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。

RENESAS

アプリケーション・ノート

78K0S/Kx1+

サンプル・プログラム（シリアル・インタフェースUART6）

受信リング・バッファによる全二重通信編

この資料は，サンプル・プログラムの動作概要や使用方法，およびシリアル・インタフェースUART6の設定方法や活用方法を説明したものです。サンプル・プログラムでは，ボー･レートを9600 bpsに設定して，シリアル通信を行い，1キャラクタのデータ受信に応じて，4キャラクタのデータ送信を行います。受信エラー時の場合も同様に，エラー内容に応じて，4キャラクタのデータ送信を行います。

対象デバイス

78K0S/KA1+マイクロコントローラ
78K0S/KB1+マイクロコントローラ

目　次

資料番号　U18914JJ2V0AN00（第2版）
発行年月　July 2008 NS

Windows, Windows XPは，米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

- 本資料に記載されている内容は2008年7月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品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，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，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，ソフトウエアおよびこれらに関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，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「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。

  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

  特別水準：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

  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，生命維持のための装置またはシステム等

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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

第1章　概　　要

このサンプル・プログラムでは，全二重通信が可能なシリアル・インタフェースUART6の使用例を示しています。ボー･レートを9600 bpsに設定して，シリアル通信を行い，1キャラクタのデータ受信に応じて，4キャラクタのデータ送信を行います。受信エラー時の場合も同様に，エラー内容に応じて，4キャラクタのデータ送信を行います。

1. 1　初期設定の主な内容

初期設定の主な内容は，次のとおりです。

●システム・クロック・ソースとして，水晶／セラミック発振クロックを選択[注]
●ウォッチドッグ・タイマの動作停止
●V_{LVI}（低電圧検出電圧）を4.3 V±0.2 Vに設定
●V_{DD}（電源電圧）≧V_{LVI}になったあとに，V_{DD}＜V_{LVI}を検出した場合，内部リセット（LVIリセット）信号を発生
●CPUクロック周波数を8 MHzに設定
●入出力ポートの設定
●シリアル・インタフェースUART6の設定
　・ボー・レート：9600 bps
　・データのキャラクタ長：7ビット
　・パリティ指定：偶数パリティ
　・ストップ・ビット数：1ビット
　・先頭ビットの指定：LSBファースト
　・TxD6出力：通常出力
　・エラー発生時の割り込みにINTSRE6が発生
　・内部動作クロックの動作許可
　・送信動作許可
　・受信動作許可

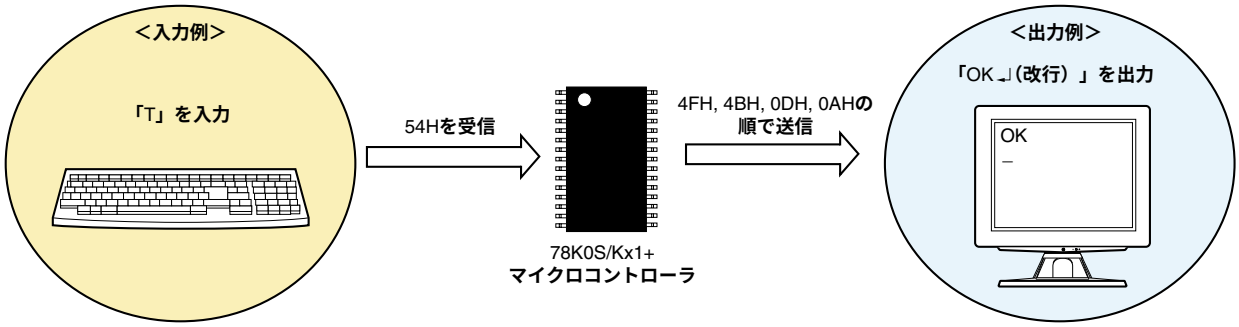
注　オプション・バイトで設定します。

【コラム】全二重通信とは

全二重通信とは，送信動作と受信動作を独立して，同時に動作することが可能な通信のことです。シリアル・インタフェースUART6は，この全二重通信に対応していて，送信と受信を同時に使用することができます。

1. 2　メイン・ループ以降の内容

初期設定完了後は，RxD6端子からのデータ入力により，シリアル通信の受信動作を開始します。このサンプル・プログラムでは，ASCIIコードの送受信を想定しており，1キャラクタのデータ受信に応じて，4キャラクタのデータ送信を行います。受信エラー時の場合も同様に，エラー内容に応じて，4キャラクタのデータ送信を行います。



RAM領域には受信データの格納用バッファとして，50バイト（= 1バイト（1データ）×50）の領域を確保しています。

受信データは，割り込み処理時にバッファに格納するため，連続受信が可能で，バッファ領域の先頭から順次格納していきます。また，バッファをリング・バッファ構成とするため，バッファ領域の最終尾に受信データが達したあとは，再びバッファ領域の先頭から格納していきます。受信データをバッファに格納するときに，バッファに空きがあれば，受信データを格納しますが，バッファに空きがなければ，受信データを格納せずに破棄します。

注意　デバイス使用上の注意事項については，各製品のユーザーズ・マニュアル（78K0S/KA1+, 78K0S/KB1+）を参照してください。

【コラム】ASCIIコードとは

7ビットで1つの文字（アルファベット，数字，記号，制御文字）を表す文字コードです。

【コラム】リング・バッファとは

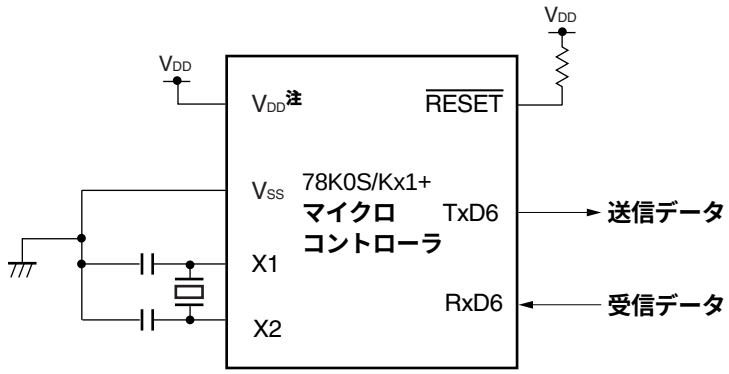
送信・受信バッファの制御方法です。固定のバッファ領域を用意し，バッファに転送された順でデータを処理し，バッファ領域の最終尾の次のアドレスが，先頭アドレスになるようにバッファを制御します。バッファを輪のように使用するため，リング・バッファと呼ばれています。

第2章　回 路 図

この章では，このサンプル･プログラムで使用する回路図を説明します。

2. 1　回 路 図

回路図を次に示します。



注　4.5 V≦VDD≦5.5 Vの電圧範囲で使用してください。

注意1. AVREF端子はVDDに直接接続してください。

2. AVSS端子はGNDに直接接続してください（78K0S/KB1+マイクロコントローラのみ）。
3. 回路図中の端子およびAVREF, AVSS端子以外の未使用端子はすべて出力ポートのため，オープン（未接続）にしてください。

第3章　ソフトウエアについて

この章では，ダウンロードする圧縮ファイルのファイル構成，使用するマイコンの内蔵周辺機能，サンプル・プログラムの初期設定と動作概要，およびフロー・チャートを説明します。

3. 1　ファイル構成

ダウンロードする圧縮ファイルのファイル構成は，次のようになっています。

ファイル名	説 明	同封圧縮（*.zip）ファイル	
			
main.asm （アセンブリ言語版） main.c （C言語版）	マイコンのハードウエア初期化処理とメイン処理のソース・ファイル	●注	●注
op.asm	オプション・バイト設定用アセンブラ・ソース・ファイル （システム・クロック・ソースなどを設定）	●	●
uart6.prw	統合開発環境 PM+用ワーク･スペース・ファイル		●
uart6.prj	統合開発環境 PM+用プロジェクト・ファイル		●

注　アセンブリ言語版には「main.asm」，C言語版には「main.c」が同封されています。

備考　  　：ソース・ファイルのみ同封

 　：統合開発環境 PM+で使用するファイルを同封

3. 2　使用する内蔵周辺機能

このサンプル・プログラムでは，マイコンに内蔵する次の周辺機能を使用します。

- シリアル通信　　：シリアル・インタフェースUART6
- V_{DD}＜V_{LVI}検出　　：低電圧検出（LVI）回路
- データ入力，出力　　：RxD6, TxD6

