AM1808搭載CPUボード 「ARM-Y」	株式会社ＹＯＯＤＳ 〒754-0011 山口県山口市小郡御幸町4番地9	tel. 083-976-0022
		fax. 083-976-0023
		info_yoods@yoods.co.jp

ARM-Yユーザーマニュアル

2013年9月16日
文書番号: YD13065-v1.02

AM1808搭載CPUボード「ARM-Y」	株式会社ＹＯＯＤＳ 〒754-0011 山口県山口市小郡御幸町4番地9	tel. 083-976-0022
		fax. 083-976-0023
		info_yoods@yoods.co.jp

目次

ARM CPUボードについて

ARM-Yは...

省電力仕様　「ARM-Y」は様々な機器に組み込んでIO監視，通信GWなどの用途に使うためのインダストリアル仕様ARM9 CPUボードです。組込用途として使うため、低消費電力で動作し、CPUの能力,メインメモリ，フラッシュメモリについても、様々な用途の最大公約数となるように構成を検討されています。

インダストリアル仕様　長期安定供給できる部品選定を行い、日本国内で製造しています。動作温度範囲も広く(-20°C〜80°C)，RoHSにも対応しています。

使い易さ　使いやすさへの配慮として、DC5Vを供給するだけで動作するようにARM-Y上に電源回路を搭載してあります。また34×2の2.54ピッチヘッダピンによりペリフェラルボードの試作,設計を容易にして、各ピンのマルチプレクス設定に対応したドライバを標準で添付しています。

セルフ開発環境　ソフトウェアの開発を容易にするため、マイクロSDからのLinux開発環境をブートしてGCCを使ったセルフ環境での開発が可能です。USBのOTG機能によりパソコンからEthernetデバイスとして認識して、SSH, HTTP等による接続ができます。

ブラウザ開発環境「JWT」　ブラウザでHTTP接続してWEB上でのプログラム開発可能な環境「JWT」を標準搭載しています。ブラウザ上で直接プログラムソースや設定ファイルを書換え,保存ができて、make機能によるコンパイルも可能です。

プログラミングレス開発　ソフトウェアPLC「JUNKWare」によりラダー記述によるソフトウェア開発も可能です。シーケンサのコイルの機能としてシェルスクリプトや外部プログラムを起動することもできるので、C/C++による開発無しでシステムの構築が可能です。

ARM-Yの概要

ARM-Yの機能概要を下表に示します。

項　　目	仕様・内容
CPU	TI AM1808 CPUクロック456MHz
メモリ	LPDDR64MByte
フラッシュメモリ	NOR 32MByte
外部記憶メモリ	マイクロSD
USB	High Speed 1ch, mini Bコネクタ, OTG対応
LED	電源確認, 汎用2ch
拡張インターフェイス	34PIN×2, 2.54ピッチ(ヘッダピン未実装) UART0,1,2, LCD, EMAC, SPI, I2C, USB, GPIOを拡張可能 ※UART2はARM-Y上の6×2mmピッチヘッダピンからもシリアルコンソールとして利用可能
電源	DC5V単一電源(5V±10%) 消費電流 Typ. 0.09A, max. 0.17A (ARM-Y単体) ※基板上ジャンパによりUSBからDC5V供給も可能
サイズ	67mm × 41.5mm
ソフトウェア仕様	U-boot, Linux Kernel 3.1.0, ルートファイルシステム書込済み マイクロSDルートファイルシステムによりセルフ開発可能
ブートモード	UARTブート, フラッシュメモリブートをDIP SWで選択可能
その他	RoHS対応

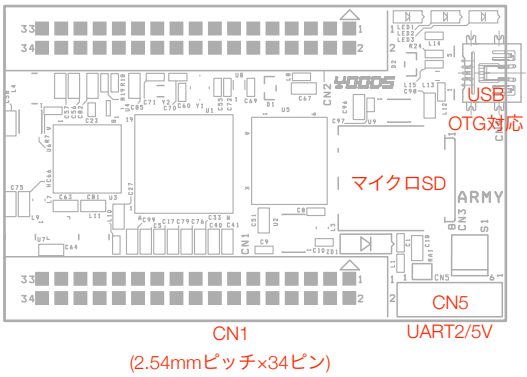


図1 ARM-Yのインタフェース

電源の接続

ヘッダピンによる電源供給

ARM-YのCN1-1,2番ピン，CN5-1,2番ピンは共通で、ARM-Y用DC5V電源供給用に使用できます。CN5を使用する場合は日本圧着端子製造(株)のPHコネクタ等を用いてください。その場合、電源供給と2〜6番ピンにシリアルコンソールを接続することも可能です。(付録参照)

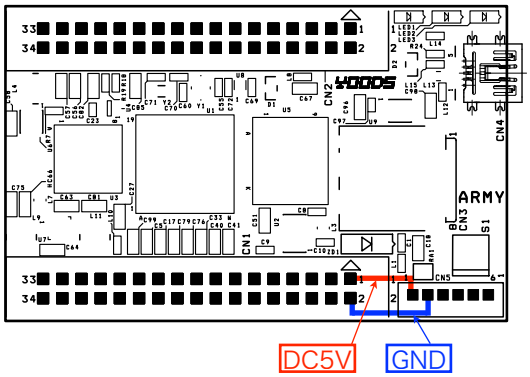


図2 ARM-YのDC5V電源供給

USBによる電源供給

USBからDC5V供給することもできます。この場合は基板裏のJP1をゼロオーム抵抗やリード線をハンダ付けしてジャンパーする必要があります。USBの規格上0.5A以上の電流を流すことはできませんので、注意してください。また、ヘッダピンとUSBの両方から電源を供給しないでください。ARM-Yや電源供給源の故障を発生する可能性があります。

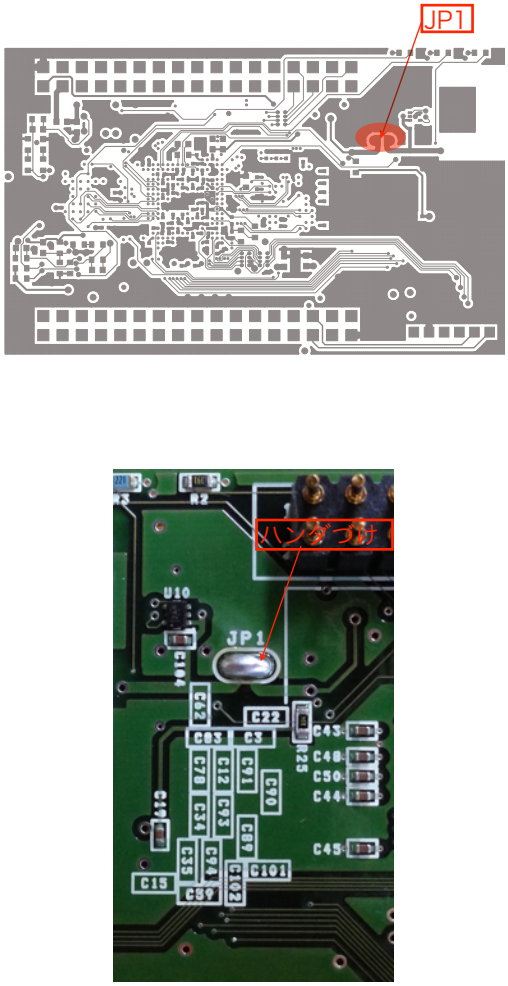


図3 ARM-YのUSBによるDC5V電源供給

OTGによるEthernet接続方法

ARM-Yを単体で使用する場合、そのままで使えるインターフェイスはUSBしかありません。ARM-Y付属のHIGH SPEED USBポートはOTG対応ので、パソコンと直接イーサーネット接続することができます。

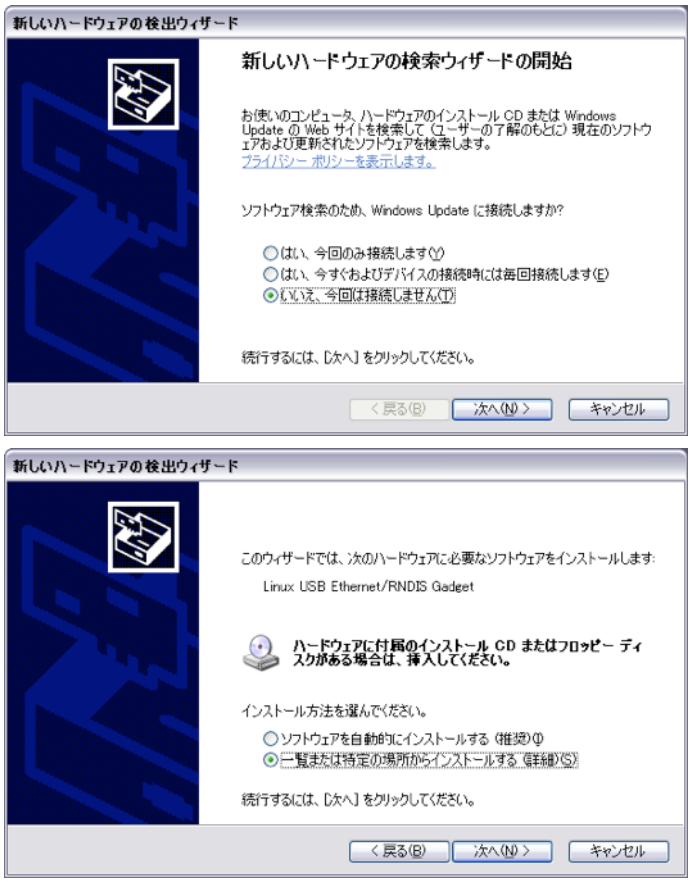
この章では、まずARM-Yをイーサーネットに接続してパソコンから設定ができるようにします。ARM-Yとパソコンを接続するUSBケーブルはOTGケーブルとして販売されていますので、別途入手しておいてください。

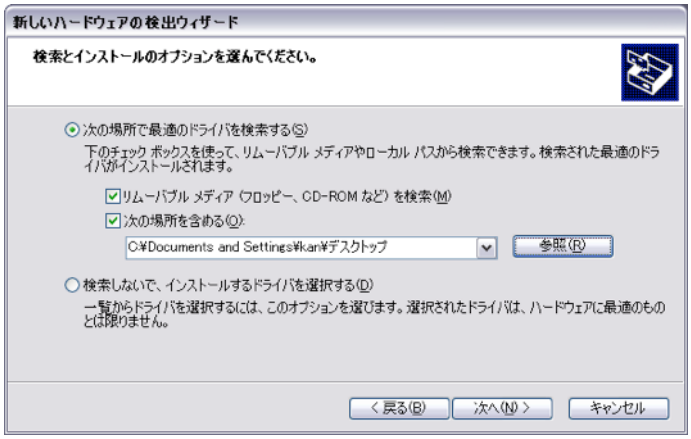
Windows XPへの接続方法

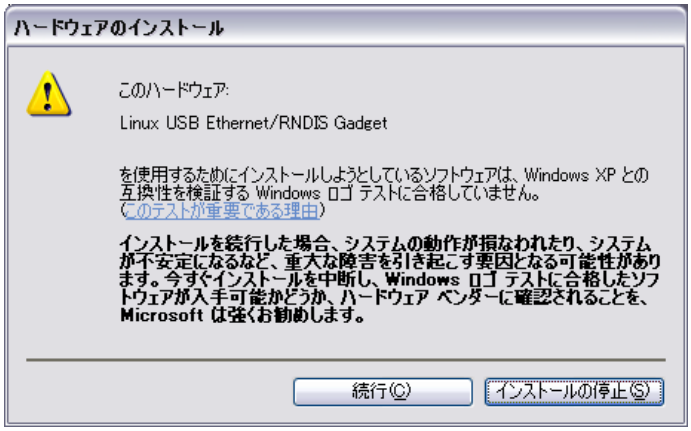
1.「付録」にあるダウンロードサイトより、ドライバ情報ファイルlinux.infをWindows XPにコピーしてください。

2.USB OTGケーブルでパソコンとARM-Yを接続してください。

3.ARM-Yの電源を投入してください。ARM-Yが起動して、パソコンがUSB上にARM-Yを検出すると、以下のようにデバイスドライバインストールのダイアログが開きます。通常のドライバインストール手順に従ってデバイスドライバをインストールします。ドライバのINFファイルの場所はlinux.infを配置したフォルダを指定してください。以下の例ではデスクトップにinfファイルを配置しています。







4.インストールが完了すると、デバイスドライバに「Linux USB Ethernet/RNDIS Gadget」が登録されていることが確認できます。

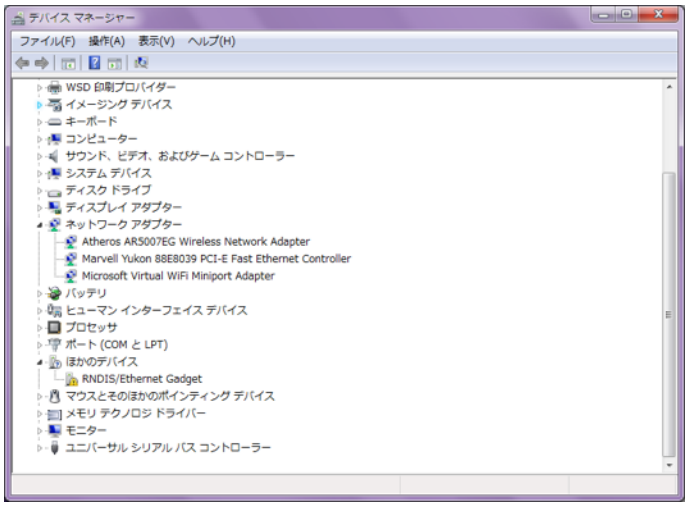
5.認識したデバイスにネットワーク設定を行ってください。出荷時のARM-Yのアドレス192.168.2.252と設定されているので、認識したネットワークデバイスには192.168.2.xxxのアドレスを設定します。

✓ARM-YがWindows PCを経由してインターフェイスに接続することも可能です。方法についてはインターネット等で調べてください。

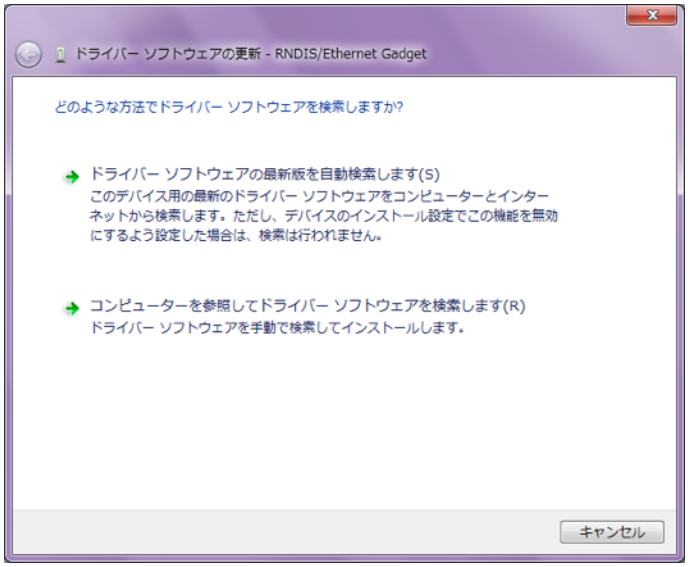
Windows 7への接続方法

1.USB OTGケーブルでパソコンとARM-Yを接続してください。

2.ARM-Yの電源を投入してください。ARM-Yが起動して、パソコンがUSB上にARM-Yを検出すると、デバイスドライバが自動的にインストールされます。　デバイスドライバダイアログから「他のデバイス/RNDIS/Ethernet Gadget」を選択して開いてください。

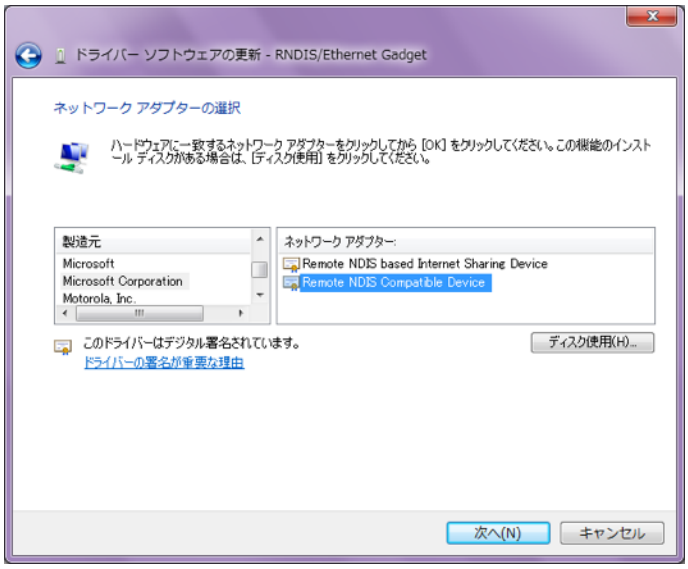


3.ドライバ詳細ダイアログから「ドライバの更新」を行います。

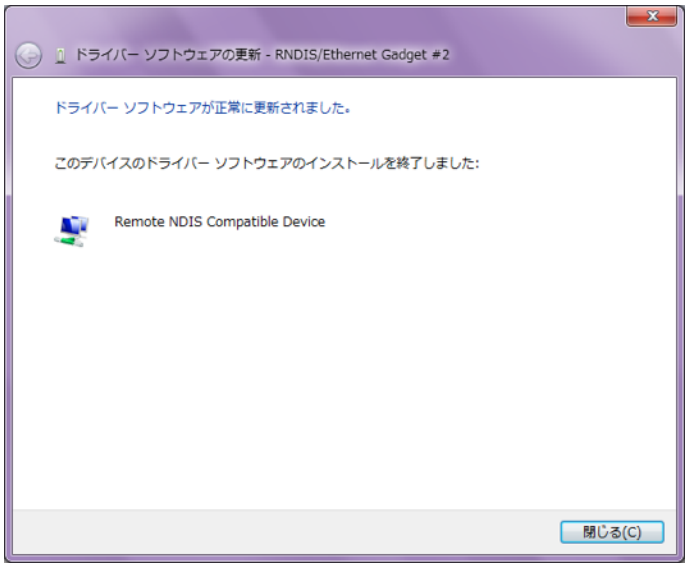


「コンピュータを参照してドライバーソフトウェアを検索します」を選択してください。

4.ネットワークアダプタを選択して、更にその中から「Microsoft Corporation」の「Remote NDIS Compatible Device」を選択してインストールしてください。



