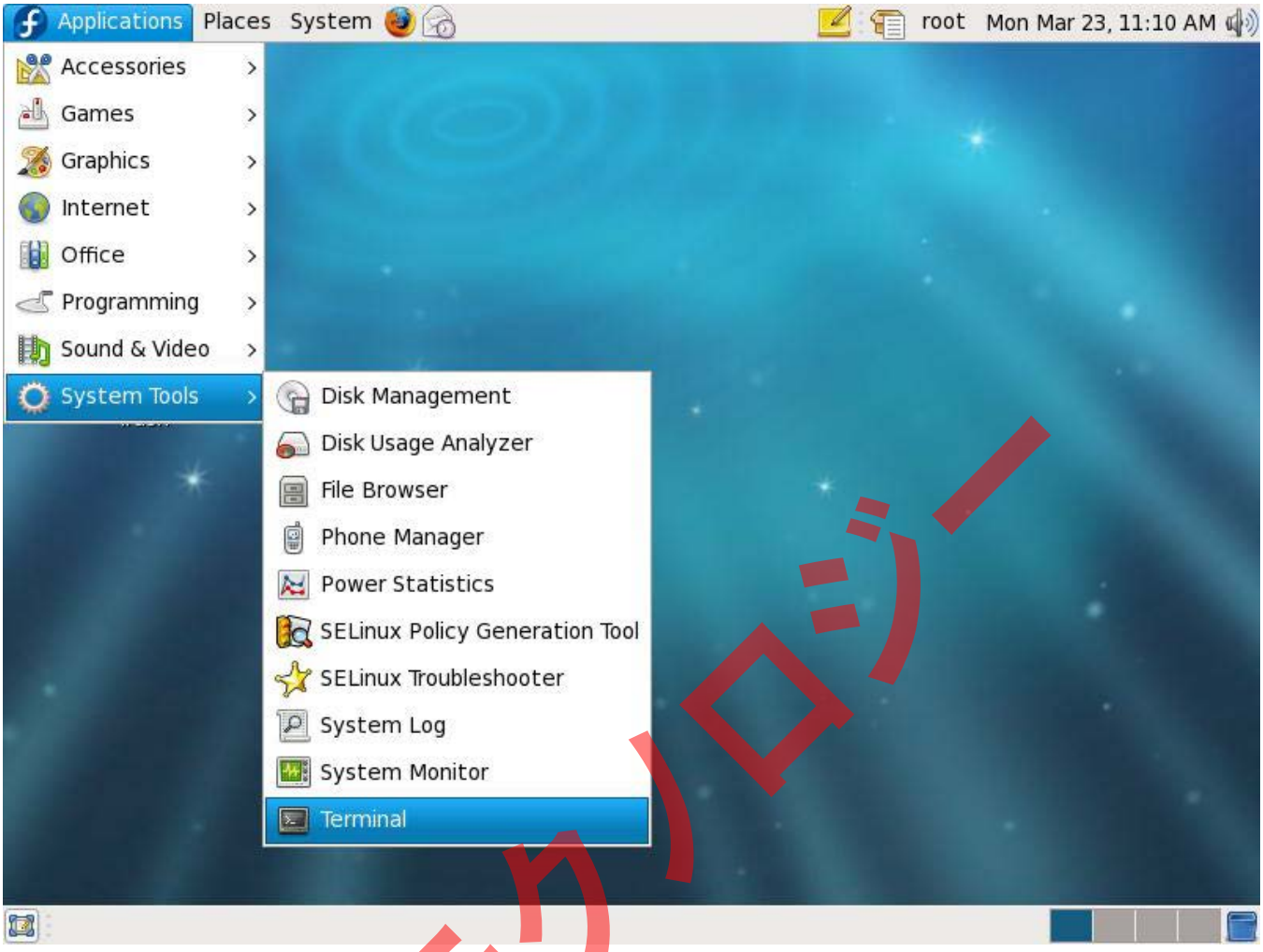
先ほど設定したパスワードを入力する：



ログイン後提示画面が出てきて、root ユーザーでログインすれば、ログインする度提示画面が出る。「Continue」をクリックすればいい。

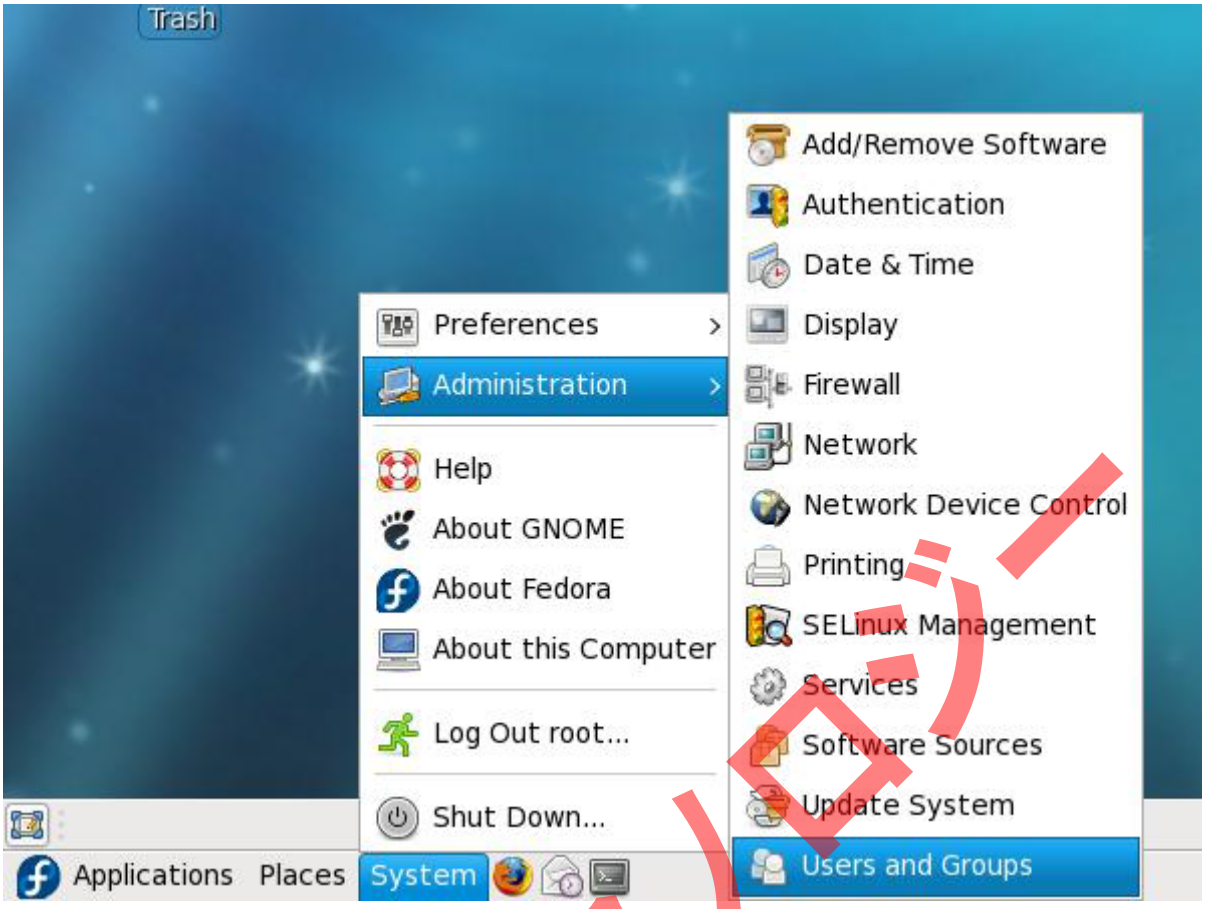


ログイン後の画面は下図の通り。Windows 或いは Ubuntu に類似している：

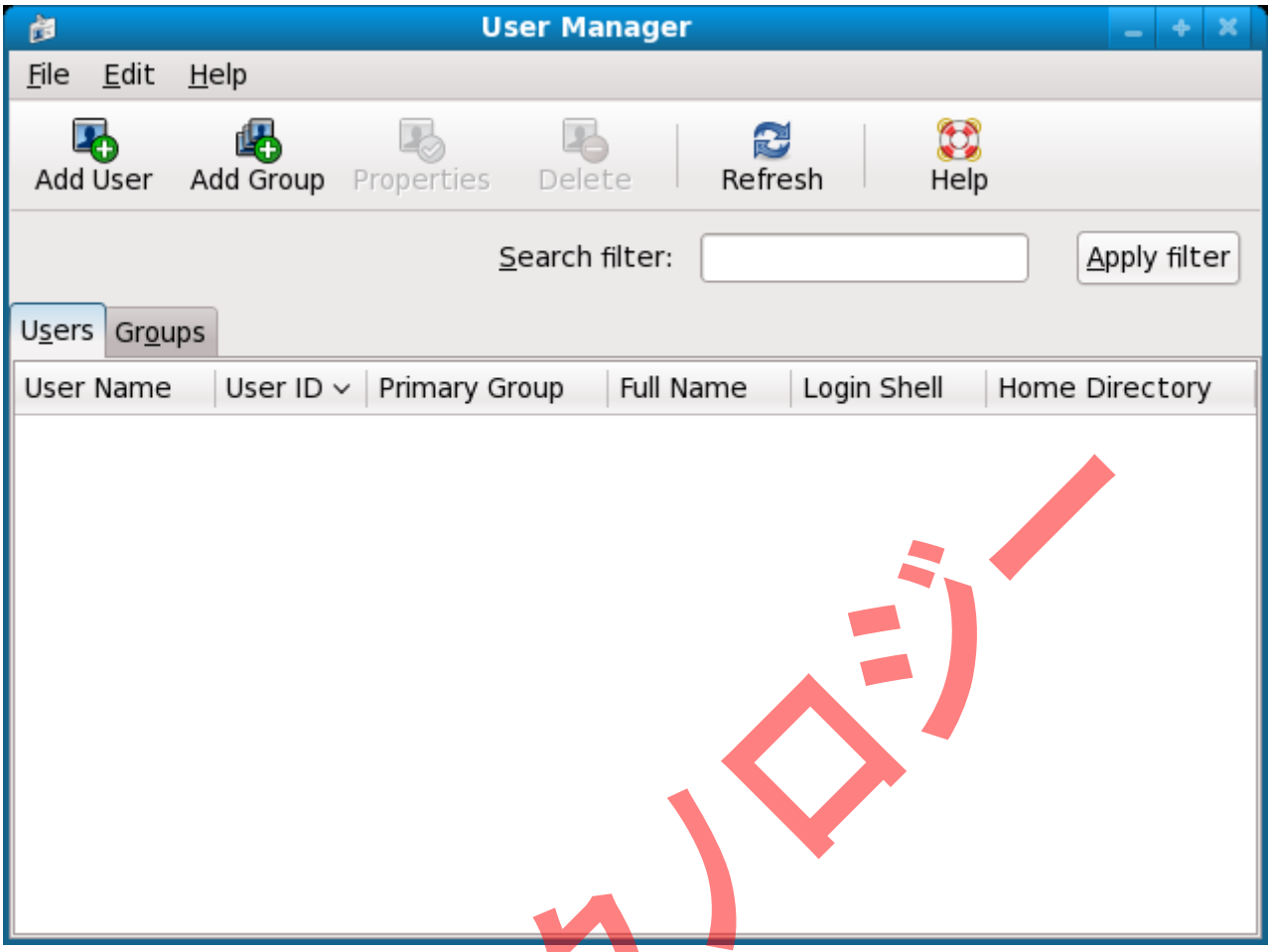


4.3.2 新しいユーザーを作成する

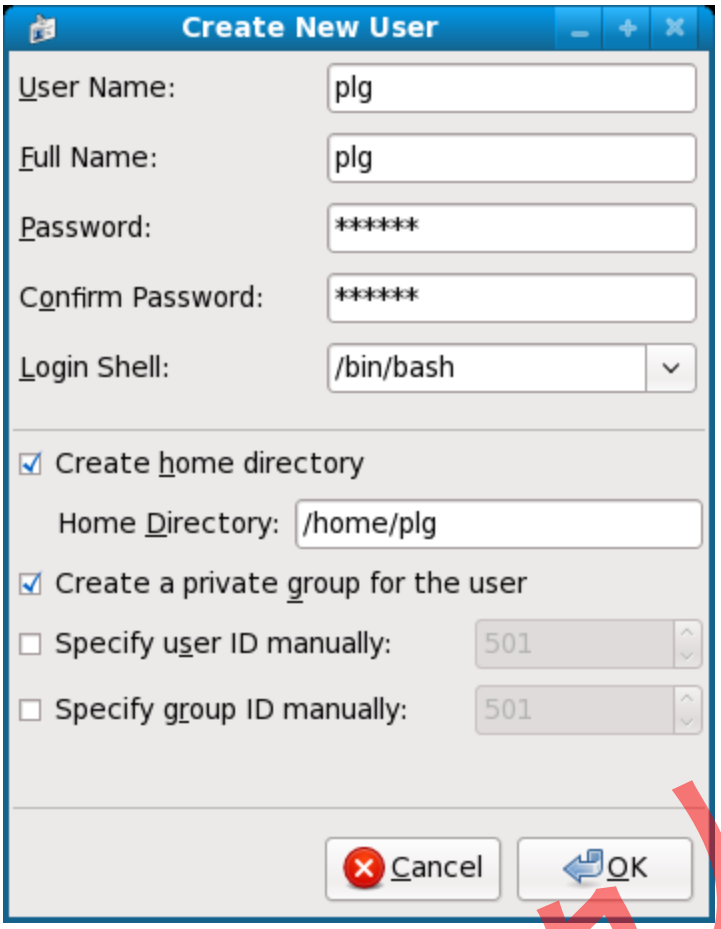
Step1：下図のように「Users and Groups」を開く：



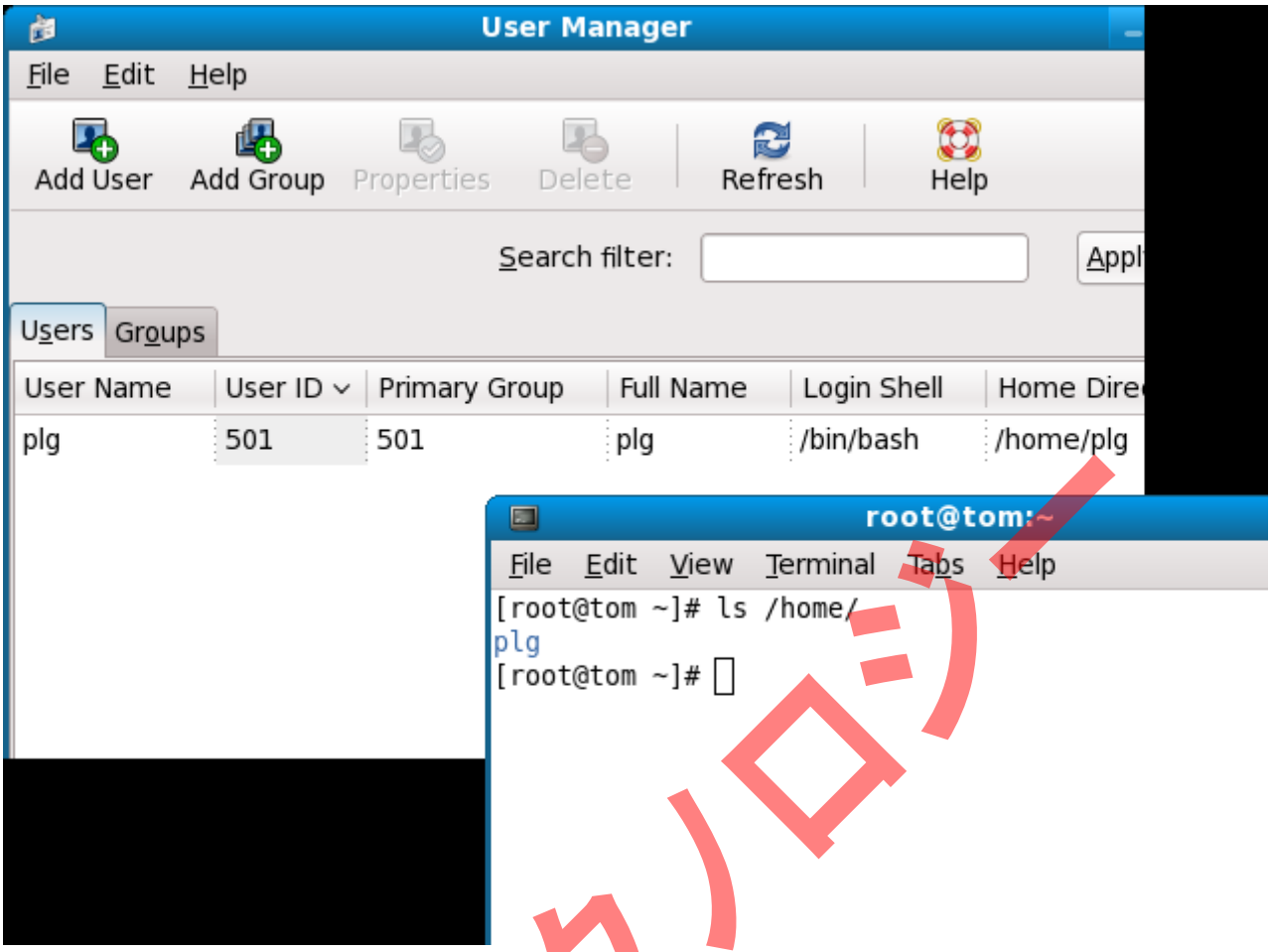
