<b>4 14 0 00 株 弐 0 1 1 デ </b> ド		tel. 083-976-0022
AMI 1606 拾戦CPU小一ト	株式会社YOODS 〒754-0011	fax. 083-976-0023
<sup>r</sup> ARM-Y」	山口県山口市小郡御幸町4番地9	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

<b>▲▲▲▲●○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○</b>		tel. 083-976-0022	
AMI808拾載CPU小一ト	株式会社YOODS 〒754-0011	fax. 083-976-0023	
<sup>r</sup> ARM-Y」	山口県山口市小郡御幸町4番地9	<pre>info_yoods@yoods.co.jp</pre>	

# 目次

ARM CPUボードについて	1
ARM-Yは	1
ARM-Yの概要	2
電源の接続	3
ヘッダピンによる電源供給	3
<b>USB</b> による電源供給	4
OTGによるEthernet接続方法	5
Windows XPへの接続方法	6
Windows 7への接続方法	8
Linux(Ubuntu)への接続方法	10
ARM-Yの起動と接続確認	11
<i>ping</i> による接続確認	11
JWTによるパソコンとARM-Yの接続	12
JWTによる接続	12
ポップアップウインドウブロックの解除	13
SSHによる接続	14
時刻同期について	15
手動時刻合わせ	15
PIN MUXの設定	16
CN1のPIN配列	17

∧м1000広戦ひロリポード		tel. 083-976-0022
	│ 株式会社YOODS   〒754-0011	fax. 083-976-0023
'ARM-Y」	山口県山口市小郡御幸町4番地9	<pre>info_yoods@yoods.co.jp</pre>
CN2のPIN配列		18
PIN MUXの設定		15
PIN MUXを切替える方法		15
PIN MUXをどこで設定するか		20
サンプル回路の作成とJUNKWare	を使ったプログラム作成	21
練習用回路図		2
JUNKWareによる動作確認		23
出力ピンの設定		23
入力ピンの設定		24
再起動による設定の反映		25
試作ボードの動作確認(dispparによ	ର)	20
応用〜JUNKWareによるLEDとス	イッチの連動	30
動作シーケンスの記述		30
動作確認		30
JUNKWareによるARM-Y上のLED常	间御方法	32
付録		1
ARM-Yの再起動方法		
開発キットについて		
開発キットの内容		-
開発環境(マイクロSD)の使用方法	<del>.</del>	2
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	2
開発用マイクロ <i>SD</i> の内容について		:
シリアルコンソールの使い方		ł

		tel. 083-976-0022
AM1808拾載CPU小一ト	株式会社YOODS 〒754-0011	fax. 083-976-0023
「ARM-Y」	山口県山口市小郡御幸町4番地9	<u>info_yoods@yoods.co.jp</u>
CN5(シリアルコンソールポート)	の配線	5
シリアルコンソールの通信パラン	ペータ	6
u-bootの環境変数の設定方法(1)		7
u-bootの環境変数の設定方法(2)		8
カーネルのインストール方法		ç
ユーザーランドイメージのインス	ストール方法	ç
RTC(カレンダータイマ)のバックア	ップ方法	10
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	訪法 11
ubuntuの設定		11
ARM-Yの設定		11
NTPサーバによる時刻合わせ		13
Cコンパイラによるソフトウェア開	発方法	14
JWT上でのソースコード作成		14
ディレクトリ作成		14
ソースファイルの作成		15
GPIOドライバのアクセス方法		16
ピンの設定		17
makeファイルの作成とコンパイ	N	18
実行		19
マイクロSD使用中のSambaの利用	について	21
フラッシュメモリのマウント/アンマ	マウント方法	22
ネットワーク設定		23
ブートローダー(u-boot)のインスト	ール方法	24

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
AM1808のWatchDogについて /home/jwappの復旧方法		25

