V850 マイクロコントローラ

V850ES/Jx3-L

UARTA を使用した UART 通信

R01AN0780JJ0100
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 シリーズのアシンクロナス・シリアル・インタフェース A（UARTA）を使用して対向機器との UART 通信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、対向機器から送られてくる ASCII 文字を解析し、応答処理を行います。

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。

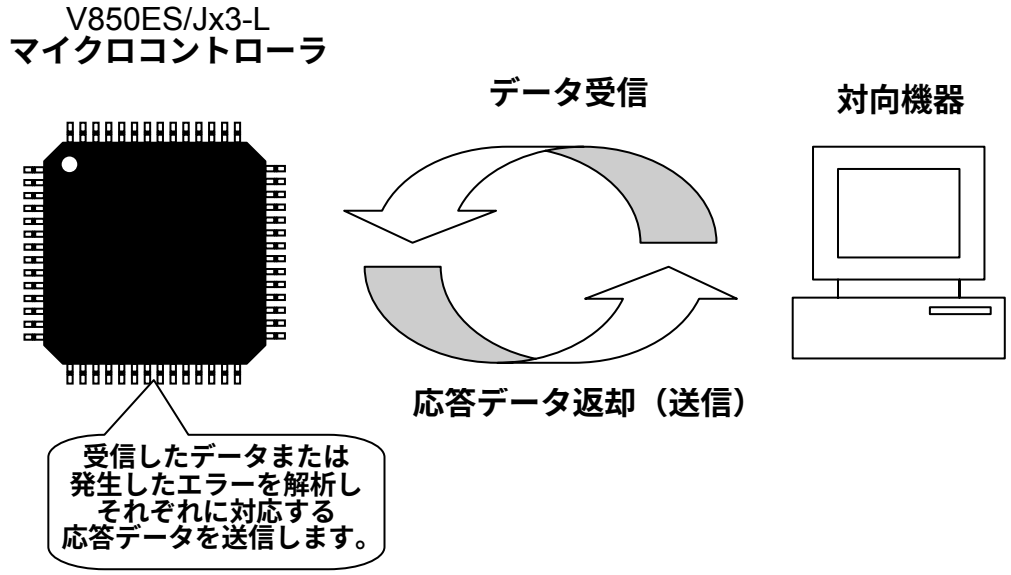
目次

1. 仕様

本アプリケーションノートでは、アシンクロナス・シリアル・インタフェース A（UARTA）の使用例を示しています。対向機器と UART 通信を行い、送られてくる ASCII 文字の解析と応答処理を行います。

サンプル・コードの動作概要を次に示します。

【動作概要】



対向機器との UART 通信では、データを正常に受信した場合、受信データを格納し、受信データに対応した値を送信します。送信するデータについては、表 1.1 に示します。

また、受信エラーが発生した場合、発生したエラーをチェックし、エラーに対応した値を送信します。送信するデータについては表 1.2 に示します。

表 1.1 受信データと返却する（送信）データ

受信データ	返却する（送信）データ
T（54H）	O（4FH）、 K（4BH）、 "CR"（0DH）、 "LF"（0AH）
t（74H）	o（6FH）、 k（6BH）、 "CR"（0DH）、 "LF"（0AH）
上記以外	U（55H）、 C（43H）、 "CR"（0DH）、 "LF"（0AH）

表 1.2 発生したエラーと返却する（送信）データ

受信データ	返却する（送信）データ
パリティ・エラー	P（50H）、E（45H）、"CR"（0DH）、"LF"（0AH）
フレーミング・エラー	F（46H）、E（45H）、"CR"（0DH）、"LF"（0AH）
オーバーラン・エラー	O（4FH）、E（45H）、"CR"（0DH）、"LF"（0AH）

表 1.3 に使用する周辺機能と用途を、表 1.4 に UART 通信の設定を示します。

表 1.3 使用する周辺機能と用途

周辺機能	用途
アシンクロナス・シリアル・インタフェース A（UARTA）	対向機器との UART 通信

表 1.4 UART 通信の設定

項目	設定
ボー・レート設定	38400bps
データ長設定	8 ビット
ストップ・ビット長設定	1 ビット
パリティ設定	偶数パリティ
転送方向設定	LSB ファースト

備考　本アプリケーションノートでは、アシンクロナス・シリアル・インタフェース A（UARTA）を使用して UART 通信を行いますが,この使用例はアシンクロナス・シリアル・インタフェース C（UARTC）にも応用することができます。

なお、V850ES/Jx3-L シリーズの UART 機能は、製品ごとに搭載している数が異なります。表 1.5 に UART 機能の違いを示します。

表 1.5　UART 機能の違い

愛称	品名	Flash/RAM	搭載している UART 機能
V850ES/JC3-L（40pin）	μPD70F3797 μPD70F3798 μPD70F3799 μPD70F3800 μPD70F3838	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	UARTA：2 チャネル
V850ES/JC3-L（48pin）	μPD70F3801 μPD70F3802 μPD70F3803 μPD70F3804 μPD70F3839	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	UARTA：3 チャネル
V850ES/JE3-L（64pin）	μPD70F3805 μPD70F3806 μPD70F3807 μPD70F3808 μPD70F3840	16KB/8KB 32KB/8KB 64KB/8KB 128KB/8KB 256KB/16KB	UARTA：3 チャネル
V850ES/JF3-L（80pin）	μPD70F3735 μPD70F3736	128KB/8KB 256KB/16KB	UARTA：3 チャネル
V850ES/JG3-L	μPD70F3737 μPD70F3738	128KB/8KB 256KB/16KB	UARTA：3 チャネル
	μPD70F3792 μPD70F3793 μPD70F3841 μPD70F3842	384KB/32KB 512KB/40KB 768KB/80KB 1MB/80KB	UARTA：6 チャネル UARTC：1 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	UARTA：6 チャネル UARTC：1 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）
使用ツール（対向機器）	ハイパーターミナル
使用ツールの設定	● ビット/秒 ：38400 ● データ ビット ：8 ● パリティ ：偶数 ● ストップ ビット ：1 ● フロー制御 ：なし

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（初期設定） LED 点灯のスイッチ制御編（R01ANxxxxJJ0100）

4. ハードウエア説明

4.1 使用端子一覧

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
P90/TXDA1	出力	UARTA1 送信出力
P91/RXDA1	入力	UARTA1 受信入力

