

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル

ルネサス エレクトロニクス社 RX651/RX671(QFP-144 ピン)搭載 HSB シリーズマイコンボード向け I/O ボード

-本書を必ずよく読み、ご理解された上でご利用ください





注意事項…		1
安全上のご	注意	2
概要		4
開発環境…		5
サンプルプ	コグラムの動作説明	6
スマート・コ	ンフィグレータを使用した初期設定	10
1. サンプ	ルプログラム説明(V1 2)	23
11 J.	✓ ↓ ✓ ✓ ✓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
1.1. >-	ン) 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅 𝔅	23 24
112	ブザーの国法教変更(*2)	24
1.1.2.	○ ○ ○ □ 次気交叉(2) Δ/D 変換(*3)	20
1.1.3.	べし 交供(3) I ED8 PW/M デューティ変更(*1)	20 26
1.1.4.	CEDO I WW / エーノイ変史(+)	20
1.1.5.	7 ビノアンド こし 時間((3)	27
1 1 7	、1 , , , , , , , , , , , , , , , , , , ,	20
118	ステッピングモータ(右回転)(*8)	
1.1.0.		20
1.2. 司 1.2.1	ッとの周数 タイマ(CMT0)割り込み処理	30
1.2.2.	7 ヤグメント LED 表示処理(*1)	
1.2.3.	マトリックススイッチスキャン処理(*2)	32
1.2.1.	SW8(IRQ15)割り込み処理	34
1.2.2.	SW9(IRQ13)割り込み処理	34
1.2.3.	SW10(IRQ4)割り込み処理	35
بت 13		36
1.3.1.	io_board.c に含まれる関数	36
1.3.2.	 lcd_1602.c に含まれる関数	40
1.3.3.	sci.c に含まれる関数	44
1.4. グ	ローバル変数	49
1.4.1.	io_board.c で定義している変数	49
1.4.2.	sci.c で定義している変数	53
1.5. 定	義値	53
1.5.1.	io_board.h で定義している値	53
1.5.1.	lcd_1602_RX65.h で定義している値	53
1.5.2.	sci.h で定義している値	54
2. マイコン	/使用機能	55
21 庙	田楼能一覧	55
2.1. 区		00

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社

	2.2.	I/O ボードの各機構に割り当てられているマイコン機能	55
	2.3.	使用割り込み一覧	56
	2.4.	I/O ポート初期設定	56
	2.5.	SCI5(USB-SERIAL)を使用したキャラクタの送受信	57
3.	マイ	コンボードへのプログラム書き込み方法	59
	3.1.	デバッガを使用した書き込み	59
	3.2.	USB ケーブルを使用した書き込み	63
付	録		69
	取扱該	说明書改定記録	69
	お問合	6世窓口	69



1

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

- 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
- 2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
- 3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複 写・複製・転載はできません。
- 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては 製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更 することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
- 5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
- 6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

- 1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
- 2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

- 1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
- 2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
- 3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
- 4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的・用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず 一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用 には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。 ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊 社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に 一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。 保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転 売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。 本製品を使った二次製品の保証は致し兼ねます。

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上で お読み下さい。

表記の意味



取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性が ある事が想定される

取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが 可能性がある事が想定される

絵記号の意味

0	一般指示 使用者に対して指示に基づく行為を 強制するものを示します	\bigcirc	一般禁止 一般的な禁止事項を示します
	電源プラグを抜く 使用者に対して電源プラグをコンセ ントから抜くように指示します		一般注意 一般的な注意を示しています













概要

本書は、RX651 マイコンを搭載したマイコンボード(HSBRX651F144A)、及び RX671 マイコンを搭載したマイコンボ ード(HSBRX671F144)向けの、I/O ボード(スイッチや LED, LCD 等が搭載されており、マイコンのプログラムの動作 をモニタするためのボード)である、HSBRX65-IO-BOARD を動作させるサンプルプログラムの説明を行っている資 料です。

サンプルプログラムは、あくまで I/O ボードを動作させる一例です。サンプルプログラムで使用しているものとは、別なタイマ機能を用いたり、別な手法を用いてプログラミングを行ってみる事を推奨致します。





開発環境

本サンプルプログラムは、CS+向けのプロジェクトとなっていますので、予め

CS+forCC(Ver8.07 以降) [ルネサスエレクトロニクス製]

をダウンロードし、インストールを行ってください。本サンプルプログラムをビルドする目的であれば、無償版で十分 です。(無償版のリンクサイズ制約が問題となる様な大規模なプログラムではありません)

本書では、CS+と連携して使用できる、スマート・コンフィグレータを使用しています。スマート・コンフィグレータは、 自動的にコードを生成してくれる便利なツールですので、「RX スマート・コンフィグレータ(Ver2.12 以降)」も合わせて ダウンロード、インストールを行ってください。

作成したプログラムをデバッグする際は、デバッガ

E1

E2

E2Lite

E20

(いずれか)があると便利です。デバッガは、マイコンボードへのプログラムの書き込みと、デバッグ、両方の用途で 使用可能です。

なお、デバッガをお持ちでなくても、マイコンボードへのプログラムの書き込みは、付属の USB ケーブル(USB AminiB)で行う事ができます。USB ケーブルでプログラムの書き込みを行う際は、RenesasFlashProgrammer(Ver3 以降)が必要です。(基本的には、CS+のインストール時に、同時にインストールされます)

本書では、CS+以外の開発環境を使用する場合の説明はありませんが、e2studio やサードパーティ製の開発環境を使用する事も可能です。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株



サンプルプログラムの動作説明

本サンプルプログラムをマイコンボードに書き込んだ場合の動作に関して説明します。



(1)SW-LED 連動

ボード上のプッシュスイッチ SW0-7 を押すと、押している間、LED0-7 が点灯します。 (これは、SW0-7 と LED0-7 がボード上で接続されているわけではなく、マイコンボードで実行されているプログラ ムが、SW の読み取りと、LED の制御を行っています。)

(2)マトリックススイッチの読み取り

マトリックススイッチ SW11-26 を押すと、LCD 画面に押しているスイッチが表示されます。

※LCD 画面が薄い、または濃く塗りつぶされている場合は、ボード上の R11(水色の中に黄色の十字がある部品) を、精密ドライバ等で調整してください

※複数のキーを同時に押すと、「SW26*」の様に、末尾にアスタリスク(*)が表示されます(SWxxのxxは、押されているキーの中で番号の一番大きなものが表示されます)





(3)ブザー鳴動

※JP2を2-3ショート(上側の2ピンにショートジャンパを挿す)にしてください

SW8を押すと、ブザーが鳴ります。SW8はトグル動作になっており、もう一度押すとブザーは止まります。 ブザーが鳴っている間、SW0-7を押すと、ブザーの音程が変わります。

(4)LED 点滅

※JP1を1-2ショート(下側の2ピンにショートジャンパを挿す)にしてください

SW9を押すと、LED8 が点滅します。(1 秒周期)もう一度 SW9を押すと、LED8 は消灯します(SW9 はトグル動作です)。

LED8 が点滅している間、R29 を回すと、点灯時間が変わります。

(5)7 セグメント LED

7 セグメント LED は、R29 の角度に応じた数値が表示されます(0~3FF)。 SW10 を押す度に、「表示数値が固定(HOLD)」「消灯(OFF)」「R29 に応じた値(ON)」の様に切り替わります。

(6)LCD

LCD 画面には、1 行目には

HSBRX65-IO-BOARD

の表示が出力されます。2 行目は、マトリックススイッチを押している間「SW11-26」の表示、及び SW8-10 を押した際「IRQ15, IRQ13, IRQ4」の表示(IRQ 表示は、他の情報が出力されるまで、表示は維持)が出力されます。

(7)USB-Serial

J7(USB-miniB)をPCと接続。PC 側で、

115,200bps, 8bit, パリティなし, ストップビット 1bit, フロー制御なし

で端末(Teraterm 等)を開くと、端末にメッセージが表示されます。

端末から、ブザーの鳴動を ON/OFF したり、7 セグメント LED のデモを実行したりする事が可能です。

※USB-Serial を使用する際は、PC 側に Prolific の USB-Serial のドライバがインストールされている必要があり ます(ドライバのインストールに関しては、取扱説明書参照)

※仮想 COM ポート番号(COM4 等)は、PC 環境によって異なりますので、USB ケーブルの挿抜で、PC 上に出 現する COM ポート番号を選んで、端末を接続してください

※Ver2.0 以降のボードでは USB-Serial 変換使用時、JP9 をショートに設定してください(~Ver1.1 ボードではジャンパ設定は不要です)

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



(8)ステッピングモータ

※JP3 をショートに設定してください

J5 1-6 番ピンに、ステッピングモータ(推奨モータ: PF25-48C1[日本パルスモータ製])を、接続し端末からステッ ピングモータを制御するコマンド("s" or "r")を入力すると、ステッピングモータを右回転または左回転させる事ができ ます。

-USB-Serialに接続した端末からのコマンド-

HSBRX671F144 ボード向けのサンプルプログラムでは この部分の表示が HSBRX671F144 になります

ボードを起動すると、端末には下記のメッセージが表示されます。

```
HSBRX65-IO-BOARD[since Ver1.1](HSBRX651F144A) Sample program [Rev1.2].
Copyright (C) 2019-2022 HokutoDenshi. All Rights Reserved.
TEST MENU:
    b : buzzer(P17) [ON/OFF toggle]
    l : LED8(P15) [ON/OFF toggle]
    m : key matrix
    s : stepping motor[CCW]
    r : stepping motor[CCW]
    7 : 7seg Demo
    h : help
>
```

・b コマンド(SW8 と等価)

キーボードから、b(1 文字、エンター不要)を入力すると、ブザー(P17)を鳴らします。もう一度 b コマンドを入力すると、ブザーは止まります。

```
>b
buzzer (P17)>
->buzzer ON
>b
buzzer (P17)>
->buzzer OFF
```

・1 コマンド(SW9 と等価)

キーボードから、Iコマンドを入力すると、LED8 が点滅します。もう一度 Iコマンドを入力すると、LED8 は OFF と

なります。

```
>1
LED8 (P15)>
->LED8 blink ON
>1
LED8 (P15)>
->LED8 OFF
```





・m コマンド

キーボードから、mコマンドを入力すると、マトリックススイッチの状態を表示します。何かキーを押すと停止します。

```
>m
-matrix sw (press any key to exit)-
SW11 SW12 SW13 SW14 SW15 SW16 SW17 SW18 SW19 SW20 SW21 SW22 SW23 SW24 SW25 SW26
0 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0
```

押している、キーが o で表示されます。(複数キーの同時押しも認識します) ※m コマンド実行中は、LCD 画面には、SW11-26 の情報が表示されません

・sコマンド

>s

キーボードから、sコマンドを入力すると、ステッピングモータ端子(J5)から、ステッピングモータを左回りに回転させる信号が出力されます。何かキーを押すと停止します。

-stepping motor drive [CCW] (press any key to exit)-

・rコマンド

キーボードから、rコマンドを入力すると、ステッピングモータ端子(J5)から、ステッピングモータを右回りに回転させる信号が出力されます。何かキーを押すと停止します。

```
>r
-stepping motor drive [CW] (press any key to exit)-
```

・7 コマンド

キーボードから、7 コマンドを入力すると、7 セグメント LED のデモ(全表示から、1111, 2222, …FFFF まで順次 表示)が実行されます(デモの後は、7 セグメント LED が ON となります)。

>7 7seg LED ->7seg led DEMO -> ON

・hコマンド

キーボードから、hコマンドを入力すると、ヘルプの画面が出力されます。

```
>h
b : buzzer(P17) [ON/OFF toggle]
l : LED8(P15) [ON/OFF toggle]
m : key matrix
s : stepping motor[CCW]
r : stepping motor[CW]
7 : 7seg Demo
h : help
```