Step2：ユーザー管理画面が出てくる：



Step3：「Add User」をクリックし、新しいユーザーを作成し、パスワードを設定する：



「OK」をクリックし、下図のように plg ユーザーが見られる。それと同時に/home フォルダに plg ユーザーフォルダも出てくる。

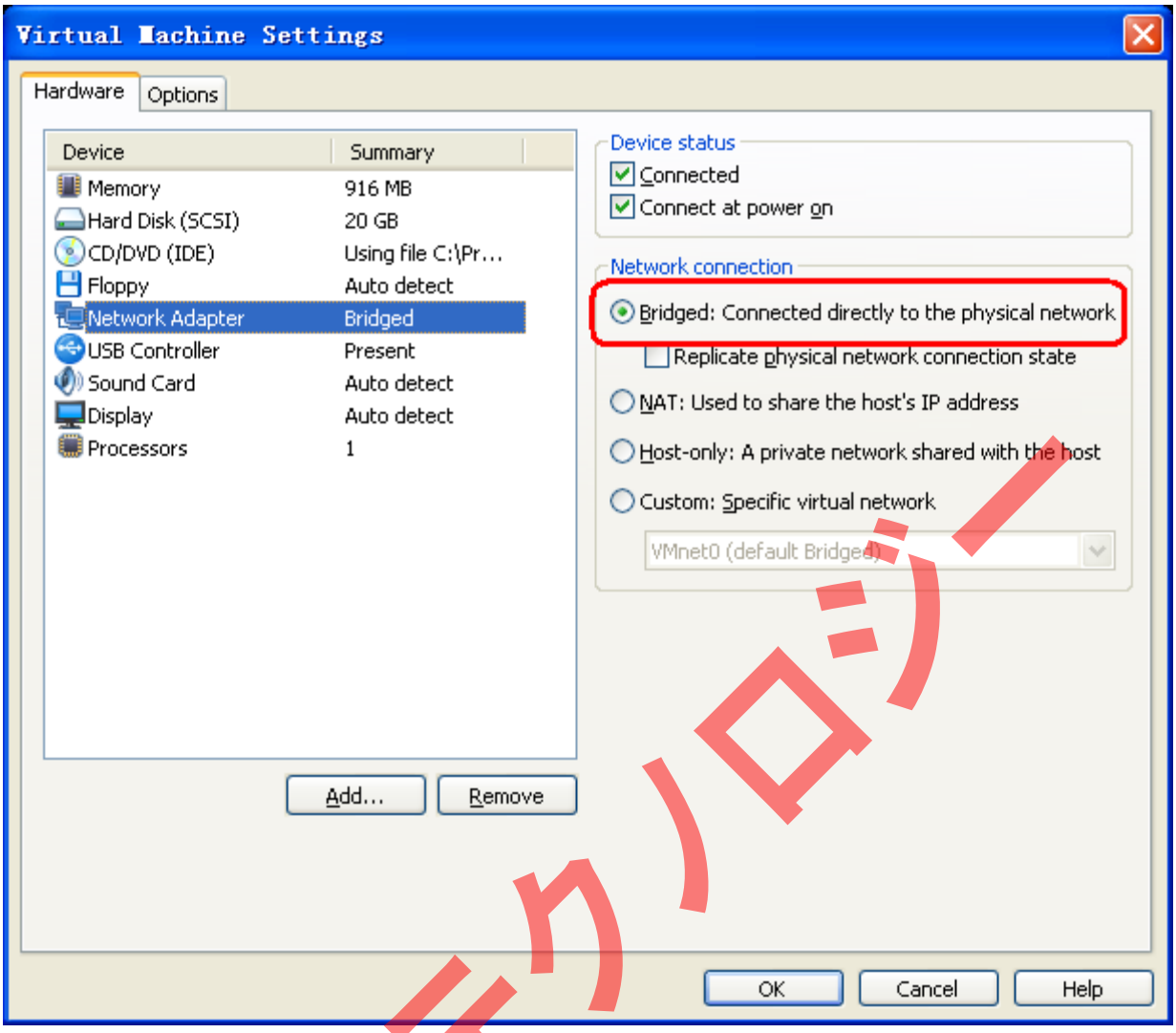


Add User をクリックし、提示に従って操作すればいい。

4.3.3 Windows システムのファイルにアクセスする

二つのシステムのネットワークが相互通信できれば、便利に Windows システムの共有ファイルにアクセスできる。

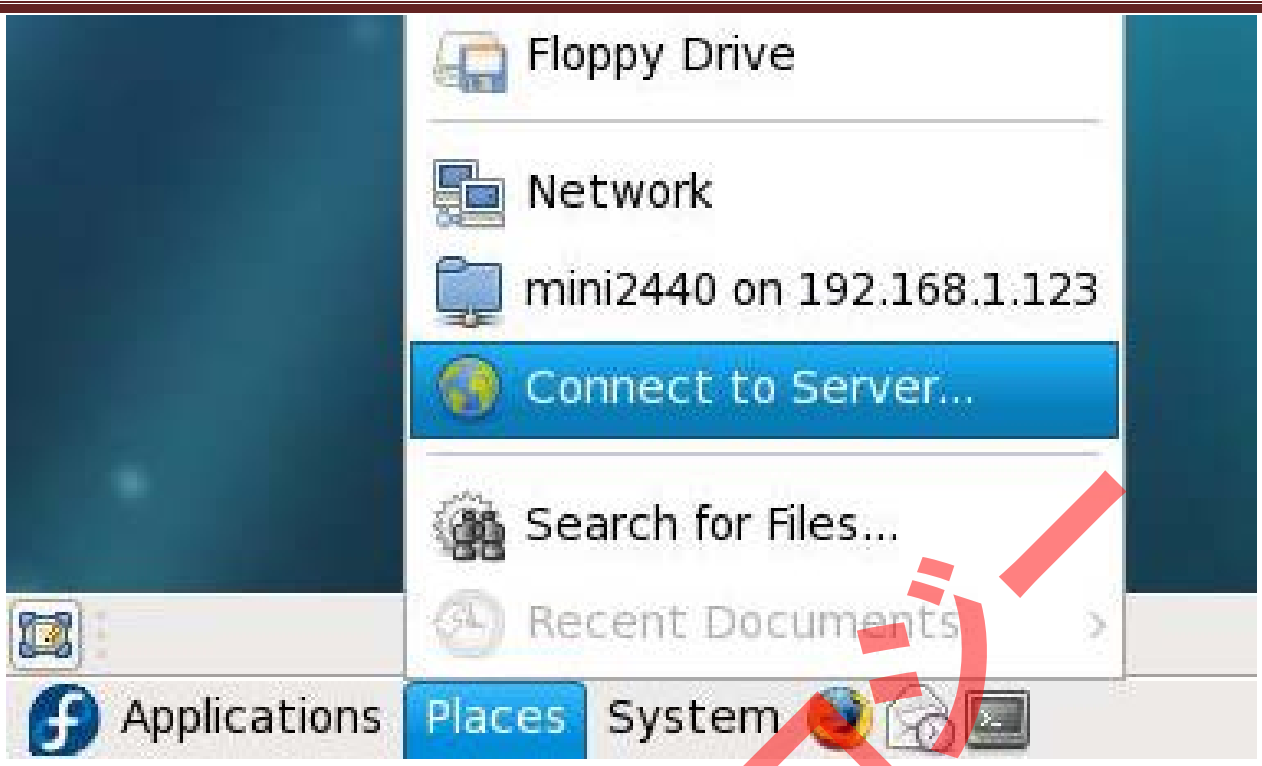
バーチャルマシンでネットワークを使用する場合、一番簡単な方法として下図のように“Guest”を“Bridges”に設定すればいい：