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。

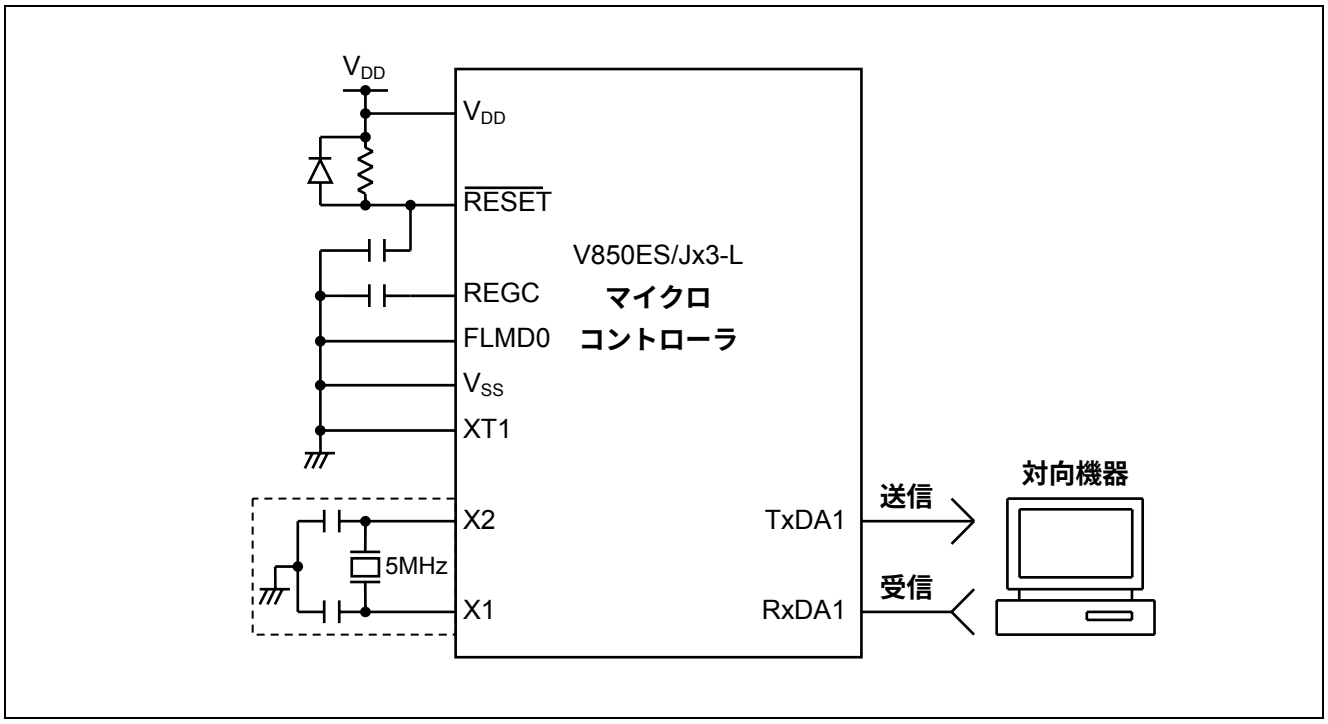


図 4.1 ハードウエア構成

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、アシンクロナス・シリアル・インタフェース A（UARTA）を使用し、対向機器との UART 通信を行います。対向機器からデータを受信すると、応答用のデータを対向機器に送信します。また、受信エラーが発生した場合は、エラーの種類に応じたデータを対向機器に送信します。

＜設定条件＞

- ボー・レートを 38400bps に設定します。
- データ長を 8 ビットに設定します。
- ストップビットを 1 ビットに設定します。
- 偶数パリティに設定します。
- 転送方向を LSB ファーストに設定します。
- 受信完了割り込み（INTUA1R）、送信許可割り込み（INTUA1T）を使用します。

(1) アシンクロナス・シリアル・インタフェース A（UARTA）の初期設定を行います。

(2) 受信完了割り込み（INTUA1R）を許可し、HALT モードに移行します。

(3) 対向機器からのデータ受信または受信エラー発生により受信完了割り込み（INTUA1R）が発生した場合、受信データまたはエラーステータスの解析を行い、それぞれに応じたデータを送信用に設定します。また、送信許可割り込み（INTUA1T）を許可し、対向機器へのデータ送信を開始します。

(4) データを 1 バイト送信するごとに送信許可割り込み（INTUA1T）が発生し、次のバイトの送信を開始します。すべての送信が完了したとき、送信許可割り込み（INTUA1T）を禁止し、HALT モードに移行します。

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
BUFFSIZE	16	受信データ用バッファのサイズ
TXDNUM	4	送信データのバイト数
CODE_NORMAL1	0x54	アスキーコードの‘T’
CODE_NORMAL2	0x74	アスキーコードの‘t’
CODE_ERRORP	0xFF	パリティエラーを示すコード
CODE_ERRORF	0xFE	フレーミングエラーを示すコード
CODE_ERRORO	0xFD	オーバーランエラーを示すコード
REP_NONE	0x00	未応答ステータス：未応答データなし
REP_STORE	0x01	未応答ステータス：未応答データあり
REP_FULL	0x02	未応答ステータス：未応答データフル

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
gStringOK	O K ¥r ¥n	‘T’受信時の送信データ
gStringok	o k ¥r ¥n	‘t’受信時の送信データ
gStringUC	U C ¥r ¥n	その他の文字コード受信時の送信データ
gStringPE	P E ¥r ¥n	パリティエラー発生時の送信データ
gStringFE	F E ¥r ¥n	フレーミングエラー発生時の送信データ
gStringOE	O E ¥r ¥n	オーバーランエラー発生時の送信データ

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*	gpUarta1TxAddress	送信データ用のポインタ	MD_INTUA1T MD_TxControl
USHORT	gUarta1TxCnt	送信したデータ数	MD_INTUA1T MD_TxControl
UCHAR*	gpUarta1RxAddress	受信バッファ用のポインタ	MD_INTUA1R
USHORT	gUarta1RxCnt	受信したデータ数	MD_INTUA1R MD_TxControl
USHORT	gUarta1RxLen	受信データ数の最大値	MD_INTUA1R
UCHAR	gRxBuffer	受信バッファ	MD_INTUA1R
UCHAR	gTxPointer	受信バッファ内の応答済みデータを示すポインタ	MD_INTUA1T MD_TxControl
UCHAR	gRepStatus	未応答ステータス	MD_INTUA1R MD_INTUA1T MD_TxControl

5.5 関数一覧

表 5.4 に 関数を示します。

表 5.4 関数

関数名	概要
MD_UartVarInit	通信動作に使用する変数の初期値設定
MD_TxControl	送信制御処理
MD_INTUA1R	INTUA1R 割り込み処理
MD_INTUA1T	INTUA1T 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_UartVarInit

概要	通信動作に使用する変数の初期値設定
ヘッダ	CG_serial.h
宣言	void MD_UartVarInit(void)
説明	通信動作に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_INTUA1R

概要	INTUA1R 割り込み処理
ヘッダ	CG_serial.h
宣言	__interrupt void MD_INTUA1R(void)
説明	受信したデータまたは発生したエラーをバッファに保存します。 ただし、未応答データでバッファが一杯になった場合、保存は行いません。
引数	なし
リターン値	なし
備考	なし

[関数名] MD_INTUA1T

概要	INTUA1T 割り込み処理
ヘッダ	CG_serial.h
宣言	__interrupt void MD_INTUA1T(void)
説明	データを送信します。
引数	なし
リターン値	なし
備考	なし

[関数名] MD_TxControl

概要	送信制御処理
ヘッダ	CG_serial.h
宣言	void MD_TxControl(void)
説明	受信データまたはエラーステータスに応じた送信データを設定します。 また、送信動作の開始および停止を実行します。
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、UARTA の設定などを行います。

初期設定完了後は、HALT モードに移行します。対向機器との UART 通信が開始されると、受信したデータまたはエラーを解析し、それぞれに対応した応答データを対向機器に送信します。応答データの送信が完了した場合、再び HALT モードに移行します。