9



スマート・コンフィグレータを使用した初期設定

CS+を使用して、本ボード向けのプログラムを作成する際は、スマート・コンフィグレータを使用すると、クロックの初 期設定や割り込みの設定等のコードを自動生成してくれますので、手間の掛かる初期設定のコードを書く手間が省け ます。

CS+を起動し、

ファイルー新規作成ー新しいプロジェクトを作成



使用するマイクロコントローラ: R5F5651EDxFB …HSBRX651F144Aの場合 R5F5671EHxFB …HSBRX671F144の場合

プロジェクト名: 任意の名称

「作成」ボタンを押す。





🚳 RX65-IO - CS+ for CC - [プロパティ]								
ファイル(E) 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(I) ウインドウ(W) ヘルプ(H)								
	0 ℃ 🏭 🚆 🏯	■ 100%■	🖬 🚮 DefaultBuild	•				
중 중 🖉 😵 😵 🗆 후 다 이 호 🦉	┛ ソリューション一覧(<u>S</u>)							
🐺 לבטֿדלאיש 🕂 🗰	🚰 ว่อเกรา							
2 2 2 2 2 2 2 2 2 2 2 2 2 2	 ▶ RX65-10 のプロパティ ▲ ファイル > ファイル名 絶対パス ▶ ライセンス ▶ 記録 							

「スマート・コンフィグレータ(設計ツール)」をダブルクリックで起動。

🚺 新規スマート・コンフィグレータファ	()L			
スマート・コンフィグレータ詞	定			
RTOSの種類とバージョンを選択				
RTOS: None				•
RTOSバージョン:				·
				RTOSバージョンの管理…
	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

RTOS は None を選択して「終了」を押す。



11

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



ファイル ウィンドウ ヘルプ								
e Kxos-io.scig × 概説								
							コードの生成レポートの生成	
▼ 機能概要								
VMIRE 構築 成型 成型 成型 成型 成型 成型 成型 成型 成型 人力 成型 人力 反 Application Code Software Components Drivers Device Drivers Device Drivers Device Drivers MCU Hardware							Renesas	
▼ 現在の設定状態								
使用しているボード/デバイス: R5F5651EDxFB (R	ROM size: 2MB, RAM si:	e: 640KB, Pin count:	144)					
生成先ロケーション (PROJECT_LOC¥): src¥smc	_gen		編集					
使用しているコンポーネント:								
コンポーネント	バージョン	設定						
 Board Support Packages. (r_bsp) 	7.00	r_bsp(使用中)						
概要 ボード クロック システム コンポーネント 対	端子 割り込み						•	▶ 凡例
■ コンソール			et 🗉 🔻 📑	+	🌡 コンフィグレーションチ	エック		7:
現在、表示するコンソールがありません。				0	項目			
					記述/説明		*	型
					•		п	- F

「ボード」タブを選択。

■ スマート・コンフィグレータ		
ファイル ウィンドウ ヘルプ		
		😰 🛐
👹 RX65-IO.scfg 😫	- 8	🛃 MC 🛛 🗖 🗖
デバイス選択	5	
テルノフ巡辺		🔚 🔺 🄎 »
ポード: カスタムユーザボード ▼		
デバイス: R5F5651EDxFB		
		-₹enesas
概要 ボード クロック コンボーネント 端子 割り込み		▶ 凡例
🖳 א-עעב 🛛 🗎 💀 🔄 ד 🖻 🕄	□ンフィグレーションチェック 🛛	⊉ ▽ □ □
スマート・コンフィグレータ出力	0 項目	
M05000001: 端子 22 (こEXTAL の機能が割り当てられています M05000001: 端子 20 (こXTAL の機能が割り当てられています	記述/説明	タイプ
	4	
		1

右上の「読み込み」ボタンを押す。

サンプルプロジェクトフォルダ(HSBRX65-IO-BOARD_SAMPLE/HSBRX65-IO-BOARD_RX671)内の、.bdf ファイ ルを指定して読み込みます

HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子

12



読み込むファイル:

HSBRX65-IO-BOARD_SAMPLE¥HSBRX651F144A_IO-BOARD_V1.00.bdf …HSBRX651F144Aの場合 HSBRX65-IO-BOARD_SAMPLE_RX671¥HSBRX671F144_IO-BOARD_V1.00.bdf …HSBRX671F144の場合

※Vの後のバージョン番号は変わる可能性があります

【 ズマート・コンフィグレータ	
ファイル ウィンドウ ヘルプ	
🌼 *RX65-IO.scfg 🛛	Mc X
デバイス選択	
デバイス選択	
ボード: カスタムユーザボード ▼ デバイス: R5F5651EDxFB	
	·?ENCES/OS
概要 ポード クロック コンポーネント 端子 割り込み	· · · · · · · · · · · · · · · · · · ·
スマート・コンフィグレータ出力	0 項目
M05000001: 端子 22 (こ EXTAL の機能が書的当てられています M05000001: 端子 20 (こ XTAL の機能が書的当てられています M02000002: 設定をインボート: <u>C:\Users\win64-5\Documents\cubesuite\HSBRX65-IO-BOARD SAMPLE\HSE</u>	記述/說明 ^ タイプ
· · · · · · · · · · · · · · · · · · ·	
4	

ボード欄、「HSBRX651F144A_IO-BOARD_V1.00」を選択。…HSBRX651F144Aの場合 「HSBRX671F144_IO-BOARD_V1.00」を選択。…HSBRX671F144の場合 (Vの後のバージョン番号は変わる可能性があります)

確認画面がでますので、「続ける」(もしくは「保存して続ける」)を押してください。



13

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



KS.	スマート・コ	ンフィグレータ							
7:	ァイル ウィ	ンドウ ヘルプ							
: 🗗									: 🖻 I 🐼
583	*PX65_IO_cc	fa ∞							
252	-KA03-10.50	ig a						U	
端	子設定							🔁 🖆	
ýä	子番号							H B 35.7	📕 🔺 🔎 »
	フィルタ入力						すべて	•	
	端子番号	端子名	デフォルト機能	機能	方向	備考	コメント	•	
	112	P64/WE#/D3/CS4#	P64	設定されてい	なし		Matrix-SW-ROW-1		
	113	P63/CAS#/D2/CS3#	P63	設定されてい	なし		Matrix-SW-COL-4		
	114	P62/RAS#/D1/CS2#	P62	設定されてい	なし		Matrix-SW-COL-3		
	115	P61/SDCS#/D0/CS1#	P61	設定されてい	なし		Matrix-SW-COL-2		
	116	VSS		VSS	-	読み取り専用			
	117	P60/CS0#	P60	設定されてい	なし		Matrix-SW-COL-1		
	118	VCC		VCC	-	読み取り専用			Benerat
	119	PD7/D7/MTIC5U/POE0#/SSLC3-A/QMI-B/QIO1	PD7	設定されてい	なし		SW7		-CEINES/35
	120	PD6/D6/MTIC5V/MTIOC8A/POE4#/SSLC2-A/QM	PD6	設定されてい	なし		SW6		Not Control of Control
	121	PD5/D5/MTIC5W/MTIOC8C/POE10#/SSLC1-A/	PD5	設定されてい	なし		SW5		
	122	PD4/D4/MTIOC8B/POE11#/SSLC0-A/QSSL-B/S	PD4	設定されてい	なし		SW4		1
	123	PD3/D3/MTIOC8D/TOC2/POE8#/RSPCKC-A/QIO	PD3	設定されてい	なし		SW3		
	124	PD2/D2/MTIOC4D/TIC2/MISOC-A/CRX0/QIO2-B	PD2	設定されてい	なし		SW2		
	125	PD1/D1/MTIOC4B/POE0#/MOSIC-A/CTX0/LCD	PD1	設定されてい	なし		SW1	E	
	126	PD0/D0/POE4#/LCD_EXTCLK-B/IRQ0/AN108	PD0	設定されてい	なし		SW0		
	127	P93/A19/POE0#/CTS7#/RTS7#/SS7#/AN117	P93	設定されてい	なし		LCD-DB7		
	128	P92/A18/POE4#/RXD7/SMISO7/SSCL7/AN116	P92	設定されてい	なし		LCD-DB6	-	
端	子機能	業子番号							
根理	要 ボード ク	ロック コンポーネント 端子 割り込み							▶ 月.(利
)				
	コンソール 8	×	🖳 🔐 🕅 🛛	🛃 🖳 🔻 📑 🔻		3 コンフィグレーションラ 	チェック 🛛		
지	マート・コンフ	フィグレータ出力				0項目			
MO	8000005: Se 8000016: Pi	etting sub-clock source to 32.768 kHz			^	記述/説明			タイプ
MØ	5000016: Pi	in 22 is assigned to EXTAL			-				
					-				
•		III			+	•	III		

端子タブー端子番号タブ

を選択すると、コメントの欄に、どの端子がどの信号に接続されているかのコメントがありますので、端子機能の割り 当てを行った際等、確認の目安としてください。





【 スマート・コンフィグレータ			
ファイル ウィンドウ ヘルプ			
			🖻 📓
∰ *RX65-IO.scfg ⊠			MCU 🛛
クロック設定			
	×1/4 -	60.0 (MHz)	-
	SCKCR (ICLK[3:0])	システムクロック (ICLK)	
発掘源: 発振子 - ×	► x1/2 ▼	120.0 (MHz)	120MHz
周波数: 24 (MHz)	SCKCR (PCLKA[3:0])	周辺モジュールクロックA (PCLKA)	
	×1/2 •	120.0 (MHz)	
9980 (us) 実際の値: 10000.000 us	SCKCR (PCLKB[3:0])	周辺モジュールクロックB (PCLKB)	COM 11-
	×1/4 ×	60.0 (MHz)	60IVIHZ
	SCKCR (PCLKC[3:0])	周辺モジュールクロックC (PCLKC)	
	×1/4 -	60.0 (MHz)	₹ENESAS
▼ 5750555 ■ 第次数: 22.758 (kHz)	SCKCR (PCLKD[3:0])	周辺モジュールクロックD (PCLKD)	1800 NY TALANA
	• ×1/4 •	60.0 (MH2)	
Oscillator drive capacity: 標準CL 👻 -	SCKCR (BCK[3:0])	外部バスクロック選択 (BCLK)	
発振安定時間:		120.0 (Mi12)	
2000 (ms) 実際の値: 2047.939 ms		LKDIV)外部バスクロック端子(BCLK端子)	
		SDRAM9099 (SDCLK)	
носола»	SCKCR2 (UCK[3:0])		
	×1/5 ×	48.0 (MHz)	-
概要 ボード クロック コンポーネント 端子 割り込み			▶ 凡例
ארעלב 🖳 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖬 ארעלב 🖬	□ □ 🔝 コンフィグレーションチェック 🛙		* ▽ □ □
スマート・コンフィグレータ出力	0 項目	*	
M03000005: Setting sub-clock source to 32.768 kHz M05000016: Pin 20 is assigned to XTAL	▲ 記述/説明		タイプ
M05000016: Pin 22 is assigned to EXTAL M05000001: 端子 18 に XCOUT の機能が実的当てられています	E		
M05000001: 端子 17 に XCIN の機能が書的当てられています	•		
		V RT	CSCLK
		32.76	8 (kHz)
次に「クロック」々づを問き			
ベニックロフノコアノで同じ、	※★サンプ	ルズけキ体田ですが	ミニマルタイトクロック
	本牛リノノ	ルビは不使用じりり	······································

システムクロック:120MHz

周辺モジュールクロック:60MHz

になっている事を確認してください。(基本的には初期設定のまま変更不要です)

必要に応じて、以下設定してください。

サブクロック(チェックを入れる)

※マイコンボードにはサブクロックが実装されています、プログラムで RTC の機能を使用しないときはチェックは必要ありません

SCKCR2(チェックを入れる)

※マイコンボードには USB 関連の回路が実装されています、プログラムでマイコンボード側の USB(ホスト、ファンク ション)の機能を使用しないときはチェックは必要ありません

(USB クロック使用時は、48MHz になるよう設定してください)