Windows システムの共有ファイルにアクセスする手順は下記の通り：

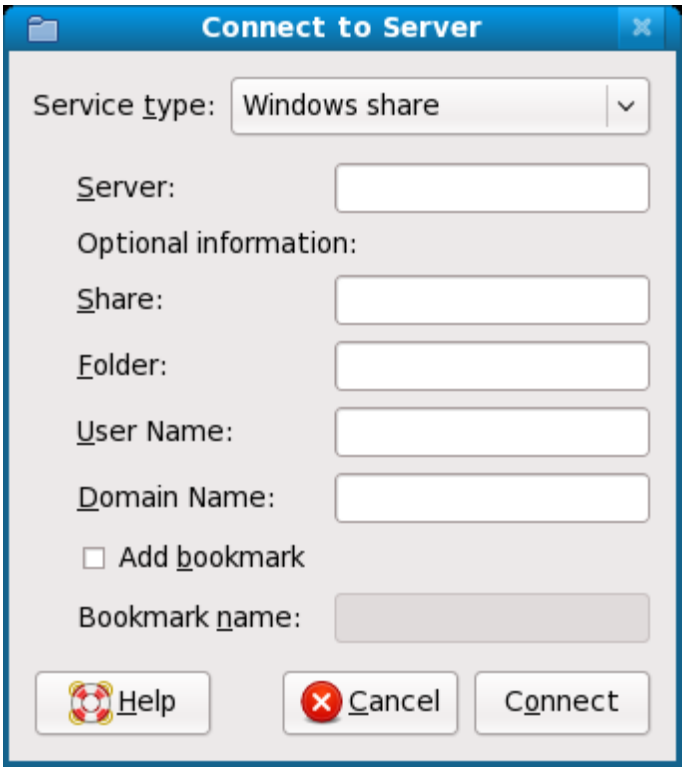
Step1：Windows に共有フォルダ「share_f9」（例）を設定する：

Step2：Fedora9 に下図の操作を行う：

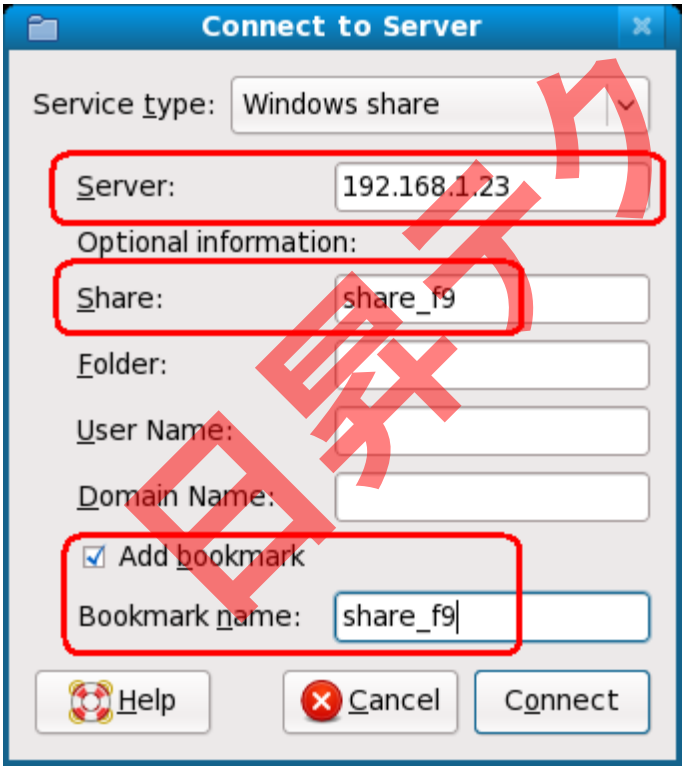


下図の画面を開く：

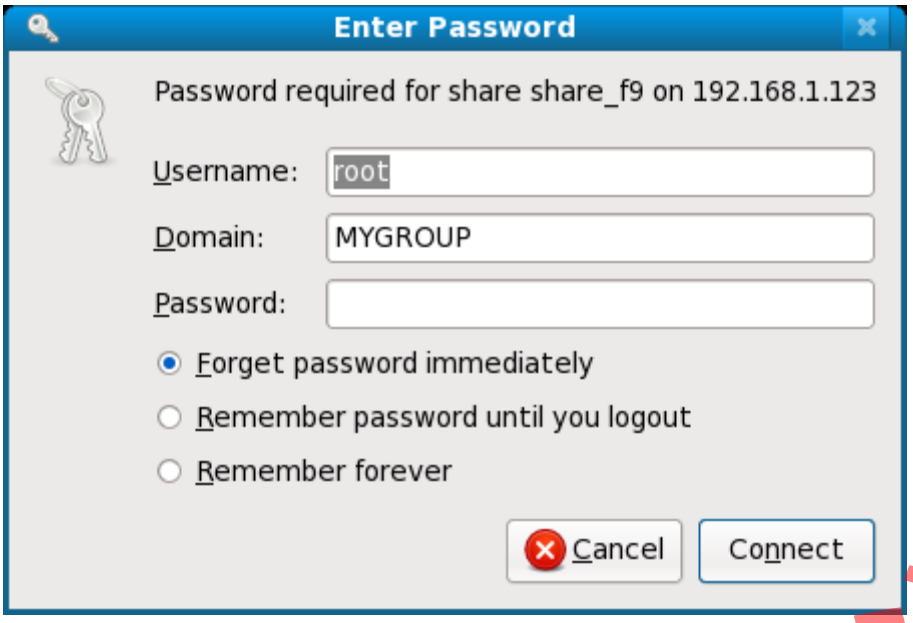
Service type リストに Windows share を選択し、下図の画面が出てくる：



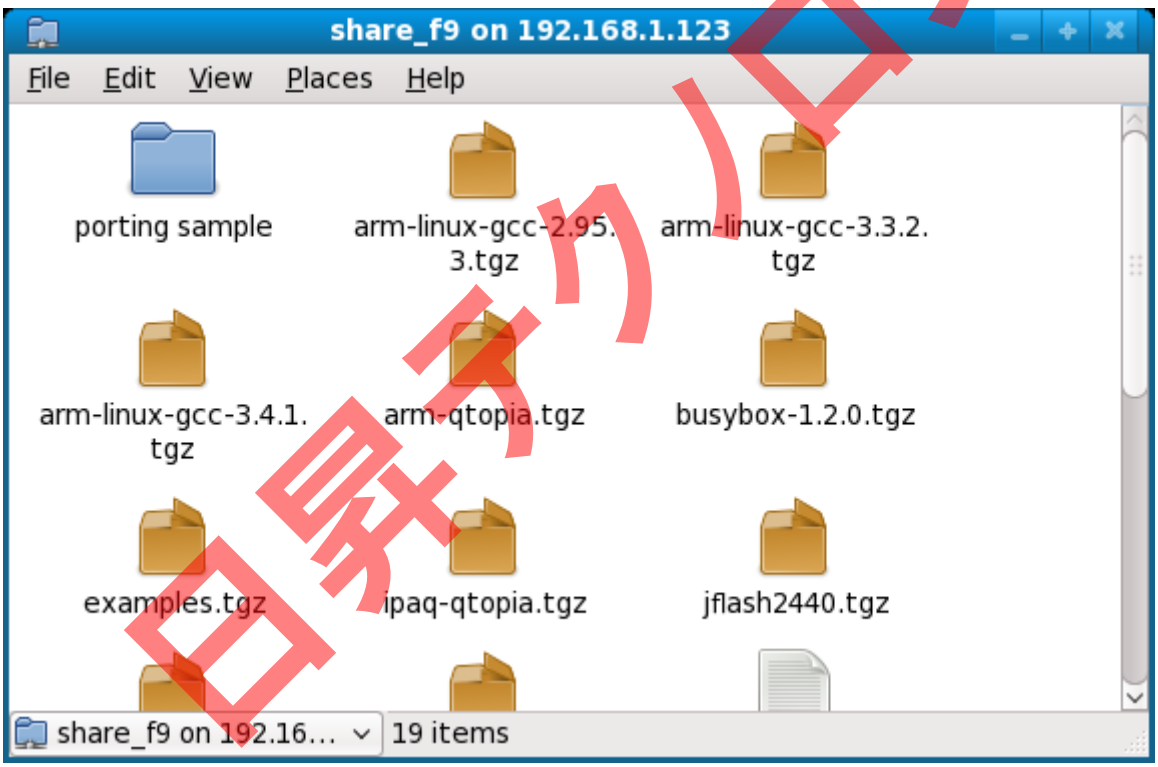
共有する Windows ホストの IP アドレスと共有フォルダを入力する：



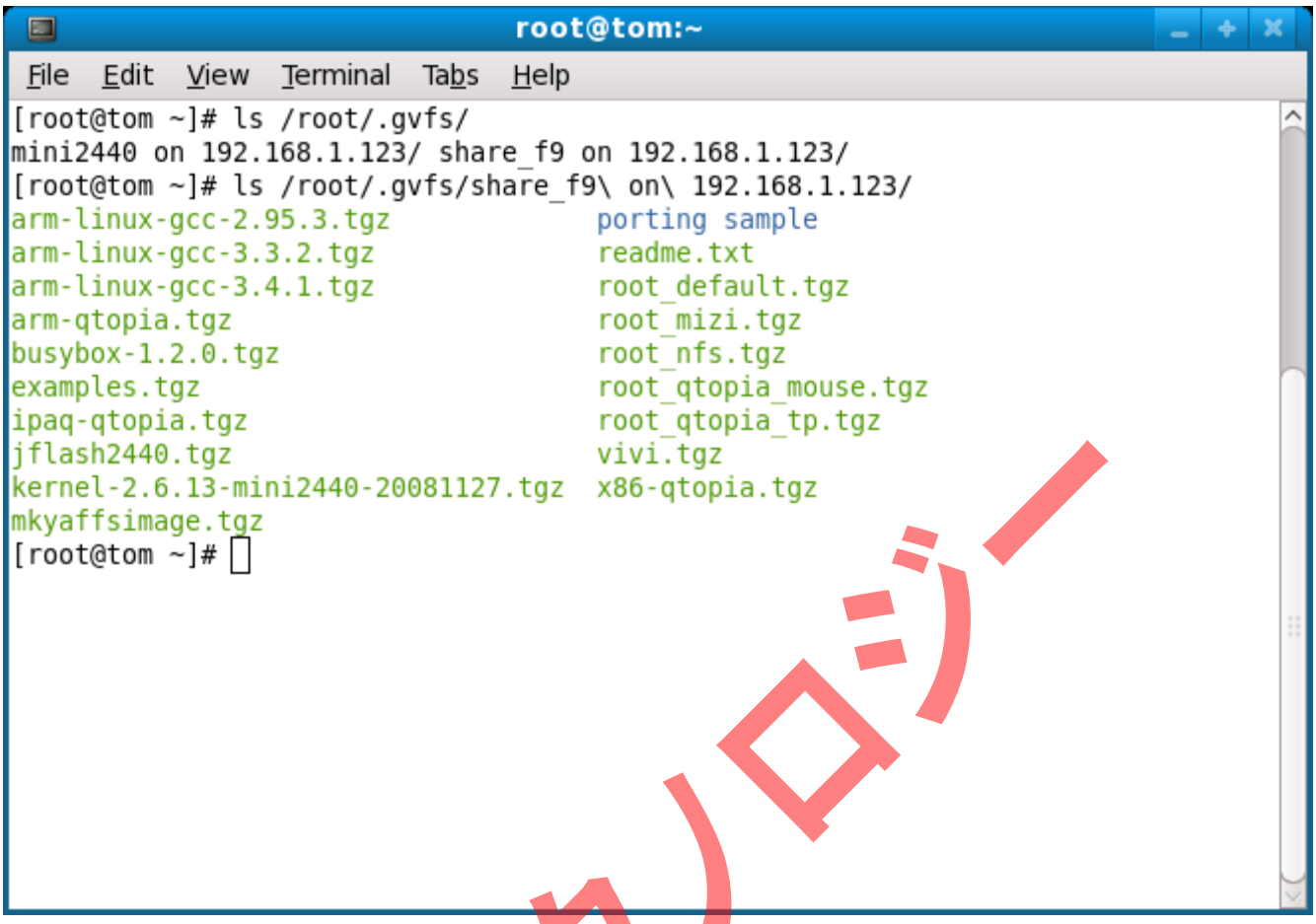
「Connect」をクリックし、下図の提示画面が出てくる：



そのまま「Connect」をクリックし、Windows 共有フォルダの内容が見られる。他のフォルダと同じく操作できる。



CMD から上記のフォルダを使用する場合、下図の操作を行う：



共有フォルダを切断する場合、デスクトップの共有フォルダに右ボタンで下図の操作を行う：

4.3.4 クロスコンパイル環境作成

Linux 上ボード用のカーネル、GUI の Qtopia/qt4、bootloader や他のアプリをコンパイルするにはクロスコンパイラが必要。ここでは arm-linux-gcc-4.5.1 を使用し、デフォルトで armv6 命令セットを使用し、ハードウェア浮動小数点演算をサポートする。

以下は作成の詳細手順である。

Step1： 付属 DVD の Linux フォルダの「arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz」を Fodora9 の tmp/ファイル（例）にコピーして、下記コマンドで解凍する：

