Microcomputer embedded software beginner's guide

(Training kit use for「R8C/2D」made by Renesas Technology Corp.)

Step3

株式会社ピーアイシステム

ご使用になられる前に

1. 本資料は、教育・研修に使用することを目的としております。 従って記載のプログラム、アルゴリズム等を機器等の製品に組み込んだ場合、弊社は責任を負いません。
2. 本資料は参考資料であり、弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
3. 本資料に記載の全ての情報に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
4. 本資料の輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要は手続を行ってください。
5. 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。

ご注意

■ HEW,High-performance Embedded Workshop は株式会社ルネサスエレクトロニクス社の登録商標です。

目次

5. Step3(キーマトリクス入力)

5.1. 概要

複数のキースイッチの状態を知るためには本来、キースイッチの数だけ入力ポートが必要になります。
しかし、例えばキーボードのように大量のキースイッチがあってはマイコンの入力ポートが不足してしまうことになります。 このようなときに使われる方法がキーマトリクスです。
本章では 4×4 の計 16 個のキー入力方法について学習します。

5.2. 解説

キー入力の基本的な構成を下図に示します。

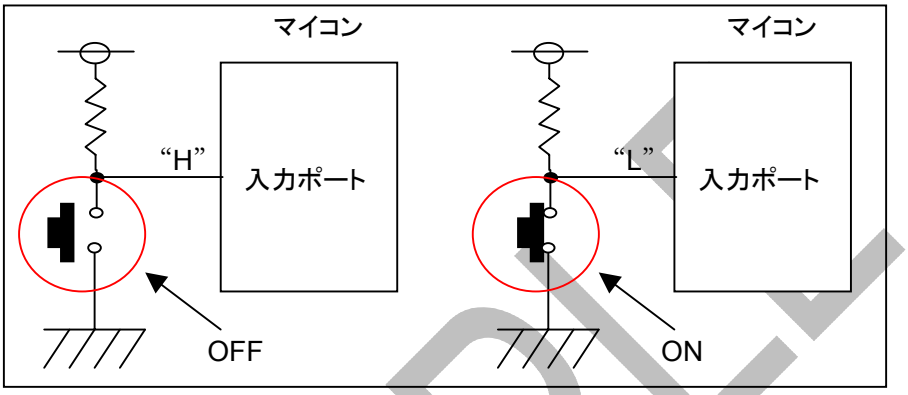


図 5.2.1 キー入力構成図

上図ではキースイッチが OFF(押されていない)時は入力ポートは“L”レベルが入力され、キースイッチが ON(押された)時は入力ポートは“H”レベルが入力されます。
キースイッチが少ない場合は上図の構成で良いのですが、キースイッチの数が多くなるとその分ポートが必要となり、効率的ではありません。 このようなときは使用ポート数を少なくする為にキーマトリクスで構成します。
キーマトリクスとは図 5.3.1 のように入力ポートと出力ポートを組み合わせたキー入力構成のことです。 キースイッチが 16 個必要な場合、上図のような構成では 16 のポートが必要になりますが、キーマトリクスの場合は 8 つ(入力ポートが 4 つ、出力ポートが 4 つ)で済みます。

本章ではキースイッチに接続されている入力ポートをキー入力、キースイッチに接続されている出力ポートをスキャン出力と呼称します。
P101MS0008-A ではキー入力は内部プルアップ抵抗に接続する為、キースイッチが押されていない時は“H”レベルが入力されます。(内部プルアップ抵抗については項番 5.4.3 ⑤の解説を参照して下さい。) キースイッチが押された時はスキャン出力の出力レベルがそのまま入力されます。 つまり、スキャン出力を“L”レベル出力にした時にキー入力が“L”レベルであれば「接続されているキースイッチが押されている」と判定することができます。

5.3. 仕様

キーマトリクス入力処理の仕様を以下に示します。

1. 10ms 毎にスキャン出力を切り替え、キースイッチを入力する。
2. 全キースイッチの入力結果が 2 度一致した時、キー入力確定とする。
   従って、キースイッチ確定は最初のキー入力があってから 80ms 後となる。
3. キースイッチの多重押しや渡り押しは禁止とし、キースイッチ動作させない。
4. 押下したキーコードを 16 進数で LED へ表示する。多重押し及び渡り押しの場合は、“FF”を表示する。

P101MS0008-A のキーマトリクス接続図、入出力論理とキーマトリクス入力処理の入出力タイミングチャートを以下に示します。

5.3.1. 接続図

キースイッチが 16 個ある場合、キーマトリクス入力はスキャン出力 4 本、キー入力 4 本、計 8 本のポートを使用します。

キーマトリクスの接続図を下図に示します。

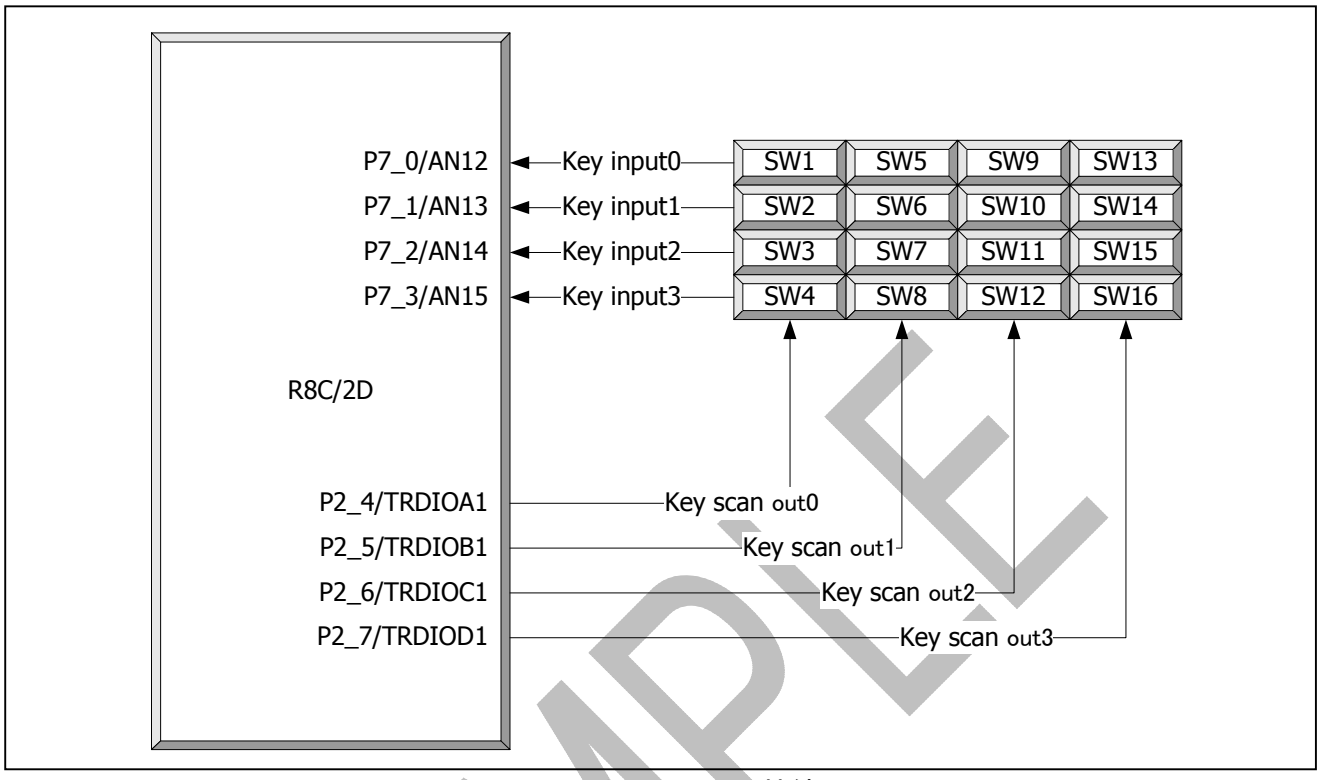


図 5.3.1 キーマトリクス接続図

キーマトリクス入力はスキャンラインを順次切り替え、ライン全てでのキー入力を読み取ったところで、最初のキー入力が確定します。 確定したキー入力データをもとにキーコードを生成します。 各キースイッチのキーコードを下表に示します。

表 5.1 キースイッチとキーコード

	P7_3		P7_2		P7_1		P7_0	
ライン	キー	キーコード	キー	キーコード	キー	キーコード	キー	キーコード
P2_4	SW4	04h	SW3	03h	SW2	02h	SW1	01h
P2_5	SW8	08h	SW7	07h	SW6	06h	SW5	05h
P2_6	SW12	0Ch	SW11	0Bh	SW10	0Ah	SW9	09h
P2_7	SW16	10h	SW15	0Fh	SW14	0Eh	SW13	0Dh

キー入力ポートは P7 レジスタの下位 4bit となりますので、一例として、
「スキャンラインが P2_5(SW5～SW8)の P7 レジスタ値が 0x0B(00001011b)だった場合は、SW7 が押されている」という事になります。

5.3.2. 入出力論理

ポートの出力論理について下表に示します。

表 5.2　ポート出力論理

ポート	PORT 名称	I/O	Active	Initial
P2_4/TRDIOA1	Key scan out0	O	L	H
P2_5/TRDIOB1	Key scan out1	O	L	H
P2_6/TRDIOC1	Key scan out2	O	L	H
P2_7/TRDIOD1	Key scan out3	O	L	H
P7_0/AN12	Key input0	I	L	-
P7_1/AN13	Key input1	I	L	-
P7_2/AN14	Key input2	I	L	-
P7_3/AN15	Key input3	I	L	-

