お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

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

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

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

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

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

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

H8/300H Tiny シリーズ
モニタソフト

要旨

パソコンと H8/3664 を RS-232C ドライバを使用して接続し、パソコン上のターミナルウィンドウ（ハイパーターミナル）からメモリのダンプおよびメモリ（RAM）のエディットを行います。

動作確認デバイス

H8/3664

目次

1. 仕様

1. 図 1.1 に示すようにパソコンと H8/3664 を RS-232C ドライバを使用して接続し、パソコン上のターミナルウィンドウ（ハイパーターミナル）からメモリのダンプおよびメモリ（RAM）のエディットを行います。
2. 送信データの通信フォーマットは、データ長を 8 ビット、パリティビットなし、ストップビット長を 1 ビット、ビットレートは 9600（bit/s）に設定し送受信を行います。

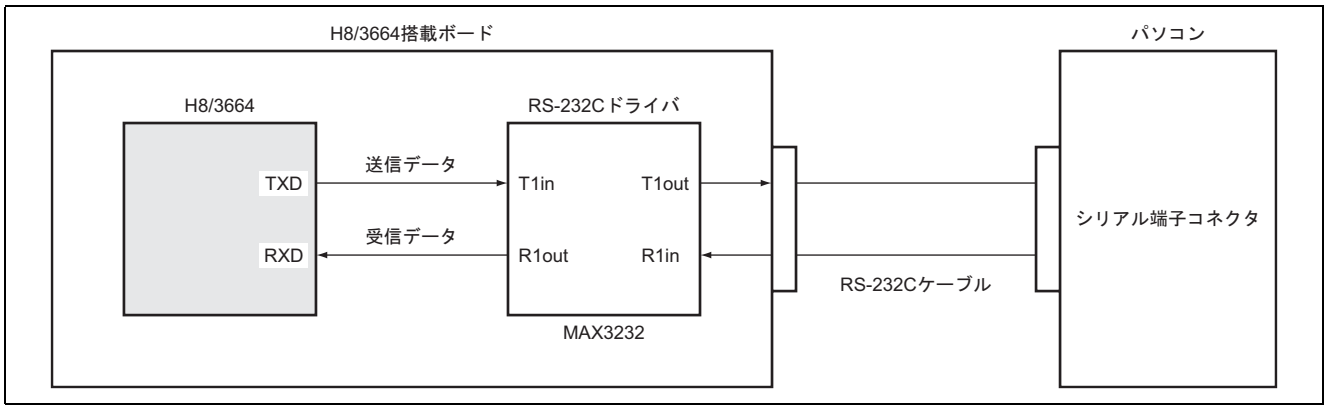
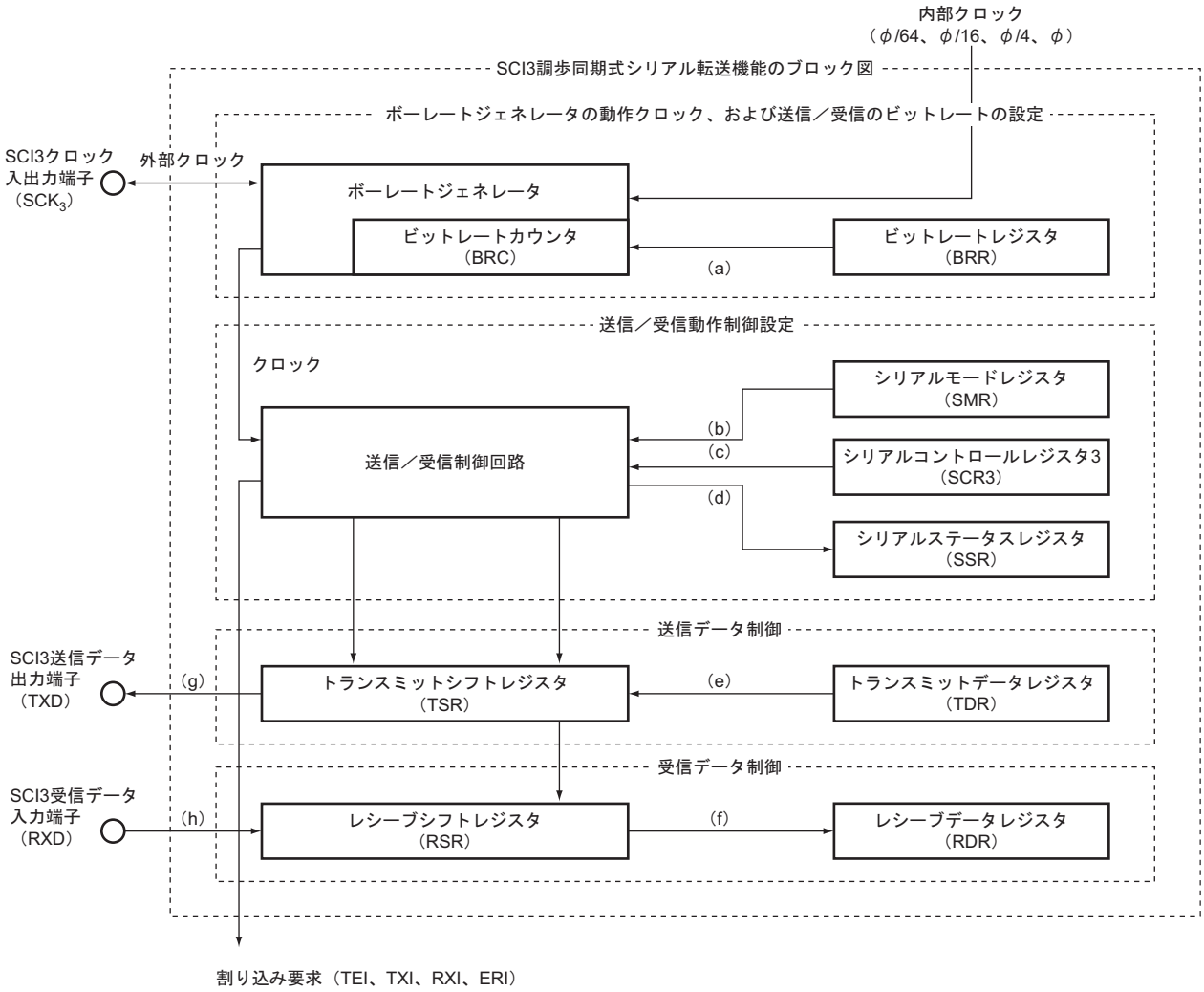


図 1.1　H8/3664 搭載ボードーパソコン接続図

2. 使用機能説明

1. 本タスク例では、シリアルコミュニケーションインタフェース（SCI：Serial Communication Interface）を使用して、調歩同期式のシリアルデータの同時送受信を行います。図 2.1 に調歩同期式シリアルデータ同時送受信のブロック図を示します。以下に調歩同期式シリアルデータ同時送受信のブロック図について説明します。

- 調歩同期式モードは、キャラクタ単位で同期をとる調歩同期方式でシリアルデータ通信を行います。
- Universal Asynchronous Receiver/Transmitter（UART）や、Asynchronous Communication Interface Adapter（ACIA）などの標準の調歩同期式通信用 LSI とのシリアルデータ通信ができます。
- 複数のプロセッサとシリアル通信ができるマルチプロセッサ間通信機能を備えています。
- 通信フォーマットを 12 種類のフォーマットから選択できます。
- 独立した送信部と受信部を備えているので、送信と受信を同時に行うことができます。また、送信部および受信部ともにダブルバッファ構造になっているため、連続送信・連続受信ができます。
- 内蔵のボーレートジェネレータで任意のビットレートを選択可能です。
- 送受信クロックソースを内部クロック、または外部クロックから選択可能です。
- 割り込み要因には送信終了、送信データエンプティ、受信データフル、オーバランエラー、フレーミングエラー、パリティエラーの 6 種類の割り込み要因があります。
- レシーブシフトレジスタ（RSR）は、シリアルデータを受信するためのレジスタです。RSR に RXD 端子から入力されたシリアルデータを、LSB（ビット 0）から受信した順にセットしパラレルデータに変換します。1 バイトのデータを受信すると、データは自動的に RDR へ転送されます。CPU から RSR を直接リード／ライトすることはできません。
- レシーブデータレジスタ（RDR）は、受信したシリアルデータを格納する 8 ビットのレジスタです。1 バイトのデータの受信が終了すると、受信したデータを RSR から RDR へ転送し、受信動作を完了します。その後、RSR は受信可能となります。RSR と RDR はダブルバッファになっているため連続した受信動作が可能です。RDR は受信専用レジスタなので CPU からライトできません。
- トランスミットシフトレジスタ（TSR）は、シリアルデータを送信するためのレジスタです。
  TDR から送信データをいったん TSR に転送し、LSB（ビット 0）から順に TXD 端子に送出することでシリアルデータ送信を行います。1 バイトのデータを送信すると、自動的に TDR から TSR へ次の送信データを転送し、送信を開始します。ただし、TDR にデータが書き込まれていない（TDRE に"1"がセットされている）場合には TDR から TSR へのデータ転送は行いません。CPU から TSR を直接リード／ライトすることはできません。
- トランスミットデータレジスタ（TDR）は、送信データを格納する 8 ビットのレジスタです。TSR の"空"を検出すると、TDR に書き込まれた送信データを TSR に転送し、シリアルデータ送信を開始します。TSR のシリアルデータ送信中に、TDR に次の送信データをライトしておくと、連続送信が可能です。TDR は、常に CPU によるリード／ライトが可能です。
- シリアルモードレジスタ（SMR）は、シリアルデータ通信フォーマットの設定と、ボーレートジェネレータのクロックソースを選択するための 8 ビットのレジスタです。SMR は、常に CPU によるリード／ライトが可能です。
- シリアルコントロールレジスタ 3（SCR3）は、送信／受信動作、調歩同期式モードでのクロック出力、割り込み要求の許可／禁止、および送信／受信クロックソースの選択を行う 8 ビットのレジスタです。SCR3 は、常に CPU によるリード／ライトが可能です。



【注】（a）SMRで選択されるボーレートジェネレータの動作クロックと合わせて、送信／受信のビットレートを設定します。本タスク例では、送信のビットレートを9600bit/sにて設定しています。
（b）シリアルデータ通信フォーマットの設定と、ボーレートジェネレータのクロックソースを選択します。本タスク例でシリアルデータ通信フォーマットは動作モードを調歩同期式に、データ長を8ビットに、パリティビットなし、ストップビット長を1ビットに内蔵ボーレートジェネレータのクロックソースをφクロックに設定しています。
（c）送信／受信動作、調歩同期式モードでのクロック出力、割り込み要求の許可／禁止を選択します。本タスク例では、調歩同期式モードでのクロック出力は、コミュニケーションモードを調歩同期式に、クロックソースを内部ボーレートジェネレータに設定しています。また、割り込み要求の許可／禁止は送信データエンプティ割り込み要求を禁止、受信データ、フル割り込み要求を禁止、受信エラー割り込み要求を禁止に設定しています。
（d）ステータスフラグ（トランスミットデータレジスタエンプティ、レシーブデータレジスタ、オーバランエラー、フレーミングエラー、パリティエラー、トランスミットエンド）によりSCI3 の動作状態を示す。
（e）TSR の"空"を検出することにより、TDRに書き込まれた送信データをTSRに転送。
（f）1バイトのデータの受信が終了すると、受信したデータをRSRからRDRへ転送。
（g）送信データ。
（h）受信データ。

図 2.1　歩同期式シリアルデータ同時送受信のブロック図

- シリアルステータスレジスタ（SSR）は、SCI3 の動作状態を示すステータスフラグと、マルチプロセッサビットを内蔵した 8 ビットのレジスタです。SSR は常に CPU からリード／ライトできます。ただし、TDRE、RDRF、OER、PER、FER へ"1"をライトすることはできません。また、これらに"0"をライトしてクリアするためには、あらかじめ"1"をリードしておく必要があります。また、TEND および MPBR はリード専用であり、ライトすることはできません。
- ビットレートレジスタ（BRR）は、SMR の CSK1、CKS0 で選択されるボーレートジェネレータの動作クロックとあわせて、送信／受信のビットレートを設定する 8 ビットのレジスタです。BRR は常に CPU によるリード／ライトが可能です。
- 表 2.1 に、調歩同期式モードの BRR の設定例を示します。表 2.1 はアクティブモードで、OSC が 16MHz のときの値を示しています。

表 2.1　ビットレートに対する BRR の設定例（調歩同期式モード）

R ビットレート（bit/s）	110	150	300	600	1200	2400	4800	9600	19200	31250	38400
n	3	2	2	1	1	0	0	0	0	0	0
N	70	207	103	207	103	207	103	51	25	15	12
誤差（%）	0.03	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.00	0.16

【注】 1. 誤差は、1%以内となるように設定します。

2. BRR の設定値は以下の計算式で求まります。

$$N = \frac{OSC}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

B：ビットレート（bit/s）
N：ボーレートジェネレータの BRR の設定値（0≦N≦255）
OSC：ϕ_{OSC} の値（MHz）＝16MHz
n：SMR の CKS1、CKS2 の設定値（0≦n≦3）（n とクロックの関係は表 2.2 を参照）

表 2.2　n とクロックの関係

n	クロック	SMR の設定値	
		CKS1	CKS
0	ϕ	0	0
1	ϕ/4	0	1
2	ϕ/16	1	0
3	ϕ/64	1	1

3. 表 2.1 に誤差は以下の計算式で求めた値を小数点第 3 位を四捨五入して表示してあります。

$$誤差（\%）= \left\{ \frac{\phi \times 10^6}{(N+1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

4. OSC が 16MHz のときの最大ビットレート（調歩同期式モード）は、500000（bit/s）になります。ただし、設定値は、n＝0、N＝0 のときです。

- 調歩同期式モードは、通信開始を意味するスタートビットと通信終了を意味するストップビットとをデータに付加したキャラクタを送信／受信し、1 キャラクタ単位で同期をとりながらシリアル通信を行うモードです。
- SCI3 内部では、送信部と受信部は独立しているので、全二重通信を行うことができます。また、送信部と受信部がともにダブルバッファ構造になっているので、送信中にデータのライト、受信中にデータのリードができ、連続送信／受信が可能です。
- 図 2.2 に調歩同期式通信のデータフォーマットを示します。調歩同期式通信では、通信回線は通常マーク状態（"High"レベル）に保たれています。SCI3 では通信回線を監視し、スペース（"Low"レベル）になったところをスタートビットとみなしてシリアル通信を開始します。
- 通信データの 1 キャラクタはスタートビット（"Low"レベル）から始まり、送信／受信データ（LSB ファースト：最下位ビットから）、パリティビット（"High"または"Low"レベル）、最後にストップビット（"High"レベル）の順で構成されます。
- 調歩同期式モードでは、受信時にスタートビットの立ち下がりエッジで同期化を行います。また、データを 1 ビット期間の 16 倍の周波数のクロックの 8 番目でサンプリングするので、各ビットの中央で通信データを取り込みます。