#cd /tmp

#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz —C /

注意事項：C の後ろにスペースがあり、C は全角大文字。英語「Change」の先頭文字で、ここではフォルダを変更するという意味を表している。

このコマンドを実行すると、arm-linux-gcc を**/opt/FriendlyARM/toolschain/4.5.1** フォルダにインストールする。

Step2：コンパイラパスを環境変数に追加する、下記のコマンドを実行する：

#gedit /root/.bashrc

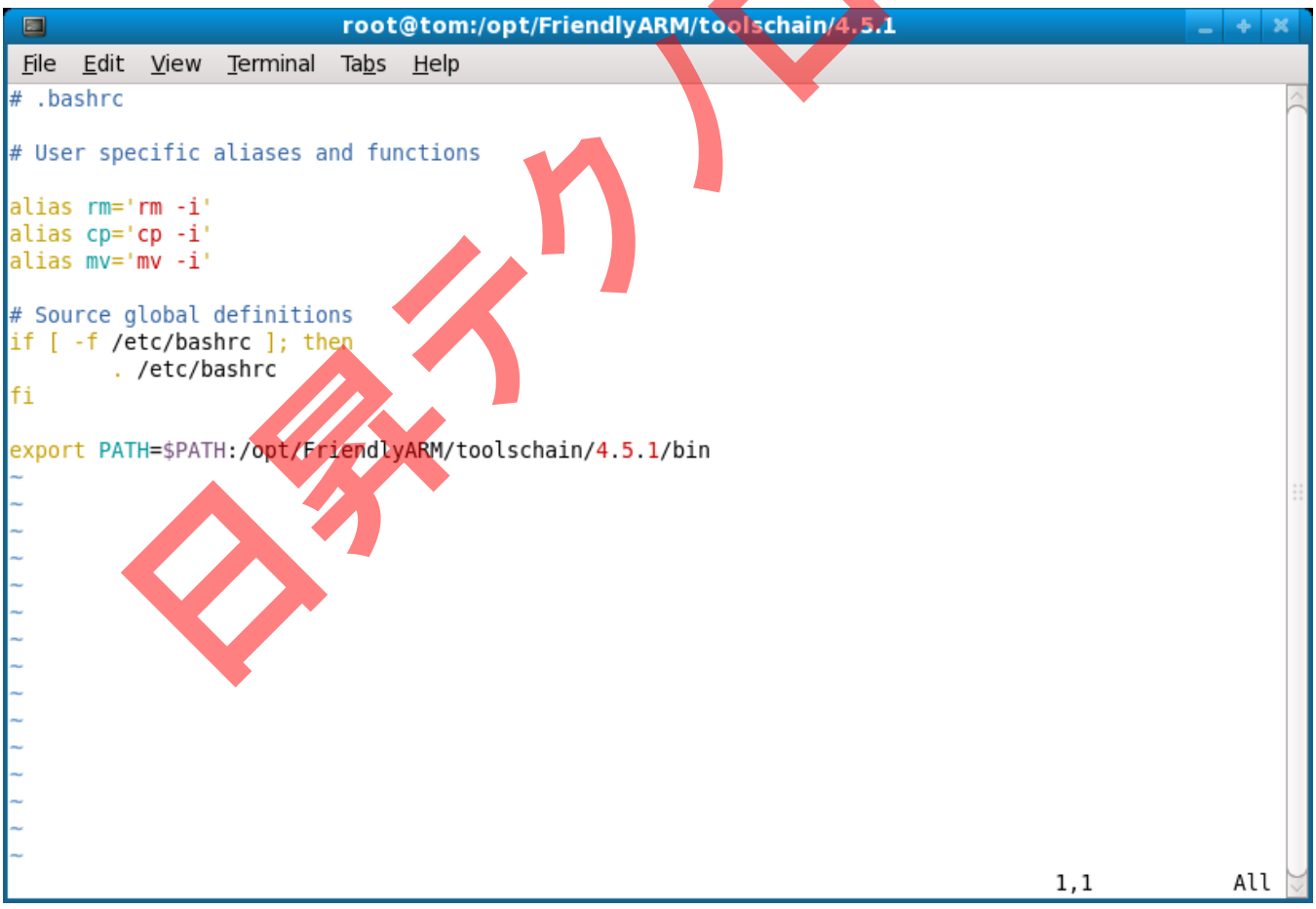
「/root/.bashrc」ファイルを編集する、"bashrc"の前に"."がある。

最後の一行に下記の内容を追加する：

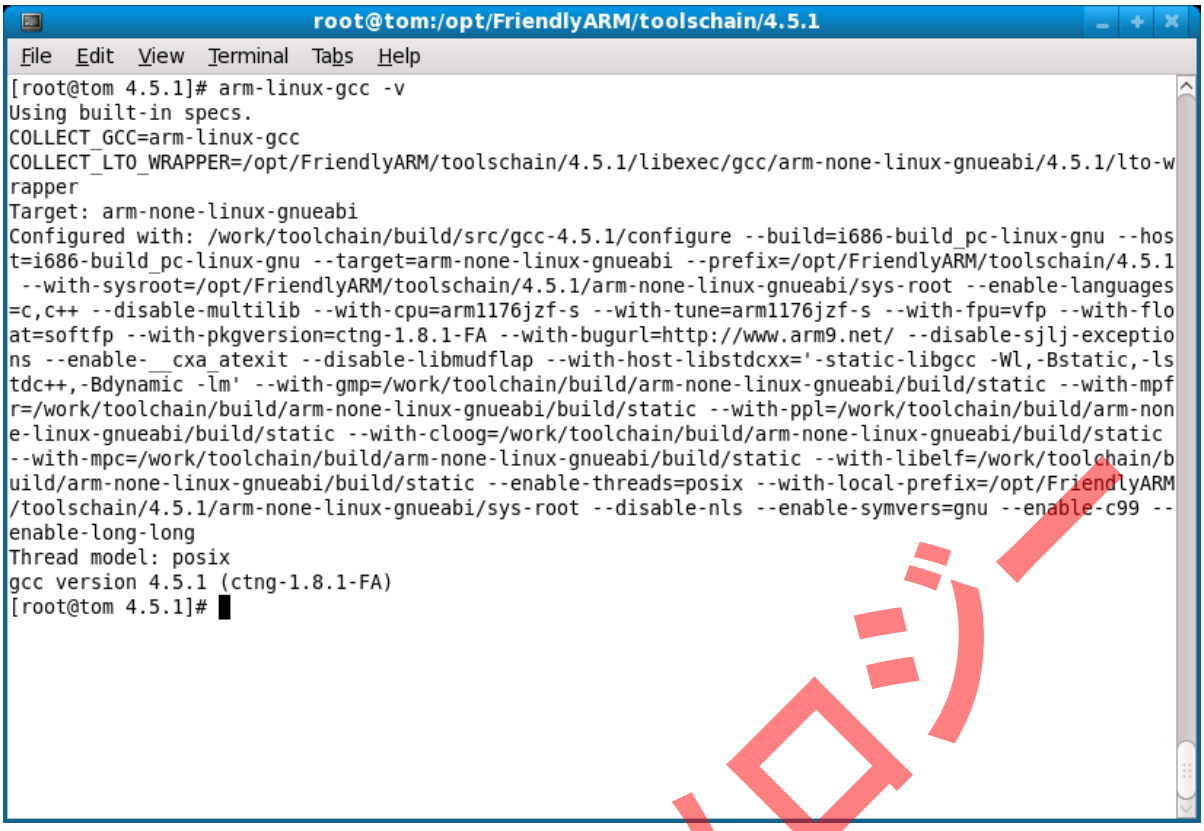
export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin

パスは必ず正しくご記入ください。

下図の通り修正して保存する：



システムに再ログインし（Start->logout）、上記の設定を有効にする。CMD に arm-linux-gcc -v を入力し、下図の情報が出てきて、クロスコンパイラ環境が構築できている：



4.4 ソースコードと他のツールの解凍とインストール

本節では全てのソースコード及びツールを解凍とインストールする、それらは：

-Linux カーネルソースコード

-Qtopia-2.2.0 プラットフォームソースコード（x86 と arm バージョン）

-arm-qt-extended-4.4.3 プラットフォームソースコード（x86 と arm バージョン）

-QtE-4.7.0 プラットフォームソースコード（arm バージョン）

-busybox-1.17 ソースコード

-Linux サンプルソースコード

-ファイルシステムフォルダ

-GUI の Linux logo 作成ツール **logomaker**

注意事項：全てのソースコードとツールは解凍でインストールし、全てのソースコードはコンパイラ arm-linux-gcc-4.5.1 によりコンパイルされる.

以下は詳細な解凍インストールプロセスと簡単な説明である。

4.4.1 ソースコードの解凍とインストール

先ずはフォルダ/opt/FriendlyARM/tiny4412/linux を作成する

CMD に

#mkdir -p /opt/FriendlyARM/tiny4412/linux

下記の手順のソースコードは全てこのフォルダに解凍する

（1） Linux ソースコードの用意

Fedora9 の/Tmp フォルダに/tmp/linux を作成する

#mkdir /tmp/linux

付属 DVD の Linux フォルダにある全てのファイルを/tmp/linux にコピーする

（2） Linux カーネルソースコードのインストール

/opt/FriendlyARM/tiny4412/linux フォルダで下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/linux-3.5-20131010.tar.gz

実行後 linux-3.5 フォルダが作成され、Linux カーネルソースコードが含まれている。

注意事項：20131010 は更新日付の一つの例。

（3）ファイルシステムのインストール

下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/rootfs_qtopia_qt4-20131010.tgz

rootfs_qtopia_qt4 フォルダが作成される。

注意事項：20131010 は更新日付の一つの例。

（4）組み込み GUI システム qtopia のインストール

/opt/FriendlyARM/tiny4412/linux フォルダに下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qtopia-20100420.tar.gz

#tar xvzf /tmp/linux/arm-qtopia-20101105.tar.gz

実行後 x86-qtopia フォルダと arm-qtopia フォルダが作成され，ソースコードが含まれている。

注意事項：x86-qtopia と arm-qtopia の後に日付が付いている可能性がある、リリース或いは更新日付を表示している。組み込みブラウザ konquor のソースコードも含まれている

（5）組み込み GUI システム qt-extended-4.4.3 のインストール

/opt/FriendlyARM/tiny4412/linux フォルダに下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qt-extended-4.4.3-20101003.tgz

#tar xvzf /tmp/linux/arm-qt-extended-4.4.3-20101105.tgz

実行後 x86-qt-extended-4.4.3 フォルダと arm-qt-extended-4.4.3 フォルダが作成され、ソースコードが含まれている。

注意事項：x86-qt-extended-4.4.3 と arm-qt-extended-4.4.3 の後に日付が付いている可能性がある、リリース或いは更新日付を表示している。

（6）QtE-4.7.0 のインストール

/opt/FriendlyARM/tiny4412/linux フォルダに下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/x86-qte-4.6.1-20100516.tar.gz

#tar xvzf /tmp/linux/arm-qte-4.7.0-20101105.tar.gz

実行後 x86-qte-4.6.1 フォルダと arm-qte-4.7.0 フォルダが作成され、ソースコードが含まれている。

注意事項：x86-qte-4.6.1 と arm-qte-4.7.0 の後に日付が付いている可能性がある、リリース或いは更新日付を表示している。x86-qte-4.6.1 は主に Creator プラットフォームの作成に利用され、バージョンが比

較的に低いが、開発に影響がない。

（7）busybox のインストール

ここでは busybox-1.13.3 バージョンを利用しており、最新バージョンは下記 URL をご参照ください：

http://www.busybox.net

/opt/FriendlyARM/tiny4412/linux フォルダに下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/busybox-1.13.3-20101120.tgz

実行後busybox-1.13.3が作成され、ソースコードが含まれている。

注意事項：ユーザーが便利に使用できる様にデフォルトで設定ファイルfa.configが作成される。

（8）Linuxサンプルソースコードのインストール

下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/examples-tiny4412-20131010.tgz

実行後examplesフォルダが作成され、サンプルソースコードが含まれている。

注意事項：20131010 は更新日付の一つの例。

4.4.2 ファイルシステムのインストール

下記のコマンドを実行する：

#cd /opt/FriendlyARM/tiny4412/linux

#tar xvzf /tmp/linux/rootfs_qtopia_qt4-20131010.tgz

実行後 rootfs_qtopia_qt4 が作成され、開発ボードのファイルシステムと完全に同じである。

注意事項：20131010 は更新日付の一つの例。

該当ファイルシステムは qtopia-2.2.0、Qtopia4、 QtE-4.7.0、busybox 等を含め、前のと比べ、下記の特性がある：

- 自動的にタッチスクリーンに繋がっているか確認し、初回使用の時校正が必要かを判断する。繋がっていない場合、自動的にシステムに入り、マウスを使用すればいい。でない場合、先ずはタッチスクリーンの校正を行う。
- 自動的に普通或いは高速 SD カード（最大 32G サポート）と USB メモリを識別する。
- 自動的に USB マウス或いはタッチスクリーンを検測する。
- USB マウスとタッチスクリーンの同時利用をサポートする。