*IOピンのDC*特性

フラッシュメモリの更新方法

各種リソースの提供について

29

27

28

### 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

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	

### ARM-Yの機能概要を下表に示します。



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。



新しいハードウェアの検出ウィザード
検索とインストールのオブションを選んでください。
<ul> <li>○ 次の場所で最適のドライバを検索する(S) 下のチェック ボックスを使って、リムーバブル メディアやローカル パスから検索できます。検索された最適のドラ イバがインストールされます。</li> <li>マリムーバブル メディア (フロッピー、CD-ROM など) を検索(M)</li> <li>マ )次の場所を含める(Q).</li> <li>○ XDのguments and Settions X arX 72 7 2 7 2 7</li> </ul>
○社会をはればいるは、日本はないで、インストールするドライバを選択する(Q) 一覧からドライバを選択するには、このオブションを選びます。選択されたドライバは、ハードウェアに最適のもの とば取りません。
〈戻る(2) 次へ(2) キャンセル



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。

オブション								X
一般	タブ	(1) しょうしょうしょうしょうしょうしょうしょうしょうしょうしょうしょうしょうしょうし	שפעםל גפעםל	<b>つつ</b> プライバシー	25-21)ティ	Sync Sync	<b>☆</b> 詳細	
<ul> <li>✓ ボッブ</li> <li>✓ 画像</li> <li>✓ Java</li> </ul>	アップウィン を自動的(Ci Script を有	<sup>ドウをブロックす</sup> 読み込むの 劾にする( <u>」</u> )	'ଟ( <u>B</u> )				許可サ- 例外サ- 詳細設	<u>(┣@)</u> (┣::::::::::::::::::::::::::::::::::::
ー フォントと配 既定のフォ	記色 ォント( <u>D</u> ): [	MS Pゴシック			<b>र</b> मिर:	ズ(S): 16	<ul> <li>詳細設</li> <li>配色設</li> </ul>	定( <u>A</u> ) 定( <u>C</u> )
言語 Web ペー	ジの表示に	使用する言語	の優先順位を	設定できます。			言語談	定②
					ОК	**	1211 N	ルプ(円)

### 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。

 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。



192.16	8.2.252	:22 - Te	ra Term VT					
ファイル(E)	編集(E)		コントロール(Q)	ウルドウ型	漢字コード(ど)	ヘルプ(ゼ)		
								^
[jwapp@s [jwapp@s [jwapp@s	irm-y ( irm-y ( irm-y (	ttyp0) ttyp0) ttyp0)	~]\$ ~]\$ ~]\$					
								~

## 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。

JUNKWare-Tools	+		
€ € 192.168.2.252		☆ マ C 🚼 - Google	<i>P</i> 1
home	/home/jwapp/clock_set [cl	ock_set file]	
Giver provide a state of the state of	日時設定: 2013/7/3 20:36:5 夏新		
	Ş		
(1)JWT (2)右側 (3)その Yの時刻	の左側ファイルツリーから「d ペインに操作中のパソコンの5 まま「更新」ボタンをクリッ?  を(2)の時刻に設定します。	clock_set」を選択してください。 見在の時刻が表示されます。 りしてください。ARM-	

✓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P	N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
4	UART2_TXD				
5	UART2_CTS				「コンシールとしてドランシー」
6	UART2_RTS				
7	LCD_D0	GPIO7_8		PD	
8	LCD_D1	GPIO7_9		PD	
9	LCD_D2	GPI07_10		PD	
10	LCD_D3	GPI07_11		PD	
11	LCD_D4	GPI07_12		PD	
12	LCD_D5	GPI07_13		PD	
13	LCD_D6	GPI07_14		PD	
14	LCD_D7	GPI07_15		PD	
15	LCD_D8	GPIO7_0		PD	ブートモード設定に使用する
16	LCD_D9	GPIO7_1		ブートモ	ため、電源投入時は使用でき
17	LCD_D10	GP107_2		— หี รพ	ません。また、使用する場合
18	LCD_D11	GPIO7_3		の設定	は、フートモードDIP SWと
19	LCD_D12	GPIO7_4		を化し	成日しないようにパックティー で分離する必要があります。
20	LCD_D13	GPIO7_5		ます。	
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	GP108_9		PU	
26	LCD_MCLK	GPI08_10		PU	
27	LCD_PCLK	GPI08_11		PU	
28	AXR0	CLKS0	GPIO8_7	PU	
29	AXR1	DX0	GPIO1_9	PU	
30	AXR2	DR0	GPIO1_10	PU	
31	AXR3	FSX0	GPI01_11	PU	
32	AXR5	CLKX0	GPI01_13	PU	
33	AHCLKX	UART1_CTS	GPIO0_10	PU	UART1
34	AHCLKR	UART1_RTS	GPI00_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	GPI06_15		PU	
10	CLKOUT	GPI06_14		PU	
11	USB1_DP				
12	USB1_DM				
13	SPI1_CLK	GPI02_13		PU	
14	SPI1_ENA	GPI02_12		PU	
15	SPI1_SOMI	GPI02_11		PU	
16	SPI1_SIMO	GPI02_10		PU	
17	SPI1_SCS7	12C0_SCL	GPIO1_5	PU	12C0
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	GPI02_15		PU	
22	SPI1_SCS0	GPI02_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	GP108_5		PU	
32	+VIO				DC3.3V 拡張ボードのIO用電
33	GND				源, 最大電流500mA
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。