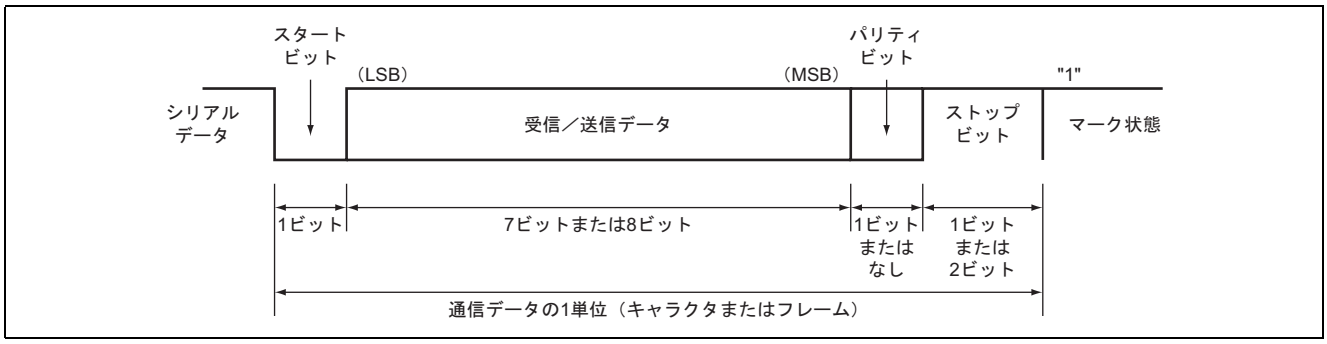


図 2.2　調歩同期式通信のデータフォーマット

- SCI3 クロック（SCK3）は、SCI3 のクロック入出力端子です。
- SCI3 レシーブデータ入力（RXD）は、SCI3 の受信データ入力端子です。
- SCI3 トランスミットデータ出力（TXD）は、SCI3 の送信データ出力端子です。
- SCI3 の割り込み要因には、送信終了、送信データエンプティ、受信データフルおよび 3 種類の受信エラー（オーバランエラー、フレーミングエラー、パリティエラー）の計 6 種類があり、共通のベクタアドレスが割り付けられています。
- 各割り込み要求は、SCR3 の TIE、RIE で許可／禁止できます。
- SSR の TDRE が"1"にセットされると TXI が発生します。SSR の TEND が"1"にセットされると、TEI が発生します。この 2 つの割り込みは送信時に発生します。
- SSR の TDRE は初期値が"1"になっています。したがって送信データを TDR へ転送する前に SCR3 の TIE を"1"にセットして送信データエンプティ割り込み要求（TXI）を許可すると、送信データが準備されていなくても TXI が発生します。
- SSR の TEND は初期値が"1"になっています。したがって、送信データを TDR へ転送する前に SCR3 の TEIE を"1"にセットして送信終了割り込み要求（TEI）を許可すると、送信データが送信されていなくても TEI が発生します。
- 送信データを TDR へ転送する処理を割り込み処理ルーチンの中で行うようにすることで、これらの割り込みを有効に利用できます。また、これらの割り込み要求（TXI、TEI）の発生を防ぐためには、送信データを TDR へ転送した後に、これらの割り込み要求に対応する許可ビット（TIE、TEIE）を"1"にセットします。
- SSR の RDRF が"1"にセットされると RXI が発生します。OER、PER、FER のいずれかが"1"にセットされると ERI が発生します。この 2 つの割り込み要求は受信時に発生します。

2. 表 2.3 に本タスク例の機能割り付け示します。表 2.3 に示すように機能を割り付け、調歩同期式シリアルデータ同時送受信を行います。

表 2.3　機能割り付け

機能	機能割り付け
RSR	シリアルデータを受信するためのレジスタ
RDR	受信データを格納するレジスタ
SMR	シリアルデータ通信フォーマット、ボーレートジェネレータのクロックソースの設定
SSR	SCI3 の動作状態を示すステータスフラグ
BRR	送信／受信のビットレートを設定
PMR1	TXD 出力端子設定
SCK3	SCI3 のクロック出力端子
TXD	SCI3 の送信データ出力端子
RXD	SCI3 の受信データ入力端子

3. 動作原理

図 3.1 に動作原理を示します。図 3.1 に示すようなハードウェア処理、およびソフトウェア処理により調歩同期式シリアルデータ同時送受信を行います。

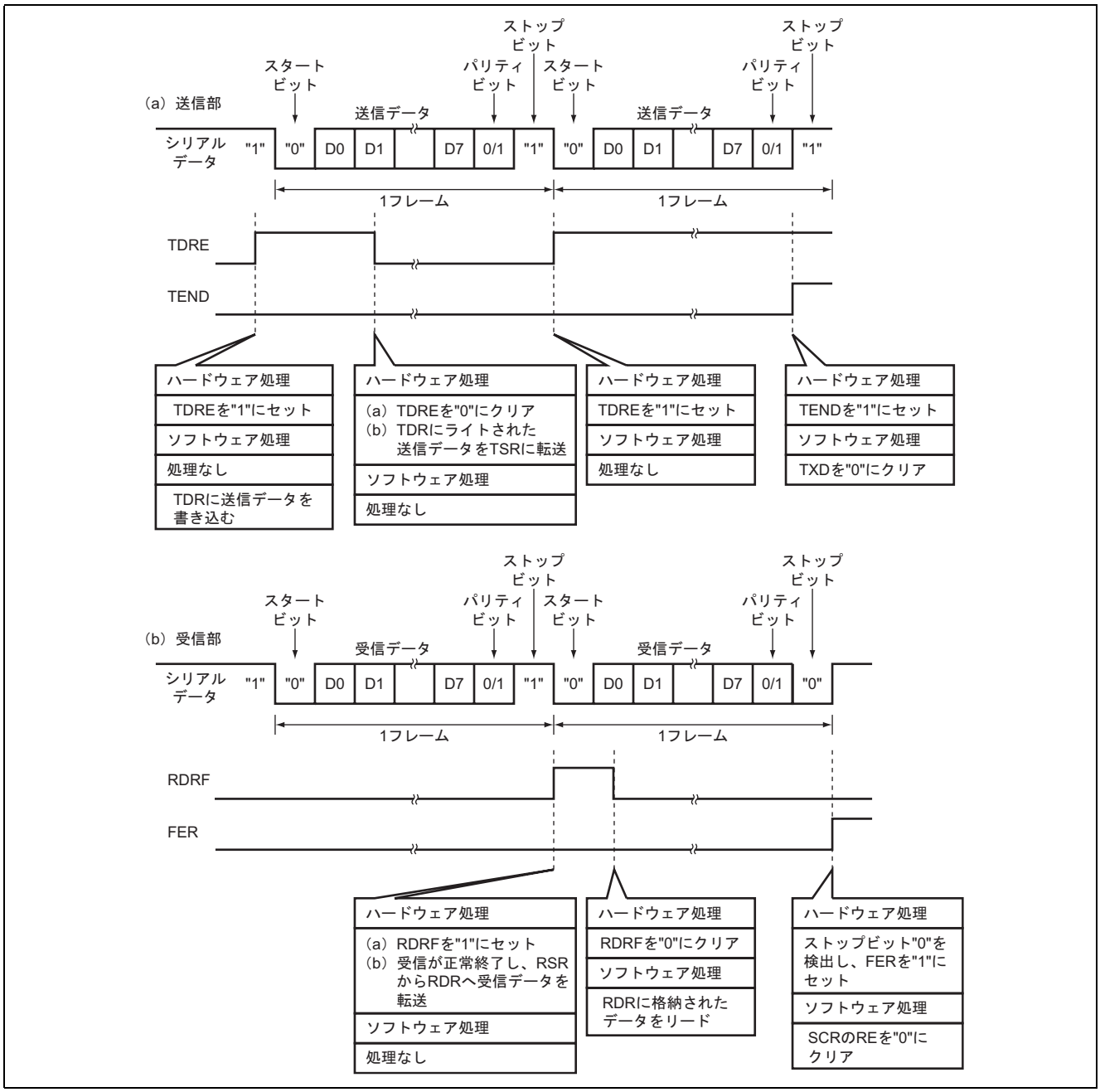


図 3.1　調歩同期式シリアルデータ同時送受信の動作原理

4. ソフトウェア説明

4.1 モジュール説明

本タスク例におけるモジュール説明（引数・リターン値）を表 4.1 に示します。

表 4.1 モジュール説明

モジュール（関数）名	引数	リターン値	機能
INIT（アセンブリ言語）	なし	なし	スタックポインタの設定（R7 に H'FF80 をセット）CCR の設定（割り込み禁止）main へジャンプする
main	なし	なし	メインモジュール
Init_sci	なし	なし	シリアルステータスレジスタを 0 クリア シリアル通信用設定（調歩同期モード、データ長 8bit、パリティなし、ストップビット 1、ビットレート 9600bps）TXI、RXI、ERI 割り込み処理と送受信を許可
Dsp_Init	なし	なし	パソコンのターミナルウィンドウにプログラムのバージョンを表示するためのデータを設定
Int_SCI3	なし	なし	シリアル割り込み処理シリアルステータスレジスタがエラー、送信、受信の状態か判定
Int_s3err	なし	なし	受信カウンタとシリアルステータスレジスタを 0 にクリア
Int_s3rx	なし	なし	シリアル受信処理
Int_s3tx	なし	なし	シリアル送信処理
Ans_send	なし	なし	返答データセット処理
Dump_send	なし	なし	メモリダンプデータセット処理
Dsp_mem	なし	なし	メモリエディット前データセット処理
Dsp_err	なし	なし	エラー用データセット処理
Set_send1	work（送信データ）	なし	送信データセット処理
Dsp_idle	なし	なし	改行用データセット処理
Dump_start	adrs （ダンプ開始アドレス）	なし	メモリダンプ処理用変数セット
Cnvt_AsctoAddress	なし	なし	アドレスデータ変換 （4ByteASCII→ByteHex）
Cnvt_AsctoHex	*ptr （変換データ格納アドレス）	data （1ByteHex データ）	2ByteASCII データを 1ByteHex データに変換
Cnvt1_AsctoHex	*ptr （変換データ格納アドレス）	data （4bitHex データ）	1ByteASCII データを 4bitHex データに変換
Cnvt1_HextoAsc	data （1ByteHex データ）	*ptr （ASCII 変換データ格納アドレスへのポインタ）	1ByteHex データを 2ByteASCII データに変換
Int_trap	なし	なし	トラップ割り込み処理
NMI	なし	なし	NMI 割り込み処理（処理は何も行わない）
Int_s3te	なし	なし	SCI1TEI 割り込み処理 （処理は何も行わない）

4.2 構成ファイル

本タスクのファイル一覧表を表 4.2 に示します。

表 4.2 ファイル構成表

ファイル名	処理内容
DBSCT.C	未初期化エリアの初期化処理（HEW 自動生成）
INIT.SRC	スタックポインタ・CCR 設定（RESET 時）
KMONI.C	メインプログラム、シリアル割り込み処理、通信サブルーチン処理、ベクタテーブル定義など
ASCDEFINE.H	アスキーコード定義（ヘッダーファイル）

4.3 セクション定義

本タスクのセクション定義を表 4.3 に示します。

表 4.3 セクション定義表

アドレス	セクション	説　　明
H'0000	CV1	RESET ベクタアドレス
H'0010	CV2	TRAP ベクタアドレス
H'002E	CV3	SCI ベクタアドレス
H'0100	P	プログラム領域
	C$DSEC	初期化データ領域（DBSCT.C にて定義）
	C$BSEC	未初期化データ領域（DBSCT.C にて定義）
	D	初期化データ領域
	C	定数領域
H'FB80	B	未初期化データ領域
	R	初期化データ領域

4.4 グローバル変数の説明

本タスク例におけるグローバル変数の説明を表 4.4 に示します。

表 4.4 使用グローバル変数説明

変数名	型	サイズ	用　　途
s3rx_buf	unsigned char	16	受信データバッファ
s3tx_buf	unsigned char	60	送信データバッファ
s3rx_cnt	unsigned char	1	受信済みデータのビット数
s3tx_cnt	unsigned char	1	送信済みデータのビット数
s3tx_cnt	unsigned char	1	送信データのビット長
dummy	unsigned char	1	シリアルステータスレジスタダミーライト用エリア
trap_flag	unsigned char	1	トラップ割り込み処理用フラグ
moni_code	unsigned char	1	データ受信後、再受信時の動作選択用
ans_code	unsigned char	1	返答データ生成時の動作選択用
dump_ycnt	unsigned char	1	メモリダンプ時のダンプ行数カウント
*cr_adrs	unsigned char	1	メモリダンプ／メモリエディットを行うアドレス
*dump_adrs	unsigned char	1	メモリダンプ／メモリエディットを行うアドレス
CMem.data	unsigned char	2	メモリダンプ／メモリエディットを行うアドレス （*CMem.adrs と共用体）
*CMem.adrs	unsigned char	1	メモリダンプ／メモリエディットを行うアドレス （*CMem.data と共用体）

4.5 使用内部レジスタの説明

本タスク例で使用する内部レジスタ説明を表 4.5 に示します。

表 4.5 使用内部レジスタの説明

レジスタ名		機　　能	操作	設定値
SMR	COM	コミュニケーションモードを調歩同期式モードに設定	設定	0
	CHR	調歩同期式モード時におけるデータ長を 8 ビットデータに設定	設定	0
	PE	調歩同期式モードで、送信時にパリティビットの付加およびチェックを行わない	設定	0
	PM	SMR の PE＝"0"のため無効	設定	0
	STOP	調歩同期式モードでのストップビットの長さを 1 ビットに設定	設定	0
	MP	マルチプロセッサ通信機能を禁止	設定	0
	CKS1 CKS0	内蔵ボーレートジェネレータのクロックソースをφクロックに設定	設定	CKS1＝"0" CKS0＝"0"
BRR		SMR の CKS1、CKS0 で選択されるボーレートジェネレータの動作クロックとあわせて送信のビットレートを 9600（bit/s）に設定	設定	H'51
SCR3	TIE	"1"のとき、TXI 割り込み要求を許可	設定	0/1
	RIE	RXI および ERI 割り込み要求を許可	設定	1
	TE	送信動作を許可	設定	1
	RE	受信動作を許可	設定	1
	MPIE	SMR の MP＝"0"のため無効	設定	0
	TEIE	TEI 割り込み要求を拒否	設定	0
	CKE1 CKE0	クロックソースを内部ボーレートジェネレータに設定	設定	CKE1＝"0" CKE0＝"0"
TDR		送信データを格納する 8 ビットのレジスタ	格納	－
RDR		受信データを格納する 8 ビットのレジスタ	格納・参照	－
SSR	TDRE	"0"：TDR に送信データをライトしたとき、もしくは"1"の状態をリードした後、"0"をライトしたとき "1"：TDR から TSR にデータが転送されたとき	格納・参照	0/1
	RDRF	"0"：RDR のデータをリードしたとき、もしくは"1"の状態をリードした後、"0"をライトしたとき "1"：受信が正常終了し、RSR から RDR へデータが転送されたとき	格納・参照	0/1
	OER	"0"："1"の状態をリードした後、"0"をライトしたとき "1"：受信中にオーバランエラーが発生した時	格納・参照	0/1
	FER	"0"："1"の状態をリードした後、"0"をライトしたとき "1"：受信中にフレーミングエラーが発生した時	格納・参照	0/1
	PER	"0"："1"の状態をリードした後、"0"をライトしたとき "1"：受信中にパリティエラーが発生した時	格納・参照	0/1
	TEND	"0"：TDR へ送信データをライトしたとき、もしくは"1"の状態をリードした後、"0"をライトしたとき "1"：送信キャラクタの最後のビットの送信時、TDRE が"1"のとき	格納・参照	0/1
	MPBR	受信キャラクタ中のマルチプロセッサビットを格納 SCR3 の RE が"0"のときは変化なし	参照	－
	MPBT	送信キャラクタ中のマルチプロセッサビットの値を指定	参照	－
PMR1		$P2_2$/TXD 端子機能を TXD 端子機能に設定	設定	0x02