本章では P1_1、P1_2、P0_0、P0_1、P0_2、P0_3、P7_4、P7_5、P7_6、P7_7 ポートとして使用します。

5.3.3. 出力タイミングチャート

入出力タイミングチャートについて下図に示します。

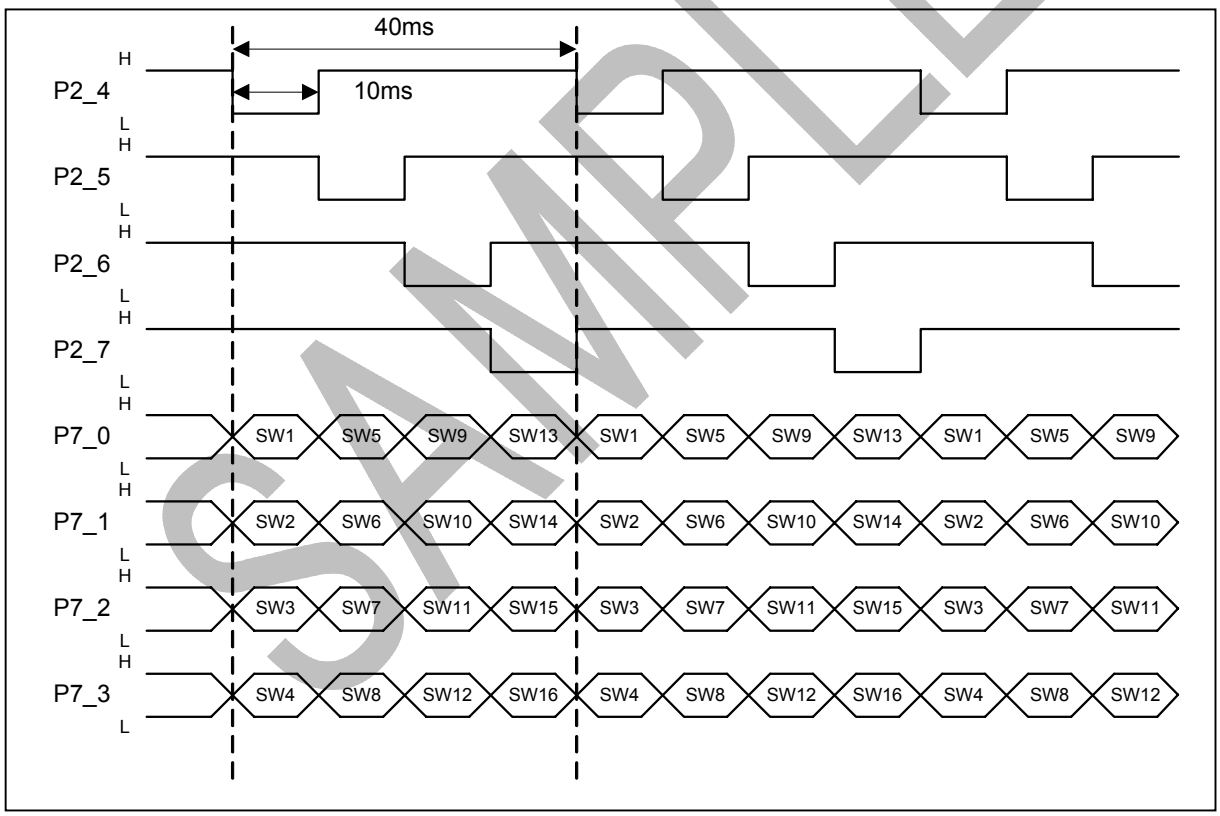


図 5.3.2 キーマトリクス入出力タイミング

キーマトリクス制御の為の SFR 設定手順を次項のフローチャートにて解説します。

5.4. フローチャート

キーマトリクス入力処理のフローチャートを以下に示します。

フローチャート内の太字は SFR のマクロ定義名です。 SFR のマクロ定義名は HEW のプロジェクト作成時に自動生成される R8C/2D SFR 定義ファイル“sfr_r82d.h”で定義されています。

5.4.1. メイン処理

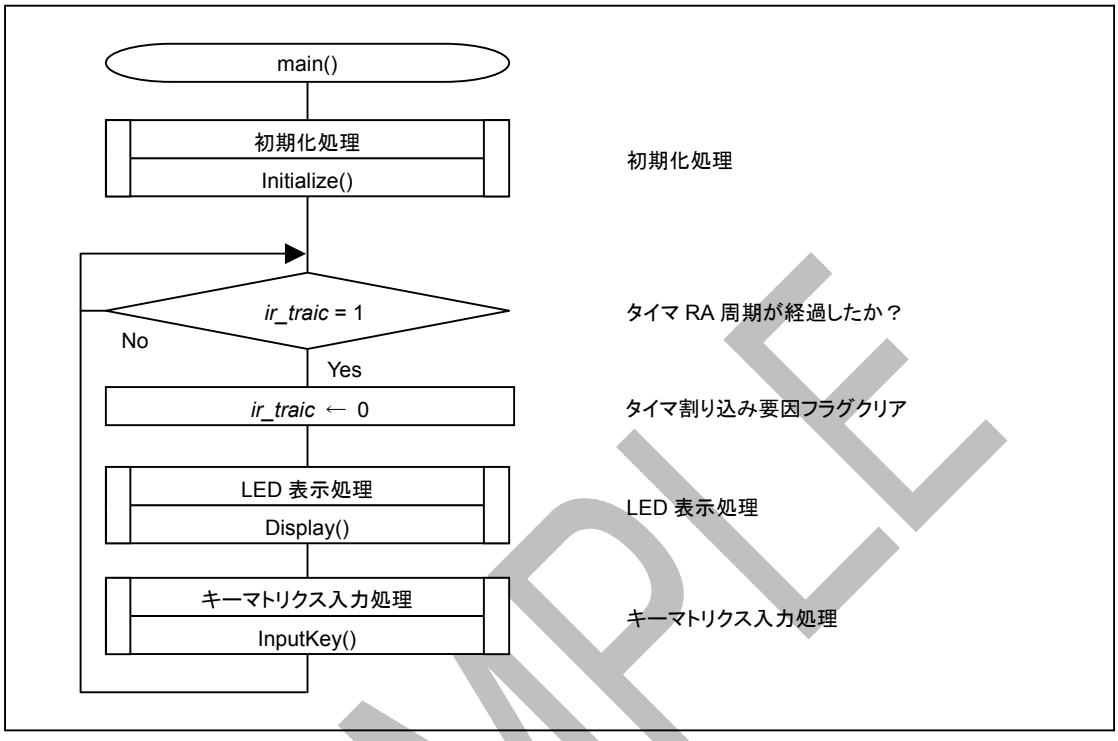


図 5.4.1 メイン処理(サンプルソースはリスト 5.5.1 ①参照)

5.4.2. 初期化処理

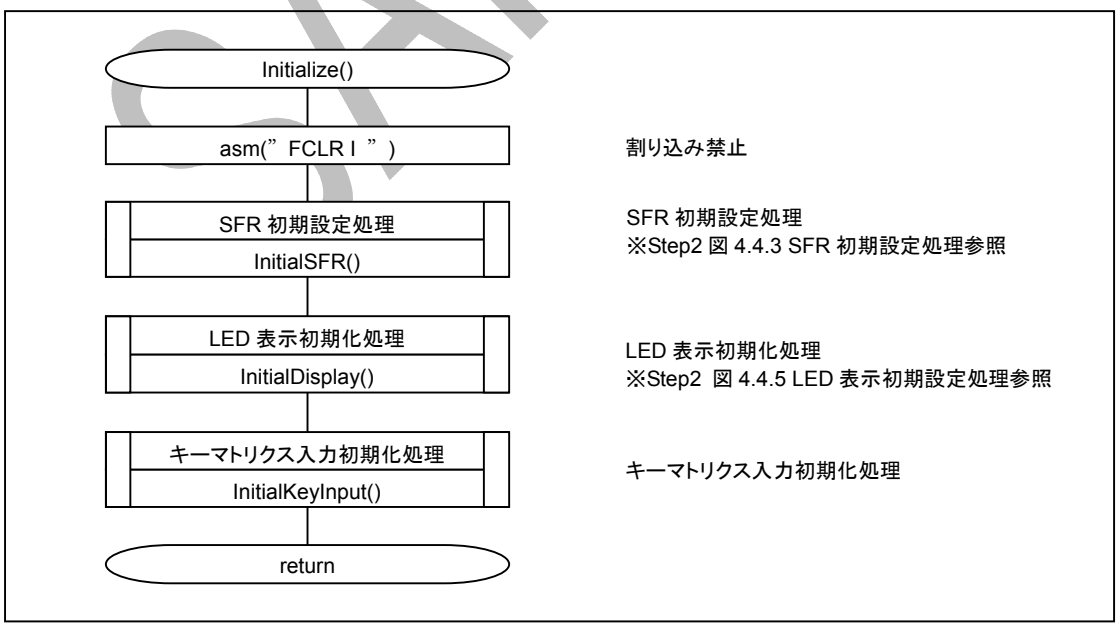


図 5.4.2 初期化処理(サンプルソースはリスト 5.5.2 ①参照)

