V850 マイクロコントローラ

V850ES/Jx3-L

DMA 転送を使用した CSI 連続送受信（マスタ）

R01AN0782JJ0100
Rev.1.00
2011.09.29

要旨

本アプリケーションノートでは、V850ES/Jx3-L シリーズのクロック同期式シリアル・インタフェース B（CSIB）と DMA コントローラを使用して CSI 連続送受信（マスタ）を行う方法を説明します。

本アプリケーションノートでは、DMA 転送で CSIB2 の送受信を制御し、16 バイト（1 バイト×16 回）の CSI 連続送受信（マスタ）を行います。

対象デバイス

V850ES/JC3-L

V850ES/JE3-L

V850ES/JF3-L

V850ES/JG3-L

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 仕様

本アプリケーションノートでは、クロック同期式シリアル・インタフェース B（CSIB）と DMA コントローラの使用例を示しています。2 チャネルの DMA 転送で CSIB2 の送受信を制御し、16 バイト（1 バイト×16 回）の CSI 連続送受信（マスタ）を行います。

【動作概要】

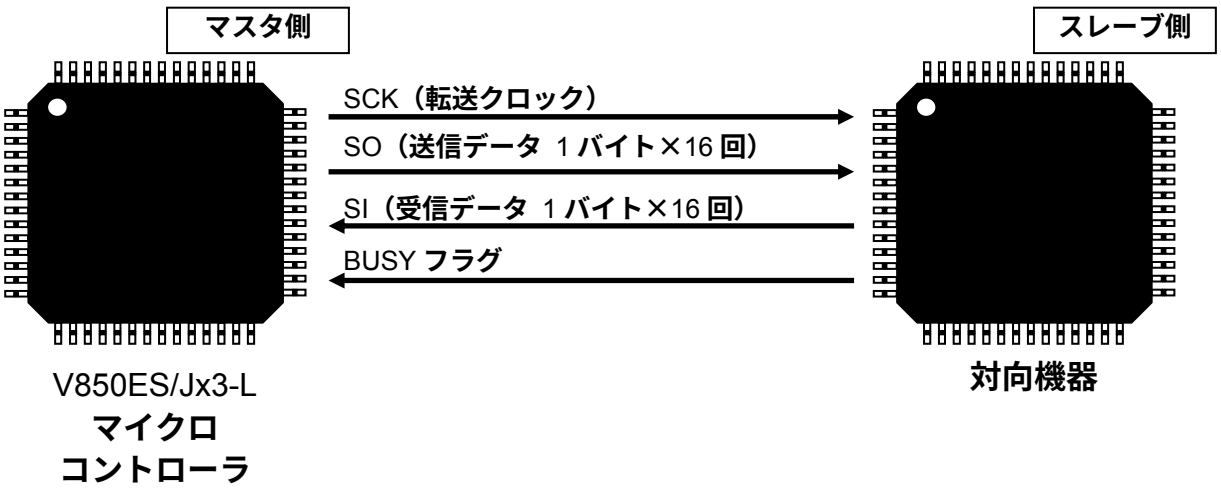


表 1.1 に使用する周辺機能と用途を、表 1.2 に CSIB2 の設定を示します。

表 1.1 使用する周辺機能と用途

周辺機能	用途
CSIB2	CSI 連続送受信（マスタ）
DMA コントローラ 0（DMA0）	受信動作の制御
DMA コントローラ 1（DMA1）	送信動作の制御

表 1.2 CSIB2 の設定

項目	設定
データ長	8 ビット
転送方向	MSB ファースト
データ送受信タイミング	タイプ 1
転送クロック	5MHz

また、送信データとして使用する 16 バイトデータを表 1.3 に示します。

表 1.3 送信用の 16 バイトデータ

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f

備考 V850ES/Jx3-L シリーズの CSIB は、製品ごとに搭載しているチャネル数が異なります。表 1.4 に CSI 機能の違いを示します。

表 1.4 CSI 機能の違い

愛称	品名	Flash/RAM	搭載しているチャネル数
V850ES/JC3-L（40pin）	μPD70F3797 μPD70F3798 μPD70F3799 μPD70F3800 μPD70F3838	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	2 チャネル
V850ES/JC3-L（48pin）	μPD70F3801 μPD70F3802 μPD70F3803 μPD70F3804 μPD70F3839	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	4 チャネル
V850ES/JE3-L（64pin）	μPD70F3805 μPD70F3806 μPD70F3807 μPD70F3808 μPD70F3840	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	5 チャネル
V850ES/JF3-L（80pin）	μPD70F3735 μPD70F3736	128KB/8KB 256KB/16KB	3 チャネル
V850ES/JG3-L	μPD70F3737 μPD70F3738 μPD70F3792 μPD70F3793 μPD70F3841 μPD70F3842	128KB/8KB 256KB/16KB 384KB/32KB 512KB/40KB 768KB/80KB 1MB/80KB	5 チャネル
V850ES/JG3-L（USB 搭載品）	μPD70F3794 μPD70F3795 μPD70F3796 μPD70F3843 μPD70F3844	256KB/40KB 384KB/40KB 512KB/40KB 768KB/80KB 1MB/80KB	5 チャネル

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2.1 動作確認条件

項目	内容
使用マイコン	V850ES/JG3-L（USB 搭載品）（μPD70F3796GC）
動作周波数	● CPU クロック ：20MHz ● 周辺クロック ：20MHz ● メイン・クロック発振周波数 ：5MHz
動作電圧	3.0V（2.7V～3.6V） （CPU クロックが 20MHz 動作のとき）
統合開発環境	ルネサス エレクトロニクス製 CubeSuite+ V1.00.01
C コンパイラ	ルネサス エレクトロニクス製 CA850 V3.50
使用ボード	V850ES/JG3-L(USB) ターゲット･ボード （QB-V850ESJG3LUSB-TB）
スレーブ機器	V850ES/JG3-L(USB) ターゲット･ボード （QB-V850ESJG3LUSB-TB）
スレーブ機器の設定[注]	● データ長 ：8 ビット ● 転送方向 ：MSB ファースト ● データ送受信タイミング ：タイプ 1

注． V850ES/Jx3-L アプリケーションノート DMA 転送を使用した CSI 連続送受信（スレーブ）のサンプルコードを使用しています。

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

V850ES/Jx3-L アプリケーションノート LED 点灯のスイッチ制御編（U19479JJ）
V850ES/Jx3-L アプリケーションノート DMA 転送を使用した CSI 連続送受信（スレーブ）
（R01AN0781JJ0100）

4. ハードウエア説明

4.1 使用端子一覧

表 4.1 に使用端子と機能を示します。

表 4.1 使用端子と機能

端子名	入出力	内容
P53/SIB2	入力	CSIB2 受信入力
P54/SOB2	出力	CSIB2 送信出力
P55/SCKB2	出力	CSIB2 クロック出力
P50	入力	スレーブ BUSY 信号の入力

4.2 ハードウエア構成例

図 4.2 に本アプリケーションノートで使用するハードウエア構成例を示します。

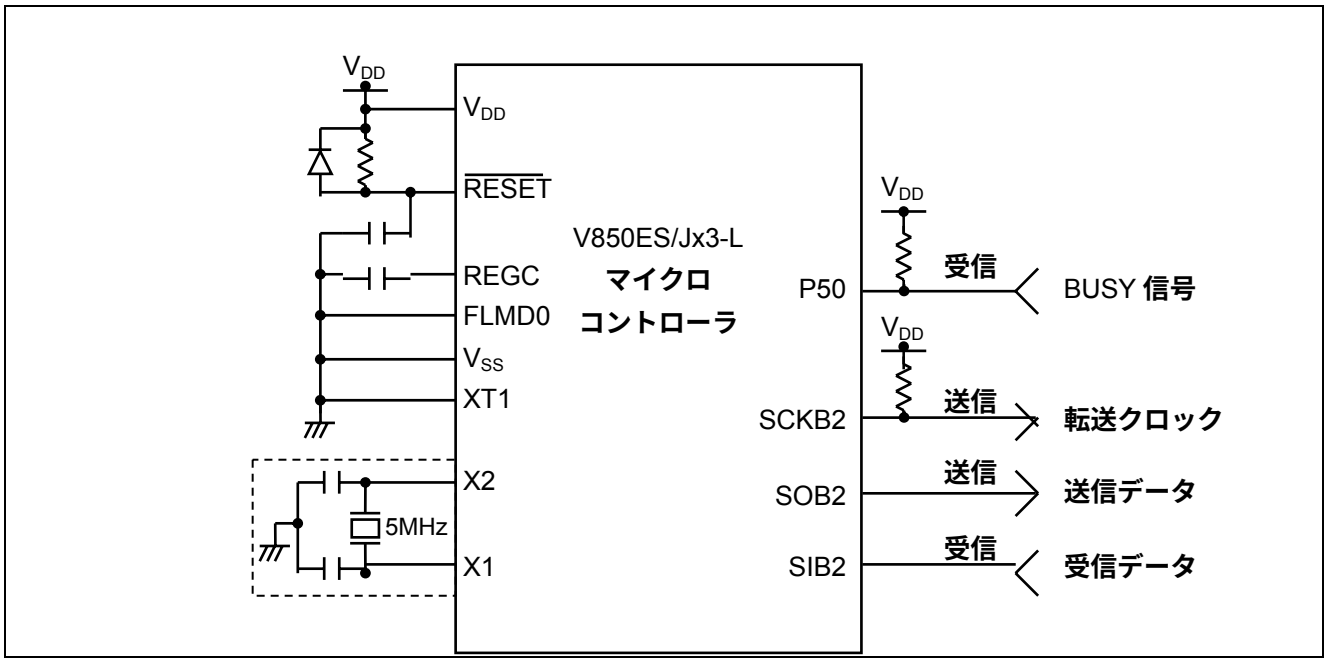


図 4.2 ハードウエア構成

注意 1 V_{DD} は 2.7 V＜V_{DD}≦3.6 V の電圧範囲で使用してください。なお、この電圧範囲は CPU クロックが 20MHz 動作のときのものです。

2 EV_{DD} 端子，AV_{REF0} 端子は V_{DD} と同電位にしてください。

3 EV_{SS} 端子は GND と同電位にしてください。

4 REGC はコンデンサ（推奨値：4.7 μF）を介し，GND に接続してください。

5 FLMD0 端子は，通常動作モード時は GND に接続してください。

6 未使用ポートについては，出力ポートとして処理するため，すべてオープンとしてください。

7 メイン・クロック発振回路は，配線容量などの影響を避けるために，図中の破線の部分を次のように配線してください。

- 配線は極力短くする。
- 他の信号線と交差させない。
- 変化する大電流が流れる線に接近させない。
- 発振回路のコンデンサの接地点は，常に V_{SS} と同電位になるようにする。
- 大電流が流れるグランド・パターンに接地しない。
- 発振回路から信号を取り出さない。