4.6 モニタ制御方法

H8/3664 モニタ制御方法を下記に示します。

入力は半角英数字を使用してください。

入力文字を間違った場合は「Back Spece」で 1 つ前の文字を消すことが可能です。

メモリダンプ制御時の最初の文字「D」とアドレス部分のアルファベット（A～F）および、メモリエディット時の最初の文字「E」とアドレス、データ部分のアルファベット（A～F）は小文字での入力も可能です。

1. メモリのダンプを行う場合
   D を入力した後、ダンプデータの先頭アドレスを 4 ビットの HEX コードで入力し、[enter]を押すと指定されたアドレスから 128 ビット分のダンプデータが表示されます。

   例：0xFE80 番地からのダンプデータを表示したい場合→DFE80（図 4.1）

図 4.1　メモリデータダンプ（0xFE80 番地の例）

ダンプデータが表示された後、何も文字を入力せずに[enter]を押すと、ダンプデータの続きを 128 ビット分表示します。

例：図 4.1 の状態（0xFEFF 番地までのダンプデータを表示）で[enter]を押す
→0xFF00 番地からのダンプデータを 128 ビット分表示（図 4.2）

図 4.2　メモリデータダンプ（連続ダンプ）

2. メモリのエディットを行う場合
   E を入力した後、エディットしたいのアドレスを 4 ビットの HEX コードで入力し、[enter]を押します。
   アドレスを入力すると指定されたアドレスと、指定されたアドレスの現在の値が表示されます。

   例：0xFE00 番地のデータをエディットする場合→EFE00（図 4.3）

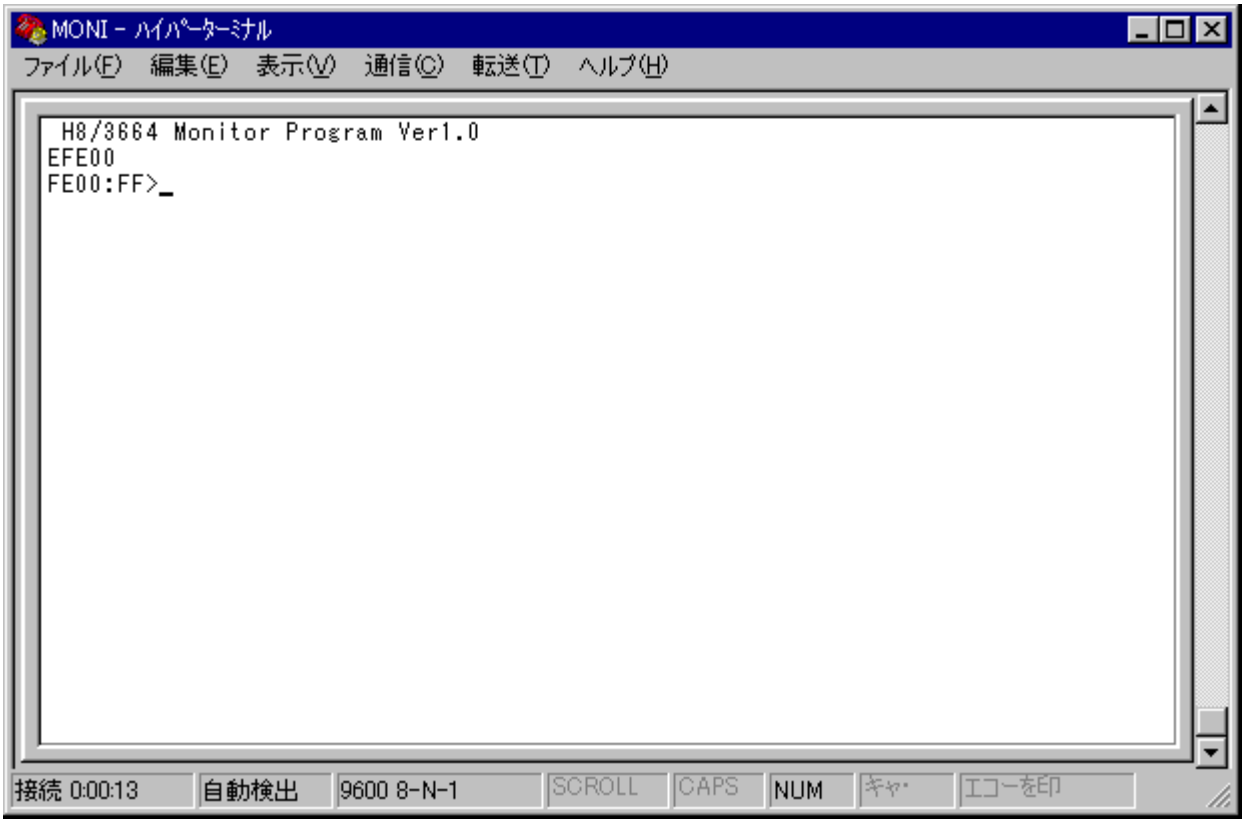


図 4.3　メモリデータエディット（アドレス指定）

アドレスを指定し、指定されたアドレスのデータが表示されている状態で、変更したい値を入力し、[enter]を押します。

例：図 4.3 の状態で 0xFE00 番地のデータを FF から 00 に変更する→FE00：FF>00（図 4.4）

図 4.4　メモリデータエディット（データ変更）

アドレスと現在の値が表示されている状態で、何も文字を入力せずに[enter]を押すと、1 つ後ろのアドレスのデータをエディットすることが出来ます（図 4.5）

図 4.5　メモリデータエディット（アドレス送り）

アドレスと現在の値が表示されている状態で、何も文字を入力せずに「^」を押すと、1 つ前のアドレスのデータをエディットすることが出来ます（図 4.6）

図 4.6　メモリデータエディット（アドレス戻し）

メモリのエディットを終了したい場合は何も文字を入力せずに、「.」を押すと、メモリのエディットを終了することが出来ます。（図 4.7）

図 4.7　メモリデータエディット（エディット終了）

4.7 モジュール階層図

モジュール階層図を図 4.8 に示します。

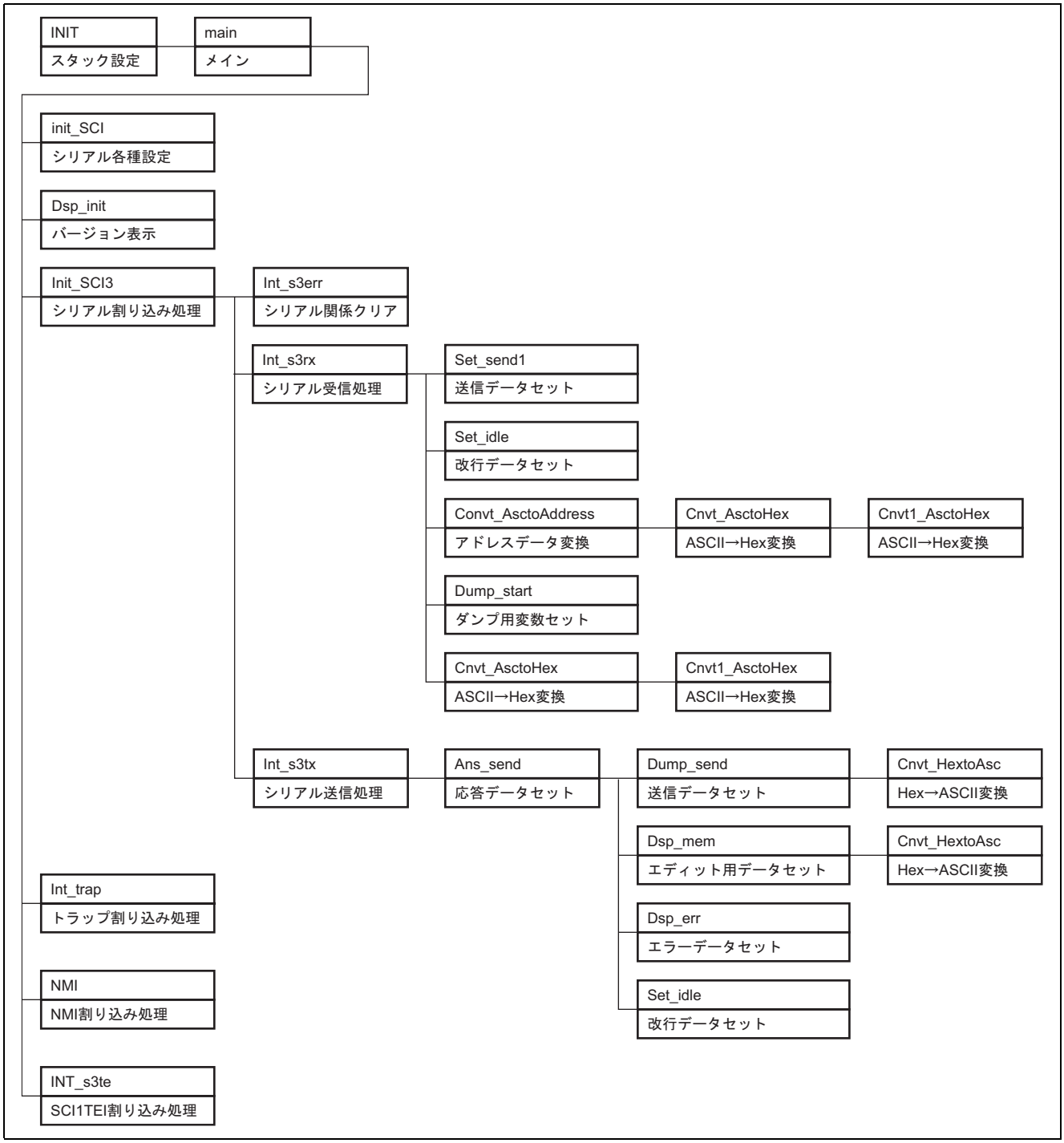
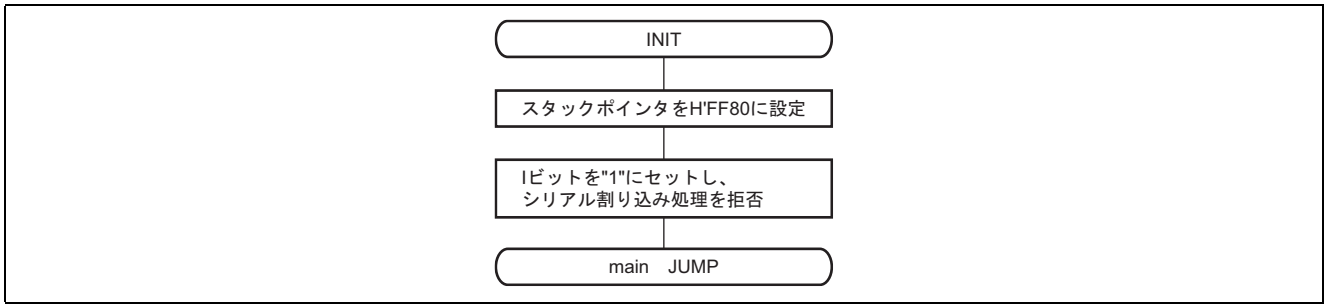
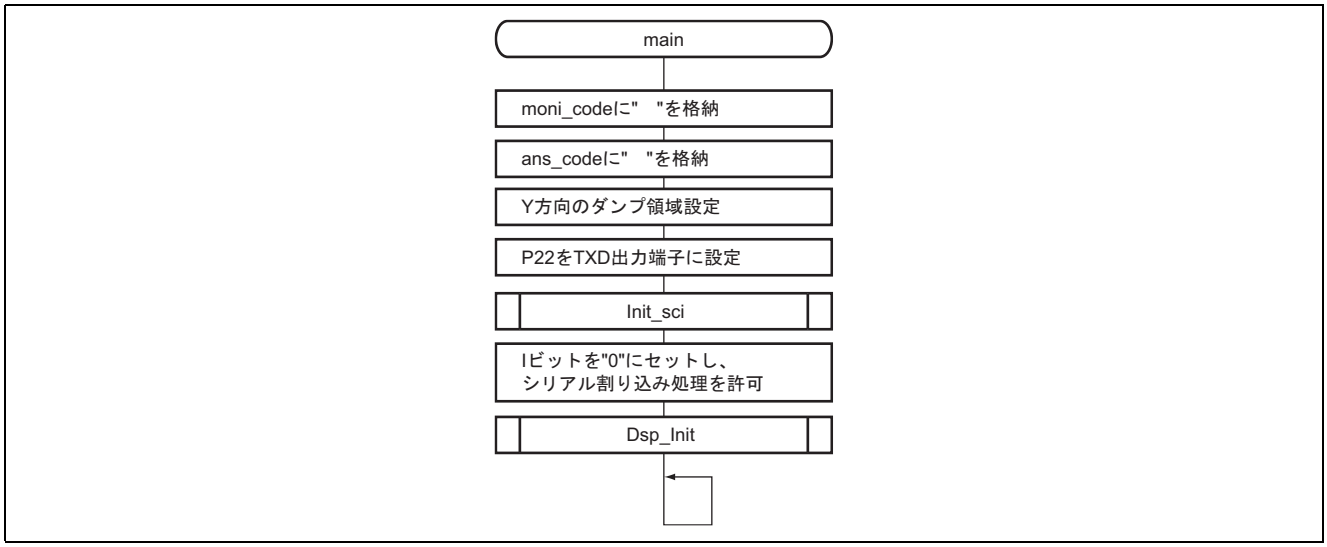
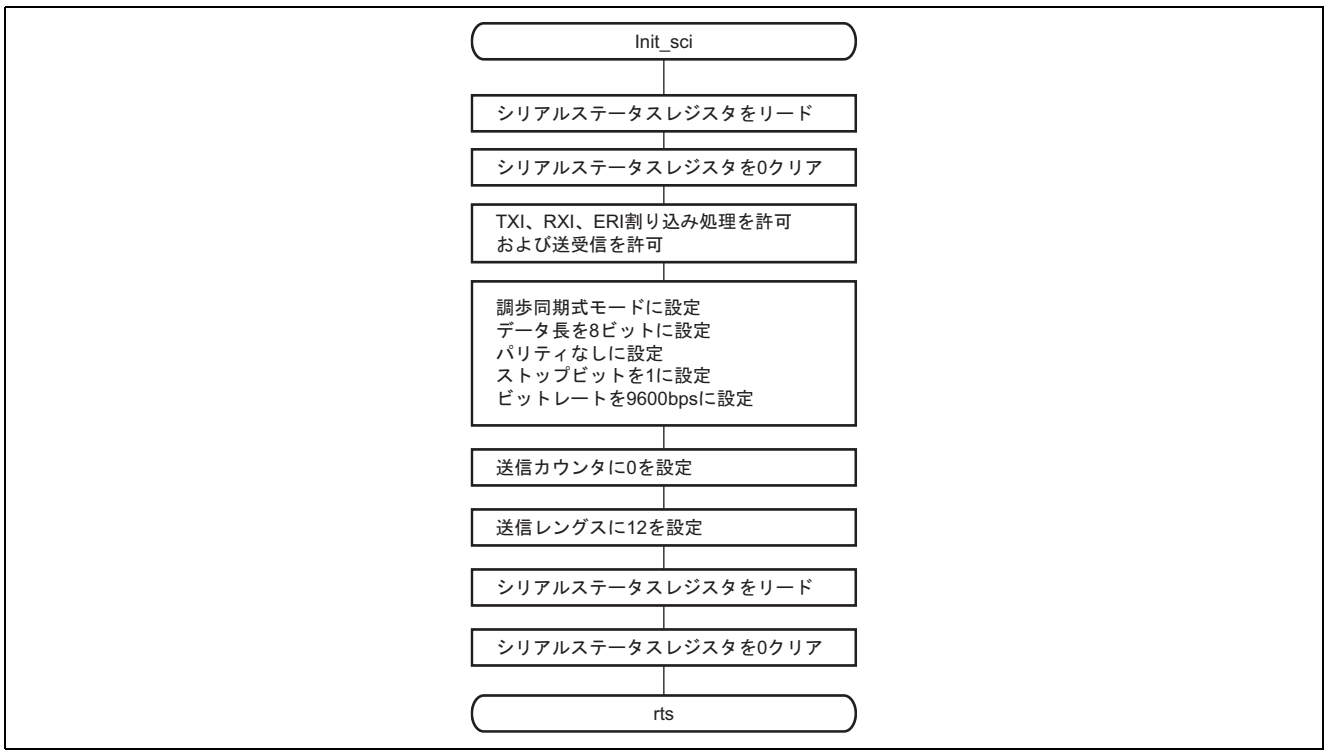