3. 3　初期設定と動作概要

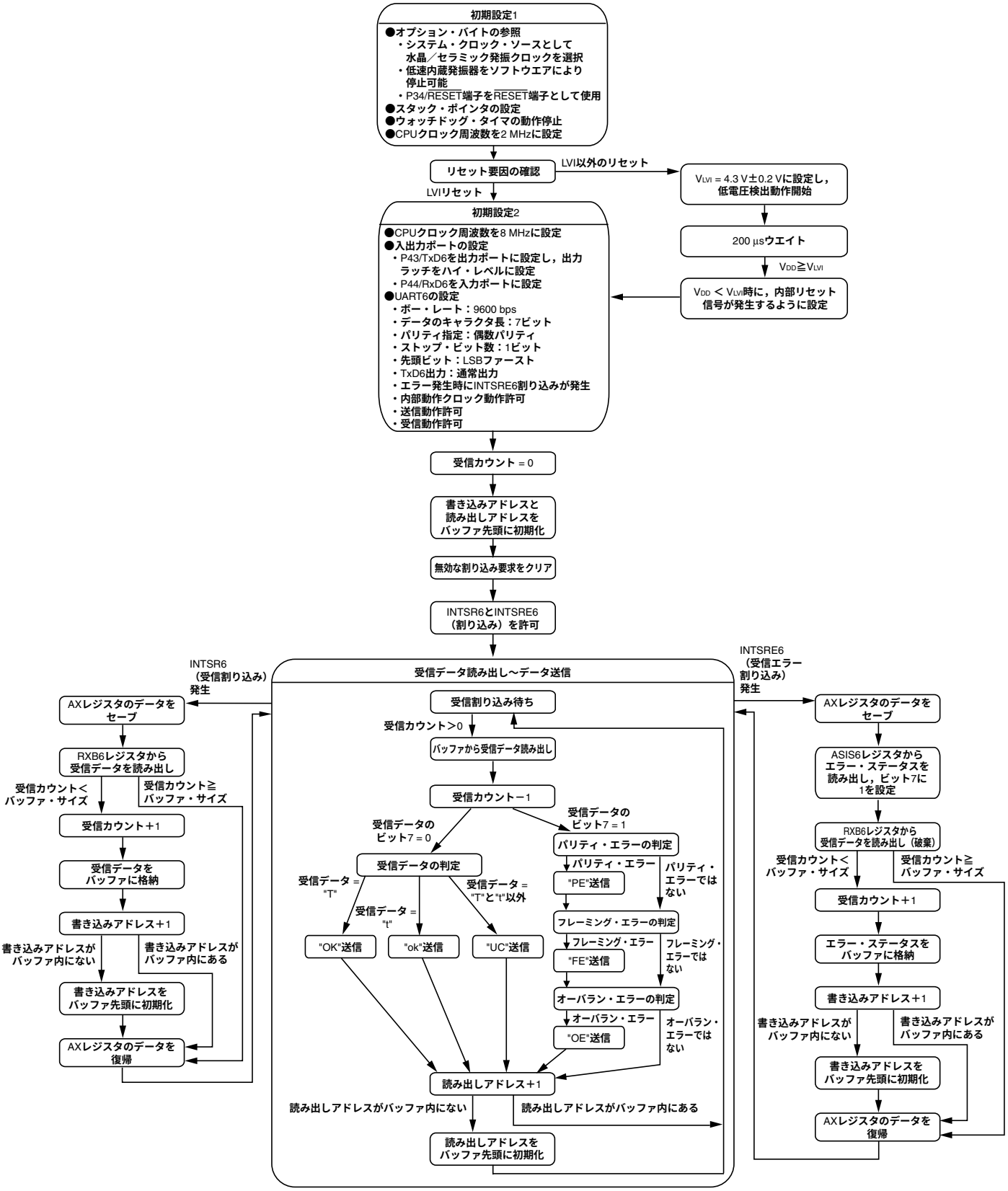
このサンプル・プログラムでは，初期設定にて，低電圧検出機能の設定，クロック周波数の選択，入出力ポートの設定，シリアル・インタフェースUART6の設定などを行います。

初期設定完了後は，RxD6端子からのデータ入力により，シリアル通信の受信動作を開始します。このサンプル・プログラムでは，ASCIIコードの送受信を想定しており，1キャラクタのデータ受信に応じて，4キャラクタのデータ送信を行います。受信エラー時の場合も同様に，エラー内容に応じて，4キャラクタのデータ送信を行います。

RAM領域には受信データの格納用バッファとして，50バイト（= 1バイト（1データ）×50）の領域を確保しています。

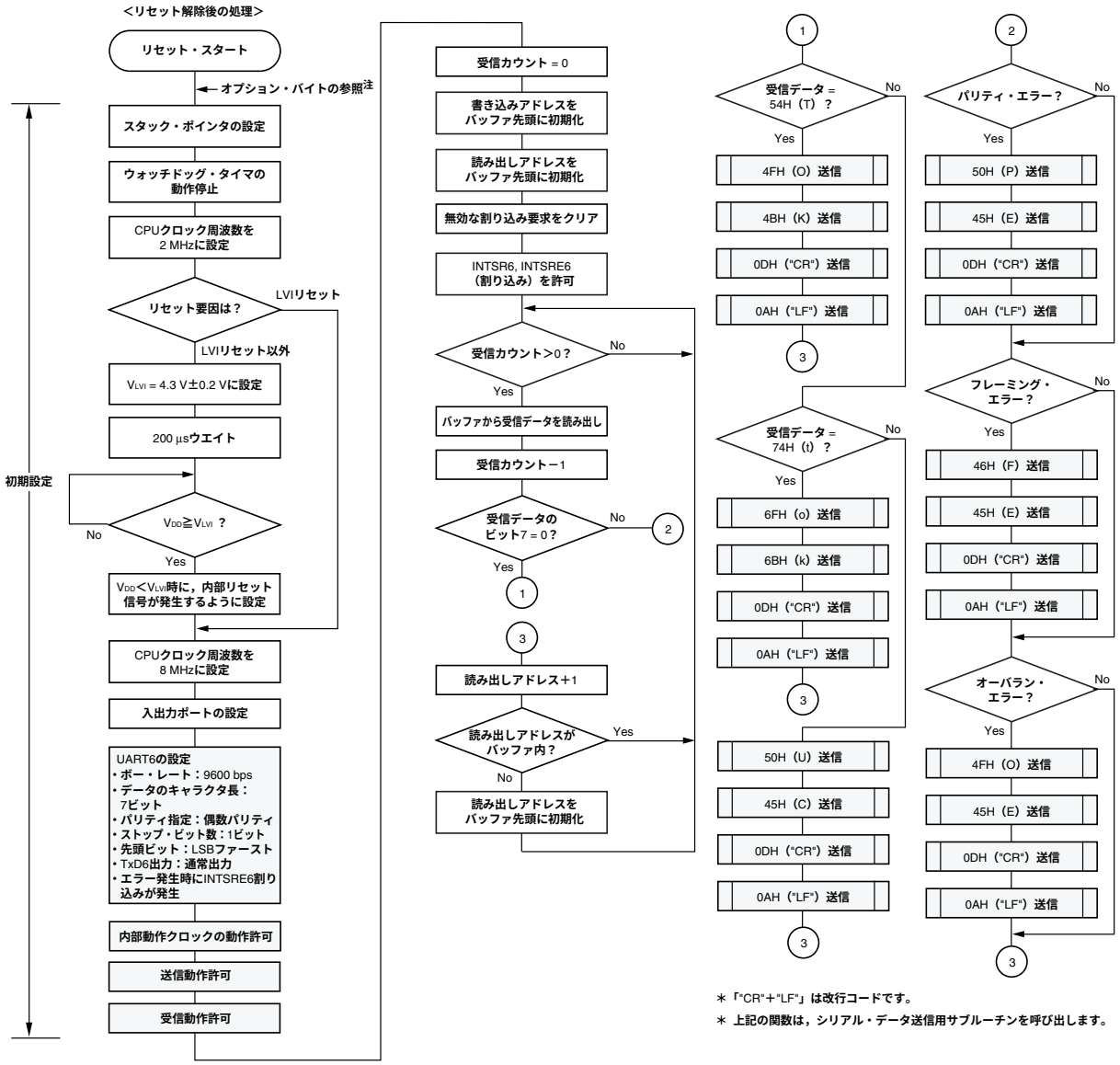
受信データは，割り込み処理時にバッファに格納するため，連続受信が可能で，バッファ領域の先頭から順次格納していきます。また，バッファをリング・バッファ構成とするため，バッファ領域の最終尾に受信データが達したあとは，再びバッファ領域の先頭から格納していきます。受信データをバッファに格納するときに，バッファに空きがあれば，受信データを格納しますが，バッファに空きがなければ，受信データを格納せずに破棄します。

