2.12 クロック同期式シリアルデータ受信

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

仕様

(1) 図1に示すようにクロック同期式シリアル転送機能を使用して、4バイトの8ビットデータの受信を行ないます。
(2) 転送クロックは、外部クロックを使用します。
(3) 受信するデータのデータ長は8ビットで、データの最下位ビットから受信するLSBファースト方式による受信を行います。

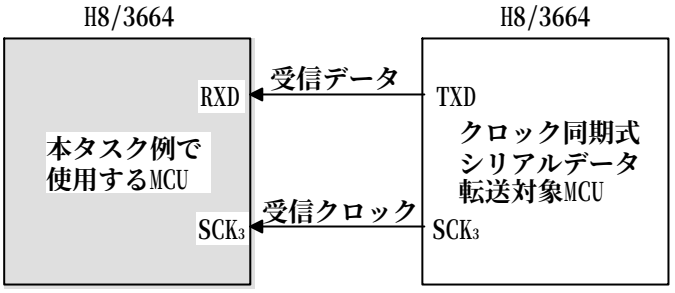


図1　クロック同期式シリアルデータ受信

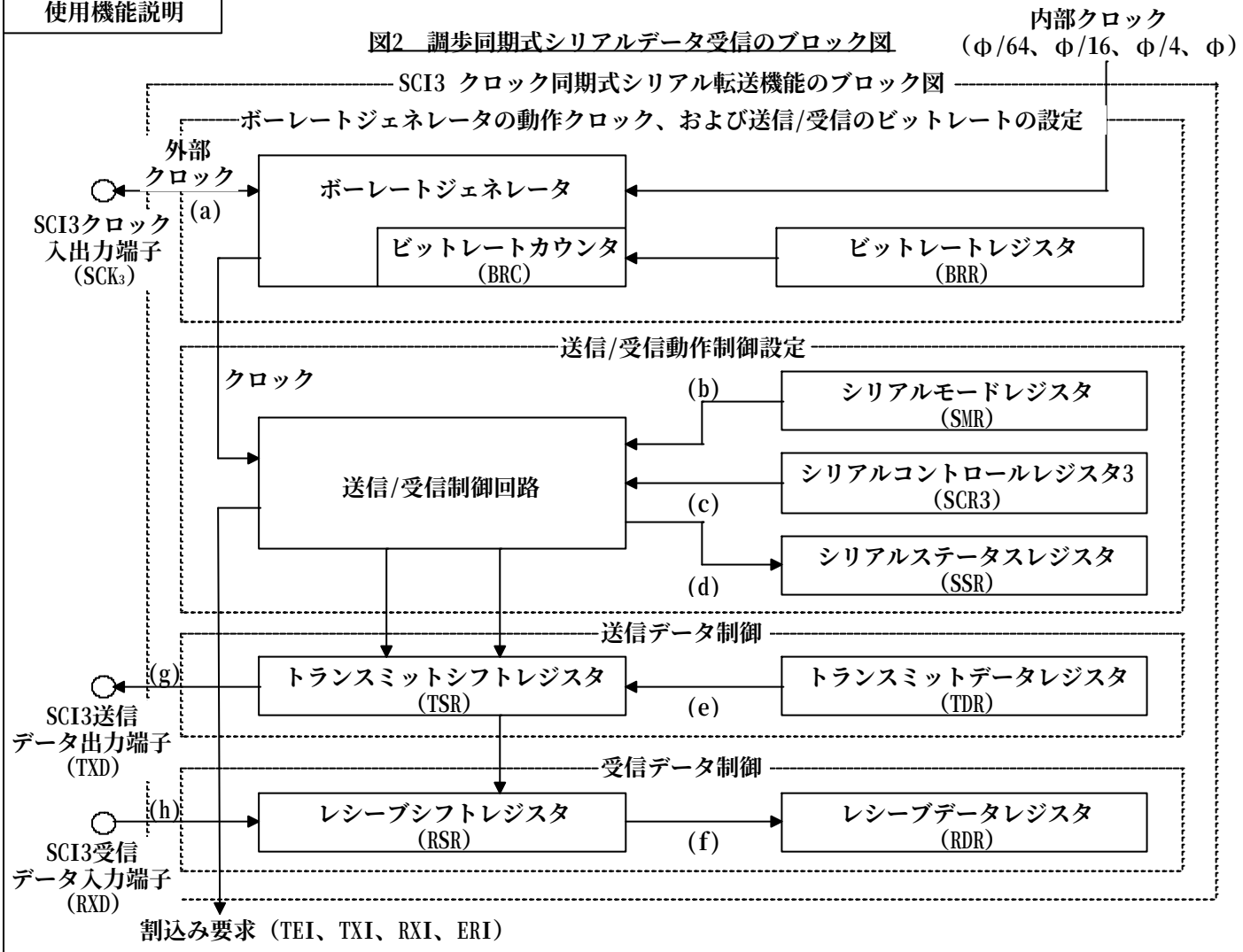
使用機能説明

(1) 本タスク例では、シリアルコミュニケーションインタフェース（SCI：Serial Communication Interface）を使用して、クロック同期式のシリアルデータの受信を行ないます。図2にクロック同期式シリアルデータ受信のブロック図を示します。以下にクロック同期式シリアルデータ受信のブロック図について説明します。。
- 受信エラーの検出には、オーバランエラーのみを行います。
- クロック同期式モードではデータ長は8ビットになります。
- レシーブシフトレジスタ（RSR）は、シリアルデータを受信するためのレジスタです。RSRにRXD端子から入力されたシリアルデータを、LSB（ビット0）から受信した順にセットしパラレルデータに変換します。1バイトのデータを受信すると、データは自動的にRDRへ転送されます。CPUからRSRを直接リード/ライトすることはできません。
- レシーブデータレジスタ（RDR）は、受信したシリアルデータを格納する8ビットのレジスタです。1バイトのデータの受信が終了すると、受信したデータをRSRからRDRへ転送し、受信動作を完了します。その後、RSRは受信可能となります。RSRとRDRはダブルバッファになっているため連続した受信動作が可能です。RDRは受信専用レジスタなのでCPUからライトできません。
