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。

アプリケーション・ノート

RENESAS

CC78Kシリーズ Cコンパイラ

78K/Ⅲシリーズ用

資料番号　EEA-608（第1版）
発行年月　October 1989 P

はじめに

対 象 者　このアプリケーション・ノートは，μPD78320の機能を理解し，それを用いたC言語のアプリケーション・プログラムを設計するエンジニアを対象とします．

目　　的　このアプリケーション・ノートは，C言語によるプログラミングについて，μPD78320のUART機能を用いてユーザに理解していただくことを目的とします．

構　　成　このアプリケーション・ノートは，大別して次の内容で構成されています．

- 概　説
- ハードウエア編
- ソフトウエア編
- 開発方法
- Cコンパイラの注意事項

読 み 方　このアプリケーション・ノートの読者には，電気，論理回路，マイクロコンピュータおよびC言語に関する一般知識を必要とします．また，このアプリケーション・ノートでは，μPD78320を代表品種として記述してあります．
μPD78320を対象デバイスと読み替えてご利用ください．

凡　　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称の上に上線）
注	：本文中につけた注の説明
注意	：気を付けて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	：
	2進数 … ××××または××××B
	10進数 … ××××
	16進数 … ××××H

関連資料　○Cコンパイラに関する資料

・ユーザーズ・マニュアル　言語編／操作編(EEU-655/EEU-656)

○アセンブラ，リンカに関する資料

・ユーザーズ・マニュアル(EEU-662)

○μPD78320に関する資料

・ユーザーズ・マニュアル(EEU-619)

目次要約

目　次

図の目次

表の目次

第1章　概　要

μPD78320は，μCOM-78K/Ⅲシリーズに属する16/8ビット・シングルチップ・マイクロコンピュータです．

次のようなハードウエアを内蔵しています．

・ＲＡＭ：６４０バイト
・リアルタイム・パルス・ユニット
・２チャネル・シリアル・インタフェース
・１０ビット分解能Ａ／Ｄコンバータ
・ウォッチドッグ・タイマ
・割り込みコントローラ

このアプリケーション・ノートでは，μPD78320を，Ｃ言語で開発する際に参考にしていただくために，作成しました．

今回は，割り込み機能を用いて，アシンクロナス・シリアル・インタフェースを行うプログラムを，Ｃ言語で作成しています．

本書では，そのプログラムについて，次の構成で説明します．

(1)ハードウエア編

メモリ・マップ，EBボードの変更箇所の説明，使用レジスタおよびレジスタ設定例について説明します．

(2)ソフトウエア編

モジュール，変数について説明します．

モジュールの説明では，処理内容，使用サブルーチン，パラメータについて説明し，フロー・チャートを添付しています．

(3)開発方法

オブジェクト・モジュール・ファイル，ロード・モジュール・ファイル，HEX形式オブジェクト・モジュール・ファイルの作成について，それぞれ具体例で説明します．また，EBボードへのダウン・ロードについても説明します．

(4)Ｃコンパイラの注意事項

Ｃソース・プログラム作成前，Ｃソース・プログラム作成時，ロード・モジュール・ファイル作成時それぞれの注意事項を示します．

(5)プログラム・リスト

1.1 ハードウエア編の見方

ここでは，今回作成したプログラムを動作させる環境について記述しています．今回は，EBボード(EB-78320-98)を用いていますが，このEBボードを多少変更して使用しています．

また，使用した特殊機能レジスタの一覧と，その設定例を記述しています．

1.2 ソフトウエア編の見方

ここでは，今回作成したプログラムについて，モジュール別に説明しています．項目は以下のとおりです．

- ・処理内容　　：そのモジュールの動作内容を記述しています．
- ・使用サブルーチン：そのモジュールで使用しているサブルーチンを記述しています．
- ・パラメータ　　：そのモジュールの入出力パラメータを記述しています．
- ・備考　　：そのモジュールについての補足説明をしています．

1.3 開発方法の見方

ここでは，今回作成したプログラムをEBボード上で動作させるために，HEX形式オブジェクト・モジュール・ファイルにするまでの手順について記述しています．

ツールとして，以下のものが必要です．

- ・Cコンパイラ　　：CC78K3
- ・アセンブラ　　：RA78K3
- ・リンカ　　：LK78K3
- ・オブジェクト・コンバータ：OC78K3

1.4 Cコンパイラの注意事項の見方

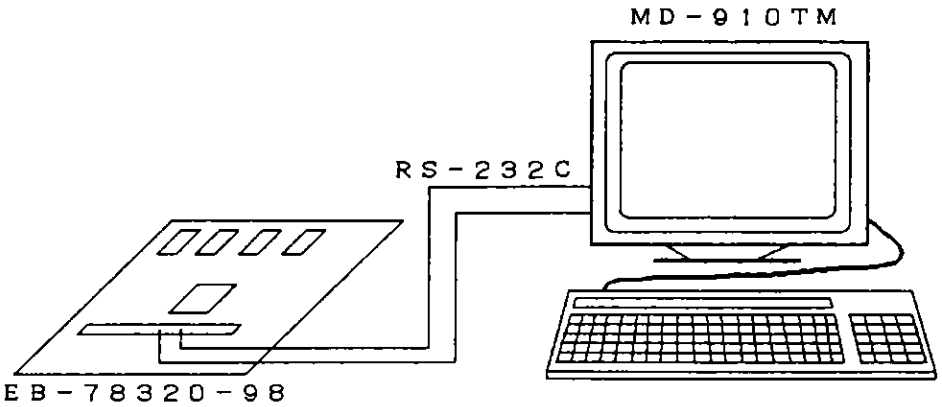
ここでは，C言語でプログラムを作成するにときの注意事項について記述しています．

第２章　ハードウエア編

本章では，今回使用したハードウエアについて説明します．

今回は，ターゲットとしてEBボード(EB-78320-98)，ターミナルとしてMD-910TMを使用し，双方をRS-232Cを介して接続しています．図2-1にターゲット・システムの構成を示します．

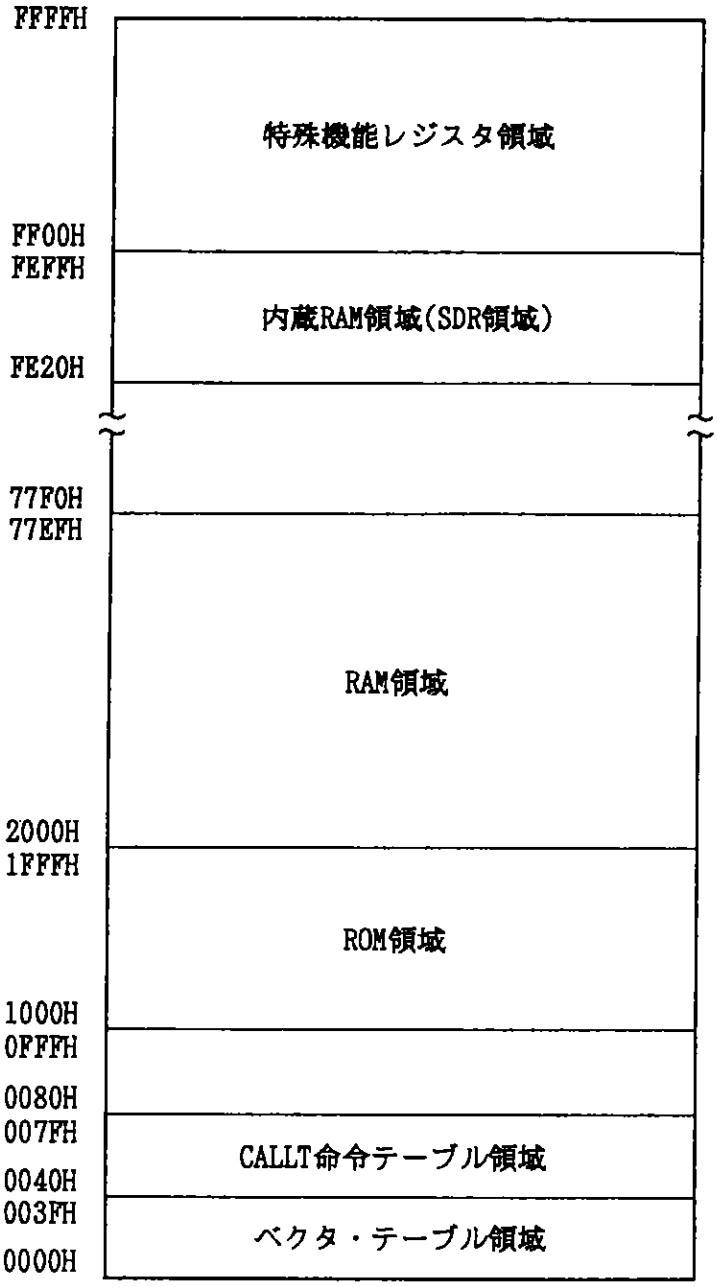
図2-1　ターゲット・システムの構成



2.1 メモリ・マップ

EBボードおよび，今回作成したプログラムについて，図2-2にメモリ・マップを示します．

図2-2　メモリ・マップ



2.2 EBボード変更

EBボードは，通信用のデバイスとしてμPD71051を搭載していますが，今回はμPD78320内部のUARTを使用します．そのため，RS-232Cと接続できるようにするために，EBボードとRS-232Cの間にMAX232Cを接続し，通信が可能となるようにします．図2-3，図2-4に変更部分について示します．

図2-3　EBボード変更部分

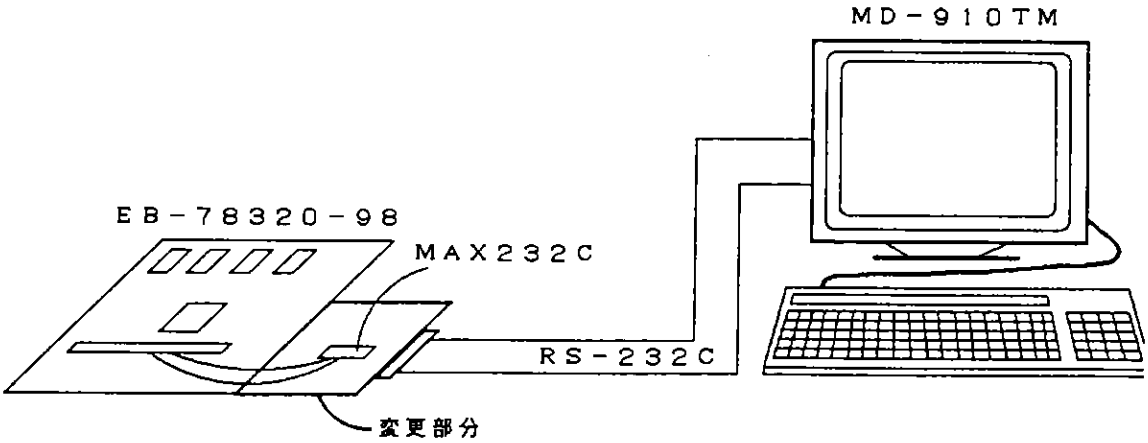
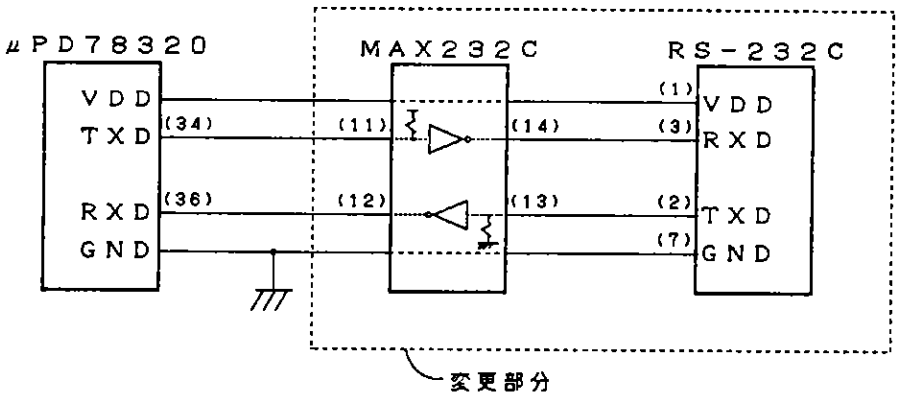