(時計機能)を使用する際は RTCSCLK にチェックを

入れてください

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 材



スマート・コンフィグレータ	
ファイル ウィンドウ ヘルプ	
🏶 *RX65-IO.scfg 🔀	X M 2
クロック設定	
	ロードの生成
VCC: 3.3 (V) PLID路 分周比: X1 ・ ig(#)・	SCKCR (FCLK[3:0]) x1/4 SCKCR (ICLK[3:0]) システムクロック (ICLK)
ボルジン 24 「MHz) 発展安定時間: 9980 (us) 実際の値: 10000.000 us	x1/2 ・ 120.0 (MHz) SCKCR (PCLKA[3:0]) 用辺モジュールクロックA (PCLKA) x1/2 ・ 120.0 (MHz) SCKCR (PCLKB[3:0]) 用辺モジュールクロックA (PCLKB) x1/4 - 60.0 (MHz)
▼ サブクロック 周波数: 32.768 (kHz) Oscillator drive capacity: 標準CL ▼	SCRCR (PCLRC[3:0]) 周辺モジュールクロック (PCLRC) ×1/4 60.0 (MHz) SCRCR (PCLKD[3:0]) 周辺モジュールクロック (PCLRD) ×1/4 60.0 (MHz) SCRCR (BCK[3:0]) 外部パスクロック 遠沢 (BCLK) ×1/2 120.0 (MHz)
概要 ボード クロック コンボーネント 端子 割り込み	▶ 凡例
עכב 🛛 🚽 🖂 🚽 🖓 🗸 🖓 🖓	🔝 コンフィグレーションチェック 🛛 🌐 🍄 🖓 🗖
スマート・コンフィグレータ出力 M03000005: Setting sub-clock source to 32.768 kHz	0 項目 記述/説明 クイブ
M05000016: Pin 20 is assigned to XTAL M05000016: Pin 22 is assigned to EXTAL	
۲	III →

一通り設定が終わったら、右上の「コードの生成」ボタンを押してください。

10 セク	ション設定 ×
?	プロジェクトの現在のセクション設定は、スマート・コンフィグレータに対応していません。 セクションの設定を変更しますか。?
	現在のセクション設定: B_1,R_1,B_2,R_2,B,R,SU,SI/4,PResetPRG/FFE00000,C_1,C_2,C,C\$DSEC,C\$BSEC,C\$INIT,C\$VTBL,C\$VECT,D_1,D_2,D,P,PIntPRG,W_1,W_2,W,L /FFE00100,EXCEPTVECT/FFFFF80,RESETVECT/FFFFFFC 変更後のセクション設定: SU,SI,B_1,R_1,B_2,R_2,B,R/0x00000004,C_1,C_2,C,C\$*,D*,W*,L,P*/0xFFE00000,EXCEPTVECT/0xFFFFFF80,RESETVECT/0xFFFFFFC
	変更する場合は[はい]を、 変更しない場合は[いいえ]を、 コード生成をキャンセルする場合は[キャンセル]をクリックしてください。
 	後、このメッセージを表示しない (はい(Y) いいえ(N) キャンセル

セクションの変更のメッセージが表示された際は、「はい」を選択してください。







上記枠内が、スマート・コンフィグレータが生成したソースコードです。(この時点では、クロックの設定程度しか入っ ていませんが、スマート・コンフィグレータでコンポーネントを追加していくと、この部分が増えていきます)



17

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



[参考]

🚳 HSE	BRX65-IO-BOARD_SAMPLE - CS+ for CC -	[プロジェ	:クト・ツリー]		
ファイ	ル(<u>E)</u> 編集(<u>E</u>) 表示(<u>V</u>) プロジェクト(<u>P</u>)	ビルド(<u>B</u>) デバッグ(<u>D</u>) ツール(<u>T</u>) ウインドウ(<u>W</u>) へ	レプ(<u>H</u>)	
- 🙉 Z	スタート(5) 🛃 🔒 🎒 🐰 🐚 🚳 🖌	0 CH A	🖁 🛱 🗂 🔽 🕇	🔐 ன DefaultBuild	- 🔨 🐻 [
- 💎 (7 🖉 🖗 😐 🖓 🗣 🔍 🗗 🕻	בעע	ーション一覧(<u>S</u>)		
70	ייער איזעד איזעד איזעד איזעד איזעד איזעד איזעד איזעד איזעד איז איזעד איז איזעד איז איזעד איזעד איזעד איזער איז איז איז איזעד איז איזעד איז איזעד איזעד איזעד איזעד איזעד איזעד איזעד איזעד איז איזעד איז איזעד איזעד איזעד איז	וסל 🚰	ペティ 🥤 HSBRX65-IO-BOARD_SAMPLE: 🏼 🏹 sci	c 🔄 Config_SCI5_user.c	🔮 Config_SCI5h 🛛 🦉 C
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		50   B	1   => ~ <b>ヽ</b>   カラム・		
Ĥ∰ ∰ ◀ ◀ ॡ ⋽ ⊥ · · · ·	BXX65-IO-BOARD SAMPLE (フロシエクト) RSF5651EDxFB (マイクロコントローラ) スマート・コンフィグレータ (設計ツール) CC-RX (ビルド・ツール) アイル プログラム解析 (解析ツール) ファイル HSBRX65-IO-BOARD SAMPLE.c Smart Configurator 	1           2           4           5           7           8           10           11           23           4           56           7           8           10           112           13           14           15           16           17           201           223           223           223           225	<pre>/#************************************</pre>	****	**************************************
	B II r_pincfg I sci I lcd_1602	27 28 29	#ifdef _cplusplus  //#include <ios>  //_SINT ios_base::Init::init_cnt;</ios>	// Remove the commen // Remove the commen	t when you use ios t when you use ios

スマート・コンフィグレータでコンポーネントを追加すると、上記の枠内にソースコードが追加されます。

<u>ड</u> ि रू	/-ト・コ	コンフィグレータ							
ファイ	ルウィ	ンドウ ヘルプ							
: 🖻 🏲	۵ (								: 📄 I 🐼
253 H	SBKX03-	10-BOARD_SAMPLEISUY &						U	
端子	設定							🔁 🖆	
端子都	畤							- <b>1 1 1 1 1 1 1 1 1 1</b>	
71	ルタ入力	1					すべて	•	
端	子番号	端子名	デフォルト機能	機能	方向	備考	 コメント	*	
	65	P80/TRDATA0/EDREQ0/MTIOC3B/PO26/ET0_TX		設定されてい	なし				
	66	PC4/A20/CS3#/MTIOC3D/MTCLKC/TMCI1/PO25		設定されてい	たし,				
	67	PC3/A19/MTIOC4D/TCLKB/PO24/ET0_TX_ER/T		TXD5	0		SCI5-TX		
	68	P77/TRDATA7/CS7#/PO23/ET0_RX_ER/RMII0_R		P77	IO		7SEG-LED-COM-4		
	69	P76/TRDATA6/CS6#/PO22/ET0_RX_CLK/REF50		P76	IO		7SEG-LED-COM-3		
	70	PC2/A18/MTIOC4B/TCLKA/PO21/ET0_RX_DV/R		RXD5	I		SCI5-RX	E	- RUNNEAS
	71	P75/TRSYNC1/CS5#/PO20/ET0_ERXD0/RMII0		P75	IO		7SEG-LED-COM-2		· · · · · · · ·
	72	P74/TRDATA5/A20/CS4#/P019/ET0_ERXD1/RM		P74	IO		7SEG-LED-COM-1		
	73	PC1/A17/MTIOC3A/TCLKD/PO18/ET0_ERXD2/S		設定されてい	なし				
	74	VCC		VCC	-	読み取り専用			
	75	PC0/A16/MTIOC3C/TCLKC/PO17/ET0_ERXD3/C		設定されてい	なし				
	76	VSS		VSS	-	読み取り専用			
	77	P73/TRDATA4/CS3#/PO16/ET0_WOL		設定されてい	なし			•	
端子模	制能	端子番号							
概要 7	ボード く	フロック コンポーネント 端子 割り込み							▶ 凡例
	シソール	8			ことこことの	パイグレーションチェック	8		
現在、	表示する:	コンソールがありません。			0項目				
					記述/説明	Ą	*		タイブ
					•		III		Þ

また、端子の「機能」「方向」の欄に選択している機能が反映されます。



HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子



ースマート・コンフィグレータによるブザー向けタイマ TMR1 の設定に関してー

スマート・コンフィグレータを使用して、ブザーを鳴らすためのタイマ(TMR1)を設定する例を示します。

ファイル ウィンドウ ヘルプ	
	😰   📓
∰ *RX65-IO.scfg ⊠	- <b>M</b> M M - <b>D</b>
ソフトウェアコンポーネント設定	
コンボーネント 🎝 🕞 🕀 🌲 🗸 設定	
<u> フィルタ入力 コンポーネントの追加 ・ ② ジェネリック ・ 」 ジェネリック ・ 」 「」 bsp </u>	
2 コンジール ☆    2 = ↓ 1 =    2 = ↓ 1 =    3 = Jンジィクレーションデェック ☆    3 = Jンジィクレータ出力    3 = Jンジィクレータ	- <del>1</del> 7
「「「」「」「」」「」」「」」「」」「」」「」」「」」「」」「」」「」」」「」」」「」」」「」」」「」」」	タイン

#### 「コンポーネントの追加」を選択。

1) (2)					
能全て					
イルタ					
コンポーネント^	タイプ	バージョン			
8 ビットタイマ	コード生成	1.4.0			
CRC 演算器	コード生成	1.4.0			
D/A コンバータ	コード生成	1.4.0			
DMA コントローラ	コード生成	1.3.1			
I2C スレーブモード	コード生成	1.4.0			
I2C マスタモード	コード生成	1.4.0			
PWMモードタイマ	コード生成	1.4.0			
r_bsp	FIT	4.00			
最新バージョンのみ表示 明					
:MCU は、8 ビットのカウン	タをベースにした2 チャネ	ルの8 ビットタイマ (TMR)	を2 ユニット (ユニ	:ット0、ユニット1)、合	計4 チャネル内蔵しています。
かいつトウェアコンポーネン	<u>トをダウンロードする</u>				
<u>/////////////////////////////////////</u>					

「8ビットタイマ」を選択して、「次へ」。



19



【 コンポーネントの追加		
選択したコンポーネ: 加します	ントのコンフィグレーションを追	+
8 ビットタイマ		
コンフィグレーション名	Config_TMR1	
カウントモード:	8 ビット	-
リソース:	TMR1	•
(?)	< 戻る(B) 次へ(N) > 終了(E)	キャンセル
L		

「TMR1」を選択して、「終了」。

【☆ スマート・コンフィグレータ			
ファイル ウィンドウ ヘルプ			
			😰   🔐
∰ *RX65-IO.scfg ⊠			X M 2
ソフトウェアコンポーネント設定			
コンポーネント 🎝 🛛 🗄 🎝 🔻	設定		
6.5	○カウント設定		
フィルタ入力	クロックソース	PCLK/1024 -	58.594 (kł
🔺 🧀 Startup	カウンタクリア	コンペアマッチBによりクリア -	
▲ ≧ ジェネリック ♪ r hsp	コンペアマッチAの値(TCORA)	50 カウント -	(実際の値:50)
▲ ➢ Drivers	コンペアマッチBの値(TCORB)	100 カウント	(実際の値:100)
▲ ≽ タイマ Config TMP1	TMO1出力設定		
Coning_TMR1	▼ TMO1出力許可		
	コンペアマッチA時の出力レベル	1出力 👻	- Child Solar
	コンペアマッチB時の出力レベル	0出力 👻	
	割り込み設定		
	□ TCORAコンペアマッチ割り込みを許可(CMIA1)	優先順位 レベル15 (最高) 👻	
	□ TCORBコンペアマッチ割り込みを許可(CMIB1)	優先順位 レベル15(最高)	
	□ TCNTオーバフロー割り込みを許可(OVI1)	優先順位 レベル15(最高)	
-			Þ
概要 ボード クロック コンポーネント 端子 割	0込み		▶ 凡例
ロー 2000 日本 100000000000000000000000000000000		コンフィグレーションチェック XX	
スマート・コンフィグレータ出力		0項目	
M05000001: 端子 31 (こ TMO1 の機能が割り当てられ	,ています ^	記述/説明	タイン
<		٠ [ الا	4