- トランスミットシフトレジスタ（TSR）は、シリアルデータを送信するためのレジスタです。TDRから送信データをいったんTSRに転送し、LSB（ビット0）から順にTXD端子に送出することでシリアルデータ送信を行ないます。1バイトのデータを送信すると、自動的にTDRからTSRへ次の送信データを転送し、送信を開始します。ただし、TDRにデータが書き込まれていない（TDREに"1"がセットされている）場合にはTDRからTSRへのデータ転送は行ないません。CPUからTSRを直接リード/ライトすることはできません。
- トランスミットデータレジスタ（TDR）は、送信データを格納する8ビットのレジスタです。
  TSRの"空"を検出すると、TDRに書き込まれた送信データをTSRに転送し、シリアルデータ送信を開始します。TSRのシリアルデータ送信中に、TDRに次の送信データをライトしておくと、連続送信が可能です。
  TDRは、常にCPUによるリード/ライトが可能です。
- シリアルモードレジスタ（SMR）は、シリアルデータ通信フォーマットの設定と、内蔵ボーレートジェネレータのクロックソースを選択するための8ビットのレジスタです。
- シリアルコントロールレジスタ3（SCR3）は、送信/受信動作および送信/受信クロックソース の選択を行なう8ビットのレジスタです。
- シリアルステータスレジスタ（SSR）は、SCI3のステータスフラグと送受信マルチプロセッサビットで構成されています。TDRE、RDRF、OER、PER、FERはクリアのみ可能です。
- 転送クロックは、8種類の内部クロックと外部クロックから選択できます。内部クロックを選択した場合は、SCK3端子は出力端子となります。クロック連続出力モードに設定すると選択したクロックをSCK3端子から連続して出力します。外部クロックを選択した場合は、SCK3端子はクロック入力端子となります。
- 本タスク例では、転送クロックソースを外部クロックに設定しています。
- SCI3の転送フォーマットは8ビットのデータを選択可能です。データの最下位ビットから送受信されるLSBファースト方式による転送を行ないます。送信データは、転送クロックの立ち下がりから次の立ち上がりまで出力されます。また、受信データは転送クロックの立ち上がりで取り込まれます。
- 本タスク例では、動作モードを8ビットモードに設定し、8ビットのデータ受信を行ないます。
- SCI3クロック（SCK3）は、SCI3のクロック入出力端子です。
- SCI3レシーブデータ入力（RXD）は、SCI3の受信データの入力端子です。