4.4.3 LogoMaker のインストール

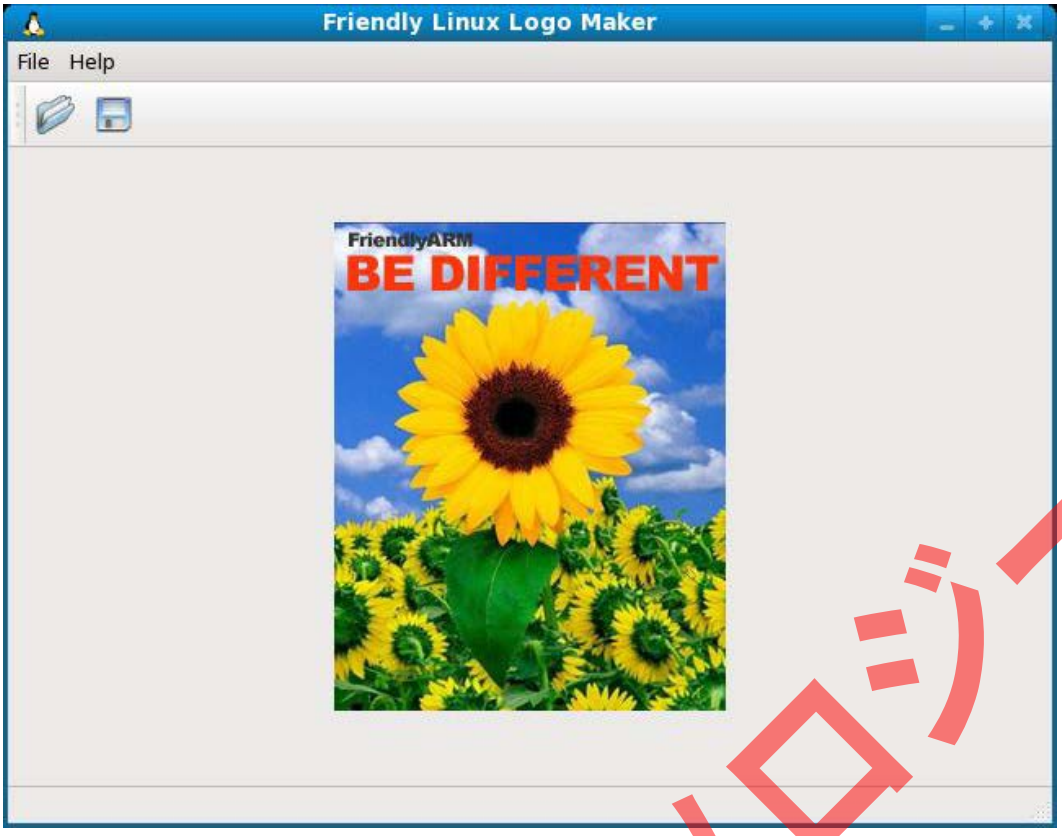
LogoMaker はLinux Logo を制作するツールであり、Fedora9 を基に開発したGUI 化したバージョンである。

下記のコマンドを実行する：

#tar xvzf /tmp/linux/logomaker.tgz -C /

注意事項：C は大文字で、後ろにスペースがある。インストールフォルダをチェンジするという意味を表している。

上記のコマンドを実行すると、LogoMaker は/usr/sbin フォルダにインストールされる。インストール完了後、コマンドラインに logomaker を入力し下図の画面が出てくる：



4.5 カーネルのコンフィグとコンパイル

Linux カーネルは Android カーネルと同じソースコードを使用するが、コンフィグが違い、下記のコマンドでカーネルをコンパイルする：

```
#cd /opt/FriendlyARM/tiny4412/android/linux-3.5
#cptiny4412_linux_defconfig.config  ;config の前に“. ”がある
```

make menuconfig コマンドを実行しコンフィグを修正できる。修正完了後、make を出力し、コンパイルする：

```
#make
```

コンパイル完了後、arch/arm/boot フォルダに zImage が作成され、SD カードの images/Linux/フォルダの zImage を取り替え Tiny4412 に書き込めばいい。

4.6 ファイルシステムイメージの作成

付属 DVD の tools フォルダの linux_tools.tgz を Ubuntu ルートディレクトリに解凍する。インターネットから iso ファイルをダウンロードする場合、下記のコマンドで iso をロードし解凍する：

```
# mkdir -p /mnt/iso
# mount -o loop Tiny4412-20130707.iso /mnt/iso
# cd /
# tar xvzf /mnt/iso/tools/linux_tools.tgz
```

```
#cd /opt/FriendlyARM/tiny4412/linux/
#make_ext4fs -s -l 314572800 -a root -L linux rootfs_qtopia_qt4.img rootfs_qtopia_qt4
```

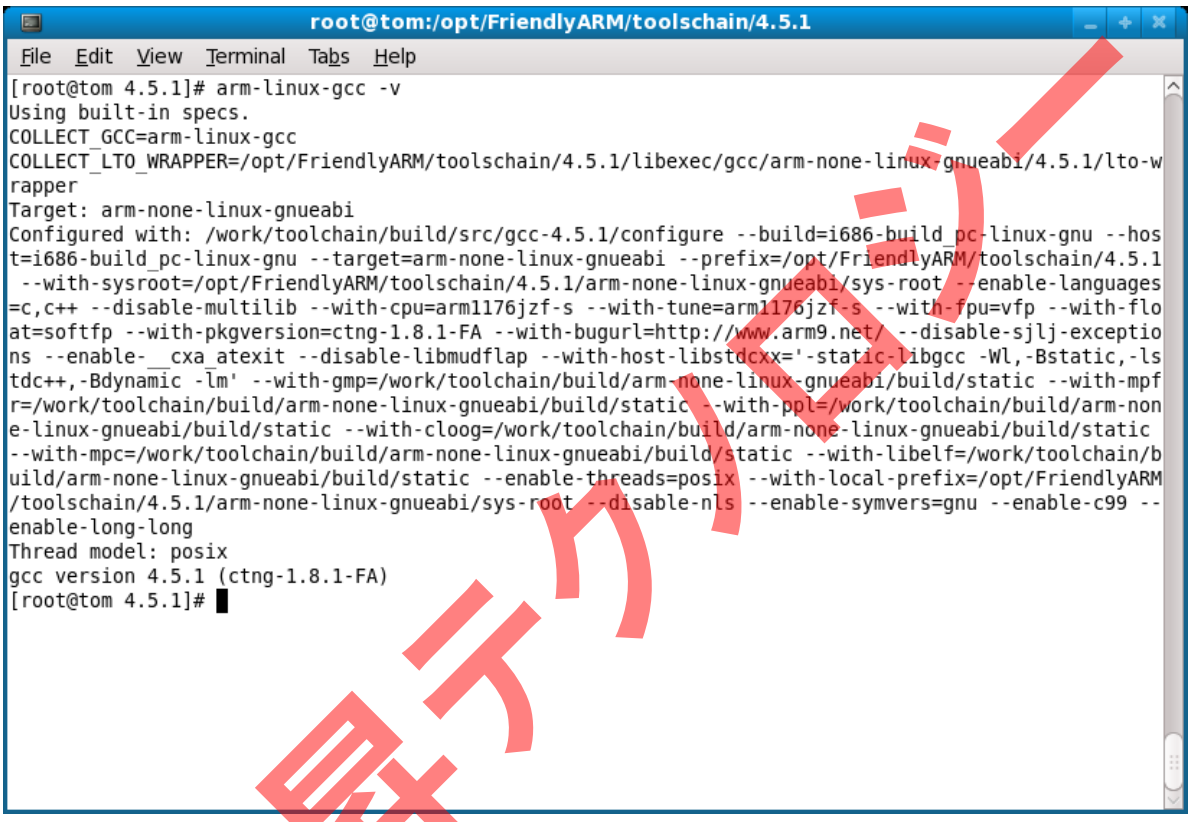
SD カードの images/Linux/フォルダの **rootfs_qtopia_qt4.img** を **rootfs_qtopia_qt4.img** に書き換え、Tiny4412 に書き込めばいい。

4.7 Linux 組み込みアプリ開発

本節は簡単な例で Linux アプリの編集、コンパイル及びボードに書き込んで実行するまで説明する。5.5.1 の手順を実行すれば、/opt/FriendlyARM/tiny4412/examples に下記のサンプルが見られる。

注意事項：下記のサンプルアプリが利用するコンパイラーは arm-linux-gcc-4.5.1-v6-vfp である。他のバージョンのクロスコンパイラーでコンパイルした場合、開発ボードで起動できない可能性がある。

クロスコンパイラーのバージョンを確認する時、下図のように arm-linux-gcc-v コマンドを実行する：



4.7.1 Hello,World!

/tmp/フォルダに hello.c ファイルを作成する：

```
#include <stdio.h>

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

Step1：コンパイル

ソースコードディレクトリに入り、make コマンドを実行する：

#cd /tmp/

#arm-linux-gcc hello.c -o hello

hello の実行ファイルが作成される。file コマンドで ARM 環境で実行できるか確認できる。開発ボードに実行出来る実行ファイルの出力は：