5.これでインストールが完了です。



新しく出来たネットワークドライバに192.168.2.1/255.255.255.0のアドレスを設定してください。

その後、ARM-Yの電源を入れ直すと接続ができます。

Linux(Ubuntu)への接続方法

1.特に設定は必要ありません。ARM-Yが起動すると「usb0」というネットワークデバイスで認識するので、/etc/network/interfacesにusb0デバイス用設定を設定してください。

2.認識したデバイスにネットワーク設定を行ってください。出荷時のARM-Yのアドレス192.168.2.252と設定されているので、認識したネットワークデバイスには192.168.2.xxxのアドレスを設定します。

ARM-Yの起動と接続確認

ここまでで、ARM-Yに電源を供給してネットワーク接続することができました。

pingによる接続確認

パソコン(Windows,Linux等)がARM-Yを認識しているかどうか確認するために、パソコンからpingを打ってみてください。

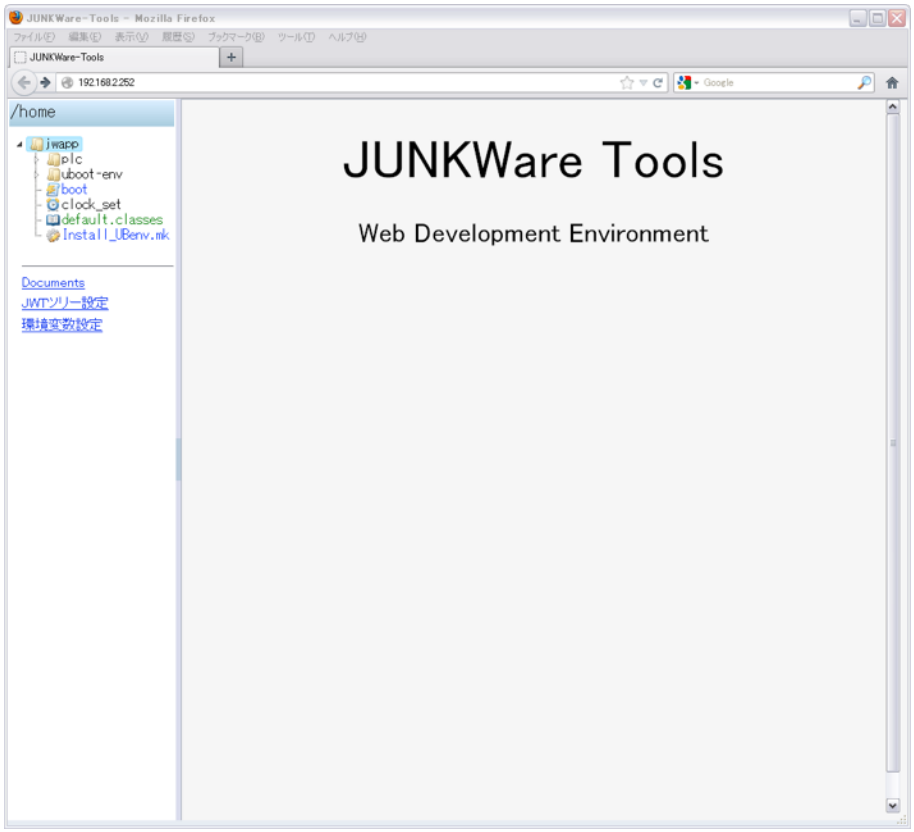
```
$ ping 192.168.2.252
PING 192.168.2.252 (192.168.2.252): 56 data bytes
64 bytes from 192.168.2.252: icmp_seq=0 ttl=64 time=1.918 ms
64 bytes from 192.168.2.252: icmp_seq=1 ttl=64 time=0.757 ms
64 bytes from 192.168.2.252: icmp_seq=2 ttl=64 time=0.742 ms
64 bytes from 192.168.2.252: icmp_seq=3 ttl=64 time=0.829 ms
64 bytes from 192.168.2.252: icmp_seq=4 ttl=64 time=0.801 ms
64 bytes from 192.168.2.252: icmp_seq=5 ttl=64 time=0.757 ms
^C
--- 192.168.2.252 ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.742/0.967/1.918/0.426 ms
```

接続できている場合には、上記のようにARM-Y(192.168.2.252)からpingの応答があります。

JWTによるパソコンとARM-Yの接続

JWTによる接続

ARM-Yには出荷時、Linux及び、Webベースの開発環境「JWT」がインストールされています。JWTを用いて各種設定の変更作業等が可能です。ブラウザを開いて、`http://192.168.2.252/`にアクセスしてみてください。

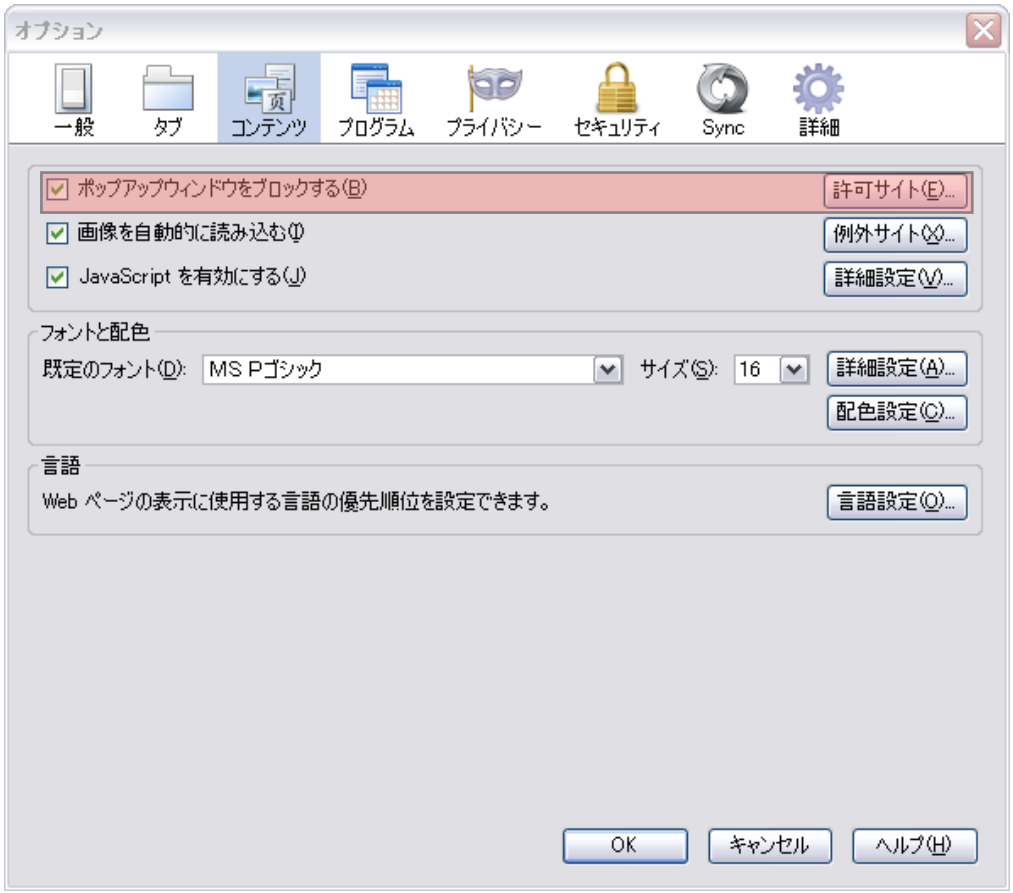


上記のような画面が開きます。JWTは左側のペインからファイルを選択することにより、そのファイルを編集したり、起動させる等の処理を提供しています。

※ブラウザはFirefoxを使用してください。他のブラウザでも動作しますが、レイアウト等が崩れる場合があります。

ポップアップウインドウブロックの解除

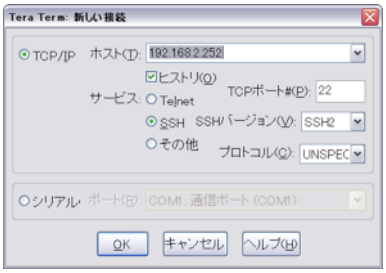
ブラウザはセキュリティのためにデフォルトでポップアップウインドウをブロックする設定がされていることがあります。オプション設定でARM-Y(192.168.2.252)からのポップアップ要求をブロックしないように設定しておいてください。

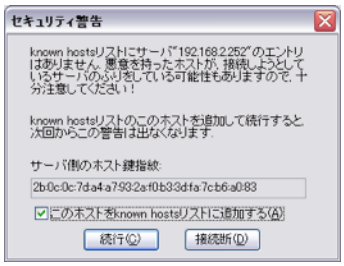


SSHによる接続

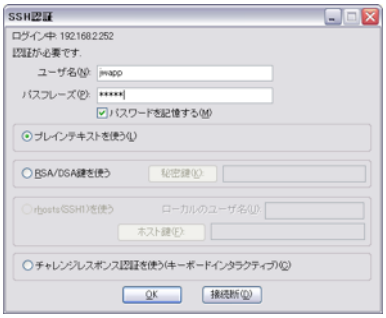
JWT以外に、汎用SSHクライアントで接続することも可能です。以下では、Windows上でTeratermを用いて接続する場合の手順を示します。

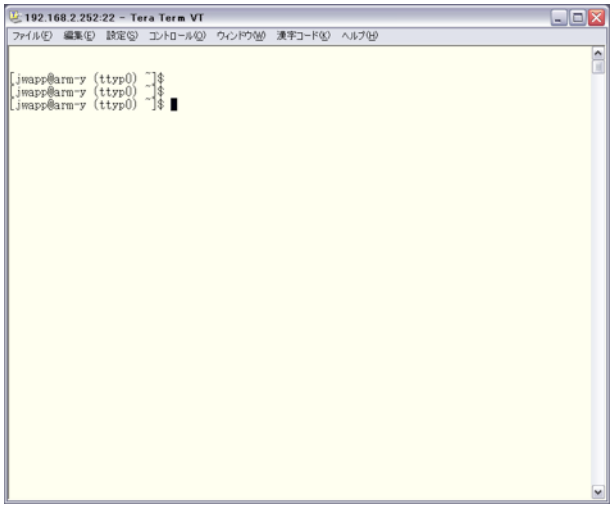
(1) Teratermを起動してTCP/IP, SSH接続を選択します。接続ホストはARM-Yのアドレス(192.168.2.252)を指定してください。セキュリティ警告は書きダイアログのように対応してください。





(2) ユーザーID/パスフレーズはjwapp/jwappを使ってください。

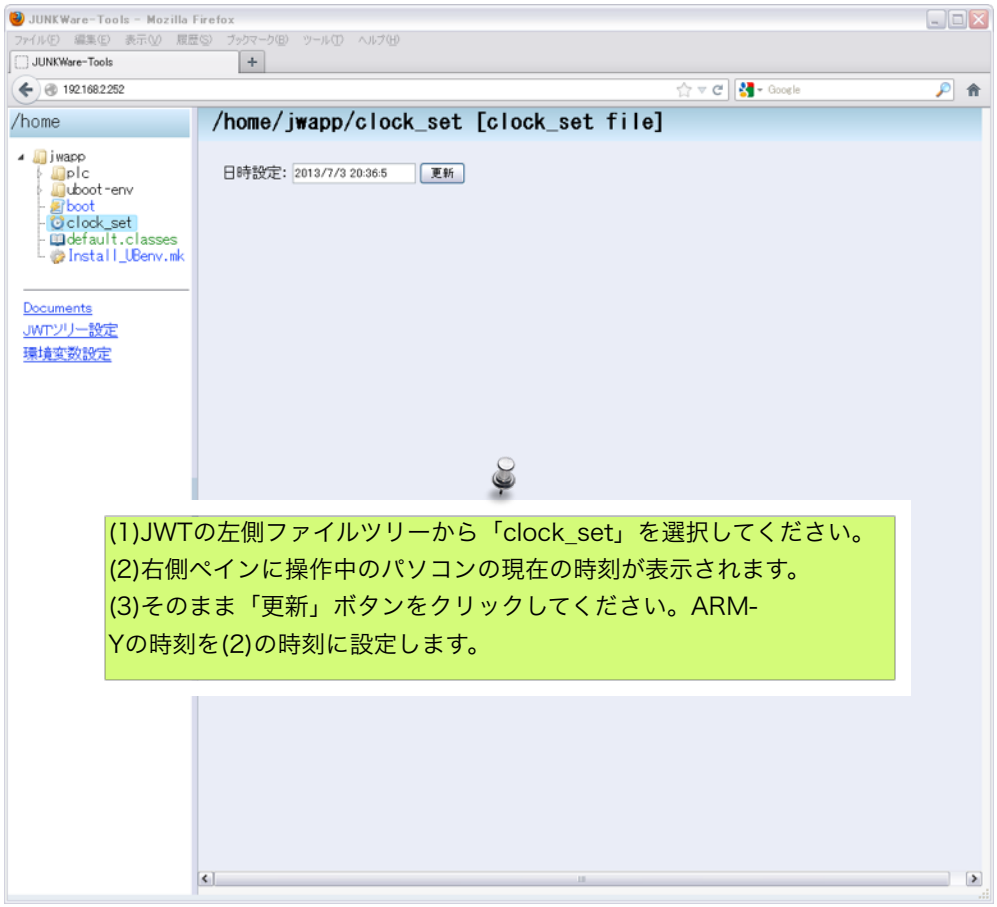




時刻同期について

手動時刻合わせ

ARM-Yはカレンダータイマのバックアップを持っていないため、電源投入時はカレンダーが初期化されています。JWTでは簡単に時刻を設定する機能を提供しているので、以下の手順で時刻を設定してください。



✓時刻を設定することで、開発中に作成・変更したファイルのタイムスタンプの整合性をとることができ、デバック等の作業が楽になります。

PIN MUXの設定

ARM-YはCPUのIOピンをCN1,CN2の2.54mmピッチヘッダーピンに接続しています。ほとんどのピンはマルチプレクスされているため、マルチプレクス設定を変更することにより、各種IOデバイスを接続することができます。

ARM-Yはピンマルチプレクス設定用のドライバ(arm-y_drv.ko)をインストールしています。このドライバの設定を書き換えて、使いたいピンの設定を変更することができます。具体的な設定方法を以下に示します。

CN1のPIN配列

ピン番号	mux(1)	mux(2)	mux(3)	PU/PD	備考
1	DC5V				ARM-Y駆動電源入力
2	GND				
3	UART2_RXD				UART2 コンソールとしてトランシーバー実装済み。
4	UART2_TXD				
5	UART2_CTS				
6	UART2_RTS				
7	LCD_D0	GPIO7_8		PD	
8	LCD_D1	GPIO7_9		PD	
9	LCD_D2	GPIO7_10		PD	
10	LCD_D3	GPIO7_11		PD	
11	LCD_D4	GPIO7_12		PD	
12	LCD_D5	GPIO7_13		PD	
13	LCD_D6	GPIO7_14		PD	
14	LCD_D7	GPIO7_15		PD	
15	LCD_D8	GPIO7_0		PD ブートモードSWの設定により変化します。	ブートモード設定に使用するため、電源投入時は使用できません。また、使用する場合は、ブートモードDIP SWと競合しないようにバッファ等で分離する必要があります。
16	LCD_D9	GPIO7_1			
17	LCD_D10	GPIO7_2			
18	LCD_D11	GPIO7_3			
19	LCD_D12	GPIO7_4			
20	LCD_D13	GPIO7_5			
21	LCD_D14	GPIO7_6			
22	LCD_D15	GPIO7_7			
23	LCD_AC_ENB	GPIO_6_0		PU	
24	LCD_VSYNC	GPIO8_8		PU	
25	LCD_HSYNC	GPIO8_9		PU	
26	LCD_MCLK	GPIO8_10		PU	
27	LCD_PCLK	GPIO8_11		PU	
28	AXR0	CLKS0	GPIO8_7	PU	
29	AXR1	DX0	GPIO1_9	PU	
30	AXR2	DR0	GPIO1_10	PU	
31	AXR3	FSX0	GPIO1_11	PU	
32	AXR5	CLKX0	GPIO1_13	PU	
33	AHCLKX	UART1_CTS	GPIO0_10	PU	UART1
34	AHCLKR	UART1_RTS	GPIO0_11	PU	