詳細については，次の状態遷移図（ステート・チャート）に示します。



3. 4　フロー・チャート

このサンプル・プログラムのフロー・チャートを次に示します。



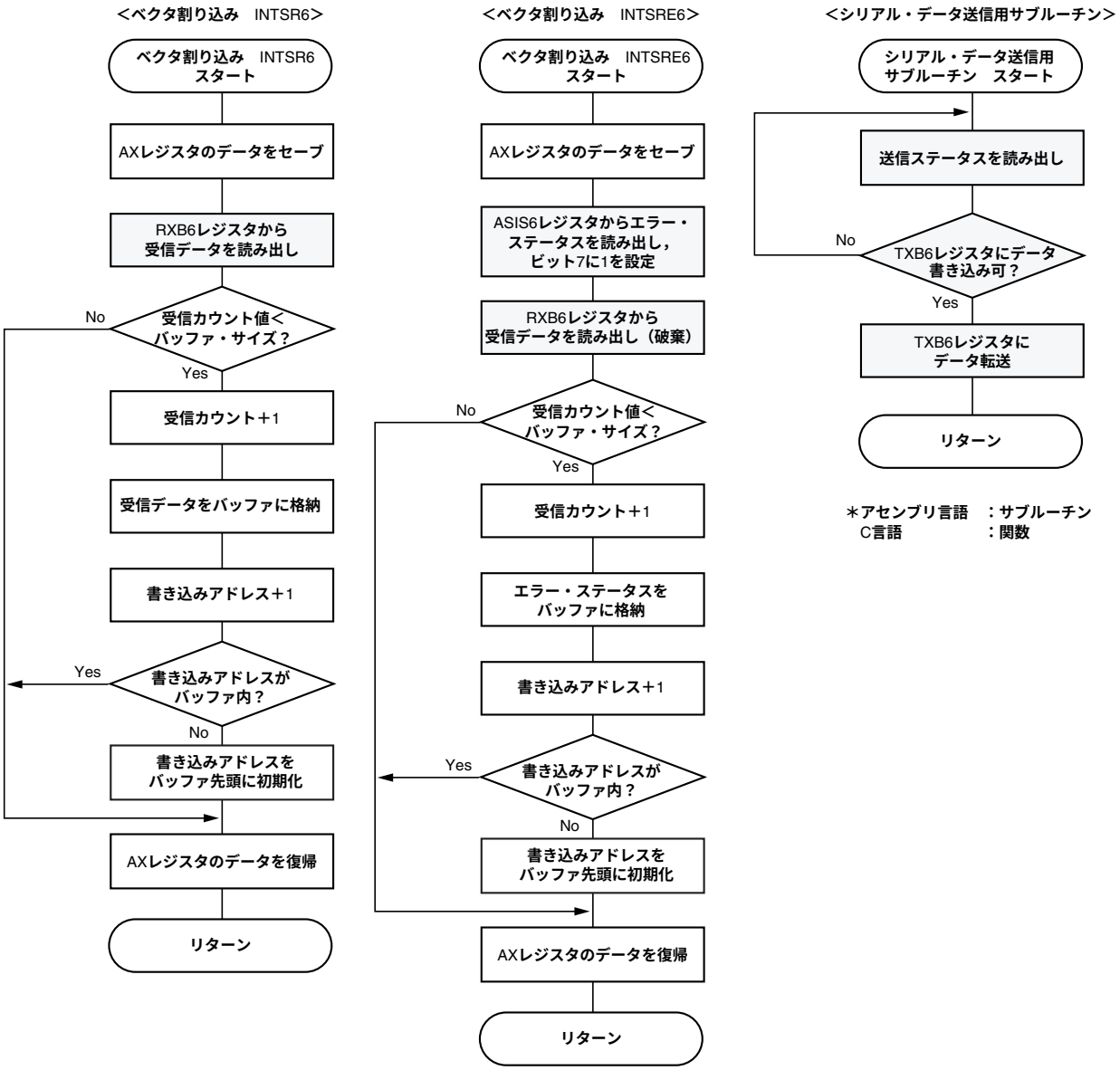
＊「"CR"＋"LF"」は改行コードです。

＊ 上記の関数は，シリアル・データ送信用サブルーチンを呼び出します。

注　オプション・バイトの参照は，リセット解除後に，マイコンが自動的に行います。このサンプル・プログラムでは，オプション・バイトの参照により，次の内容が設定されます。

- **システム・クロック・ソースとして，水晶／セラミック発振クロックを使用**
- **低速内蔵発振器をソフトウエアで停止可**
- **P34/$\overline{\text{RESET}}$端子を$\overline{\text{RESET}}$端子として使用**

備考　＜ベクタ割り込み INTSR6**＞，＜ベクタ割り込み** INTSRE6**＞，＜シリアル・データ送信用サブルーチン＞のフロー・チャートは，次ページにあります。**



＊アセンブリ言語　：サブルーチン
　C言語　　：関数

第4章　設定方法について

この章では，シリアル・インタフェースUART6の設定について説明します。

その他の初期設定については，78K0S/Kx1+　サンプル・プログラム（初期設定）　LED点灯のスイッチ制御編　アプリケーション・ノートを，割り込みについては，78K0S/Kx1+　サンプル・プログラム（割り込み）　スイッチ入力による外部割り込み編　アプリケーション・ノートを，低電圧検出（LVI）については，78K0S/Kx1+　サンプル・プログラム（低電圧検出）　2.7 V未満検出時リセット発生編　アプリケーション・ノートを参照してください。

レジスタ設定方法の詳細については，各製品のユーザーズ・マニュアル（78K0S/KA1+，78K0S/KB1+）を参照してください。

アセンブラ命令については，78K/0Sシリーズ　命令編　ユーザーズ・マニュアルを参照してください。

4. 1　シリアル・インタフェースUART6の設定

シリアル・インタフェースUART6は，主に次の11種類のレジスタを使用します。

- クロック選択レジスタ6（CKSR6）
- ボー・レート・ジェネレータ・コントロール・レジスタ6（BRGC6）
- アシンクロナス・シリアル・インタフェース動作モード・レジスタ6（ASIM6）
- アシンクロナス・シリアル・インタフェース・コントロール・レジスタ6（ASICL6）
- 送信バッファ・レジスタ6（TXB6）
- 受信バッファ・レジスタ6（RXB6）
- アシンクロナス・シリアル・インタフェース送信ステータス・レジスタ6（ASIF6）
- アシンクロナス・シリアル・インタフェース受信エラー・ステータス・レジスタ6（ASIS6）
- 入力切り替え制御レジスタ（ISC）
- ポート・モード・レジスタx（PMx）注1
- ポート・レジスタx（Px）注1

注1. シリアル・インタフェースUART6で使用する端子は，次のように設定してください。

POWER6	TXE6	RXE6	PM43	P43	PM44	P44	UART6の動作	端子機能	
								TxD6/INTP1/P43	RxD6/P44
0	0	0	×注2	×注2	×注2	×注2	停止	P43	P44
1	0	1	×注2	×注2	1	×	受信	P43	RxD6
	1	0	0	1	×注2	×注2	送信	TxD6	P44
	1	1	0	1	1	×	送受信	TxD6	RxD6

2. ポート機能として設定することができます。

備考　×　：don't care
POWER6　：ASIM6レジスタのビット7
TXE6　：ASIM6レジスタのビット6
RXE6　：ASIM6レジスタのビット5

＜シリアル・インタフェースUART6**の基本的な動作設定手順例＞**

① CKSR6**レジスタで**UART6**の基本クロック（**f_{XCLK6}**）を設定**
② BRGC6**レジスタで**UART6**の基本クロック（**f_{XCLK6}**）の分周値を設定**
} **ボー・レート設定**

③ ASIM6**レジスタで，パリティ，キャラクタ長，ストップ・ビット，エラー割り込みを設定**
④ ASICL6**レジスタで，先頭ビット，**TxD6**出力反転の許可／禁止を設定**
⑤ POWER6**をセット（1）：内部動作クロックの動作許可**
⑥ TXE6**をセット（1）：送信動作許可**
⑦ RXE6**をセット（1）：受信動作許可**

注意1. **送信を開始する場合は，⑥のあと，**UART6**の基本クロック（**f_{XCLK6}**）**1**クロック以上待ってから，**TXB6**レジスタに送信データを書き込んでください。**

2. **⑦のあと，**UART6**の基本クロック（**f_{XCLK6}**）**1**クロック経過後に，受信許可状態になります。**

3. PMx**レジスタと**Px**レジスタの設定手順は，通信相手との関係を考慮して行ってください。また，**TxD6**端子を使用する場合は，意図しないスタート・ビット（立ち下がり信号）を発生させないために，**P43**を**1**に設定したあとに，**PM43**を**0**（出力）に設定してください。**

（1**）**CKSR6**レジスタの設定**

CRSR6**レジスタは，シリアル・インタフェース**UART6**の基本クロック（**f_{XCLK6}**）を選択するレジスタです。**

図4－1　**クロック選択レジスタ**6（CKSR6）**のフォーマット**

CKSR6

0	0	0	0	TPS63	TPS62	TPS61	TPS60

基本クロック（f_{XCLK6}**）選択**

0	0	0	0	f_{XP}
0	0	0	1	$f_{XP}/2$
0	0	1	0	$f_{XP}/2^2$
0	0	1	1	$f_{XP}/2^3$
0	1	0	0	$f_{XP}/2^4$
0	1	0	1	$f_{XP}/2^5$
0	1	1	0	$f_{XP}/2^6$
0	1	1	1	$f_{XP}/2^7$
1	0	0	0	$f_{XP}/2^8$
1	0	0	1	$f_{XP}/2^9$
1	0	1	0	$f_{XP}/2^{10}$
1	0	1	1	$f_{XP}/2^{11}$
上記以外				**設定禁止**

注意　TPS63-TPS60**を書き換える場合は，**POWER6 = 0**に設定してから行ってください。**

備考1. **通信動作中（**POWER6 = 1**かつ**TXE6 = 1**，または**POWER6 = 1**かつ**RXE6 = 1**）に，ソフトウエアで**CKSR6**レジスタへのリフレッシュ（同値書き込み）動作を行うことができます。**

2. f_{XP}**：周辺ハードウエアへのクロックの発振周波数**

(2) BRGC6レジスタの設定

BRGC6レジスタは，シリアル・インタフェースUART6の基本クロック（f_{XCLK6}）の分周値を選択するレジスタです。

図4－2　ボー・レート・ジェネレータ・コントロール・レジスタ6（BRGC6）のフォーマット

BRGC6

MLD67	MLD66	MLD65	MLD64	MLD63	MLD62	MLD61	MLD60

MLD67	MLD66	MLD65	MLD64	MLD63	MLD62	MLD61	MLD60	k	8ビット・カウンタの出力クロック選択
0	0	0	0	0	x	x	x	x	設定禁止
0	0	0	0	1	0	0	0	8	f_{XCLK6}/8
0	0	0	0	1	0	0	1	9	f_{XCLK6}/9
0	0	0	0	1	0	1	0	10	f_{XCLK6}/10
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
1	1	1	1	1	1	0	0	252	f_{XCLK6}/252
1	1	1	1	1	1	0	1	253	f_{XCLK6}/253
1	1	1	1	1	1	1	0	254	f_{XCLK6}/254
1	1	1	1	1	1	1	1	255	f_{XCLK6}/255

