お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 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、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

M32C/85 グループ

DMAC を利用したシリアル I/O 連続送信/連続受信手順

1. 要約

この資料では、DMAC を利用したシリアル I/O 連続送信/連続受信手順と使用例を紹介しています。

2. はじめに

この資料で説明する応用例は次のマイコンに適用されます。

・ マイコン：M32C/85 グループ

M32C/85 グループと同様の SFR(周辺機能制御レジスタ)を持つ他の M16C ファミリでも本プログラムを使用することができます。ただし、一部の機能を機能追加等で変更している場合がありますのでマニュアルで確認してください。
このアプリケーションノートをご使用に際しては十分な評価を行ってください。

3. 使用例の説明

DMAC の要求要因にシリアル I/O の送信(または受信)を選択し、シリアル I/O の送信(または受信)のタイミングで、高速に送信バッファへ次のデータを書き込みます(または受信バッファから読み出します)。これを DMAC の転送回数分連続して行います。

3.1 接続例

図 1 に連続送信/連続受信時の接続例を示します。

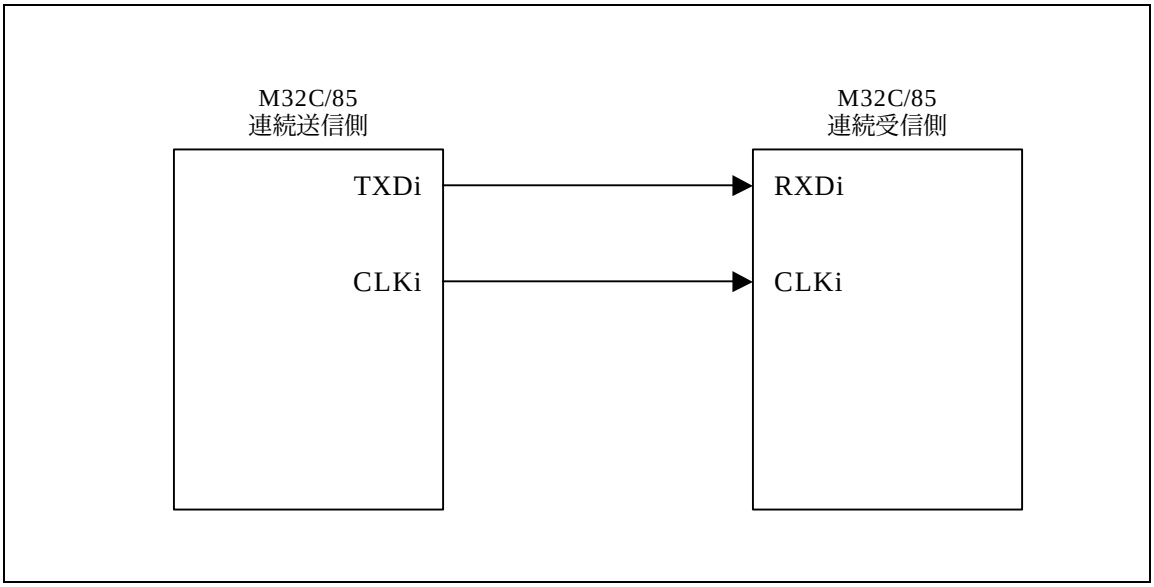


図 1. 連続送信/連続受信時の接続例

3.2 連続送信設定方法

8 バイトのデータを連続送信する場合の設定方法を以下に示します。

使用例：

- システム条件
  VCC1=VCC2=5V、XIN=32MHz
- DMAC 設定
  DMA 要求要因=UART0 送信、単転送、転送単位=8 ビット、転送方向=メモリ(順方向)→固定(U0TB レジスタ)
- シリアル I/O 設定
  クロック同期シリアル I/O モード、CTS/RTS 機能禁止、BRG カウントソース=f1、転送速度=125kbps(BRG=127)、
  送信割り込み要因=送信バッファ空

動作：

DMAC の要求要因に UART0 送信を指定し、最初の 1 バイトを UART0 送信バッファに書き込んだ後、UART0 送信割り込み要求をトリガにして残りの 7 バイトのデータを連続送信する。

図 2 に、連続送信のタイミング図を示す。

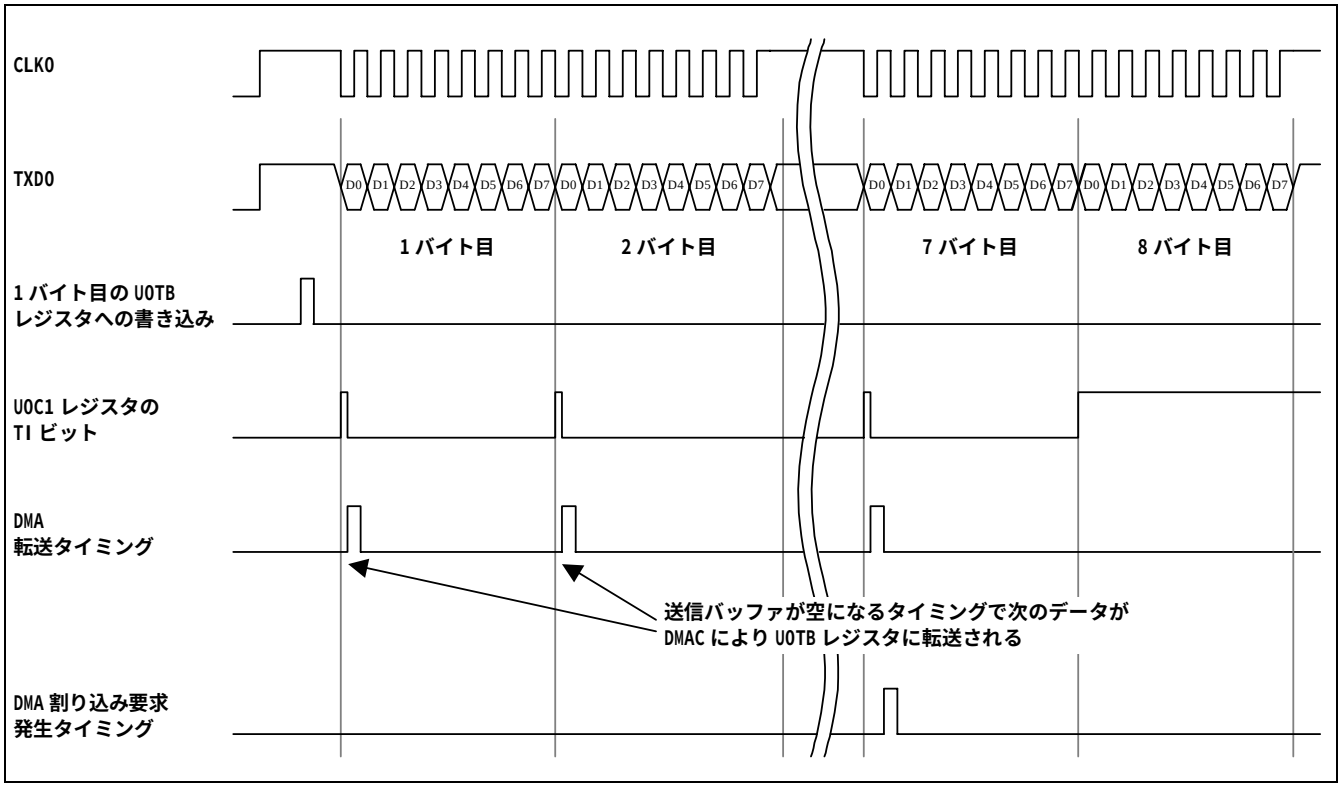
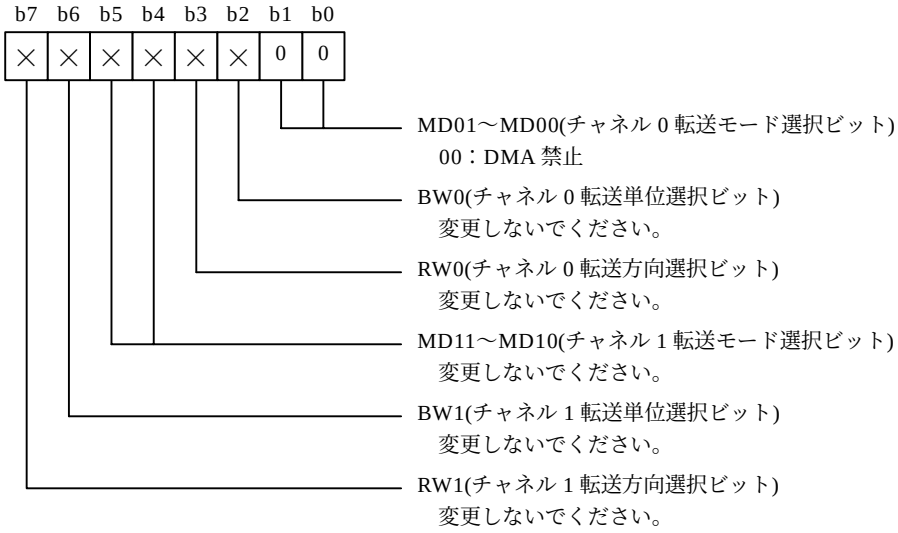