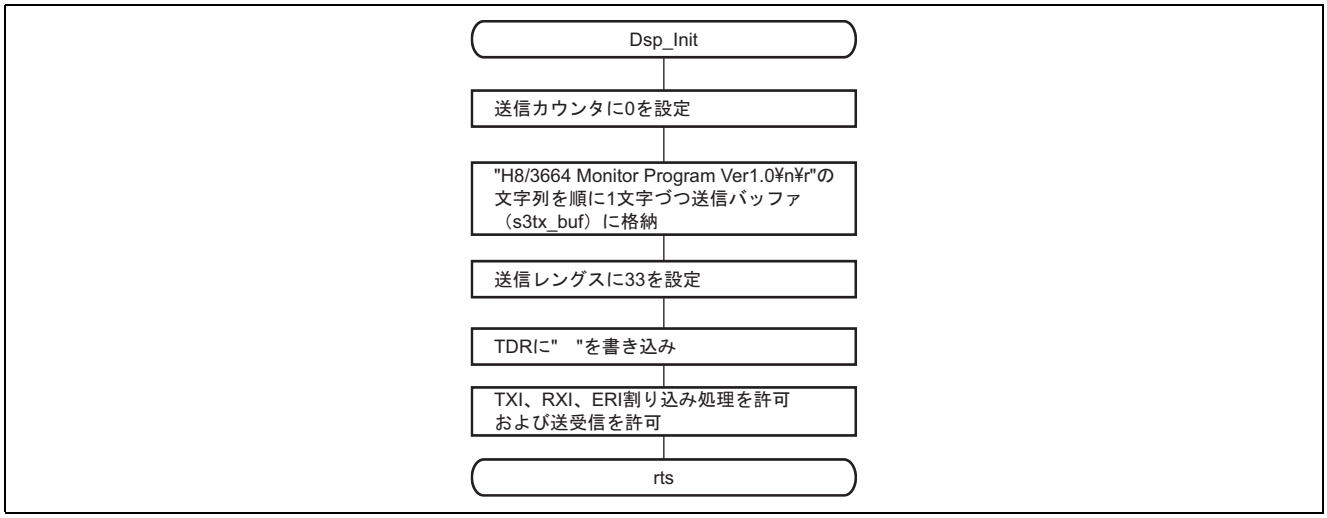
図 4.8　モジュール階層図

5. フローチャート

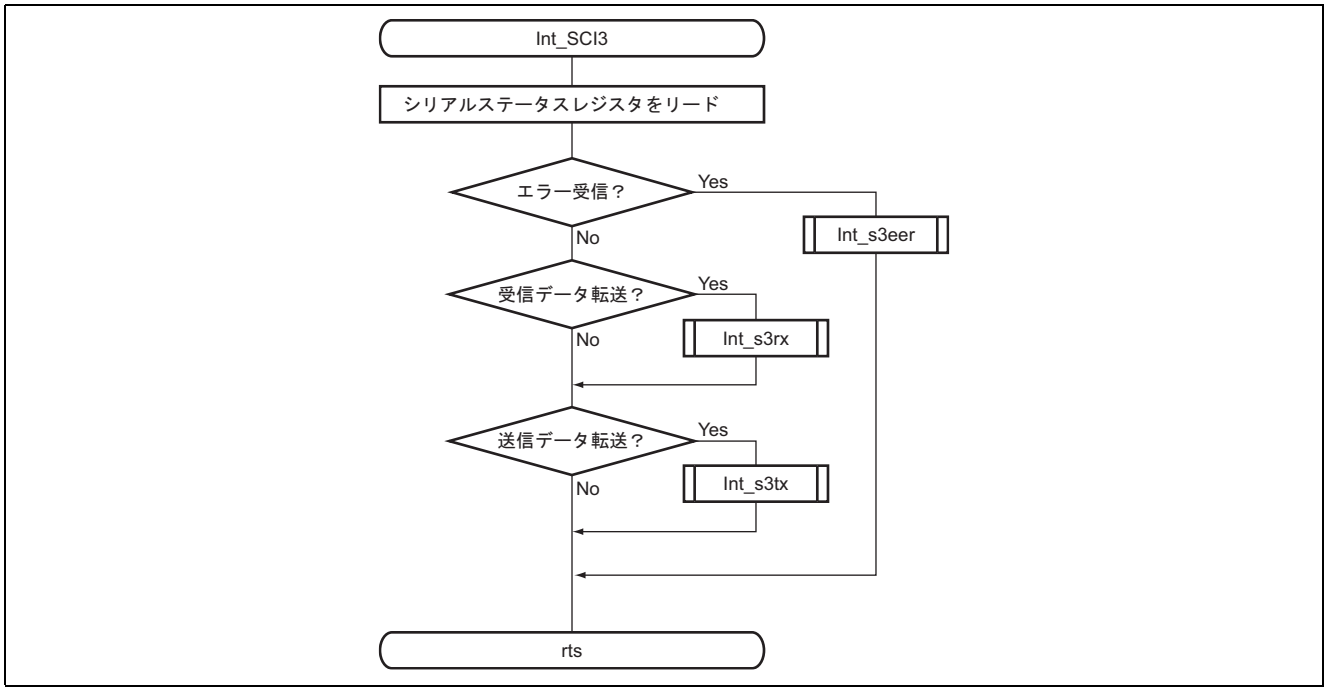

INIT
スタックポインタをH'FF80に設定
Iビットを"1"にセットし、
シリアル割り込み処理を拒否
main JUMP

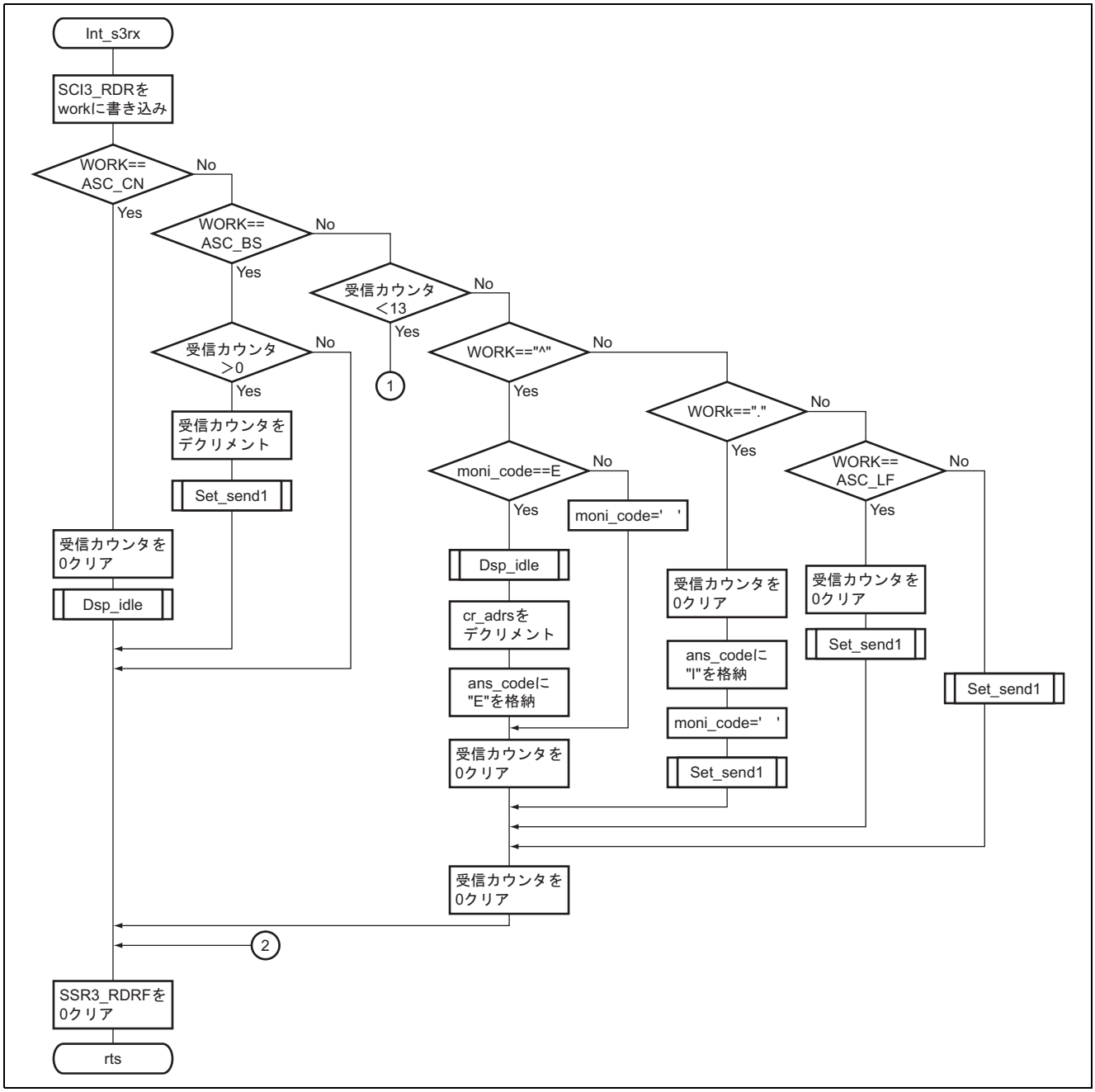

main
moni_codeに" "を格納
ans_codeに" "を格納
Y方向のダンプ領域設定
P22をTXD出力端子に設定
Init_sci
Iビットを"0"にセットし、
シリアル割り込み処理を許可
Dsp_Init

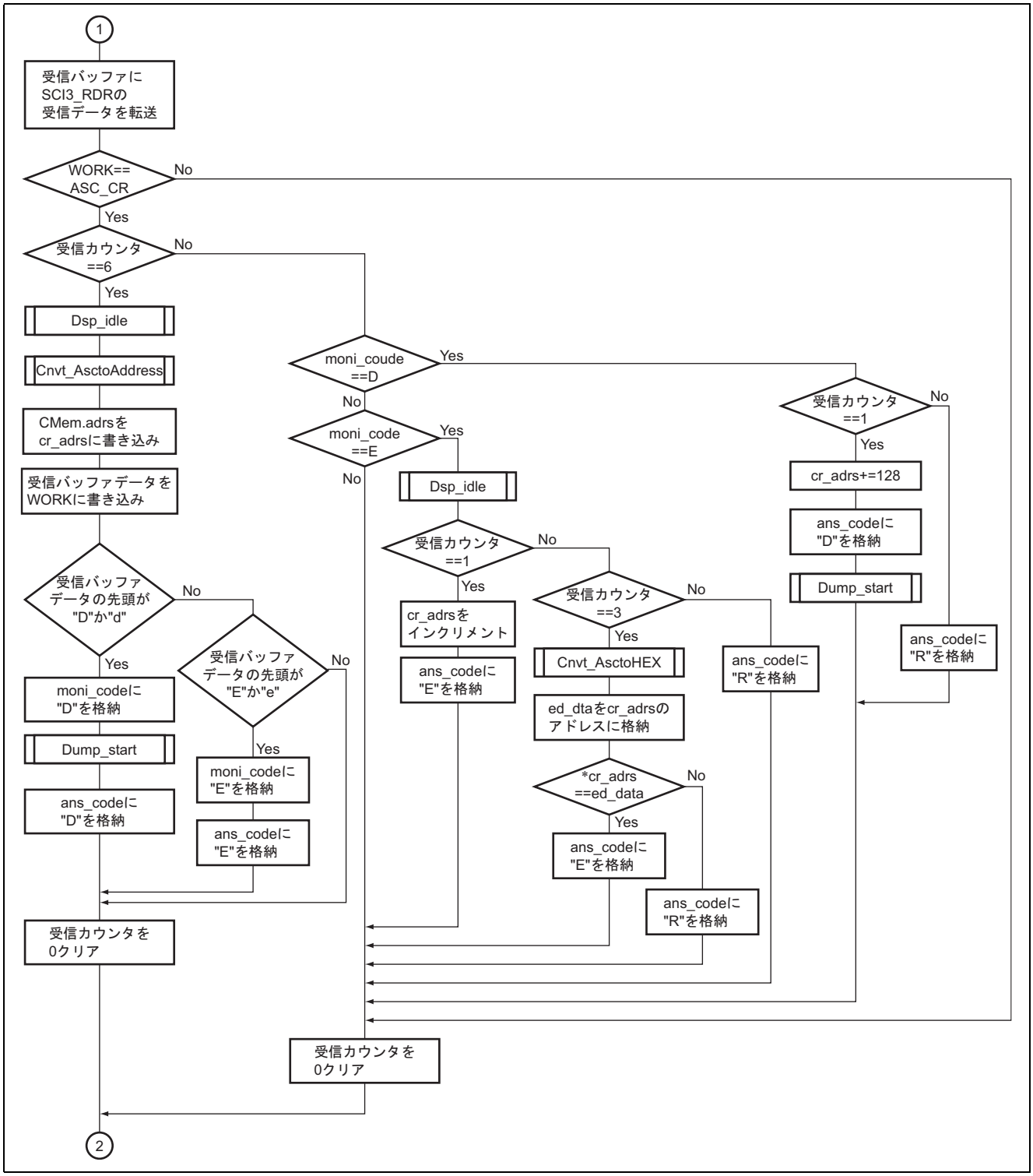

Init_sci
シリアルステータスレジスタをリード
シリアルステータスレジスタを0クリア
TXI、RXI、ERI割り込み処理を許可
および送受信を許可
調歩同期式モードに設定
データ長を8ビットに設定
パリティなしに設定
ストップビットを1に設定
ビットレートを9600bpsに設定
送信カウンタに0を設定
送信レングスに12を設定
シリアルステータスレジスタをリード
シリアルステータスレジスタを0クリア
rts