5. ソフトウエア説明

5.1 動作概要

本アプリケーションノートでは、CSIB2、DMA0、DMA1 を使用し、16 バイト（1 バイト×16 回）の CSI 連続送受信（マスタ）を行います。

DMA1 転送による送信データの書き込みと DMA0 転送による受信データの保存を繰り返すことで、連続送受信動作を実現します。

＜設定条件＞

- CSIB2 の動作を連続転送モードに設定します。
- CSIB2 の転送クロックを 5MHz に設定します。
- CSIB2 のデータ長を 8 ビットに設定します。
- CSIB2 の転送方向を MSB ファーストに設定します。
- CSIB2 の送受信タイミングをタイプ 1 に設定します。
- DMA0 の転送データサイズを 8 ビットに設定します。
- DMA0 の転送元を CB2RXL レジスタ、転送先を RAM に設定します。
- DMA0 の転送回数を 16 に設定します。
- DMA0 の起動要因を INTCB2R に設定します。
- DMA0 転送終了割り込み（INTDMA0）を使用します。
- DMA1 の転送データサイズを 8 ビットに設定します。
- DMA1 の転送元を RAM、転送先を CB2TXL レジスタに設定します。
- DMA1 の転送回数を 16 に設定します。
- DMA1 の起動要因を INTCB2T に設定します。

(1) CSIB2、DMA0、DMA1 の初期設定を行います。

(2) DMA0 転送終了割り込み（INTDMA0）を許可し、CSIB2 の動作を許可します。

(3) スレーブ機器からの BUSY 信号が解除（ロウ・レベル出力）されたとき、CSIB2 のマスタ送受信を開始します。また、CPU は HALT モードに移行します。

(4) 1 バイトの送受信を行うごとに、INTCB2R 割り込みをトリガとした DMA0 転送で受信データを RAM に保存します。また、INTCB2T 割り込みをトリガとした DMA1 転送で次の送信データを設定します。送信データの設定により、次の送受信が開始されます。

(5) 16 バイト目の送受信により、DMA0 転送終了割り込み（INTDMA0）が発生し、CPU の HALT モードは解除されます。

(6) 以降は（3）から（5）の処理を繰り返します。

5.2 オプション・バイトの設定一覧

表 5.1にオプション・バイト設定を示します。

表 5.2 オプション・バイト設定

アドレス	設定値	内容
0000007AH	00000101B	ウォッチドッグ・タイマ 2 の動作クロック（f_X/f_T/f_R）選択可能、INTWDT2/WDTRES モード選択可能
		ソフトウエアにより内蔵発振器停止可能
		発振安定時間 $2^{15}/f_X$

5.3 定数一覧

表 5.2.1 にサンプルコードで使用する定数、表 5.2.2 にサンプルコードで使用するテーブルを示します。

表 5.2.1 サンプルコードで使用する定数

定数名	設定値	内容
TXDNUM	16	送信データのバイト数
BUFNUM	3	受信バッファの保存開始位置の更新回数

表 5.2.2 サンプルコードで使用するテーブル

テーブル名	設定値	内容
gTxTable	0x00,0x01,0x02・・0x0f	送信用の 16 バイトデータ

5.4 変数一覧

表 5.3 にグローバル変数を示します。

表 5.3 グローバル変数

型	変数名	内容	使用する関数
UCHAR	gRxBuffer	受信バッファ（48 バイト）	MD_StartCommunication
UCHAR	gTxBuffer	送信バッファ（16 バイト）	MD_VarInit
UCHAR	gBuffCounter	受信バッファの保存開始位置	MD_StartCommunication

5.5 関数一覧

表 5.4 にエラー! 参照元が見つかりません。を示します。

表 5.4 関数

関数名	概要
MD_VarInit	使用する変数の初期値設定
MD_StartCommunication	送受信開始処理
MD_INTDMA0	INTDMA0 割り込み処理

5.6 関数仕様

サンプルコードの関数仕様を示します。

[関数名] MD_VarInit

概要	使用する変数の初期値設定
ヘッダ	CG_dma.h
宣言	void MD_VarInit(void)
説明	使用する変数に初期値を設定します。
引数	なし
リターン値	なし
備考	なし

[関数名] MD_StartCommunication

概要	送受信開始処理
ヘッダ	CG_dma.h
宣言	void MD_StartCommunication(void)
説明	DMA0、DMA1 転送動作を許可し、CSI 送受信動作を開始します。
引数	なし
リターン値	なし
備考	なし

[関数名] MD_INTDMA0

概要	INTDMA0 割り込み処理
ヘッダ	CG_dma.h
宣言	__interrupt void MD_INTDMA0(void)
説明	CPU のスタンバイを解除します。また、TC1 フラグをクリアします。
引数	なし
リターン値	なし
備考	なし

5.7 状態遷移図

このサンプルコードでは、初期設定で、クロック周波数の選択や、ウォッチドッグ・タイマ 2 の停止設定、入出力ポートの設定、CSIB2 の設定、DMA0 の設定、DMA1 の設定などを行います。

初期設定完了後、スレーブからのビジー信号をチェックします。ビジーの解除が確認できた場合、CSI 連続送受信を開始し、CPU は HALT モードに移行します。以降は、16 バイト（1 バイト×16 回）の送受信が完了するごとに、ビジー信号のチェックと送受信の開始を繰り返します。