図 2. 連続送信のタイミング図

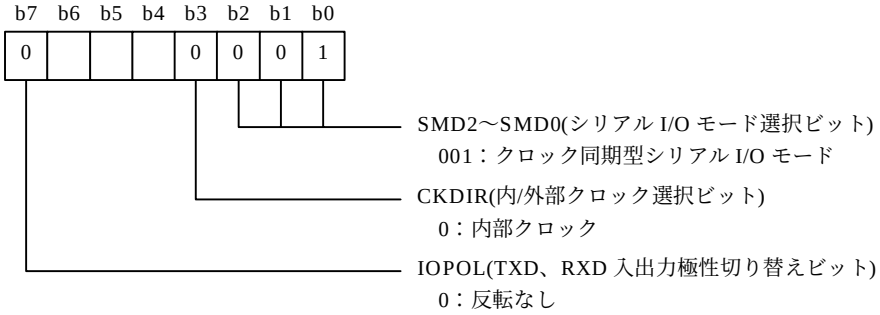
(1) DMA 禁止設定

・ DMD0 レジスタの MD01～MD00 ビットを“00b”にして DMA チャネル 0 を DMA 禁止にする。

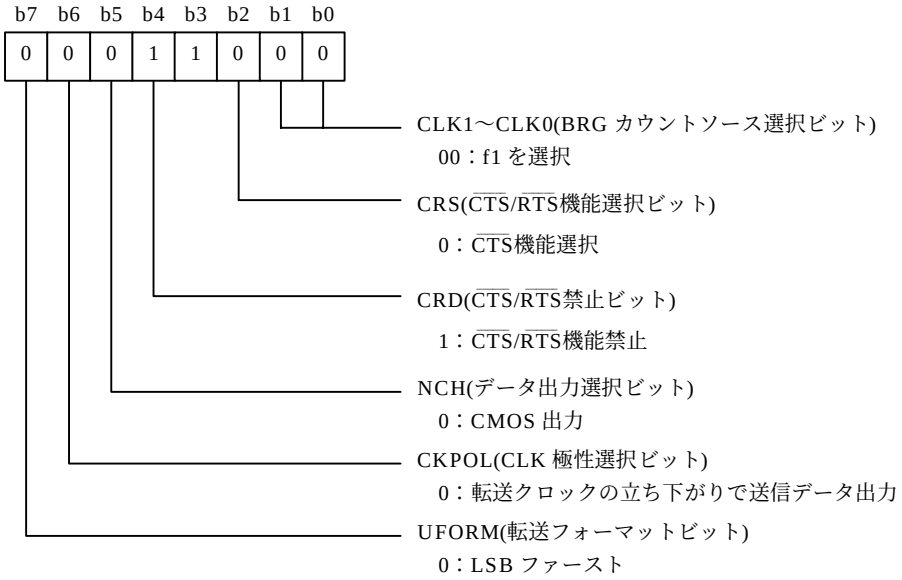


(2) シリアル I/O 設定

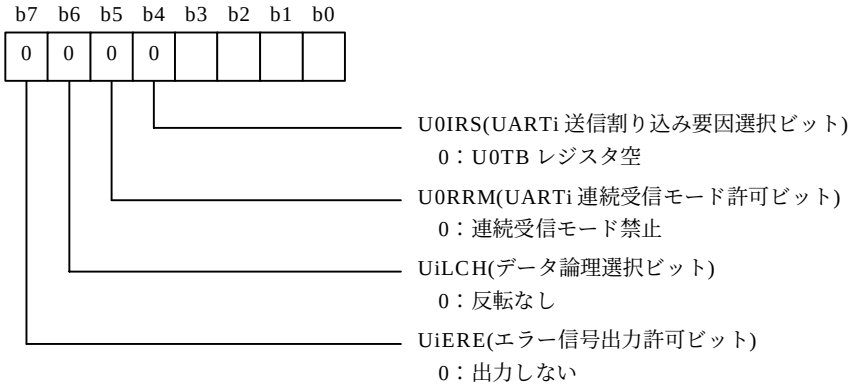
・ U0MR レジスタ(UART0 送受信モードレジスタ)を設定する。



・ U0C0 レジスタ(UART0 送受信制御レジスタ 0)を設定する。

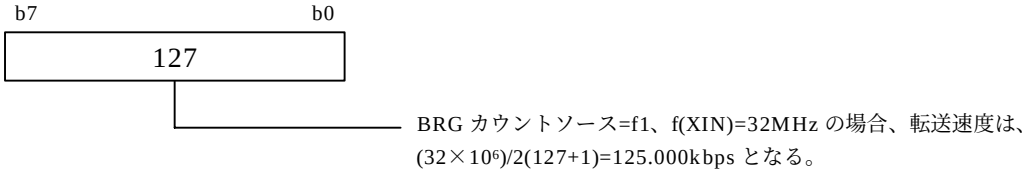


・ U0C1 レジスタ(UART0 送受信制御レジスタ 1)を設定する。

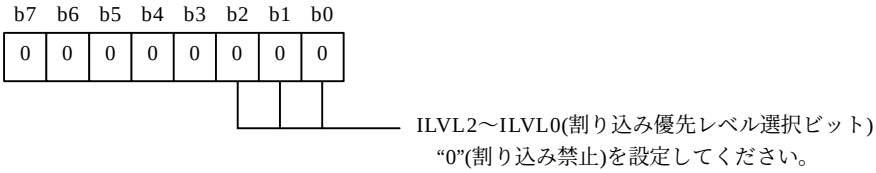


・ U0SMR レジスタ(UART0 特殊モードレジスタ)、 U0SMR2(UART0 特殊モードレジスタ 2)、U0SMR3 レジスタ(UART0 特殊モードレジスタ 3)、U0SMR4 レジスタ(UART0 特殊モードレジスタ 4)に"00h"を設定する。

・ U0BRG レジスタ(UART0 転送速度レジスタ)を設定する。

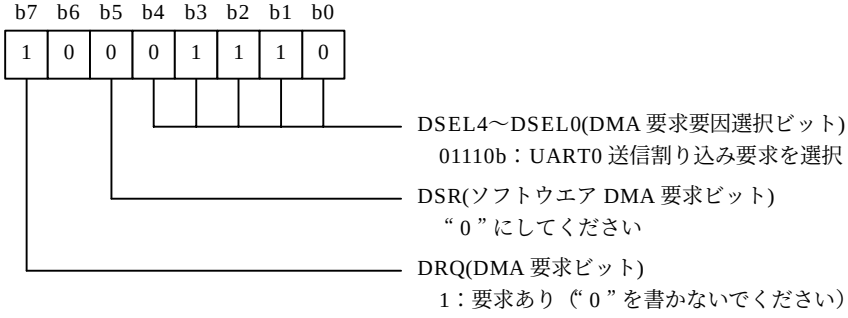