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。

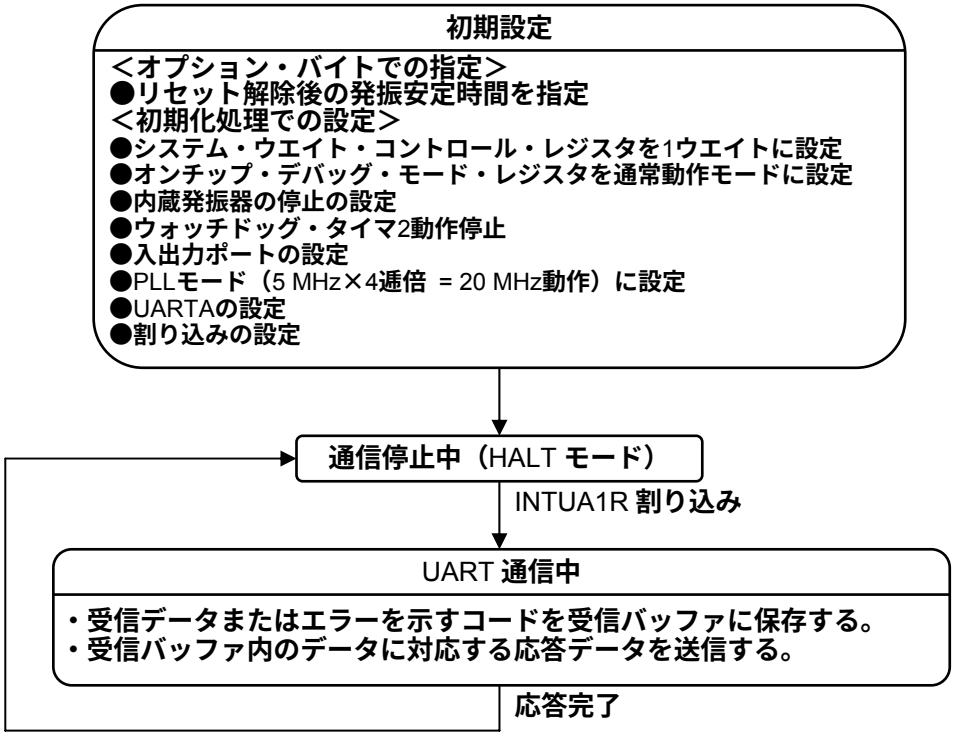


図 5.1 状態遷移図

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。

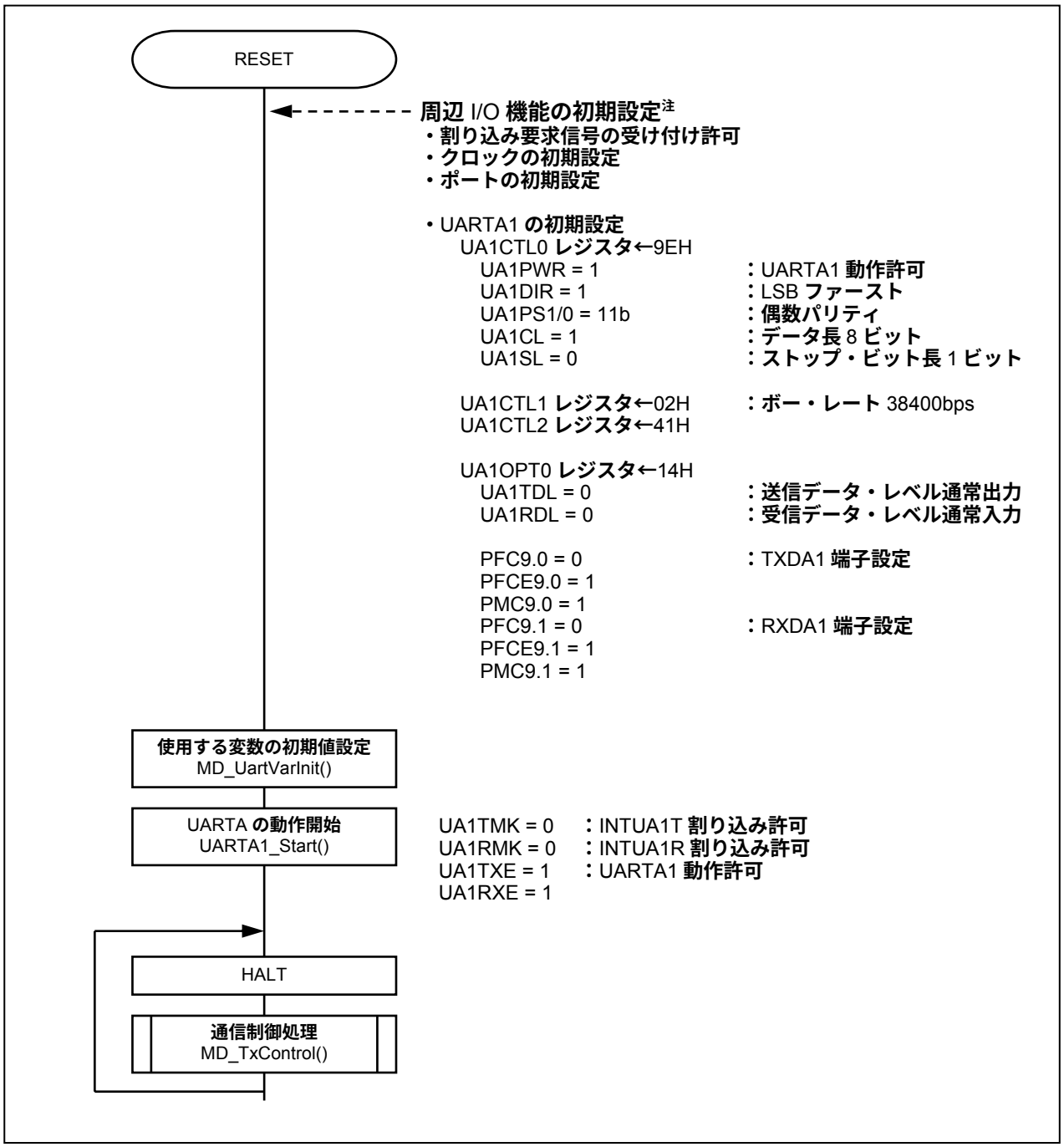


図 5.2 メイン処理

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。

5.8.2 割り込み処理