詳細については、次の状態遷移図（ステート・チャート）に示します。

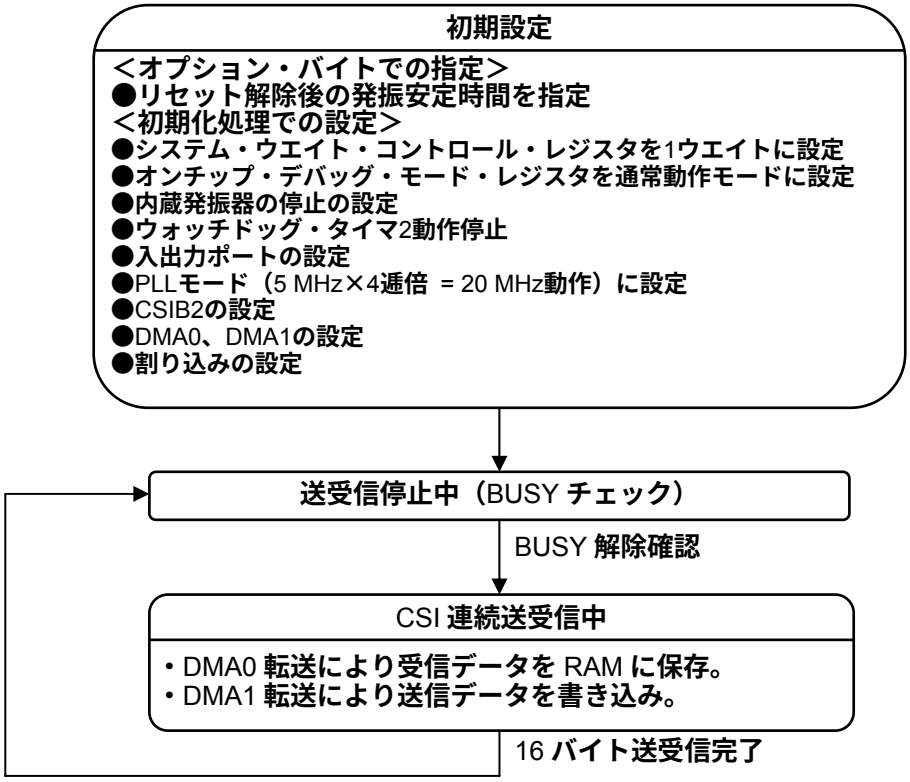


図 5.1 状態遷移図

5.8 フローチャート

5.8.1 メイン処理

図 5.2 にメイン処理のフローを示します。

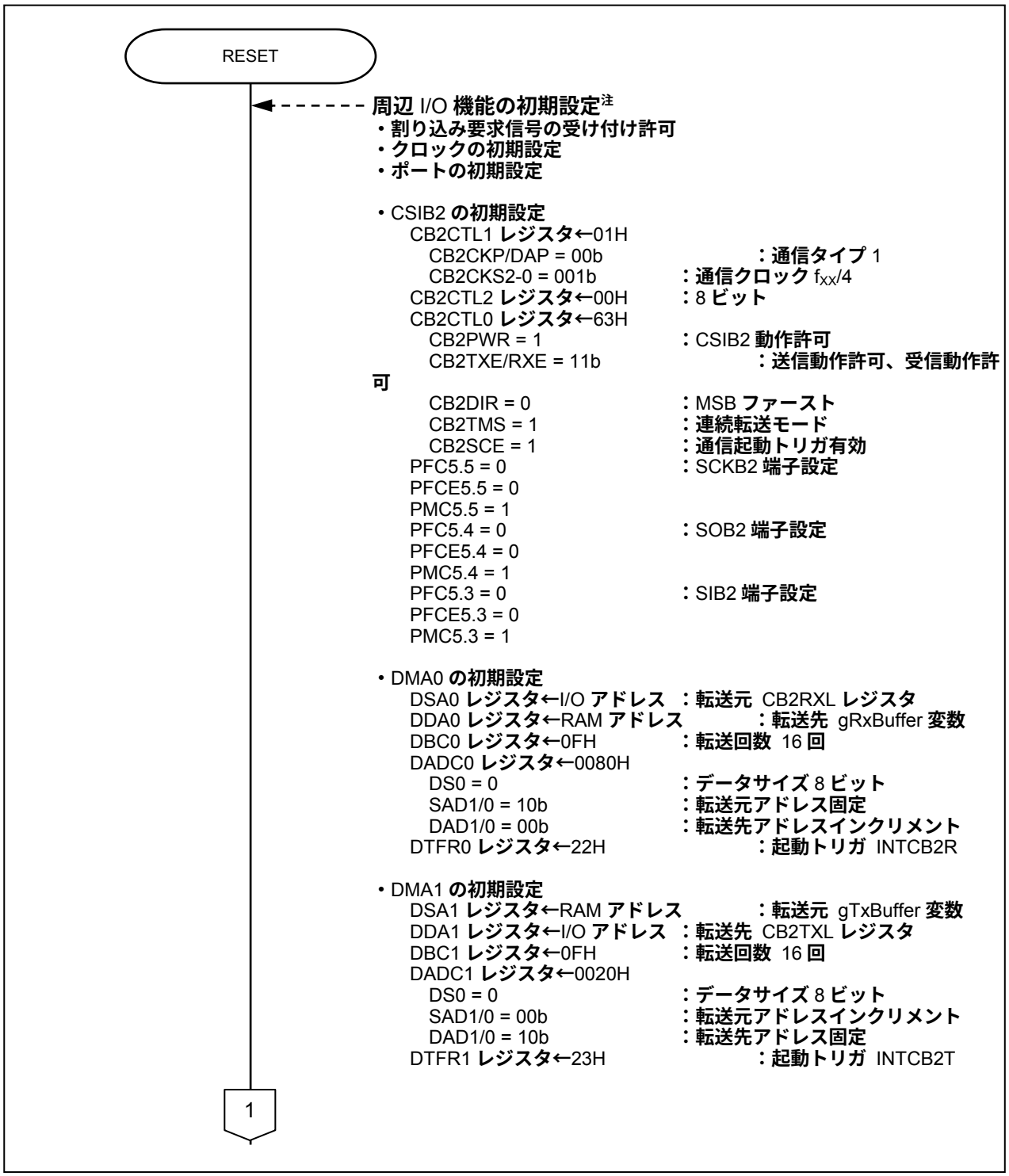


図 5.2 メイン処理（1/2）

注． 周辺 I/O 機能の初期設定はスタート・アップ処理（CG_start.s）で実行されます。

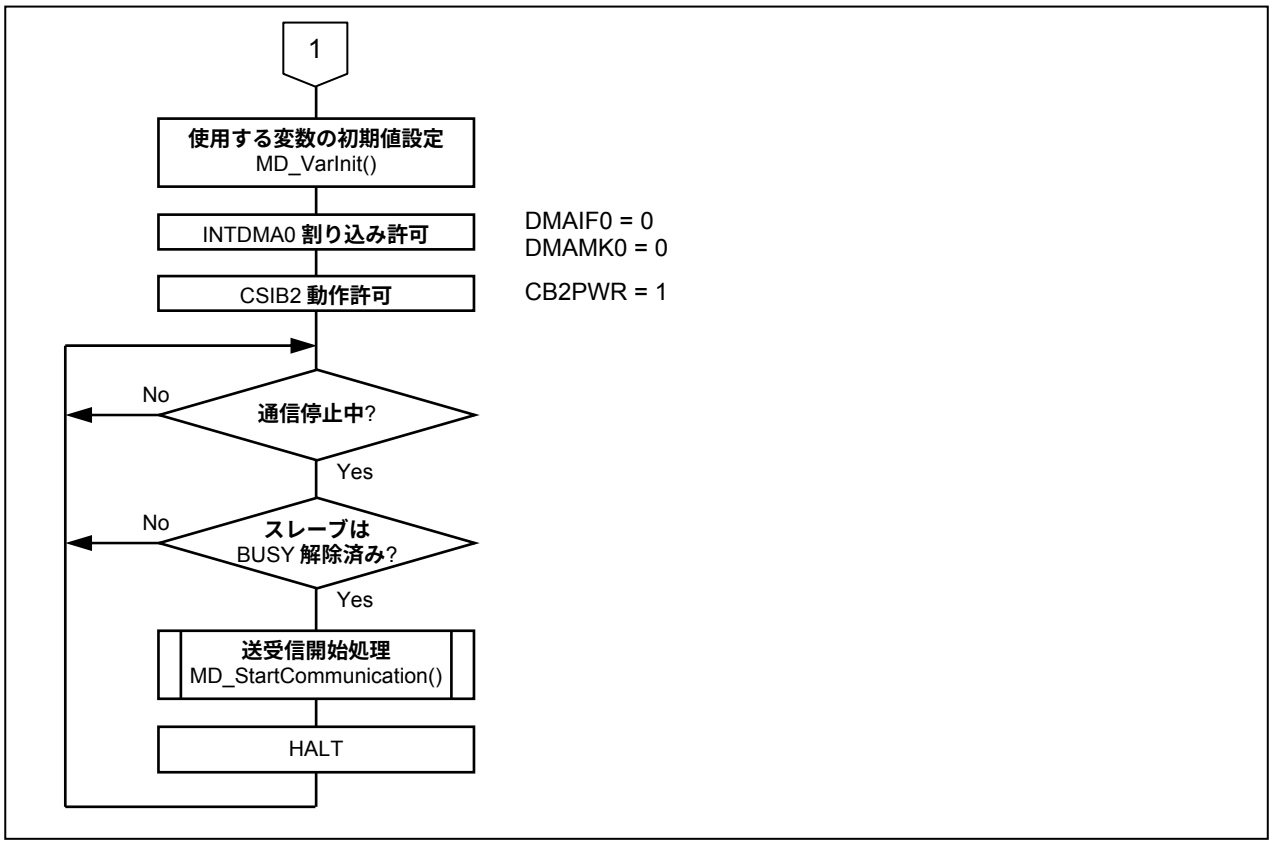

1
使用する変数の初期値設定
MD_VarInit()
INTDMA0 割り込み許可
DMAIF0 = 0
DMAMK0 = 0
CSIB2 動作許可
CB2PWR = 1
No
通信停止中?
Yes
No
スレーブは
BUSY 解除済み?
Yes
送受信開始処理
MD_StartCommunication()
HALT