・ S0TIC レジスタ(UART0 送信割り込み制御レジスタ)を設定する。

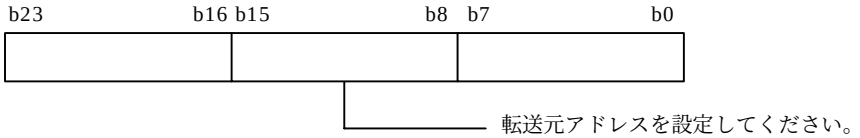


(3) DMAC 設定

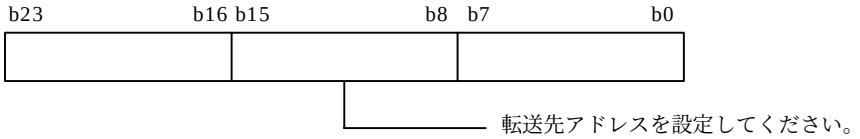
・ DM0SL レジスタ(DMA0 要因選択レジスタ)を設定する。



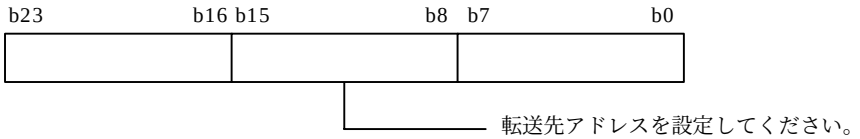
・ DSA0 レジスタ(DMA0SFR アドレスレジスタ)を設定する。



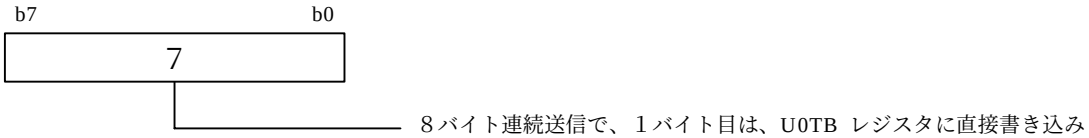
・ DMA0 レジスタ(DMA0 メモリアドレスレジスタ)を設定する。



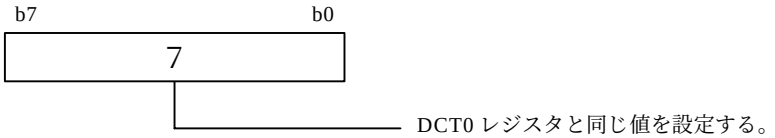
・ DRA0 レジスタ(DMA0 メモリアドレスリロードレジスタ)を設定する。



・ DCT0 レジスタ(DMA0 転送カウントレジスタ)を設定する。



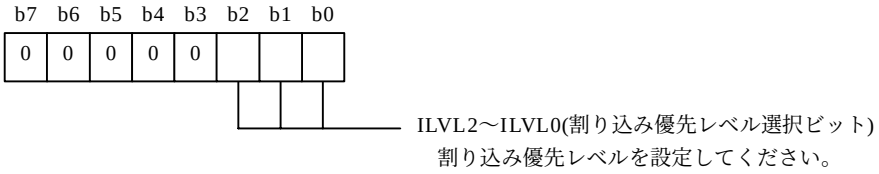
・ DRC0 レジスタ(DMA0 転送カウントリロードレジスタ)を設定する。



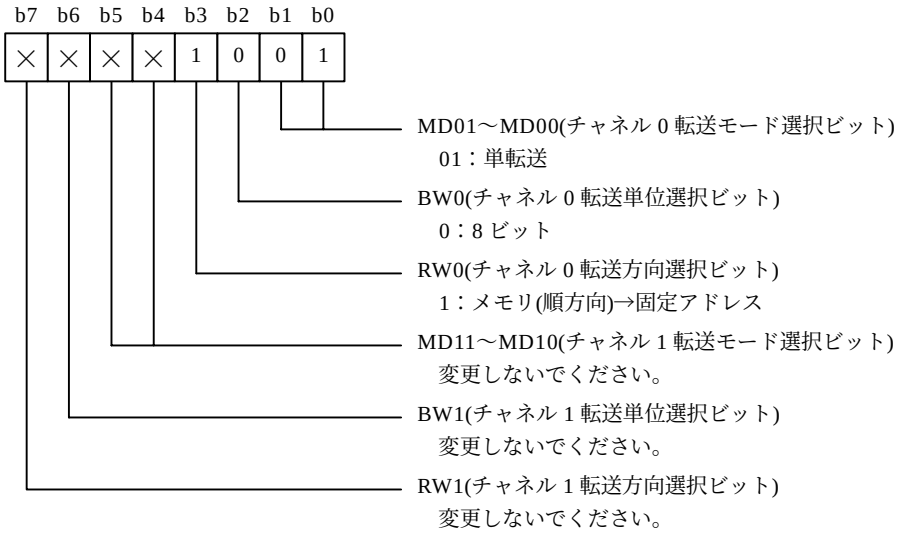
・ ダミーサイクル挿入

DM0SL レジスタを設定してから、BCLK で６サイクル分待ってから DMA を許可してください。ここでは、NOP を６個挿入することで、６サイクル分待ちます。