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

使用機能説明

図2　調歩同期式シリアルデータ受信のブロック図



【注】(a)外部クロックを受信する。
(b)シリアルデータ通信フォーマットの設定を行う。
(c)送信/受信動作、クロック同期式モードでのクロック入力の選択を行う。
(d)ステータスフラグ（レシーブデータレジスタフル、オーバランエラー）によりSCI3の動作状態を示す。
(e)TSRの"空"を検出することにより、TDRに書き込まれた送信データをTSRに転送。
(f)1バイトのデータの受信が終了すると、受信したデータをRSRからRDRへ転送。
(g)送信データを送信する。
(h)受信データを受信する。

(2) 表1に本タスク例の機能割付け示します。表1に示すように機能を割り付け、クロック同期式シリアルデータ受信を行ないます。

表1　機能割付け

機能	機能割付け
RSR	シリアルデータを受信するためのレジスタ
RDR	受信データを格納するレジスタ
SMR	シリアルデータ通信フォーマットの設定
SSR	SCI3の動作状態を示すステータスフラグ
SCR3	受信動作、SCK3の端子機能をクロック入力端子設定
SCK3	SCI3のクロック入力端子
RXD	SCI3の受信データ入力端子

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

動作原理

(1) 図3に動作原理を示します。図3に示すようなハードウェア処理、およびソフトウェア処理によりクロック同期式シリアルデータ受信を行ないます。

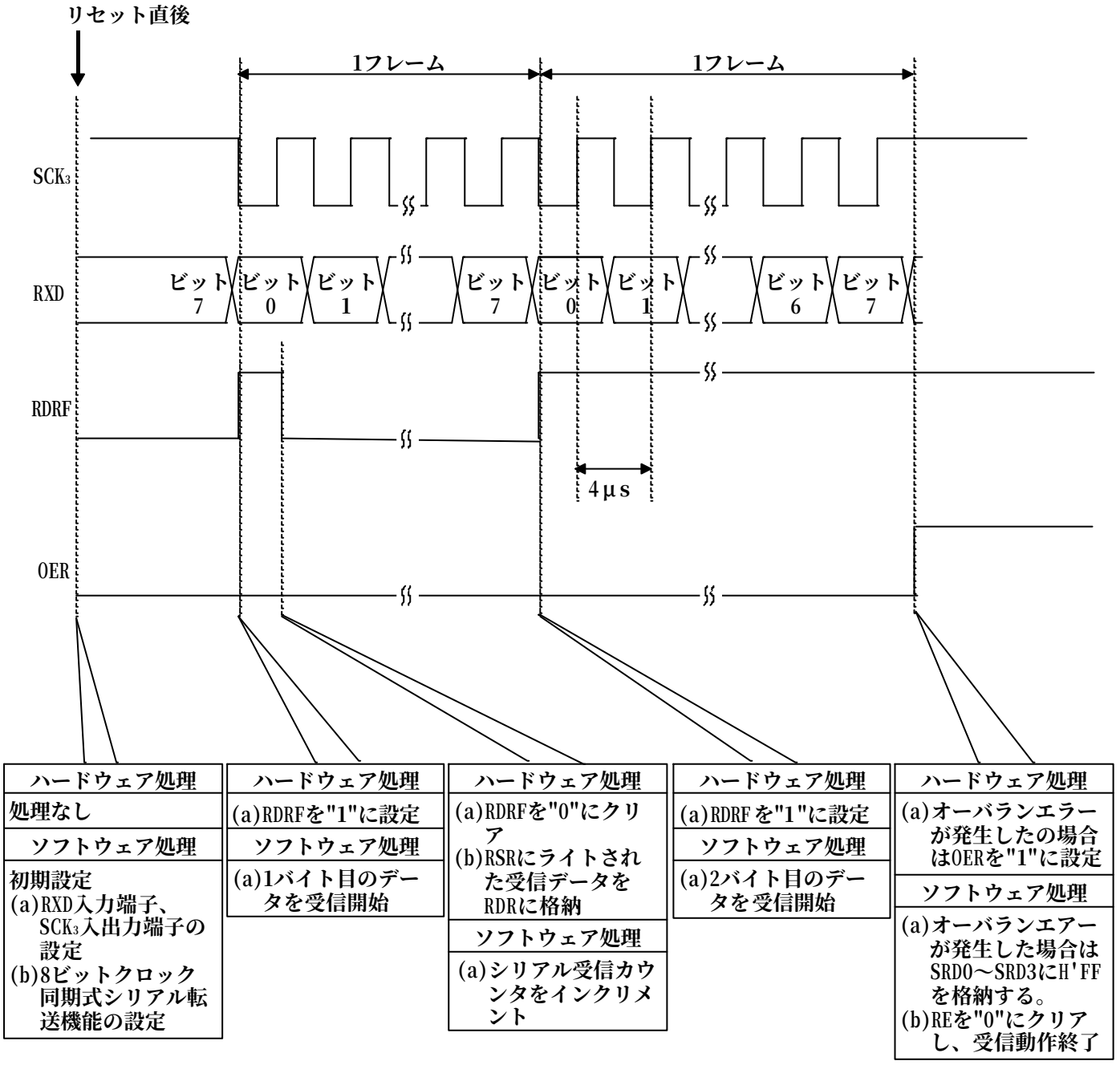


図3　クロック同期式シリアル受信の動作原理

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

ソフトウェア説明

(1) モジュール説明

表2に本タスク例におけるモジュール説明を示します。

表2　モジュール説明

モジュール名	ラベル名	機能