図 5.2 メイン処理（2/2）

5.8.2 送受信開始処理

図 5.3 に送受信開始処理のフローを示します。

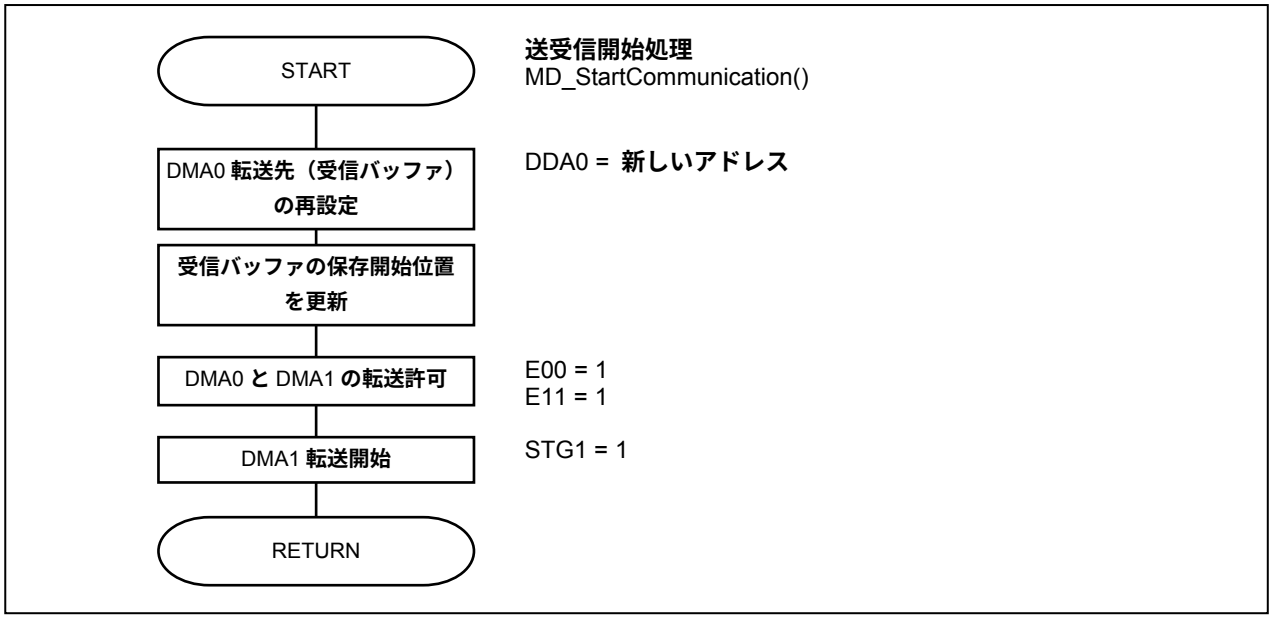


図 5.3 送受信開始処理

5.8.3 INTDMA0 割り込み処理

図 5.4 に INTDMA0 割り込み処理のフローを示します。

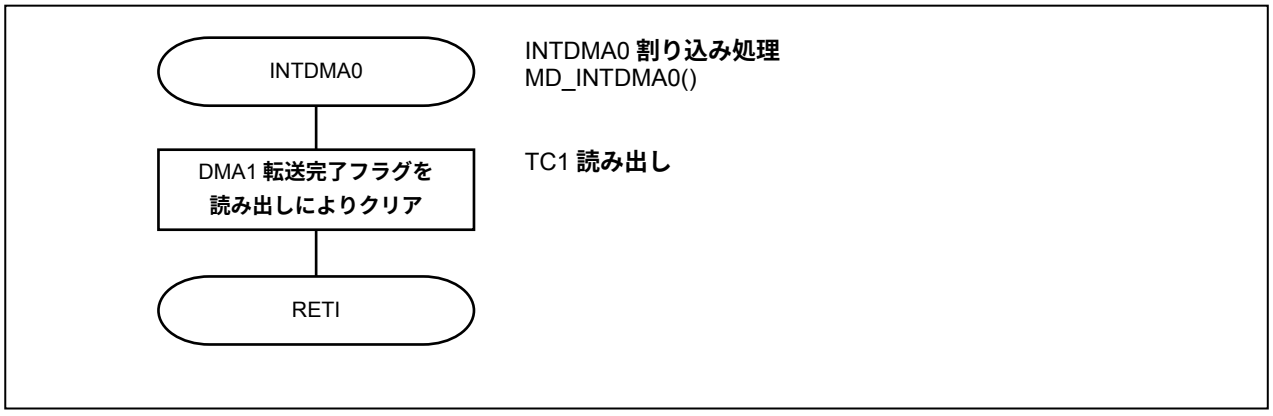
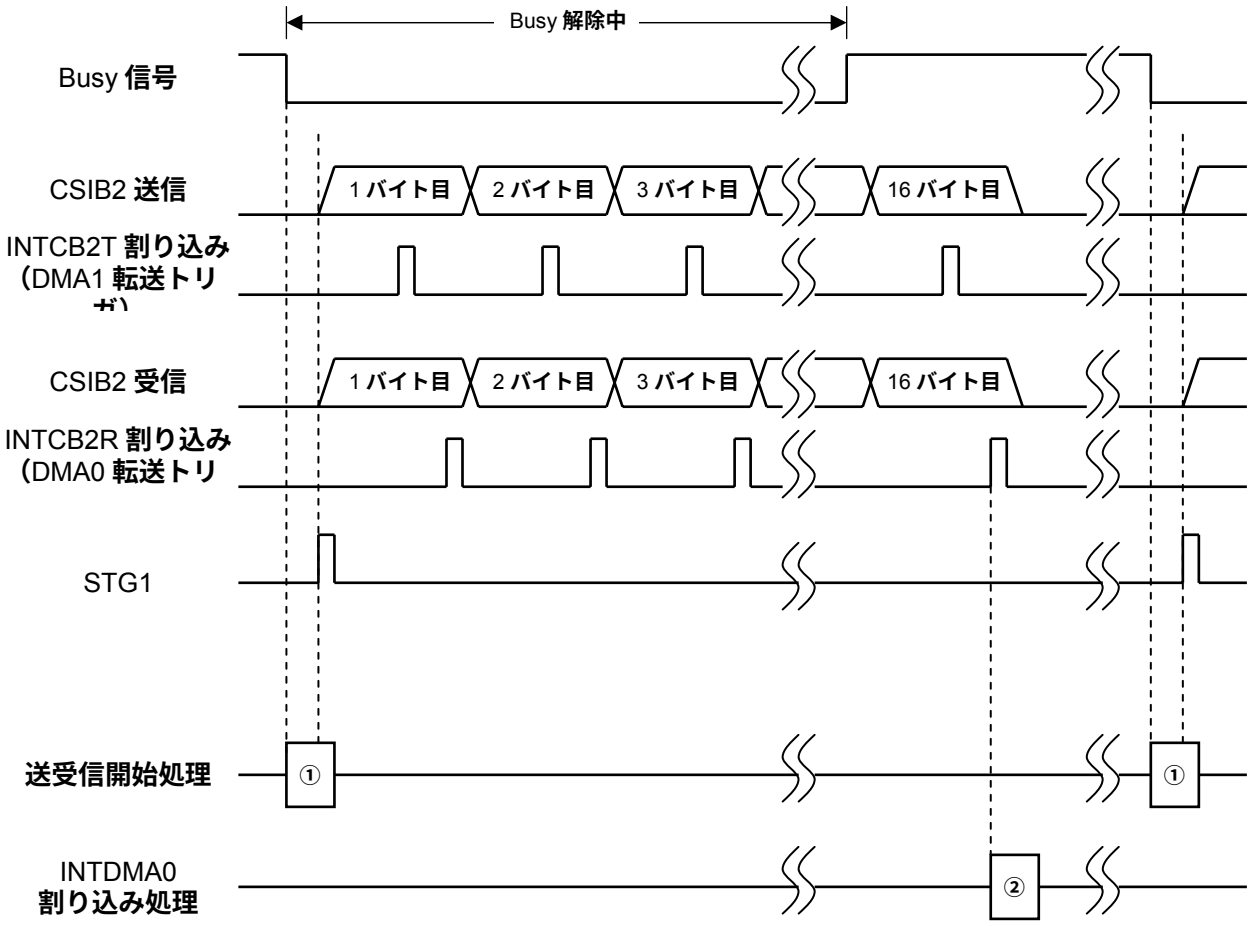


図 5.4 INTDMA0 割り込み処理

また、図 5.5 に 16 バイト連続送受信のタイミングを示します。



① スレーブからの Busy 信号が解除（ロウ・レベル出力）されたときに処理を開始する。DMA0 転送先（受信バッファ）の再設定と DMA0 および DMA1 の転送動作の許可を行い、ソフトウエア起動トリガ（STG1）により DMA1 の転送動作（送信データの書き込み）を開始する。これにより、送受信動作が開始される。また、処理が終了したら、CPU は HALT モードに移行する。

② DMA0 転送（16 回）の終了により割り込み処理が開始され、CPU の HALT モードが解除される。また、割り込み処理内では TC1 フラグをクリアする。

図 5.5 16 バイト連続送受信のタイミング

5.9 CSIB の設定

5.9.1 CSIB2 制御レジスタ 0（CB2CTL0）

CSIB2 のシリアル転送動作を制御する 8 ビット・レジスタです。

8/1 ビット単位でリード／ライト可能です。

リセットにより 01H になります。

CSIB2 制御レジスタ 0（CB2CTL0）
アドレス： FFFFFD20H

7	6	5	4	3	2	1	0
CB2PWR	CB2TXE	CB2RXE	CB2DIR	0	0	CB2TMS	CB2SCE

CB2PWR	CSIB2動作禁止／許可の指定
0	CSIB2動作禁止，CB2STRレジスタをリセットする
1	CSIB2動作許可

CB2TXE	送信動作禁止／許可の指定
0	送信動作禁止
1	送信動作許可

CB2RXE	受信動作禁止／許可の指定
0	受信動作禁止
1	受信動作許可

CB2DIR	データ転送順序
0	MSBファースト
1	LSBファースト

CB2TMS	転送モードの指定
0	シングル転送モード
1	連続転送モード

CB2SCE	起動転送無効／許可の指定
0	通信起動トリガ無効
1	通信起動トリガ有効

図 5.6.1 CSIB2 制御レジスタ 0（CB2CTL0）のフォーマット

5.9.2 CSIB2 制御レジスタ 1（CB2CTL1）

CSIB2 のシリアル転送動作モードを指定する 8 ビット・レジスタです。

8/1 ビット単位でリード／ライト可能です。

リセットにより 00H になります。

CB2CTL0.CB2PWR ビット = 0 の場合のみ書き換えが可能です。

CSIB2 制御レジスタ 1（CB2CTL1）
アドレス： FFFFFD21H

7	6	5	4	3	2	1	0
0	0	0	CB2CKP	CB2DAP	CB2CKS2	CB2CKS1	CB2CKS0

CB2CKP	CB2DAP	SCKB2に対するデータの送受信タイミングの指定
0	**0**	**通信タイプ1**
0	1	通信タイプ2
1	0	通信タイプ3
1	1	通信タイプ4

CB2CKS2	CB2CKS1	CB2CKS0	通信クロック（f_{CCLK}）	モード
0	0	0	$f_{XX}/2$	マスタ・モード
0	**0**	**1**	$f_{XX}/4$	マスタ・モード
0	1	0	$f_{XX}/8$	マスタ・モード
0	1	1	$f_{XX}/16$	マスタ・モード
1	0	0	$f_{XX}/32$	マスタ・モード
1	0	1	$f_{XX}/64$	マスタ・モード
1	1	0	f_{XX}/f_{BRG2}	マスタ・モード
1	1	1	外部クロック（SCKB2）	スレーブ・モード

注．通信クロック（f_{CCLK}）は、8MHz 以下になるように設定してください。

図 5.6.2 CSIB2 制御レジスタ 1（CB2CTL1）のフォーマット

5.9.3 CSIB2 制御レジスタ 2（CB2CTL2）

CSIB2 のシリアル転送データ長を指定する 8 ビットのレジスタです。

8 ビット単位でリード／ライト可能です。

リセットにより 00H になります。

CB2CTL0.CB2PWR ビット = 0，または CB2TXE, CB2RXE ビット = 0 の場合のみ書き換えが可能です。

CSIB2 制御レジスタ 2（CB2CTL2）
アドレス： FFFFFD22H

7	6	5	4	3	2	1	0
0	0	0	0	CB2CL3	CB2CL2	CB2CL1	CB2CL0

CB2CL3	CB2CL2	CB2CL1	CB2CL0	転送データ量
0	**0**	**0**	**0**	8ビット
0	0	0	1	9ビット
0	0	1	0	10ビット
0	0	1	1	11ビット
0	1	0	0	12ビット
0	1	0	1	13ビット
0	1	1	0	14ビット
0	1	1	1	15ビット
1	x	x	x	16ビット

備考 1． 転送データ長が 8/16 ビットではない場合には，CB2TX, CB2RX レジスタの最下位ビットから詰めてデータを準備して使用してください。
2． x：don`t care

図 5.6.3 CSIB2 制御レジスタ 2（CB2CTL2）のフォーマット

5.9.4 CSIB2 の端子設定

CSIB2 を使用する場合，CSIB2 のシリアル・クロック入出力端子（SCKB2）、CSIB2 のシリアル・データ入力端子（SIB2）、CSIB2 のシリアル・データ出力端子（SOB2）を設定する必要があります。

SCKB2 は、ポート 5 ファンクション・コントロール・レジスタ（PFC5）のビット 5、ポート 5 ファンクション・コントロール拡張レジスタ（PFCE5）のビット 5、ポート 5 モード・コントロール・レジスタ（PMC5）のビット 5 により設定します。

表5.5.1 SCKB2の設定

PMC55	PFC55	PFCE55	端子機能の指定
0	x	x	入出力ポート（P55）
1	**0**	**0**	SCKB2入出力
1	1	0	KR5入力
1	1	1	RTP05出力

備考 x：no care

SOB2 は、ポート 5 ファンクション・コントロール・レジスタ（PFC5）のビット 4、ポート 5 ファンクション・コントロール拡張レジスタ（PFCE5）のビット 4、ポート 5 モード・コントロール・レジスタ（PMC5）のビット 4 により設定します。

表5.5.2 SOB2の設定

PMC54	PFC54	PFCE54	端子機能の指定
0	x	x	入出力ポート（P54）
1	**0**	**0**	SOB2入出力
1	1	0	KR4入力
1	1	1	RTP04出力

備考 x：no care

SIB2 は、ポート 5 ファンクション・コントロール・レジスタ（PFC5）のビット 3、ポート 5 ファンクション・コントロール拡張レジスタ（PFCE5）のビット 3、ポート 5 モード・コントロール・レジスタ（PMC5）のビット 3 により設定します。

表5.5.3 SIB2の設定

PMC53	PFC53	PFCE53	端子機能の指定
0	x	x	入出力ポート（P53）
1	**0**	**0**	SIB2入出力
1	0	1	TIQ00入力/KR3入力
1	1	0	TOQ00出力
1	1	1	RTP03出力

備考 x：no care

5.10 DMA 転送の設定

5.10.1 DMA ソース・アドレス・レジスタ 0、1（DSA0、DSA1）

DMA チャネル 0、1 の DMA 転送元アドレス（26 ビット）を設定します。

このレジスタは，DSAnH, DSAnL の 2 つの 16 ビット・レジスタに分かれます。（n = 0-1）

16 ビット単位でリード／ライト可能です。

DMA ソース・アドレス・レジスタ 0（DSA0）
アドレス： DSA0H FFFFF082H、DSA0L FFFFF080H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSA0H	IR	0	0	0	0	0	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSA0L	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0