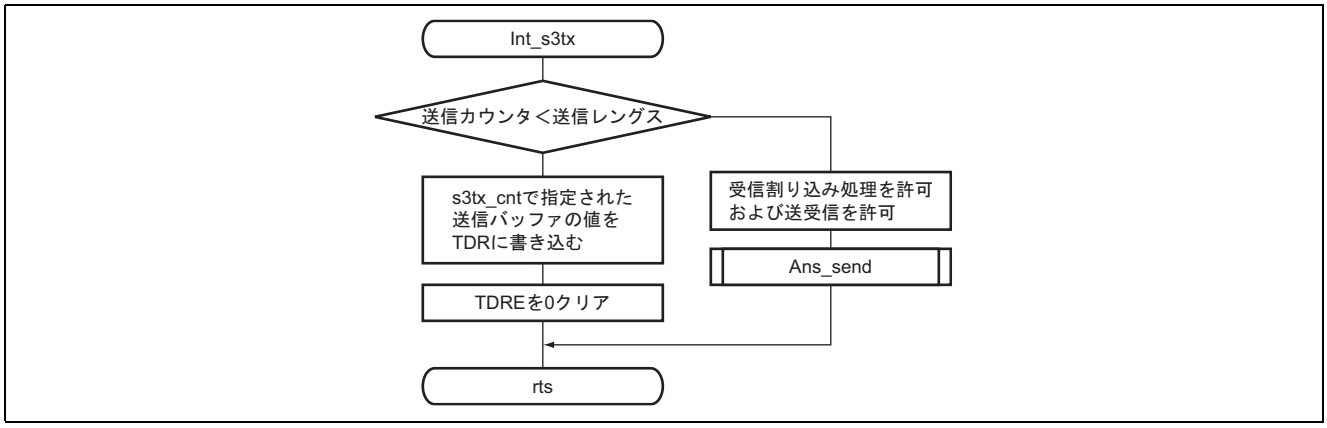

Dsp_Init
送信カウンタに0を設定
"H8/3664 Monitor Program Ver1.0¥n¥r"の
文字列を順に1文字づつ送信バッファ
（s3tx_buf）に格納
送信レングスに33を設定
TDRに" "を書き込み
TXI、RXI、ERI割り込み処理を許可
および送受信を許可
rts

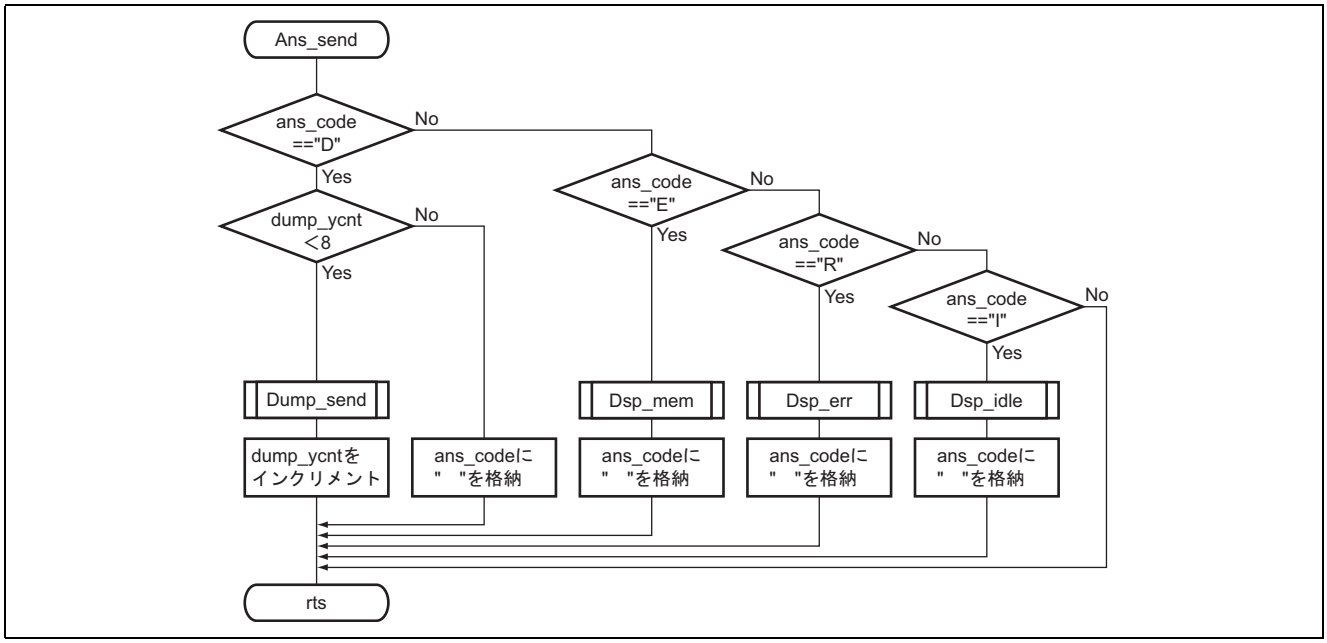
シリアル割り込み処理ルーチン

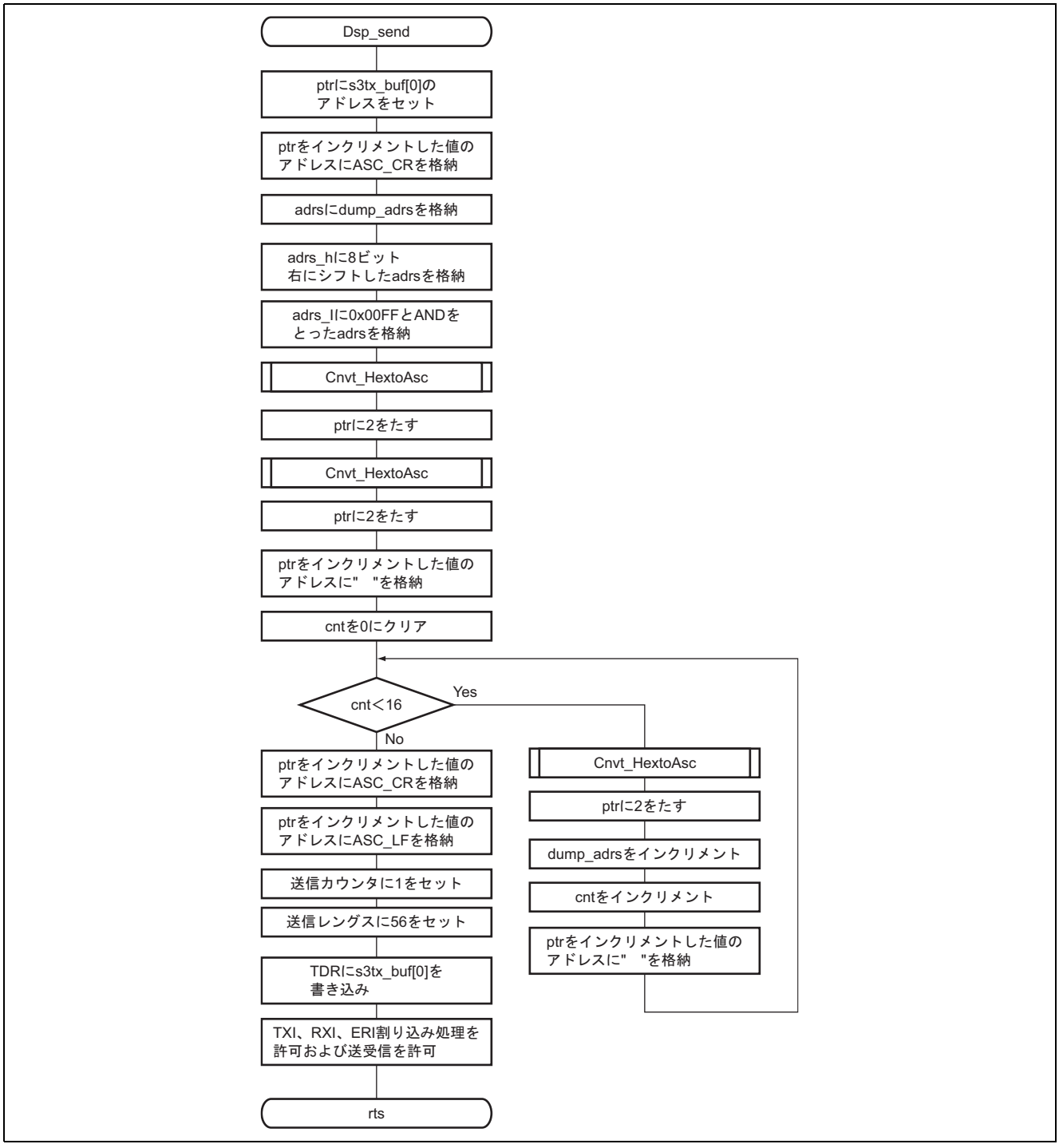


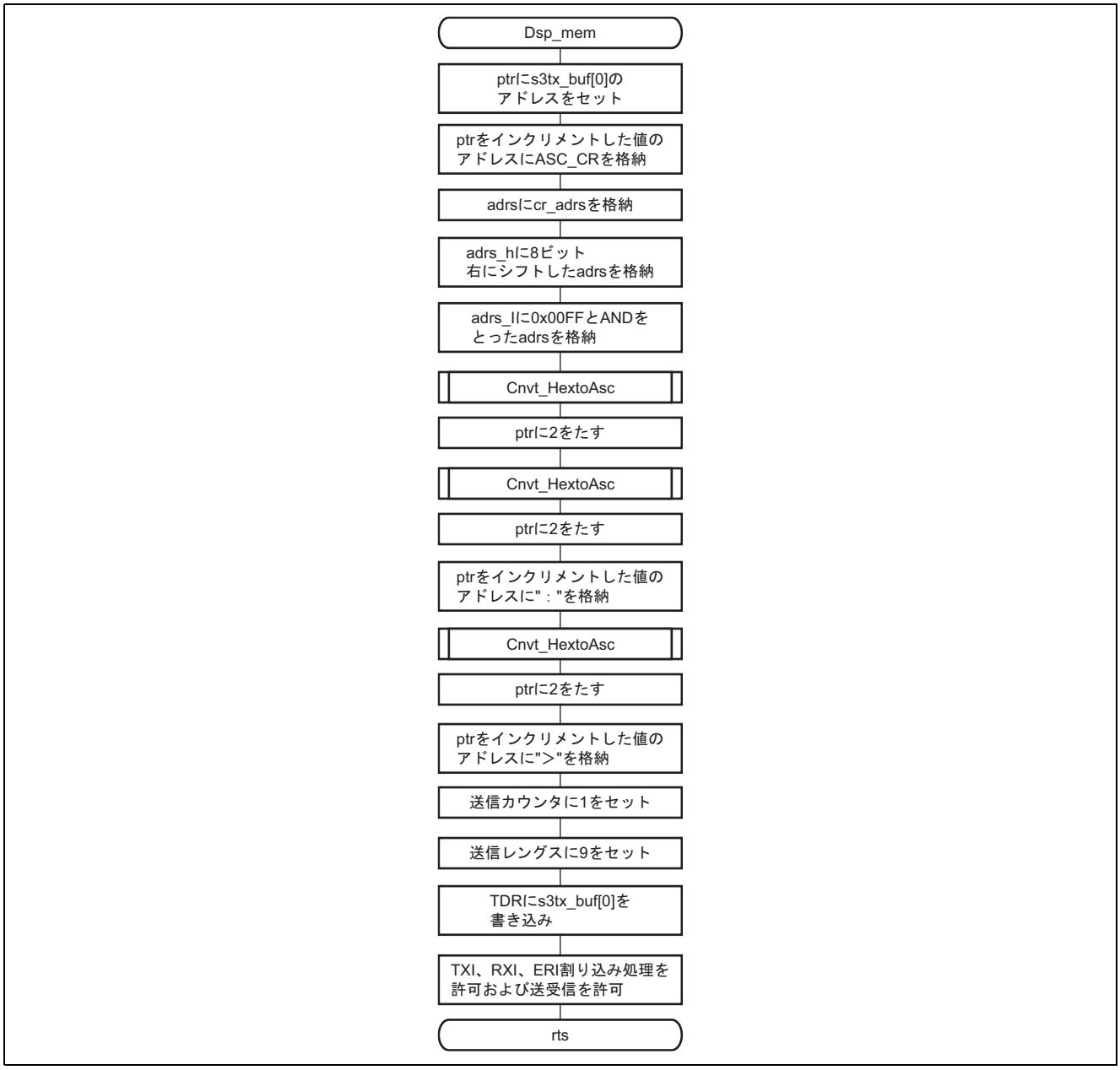

Int_s3rx
SCI3_RDRを workに書き込み
WORK== ASC_CN
No
Yes
WORK== ASC_BS
No
Yes
受信カウンタ ＜13
No
Yes
1
受信カウンタ ＞0
No
Yes
受信カウンタを デクリメント
Set_send1
WORK=="^"
No
Yes
moni_code==E
No
Yes
moni_code=' '
Dsp_idle
cr_adrsを デクリメント
ans_codeに "E"を格納
受信カウンタを 0クリア
WORk=="."
No
Yes
受信カウンタを 0クリア
ans_codeに "I"を格納
moni_code=' '
Set_send1
WORK== ASC_LF
No
Yes
受信カウンタを 0クリア
Set_send1
Set_send1
受信カウンタを 0クリア
受信カウンタを 0クリア
2
SSR3_RDRFを 0クリア
rts