クロックソース「PCLK/1024」 カウンタクリア「コンペアマッチ B によるクリア」 コンペアマッチ A の値「50」カウント ※仮値 コンペアマッチ B の値「100」カウント ※仮値



HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子



TMO1 出力許可

コンペマッチ A の出力レベル「1 出力」

コンペマッチ B の出力レベル「0 出力」

端子 31 に TMO1 の機能が割り当てられています

→31 番ピンは、P26 です。ここでは、P17 に、TMO1 を割り当てたいので、変更します。

									: 🖽 💌 🗖
ser - KX65-IO.scrg 🐹									
端子設定								۵ 🗗	
ハードウェアリソース 🕀 🖻 🛱	뮯	端子機能					3	📕 🔚 è Z	
フィルタ文字列を入力		フィルタス	カ			3	<u>~</u> т	•	
📦 TPU4	*	使用する	機能	端子割り当て		端子番号	方向	備考	
€ TPU5			TMCI1	✓ 設定されていませ	6	/ 設定されて	. なし		
		<b>V</b>	TMO1	P26/TD0/CS6#/	MTIOC2A/TMO1/PO6/TXD1/	/ 31	0		
PPG0			TMRI1	/ 設定されていませ	6	/ 設定されて	. なし		
▲ (0,8ビットタイマ)									
TMR0	=								
TMR1									Acresses
TMR2									
TMR3									
▲ (2) コンペアマッチタイマW									
▲ ## シリアルコミュニケーションインタ									
€ SCI0									
SCI1	-								
4 11 11		•						Þ.	
端子機能 端子番号									
既要 ボード クロック コンポーネント 端子	割り辺	<b>\</b> ∂}							▶ 凡例
			R. 67 R.	≓ E + P T		w <b>7</b> %			1
マート・コンフィグレータ出力					0項目				→I [*]
105000001: 端子 31 (こTMO1 の機能が割り当て	6ท7เ	ます		A	記述/説明		*		9-
				-					
(				Þ	•				

「端子」タブを選択 「TMR1」の設定を開く TMO1 の端子割り当て





スマート・コンフィグレータ								
ファイル ウィンドウ ヘルプ								
								😰   🛐
#RX65-IO.scfg ※							- 6	I 🛃 M 💥 🗖 🗖
端子設定							😼 🖻	
ハードウェアリソース 🕀 🖻 🖧 👗	端子機能					ર	<b>11</b>   <b>15</b>   <b>14</b>   <b>14</b>	
フィルタ文字列を入力	フィルタ入	. <del>.</del>				ৰ্বন্দ	•	
📦 TPU4 🔺	使用する	機能	端子割り当て		端子番号	方向	備考	
TPU5		TMCI1	✓ 設定されていません	J	/ 設定されて	て なし		
▲ @ フログラマフルバレスジェネレータ		TMO1	P26/TDO/CS6#/M	TIOC2A/TMO1/PO6/TXD1/	/ 31	0		
PPG0		TMRI1	/ 設定されていませ/	i.				
			P26/TDO/CS6#/N	1TIOC2A/TMO1/PO6/TXD1/SM	OSI1/SSDA1	/CTS3#/RTS3	#/SS3#/MOSIB-A	
			P17/MTIOC3A/MT	IOC3B/MTIOC4B/TIOCB0/TCL	KD/TMO1/PC	015/POE8#/SC	CK1/TXD3/SMOSI3/S	SDA3/SDA2/SDHI_D3-C/P
TMR2								Revesas
TMR3								
▲ 偽 コンペアマッチタイマW								
CMTW0								
CMTW1								
▲ # # シリアルコミュニケーションインタ:								
SCI0								
😜 SCI1 🔻								
	•						Þ.	
端子機能 端子番号								
概要 ボード クロック コンポーネント 端子 割り込	д							▶ 凡例
ロコンソール ※		B. 🖬 😡 🖂		🕄 コンフィグレーションチェ・	y <b>ク</b> 窓			
スマート・コンフィグレータ出力				0項目				
			*	記述/説明		*		91
			-					
<			- F	•	"	1		

TMO1 の端子割り当ての欄をクリックして、P17 側に変更する。

以上で、スマート・コンフィグレータを使用して、P17にTMO1を割り当てる設定は完了です。

(スマート・コンフィグレータ上の「コード生成」を行うと、TMO1を P17 から出力するコードが生成されます)



コンペアマッチ A の値(TCORA)を 50, コンペアマッチ A の値(TCORB)を 100 に設定した場合、TMO1(P17)からは、上記のような波形が出力されます。

※デューティ比を 50%に設定したい場合は、TCORA=n, TCORB=2n-1 に設定してください



# 1. サンプルプログラム説明(V1.2)

# 1.1. メイン関数

サンプルプログラムのメイン関数の処理内容を以下に示します。

#### ーフローチャートー

#### メイン関数 main()



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



メイン関数は各種初期化後、

・キーボードからの指令読み取り

・プッシュスイッチ SW0-7 の読み取り→LED0-7 の ON/OFF 制御

・P17(ブザー)の周波数を SW0-7 の押したキーに応じて変更(ブザー鳴動時のみ)

・R29(ボリューム)の回転角度を A/D 変換

(7 セグメント LED の表示値、及び P15 のデューティ比に結果を使用)

・P15(LED8)のデューティ比をボリュームの回転角度に連動(LED8 点滅時のみ)

・7 セグメント LED の表示値をボリュームの回転角度に連動(7 セグメント LED ON, HOLD 無効時)

・マトリックススイッチ読み取り、LCD 画面に表示

上記の処理を無限ループします。

メイン関数とは別に

・タイマ(CMT0)による定期的な割り込み処理(100us 毎)

・SW8~SW10 押下による割り込み処理

が実行されます。

main loop は、処理内容により実行時間は前後しますが、最短で 6.5us 程度で 1 ループします。 (LCD 出力の処理が入ると、実行時間は 200us 程度掛かります)

#### 1.1.1. SW0-7 読み取り LED0-7 制御(*1)

SW0-7の状態を読み取り、LED0-7をON/OFF します。



SW0-7 の端子状態を示すレジスタ(PORTD.PIDR)値を、LED0-7 の出力ポート(PORTE.PODR)にコピーしています。 プログラムのコードとしては、1 行のみです。

※マイコンのプログラムは始めてという方は、この部分の処理の変更を行ってみると、ポートの入出力の手法という、 マイコンの制御で最も単純な制御に関しての理解が深まるかと思います

※SW の押している個数を LED の点灯状態に反映させるサンプルが、オリジナルのソースにコメントアウトされて記載されています



#### 1.1.2. ブザーの周波数変更(*2)

SW0-7の状態を読み取り、ブザーの周波数(音程)を変化させます。



ブザー鳴動時に、SW0-7を押すと、ブザーの音程が変化します。音程に対応する周期は、予め配列変数に設定してあり、押したスイッチによって選ぶようになっています。

(音程を変えたい場合は、g_buzz_cycle_arの定義値を変えてみてください。指定可能なのは~255以内です。周期の値なので、値が小さいほど高音で、28 で 2kHz 位です。)



25

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式



#### 1.1.3. A/D 変換(*3)

R29の回転角度を、A/D変換で読み取ります。



プログラム内では、A/D 変換処理中のフラグを確認し、A/D 変換実行中でなければ、A/D 変換を行うよう指示を出し ます。A/D 変換処理中を示すフラグのクリアと、A/D 変換結果の回収は、A/D 変換の処理が終了した時に呼び出され る、割り込み処理内で行っています。

#### 1.1.4. LED8 PWM デューティ変更(*4)

LED8 点滅中は、R29 の回転角度に応じて点灯時間が変化します。



LED8 が点灯している時間を、R29 の A/D 変換値(0~4095)に応じた値に設定しています。 (LED8 は、TPU2 タイマを使用しており、周期レジスタを TGRA, ON 時間を TGRB で設定しています) ※TGRB レジスタの設定は、決められたタイミング(カウンタ値リセットされるタイミング)で行っています (詳細は後述)





#### 1.1.5. 7 セグメント LED 制御(*5)

7 セグメント LED の表示値を設定します。



7 セグメント LED の点灯制御は、100us 毎に定期的に実行されるコンペアマッチタイマ(CMT0)内で行われています。メイン関数内の本処理の目的は、「7 セグ表示変数」に表示値を設定する事です。

A/D 変換値=表示値とすると、A/D 変換値の微妙な変動(0x123→0x124→0x123→…等)が、表示値に反映され、 値が読みにくいので、現在の表示値と一定値(プログラム内の設定値は 3)以上変化した時のみ、表示値を更新する 事としています。なお、この様な処理を行うと、0x002→0x000の様な変更が永遠に反映されないので、強制的に一 定時間(設定値は 2 秒)毎に表示更新を行っています。

7 セグメント LED の状態フラグが、OFF の時は表示フラグをクリア。HOLD の時は値の更新処理は実行せず、表示フラグのセットのみ行っています。

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



27



## 1.1.6. マトリックススイッチ(*6)

マトリックススイッチのスキャン結果を処理します。



マトリックススイッチのスキャン自体は、100us毎のコンペアマッチタイマ(CMT0)処理内で行われており、スキャン結果がスキャン値に格納済みとなっています。メイン関数内では、スキャン値を押されているスイッチ番号(11~26)に変換し、LCDへの表示を行います。

また、2つ以上のキーが押されているかを判断し、2つ以上のキーが押されている場合は、LCD に*(アスタリスク) を表示します。



## 1.1.7. ステッピングモータ(左回転)(*7)

ステッピングモータを左回転させます。



PB4-7 で駆動されている 4 つのトランジスタ(Q5-8)の内、2 素子を L にドライブします。L にドライブする素子は、 250ms 毎に、切り替えを行います。

pos	(PB7)	(PB6)	(PB5)	(PB4)
	Q5	Q6	Q7	Q8
0	L		L	
1	L			L
2		L		L
3		L	L	

(PBn=H のとき、Qx=L)



#### 1.1.8. ステッピングモータ(右回転)(*8)

ステッピングモータを右回転させます。

左回転の場合は、posを0,1,2,3の順で変化させますが、逆回転の場合は、pos=3,2,1,0の順で stepping_motor_drive() 関数を呼び出すという違いのみです。

## 1.2. 割り込み関数

#### 1.2.1. タイマ(CMT0)割り込み処理

r_Config_CMT0_cmi0_interrupt()



タイマ割り込みは、100us 毎に実行される割り込み関数です。定期的に実行したい処理をこの関数内で実行しています。

※CMT0の割り込み優先度は、6に設定しています

6より優先度が高い割り込みは、本ルーチン実行中でも割り込みルーチンに飛びます(多重割り込み許可)

※V1.0 では、CMT0 内で SCI5 の文字表示を行っていましたが、V1.2 では割り込み(スマートコンフィグレータで出 カする SCI5 の API 関数を使用)で文字表示を行う様に変更しました





1.2.2. 7 セグメント LED 表示処理(*1)

※1234を表示する場合

(1)COM1 を選択(P74=H, P75-77=L)、セグメント(PA)を表示値"4"にセット
(2)COM2 を選択(P75=H, P74,P75-77=L)、セグメント(PA)を表示値"3"にセット
(3)COM3 を選択(P76=H, P74-75,P77=L)、セグメント(PA)を表示値"2"にセット
(4)COM4 を選択(P77=H, P74-76=L)、セグメント(PA)を表示値"1"にセット