・ DM0IC レジスタ(DMA0 割り込み制御レジスタ)を設定する。

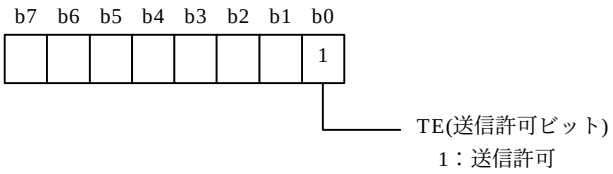


・ DMA0 を転送許可にする。
DMD0 レジスタ(DMA モードレジスタ 0)を設定する。



(4) 割り込みを許可（I フラグ=“1”）する。

(5) 送信許可
U0C1 レジスタの TE ビットを“1”(送信許可)にする。



(6) 連続送信開始
U0TB レジスタに連続送信するデータの 1 バイト目を書き込む。 これ以降は、UART0 送信割り込みを要因とする DMAC 転送により DMA 転送カウンタに設定したカウンタ分、連続送信が行われる。

(7) DMAC 転送完了割り込み処理
DMAC 転送完了フラグをセットする。

3.3 連続受信設定方法

8 バイトのデータを連続受信する場合の設定方法を以下に示します。

使用例：

- システム
  VCC1=VCC2=5V、XIN=32MHz
- DMAC 設定
  DMA 要求要因=UART0 受信、単転送、転送単位=16 ビット(エラーフラグを含む)、
  転送方向=固定(U0RB レジスタ)→メモリ(方向)
- シリアル I/O 設定
  クロック同期シリアル I/O モード、外部クロック(注 1)、CTS/RTS 機能禁止、連続受信モード許可