注意1. MLD67-MLD60を書き換える場合は，TXE6 = 0, RXE6 = 0に設定してから行ってください。

2. 8ビット・カウンタの出力クロックをさらに1/2分周したものが，ボー・レート値となります。

・ボー・レート = $\frac{f_{XCLK6}}{2 \times k}$ [bps]

備考1. 通信動作中（POWER6 = 1かつTXE6 = 1，またはPOWER6 = 1かつRXE6 = 1）に，ソフトウエアでBRGC6レジスタへのリフレッシュ（同値書き込み）動作を行うことができます。

2. f_{XCLK6}： TPS63-TPS60で選択した基本クロックの周波数

3. k：　MLD67-MLD60で設定した値（k = 8, 9, 10, …, 255）

4. x：　任意

(3) ASIM6レジスタの設定

ASIM6レジスタは，シリアル・インタフェースUART6のシリアル通信動作を制御するレジスタです。

図4－3　アシンクロナス・シリアル・インタフェース動作モード・レジスタ6（ASIM6）のフォーマット

ASIM6

POWER6	TXE6[注1]	RXE6[注2]	PS61	PS60	CL6	SL6	ISRM6

エラー発生時の割り込み指定

0	INTSRE6
1	INTSR6（受信完了時の割り込みと同じ割り込み）

送信データのストップ・ビット数指定

0	ストップ・ビット数 = 1
1	ストップ・ビット数 = 2

送受信データのキャラクタ長指定

0	データのキャラクタ長 = 7ビット
1	データのキャラクタ長 = 8ビット

パリティ・ビットの指定

		送信動作	受信動作
0	0	パリティ・ビットを出力しない	パリティなしで受信
0	1	0パリティ出力	0パリティとして受信[注3]
1	0	奇数パリティ出力	奇数パリティとして判定
1	1	偶数パリティ出力	偶数パリティとして判定

受信動作許可／禁止

0	受信動作禁止（受信回路を同期リセットする）
1	受信動作許可

送信動作許可／禁止

0	送信動作禁止（送信回路を同期リセットする）
1	送信動作許可

内部動作クロックの動作許可／禁止

0[注4]	内部動作クロックの動作禁止（ロウ・レベル固定），内部回路を非同期リセットする[注5]
1[注6]	内部動作クロックの動作許可

備考　通信動作中（POWER6 = 1かつTXE6 = 1，またはPOWER6 = 1かつRXE6 = 1）に，ソフトウエアでASIM6レジスタへのリフレッシュ（同値書き込み）動作を行うことができます。

（注，注意は次ページにあります）

注1. TXE6**は，**CKSR6**レジスタで設定した基本クロック（**f_{XCLK6}**）により，同期化されています。再び送信動作を許可する場合は，**TXE6 = 0**に設定してから基本クロック（**f_{XCLK6}**）1クロック以降に**TXE6 = 1**を設定してください。基本クロック（**f_{XCLK6}**）1クロック以内に設定すると，送信回路を初期化できない場合があります。**

2. RXE6**は，**CKSR6**レジスタで設定した基本クロック（**f_{XCLK6}**）により，同期化されています。再び受信動作を許可する場合は，**RXE6 = 0**に設定してから基本クロック（**f_{XCLK6}**）1クロック以降に**RXE6 = 1**を設定してください。基本クロック（**f_{XCLK6}**）1クロック以内に設定すると，受信回路を初期化できない場合があります。**
3. **「0パリティとして受信」を設定すると，パリティ判定を行いません。したがって，**ASIS6**レジスタの**PE6**フラグはセットされないため，エラー割り込みも発生しません。**
4. **送信中に**POWER6 = 0**にすると，**TxD6**端子の出力はハイ・レベルになり，**RxD6**端子からの入力はハイ・レベルに固定されます。**
5. **リセットされるのは，**ASIS6**レジスタ，**ASIF6**レジスタ， **ASICL6**レジスタの**SBRF6**と**SBRT6**，**RXB6**レジスタです。**
6. POWER6 = 1**に設定し，基本クロック（**f_{XCLK6}**）が1クロック経過したあと，内部動作クロックとして，基本クロック（**f_{XCLK6}**）が供給されます。**

注意1. **起動時は**POWER6 = 1**に設定してから**TXE6 = 1**に設定したあと，基本クロック（**f_{XCLK6}**）1クロック以上待ってから**TXB6**レジスタに送信データを設定すると，送信動作を開始します。送信動作を停止するときには**TXE6 = 0**に設定してから，**POWER6 = 0**に設定してください。**

2. **起動時は**POWER6 = 1**に設定してから**RXE6 = 1**に設定したあと，基本クロック（**f_{XCLK6}**）1クロック経過後に，受信許可状態になります。受信動作を停止するときには**RXE6 = 0**に設定してから，**POWER6 = 0**に設定してください。**
3. RxD6**端子にハイ・レベルが入力された状態で**POWER6 = 1→RXE6 = 1 **と設定してください。ロウ・レベルのときに**POWER6 = 1→RXE6 = 1 **と設定すると，受信を開始し，正常なデータが受信されません。**
4. PS61, PS60, CL6**を書き換えるときは，**TXE6, RXE6**をクリア（0）してから行ってください。**
5. LIN**通信動作で使用する場合，**PS61, PS60**を0に固定してください。**
6. SL6**を書き換えるときは，**TXE6 = 0**にしてから行ってください。また，受信は常に“ストップ・ビット数 = 1”として動作するので，**SL6**の設定値の影響は受けません。**
7. ISRM6**を書き換えるときは，**RXE6 = 0**にしてから行ってください。**

(4) ASICL6レジスタの設定

ASICL6レジスタは，シリアル・インタフェースUART6のシリアル通信動作を制御するレジスタです。

図4－4　アシンクロナス・シリアル・インタフェース・コントロール・レジスタ6（ASICL6）のフォーマット

ASICL6

SBRF6	SBRT6	SBTT6	SBL62	SBL61	SBL60	DIR6	TXDLV6

TxD6出力反転許可／禁止

0	TxD6通常出力
1	TxD6反転出力

先頭ビットの指定

0	MSBファースト
1	LSBファースト

SBF送信出力幅制御

1	0	1	SBFは13ビット長で出力
1	1	0	SBFは14ビット長で出力
1	1	1	SBFは15ビット長で出力
0	0	0	SBFは16ビット長で出力
0	0	1	SBFは17ビット長で出力
0	1	0	SBFは18ビット長で出力
0	1	1	SBFは19ビット長で出力
1	0	0	SBFは20ビット長で出力

SBF送信トリガ

0	－
1	SBF送信トリガ

SBF受信トリガ

0	－
1	SBF受信トリガ

SBF受信状態フラグ（リード・オンリー）

0	POWER6 = 0およびRXE6 = 0に設定したとき，またはSBF受信が正常の終了したとき
1	SBF受信中

備考　SBRT6とSBTT6が0のデータをASICL6に書き込んだ場合，通信動作中（POWER6 = 1かつTXE6 = 1，またはPOWER6 = 1かつRXE6 = 1）に，ソフトウエアでASICL6レジスタへのリフレッシュ（同値書き込み）動作を行うことができます。

（注意は次ページにあります）

注意1. LIN通信動作で使用するときに，SBF送信およびSBF受信を使用します。SBF送信およびSBF受信の詳細については，各製品のユーザーズ・マニュアル（78K0S/KA1+, 78K0S/KB1+）を参照してください。

2. SBF受信エラー時は，再びSBF受信モードに戻ります。SBRF6フラグの状態は保持（1）されます。
3. SBRT6は，POWER6 = 1，かつRXE6 = 1に設定してからセット（1）してください。また，セット（1）後，SBF受信が終了（割り込み要求信号が発生）する前に，SBRT6をクリア（0）しないでください。
4. SBRT6のリード値は常に0です。SBF受信正常終了後，SBRT6は自動的にクリア（0）されます。
5. SBTT6は，POWER6 = 1，かつTXE6 = 1に設定してからセット（1）してください。また，セット（1）後，SBF送信が終了（割り込み要求信号が発生）する前に，SBTT6をクリア（0）しないでください。
6. SBTT6のリード値は常に0です。SBF送信終了後，SBTT6は自動的にクリア（0）されます。
7. DIR6, TXDLV6を書き換えるときは，TXE6, RXE6をクリア（0）にしてから行ってください。