図 5.3.1 に INTUA1T 割り込み処理のフロー、図 5.3.2 に INTUA1R 割り込み処理のフローを示します。

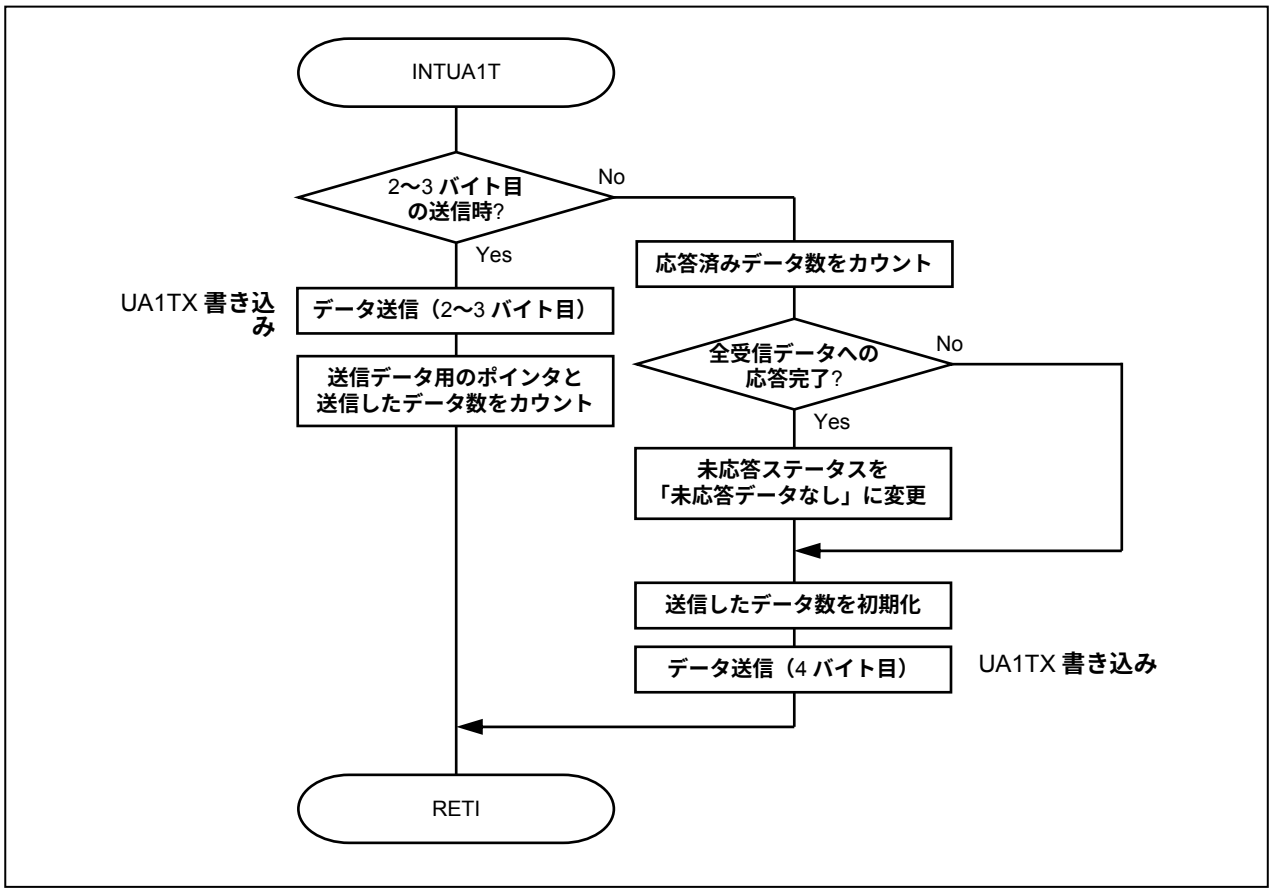


図 5.3.1 INTUA1T 割り込み処理

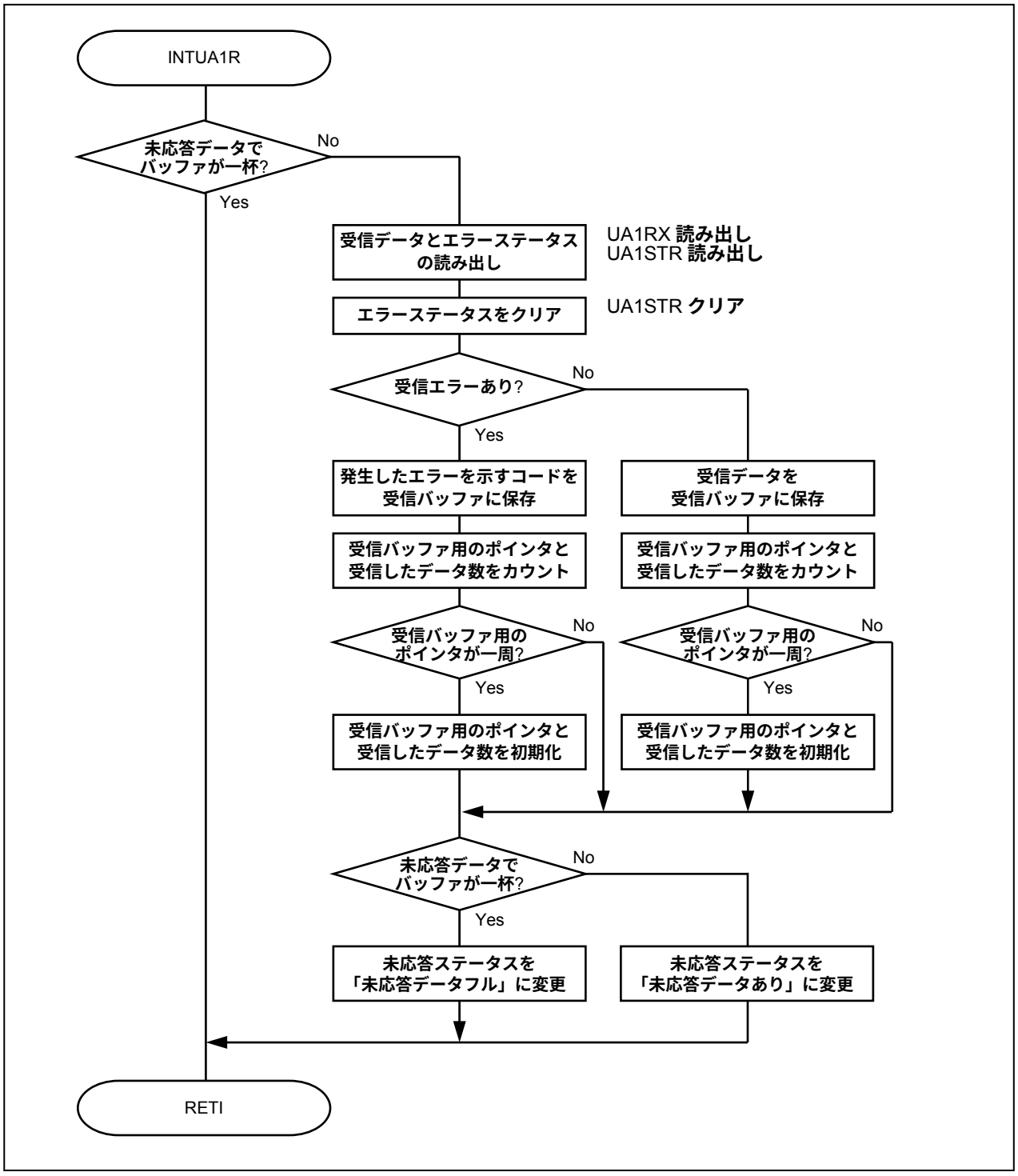


図 5.3.2 INTUA1R 割り込み処理

5.8.3 通信制御処理

図 5.4 に通信制御処理のフローを示します。

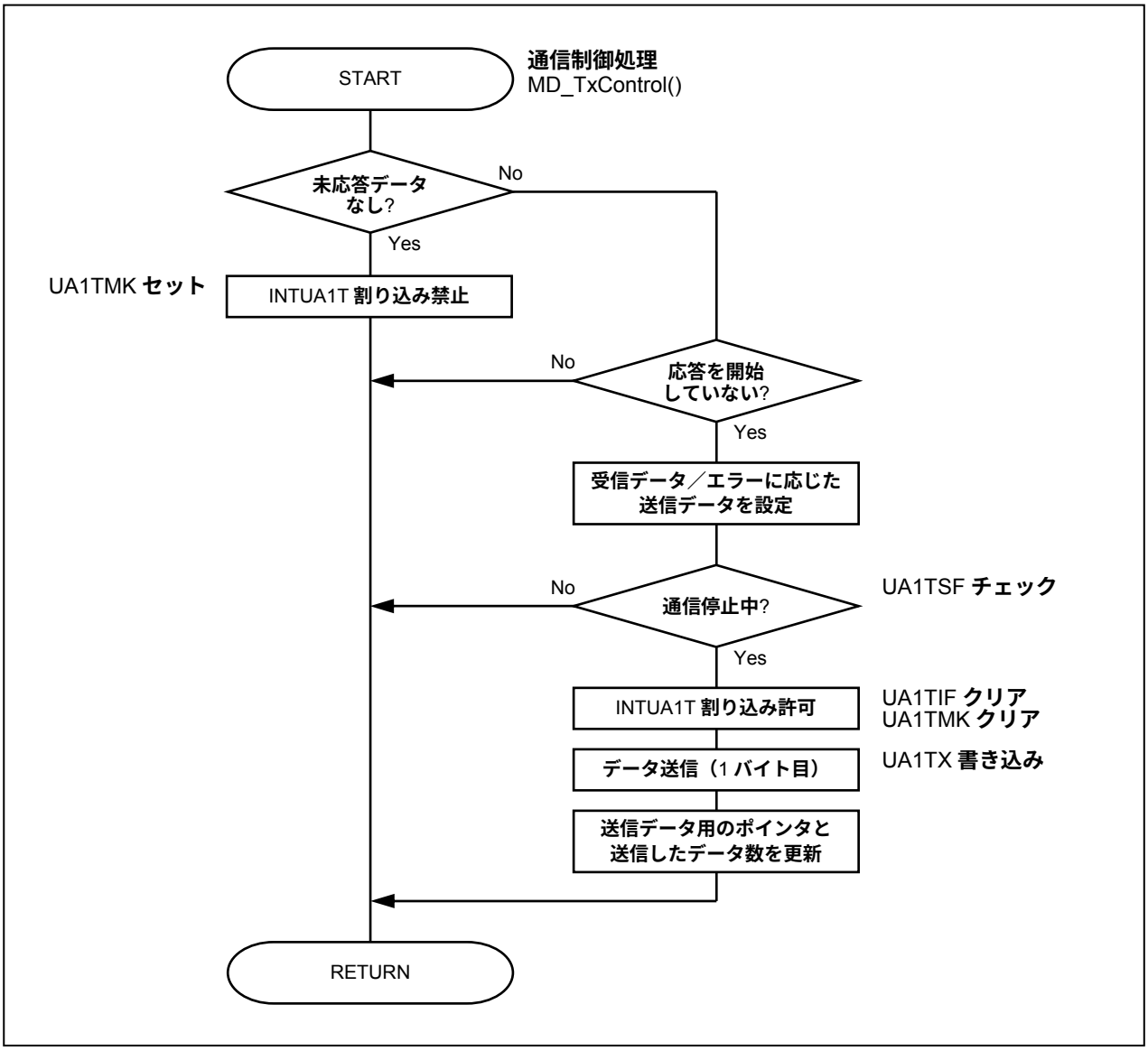
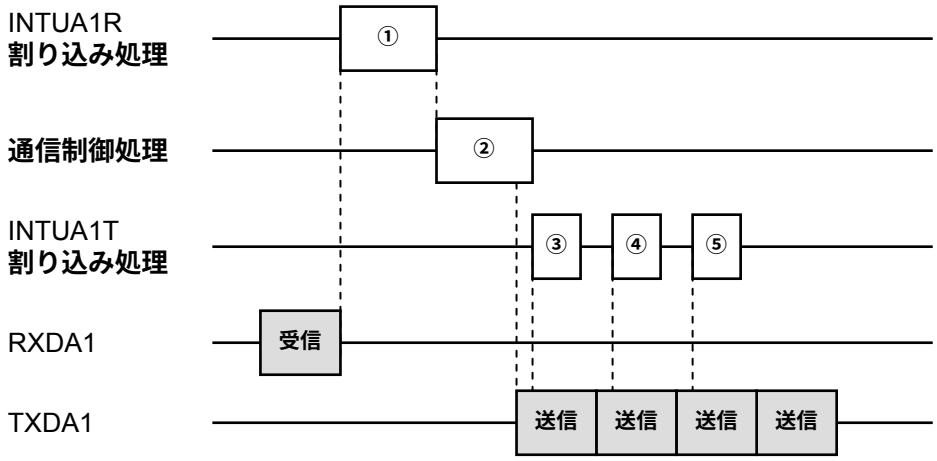