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。

✓ <u>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</u>
 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。

000	JUNKWare-Tools	1277
JUNKWare-Tools	+	
/home	<pre>/home/jwapp/plc/objs/ [objs directory]</pre>	
<pre>&gt; jwapp configs plc docs beat -&gt; fc_reset -&gt; reboot -&gt; scan boot -&gt; comment.rc -&gt; disp.rc -&gt; plc -&gt; plc -&gt; plc -&gt; plc -&gt; plc -&gt; sequence.rc -&gt; sequence.rc -&gt; sequence.rc -&gt; sequence.rc</pre>	<ul> <li>作 成</li> <li>夕グ作成機能</li> <li>・ 基本情報</li> <li>ダグ名 (n1.8.led)</li> <li>ダブ名 (n1.8.led)</li> <li>グラス (AMCPIO):</li> <li>ボビー:</li> <li>範囲</li> <li>正AD)</li> <li>・ 初期値</li> <li>・ 共通</li> </ul>	
<pre></pre>	Status Index パラメータ コオ	・クター
□ derault.Classes ◎ Install_UBenv.mk □ <u>Documents</u> <u>JWTッリー設</u> 現境変数設定	ご。(() (	

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。

UNKWare-Tools - Mozi	lla Firefox		
JUNKWare-Tools	履歴ら フックマーク(B) サール(D) ヘルフ(D) +		
€ @ 192.168.2.252		☆ マ C 🔀 - Go	ogle 🔎 🦊 🏫
/home	/home/jwapp/plc/objs/	[objs directory]	
<pre>&gt; jvapp &gt; configs &gt; plc &gt; docs &gt; docs &gt; docs &gt; cnl_8_led - fc_reset - led_R - led_R - led_Y - reboot - scan _ watchdog &gt; scripts - boot - connent.rc - connent.rc - disp.rc - env.rc</pre>	作 成 タグ作成機能 ・基本情報 ダグ名 cn1,12_sw クラス AMGP10 × ゲート scan0 × 範囲 LEAD × ・ ・ ・ 初期値 ・ 共通		
- plc - plct.rc - sequence.rc - svn-connit.tnp	Status Index	パラメータ	コネクター 〇 ① ③
<ul> <li>Auboot-env</li> <li>Stoot</li> <li>Octock_set</li> <li>Odefault.classes</li> <li>Mathematical Procession</li> </ul>			
<u>Documents</u> <u>JMTツリー設定</u> 環境変数設定	プ名,クラス名,ゲート,パラメ- 戈ボタンを押下してください,	ータを設定してください。 。	

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。

😻 JUNKWare-Tools - Mozi	lla Firefox	
ファイル(E) 編集(E) 表示(V)	履歴(S) ブックマーク(B) ツール(T) ヘルブ(H)	
JUNKWare-Tools	+	
/home	/home/jwapp/Install_UBenv.mk [Makefile]	
▲ ■ivapp ◆ ■configs ● ■pic ● ■pic ● ■boot ● ○ clock_set ● ○ clock_set ■	保存編集破棄 削除 (Ctrl+Sでも保存可能) 1: ENV1=/home/jwapp/uboot-env/for_NOR.img 2: ENV3=/home/jwapp/uboot-env/for_MIC.img 3: ENV3=/home/jwapp/uboot-env/for_NFS.img 4: UPART=/dev/mtd2 5: 6: NOR: 7: @echo U-Boot Environment flash area erasing 8: sudo flash_eraseall \$(UPART) 9: @echo writing environment for NOR Flash Memory 10: sudo dd if=\$(ENV1) of=\$(UPART) 11: @echo finished. 12: 13: MMC: 14: @echo U-Boot Environment flash area erasing 15: sudo flash_eraseall \$(UPART) 16: @echo writing environment for MMC File System 17: sudo dd if=\$(ENV2) of=\$(UPART) 18: @echo finished. 19: 20: reboot: 21: sudo reboot 22: 23: shutdown:	
*reboot示	ダン押下により、冉起動してくたさい。	
	この Makefileを実行 NOR MMC reboot shutdown	
	<b>K</b>	>