（5）TXB6レジスタの動作

TXB6レジスタは，シリアル・インタフェースUART6の送信データを設定するバッファ・レジスタです。

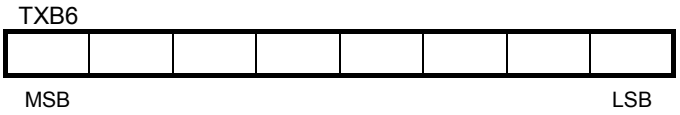
TXB6レジスタへ送信データを書き込むことにより，送信動作が開始されます。

データ長を7ビットに指定した場合は次のようになります。

・LSBファースト送信時では，TXB6のビット0-6が転送され，TXB6のMSBは送信されません。

・MSBファースト送信時では，TXB6のビット7-1が転送され，TXB6のLSBは送信されません。

図4－5　送信バッファ・レジスタ6（TXB6）のフォーマット



注意1. 送信を開始する場合は，TXE6 = 1に設定したあと，基本クロック（f_{XCLK6}）1クロック以上待ってから，TXB6に送信データを書き込んでください。

2. ASIF6レジスタのTXBF6が1のとき，TXB6レジスタにデータを書き込まないでください。
3. 通信動作中（POWER6 = 1かつTXE6 = 1，またはPOWER6 = 1かつRXE6 = 1）に，ソフトウエアでTXB6レジスタへのリフレッシュ（同値書き込み）動作を行わないでください。連続送信で同値を出力する場合は，必ずTXBF6が0であることを確認してから，TXB6レジスタに同値を書き込んでください。

【コラム】LINとは

Local Interconnect Networkの略称で，車載ネットワークのコストダウンを目的とする低速（1～20 kbps）のシリアル通信プロトコルです。シリアル・インタフェースUART6は，LIN-busに対応しています。

LINの通信はシングル・マスタ通信で，1つのマスタに対し最大15のスレーブが接続可能です。

LINのスレーブは，スイッチ，アクチュエータ，センサなどの制御に使用され，これらがLINのネットワークを介してLINのマスタに接続されます。

LINのマスタは通常，CAN（Controller Area Network）などのネットワークに接続されます。

また，LINバスはシングル・ワイヤ方式で，ISO9141に準拠したトランシーバを介して各ノードが接続されます。

LINのプロトコルでは，マスタはフレームにボー・レート情報をつけて送信し，スレーブはこれを受信してマスタとのボー・レート誤差を補正します。マスタとスレーブのボー・レート誤差が±15 %以下であれば，補正可能です。

(6) RXB6レジスタの動作

RXB6レジスタは，シリアル・インタフェースUART6の受信データを格納するバッファ・レジスタです。データを1バイト受信するごとに，受信データが転送されます。

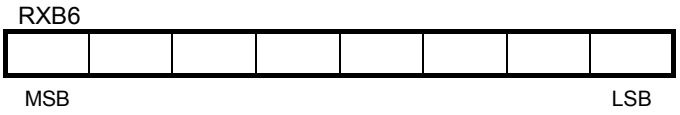
データ長を7ビットに指定した場合は次のようになります。

・LSBファースト受信時では，受信データはRXB6のビット0-6に転送され，RXB6のMSBは必ず0になります。

・MSBファースト受信時では，受信データはRXB6のビット7-1に転送され，RXB6のLSBは必ず0になります。

オーバラン・エラー（OVE6）が発生した場合，そのときの受信データはRXB6には転送されません。

図4－6　受信バッファ・レジスタ6（RXB6）のフォーマット



注意1. RXE6 = 1**に設定したあと，基本クロック（**f_{XCLK6}**）1クロック経過後に，受信許可状態になります。**

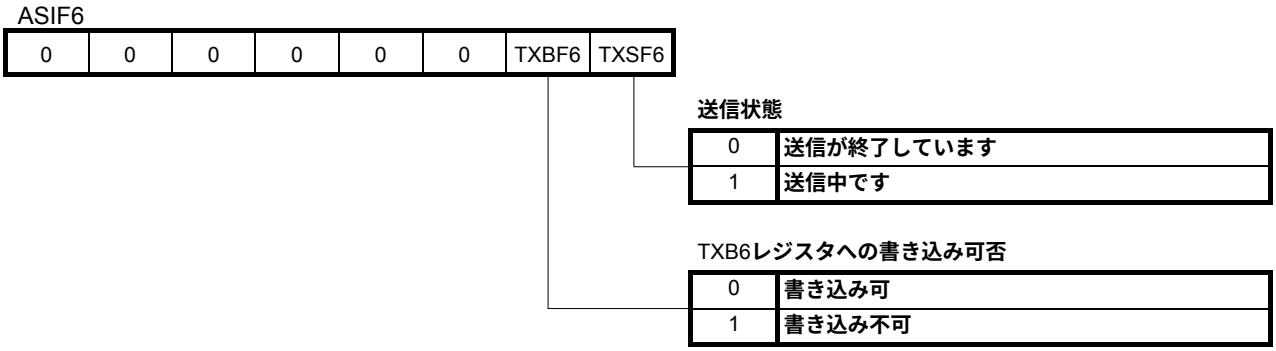
2. **受信エラー発生時にも**RXB6**レジスタは必ず読み出してください。**RXB6**レジスタを読み出さないと，次のデータ受信時にオーバラン･エラーが発生し，いつまでも受信エラーの状態が続いてしまいます。**

(7) ASIF6レジスタの動作

ASIF6レジスタは，シリアル・インタフェースUART6の送信時のステータスを示すリード・オンリーのレジスタです。

TXB6レジスタのデータ転送をASIF6レジスタで確認し，次のデータをTXB6レジスタに書き込むことで，割り込み期間中も途切れることなく送信を続けることができます。

図4－7　アシンクロナス・シリアル・インタフェース送信ステータス・レジスタ6（ASIF6）のフォーマット



注意1. **連続送信を行う場合は，最初の送信データ（1バイト目）を**TXB6**レジスタに書き込んだあと，必ず**TXBF6**が“**0**”であることを確認してから次の送信データ（2バイト目）を**TXB6**レジスタに書き込んでください。**TXBF6**が“**1**”のときに**TXB6**レジスタにデータを書き込んだ場合の送信データは保証できません。**

2. **連続送信完了時に送信ユニットを初期化する場合は，送信完了割り込み発生後に，必ず**TXSF6**が“**0**”であることを確認してから初期化を実行してください。**TXSF6**が“**1**”のときに初期化を実行した場合の送信データは保証できません。**

(8) ASIS6レジスタの動作

ASIS6レジスタは，シリアル・インタフェースUART6の受信終了時のエラー・ステータスを示すリード・オンリーのレジスタです。

図4－8　アシンクロナス・シリアル・インタフェース受信エラー・ステータス・レジスタ6（ASIS6）のフォーマット

ASIS6

0	0	0	0	0	PE6	FE6	OVE6

オーバラン・エラーを示すステータス・フラグ

0	POWER6 = 0およびRXE6 = 0に設定したとき，またはASIS6レジスタのリード
1	RXB6レジスタに受信データがセットされ，それを読み出す前に次の受信動作が完了したとき（オーバラン・エラー）

フレーミング・エラーを示すステータス・フラグ

0	POWER6 = 0およびRXE6 = 0に設定したとき，またはASIS6レジスタのリード
1	受信完了時，ストップ・ビットが検出されないとき（フレーミング・エラー）

パリティ・エラーを示すステータス・フラグ

0	POWER6 = 0およびRXE6 = 0に設定したとき，またはASIS6レジスタのリード
1	受信完了時，送信データのパリティとパリティ・ビットが一致しないとき（パリティ・エラー）

注意1. PE6の動作は，ASIM6レジスタのPS61, PS60の設定値により異なります。

2. 受信データのストップ・ビットはストップ・ビット数に関係なく最初の1ビットだけをチェックします。
3. オーバラン・エラーが発生した場合，次の受信データはRXB6レジスタには書き込まれず，データは破棄されます。
4. RXB6レジスタを読み出す前に，必ずASIS6レジスタを読み出してください。

(9) ISCレジスタの設定

ISCレジスタは，LIN受信時に，マスタから送信されるステータス信号を受信するときに設定するレジスタです。ISCレジスタの設定により，RxD6端子からの入力信号を，外部割り込み（INTP0）および16ビット・タイマ／イベント・カウンタ00へ入力することができます。

図4－9　入力切り替え制御レジスタ（ISC）のフォーマット

ISC

0	0	0	0	0	0	ISC1	ISC0

INTP0入力ソースの選択

0	INTP0（P30）
1	RxD6（P44）

TI000入力ソースの選択

0	TI000（P30）
1	RxD6（P44）

注意　LIN送信およびLIN受信の詳細については，各製品のユーザーズ・マニュアル（78K0S/KA1+, 78K0S/KB1+）を参照してください。

【　例　】シリアル・インタフェースUART6を次のように設定して，シリアル送受信動作を開始する場合

- **ボー・レート：** 9600 bps
- **データのキャラクタ長：** 7**ビット**
- **パリティ・ビット指定：** **偶数パリティ**
- **ストップ・ビット数：** 1**ビット**
- **先頭ビット：** LSB**ファースト**
- TxD6**出力：** **通常出力**
- **エラー発生時の割り込み：** INTSRE6

（周辺ハードウエアへのクロックの発振周波数（f_{XP}）= 8 MHz，LIN通信は行わない）

（本サンプル・プログラム・ソースと同内容）

（1）レジスタの設定内容

① CKSR6

0	0	0	0	0	0	0	1

基本クロック（f_{XCLK6}）選択

0	0	0	1	f_{XP}/2

- **基本クロック（f_{XCLK6}）** = f_{XP}/2 = 8 MHz/2 = 4 MHz

② BRGC

1	1	0	1	0	0	0	0

k	8ビット・カウンタの出力クロック選択
208	f_{XCLK6}/208

- **ボー・レート** = $\frac{f_{XCLK6}}{2 \times k}$ [bps] = $\frac{4\ \text{MHz}}{2 \times 208}$ [bps] = $\frac{4000000}{2 \times 208}$ [bps]　　9600 [bps]