5.4.3. ポート初期設定処理

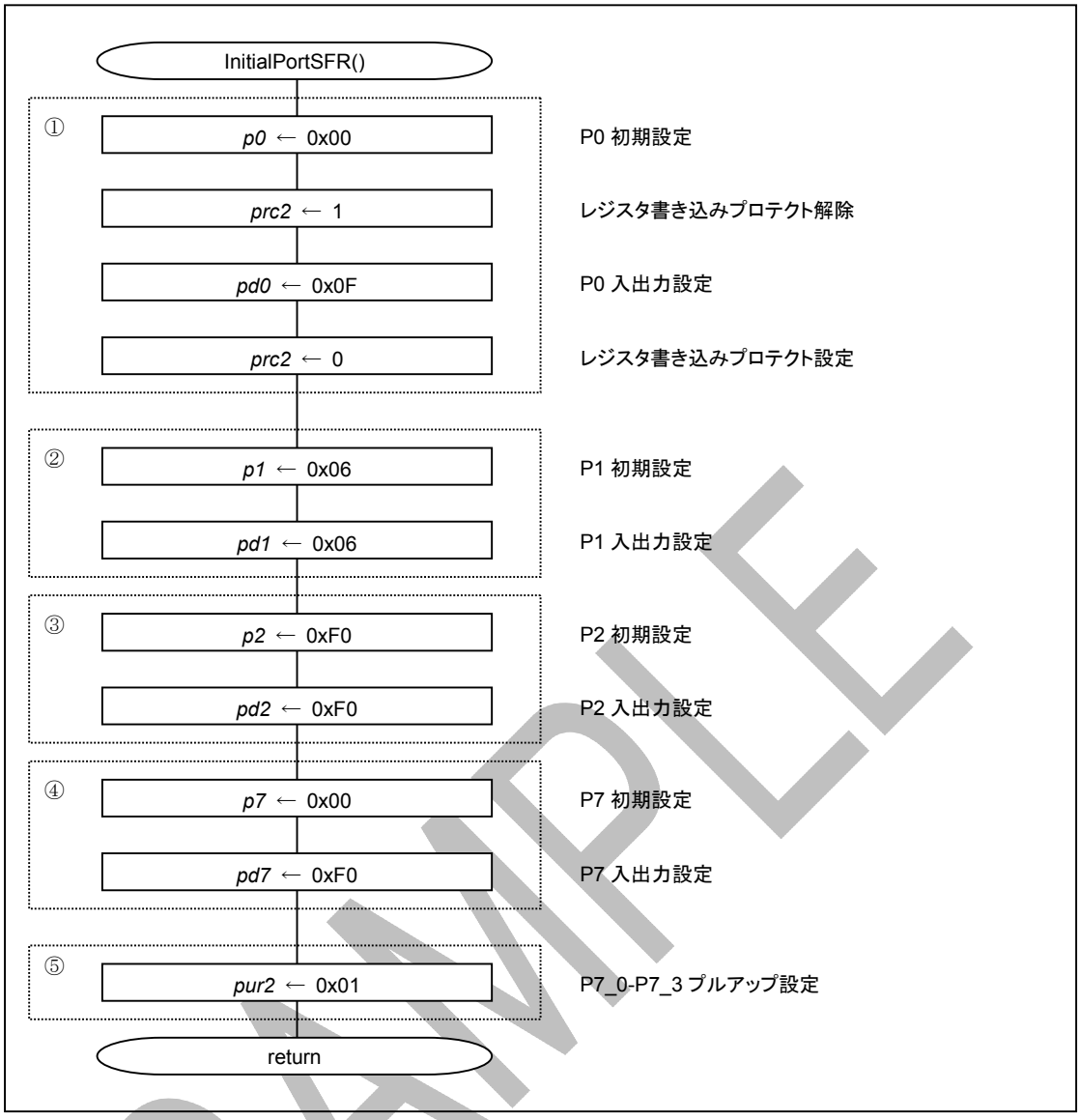


図 5.4.3 ポート初期設定処理(サンプルソースはリスト 5.5.3 ①参照)

①出力ポートに設定した際に不定なレベルが出力されないようにするため、ポート P0_0～P0_3 を出力ポートに設定する前にポート P0_0～P0_3 に非アクティブレベル値を設定します。
PD0 レジスタへの書き込みプロテクトを解除した後、ポート P0_0～P0_3 を出力ポートに設定し、PD0 レジスタへの書き込みプロテクトを再設定します。
②ポート P1_1～P1_2 に非アクティブレベル値を設定し、ポート P1_1～P1_2 を出力ポートに設定します。
③ポート P2_4～P2_7 に非アクティブレベル値を設定し、ポート P2_4～P2_7 を出力ポートに設定します。
④ポート P7_4～P7_7 に非アクティブレベル値を設定し、ポート P7_4～P7_7 を出力ポートに設定します。
⑤ポート P7_0～P7_3 に内部プルアップ抵抗を接続します。 プルアップ抵抗とは入力ポートに何も接続されていない時にそのポートを”H”レベルに固定するために用いる抵抗のことです。
ポート P7_0～P7_3 は入力ポートで、接続図のようにキースイッチが接続されています。
キースイッチが押下されるとポート P2_4～P2_7 とポート P7_0～P7_3 が接続され、
ポート P2_4～P2_7 の出力をポート P7_0～P7_3 で入力できます。ですが、キースイッチが押下されていない状態ではポート P7_0～P7_3 に何も接続されていない状態となるため、
ポート P7_0～P7_3 の値が不定となります。これを防ぐためにプルアップ抵抗が必要となります。
R8C/2D はマイコン内部にプルアップ抵抗を持っているため、本書ではこの内部プルアップ抵抗を使用することとします。

5.4.4. キーマトリクス入力初期化処理

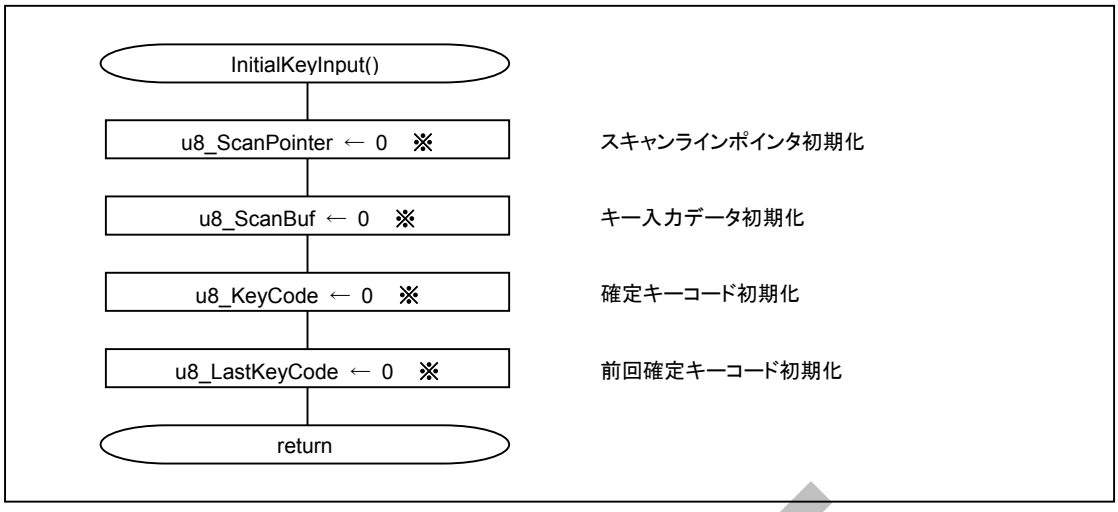


図 5.4.4 キーマトリクス入力初期化処理(サンプルソースはリスト 5.5.7 ②参照)

※各変数の定義は以下の通り。

```
static u8 u8_ScanPointer;

static u8 u8_ScanBuf[ 4 ];

static u8 u8_KeyCode;

static u8 u8_LastKeyCode;
```

5.4.5. LED 表示処理

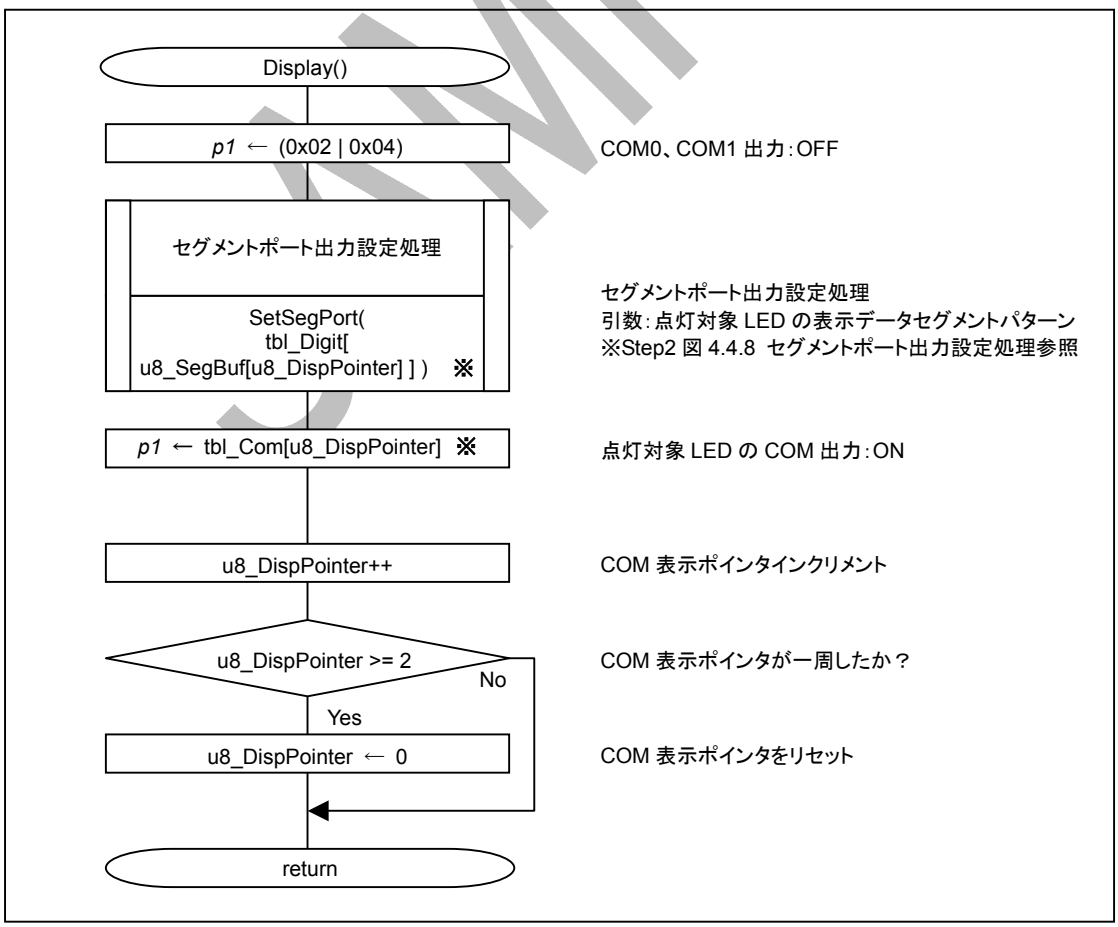


図 5.4.5 LED 表示処理(サンプルソースはリスト 5.5.5 ①参照)