```
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked
(uses shared libs), for GNU/Linux 2.6.14, not stripped
```

Step2：開発ボードにダウンロード

コンパイルした実行ファイルをボードにダウンロードするには主に下記三種類の方法がある：

(1)ftp でファイルをボードに送信する（**お勧め**）

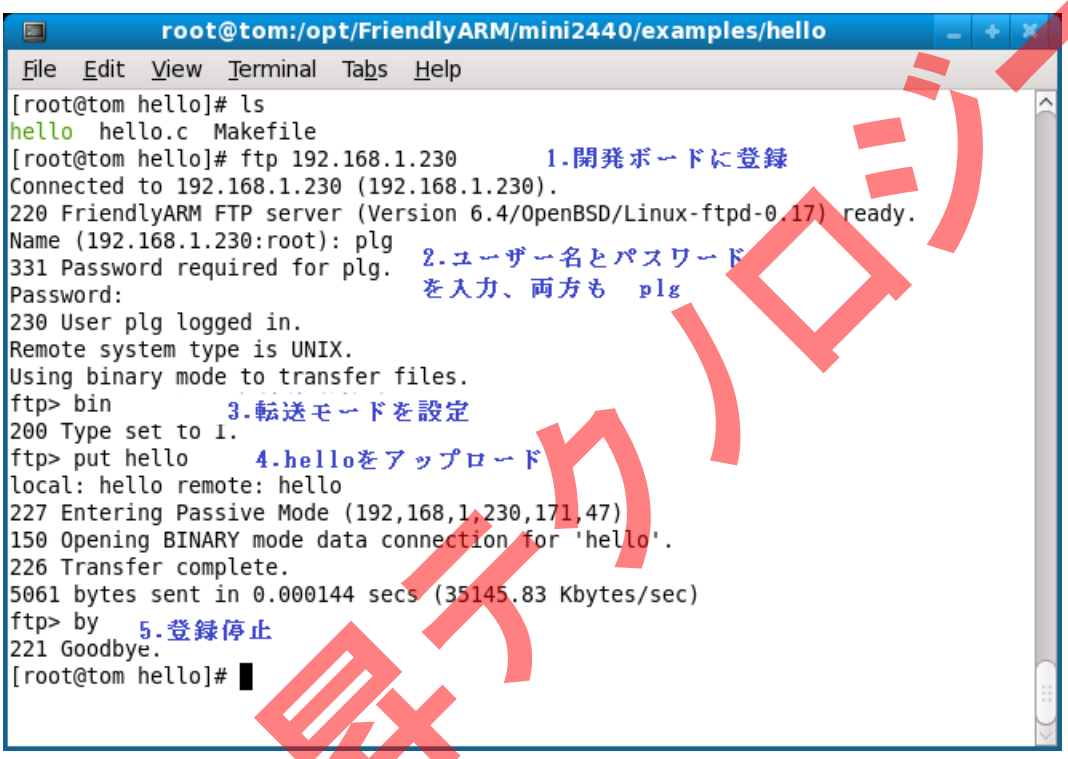
(2)メディア（USB メモリなど）にコピーする

(3)シリアルポートを通じてファイルをボードに送信する

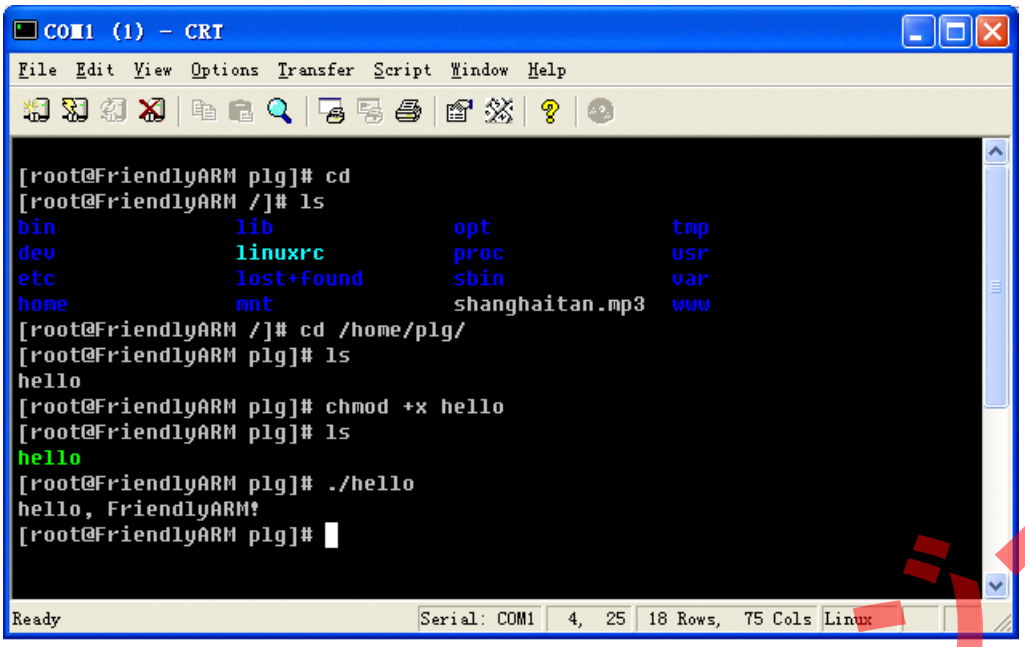
以下は一つづつ説明する：

(1)ftp でファイルをボードに送信する（お勧め）

先ずは下図のように PC 側で実行する：



次は下図のようにボード側で実行する：



(2)メディア（USB メモリなど）にコピーする

Step1： USB メモリにコピーする

#mount /dev/sda1 /mnt

#cp hello /mnt

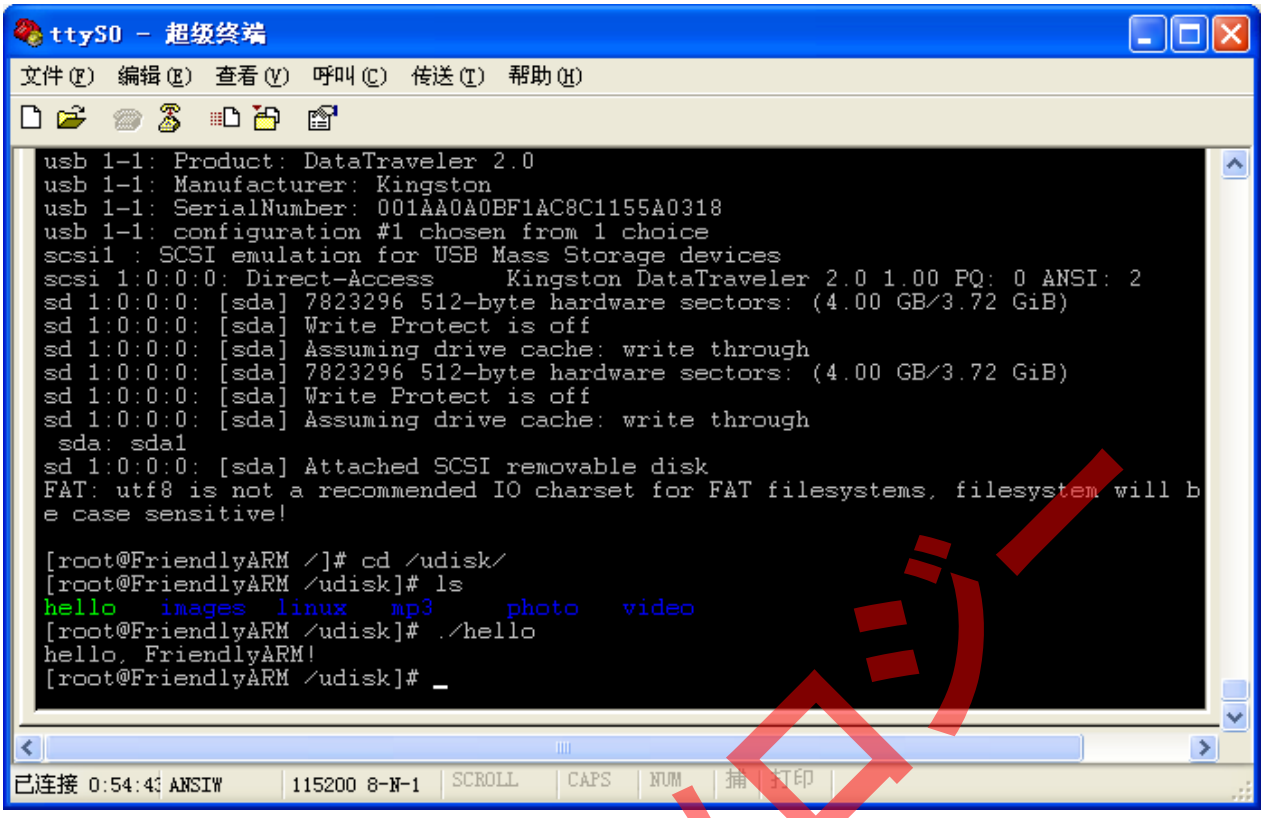
#umount /mnt

Step2： USB メモリからボードにコピーして実行する

USB メモリをボードの USB Host インタフェースに挿入すると自動的に/udisk にマウントする。下記コマンドで実行できる。

#cd /udisk

#./hello



(3)シリアルポートを通じてファイルをボードに送信する

送信方法は 4.2.3 節をご参照ください。

実行する前、**#chmod +x hello** で権限を修正する必要がある。

注意事項：USB 転換ケーブルを使用する場合、転換器の性能がよくないならボードに送信できないため、ftp でファイルの送信をお勧め。

4.7.2 LED テスト

プログラムリスト

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>

int main(int argc, char **argv)
{
    int on;
    int led_no;
    int fd;

/* ledが制御するパラメーターをチェックし、なかったらexit*/
if (argc != 3 || sscanf(argv[1], "%d", &led_no) != 1 || sscanf(argv[2],"%d", &on) != 1 ||
    on < 0 || on > 1 || led_no < 0 || led_no > 3) {
    fprintf(stderr, "Usage: leds led_no 0|1¥n");
```

```
    exit(1);
  }
  /*/dev/ledsをオープンする*/
    fd = open("/dev/leds0", 0);
    if (fd < 0) {
        fd = open("/dev/leds", 0);
  }
    if (fd < 0) {
        perror("open device leds");
        exit(1);
  }

  /*ioctlと入力したパラメーターでledを制御する*/
    ioctl(fd, on, led_no);
  /*設定をクロース*/
    close(fd);
    return 0;
}
```

前述のhelloポログラムと同じ様に実行ファイル「led」をコンパイルし、ボードにダウンロードして実行できる。

4.7.3 ボタンテスト

プログラムリスト

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>

int main(void)
{
  int buttons_fd;
  char buttons[6] = {'0', '0', '0', '0', '0', '0'};

  buttons_fd = open("/dev/buttons", 0);
  if (buttons_fd < 0) {
    perror("open device buttons");
    exit(1);
```

```
  }

  for (;;) {
    char current_buttons[6];
    int count_of_changed_key;
    int i;
    if (read(buttons_fd, current_buttons, sizeof current_buttons) != sizeof current_buttons)
{
    perror("read buttons:");
    exit(1);
  }
  for (i = 0, count_of_changed_key = 0; i < sizeof buttons / sizeof buttons[0]; i++) {
    if (buttons[i] != current_buttons[i]) {
      buttons[i] = current_buttons[i];
      printf("%skey %d is %s", count_of_changed_key? ", ": "", i+1, buttons[i] == '0' ? "up" :
"down");
      count_of_changed_key++;
    }
  }
  if (count_of_changed_key) {
    printf("¥n");
   }
  }
  close(buttons_fd);
  return 0;
}
```

前述のhelloポログラムと同じ様に実行ファイル「buttons」をコンパイルし、ボードにダウンロードして実行できる。

4.7.4 PWM ブザーテスト

プログラムリスト

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#define PWM_IOCTL_SET_FREQ   1
#define PWM_IOCTL_STOP        2

#defineESC_KEY     0x1b

static int getch(void)
{
  struct termios oldt,newt;
  int ch;
```

```
if (!isatty(STDIN_FILENO)) {
fprintf(stderr, "this problem should be run at a terminal¥n");
exit(1);
}
// save terminal setting
if(tcgetattr(STDIN_FILENO, &oldt) < 0) {
    perror("save the terminal setting");
    exit(1);
}
// set terminal as need
newt = oldt;
newt.c_lflag &= ~( ICANON | ECHO );
if(tcsetattr(STDIN_FILENO,TCSANOW, &newt) < 0) {
    perror("set terminal");
    exit(1);
}
ch = getchar();

// restore termialsetting
if(tcsetattr(STDIN_FILENO,TCSANOW,&oldt) < 0) {
     perror("restore the termial setting");
     exit(1);
  }
 return ch;
 }

 static int fd = -1;
 static void close_buzzer(void);
 static void open_buzzer(void)
 {
     fd = open("/dev/pwm", 0);
     if (fd < 0) {
         perror("open pwm_buzzer device");
         exit(1);
   }
     // any function exit call will stop the buzzer
     atexit(close_buzzer);
 }
 static void close_buzzer(void)
 {
     if (fd >= 0) {
         ioctl(fd, PWM_IOCTL_STOP);
         close(fd);
         fd = -1;
```

```
    }
  }
  static void set_buzzer_freq(int freq)
  {
      // this IOCTL command is the key to set frequency
      int ret = ioctl(fd, PWM_IOCTL_SET_FREQ, freq);
      if(ret < 0) {
          perror("set the frequency of the buzzer");
          exit(1);
    }
  }
  static void stop_buzzer(void)
{
      int ret = ioctl(fd, PWM_IOCTL_STOP);
      if(ret < 0) {
          perror("stop the buzzer");
          exit(1);
    }
  }
  int main(int argc, char **argv)
  {
      int freq = 1000 ;

      open_buzzer();

      printf( "¥nBUZZER TEST ( PWM Control )¥n" );
      printf( "Press +/-to increase/reduce the frequency of the BUZZER¥n" ) ;
      printf( "Press 'ESC' key to Exit this program¥n¥n" );

      while( 1 )
    {
          int key;

          set_buzzer_freq(freq);
          printf( "¥tFreq = %d¥n", freq )
          ;
          key = getch();

          switch(key) {
          case '+':
              if( freq < 20000 )
                  freq += 10;
              break;

          case '-':
```

```
            if( freq > 11 )
                freq -= 10 ;
            break;

        case ESC_KEY:
        case EOF:
        stop_buzzer();
        exit(0);

     default:
        break;
      }
   }
}
```

前述のhelloポログラムと同じ様に実行ファイル「pwm_test」をコンパイルし、ボードにダウンロードして実行できる。

4.7.5 I2C-EEPROM テスト

下記のプログラムに同じディレクトリの 24cXX.c プログラムが必要となる

```
#include <stdio.h>
#include <fcntl.h>
#include <getopt.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "24cXX.h"

#define usage_if(a) do { do_usage_if( a , __LINE__); } while(0);
void do_usage_if(int b, int line)
{
    const static char *eeprog_usage =
        "I2C-24C08(256 bytes) Read/Write Program, ONLY FOR TEST!¥n"
        "FriendlyARM Computer Tech. 2009¥n";
    if(!b)
        return;
    fprintf(stderr, "%s¥n[line %d]¥n", eeprog_usage, line);
    exit(1);
}

#define die_if(a, msg) do { do_die_if( a , msg, __LINE__); } while(0);
void do_die_if(int b, char* msg, int line)
```

```
{
    if(!b)
        return;
    fprintf(stderr, "Error at line %d: %s¥n", line, msg);
    fprintf(stderr, "sysmsg: %s¥n", strerror(errno));
    exit(1);
}

static int read_from_eeprom(struct eeprom *e, int addr, int size)
{
    int ch, i;
    for(i = 0; i < size; ++i, ++addr)
    {
        die_if((ch = eeprom_read_byte(e, addr)) < 0, "read error");
        if( (i % 16) == 0 )
            printf("¥n %.4x|  ", addr);
        else if( (i % 8) == 0 )
            printf("  ");
        printf("%.2x ", ch);
        fflush(stdout);
     }
      fprintf(stderr, "¥n¥n");
      return 0;
}

static int write_to_eeprom(struct eeprom *e, int addr)
{
      int i;
      for(i=0, addr=0; i<256; i++, addr++)
     {
          if( (i % 16) == 0 )
              printf("¥n %.4x|  ", addr);
          else if( (i % 8) == 0 )
              printf("  ");
          printf("%.2x ", i);
          fflush(stdout);
          die_if(eeprom_write_byte(e, addr, i), "write error");
  }
  fprintf(stderr, "¥n¥n");
    return 0;
}

int main(int argc, char** argv)
{
```

```
        struct eeprom e;
        int op;

        op = 0;

        usage_if(argc != 2 || argv[1][0] != '-' || argv[1][2] != '¥0');
        op = argv[1][1];

        fprintf(stderr, "Open /dev/i2c/0 with 8bit mode¥n");
    die_if(eeprom_open("/dev/i2c/0", 0x50, EEPROM_TYPE_8BIT_ADDR, &e) < 0,
          "unable to open eeprom device file "
          "(check that the file exists and that it's readable)");
    switch(op)
    {
    case 'r':
        fprintf(stderr, "  Reading 256 bytes from 0x0¥n");
        read_from_eeprom(&e, 0, 256);
        break;
    case 'w':
        fprintf(stderr, "  Writing 0x00-0xff into 24C08 ¥n");
        write_to_eeprom(&e, 0);
        break;
    default:
        usage_if(1);
        exit(1);
     }
    eeprom_close(&e);

    return 0;
}
```