動作：

DMAC の要求要因に UART0 受信を指定し、UART0 送信バッファにダミーデータを書き込んだ後、UART0 受信割り込みをトリガに連続受信する。

図 3 に、連続受信のタイミング図を示す。

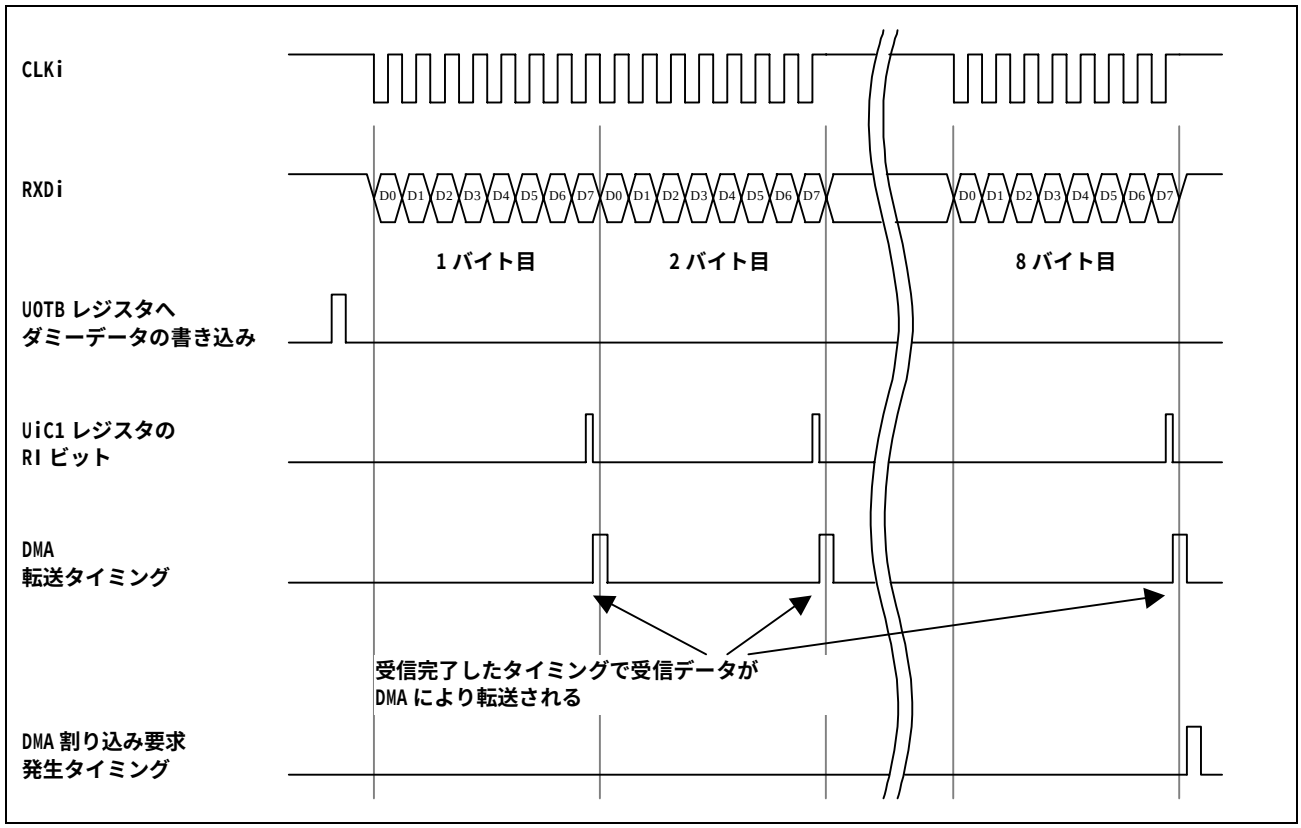


図 3. 連続受信のタイミング図

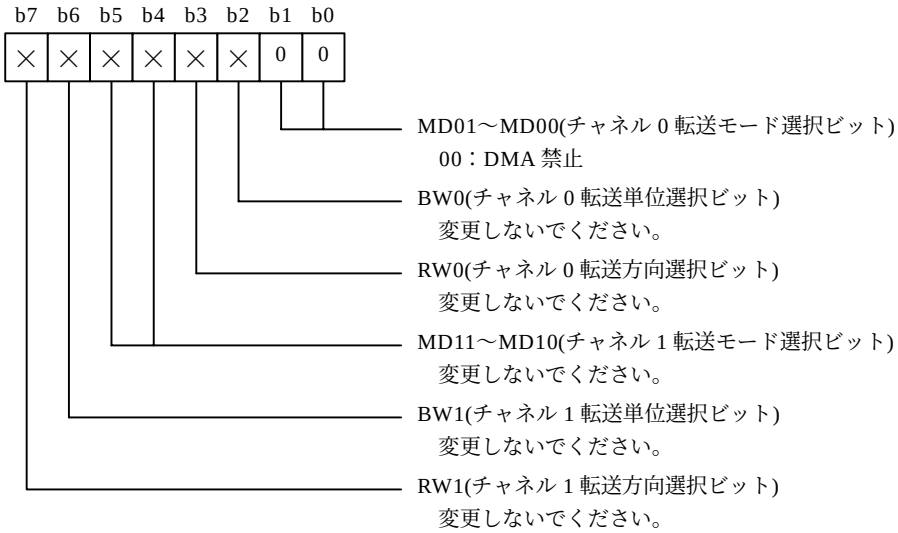
注 1.

データ受信前の CLKi 端子への入力が“H”のとき(UiC0 レジスタの CKPOL ビットが“1”なら“L”のとき)に、次の条件を満たしてください。

- UiC1 レジスタの TE ビット=1（送信許可）
- UiC1 レジスタの RE ビット=1（受信許可）
- UiTB レジスタにダミーデータを書く(または、UiRB レジスタを読む)

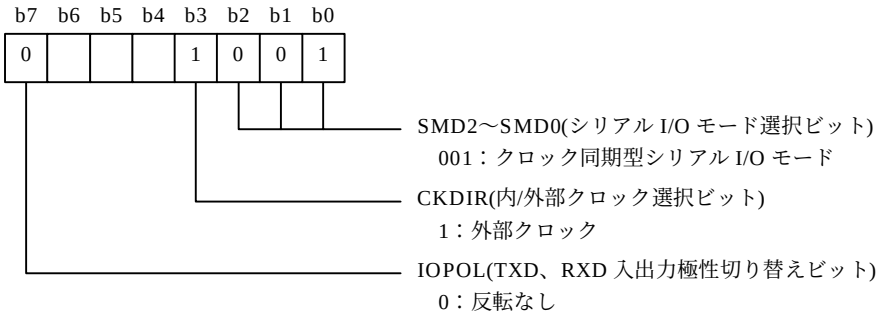
(1) DMA 禁止設定

・ DMD0 レジスタの MD01～MD00 ビットを“00b”にして DMA チャネル 0 を DMA 禁止にする。

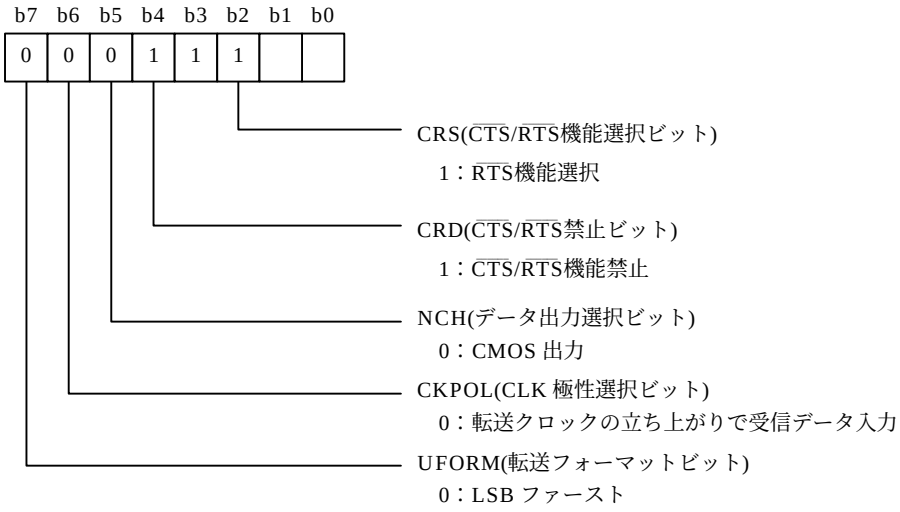


(2) シリアル I/O 設定

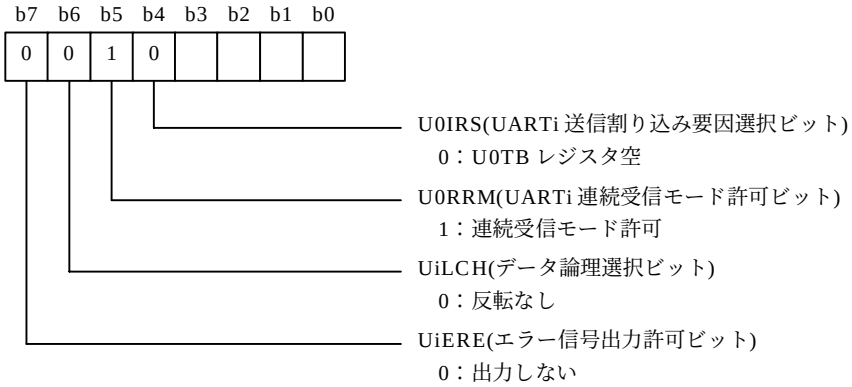
・ U0MR レジスタ(UART0 送受信モードレジスタ)を設定する。



・ U0C0 レジスタ(UART0 送受信制御レジスタ 0)を設定する。

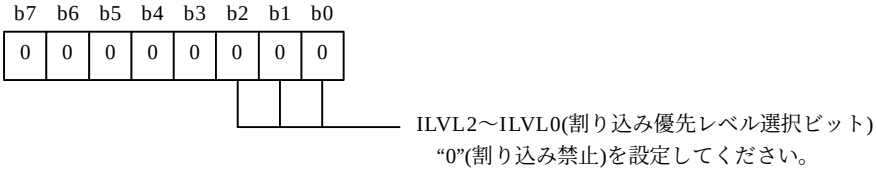


・ U0C1 レジスタ(UART1 送受信制御レジスタ 1)を設定する。



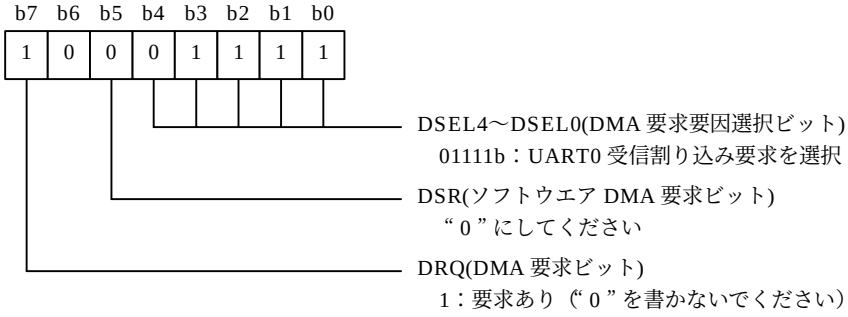
・ U0SMR レジスタ(UART0 特殊モードレジスタ) 、U0SMR2(UART0 特殊モードレジスタ 2)、U0SMR3(UART0 特殊モードレジスタ 3)、U0SMR4 レジスタ(UART0 特殊モードレジスタ 4)に"00h"を設定する。