図2-4　接続方法



注意　（）内はピン番号．

さらに，EBボードには，16MHzのクロック・モジュールが搭載されていますが，UARTのボー・レートの関係から，今回は，10MHzのクロック・モジュールを使用します．

2.3 特殊機能レジスタ

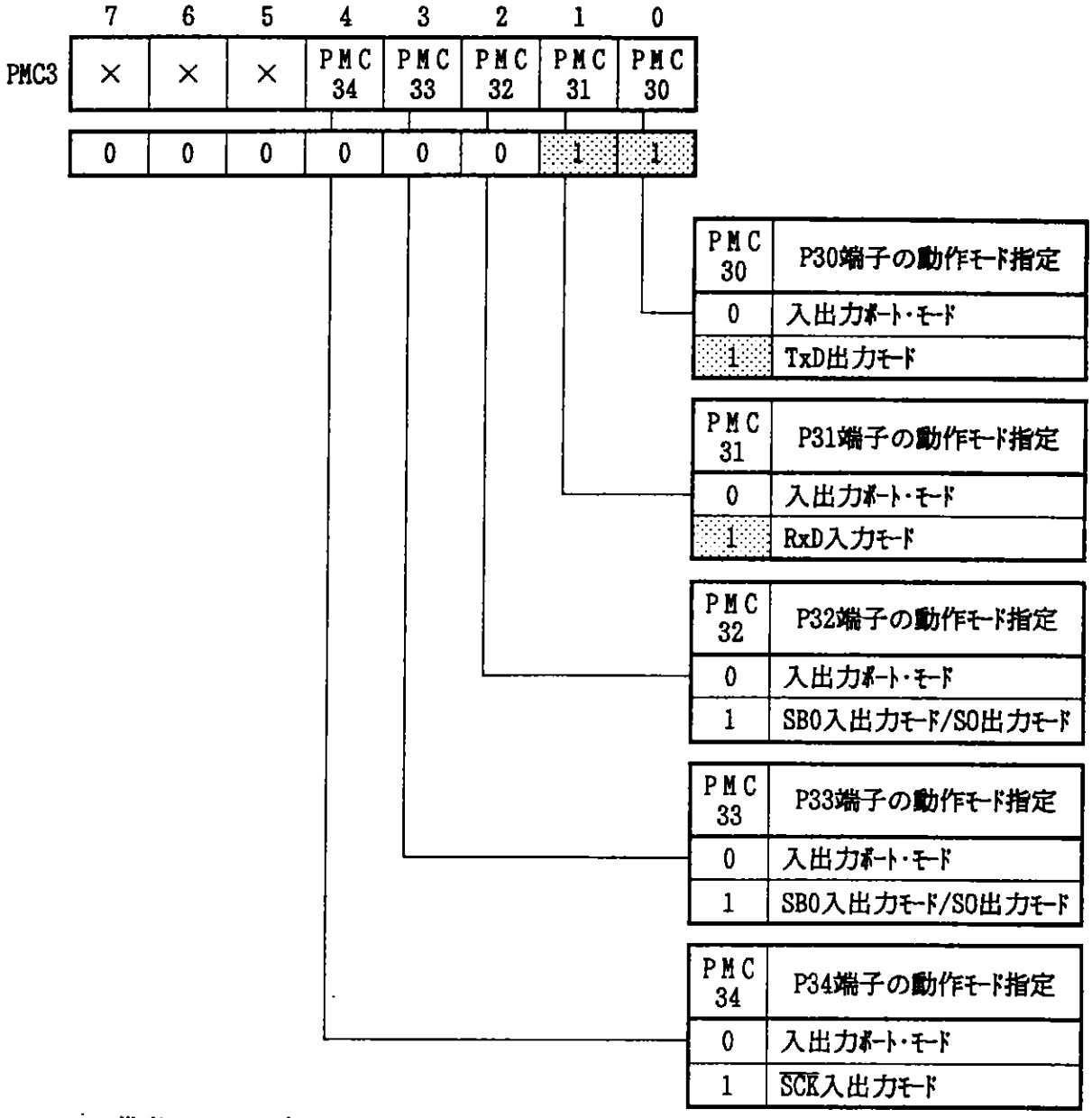
本節では，今回のプログラムで使用した特殊機能レジスタについて説明します．表2-1に，特殊機能レジスタのアドレスを示します．

表2-1　特殊機能レジスタ・アドレス一覧

ｱﾄﾞﾚｽ	特殊機能レジスタの名称	略　号
FF43H	ポート３モード・コントロール・レジスタ	PMC3
FF4CH ------ FF4DH	ボー・レート・ジェネレータ・レジスタ	BRG
FF88H	アシンクロナス・シリアル・インタフェース・モード・レジスタ	ASIM
FF8AH	アシンクロナス･シリアル･インタフェース･ステータス･レジスタ	ASIS
FF8CH	シリアル受信バッファ：UART	RXB
FF8EH	シリアル送信シフト・レジスタ：UART	TXS
FFB1H	ボー・レート・ジェネレータ・モード・レジスタ	BRGM
FFE2H	割り込み要求フラグ・レジスタ　1L	IF1L
FFE5H	割り込みマスク・フラグ・レジスタ　0H	MK0H
FFE6H	割り込みマスク・フラグ・レジスタ　1L	MK1L
FFEDH	割り込み処理モード指定レジスタ　0H	ISM0H
FFF1H	コンテキスト・スイッチング許可フラグ・レジスタ　0H	CSE0H