図 5.4 通信制御処理

また、図 5.5 にデータ受信から応答までのタイミングを示します。



① INTUA1R 割り込みにより処理開始。受信したデータを受信バッファに保存する。
② 受信バッファに保存されたデータを解析し、対応する応答データを選択する。また、INTUA1T 割り込みを許可し、応答データの 1 バイト目を UA1TX に書き込む。書き込み後は HALT モードに移行する。
③ INTUA1T 割り込みにより処理開始。応答データの 2 バイト目を UA1TX に書き込む。書き込み後は HALT モードに移行する。
④ INTUA1T 割り込みにより処理開始。応答データの 3 バイト目を UA1TX に書き込む。書き込み後は HALT モードに移行する。
⑤ INTUA1T 割り込みにより処理開始。応答データの 4 バイト目を UA1TX に書き込む。また、INTUA1T 割り込みを禁止し、HALT モードに移行する。

図 5.5 データ受信から応答までのタイミング

5.9 UARTA の設定

5.9.1 UARTA1 制御レジスタ 0（UA1CTL0）

UA1CTL0 レジスタは，UARTA1 シリアル転送動作を制御する 8 ビットのレジスタです。

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 10H になります。

UARTA1 制御レジスタ 0（UA1CTL0）
アドレス： FFFFFA10H

7	6	5	4	3	2	1	0
UA1PWR	UA1TXE	UA1RXE	UA1DIR	UA1PS1	UA1PS0	UA1CL	UA1SL

UA1PWR	UARTA1の動作の制御
0	UARTA1動作禁止（UARTA1を非同期にリセット）
1	**UARTA1動作許可**

UA1TXE	送信動作許可
0	送信動作禁止
1	**送信動作許可**

UA1RXE	受信動作許可
0	受信動作禁止
1	**受信動作許可**

UA1DIR	データ転送順序
0	MSBファースト
1	**LSBファースト**

UA1PS1	UA1PS0	送信時のパリティ選択	受信時のパリティ選択
0	0	パリティを出力しない	パリティなしで受信
0	1	0パリティを出力	0パリティとして受信
1	0	奇数パリティを出力	奇数パリティとして判定を行う
1	**1**	**偶数パリティを出力**	**偶数パリティとして判定を行う**

UA1CL	送受信データ1フレームのデータ・キャラクタ長指定
0	7ビット
1	**8ビット**

UA1SL	送信データのストップ・ビット長指定
0	**1ビット**
1	2ビット

図5.6.1 UARTA1制御レジスタ0（UA1CTL0）のフォーマット

5.9.2 UARTA1 制御レジスタ 1（UA1CTL1）

UA1CTL1 レジスタは，UARTA1 の基本クロックを選択するための 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。

注意 UA1CTL1レジスタを書き換える場合は，UA1CTL0.UA1PWRビット = 0にしてから行ってください。

UARTA1 制御レジスタ 1（UA1CTL1）
アドレス： FFFFFA11H

7	6	5	4	3	2	1	0
0	0	0	0	UA1CKS3	UA1CKS2	UA1CKS1	UA1CKS0

UA1CKS3	UA1CKS2	UA1CKS1	UA1CKS0	基本クロック（f_{UCLK}）の選択
0	0	0	0	f_{XX}
0	0	0	1	$f_{XX}/2$
0	**0**	**1**	**0**	$f_{XX}/4$
0	0	1	1	$f_{XX}/8$
0	1	0	0	$f_{XX}/16$
0	1	0	1	$f_{XX}/32$
0	1	1	0	$f_{XX}/64$
0	1	1	1	$f_{XX}/128$
1	0	0	0	$f_{XX}/256$
1	0	0	1	$f_{XX}/512$
1	0	1	0	$f_{XX}/1024$
1	0	1	1	外部クロック[注]（ASCKA0端子）
上記以外				設定禁止

注．UARTA0 のみ有効。

備考．f_{XX}：メイン・クロック周波数

図5.6.2 UARTA1制御レジスタ1（UA1CTL1）のフォーマット

5.9.3 UARTA1 制御レジスタ 2（UA1CTL2）

UA1CTL2 レジスタは，UARTA1 のボー・レート（シリアル転送スピード）クロックを選択するための 8 ビットのレジスタです。UAnCTL2 レジスタで設定したクロックを 2 分周したクロックがボー・レート・クロックになります。

8 ビット単位でリード／ライト可能です。

リセットにより FFH になります。

UARTA1 制御レジスタ 2（UA1CTL2）
アドレス： FFFFFA12H

7	6	5	4	3	2	1	0
UA1BRS7	UA1BRS6	UA1BRS5	UA1BRS4	UA1BRS3	UA1BRS2	UA1BRS1	UA1BRS0

UA1 BRS7	UA1 BRS6	UA1 BRS5	UA1 BRS4	UA1 BRS3	UA1 BRS2	UA1 BRS1	UA1 BRS0	規定値 (k)	シリアル・クロック
0	0	0	0	0	0	X	X	X	設定禁止
0	0	0	0	0	1	0	0	4	$f_{UCLK}/4$
0	0	0	0	0	1	0	1	5	$f_{UCLK}/5$
0	0	0	0	0	1	1	0	6	$f_{UCLK}/6$
:	:	:	:	:	:	:	:	:	:
0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	65	$f_{UCLK}/65$
:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	0	0	252	$f_{UCLK}/252$
1	1	1	1	1	1	0	1	253	$f_{UCLK}/253$
1	1	1	1	1	1	1	0	254	$f_{UCLK}/254$
1	1	1	1	1	1	1	1	255	$f_{UCLK}/255$

