GKRX7_1M マイコン学習セット 入門編

初版 2018.1.26

【 製品概要 】

本マニュアルはマイコンの学習を行うためのハードウエア、ソフトウエアの入門編です。

入門編では開発環境の構築からマイコンプログラムの作成、実行までを詳細に解説。また、基本的な周辺 機能である I/Oポート、USB、A/Dコンバータ、D/Aコンバータの使い方を平易なサンプルプ ログラムをもとに説明します。

ハードウエアはマイコン基板 BCRX7_1M CPUボード+専用ハードウエアで構成される学習ボ ードです。

ソフトウエアはルネサス社の無償統合開発環境CS+ for CC上で動くサンプルプログラムです。 C言語で書かれています。めんどくさい初期設定はCS+の「コード生成」機能で簡単に行え、240M Hzクロック、FPU内蔵で驚異の演算速度を体験出来ます。

※本学習ボード開発にはルネサスエレクトロニクス社製E1が別途必要です。



1. 開発環境、事前準備

- 1-1. 開発環境
 - a:開発セット 同梱物
 - b:BCRX7_1M CPUボードの特徴
 - c: E1エミュレータ (デバッカ)
 - d:無償のCS+、RX用Cコンパイラのダウンロード
 - e:CDコピー、デバイスドライバのインストゥール
- 1-2 動作、デバック
 - a:CS+起動、コンパイル、書き込み、動作
 - b:新しいプログラムを作る CS+ 操作
 b-1:初めにIOポートの設定

- b-2:プログラムの書き込み、書く場所の注意
- b-3:パワーオンリセットを設定する
- b-4:デバッカの設定がデフォルトはエミュレータなので注意
- b-5:E1から電源供給
- b-6:クロック発生回路を設定する必要があります
- c : その他
 - c-1:動作中に変数の変化を見るには?
 c-2:サンプルを走らせるときにvect.hの重複するアドレスを削除
 c-3:三角関数math.hはインクルードもCS+の設定も必要
 c-4:割り込みが入っているか?周期は?簡単なチェック方法
 c-5:既存のプログラムを雛形として新しいプログラムを作る
 c-6:コード生成と見落としがちな注意点

2. サンプルプログラム

2 — 1.	sample1	出力ポートのON,OFF	LEDをチカチカさせます。
2 — 2.	sample2	SIO (USB)	USB経由でパソコンとやりとり
2 — 3.	sample3	A/D変換をUSB出力	アナログ量をデジタルに変換し、
			パソコンに出力します。
2 — 4 .	sample4	割り込み	1msec間隔定周期タイマ割込み
2 — 5 .	sample5	PWM出力	PWMでLEDの明るさを変えます
2 — 6 .	sample6	三角、対数、平方根関数	関数演算速度を検証
2 — 7 .	sample7	D/Aにsin演算した正弦派	皮を出力する
			音の高低をスピーカーで聞く
2 — 8 .	sample8	液晶表示	ADの値を液晶表示します。
2 — 9.	sample9	ステージを動かす	パルスモータ制御の基本を学習

1-1. 開発環境

a:学習セット同梱物

BCRX7_1M 学習ボード CD(サンプルプログラム、デバイスドライバ、ドキュメント) マニュアル(入門編 本誌、応用編) ACアダプタ、USB(フルサイズ)ケーブル、パルスモータステージ ※開発に必要なルネサスエレクトロニクス社製デバッカE1は同封されておりません。別途必要です。



b:BCRX7_1M CPUボードの特徴(R5F571MLCDFP 100ピン搭 載)

●ルネサス独自のRX CPUコア、内部32ビットデータバス幅マイクロコンピュータ。3.3V 2 40MHz動作可能。従来のRX製品に搭載されたコアとの互換性を踏襲しながらも更に強力に進化した RXv2コアを採用し、フラッシュ内蔵マイコンとして最高クラスとなる1044coremarkを実 現。フラッシュメモリ向けに最適化したキャッシュ(AFU)により240MHzノーウェイト相当のフ ラッシュメモリアクセスが可能。

●FPU 単精度浮動小数点数(32ビット) IEEE754に準拠したデータタイプ、および例外

●メモリ容量 内蔵フラッシュROM 4Mバイト、内蔵RAM512Kバイト 内蔵データフラッシュ 64Kバイト

●A/Dコンバータ : 12ビット分解能×22 変換速度0.48μsec/1ch(ADCLK=6 0MHz)

●D/Aコンバータ:12ビット分解能×1

●外部バス拡張機能:あり(外部にデータバス、アドレスバス等出力できます)

● I /Oポート:入出力:78、入力:1、プルアップ抵抗:78 オープンドレイン出力:78 5V トレラント:17

●タイマ:16ビットタイマパルスユニット(TPUa)(16ビット×6チャンネル)、ポートアウトプ ットイネーブル3(POE3a)、汎用PWMタイマ(GPTa)(16ビット×4チャンネル)、プログラ マブルパルスジェネレータ(PPG)、8ビットタイマ(TMRb)(8ビット×2チャンネル)×2ユニ ット、コンペアマッチタイマ(CMT)(16ビット×2)×2ユニット、コンペアマッチタイマW(CM TW)(32ビット×1チャンネル)×2ユニット、リアルタイムクロック(RTCd)、ウオッチドッグ タイマ(WDTA)、独立ウオッチドグタイマ(IWDTAa)

 ●イーサネットコントローラ(ETHERC)2チャンネル、USB2.0FSホスト/ファンクション モジュール(USBb)1ポート、シリアルコミュニュケーションインターフェイス(SCIg、SCI h)9チャンネル、FIFO内蔵シリアルコミュニュケーションインターフェイス(SCIFA)4チャ ンネル、IICバスインターフェイス(RIICa)2チャンネル、CANモジュール(CAN)3チャ ンネル、シリアルペリフェラルインターフェイス(RSPIa)2チャンネル、クワッドシリアルペリフ ェラルインターフェイス(QSPI)1チャンネル ●シリアルサウンドインターフェイス(SSI)2チャンネル、サンプリングレートコンバータ(SRC)、 MMCホストインターフェイス(MMCIF)、パラレルデータキャプチャユニット(PDC)、温度セン サ等内蔵。

●オンチップデバッキングシステム:(FINEインターフェイス)

●動作周囲温度:-40~+85℃

●EEPROM: 25LC256(32Kバイト) 電源OFFでもデータ保持。 ※オプション(実 装品はご相談下さい)

●電源 2.7V~3.6V 単一 40mA(240MHz動作 TYPE)

E1デバッカを使用して動作させる時E1から3.3Vの電源を供給できます。 デバック時など200mA以内の使用であれば他に用意する必要はありません。

●クリスタル:メイン 12MHz(×4逓倍で50MHz作成)実装済み。

●デバックコネクタ: E1用(FINEインターフェイス)デバックコネクタ実装済み。

●基板サイズ 64×48×13(H)mm

●基板仕上げ 金メッキ RoHS指令準拠 基板、部品、半田付け全ての工程でRoHS指令準拠仕様。



基板大きさ(部品面)

c: E1エミュレータ



概要

E1エミュレータは、ルネサス主要マイコンに対応したオンチップデバッギングエミュレータです。基本 的なデバッグ機能を有した低価格の購入しやすい開発ツールで、フラッシュプログラマとしても使用可能 です。