※tbl_Digit[]、tbl_Com[]のテーブル値は以下の通り。

```
const static u8 tbl_Digit[]={
  { 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x27,0x7F,0x6F,0x77,0x7C,0x58,0x5E,0x79,0x71 }
};

const static u8 tbl_Com[]={
  { 0xFD,0xFB }
};
```

5.4.6. キーマトリクス入力処理

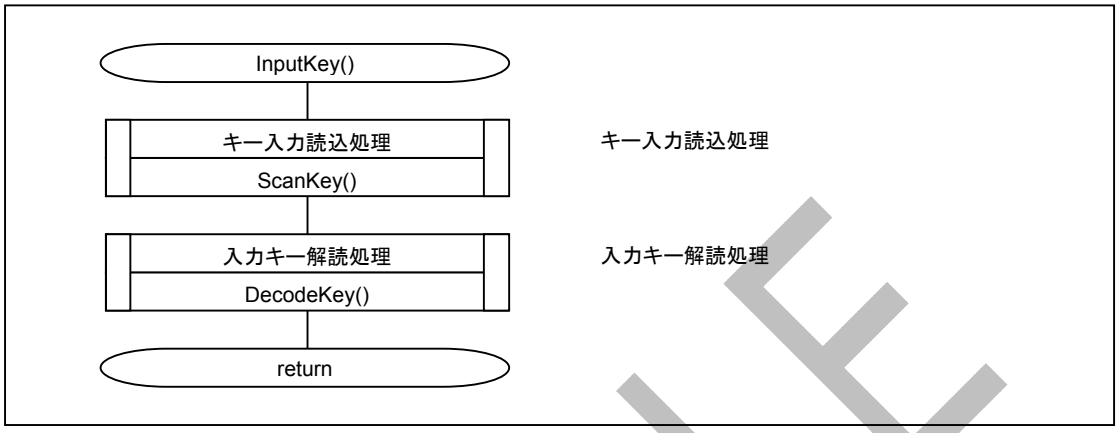


図 5.4.6 キーマトリクス入力処理(サンプルソースはリスト 5.5.7 ①参照)

5.4.7. キー入力読込処理

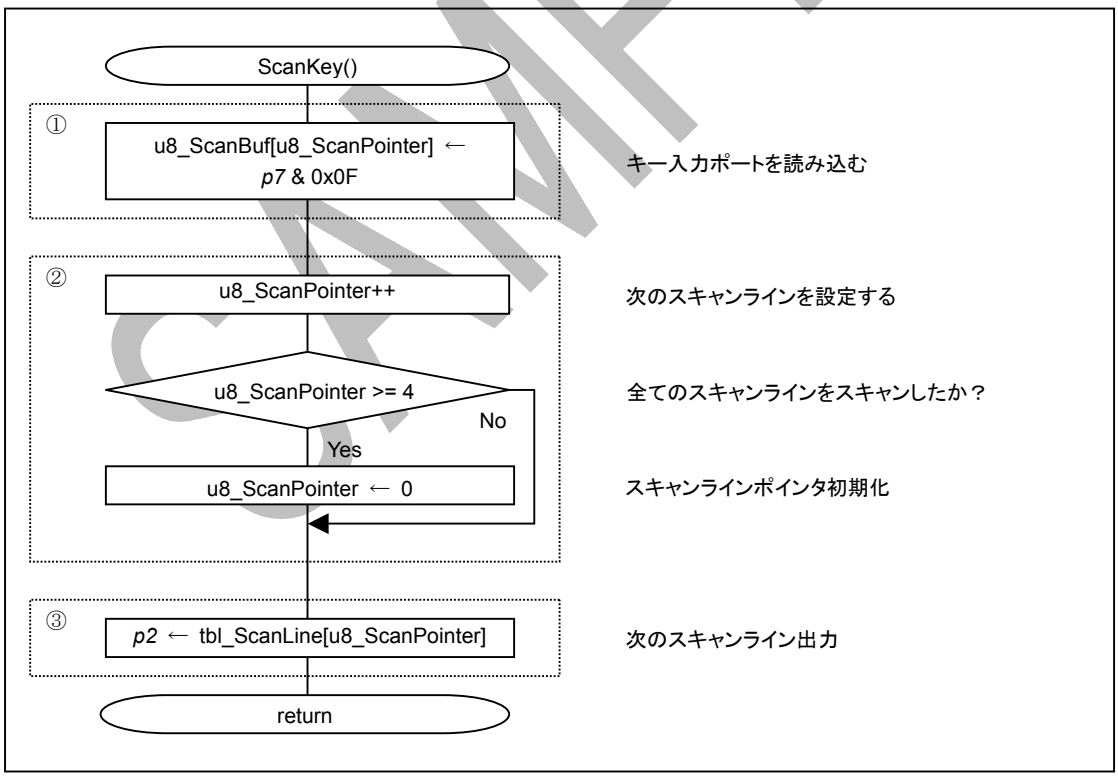


図 5.4.7 キー入力読込処理(サンプルソースはリスト 5.5.8 ④参照)

①キースキャンラインポインタが示すスキャンラインのキー入力ポートの値を入力データバッファに格納します。
②キースキャンラインポインタをインクリメントし、値が”4”以上になったら”0”に初期化します。
③次回にスキャンするラインのスキャン出力ポートに”L”出力します。

※tbl_ScanLine[]のテーブル値は以下の通り。

```
const static u8 tbl_ScanLine []={
   0xE0, 0x06, 0xB0, 0x70
};
```

5.4.8. 入力キー解読処理

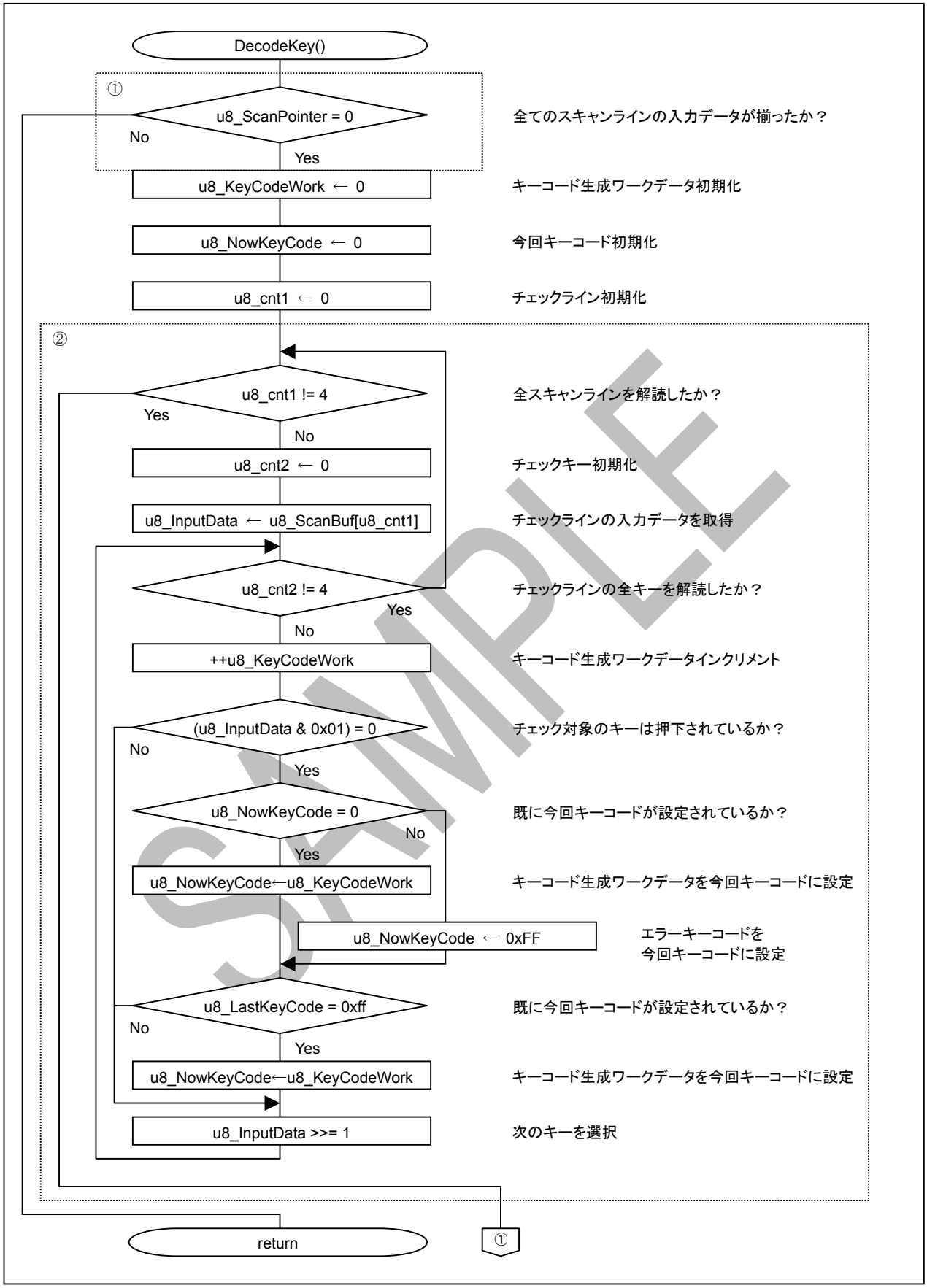


図 5.4.8 入力キー解読処理 - 1(サンプルソースはリスト 5.5.8 ⑤参照)