③ ASIM6

POWER6	TXE6	RXE6	1	1	0	0	0

エラー発生時の割り込み指定

0	INTSRE6

送信データのストップ・ビット数指定

0	ストップ・ビット数 = 1

送受信データのキャラクタ長指定

0	データのキャラクタ長 = 7ビット

パリティ・ビットの指定

		送信動作	受信動作
1	1	偶数パリティ出力	偶数パリティとして判定

受信動作許可／禁止

0	受信動作禁止（受信回路を同期リセットする）
1	受信動作許可

送信動作許可／禁止

0	送信動作禁止（送信回路を同期リセットする）
1	送信動作許可

内部動作クロックの動作許可／禁止

0	内部動作クロックの動作禁止（ロウ・レベル固定），内部回路を非同期リセットする
1	内部動作クロックの動作許可

④ ASICL6：**初期設定のまま使用（先頭ビット：LSBファースト，TxD6出力：通常出力）**

⑤ PMx, Px

PM43	P43	PM44	P44	UART6の動作	端子機能	
					TxD6/INTP1/P43	RxD6/P44
0	1	1	×	送受信	TxD6	RxD6

備考　×　：don't care

(2) サンプル・プログラム

①　アセンブリ言語の場合

```
SET1   P4.3
CLR1   PM4.3
SET1   PM4.4
MOV    CKSR6  #1
MOV    BRGC6  #208
MOV    ASIM6, #00011000B
SET1   POWER6
SET1   TXE6
SET1   RXE6
```

②　C言語の場合

```
P4.3 = 1;
PM4.3 = 0;
PM4.4 = 1;
CKSR6 = 1;
BRGC6 = 208;
ASIM6 = 0b00011000;
POWER6 = 1;
TXE6 = 1;
RXE6 = 1;
```

【本サンプル・プログラム・ソースからの抜粋】

付録A　プログラム・リストから，シリアル・インタフェースUART6の機能に関する抜粋部分を示します（前述の【　例　】と同内容）。

（1）アセンブリ言語

```
XMAIN  CSEG   UNIT
IRESET:
                    .
                    .
                    .
        MOV     CKSR6, #1           ; ボー・レートを9600bpsに設定
        MOV     BRGC6, #208         ; （同上）
        MOV     ASIM6, #00011000B   ; 偶数パリティ出力、キャラクタ長7bit、ストップビット数1
                                    ; エラー発生時の割り込みにINTSRE6が発生
        SET1    POWER6              ; 内部動作クロックの動作許可
        SET1    TXE6                ; 送信動作許可
        SET1    RXE6                ; 受信動作許可

MMAINLOOP:
                    .
                    .
                    .
        MOV     IF0,    #00H        ; 無効割り込み要求をクリアしておく
        CLR1    SRMK6               ; INTSR6（シリアル受信）割り込み許可
        CLR1    SREMK6              ; INTSRE6（受信エラー）割り込み許可
        EI                          ; ベクタ割り込み許可
                    .
                    .
                    .
IINTSR6:
        PUSH    AX                  ; AXレジスタのデータをスタックへ退避
        MOV     A, RXB6             ; シリアル受信データを読み出し
                    .
                    .
                    .
IINTSRE6:
        PUSH    AX                  ; AXレジスタのデータをスタックへ退避
        MOV     A, ASIS6            ; エラー・ステータスを読み出し
        SET1    A.7                 ; ビット7に受信エラー・フラグをセット
        XCH     A,      X           ; エラー情報を退避
        MOV     A, RXB6             ; シリアル受信データを読み出し（破棄）
        XCH     A,      X           ; エラー情報を復帰
                    .
                    .
                    .
JTXS100:
        MOV     A, ASIF6
        BT      A.1, $JTXS100       ; 送信許可待ち
        XCH     A,      X           ; 送信データをXレジスタから復帰
        MOV     TXB6, A             ; シリアル送信
                    .
                    .
                    .
```

ボー・レートの設定

通信フォーマットとエラー割り込みの設定

動作クロックの動作許可

送信動作許可

受信動作許可

INTSR6（受信）割り込み処理を許可

INTSRE6（受信エラー）割り込み処理を許可

RXB6レジスタより受信データを読み出し

ASIS6レジスタよりエラー・ステータスを読み出してから，RXB6レジスタより受信データを読み出し

TXBF6フラグが0（送信可能）になったら，TXB6レジスタに送信データを書き込む→データ送信

(2) C言語

```
void hdwinit(void){
        unsigned char ucCnt200us;  /* 200usウェイト用8ビット変数 */
                        •
                        •
                        •
        CKSR6 = 1;                  /* ボー・レートを9600bpsに設定 */
        BRGC6 = 208;                /* （同上） */
        ASIM6 = 0b00011000;         /* 偶数パリティ出力、キャラクタ長7bit、ストップビット数1 */
                                    /* エラー発生時の割り込みにINTSRE6が発生 */
        POWER6 = 1;                 /* 内部動作クロックの動作許可 */
        TXE6  =1;                   /* 送信動作許可 */
        RXE6  =1;                   /* 受信動作許可 */
        return;
}

void main(void)
{
                        •
                        •
                        •
        IF0 = 0x00;                 /* 無効割り込み要求をクリアしておく */
        SRMK6 = 0;                  /* INTSR6(シリアル受信)割り込み許可 */
        SREMK6 = 0;                 /* INTSRE6(受信エラー)割り込み許可 */
        EI();                       /* ベクタ割り込み許可 */
                        •
                        •
                        •
}

__interrupt void fn_intsr6()
{
        unsigned char ucData;
        ucData = RXB6;                      /* シリアル受信データを読み出し */
                        •
                        •
                        •
}

__interrupt void fn_intsre6()
{
        unsigned char ucData;
        unsigned char ucTemp;
        ucData = ASIS6 | 0b10000000;        /* bit7にエラー・フラグをセットしてエラー情報を格納 */
        ucTemp = RXB6;                      /* シリアル受信データを読み出し(破棄) */
                        •
                        •
                        •
}

void fn_uart_send(unsigned char ucTxData)
{
        g_ucAsif6 = ASIF6;          /* 送信ステータス読み出し */
                while (g_ucAsif6.1)         /* 送信許可待ち */
        {
                g_ucAsif6 = ASIF6;  /* 送信ステータス読み出し */
        }
        TXB6 = ucTxData;            /* シリアル送信 */
        return;
}
```

ボー・レートの設定

通信フォーマットとエラー割り込みの設定

動作クロックの動作許可

送信動作許可

受信動作許可

INTSR6（受信）割り込み処理を許可

INTSRE6（受信エラー）割り込み処理を許可

RXB6レジスタより受信データを読み出し

ASIS6レジスタよりエラー・ステータスを読み出してから，RXB6レジスタより受信データを読み出し

TXBF6フラグが0（送信可能）になったら，TXB6レジスタに送信データを書き込む→データ送信

4.2　受信データまたは受信エラー内容と送信データ

このサンプル・プログラムでは，シリアル・インタフェースUART6でシリアル通信を行い，受信したデータまたは受信エラー内容に応じて，データを送信します。受信データはASCIIコードの1キャラクタを，送信データはASCIIコードの4キャラクタを想定しています。

データのキャラクタ長は7ビットに，先頭ビットはLSBファーストに設定しているので，受信が正常に終了した場合，受信データの先頭ビットは受信バッファ・レジスタ（RXB6）のビット0に転送され，ビット6まで順に転送されます。このとき，ビット7（MSB）は必ず0になります。このRXB6レジスタの受信データは，割り込み（INTSR6）処理時に，バッファに格納します。

受信エラー発生の場合には，割り込み（INTSRE6）処理時に，バッファのビット0-6にASIS6レジスタのエラー・ステータスを格納し，ビット7に1をセットします。

これにより，バッファ読み出し時には，ビット0-6のデータが受信データかエラー・ステータスかを，ビット7で判断してから，ビット0-6のデータの内容を判断します。

読み出した受信データまたは受信エラー内容に対応する送信データは，次のとおりです。

● 正常受信（ビット7が0）の場合

1キャラクタ受信データ（16進数データ）	4キャラクタ送信データ（16進数データ）			
T（54H）	O（4FH）	K（4BH）	“CR”（0DH）	“LF”（0AH）
t（74H）	o（6FH）	k（6BH）	“CR”（0DH）	“LF”（0AH）
上記以外	U（55H）	C（43H）	“CR”（0DH）	“LF”（0AH）

● エラー受信（ビット7が1）の場合

受信エラー内容（ASIS6レジスタのフラグの値）	4キャラクタ送信データ（16進数データ）			
パリティ・エラー（PE6が1の場合）	P（50H）	E（45H）	“CR”（0DH）	“LF”（0AH）
フレーミング・エラー（FE6が1の場合）	F（46H）	E（45H）	“CR”（0DH）	“LF”（0AH）
オーバラン・エラー（OVE6が1の場合）	O（4FH）	E（45H）	“CR”（0DH）	“LF”（0AH）