C言語ソースデバックが可能で、1 行実行、ブレークポイント設定、変数、レジスタ、メモリ参照等々、 従来であれば高価なICEしか出来なかった機能が、安価に実現されています。また、使い方もHEW(統 合開発環境)のE8aと同じで、経験があれば半日で、無くても1日で必要な操作を会得することが出来 ると思います。

マイコンとの通信として、シリアル接続方式とJTAG接続方式の2種類に対応しています。使用可能な デバッグインタフェースは、ご使用になるマイコンにより異なります。

また、基本デバッグ機能に加え、ホットプラグイン機能(動作中のユーザシステムに後からE1エミュレ ータを接続して、プログラムの動作確認を行うことが可能)を搭載しているため、プログラムのデバッグ・ 性能評価に大きく貢献できます。

対応MPU

- RH850,V850 ファミリ
- RX ファミリ
- RL78 ファミリ
- R8C ファミリ
- 78K ファミリ



E1を購入するとCDが添付されています。インターネットが動作する環境で、パソコンにCDをセット し、README__j.HTMを実行、ネットから最新E1ドライバーのインストールを行ってください。 続いて、ネットから開発環境CS+とCコンパイラの最新版ダウンロードを行います。 d:無償版RX用Cコンパイラのダウンロード

省略

1-2 動作、デバック

a:CS+ for CC 起動、コンパイル、書き込み、動作



CDに添付しているサンプルプログラムを使って、コンパイル、書き込み、動作の方法を示します。

CS+ for CCを起動します。ここでは例としてRX7_1A_sutdy_sample¥入門 ¥sample1を動作させます。基板上のLED D1が点滅するプログラムです。 ファイル → ファイルを開く → sample1.mtpjをダブルクリックします。

🙆 ファイルを開く					x
← → ~ ↑ 📜 « RX7	71M_STUDY_sample > Introduction > sample1	› ~ ٽ	sample1の検索		9
整理▼ 新しいフォルダー				· 🔳	?
学習ステージ2017(^	名前	更新日時	種類	サイズ	
🔜 デスクトップ	📙 cg_src	2017/07/13 16:24	ファイル フォルダー		
neDrive	DefaultBuild	2017/07/13 16:24	ファイル フォルダー		
 ジドキュメント 浸画像 ≡ 2 Beyond the river 	 Sample1.mtpj 種類: MTPJ ファイル サイズ: 17.0 MB 更新日時: 2016/09/23 15:23 	2016/09/23 15:23	MTPJ ファイル	17,45	8 KB

プロジェクトツリーが表示されます。



sample1. cが中央に表示されます。とりあえず、実行してみます。E1のケーブルを基板のCN 1に挿入します。電源は添付のACアダプタより供給します。(写真ご参考)

電源アダプタをコンセントに入れて電源を供給すると9 V側LED D24が緑色に点灯、3.3 V側L ED D25が赤色に点灯します。E1のVCCが緑色に点灯します。→ 以上3点のLEDが点灯しな い場合、異常です。何か理由があります。コンセントからACアダプタを抜き、原因不明の場合、弊社に ご連絡ください。



「デバック・ツールヘプログラムを転送」をクリック。



上手く転送できると、今まで表示されていなかったプログラムの絶対番地が表示されます。カーソルの黄 色い帯が表示されます。

行	-	アドレス	🗖 🖨		
67 68 69				* Arguments : Hene * Return Value : None ************************************	**********
70		ffc0057e		void main(void)	
71 72 73 74 75 76		ffc00580	I	∃\ R_MAIN_UserInit(); /* Start user code. Do not edit comment genera while (1U) {	ted here */
77 78 79 80 81	1	ffc00599 ffc0059c ffc0059e		PORT2.PODR.BYTE = 0×06; PORT5.PODR.BIT.B5 = 1; PORTC.PODR.BIT.B0 = 1;	//PWMにつかう赤、緑 ON //D21 ON //CPU基板上のD1 ON
82		ffc00597	1	main_wait(5000000);	
83 84 85 86 87	•	ffc005a5 ffc005a8 ffc005aa		PORT2.PODR.BYTE = 0×08; PORT5.PODR.BIT.B5 = 0; PORTC.PODR.BIT.B0 = 0;	//PWMにつかう青 ON //D21 OFF //CPU基板上のD1 OFF
88		ffc005a3	1	main_wait(5000000);	
90 89				}	

ここまでいかなかった場合、E1のデバイスドライバインストゥールをご検証願います。

次に、プログラムを動作させます。「CPUリセット後、プログラムを実行」をクリック。



セーブして

E) 表	示(V) プロジェクト(P) ビルド(B) デバッグ	/(D) ツー
3	🛛 🕼 🖄 🗠 🗠 📾 🖉	- 10
	ファイルに指定された項目を保存します。((Ctrl+S)
「ビル	レド後、デバック・ツールヘプログラムを転送」	をクリック。



「CPUリセット後、プログラムを実行」をクリック。 LEDの点滅が先ほどより、遅くなったのが目視できましたでしょうか?

次に、ブレークポイントの設定を行ってみます。一度、プログラムを停止させます。 ブレークポイントを2か所 設定しました。ここをマウスでダブルクリック。ブレークポイント設定マー クとオレンジのカーソルが表示されます。



「CPUリセット後、プログラムを実行」します。





プログラムはブレークポイントで停止し、黄色いカーソル(現在のプログラムカウンタの位置)で停止し ます。カーソルのある行はまだ実行されていません。その1行前のPORT2. PODR. BYTE=0 ×06;命令によりPORT2 = 0×06となるので、赤と緑のLEDが点灯します。理由、詳細は sample1の解説で行います。

更に「プログラムを現在の位置から実行」をクリックすると、もう一つのブレークポイントで停止し、P ORT2. PODR. BYTE=0×08;命令実行によりPORT2は先ほどの赤と緑のLEDは消灯 し、青のLEDが点灯します





以上が、プログラムのコンパイル、E1へのダウンロード、実行、修正、ブレークポイント設定、動作の 概要です。

b:新しいプログラムを作る

省略

sample1. cを参考にもっとも簡単なプログラムを記入してみます。基板上のLEDを点滅させる プログラムです。基板上のLED D1はPORTCのPC0に接続されています。

b-1:初めに I Oポートの設定

ここが従来のRXマイコン開発と変わった部分になります。コード生成ツールが完備してIO設定に限らず、あらゆるペリフェラルの初期設定をプログラムレスで開発できるようになりました(2016.9)。 RL78のCS+開発と同じように行えます。