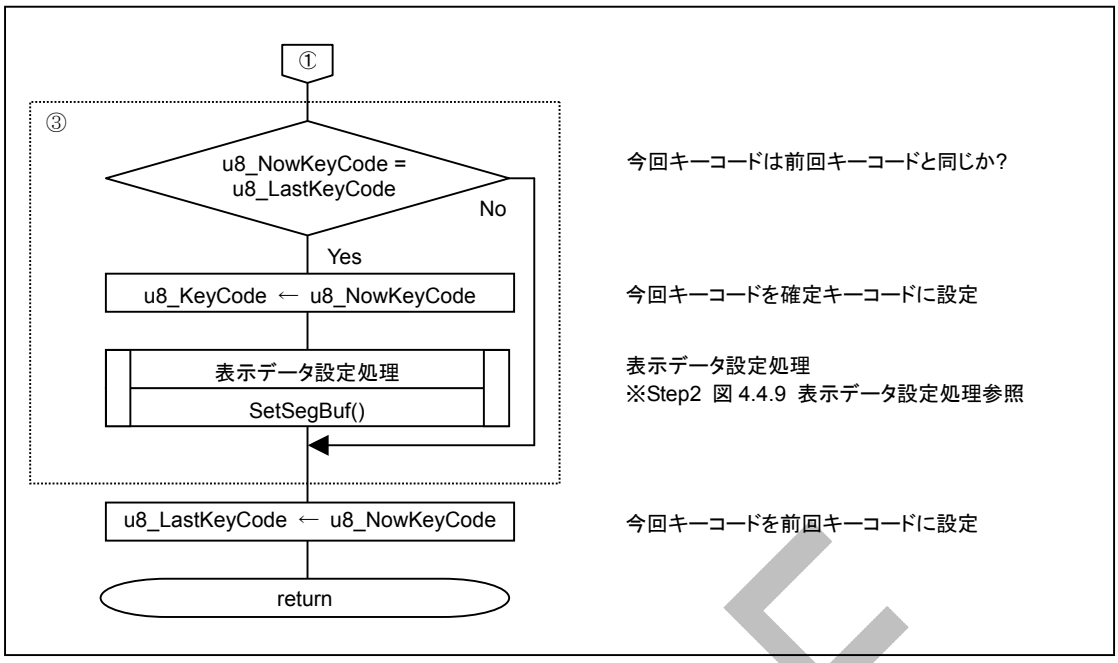


図 5.4.9 入力キー解読処理 - 2(サンプルソースはリスト 5.5.8 ⑤参照)

①全スキャンラインを読み込んだか判定し、全スキャンラインを読み込んでない場合は何もせずに処理を終了します。読み込んだ場合は以降の処理を行います。
②キースイッチが押下されているかスキャンラインの 1 ラインずつチェックし、4 ライン分チェックします。チェック対象スキャンラインの入力データを 1 ビットずつ“0”(キースイッチが押下されている)かどうか判定し、“0”であればキーコード判定をします。 キーコード判定ではキーコードが“0”の場合、他のキーは押下されていないとみなし、キーコード生成ワークデータを今回キーコードに設定します。そうでない場合、他のキースイッチも押下されている(多重押し)、または渡り押しされたとみなし、キーコードに”0xFF”を設定します。
③キーコードが 2 度一致しているか確認し、一致していたら確定キーコードとし、表示データを設定します。

5.4.9. キーコード取得処理

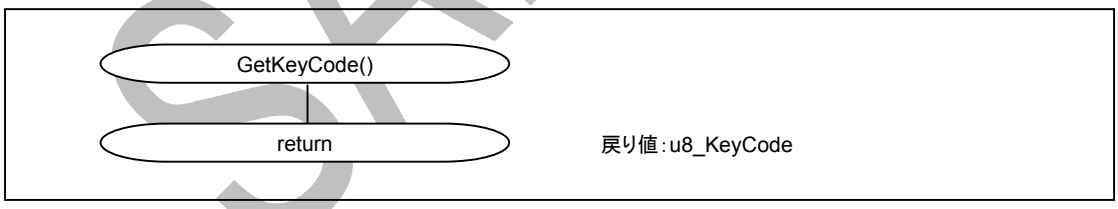


図 5.4.10 キーコード取得処理(サンプルソースはリスト 5.5.7 ③参照)

5.5. サンプルソース解説

キーマトリクス入力処理のサンプルソースファイルの一覧を下表に示します。

表 5.3 サンプルソースファイル一覧

No	ファイル名	内容	備考
1	ncrt0.a30	スタートアップルーチン	HEW プロジェクト生成時に自動生成されるファイル
2	main.c	メイン処理	本 Step で作成するファイル
3	initial.c	初期設定処理	本 Step で作成するファイル
4	disp.c	LED 表示処理	本 Step で作成するファイル
5	keyin.c	キーマトリクス入力処理	本 Step で作成するファイル
6	sect30.inc	セクション定義	HEW プロジェクト生成時に自動生成されるファイル
7	nc_define.inc	マクロシンボル定義	HEW プロジェクト生成時に自動生成されるファイル
8	sfr_r82d.h	R8C/2D SFR 定義	HEW プロジェクト生成時に自動生成されるファイル
9	common.h	共通ヘッダーファイル	本 Step で作成するファイル
10	initial.h	初期設定ヘッダーファイル	Step0 で作成したファイル
11	disp.h	LDE 表示処理ヘッダーファイル	本 Step で作成するファイル
12	keyin.h	キーマトリクス入力処理関連 ヘッダーファイル	本 Step で作成するファイル

サンプルソースで使用する静的変数について下表に示します。

表 5.4 静的変数

No	変数名	データ型	サイズ	内容
1	u8_ScanPointer	u8(unsigned char)	1byte	スキャンラインポインタ
2	u8_ScanBuf[SCAN_LINE_NUM]	u8(unsigned char)	4byte	キー入力データ SCAN_LINE_NUM=4
3	u8_KeyCode	u8(unsigned char)	1byte	確定キーコード
4	u8_LastKeyCode	u8(unsigned char)	1byte	前回確定キーコード

各ソースファイルの解説を以下に示します。

5.5.1. ncrt0.a30

R8C/Tiny 用スタートアップルーチンを含むアセンブラソースファイルです。

スタートアップルーチンとはリセットスタート後に最初に実行されるプログラムのことで、プログラムの実行に必要な環境設定をした後、main 関数を実行します。

このファイルは HEW でプロジェクトを作成すると自動生成されます。

5.5.2. main.c

リスト 5.5.1 サンプルソース main.c

```
/***********************************************************************************
 * System Name    : Starting Kit for R8C/2D
 * File Name      : main.c
 * Version        : Ver 0.02
 * Description    : Main process
 * CPU            : R8C/2D
 * Compiler       : M16C Series,R8C Family C Compiler Ver 5.45 Release 00
 * OS             : No Use
 * Author         : P.I.SYSTEM TOKYO
 * Notes          :
 ***********************************************************************************
 * Copyright,2010 P.I.SYSTEM CORP.
 ***********************************************************************************
* History         : Ver 0.01                start
***********************************************************************************/

/***********************************************************************************/
/*****    Include files                                                        *****/
/***********************************************************************************/
#include "common.h"
#ifdef DEF_DEBUG_STEP0
 #include "sfr_r82d.h"
#endif   // DEF_DEBUG_STEP0
#include "initial.h"

/***********************************************************************************/
/*****    Internal macro definition                                            *****/
/***********************************************************************************/

/***********************************************************************************/
/*****    Internal original data type definition                               *****/
/***********************************************************************************/

/***********************************************************************************/
/*****    Internal variants                                                    *****/
/***********************************************************************************/

/***********************************************************************************/
/*****   Prototype of internal function                                        *****/
/***********************************************************************************/
void main(void);

/***********************************************************************************
 * Function Name  : main
 * Description    : メイン処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  : Initialize()
 * Process        :
 * Notes :
***********************************************************************************/
void main(void)
{
  Initialize();                             // 初期設定処理

  while(1){
    if( ir_traic == 1 ){                    // 10ms 経過
      ir_traic = 0;                         // 割り込み要因ビットクリア
      Display();                            // LED 表示処理
      InputKey();                           // キーマトリクス入力処理
    }
  }
}
```

メイン処理
(図 5.4.1 のプログラム例)

①

5.5.3. initial.c

InitialSFR()に関しては Step2 のリスト 4.5.3、InitialClockSFR()に関しては Step0 のリスト 2.4.3、InitialTimerRASFR()に関しては Step1 のリスト 3.4.3 を参照して下さい。

リスト 5.5.2 サンプルソース initial.c - 1

```
/************************************************************************************
 * System Name    : Starting Kit for R8C/2D
 * File Name      : initial.c
 * Version        : Ver 0.01
 * Description    : Initialize process
 * CPU            : R8C/2D
 * Compiler       : M16C Series,R8C Family C Compiler Ver 5.45 Release 00
 * OS             : No Use
 * Author         : P.I.SYSTEM TOKYO
 * Notes          :
 ************************************************************************************
 * Copyright,2009 P.I.SYSTEM CORP.
 ************************************************************************************
 * History        : Ver 0.01                start
************************************************************************************/

/************************************************************************************/
/*****    Include files                                                         *****/
/************************************************************************************/
#include "sfr_r82d.h"
#include "common.h"

#define _GLOBAL_INITIAL_
#include "initial.h"
#include "keyin.h"
#include "disp.h"

/************************************************************************************/
/*****    Internal macro definition                                             *****/
/************************************************************************************/
#define DEF_TIMERRA_TRAPRE          ( 250-1 )         // TimerRA Pre
#define DEF_TIMERRA_TRA             ( 100-1 )         // TimerRA
/************************************************************************************/
/*****    Internal original data type definition                                *****/
/************************************************************************************/


/************************************************************************************/
/*****    Internal variants                                                     *****/
/************************************************************************************/


/************************************************************************************/
/*****   Prototype of internal function                                         *****/
/************************************************************************************/
void InitialSFR(void);
void InitialClockSFR(void);
void InitialPortSFR(void);
void InitialTimerRASFR(void);
/************************************************************************************
 * Function Name  : Initialize
 * Description    : 初期設定処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  : InitialSFR(),InitialDisplay()
 * Process        :
 * Notes          :
************************************************************************************/
void Initialize(void)
{
  asm(" FCLR I ");          // 割り込み禁止
  InitialSFR();             // SFR 初期設定処理
  InitialDisplay();         // LED 表示初期化処理
  InitialKeyInput();        // キーマトリクス入力初期化処理
}
```