CN2のPIN配列

ピン番号	mux(1)	mux(2)	mux(3)	PU/PD	備考
1	RMII_MHZ_50				
2	RMII_RXD1				
3	RMII_RXD0				
4	RMII_RXER				
5	RMII_CRS_DV				
6	RMII_TXEN				
7	RMII_TXD0				
8	RMII_TXD1				
9	RESETOUT	GPIO6_15		PU	
10	CLKOUT	GPIO6_14		PU	
11	USB1_DP				
12	USB1_DM				
13	SPI1_CLK	GPIO2_13		PU	
14	SPI1_ENA	GPIO2_12		PU	
15	SPI1_SOMI	GPIO2_11		PU	
16	SPI1_SIMO	GPIO2_10		PU	
17	SPI1_SCS7	I2C0_SCL	GPIO1_5	PU	I2C0
18	SPI1_SCS6	I2C0_SDA	GPIO1_4	PU	
19	SPI1_SCS3	UART1_RXD	GPIO1_1	PU	UART1
20	SPI1_SCS2	UART1_TXD	GPIO1_0	PU	
21	SPI1_SCS1	GPIO2_15		PU	
22	SPI1_SCS0	GPIO2_14		PU	
23	SPI0_SCS5	UART0_RXD	GPIO8_4	PU	UART0
24	SPI0_SCS4	UART0_TXD	GPIO8_3	PU	
25	SPI0_SCS3	UART0_CTS	GPIO8_2	PU	
26	SPI0_SCS2	UART0_RTS	GPIO8_1	PU	
27	SPI0_SCS1	GPIO1_7	MDIO_CLK	PU	SPI0
28	SPI0_SCS0	GPIO1_6	MDIO_D	PU	
29	SPI0_CLK	GPIO1_8		PU	
30	SPI0_SOMI	GPIO8_6		PU	
31	SPIO0_SIMO	GPIO8_5		PU	
32	+VIO				DC3.3V 拡張ボードのIO用電源, 最大電流500mA
33	GND				
34	RTC+VIN				RTCバックアップ電源を接続

✓上記CN1,CN2のPIN配列表ではMUXのデフォルト設定が緑で着色してあります。

PIN MUXの設定

PIN MUXを切替える方法

通常、PIN MUXを切替えるためにはカーネルのソースを変更しますが、ARM-Yではこの変更をテキストリソースで変更できる手段を提供しています。具体的には、以下のフォーマットで/dev/armyioにテキストを書込みます。

```
cn[1/2]_[pin no]_[mux_type](_[I/L/H])
```

[mux_type]は、 `gpio / lcd / mcasp / mcbsp / resetout / clkout / i2c / spi / mdio` を指定してください。`uart`は特に設定コマンドを用意していません。`uart0,1,2`のいずれもデフォルト設定となっているので、`MUX`設定を変更せずに使ってください。

最後の `(_[I/L/H])` 部分はGPIOの場合のみのオプションで、`I`は入力モード, `L/H`は出力モード時の初期値を設定します。

例えば、CN1のPIN番号8をGPIOに設定、更に出力モードにして初期値をHigh Levelとする場合は以下のようにします。

```
echo "cn1_8_gpio_H" >/dev/armyio
```

MUX設定が複数ある場合は、設定を書込んだファイルを作成して(以下の例ではpinmux.conf)、まとめて書込むことも可能です。

(1) /home/jwapp/configs/pinmux.confの作成

```
cn1_8_gpio_H
cn1_9_gpio_L
cn2_23_GPIO_I
cn2_24_GPIO_I
cn2_25_GPIO_I
cn2_26_GPIO_I
```

(2) PIN MUXの設定

```
cat /home/jwapp/configs/pinmux.conf >/dev/armyio
```

PIN MUXをどこで設定するか

上記方法でPIN　MUXはテキストコマンドにより変更できますが、ARM-Yを使ったシステムのどこでそれを設定すれば良いでしょうか？PIN MUXはシステム起動時に一度だけ実行すれば良いので、何度も呼ばれる場所は適当ではありません。

※rc.localはLinuxのブートプロセスで最後に呼び出されます。ここにPIN MUXの設定変更を書いておきましょう。

JWTでARM-Yに接続して、上図のように/home/jwapp/configs/rc.localを開いて、PIN MUXコマンドを書込んでおきます。

✓現在のところ、この方法でのPIN　MUXの設定は、Multichannel Audio Serial Ports (McASP), Multichannel Buffered Serial Ports (McBSP),Liquid Crystal Display Controller(LCD)には対応していません。

サンプル回路の作成とJUNKWareを使ったプログラム作成

ARM-Yを使ったサンプルとして、ARM-Yをユニバーサル基板に載せて周辺回路を作ってみます。また、ARM-YからIOへのアクセスは、プリインストールされているJUNKWareを使います。

✓JUNKWareを使わずにプログラムでIOにアクセスするためには、コンパイラ等を含む開発用マイクロSDが必要になります。この方法については「 開発環境(マイクロSD)の使用方法」を参照してください。

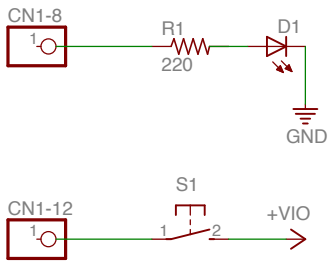
練習用回路図

ここで作成する回路では、以下のことを前提とします。

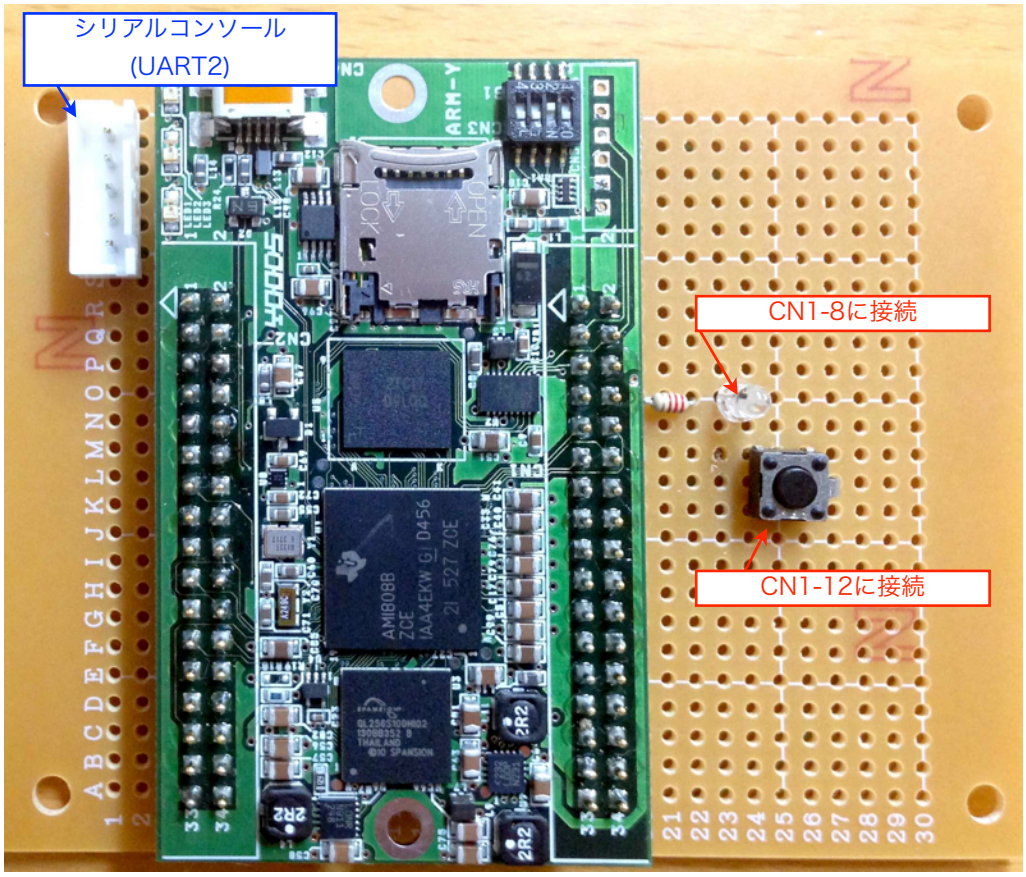
•電源はUSBから供給するようにARM-YのJP1をハンダで接続

•IO用電源はARM-YのCN2/32,33の+VIO,GNDを使用

それでは、CN1の8,12番ピンにLEDとスイッチを接続します。回路図で表すと以下のようになります。



実際のボードは以下のようになります。



✓このサンプルボードではシリアルコンソールが使えるように、CN1-2～CN1-6のピンをXH5ピンコネクタに引き出しています。

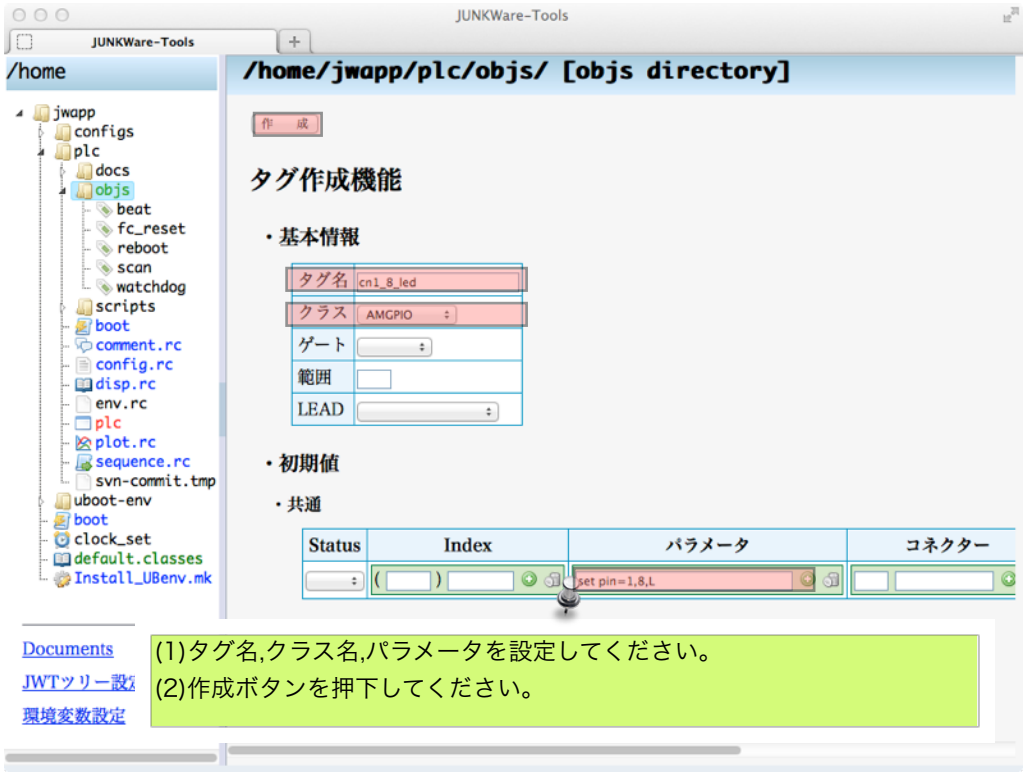
JUNKWareによる動作確認

作成したボードの動作確認をJUNKWareを使って行います。JUNKWareを使うことにより、プログラミングレスでGPIO入出力の設定・動作確認まで行うことができます。

予め用意してあるJUNKWareのクラス「AMGPIO」から生成したオブジェクトを使用するピンに割当てます。JUNKWareのオブジェクトは/home/jwapp/plc/objs/にまとめて定義します。基本的にひとつのファイルがひとつのオブジェクトを表します。　/home/jwapp/plc/objs/を選択すると新しいオブジェクトを定義する画面が開くので、この機能を使ってCN1/8,12番ピンのGPIOオブジェクトを作成してください。

i)出力ピンの設定

作成したボードはCN1の8番ピンにLEDを接続しているので、このピンを出力GPIOとするオブジェクトを定義します。オブジェクトの名前が「タグ名」になるので「cn1_8_led」という名前にします(タグ名は半角15文字以内で任意です)。クラスとしてAMGPIOを選択して、パラメータの部分に使用するピンパラメータを記述します。「`set pin=1,8,L`」と書いてください。設定内容はコネクタ番号,ピン番号,初期レベルです。初期レベル「`H`」を指定することもできます。



全てのパラメータ設定後、「作成」ボタンを押下してください。左側のファイルツリーに「cn1_8_led」が追加されます。

ii)入力ピンの設定

同様にして、以下の図を参考に12番ピンをスイッチ入力として定義してください。

入力の場合JUNKWareでは割込を使わないので、周期的にポーリングする必要があります。そのスキャン周期をゲートオブジェクトで指定します。ここではひとまずscan.0を指定しておいてください。

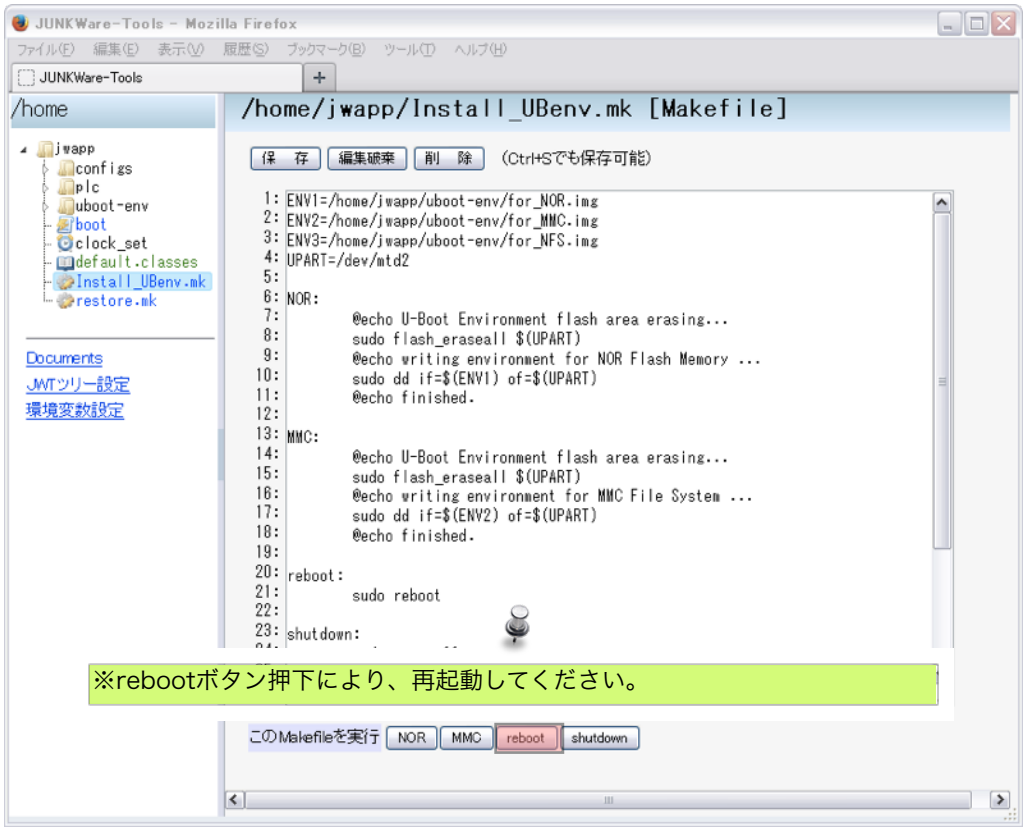
パラメータの最後は入力ピンとして使用するために「I」(input)を指定してください。

またGPIOはActive　Lowのため、ボタン解放時ONとなっています。そこでinvertパラメータをonに設定しています。これによりボタンリリース時off，押下時onになります。