ここではPORTCのPCOを出力設定します。コード生成(設計ルーツ)→ 周辺機能 → I / Oポー トを選択。PORTCのPCOを出力にチェック。

 $\backslash \setminus$

٢	test_sample - CS+ for CC - [周辺機能					
ファ	/ル(F) 編集(E) 表示(V) プロジェクト(F	P) ビルド(B) デバッグ(D) ツール	(T) ウイン	で(W) ヘルプ(H)		
6	、スタート(S) 🔒 🔒 🗿 🗄 🔉 🐚 📸	う (m) 単 単 単		00% 🔫 😽 😽 DefaultBuild	- 🔨	🛛 🖓 🎝 🤭 I 🖲 🕞 🖸
	I 🖗 🔍 🗗					
プロ:	ジェクト・ツリー - + ×	🔄 プロパティ 🎏 周辺機能				
ĝ	020	🕅 コードを生成する ا 🙇 🖨	//////////////////////////////////////		0.000	🗏 🗟 🖧 ភី ភី 📲 ភី
-	Test_sample (プロジェクト)* ^	Port0 Port1 Port2 Port	3 Port4	Port5 PortA PortB PortC Po	rtD PortE P	ortJ
	- 〒 R5F5/1MLCXFP (マイクロコント □ □ コード生成 (設計ツール)	-PC0	•			
	国▲端子図	● 使用しない ○ 入力	● 出力	□ 内蔵プルアップ CMOS 出力	× [_ 1を出力 高駆動出力
	日間周辺機能		○ # +			
	● クロック発生回路		0 10			_ (2U)] (antenu))
	● ● ● クロック周波数精度測定	● 使用しない ○ 入力	〇 出力	□ 内蔵ブルアップ OMOS 出力	× [1を出力 高駆動出力
	🥄 消費電力低減機能	PC3				
		● 使用しない ○ 入力	〇 出力	□ 内蔵ブルアップ CMOS 出力	× [1を出力 高駆動出力
		-PC4	~			
	データトランスファコントロー =	● 使用しない ○ 入力	〇出力	 内蔵ブルアップ CMOS 出力 	Y	_1を出力 高駆動出力
	▲ イベントリンクコントローヺ =	-PC5	0 4 1			□1を中井□ 直販新中市
		PC6			L	
	■ ペルテノアノウンヨノウ1 マ/	● 使用しない ○ 入力	〇 出力	 内蔵ブルアップ OMOS 出力 	× [1を出力 高駆動出力
	■ [●] 汎用PWMタイマ	-PC7				
	由 ● 16ビットタイマパルスユニッ	● 使用しない ○ 入力	〇 出力	□ 内蔵プルアップ CMOS 出力	× [1を出力 高駆動出力

これで「コードを生成する」をクリックするとPORTCのPCOを出力に設定する初期化プログラムが 自動生成されます。

b-2: プログラムの書き込み、書く場所の注意

LED D1が接続されているPORTCのPCOをON, OFFさせるプログラムを書き込みます。

```
Global variables and functions
****
/* Start user code for global. Do not edit comment generated here */
void lwait(long wdata)
⊟{
      while(wdata != 0)
      ł
            wdata--:
      3
]
 /* End user code. Do not edit comment generated here */
void R MAIN UserInit(void);
* Function Name: main
* Description : This function implements main function.
* Arguments : None
* Return Value : None
void main(void)
Ð{
   R MAIN UserInit();
   /* Start user code. Do not edit comment generated here */
   while (1U)
   ł
      PORTC.PODR.BIT.BO = 1:
      lwait(500000);
PORTC.PODR.BIT.B0 = 0;
      lwait(500000);
   /* End user code. Do not edit comment generated here */
3
```

このときに、必ず

/* Start user code for global. Do not edit comment generated here */

プログラム、define等定義、int, short等変数定義等 ユーザーが書くもの全て

/* End user code. Do not edit comment generated here */

の間にプログラムや定義文を書くようにしてください。これは新しいペリフェラルを追加したり、変更したりした場合、新たに「コード生成」させる必要がありますが、このコメントサンドイッチ中のコードは「コード生成」しても消えません。それ以外は消えますので、注意が必要です。

ビルド後、「デバックツールへダウンロード」 をクリックするとノーエラーでコンパイルされ、ダウンロ ード、実行されますが、LEDは光りません。 b-3:パワーオンリセットを設定する



b-4:デバッカの設定がデフォルトはエミュレータなので注意

省略

b- 5: E1から電源供給

省略

b-6:クロック発生回路を設定する必要があります

ここから



クロック発生をクリック。クロック設定で上記のように設定します。

しシュテムカロック語空			
クロックソース	PLL回路	~	
システムクロック(IOLK)	×1 ~	240	(MHz)
周辺モジュールクロック(PCLKA)	×1/2 🗸	120	(MHz)
周辺モジュールクロック(PCLKB)	×1/4 👻	60	(MHz)
周辺モジュールクロック(PCLKC)	×1/4 ¥	60	(MHz)
周辺モジュールクロック(PCLKD)	×1/4 ¥	60	(MHz)
外部バスクロック選択(BCLK)	×1/2 👻	120	(MHz)
FlashIFクロック(FCLK)	×1/4 ¥	60	(MHz)
USBクロック(UCLK)	x 1/5 🗸 🗸	48	(MHz)

更に、各々の部分のクロックを上記のように設定します。「コード生成」を行い、全てのファイルを変更、 コンパイル、ダウンロード、実行 でCPUはやっと240MHzで動作し、sample1とほぼ同じ 間隔でLEDが点滅するのが確認できます。

c:その他

c-1:動作中に変数の変化を見るには?

省略

ウォッチ1			д Х
ا 🗙 🕼 🧏 🧠 ا 🏽	表記(N)▼ 👼		
ウォッチ式		型情報(バイト数)	アドレス メイ
🐻 SCI1.SSR	0x84	IOR(1)	0x0008a024
🖽 🔍 rx_data		unsigned char…	0×00000004
😜 adO	3711 (0x0e7f)	uint16_t(2)	0x000000c4
🔍 templ	1999 (OxO7cf)	uint16_t(2)	0x000000c6 =
🔍 stvolt	1567 (OxO61f)	uint16_t(2)	0x00000c8
⊟ 🔍 tx_buffer	"ADO = 2.991V…	unsigned char…	0x0000018
€ [0]	'A' (0x41)	unsigned char(1)	0x00000018
😜 [1]	'D' (0x44)	unsigned char(1)	0×00000019
😜 [2]	'O' (Ox3O)	unsigned char(1)	0x0000001a
😜 [3]	''(0x20)	unsigned char(1)	0x000001b
😜 [4]	'=' (0x3d)	unsigned char(1)	0x0000001c
😜 [5]	''(0x20)	unsigned char(1)	0x0000001d
€ [6]	'2' (0x32)	unsigned char(1)	0x0000001e
🔍 [7]	'.' (Ox2e)	unsigned char(1)	0x0000001f
😜 [8]	'9' (0x39)	unsigned char(1)	0x0000020
😜 [9]	'9' (0x39)	unsigned char(1)	0x0000021
🔍 [10]	'4' (0x34)	unsigned char(1)	0x0000022
😜 [11]	'V' (0x56)	unsigned char(1)	0x0000023
😜 [12]	''(0x20)	unsigned char(1)	0x0000024
😜 [13]	'T' (0x54)	unsigned char(1)	0x0000025
😜 [14]	'E' (0x45)	unsigned char(1)	0x0000026
😜 [15]	'M' (0x4d)	unsigned char(1)	0×00000027
😜 [16]	'P' (0x50)	unsigned char(1)	0×00000028
😜 [17]	''(0x20)	unsigned char(1)	0×00000029
😜 [18]	'=' (0x3d)	unsigned char(1)	0x000002a 🗸
<	III		>