リスト 5.5.3 サンプルソース initial.c - 2

```
/*****************************************************************************
 * Function Name  : InitialPortSFR
 * Description    : ポート SFR 初期設定処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  :
 * Process        :
*****************************************************************************/
void InitialPortSFR(void)
{
  p0 = 0x00;               // Port 0
  prc2 = 1;                // レジスタ書き込みプロテクト解除
  pd0 = 0x0F;              // I/O direction
//      00001111
//      ||||||||[ bit-no.PIN-no(IN/OUT)   : Port Description ]
//      |||||||+-----b0.pin66(OUT)          : 7seg-LED-a (initial L,active H)
//      ||||||+------b1.pin67(OUT)          : 7seg-LED-b (initial L,active H)
//      |||||+-------b2.pin68(OUT)          : 7seg-LED-c (initial L,active H)
//      ||||+--------b3.pin69(OUT)          : 7seg-LED-d (initial L,active H)
//      |||+---------b4.pin70(IN)                    : -
//      ||+----------b5.pin73(IN)                    : -
//      |+-----------b6.pin74(IN)                    : -
//      +------------b7.pin76(IN)                    : -
  prc2 = 0;                // レジスタ書き込みプロテクト
  p1 = 0x06;               // Port 1
  pd1 = 0x06;              // I/O direction
//      00000110
//      ||||||||[ bit-no.PIN-no(IN/OUT)   : Port Description ]
//      |||||||+-----b0.pin57(IN)                    : -
//      ||||||+------b1.pin56(OUT)          : 7seg-LED-com0 (initial H,active L)
//      |||||+-------b2.pin55(OUT)          : 7seg-LED-com1 (initial H,active L)
//      ||||+--------b3.pin54(IN)                    : -
//      |||+---------b4.pin34(IN)                    : -
//      ||+----------b5.pin33(IN)                    : -
//      |+-----------b6.pin32(IN)                    : -
//      +------------b7.pin31(IN)                    : -
  p2 = 0xF0;               // Port 2
  pd2 = 0xF0;              // I/O direction
//      11110000
//      ||||||||[ bit-no.PIN-no(IN/OUT)   : Port Description ]
//      |||||||+-----b0.pin30(IN)                    : -
//      ||||||+------b1.pin29(IN)                    : -
//      |||||+-------b2.pin28(IN)                    : -
//      ||||+--------b3.pin27(IN)                    : -
//      |||+---------b4.pin26(OUT)          : Key scan output0 (initial H,active L)
//      ||+----------b5.pin25(OUT)          : Key scan output1 (initial H,active L)
//      |+-----------b6.pin24(OUT)          : Key scan output2 (initial H,active L)
//      +------------b7.pin23(OUT)          : Key scan output3 (initial H,active L)
  p7 = 0x00;               // Port 7
  pd7 = 0xF0;              // I/O direction
//      11110000
//      ||||||||[ bit-no.PIN-no(IN/OUT)   : Port Description ]
//      |||||||+-----b0.pin65(IN)                    : Key input0
//      ||||||+------b1.pin64(IN)                    : Key input1
//      |||||+-------b2.pin63(IN)                    : Key input2
//      ||||+--------b3.pin62(IN)                    : Key input3
//      |||+---------b4.pin61(OUT)          : 7seg-LED-e (initial L,active H)
//      ||+----------b5.pin60(OUT)          : 7seg-LED-f (initial L,active H)
//      |+-----------b6.pin59(OUT)          : 7seg-LED-g (initial L,active H)
//      +------------b7.pin58(OUT)          : 7seg-LED-dp (initial L,active H)
  pur2 = 0x01;             // Port プルアップ設定
//       00000001
//       ||||||||
//       |||||||+-----P70-P73 pullup        : ON
//       ||||||+------P74-P77 pullup        : OFF
//       |||||+-------P80-P83 pullup        : OFF
//       ||||+--------P84-P87 pullup        : OFF
//       |||+---------P90-P93 pullup        : OFF
//       +++----------Not support           : "0" fix
}
```

ポート初期設定処理
(図 5.4.3 のプログラム例)

①

5.5.4. disp.c

SetComPort()に関しては Step2 のリスト 4.5.7、SetSegPort()、SetSegBuf()、SetDpSeg()に関しては Step2 のリスト 4.5.8 を参照して下さい。

リスト 5.5.4 サンプルソース disp.c - 1

```
/************************************************************************************
 * System Name    : Starting Kit for R8C/2D
 * File Name      : disp.c
 * Version        : Ver 0.01
 * Description    : display process
 * CPU            : R8C/2D
 * Compiler       : M16C Series,R8C Family C Compiler Ver 5.45 Release 00
 * OS             : No Use
 * Author         : P.I.SYSTEM TOKYO
*************************************************************************************
 * Copyright,2009 P.I.SYSTEM CORP.
 ************************************************************************************
 * History      : Ver 0.01         start
*************************************************************************************/

/************************************************************************************/
/*****    Include files                                                        *****/
/************************************************************************************/
#include "sfr_r82d.h"
#include "common.h"
#define _GLOBAL_DISP_
#include "disp.h"

/************************************************************************************/
/*****    Internal macro definition                                            *****/
/************************************************************************************/
#define SEG_NUM           2        // 7seg-LED 数
#define MAX_COM_POINT     2        // =SEG_NUM
#define u8_ComPort        p1
#define COM_0             0xFD     // COM0 のビット位置
#define COM0_OFF          0x02
#define COM0_ON           0xFD

#define COM_1             0xFB     // COM1 のビット位置
#define COM1_OFF          0x04
#define COM1_ON           0xFB
#define COM_OFF           (COM0_OFF | COM1_OFF)
#define COM_ON            (COM0_ON & COM1_ON)

#define u8_Seg0Port       p0
#define u8_Seg1Port       p7

#define SEG_off           0x00
#define SEG_a             0x01     // セグメント a のビット位置
#define SEG_b             0x02     // セグメント b のビット位置
#define SEG_c             0x04     // セグメント c のビット位置
#define SEG_d             0x08     // セグメント d のビット位置
#define SEG_e             0x10     // セグメント e のビット位置
#define SEG_f             0x20     // セグメント f のビット位置
#define SEG_g             0x40     // セグメント g のビット位置
#define SEG_dp            0x80     // セグメント dp のビット位置

#define DIGIT_0   ( SEG_a + SEG_b + SEG_c + SEG_d + SEG_e + SEG_f )     // "0"のセグメントパターン
#define DIGIT_1   ( SEG_b + SEG_c )                                     // "1"のセグメントパターン
#define DIGIT_2   ( SEG_a + SEG_b + SEG_d + SEG_e + SEG_g )             // "2"のセグメントパターン
#define DIGIT_3   ( SEG_a + SEG_b + SEG_c + SEG_d + SEG_g )             // "3"のセグメントパターン
#define DIGIT_4   ( SEG_b + SEG_c + SEG_f + SEG_g )                     // "4"のセグメントパターン
#define DIGIT_5   ( SEG_a + SEG_c + SEG_d + SEG_f + SEG_g )             // "5"のセグメントパターン
#define DIGIT_6   ( SEG_a + SEG_c + SEG_d + SEG_e + SEG_f + SEG_g )     // "6"のセグメントパターン
#define DIGIT_7   ( SEG_a + SEG_b + SEG_c + SEG_f )                     // "7"のセグメントパターン
#define DIGIT_8   ( SEG_a + SEG_b + SEG_c + SEG_d + SEG_e + SEG_f + SEG_g ) // "8"のセグメントパターン
#define DIGIT_9   ( SEG_a + SEG_b + SEG_c + SEG_d + SEG_f + SEG_g )     // "9"のセグメントパターン
#define DIGIT_a   ( SEG_a + SEG_b + SEG_c + SEG_e + SEG_f + SEG_g )     // "A"のセグメントパターン
#define DIGIT_b   ( SEG_c + SEG_d + SEG_e + SEG_f + SEG_g )             // "b"のセグメントパターン
#define DIGIT_c   ( SEG_d + SEG_e + SEG_g )                             // "c"のセグメントパターン
#define DIGIT_d   ( SEG_b + SEG_c + SEG_d + SEG_e + SEG_g )             // "d"のセグメントパターン
#define DIGIT_e   ( SEG_a + SEG_d + SEG_e + SEG_f + SEG_g )             // "E"のセグメントパターン
#define DIGIT_f   ( SEG_a + SEG_e + SEG_f + SEG_g )                     // "F"のセグメントパターン
#define DIGIT_bar ( SEG_g )                                             // "-"のセグメントパターン
#define DIGIT_dp  ( SEG_h )                                             // "."のセグメントパターン
```