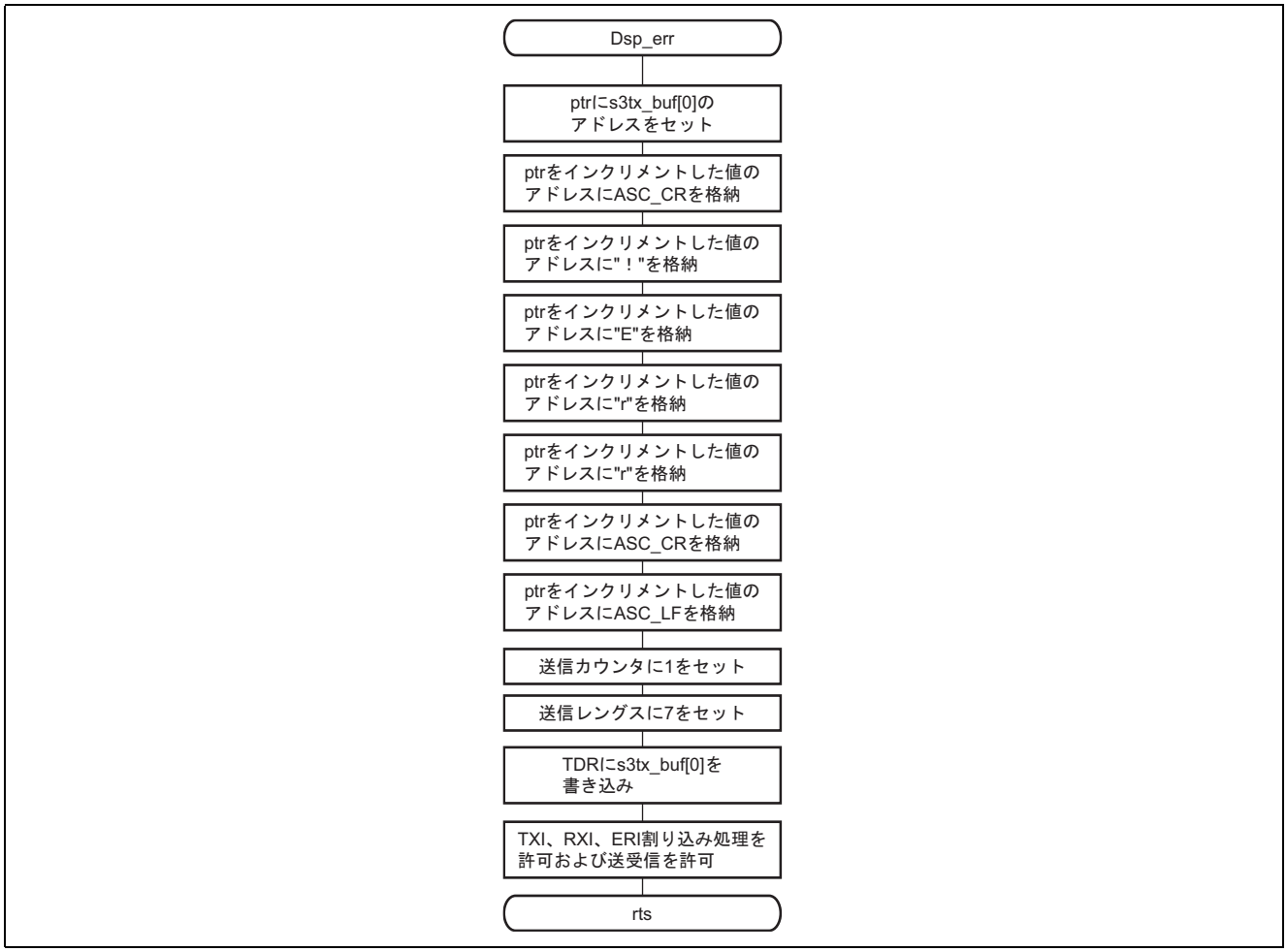

1
受信バッファに
SCI3_RDRの
受信データを転送
WORK==
ASC_CR
No
Yes
受信カウンタ
==6
No
Yes
Dsp_idle
Cnvt_AsctoAddress
CMem.adrsを
cr_adrsに書き込み
受信バッファデータを
WORKに書き込み
受信バッファ
データの先頭が
"D"か"d"
No
Yes
moni_codeに
"D"を格納
Dump_start
ans_codeに
"D"を格納
受信バッファ
データの先頭が
"E"か"e"
No
Yes
moni_codeに
"E"を格納
ans_codeに
"E"を格納
受信カウンタを
0クリア
moni_coude
==D
Yes
No
moni_code
==E
Yes
No
Dsp_idle
受信カウンタ
==1
No
Yes
cr_adrsを
インクリメント
ans_codeに
"E"を格納
受信カウンタ
==3
No
Yes
Cnvt_AsctoHEX
ed_dtaをcr_adrsの
アドレスに格納
*cr_adrs
==ed_data
No
Yes
ans_codeに
"E"を格納
ans_codeに
"R"を格納
ans_codeに
"R"を格納
受信カウンタを
0クリア
受信カウンタ
==1
No
Yes
cr_adrs+=128
ans_codeに
"D"を格納
Dump_start
ans_codeに
"R"を格納
2

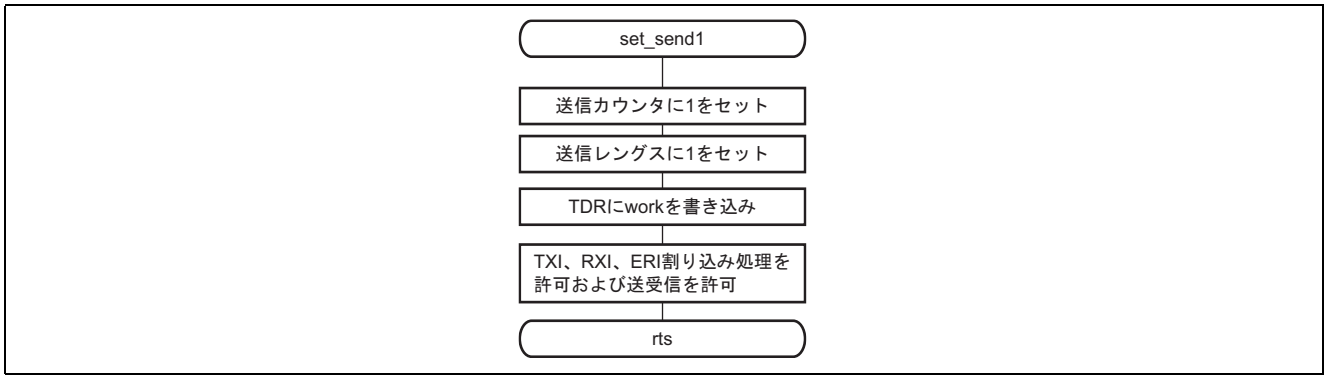

Int_s3tx
送信カウンタ＜送信レングス
s3tx_cntで指定された
送信バッファの値を
TDRに書き込む
TDREを0クリア
受信割り込み処理を許可
および送受信を許可
Ans_send
rts

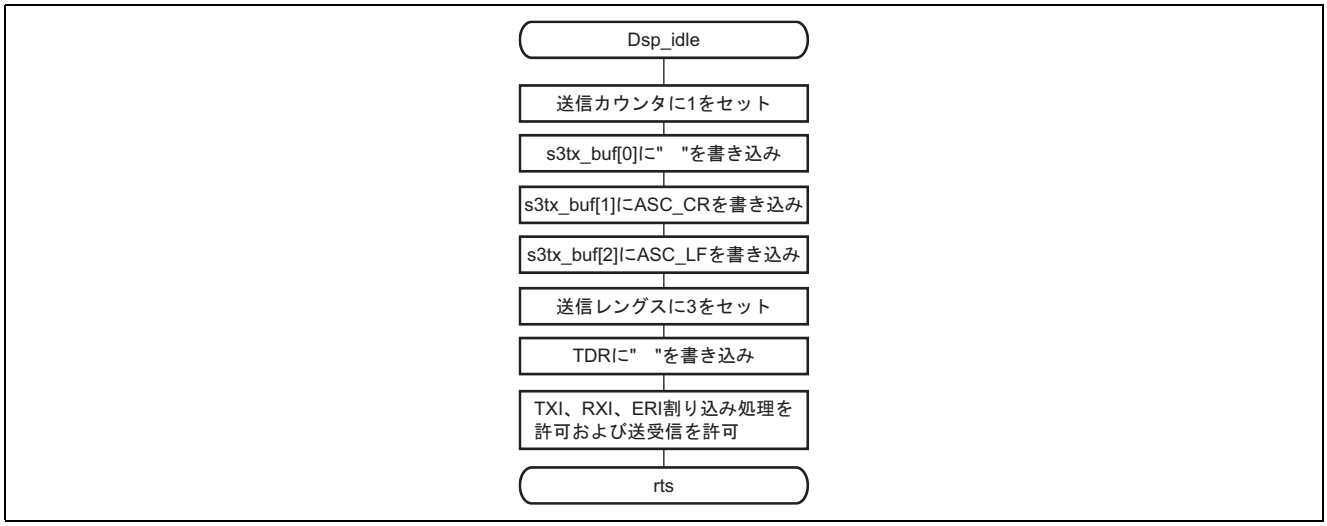

Ans_send
ans_code
=="D"
No
Yes
dump_ycnt
＜8
No
Yes
ans_code
=="E"
No
Yes
ans_code
=="R"
No
Yes
ans_code
=="I"
No
Yes
Dump_send
Dsp_mem
Dsp_err
Dsp_idle
dump_ycntを
インクリメント
ans_codeに
" "を格納
ans_codeに
" "を格納
ans_codeに
" "を格納
ans_codeに
" "を格納
rts

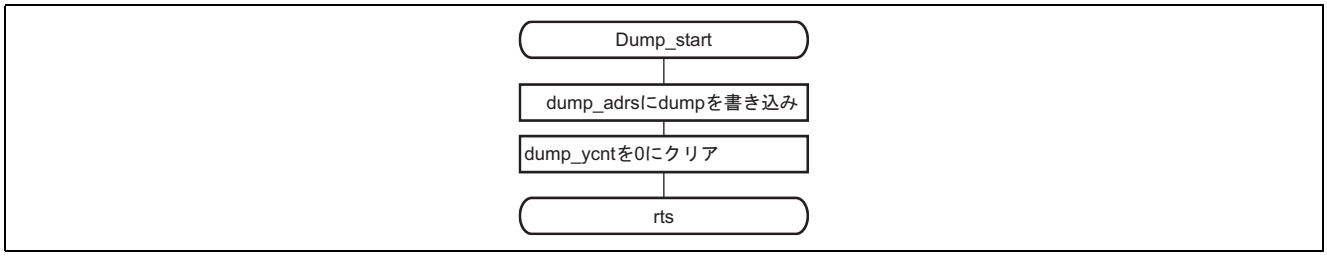

Dsp_send
ptrにs3tx_buf[0]の
アドレスをセット
ptrをインクリメントした値の
アドレスにASC_CRを格納
adrsにdump_adrsを格納
adrs_hに8ビット
右にシフトしたadrsを格納
adrs_lに0x00FFとANDを
とったadrsを格納
Cnvt_HextoAsc
ptrに2をたす
Cnvt_HextoAsc
ptrに2をたす
ptrをインクリメントした値の
アドレスに" "を格納
cntを0にクリア
cnt＜16
Yes
No
ptrをインクリメントした値の
アドレスにASC_CRを格納
ptrをインクリメントした値の
アドレスにASC_LFを格納
送信カウンタに1をセット
送信レングスに56をセット
TDRにs3tx_buf[0]を
書き込み
TXI、RXI、ERI割り込み処理を
許可および送受信を許可
rts
Cnvt_HextoAsc
ptrに2をたす
dump_adrsをインクリメント
cntをインクリメント
ptrをインクリメントした値の
アドレスに" "を格納

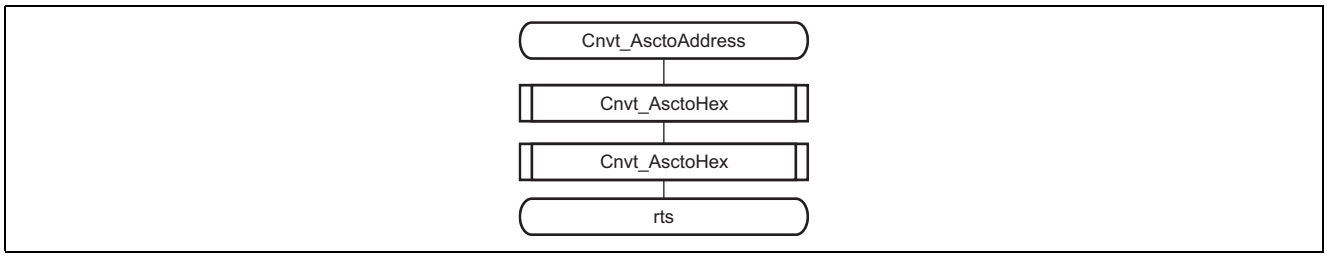

Dsp_mem
ptrにs3tx_buf[0]の
アドレスをセット
ptrをインクリメントした値の
アドレスにASC_CRを格納
adrsにcr_adrsを格納
adrs_hに8ビット
右にシフトしたadrsを格納
adrs_lに0x00FFとANDを
とったadrsを格納
Cnvt_HextoAsc
ptrに2をたす
Cnvt_HextoAsc
ptrに2をたす
ptrをインクリメントした値の
アドレスに" : "を格納
Cnvt_HextoAsc
ptrに2をたす
ptrをインクリメントした値の
アドレスに">"を格納
送信カウンタに1をセット
送信レングスに9をセット
TDRにs3tx_buf[0]を
書き込み
TXI、RXI、ERI割り込み処理を
許可および送受信を許可
rts

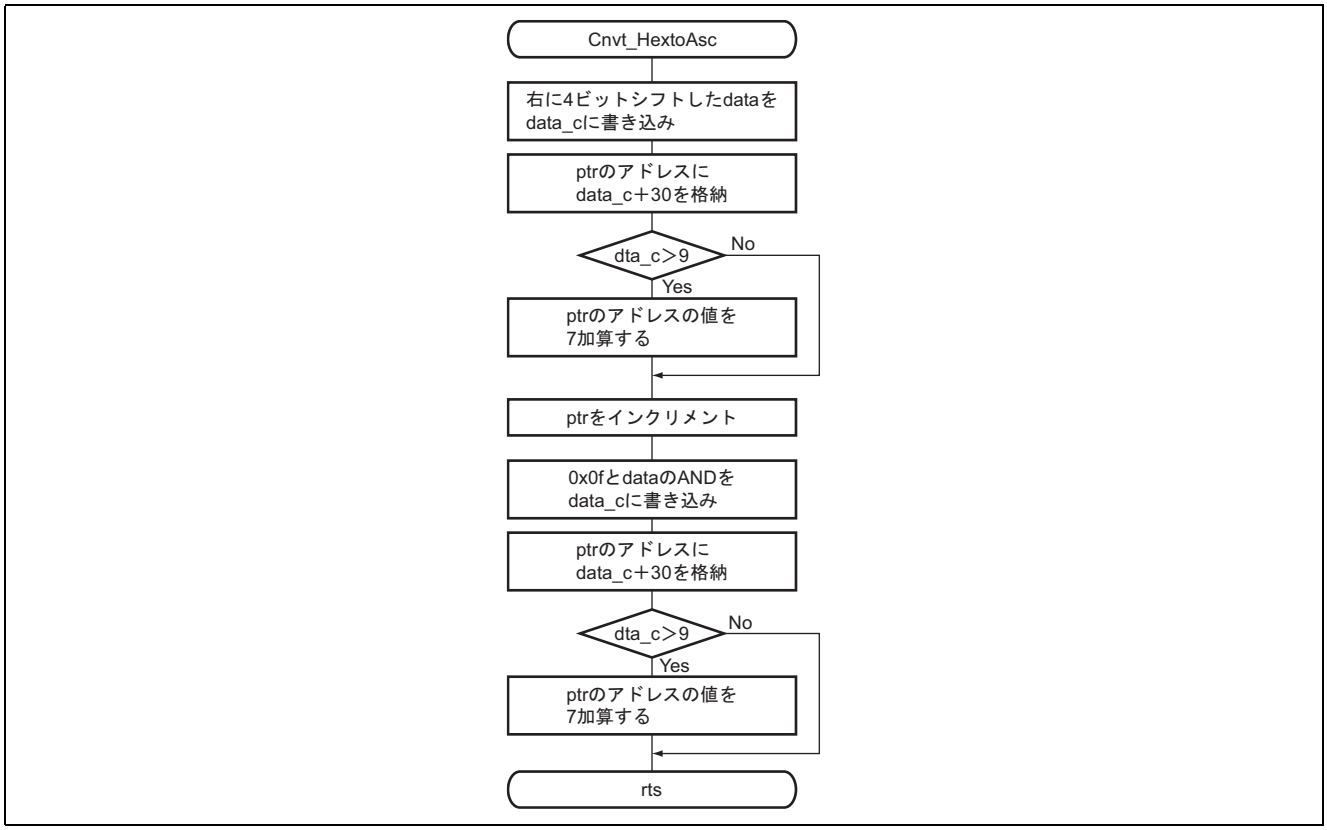

Dsp_err
ptrにs3tx_buf[0]の
アドレスをセット
ptrをインクリメントした値の
アドレスにASC_CRを格納
ptrをインクリメントした値の
アドレスに"！"を格納
ptrをインクリメントした値の
アドレスに"E"を格納
ptrをインクリメントした値の
アドレスに"r"を格納
ptrをインクリメントした値の
アドレスに"r"を格納
ptrをインクリメントした値の
アドレスにASC_CRを格納
ptrをインクリメントした値の
アドレスにASC_LFを格納
送信カウンタに1をセット
送信レングスに7をセット
TDRにs3tx_buf[0]を
書き込み
TXI、RXI、ERI割り込み処理を
許可および送受信を許可
rts