以下に，各レジスタのフォーマットを示します．各レジスタについての詳細は，「μPD78320　ユーザーズ・マニュアル」(EEU-619)を参照してください．

図2-5　ポート３モード・コントロール・レジスタのフォーマット



備考　×：don't　care

図2-6　アシンクロナス・シリアル・インタフェース・モード・レジスタのフォーマット

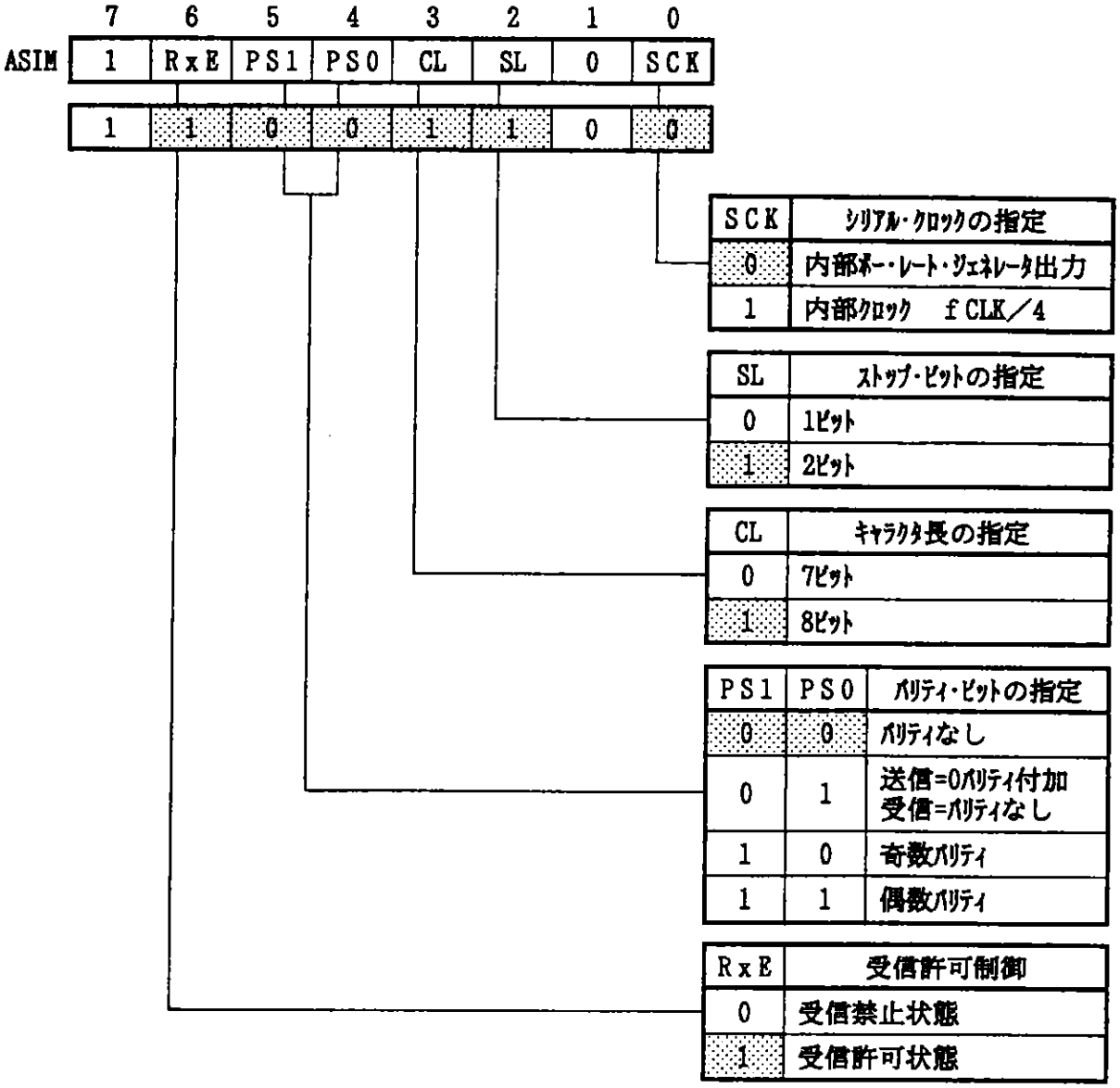


図2-7　アシンクロナス・シリアル・インタフェース・ステータス・レジスタのフォーマット

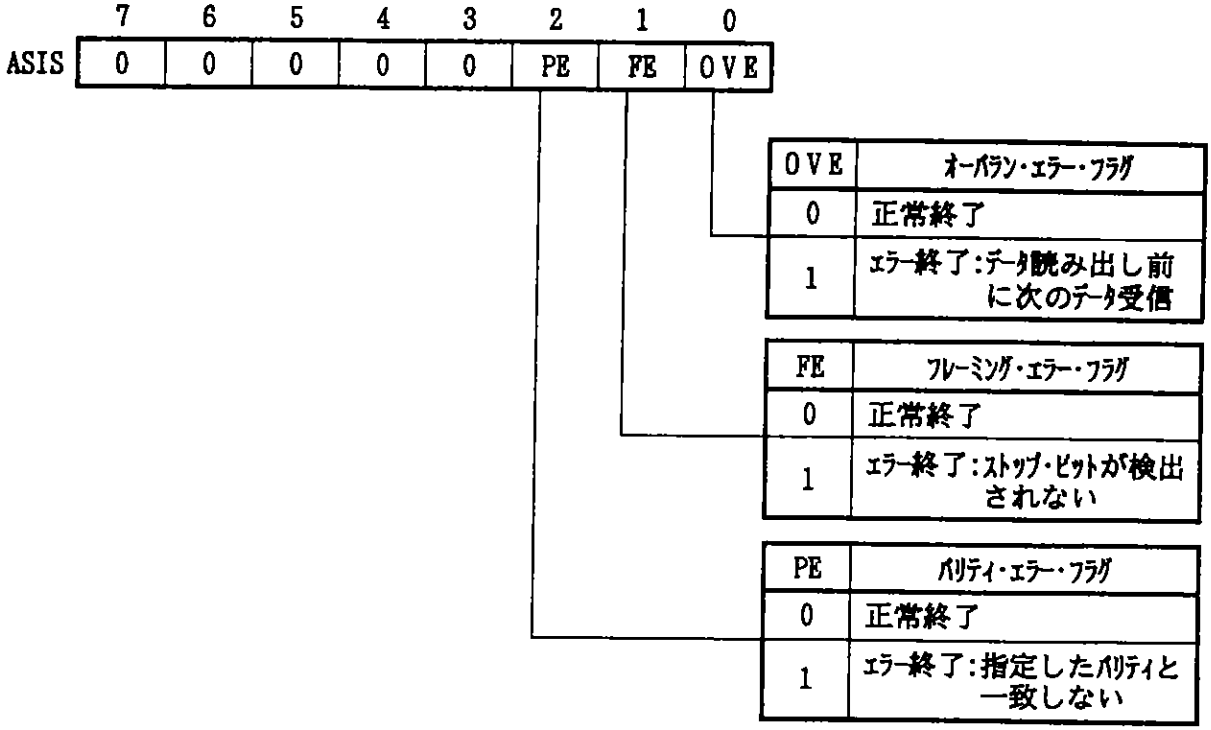


図2-8　割り込み要求フラグ・レジスタのフォーマット

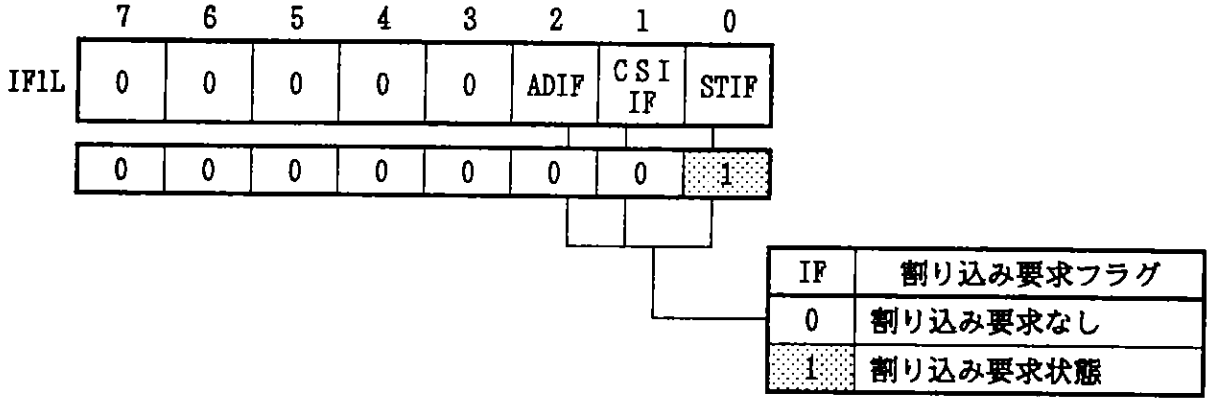


図2-9　割り込みマスク・フラグ・レジスタのフォーマット(MK0H)

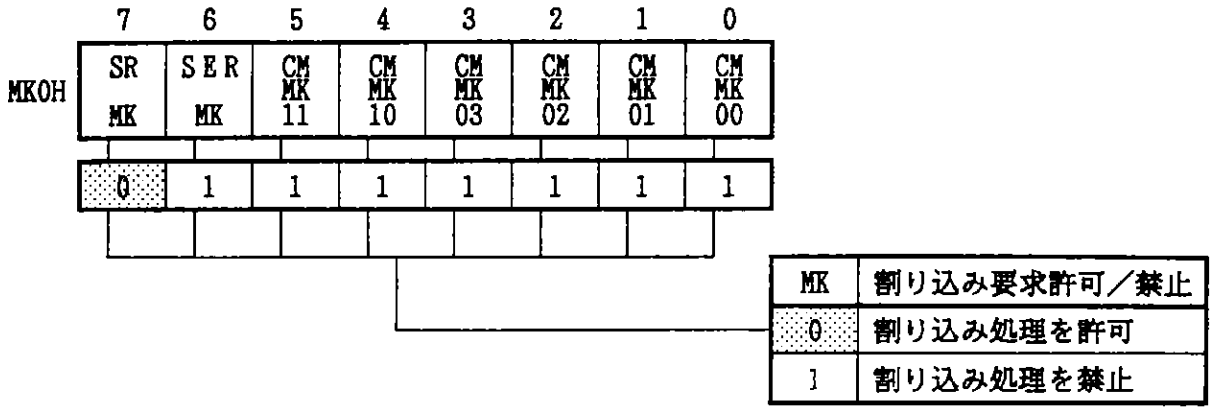


図2-10　割り込みマスク・フラグ・レジスタのフォーマット(MK1L)

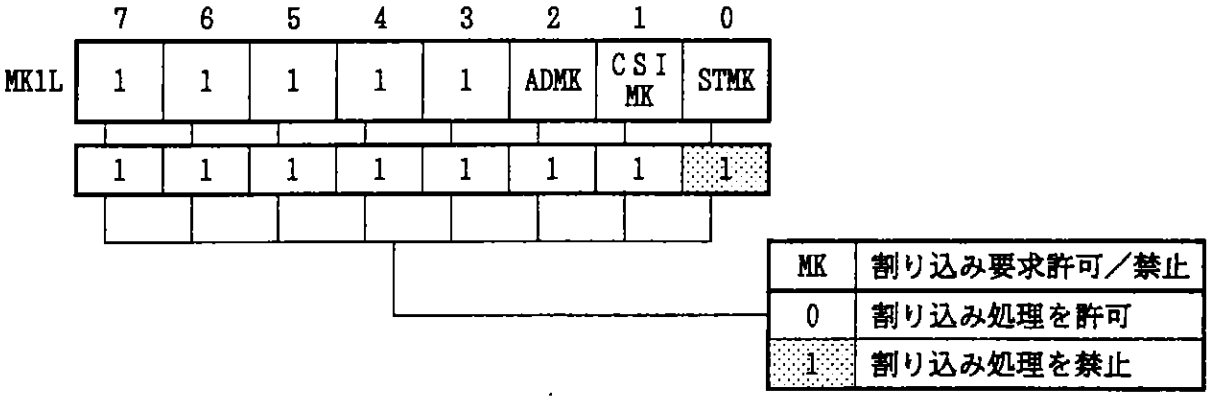
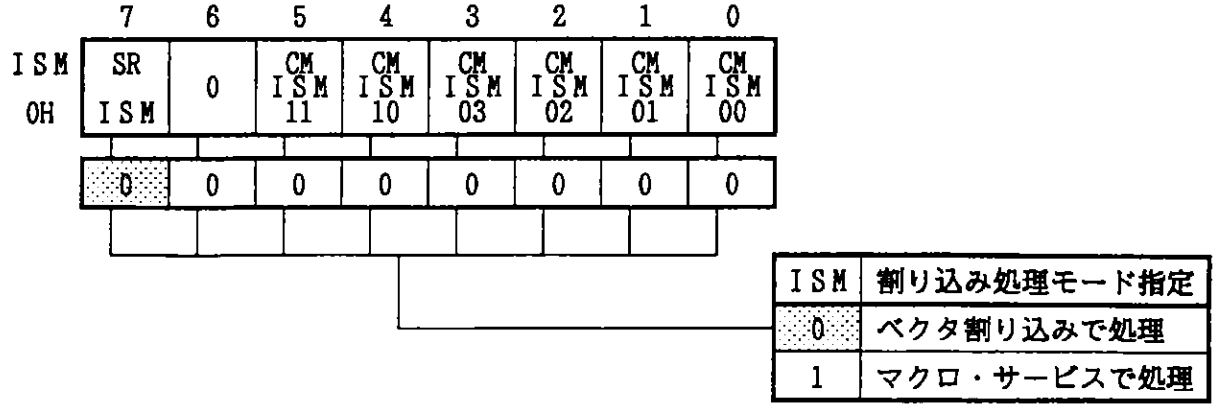
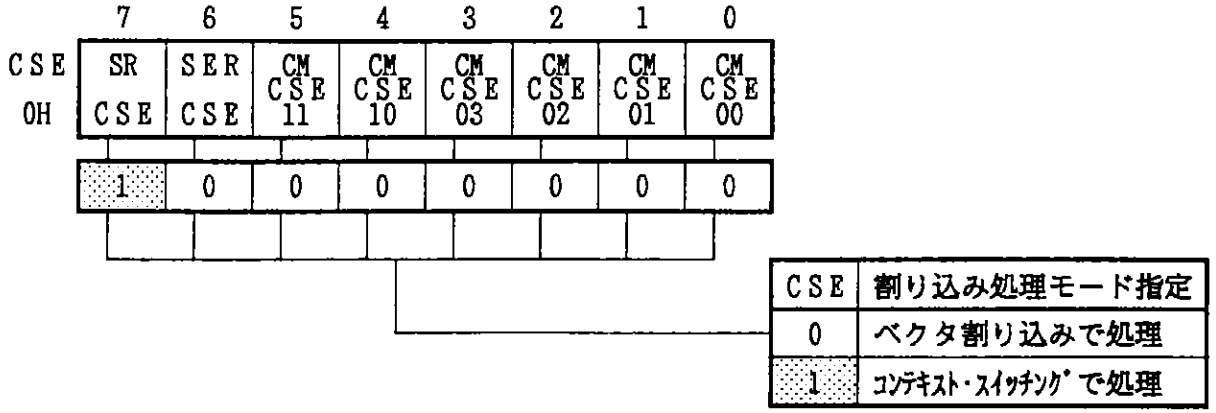


図2-11　割り込み処理モード指定レジスタのフォーマット



ISM	割り込み処理モード指定
0	ベクタ割り込みで処理
1	マクロ・サービスで処理

図2-12　コンテキスト・スイッチング許可フラグ・レジスタのフォーマット



CSE	割り込み処理モード指定
0	ベクタ割り込みで処理
1	ｺﾝﾃｷｽﾄ･ｽｲｯﾁﾝｸﾞで処理

第３章　ソフトウエア編

本章では，今回作成した送受信処理のプログラムについて，モジュール，変数について説明します．

3.1 ﾓｼﾞｭｰﾙ説明

本節では，今回作成した送受信処理のプログラムのモジュールについて説明します．
表3-1に，各モジュールの概要を示します．

表3-1　各モジュールの処理概要

モジュール名	処　理　概　要
メイン処理	送受信処理を行うために必要な変数，バッファの初期化，UARTのイニシャライズ関数の呼び出し，送受信関数の呼び出しを行う．
UARTｲﾆｼｬﾗｲｽﾞ処理	UARTを使用する際に必要となるイニシャライズを，メイン処理からのパラメータを基に行う．
受信処理	ターミナル(MD-910TM)よりデータを受信したときに，その受信したデータをメイン処理で指定したアドレスへ格納する．
送信処理	メイン処理で指定したデータを，ターミナル(MD-910TM)へ出力する．送信中に送信制御コードの受信があった場合は，その処理をする．
受信完了割り込みﾊﾝﾄﾞﾗ	ターミナル(MD-910TM)からデータを受信したときに，受信完了割り込みが発生し，起動する． 受信したデータを，受信バッファに格納する．ただし，受信したデータが，送信を制御するコードであった場合には，制御に必要な処理に移る（受信バッファには格納しない）．

3.1.1 メイン処理

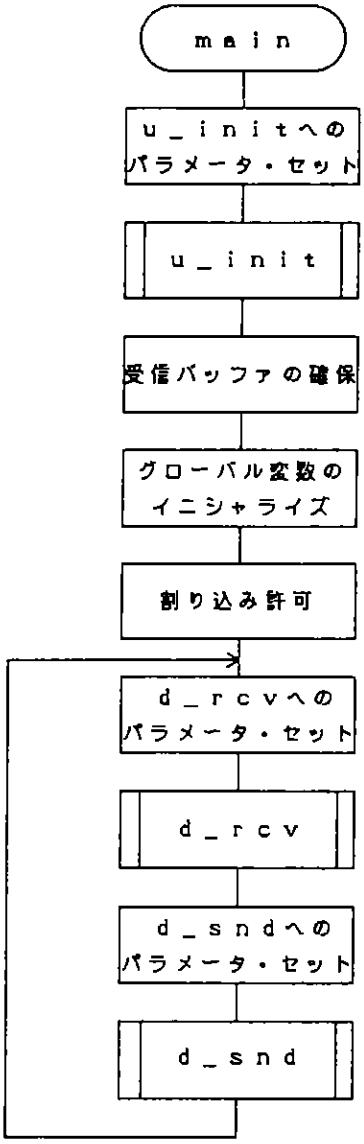
モジュール名	メイン処理：void main()
処理内容	①UARTイニシャライズ関数の呼び出し パラメータ：brgm=80H,brg=01H,s_bit=2,p_bit=0,c_length=8 (注) ②受信バッファの確保 malloc(size_t size)関数を使用．256バイトの受信バッファを確保する． ③グローバル変数の初期化 読み出しポインタ，書き込みポインタ＝受信バッファの先頭 受信データ数カウンタ，送信データ数カウンタ，送信制御用のフラグ＝0 ④割り込み許可 ⑤受信関数の呼び出し パラメータ：*r_adr ⑥送信関数の呼び出し パラメータ：*s_adr,s_cnt
使用サブルーチン	malloc(size_t size) U_INIT(int brgm,int brg,int s_bit,int p_bit,int c_length) d_rcv(unsigned char *r_adr) d_snd(unsigned char *s_adr,int s_cnt) r_int()
パラメータ 入力	なし
パラメータ 出力	なし