リスト 5.5.5 サンプルソース disp.c - 2

```
/*****************************************************************************/
/*****    Internal original data type definition                         *****/
/*****************************************************************************/

/*****************************************************************************/
/*****    Internal variants                                              *****/
/*****************************************************************************/
static u8 u8_DispPointer;
u8 u8_SegBuf[SEG_NUM] = {0,0};
static u8 u8_DpFlg = 0;
/*****************************************************************************/
/*****    Prototype of internal function                                 *****/
/*****************************************************************************/
void SetComPort(u8 data);
void SetSegPort(u8 data);
/*****************************************************************************/
/*****    Data table                                                     *****/
/*****************************************************************************/
const static u8 tbl_Com[]={                 // COM ポートテーブル
         COM_0,   COM_1
};
const static u8 tbl_Digit[]={               // セグメントパターンテーブル
         DIGIT_0, DIGIT_1, DIGIT_2, DIGIT_3,
         DIGIT_4, DIGIT_5, DIGIT_6, DIGIT_7,
         DIGIT_8, DIGIT_9, DIGIT_a, DIGIT_b,
         DIGIT_c, DIGIT_d, DIGIT_e, DIGIT_f
};
/*****************************************************************************
 * Function Name  : Display
 * Description    : LED 表示処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  : SetComPort(u8 data),SetSegPort(u8 data)
 * Process        :
*****************************************************************************/
void Display(void)
{
  SetComPort(COM_OFF);                                  // COM 出力 OFF
  SetSegPort(tbl_Digit[u8_SegBuf[u8_DispPointer]]);     // SegmentPort 出力設定
  SetComPort(tbl_Com[u8_DispPointer]);                  // COM 出力 ON
  u8_DispPointer++;                                     // COM ラインカウンタインクリメント
  if(u8_DispPointer >= MAX_COM_POINT){
    u8_DispPointer = 0;                                 // COM ラインカウンタ初期化
  }
}
/*****************************************************************************
 * Function Name  : InitialDisplay
 * Description    : LED 表示初期化処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  : SetComPort(u8 data),SetSegPort(u8 data)
 * Process        :
 * Notes          :
*****************************************************************************/
void InitialDisplay(void)
{
  u8 u8_c;
  u8_DispPointer = 0;                           // COM 表示ポインタ初期化
  for(u8_c=0;u8_c < SEG_NUM;u8_c++){
    u8_SegBuf[u8_c] = 0;    // セグメント表示データ初期化
  }
  u8_DpFlg = 0;                                 // 小数点表示フラグ初期化
  SetComPort(COM_OFF);              // グリッド出力 OFF
  SetSegPort(SEG_off);              // セグメント出力 OFF}
}
```

LED 表示処理
(図 5.4.5 のプログラム例)

①

5.5.5. keyin.c

リスト 5.5.6 サンプルソース keyin.c - 1

```
/************************************************************************************
* System Name     : Starting Kit for R8C/2D
 * File Name      : keyin.c
 * Version        : Ver 0.01
 * Description    : Key Input process
 * CPU            : R8C/2D
 * Compiler       : M16C Series,R8C Family C Compiler Ver 5.45 Release 00
 * OS             : No Use
 * Author         : P.I.SYSTEM TOKYO
************************************************************************************
 * Copyright,2009 P.I.SYSTEM CORP.
 ***********************************************************************************
 * History        : Ver 0.01                start
************************************************************************************/

/************************************************************************************/
/*****    Include files                                                        *****/
/************************************************************************************/
#include "sfr_r82d.h"
#include "common.h"

#define _GLOBAL_KEYIN_
#include "keyin.h"
#include "disp.h"

/************************************************************************************/
/*****    Internal macro definition                                            *****/
/************************************************************************************/
#define SCAN_LINE_NUM      4        // スキャンライン数
#define MAX_SCAN_POINT     4        // =SCAN_LINE_NUM
#define BUTTON_NUM         4        // 1 ラインのボタン数

#define u8_KeyInPort       p7
#define u8_KeyInAssign     0x0F     // 入力ポートの割り当ては下位(p7_0～p7_3)
#define u8_KeyJudgePort    0x01     // 最下位 bit より判定
#define u8_KeyScanPort     p2
#define SCAN_LINE0         0xE0
#define SCAN_LINE1         0xD0
#define SCAN_LINE2         0xB0
#define SCAN_LINE3         0x70

/************************************************************************************/
/*****    Internal original data type definition                               *****/
/************************************************************************************/

/************************************************************************************/
/*****    Internal variants                                                    *****/
/************************************************************************************/
static u8 u8_ScanPointer;
static u8 u8_ScanBuf[SCAN_LINE_NUM];
static u8 u8_KeyCode;
static u8 u8_LastKeyCode;

/************************************************************************************/
/*****    Prototype of internal function                                       *****/
/************************************************************************************/
void ScanKey(void);
void DecodeKey(void);

/************************************************************************************/
/*****    Data table                                                           *****/
/************************************************************************************/
const static u8 tbl_ScanLine[]={            // キーKey スキャンラインテーブル
   SCAN_LINE0,     SCAN_LINE1,       SCAN_LINE2,       SCAN_LINE3
};
```

リスト 5.5.7 サンプルソース keyin.c - 2

```
/*****************************************************************************
 * Function Name  : InputKey
 * Description    : キーマトリクス入力処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  : ScanKey(),DecodeKey()
 * Process        :
 * Notes          :
*****************************************************************************/
void InputKey(void)
{
  ScanKey();                // キー入力読込処理
  DecodeKey();              // 入力キー解読処理
}

/*****************************************************************************
 * Function Name  : InitialKeyInput
 * Description    : キーマトリクス入力初期化処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  :
 * Process        :
 * Notes          :
*****************************************************************************/
void InitialKeyInput(void)
{
  u8 u8_c;
  u8_ScanPointer = 0;
  for(u8_c=0;u8_c < SCAN_LINE_NUM;u8_c++){
    u8_ScanBuf[u8_c] = 0;
  }
  u8_KeyCode = 0;
  u8_LastKeyCode = 0;
}
/*****************************************************************************
 * Function Name  : GetKeyCode
 * Description    : キーコード取得処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : u8
 * Use Functions  :
 * Process        :
 * Notes          :
*****************************************************************************/
u8 GetKeyCode(void)
{
  return u8_KeyCode;
}
```

キーマトリクス入力処理
(図 5.4.6 のプログラム例)

①

キーマトリクス入力初期化処理
(図 5.4.4 のプログラム例)

②

キーコード取得処理
(図 5.4.10 のプログラム例)

③

リスト 5.5.8 サンプルソース keyin.c - 3

```
/*****************************************************************************
 * Function Name  : ScanKey
 * Description    : キー入力読込処理
 * Inputs         :
 *   Parameters   : なし
 *   Globals      : なし
 * Outputs        :
 *   Parameters   : なし
 *   Globals      : なし
 *   Return       : なし
 * Use Functions  :
 * Process        :
 * Notes          :
*****************************************************************************/
void ScanKey(void)
{
  u8_ScanBuf[u8_ScanPointer] = (u8_KeyInPort & u8_KeyInAssign);        // キー入力ポートを読み込む
  u8_ScanPointer++;                                                    //次のスキャンポイントを設定する

  if( u8_ScanPointer >= MAX_SCAN_POINT ){                    // 全てのスキャンポートをスキャンしたか？
    u8_ScanPointer = 0;                                                // スキャンポート初期化
  }

  u8_KeyScanPort = tbl_ScanLine[u8_ScanPointer];                       // 次のスキャンポート出力
}

/*****************************************************************************
 * Function Name : DecodeKey
 * Description   : 入力キー解読処理
 * Inputs        :
 *   Parameters  : なし
 *   Globals     : なし
 * Outputs       :
 *   Parameters  : なし
 *   Globals     : なし
 *   Return      : なし
 * Use Functions : SetSegBuf(u8 data)
 * Process       :
 * Notes         :
*****************************************************************************/
void DecodeKey(void)
{
  u8 u8_KeyCodeWork = 0, u8_NowKeyCode = 0, u8_InputData = 0;
  u8 u8_cnt1 = 0, u8_cnt2 = 0;
  if( u8_ScanPointer == 0 ){                                 // 全てのスキャンラインのデータ      か？
    /* スキャンライン分ループする */
    for( u8_cnt1 = u8_KeyCodeWork = u8_NowKeyCode = 0; u8_cnt1 != SCAN_LINE_NUM; ++u8_cn
      /* 1スキャンラインの釦数(4)分ループする */
      for( u8_cnt2 = 0 , u8_InputData = u8_ScanBuf[u8_cnt1] ; u8_cnt2 != BUTTON_NUM ; ++u8_c      ){
        ++u8_KeyCodeWork;                                        // キーコードを+1 する
        if( (u8_InputData & u8_KeyJudgePort) == 0 ){             // キー入力があるか？
          if( u8_NowKeyCode == 0 ){                              // 既にキー入力があったか？(多重キー)
            u8_NowKeyCode = u8_KeyCodeWork;                      // キーコードをセーブする
          }else{
            u8_NowKeyCode = 0xff;                                // 多重キーコードを設定
          }

          if(u8_LastKeyCode == 0xff){                            // 渡り押しか？
            u8_NowKeyCode = 0xff;
          }
        }
        u8_InputData >>= 1;                                      // 次の入力ビットをセットする
      }
    }

    if( u8_NowKeyCode == u8_LastKeyCode ){                       // チャッタチェック
      u8_KeyCode = u8_NowKeyCode;                                // 新しいキーコードを格納する
      SetSegBuf(u8_KeyCode);
    }
    u8_LastKeyCode = u8_NowKeyCode;                              // 今回の入力キーコードを設定する
  }
}
```

キー入力読込処理
(図 5.4.7 のプログラム例)

④

入力キー解読処理
(図 5.4.8、図 5.4.9 のプログラム例)

⑤

5.5.6. sect30.inc

セクション定義ファイルです。
セクションとはプログラムやデータを格納するエリアの単位です。
このファイルは HEW でプロジェクトを作成すると自動生成されます。

5.5.7. nc_define.inc

マクロシンボル定義ファイルです。
ヒープサイズ、スタックサイズ等のスタートアップルーチンで使用するマクロの定義が記述されています。
このファイルは HEW でプロジェクトを作成すると自動生成されます。

5.5.8. sfr_r82d.h

R8C/2D SFR 定義ファイルです。
このファイルは HEW でプロジェクトを作成すると自動生成されます。

5.5.9. common.h

Step0 のリスト 2.4.5、リスト 2.4.6 を参照して下さい。
ただし、「DEF_DEBUG_STEP0」、「H_SPEED_OSC」の記述は削除してください。