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。

JUNKWare-Tools	+			
<ul> <li>♦ □ 192168.2252</li> <li>∕home</li> </ul>				
/home			☆ マ C 🚼 - Google	<i>P</i> <b></b>
	/home/jwapp/plo	c/objs/cn1_8_le	ed [tag file]	
<pre>     jwapp     Configs     plc     configs     docs     docs     Configs     configs</pre>	(保存) 削除 タグ編集4 モニタ ・基本情報 タグ名 co1 8 jed クラス AMGPD ▼ ゲート ▼ 範囲 LEAD ・初期値 ・共通	<sup>〒弁入ト表示</sup> <u>モニター</u> ボタンを押下して <sup>□</sup>	♀ モニタウインドウをŀ	開きます。
esquence.rc ⇒ boot ⇒ clock_set ⇒ install_UBenv.r <u>Documents</u>		Index Set pin=1,8 Set pin=1,8		
		.111		le le
Tag Input ( cn1_8_led ステータス表:	Dutput 示/設定 リアルタ	ジ イムモニタウイント	<sup>۲</sup> ウ	

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⇒⊲<sup>2</sup>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。

Tag	Input	Output				
cn1_8_lec	CANCEL ON OFF					
				) Į		
	(1) ■(ステ (2) ON/OF	ータス)にフ 下ステータン	ォーカスし <sup>-</sup> スを選択して	ক ⊄` ক		

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。

🥹 JUNKWare-Tools - Mozilla F	irefox			
ファイル(E) 編集(E) 表示(V) 履歴 UNKWare-Tools	⑤ ブックマーク(B) ツール(D) ヘルプ(E) +	0		
€ € 192.168.2.252		☆ マ C 🛛 🚼 - Google	P	+ 1
/home	/home/jwapp/ [d	irectory]		
▲ 🛄 jwapp 🝓 ファイル作成 崎 ディレクトリ作成	機能	パラメータ	実 行	
► BURIC - Octock_set	ファイル・ディレクトリを削除	<ul><li>(ワイルドカード可)</li></ul>	削除	
Install_UBenv.mk	ファイル実行権限	(קרוולד) ● +x ○ -x	権限変更	)
<u>Documents</u> JWTソリー設定 環境変数設定	ディレクトリ内検索		検索	
(1) /home/jwap (2) 新しく作った	opを右クリックして、 :ファイルのファイル	「ファイル作成」を選択してくた 名を「disp1.rc」としてください	ださい。 。	
	ローカルファイルを転送		実 行	
192.168.2.252/#	< ]	Ш		

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。

😻 JUNKWare-Tools - Mozilla Firefox		_ [	
ファイル(E) 編集(E) 表示(V) 履歴(S) ブックマーク(B) 、	ν−ル① ヘルプ(E)		
JUNKWare-Tools +			
♦ ③ 192.168.2252	☆ マ C 🛛 🚼 ◄ Google 🖉	•	A
/home /home/jw	app/disp1.rc [disp resource file]		^
imapp         iff iff iff iff           imapp         iff iff iff           imapp         iff iff iff           imapp         imapp           imapp         imapp<	<sup>実</sup> 破楽 <u>削 除 モニター</u> (Ctrl+Sでも保存可能) ed sM		
<ul><li>(1) disp1.rcにcn1_8_led,c</li><li>(2) 「モニター」ボタンを押</li></ul>	n1_12_swを列記して保存します。 下するとdispparのモニタウインドウが開きます。		
10: 17: 18: 19: 20: 21: 22: 23: 24: 25:	Ш		>