✓scan.0,scan.1,,,の動作周期は起動用bootスクリプト内で設定しています。

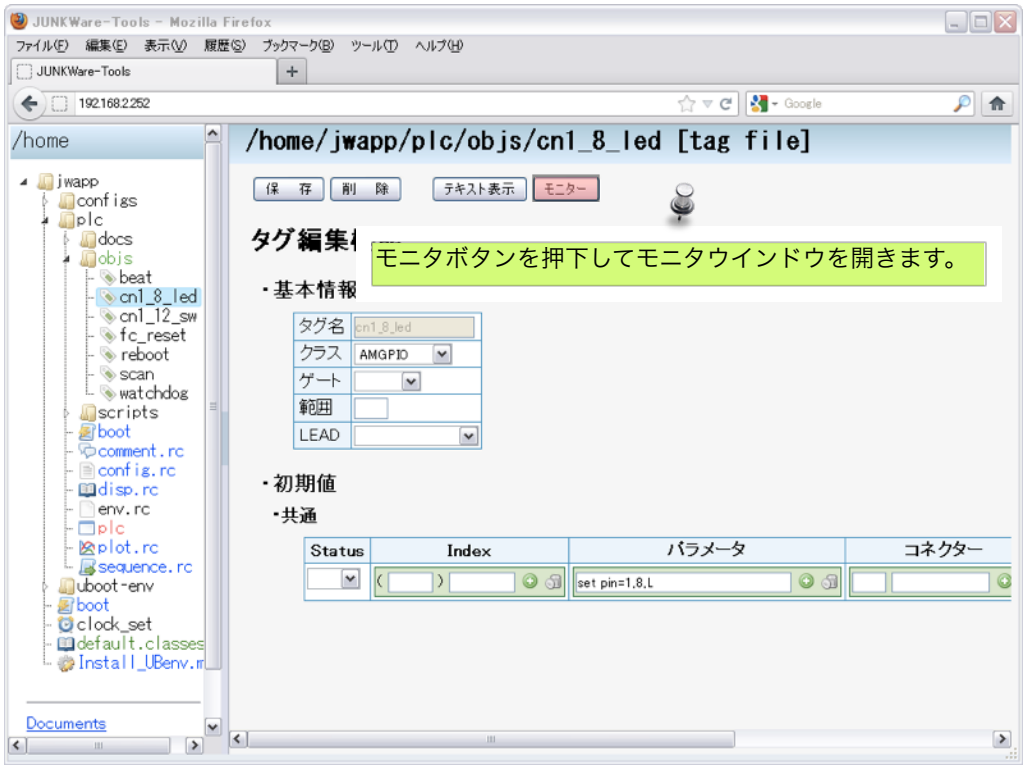
iii)再起動による設定の反映

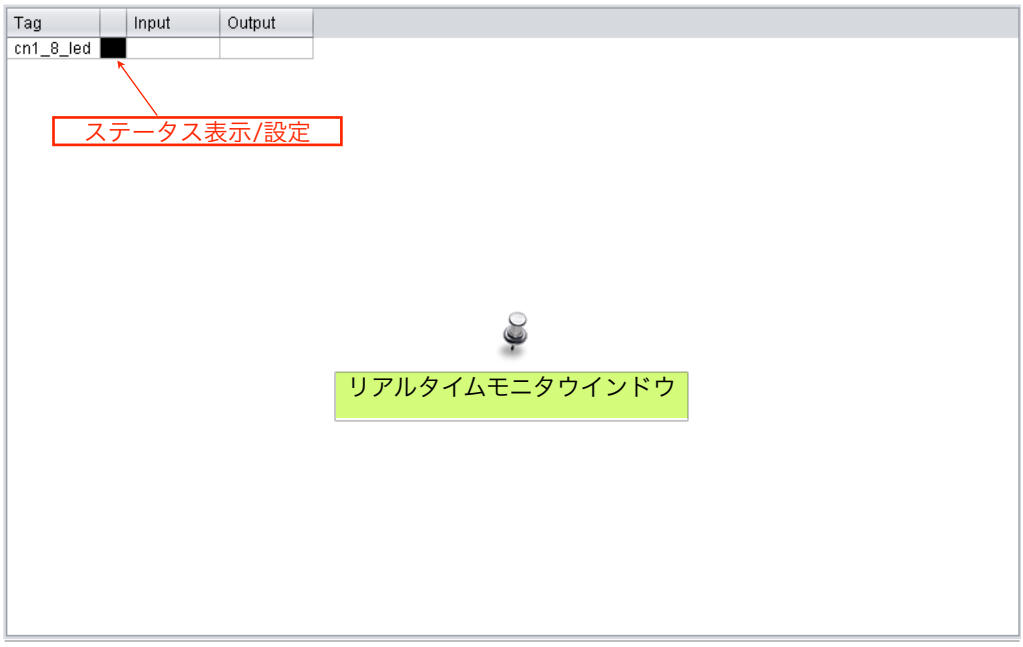
これで2本のピンの設定を終了しました。/home/jwapp/Install_UBenv.mkを選択するとこのmakefileに再起動を実行するターゲットが定義してあります。下部の「reboot」ボタンを押下してARM-Yを再起動してください。



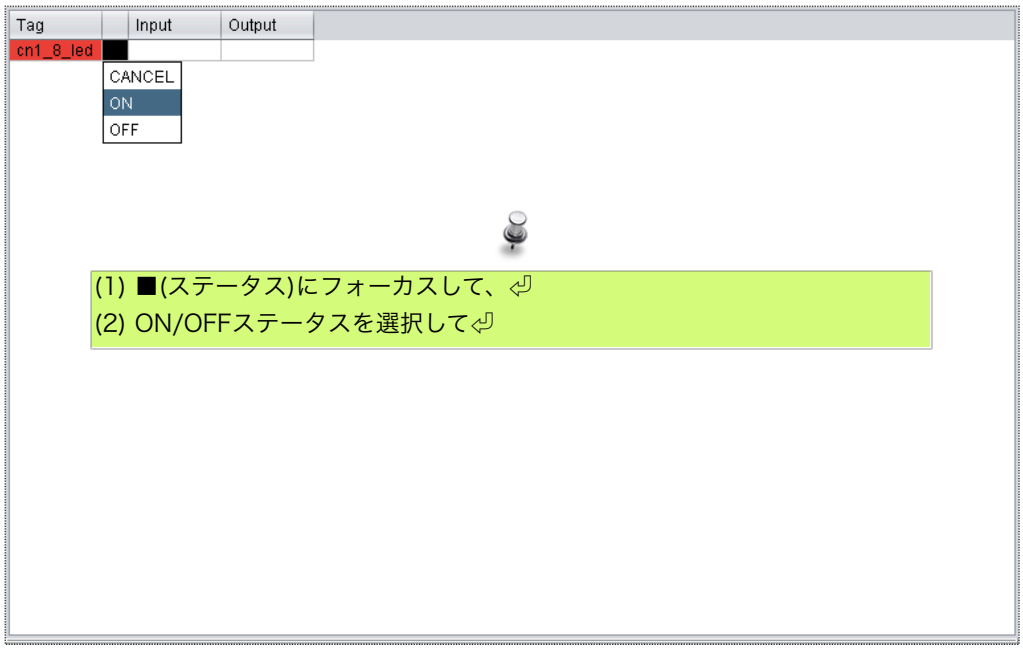
iv)試作ボードの動作確認(dispparによる)

JWTのオブジェクトのリアルタイムモニタ機能を使ってARM-Yに接続したLEDとスイッチの動作を確認してみます。/home/jwapp/plc/objs/cn1_8_ledを選択してください。右側ペインにの上部にある「モニタ」ボタンをクリックするとリアルタイムモニタ用ウインドウが開きます。JUNKWareではこのウインドウを「disppar」と呼びます。

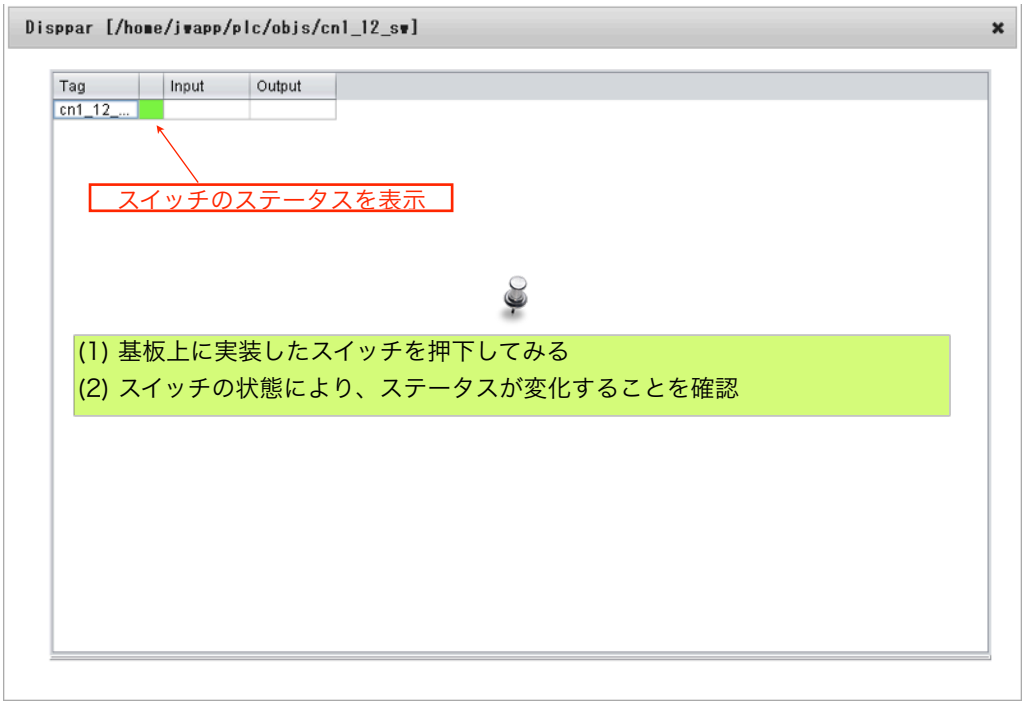




dispparウインドウでタグ名の横にある■は、このオブジェクトのステータスを示しています。この部分をクリックしてフォーカスをあててください。その後⏎キーを叩くと以下のようなダイアログが現れて、「カーソルキー選択⇒⏎確定」により、このLEDを接続したオブジェクトのステータスを変更できます。

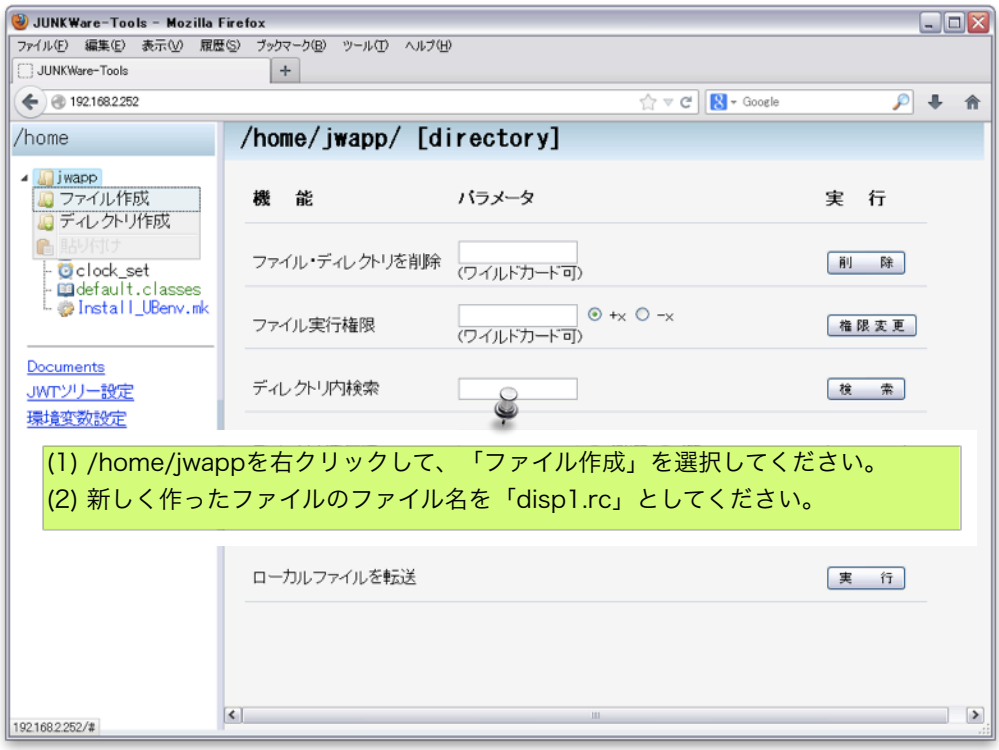


同様にして、今度はスイッチ用オブジェクトをdispparで開いてください。



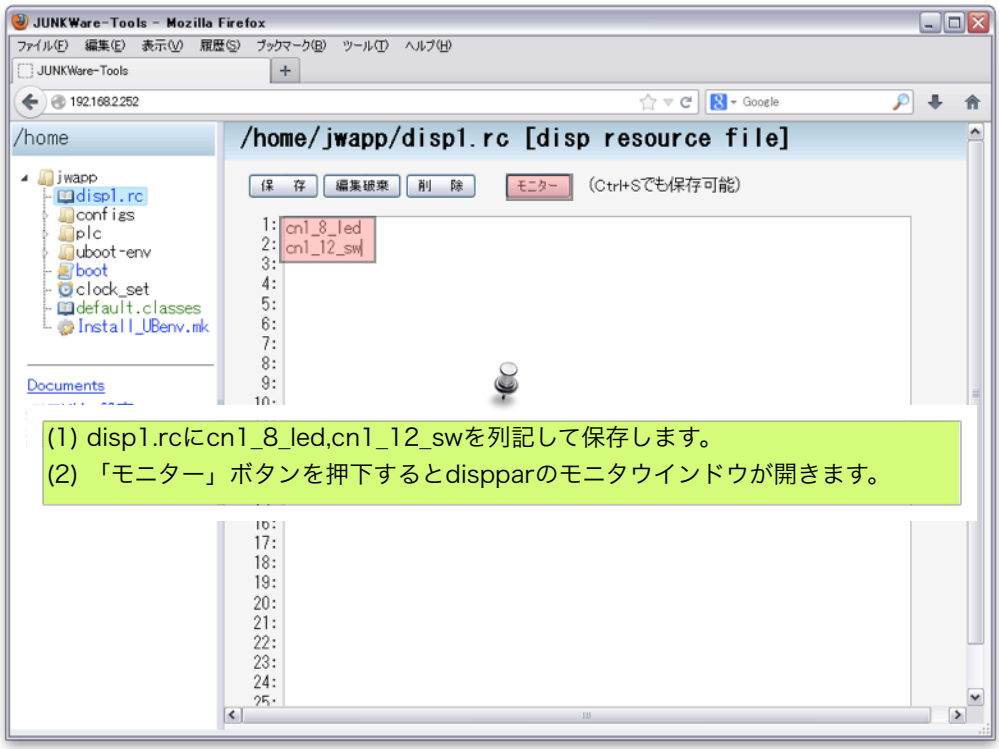
スイッチのON/OFF状態がcn1_12_swオブジェクトのステータスにリアルタイムに表示されていることが確認できればOKです。

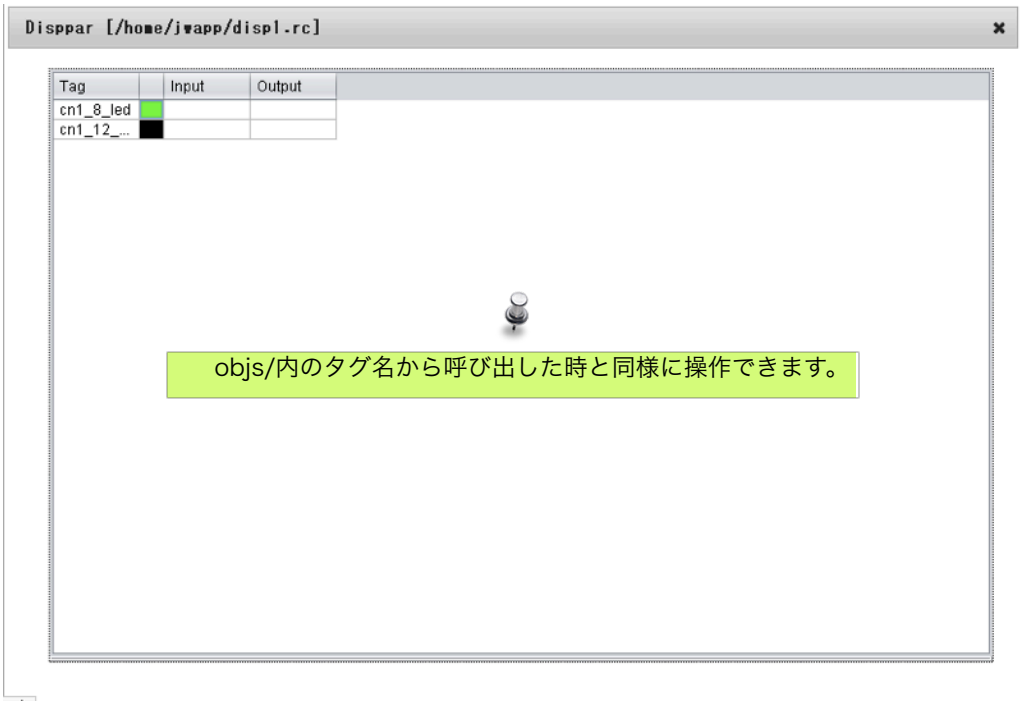
dispparでモニタするタグを、まとめて一度に表示することができます。今回の場合、cn1_8_ledとcn1_12_swを一度にdispparに表示してみます。



場所はどこでも構いませんが、任意のディレクトリ(objs等は除く)を右クリックして、そのディレクトリ内に新しいファイルをdisp*.rcと名前をつけます(*はワイルドカード)。JWTはこのファイル名で動作を決定しています。

作成したdisp1.rcにモニタしたいタグ名を列記して保存してください。その後、「モニター」ボタンを押下するとdisp1.rcに記述したタグを一度に表示したdispparウインドウが開きます。