c-2:サンプルを走らせるときにvect. hの重複するアドレスを削除

省略

c-3:三角関数math.hはインクルードもCS+の設定も必要 省略

c-4:割り込みが入っているか?周期は?簡単なチェック方法

省略

c-5:既存のプログラムを雛形として新しいプログラムを作る

省略

今まで見てきたように新規でプロジェクトを設定すると、新たに設定しなくてはならない項目が多く、従 来のプロジェクトをもとに新しいプロジェクトを作成できると便利です。test_sampleを元に 製作してみます。

lb test_sample
lb test_sample2

c-6:コード生成と見落しがちな注意点

省略

コード生成で変更したつもりになっていても、これを忘れると変わっていないので、ご注意下さい。

2. サンプルプログラム

初めにCN1にE1のケーブルを差します。E1はCS+ for CCが動作しているパソコンのUSB に接続して下さい。



次に添付のACアダプターをJ1に差して下さい。ACアダプタをAC100V コンセントに挿入し、 LED D24 が緑(9V)、LED D25が赤(3.3V)に点灯すれば正常です。



サンプルプログラムは全てコンパイル、動作確認済みです。「デバックツールへプログラムをダウンロード」後 」



sample1 ポートのON/OFF

【 概要 】

CPU基板上のLED D1(ポートPCO)、D21(ポートP55)をビット命令で、3色のLED D16、D17、D18(ポートP21, 22, 23)をバイト命令で、ポートのON/OFFにより点滅 を繰り返します。

【周辺機能の説明】

サンプルプログラムは周辺機能→I/Oポート、ポートPCO、P55、P21, 22, 23を出力に設 定し、「コード生成」してあります。この機能により、ユーザーはポートの初期設定を文章で記入する必 要が無く、「コード生成」で自動的に作成され、電源投入時、自動的に実行されます。



【 プログラム 】

/* End user code. Do not edit comment generated here */

void R_MAIN_UserInit(void);

* Function Name: main

* Description : This function implements main function.

* Arguments : None

* Return Value : None ********* void main(void) { R_MAIN_UserInit(); /* Start user code. Do not edit comment generated here */ 3 while (1U) { 4 PORT2.PODR.BYTE = 0x06; //PWMにつかう赤、緑 ON //D21 ON PORT5.PODR.BIT.B5 = 1; PORTC.PODR.BIT.B0 = 1; //CPU基板上のD1 ON 5 main_wait(500000); 6 PORT2.PODR.BYTE = 0x08;//PWMにつかう青 ON PORT5.PODR.BIT.B5 = 0; //D21 OFF PORTC.PODR.BIT.B0 = 0;//CPU基板上のD1 OFF \bigcirc main wait(5000000); }

/* End user code. Do not edit comment generated here */

【解説】

)/* Start user code for global. Do not edit comment generated here */

```
ユーザープログラム
```

/* End user code. Do not edit comment generated here */ 先も書きましたが、ユーザープログラムはこのコメントの内側に作成してください。でないと、「コード 生成」を行うと消えてしまいます。

```
②void main_wait(long ltime)
```

```
while(ltime != 0)
{
    ltime--;
}
```

}

{

LEDのON, OFFを人間の目で確認するためには時間の早すぎるON, OFFではだめで、数100 msecの時間(ウエイト)を作るためのプログラムです。

 (3) /* Start user code. Do not edit comment generated here */ while (1U) {

プログラムは/* Start user code. 以下に記入してください。

(4)

PORT2.PODR.BYTE = 0x06;	//PWMにつかう赤、緑 ON
PORT5.PODR.BIT.B5 = 1;	//D21 ON
PORTC.PODR.BIT.B0 = 1;	//CPU基板上のD1 ON

バイト命令で0×06=00000110Bです。1が立つビットはP21とP22で、それぞれの

P 2 7	P 2 6	P 2 5	P 2 4	P 2 3	P 2 2	P 2 1	P 2 0
0	0	0	0	0	1	1	0

ポートに1 = 3.3 Vが出力され、トランジスタQ2、Q3がONし、赤、緑のLEDが点灯します。

PORT5のB5、PORTCのPC0に繋がれているLED もビット命令'1'で電流が流れて点灯 します。CH3はTP19 Q4トランジスタをドライブしている信号、CH4はQ4コレクタ側の信号 です。



500000という数を減算して0になるまでの間、ここで時間が消費されます。

6

PORT2.PODR.BYTE = 0x08;	//PWMにつかう青	ON
PORT5.PODR.BIT.B5 = 0;	//D21 OFF	
PORTC.PODR.BIT.B0 = 0;	//CPU基板上のD1	OFF

0 x 0 8 を出力しています。

0x08=00001000Bです。PC3のみ1でP23に接続された青のLED D18がONしま す。

0	0	0	0	1	0		0
0	5	0	0	1	0	0	0

PORT5のP55に繋がれているLED D21もデータ'0'で電流が止まり、消灯します。 PORTCのPC0に繋がれているLED D1もデータ'0'で電流が止まり、消灯します。

⑦ main_wait(500000);

消灯している間も点灯同様に時間を保持しています。

wait()に入れる数を変えたり、一方だけ変えて、セーブ、コンパイル、CPUリセット後、実行でLEDの点灯時間が変わるのを確認できます。

2-2 sample2 SIO(USB) USB経由でパソコンとやりとり

【 概要 】

USB出力をパソコンと接続し、データのやり取りを行います。お手数ですが、テラタームやハイパータ ーミナルなどのターミナルプログラムを使用しますので、無い方は、ネットで検索し、インストゥール願 います。例ではテラタームで行います。ボーレートは38400bpsに設定して下さい。



スタート→右クリック→デバイスマネージャー → ポート(COMとLPT)でUSB Serial Port (COMxx)があることを確認して下さい (Windows10の場合)。例ではCOM4となっています。



Tera Tremをシリアルポート COM4 \rightarrow OKとします。

Tera Term: 新しい接続	x
О тор/ір ホスト サーヒ	 (T): 192.168.11.21 ▼ ビヒストリ(O) (ス: O Telnet TCPポート#(P): 22 ● SSH SSHバージョン(V): SSH2 ▼ ● その他 プロトコル(C): UNSPEC ▼
●シリアル ポート	(R): COM4: USB Serial Port (COM4) 🗸
0	K キャンセル ヘルブ(H)
設定→シリアル	
Tera Term: シリアルポート	、 設定 X
ポート(P):	COM4 V OK
ボー・レート(E	3): 38400 🗸
データ(D):	8 bit
パリティ(A):	none v
ストップ(S):	1 bit ~ ヘルプ(H)
フロー制御(F); none v
送信遅延 0 ミ	リ秒/字(C) 0 ミリ秒/行(L)

CS+でsample2を開き、デバック・ツールヘプログラムダウンロード→CPUリセット後、プロ グラム実行。



USB Test、、 と表示され、PCのキーボードを何か押すたびに、押した文字が表示されると動作としてはOKです。

プログラムはパソコンのキーボードを押した文字がCPU基板に送信され、それを返信(エコーバック) し、表示されるようになっています。 【周辺機能の説明】

省略

【 プログラム 】

省略

【解説】

省略

2-3 sample3 A/D変換をUSB出力

【 動作概要 】

RX71M CPUは2ユニットの12bit ADコンバータを持っています。ユニット0は最大8chのアナログ入力を選択できます。ユニット1は最大21チャンネルのアナログ入力、内部温度、内部基準電圧を選択できます。