・ S0RIC レジスタ(UART0 受信割り込み制御レジスタ)を設定する。

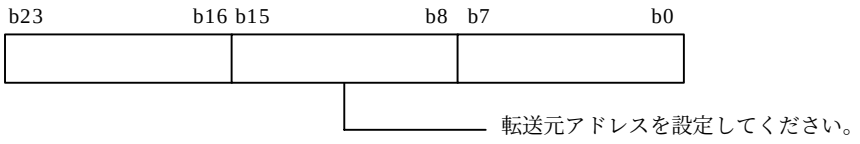


(3) DMAC 設定

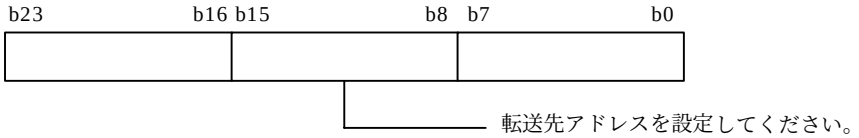
・ DM0SL レジスタ(DMA0 要因選択レジスタ)を設定する。



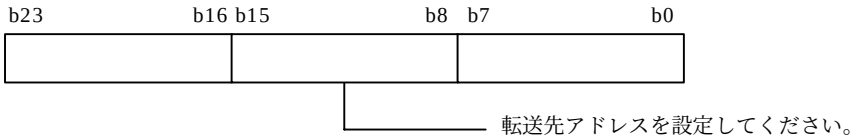
・ DSA0 レジスタ(DMA0SFR アドレスレジスタ)を設定する。



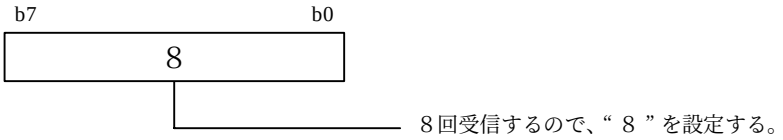
・ DMA0 レジスタ(DMA0 メモリアドレスレジスタ)を設定する。



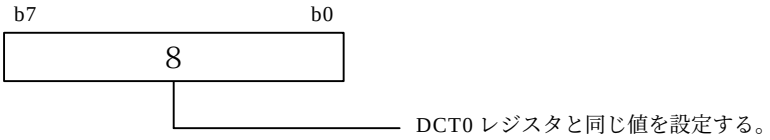
・ DRA0 レジスタ(DMA0 メモリアドレスリロードレジスタ)を設定する。



・ DCT0 レジスタ(DMA0 転送カウントレジスタ)を設定する。

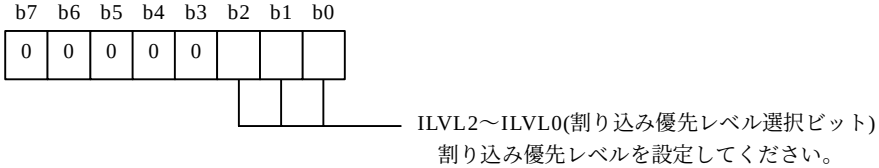


・ DRC0 レジスタ(DMA0 転送カウントリロードレジスタ)を設定する。

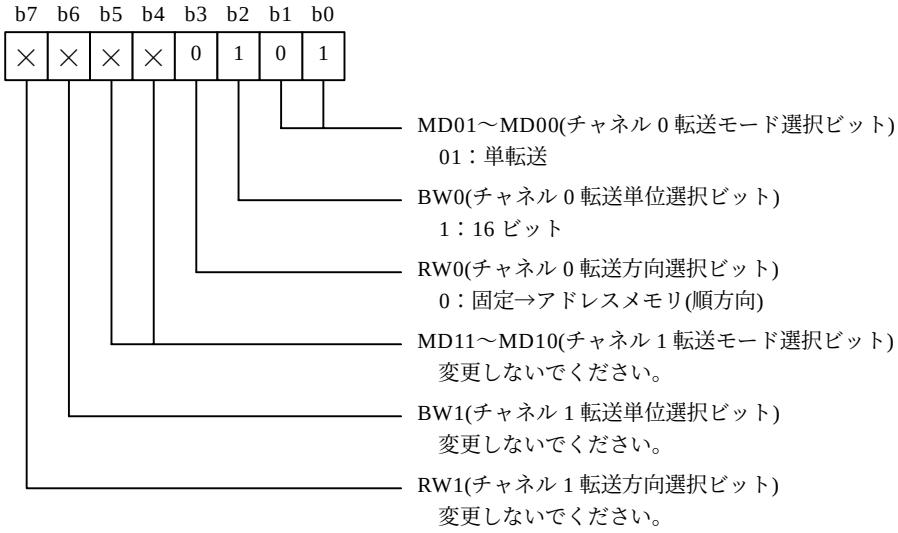


・ ダミーサイクル挿入
　DM0SL レジスタを設定してから、BCLK で 6 サイクル分待ってから DMA を許可してください。ここでは、NOP を 6 個挿入することで、6 サイクル分待ちます。

・ DM0IC レジスタ(DMA0 割り込み制御レジスタ)を設定する。



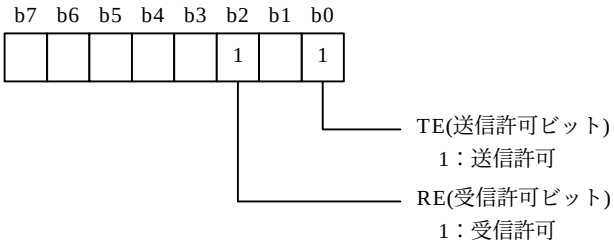
・ DMA0 を転送許可にする。
DMD0 レジスタ(DMA モードレジスタ 0)を設定する。