Ŷ



## 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。

🕘 JUNKWare-Tools - Mozi	lla Firefox			
ファイル(E) 編集(E) 表示(V)	履歴(S) ブックマーク(B) ツール(T) ヘルブ(H)			
JUNKWare-Tools	+			_
€ € 192.168.2.252		☆ マ C Soogle	₽ ♣	⋒
/home	/home/jwapp/plc/sequence.rc	[sequence file]		^
	保存 認定反映 ラダー表示 (Otri+Sでも		.::	Ξ
(1) /none/jwa	pp/pic/sequence.icを選択して、 たち効にするために「恐空反映、ず	エ記のように記処,休住しより。 ないた畑下してください		
(Z) 9-9 7 X		フノを押下してくたさい。		
	<	10		~

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。

SONK Ware-Tools - Mozilia	Firefox				
ファイル(E) 編集(E) 表示(V) 履)	歴(5) ブックマーク(8) ツール(1) ヘルプ(H)				
JUNKWare-Tools	+				_
♦ ♦ ④ 192.1682.252		☆ マ C Google	P	+	A
/home	/home/jwapp/plc/sequ	ence.rc [sequence file]			^
<pre>iwapp configs plc docs docs docs docs docs docs docs doc</pre>	(& 存) 編集破乗) 削除 1: scan.0 cnl_12_sw(a) cnl_8_le 2: 3: 4: 5: 6: 7: ラダー表示ボタ	<u>&gt;タ-表示</u> (Ctrl+Sでも保存可能) d(coil) ンを押下してください。			HI.
Lissedueroy Doot-env ジ clock_set ・ 回 default.classes ・ 回 displ.rc ジ Install_UBenv.mk Documents JWTツリー設定 環境変数設定	12: 13: 14: 15: scan.0 onl J2,3**		cn1.8.Jed		

Ţ

JUNKWare-Tools - Mozilla F ファイル(F) 編集(F) 表示(A) 開閉	ີirefox ເດິງ 1∞ກອ⇒ກ(R) ທຸ⊨ແ/T) ∧ ແ//(ຟ)	-0
UNKWare-Tools	+	
€ € 192.168.2.252	j_ ⊽ ⊄	P + 1
/home	/home/jwapp/plc/sequence.rc [tag file]	
<pre>image image i</pre>		cnl_@_led #
(1) スイッチ押 (2) (1)の動きか	下でLEDが点灯することを確認してくたさい。 「このラダーモニタ画面でも確認できます。	
□ disp1.rc ⊘ Install_UBenv.mk <u>Documents</u> <u>JWTツリー設定</u> 環境変数設定	▲	

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。

000	JUNKWare-Tools	1221
JUNKWare-Tools	+	
/home	<pre>/home/jwapp/Install_UBenv.mk [Makefile]</pre>	
<pre>jwapp configs initiab interfaces junkport junkware mailrc rc.local resolv.conf resolv.conf plc uboot-env b c</pre>	<ul> <li>※ 件 編集磁像 画 除 (Ctrl+Sでも保存可能)</li> <li>1: ENV1=/home/jwapp/uboot-env/for_NOR.img</li> <li>2: ENV2=/home/jwapp/uboot-env/for_NFS.img</li> <li>3: ENV3=/home/jwapp/uboot-env/for_MMC.img</li> <li>4: UPART=/dev/mtd2</li> <li>5: NOR:</li> <li>7: @echo U-Boot Environment flash area erasing</li> <li>8: sudo flash_eraseall \$(UPART)</li> <li>9: @echo writing environment for NOR Flash Memory</li> <li>10: sudo dd if=\$(ENV1) f=\$(UPART)</li> <li>11: @echo finished.</li> </ul>	
□ di Unitaria □ Documents <u>JWTツリー設定</u> 環境変数設定	15:       sudo flash_eraseall \$(UPART)         16:       @echo writing environment for NFS Root File System         17:       sudo di f=\$(ENV2) of=\$(UPART)         18:       @echo finished.         19:       20:         20:       MMC:         21:       @echo U-Boot Environment flash area erasing         22:       sudo flash_eraseall \$(UPART)         23:       @echo writing environment for MMC File System         24:       sudo di f=\$(ENV3) of=\$(UPART)         @echo finished          24:       sudo di f=\$(ENV3) of=\$(UPART)         @echo #inished          24:       Sudo di f=\$(ENV3) of=\$(UPART)         @echo #inished          CoMakefileを実行       NOR	4