4.7.6 パイププログラムサンプル-ウェブで LED の制御

原理説明
ボード起動後、ウエブでコマンドを発行しボード上の LED をコントロールできる。これはプロセスの間の通信でリソースを共有するサンプルである。プロセスの間の通信は IPC（InterProcess Communication)。プロセスの間に通信の目的は主に下記の五つ： (1)データ伝送 (2)データ共有 (3)イベント通知 (4)リソース共有 (5)プロセスコントロール Linux では数多くの IPC メカニズムをサポートし、信号とパイプはその二つである。詳しい説明は他の Linux プログラミング本にあるため、ここではこれ以上説明しない。

ウェブで LED の制御はパイプメカニズムで実現され、LED は共有リソースとなり、led-player はバックグランドアプリである。このアプリを起動すると、「/tmp/led-control」というパイプを作成する。(mknod コマンドで作成できるが、プログラムに修正が必要となり、興味のある方はご自分でお試しください)。このパイプに入力されたデータを監視しパラメーター（モード：type、サイクル：period）により LED の表示モードを変更させる。led.cgi は CGI プログラムで、ウェブからのキャラクターコマンド（ping：マーキーモードあるいは卓球モード、counter：カウンタモード、stop：停止モード、slow：サイクルが 0.25m、normal：サイクルが 0.125m、fast：サイクルが 0.0625m）を受け、コマンドに実際の数値を付け、最後に「echo」コマンドを実行しパイプ「/tmp/led-control」に出力して LED の制御を実現する。

プログラムリスト

```
#include <stdio.h>
#include<stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <string.h>
static int led_fd;
static int type = 1;

static void
push_leds(void)
{
    static unsigned step;
    unsigned led_bitmap;
    int i;

    switch(type) {
    case 0:
    if (step >= 6) {
      step = 0;
  }
    if (step < 3) {
      led_bitmap = 1 << step;
    } else {
      led_bitmap = 1 << (6 -step);
  }
    break;
 case 1:
    if (step > 255) {
      step =0;
  }
    led_bitmap = step;
```

```
    break;
  default:
      led_bitmap = 0;
  }
  step++;
  for (i = 0; i < 4; i++) {
      ioctl(led_fd, led_bitmap & 1, i);
      led_bitmap >>= 1;
  }
}
int main(void)
{
  int led_control_pipe;
  int null_writer_fd; // for read endpoint not blocking when control process exit

  double period = 0.5;

  led_fd = open("/dev/leds0", 0);
  if (led_fd < 0) {
      led_fd = open("/dev/leds", 0);
  }
  if (led_fd < 0) {
      perror("open device leds");
      exit(1);
  }
  unlink("/tmp/led-control");
  mkfifo("/tmp/led-control", 0666);

  led_control_pipe = open("/tmp/led-control", O_RDONLY | O_NONBLOCK);
  if (led_control_pipe < 0) {
      perror("open control pipe for read");
      exit(1);
  }
  null_writer_fd = open("/tmp/led-control", O_WRONLY | O_NONBLOCK);
  if (null_writer_fd < 0) {
      perror("open control pipe for write");
      exit(1);
  }
  for (;;) {
      fd_set rds;
      struct timeval step;
      int ret;

      FD_ZERO(&rds);
      FD_SET(led_control_pipe, &rds);
```

```
        step.tv_sec  = period;
        step.tv_usec = (period -step.tv_sec) * 1000000L;

        ret = select(led_control_pipe + 1, &rds, NULL, NULL, &step);
        if (ret < 0) {
            perror("select");
          exit(1);
        }
        if (ret == 0) {
            push_leds();
        } else if (FD_ISSET(led_control_pipe, &rds)) {
              static char buffer[200];
              for (;;) {
                char c;
                int len = strlen(buffer);
                if (len >= sizeof buffer -1) {
                    memset(buffer, 0, sizeof buffer);
                    break;
                }
                if (read(led_control_pipe, &c, 1) != 1) {
                    break;
                }
                if (c == '\r') {
                    continue;
                }
                if (c == '\n') {
                int tmp_type;
                double tmp_period;
                if (sscanf(buffer,"%d%lf", &tmp_type, &tmp_period) == 2) {
                    type = tmp_type;
                    period = tmp_period;
                }
                fprintf(stderr, "type is %d, period is %lf\n", type, period);
                memset(buffer, 0, sizeof buffer);
                break;
            }
             buffer[len] = c;
            }
        }
    }

    close(led_fd);
    return 0;
}
```

make コマンドで直接に実行ファイル「led-player」をコンパイルできる、サーバーとしてボ

ードの「/sbin」に置いてある。

Leds.cgi CGIプログラムはボード上の「/www/leds.cgi」にあり、シェルスクリプトである。アクションの一つとしてウェブページ「leds.html」から呼び出される。Leds.cgiスクリプトリストは下記の通り：

```
#!/bin/sh

type=0
period=1

case $QUERY_STRING in
        *ping*)
         type=0
         ;;
 *counter*)
         type=1
         ;;
 *stop*)
         type=2
         ;;
 esac

 case $QUERY_STRING in
 *slow*)
         period=0.25
         ;;
 *normal*)
         period=0.125
         ;;
 *fast*)
         period=0.0625
         ;;
 esac

 /bin/echo $type $period > /tmp/led-control
 echo "Content-type: text/html; charset=gb2312"
 echo
 /bin/cat led-result.template

exit 0
```


4.8 Qtopia-2.2.0 のコンパイル

Qtopia-2.2.0 のコンパイルは複雑であるため、コンフィグとコンパイルの手順を build スクリプトに作成し、初心者の勉強と使用を便利になる。該当スクリプトを実行すると、Qtopia プラットフォーム及び各プロ

グラムがコンパイルできる。「run」スクリプトで実行できる。X86 バージョンと arm バージョンの手順は基本的に同じであるが、スクリプトの内容には少々違いがあり、詳細は下記の内容をご参照ください。

4.8.1 x86 バージョンの Qtopia-2.2.0 のコンパイルと実行

ここでのソフトウェア開発とテストは全部 Fedora9 プラットフォームを基に開発したものであり、他のプラットフォームでテストしたことがない。Linux に十分詳しいなら、エラーを解決できると思われる。多くのエラーは対象のプラットフォームにあるライブラリ或いはツールが足りないことによるものである。初心者に対しては、弊社と同じプラットフォーム（Fedora9.0）のご使用をお勧め。

該当会社のホームページからダウンロードできる。

ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso

インストールする時、コンポーネントを漏らさないように、マニュアルに従って実施してください。

Linux ディストリビューションは多いため、全てのインストール手順を作成するのは不可能であることをご了承ください。

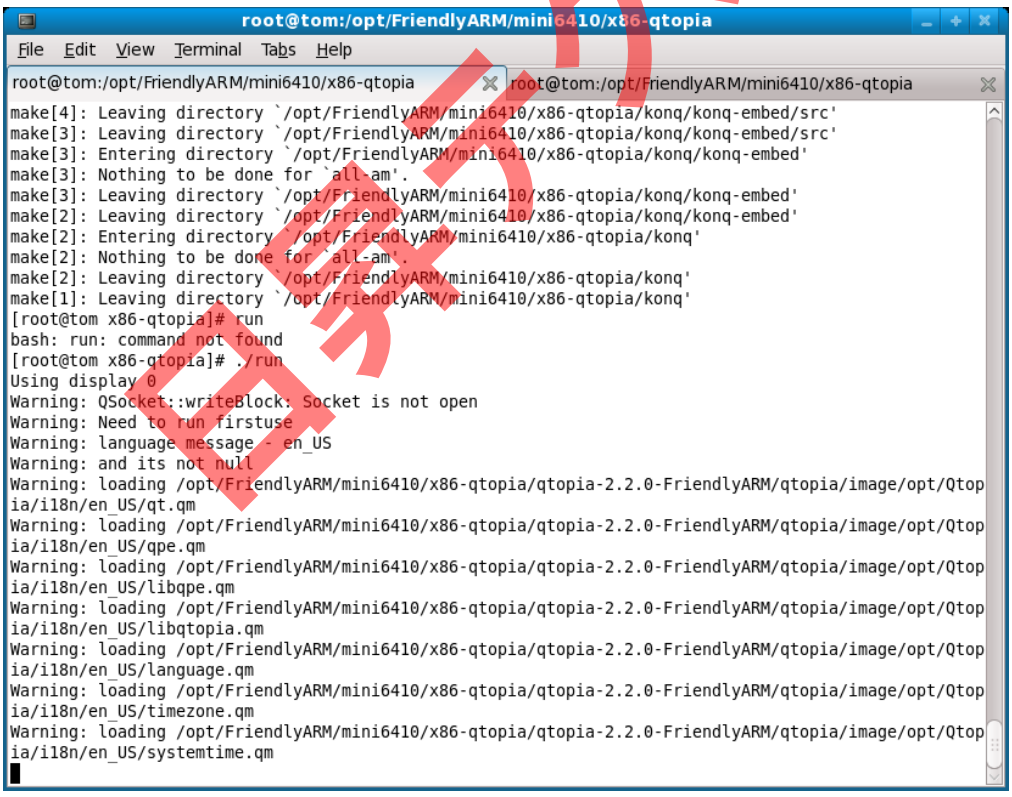
下記のコマンドを実行し、Qtopia と組み込みブラウザーをコンパイルする：

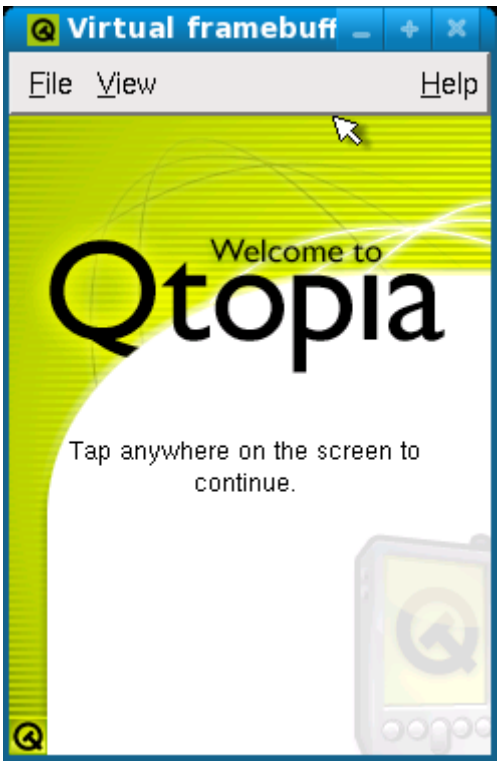
```
#cd /opt/FriendlyARM/tiny4412/linux/x86-qtopia
#./build-all            (30 分ぐらいかかる)
```

順調に実行できない場合、上記の赤い部分をご参照ください。

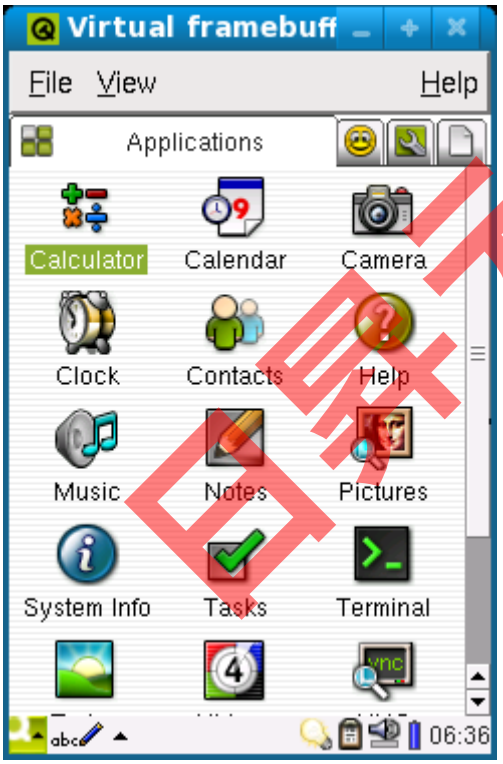
#./run コマンドでコンパイルした Qtopia システムを起動させる。“/”の前に“.”がある。