ハードウエアはサーミスタが繋がれているANOOO(P40 CN4 16番)、CPU内部温度、基準 電圧を入力とし、A/D変換した値をUSBからパソコンに送り、表示しています。

また、温度の変化はR8サーミスタを指で触ると加熱され、抵抗値が下がり、分圧されている電圧値が下 がるのが目視出来ます。



【周辺機能の説明】

SCI2に加えて、12ビットA/Dコンバータを使用しています。



S12AD	0 S12AD1						
設定1	1 設定 2						
-S12A	DO 動作設定 ——						
0	使用しない		• 使月	月する			
注意	12 ビットA/D コンパ を、ポート02, 01, 00、 大値と最小値を除い	ータのユニット0 を使 ポート9、ポートロ、 て 平林族 >スカジル:	明する ポートE 対策を行	昜合は、ポート07 を出力端子として テーテーノださい	, 05, 03, ポー 使用する場合	ヽ4 のポート出フ 含は、A/D 変換	りは使用しないでくださし を複数回実施し、
-動作1	モードの設定 ―――						
	シングルスキャンモート	ų.	○ グル	ープスキャンモード	!	○ 連続ス	キャンモード
-ダブル	トリガモード 設定 —						
	禁止		○ 許				
- <u>6</u> 28	診断 設定 ———						
- E	-٣		未使用	1	\sim		
使	用電圧		0Vの冒	電圧を使って自己	診断を行う	\checkmark	
-断線	検出 アシスト 設定・						
÷ ۲	ャージ 設定		未使用	1	\sim		
ج-	ャージ期間		1 ADC	LK	\sim		
-グルー	-プスキャン優先 設定						
IJ.	ループA優先設定		グルー	ĴAの優先制御	動作を行わな(,\ V	
<i>Б.</i>	ループBアクション		A/D	[換動作中断後]	の再起動をした	30 ~	
-A / D)変換値数 設定 —						
	加算モード		€平 ●	タモード			
- アナ ロ	り入力チャネル設定						
		変換 (グループA)	変	換 (グループB)	AD変換値	を加算/平均	専用サンプルホールド
AN	1000	~					v

ANOOO入力のための設定です。S12ADOを選択しています。

温度センサ、内部基準電圧を読み込むためのS12AD1の設定です。

S12AD0 S12AD1			
設定1 設定2			
-S12AD1 動作設定			
○ 使用しない	 使用する 		
注12 ビットA/D コンパータのユニット1	を使用する場合で、ポート02,01,00	、ボート9、ボートロ、ボートEを出力端子として使用する場合は、	
A/D 変換を複数回美施し、取入10C	取り加速を除いてキャンをとるねとの対象	RITOLAZEU.	
- 動作モードの設定			
 シングルスキャンモード 	○ グループスキャンモード	○ 連続スキャンモード	
- ダブルトリガモード 設定			
0 #1	O #*		
(•) # It	() #+		

- アナログ 入力チャネル設定			
, , , , , , , , , , , , , , , , , , ,	変換 (グループA)	変換 (グループB)	AD変換値を加算/平均
AN1 00			
AN1 01			
AN1 02			
AN1 03			
AN1 04			
AN1 05			
AN1 06			
AN1 07			
AN1 08			
AN1 09			
AN110			
AN111			
AN112			
AN113			
温度センサ出力	✓		
内部基準電圧	✓		

レジスタ等の詳細はハードウエアマニュアル REV1.00 2672頁等参照。

【 プログラム 】

void main(void)

- {
- R_MAIN_UserInit();

/* Start user code. Do not edit comment generated here */

R_SCI2_Start();

//SCI2動作開始

① ②	R_S12AD0_Start(); //AD0動 R_S12AD1_Start(): //AD1動	作開始 作開始	
0	R_SCI2_Serial_Receive(rx_data,1); rx_flg2 = 0; tx_end_flg2 = 0;		
	<u></u>		
	PORTC.PODR.BIT.B0 = 0;	//LED1 OFF	
	R_SCI2_Serial_Send(String_0,37); tx_end_wait2();	//Opening message //送信終了まち	
	while (1U) {		
3	S12AD.ADCSR.BIT.ADS	T = 1;	//ADスタート
	S12AD1.ADCSR.BIT.ADS	ST = 1;	//AD1スタート
4	while(S12AD.ADCSR.BIT	T.ADST)	
	,		
	ad0 = S1	2AD.ADDR0;	//AD/恒
5	while(S12AD1.ADCSR.BI	T.ADST)	
	;		
	temp1 = S12AD1.A	DTSDR;	//温度
	stvolt = S12AD1.AI	DOCDR;	//基準電圧
6	fdata1 = ad0/(4095/2)	3.3);	//データ→電圧V換算
\bigcirc	fdata2 = temp1/(4.09)	95/3.3);	//データ電圧mV換算
	$fdata2 \neq 4.1;$		//温度= (データ/4.1mV) -277.4
	fdata2 -= 277.4;		//0℃電上1137.5mⅤ/4.1mV
9	fdata3 = stvolt/(4093)	5/3.3);	//データ→電圧換算 type1.25V 1.20~1.30V
10	sprintf(tx buffer,"AD0 = %.3fV	TEMP = %.2 fd STD	VOLT = %.3 f V n r'', fdata1, fdata2, fdata3);
(11)	R_SCI2_Serial Sen	d(tx_buffer,sizeof(tx_bu	iffer)); //データをUSB出力
	tx_end_wait2();		//送信終了まち

PORTC.PODR.BIT.B0 = 1;	//LED1 ON
main_wait(10000000);	
PORTC.PODR.BIT.B0 = 0;	//LED1 OFF
main_wait(10000000);	



12



ウォッチ1	
📓 🧠 🐉 🖏 🗙 着	€記(N)▼ 🔤
ウォッチ式	値
🗊 SCI1.SSR	0×84
🗉 🔍 rx_data	""
🛯 adO	124 (0x007c)
👽 temp1	1567 (Ox061f)
🛯 stvolt	1557 (Ox0615)
🗏 🔍 tx_buffer	"ADO = 0.100V…
€ [0]	'A' (0x41)
😜 [1]	'D' (0x44)
🔍 [2]	'O' (Ox3O)
😜 [3]	''(0x20)
😜 [4]	'=' (0x3d)
😜 [5]	''(0x20)
😜 [6]	'O' (Ox3O)
😜 [7]	'.' (Ox2e)
😜 [8]	'1' (0x31)
😜 [9]	'0' (0x30)
😜 [10]	'O' (Ox3O)
😜 [11]	'V' (Ox56)
😜 [12]	''(0x20)
😜 [13]	'T' (0x54)
€ [14]	'E' (0x45)
😜 [15]	'M' (0x4d)
😜 [16]	'P' (0x50)
😜 [17]	''(0x20)
😜 [18]	'=' (0x3d)