メインルーチン	main	クロック同期式シリアルデータ受信の設定、受信エラーが発生した場合にはSRD0～SRD3に'HFFを格納、4バイトのデータを受信したところで終了

(2) 引数の説明

表3に本タスク例で使用する引数を示します。

表3　引数の説明

引数名	機能	使用モジュール名	データ長	入出力
SRD0～SRD3	クロック同期式シリアル受信データ	メインルーチン	1バイト	入力

(3) 使用内部レジスタ説明

表4に本タスク例における使用内部レジスタ説明を示します。

表4　使用内部レジスタ説明

レジスタ名		機能	アドレス	設定値
SMR	COM	シリアルモードレジスタ（コミュニケーションモード） ：COM="1"のとき、コミュニケーションモードをクロック同期式モードに設定	H'FFA8 ビット7	1
	MP	シリアルモードレジスタ（マルチプロセッサモード） ：MP="0"のとき、マルチプロセッサ通信機能を禁止	H'FFA8 ビット2	0
SCR3	RE	シリアルコントロールレジスタ3 （レシーブイネーブル） ：RE="1"のとき、受信動作を許可	H'FFAA ビット4	1
	CKE1 CKE0	シリアルコントロールレジスタ3 （クロックイネーブル1、0） ：CKE1="1"、CKE0="0"のとき、クロック同期式モードにおいてクロックソースを外部クロック、SCK_3端子機能をクロック入力に設定	H'FFAA ビット1 ビット0	CKE1="1" CKE0="0"
RDR		レシーブデータレジスタ ：受信データを格納する8ビットのレジスタ	H'FFAD	-
SSR	RDRF	シリアルステータスレジスタ（レシーブデータレジスタフル） ：RDRF="0"のとき、RDRに受信データが格納されていない。 ：RDRF="1"のとき、RDRに受信データが格納されている。	H'FFAC ビット6	1
	OER	シリアルステータスレジスタ（オーバランエラー） ：OER="0"のとき、受信中、または受信を完了 ：OER="1"のとき、受信時にオーバランエラーが発生	H'FFAC ビット5	0