100us 毎に、(1)→(2)→(3)→(4)→(1)…の繰り返しとしています。



COM 選択変数は、0→1→2→3→0→…の様に変化します。選択された COM に対応する、SEG 信号をドライブ (設定)するという動作となります。

31

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



## 1.2.3. マトリックススイッチスキャン処理(*2)

(1)ROW-1を選択(P64=L, P65-67=H), P60-63を読み取り(SW11-14の状態を読み取り)
(2)ROW-2を選択(P65=L, P64, P66-67=H), P60-63を読み取り(SW15-18の状態を読み取り)
(3)ROW-3を選択(P66=L, P64-65, P67=H), P60-63を読み取り(SW19-22の状態を読み取り)
(4)ROW-4を選択(P67=L, P64-66=H), P60-63を読み取り(SW23-26の状態を読み取り)

100us 毎に、(1)→(2)→(3)→(4)→(1)…の繰り返しとしています。



ROW 選択変数は、0→1→2→3→0→…の様に変化します。選択された ROW に対応する P60-63(COLUMN)の 状態を読み取り、変数に格納します。


(#1)マトリックススイッチ読み取りのウェイトに関して



SW12を押している状態で、ROW 選択が P64 から P65 に移るケースを考えると、以下の様になります。

- ・P64 が選択(=L)の時は、P61 は L
- •P64 を H 設定
- P61 がHに遷移

※12kΩ のプルアップ抵抗で、マイコンの入力端子をHに引き上げるので、ある程度の時間が必要 ※マイコンの入力端子の端子容量(8pF(max))他で、トータル 20pF の場合 12kΩ × 20pF = 240ns 程度の 時定数となる

P65をLに設定

※この時点では P61 の H への遷移は完了していない

- <u>1us のウェイト</u>(240ns の時定数に対し、余裕を持った時間)
   ※P61 は H に遷移済み
- ・P60-63を読み取り



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



### 1.2.1. SW8(IRQ15)割り込み処理

•r_Config_ICU_irq15_interrupt()



### 1.2.2. SW9(IRQ13)割り込み処理

•r_Config_ICU_irq13_interrupt()



HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル



#### 1.2.3. SW10(IRQ4)割り込み処理

r_Config_ICU_irq4_interrupt()



SW8-10の割り込み処理は、

・LCD 表示

·SCI 表示

・フラグ変数の切り替え(トグル)

となります。

SW8(IRQ15)は、ブザー鳴動の ON/OFF。SW9(IRQ13)は、LED8 の ON/OFF。SW10(IRQ4)は、7 セグメント LED の、ON/HOLD/OFF の切り替えとなります。





## 1.3. ユーザ定義関数

ユーザ定義関数は、

HSBRX65-IO-BOARD_SAMPLE(_RX671)¥src¥user

以下に格納されているものです。

## 1.3.1. io_board.c に含まれる関数

#### mdelay

概要:ウェイト関数

#### 宣言:

void mdelay( unsigned long msec );

#### 説明:

指定された時間待ちます

```
※100us(0.1ms)刻みのタイマでカウントしているので、精度としては 0.1ms オーダです
```

#### 引数:

msec: 待ち時間、ms 単位

#### 戻り値:

なし

#### 使用例:

mdelay(500); //この行で 500ms 待ちます

#### sw_led

概要:SW0-7 読み取り LED0-7 制御関数

#### 宣言:

void sw_led(void);

説明:

SW0-7を読み取り、押されている SWn(n=0-7)に対応する、LEDn(n=0-7)を点灯させます

引数:

なし

#### 戻り値:

なし

seg_led

概要:7 セグメント LED 制御関数

宣言:

void seg_led(unsigned short val, unsigned short dp);



HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル



説明:

指定された値を処理してグローバル変数に代入します 引数:

val:表示值 ※hex で指定、0x0000~0xFFFF

dp: 少数点 ※0x0~0xf

4つの少数点を全点灯させる場合は、Oxfを指定

1.234の様に最上位桁の小数点を表示させる場合は、0x8

戻り値:

なし

その他:

val を桁毎に分解し、g_seg_char[0]~g_seg_char[3]に代入します(g_seg_char[]は、0x0~0xf)

dp を桁毎に分解し、g_seg_dp[0]~g_seg_dp[3]に代入します(g_seg_dp[]は、0 または 1)

使用例:

seg_led(0x1234, 0); //7 セグメント LED に、"1234"を表示させる(*1)(※指定するのが hex 値な事に注意) seg_led(0xabcd, 0); //7 セグメント LED に、"ABCD"を表示させる(*1) seg_led(0x0123, 0x8); //7 セグメント LED に、"0.123"を表示させる(*1) seg_led(0x0123, 0x4); //7 セグメント LED に、"01.23"を表示させる(*1)

(*1)正しくは、本関数では表示用の変数に値を代入するのみです

seg_drive

概要:7 セグメント LED セグメント信号ドライブ関数

宣言:

unsigned char seg_drive(unsigned char seg_char, unsigned char dp);

説明:

7 セグメント LED のセグメント信号をドライブするレジスタ値を返します

引数:

seg_char: 表示させる文字(0x0~0xf)

dp: 小数点(点灯:1, 消灯:0)

戻り値:

セグメント信号の、PORTA.PODR.BYTE に相当するレジスタ値 使用例:

PORT7.PODR.BYTE=0x10; //P74のみH(COM1を選択) PORTA.PODR.BYTE = seg_drive(0x3, 1); //1桁目に、"3."を表示

seg_char	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
7 セグ表示	B	Β			$\Box$	5	Б	B
戻り値	0x3f	0x06	0x5b	0x4f	0x66	0x6d	0x7d	0x7





seg_char	0x8	0x9	0xa	0xb	0xc	0xd	0xe	Oxf
7 セグ表示	B	$\Box$	Ħ	B			E	E
戻り値	0x3f	0x6f	0x77	0x7c	0x39	0x5e	0x79	0x71

※戻り値は、dp=1(少数点点灯)の場合は、上記の値に 0x80 を OR した値となります

・セグメントと戻り値の関係



数値の1を表示させる場合は、SegBとSegCを点灯させれば良いので、bit1, bit2を1に設定する。 →0b00000110 = 0x6

-seg_drive 関数ではサポートされていない表示に関して-

符号のマイナス(-)を表示させる場合

→bit6のみ1に設定 0b0100 0000 = 0x40 PORTA.PODR.BYTE=0x40;

アルファベットの日を表示させる場合

→bit5, bit4, bit6, bit1, bit2を1に設定 0b0111 0110 = 0x76 PORTA.PODR.BYTE=0x76;

上記の様に、7つのセグメントで表示可能なものであれば、表示可能です。

#### stepping_motor_drive

概要:ステッピングモータ駆動関数

宣言:

void stepping_motor_drive(unsigned char pattern);

説明:

指定された値に応じて、ステッピングモータ駆動端子を ON/OFF します

引数:

pattern: 0~3を指定

戻り値:

なし





ー指定する pattern とステッピングモータに流す電流の方向の関係ー



pattern	PB7	PB6	PB5	PB4
	А	Ā	В	B
0	L		L	
1	L			L
2		L		L
3		L	L	

pattern 引数に、0~3の値を入れて本関数を呼び出した場合、上記の方向の電流を流します。

#### 使用例:

while(1)

{

```
stepping_motor_drive(0);
mdelay(250);
stepping_motor_drive(1);
mdelay(250);
stepping_motor_drive(2);
mdelay(250);
stepping_motor_drive(3);
```

```
mdelay(250);
```

}

250ms 毎に、モータ軸を回転させる例

(pattern を 3,2,1,0 の順で変化させた場合は、回転方向は逆となります。)





stepping_motor_en 概要:ステッピングモータ出力イネーブル関数 宣言: void stepping_motor_en(unsigned char on_off); 説明: ステッピングモータ駆動端子の出力モード/Hi-Zを切り替えます 引数: on_off: OFF(0), ON(1)を指定 戻り値:

なし

buzz_pitch

概要:ブザーの音程関数

宣言:

void buzz_pitch(unsigned char cycle);

説明:

TMR1 タイマ(P17/TMO1)の出力周期を変更します

引数:

cycle: 周期(~255)を指定(1/60MHz×1024=17.07us, cycle×17.07usの周期となります) 戻り値:

なし

### 1.3.2. Icd_1602.c に含まれる関数

lcd_init

概要:LCD 初期化関数

宣言:

void lcd_init(void);

説明:

LCD(SC1602)の初期化を行います

引数:

なし

戻り値:

なし

lcd_cmd

概要:LCD コマンド送信関数

宣言:

void lcd_cmd(unsigned char c);







説明:

LCD にコマンドを送信します

引数:

c: 送信するコマンド

#### 戻り値:

なし

使用例:

lcd_cmd(0xc4);

カーソルを2行目の5桁目に移動

-LCD のカーソル移動(任意の場所に文字表示する方法)-

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x40	0x41	0x42	0x43	<b>0x44</b>	0x45	0x46	0x47	0x48	0x49	0x4a	0x4b	0x4c	0x4d	0x4e	0x4f

LCD を、2 行モードで使用した場合、2 行 × 16 文字表示となりますが、文字(キャラクタ)データを保管するメモリ (DDRAM)のアドレスは、上記となります。(2 行目が 0x40 から始まります)

カーソル移動のコマンドは、0b1nnn nnnn(n:address)ですので、n のところに 0x44(0b0100 0100)を設定すると、コ マンドとしては、(0b1100 0100) 0xc4 となります。

lcd_hs1

lcd_hs2

概要:LCD カーソル移動関数

宣言:

void lcd_hs1(void);

void lcd_hs2(void);

説明:

LCD のカーソル(文字を表示する位置)を変更します

lcd_hs1 は、1 行目 1 文字目、lcd_hs2 は、2 行目 1 文字目です

引数:

なし

戻り値:

なし

lcd_clear 概要:LCD 表示クリア関数 宣言:

void lcd_clear(void);





説明:

LCD 表示をクリアします

引数:

なし

戻り値:

なし

lcd write char

概要:LCD 文字表示関数

宣言:

void lcd_write_char(unsigned char c);

説明:

LCD に1 文字表示させます(表示は、現在のカーソル位置です)

引数:

c: 表示する文字を指定

戻り値:

なし



lcd write hex

概要:LCD 表示関数

宣言:

void lcd_write_hex(unsigned char c);

説明:

LCD に1 バイトの hex コードを表示させます



HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子





引数:

c: 表示するコード

#### 戻り値:

#### なし

#### 使用例:

lcd_write_hex(0x2D); //LCD画面に"2D"を表示させます

#### lcd_write_byte_int

概要:LCD 表示関数

#### 宣言:

void lcd_write_byte_int( unsigned char num );

#### 説明:

LCD に1 バイトを数値で表示します

#### 引数:

num: 表示する数値(0~255)

#### 戻り値:

なし

#### 使用例:

Icd_write_byte_int(125); //LCD画面に"125"を表示させます

#### lcd_write_short_int

概要:LCD 表示関数

#### 宣言:

void lcd_write_short_int( unsigned short num );

#### 説明:

LCDに2バイトを数値で表示します

#### 引数:

num: 表示する数値(0~65535)

#### 戻り値:

なし

#### 使用例:

lcd_write_byte_int(1200); //LCD画面に"1200"を表示させます

lcd_write_str

概要:LCD 文字列表示関数

#### 宣言:

void lcd_write_str(unsigned char *str);

説明:

LCD に文字列を表示します





引数:

*str: 表示する文字列のポインタ(¥0 終端)

戻り値:

なし

使用例:

lcd_write_byte_int("ABC"); //LCD画面に"ABC"を表示させます

## 1.3.3. sci.c に含まれる関数

※V1.2 で変更

sci_start

概要:SCI 初期化関数

宣言:

void sci_start( void );

説明:

SCI の動作開始を行います

引数:

なし

戻り値:

なし

sci_write_char

概要:SCI1文字出力関数

宣言:

void sci_write_char( unsigned char c );

説明:

SCIに1文字出力行います

引数:

c: 表示する文字の文字コード(1を表示する場合は、0x31 または'1')

戻り値:

なし

sci_write_str

概要:SCI 文字列出力関数

宣言:

void sci_write_str( const char *str );





説明:

SCI に文字列を出力します

引数:

*str: 文字列(¥0 終端)

#### 戻り値:

なし

説明:

```
文字列に、"¥n"(0x0a)が含まれる場合は、"¥r¥n"(0x0d,0x0a)に変換されます
```

sci_write_uint8_hex

sci_write_uint16_hex

sci_write_uint32_hex

概要:SCI hex 出力関数

#### 宣言:

```
void sci_write_uint8_hex( unsigned char c );
void sci_write_uint16_hex( unsigned short s );
void sci_write_uint32_hex( unsigned long I );
```

説明:

SCI に hex コードを出力します

引数:

```
c: hex コード(8bit)
```

```
s: hex ⊐−ド(16bit)
```

```
I: hex コード(32bit)
```

#### 戻り値:

なし

#### 使用例:

```
sci_write_str("0x");
sci_write_uint8_hex( 0x5a );
端末に、"0x5a"が表示されます。
```

sci_write_uint8

```
sci_write_uint16
```

sci write uint32

概要:SCI 数值出力関数

宣言:

void sci_write_uint8( unsigned char c ); void sci_write_uint16( unsigned short s ); void sci_write_uint32( unsigned long l );





説明:

SCI に符号なしで数値を出力します

引数:

- c: 表示させる数値(8bit)
- s: 表示させる数値(16bit)
- l: 表示させる数値(32bit)

戻り値:

なし

使用例:

unsigned short s = 0x8000; sci_write_uint16( s ); 端末に、"32768"が表示されます。

sci_write_int8

sci_write_int16

sci write int32

概要:SCI 数值出力関数

宣言:

void sci_write_int8( char c ); void sci_write_int16( short s ); void sci_write_int32( long I );

説明:

SCI に(負の数値の場合)符号付きで数値を出力します

引数:

c: 表示させる数値(8bit)

s: 表示させる数値(16bit)

I: 表示させる数値(32bit)

戻り値:

なし

使用例:

short s = 0x8000; sci_write_int16( s ); 端末に、"-32768"が表示されます





floart2str float2str_eformat double2str double2str eformat 概要:浮動小数点数-文字列変換関数 宣言: void float2str( float value, int num, char *str ); void float2str_eformat( float value, int num, char *str ); void double2str( double value, int num, char *str ); void double2str_eformat( double value, int num, char *str ); 説明: 浮動小数点数を文字列に変換します (_eformat は 1.23e-3 の様に e 形式に変換を行う関数です) 引数: value: 表示させる数値(float/double 型) num: 小数点以下の桁数 *str: 文字列変換後の先頭ポインタ 戻り値: なし 使用例: float f = 1.23456f; char buf[20]; float2str(f, 2, buf); sci_write_str(buf); 端末に、"1.23"が表示されます。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株式会社



#### sci_read_char

概要:SCI1文字読み出し関数

#### 宣言:

unsigned short sci_read_char( unsigned char *c );

#### 説明:

SCIから1文字読み出しを行います

#### 引数:

c: 読み出し結果(ポインタ)

#### 戻り値:

0: 受信データがあり、*c に読み出した文字(ポインタ)を格納

0xffff(SCI_RECEIVE_DATA_EMPTY): バッファに溜まっている受信データがない

#### sci_read_str

概要:SCI 文字列読み出し関数

#### 宣言:

unsigned short sci_read_str( char *str, unsigned short size );

#### 説明:

SCI から文字列を読み出します

#### 引数:

str: 読み出し結果(ポインタ)

size: 読み出すバイト数

#### 戻り値:

0: 受信データが size バイト以上あり、*str に読み出した文字の先頭ポインタを格納 0xfff(SCI_RECEIVE_DATA_EMPTY): バッファに size バイトの受信データが溜まっていない





## 1.4. グローバル変数

## 1.4.1. io_board.c で定義している変数

・ウェイト関数向け

unsigned long g_wait_x100_us_counter

ウェイト関数(mdelay)向け変数。0以外の時は、100us毎に減算されます。mdelay関数では、この変数値が0になるのを待ちます。

・7セグメントLED向け

unsigned short g_seg_state //OFF, ON, HOLD

7セグメントLEDをON(1), OFF(0), HOLD(2)制御する状態制御フラグ変数。

unsigned short g_seg_flag //OFF, ON

7セグメントLEDの表示・非表示を制御するフラグ変数。

unsigned short g_seg_com

7セグメントLEDのCOM選択を保存する変数。0~3。

unsigned char g_seg_char[4]

7セグメントLEDの表示値を保存しておく変数。4:桁数分。

unsigned char g_seg_dp[4]

7セグメントLEDの小数点の表示値を保存しておく変数。4:桁数分。

・マトリックススイッチ

unsigned short g_matrix_scan

マトリックススイッチのROW選択を保存する変数。0~3。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



### unsigned short g_matrix_sw[4]

マトリックススイッチの読み取り値を保存する変数。4:ROW数分。

・A/D変換

unsigned short g_adc_flag

A/D変換中を示すフラグ。A/D変換実行前に1にセットし、A/D変換終了時に0にクリアされます。

unsigned short g_adc_val[1]

A/D変換結果を代入する変数。A/D変換終了時に、A/D変換値が代入されます。1:1ch分。

#### unsigned short g_adc_val_update

A/D変換結果を強制的に7セグメントLEDに反映させる変数。

unsigned long g_adc_val_update_counter

A/D変換結果を強制的に7セグメントLEDに反映させるためのカウンタ変数。





-A/D変換値を7セグメントLEDに表示させるフローー



ボリューム(R29)の回転角度に応じた A/D 変換値が、7 セグメント LED に表示される動作となっていますが、A/D 変換値は、ばらつきが出るため、「A/D 変換値=7 セグメント LED 表示値」とした場合、表示が高速に切り替わり、値 が読みにくいケースがあります。

そのため、

・現在の表示値から3以上差分が付いた場合

または

・2 秒に1回

表示値を更新するプログラムコードとしています。

・ブザー、LED8

unsigned short g_buzz_flag unsigned short g_led8_flag

ブザーとLED8をON/OFFする変数。

unsigned short g_tpu2_tgrb

TPU2のコンペアマッチB(TPU2.TGRB=PWMのデューティ比を決めるレジスタ値)の設定値を代入する変数。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル 株



#### ・任意のタイミングでTPU2.TGRBを更新した場合



任意のタイミングでTGRBを更新した場合、点滅の周期として設定している1秒より長く点灯するタイミングが来るケ ースがあります。(点灯時間を、60→20へと変更したにも拘わらず、120となるケースがでてきます)

TGRAコンペアマッチのタイミングでTPU2.TGRBを更新



任意のタイミングで、グローバル変数(g_tpu2_tgrb)に値をセットしておき、レジスタ値(TPU2.TGRB)に反映させるタ イミングを、TGRAのコンペアマッチ割り込みのタイミングとする事により、設定周期より長いON時間となる事が起きな いようにしています。

const unsigned char g_buzz_cycle_ar[8]={28,25,22,21,19,17,15,14};

ブザー周期(周波数)の設定値。1 サイクル=1/60MHz×1024=17.07[us] 28:2093[Hz] ※初期値 25:2344[Hz] 22:2663[Hz] 21:2790[Hz] 19: 3084[Hz] 17: 3447[Hz] 15: 3906[Hz] 14: 4185[Hz] 

52

HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル



## 1.4.2. sci.c で定義している変数

unsigned char sci_send_buf[2][SCI_SEND_BUF_SIZE]

SCI表示バッファ(デフォルトでは、1024文字×2)。

unsigned char sci_recv_buf[SCI_RECV_BUF_SIZE]

SCI受信バッファ(デフォルトでは、16文字)。

1.5. 定義値

#### 1.5.1. io_board.h で定義している値

#define OFF 0 #define ON 1 #define HOLD 2

ブザー, LED, 7セグメントLED等のON/OFFフラグ変数に設定、比較する定数。

#define ADC_VAL_HIST 3 //3以上変化した場合表示を更新 #define ADC_VAL_UPDATE_INTERVAL 20000 //2秒毎に強制的に更新

7 セグメント LED の表示安定化動作を決める定数。

#### 1.5.1. lcd_1602_RX65.h で定義している値

#define LCD_1602_RS_PFS MPC.P50PFS.BYTE=0x00; #define LCD_1602_E_PFS MPC.P52PFS.BYTE=0x00; #define LCD_1602_DB4_PFS MPC.P90PFS.BYTE=0x00;

#define LCD_1602_DB5_PFS MPC.P91PFS.BYTE=0x00; #define LCD_1602_DB6_PFS MPC.P92PFS.BYTE=0x00; #define LCD_1602_DB7_PFS MPC.P93PFS.BYTE=0x00;

#define LCD_1602_RS_PMR PORT5.PMR.BIT.B0=0; #define LCD_1602_E_PMR PORT5.PMR.BIT.B2=0; #define LCD_1602_DB4_PMR PORT9.PMR.BIT.B0=0;



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル

## Hohuto

#define LCD_1602_DB5_PMR PORT9.PMR.BIT.B1=0; #define LCD_1602_DB6_PMR PORT9.PMR.BIT.B2=0; #define LCD_1602_DB7_PMR PORT9.PMR.BIT.B3=0;

#define LCD_1602_RS_PDRPORT5.PDR.BIT.B0#define LCD_1602_E_PDRPORT5.PDR.BIT.B2#define LCD_1602_DB4_PDRPORT9.PDR.BIT.B0#define LCD_1602_DB5_PDRPORT9.PDR.BIT.B1#define LCD_1602_DB6_PDRPORT9.PDR.BIT.B2#define LCD_1602_DB7_PDRPORT9.PDR.BIT.B3

#define LCD_1602_RS_PODRPORT5.PODR.BIT.B0#define LCD_1602_E_PODRPORT5.PODR.BIT.B2#define LCD_1602_DB4_PODRPORT9.PODR.BIT.B0#define LCD_1602_DB5_PODRPORT9.PODR.BIT.B1#define LCD_1602_DB6_PODRPORT9.PODR.BIT.B2#define LCD_1602_DB7_PODRPORT9.PODR.BIT.B3

LCD の各制御端子の割付を指定。

## 1.5.2. sci.h で定義している値

#### #define SCI_SEND_BUF_SIZE 1024

SCIの送信バッファサイズ(バイト数)を定義。このサイズのバッファを2つ使用しますので、実際に消費されるメモリは、1024×2=2048[bytes]になります。

#### #define SCI_RECV_BUF_SIZE 16

SCIの受信バッファサイズ(バイト数)を定義。

(※キーボードからの入力を受け取るバッファを想定しているので、16文字としています。データの受信に使用する場 合等バッファサイズを増やす事は可能です。)





# 2. マイコン使用機能

## 2.1. 使用機能一覧

機能	用途	備考
CMT0	各種(7 セグメント LED、マトリックススイッチ)	100us 周期
コンペアマッチタイマ 0	周期処理	
IRQ4	SW10 押下処理	立下りエッジ、デジタルフィルタ
		使用
IRQ13	SW9 押下処理	立下りエッジ、デジタルフィルタ
		使用
IRQ15	SW8 押下処理	立下りエッジ、デジタルフィルタ
		使用
汎用ポート	7 セグメント LED	
	マトリックススイッチ	
	プッシュスイッチ	
	LED	
	ステッピングモータ	
S12AD0	R29の回転角度読み取り	
12 ビット A/D コンバータ		
SCI5	PC との通信	
シリアルコミュニケーション		
インタフェース		
TMR1	ブザー	
8ビットタイマ		
TPU2	LED8	
16ビットバルスタイマユニット		