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	rootfsからrootfs.imgを更新する
	_Base_img.mk	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	NORフラッシュへrootfs.img,arm-
	mk	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.	rootfsの/lib/modulesを更新する
	mk	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	J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。



B6B-PH-K-S

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
јwарр	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。

<ul> <li>● ● ●</li> <li>● ●</li> <li>●</li> </ul>	スクトップ — minicom — 79×23	R <sub>M</sub>
minicom	bash	
Parkashing system		8
U-Boot 2011.09 (Mar 08 2013 - 1	6:14:41) YOODS ARM-Y V1.0	
I2C: ready DRAM: 128 MiB Flash: 64 MiB MMC: davinci: 0 In: serial Out: serial Err: serial ARM: 456 MHz		
Net: No ETH PHY detected!!! Error: Ethernet init failed! Board Net Initialization Eailed	I.	
DaVinci-EMACWarning: failed to	set MAC address	
Hit any key to stop autoboot:	0	

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	<pre>setenv bootargs 'mem=64M console=- ttyS2,115200n8 otg_mode=otg noinitrd root- =/dev/mtdblock5 rw rootwait rootfstype=jffs2'</pre>
SDカード	<pre>setenv bootargs 'mem=64M console=- ttyS2,115200n8 otg_mode=otg noinitrd root=- /dev/mmcblk0p1 rw rootwait rootfstype=ext3'</pre>

\*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ulmage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ulmage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 ulmage

(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ulmage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</pre>

# 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。

000	JUNKWare-Tools	12 <sup>21</sup>
JUNKWare-Tools	+	
/home	<pre>/home/jwapp/configs/ntp [executable file]</pre>	
<pre>jwapp configs hosts inittab interfaces junkport junkware mailrc ntp rc.local resolv.conf resolv.conf.or plc uboot-env brot i (1)NTPSEF (2)ntpdate</pre>	<pre>     # ###############################</pre>	
Documents		
JWTツリー設定		
環境変数設定		
21.20.5.20.05.20		1
	DI C 起動 マゴリお動	
	KG300 (19-11: ) (19-KG300 ) (98-68 ) 71.300 (85-68 ) (85-	
		_

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。

😻 JUNKWare-Tools - Mozi	lla Firefox		
ファイル(E) 編集(E) 表示(V)	履歴(S) ブックマーク(B) ツール(T) ヘルブ(H)		
JUNKWare-Tools	+		_
<ul> <li>Ig2.168.2.252</li> </ul>	コジェクト用ディレクトリ作成 🛛 🕆 マ 🛛 🔀 - Google 🔎	+	A
/home			^
・         ・	JUNKWare Tools		
i juboot-env L @Install_UBenv.mk	Web Development Environment		
Documents JWTッリー設定			
環境変数設定			-
			►