IR	DMA転送元の指定
0	**外部メモリ，内蔵周辺I/O**
1	内蔵RAM

SA25-SA16	DMA転送元のアドレス（A25-A16）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送元アドレスを保持します。 DMA転送が完了すると，最初に設定されたDMAアドレスが保持されます。

SA15-SA0	DMA転送元のアドレス（A15-A0）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送元アドレスを保持します。 DMA転送が完了すると，最初に設定されたDMAアドレスが保持されます。

注意 1． DSA0H レジスタのビット 14-10 には，必ず “ 0 ” を設定してください。
2． DSA0H, DSA0L レジスタの設定は，DMA 転送禁止状態（DCHC0.E00 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC0.INIT0 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC0.TC0 ビット = 1 の状態）から次の DMA 転送起動までの期間

3． DSA0 レジスタの値を読み出す際，DSA0H レジスタと DSA0L レジスタの 2 つの 16 ビット・レジスタごとに読み出すため，読み出しと更新のタイミングが競合した場合，更新途中の値が読み出されることがあります。
4． 16 ビット・バス幅のミスアライン・データの DMA 転送はサポートしていません。奇数アドレスを転送元に指定した場合,アドレスの最下位ビットは強制的に 0 として扱われます。
5． リセット後，DMA 転送を開始する前に DSA0H, DSA0L, DDA0H, DDA0L, DBC0 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません。

図 5.7.1.1 DMA ソース・アドレス・レジスタ 0（DSA0）のフォーマット

DMA ソース・アドレス・レジスタ 1（DSA1）
アドレス： DSA1H FFFFF08AH、DSA1L FFFFF088H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSA1H	IR	0	0	0	0	0	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSA1L	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0

IR	DMA転送元の指定
0	外部メモリ，内蔵周辺I/O
1	内蔵RAM

SA25-SA16	DMA転送元のアドレス（A25-A16）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送元アドレスを保持します。 DMA転送が完了すると，最初に設定されたDMAアドレスが保持されます。

SA15-SA0	DMA転送元のアドレス（A15-A0）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送元アドレスを保持します。 DMA転送が完了すると，最初に設定されたDMAアドレスが保持されます。

注意 1． DSA1H レジスタのビット 14-10 には，必ず “ 0 ” を設定してください。
2． DSA1H, DSA1L レジスタの設定は，DMA 転送禁止状態（DCHC1.E11 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC1.INIT1 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC1.TC1 ビット = 1 の状態）から次の DMA 転送起動までの期間

3． DSA1 レジスタの値を読み出す際，DSA1H レジスタと DSA1L レジスタの 2 つの 16 ビット・レジスタごとに読み出すため，読み出しと更新のタイミングが競合した場合，更新途中の値が読み出されることがあります。
4． 16 ビット・バス幅のミスアライン・データの DMA 転送はサポートしていません。奇数アドレスを転送元に指定した場合,アドレスの最下位ビットは強制的に 0 として扱われます。
5． リセット後，DMA 転送を開始する前に DSA1H, DSA1L, DDA1H, DDA1L, DBC1 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません。

図 5.7.1.2 DMA ソース・アドレス・レジスタ 1（DSA1）のフォーマット

5.10.2 DMA デスティネーション・アドレス・レジスタ 0、1（DDA0、DDA1）

DMA チャネル 0、1 の DMA 転送先アドレス（26 ビット）を設定します。

このレジスタは，DDAnH, DDAnL の 2 つの 16 ビット・レジスタに分かれます。（n = 0-1）

16 ビット単位でリード／ライト可能です。

DMA デスティネーション・アドレス・レジスタ 0（DDA0）
アドレス： DDA0H FFFFF086H、DDA0L FFFFF084H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDA0H	IR	0	0	0	0	0	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDA0L	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

IR	DMA転送先の指定
0	外部メモリ，内蔵周辺I/O
1	内蔵RAM

DA25-DA16	DMA転送先のアドレス（A25-A16）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送先アドレスを保持します。 DMA転送が終了すると，最初に設定されたDMA転送元アドレスを保持します。

DA15-DA0	DMA転送先のアドレス（A15-A0）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送先アドレスを保持します。 DMA転送が終了すると，最初に設定されたDMA転送元アドレスを保持します。

注意 1． DDA0H レジスタのビット 14-10 には，必ず “ 0 ” を設定してください。
2． DDA0H, DDA0L レジスタの設定は，DMA 転送禁止状態（DCHC0.E00 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC0.INIT0 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC0.TC0 ビット = 1 の状態）から次の DMA 転送起動までの期間

3． DDA0 レジスタの値を読み出す際，DDA0H レジスタと DDA0L レジスタの 2 つの 16 ビット・レジスタごとに読み出すため，読み出しと更新のタイミングが競合した場合，更新途中の値が読み出されることがあります。
4． 16 ビット・バス幅のミスアライン・データの DMA 転送はサポートしていません。奇数アドレスを転送元に指定した場合,アドレスの最下位ビットは強制的に 0 として扱われます。
5． リセット後，DMA 転送を開始する前に DSA0H, DSA0L, DDA0H, DDA0L, DBC0 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません

図 5.7.2.1 DMA デスティネーション・アドレス・レジスタ 0（DDA0）のフォーマット

DMA デスティネーション・アドレス・レジスタ 1（DDA1）
アドレス：　DDA1H FFFFF08EH、DDA1L FFFFF08CH

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDA1H	IR	0	0	0	0	0	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDA1L	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

IR	DMA転送先の指定
0	**外部メモリ，内蔵周辺**I/O
1	**内蔵**RAM

DA25-DA16	DMA転送先のアドレス（A25-A16）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送先アドレスを保持します。 DMA転送が終了すると，最初に設定されたDMA転送元アドレスを保持します。

DA15-DA0	DMA転送先のアドレス（A15-A0）を設定してください（初期値不定）。 DMA転送中は，次のDMA転送先アドレスを保持します。 DMA転送が終了すると，最初に設定されたDMA転送元アドレスを保持します。

注意 1．DDA1H レジスタのビット 14-10 には，必ず“0”を設定してください。
2．DDA1H, DDA1L レジスタの設定は，DMA 転送禁止状態（DCHC1.E11 ビット = 0）である次のいずれかのタイミングで行ってください。
・リセット後から最初の DMA 転送起動までの期間
・DCHC1.INIT1 ビットによるチャネル初期化後から DMA 転送起動までの期間
・DMA 転送完了後（DCHC1.TC1 ビット = 1 の状態）から次の DMA 転送起動までの期間
3．DDA1 レジスタの値を読み出す際，DDA1H レジスタと DDA1L レジスタの 2 つの 16 ビット・レジスタごとに読み出すため，読み出しと更新のタイミングが競合した場合，更新途中の値が読み出されることがあります。
4．16 ビット・バス幅のミスアライン・データの DMA 転送はサポートしていません。奇数アドレスを転送元に指定した場合,アドレスの最下位ビットは強制的に 0 として扱われます。
5．リセット後，DMA 転送を開始する前に DSA1H, DSA1L, DDA1H, DDA1L, DBC1 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません。

図 5.7.2.2　DMA デスティネーション・アドレス・レジスタ 1（DDA1）のフォーマット

5.10.3 DMA 転送カウント・レジスタ 0、1（DBC0、DBC1）

DMA チャネル 0、1 の転送数を設定する 16 ビット・レジスタです。

DMA 転送中は，残りの転送数を保持します。

転送データ単位（8/16 ビット）にかかわらず，1 回の転送につき 1 ずつデクリメントされ，ボローが発生すると転送を終了します。

DMA 転送が完了すると，最初に設定された転送データ数を保持します。

16 ビット単位でリード／ライト可能です。

DMA 転送カウント・レジスタ 0（DBC0）
アドレス： FFFFF0C0H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

BC15-BC0	転送数の設定，またはDMA転送中の残りの転送数
0000H	1回の転送，または残り転送数
0001H	2回の転送，または残り転送数
:	:
000FH	16回の転送，または残り転送数
:	:
FFFFH	65536（2^{16}）回の転送，または残り転送数

注意 1． DBC0 レジスタの設定は，DMA 転送禁止状態（DCHC0.E00 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC0.INIT0 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC0.TC0 ビット = 1 の状態）から次の DMA 転送起動までの期間

2． リセット後，DMA 転送を開始する前に DSA0H, DSA0L, DDA0H, DDA0L, DBC0 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません。

図 5.7.3.1 DMA 転送カウント・レジスタ 0（DBC0）のフォーマット

DMA 転送カウント・レジスタ 1（DBC1）
アドレス： FFFFF0C2H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

BC15-BC0	転送数の設定，またはDMA転送中の残りの転送数
0000H	1回の転送，または残り転送数
0001H	2回の転送，または残り転送数
：	：
000FH	16回の転送，または残り転送数
：	：
FFFFH	65536（2^{16}）回の転送，または残り転送数

注意 1． DBC1 レジスタの設定は，DMA 転送禁止状態（DCHC1.E11 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC1.INIT1 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC1.TC1 ビット = 1 の状態）から次の DMA 転送起動までの期間

2． リセット後，DMA 転送を開始する前に DSA1H, DSA1L, DDA1H, DDA1L, DBC1 レジスタを設定してください。これらのレジスタを設定しないで DMA 転送を開始した場合は，動作を保証しません。

図 5.7.3.2 DMA 転送カウント・レジスタ 1（DBC1）のフォーマット

5.10.4 DMA アドレシング・コントロール・レジスタ 0、1（DADC0、DADC1）

DMA チャネル 0、1 の DMA 転送モードを制御する 16 ビット・レジスタです。

16 ビット単位でリード／ライト可能です。

リセットにより 0000H になります。

DMA アドレシング・コントロール・レジスタ 0（DADC0）

アドレス： FFFFF0D0H

15	14	13	12	11	10	9	8
0	DS0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
SAD1	SAD0	DAD1	DAD0	0	0	0	0

DS0	転送データ・サイズの設定
0	**8ビット**
1	16ビット

SAD1	SAD0	転送元アドレスのカウント方向の設定
0	0	インクリメント
0	1	デクリメント
1	**0**	**固定**
1	1	設定禁止

DAD1	DAD0	転送先アドレスのカウント方向の設定
0	**0**	**インクリメント**
0	1	デクリメント
1	0	固定
1	1	設定禁止

注意 1． DADC0 レジスタのビット 15, 13-8, 3-0 には，必ず“0”を設定してください。

2． DADC0 レジスタの設定は，DMA 転送禁止状態（DCHC0.E00 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC0.INIT0 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC0.TC0 ビット = 1 の状態）から次の DMA 転送起動までの期間