5.5.10. initial.h

Step0 のリスト 2.4.7 を参照して下さい。

5.5.11. disp.h

Step2 のリスト 4.5.10 を参照して下さい。

5.5.12. keyin.h

リスト 5.5.9 サンプルソース keyin.h

```
/***********************************************************************************
 * System Name  : Starting Kit for R8C/2D
 * File Name    : keyin.h
 * Version      : Ver 0.01
 * Description  :
 * CPU          : R8C/2D
 * Compiler     : M16C Series,R8C Family C Compiler Ver 5.45 Release 00
 * OS           : No Use
 * Author       : P.I.SYSTEM TOKYO
 * Notes        :
 ***********************************************************************************
 * Copyright,2009 P.I.SYSTEM CORP.
 ***********************************************************************************
 * History      : Ver 0.01                start
***********************************************************************************/
#ifndef _KEYIN_H_
#define _KEYIN_H_

#ifdef _GLOBAL_KEYIN_
#define KEYIN_EXT
#else // #ifndef _GLOBAL_KEYIN_
#define KEYIN_EXT                extern
#endif // _GLOBAL_KEYIN_

/***********************************************************************************/
/*****    Global macro definition                                          *****/
/***********************************************************************************/
#define KEYIN_NOKEY      0x00
#define KEYIN_SW1        0x01
#define KEYIN_SW2        0x02
#define KEYIN_SW3        0x03
#define KEYIN_SW4        0x04
#define KEYIN_SW5        0x05
#define KEYIN_SW6        0x06
#define KEYIN_SW7        0x07
#define KEYIN_SW8        0x08
#define KEYIN_SW9        0x09
#define KEYIN_SW10       0x0A
#define KEYIN_SW11       0x0B
#define KEYIN_SW12       0x0C
#define KEYIN_SW13       0x0D
#define KEYIN_SW14       0x0E
#define KEYIN_SW15       0x0F
#define KEYIN_SW16       0x10

/***********************************************************************************/
/*****    Global original data type definition                             *****/
/***********************************************************************************/


/***********************************************************************************/
/*****    Global variants                                                  *****/
/***********************************************************************************/


/***********************************************************************************/
/*****   Prototype of grobal function                                      *****/
/***********************************************************************************/
KEYIN_EXT void InputKey(void);
KEYIN_EXT void InitialKeyInput(void);
KEYIN_EXT u8 GetKeyCode(void);

#endif // KEYIN H
```

5.6. 演習

5.6.1. 演習課題

フローチャートとサンプルソースを参考にプログラムを作成し、入力したキースイッチに対応するキーコードが生成され、確定したキースイッチが受け付けられている事を確認して下さい。

表 5.5 動作確認におけるチェックポイント

No	チェック内容	結果
1	10ms 周期でスキャン出力が切り替わっている事が確認できたか	
2	押下したキースイッチに対応するキーコードが 7 セグメント LED で表示できたか	

5.6.2. HEW の使用方法

HEW の使用方法に関しては Step0 の 2.5.3.を参照して下さい。

5.6.2.1. サンプルプログラム実行手順

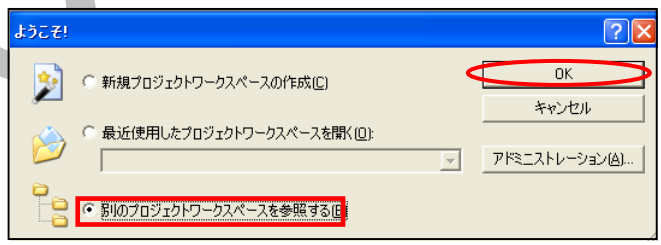
本テキスト付属 CD-ROM には本 Step のサンプルプログラムが収録されています。

サンプルプログラムは以下の環境で開発しています。 各ツールのバージョンが以下より古い場合は、必ず最新バージョンにしてください。 バージョンは HEW のメニューの「ヘルプ」→「High-performance Embedded Workshop のバージョン情報」を選択し、「High-performance Embedded Workshop について」ウィンドウの「詳細」ボタンをクリックすることで確認できます。

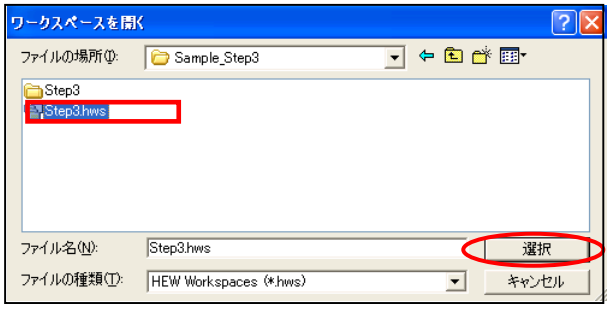
HEW	: Version 4.06.00
M16C Series, R8C Family C Compiler	: V.5.45 Release 00
E8a Emulator Software	: V.1.03 Release 02

サンプルプログラムを実行するまでの手順を以下に示します。

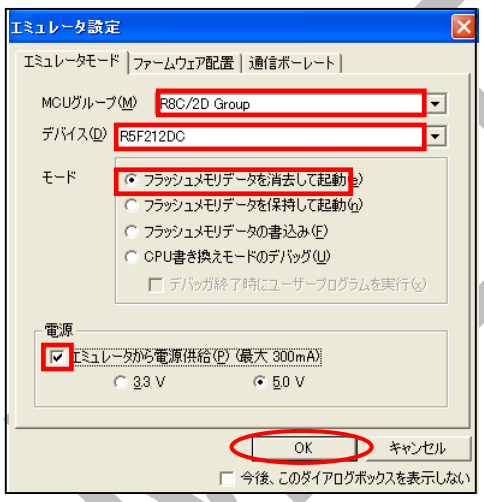
(1) 本テキスト付属 CD-ROM をセットし、エクスプローラでこの CD-ROM を開きます。
(2) “**Step3**”フォルダを開きます。
(3) “**Sample_Step3**”フォルダをサンプルプログラムを実行する PC の“**C:¥WorkSpace**”にコピーします。
(4) P101MS0008-A と E8a を接続し、E8a を PC に接続します。
(5) スタートメニューより「すべてのプログラム」→「Renesas」→「High-performance Embedded Workshop」→「High-performance Embedded Workshop」を選択し、HEW を起動します。
(6) 「別のプロジェクトワークスペースを参照する」を選択し、「OK」をクリックします。



(7) “**C:¥WorkSpace¥Sample_Step3¥R8C2D_StarterKit.hws**”ファイルを選択し、「選択」をクリックします。



(8) MCU グループは「R8C/2D Group」を選択します。本章ではマイコンは型名「R5F212DC」を使用することとするためデバイスは「R5F212DC」を選択します。
今回はデバッガで動作確認するため、モードは「フラッシュメモリデータを消去して起動」を選択します。
エミュレータから電源供給チェックボックスにチェックし、「5.0V」を選択します。
入力が済みましたら「OK」をクリックして下さい。



(9) 「すべてをビルド」ボタンをクリックします。
(10) ビルドが正常に完了すると、画面下の「Build」ウィンドウに「Build Finished 0 Errors, 0 Warnings」と表示されます。
(11) 以降の手順は Step0　2.5.3.5.の(10)以降と同様です。

5.6.3. 動作確認波形

プログラムが意図通りに動作している場合、以下波形図のように 10ms 間隔でライン出力が切り換わる事が確認できます。

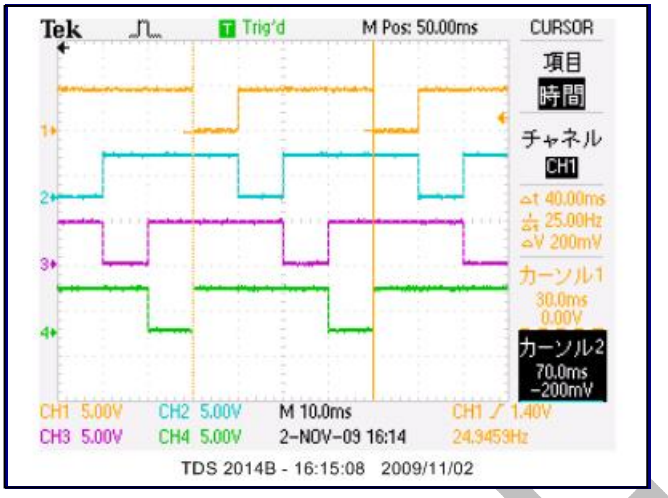


図 5.6.1 スキャン出力波形

(CH1:Key scan out0、CH2:Key scan out1、CH3:Key scan out2、CH4：Key scan out3)

プログラムが意図通りに動作している場合、SW1、SW10、SW11、SW16 押下時は以下波形図のように各入力が 40ms 間隔で 10ms 間“L”出力となることが確認できます。

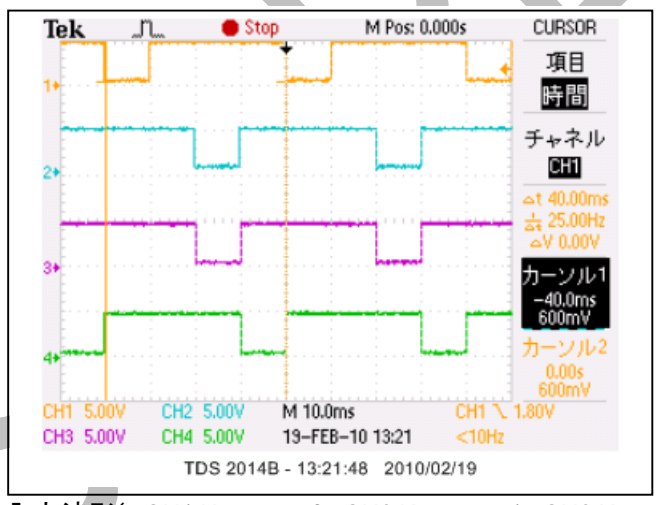


図 5.6.2 SW1 押下時キー入力波形(CH1:Key input0、CH2:Key input1、CH3:Key input2、CH4：Key input3)

改訂履歴

Rev	日付	改定内容
sample-1.0	2010.02.22	サンプル版

株式会社ピーアイシステム

本社　〒532-0004 大阪府大阪市淀川区西宮原 1-8-29 テラサキ第二ビル
TEL：06-6150-3001　FAX：06-6150-3005
東京支店　〒105-0013 東京都港区浜松町 2-9-6 浜松町エムプレスビル
TEL：03-6402-7268　FAX：03-3432-6516
http://www.pis.co.jp