注　これらのパラメータは，それぞれ以下の値を設定することができます．

- brgm(ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾓｰﾄﾞ･ﾚｼﾞｽﾀ)=80H
- brg(ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾚｼﾞｽﾀ)=01H(9,600ﾎﾞｰ),03H(4,800ﾎﾞｰ)
- s_bit(ｽﾄｯﾌﾟ･ﾋﾞｯﾄ)=1(1ﾋﾞｯﾄ),2(2ﾋﾞｯﾄ)
- p_bit(ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ)=0(ﾊﾟﾘﾃｨなし),1(0ﾊﾟﾘﾃｨ),2(奇数ﾊﾟﾘﾃｨ),3(偶数ﾊﾟﾘﾃｨ)
- c_length(送受信ﾃﾞｰﾀ･ｷｬﾗｸﾀ長)=7(7ﾋﾞｯﾄ),8(8ﾋﾞｯﾄ)

図3-1 に，メイン処理のフロー・チャートを示します．

図3-1　メイン処理フロー・チャート



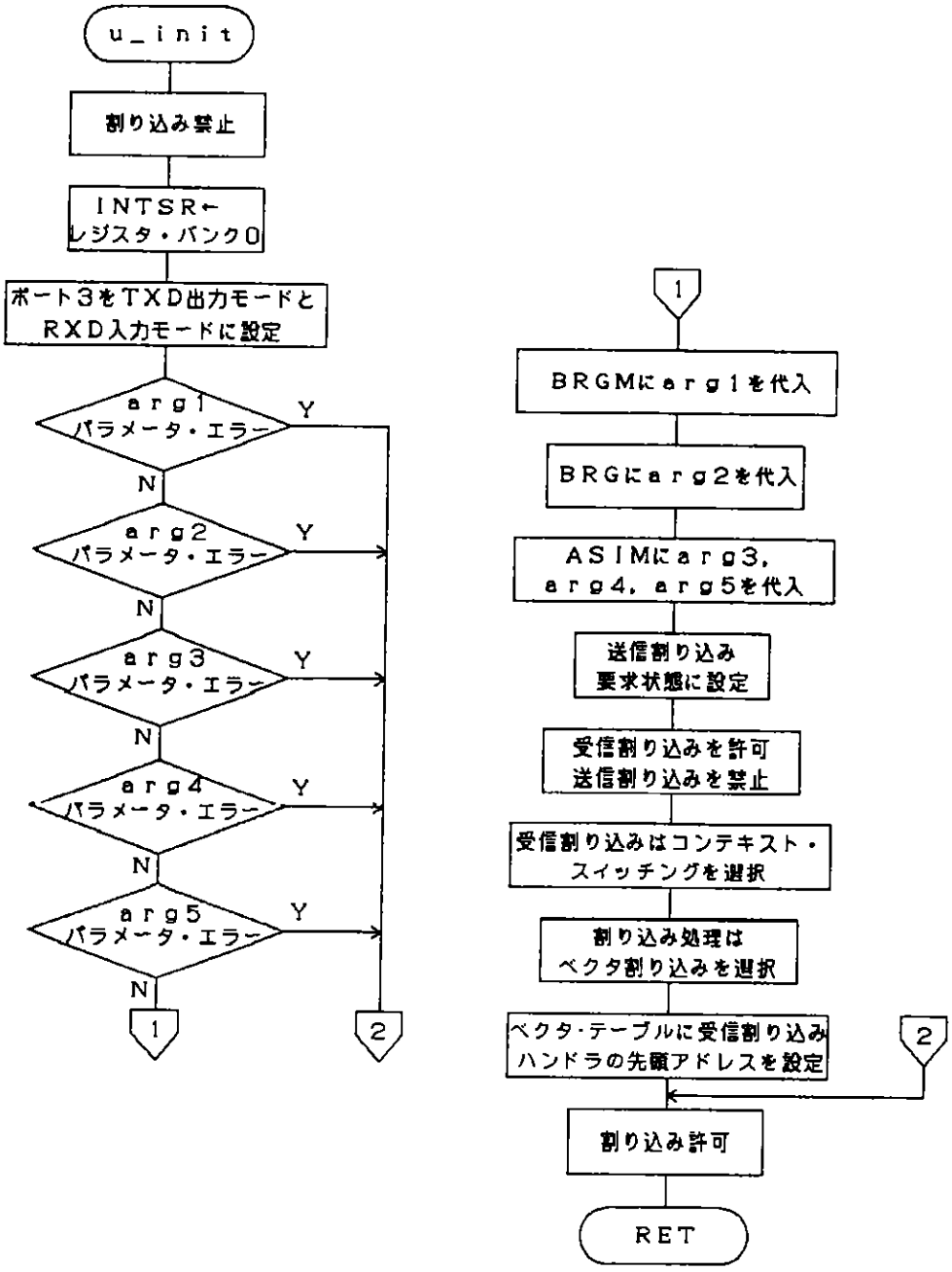
備考　この処理は，ユーザが送受信処理を行いたいときに自由にコーディングする処理です．関数への引き数を正しく設定すれば，この処理のとおりでなくても送受信処理を行うことができます．ただし，すでに使用している変数名等は使用しないでください．

3.1.2 UARTイニシャライズ処理

モジュール名		UARTイニシャライズ処理：U_INIT
処理内容		①受信割り込み発生時のレジスタ・バンクを０に設定 ②レジスタを退避 ③ポート３のイニシャライズ：P30=TXD出力モード，P31=RXD入力モード ④UARTのイニシャライズ メイン処理で渡されたパラメータのエラー・チェック エラー時は，それぞれの引き数の番号をエラー番号として，終了する． パラメータをセット ⑤割り込みの設定 送信完了割り込み発生状態に設定 受信完了割り込みを許可，受信エラー割り込み，送信完了割り込みを禁止 受信完了割り込みは，コンテキスト・スイッチングを使用 受信完了割り込みは，ベクタ割り込みで処理 ⑥受信完了割り込みハンドラの先頭アドレスをレジスタ・バンク０にセット ⑦レジスタを復帰
使用サブルーチン		なし
パラメータ	入力	arg1=brgm arg2=brg arg3=s_bit arg4=p_bit arg5=c_length
	出力	ｎ＝argnがエラー

図3-2 に，UARTイニシャライズ処理のフロー・チャートを示します．

図3-2　UARTイニシャライズ処理フロー・チャート



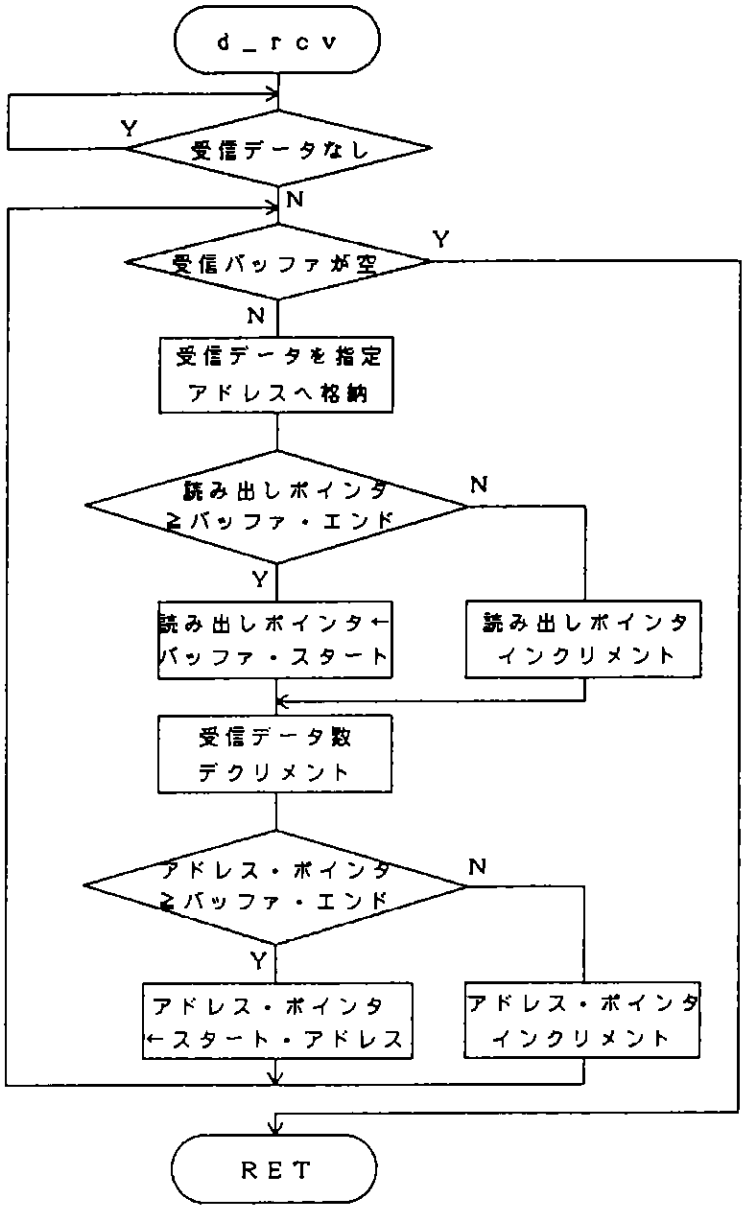
備考　この処理は，送受信を行う際は必ず最初に呼び出します．この処理をせずに送受信を行った場合の動作は，保証しません．
この処理は，アセンブラで記述しています．

3.1.3 受信処理

モジュール名		受信処理：d_rcv(unsigned char *r_adr)
処理内容		①データを受信するまで待つ ②受信したデータをすべて処理するまで，③の処理を繰り返す ③データを格納する ・受信バッファの読み出しポインタの指す内容を，メイン処理で指定されたアドレスへ格納 ・読み出しポインタをインクリメント（バッファの最後であった場合は，バッファの先頭に設定） ・受信データ数カウンタをデクリメント ・メイン処理で指定されたアドレスをインクリメント（バッファの最後であった場合は，バッファの先頭に設定）
使用サブルーチン		なし
パラメータ	入力	unsigned char *r_adr=受信データを格納するバッファのアドレス･ポインタ
	出力	なし