以上が、ADデータの取り込み、USB出力、制御の簡単なプログラムになります。

}

2-4 sample4 割り込み

【 動作概要 】

sample4を動作させます。ここでは定周期割り込みについてサンプルを示します。sample3のループ時間を割込みを使って正確に1秒とし、データを出力しています。新たに経過秒と出力を示しています。

COM11:38400baud - Tera Term VT	_ 🗆 X	
ー ファイル(E) 毎年(E) 設定(C) フントロール(O) ウインボウ(MA) Aルゴ(山)		
フパイル(F) 編集(E) 設定(3) コンドロール(O) ウインドウ(W) ハルフ(H)	/	
ADU = U./UTV TEMP = 32.1/d STU_VULT = 1.25/ V Timer = 205		<u>^</u>
ADU = 0.695V TEMP = 32.37d STD_VOLT = 1.254 V Timer = 206		
ADU = 0.5977 IEMP = 32.37d SID_VULI = 1.254 V LIMER = 207	-	
ADU = 0.704V IEMP = 32.37d SID_VULI = 1.206 V IIMER = 208 ADO = 0.606V TEMP = 32.37d STD_VOLT = 1.254 V Timer = 200		
ADU - ULOMOV TEMP - 32.370 STD_VULT - 1.254 V TIMEE - 200 ADO - 0.007V TEMP - 22 Fed STD VOLT - 1.255 V Timer - 210		
ADU - U.6977 IEMP - 32.000 SID_VULI - 1.200 V IIMER - 210 - ADO - 0.60777 TEMP - 00 764 STD VOLT - 1.054 V Timer - 211 -		
ADU - 0.0377 IENNE - 32.700 SID_YULI - 1.234 Y IIMEE - 211 ADD - 0.70217 TENNE - 22.564 STD VOLT - 1.255 V Timer - 212		
ADO - 0.7037 TEMP - 32.300 STD_YOLT - 1.233 7 TIMEE - 212 ADO - 0.607V TEMP - 22.17d STD VOLT - 1.255 V Timer - 212		
ADO = 0.60677 TEMP = 32.17d OTD_70ET = 1.255 V Timer = 210 ADO = 0.696V TEMP = 32.37d STD VOLT = 1.255 V Timer = 214		
ADO = 0.607V TEMP = 32.56d STD VOLT = 1.255 V Timer = 215		
ADD = 0.704V TEMP = 32.56d STD_VOLT = 1.255 V Timer = 216		
ADD = 0.697V TEMP = 32.37d STD VOLT = 1.255 V Timer = 217		
ADD = 0.703V TEMP = 32.37d STD VOLT = 1.256 V Timer = 218		
ADO = 0.704V TEMP = 32.17d STD VOLT = 1.255 V Timer = 219		
ADO = 0.703V TEMP = 32.37d STD VOLT = 1.256 V Timer = 220		
ADO = 0.703V TEMP = 32.37d STD_VOLT = 1.253 V Timer = 221		
ADO = 0.704V TEMP = 32.17d STD_VOLT = 1.256 V Timer = 222		
ADO = 0.698V TEMP = 32.56d STD_VOLT = 1.255 V Timer = 223		
ADO = 0.697V TEMP = 32.37d STD_VOLT = 1.255 V Timer = 224		
ADO = 0.697V TEMP = 32.37d STD_VOLT = 1.254 V Timer = 225		
ADO = 0.704V TEMP = 32.37d STD_VOLT = 1.255 V Timer = 226		-
AD0 = 0.697V TEMP = 32.56d STD_VOLT = 1.254 V Timer = 227		=
		×

オシロスコープがあればPCO CN6 20番を観測すると、以下のような幅100nsec、1ms ec周期(1.00005KHz)の波形が観測できます。



【周辺機能の説明】

sample3で使用している周辺機能に加えて、8ビットタイマ TMR0を使って、1msec 定 周期割り込みを実現させています。 🕥 sample4 - RX E1(Serial) - CS+ for CC - [プロジェクト・ツリー] ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウインドウ(W) ヘルプ(A) - 100% 🗑 😽 DefaultBuild n D - 15) 📵 🕟 🕞 🐂 🔞 🖘 🗊 プロジェクト・ツリー 🕈 🗙 🐔 逆アセンブル1 🖁 វី៨/ក្រៃវ 🧃 r_cg_tmr_user.c 🧃 r_cg_main.c 🧃 cg_tmr.c 🥑 r_cg_sci_user.c 羄 周辺 2 🕜 🙎 🛛 🗃 🐻 コードを生成する | 🔬 🗋 🚜 🛫 💓 🖓 🛔 🍰 🗯 🎲 🚳 🖏 0 0. 😃 😃 🙂 🖃 🖉 🎜 🎜 -般設家 IMRO TMR1 TMR2 TMR3 🔚 コード生成 (設計ツール) カウンと設定 ★ 🔏 端子図 ロックソース POLK ✓ 60000 (kHz) ■ 题 周辺機能 外部クロック端子TMOI1 P12 🧳 クロック発生回路 ■ 💕 電圧検出回路 カウンタクリア コンペアマッチ によりクリア V ● クロック周波数精度測定回路 外部リセット端子TMRIO P20 🔋 消費電力低減機能 コンペアマッチAの値(TCORA) 1000 μs ▼ (実際の値: 1000) …💣 割り込みコントローラ ■ [●] バス ■ [●] DMAコントローラ = S12AD A/D変換開始要求 コンペアマッチBの値(TCORB) 20 μs (実際の値: 20) 🤎 データトランスファコントローラ ● イベントリンクコントローラ TMO0出力設定 □ TMOO出力許可 ■ 🗳 マルチファンクションタイマパルフユニット3 ~ TMOO端子 P22 * ポートアウトプットイネーブル コンペアマッチA時の出力レベル 変化しない V 🛓 🌳 汎用PWMタイマ コンペアマッチB時の出力レベル 変化しない 🖻 🧉 16ビットタイマパルス ニット 🖃 🏴 プログラマブルパルズジェネレータ 割り込み設定 ¯ ■
[●] 8ビットタイマ ▼ TCORAコンペアマッチ割り込みを許可(CMIAO) 優先順位レベル15 ~ - C TMR0 □ TCORBコンペアマッチ割り込みを許可(CMIBO) 優先順位 レベル15 TMR1 TMR2 □ TCNTオーバフロー割り込みを許可(OVIO) 優先順位レベル15 TMR3

カウンタクリアを「コンペアマッチAによりクリア」にします。割込み設定をセット

「コード生成」で r_cg_tmer.c r_cg_tmer_user.c 2つのプログラムが自動作成されます。

【 プログラム 】

省略

【 解説 】

省略

このように、割り込みで時間を作り、メインで使用すれば、極めて正確なタイマーが多数作成可能です。 定周期割り込みはその中に様々なプログラムを作成することも可能で、機能的なプログラム作成に不可欠 な知識、要素です。

 \widehat{O} sprintf(tx_buffer,"AD0 = %.3fV TEMP = %.2fd STD_VOLT = %.3f V Timer = %4d \Re ⁴/₁% \Re ⁴/₁%