## 2.2. I/O ボードの各機構に割り当てられているマイコン機能

I/O ボード機構	端子	マイコン機能	マイコン機能	マイコン機能	マイコン機能
		(1)	(2)	(3)	(4)
7 セグメント LED(SEG1-2)	PA0-7, P74-77	汎用ポート			
キャラクタ LCD(J4)	P50,52,P90-93	汎用ポート			
ブザー(B1)	P17	汎用ポート	TMO1	TIOCB0	MTIOC3A
	P15		TIOCB2	MTIOC0B	
LED(LED8)	P17	汎用ポート	TMO1	TIOCB0	MTIOC3A
	P15		TIOCB2	MTIOC0B	
ボリューム(R29)	P40	汎用ポート	AN000		
割り込み用スイッチ(SW8)	P07	汎用ポート	IRQ15	*ADTRG0	
割り込み用スイッチ(SW9)	P05	汎用ポート	IRQ13		
割り込み用スイッチ(SW10)	P14	汎用ポート	IRQ4		
マトリックススイッチ	P60-67	汎用ポート			
(SW11-26)					
プッシュスイッチ(SW0-7)	PD0-7	汎用ポート			
LED(LED0-7)	PE0-7	汎用ポート			
ステッピングモータ(J5)	PB4-7	汎用ポート	PO28-31		
USB-Serial(J7)	PC2-3	汎用ポート	RXD5,TXD5		

※赤太字は、本サンプルプログラムでの使用機能





## 2.3. 使用割り込み一覧

本サンプルプログラムで使用している割り込みの一覧を示します。

割り込み(機能名)	優先度	用途
CMI0(CMT0)	6	7 セグメント LED, マトリックススイッチ, SCI バッファ出力
コンペアマッチタイマ		
IRQ4(ICU)	10	SW10 押下検出
端子割り込み		
IRQ13(ICU)	10	SW9 押下検出
端子割り込み		
IRQ15(ICU)	10	SW8 押下検出
端子割り込み		
RXI5(SCI5)	8	USB-Serial 受信
SCI5 受信割り込み		
TXI5(SCI5)	8	USB-Serial 送信処理
SCI5 送信データエンプティ		
TEI5(SCI5)	8	USB-Serial 送信処理
SCI5 送信完了割り込み		
ERI5(SCI5)	8	USB-Serial エラー処理
SCI5 エラー割り込み		
TGI2A(TPU2)	4	TPU2.TGRB レジスタ更新
TPU2 コンペアマッチ		
S12ADI(S12AD)	12	A/D 変換値を変数に格納
A/D 変換完了		

※優先度は、数値の大きなほうが優先です

## 2.4. I/O ポート初期設定

汎用ポートの初期設定を以下に示します。(初期設定は、スマート・コンフィグレータで設定しています)

I/O ボード機構	端子	ポート設定	備考
7 セグメント LED(SEG1-2)	PA0-7, P74-77	出力(初期値 0)	消灯
キャラクタ LCD(J4)	P50,52,P90-93	出力(初期値 0)	
マトリックススイッチ	P60-63	入力	
(SW11-26)	P64-67	出力(初期値1)	ROW 非選択
プッシュスイッチ(SW0-7)	PD0-7	入力	
LED(LED0-7)	PE0-7	出力(初期値1)	LED は消灯
ステッピングモータ(J5)	PB4-7	出力(初期値 0)	



## 2.5. SCI5(USB-Serial)を使用したキャラクタの送受信

サンプルプログラム V1.2 で SCI の使用方法を大きく変えています。

~V1.1 では、SCIの初期化のみスマートコンフィグレータで行い、文字出力のところはオリジナルのリングバッファ構成としていました。V1.2 では、スマートコンフィグレータの API 関数(文字出力、文字入力)を使う構成としています。



文字出力用のバッファは2面(2つ)あり、表示用のバッファはAPI関数(R_Config_SCI5_Send)に渡され、シリア ル端末に順次データが送られます。シリアルは、115,200bps設定なので、1文字を送信するのに、87us(1/115,200 ×10)掛かります。これは、マイコンの処理(120MHz動作、1クロックあたり8.3ns)に比べて非常に遅いので、文字 出力が終わるまで待っていると効率が悪いためバックグラウンドで送信処理を行わせています。

※UART(SCI)では、スタートビット=1, データ(1文字)=8bit, ストップビット=1 で合計 10bit で1文字送信します





表示用バッファのデータを全て出力した時点で、今度はデータ格納バッファのデータを API 関数に渡して、バッファの表(表示用)と裏(データ格納用)を入れ替えます。



API 関数を使用する事で移植性(サンプルコードでは、条件分けを行い、RX 向けのコードを有効化していますが、 RL78 や RA マイコンでも同じユーザ関数を使える様になっています)が高まります。





# 3. マイコンボードへのプログラム書き込み方法

## 3.1. デバッガを使用した書き込み

#### CS+を起動し、

HSBRX65-IO-BOARD_SAMPLE - CS+ for CC -	[プロパティ]	
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(H)	
	🕐 🖓 🏯 🔍 👻 100% 👻 ன ன DefaultBuild	- 🗶 : 👧 📭 🐂 🔳 🕟 🕞 🗠 🚳 🖙 🚛 🚈 🕌
4 x	MSBRX65-IO-BOARD_SAMPLE。 🚰 プロパティ	
	🔐 RX E1(Serial) のプロパティ	
HSBRX65-IO-BOARD SAMPLE (70	▲ 内蔵ROM/RAM	0010
	内蔵ROMサイス[K/バイト]       内蔵RAMサイズ[K/バイト]	2048 648
CC-RX (ビルド・ツール)	データフラッシュ・メモリ・サイズ[Kバイト]	32
	メイン・クロック・ソース	EXTAL
	メイン・クロック周波数[MHz] 動作国的数[MHz]	24.0000
	内蔵フラッシュ・メモリ書き換え時のクロック操作を許可する	いいえ
	▲ Iミュレータとの接続 エミュレーが川アルNo	
HSBRX65-IO-BOARD_SAMPLE.c	▲ ターゲット・ボードとの接続	
Config CMT0	エミュレータから電源供給をする(最大200mA) 通信方式	CVC/Z ETNE
Config ICU	FINEボーレート[bps]	2000000
Gonfig_PORT	▲ フラッシュ 10コードの入力モード	10コードを166第32桁で指定
		FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
B-□ Config_SCI5		1000 1280
Config_TMR1	▲ CPUの動作モード	
Config_TPU2	モード端子設定	シングルチップモード シングルチップモード
general	エンディアン	Little-endianデータ
er config	メイン・クロック周波数[MHz]	
	メイン・クロックカーEXTALの場合、EXTAL局版報値をUUUU1 - 99,999の間で人力してください。	
sci		1

デバッグ・ツール(右クリック)-使用するデバッグツール

で、お手持ちのデバッガ(E1, E2, E2Lite, E20)を選択してください。

※E1, E20の場合は、SerialとJTAGが選べますが、どちらを選んでも問題ありません

※E2, E2Lite の場合は、デバッガ選択後のプロパティで、通信方式:"JTAG", "FINE"のどちらでも選択可能です

メイン・クロック周波数[MHz] 24.0000 動作周波数[MHz] 120.0000

クロック設定で、上記値を入力してください。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



6	HSBRX65-IO-BOARD_SAMPLE - CS+ for CC -	[HSBRX6	5-IO-	BOARD_SAMPLE.c]			
フ	ァイル(F) 編集(E) 表示(V) プロジェクト(P)	ビルド(B)	デノ	「ッグ(D) ツール(T) ウインドウ(W) へい	ルプ(H)	_	
1	🔅 スタート(S)   退 📄 🗿 🕌 📸 🖷	e   #	L D.	デバッグ・ツールヘダウンロード(D)		uild	- 🔨
1	R R R R R R D R R R R B	בעע-	6	ビルド&デバッグ・ツールヘダウンロード	^с (В) F6		
	לםטורייער סביד די ד	📝 HSB	5	リビルド&デバッグ・ツールヘダウンロー	- ド(W)		
بر	2 🕜 🙎 🔳	#b   #b	- 00	デバッグ・ツールへ接続(C)			
	B ISBRX65-IO-BOARD SAMPLE (71		D)	デバッグ・ツールからアップロード(U)			
N.		81	2	デバッグ・ツールから切断(N)	Shift+F6	-	
171	- · · · · · · · · · · · · · · · · · · ·	82		使用するデバッグ・ツール(L)	•	-	
		84 85			Shift+E5	-	
	🕀 プログラム解析 (解析ツール)	86 87		(F) 実行(G)	F5		
		88	0	ブレークせずに 実行(F)	F8	の文明期日化	
	■ I Eルト・ツール主成ノアイル ■ HSBRX65-IO-BOARD SAMPLE C	90			F11	0010#411C	
	Smart Configurator	91	7.E	$7 = \sqrt{2} \cdot \frac{1}{2} \cdot 1$	F10		
	Config_CMT0	93 94	≡ريا ح∽				
		95 96					
	Config_PORT	97 98	Leto.		CUH+F5		
		99	1		01 514		
		101	(44)	テバック・ツールの状態を巻き戻り(W)	Ctrl+F11	-	
	Emeral Config_TPU2	102		デバッグ・ツールの状態保存(V)	•		
		104		switch(xc) {			
	ten la r_config	106		case 'b':			
		108		sci_print_buf("¥n_buzz	er (P17)>¥n″);		

デバッグービルド&デバッグ・ツールへダウンロード

※プログラムのビルドとマイコンボードへのプログラムのダウンロード(書き込み)を同時に行う場合

HSBRX65-IO-BOARD SAMPLE - BX E1(Serial)	- CS+ for CC - [HSBR	X65-IO-BOARD SAMPLE ()	
ファイル(F) 編集(F) 表示(V) プロジェクト(P) ト	ーード(B) デバッグ(D		
	(* 88 * *		
	ソリューション一覧(		
😳 לםטיבטאישי איש איש איש איש איש איש איש איש איש	MSBRX65-IO-BOA	RD_SAMPLE。 習うの行う 高速アセンブル1	IOR
2 3 2 3	81 (8) ( <b>-&gt;</b> 🔿 🖝	カラム・	🛛 🖉 🦃 🖏 🗙 表記(
<u>「 HSBRX65-IO-BOARD SAMPLE (プロジェク</u>	行 PHP PD PA		PORT1
A ■ R5F5651EDxFB (マイクロコントローラ)	38 ffe009e4	i koid main (void)	IOR
	40		BSC 810
■ Strail (デバッグ・ツール)	41 49	unsigned char xc; =	
プログラム解析 (解析ツール)	43	unsigned char buzz_cycle_tmp;	🕀 🗂 CAN 1
🗊 ファイル	44 45 ffe009e7	unsigned for add_disp_val=0;	
■ 🚺 ビルド・ツール生成ファイル	46 47 ffe009ea	short adc_disp_diff; unsigned_bar_grew_matrix_sw_nn=0:	1 CMT 1
HSBRX65-IO-BOARD_SAMPLE.c	48 ffe009ed	unsigned char watrīx_sw_no=0;	E CMT2
Smart Configurator	50 ffe009f3	unsigned than matrix sw pushed count-0, unsigned than matrix sw pushed count-0;	E CMT WO
Config_CMT0	51 ffe009f6 52 ffe009f9	unsigned short stepping_motor_pos=0;	
Config PORT	53	unsigned short i;	⊞ 🗂 DA
	54 55 ffe009fc	R Config SCI5 Start():	🕀 🗂 DMAC
	56 ffe00a00	sci_init();	DMACU
	58 ffe00a04	R_Config_CMUTO_Start();	DMAC2
Config_TPU2	59 60 ffe00a08	R Config TCH IR04 Start O:	DMAC3
⊕- ll general	61 ffe00a0c	R_Config_ICU_IRQI3_Start();	E CI DMAC5
the second seco	63	K_outris_tod_tkuij_otart(),	E C DMAC6
⊕ I _ comig	64 ffe00a14 65 ffe00a1d	g_adc_flag = 1; B_Config_S12ADD_Start():	E DMAC/
e- sci	66 07		
	68 ffe00a26	PORTI_PMR-BIT-B5 = 0;	出力
io_board	60 66a00a0a	1 11 DANTI DAN DIT 07 - 0.	LEVEL1
			Communi.dll