3． DS00 ビットは転送データ・サイズを設定するものであり，バス・サイジングを制御するものではありません。

4． 転送データ・サイズを 16 ビットに設定した場合（DS0 ビット = 1），奇数アドレスから始まる転送はできません。下位アドレスの 1 ビットを“0”にアラインしたアドレスから必ず転送を開始します。

5． 内蔵周辺 I/O レジスタを対象（転送元／転送先）とする DMA 転送の場合，必ずレジスタ・サイズと同じ転送サイズを指定してください。たとえば，8 ビットのレジスタに対する DMA 転送の場合は，必ず（8 ビット）転送を指定してください。

図 5.7.4.1 DMA アドレシング・コントロール・レジスタ 0（DADC0）のフォーマット

DMA アドレシング・コントロール・レジスタ 1（DADC1）
アドレス： FFFFF0D2H

15	14	13	12	11	10	9	8
0	DS0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
SAD1	SAD0	DAD1	DAD0	0	0	0	0

DS0	転送データ・サイズの設定
0	**8ビット**
1	16ビット

SAD1	SAD0	転送元アドレスのカウント方向の設定
0	**0**	**インクリメント**
0	1	デクリメント
1	0	固定
1	1	設定禁止

DAD1	DAD0	転送先アドレスのカウント方向の設定
0	0	インクリメント
0	1	デクリメント
1	**0**	**固定**
1	1	設定禁止

注意 1. DADC1 レジスタのビット 15, 13-8, 3-0 には，必ず“0”を設定してください。
2. DADC1 レジスタの設定は，DMA 転送禁止状態（DCHC1.E11 ビット = 0）である次のいずれかのタイミングで行ってください。
   - リセット後から最初の DMA 転送起動までの期間
   - DCHC1.INIT1 ビットによるチャネル初期化後から DMA 転送起動までの期間
   - DMA 転送完了後（DCHC1.TC1 ビット = 1 の状態）から次の DMA 転送起動までの期間
3. DS10 ビットは転送データ・サイズを設定するものであり，バス・サイジングを制御するものではありません。
4. 転送データ・サイズを 16 ビットに設定した場合（DS0 ビット = 1），奇数アドレスから始まる転送はできません。下位アドレスの 1 ビットを“0”にアラインしたアドレスから必ず転送を開始します。
5. 内蔵周辺 I/O レジスタを対象（転送元／転送先）とする DMA 転送の場合，必ずレジスタ・サイズと同じ転送サイズを指定してください。たとえば，8 ビットのレジスタに対する DMA 転送の場合は，必ず（8 ビット）転送を指定してください。

図 5.7.4.2 DMA アドレシング・コントロール・レジスタ 1（DADC1）のフォーマット

5.10.5 DMA チャネル・コントロール・レジスタ 0、1（DCHC0、DCHC1）

DMA チャネル 0、1 の DMA 転送動作を制御する 8 ビット・レジスタです。

8/1 ビット単位でリード／ライト可能です（ただし，ビット 7 はリードのみ，ビット 1, 2 はライトのみ可能です。ビット 1, 2 をリードした場合は 0 が読み出されます。）。

リセットにより 00H になります。

DMA チャネル・コントロール・レジスタ 0（DCHC0）
アドレス： FFFFF0E0H

7	6	5	4	3	2	1	0
TC0	0	0	0	0	INIT0	STG0	E00

TC0	DMAチャネル0のDMA転送の完了／未完了を示すステータス・フラグ
0	DMA転送未完了
1	DMA転送完了

INIT0	DMA転送が禁止された状態で（E00ビット = 0），INIT0ビットをセット（1）するとDMA転送のステータスを初期化できます。リードのみ可能です。

STG0	DMA転送のソフトウエア起動トリガです。 DMA転送が許可の状態（TC0ビット = 0, E00ビット = 1）でこのビットをセット（1）するとDMA転送を開始します。

E00	DMAチャネル0のDMA転送の許可／禁止の設定
0	DMA転送の禁止
1	DMA転送の許可 （DMA転送が完了（ターミナル・カウント発生）すると，自動的にクリア（0）されます。）

注意 1． DCHC0 レジスタのビット 6-3 には，必ず“ 0 ”を設定してください。
2． DMA 転送完了時（ターミナル・カウント時）は，E00 ビットのクリア（0）→TC0 ビットのセット（1）の順で各ビットの更新が行われます。そのため，DCHC0 レジスタの各ビットの更新途中に DCHC0 レジスタを読み出した場合，「転送未完了，かつ転送禁止」の状態を示す値（TC0 ビット = 0，かつ E00 ビット = 0）が読み出されることがあります。

図 5.7.5.1 DMA チャネル・コントロール・レジスタ 0（DCHC0）のフォーマット

DMA チャネル・コントロール・レジスタ 1（DCHC1）
アドレス： FFFFF0E2H

7	6	5	4	3	2	1	0
TC1	0	0	0	0	INIT1	STG1	E11

TC1	DMAチャネル1のDMA転送の完了／未完了を示すステータス・フラグ
0	DMA転送未完了
1	DMA転送完了

INIT1	DMA転送が禁止された状態で（E11ビット = 0），INIT1ビットをセット（1）するとDMA転送のステータスを初期化できます。リードのみ可能です。

STG1	DMA転送のソフトウエア起動トリガです。 DMA転送が許可の状態（TC1ビット = 0, E11ビット = 1）でこのビットをセット（1）するとDMA転送を開始します。

E11	DMAチャネル1のDMA転送の許可／禁止の設定
0	DMA転送の禁止
1	DMA転送の許可 （DMA転送が完了（ターミナル・カウント発生）すると，自動的にクリア（0）されます。）

注意 1． DCHC1 レジスタのビット 6-3 には，必ず“ 0 ”を設定してください。
2． DMA 転送完了時（ターミナル・カウント時）は，E11 ビットのクリア（0）→TC1 ビットのセット（1）の順で各ビットの更新が行われます。そのため，DCHC1 レジスタの各ビットの更新途中に DCHC1 レジスタを読み出した場合，「転送未完了，かつ転送禁止」の状態を示す値（TC1 ビット = 0，かつ E11 ビット = 0）が読み出されることがあります。

図 5.7.5.2 DMA チャネル・コントロール・レジスタ 1（DCHC1）のフォーマット

5.10.6 DMA トリガ要因レジスタ 0、1（DTFR0、DTFR1）

内蔵周辺 I/O からの割り込み要求信号による DMA 転送開始トリガを制御する 8 ビット・レジスタです。

このレジスタで設定した割り込み要求信号が，DMA 転送の起動要因になります。

8 ビット単位でリード／ライト可能です。ただし，DFn ビットのみ 1 ビット単位でリード／ライト可能です。

リセットにより 00H になります。

DMA トリガ要因レジスタ 0（DTFR0）
アドレス： FFFFF810H

7	6	5	4	3	2	1	0
DF0	0	IFC05	IFC04	IFC03	IFC02	IFC01	IFC00

DF0	DMA転送要求ステータス・フラグ
0	DMA転送要求なし
1	DMA転送要求あり

IFC05	IFC04	IFC03	IFC02	IFC01	IFC00	割り込み要因
0	0	0	0	0	0	割り込みによるDMA要求禁止
0	0	0	0	0	1	INTP0
0	0	0	0	1	0	INTP1
：						：
1	**0**	**0**	**0**	**1**	**0**	INTCB2R
1	0	0	0	1	1	INTCB2T
：						：

注意 1． IFC05-IFC00 ビットの設定は，DMA 転送禁止状態（DCHC0.E00 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC0.INIT0 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC0.TC0 ビット = 1 の状態）から次の DMA 転送起動までの期間

2． DTFR0 レジスタの設定を変更する場合は必ず次の手順で行ってください。
- DMA0 動作を停止（DCHC0.E00 ビット = 0）する。
- DTFR0 レジスタの設定を変更する。
  （必ず DF0 ビット = 0 とし，かつ 8 ビット操作で行ってください）
- DF0 ビット = 0 であることを確認する。
  （あらかじめ，割り込み発生要因の動作を停止しておいてください）
- DMA0 動作を許可（E00 ビット = 1）する。

3． スタンバイ・モード（IDLE1, IDLE2, STOP, サブ IDLE モード）中に発生した割り込み要求は，DMA 転送サイクルの起動要因にはなりません（DF0 ビットもセット（1）されません）。

4． IFC05-IFC00 ビットで任意の DMA 起動要因を選択したあとは，DMA 転送の許可／禁止にかかわらず，選択した内蔵周辺 I/O からの割り込みが発生すると DF0 ビットはセット（1）されます。この状態で DMA 許可とした場合，ただちに DMA 転送が起動されます。

図 5.7.6.1 DMA トリガ要因レジスタ 0（DTFR0）のフォーマット

DMA トリガ要因レジスタ 1（DTFR1）
アドレス： FFFFF812H

7	6	5	4	3	2	1	0
DF1	0	IFC15	IFC14	IFC13	IFC12	IFC11	IFC10

DF1	DMA転送要求ステータス・フラグ
0	DMA転送要求なし
1	DMA転送要求あり

IFC15	IFC14	IFC13	IFC12	IFC11	IFC10	割り込み要因
0	0	0	0	0	0	割り込みによるDMA要求禁止
0	0	0	0	0	1	INTP0
0	0	0	0	1	0	INTP1
			:			:
1	0	0	0	1	0	INTCB2R
1	**0**	**0**	**0**	**1**	**1**	INTCB2T
			:			:

注意 1. IFC15-IFC10 ビットの設定は，DMA 転送禁止状態（DCHC1.E11 ビット = 0）である次のいずれかのタイミングで行ってください。
- リセット後から最初の DMA 転送起動までの期間
- DCHC1.INIT1 ビットによるチャネル初期化後から DMA 転送起動までの期間
- DMA 転送完了後（DCHC1.TC1 ビット = 1 の状態）から次の DMA 転送起動までの期間

2. DTFR1 レジスタの設定を変更する場合は必ず次の手順で行ってください。
- DMA1 動作を停止（DCHC1.E11 ビット = 0）する。
- DTFR1 レジスタの設定を変更する。
  （必ず DF1 ビット = 0 とし，かつ 8 ビット操作で行ってください）
- DF1 ビット = 0 であることを確認する。
  （あらかじめ，割り込み発生要因の動作を停止しておいてください）
- DMA1 動作を許可（E11 ビット = 1）する。