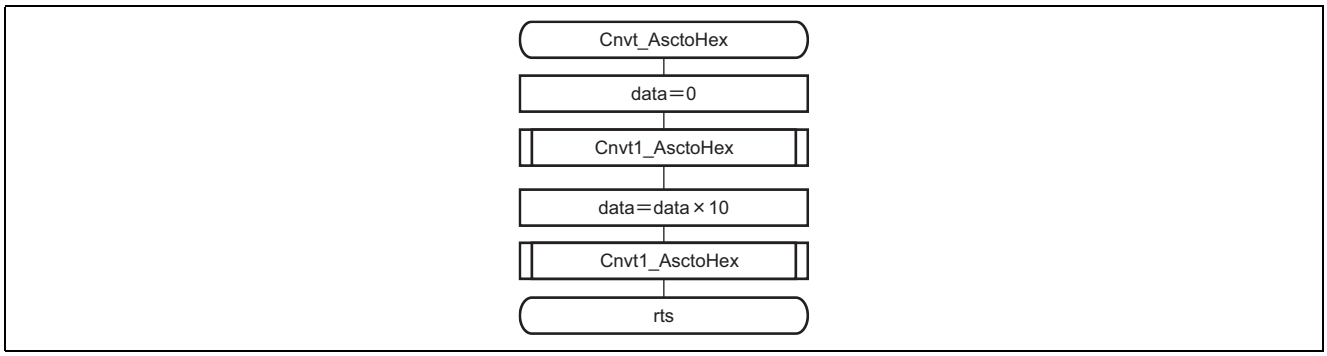

set_send1
送信カウンタに1をセット
送信レングスに1をセット
TDRにworkを書き込み
TXI、RXI、ERI割り込み処理を
許可および送受信を許可
rts

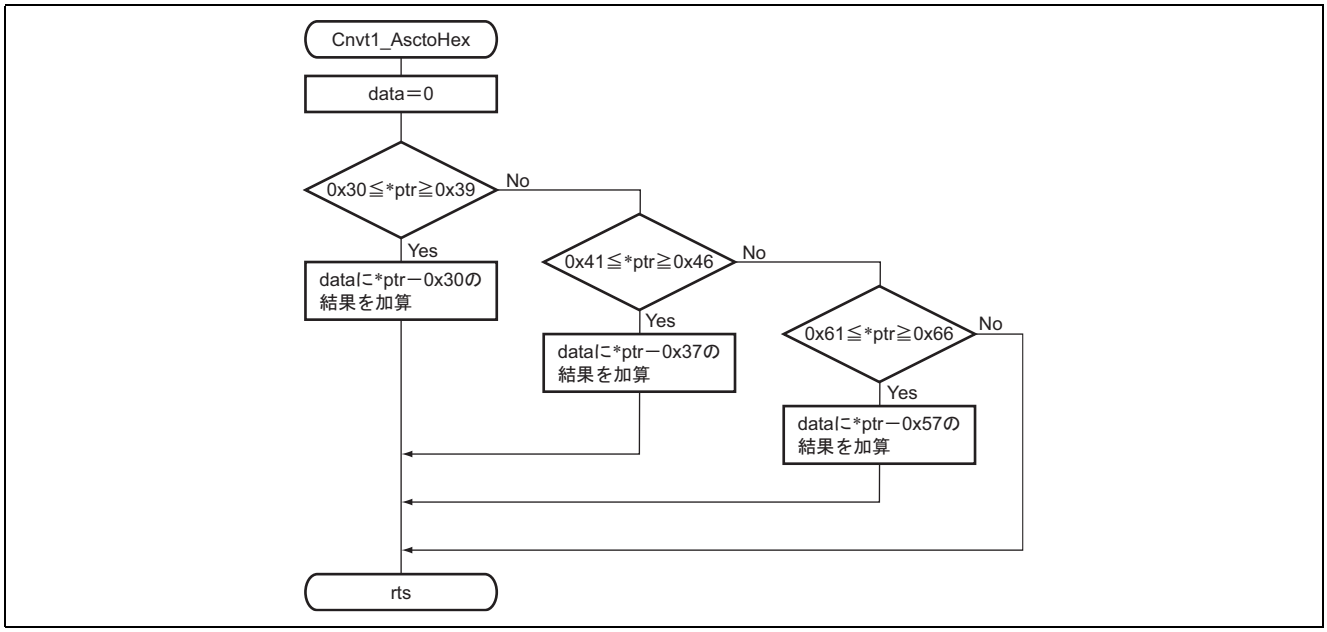

Dsp_idle
送信カウンタに1をセット
s3tx_buf[0]に" "を書き込み
s3tx_buf[1]にASC_CRを書き込み
s3tx_buf[2]にASC_LFを書き込み
送信レングスに3をセット
TDRに" "を書き込み
TXI、RXI、ERI割り込み処理を
許可および送受信を許可
rts


Dump_start
dump_adrsにdumpを書き込み
dump_ycntを0にクリア
rts


Cnvt_AsctoAddress
Cnvt_AsctoHex
Cnvt_AsctoHex
rts


Cnvt_HextoAsc
右に4ビットシフトしたdataを
data_cに書き込み
ptrのアドレスに
data_c＋30を格納
dta_c＞9
No
Yes
ptrのアドレスの値を
7加算する
ptrをインクリメント
0x0fとdataのANDを
data_cに書き込み
ptrのアドレスに
data_c＋30を格納
dta_c＞9
No
Yes
ptrのアドレスの値を
7加算する
rts


Cnvt_AsctoHex
data＝0
Cnvt1_AsctoHex
data＝data×10
Cnvt1_AsctoHex
rts


Cnvt1_AsctoHex
data＝0
0x30≦*ptr≧0x39
No
Yes
dataに*ptr－0x30の
結果を加算
0x41≦*ptr≧0x46
No
Yes
dataに*ptr－0x37の
結果を加算
0x61≦*ptr≧0x66
No
Yes
dataに*ptr－0x57の
結果を加算
rts

6. プログラムリスト

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

```
/************************************************************************************************************************/
/*                                                                                                                      */
/*   FILE           :kmoni.c                                                                                            */
/*   DATE           :Man, Jul 11, 2001                                                                                  */
/*   DESCRIPTION    :Main Program                                                                                       */
/*   CPU TYPE       :H8/3664F                                                                                           */
/*                                                                                                                      */
/*   This file is generated by Renesas Project Generator (Ver.1.2).                                                     */
/*                                                                                                                      */
/************************************************************************************************************************/
      #include  "machine.h"
      #include  "ascdefine.h"
      #include  "string.h"

#ifdef __cplusplus
extern "C" {
#endif
void abort(void);
#ifdef __cplusplus
}
#endif

struct BIT {
      unsigned char  b7:1;       /* ビット 7 */
      unsigned char  b6:1;       /* ビット 6 */
      unsigned char  b5:1;       /* ビット 5 */
      unsigned char  b4:1;       /* ビット 4 */
      unsigned char  b3:1;       /* ビット 3 */
      unsigned char  b2:1;       /* ビット 2 */
      unsigned char  b1:1;       /* ビット 1 */
      unsigned char  b0:1;       /* ビット 0 */
};

#define SCI3_SMR    (*(volatile unsigned char *)0xFFA8)                    /* SCI3  Address                    */
#define SCI3_BRR    (*(volatile unsigned char *)0xFFA9)                    /* SCI3  Address                    */
#define SCI3_SCR    (*(volatile unsigned char *)0xFFAA)                    /* SCI3  Address                    */
#define SCI3_TDR    (*(volatile unsigned char *)0xFFAB)                    /* SCI3  Address                    */
#define SCI3_SSR    (*(volatile unsigned char *)0xFFAC)                    /* SCI3  Address                    */
#define SCI3_RDR    (*(volatile unsigned char *)0xFFAD)                    /* SCI3  Address                    */
#define ssr3        (*(struct BIT *)0xFFAC)
#define SSR3_RDRF   ssr3.b6
```

```
#define SSR3_TDRE        ssr3.b7

#define PMR1        *(volatile unsigned char *)0xFFE0

#pragma         interrupt(Int_SCI3)
#pragma         interrupt(Int_trap)
#pragma         interrupt(NMI)

unsigned char  s3rx_buf[16];
unsigned char  s3tx_buf[60];
unsigned char  s3rx_cnt;
unsigned char  s3tx_cnt;
unsigned char  s3tx_len;
unsigned char  dummy;
unsigned char  trap_flag;

unsigned char  moni_code;
unsigned char  ans_code;
unsigned char  dump_ycnt;
unsigned char  *cr_adrs;
unsigned char  *dump_adrs;
union     {
     unsigned char  data[2];
     unsigned char  *adrs;
          } CMem;

/*;************************************************************************************************************************/
/*;  関数定義                                                                                                            */
/*;************************************************************************************************************************/
extern          void INIT(void);
void            main ( void );
void            Init_sci ( void );
void            Int_SCI3( void );
void            Int_s3tx ( void );
void            Int_s3rx ( void );
void            Int_s3err( void );
void            Int_trap( void );
void            Cnvt_HextoAsc(unsigned char data ,unsigned char *ptr);
unsigned char   Cnvt_AsctoHex ( unsigned char *ptr );
unsigned char   Cnvt1_AsctoHex(unsigned char *ptr);
void            Cnvt_AsctoAddress( void );

void            Ans_send( void );
void            Set_send1 ( unsigned char work );
void            Dsp_mem( void );
void            Dsp_err( void );
void            Dsp_idle( void );
void            Dsp_init( void );
void            Dump_start( unsigned char *adrs );
void            Dump_send( void );

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

```
    INIT,                                                    /* 00 リセット                    */
    INIT
};
#pragma   section        V2                                  /* VECTOR SECTOIN SET             */
void (*const VEC_TBL2[])(void) = {
    Int_trap,                                                /* 10  trap#0                     */
    Int_trap,                                                /* 12  trap#1                     */
    Int_trap,                                                /* 14  trap#2                     */
    Int_trap                                                 /* 16  trap#3                     */
};
#pragma   section        V3                                  /* VECTOR SECTOIN SET             */
void (*const VEC_TBL3[])(void) = {
    Int_SCI3,                                                /* 2E SCI3_RX  Interrupt          */
    Int_SCI3                                                 /* 2E SCI3_RX  Interrupt          */
};

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

    moni_code = ' ';
    ans_code = ' ';
    dump_ycnt = 8;

    PMR1 = 0x02;
    Init_sci();
    set_imask_ccr(0);                                        /* コンディションコードの設定     */
    Dsp_init();

    while (1) {
        ;
    }
}

void abort(void)
{

}
/*-------------------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Init_sci                                                                                                           */
/*   <関数機能>     シリアル設定                                                                                                       */
/*   <引き数 >      なし                                                                                                               */
/*   <戻り値 >      なし                                                                                                               */
/*   <外部変数>     なし                                                                                                               */
/*   <使用関数>     なし                                                                                                               */
/*-------------------------------------------------------------------------------------------------------------------------------------*/
void Init_sci ( void )   {

    unsigned long *plPtr_0;
    unsigned long *plPtr_1;
    unsigned char  speed;
```

```
/*-------------------------------------------------------------------------------------------------------------------------------*/
/*    <対UNIT通信>                                                                                                               */
/*    SCIレジスタ                 7    6    5    4    3    2    1    0                                                          */
/*    シリアルステータス  (SSR): TDRE RDRF ORER FER  PER  ---  ---  ---                                                          */
/*    シリアルモード          (SMR): C/A  8/7  PE   O/E  STP  MP   CKS1 CKS0                                                     */
/*                                0    0    0    0    0    0    0    0                                                          */
/*    ビットレート            (BRR):                                                                                             */
/* シリアルコントロール   (SCR): TIE  RIE  TE   RE   MPI  TEI  CKE1 CKE0                                                       */
/*                           1    1    1    1    0    0    0    0                                                               */
/*-------------------------------------------------------------------------------------------------------------------------------*/

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
     SCI3_SCR = 0xf0;
     SCI3_BRR = 51;                                                    /*  9600bps CLK:16MHz                                    */

     SCI3_SMR = 0x00;                                                  /* PE:0                                                  */

     s3rx_cnt = 0;

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
}
/*-------------------------------------------------------------------------------------------------------------------------------*/
/*    <関数名 >       Int_SCI3                                                                                                   */
/*    <関数機能>      シリアル割込み処理                                                                                         */
/*    <外部変数>                                                                                                                 */
/*    <使用関数>      Int_s3rx,Int_s3tx,int_s3err                                                                                */
/*-------------------------------------------------------------------------------------------------------------------------------*/
void Int_SCI3( void )    {
 unsigned char work;

     work = SCI3_SSR;
     if ((work & 0x38) != 0)  {
          Int_s3err();
     }
     else {
          if ((work & 0x40) != 0)  {
               Int_s3rx();
          }
          if ((work & 0x80) != 0)  {
               Int_s3tx();
          }
     }
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Int_s3tx                                                                                              */
/*   <関数機能>     シリアル送信処理                                                                                      */
/*   <外部変数>     s1tx_cnt,s1tx_len                                                                                     */
/*   <使用関数>     Ans_send                                                                                              */
/*------------------------------------------------------------------------------------------------------------------------*/
void Int_s3tx( void )    {

     if (s3tx_cnt < s3tx_len) {
          SCI3_TDR = s3tx_buf[s3tx_cnt++];
          SSR3_TDRE = 0;
     }
     else {
          SCI3_SCR = 0x70;         /* TEIE:0 */
          Ans_send();
     }
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Int_s3rx                                                                                              */
/*   <関数機能>     シリアル受信処理                                                                                      */
/*   <外部変数>     s3rx_buf,s3rx_cnt,                                                                                    */
/*   <使用関数>     Dsp_idle,Set_send1,Cnvt_AsctoAddress,Dump_start,                                                      */
/*                  Cnvt_AsctoHex                                                                                         */
/*------------------------------------------------------------------------------------------------------------------------*/
/*------------------------------------------------------------------------------------------------------------------------*/
/*                        1   2   3   4   5   6   7   8   9   10  11  12                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
/*   memory Dump    : D | x   x   x   x   CR                                                                             */
/*                  : CR                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
/*   memory Edit    : M | x   x   x   x   CR                                                                             */
/*                  : CR                                                                                                  */
/*                  : ^                                                                                                   */
/*                  : x   x     CR                                                                                        */
/*------------------------------------------------------------------------------------------------------------------------*/
void Int_s3rx( void )    {
 unsigned char work,ed_data;

     work = SCI3_RDR;

     if (work == ASC_CN) {
          s3rx_cnt = 0;
          Dsp_idle();
     }
     else if (work == ASC_BS) {
          if (s3rx_cnt > 0)   {
               s3rx_cnt--;
               Set_send1(work);
          }
     }
     else if (s3rx_cnt < 13) {
          s3rx_buf[s3rx_cnt++] = work;

          if (work == ASC_CR) {
               if (s3rx_cnt == 6)  {
                    Dsp_idle();
                    Cnvt_AsctoAddress();
```