下図の画面が見られる：





提示に従ってクリックし、下図のようにQtopiaシステムが出てくる：

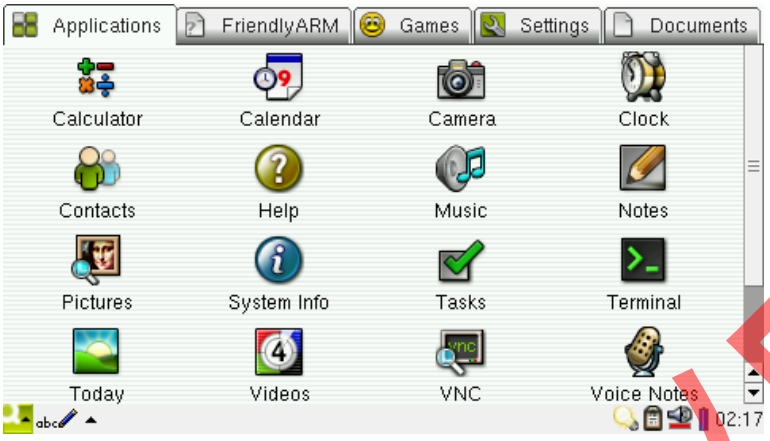


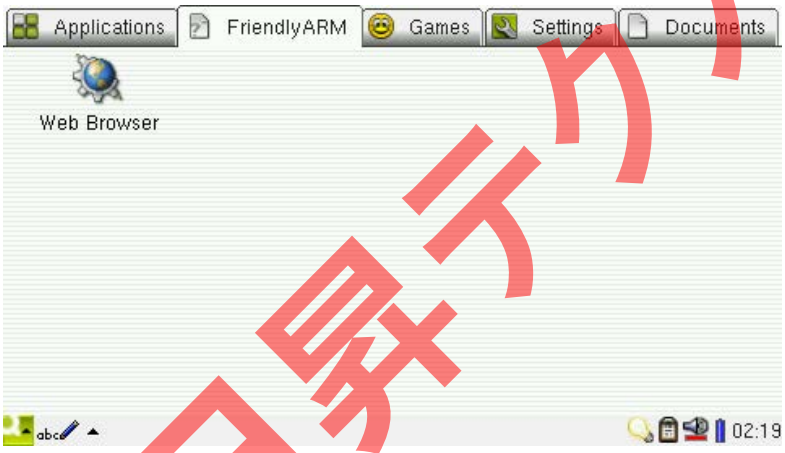
4.8.2 arm バージョンの Qtopia-2.2.0 のコンパイルと実行

arm-linux-gcc-4.5.1 コンパイラーと Fedora9 プラットフォームを確保した上、下記のコマンドを実行し Qtopia と組み込みブラウザーをコンパイルする：

```
#cd /opt/FriendlyARM/tiny4412/linux/arm-qtopia
```

#./build-all　　（30分ぐらいかかる）
#./mktarget　　（ルートファイルシステムのイメージファイル target-qtopia-konq.tgz を作成）
順調に実行できない場合、4.8.1 の最初の赤い部分をご参照ください。
/opt フォルダのファイルを全部削除し、ボードの元の Qtopia システムが削除できる。作成した target-qtopia-konq.tgz を USB メモリ或いは他の方法でボードのルートフォルダに解凍する。Ftp で /home/plg フォルダにダウンロードしたと仮定し、ハイパーターミナルに下記コマンドを入力する：
#tar xvzf /home/plg/target-qtopia-konq.tgz -C /
“C”は Change の意味で、“/”はルートディレクトリに解凍することを表示する。実行後、ボードをリスタートすると、下図のように全ての画面は英語になり、「FriendlyARM」タブにあるアプリはブラウザーだけ：





4.9 QtE-4.8.5 のコンパイル及びインストール

4.9.1 arm バージョンの QtE-4.8.5 のコンパイルと実行

Qtopia-2.2.0 と同じように build-sh スクリプトを提供しており、ソースコードディレクトリに下記のコマンドを実行する：
#cd /opt/FriendlyARM/tiny4412/linux/arm-qte-4.8.5
#./build.sh
かなり時間かかるため、暫くお待ちください。
実行後、mktarget スクリプトを実行し、コンパイルされたターゲットファイルディレクトリーより target-qte-4.8.5-to-devboard.tgz（ライブラリーファイル）と target-qte-4.8.5-to-hostpc.tgz（2 進

法事例）を抽出する。Disc の Linux ディレクトリーにも格納されている。

target-qte-4.8.5-to-devboard.tgz は開発ボード向けのバージョン（開発ツールを削除したもの）で、target-qte-4.8.5-to-hostpc.tgz は PC 向けのバージョン（開発及びコンパイル用のバージョンで、qmake などの Qt ツール及びコンパイルに必要なヘッダーなどを含んで、QtCreator 開発ツールの設定に使える）である。

QtE-4.8.5 を開発ボードにインストールする方法：

target-qte-4.8.5-to-devboard.tgz を開発ボードのルートディレクトリに解凍し、下記のコマンドを実施する。（圧縮バックが SD カードの直下に格納されている場合、）：

```
# rm -rf /usr/local/Trolltech/QtEmbedded-4.8.5-arm
# tar xvzf /sdcard/target-qte-4.8.5-to-devboard.tgz -C /
```

QtE-4.8.5 を PC にインストールする方法：

target-qte-4.8.5-to-hostpc.tgz を PC のルートディレクトリーに解凍し、下記のコマンドを実施する：

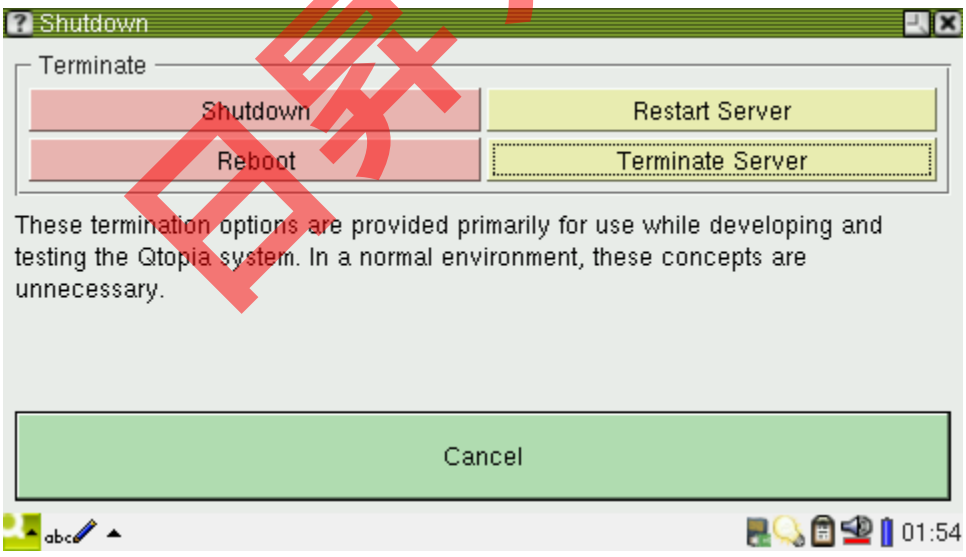
```
# tar xvzf target-qte-4.8.5-to-hostpc.tgz -C /
```

QtE-4.8.5 は/usr/local/Trolltech/QtEmbedded-4.8.5-arm の下にインストールされる。

4.9.2 QtE4.8.5APP の開発及び動作

4.9.2.1 Qtopia-2.2.0 環境での Qt プログラムのテスト

QtE-4.8.5 を実行するには、Qtopia-2.2.0 を停止する必要がある。「設定」タブの「シャットダウン」をクリックし、下記画面で「Terminate Server」をクリックし Qtopia-2.2.0 を停止できる：



Qtopia-2.2.0 を停止しコマンドラインに qt4 を入力すると、下図のように QtE-4.8.5 を起動させる：

自分でコンパイルした Qt4 プログラムを起動する場合、関連の環境変量を設定しなければいけない。/bin/setqt4env スクリプト設定を利用できる。/bin/qt4 スクリプトサンプルをご参考ください。

4.9.2.2　Qt4 プログアムのオート起動

QtE-4.8.5 で開発したプログラムを起動する手順：

（1）起動スクリプト/etc/init.d/rcS を Edit する。qtopia 起動項目を削除し、最後に/bin/qt4& を追記する。

（2）スクリプト/bin/qt4 を Edit する。最後の 2 行を変更し上、再起動するといい。

変更前

```
cd /usr/local/Trolltech/QtEmbedded-4.8.5-arm/demos/embedded/fluidlauncher
./fluidlauncher -qws
```

変更後

注意：プログラムは/opt/ディレクトリーに格納されている場合（プログラム名：helloworld)。

```
cd /opt
./helloworld -qws
```

4.9.2.3 Qt4 プログラムのスクリーン回転

横画面を縦画面に設定する場合、開発ボードのスクリプト/bin/setqt4env を Edit し、Qt4 プログラムを先どうするといい。

変更前：

```
export QWS_DISPLAY=:1
```

変更後：

```
export QWS_DISPLAY=Transformed:Rot90:1
```

export QWS_DISPLAY=:1 は回転なしの意味である。上記の変更で 90 度の回転を実現した。他に 120 度、270 などの回転も実現できる。

4.10 Qtopia4(Qt-Extended-4.4.3)のコンパイル

4.10.1 x86 バージョンの Qt-Extended-4.4.3 のコンパイルと実行

Qtopia-2.2.0 と同じように build-all スクリプトを提供しており、ソースコードディレクトリに下記のコマンドを実行する：

```
#cd /opt/FriendlyARM/tiny4412/linux/x86-qt-extended-4.4.3
#./build
```

かなり時間かかるため、暫くお待ちください。

#./run コマンドでコンパイルした Qtopia システムを起動させる。“/”の前に“.”がある。

下図の画面が見られる：

4.10.2 arm バージョンの Qt-Extended-4.4.3 のコンパイルと実行

Qtopia-2.2.0 と同じように build-all スクリプトを提供しており、ソースコードディレクトリに下記のコマンドを実行する：

```
#cd /opt/FriendlyARM/tiny4412/linux/arm-qt-extended-4.4.3
#./build
```

かなり時間かかるため、暫くお待ちください。

実行後、mktarget スクリプトを実行し、ルートファイルシステムのイメージ target-qtopia4.tgz を作成する。開発ボードのルートディレクトリに解凍し、下記のコマンドで使用できる：

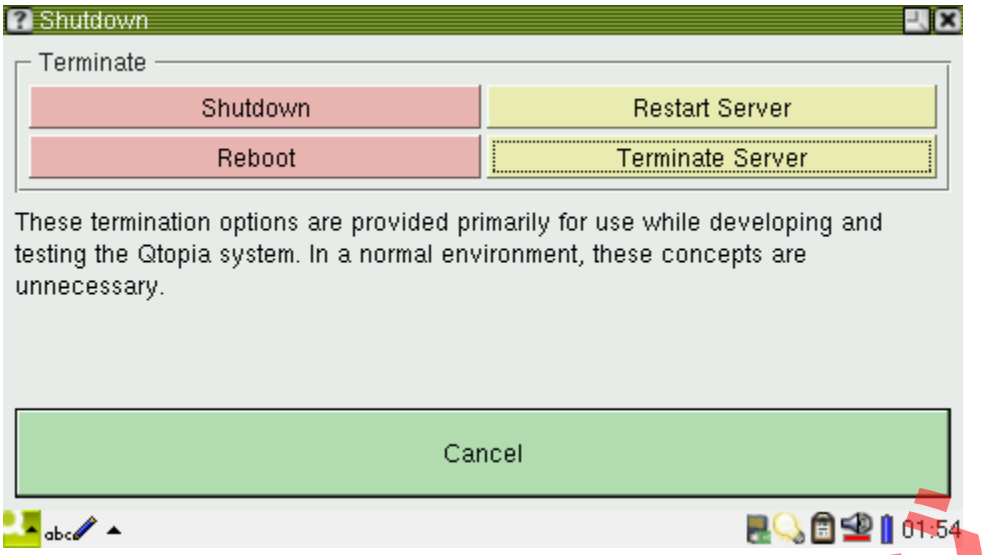
```
#tar xvzf target- qtopia4.tgz-C /
```

/opt フォルダに Qtopia4.4.3 フォルダが生成される。

ボードにプリインストールされた Linux には QtE-4.7.0 があるため、テストする前に rm コマンドで Qtopia4.4.3 フォルダにあるファイルを全部削除すればいい。

Qtopia4 を実行するには、Qtopia-2.2.0 を停止する必要がある。「設定」タブの「シャットダウン」をク

リックし、下記画面で「Terminate Server」をクリックしQtopia-2.2.0を停止できる：



他の方法として、起動スクリプトファイル/etc/init.d/rcSからqtopiaをコメントアウトして再起動、killallコマンドで関連するプロセスをキルするなどがある。

Qtopia-2.2.0を停止しコマンドラインにqtopia4 &（&はバックグラウンドを表示する）を入力すると、下図のようにQtopia4を起動させる：

第五章 Linux アプリ開発マニュアル

Linux アプリの開発は 6410 プラットフォームと基本的に同じであるため、Mini6410 の開発ファイル「Mini6410-Qt_Qtopia-programingManual.pdf」をご参照ください。