3. スタンバイ・モード（IDLE1, IDLE2, STOP, サブ IDLE モード）中に発生した割り込み要求は，DMA 転送サイクルの起動要因にはなりません（DF1 ビットもセット（1）されません）。

4. IFC15-IFC10 ビットで任意の DMA 起動要因を選択したあとは，DMA 転送の許可／禁止にかかわらず，選択した内蔵周辺 I/O からの割り込みが発生すると DF0 ビットはセット（1）されます。この状態で DMA 許可とした場合，ただちに DMA 転送が起動されます。

図 5.7.6.2 DMA トリガ要因レジスタ 1（DTFR1）のフォーマット

6. サンプルコード

以下に V850ES/JG3-L 用のサンプルコードを記載します。

・CG_main.c

```
/*
***********************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_main.c
* Abstract:       This file implements main function.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
***********************************************************************
*/

/*
***********************************************************************
** Pragma directive
***********************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************
** Include files
***********************************************************************
*/
```

```
#include "CG_macrodriver.h"
#include "CG_system.h"
#include "CG_port.h"
#include "CG_serial.h"
#include "CG_dma.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

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

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function implements main function.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void main(void)
{
    /* Start user code. Do not edit comment generated here */

    MD_VarInit();    /* Variable initialization */

    DMAIF0 = 0;     /* Clear INTDMA0 flag */
    DMAMK0 = 0;    /* Enable INTDMA0 */
```

```
        CB2PWR = 1;   /* Enable CSIB2 */

        while (1U)
        {
                /* Communication stopped */
                if( CB2STR.7 == 0 )
                {
                        /* Slave is ready */
                        if( P5.0 == 0 )
                        {
                                MD_StartCommunication();        /* Start communication */

                                HALT();                         /* Cpu standby */

                                NOP();
                                NOP();
                                NOP();
                                NOP();
                                NOP();
                        }
                }
        }

        /* End user code. Do not edit comment generated here */
}
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

・CG_serial.c

下記の関数は本サンプルコードでは使用しません。

・CSIB2_Start

→CSIB2 の動作を開始します。

・CSIB2_Stop

→CSIB2 の動作を停止します。

・CSIB2_SendReceiveData

→CSIB2 の送信処理を行います。

```
/*
************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_serial.c
* Abstract:       This file implements device driver for Serial module.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
************************************************************************
*/

/*
************************************************************************
** Pragma directive
************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
```

```
************************************************************************
** Include files
************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

/*
************************************************************************
**  Global define
************************************************************************
*/
/*volatile UCHAR   *gpCsib2RxAddress;*/        /* csib2 receive buffer address */
/*volatile USHORT gCsib2RxLen;*/               /* csib2 receive data length */
/*volatile USHORT gCsib2RxCnt;*/               /* csib2 receive data count */
/*volatile UCHAR   *gpCsib2TxAddress;*/        /* csib2 send buffer address */
/*volatile USHORT gCsib2TxLen;*/               /* csib2 send data length */
/*volatile USHORT gCsib2TxCnt;*/               /* csib2 send data count */
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes the CSIB2 module.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void CSIB2_Init(void)
```

```
{
        CB2PWR = 0U; /* disable CSIB2 */
        CB2TMK = 1U;  /* disable INTCB2T */
        CB2TIF = 0U;    /* clear INTCB2T flag */
        CB2RMK = 1U; /* disable INTCB2R */
        CB2RIF = 0U;    /* clear INTCB2R flag */
        /* Set INTCB2T level low priority */
        CB2TIC |= 0x07U;
        /* Set INTCB2R level low priority */
        CB2RIC |= 0x07U;
        /* Serial clock 5MHz */
        CB2CTL1 = _00_CSIB_DATA_TIMING1 | _01_CSIB_CLOCK_2;
        /* Data length 8bits */
        CB2CTL2 = _00_CSIB_DATA_LENGTH_8;
        /* Enable transmit/receive operation, MSB-first transfer,
            Continuous transfer mode, Communication start trigger valid */
        CB2CTL0 = _40_CSIB_TRANSMIT_ENABLE | _20_CSIB_RECEIVE_ENABLE |
_00_CSIB_DATA_MSB | _02_CSIB_MODE_CONTINUOUS | _01_CSIB_STARTTRG_VALID;
        /* Set SCKB2 pin */
        PFC5 &= 0xDFU;
        PFCE5 &= 0xDFU;
        PMC5 |= 0x20U;
        /* Set SOB2 pin */
        PFC5 &= 0xEFU;
        PFCE5 &= 0xEFU;
        PMC5 |= 0x10U;
        /* Set SIB2 pin */
        PFC5 &= 0xF7U;
        PFCE5 &= 0xF7U;
        PMC5 |= 0x08U;
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function starts CSIB2 transfer.
**
**  Parameters:
**      None
```

```
**
**    Returns:
**        None
**
**-------------------------------------------------------------------------
*/
/*void CSIB2_Start(void)
{
        CB2RIF = 0U;*/ /* clear INTCB2R flag */
/*      CB2RMK = 0U;*/          /* enable INTCB2R */
/*      CB2TIF = 0U;*/ /* clear INTCB2T flag */
/*      CB2TMK = 0U;*/          /* enable INTCB2T */
/*      CB2PWR = 1U;*/          /* enable CSIB2 */
/*}*/
/*
**-------------------------------------------------------------------------
**
**    Abstract:
**        This function stops CSIB2 transfer.
**
**    Parameters:
**        None
**
**    Returns:
**        None
**
**-------------------------------------------------------------------------
*/
/*void CSIB2_Stop(void)
{
        CB2PWR = 0U;*/          /* disable CSIB2 */
/*      CB2RMK = 1U;*/          /* disable INTCB2R */
/*      CB2RIF = 0U;*/ /* clear INTCB2R flag */
/*      CB2TMK = 1U;*/          /* disable INTCB2T */
/*      CB2TIF = 0U;*/ /* clear INTCB2T flag */
/*}*/
/*
**-------------------------------------------------------------------------
**
```

```
**  Abstract:
**      This function sends and receives CSIB2 data.
**
**  Parameters:
**      txbuf: transfer buffer pointer
**      txnum: buffer size
**      rxbuf: receive buffer pointer
**
**  Returns:
**      MD_OK
**      MD_ARGERROR
**
**-------------------------------------------------------------------------
*/
/*MD_STATUS CSIB2_SendReceiveData(UCHAR *txbuf, USHORT txnum, UCHAR *rxbuf)
{
    MD_STATUS status = MD_OK;

    if (txnum < 1U)
    {
        status = MD_ARGERROR;
    }
    else
    {
        gCsib2TxLen = txnum;*//* send data length */
/*      gCsib2TxCnt = txnum;*/ /* send data count */
/*      gpCsib2TxAddress = txbuf;*/     /* send buffer pointer */
/*      gpCsib2RxAddress = rxbuf;*/     /* receive buffer pointer */
/*      gCsib2RxCnt = 0U;
        if (CB2CTL2 !=  _00_CSIB_DATA_LENGTH_8)
        {*/
            /* Transfer data length is more than 8 bit */
            /* The data number should be even */
/*          if ((txnum & 0x1U) == 0x1U)
            {
                status = MD_ARGERROR;
            }
            else
            {
```

```
                                CB2TMK = 1U;*/          /* disable INTCB2T */
/*                              CB2TX = *(USHORT* )gpCsib2TxAddress;*/      /* start by writing data to
CB2TX */
/*                              gpCsib2TxAddress += 2U;
                                gCsib2TxCnt -= 2U;
                                CB2TMK = 0U;*/          /* enable INTCB2T */
/*                      }
                }
                else
                {*/
                        /* Transfer data length is 8 bit */
/*                      CB2TMK = 1U;*/          /* disable INTCB2T */
/*                      CB2TXL = *gpCsib2TxAddress;*/           /* start by writing data to CB2TX */
/*                      gpCsib2TxAddress++;
                        gCsib2TxCnt--;
                        CB2TMK = 0U;*/          /* enable INTCB2T */
/*              }
        }

        return (status);
}*/

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

・CG_dma.c

下記の関数は本サンプルコードでは使用しません。

・DMA0_Enable

→DMA0 **転送を許可します。**

・DMA0_Disable

→DMA0 **転送を禁止します。**

・DMA1_Enable

→DMA1 **転送を許可します。**

・DMA1_Disable

→DMA1 **転送を禁止します。**

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_dma.c
* Abstract:       This file implements device driver for DMA module.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
```

```
#pragma interrupt INTDMA0 MD_INTDMA0
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_dma.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/
/* Start user code for global. Do not edit comment generated here */
        #define TXDNUM      16      /* Number of transmit data */
        #define BUFNUM      3       /* Number of buffer area */
static UCHAR gRxBuffer[(TXDNUM * BUFNUM)];      /* Receive data buffer */
static UCHAR gTxBuffer[TXDNUM];                 /* Transmit data buffer */
static UCHAR gBuffCounter;                      /* Counter for buffer area */
static const UCHAR gTxTable[TXDNUM] = {         /* Transmit data */
          0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
        ,0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f
};
/* End user code. Do not edit comment generated here */

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes the DMA0 module.
**
**  Parameters:
**      None
**
```

```
**    Returns:
**          None
**
**------------------------------------------------------------------------
*/
void DMA0_Init(void)
{
          E00 = 0U;          /* disable DMA0 transfer */
          DMAMK0 = 1U; /* disable INTDMA0 interrupt */
          DMAIF0 = 0U;   /* clear INTDMA0 interrupt flag */
          /* Set INTDMA0 level low priority */
          DMAIC0 |= 0x07U;
          INIT0 = 1U;        /* enable DMA0 initialization */
          /* Set CB2RXL address for DMA0 source address */
          DSA0H = _0000_DMA_SRC_AREA_EXMEMORIO | _03FF_DMA0_SRC_ADDRESS_HIGH;
          DSA0L = _FD24_DMA0_SRC_ADDRESS_LOW;
          /* Set receive data buffer address for DMA0 destination address */
          DDA0H = _8000_DMA_DST_AREA_INRAM | _03FF_DMA0_DST_ADDRESS_HIGH;
          /*DDA0L = _510C_DMA0_DST_ADDRESS_LOW;*/
          DDA0L = (USHORT)&gRxBuffer;
          /* DMA0 transfer count : 16 */
          DBC0    = _000F_DMA0_NUM_COUNT;
          /* Transfer data 8bits, Transfer source address fix, Destination address increment */
          DADC0 = _0000_DMA_DATA_SIZE_8 | _0080_DMA_SRC_DIR_FIX | _0000_DMA_DST_DIR_INC;
          /* DMA0 trigger : INTCB2R */
          DTFR0 = _22_DMA_TRIGGER_INTCB2R;
}
/*
**------------------------------------------------------------------------
**
**    Abstract:
**          This function enables DMA0 module operation.
**
**    Parameters:
**          None
**
**    Returns:
**          None
**
```