備考　“CR”と“LF”は制御文字です。“CR”と“LF”を合わせると，改行コードになります。

【コラム】マクロとサブルーチン

このアセンブリ言語のサンプル・プログラムでは，記述を簡潔にするために，マクロとサブルーチンを使用しています。マクロとサブルーチンを使用すると，処理プログラムの変更が1箇所で済み，かつプログラムを理解しやすくなるため，プログラムのメンテナンスが簡単になります。

① マクロ

使用頻度の高い処理プログラムなどに名前を定義し，この名前を使用することで，記述を簡略化する手法です。マクロを使用すると，余分なCALL, CALLT命令やRET命令が入らずに，マクロの参照箇所にマクロで定義したプログラムのコードが展開されるため，プログラム実行時間を短縮できます。また，パラメータ定義ができるので，パラメータを変更するだけで，類似処理プログラムを容易に定義できます。

② サブルーチン

まとまった処理プログラムを，メイン・ルーチンから参照できるように切り出すことで，記述を簡略化する手法です。

<本サンプル・プログラム・ソースからの抜粋>

マクロ名「_SEROUT」，仮パラメータ「RTXDATA」のマクロ定義

```
_SEROUT     MACRO  RTXDATA
    PUSH    AX                  ; AXレジスタのデータをスタックへ退避
    MOV     A,     RTXDATA      ; 仮パラメータRTXDATAをAレジスタに格納
    CALLT   [ZTXSUB]            ; UART送信用サブルーチンを実行
    POP     AX                  ; AXレジスタのデータを復帰
    ENDM
```

マクロの実行内容

マクロ定義の終了

サブルーチン参照

```
XCALTCSEG   CALLT0
ZTXSUB:     DW     STXSUB       ; UART送信用サブルーチン
```

サブルーチン「STXSUB」をCALLT命令で呼び出せるように，CALLT命令テーブル領域にアドレス「ZTXSUB」を登録

```
MMAINLOOP:
LMLP200:
    CMP     A, #'T'             ; 受信データをT(54H)と比較
    BNZ     $LMLP210            ; 受信データがT(54H)でなければ分岐
    SEROUT  #'O'                ; O(4FH)送信
    SEROUT  #'K'                ; K(4BH)送信
    SEROUT  #0DH                ; 改行コード"CR"送信
    SEROUT  #0AH                ; 改行コード"LF"送信
    BR      !LMLP400            ; LMLP400へ分岐
```

実パラメータ

マクロの参照（上記の4行が展開）

サブルーチン

```
STXSUB:
    XCH     A,     X            ; 送信データをXレジスタへ退避
JTXS100:
    MOV     A, ASIF6
    BT      A.1, $JTXS100       ; 送信許可待ち
    XCH     A,     X            ; 送信データをXレジスタから復帰
    MOV     TXB6, A             ; シリアル送信
    RET                         ; サブルーチンから復帰
```

サブルーチンの処理内容

メイン・ルーチンに戻る

第5章　デバイスでの動作確認

この章では，ダウンロードしたサンプル・プログラムを使用し，ビルドからデバイスでの動作確認までの流れを説明します。

★　5. 1　サンプル・プログラムのビルド

サンプル・プログラムのビルド方法について，  のアイコンからダウンロードしたサンプル・プログラム（ソース・ファイル＋プロジェクト・ファイル）を使用し，説明します。その他のダウンロードしたプログラムのビルド方法については，78K0S/Kx1+　サンプル・プログラム　スタートアップ・ガイド　アプリケーション・ノートのを参照してください。

また，PM+操作方法の詳細については，PM+ プロジェクト・マネージャ　ユーザーズ・マニュアルを参照してください。

【コラム】ビルドのエラー

PM+でビルドしているときに「A006 File not found 'C:¥NECTOOLS32¥LIB78K0S¥s0sl.rel'」または，「*** ERROR F206 Segment '@@DATA' can't allocate to memory - ignored.」というエラー・メッセージが出た場合，次の手順にてコンパイラオプションの設定を変更してください。

①［ツール］→［コンパイラオプションの設定］を選択してください。
②［コンパイラオプションの設定］ダイアログが開いたら，「スタートアップ・ルーチン」タグを選択してください。
③「標準ライブラリ固定領域を使用する」のチェックを外してください（それ以外のチェックは，そのまま）。

「標準ライブラリ固定領域を使用する」のチェックを外すと，標準ライブラリ固定領域として確保されていた118バイトのRAM領域が使用可能になりますが，標準ライブラリ（getchar関数やmalloc関数など）を使用できなくなります。

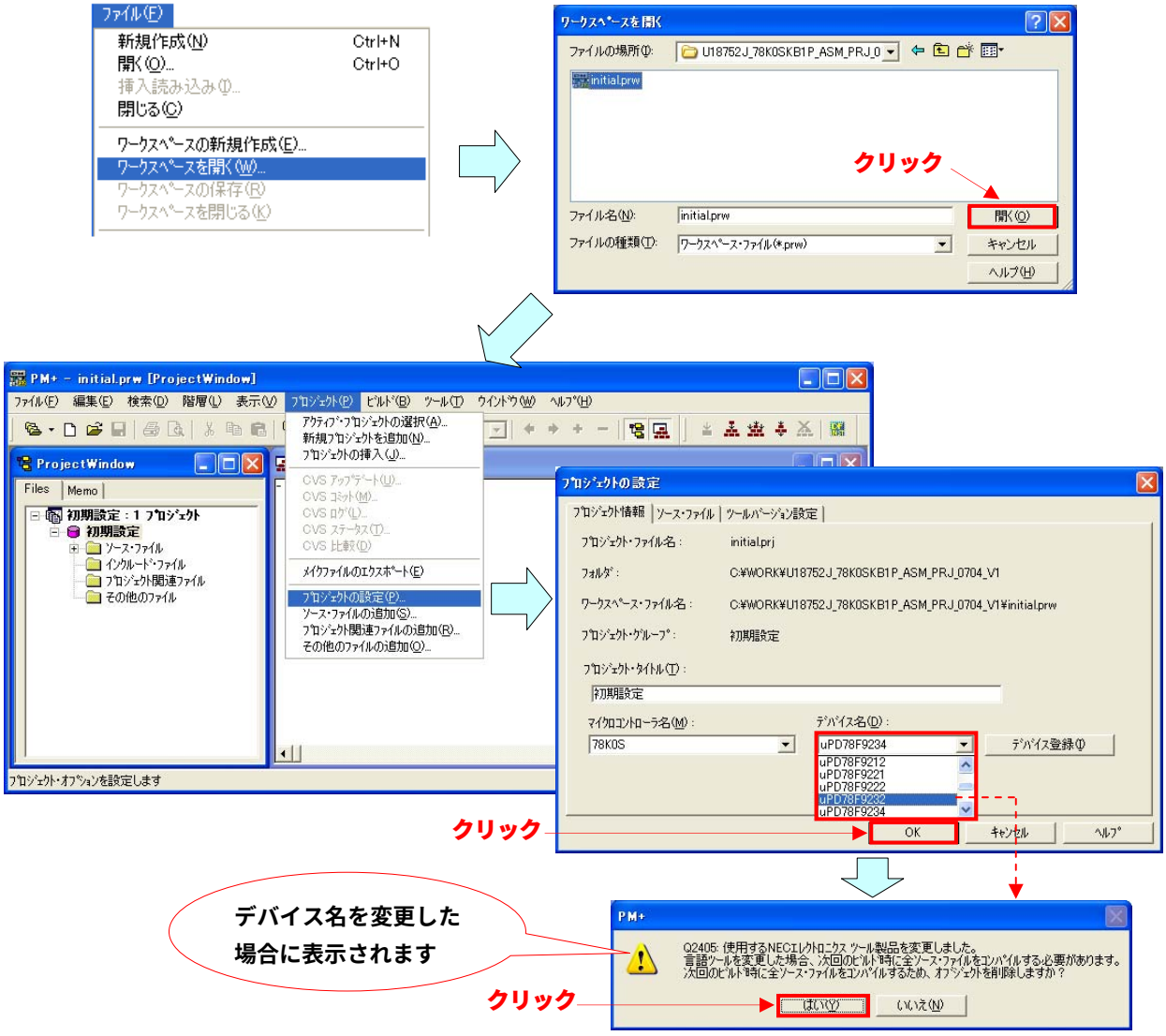
このサンプル・プログラムでは，  のアイコンを選択してダウンロードしたファイルを使用する場合，デフォルトで「標準ライブラリ固定領域を使用する」のチェックが外されています。

(1) PM+を起動してください。

(2) ［ファイル］→［ワークスペースを開く］から，「uart6.prw」を選択し，［開く］ボタンをクリックしてください。ワークスペースが作成され，その中にソース・ファイルが自動的に読み込まれます。

(3) ［プロジェクト］→［プロジェクトの設定］を選択してください。［プロジェクトの設定］画面が立ち上がったら，使用するデバイス名を選択（デフォルトでは，ROM/RAMサイズの最も大きいデバイスが選択）し，［OK］ボタンをクリックしてください。