図3-3 に，受信処理のフロー・チャートを示します．

図3-3　受信処理フロー・チャート



備考　この処理は，受信完了割り込みハンドラ(r_int)でデータを受信していないと，データを受信するまで永久に待ち続けます．

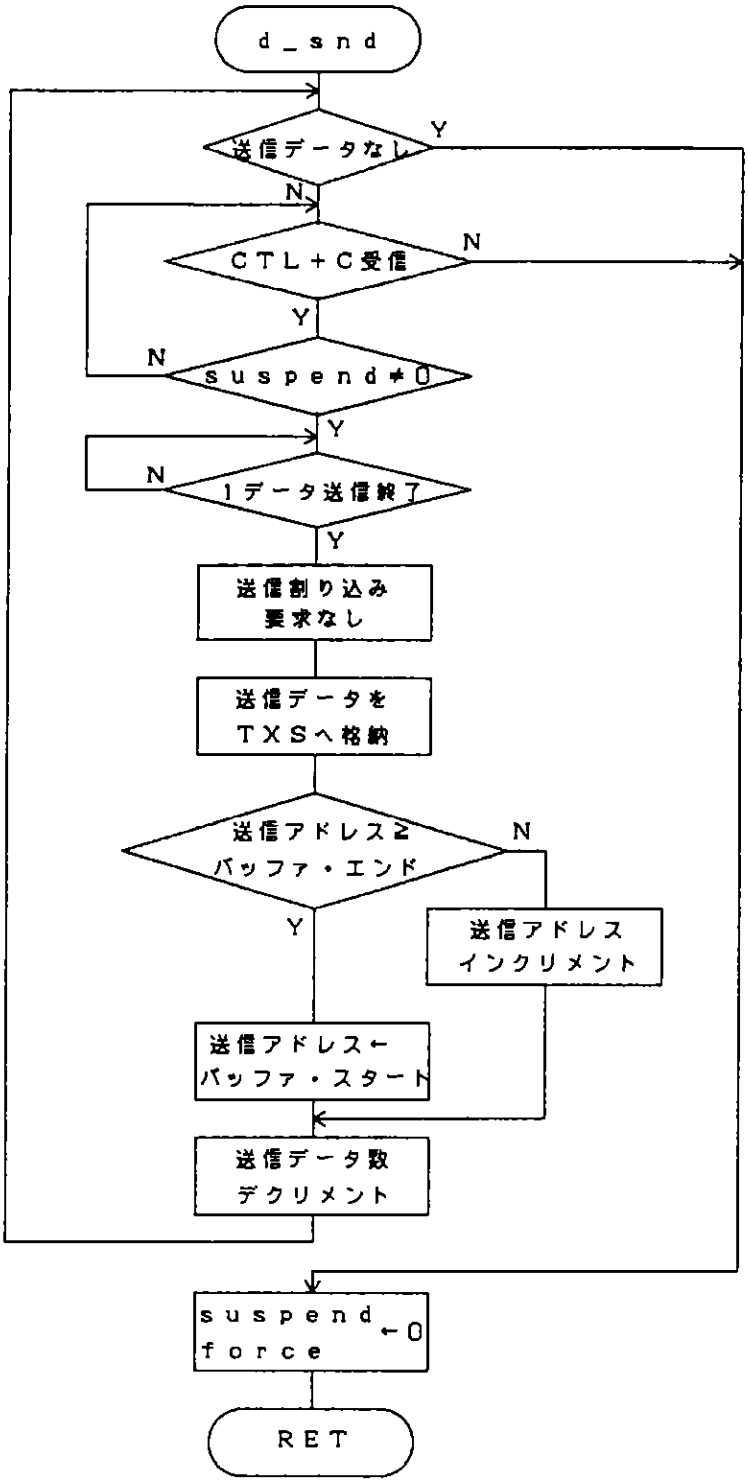
3.1.4 送信処理

モジュール名	送信処理：d_snd(unsigned char *s_adr,int s_cnt)	
処理内容	①送信するデータをすべて処理するまで，②から④の処理を繰り返す ②送信を制御するコートが受信されているかどうかのチェック ・コントロールＣ受信(注1) 送信制御用のフラグをクリアし，送信を終了 ・コントロールＳ受信(注2) コントロールＱが受信されるまで，送信を停止．ただし，コントロールＣが受信されたら，コントロールＣ受信時の処理をする ③１バイトの送信が完了するまで待つ ④データを格納する ・メイン処理で指定されたアドレスの内容（送信データ）をTXSへ格納 ・メイン処理で指定されたアドレスをインクリメント（バッファの最後であった場合は，バッファの先頭に設定） ⑤送信制御用のフラグをクリア	
使用ルーチンサブ	なし	
パラメータ	入力	unsigned char *r_adr=受信データを格納するバッファのアドレス･ポインタ int s_cnt　　=送信するデータ数カウンタ
	出力	なし

注１．force=1:コントロールＣ受信
　２．suspend=1:コントロールＳ受信

図3-4 に，送信処理のフロー・チャートを示します．

図3-4　送信処理フロー・チャート

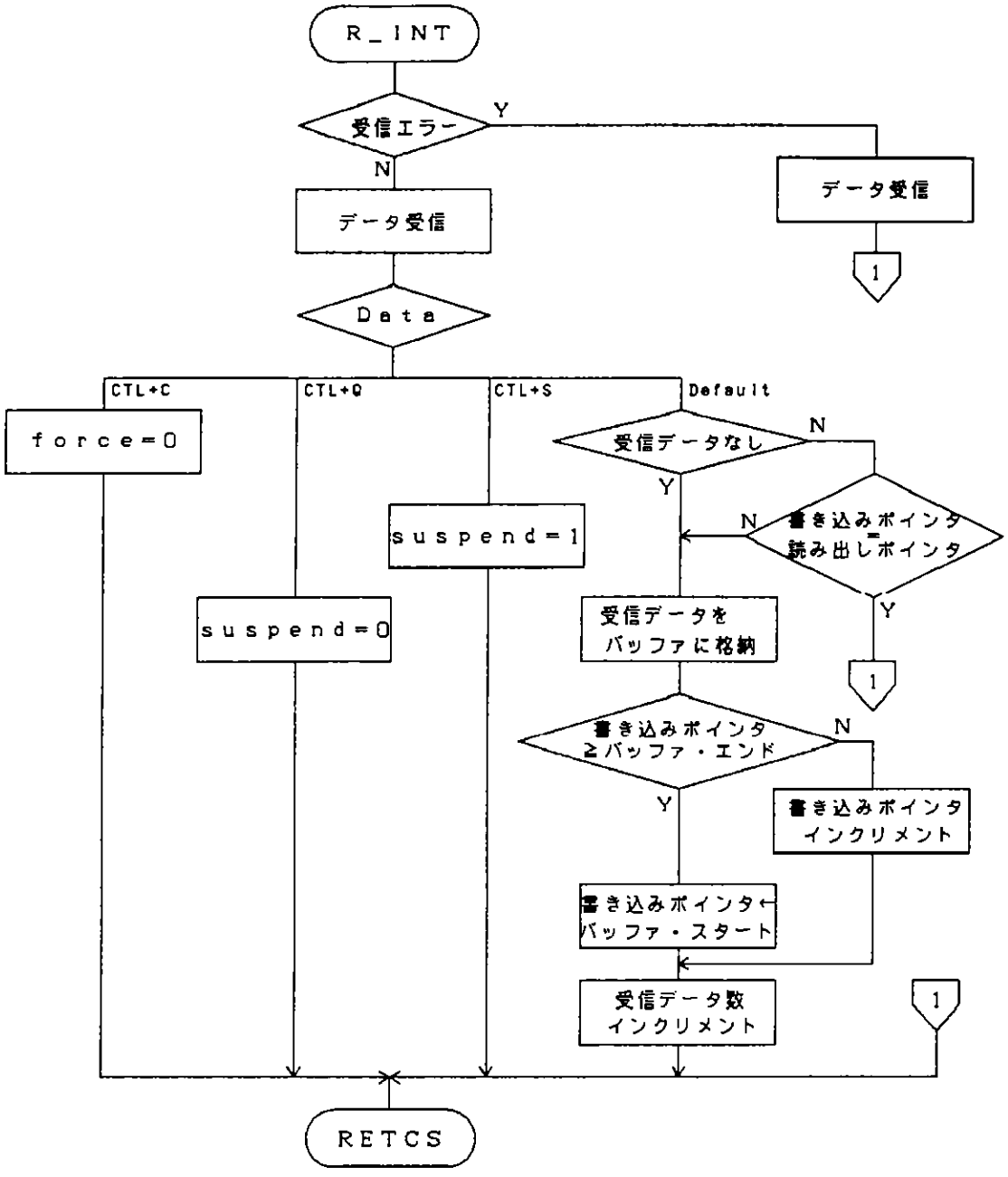


3.1.5　受信完了割り込みハンドラ

モジュール名		受信完了割り込みハンド：r_int()
処理内容		①受信エラーが発生しているかどうかのチェック ・受信エラー発生時 RXBからデータを読み出し(ASISレジスタ・クリア)，ハンドラを終了 ②受信データをRXBから取り出す ③受信データの解析 ・受信データ＝コントロールC コントロールC受信フラグ(force)をセット(1)し，ハンドラを終了 ・受信データ＝コントロールQ コントロールS受信フラグ(suspend)をリセット(0)し，ハンドラを終了 ・受信データ＝コントロールS コントロールS受信フラグ(suspend)をセット(1)し，ハンドラを終了 ・受信データ＝上記以外 (ⅰ)受信バッファにすべてデータが格納されている場合は，何もせずにハンドラを終了 (ⅱ)RXBから取り出したデータを受信バッファの書き込みポインタの指すところに格納 (ⅲ)書き込みポインタをインクリメント（バッファの最後であった場合は，バッファの先頭に設定） (ⅳ)受信データ数カウンタをインクリメント
使用サブルーチン		なし
パラメータ	入力	なし
	出力	なし

図3-5 に，受信完了割り込みハンドラのフロー・チャートを示します・

図3-5　受信完了割り込みハンドラ・フロー・チャート



備考　受信バッファにすべてデータが格納されている場合は，新たに受信したデータは，読み飛ばされます．

3.2 変数説明

本節では，今回作成した送受信処理のプログラムで使用した変数について説明します．

変数はすべてグローバルに宣言しており，どのモジュールからも呼び出しが可能となっています．

表3-2　変数一覧

変数名	型	意　味
r_adr	unsigned char	受信ﾃﾞｰﾀを格納するﾊﾞｯﾌｧのｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ（ﾒｲﾝ処理で指定）
r_start	unsigned char	受信ﾃﾞｰﾀを格納しておく，受信ﾊﾞｯﾌｧの先頭ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ
r_end	unsigned char	受信ﾃﾞｰﾀを格納しておく，受信ﾊﾞｯﾌｧの最後のｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ
r_wp	unsigned char	受信バッファの書き込みポインタ
r_rp	unsigned char	受信バッファの読み出しポインタ
s_adr	unsigned char	送信ﾃﾞｰﾀが格納されているﾊﾞｯﾌｧのｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ
s_start	unsigned char	送信ﾃﾞｰﾀが格納されているﾊﾞｯﾌｧの先頭ｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ
s_end	unsigned char	送信ﾃﾞｰﾀが格納されているﾊﾞｯﾌｧの最後のｱﾄﾞﾚｽ･ﾎﾟｲﾝﾀ
r_cnt	int	受信したデータ数
s_cnt	int	送信するデータ数
r_tmp	int	受信データ数の一時退避領域
force	int	コントロールＣ受信時の判断用のフラグ
suspend	int	コントロールＳ受信時の判断用のフラグ
data	int	RXBから取り出したﾃﾞｰﾀを代入する変数
error	int	RXBから取り出したﾃﾞｰﾀを代入する変数(受信ｴﾗｰ発生時)
i	int	総受信データ数
brgm	int	ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾓｰﾄﾞ･ﾚｼﾞｽﾀ(UARTｲﾆｼｬﾗｲｽﾞ関数へのﾊﾟﾗﾒｰﾀ)
brg	int	ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾚｼﾞｽﾀ(UARTｲﾆｼｬﾗｲｽﾞ関数へのﾊﾟﾗﾒｰﾀ)
s_bit	int	ｽﾄｯﾌﾟ･ﾋﾞｯﾄ(UARTｲﾆｼｬﾗｲｽﾞ関数へのﾊﾟﾗﾒｰﾀ)
p_bit	int	ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ(UARTｲﾆｼｬﾗｲｽﾞ関数へのﾊﾟﾗﾒｰﾀ)
c_length	int	送受信ﾃﾞｰﾀのｷｬﾗｸﾀ長(UARTｲﾆｼｬﾗｲｽﾞ関数へのﾊﾟﾗﾒｰﾀ)

第４章　開発方法

本章では，作成したプログラムを78Kシリーズのｃコンパイラ，アセンブラを用いてそれぞれコンパイル，アセンブルし，HEX形式にするまでの手順を示します．

なお，本章では具体例として，今回作成した際に使用した方法で説明します．詳細は，各ユーザーズ・マニュアルを参照してください．

4.1 オブジェクト・モジュール・ファイル作成

本節では，任意のエディタで作成したＣ言語およびアセンブラのプログラムをコンパイル，アセンブルする方法を示します．

4.1.1 コンパイル

任意のエディタで作成したＣ言語のプログラム(B:C_DRV.C)を，CC78K3を用いてコンパイルします．以下に起動方法を示します．

```
A>CC78K3 -C320 B:C_DRV.C -AB:↵
          ①              ②
```

①デバイス種別指定オプション：ターゲット・デバイスは，μPD78320を使用します．

②ｱｾﾝﾌﾞﾗ･ｿｰｽ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ作成指定ｵﾌﾟｼｮﾝ：アセンブラ・ソース・モジュール・ファイルをＢドライブに作成します．