メイン関数(void main(void))の行でプログラムが停止、アドレスのところにプログラムのアドレスが表示されていれば、プログラムのダウンロードは成功しています。(プログラムがマイコンボードに書き込まれています)

HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子





デバッグを終了させる場合は、

デバッグ-デバッグ・ツールから切断

を選んで、デバッガとマイコンボードの接続を切った後で、ボード電源断等行ってください。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



#### - デバッガの設定に問題があるケース-



デバッガの設定で「RX シミュレータ」を選択した場合、プログラムのダウンロードは実行されますが、マイコンボード に対しての書き込みは行われていません。

(プログラムは、メイン関数実行より前で止まります)

RX シミュレータの場合は、ツールがマイコンの動作を模擬するモードですので、実際のマイコンボードに対しての書込み、マイコンボード上でのプログラム実行等は行われません。





## 3.2. USB ケーブルを使用した書き込み

製品付属の USB ケーブル(USB-A-miniB)を、

マイコンボードの J5

に挿して、PCと接続してください。



マイコンボードに給電を行ってください。

(1)I/O ボード J14 に AC アダプタを接続

(2)マイコンボード J6 に 5V 電源を接続

(3)マイコンボード J9 をショートに設定し、USB ケーブル経由で電源を供給

※(1)~(3)のいずれかを選んで、給電を行ってください

マイコンボード、SW1(DIP-SW)の2のみ、ON(下側)に設定(他は OFF)。 SW2(リセット)を押す。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



#### RFP(RenesasFlashProgrammer)を起動

📕 Renesas Flash Programmer V3.05.01 (無	價版)
ファイル(F) デバイス情報(D) ヘルプ(H)	
新しいプロジェクトを作成(N)	接続設定 ユニークコード
プロジェクトを開く(0)	
フロジェクトを保存(S)	
イメージファイルを保存(I)	
ファイルチェックサム(C)	
1 F14_CSUM.rpj	
2 G140_CSUM.rpj	参照(B)
3 G10_CSUM.rpj	
4 RX231.rpj	
終了(X)	
スタート	(S)
スタート( Renesas Flash Programmer V3.05.01 [3 Sep 2018] プロジェクトをロードします。(C.¥Users¥win64-5¥Docum CSUMrpj)	(S) (無償版) ents¥Renesas Flash Programmer¥V3.05¥F14_CSUM¥F14

ファイルー新しいプロジェクトを作成

📓 新しいプロジェクトの	作成	
プロジェクト情報		
マイクロコントローラ( <u>M</u> ):	RX65× -	
プロジェクト名( <u>N</u> ):	RX65IO	
作成場所( <u>F</u> ):	C:¥Users¥win64-5¥Documents¥Renesas Flash Pr	参照( <u>B</u> )
通信 ツール(I): COM インタフェース(I): 2 wire UART - ツール詳細(D) 番号: COM6		
	接続( <u>0</u> )	*+>セル(C)

マイクロコントローラ RX65x を選択(HSBRX671F144 の場合は RX67x を選択) プロジェクト名 任意の名称 を入力 ツール COM を選択

ツール詳細ボタンを押す





ジール詳細 (COM)	
ツール選択	
COM6 : Prolific USB-to-Serial Comm Port	
<u></u> K	キャンセル(C)

PC 上に、COM ポートが複数存在している場合は、選択ができますので、

RX USB Boot(CDC)

となっている、COM ポート(上記では、COM21)を選択してください。

※RX USB Boot のデバイスが見えない場合は、マイコンボードの DIP-SW(SW1)の設定を確認後、マイコンボードを リセット(もしくは電源の再投入)してください

「OK」を押す。

🚺 新しいプロジェクトの	作成	
プロジェクト情報		
マイクロコントローラ( <u>M</u> ):	RX65x •	
プロジェクト名( <u>N</u> ):	RX65IO	
作成場所( <u>F</u> ):	C:¥Users¥win64-5¥Documents¥Renesas Flash Pri	参照(B)
通信 ツール(I): COM インタフェース(I): 2 wire UART マ ツール詳細(D) 番号: COM21		
		キャンセル(O)

「接続ボタン」を押す。

🚺 IDコードの設定	
IDコード認証 IDコード(1):	
	<u>OK</u> (**)±1/(0)

出荷時状態、特段 ID コードを設定していない状態ですと、デフォルトで FF(x16)となっていますので、「OK」を押す。

HSBRX65-IO-BOARD サンプルソフトウェア マニュアル #





🜠 Renesas Flash Programmer V3.05.01 (無償版)	
ファイル(E) デバイス情報(D) ヘルプ(H)	
操作 操作設定 ブロック設定 接続設定 ユニークコード	
プロジェクト情報	
現在のプロジェクト: RX65IOrpj	
マイクロコントローラ: RX Group	エンディアン( <u>E</u> ): リトル 🔹
プログラムファイル	
	参照(B)
フラッシュ操作	
消去 >> 書き込み >> ベリファイ	
7h - h(s)	
デバイス情報を取得します。 デバイス - BY Group	*
Code Flash 1 (アドレス: 0xFFFF0000、サイズ: 64 K、消去サイズ: 8 K)	
Code Flash 1 (アドレス:0xFFE00000、サイズ:1984 K、消去サイズ:32 K)  Data Flash 1 (アドレス:0x00100000、サイズ:32 K、消去サイズ:64)	
Config Area (アドレス:0xFE7F5D00、サイズ:128、消去サイズ:0)	
ツールから切断します。	=
採作か成功しました。	
	<b>•</b>
	ステータスとメッセージのクリア( <u>C</u> )

デバイス情報取得後、「操作が成功しました」という表示が出れば、問題ありません。

※接続に失敗した場合は、マイコンボードをリセット(もしくは電源の再投入)後に、再度「接続」を行ってください

プログラムファイル「参照」ボタンを押し、ビルドした mot ファイルを指定してください。

プログラムファイルを指定	じてください。				×
	リ ▶ ドキュメント ▶ cubesuite ▶ HSBRX65-IO-B	OARD_SAMPLE   DefaultBuild	<b>- 4</b> <del>9</del>	DefaultBuildの検索	Q
整理 ▼ 新しいフォル	ダー			•=== •	
★ お気に入り ダウンロード	ドキュメント ライブラリ DefaultBuild			並べ替え: フォルタ	<b>ヺ</b> — ▼
📃 デスクトップ	名前	更新日時	種類	サイズ	
1910 最近表示した場所 名 OneDrive	HSBRX65-IO-BOARD_SAMPLE.mot	2019/04/17 11:56	MOT ファイル	2 61 KB	
<ul> <li>⇒ ライブラリ</li> <li>⇒ Subversion</li> <li>⇒ ドキュメント</li> <li>⇒ ピクチャ</li> <li>≅ ピデオ</li> <li>⇒ ミュージック</li> <li>▲ コンピューター ↓</li> </ul>					

66

HSBRX65-IO-BOARD サンプル ソフトウェア マニュアル 株式会社 北手電子



mot ファイル(ビルドしたプログラムのバイナリを書き込み用にテキストフォーマット化したファイル)は、プロジェクトフ オルダの、DefaultBuild の下にあります。

マイコンボードをリセット(マイコンボードの SW2 を押す)(もしくは電源再投入)してください。 ※重要

※作成済みのプロジェクトを読み出して再度実行する場合は、この時点でのリセットは不要です

🜠 Renesas Flash Programmer V3.05.01 (無償版)	
ファイル( <u>E</u> ) デバイス情報( <u>D</u> ) ヘルプ( <u>H</u> )	
操作 操作設定 ブロック設定 接続設定 ユニークコード	
プロジェクト情報	
現在のプロジェクト: RX65IOrpj	
マイクロコントローラ: RX Group	エンディアン(E): リトル 🔹
プログラムファイル	
C:¥Users¥win64-5¥Documents¥cubesuite¥HSBRX65-IO-BC	ARD_SAMPLE¥DefaultBuik 参照_(B)
	CRC-32 : EBCBC7AB
フラッシュ操作	
消去 >> 書き込み >> ベリファイ	
スタート( <u>S</u> )	
デバイス情報を収得します。 デバイス名:RX Group	<b>^</b>
Code Flash 1 (アドレス:0xFFFF0000、サイズ:64 K、消去サイズ:8 K Code Flash 1 (アドレス:0xEFF00000、サイズ:1984 K、消去サイズ:8	) 2 K)
Data Flash 1 (アドレス: 0x00100000、サイズ: 32 K、消去サイズ: 64)	
Config Area (アトレス: 0xFE7F5D00、サイス: 128、)自去サイス: 0)	
ツールから切断します。 操作が成功しました。	=

スタートボタンを押します。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル



Renesas Flash Programmer V3.05.01	<mark>(</mark> 無償版)	
ファイル( <u>E)</u> デバイス情報( <u>D</u> ) へル	プ( <u>H</u> )	
操作 操作設定 ブロック設定 接続設定	ユニークコード	
プロジェクト情報		
現在のプロジェクト: RX65IOrpj		
マイクロコントローラ: RX Group	I	ッディアン(E): リトル ・
プログラムファイル		
C:¥Users¥win64-5¥Documents¥cubesu	ite¥HSBRX65-IO-BOARD_SAMPL	.E¥DefaultBuil 参照( <u>B</u> )
	CRC-3	2 : EBCBC7AB
フラッシュ操作		
消去 >> 書き込み >> ベリファイ		
7.6		
79-	· r( <u>5</u> )	止吊終」
[Config Area] 0xFE7F5D00 - 0xFE7F5D2F	サイズ:48	A
[Config Area] 0xFE7F5D40 - 0xFE7F5D7F	サイズ:64	
ベリファイを実行します。		
[Config Area] 0xFE7F5D00 - 0xFE7F5D2F	サイズ:48	
[Config Area] UXFE7F5D4U - UXFE7F5D7F	サイズ:64	
ツールから切断します。		
深作が成功しました。		=
		-
		ステータスとメッセージのクリア( <u>C</u> )

消去、書き込み、ベリファイ後、「操作が成功しました」の表示が出力されれば、プログラムの書き込みは成功しています。

※書き込みに失敗した場合は、マイコンボードをリセット(もしくは電源の再投入)後に、再度「スタート」を行ってください

USB ケーブルをマイコンボード、J5 から抜いてください。(マイコンボードの電源を切断してください)

マイコンボードの SW1(DIP-SW)を全て OFF 側に設定してください。

マイコンボードに電源を接続してください。

上記手順で、マイコンボードに書き込まれたプログラムが実行を開始します。

デバッガが手元にないといった場合でも、USB ケーブルのみでプログラムの書き込みを行う事が可能です。


### 付録

#### 取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2019.4.18	—	初版発行
REV.1.1.0.0	2022.4.13	P8,10-13,36,64 P15 P30,44-48,53,54,56-58	HSBRX671F144 の記載を追加 コメント追加 SCI 処理変更に伴う修正
REV.1.2.0.0	2022.5.31	P6,7	ボード Ver2.0 の注意事項追加

#### お問合せ窓口

最新情報については弊社ホームページをご活用ください。 ご不明点は弊社サポート窓口までお問合せください。

## 株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7 TEL 011-640-8800 FAX 011-640-8801 e-mail:support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用) URL:https://www.hokutodenshi.co.jp

商標等の表記について

- ・ 全ての商標及び登録商標はそれぞれの所有者に帰属します。
- ・ パーソナルコンピュータを PC と称します。



HSBRX65-IO-BOARD サンプルソフトウェア マニュアル

# HSBRX65-IO-BOARD サンプルソフトウェア マニュアル

©2019-2022 北斗電子 Printed in Japan 2022 年 5 月 31 日改訂 REV.1.2.0.0 (220531)

_{株式会社} 上手電子

ルネサス エレクトロニクス社 RX651/RX671(QFP-144 ピン)搭載 HSB シリーズマイコンボード向け I/O ボード