(4) 割り込みを許可（I フラグ=“1”）する。

(5) 送受信許可

U0C1 レジスタの TE ビットを“1”(送信許可)、RE ビットを“1”(受信許可）にする。



(6) 連続受信開始

U0TB レジスタにダミーデータを書き込み、連続受信を開始する。

(7) DMA 転送完了割り込み処理

受信したデータのエラーチェックを行い、必要ならエラー処理としてシリアル I/O の初期化を実施する。

4. 参考ドキュメント

ハードウエアマニュアル

M32C/85 グループ ハードウェアマニュアル
（最新版をルネサス テクノロジホームページから入手してください。）

5. ホームページとサポート窓口

ルネサス テクノロジホームページ

http://www.renesas.com/jpn/

M16C ファミリ MCU 技術サポート窓口

E-mail:support_apl@renesas.com

6. 参考プログラム例

6.1 連続送信プログラム例

DMAC を利用して 8 バイト分のデータを連続送信するプログラム例を示します。
ここでは、DMAC およびシリアル I/O は下記の仕様として設定します。

- DMAC 仕様
  DMA 要求要因=UART0 送信、単転送、転送単位=8 ビット、転送方向=順方向(メモリ)→固定(U0TB レジスタ)

- シリアル I/O 仕様
  クロック同期シリアル I/O モード、CTS/RTS 機能禁止、BRG カウントソース=f1、転送速度=125000bps((XIN=32MHz 時)
  送信割り込み要因=送信バッファ空

```
/***********************************************************************/
/*                                                                     */
/*  M32C/85 Group Program Collection                                   */
/*                                                                     */
/*  FILE NAME : rjj05b0578_snd.c                                       */
/*  CPU       : M32C/85 Group                                          */
/*  FUNCTION  : The sample program of the serial I/O continuation      */
/*              transmission using DMAC.                               */
/*  HISTORY   : 2004.08.16 Ver 1.00                                    */
/*                                                                     */
/*  Copyright (C) 2004. Renesas Technology Corp.                       */
/*  Copyright (C) 2004. Renesas Solutions Corp.                        */
/*  All right reserved.                                                */
/*                                                                     */
/***********************************************************************/

/****************************************/
/*    include file                      */
/****************************************/
#include "sfr32c8586.h"         // Special Function Register Header File

/****************************************/
/*    Function declaration              */
/****************************************/
void sio_init(void);            // Serial-I/O initialize routine
void dma0_int(void);            // DMA0 interrupt routine

/****************************************/
/*    Global variable declaration       */
/****************************************/
                                // Transfer data area.
unsigned char   snd_data[8] = {0x01, 0x03, 0x07, 0x0f, 0x1f, 0x3f, 0x7f, 0xff, 0x00};
unsigned char   dma_flg;        // Dma transmit complate flag. 1=complated.

/****************************************/
/*    #pragma declaration               */
/****************************************/
                                // CPU internal registor
unsigned short  dmd0;
#pragma DMAC    dmd0    DMD0    // DMD0(DMA mode register0)
unsigned short  dct0;
#pragma DMAC    dct0    DCT0    // DCT0(DMA0 transfer count register)
unsigned short   drc0;
```

```
#pragma DMAC    drc0    DRC0    // DRC0(DMA0 transfer count reload register)
void _far       *dma0;
#pragma DMAC    dma0    DMA0    // DMA0(DMA0 memory address register)
void _far       *dsa0;
#pragma DMAC    dsa0    DSA0    // DSA0(DMA0 SFR address register)
void _far       *dra0;
#pragma DMAC    dra0    DRA0    // DRA0(DMA0 memory address reload register)

/****************************************/
/*    Main Program                      */
/****************************************/
void main(void)
{
    unsigned short   dmd0_tmp;      // DMD0 register temp

    dmd0_tmp =  dmd0;               // (1)DMA0 inhibit(DMD0)
    dmd0_tmp &= 0x00fc;             //  <MD00-01> : DMA inhibit
    dmd0     =  dmd0_tmp;           //

    sio_init();                     // Serial-I/O initialization.

    dm0sl = 0x8e;                   // Set DM0SL register.
                                    //  <DSEL4-0> : UART0 transmit
                                    //  <DRQ>     : DMA requested

    dsa0  = &u0tb;                  // Set DSA0 register.
    dma0  = &snd_data[1];           // Set DMA0 register.
    dra0  = &snd_data[1];           // Set DRA0 register.
    dct0  = 7;                      // Set DCT0 register.
    drc0  = 7;                      // Set DRC0 register.
                                    // Dummy cycle insertion
    asm("NOP    ");                 //  It waits by 6 cycles by BCLK.
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");

    dm0ic = 4;                      // Set DMA0 interrupt priority-level = 4.

    dmd0_tmp |= 0x09;               // DMA0 permission(DMD0)
    dmd0      = dmd0_tmp;           //  <MD01-00> : single transfer
                                    //  <BW0>     : 8bit
                                    //  <RW0>     : Memory to Fixed address

    asm("fset i");                  // Interrupt enabled

    te_u0c1 = 1;                    // U0C1 register re-setup.
                                    //  <TE>      : transmit enabled

    u0tb = snd_data[0];             // First byte transmission.

    while(1);

}

/****************************************/
/*  Serial-I/O initialize routine       */
/****************************************/
void sio_init(void)
```

```
{
    u0mr = 0x01;            // Set U0MR register.
                            //  <SMOD2-0> : Clock-synchronous
                            //  <CKDIR>   : Internal-clock
                            //  <IOPOL>   : No reverse

    u0c0 = 0x18;            // Set U0C0 register.
                            //  <CLK1-0>  : f1
                            //  <CRS>     : select CTS function
                            //  <CRD>     : CTS/RTS function disabled
                            //  <NCH>     :
                            //  <CKPOL>   : transfer data is output at falling edge
                            //  <UFORM>   : LSB first

    u0c1 = 0x00;            // Set U0C1 register.
                            //  <TE>      : transmit disabled
                            //  <RE>      : receive disabled
                            //  <U0IRS>   : Transmit interrupt cause = Buffer empty
                            //  <U0LCH>   : No reverse
                            //  <U0ERE>   : Error signal output disable

    u0smr  = 0x00;          // Set U0SMR register.
    u0smr2 = 0x00;          // Set U0SMR2 register.
    u0smr3 = 0x00;          // Set U0SMR3 register.
                            //  <NODC> : CLK0 is CMOS output
    u0smr4 = 0x00;          // Set U0SMR4 register.

    u0brg = 127;            // Set U0BRG register.
                            //   125000bps(XIN=32MHz)

    pd6 = 0x00;             // Set PD6 register.
    ps0 = 0x0a;             // Set PS0 register.
                            //  <Select CLK0,TXD0 output>
    s0tic = 0;              // Set UART0 transmit interrupt priority-level = 0.

}

/****************************************/
/*  DMA0 interrupt routine              */
/****************************************/
#pragma INTERRUPT/B dma0_int
// "/B" = Instead of saving the registers to the stack,
//        you can switch to the alternate registers.

void dma0_int(void)
{
    dma_flag = 1;           // DMA transmit complate set.
    p10  = 0xff;            // Transmit complate display.
    pd10 = 0xff;

}
```

6.2 連続受信プログラム例

DMA を利用して８バイト分のデータを連続受信するプログラム例を示します。
DMAC およびシリアル I/O は下記の仕様として設定します。

- DMAC 仕様
  DMA 要求要因=UART0 受信、単転送、転送単位=16 ビット(エラーフラグも含む)、
  転送方向=固定(U0RB レジスタ)→順方向(メモリ)