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。

● JUNKWare-Tools - Mozi ファイル(E) 編集(E) 表示(V) □ JUNKWare-Tools	lla Firefox 履歴(な) ブックマーク(空) ツール(① ヘルブ(吐) ・	
< € 192.168.2252	プログラムソースファイル作成 ☆ ♂ C Socele	🥬 ∔ 🏫
/home ↓ jvapp ↓ test1 ↓ ファイル作成 ↓ ディレクリ作成 ↓ 名前変更 ↓ 肖除 ↓ 切り取り	JUNKWare Tools Web Development Environment	
 <u>Doc</u> ●● USUはす <u>JMTツリー設定</u> 環境変数設定		n
● JUNKWare-Tools - Mozi ファイル(F) 編集(E) 表示(2)	lla Firefox 履歴(S) ブックマーク(B) ツール(T) ヘルブ(H)	
JUNKWare-Tools	+ (1) ソースファイルを保存	
€ € 192.168.2.252	(Ctrl+S) ☆ マ C S - Google	₽ 🖡 🏫
/home	/home/j appresent to the source file]	^
「jwapp     Ltest]     Lotest].c     Jarn-y     Jhtnl     JInstall     Packages     Juboot-env     Documents     JMTッリー最定	保存 編集破棄 剤 除 (Ctri+Sでも保存可能) 1: Ħinclude <stdio.h> 2: Ħinclude <fontl.h> 3: Ħinclude <fontl.h> 4: Ħinclude <sys ioctl.h=""> 5: Ħinclude (sys/ioctl.h&gt; 5: Ħinclude (sys/ioctl.h&gt;) 5: Ħinclude (sys/ioctl.h&gt;) 6: 1: int main(int argc,char **argv) { 8: int fd=open("/dev/armyio",0_RDWR); 9: 1: int substance (for the system) { 1: int substance (for the sy</sys></fontl.h></fontl.h></stdio.h>	
環境変数設定	<pre>10: ARMY_GPID d; 11: while(1) { 12: d_pin=GPID_ONI_12; 13: ioctl(fd.ARMY_GPID_GET,&amp;d); 14: printf("ONI_12:XdWn".d.stat); 15: d_pin=GPID_ONI_08; 16: ioctl(fd.ARMY_GPID_SET,&amp;d); 17: usleep(1000000); 18: } 19: close(fd); 20: return 0; 21: } (1) ソースコードを入力して「保存」 (2) 「コンパイル」ボタンを押下してソースコードの確認</pre>	H
<u>環境変数設定</u>	10:       ARMY_GPIO_d;         11:       while(1) {         12:       d.pin=GPIO_QNI_12;         13:       ioctl(fd.ARMY_GPIO_GET.&d);         14:       printf("OII_12:Xdhn",d.stat);         15:       d.pin=GPIO_QNI_0;         16:       ioctl(fd.ARMY_GPIO_SET.&d);         17:       usleep(1000000);         18:       }         19:       close(fd);         20:       return 0;         21:       >         (1)       ソースコードを入力して「保存」         (2)       「コンパイル」ボタンを押下してソースコードの確認	

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。



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。

JUNKWare-Tools - Mozil	la Firefox	1.42.6	
JUNKWare-Tools	+	NY 11	
♦ ④ 192.168.2.252		r ⊂ C   🔀 - Google	₽ ♦ 🏫
/home	/home/jwapp/test	1/ [directory]	
▲ivapp ▲test1 	<b>微</b> 能	NFX bootに実行権限を与える	実 行
- ∰Makefile L⊕testl.c ⊖arm-y	ファイル・ディレクトリを削除	(フィルドカード可)	削除
Packages uboot-env	ファイル実行権限	boot (フイルドカード可)) ● +x ○ -x	権限変更
L @Install_UBenv.mk	ディレクトリ内検索		検索
<u>JWTッリー設定</u> <u>環境変数設定</u>	ディレクトリを圧縮	◎ tar.gz ○ tgz (フイルドカード可)	圧 縮
	ブロジェクトテンプレート		配置
	ローカルファイルを転送	€ Ĵ	実 行
(1) test1 (2) test1 (3) ファイ	の下にbootを作成 ディレクトリを選択 ´ル実行権限にbootを	≥記載後、「権限変更」を押下してくた	±さ
	< ]	10	

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)	
<pre>echo 'cn1_8_gpio_L' &gt;/dev/armyio echo 'cn1_12_gpio_I' &gt;/dev/armyio /home/jwapp/test1/test1 &amp;</pre>	
77	
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
јwарр	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。

000	JUNKWare-Tools	H <sub>M</sub>
/home	/home/jwapp/configs/interfaces [file]	
→ □jwapp → □configs → hosts → inittab → inittab → inittab → inittab → inittab → inittab → inittab → issue → junkyort → junkyort → junkyort → mailrc → rc.local → resolv.conf → ieee1888 → C → Uibs → php → g exec.sh → test1.c → test1.c	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	

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。

	JUNKWare	Tools	2
/home	/home/jwapp/plc/objs/w	atchdog [tag file]	
<pre>Jumpp Jumpp Jump Jump Jump Jump Jump Jum</pre>	棄		
- 😹 sequence.rc	Status Index	パラメータ	コネクター
- 🛃 boot - 🔯 clock_set	ON + ( in ) 45 O		
□ default.classes □ Jistall_UBenv.mk □ restore.mk <u>Documents</u> <u>JWTツリー設定</u> 環境変数設定			