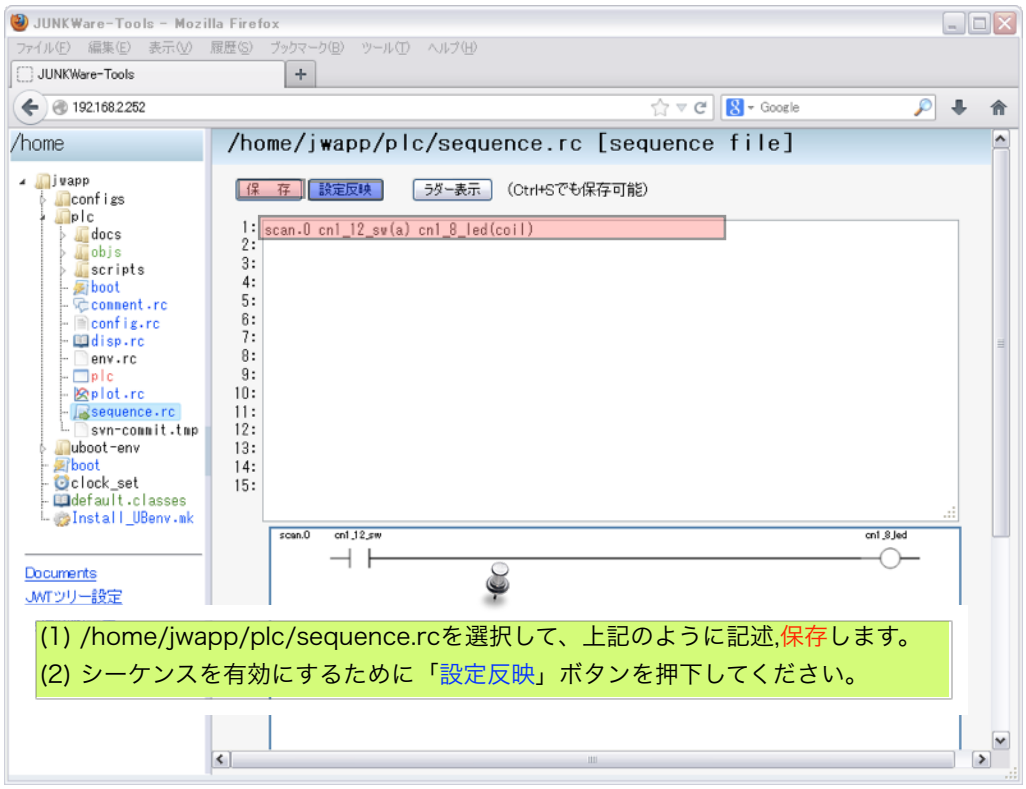
このように、モニタや設定を行いたいオブジェクトをまとめてdispparに表示することができます。

応用～JUNKWareによるLEDとスイッチの連動

もうひとつ、応用としてスイッチとLEDを連動してみます。

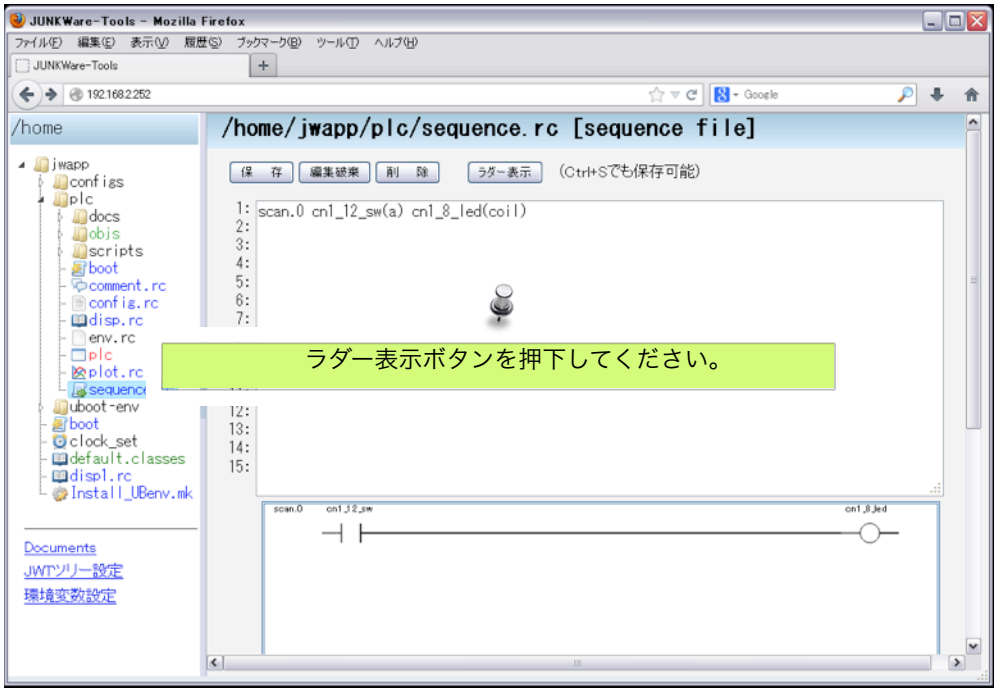
i)動作シーケンスの記述

「スイッチが押された→LEDを点灯」という動作をシーケンス回路と呼びます。これを/home/jwapp/plc/sequence.rcに記述･保存します。保存後、シーケンス回路を有効にするために「設定反映」ボタンを押下してください。

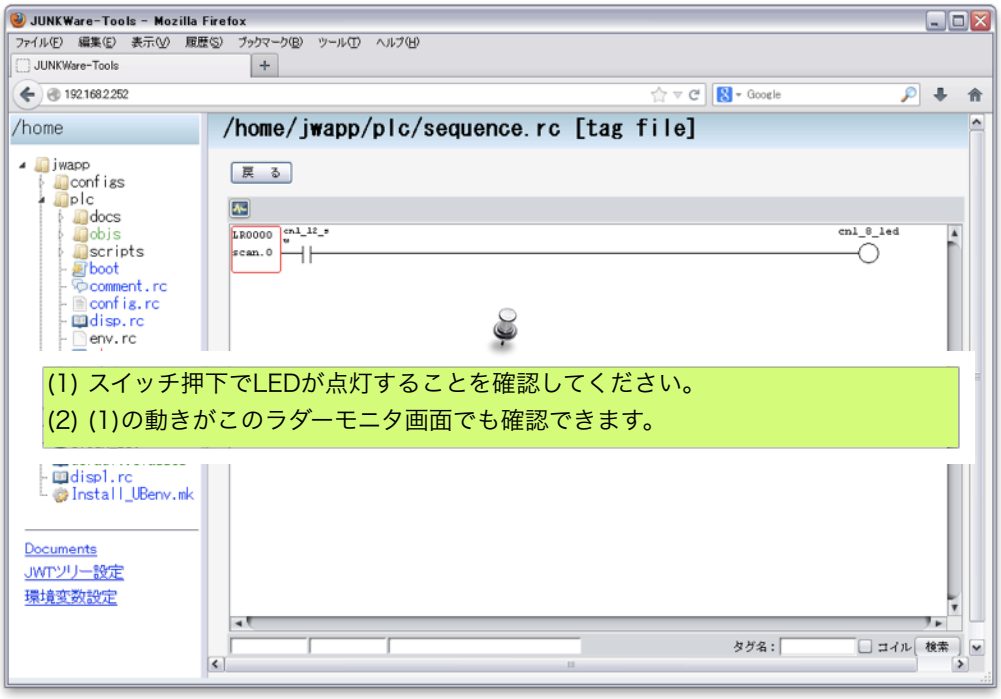


ii)動作確認

設定反映完了後、スイッチを押下するとLEDが点灯するようになればOKです。先程編集したsequence.rcを選択して、上段の「ラダー表示」ボタンを押下してください。シーケンス回路を表示したラダーモニタウインドウが開きます。スイッチを押下すると、スイッチのステータスがONになり、LEDが点灯する動作を確認できます。



⇩



JUNKWareを用いることにより、更に

•スイッチのステータスOFF時にLEDを点灯させる(反転)。

•LEDをフリッカーさせる。

等のプログラムが簡単に作成できます。詳しくはJUNKWareのマニュアルを参照してください。

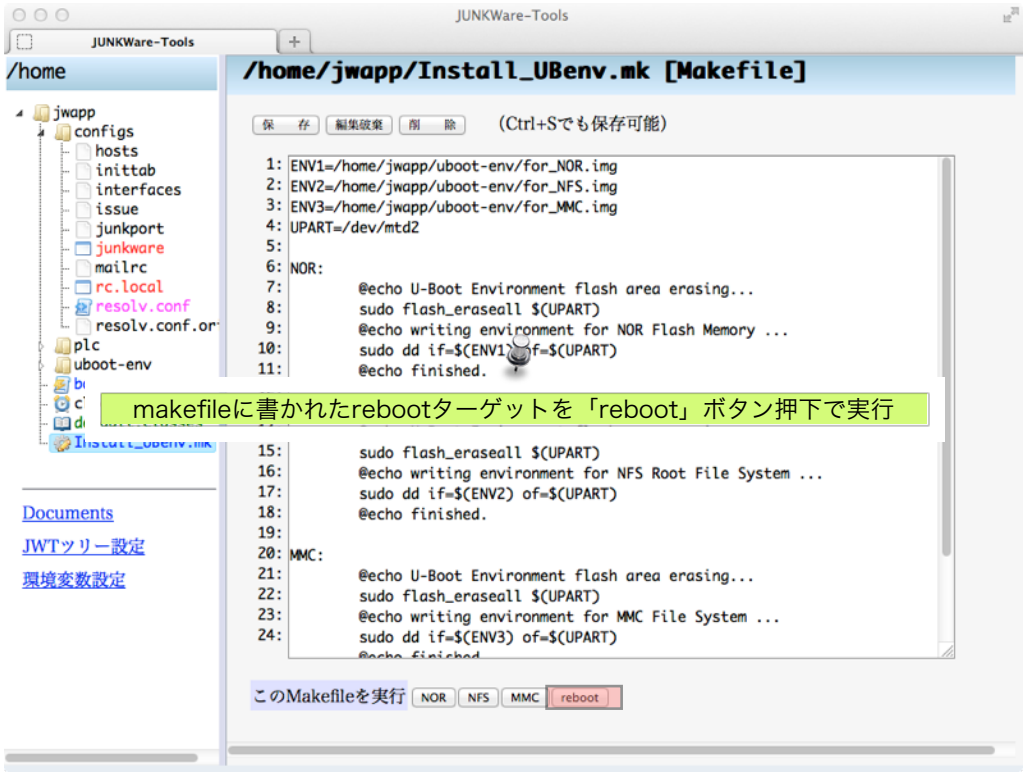
JUNKWareによるARM-Y上のLED制御方法

ARM-Yには汎用LEDとして、赤と黄色のLEDが実装されています。これらのLEDはプログラムで制御できます。JUNKWareではobjs/led_R, led_Yとして定義してあります。これまでの応用として、このLEDとスイッチを連動させるなど、トライしてみてください。

付録

ARM-Yの再起動方法

設定後再起動が必要な場合等、JWTから再起動をかけることができます。/home/jwapp/Install_UBenv.mkを選択して、以下の方法で再起動をかけてください。



開発キットについて

ARM-Yのベースのシステム開発ツールとして別売の開発キット(8,000円)があります。必ずしもこれが必要という訳ではありませんが、必要に応じてお求めください。

開発キットの内容

- 開発環境マイクロSD(Debian-Lennyベース)
- CN5用シリアルコンソール,電源用ケーブル(約10cm)
- CN5用PH6ピンコネクタ

✓マイクロSDの開発環境はYOODSのホームページ(http://www.yoods.co.jp)からダウンロード可能です。また、シリアルコンソールケーブルのケーブル図は本付録に記載されているので自作も可能です。

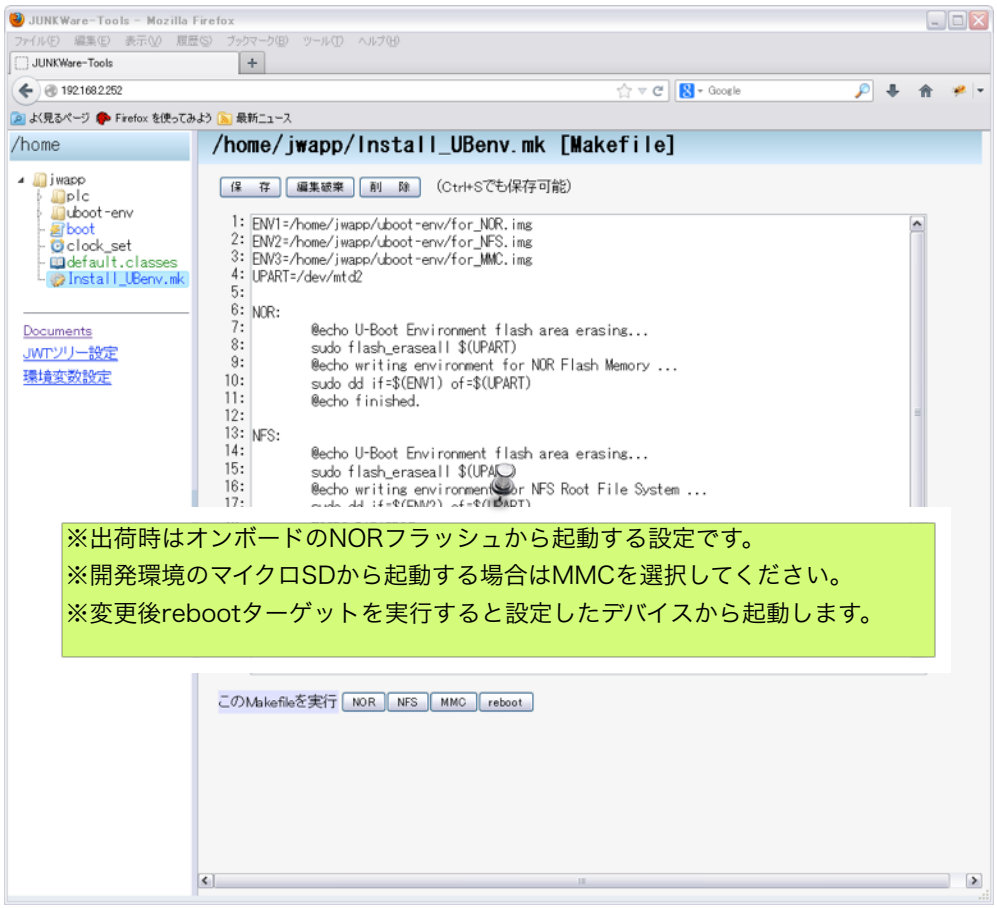
開発環境(マイクロSD)の使用方法

開発環境をインストールしたマイクロSDをARM-Yに搭載して、このマイクロSDからLinuxをブートすることにより、Debian　Lennyの開発環境によるソフトウェア開発を行うことができます。以下にその手順を説明します。

i)u-bootの起動パラメータの変更によるSDからのシステム起動

ARM-Yの起動デバイスはu-boot(ブートローダー)の起動オプションで指定します。起動オプションを直接u-boot上で書き換えることも可能ですが、ここではLinux上で書き換える方法を説明します。

書換えの手続きを/home/jwapp/Install_UBenv.mkに定義してありますので、下図中の説明に従って起動オプションを書き換えてください。



✓開発環境のマイクロSDがない状態でMMCブートを選択するとLinuxを起動することができなくなり、この方法でNORブートに戻すことができなくなります。このような状態になった場合は、付録に記載のシリアルコンソールによるu-boot起動パラメータの書換えにより、NORブートの設定を行ってください。

✓シリアルコンソールによる設定ができない場合は、実費(2000円/枚)にて設定サービスを致します。詳しくは(株)YOODSまでお問合せください。

開発用マイクロSDの内容について

開発用マイクロSDはDebian/Lennyをベースにしています。開発用ディレクトリとして提供している/home/jwappについて、以下に説明します。

/home/jwapp/	ディレクトリ/ファイル	説　　明
arm-y	classes	ARM-Y用JUNKWareクラスのソース(AMGPIO, WATCHDOG)
	configs	hosts,inittab,interfaces,issue,mailrc,ntp,rc.local,resolv,conf,JUNKWare関連設定ファイル
	plc	JUNKWareプロセス"plc"の構成ファイル群
	boot	JUNKWareプロセス"plc"の起動/停止スクリプト
	clock_set	JWT時刻合わせツール
	default.classes	JUNKWareクラス説明ドキュメント
	Makefile	JUNKWareプロセス"plc", NORフラッシュインストール用makefile
html	JWT構成ファイル/var/www/htmlへのシンボリックリンク	
Install	images	ARM-Yインストール用JFFS2イメージディレクトリ。 (1) rootfs.img: 基本システム (2) arm-y.jfs: ARM-Y出荷時システム
	rootfs	NORフラッシュ書込み用ファイルシステム(/home/jwappを含まない)
	Create_Userland_Base_img.mk	rootfsからrootfs.imgを更新するmakefile
	Install.in	Make用基本情報
	Install_Kernel.mk	NORフラッシュへkernelを書込むmakefile
	Install_Userland_.mk	NORフラッシュへrootfs.img,arm-y.jfsを書込む、又は書き出すmakefile

/home/jwapp/	ディレクトリ/ファイル	説　　明
	rootfs.tar.gz	rootfsのtarアーカイブ
	Update_modules.mk	rootfsの/lib/modulesを更新するmakefile
Packages	junkware	JUNKWareディレクトリへのシンボリックリンク
	kernel	カーネルソースへのシンボリックリンク
uboot-env	u-boot環境変数格納パーティションのイメージ群 (1) for_MMC.img: マイクロSDブート用設定 (2) for_NOR.img: NORブート用設定	
Install_UBenv.mk	u-boot環境変数格納パーティション更新用makefile	