今回は，Ｃ言語のプログラム中に“ASM文”を使用しているため，コンパイラでオブジェクト・モジュール・ファイルは生成しません．これは，コンパイル時に，以下のようなエラーが出力されることから分かります．したがって，コンパイラでアセンブラ・ソース・モジュール・ファイルを作成し，さらにアセンブルするという方法で，オブジェクト・モジュール・ファイルを生成します．

エラー・メッセージ　W837:Output assembler source file, not object file

Ｃコンパイラについての詳細は，「ＣＣ７８Ｋシリーズ　Ｃコンパイラ操作編　78K/Ⅲｼﾘｰｽﾞ用　ユーザーズ・マニュアル」(EEU-656)を参照してください．

4.1.2　アセンブル

任意のエディタで作成したアセンブラのプログラム(B:A_DRV.ASM)およびコンパイラ(CC78K3)で生成されたアセンブラ・ソース・モジュール・ファイル(B:C_DRV.ASM)を，RA78K3を用いてアセンブルします．以下に起動方法を示します．

```
A>RA78K3 B:A_DRV.ASM -NCA -OB: -PB:↵
A>RA78K3 B:C_DRV.ASM -NCA -OB: -PB:↵
                     ①   ②   ③
```

①シンボル名ケース指定オプション：シンボル名の大文字，小文字の区別をするように指定します．
Ｃ言語のプログラムとリンクするときは，必ずこのオプションを指定してください．

②ｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ出力指定ｵﾌﾟｼｮﾝ：オブジェクト・モジュール・ファイルをＢドライブに作成します．

③ｱｾﾝﾌﾞﾙ･ﾘｽﾄ･ﾌｧｲﾙ出力指定ｵﾌﾟｼｮﾝ：アセンブル・リスト・ファイルをＢドライブに作成します．

アセンブラについての詳細は，「ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ操作編 78K/Ⅲシリーズ用　ユーザーズ・マニュアル　第４章　アセンブラ」(EEU-662)を参照してください．

4.2 ロード・モジュール・ファイル作成

本節では，コンパイルおよびアセンブルで生成されたオブジェクト・モジュール・ファイルをリンクする方法を示します．

4.2.1 リンク

4.1.2で作成したオブジェクト・モジュール・ファイル(B:C_DRV.REL,B:A_DRV.REL)を，LK78K3を用いてリンクします．以下に起動方法を示します．

```
A>LK78K3 -FB:LINK.JOB↵
          ①
```

①パラメータ・ファイル指定オプション：入力ファイル名，オプションを“B:LINK.JOB”というパラメータ・ファイルより入力します．

・パラメータ・ファイルの内容

```
-BCL320.LIB -DB:DRIVER.JOB -S -PB:DRIVER.MAP -OB:DRIVER.LNK
    ①           ②        ③       ④            ⑤
CS320R.REL B:C_DRV.REL B:A_DRV.REL ROM320.REL
  ⑥                    ⑦
```

①ライブラリ・ファイル指定オプション：Cコンパイラが提供している“CL320.LIB”というライブラリをライブラリ・ファイルとして指定します．

②ディレクティブ・ファイル指定オプション：“B:DRIVER.JOB”をディレクティブ・ファイルとして指定します(4.2.2　参照)．

③スタック解決用シンボル生成指定オプション：スタック解決用のパブリック・シンボル“_@STBEG”と“_@STEND”を自動生成します．

④リンク・リスト・ファイル出力指定オプション：リンク・リスト・ファイルを“B:DRIVER.MAP”とします．

⑤ロード・モジュール・ファイル出力指定オプション：ロード・モジュール・ファイルを“B:DRIVER.LNK”とします．

⑧ｽﾀｰﾄｱｯﾌﾟ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ(注)：オブジェクト･プログラムの初期化を行うものです．C言語のプログラムをリンクするときは，スタートアップ･モジュール･ファイルをリンクする必要があります．スタートアップ･モジュール･ファイルは，リンクするファイルの最初に指定します(ROM化を行わない場合は，順番の指定はありません)．

⑦ROM化を行う際のﾘﾝｸ･ﾌｧｲﾙ：ROM化を行う際に，一緒にリンクするファイルです．初期化データの終端を示します．このファイルは，リンクするファイルの最後に指定します．

4.2.2　ディレクティブ・ファイル

ディレクティブ・ファイルは，リンカに対して入力ファイルや使用可能なメモリ，セグメントの配置など，リンク時の各種の指示を行うための命令群を記述したテキスト・ファイルです．

メモリ・ディレクティブ，セグメント配置ディレクティブに分かれています．

(1)メモリ・ディレクティブ

メモリ・ディレクティブは，メモリ領域を定義するディレクティブです．定義したメモリ領域には，名前（メモリ領域名）をつけ，セグメント配置ディレクティブで参照可能にします．以下に構文を示します．

MEMORY△メモリ領域名▲:▲（▲オリジン値▲，▲サイズ▲）［／▲メモリ空間名］

・メモリ領域名：定義するメモリ領域の名前を指定します．
・オリジン値　：定義するメモリの先頭アドレスを指定します．
・サイズ　　：定義するメモリ領域のサイズを指定します．
・メモリ空間名：メモリ領域を割り付けるメモリ空間名を指定します．

備考１．　△：スペース
　　２．　▲：省略可能なスペース

(2)セグメント配置ディレクティブ

セグメント配置ディレクティブは，指定したセグメントをメモリ領域上の特定番地に配置するディレクティブです．以下に構文を示します．

MERGE△セグメント名▲:
（▲［配置型定義］［▲結合型定義］）［／▲メモリ空間名］
［▲＝▲バインド指定］［▲／▲メモリ空間名］

・セグメント名：リンカに入力するオブジェクト・モジュール・ファイル中に含まれるセグメント名です．
・配置型定義　：指定したスタート・アドレスにセグメントを配置します．
“AT▲(▲スタート・アドレス▲)”のみ指定可能です．
・結合型定義　：指定されたセグメントに対し，同名の入力セグメントが複数ある場合に，セグメントの結合方法を指定します．
・バインド指定：セグメントを配置するメモリ領域を決定するための指定です．
・メモリ空間名：セグメントを配置するメモリ空間を指定します．

備考１．　△：スペース
２．　▲：省略可能なスペース

(3)ディレクティブ・ファイルの内容

以下に，今回使用したディレクティブ・ファイル(B:DRIVER.JOB)の内容を示します．

```
memory CLT : ( 00000h , 00080h ) … CLT領域を0Hから80Hに定義します．
memory ROM : ( 01000h , 01000h ) … ROM領域を1000Hから2000Hに定義します．
memory RAM : ( 02000h , 057F0h ) … RAM領域を2000Hから77F0Hに定義します．
memory SDR : ( 0FE20h , 000E0h ) … SDR領域をFE20HからFF00Hに定義します．
merge CALT : = CLT……入力ｾｸﾞﾒﾝﾄCALTをCLT領域に割り付けます．
merge CODE : = ROM……入力ｾｸﾞﾒﾝﾄCODEをROM領域に割り付けます．
merge CNST : = ROM……入力ｾｸﾞﾒﾝﾄCNSTをROM領域に割り付けます．
merge R_DATA : = ROM……入力ｾｸﾞﾒﾝﾄR_DATAをROM領域に割り付けます．
merge DATA : = RAM……入力ｾｸﾞﾒﾝﾄDATAをRAM領域に割り付けます．
merge R_INIT : = ROM……入力ｾｸﾞﾒﾝﾄR_INITをROM領域に割り付けます．
merge INIT : = RAM……入力ｾｸﾞﾒﾝﾄINITをRAM領域に割り付けます．
merge R_DATS : = SDR……入力ｾｸﾞﾒﾝﾄR_DATSをSDR領域に割り付けます．
merge DATS : = SDR……入力ｾｸﾞﾒﾝﾄDATSをSDR領域に割り付けます．
merge R_INIS : = SDR……入力ｾｸﾞﾒﾝﾄR_INISをSDR領域に割り付けます．
merge INIS : = SDR……入力ｾｸﾞﾒﾝﾄINISをSDR領域に割り付けます．
merge BITS : = SDR……入力ｾｸﾞﾒﾝﾄBITSをSDR領域に割り付けます．
```

リンカについての詳細は，「ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ操作編 78K/Ⅲシリーズ用　ユーザーズ・マニュアル　第５章　リンカ」(EEU-662)を参照してください．

4.3 HEX形式オブジェクト・モジュール・ファイル作成

本節では，リンカで生成されたロード・モジュール・ファイルをHEX形式オブジェクト・モジュール・ファイルにする方法を示します．

4.3.1 オブジェクト・コンバータ

4.2で作成したロード・モジュール・ファイル(B:DRIVER.LNK)を，OC78K3を用いてHEX形式オブジェクト・モジュール・ファイルとして出力します．以下に起動方法を示します．

```
A>OC78K3 B:DRIVER.LNK -OB:DRIVER.HEX -SB:DRIVER.SYM↵
                       ①              ②
```

①HEX形式ｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ出力指定ｵﾌﾟｼｮﾝ：HEX形式オブジェクト・モジュール・ファイルを“B:DRIVER.HEX”として指定します．

②シンボル･テーブル･ファイル出力指定ｵﾌﾟｼｮﾝ：シンボル・テーブル・ファイルを“B:DRIVER.SYM”として指定します．

オブジェクト・コンバータについての詳細は，「ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ操作編　78K/Ⅲシリーズ用　ユーザーズ・マニュアル　第６章　オブジェクト・コンバータ」(EEU-662)を参照してください．

4.4 ダウン・ロード

本プログラムは，２章で説明したように，EB-78320-98Cとターミナル(MD-910TM)とを，RS-232Cを介して接続した回路を動作させるものです．

以下にEBボードへのダウン・ロードの方法を示します．

① EBボードの電源を入れます．

② コンソール(PC-9801VX)から，

A>EB78320↵

と入力します（これでEBボードは立ち上がります）．

③ 作成したHEX形式オブジェクト・コンバータをロードします（ロードするファイルは，“B:DRIVER”です）．

0>LOD B:DRIVER↵

これで作成したプログラムはEBボード上にダウン・ロードされました．以後，EBボード上で動作させることができます．

第５章　Ｃコンパイラの注意事項

本章では，Ｃコンパイラ(CC78K3)を使用してプログラムを作成する際の注意事項について示します．

5.1 Ｃソース・プログラム作成前の注意事項

本節では，Ｃコンパイラを使用してプログラムを作成するうえで，μPD78320の機能を利用するときの注意事項について示します．

5.1.1　マクロ・サービス使用時

Ｃコンパイラでマクロ・サービスを使用する場合，マクロ・サービス・コントロール・ワード・アドレスが，Ｃコンパイラで，sreg変数，bit型変数の領域として使われているものがあります．したがって，アドレスが重ならないマクロ・サービスは，問題なく使用できますが，アドレスが重なってしまうマクロ・サービスに関しては，sreg変数，bit型変数を使用しない場合のみ，使用できます．

また，マクロ・サービス処理を行うときは，あらかじめマクロ・サービス処理を指定できる割り込み要求に対するマクロ・サービス・モード・レジスタとチャネル・ポインタに，値を設定しておく必要があります．

5.1.2　レジスタ・バンク使用時

Ｃコンパイラでは，レジスタ・バンクは“RB7”に固定されています（スタートアップ・ルーチンで“RB7”に設定します）．“RB2”から“RB6”は，Ｃコンパイラで使用しているので，使用することはできません．しかし“RB0”，“RB1”は，Ｃコンパイラでは使用しませんので，自由に使用することができます．したがって，割り込み処理等で，レジスタ・バンクの切り替えを行いたい場合は，“RB0”または“RB1”に設定してください．

5.2 Cソース・プログラム作成時の注意事項

本節では，Cソース・プログラムを作成する際の注意事項について示します．

5.2.1 特殊機能レジスタ使用時

Cコンパイラでは，特殊機能レジスタを使用するときに，C言語のソース・レベルで記述可能となるように，“pragma指令”というものが用意されています．これは，Cソース・プログラム中で，デバイスが持つsfr名を使用することを指定するものです．この指定ののち，Cソース中では，sfr名の宣言なしに，直接sfr名を使用することができます．