```
**-------------------------------------------------------------------------
*/
/*void DMA0_Enable(void)
{
        DMAIF0 = 0U;*/ /* clear INTDMA0 interrupt flag */
/*      DMAMK0 = 0U;*/          /* enable INTDMA0 interrupt */
/*      E00 = 1U;*/     /* enable DMA0 transfer */
/*}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function disables DMA0 module operation.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
/*void DMA0_Disable(void)
{
        E00 = 0U;*/     /* disable DMA0 transfer */
/*      DMAMK0 = 1U;*/          /* disable INTDMA0 interrupt */
/*      DMAIF0 = 0U;*/ /* clear INTDMA0 interrupt flag */
/*}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes the DMA1 module.
**
**  Parameters:
**      None
**
**  Returns:
**      None
```

```
**
**------------------------------------------------------------------------
*/
void DMA1_Init(void)
{
        E11 = 0U;         /* disable DMA1 transfer */
        DMAMK1 = 1U; /* disable INTDMA1 interrupt */
        DMAIF1 = 0U;   /* clear INTDMA1 interrupt flag */
        INIT1 = 1U;       /* enable DMA1 initialization */
        /* Set transmit data buffer address for DMA1 source address */
        DSA1H = _8000_DMA_SRC_AREA_INRAM | _03FF_DMA1_SRC_ADDRESS_HIGH;
        /*DSA1L = _513C_DMA1_SRC_ADDRESS_LOW;*/
        DSA1L = (USHORT)&gTxBuffer;
        /* Set CB2TXL address for DMA1 destination address */
        DDA1H = _0000_DMA_DST_AREA_EXMEMORIO | _03FF_DMA1_DST_ADDRESS_HIGH;
        DDA1L = _FD26_DMA1_DST_ADDRESS_LOW;
        /* DMA1 transfer count : 16 */
        DBC1    = _000F_DMA1_NUM_COUNT;
        /* Transfer data 8bits, Transfer source address increment, Destination address fix */
        DADC1 = _0000_DMA_DATA_SIZE_8 | _0000_DMA_SRC_DIR_INC | _0020_DMA_DST_DIR_FIX;
        /* DMA1 trigger : INTCB2T */
        DTFR1 = _23_DMA_TRIGGER_INTCB2T;
}
/*
**------------------------------------------------------------------------
**
**    Abstract:
**        This function enables DMA1 module operation.
**
**    Parameters:
**        None
**
**    Returns:
**        None
**
**------------------------------------------------------------------------
*/
/*void DMA1_Enable(void)
{
```

```
        E11 = 1U;*/     /* enable DMA1 transfer */
/*}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function disables DMA1 module operation.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
/*void DMA1_Disable(void)
{
        E11 = 0U;*/     /* disable DMA1 transfer */
/*}*/

/* Start user code for adding. Do not edit comment generated here */
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function is INTDMA0 interrupt service routine.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTDMA0(void)
{
        UCHAR work;
```

```
        /* Read and clear TC1 for next DMA1 trigger setting */
        work = TC1;
        work = TC1;
        work = TC1;
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      Variable initialization.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void MD_VarInit( void )
{
        UCHAR count;

        /* Store transmit data for RAM */
        for( count=0; count<TXDNUM; count++ )
        {
                gTxBuffer[count] = gTxTable[count];
        }
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      Communication start function.
**
**  Parameters:
**      None
**
```

```
**    Returns:
**          None
**
**------------------------------------------------------------------------
*/
void MD_StartCommunication( void )
{
        /* Set next area of receive data buffer for DMA0 transfer destination */
        DDA0L = ( USHORT )( gRxBuffer + (gBuffCounter * TXDNUM) );

        /* Count number of buffer area */
        gBuffCounter++;
        gBuffCounter %= BUFNUM;

        E00 = 1;        /* Enable DMA0 transfer */
        E11 = 1;        /* Enable DMA1 transfer */

        STG1 = 1;       /* Set DMA1 trigger, start CSI communication */
}
/* End user code. Do not edit comment generated here */
```

• CG_systeminit.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_systeminit.c
* Abstract:       This file implements system initializing function.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_system.h"
#include "CG_port.h"
#include "CG_serial.h"
```

```
#include "CG_dma.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

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

extern unsigned long _S_romp;  /* ROMization symbol */

void systeminit(void);

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes each macro.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void systeminit(void)
{
        _rcopy(&_S_romp, -1);   /* Call ROMization */
        DI();     /* disable interrupt */
        PORT_Init();
        CSIB2_Init();
        DMA0_Init();
        DMA1_Init();
```

```
    EI();      /* enable interrupt */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

• CG_system.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_system.c
* Abstract:       This file implements device driver for System module.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_system.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

```
/*#include "CG_userdefine.h"*/

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

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes the clock generator module.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void CLOCK_Init(void)
{
    UCHAR psval = 0U;
    UINT i = 0U;
    /* Set WDT2 (stop) */
    WDTM2 = _00_WDT2_OPERMODE_STOP;
    /* Set fXX and fCPU */
    psval = _00_CG_SUBCLK_FEEDBACK_USE | _00_CG_MAINCLK_ENABLE |
_00_CG_MAINCLK_FEEDBACK_USE | _00_CG_CPUCLK_MAIN0;
    PRCMD = psval;
    PCC = psval;    /* fCPU = fXX */
    /* Select PLL Mode */
    SELPLL = 1U;
    /* Set fR (disable) */
    RSTOP = 1U;
```

```
        /* Set fBRG (disable) */
        /*BGCE0 = 0U;*/
        /* Set Stand-by function */
        /*psval = _00_CG_STANDBY_INTWDT2EN | _00_CG_STANDBY_NMIEN |
_00_CG_STANDBY_MASKIEN;
        PRCMD = psval;
        PSC = psval;*/
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

• CG_port.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename:       CG_port.c
* Abstract:       This file implements device driver for PORT module.
* APIlib:CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:         uPD70F3796
* Compiler:       CA850
* Creation date: 2011/07/19
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_port.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

```
/*#include "CG_userdefine.h"*/

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

/*
**-----------------------------------------------------------------------------
**
**    Abstract:
**        This function initializes setting for Port I/O.
**
**    Parameters:
**        None
**
**    Returns:
**        None
**
**-----------------------------------------------------------------------------
*/
void PORT_Init(void)
{
        /* P50 : Busy input */
        /* P53/SIB2 : SIB2 */
        /* P54/SOB2 : SOB2 */
        /* P55/SCKB2 : SCKB2 */
        /* P40-P42,PCM0,PDL5 : On-chip debug */
        /* Other port : Low level output(no use) */

        /* PM0    : 0b10000011 */
        PM0 = _00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT
| _00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _83_PM0_DEFAULT;
        /* PM1    : 0b11111100 */
        PM1 = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _FC_PM1_DEFAULT;
        /* PM3L : 0b00111000 */
```

```
    PM3L = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT |
_38_PM3L_DEFAULT;

    /* PM3H : 0b11111100 */

    PM3H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _FC_PM3H_DEFAULT;

    /* PM4   : No care */

    /* PM5   : 0b11011101 */

    PM5 = _01_PMn0_MODE_INPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_08_PMn3_MODE_UNUSED | _10_PMn4_MODE_UNUSED | _20_PMn5_MODE_UNUSED |
_C0_PM5_DEFAULT;

    /* PM7L : 0b00000000 */

    PM7L = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PM7H : 0b11110000 */

    PM7H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _F0_PM7H_DEFAULT;

    /* PM9L : 0b00000000 */

    PM9L = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PM9H : 0b00000000 */

    PM9H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PMCM : 0b11110000 */

    PMCM = _00_PMn0_MODE_OCD | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _F0_PMCM_DEFAULT;

    /* PMCT : 0b10101100 */

    PMCT = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn4_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _AC_PMCT_DEFAULT;

    /* PMDH : 0b11100000 */

    PMDH = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_E0_PMDH_DEFAULT;

    /* PMDLL: 0b00000000 */

    PMDLL = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_20_PMn5_MODE_UNUSED | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PMDLH: 0b00000000 */

    PMDLH = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT |
_00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;


    /* PMC0   : 0b00000000 */
```

```
    PMC0 = _00_PMCn2_OPER_PORT | _00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT |
_00_PMCn5_OPER_PORT | _00_PMCn6_OPER_PORT;
    /* PMC3L : 0b00000000 */
    PMC3L = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMC3H : 0b00000000 */
    PMC3H = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT;
    /* PMC4   : No care */
    /* PMC5   : 0b00000000 */
    PMC5 = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT;
    /* PMC9L : 0b00000000 */
    PMC9L = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMC9H : 0b00000000 */
    PMC9H = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMCCM : 0b00000000 */
    PMCCM = _00_PMCn0_OPER_OCD | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT;
    /* PMCCT : 0b00000000 */
    PMCCT = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn4_OPER_PORT |
_00_PMCn6_OPER_PORT;
    /* PMCDH : 0b00000000 */
    PMCDH = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT;
    /* PMCDLL: 0b00100000 */
    PMCDLL = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _20_PMCn5_OPER_ALTER |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMCDLH: 0b00000000 */
    PMCDLH = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

7. 参考ドキュメント

V850ES/Jx3-L ユーザーズマニュアル ハードウエア編（R01UH0018）

V850ES/JF3-L ユーザーズマニュアル ハードウエア編（R01UH0017）

V850ES/JG3-L ユーザーズマニュアル ハードウエア編（R01UH0165）

V850ES/JG3-L（USB コントローラ内蔵製品）ユーザーズマニュアル ハードウエア編（R01UH0001）

（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート/テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/inquiry

改訂記録	V850ES/Jx3-L シリーズ DMA 転送を使用した CSI 連続送受信（マスタ）

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.09.29	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. **未使用端子の処理**

   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

   CMOS**製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、**LSI**周辺のノイズが印加され、**LSI**内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。**

2. **電源投入時の処置**

   【注意】電源投入時は，製品の状態は不定です。

   電源投入時には、LSI**の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。**

   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. **リザーブアドレスのアクセス禁止**

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. **クロックについて**

   【注意】リセット時は、クロックが安定した後、リセットを解除してください。

   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. **製品間の相違について**

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0