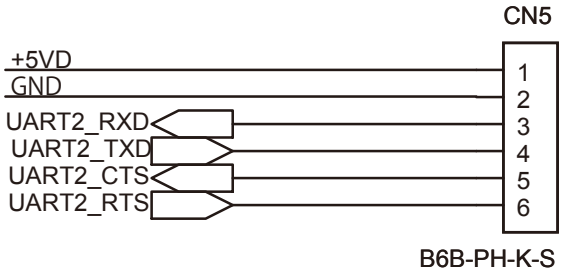
シリアルコンソールの使い方

ARM-YのCN5はUART2に接続されていて、トランシーバーを接続済みのシリアルコンソールに設定してあります。

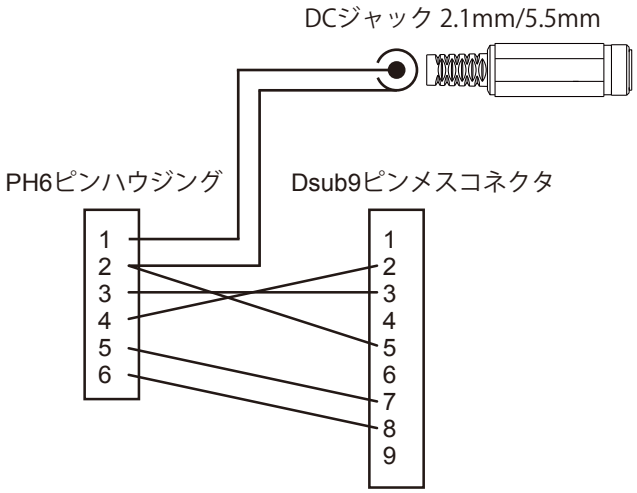
✓アプリ側でUART2を使う場合は、u-bootの設定及びLinuxの設定を変更する必要があります。

CN5(シリアルコンソールポート)の配線

CN5にPHコネクタを付けてシリアルコンソールを使用することができます。以下の図を参考にケーブルを作成してください。



開発キット付属のシリアルコンソールケーブルの仕様は、以下のようになっています。



シリアルコンソールの通信パラメータ

ARM-YのCN5にシリアルコンソールケーブルを接続することで、ARM-Yのシリアルコンソールを使用することができます。シリアル端末の通信条件は以下のように設定してください。

通信パラメータ	設定値
ボーレート	115200
データ長	8
ストップビット	1
パリティ	なし
フロー制御	ハードウェアフロー制御

ARM-Y起動後しばらくすると、シリアルコンソールに

3.1.0-ARM-Y-1.0

arm-y login:

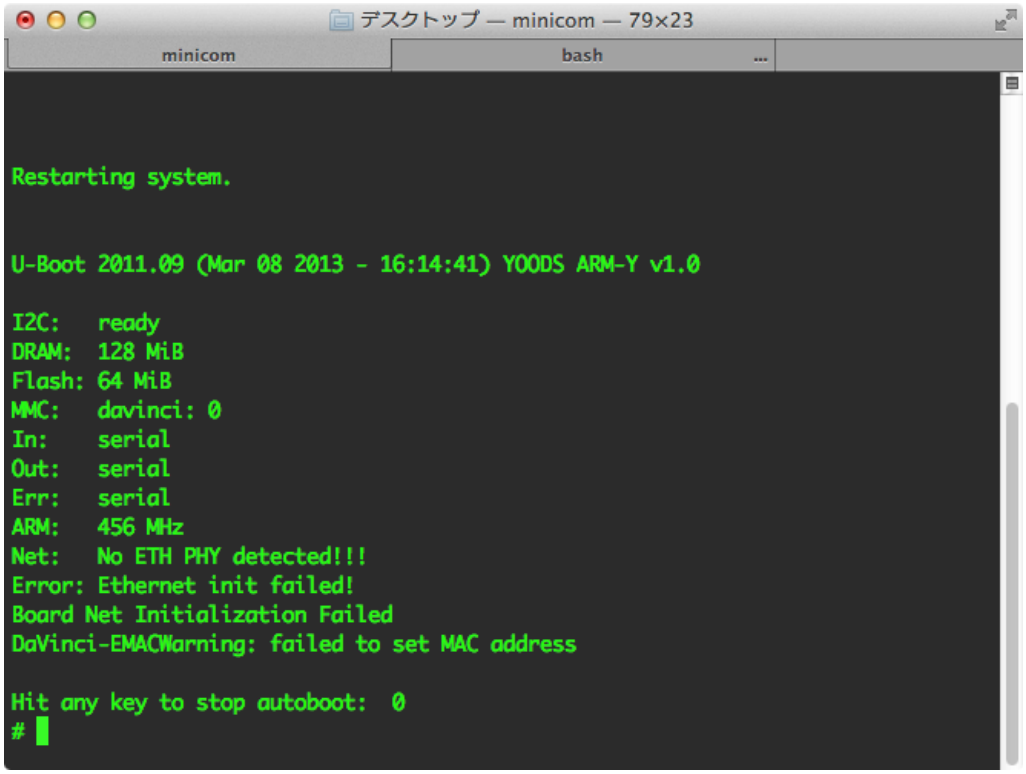
のようにログインプロンプトが表示されます。以下のID/パスワードでログインしてLinuxを操作することができます。

ID	パスワード
jwapp	jwapp

jwappユーザーでログイン後、rootになる場合はsuコマンドを使い、パスワード”yama”を使用してください。

u-bootの環境変数の設定方法(1)

システムを起動する前にシリアルコンソール接続しておくと、ARM-Yのブートローダーが起動します。kernelのロードを開始する前の2秒間に何かのキー(リターン等)を押下することにより、システムの起動を一旦停止してブートパラメタの変更やkernelの書き換え処理を行うことが可能です。



開発用SD(Debian-lenny)を使用するためには、システムの起動パラメータを変更して、ARM-Y搭載のNORフラッシュからの起動をSDカードからの起動に変更してください。

起動メディア	bootargsの設定
NORフラッシュ	`setenv bootargs 'mem=64M console=-ttyS2,115200n8 otg_mode=otg noinitrd root-=/dev/mtdblock5 rw rootwait rootfstype=jffs2'`
SDカード	`setenv bootargs 'mem=64M console=-ttyS2,115200n8 otg_mode=otg noinitrd root=-/dev/mmcblk0p1 rw rootwait rootfstype=ext3'`

※設定変更後、saveenv⏎により起動パラメータをフラッシュメモリに保存して恒久的に使用することができます。

※ARM-Y上のUSBはデフォルトでOTGモード(otg_mode=otg)に設定してあります。これ以外の設定としてホストモード(otg_mode=host)と、デバイスモード(otg_mode=device)が設定できます。

u-bootの環境変数の設定方法(2)

前述の方法に依らず、NORフラッシュメモリ内のu-boot起動パラメータ保存パーミッションのイメージを丸ごと書込むことで、NOR/SDカードのブートを切替えることができます。

(1) 添付のCD-ROMからfor_NOR.img,for_MMC.imgをUSBメモリにコピーしてARM-YのUSBミニコネクタに接続してください。USB接続用の変換ケーブルは事前に準備してください。

(2) ARM-Yに電源を投入してu-bootのコンソール画面から以下のコマンドを入力してイメージをNORフラッシュメモリに書込み、再起動してください。

(a) usb reset

(b) fatload usb 0 c0700000 for_NOR.img (またはfor_MMC.img)

(c) protect off 60100000 +$filesize

(d) erase 60080000 +$filesize

(e) cp.b c0700000 60080000 $filesize

カーネルのインストール方法

(1) 添付のCD-ROMからuImageをUSBメモリにコピーしてARM-YのUSBミニコネクタに接続してください。USB接続用の変換ケーブルは事前に準備してください。

(2) ARM-Yに電源を投入してu-bootのコンソール画面から以下のコマンドを入力してuImageをNORフラッシュメモリに書込み、再起動してください。

(a) usb reset
(b) fatload usb 0 c0700000 uImage
(c) protect off 60100000 +$filesize
(d) erase 60100000 +$filesize
(e) cp.b c0700000 60100000 $filesize

✓usb resetは一度実行してください。何度も実行する必要はありません。

ユーザーランドイメージのインストール方法

(1) 添付のCD-ROMからarm-y.jfsをUSBメモリにコピーしてARM-YのUSBミニコネクタに接続してください。USB接続用の変換ケーブルは事前に準備してください。

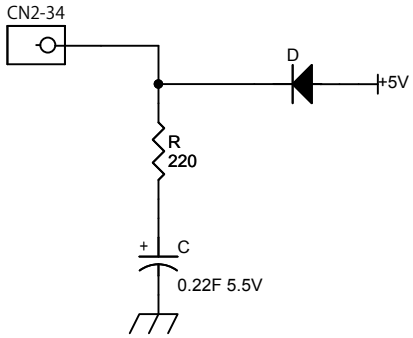
(2) ARM-Yに電源を投入してu-bootのコンソール画面から以下のコマンドを入力してuImageをNORフラッシュメモリに書込み、再起動してください。

(a) fatload usb 0 c0700000 arm-y.jfs
(b) protect off 60400000 +0x1C00000
(c) erase 60400000 +0x1C00000
(d) cp.b c0700000 60400000 $filesize

✓ユーザーランドイメージを書込むと、それまでのカスタマイズは全て無効になり、出荷時の状態に戻ります。必要に応じてバックアップをとっておいてください。

RTC(カレンダータイマ)のバックアップ方法

ARM-YはAM1808のRTCに接続する端子があります。ここに電源を接続するとにより電源OFFにしても時刻を保持することが可能です。以下に参考の回路図を掲載します。



この回路図では電気二重層コンデンサを使っていますが、容量とおよそのバックアップ時間は以下の式から計算してください。

```
t=C(V0-V1)/I

C=0.22F V0=5V V1=1.3V I=15uA とした場合

t=0.22×(5-1.3)/(15×10e-6)=54267秒=15.1時間
```

Ubuntuをゲートウェイとした場合のARM-Yのインターネット接続方法

ubuntuの設定

(1) /etc/sysctl.conf

net.ipv4.ip_forward=1

ubuntu再起動後　cat　/proc/sys/net/ipv4/ip_forwardで1になっていることを確認してください。

(2) iptablesによるNAT設定

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

iptables-save > /etc/iptables

vi /etc/network/if-pre-up.d/iptables

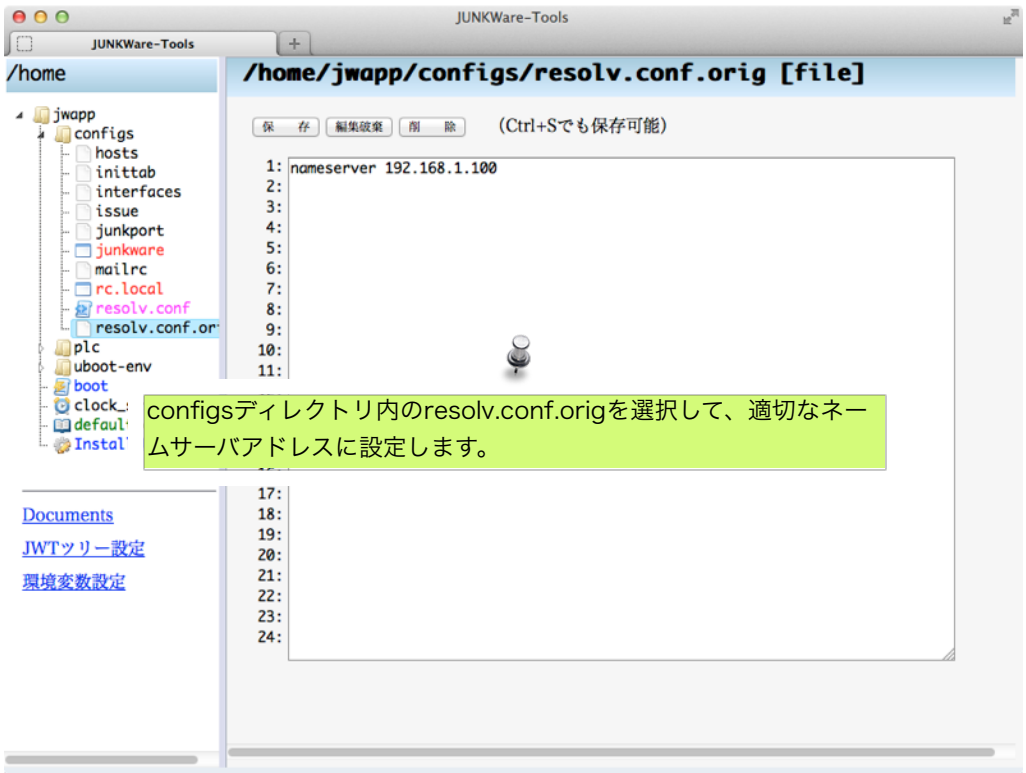
```
#!/bin/sh

  /sbin/iptables-restore < /etc/iptables
```

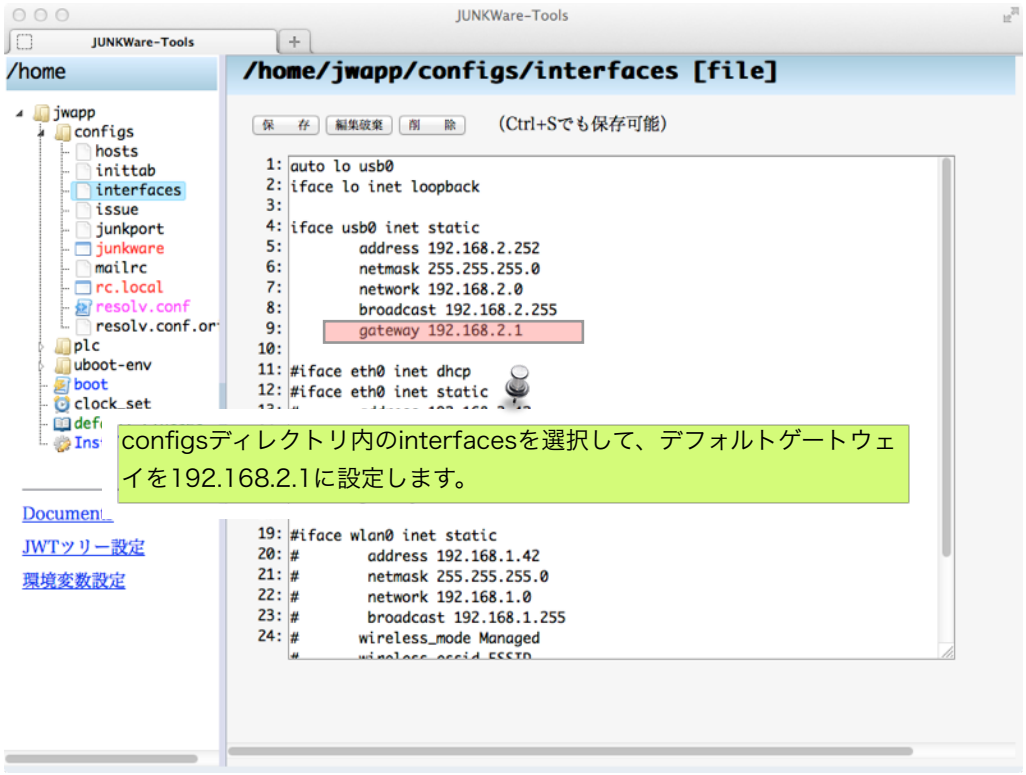
chmod 755 /etc/network/if-pre-up.d/iptables

ARM-Yの設定

(1)　/home/jwapp/configs/resolv.conf.origを使用環境のネームサーバのアドレスに設定してください。



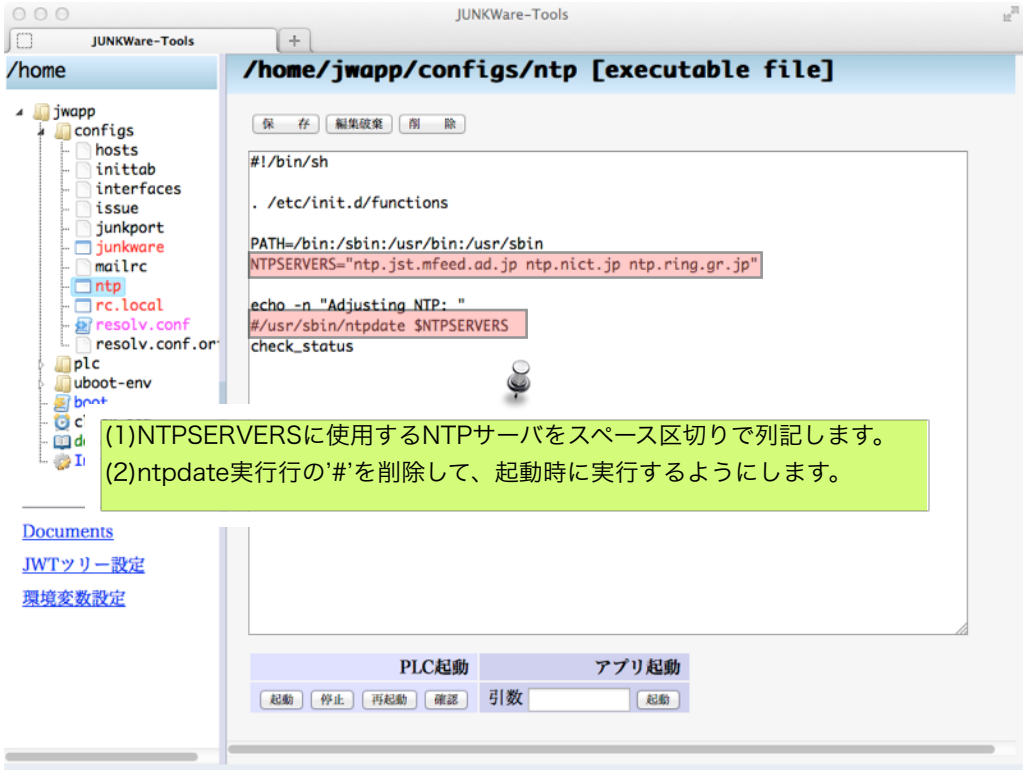
(2)デフォルトゲートウェイを適切に設定してください。標準では接続したパソコン側のインターネットアドレスとして192.168.2.1を指定してあります。