(4) 使用RAM説明

表5に本タスク例における使用RAM説明を示します。

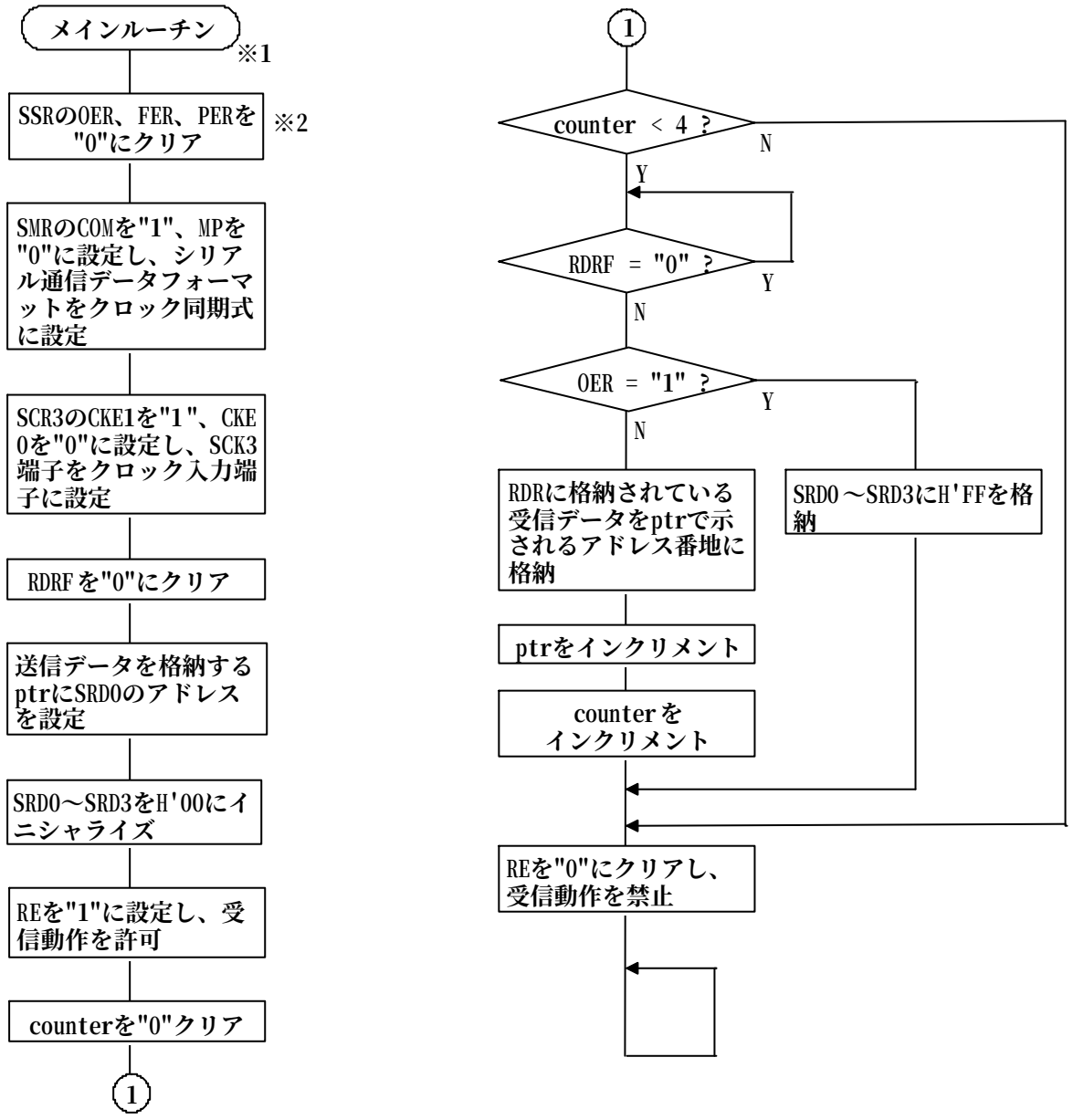
表5　使用RAM説明

ラベル名	機能	アドレス	使用モジュール名
SRD0	クロック同期式シリアル受信データの1バイト目を格納	H'FB80	メインルーチン
SRD1	クロック同期式シリアル受信データの2バイト目を格納	H'FB81	メインルーチン
SRD2	クロック同期式シリアル受信データの3バイト目を格納	H'FB82	メインルーチン
SRD3	クロック同期式シリアル受信データの4バイト目を格納	H'FB83	メインルーチン
counter	クロック同期式シリアル受信動作を4カウントする8ビットカウンタ	H'FB84	メインルーチン

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

フローチャート

(a) メインルーチン



※1：本例ではスタックポインタはINIT.SRC（アセンブリ言語）で設定してあります。
※2：クロック同期式（受信）ではOER、FER、PERを"0"クリアにしておく必要がある。