サーミスタの電圧換算AD値、CPU内部温度、基準電圧に加えて、経過時間(秒)の4つの情報をUSB に出力しています。

ウォッチ1			д х
🔊 🥮 🐉 🖏 🗙	表記(N)▼│ 쪸		
ウォッチ式	値	型情報(バイト数)	アドレス メモ
🗊 SCI1.SSR	0x00	IOR(1)	0x0008a024
🖽 🔍 rx_data		unsigned char…	0×00000004
🔍 adO	498 (0x01f2)	uint16_t(2)	0x000000f6
🔍 tempî	1586 (0x0632)	uint16_t(2)	0x000000f8
🔍 stvolt	1551 (0x060f)	uint16_t(2)	0x000000fa
🖽 🔍 tx_buffer	"ADO = 0.401V…	unsigned char…	0x0000018
👽 fdata2	3.433057E+001	float (4)	0x0000050c
🔍 fdata1	4.013187E-001	float (4)	0×00000508

なお、各種変数、バッファの内容はプログラム実行中にウオッチ窓でリアルタイムに見ることが出来ます。

2ー5 sample5 PWM出力 【 動作 】

汎用PWMタイマを使って、PWM波形を作ります。GTIOCOA端子 P23 CN5 20番、GTIOC1A端子 P22 CN5 19番、GTIOC2A端子 P21 CN5 18番に出力されます。それぞれ、青、緑、赤のLEDがドライブされま す。波形はチェックピンTP17, 18, 19を観測したものです(詳細は回路図ご参照ください)



PWM駆動により、Red, Green, Blue=RGBの色の混合比が変わりますので、丸い白色の ドームを被せて見たりすると色が混色し、単色が変化するように見えるかもしれません。TVやスマホの フルカラー表示は、光の3原色 RGBの強度の違いで様々な色を見せています。

PWM波形は上図のように、周期が変わらず、設定値によってH、Lの幅の比率が変化します。この出力 でLEDやモーターをドライブすると明るさや速度を効率よく変えることが出来るので、現代では様々な 用途に使われています。

【周辺機能の説明】

省略

【 プログラム 】

省略

【 解説 】

省略

2-6 sample6 三角、対数、平方根関数 関数演算速度を検証

【 概要 】

| og、sin、√ 演算を行い、演算結果の確認とその速度を測定します。

【周辺機能の説明】

CS+側の設定はc-3↓をご参照ください。
 c-3: 三角関数math. hはインクルードもCS+の設定も必要

【 プログラム 】

; ①#include <math.h>

②double d1,d2,d3;
③short s1,s2,s3;

(4)#define PI 3.14159265

;

void main(void)

{

R_MAIN_UserInit();

/* Start user code. Do not edit comment generated here */

//重複省略

5 6	PORTC.PODR.BIT.B0 = 1; d1 = log10(10000);	//時間マーカーON
\overline{O}	PORTC.PODR.BIT.B0 = 0;	//時間マーカーOFF
8	d2 = sin((PI/180)*45);	
	PORTC.PODR.BIT.B0 = 1;	//時間マーカーON
9	d3 = sqrt(2);	
	PORTC.PODR.BIT.B0 = 0;	//時間マーカーOFF
10	s1 = d1;	
	s2 = d2;	
	s3 = d3;	
W	vhile (1U)	

{

}

【 解説 】

①#include <math.h> //sqr 等演算を行うのに必要 math.h有効と共にこの表記も必要。
 三角関数や、対数、平方根を使うためにはmath.hをインクルードする必要があります。
 ②double d1,d2,d3;
 演算結果をセーブするダブル(浮動小数点32ビット)データです。
 ③short s1,s2,s3;
 演算結果をキャストしてセーブするショート(16ビット)データです。
 ④#define PI 3.14159265

三角関数計算で角度を入力して数値を出すために使います。

5	PORTC.PODR.BIT.B0 = 1;	//時間マーカーON
6	$d1 = \log 10(10000);$	
\overline{O}	PORTC.PODR.BIT.B0 = 0;	//時間マーカーOFF

d 1 = 1 o g 1 0 (10000)を行うのですが、演算速度を測定するためにポートを使用しています。 d 1 は 4 になるはずです。

(8) $d2 = \sin((PI/180)*45);$

sin(45°)という意味です。d2=0.7071067、となるはずです。

⑨ d3 = sqrt(2);
 √ 2という意味です。d3 = 1.41421、、となるはずです。

(10)

s1 = d1;s2 = d2;

s3 = d3;

32ビット浮動小数点データを16ビット整数にキャストしています。それぞれ、4,0,1となるはずです。例えば演算結果をDAコンバータに出力する場合、浮動小数点のままでは設定できません。小数点以下何桁まで使用したいかに応じて、doubleデータを加工してからshortに移せば最大の精度、 有効数値を得ることが出来ます。

ウォッチ1				ą 🗙
🔳 🍭 🛃 🔧	表記(N) - 🔤			
ウォッチ式	値	型情報(バイト数)	アドレス メ	ŧ
€ tx_end_flg	'' (0x00)	uint8_t (1)	0x000002d	
€rx_flg	'' (0x00)	uint8_t (1)	0x000002c	
🖽 🔍 rx_data		unsigned char…	0x00000004	
🗊 SCI1.SCR	0x50	IOR(1)	0x0008a022	
🗊 SCI1.SSR	0x84	IOR(1)	0x0008a024	
😔 d1	4.000000E+000	float (4)	0x00000460	
😜 d2	7.071068E-001	float (4)	0x00000464	
😜 d3	1.414214E+000	float (4)	0x00000468	
🔍 s1	4 (0x0004)	short (2)	0x0000054	
🔍 s2	0 (0x0000)	short (2)	0x0000056	
S3	1 (0x0001)	short (2)	0x0000058	

演算結果はそれぞれ d 1、 d 2、 d 3に入ります。正しいですね。

演算速度ですが、 | og 10(10000)が約550nsec、sin(45°)が350nsec、√ 2が100nsec程度かかるようでした。



TBS 1064 - 17:41:37 2016/09/29

例えばRL78(32MHz)では約220 μ sec、sin(45°)が130 μ sec、 $\sqrt{2}$ が100 μ sec程度ですので、それぞれ400倍、371倍、1000倍も速いことになります。マイコンに必 要な能力が演算処理速度の場合、RXを使用するのが圧倒的に有利であることが分かります。RX21A がRX71M、RX630に比べて遅いのはFPU内蔵、非内蔵の差と思われます。

CPU クロック	log	sin	\checkmark		
RX71M 240MHz	550nsec	3 5 0 n s e c	100nsec		
RX630 100MHz	1. 8µsec	800 n s e c	1. 2µsec		
RX21A 50MHz	ЗЗµѕес	2. 5µsec	3µѕес		
RL78 32MHz	220 <i>µ</i> sec	130µsec	1000 <i>µ</i> sec		

CPU別演算速度例

2-7 sample7 D/Aにsin演算した正弦波を出力する

【 概要 】

RX71Mがもつ、1ch 12ビットD/Aにsin演算結果(最大±1)を0-3.3Vに変換し出 カします。正弦波オシレーターになります。サンプルでは207Hz~10KHzまで周波数を変化させ て、デジタルアンプを経てスピーカーU6に出力します。周波数変化を耳で音を聞くことが出来ます。音 量はVR2で調整出来ます。