〔例〕 #pragma sfr

sfr領域利用時およびpragma指令についての詳細は，「ＣＣ７８Ｋシリーズ Cコンパイラ言語編 78K/Ⅲｼﾘｰｽﾞ用 ユーザーズ・マニュアル」(EEU-655)を参照してください．

5.2.2 include文使用時

Cコンパイラでは，標準ライブラリ関数として，次のものを用意しています．

・入出力関数
・文字・文字列関数
・メモリ関数
・プログラム制御関数
・数学関数
・特殊関数

これらはすべて，１１個のヘッダ・ファイルに組み込まれています．したがって，これらの標準ライブラリ関数を使用する場合，処理の最初にヘッダ・ファイルの宣言“include<>”をしなければなりません．

〔例〕 #include <stdlib.h>

ヘッダ・ファイルのうち，“stdlib.h”を使用する場合は，その前に“stddef.h”の宣言をする必要があります．これは，“stdlib.h”内で使用している型(size_t)を，“stddef.h”内で宣言しているからです．

“stdlib.h”を，“stddef.h”宣言なしで使用すると，コンパイル時にエラーとなりますので，必ず両方宣言してください．

5.2.3 ASM文使用時

Cコンパイラでは，Cソース・プログラム中に，アセンブラ・ソース・プログラムを記述できるように，“ASM指令”というものが用意されています．これは，C言語のソース・プログラム中で，次に続く文をアセンブラ・ソースとして，コンパイラが出力するアセンブラ・ソース・モジュール・ファイル中に埋め込むものです．

〔例〕
```
#asm
        EI
#endasm
```

しかし，この“ASM文”は，レジスタの退避等の処理を行わないため，レジスタ等を使用する場合は，前のレジスタの内容を壊さないように注意が必要です．また，４章でも説明したように，“ASM文”を使用すると，コンパイラでオブジェクト・モジュール・ファイルを生成しません．したがって，コンパイラでアセンブラ・ソース・モジュール・ファイルを作成し，さらにアセンブルするという方法で，オブジェクト・モジュール・ファイルを生成しなければなりません．

ASM文についての詳細は，「ＣＣ７８Ｋシリーズ　Cコンパイラ言語編　78K/Ⅲｼﾘｰｽﾞ用ユーザーズ・マニュアル」(EEU-655)を参照してください．

5.3 ロード・モジュール・ファイル作成時の注意事項

本節では，ロード・モジュール・ファイルを作成する際の注意事項について示します．

5.3.1 Cとアセンブラのリンク

本項では，C言語からアセンブリ言語ルーチンを呼び出す際の注意事項について示します．

〔例〕
```
int brgm,brg,s_bit,p_bit,c_length;
int U_INIT(int brgm,int brg,int s_bit,int p_bit,int c_length);

void main()
{
        int error;

        brgm     = 0x80;
        brg      = 0x01;
        s_bit    = 2;
        p_bit    = 0;
        c_length = 8;

        error = U_INIT(brgm,brg,s_bit,p_bit,c_length);
}
```

注意　“U_INIT”は，アセンブリ言語で記述された関数です．

(1)レジスタ

Cコンパイラが生成するプログラムでは，レジスタ変数用レジスタを退避せずにアセンブラの関数を呼び出します．したがって，アセンブラ側でこれらの変数の値を変更する場合は，事前に値の退避を行わなければなりません．

また，関数内の“PUSH”，“POP”によりスタック・ポインタ(SP)の値は変わります．したがって，スタック・ポインタを“HL”レジスタにコピーして，引き数のベース・ポインタとして使用します．

関数が終了するときは，退避したベース・ポインタとワーク・レジスタを復帰します．

(2)引き数

このプログラム例で，実行時に行われるプログラム間のインタフェースと制御の流れを以下に示します．

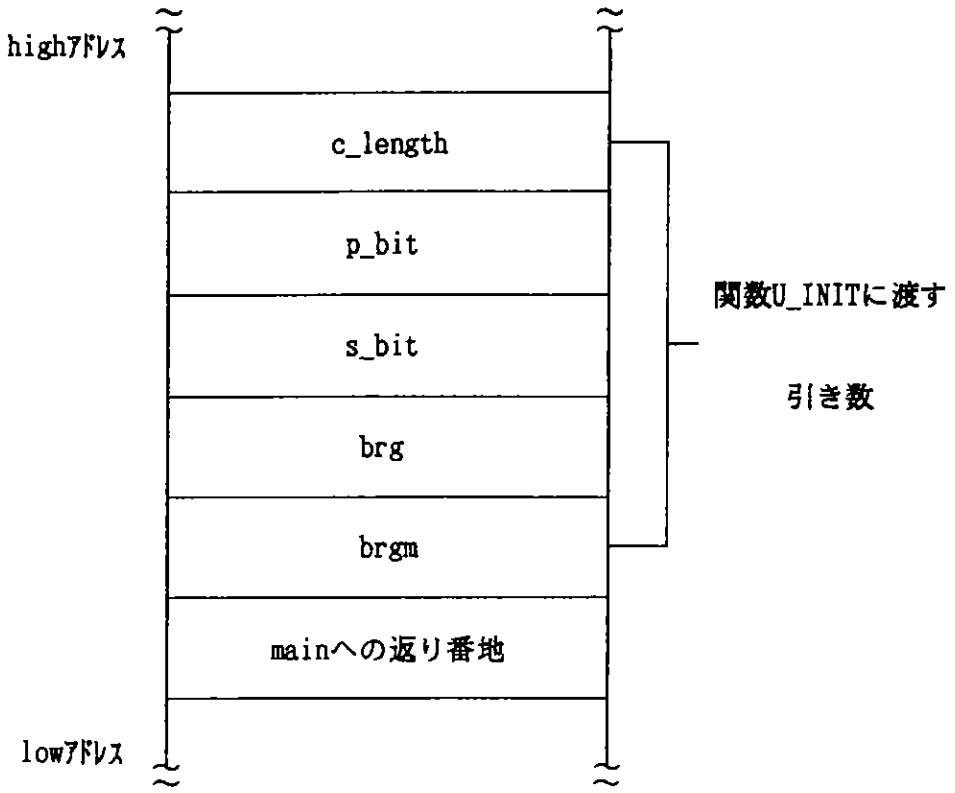
①関数mainから関数U_INITへ渡す引き数をスタックに積む

②CALL命令により関数U_INITに制御を渡す

ここで，関数mainから関数U_INITに制御が渡された直後のスタックには，mainから渡された引き数と，mainへの戻り番地が積まれます．mainから渡された引き数は，図5-1からも分かるように，Ｃ言語で記述した際に右側に記述した方から積まれます．したがって，アセンブリ言語側で引き数を読み出す時は，注意が必要です．

また，関数に戻り値がある場合，アセンブリ言語側では，戻り値を“BC”，“RP2”レジスタへセットします．

図5-1　スタック領域



(3)関数名

Cコンパイラは，出力するオブジェクト・モジュールの外部変数および参照名に“__”を付加します．したがって，アセンブリ言語側では，C言語側で呼び出した関数名に故意に“__”を付加した名前で記述する必要があります．

このプログラム例では，アセンブリ言語側では次のようになります．

〔例〕

```
        NAME    A_DRV
;
        PUBLIC  _U_INIT
;
CODE    CSEG
;
_U_INIT:
        PUSH    HL      ;ベース・ポインタをセーブ
        MOVW    AX,SP   ;スタック・ポインタをコピー
        MOVW    HL,AX
              .
              .
              .         ;本来の処理
              .
              .
        POP     HL      ;ベース・ポインタの復帰
        RET
;
        END
```