f_{UCLK}：UA1CTL1.UA1CKS3-UA1CKS0 ビットで選択したクロック周波数

図5.6.3 UARTA1制御レジスタ2（UA1CTL2）のフォーマット

次に、代表的なボー・レートの設定値を示します。

表5.5.1 ボー・レート・ジェネレータ設定データ

ボー・レート (bps)	f_{XX} = 20MHz			f_{XX} = 16MHz			f_{XX} = 10MHz		
	UA1CTL1	UA1CTL2	ERR（%）	UA1CTL1	UA1CTL2	ERR（%）	UA1CTL1	UA1CTL2	ERR（%）
300	08H	82H	0.16	07H	D0H	0.16	07H	82H	0.16
600	07H	82H	0.16	06H	D0H	0.16	06H	82H	0.16
1200	06H	82H	0.16	05H	D0H	0.16	05H	82H	0.16
2400	05H	82H	0.16	04H	D0H	0.16	04H	82H	0.16
4800	04H	82H	0.16	03H	D0H	0.16	03H	82H	0.16
9600	03H	82H	0.16	02H	D0H	0.16	02H	82H	0.16
19200	02H	82H	0.16	01H	D0H	0.16	01H	82H	0.16
31250	01H	A0H	0	01H	80H	0	00H	A0H	0
38400	01H	82H	0.16	00H	D0H	0.16	00H	82H	0.16
76800	00H	82H	0.16	00H	68H	0.16	00H	41H	0.16
153600	00H	41H	0.16	00H	34H	0.16	00H	21H	-1.36
312500	00H	20H	0	00H	1AH	-1.54	00H	10H	0
625000	00H	10H	0	00H	0DH	-1.54	00H	08H	0

備考 f_{XX}：メイン・クロック周波数
ERR：ボー・レート誤差 [%]

5.9.4 UARTA1 状態レジスタ（UA1STR）

UA1STR レジスタは，UARTA1 の転送状態と受信エラー内容を示す 8 ビットのレジスタです。

8/1 ビット単位でリード／ライト可能です。UA1TSF ビットはリードのみ可能で，UA1PE, UA1FE, UA1OVE ビットについてはリード／ライト可能ですが，ライト時は"0"ライトによるクリアのみ可能で，"1"ライトによるセット動作はできません（"1"をライトしても値を保持します）。

次にクリア条件を示します。

表5.5.2 STRレジスタのクリア条件

レジスタ／ビット	クリア条件
UA1STRレジスタ	・リセット ・UA1CTL0.UA1PWRビット = 0
UA1TSFビット	・UA1CTL0.UA1TXEビット = 0
UA1PE, UA1FE, UA1OVEビット	・0の書き込み ・UA1CTL0.UA1RXEビット = 0

UARTA1 状態レジスタ（UA1STR）
アドレス： FFFFFA14H

7	6	5	4	3	2	1	0
UA1TSF	0	0	0	0	UA1PE	UA1FE	UA1OVE

UA1TSF	転送状態フラグ
0	送信シフト・レジスタにデータなし.. ・UA1PWRビット = 0，またはUA1TXEビット = 0に設定したとき ・転送完了後に，UA1TXレジスタに次のデータ転送がなかったとき
1	送信シフト・レジスタにデータあり （UA1TXレジスタへの書き込み）

UA1PE	パリティ・エラー・フラグ
0	・UA1PWRビット = 0，またはUA1RXEビット = 0に設定したとき ・“0”をライトしたとき
1	受信したパリティ・ビットが設定と一致しないとき

UA1FE	フレーミング・エラー・フラグ
0	・UA1PWRビット = 0，またはUA1RXEビット = 0に設定したとき ・“0”をライトしたとき
1	受信時，ストップ・ビットが検出されないとき

UA1OVE	オーバラン・エラー・フラグ
0	・UA1PWRビット = 0，またはUA1RXEビット = 0に設定したとき ・“0”をライトしたとき
1	UA1RXレジスタに受信データがセットされ，それを読み出す前に次の受信動作が完了したとき

図5.6.4 UARTA1状態レジスタ（UA1STR）のフォーマット

5.9.5 UARTA1 受信データ・レジスタ（UA1RX）

UA1RX レジスタは，受信シフト・レジスタで変換したパラレル・データを格納するための 8 ビット・バッファ・レジスタです。

1 キャラクタの受信完了により受信シフト・レジスタに格納したデータを UA1RX レジスタに転送します。

8 ビット単位でリードのみ可能です。

リセット以外に，UA1CTL0.UA1PWR ビット = 0 によっても UA1RX レジスタは FFH になります。

UARTA1 受信データ・レジスタ（UA1RX）
アドレス： FFFFFA16H

	7	6	5	4	3	2	1	0
UA1RX								

図5.6.5 UARTA1受信データ・レジスタ（UA1RX）のフォーマット

5.9.6 UARTA1 送信データ・レジスタ（UA1TX）

UA1TX レジスタは，送信データを設定するための 8 ビット・レジスタです。

送信許可状態（UA1CTL0.UA1TXE ビット = 1）のときに，UA1TX レジスタへ送信データを書き込むことにより，送信動作が開始されます。UA1TX レジスタのデータを UARTA1 送信シフト・レジスタに転送終了したタイミングで，送信許可割り込み要求信号（INTUA1T）を発生します。

8 ビット単位でリード／ライト可能です。

リセットにより FFH になります。

注意 送信動作許可状態（UA1PWRビット = 1かつUA1TXEビット = 1）では，UA1TXレジスタへの書き込みは，送信トリガとして作用されるため，直前の値と同一の値を書き込むと，二度同じデータが送信されてしまいます。送信中の送信データの書き込みは，必ず送信許可割り込み要求信号（INTUA1T）が発生したあとにしてください。また，送信禁止状態（UA1PWRビット = 0またはUA1TXEビット = 0）にてUA1TXレジスタに書き込み後，送信許可状態に設定しても送信は開始されません。

UARTA1 受信データ・レジスタ（UA1TX）
アドレス： FFFFFA17H

	7	6	5	4	3	2	1	0
UA1TX								

図5.6.6 UARTA1送信データ・レジスタ（UA1TX）のフォーマット

5.9.7 UARTA1 の端子設定

UARTA1 を使用する場合，UARTA1 の送信出力用端子（TXDA1）と UARTA1 の受信入力用端子（RXDA1）を設定する必要があります。

TXDA1 は、ポート 9 ファンクション・コントロール・レジスタ（PFC9）のビット 0、ポート 9 ファンクション・コントロール拡張レジスタ（PFCE9）のビット 0、ポート 9 モード・コントロール・レジスタ（PMC9）のビット 0 により設定します。また、RXDA1 は、ポート 9 ファンクション・コントロール・レジスタ（PFC9）のビット 1、ポート 9 ファンクション・コントロール拡張レジスタ（PFCE9）のビット 1、ポート 9 モード・コントロール・レジスタ（PMC9）のビット 1 により設定します。

表5.5.7 TXDA1の設定

PMC9.0	PFC9.0	PFCE9.0	端子機能の指定
0	x	x	入出力ポート（P90）
1	**1**	**0**	TXDA1出力
1	1	1	SDA02入出力

備考 x：no care

表5.5.8 RXDA1の設定

PMC9.1	PFC9.1	PFCE9.1	端子機能の指定
0	x	x	入出力ポート（P91）
1	**1**	**0**	RXDA1入力
1	1	1	SCL02入出力

備考 x：no care

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: 2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** 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"
/* 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_UartVarInit();       /* Variable initialization for UART */
    UARTA1_Start();         /* Start UARTA1 operation */

    /* Main loop */
    while (1U)
```

```
        {
                HALT();/* Cpu standby */

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

                MD_TxControl();          /* UART transmission control */
        }

        /* 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。

• UARTA1_Stop

→UARTA1 の動作を停止します。

• MD_STATUS UARTA1_ReceiveData

→受信データをバッファに保存します。

• MD_STATUS UARTA1_SendData

→データ送信を行います。

• UARTA1_SoftOverRunCallback

→受信データ数がバッファサイズを超えた場合の処理です。

```
/*
************************************************************************
* 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: 2011/07/20
************************************************************************
*/

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