✓設定完了後は次ページ記載の方法でARM-Yを再起動してください。

NTPサーバによる時刻合わせ

ARM-Yがインターネットにアクセスできる場合、インターネット上のNTPサーバにアクセスして、自動的にカレンダータイマを設定することができます。/home/jwapp/configs/ntpを以下のように編集してください。



Cコンパイラによるソフトウェア開発方法

開発用SDで起動することにより、Debianベースで開発を行うことができます。SSHでログインしてvi等のエディタでソースを書いてコンパイル→実行→デバックを行い、でき上がったバイナリをNORフラッシュに書込む、という手順になりますが、ログインせずにブラウザのみでJWTを使って開発する方法をご紹介します。

JWT上でのソースコード作成

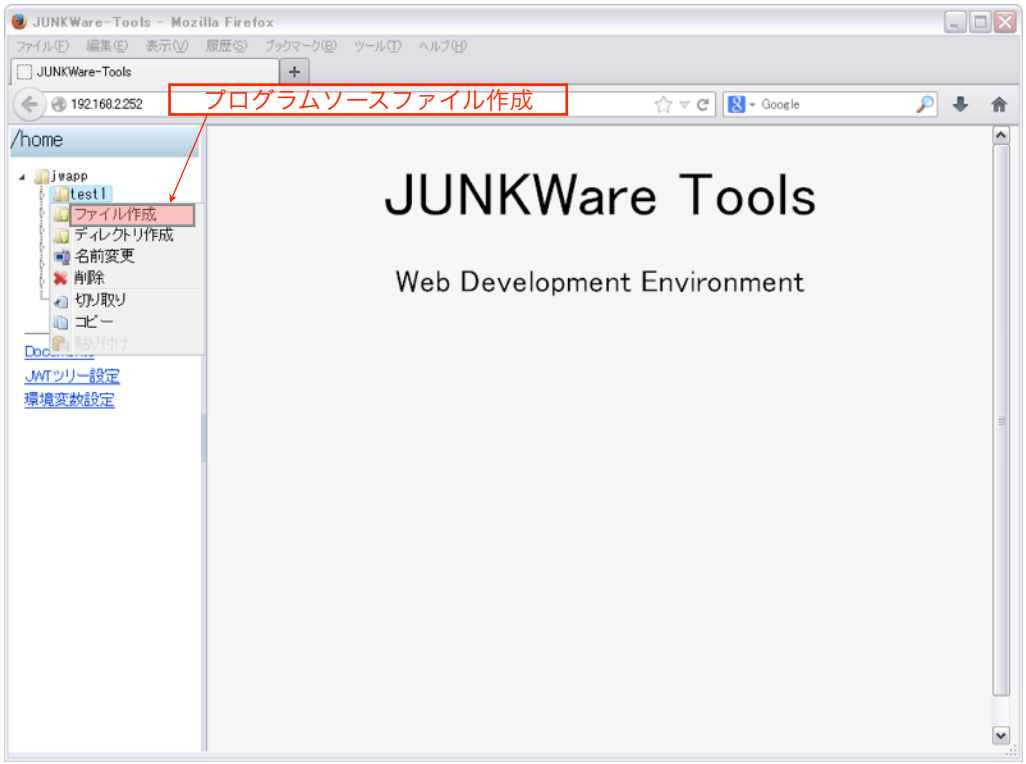
i)ディレクトリ作成

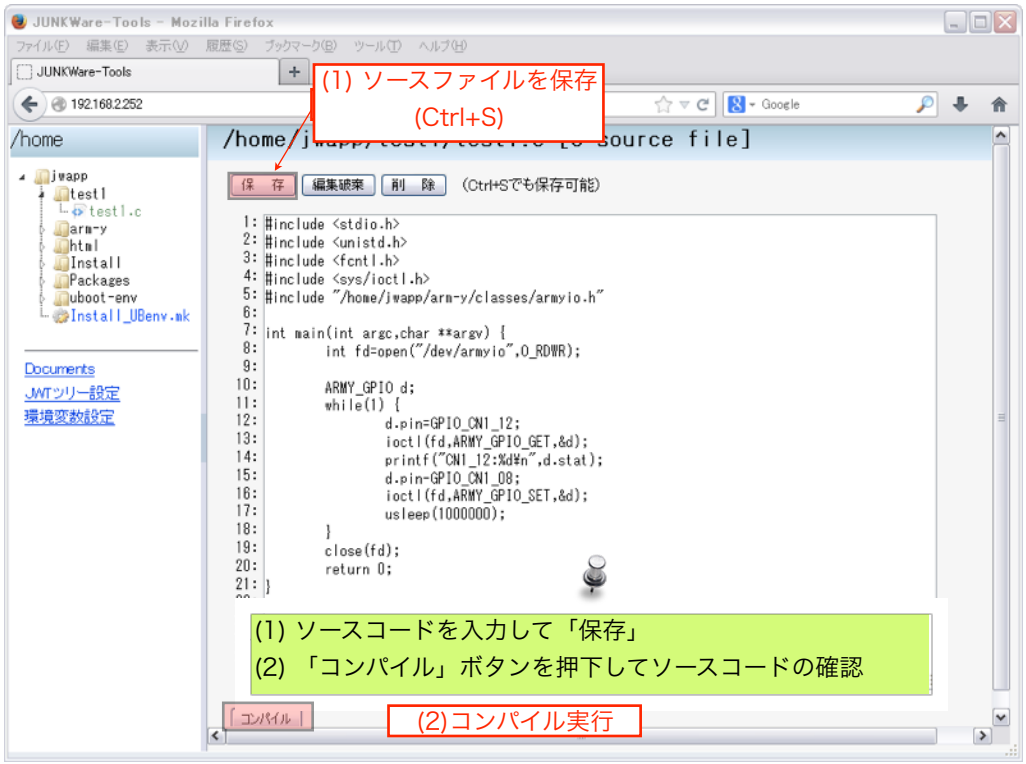
プログラム開発を行うディレクトリを/home/jwapp以下に作成します。ここではtest1というディレクトリを作成します。

上図の「ディレクトリ作成」を選択して新しいディレクトリにtest1という名前をつけてください。

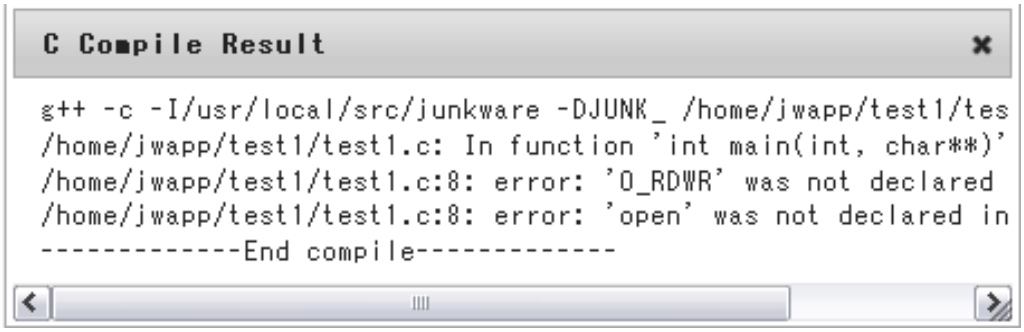
ii)ソースファイルの作成

ディレクトリの準備ができたら、そのディレクトリの下にソースファイルを作成してください。ここでは名前をtest1.cとしました。ソースファイルを作成した後、その中にプログラムを記述して保存します。





ソースコード作成後、右側ペイン下部にある「コンパイル」ボタンを押下すると、コンパイラを使ってコンパイルしてみることができます。これは文法的な確認などの目的で実施するもので、必須ではありません。



コンパイルの結果は上記のようにダイアログに表示されるので、この結果をみて、ソースコードの修正を行ってください。

GPIOドライバのアクセス方法

ここでtest1.cに入力したソースは以下のようになっています。ioctlを使ってARM-Yドライバにアクセスして、GPIOの制御を行っています。具体的には、このマニュアルで作成しているサンプル基板で定期的にスイッチの状態を読込み、LEDの点灯/消灯を切替えるものです。

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include "/home/jwapp/Packages/army-drv/armyio.h"

int main(int argc,char **argv) {
    int fd=open("/dev/armyio",O_RDWR);

    ARMY_GPIO d;
    while(1) {
          d.pin=GPIO_CN1_12;
          ioctl(fd,ARMY_GPIO_GET,&d);
          printf("CN1_12:%d\n",d.stat);
          d.pin=GPIO_CN1_08;
          ioctl(fd,ARMY_GPIO_SET,&d);
          usleep(100000);
    }

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

GPIOへのアクセスは以下のフローで行ってください。

i)ピンの設定

「PIN　MUXの設定」に従って、使用するピンのGPIO設定を行ってください。このサンプルの場合、/home/jwapp/configs/pinmux.confに以下の記述を行ってください。

```
cn1_8_gpio_L
cn1_12_gpio
```

/dev/armyioへのアクセスコードの記述方法

GPIOにアクセスするためにはioctlを使って、`/home/jwapp/Packages/army-drv/armyio.h`に定義してある ARMY_GPIO構造体を用います。

```
/* ioctl structure for GPIO access */
struct ARMY_GPIO {
    unsigned char pin;
    unsigned char stat;
};
```

ARMY_GPIOのpinには `d.pin=GPIO_CN1_12` のようにコネクタ番号とピン番号を用いた文字列を設定してください。この文字列は、実際にはarmyio.hにマクロ定義されている文字列となります。

入力ピンのステータス確認の場合は、以下のようにしてピンステータスをARMY-GPIOのstatに読込みます。

```
ARMY_GPIO d;
d.pin=GPIO_CN1_12;
ioctl(fd,ARMY_GPIO_GET,&d);
```

出力GPIOの場合は以下のコードでピンステータスをHにできます。

```
ARMY_GPIO d;
d.pin=GPIO_CN1_8;
d.stat=1;
ioctl(fd,ARMY_GPIO_SET,&d);
```

逆にd.stat=0とするとピンステータスをLに設定します。

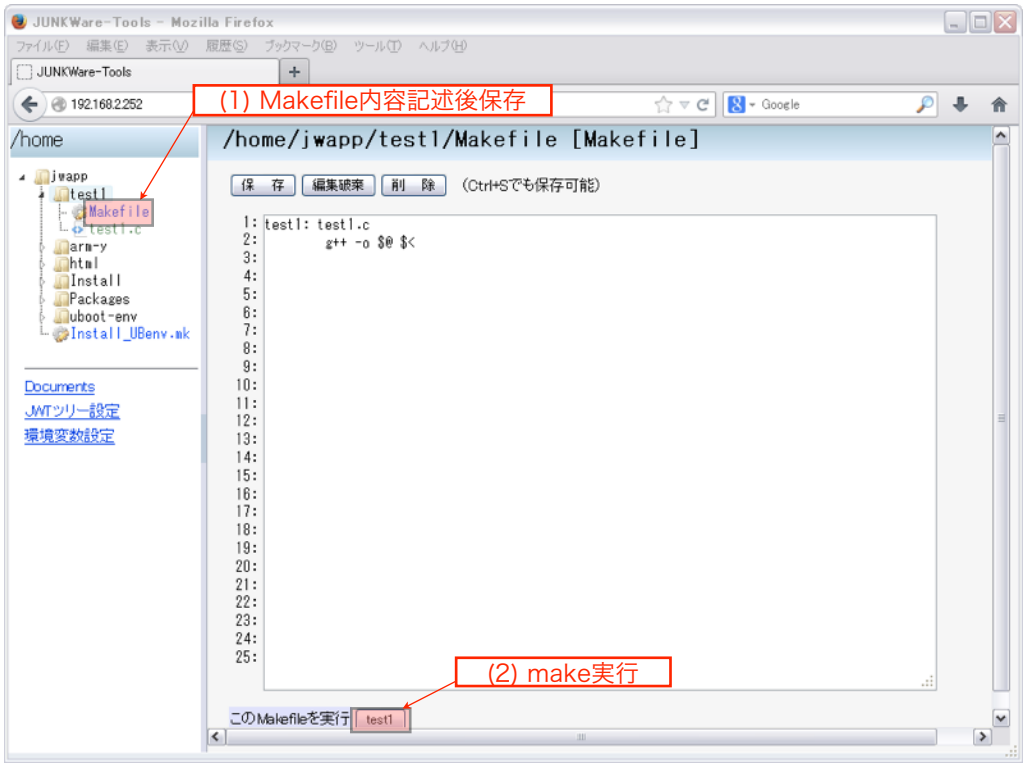
✓このソースプログラムはg++コンパイラを使うことを前提としています。gccを使う場合は、ARM_GPIOの宣言部を`struct ARMY_GPIO d;`のように書き換えてください。

makeファイルの作成とコンパイル

JWTでソースをコンパイルして実行バイナリを作成する場合、shellを使うことはできないので、JWTのmake呼出し機能を使います。test1ディレクトリ以下にMakefileを作り、以下のように記述します。

```
test1: test1.c
    g++ -o $@ $<
```

作成→保存すると下部に「test1」ボタンが生成されています。このボタンを押下することでターゲットバイナリtest1を生成できます。

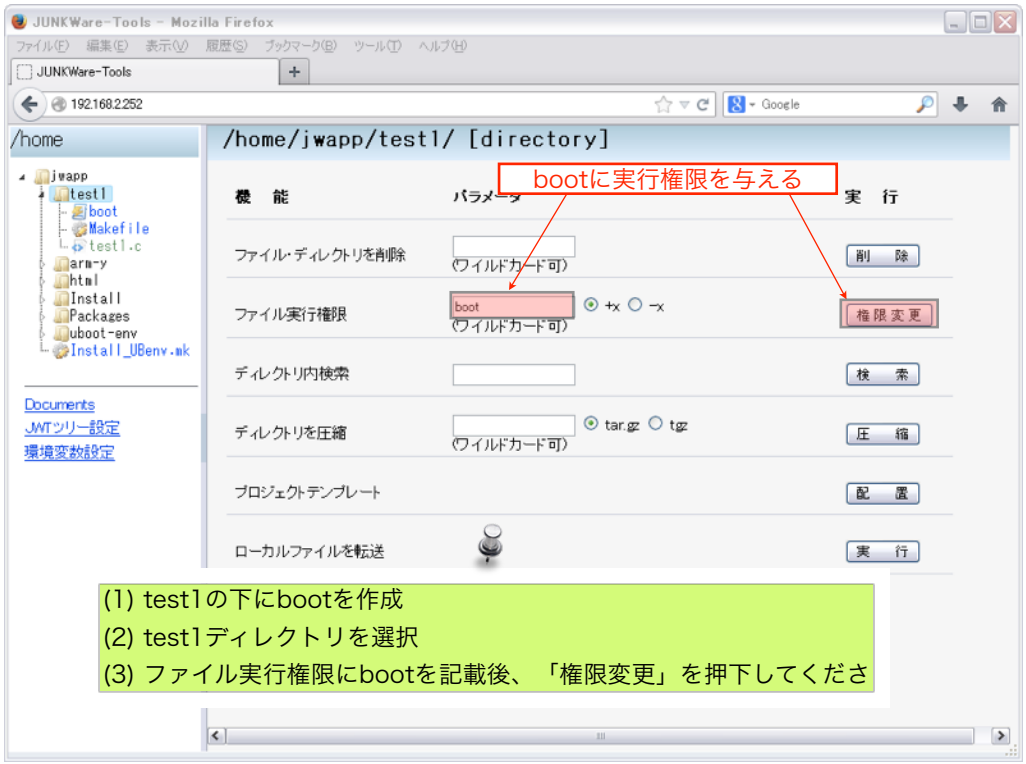


✓JWTのmake呼出し機能の詳細についてはJUNKWareのマニュアルを参照してください。

✓makeはファイルのタイムスタンプを用います。使用する前に、ARM-Yのカレンダタイマを設定するようにしてください。

実行

前ページまでで作成したバイナリを実行するために、以下の要領でbootファイルを作成してください。



bootには引数としてstart/stopを指定できます。これを利用するため以下にように記述してください。

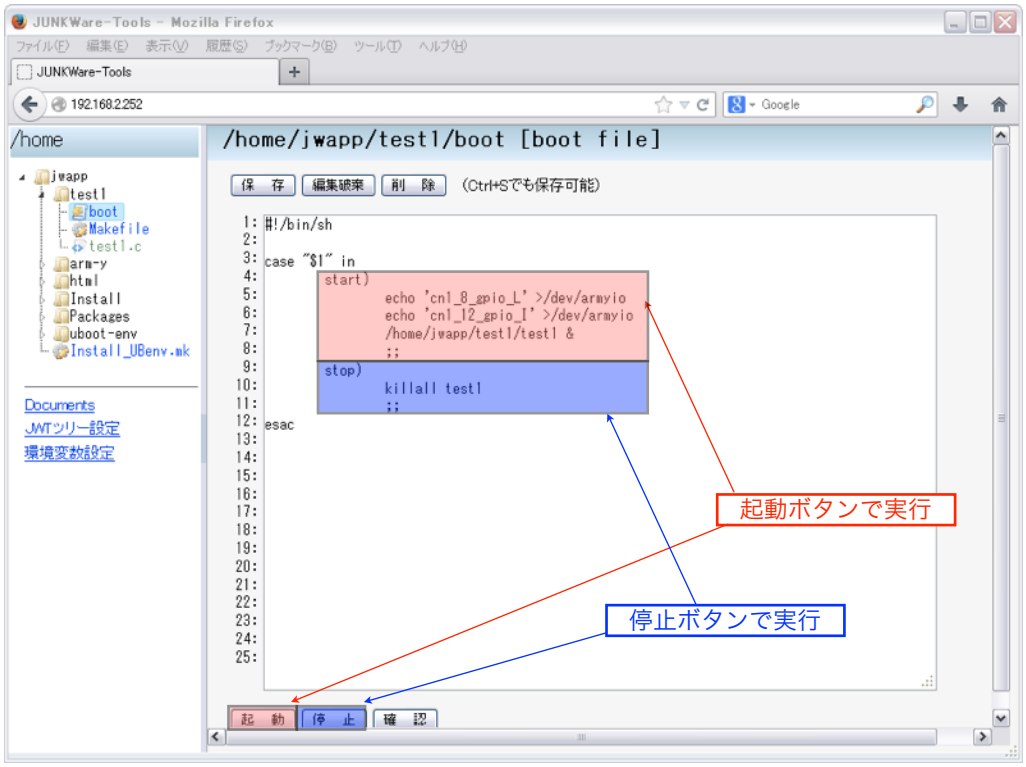
```
#!/bin/sh

case "$1" in
    start)
          echo 'cn1_8_gpio_L' >/dev/armyio
          echo 'cn1_12_gpio_I' >/dev/armyio
          /home/jwapp/test1/test1 &
          ;;
    stop)
          killall test1
          ;;
esac
```