- シリアル I/O 設定条件
  クロック同期シリアル I/O モード、外部クロック、CTS/RTS 機能禁止、連続受信モード許可

```
/*****************************************************************************/
/*                                                                           */
/*  M32C/85 Group Program Collection                                         */
/*                                                                           */
/*  FILE NAME : rjj05b0578_rcv.c                                             */
/*  CPU       : M32C/85 Group                                                */
/*  FUNCTION  : The sample program of the serial I/O continuation            */
/*              reception using DMAC.                                        */
/*  HISTORY   : 2004.08.16 Ver 1.00                                          */
/*                                                                           */
/*  Copyright (C) 2004. Renesas Technology Corp.                             */
/*  Copyright (C) 2004. Renesas Solutions Corp.                              */
/*  All right reserved.                                                      */
/*                                                                           */
/*****************************************************************************/

/*****************************************/
/*    include file                       */
/*****************************************/
#include "sfr32c8586.h"         // Special Function Register Header File

/*****************************************/
/*    Function declaration               */
/*****************************************/
void sio_init(void);            // Serial-I/O initialize routine
void dma0_int(void);            // DMA0 interrupt routine

/*****************************************/
/*    Global variable declaration        */
/*****************************************/
unsigned short  rcv_data[8];    // Repeat receive data area

/*****************************************/
/*    #pragma declaration                */
/*****************************************/
                                // CPU internal registor
unsigned short  dmd0;
#pragma DMAC    dmd0    DMD0    // DMD0(DMA mode register0)
unsigned short  dct0;
#pragma DMAC    dct0    DCT0    // DCT0(DMA0 transfer count register)
unsigned short   drc0;
#pragma DMAC    drc0    DRC0    // DRC0(DMA0 transfer count reload register)
void _far       *dma0;
```

```
#pragma DMAC    dma0    DMA0    // DMA0(DMA0 memory address register)
void _far       *dsa0;
#pragma DMAC    dsa0    DSA0    // DSA0(DMA0 SFR address register)
void _far       *dra0;
#pragma DMAC    dra0    DRA0    // DRA0(DMA0 memory address reload register)

/****************************************/
/*    Main Program                      */
/****************************************/
void main(void)
{

    unsigned short  dmd0_tmp;   // DMD0 register temp

    dmd0_tmp =  dmd0;           // (1)DMA0 inhibit(DMD0)
    dmd0_tmp &= 0x00fc;         //  <MD00-01> : DMA inhibit
    dmd0     =  dmd0_tmp;       //

    sio_init();                 // Serial-I/O initialization.

    dm0sl = 0x8f;               // Set DM0SL register.
                                //  <DSEL4-0> : UART0 receive
                                //  <DRQ>     : DMA requested

    dsa0  = &u0rb;              // Set DSA0 register.
    dma0  = &rcv_data[0];       // Set DMA0 register.
    dra0  = &rcv_data[0];       // Set DRA0 register.
    dct0  = 8;                  // Set DCT0 register.
    drc0  = 8;                  // Set DRC0 register.
                                // Dummy cycle insertion
    asm("NOP    ");             //  It waits by 6 cycles by BCLK.
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");
    asm("NOP    ");

    dm0ic = 4;                  // Set DMA0 interrupt priority-level = 4.

    dmd0_tmp |= 0x05;           // DMA0 permission(DMD0)
    dmd0      = dmd0_tmp;       //  <MD01-00> : single transfer
                                //  <BW0>     : 16bit
                                //  <RW0>     : Fixed address to Memory

    asm("fset i");              // Interrupt enabled

    u0c1 |= 0x05;               // U0C1 register re-setup.
                                //  <TE>      : transmit enabled
                                //  <RE>      : receive enabled

    u0tb = 0x55;                // dummy data Set for receive.

    while(1);

}

/****************************************/
/*  Serial-I/O initialize routine       */
/****************************************/
void sio_init(void)
```

```
{
    u0mr = 0x09;            // Set U0MR register.
                            //  <SMOD2-0> : Clock-synchronous
                            //  <CKDIR>   : External-clock
                            //  <IOPOL>   : No reverse

    u0c0 = 0x1c;            // Set U0C0 register.
                            //  <CLK1-0>  :
                            //  <CRS>     : select RTS function
                            //  <CRD>     : CTS/RTS function disabled
                            //  <NCH>     :
                            //  <CKPOL>   : receive data is input at rising edge
                            //  <UFORM>   : LSB first

    u0c1 = 0x20;            // Set U0C1 register.
                            //  <TE>      : transmit disabled
                            //  <RE>      : receive disabled
                            //  <U0RRM>   : Continuous receive mode enabled
                            //  <U0LCH>   : No reverse
                            //  <U0ERE>   : Error signal output disable

    u0smr  = 0x00;          // Set U0SMR register.
    u0smr2 = 0x00;          // Set U0SMR2 register.
    u0smr3 = 0x00;          // Set U0SMR3 register.
    u0smr4 = 0x00;          // Set U0SMR4 register.

    pd6 = 0x00;             // Set PD6 register.
    ps0 = 0x01;             // Set PS0 register.
                            //  <Select RTS0 output>
    s0ric = 0;              // Set UART0 receive interrupt priority-level = 0.

}

/****************************************/
/*  DMA0 interrupt routine              */
/****************************************/
#pragma INTERRUPT/B dma0_int
// "/B" = Instead of saving the registers to the stack,
//        you can switch to the alternate registers.

void dma0_int(void)
{

                            // Receive data display.
    pd0  = 0xff;            // P0 is an output port.
    pd1  = 0xff;            // P1 is an output port.
    pd2  = 0xff;            // P2 is an output port.
    pd3  = 0xff;            // P3 is an output port.
    pd4  = 0xff;            // P4 is an output port.
    pd5  = 0xff;            // P5 is an output port.
    pd6  = 0xff;            // P6 is an output port.
    pd7  = 0xff;            // P7 is an output port.
    p0 = rcv_data[0];
    p1 = rcv_data[1];
    p2 = rcv_data[2];
    p3 = rcv_data[3];
    p4 = rcv_data[4];
    p5 = rcv_data[5];
    p6 = rcv_data[6];
    p7 = rcv_data[7];
```

```
    prc2 = 1;
    pd9  = 0xff;
    pd10 = 0xff;
    p9   = rcv_data[7];
    p10  = (char)(rcv_data[7] >> 8);

}
```

改訂記録

Rev.	発行日		改訂内容
		ページ	ポイント
1.00	2004.09.01	-	初版発行
1.01	2005.03.25	4,7,9,12	ビットシンボル名の誤記修正
		5	レジスタ名の誤記修正

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。