```
#pragma interrupt INTUA1R MD_INTUA1R
#pragma interrupt INTUA1T MD_INTUA1T
/* 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   *gpUarta1TxAddress;  /* uarta1 transmit buffer address */
volatile USHORT gUarta1TxCnt;/* uarta1 transmit data number */
volatile UCHAR   *gpUarta1RxAddress; /* uarta1 receive buffer address */
volatile USHORT gUarta1RxCnt;         /* uarta1 receive data number */
volatile USHORT gUarta1RxLen;         /* uarta1 receive data length */
/* Start user code for global. Do not edit comment generated here */
/*
**********************************************************************************
** Private global variables and constants
**********************************************************************************
*/
        #define BUFFSIZE       16      /*   Receive data buffer size */
static UCHAR gRxBuffer[BUFFSIZE];    /* Receive data buffer */
static UCHAR gTxPointer;             /* Buffer pointer for transmission */
        #define TXDNUM                 4       /*   Transmit data number */
static UCHAR gRepStatus;             /* Reply status */
        #define REP_NONE     0x00    /*   All reply finished */
        #define REP_STORE    0x01    /*   Some no-reply data stored */
        #define REP_FULL     0x02    /*   Buffer is full of no-reply data */
```

```
/* Transmit code */
static const UCHAR gStringOK[TXDNUM] = { "OK¥r¥n" };
static const UCHAR gStringok[TXDNUM] = { "ok¥r¥n" };
static const UCHAR gStringUC[TXDNUM] = { "UC¥r¥n" };
static const UCHAR gStringPE[TXDNUM] = { "PE¥r¥n" };
static const UCHAR gStringFE[TXDNUM] = { "FE¥r¥n" };
static const UCHAR gStringOE[TXDNUM] = { "OE¥r¥n" };

/* Receive code */
#define CODE_NORMAL1        'T'     /* Normal code 1 : 'T' */
#define CODE_NORMAL2        't'     /* Normal code 2 : 't' */

/* Error code */
#define CODE_ERRORP         0xFF    /* Parity error code */
#define CODE_ERRORF         0xFE    /* Framing error code */
#define CODE_ERRORO         0xFD    /* Overrun error code */
/* End user code. Do not edit comment generated here */

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function initializes the UARTA1 module.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void UARTA1_Init(void)
{
        UA1TXE = 0U;  /* disable UARTA1 transmission operation */
        UA1RXE = 0U;  /* disable UARTA1 reception operation */
        UA1PWR = 0U; /* disable UARTA1 operation */
        UA1TMK = 1U;  /* disable INTUA1T interrupt */
        UA1TIF = 0U;    /* clear INTUA1T interrupt flag */
```

```
    UA1RMK = 1U; /* disable INTUA1R interrupt */
    UA1RIF = 0U;    /* clear INTUA1R interrupt flag */
    /* Set INTUA1T level low priority */
    UA1TIC |= 0x07U;
    /* Set INTUA1R level low priority */
    UA1RIC |= 0x07U;
    /* Baud rate : 38400bps */
    UA1CTL1 = _02_UARTA_BASECLK_FXX_4;
    UA1CTL2 = _41_UARTA1_BASECLK_DIVISION;
    /* LSB first, Even parity, Data length 8bit, Stop bit 1bit */
    UA1CTL0 = _10_UARTA_TRANSFDIR_LSB | _0C_UARTA_PARITY_EVEN |
_02_UARTA_DATALENGTH_8BIT | _00_UARTA_STOPLENGTH_1BIT;
    /* Normal output, Normal input */
    UA1OPT0 = _14_UARTA_UAnOPT0_INITIALVALUE |
_00_UARTA_TRAN_DATALEVEL_NORMAL | _00_UARTA_REC_DATALEVEL_NORMAL;
    UA1PWR = 1U; /* enable UARTA1 operation */
    /* Set TXDA1 pin */
    PFC9L &= 0xFEU;
    PFCE9L |= 0x01U;
    PMC9L |= 0x01U;
    /* Set RXDA1 pin */
    PFC9L &= 0xFDU;
    PFCE9L |= 0x02U;
    PMC9L |= 0x02U;
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function starts the UARTA1 operation.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void UARTA1_Start(void)
```

```
{
        UA1TIF = 0U;    /* clear INTUA1T interrupt flag */
        UA1TMK = 0U;  /* enable INTUA1T interrupt */
        UA1RIF = 0U;    /* clear INTUA1R interrupt flag */
        UA1RMK = 0U; /* enable INTUA1R interrupt */
        UA1TXE = 1U;  /* enable UARTA1 transmission operation */
        UA1RXE = 1U;  /* enable UARTA1 reception operation */
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function stops the UARTA1 operation.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
/*void UARTA1_Stop(void)
{
        UA1TXE = 0U;*/          /* disable UARTA1 transmission operation */
/*      UA1RXE = 0U;*/          /* disable UARTA1 reception operation */
/*      UA1TMK = 1U;*/          /* disable INTUA1T interrupt */
/*      UA1TIF = 0U;*/ /* clear INTUA1T interrupt flag */
/*      UA1RMK = 1U;*/          /* disable INTUA1R interrupt */
/*      UA1RIF = 0U;*/ /* clear INTUA1R interrupt flag */
/*}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function receives UARTA1 data.
**
**  Parameters:
**      rxbuf: receive buffer pointer
```

```
**      rxnum: buffer size
**
**  Returns:
**      MD_OK
**      MD_ARGERROR
**
**-------------------------------------------------------------------------
*/
/*MD_STATUS UARTA1_ReceiveData(UCHAR *rxbuf, USHORT rxnum)
{
        MD_STATUS status = MD_OK;

        if (rxnum < 1U)
        {
                status = MD_ARGERROR;
        }
        else
        {
                gUarta1RxCnt = 0U;
                gUarta1RxLen = rxnum;
                gpUarta1RxAddress = rxbuf;
        }

        return (status);
}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function sends UARTA1 data.
**
**  Parameters:
**      txbuf: transfer buffer pointer
**      txnum: buffer size
**
**  Returns:
**      MD_OK
**      MD_ARGERROR
**      MD_DATAEXISTS
```

```
**
**------------------------------------------------------------------------
*/
/*MD_STATUS UARTA1_SendData(UCHAR *txbuf, USHORT txnum)
{
        MD_STATUS status = MD_OK;

        if (txnum < 1U)
        {
                status = MD_ARGERROR;
        }
        else
        {
                gpUarta1TxAddress = txbuf;
                gUarta1TxCnt = txnum;
                if((UA1STR & 0x80U) == 0U)
                {
                        UA1TMK = 1U;*/          /* disable INTUA1T interrupt */
/*                      UA1TX = *gpUarta1TxAddress;
                        gpUarta1TxAddress++;
                        gUarta1TxCnt--;
                        UA1TMK = 0U;*/          /* enable INTUA1T interrupt */
/*              }
                else
                {
                        status = MD_DATAEXISTS;
                }
        }

        return (status);
}*/

/* Start user code for adding. Do not edit comment generated here */
/*
**------------------------------------------------------------------------
**
**  Abstract:
**      This function is INTUA1R interrupt service routine.
**
```

```
**    Parameters:
**          None
**
**    Returns:
**          None
**
**------------------------------------------------------------------------
*/
__interrupt void MD_INTUA1R(void)
{
        UCHARrx_data;
        UCHARerr_type;

        /* Receive data buffer is mask */
        if( gRepStatus == REP_FULL )
        {
                NOP();  /* Don't read/store receive data */
        }
        else
        {
                rx_data = UA1RX;                        /* Read receive data */
                err_type = (UCHAR)(UA1STR & 0x07U);         /* Read reception error */
                UA1STR = (UCHAR)~0x07U;             /* Clear reception error */