簡単のため、ここではstart内でGPIOのピン設定も実施しました。test1は無限loopで戻ってこないため、バックグラウンド実行しています。stopではtest1プロセスをターミネートしています。

✓start時、このbootシェルスクリプトが終了しないとJWTでの操作ができなくなります。

bootスクリプトを保存すると、start, stopに記載した内容をstart, stopボタンで実行することができます。



bootボタン押下でtest1を実行してみてください。SW押下でLEDが消灯,SWリリースでLED点灯、と連動して動けばOKです。

マイクロSD使用中のSambaの利用について

開発用マイクロSDではSambaを使ってパソコンからSDにアクセスすることができます。アクセス時のユーザーID,パスワードは

ID	パスワード
jwapp	jwapp

を使ってください。Sambaの設定は/etc/samba/smb.confにあります。必要に応じて変更してください。

フラッシュメモリのマウント/アンマウント方法

SDからブートしている場合も

```
sudo mount -t jffs2 /dev/mtdblock5 /mnt/cf/
```

で/mnt/cfを編集することで、NORフラッシュのユーザーランドをファイル単位で変更することができます。

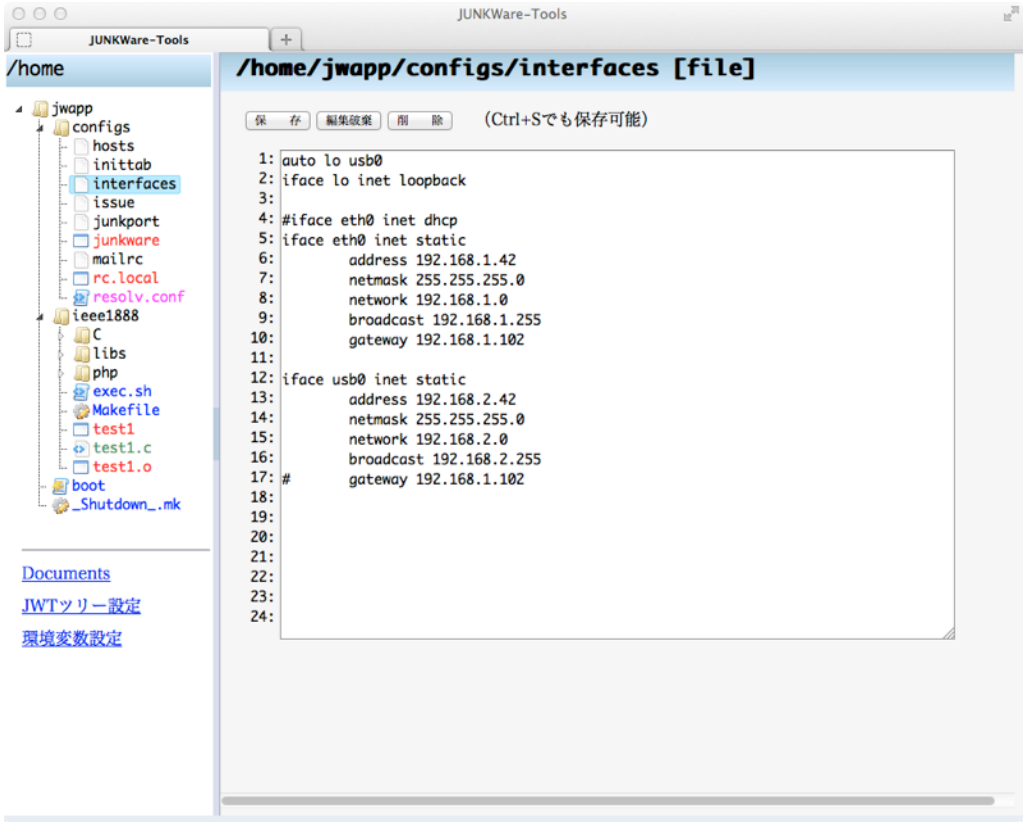
変更終了後は

```
sudo umount /mnt/cf
```

でアンマウントしてください。

ネットワーク設定

JWTでconfigs/interfacesを開いてください。このファイルは/etc/network/interfacesへのシンボリックリンクになっているので、このファイルを編集して再起動することにより、ネットワーク設定を変更することが可能です。



ブートローダー(u-boot)のインストール方法

(1) ARM-Yのシリアルコンソールと Windowsパソコンをシリアルケーブルで直結してください。ARM-Yに電源を投入する前にARM-Y上のディプスイッチを以下のように設定します(UART2ブートモード)。

```
DIP SW[1234]=[ON ON OFF ON]
```

(2)ダウンロードサイトから「ARM-Y書込み」をWindowsにコピーして開いてください。

(3)「ARM-Y書込み」フォルダ内のwrite-uboot.batをテキストエディタで開いてCOMポートを適切に修正して保存します。

(4)ダブルクリックでwrite-uboot.batを起動します。その後、ARM-Yの電源を入れてください。DOS窓が開いて書込み処理が始まります。

(5)書込みが完了してDOS窓が閉じたら、DIPスイッチを以下のようにNORブート用に戻して、ARM-Yの電源を再投入してください。

```
DIP SW[1234]=[ON OFF ON ON]
```

✓ダウンロードサイトについては 「各種リソースの提供について」を参照してください。

AM1808のWatchDogについて

AM1808にはWatchDog機能が搭載されています。ARM-Y搭載のJUNKWareではこの機能を使うクラスを用意して組み込んでいます。/home/jwapp/plc/objs/watchdogを開いてみてください。

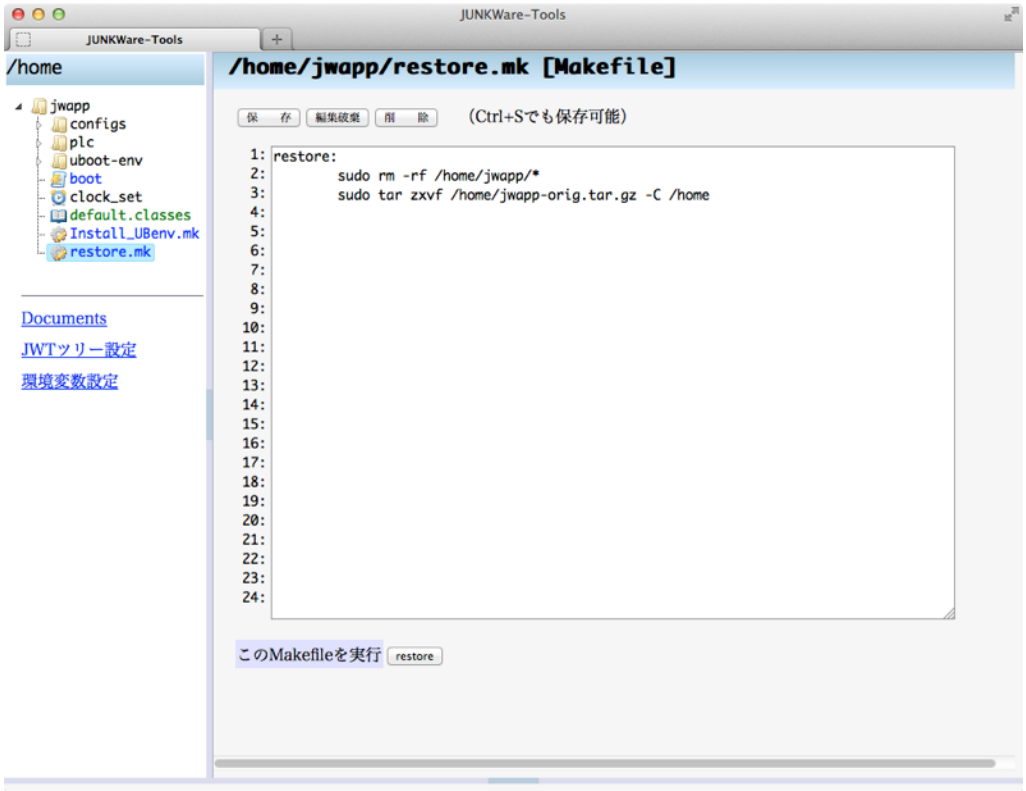
WATCHDOGというクラスから生成されたwatchdogオブジェクトの定義があります。起動時ON, input=45(秒)に設定してあるので、起動時に動作を開始して、inputの設定時間-5秒単位で/dev/watchdogにアクセスしてタイマをリセットしています。

もし、このJUNKWareプロセスが落ちてしまった場合にはWatchDogデバイスが更新されなくなるため、カウントアップと同時に(45秒経過後)、ARM-Yは自動的に再起動します。

JUNKWareプロセスを終了してもWatchDog機能を働かさないためには、プロセス終了前にwatchdogオブジェクトのステータスをOFFにする必要があります。/home/jwapp/plc/bootには、その処理が記載してあるので参考にしてください。

/home/jwappの復旧方法

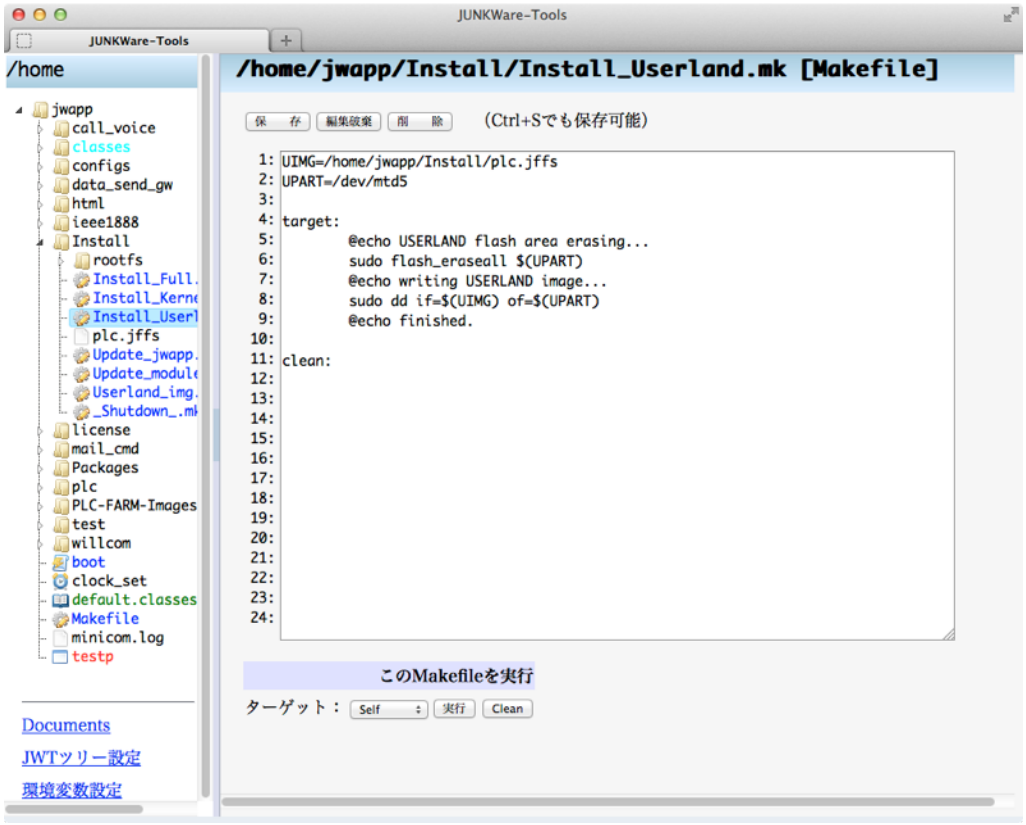
/home/jwapp/restore.mkには、/home/jwapp以下を初期状態に戻す機能を記載してあります。具体的には、/home/jwapp-orig.tar.gzにあるバックアップを上書きする機能です。



✓この機能ではconfigs/以下の設定ファイルは元に戻すことができません。これはconfigs/以下のファイルがシンボリックリンクであるためです。

フラッシュメモリの更新方法

開発用マイクロSDのInstall/Install_Userland.mkを使ってARM-YのNORフラッシュを出荷時の状態に戻すことができます。



JWT上からInstall_Userland.mkを選択して「実行」ボタンをクリックするとNORフラッシュ上のユーザーランドを更新します。更新手続きについても、このmakefileを編集することにより変更が可能です。コンソール上からmakeコマンドを直接起動しても構いません。

IOピンのDC特性

IOボードを設計する際には、下記AM1808の基準を使用してください。

Electrical Characteristics Over Recommended Ranges of Supply Voltage and Operating Junction Temperature (Unless Otherwise Noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage (dual-voltage LVCMOS IOs at 3.3V)(1)	DVDD= 3.15V, I_{OH} = -4 mA	2.4			V
		DVDD= 3.15V, I_{OH} = -100 µA	2.95			V
	High-level output voltage (dual-voltage LVCMOS IOs at 1.8V)(1)	DVDD= 1.71V, I_{OH} = -2 mA	DVDD-0.45			V
V_{OL}	Low-level output voltage (dual-voltage LVCMOS I/Os at 3.3V)	DVDD= 3.15V, I_{OL} = 4mA			0.4	V
		DVDD= 3.15V, I_{OL} = 100 µA			0.2	V
	Low-level output voltage (dual-voltage LVCMOS I/Os at 1.8V)	DVDD= 1.71V, I_{OL} = 2mA			0.45	V
I_I (2)	Input current(1) (dual-voltage LVCMOS I/Os)	V_I = VSS to DVDD without opposing internal resistor			±9	µA
		V_I = VSS to DVDD with opposing internal pullup resistor (3)	70		310	µA
		V_I = VSS to DVDD with opposing internal pulldown resistor (3)	-75		-270	µA
	Input current (DDR2/mDDR I/Os)	V_I = VSS to DVDD with opposing internal pulldown resistor (3)	-77		-286	µA
I_{OH}	High-level output current(1) (dual-voltage LVCMOS I/Os)				-6	mA
I_{OL}	Low-level output current(1) (dual-voltage LVCMOS I/Os)				6	mA
Capacitance	Input capacitance (dual-voltage LVCMOS)			3		pF
	Output capacitance (dual-voltage LVCMOS)			3		pF

(1) These IO specifications apply to the dual-voltage IOs only and do not apply to DDR2/mDDR or SATA interfaces. DDR2/mDDR IOs are 1.8V IOs and adhere to the JESD79-2A standard. USB0 I/Os adhere to the USB2.0 standard. USB1 I/Os adhere to the USB1.1 standard. SATA I/Os adhere to the SATA-I and SATA-II standards.
(2) I_I applies to input-only pins and bi-directional pins. For input-only pins, I_I indicates the input leakage current. For bi-directional pins, I_I indicates the input leakage current and off-state (Hi-Z) output leakage current.
(3) Applies only to pins with an internal pullup (IPU) or pulldown (IPD) resistor. The pull-up and pull-down strengths shown represent the minimum and maximum strength across process variation.

各種リソースの提供について

このマニュアルに記載したARM-Yのカスタマイズに使用するリソースは(株)ＹＯＯＤＳのホームページから提供されています。

http://www.yoods.co.jp/docs/index.htmlからダウンロードしてご使用下さい。

なお、開発用SDはdebian/Lennyをベースにしています。Debianに関する設定やカスタマイズ等については、Webサイト上の情報をご利用下さい。