クロック同期式シリアルデータ受信	使用機能	SCI3 : クロック同期式シリアル転送機能

プログラムリスト

INIT.SRC（プログラムリスト）

```
        .EXPORT  _INIT
        .IMPORT  _main
;
        .SECTION    P,CODE
_INIT:
        MOV.W    #H'FF80,R7
        LDC.B    #B'10000000,CCR
        JMP              @_main
;
        .END
```

```
/*******************************************************/
/*                                                     */
/*  H8/300H Tiny Series -H8/3664-                      */
/*  Application Note                                   */
/*                                                     */
/*  'Synchronous Serial Data Receotion'                */
/*                                                     */
/*  Function                                           */
/*  : Serial Communication Interface                   */
/*     Synchronous Serial Interface                    */
/*     -Receiving                                      */
/*                                                     */
/*  External Clock : 16MHz                             */
/*  Internal Clock : 16MHz                             */
/*  Sub Clock      : 32.768kHz                         */
/*                                                     */
/*******************************************************/

#include    <machine.h>

/*******************************************************/
/*  Symbol Defnition                                   */
/*******************************************************/
struct BIT {
    unsigned char    b7:1;      /* bit7 */
    unsigned char    b6:1;      /* bit6 */
    unsigned char    b5:1;      /* bit5 */
    unsigned char    b4:1;      /* bit4 */
    unsigned char    b3:1;      /* bit3 */
    unsigned char    b2:1;      /* bit2 */
    unsigned char    b1:1;      /* bit1 */
    unsigned char    b0:1;      /* bit0 */
};

#define    SMR_BIT (*(struct BIT *)0xFFA8)            /* Serial Mode Register       */
#define    COM     SMR_BIT.b7                         /* Communication Mode         */
#define    MP      SMR_BIT.b2                         /* Multiprocesor Mode         */
#define    SCR3_BIT (*(struct BIT *)0xFFAA)           /* Serial Control Register 3  */
#define    RE      SCR3_BIT.b4                        /* Receive Enable             */
#define    CKE1    SCR3_BIT.b1                        /* Clock Enable 1             */
#define    CKE0    SCR3_BIT.b0                        /* Clock Enable 0             */
#define    SSR_BIT (*(struct BIT *)0xFFAC)            /* Serial Status Register     */
#define    RDRF    SSR_BIT.b6                         /* Receive Data Register Full */
#define    OER     SSR_BIT.b5                         /* Overrun Erorr              */
#define    FER     SSR_BIT.b4                         /* Framing Erorr              */
#define    PER     SSR_BIT.b3                         /* Parity Erorr               */
#define    RDR     *(volatile unsigned char *)0xFFAD  /* Receive data Register      */

/*******************************************************/
/*  関数定義                                           */
/*******************************************************/
extern  void    INIT( void );                          /* SP Set                     */
void    main    ( void );
```

クロック同期式シリアルデータ受信	使用機能	SCI3 ： クロック同期式シリアル転送機能

プログラムリスト

```
/************************************************************/
/*   RAM Allocation                                         */
/************************************************************/
    unsigned char   SRD[4];
    unsigned char   counter;

/************************************************************/
/*  Vector Address                                          */
/************************************************************/
#pragma section     V1                          /* VECTOR SECTOIN SET         */
void (*const VEC_TBL1[])(void) = {
/* 0x00 - 0x0f */
    INIT                                        /* 00 Reset                   */
};

#pragma section                                 /* P                          */
/************************************************************/
/*   Main Program                                           */
/************************************************************/
void main ( void )
{

    unsigned char    *ptr;

    OER = 0;                                    /* Clear OER                  */
    FER = 0;                                    /* Clear FER                  */
    PER = 0;                                    /* Clear PER                  */

    COM = 1;                                    /* Initialize Communication Mode          */
    MP = 0;                                     /* Initialize Multiprocesor Mode          */

    CKE1 = 1;                                   /* Initialize Clock Enable 1              */
    CKE0 = 0;                                   /* Initialize Clock Enable 0              */

    RDRF = 0;                                   /* Clear RDRF                 */

    ptr = &SRD[0];                              /* Initialize Serial Receiving Data Addreass */

    SRD[0] = 0x00;                              /* Initialize Serial Receiving Data 0     */
    SRD[1] = 0x00;                              /* Initialize Serial Receiving Data 1     */
    SRD[2] = 0x00;                              /* Initialize Serial Receiving Data 2     */
    SRD[3] = 0x00;                              /* Initialize Serial Receiving Data 3     */

    RE = 1;                                     /* Start Serial Receiving     */

    counter = 0;                                /* Clear counter              */

    while (counter < 4){                        /* Serial Receiving Data Counter 4 Loop   */

        while(RDRF == 0){                       /* RDRF = 1 ?                 */
            ;
        }

        if (OER == 1){                          /* Overrun Erorr Flag = 1 ?   */
            SRD[0] = 0xFF;                      /* Overrun Erorr 0            */
            SRD[1] = 0xFF;                      /* Overrun Erorr 1            */
            SRD[2] = 0xFF;                      /* Overrun Erorr 2            */
            SRD[3] = 0xFF;                      /* Overrun Erorr 3            */
            break;
        }
        else    {
            *ptr = RDR;                         /* Save Serial Receiving Data             */
            ptr++;                              /* Increment Serial Receiving Data Address*/
            counter++;                          /* Increment counter          */
        }
    }
```

プログラムリスト

```
    RE = 0;                                    /* Clear RE                                    */

    while(1){
        ;
    }
}
```

リンクアドレス指定

セクション名	アドレス
CV1	H'0000
P	H'0100
B	H'FB80