                /* Error is detected */
                if (err_type != 0U)
                {
                        UARTA1_ErrorCallback(err_type);         /* Error function */
                }
                /* Reception ends normally */
                else
                {
                        /* Receive data buffer is not full */
                        if (gUarta1RxLen > gUarta1RxCnt)
                        {
                                *gpUarta1RxAddress = rx_data; /* Store receive data */
                                gpUarta1RxAddress++;            /* Increment receive buffer address */
                                gUarta1RxCnt++;                 /* Count receive data number */
```

```
                                /* Receive data buffer is end */
                                if (gUarta1RxLen == gUarta1RxCnt)
                                {
                                        UARTA1_ReceiveEndCallback();          /* Reset receive data
number */
                                }
                        }
                        /* Receive data number is over */
                        else
                        {
                                /*UARTA1_SoftOverRunCallback(rx_data);*/      /* Overrun function */
                        }
                }

                /* Receive data buffer is full of no-reply data */
                if( gTxPointer == gUarta1RxCnt )
                {
                        gRepStatus = REP_FULL;        /* Set reply status */
                }
                /* Some data stored */
                else
                {
                        gRepStatus = REP_STORE;     /* Set reply status */
                }
        }
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function is INTUA1T interrupt service routine.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
```

```
*/
__interrupt void MD_INTUA1T(void)
{
        /* During transmission */
        if (gUarta1TxCnt > 0U)
        {
                UA1TX = *gpUarta1TxAddress;  /* Output transmit data */
                gpUarta1TxAddress++;         /* Increment transmit data address */
                gUarta1TxCnt--;              /* Count data transfer */
        }
        /* Last data transfer */
        else
        {
                UARTA1_SendEndCallback();    /* Last data transfer function */
        }
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      This function is a callback function of UARTA1.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void UARTA1_ReceiveEndCallback(void)
{
        /* Start user code. Do not edit comment generated here */

        gUarta1RxCnt = 0;                               /* Clear receive data number */
        gpUarta1RxAddress = (UCHAR *)gRxBuffer;         /* Reset receive data buffer address */

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

```
/*
**-------------------------------------------------------------------------
**
**    Abstract:
**        This function is a callback function of UARTA1.
**
**    Parameters:
**        None
**
**    Returns:
**        None
**
**-------------------------------------------------------------------------
*/
void UARTA1_SendEndCallback(void)
{
        /* Start user code. Do not edit comment generated here */

        /* Pointer update */
        gTxPointer++;
        gTxPointer %= BUFFSIZE;

        /* All reply finished */
        if( gTxPointer == gUarta1RxCnt )
        {
                gRepStatus = REP_NONE;        /* Set reply status */
        }

        gUarta1TxCnt = (TXDNUM - 1); /* Clear transmit number */
        UA1TX = *gpUarta1TxAddress;  /* Output transmit data */

        /* End user code. Do not edit comment generated here */
}
/*
**-------------------------------------------------------------------------
**
**    Abstract:
**        This function is a callback function of UARTA1.
**
```

```
**    Parameters:
**          err_type: error type
**
**    Returns:
**          None
**
**------------------------------------------------------------------------
*/
void UARTA1_ErrorCallback(UCHAR err_type)
{
        /* Start user code. Do not edit comment generated here */

        /* Parity error */
        if( err_type & 0b00000100 )
        {
                /* Store parity error code for buffer */
                MD_StoreError( CODE_ERRORP );
        }

        /* Framing error */
        if( err_type & 0b00000010 )
        {
                /* Store framing error code for buffer */
                MD_StoreError( CODE_ERRORF );
        }

        /* Overrun error */
        if( err_type & 0b00000001 )
        {
                /* Store overrun error code for buffer */
                MD_StoreError( CODE_ERRORO );
        }

        /* End user code. Do not edit comment generated here */
}
/*
**------------------------------------------------------------------------
**
**    Abstract:
```

```
**      Store error code for buffer.
**
**  Parameters:
**      Error code
**
**  Returns:
**      None
**
**------------------------------------------------------------------------
*/
void MD_StoreError(UCHAR code)
{

    *gpUarta1RxAddress = code;    /* Set error code */
    gpUarta1RxAddress++;          /* Increment receive buffer address */
    gUarta1RxCnt++;               /* Count receive data number */

    /* Receive data buffer is full */
    if (gUarta1RxLen <= gUarta1RxCnt)
    {
        gUarta1RxCnt = 0;         /* Clear receive data number */

        /* Reset receive data buffer address */
        gpUarta1RxAddress = (UCHAR *)gRxBuffer;
    }

}
/*
**------------------------------------------------------------------------
**
**  Abstract:
**      Variable initialization for UART.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
```

```
**-------------------------------------------------------------------------
*/
void MD_UartVarInit( void )
{

        gUarta1TxCnt = (TXDNUM - 1);                /* Clear transmit number */
        gpUarta1RxAddress = (UCHAR *)gRxBuffer;     /* Set receive data buffer address */
        gUarta1RxLen = BUFFSIZE;                    /* Set maximum value of receive data number */

        gUarta1RxCnt = 0;               /* Clear receive data number */
        gTxPointer = 0;             /* Clear buffer pointer for transmission */
        gRepStatus = REP_NONE;      /* Reply status : All reply finished */

}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**      UART transmission control.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------
*/
void MD_TxControl( void )
{

        /* All reply finished */
        if( gRepStatus == REP_NONE )
        {
                UA1TMK = 1;    /* Disable INTUA1T */
        }
        /* No-reply data stored */
        else
```

```
{
        /* First byte of transmit data */
        if( gUarta1TxCnt >= (TXDNUM - 1) )
        {
                /* Set transmit data address */
                switch( gRxBuffer[ gTxPointer ] )
                {
                        /* Normal code 1 receive */
                        case CODE_NORMAL1:
                                /* Set transmit data 'OK' */
                                gpUarta1TxAddress = (UCHAR *)gStringOK;
                                break;
                        /* Normal code 2 receive */
                        case CODE_NORMAL2:
                                /* Set transmit data 'ok' */
                                gpUarta1TxAddress = (UCHAR *)gStringok;
                                break;
                        /* Parity error */
                        case CODE_ERRORP:
                                /* Set transmit data 'PE' */
                                gpUarta1TxAddress = (UCHAR *)gStringPE;
                                break;
                        /* Framing error */
                        case CODE_ERRORF:
                                /* Set transmit data 'FE' */
                                gpUarta1TxAddress = (UCHAR *)gStringFE;
                                break;
                        /* Overrun error */
                        case CODE_ERRORO:
                                /* Set transmit data 'OE' */
                                gpUarta1TxAddress = (UCHAR *)gStringOE;
                                break;
                        /* Other code receive */
                        default:
                                /* Set transmit data 'UC' */
                                gpUarta1TxAddress = (UCHAR *)gStringUC;
                                break;
                }
```

```
                        /* Transmission stopped */
                        if( !UA1TSF )
                        {
                                UA1TIF = 0;
                                UA1TMK = 0;    /* Enable INTUA1T */

                                gUarta1TxCnt--;            /* Count transmit number */
                                UA1TX = *gpUarta1TxAddress;  /* Data output */
                                gpUarta1TxAddress++;             /* Increment transmit data address */
                        }
                }
        }

}
/* 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: 2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** 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"
```

```
/* 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();
        UARTA1_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: 2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** 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: 2011/07/26
***********************************************************************************
*/

/*
***********************************************************************************
** 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)
{
    /* P90 : TXDA1 */
    /* P91 : RXDA1 */
    /* 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   : 0b11000000 */
        PM5 = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT
| _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_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 : 0b00000011 */
        PM9L = _01_PMn0_MODE_UNUSED | _02_PMn1_MODE_UNUSED |
_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 |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT;
        /* PMC9L : 0b00000000 */
        PMC9L = _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 アシンクロナス・シリアル・インタフェース A（UARTA）

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