```
            cr_adrs = (unsigned char *)CMem.adrs;
            work = s3rx_buf[0];
            if (work == 'D' || work == 'd') {
                moni_code = 'D';
                Dump_start(cr_adrs);
                ans_code = 'D';
            }
            else if (work == 'E' || work == 'e') {
                moni_code = 'E';
                ans_code = 'E';
            }
        s3rx_cnt = 0;
        }
        else {
            switch (moni_code) {
                case 'D':                                               /*--- Dump---*/
                    if (s3rx_cnt == 1) {
                        cr_adrs+=128;
                        ans_code = 'D';
                        Dump_start(cr_adrs);
                    }
                    else {
                        ans_code = 'R';
                    }
                    break;
                case 'E':                                               /*--- Edit ---*/
                    Dsp_idle();
                    if (s3rx_cnt == 1) {
                        cr_adrs++;
                            ans_code = 'E';
                    }
                    else if (s3rx_cnt == 3) {
                        ed_data = Cnvt_AsctoHex(s3rx_buf);
                        *cr_adrs = ed_data;
                        if (*cr_adrs == ed_data) {
                            ans_code = 'E';
                        }
                        else {
                            ans_code = 'R';
                        }
                    }
                    else {
                            ans_code = 'R';
                    }
                    break;
                default:                                                /*--- idle ---*/
                    break;
            }
            s3rx_cnt = 0;
        }
    }
    else if (work == '^')    {
        if (moni_code == 'E') {
            Dsp_idle();
            cr_adrs--;
            ans_code = 'E';
        }
```

```
                else {
                    moni_code = ' ';
                }
                s3rx_cnt = 0;
            }
            else if (work == '.')    {
                s3rx_cnt = 0;
                ans_code = 'I';
                moni_code = ' ';
                Set_send1(work);
            }
            else if (work == ASC_LF) {
                s3rx_cnt = 0;
                Set_send1(work);
            }
            else {
                Set_send1(work);
            }
        }
        else {
            s3rx_cnt = 0;
        }

        SSR3_RDRF = 0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   Set_send                                                                                                             */
/*------------------------------------------------------------------------------------------------------------------------*/
void Set_send1 ( unsigned char work ) {

        s3tx_cnt = 1;
        s3tx_len = 1;
        SCI3_TDR = work;
        SCI3_SCR = 0xf0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Ans_send                                                                                              */
/*   <関数機能>     返答データ送信                                                                                        */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                            */
/*   <使用関数>     Dump_send,Dsp_mem,Dsp_mcng,Dsp_err,Dsp_idle                                                           */
/*------------------------------------------------------------------------------------------------------------------------*/
void Ans_send ( void )   {

        switch (ans_code) {
            case 'D':                                                   /*--- Dump ---*/
                if (dump_ycnt < 8)  {
                    Dump_send();
                    dump_ycnt++;
                }
                else {
                    ans_code = ' ';
                }
                break;
            case 'E':                                                   /*--- Edit ---*/
                Dsp_mem();
```

```
                ans_code = ' ';
                break;
            case 'R':                                                   /*--- Err ---*/
                Dsp_err();
                ans_code = ' ';
                break;
            case 'I':                                                   /*--- Idle ---*/
                Dsp_idle();
                ans_code = ' ';
                break;
            default:                                                    /*---  ---*/
                break;
        }
    }
/*------------------------------------------------------------------------------------------------------------------------*/
/*    <関数名 >      Dsp_mem                                                                                              */
/*    <関数機能>     メモリ                                                                                               */
/*    <引き数 >      なし                                                                                                 */
/*    <戻り値 >      なし                                                                                                 */
/*    <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                           */
/*    <使用関数>     Cnvt_HextoAsc                                                                                        */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_mem( void )     {
 unsigned char *ptr,adrs_h,adrs_l;
 unsigned shortadrs;

     ptr = &s3tx_buf[0];
     *ptr++ = ASC_CR;
     adrs = (unsigned short)cr_adrs;
     adrs_h = (unsigned char)(adrs >> 8);
     adrs_l = (unsigned char)(adrs & 0x00ff);
     Cnvt_HextoAsc(adrs_h,ptr);
     ptr+=2;
     Cnvt_HextoAsc(adrs_l,ptr);
     ptr+=2;
     *ptr++ = ':';
     Cnvt_HextoAsc(*cr_adrs,ptr);
     ptr+=2;
     *ptr++ = '>';

     s3tx_cnt = 1;
     s3tx_len = 9;
     SCI3_TDR = s3tx_buf[0];
     SCI3_SCR = 0xf0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Dsp_idle                                                                                              */
/*   <関数機能>     エラー表示                                                                                            */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                            */
/* <使用関数>       なし                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_idle( void )    {

          s3tx_cnt = 1;
          s3tx_buf[0] = ' ';
          s3tx_buf[1] = ASC_CR;
          s3tx_buf[2] = ASC_LF;
          s3tx_len = 3;
          SCI3_TDR = ' ';
          SCI3_SCR = 0xf0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Dsp_init                                                                                              */
/*   <関数機能>     初期画面表示                                                                                          */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                            */
/*   <使用関数>     なし                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_init( void )    {

     s3tx_cnt = 0;
     strcpy((char *)s3tx_buf,"H8/3664 Monitor Program Ver1.0¥n¥r");
     s3tx_len = 33;
     SCI3_TDR = ' ';
     SCI3_SCR = 0xf0;
}

/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Dsp_err                                                                                               */
/*   <関数機能>     エラー表示                                                                                            */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                            */
/*   <使用関数>     なし                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dsp_err( void )     {
 unsigned char *ptr;

     ptr = &s3tx_buf[0];
     *ptr++ = ASC_CR;
     *ptr++ = '!';
     *ptr++ = 'E';
     *ptr++ = 'r';
     *ptr++ = 'r';
     *ptr++ = ASC_CR;
     *ptr++ = ASC_LF;

     s3rx_cnt = 0;
     s3tx_cnt = 1;
```

```
     s3tx_len = 7;
     SCI3_TDR = s3tx_buf[0];
     SCI3_SCR = 0xf0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Dump_start                                                                                            */
/*   <関数機能>     メモリダンプ開始                                                                                      */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s0tx_cnt,s0tx_len,s0tx_buf                                                                            */
/*   <使用関数>     なし                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dump_start( unsigned char *adrs ) {
     dump_adrs = adrs;
     dump_ycnt = 0;
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Dump_send                                                                                             */
/*   <関数機能>     メモリダンプデータ送信                                                                                */
/*   <引き数 >      なし                                                                                                  */
/*   <戻り値 >      なし                                                                                                  */
/*   <外部変数>     s3tx_cnt,s3x_len,s3tx_buf                                                                             */
/*   <使用関数>     Cnvt_HextoAsc                                                                                         */
/*------------------------------------------------------------------------------------------------------------------------*/
/*------------------------------------------------------------------------------------------------------------------------*/
/*         1        2        3        4        5                                                                        */
/*   12345678901234567890123456789012345678901234567890123                                                               */
/*   0000 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F                                                                */
/*------------------------------------------------------------------------------------------------------------------------*/
void Dump_send( void )   {
 unsigned char *ptr,adrs_h,adrs_l,cnt;
 unsigned shortadrs;

          ptr = &s3tx_buf[0];
          *ptr++ = ASC_CR;
          adrs = (unsigned short)dump_adrs;
          adrs_h = (unsigned char)(adrs >> 8);
          adrs_l = (unsigned char)(adrs & 0x00ff);
          Cnvt_HextoAsc(adrs_h,ptr);
          ptr+=2;
          Cnvt_HextoAsc(adrs_l,ptr);
          ptr+=2;
          *ptr++ = ' ';
          cnt = 0;
          while (cnt <16)     {
               Cnvt_HextoAsc(*dump_adrs,ptr);
               ptr+=2;
               dump_adrs++;
               cnt++;
               *ptr++ = ' ';
          }
          *ptr++ = ASC_CR;
          *ptr++ = ASC_LF;

          s3tx_cnt = 1;
          s3tx_len = 56;
          SCI3_TDR = s3tx_buf[0];
          SCI3_SCR = 0xf0;
}
```

```
/*-------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Cnvt_HextoAsc(0x30～0x39,0x41～0x46)                                                                  */
/*   <関数機能>     hexdata(1byte)ー>ASCIIdata(2byte)変換処理                                                              */
/*   <引き数 >      hexdata,(unsigned char *)ascii_data                                                                    */
/*   <戻り値 >      なし                                                                                                   */
/*   <外部変数>     なし                                                                                                   */
/*   <使用関数>     なし                                                                                                   */
/*-------------------------------------------------------------------------------------------------------------------------*/
void Cnvt_HextoAsc(unsigned char data ,unsigned char *ptr) {
 unsigned char data_c;

     data_c = data >> 4;
     *ptr = data_c + 0x30;
     if (data_c > 9)
          *ptr += 7;

     ptr++;
     data_c = data & 0x0f;
     *ptr = data_c + 0x30;
     if (data_c > 9)
          *ptr += 7;
}
/*-------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Int_s3err                                                                                              */
/*   <関数機能>     シリアル 0 受信エラー処理                                                                              */
/*   <外部変数>     s3rx_cnt                                                                                               */
/*   <使用関数>     なし                                                                                                   */
/*-------------------------------------------------------------------------------------------------------------------------*/
void Int_s3err( void )   {

     dummy = SCI3_SSR;
     SCI3_SSR = 0;
     s3rx_cnt = 0;
}
/*-------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Int_trap                                                                                               */
/*   <関数機能>     トラップ割込み処理                                                                                     */
/*   <外部変数>     trap_flag                                                                                              */
/*   <使用関数>     なし                                                                                                   */
/*-------------------------------------------------------------------------------------------------------------------------*/
void Int_trap( void )    {

     while (trap_flag == 0)
          ;
     trap_flag = 0;
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Cnvt_AsctoHex                                                                                         */
/*   <関数機能>     2byte ascii code => 1byte (7-4bit,3-0bit)                                                             */
/*   <引き数 >      アスキーコード列のポインタ                                                                            */
/*   <戻り値 >      1byte(Hex)                                                                                            */
/*   <外部変数>     なし                                                                                                  */
/*   <使用関数>     Cnvt1_AsctoHex                                                                                        */
/*------------------------------------------------------------------------------------------------------------------------*/
unsigned char Cnvt_AsctoHex(unsigned char *ptr) {
     unsigned char  data = 0;

     data = Cnvt1_AsctoHex(ptr++);
     data *= 0x10;
     data += Cnvt1_AsctoHex(ptr);

     return(data);
}

/*------------------------------------------------------------------------------------------------------------------------*/
/*   <関数名 >      Cnvt1_AsctoHex                                                                                        */
/*   <関数機能>     1byte ascii code => 1byte(bit3-0)                                                                     */
/*   <引き数 >      アスキーコード列のポインタ                                                                            */
/*   <戻り値 >      1byte(bit3-0)                                                                                         */
/*   <外部変数>     なし                                                                                                  */
/*   <使用関数>     なし                                                                                                  */
/*------------------------------------------------------------------------------------------------------------------------*/
/*   ascii     0   -   9      A   B   C   D   E   F                                                                      */
/*   hex       30  -  39     41  42  43  44  45  46                                                                      */
/*------------------------------------------------------------------------------------------------------------------------*/
unsigned char Cnvt1_AsctoHex(unsigned char *ptr) {
     unsigned char  data = 0;

     if ((0x30 <= *ptr) && (*ptr <= 0x39)) {
          data += *ptr - 0x30;
     }
     else if ((0x41 <= *ptr) && (*ptr <= 0x46)) {
          data += *ptr - 0x37;
     }
     else if ((0x61 <= *ptr) && (*ptr <= 0x66)) {
          data += *ptr - 0x57;
     }
return(data);
}
```

```
/*------------------------------------------------------------------------------------------------------------------------*/
/*    <関数名 >       Cnvt_AsctoAddress(0x30～0x39,0x41～0x46)                                                          */
/*    <関数機能>      ASCIIdata(4byte)->参照アドレス(2byte)変換処理                                                     */
/*    <引き数 >       なし                                                                                              */
/*    <戻り値 >       なし                                                                                              */
/*    <外部変数>      CMemAddress,s3rx_buf                                                                              */
/*    <使用関数>      Cnvt_AsctoHex                                                                                     */
/*------------------------------------------------------------------------------------------------------------------------*/
void Cnvt_AsctoAddress( void )
{
                    /* 参照アドレス部を変換(2バイトずつ)                                                                 */
                    /* buf[1][2][3][4]                                                                                   */
                    /*    -  -  -  -   x  x  x  x                                                                        */
     CMem.data[0] = Cnvt_AsctoHex(&s3rx_buf[1]);
     CMem.data[1] = Cnvt_AsctoHex(&s3rx_buf[3]);
}
/*------------------------------------------------------------------------------------------------------------------------*/
/*    <関数名 >       NMI                                                                                               */
/*    <関数機能>      NMI割込み                                                                                         */
/*    <引き数 >       なし                                                                                              */
/*    <戻り値 >       なし                                                                                              */
/*    <外部変数>      なし                                                                                              */
/*    <使用関数>      なし                                                                                              */
/*------------------------------------------------------------------------------------------------------------------------*/
void NMI(void) {

}
/*------------------------------------------------------------------------------------------------------------------------*/
/*    <関数名 >       Int_s3te                                                                                          */
/*    <関数機能>      SCI1TEI 割込み                                                                                    */
/*    <引き数 >       なし                                                                                              */
/*    <戻り値 >       なし                                                                                              */
/*    <外部変数>      なし                                                                                              */
/*    <使用関数>      なし                                                                                              */
/*------------------------------------------------------------------------------------------------------------------------*/
void Int_s3te(void) {

}
```

Section 設定

アドレス	セクション名
H'0000	CV1
H'0010	CV2
H'002E	CV3
H'0100	P
	C$DSEC
	C$BSEC
	D
	C
H'FB80	B
	R

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.24	—	初版発行

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

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