付　録　プログラム・リスト

```
#pragma sfr
#include <¥nec¥78k3¥include¥stddef.h>
#include <¥nec¥78k3¥include¥stdlib.h>
#define SIZE    256
#define CTL_C   0x03
#define CTL_S   0x13
#define CTL_Q   0x11

/*****  ｸﾞﾛｰﾊﾞﾙ変数の定義  *****/
/*      r_adr     - 受信ﾊﾞｯﾌｧ･ｱﾄﾞﾚｽ                                    */
/*      s_adr     - 送信ﾊﾞｯﾌｧ･ｱﾄﾞﾚｽ                                    */
/*      r_start   - 受信ﾊﾞｯﾌｧ･ｽﾀｰﾄ･ﾎﾟｲﾝﾀ                              */
/*      r_end     - 受信ﾊﾞｯﾌｧ･ｴﾝﾄﾞ･ﾎﾟｲﾝﾀ                              */
/*      r_wp      - 書き込みﾎﾟｲﾝﾀ                                       */
/*      r_rp      - 読み出しﾎﾟｲﾝﾀ                                       */
/*      r_cnt     - 受信ﾃﾞｰﾀ数ｶｳﾝﾀ                                     */
/*      s_cnt     - 送信ﾃﾞｰﾀ数ｶｳﾝﾀ                                     */
/*      force     - CTL_C受信時のﾌﾗｸﾞ  force = 1 : CTL_C受信           */
/*      suspend   - CTL_S,CTL_Q受信時のﾌﾗｸﾞ  suspend = 1 : CTL_S受信   */
/*                                          suspend = 0 : CTL_Q受信   */
/*      brgm      - ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾓｰﾄﾞ･ﾚｼﾞｽﾀ                    */
/*      brg       - ﾎﾞｰ･ﾚｰﾄ･ｼﾞｪﾈﾚｰﾀ･ﾚｼﾞｽﾀ                           */
/*      s_bit     - ｽﾄｯﾌﾟ･ﾋﾞｯﾄ                                         */
/*      p_bit     - ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ                                         */
/*      c_length  - 送受信ﾃﾞｰﾀ･ｷｬﾗｸﾀ長                                 */
unsigned char   *r_adr, *r_start, *r_end, *r_wp, *r_rp;
unsigned char   *s_adr, *s_start, *s_end;
int     r_cnt, s_cnt, r_tmp;
int     force, suspend;
int     data, error, i;
int     brgm, brg, s_bit, p_bit, c_length;
int     U_INIT(int brgm, int brg, int s_bit, int p_bit, int c_length);
void    d_rcv(unsigned char *r_adr);
void    d_snd(unsigned char *s_adr, int s_cnt);
void    r_int();
```

```
void main()
{

/*****  UARTｲﾆｼｬﾗｲｽﾞ関数の呼び出し  *****/
/*****  ﾊﾟﾗﾒｰﾀの設定  *****/
/*       brgm      = 80H                          */
/*       brg       = 03H ･･･ ﾎﾞｰ･ﾚｰﾄ:4,800ﾎﾞｰ   */
/*                 = 01H ･･･ ﾎﾞｰ･ﾚｰﾄ:9,600ﾎﾞｰ   */
/*       s_bit     = 1   ･･･ 1ﾋﾞｯﾄ                */
/*                 = 2   ･･･ 2ﾋﾞｯﾄ                */
/*       p_bit     = 0   ･･･ ﾊﾟﾘﾃｨなし            */
/*                 = 1   ･･･ 0ﾊﾟﾘﾃｨ               */
/*                 = 2   ･･･ 奇数ﾊﾟﾘﾃｨ            */
/*                 = 3   ･･･ 偶数ﾊﾟﾘﾃｨ            */
/*       c_length  = 7   ･･･ 7ﾋﾞｯﾄ                */
/*                 = 8   ･･･ 8ﾋﾞｯﾄ                */

        brgm      = 0x80;
        brg       = 0x01;
        s_bit     = 2;
        p_bit     = 0;
        c_length  = 8;

        error = U_INIT(brgm, brg, s_bit, p_bit, c_length);

/*****  受信ﾊﾞｯﾌｧの確保(256ﾊﾞｲﾄ)  *****/
        r_start = malloc(SIZE);
        r_adr   = r_start;              /*  r_start : 受信ﾊﾞｯﾌｧ･ｽﾀｰﾄ･ﾎﾟｲﾝﾀ  */
        r_end   = r_start + SIZE - 1;   /*  r_end   : 受信ﾊﾞｯﾌｧ･ｴﾝﾄﾞ･ﾎﾟｲﾝﾀ  */
        s_start = r_start;
        s_adr   = r_start;              /*  s_start : 送信ﾊﾞｯﾌｧ･ｽﾀｰﾄ･ﾎﾟｲﾝﾀ  */
        s_end   = r_start + SIZE - 1;   /*  s_end   : 送信ﾊﾞｯﾌｧ･ｴﾝﾄﾞ･ﾎﾟｲﾝﾀ  */

/*****  ｸﾞﾛｰﾊﾞﾙ変数の初期化  *****/
        r_wp = r_rp =  r_start;
        r_cnt = s_cnt = i = r_tmp = 0;
        force = suspend = data = 0;

/*****  割り込み許可  *****/
        #asm
                EI
        #endasm

        while(1){
/*****  受信関数の呼び出し  *****/
                d_rcv(r_adr);
                r_adr = r_adr + r_tmp;
                i = r_tmp + i;

/*****  送信関数の呼び出し  *****/
                s_cnt = i;
                d_snd(s_adr, s_cnt);
                s_adr = r_adr;

                i = r_tmp = 0;
        }
}
```

```
/*****  受信処理関数  *****/
void d_rcv(unsigned char *r_adr)
{

/*  ﾃﾞｰﾀを受信するまで待つ  */
        while(r_cnt == 0){
        }

/*  ﾃﾞｰﾀｾｯﾄ  */
        r_tmp = r_cnt;
        while(r_cnt != 0){
                *r_adr = *r_rp;          /*  ﾃﾞｰﾀｾｯﾄ  */

                if(r_rp == r_end){
                        r_rp = r_start;
                } else{
                        r_rp++;
                }
                r_cnt--;
                if(r_adr == r_end){
                        r_adr = r_start;
                } else{
                        r_adr++;
                }
        }
}


/*****  送信処理関数  *****/
void d_snd(unsigned char *s_adr, int s_cnt)
{

/*  ﾃﾞｰﾀがなくなるまでﾃﾞｰﾀｾｯﾄ  */
        while(s_cnt != 0){
CHK_C:
/*  受信ﾃﾞｰﾀ = CTL_C : 強制終了  */
                if(force == 0){
/*  受信ﾃﾞｰﾀ = CTL_S : CTL_Q受信まで待つ  */
                        while(suspend != 0){
                                goto CHK_C;
                        }
                        while(IF1L != 1){
                        }
                        IF1L = 0;

                        TXS = *s_adr;   /*  ﾃﾞｰﾀｾｯﾄ  */

                        if(s_adr == s_end){
                                s_adr = s_start;
                        } else{
                                s_adr++;
                        }
                        s_cnt--;
                } else{
                        break;
                }
        }
        force = suspend = 0;
}
```

```
/*****  受信完了割り込みﾊﾝﾄﾞﾗ  *****/
void r_int()
{
/*  受信ｴﾗｰ発生ならば終了  */
        if(ASIS != 0){
                error = RXB;
        } else{
/*  ﾃﾞｰﾀ受信  */
                data = RXB;

/*  ﾃﾞｰﾀ･ﾁｪｯｸ  */
                switch(data){
                        case CTL_C:force = 1;           /*  data = CTL_C : force = 1    */
                                   break;
                        case CTL_Q:suspend = 0;         /*  data = CTL_Q : suspend = 0  */
                                   break;
                        case CTL_S:suspend = 1;         /*  data = CTL_S : suspend = 1  */
                                   break;
                        default   :if(r_cnt != 0){
                                        if(r_wp == r_rp){
                                                break;
                                        }
                                   }

                                   *r_wp = data;        /*  ﾃﾞｰﾀ･ｾｯﾄ  */

                                   if(r_wp == r_end){
                                        r_wp = r_start;
                                   } else{
                                        r_wp++;
                                   }

                                   r_cnt++;
                }
        }
}
```

```
$       PROCESSOR(320)

        NAME    A_DRV
;
        EXTRN   _r_int
        PUBLIC  _U_INIT,_A_R_INT
;
        ORG     0024H           ;INTSRのﾍﾞｸﾀ･ｱﾄﾞﾚｽ = 0024H
        DW      0               ;INTSRにﾚｼﾞｽﾀ･ﾊﾞﾝｸ0を設定
;
        ORG     0FEF4H
V_ADR:  DS      2               ;ﾚｼﾞｽﾀ･ﾊﾞﾝｸ0のﾍﾞｸﾀ･ｱﾄﾞﾚｽ = FEF4H
:
```

```
CODE    CSEG
;
;---------------------------------------
;
;       UARTｲﾆｼｬﾗｲｽﾞ関数
;
;---------------------------------------
_U_INIT:
        DI
        PUSH    HL              ;ﾍﾞｰｽ･ﾎﾟｲﾝﾀをｾｰﾌﾞ
        MOVW    AX, SP          ;ｽﾀｯｸ･ﾎﾟｲﾝﾀをｺﾋﾟｰ
        MOVW    HL, AX
;
;---------- ﾎﾟｰﾄ3のｲﾆｼｬﾗｲｽﾞ
        MOV     PMC3, #03H      ;ﾎﾟｰﾄ3ﾋﾞｯﾄ0 = TxD出力ﾓｰﾄﾞ,  ﾎﾟｰﾄ3ﾋﾞｯﾄ1 = RxD入力ﾓｰﾄﾞ
;
;---------- UARTの初期化
;ﾊﾟﾗﾒｰﾀのﾁｪｯｸ
CHK1:   MOV     A, [HL+4]       ;[HL+4] = arg1
        CMP     A, #80H         ;arg1(BRGM) = 80H(ﾎﾞｰ･ﾚｰﾄ×1ﾓｰﾄﾞ)?
        BNZ     $ERROR1
CHK2:   MOV     A, [HL+6]       ;[HL+6] = arg2
        CMP     A, #01H         ;arg2(BRG) = 01H(9,600ﾎﾞｰ) or 03H(4,800ﾎﾞｰ)?
        BZ      $CHK3
        CMP     A, #03H
        BNZ     $ERROR2
CHK3:   MOV     A, [HL+8]       ;[HL+8] = arg3
        CMP     A, #1           ;arg3(ｽﾄｯﾌﾟ･ﾋﾞｯﾄ) = 1(1ﾋﾞｯﾄ) or 2(2ﾋﾞｯﾄ)?
        BZ      $CHK4
        CMP     A, #2
        BNZ     $ERROR3
CHK4:   MOV     A, [HL+10]      ;[HL+10] = arg4
        SHR     A, 2            ;arg4(ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ) = 0(ﾊﾟﾘﾃｨなし) or 1(0ﾊﾟﾘﾃｨ) or 2(奇数ﾊﾟﾘﾃｨ) or 3 (偶数ﾊﾟﾘﾃｨ)?
        BNZ     $ERROR4
CHK5:   MOV     A, [HL+12]      ;[HL+12] = arg5
        CMP     A, #07H         ;arg5(ｷｬﾗｸﾀ長) = 7(7ﾋﾞｯﾄ) or 8(8ﾋﾞｯﾄ)?
        BZ      $PUT_ARG
        CMP     A, #08H
        BNZ     $ERROR5
        BR      $PUT_ARG
;
;ｴﾗｰ処理
ERROR1:
        MOVW    BC, #01H
        BR      $ERROR
ERROR2:
        MOVW    BC, #02H
        BR      $ERROR
ERROR3:
        MOVW    BC, #03H
        BR      $ERROR
ERROR4:
        MOVW    BC, #04H
        BR      $ERROR
ERROR5:
        MOVW    BC, #05H
        BR      $ERROR
;
```

```
;ﾊﾟﾗﾒｰﾀのｾｯﾄ
PUT_ARG:
        MOV     A,[HL+4]
        MOV     BRGM,A          ;arg1 -> BRGM
        MOV     A,[HL+6]
        XCH     A,X
        MOV     A,#0
        MOVW    BRG,AX          ;arg2 -> BRG
        MOV     A,[HL+8]
        SHL     A,1
        AND     A,#04H          ;arg3(ｽﾄｯﾌﾟ･ﾋﾞｯﾄ) -> bit2
        MOV     B,A
        MOV     A,[HL+10]
        SHL     A,4             ;arg4(ﾊﾟﾘﾃｨ･ﾋﾞｯﾄ) -> bit4,bit5
        OR      B,A
        MOV     A,[HL+12]
        AND     A,#08H          ;arg5(ｷｬﾗｸﾀ長) -> bit3
        OR      A,B
        OR      A,#0C0H
        MOV     ASIM,A          ;arg3,arg4,arg5 -> ASIM
;
;--------- 割り込みの設定
        MOV     IF1L,#01H       ;STIF = 1(ﾀﾞﾐｰ送信完了割り込み)
        MOV     MK0H,#7FH       ;SRMK = 0(受信割り込み可能), SERMK = 1(受信ｴﾗｰ割り込み不可能)
        MOV     MK1L,#0FFH      ;STMK = 1(送信割り込み不可能)
        MOV     CSE0H,#80H      ;SRCSE = 1(受信割り込みはｺﾝﾃｷｽﾄ･ｽｲｯﾁﾝｸﾞを使用)
        MOV     ISM0H,#00H      ;SRISM = 0(受信割り込みはﾍﾞｸﾀ割り込みで処理)
;
        MOVW    AX,#_A_R_INT    ;RB0-R4,R5 = 受信完了割り込みﾊﾝﾄﾞﾗの先頭ｱﾄﾞﾚｽ
        MOVW    V_ADR,AX

;
ERROR:
        POP     HL              ;ﾍﾞｰｽ･ﾎﾟｲﾝﾀの復帰
        EI
        RET
;
;
;
```

```
CODE    CSEG
;
;-----------------------------------------
;
;       受信完了割り込みハンドラ
;
;-----------------------------------------
_A_R_INT:
        CALL    !_r_int
        RETCS   !_A_R_INT
;
        END
```

NEC 日本電気株式会社

本　　社	〒108 東京都港区芝五丁目33番1号(日本電気本社ビル)
半導体第一、第二販売事業部	〒108 東京都港区芝五丁目29番11号(日本電気住生ビル) 東　京(03)456－6111
関西支社半導体販売部	〒540 大阪市中央区城見一丁目4番24号(日本電気関西ビル) 大　阪(06)945－3178 大　阪(06)945－3200
中部支社半導体販売部	〒460 名古屋市中区栄四丁目15番32号(日建生住ビル) 名古屋(052)262－3611

北海道支社	札　幌	(011)231－0161
東北支社	仙　台	(022)261－5511
岩手支店	盛　岡	(0196)51－4344
郡山支店	郡　山	(0249)23－5511
いわき支店	いわき	(0246)21－5511
新潟支店	新　潟	(025)247－6101
水戸支店	水　戸	(0292)26－1717
土浦支店	土　浦	(0298)23－6161
神奈川支社	横　浜	(045)324－5511
群馬支店	高　崎	(0273)26－1255
太田支店	太　田	(0276)46－4011
宇都宮支店	宇都宮	(0286)21－2281
小山支店	小　山	(0285)24－5011
長野支店	長　野	(0262)35－1444
松本支店	松　本	(0263)35－1666
上諏訪支店	諏　訪	(0266)53－5350
甲府支店	甲　府	(0552)24－4141
沖縄支店	那　覇	(0988)66－5611
埼玉支社	大　宮	(048)641－1411
立川支社	立　川	(0425)26－0911
千葉支社	千　葉	(0472)27－5441
静岡支社	静　岡	(0542)55－2211
浜松支店	浜　松	(0534)52－2711
北陸支社	金　沢	(0762)23－1621
富山支店	富　山	(0764)31－8461
京都支社	京　都	(075)221－8511
神戸支社	神　戸	(078)332－3311
中国支社	広　島	(082)247－4111
岡山支店	岡　山	(0862)25－4455
四国支社	高　松	(0878)22－4141
松山支店	松　山	(0899)45－4111
九州支社	福　岡	(092)271－7700
北九州支店	北九州	(093)541－2887

(技術お問い合せ先)

半導体応用技術本部 第一応用システム技術部	〒108	東京都港区芝五丁目29番11号(日本電気住生ビル)	東　京 (03)798－6105
半導体応用技術本部 第二応用システム技術部	〒540	大阪市中央区城見一丁目4番24号(日本電気関西ビル)	大　阪 (06)945－3383
半導体応用技術本部	〒210	川崎市幸区堀越三丁目484番地(川崎技術センター)	川　崎(044)533－1111

インフォメーションセンター
FAX(044)548-7900
(24時間受付)