波形は実行時の 上 TP4 下 TP22を観測。





省略

「フロクラム】 省略

【 解説 】

省略

格納された配列のデータを1個ずつ読み込んでD/Aに出力しています。この方式で最高82KHz程度の正弦波が作成できています。人間の可聴帯域程度であれば十分使用可能です。

2-8 sample8 液晶表示

【 概要 】

sample3、4 プログラムをベースに16文字×2行の液晶に表示させるプログラムを追加しました。簡単な液晶表示の例を示します。



【 プログラム 】

```
① lcd_disp_0();
```

【 解説 】

① lcd_disp_0();

液晶に表示させている関数ですが、内容は以下の通りです。 void lcd_disp_0(void)

{

```
lcd_cursor_1();
lcd_data_write(tx_buffer[0]);
lcd_data_write(tx_buffer[1]);
lcd_data_write(tx_buffer[2]);
lcd_data_write(tx_buffer[3]);
lcd_data_write(tx_buffer[4]);
lcd_data_write(tx_buffer[5]);
lcd_data_write(tx_buffer[6]);
lcd_data_write(tx_buffer[7]);
lcd_data_write(tx_buffer[8]);
lcd_data_write(tx_buffer[9]);
lcd_data_write(tx_buffer[10]);
lcd_data_write(tx_buffer[11]);
```

lcd_cursor_2();

lcd_data_write(tx_buffer[46]); lcd_data_write(tx_buffer[47]); lcd_data_write(tx_buffer[48]); lcd_data_write(tx_buffer[48]); lcd_data_write(tx_buffer[50]); lcd_data_write(tx_buffer[51]); lcd_data_write(tx_buffer[52]);

```
lcd_data_write(tx_buffer[53]);
lcd_data_write(tx_buffer[54]);
lcd_data_write(tx_buffer[55]);
lcd_data_write(tx_buffer[56]);
lcd_data_write(tx_buffer[57]);
```

}

単純に tx_buffer [n] にある文字を液晶に書き込んでいるだけです。

USBには以下の命令で

sprintf(tx_buffer,"AD0 = %.3fV TEMP = %.2fd STD_VOLT = %.3f V Timer

= %4d¥n¥r",fdata1,fdata2,fdata3,timer);

テラターム画面のように出力されているのですが、液晶は1行16文字文字しかないので

ウォッチ式	値 型情報(バイト数)
🖽 👽 rx_data	"" unsigned char…
🔍 ad0	713 (0x02c9) uint16_t(2)
🔍 temp1	1588 (0x0634) uint16_t(2)
🔍 stvolt	1552 (0x0610) uint16_t(2)
🗏 👻 tx_buffer	"ADO = 0.575V unsigned char
😜 [0]	'A' (Ox41) unsigned char(1)
😜 [1]	'D' (Ox44) unsigned char(1)
😜 [2]	'O' (Ox30) unsigned char(1)
😜 [3]	''(Ox2O) unsigned char(1)
😜 [4]	'=' (0x3d) unsigned char(1)
😜 [5]	''(Ox2O) unsigned char(1)
😜 [6]	'O' (Ox30) unsigned char(1)
😜 [7]	'.' (Ox2e) unsigned char(1)
😜 [8]	'5' (0x35) unsigned char(1)
😜 [9]	'7' (0x37) unsigned char(1)
😜 [10]	'5' (0x35) unsigned char(1)
😜 [11]	'V' (Ox56) unsigned char(1)
😜 [12]	''(Ox2O) unsigned char(1)
😜 [13]	'T' (0x54) unsigned char(1)
💊 [14]	'E' (0x45) unsigned char(1)
💊 [15]	'M' (0x4d) unsigned char(1)
😜 [16]	'P' (0x50) unsigned char(1)
🔋 🛛 😜 [17]	''(Ox2O) unsigned char(1)
😜 [18]	'=' (0x3d) unsigned char(1)

上の行はtx_buffer[0]から[11]までを表示しています。

下の行は t x _ b u f	f e r [46]から[57]までを表示しています。
ウォッチ式	値 型情報(バイト数)
😜 [37]	''(0x20) unsigned char(1)
🛛 [38]	'1' (0x31) unsigned char(1)
😜 [39]	'.' (Ox2e) unsigned char(1)
🐳 [40]	'2' (0x32) unsigned char(1)
😜 [41]	'5' (0x35) unsigned char(1)
😜 [42]	'1' (0x31) unsigned char(1)
😜 [43]	''(0x20) unsigned char(1)
😜 [44]	'V' (0x56) unsigned char(1)
😜 [45]	''(0x20) unsigned char(1)
💚 [46]	'T' (0x54) unsigned char(1)
😜 [47]	'i' (0x69) unsigned char(1)
😜 [48]	'm' (Ox6d) unsigned char(1)
😜 [49]	'e' (0x65) unsigned char(1)
😜 [50]	'r' (0x72) unsigned char(1)
😜 [51]	''(0x20) unsigned char(1)
😜 [52]	'=' (0x3d) unsigned char(1)
😜 [53]	''(0x20) unsigned char(1)
😜 [54]	'1' (0x31) 🗮 nsigned char(1)
😜 [55]	'4' (0x34) unsigned char(1)
😜 [56]	'2' (0x32) unsigned char(1)
😜 [57]	'2' (0x32) unsigned char(1)
😜 [58]	'' (OxOa) unsigned char(1)
😜 [59]	''(OxOd) unsigned charX()
😜 [60]	'' (0x00) unsigned char(1入
😜 [61]	'' (0x00) unsigned char(1) 🔪
	\backslash

なお、通信でも液晶表示でも扱う文字はASCIIコードで、1を表示したいときには0×31を設定します。1を設定しても1は表示されないことに注意してください。

2-9 sample9 ステージを動かす

【 概要 】

パルスモータ制御の基本を学びます。



使用しているモーターは2相、ステップ角 3.6°、リードピッチ1mmです。360/3.6=10 0パルスで1mm移動する計算になります。

ドライバーICは最大1/16ステップまで選択できるマイクロステップドライバで、1/16を選択した場合、1パルスで1.8/16=0.1125°、360/0.1125=1600パルス1mm移動 することになります。

※液晶の表示は応用 sample35を動作させた時のものです。

※ステージの回転ねじの部分にゴミが入ると回転がスムースに行かなくなりますの保存は必ず付属の箱、 またはゴミの無い環境でお願いします。 /



^{【プログラム】} 省略 ^{(解説}】 省略 _{(演習}】 省略

それぞれはそれぞれの会社の登録商標です。

フォース

⑦及びFORCE

®は弊社の登録商標です。(ソフトウエア、ハードウエア 電子計算機、及びその

の周辺装置)

1. 本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。

2. 本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。

3. 本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。

4. 本文章は許可なく転載、複製することを堅くお断りいたします。

お問い合わせ先:

〒350-1213 埼玉県日高市高萩1141-1
TEL 042(985)6982
FAX 042(985)6720
Homepage:http//beriver.co.jp
e-mail:info@beriver.co.jp
有限会社ビーリバーエレクトロニクス ©Beyond the river Inc. 20180205