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。

00	JUNKWare-Tools	R <sub>M</sub>
/home	/home/jwapp/restore.mk [Makefile]	
<ul> <li>jwapp         <ul> <li>jwapp</li> <li>plc</li> <li>plc</li> <li>plc</li> <li>clock_set</li> <li>clock_set</li> <li>clock_set</li> <li>prestore.mk</li> </ul> </li> </ul>	(Ctrl+Sでも保存可能)     (Ctrl+Sでも保存可能)     (ctrl+Sでも保存可能)     sudo rm -rf /home/jwapp/*     sudo tar zxvf /home/jwapp-orig.tar.gz -C /home     (	

✓<u>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</u> 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。

00	JUNKWare-Tools	H <sub>21</sub>
JUNKWare-Tools	+	
/home	<pre>/home/jwapp/Install/Install_Userland.mk [Makefile]</pre>	
<pre>&gt; jwapp &gt; call_voice &gt; calses &gt; call_voice &gt; call_voice &gt; call_ses &gt; ca</pre>	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	

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 <sub>OH</sub>	High-level output voltage	DVDD= 3.15V, I <sub>OH</sub> = -4 mA	2.4			V
	(dual-voltage LVCMOS IOs at 3.3V) <sup>(1)</sup>	DVDD= 3.15V, I <sub>OH</sub> = -100 µA	2.95			V
	High-level output voltage (dual-voltage LVCMOS IOs at 1.8V) <sup>(1)</sup>	DVDD= 1.71V, I <sub>OH</sub> = -2 mA	DVDD-0.45			V
	Low-level output voltage	DVDD= 3.15V, I <sub>OL</sub> = 4mA			0.4	V
Voi	(dual-voltage LVCMOS I/Os at 3.3V)	DVDD= 3.15V, I <sub>OL</sub> = 100 µA			0.2	V
- OL	Low-level output voltage (dual-voltage LVCMOS I/Os at 1.8V)	DVDD= 1.71V, I <sub>OL</sub> = 2mA			0.45	v
	Input current <sup>(1)</sup> (dual-voltage LVCMOS I/Os)	V <sub>I</sub> = VSS to DVDD without opposing internal resistor			±9	μΑ
(2)		V <sub>I</sub> = VSS to DVDD with opposing internal pullup resistor <sup>(3)</sup>	70		310	μΑ
II (-)		V <sub>I</sub> = VSS to DVDD with opposing internal pulldown resistor <sup>(3)</sup>	-75		-270	μΑ
	Input current (DDR2/mDDR I/Os)	V <sub>I</sub> = VSS to DVDD with opposing internal pulldown resistor <sup>(3)</sup>	-77		-286	μA
I <sub>ОН</sub>	High-level output current <sup>(1)</sup> (dual-voltage LVCMOS I/Os)				-6	mA
I <sub>OL</sub>	Low-level output current <sup>(1)</sup> (dual-voltage LVCMOS I/Os)				6	mA
Capacitance	Input capacitance (dual-voltage LVCMOS)			3		pF
	Output capacitance (dual-voltage LVCMOS)			3		pF

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<sub>1</sub> applies to input-only pins and bi-directional pins. For input-only pins, I<sub>1</sub> indicates the input leakage current. For bi-directional pins, I<sub>1</sub> indicates the input leakage current and off-state (H-2) 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。

00		Www.yoods.co.jp/docs	ドキュメントー /index.html	覧 株式会社YOODS		≝ ■ ■		
				株士	会社 🖵	loos		
	ホーム	ニュースリリース	製品紹介	技術ドキュメント	会社案内			
	ホーム > 公開ドキュメ	ント > ドキュメント一覧						
	ARM-Y関連ドキュメント(右クリック→ダウンロードしてください)							
	ルートフ	ァイルシステムイメージ(NORフ	ラッシュ用)					
	linux.inf	ダウンロード〜Linux USB Ether	net/RNDIS Gadget					
	開発用マイクロSDファイルシステム							
	4GByte以上のマイクロSDの第一パーティションをlinuxパーティション,ext3フォーマットにして、このtar ポールを解凍してください。 ※このシステムについては全て自己責任でご使用下さい。基本的にサポートは行いません。							
	u-boot設定パーミッションイメージ[NORブート用, MMCブート用]							
	u-boot	込みツール(u-bootイメージを含	t)					

<u>http://www.yoods.co.jp/docs/index.html</u>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。