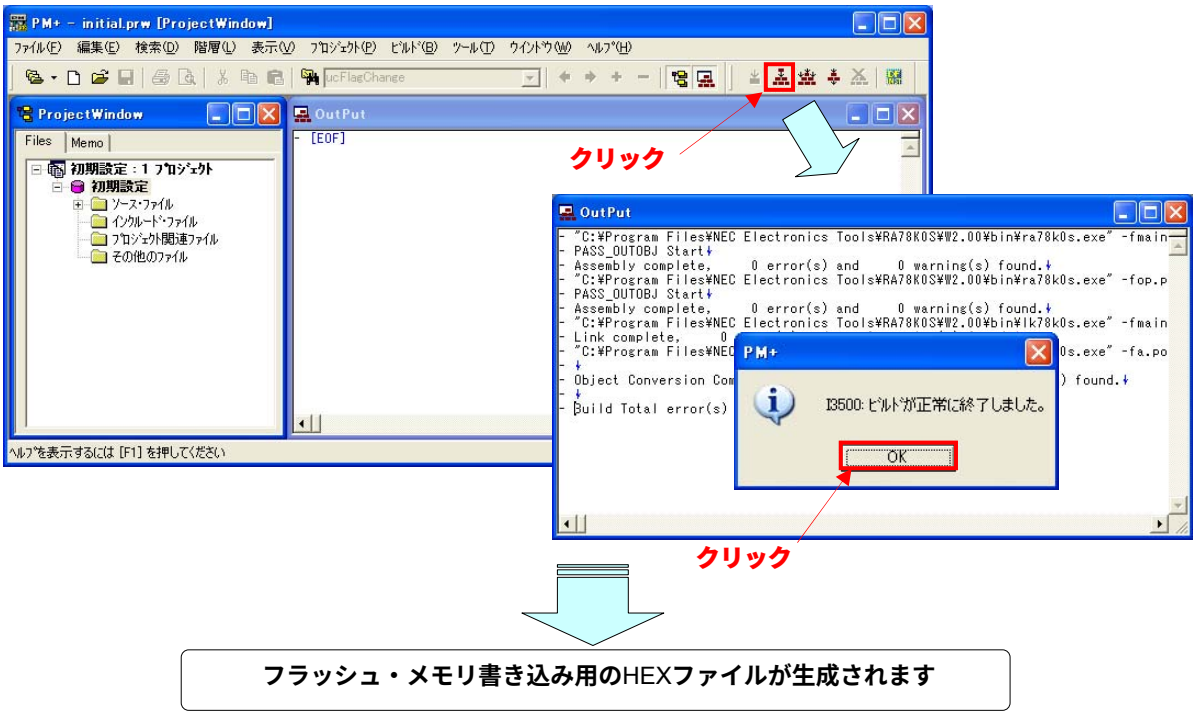
備考　下の図は，**「サンプル・プログラム（初期設定）**　LED**点灯のスイッチ制御」**の画面例です。



(4)  （「ビルド」ボタン）をクリックしてください。ソース・ファイルが正常にビルドされると，「I3500: ビルドが正常に終了しました」というメッセージ画面が立ち上がります。

(5) メッセージ画面にある［OK］ボタンをクリックしてください。フラッシュ・メモリ書き込み用のHEXファイルが作成されます。

備考　下の図は，「**サンプル・プログラム（初期設定）**　LED**点灯のスイッチ制御**」の画面例です。

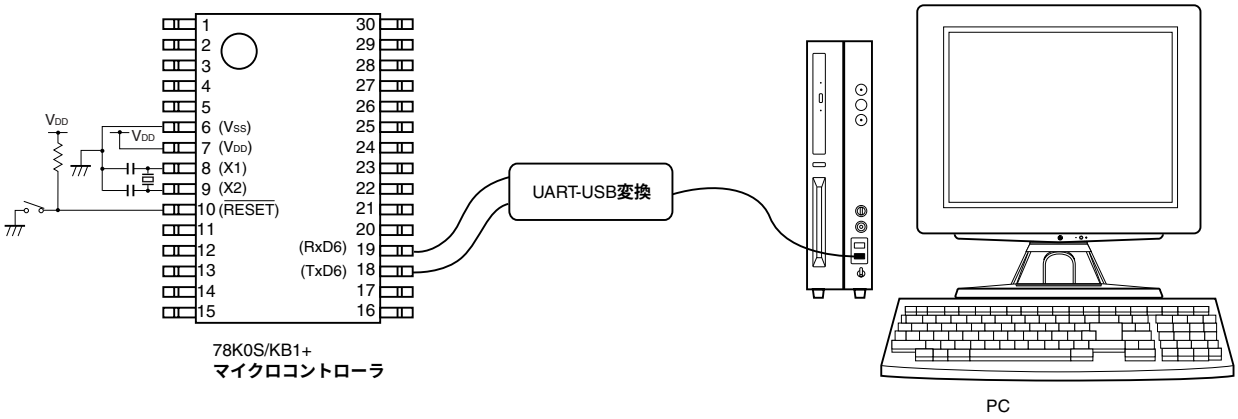


5. 2　デバイスでの動作

ここでは，デバイスでの動作確認の例を説明します。

ビルド実行により生成されたHEXファイルは，デバイスのフラッシュ・メモリに書き込みすることができます。デバイスへのフラッシュ・メモリ書き込み方法例については，各製品のフラッシュ書き込み簡単マニュアル　インフォメーション（78K0S/KA1+, 78K0S/KB1+）を参照してください。

デバイスと使用する周辺ハードウエアの接続例を，次に示します。



このサンプル・プログラムを書き込んだデバイスを上図のように接続し，Windows® 2000, Windows XP®に標準ツールとして用意されている「ハイパーターミナル」を使用した場合の動作例を，次に示します。

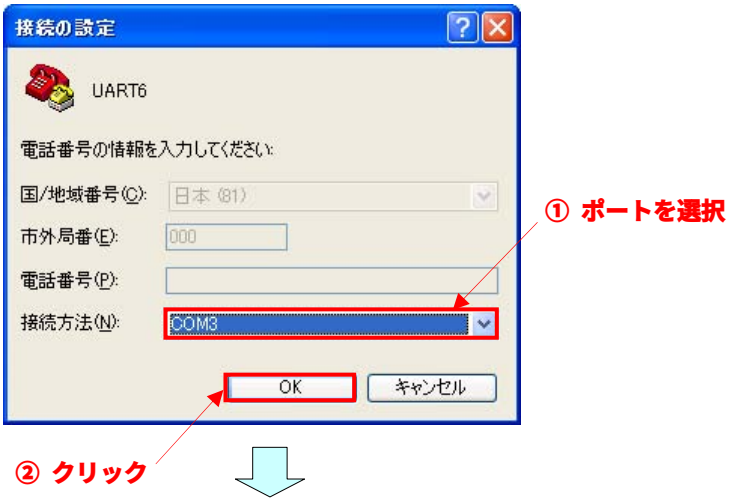
(1) このサンプル・プログラムを書き込んだデバイスを上図のように接続し，「ハイパーターミナル」を次の手順で起動してください。

・Windows 2000：　［スタート］→［プログラム］→［アクセサリ］→［ハイパーターミナル］の順に選択し，［Hyper Terminal］ウインドウが開いたら，そのウインドウ上にある " Hypertrm.exe " アイコンをダブルクリック

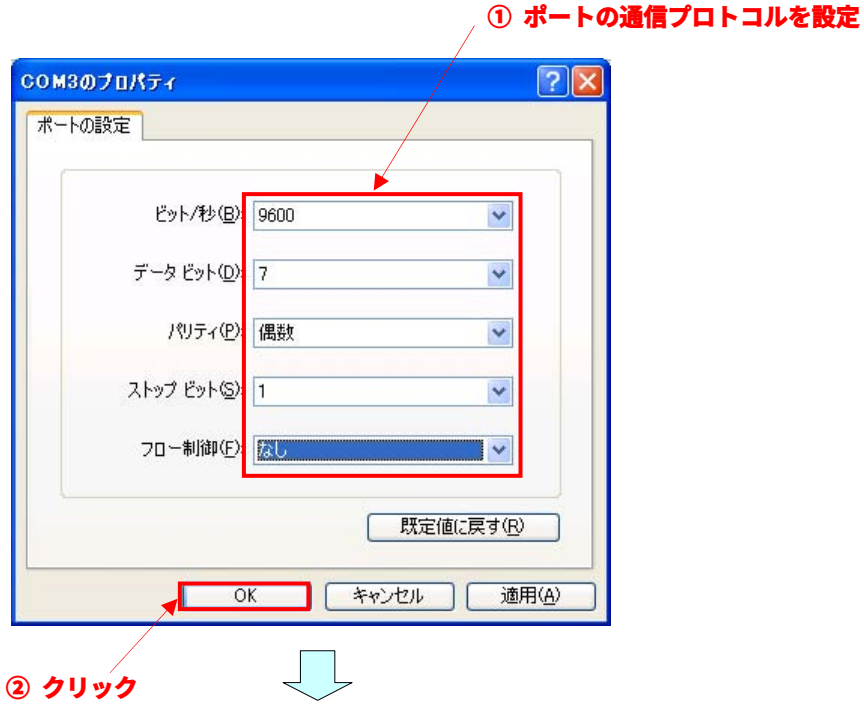
・Windows XP：　［スタート］→［すべてのプログラム］→［アクセサリ］→［通信］→［ハイパーターミナル］の順に選択

(2) ［接続の設定］画面が開きます。任意の名前入力（下記の画面例では「UART6」）と，アイコンを選択し（下記の画面例では一番左にあるアイコン選択），［OK］ボタンをクリックしてください。

(3) 設定画面が切り替わります。接続方法にUSBケーブルを接続しているポート（下記の画面例では「COM3」）を選択し，［OK］ボタンをクリックしてください。

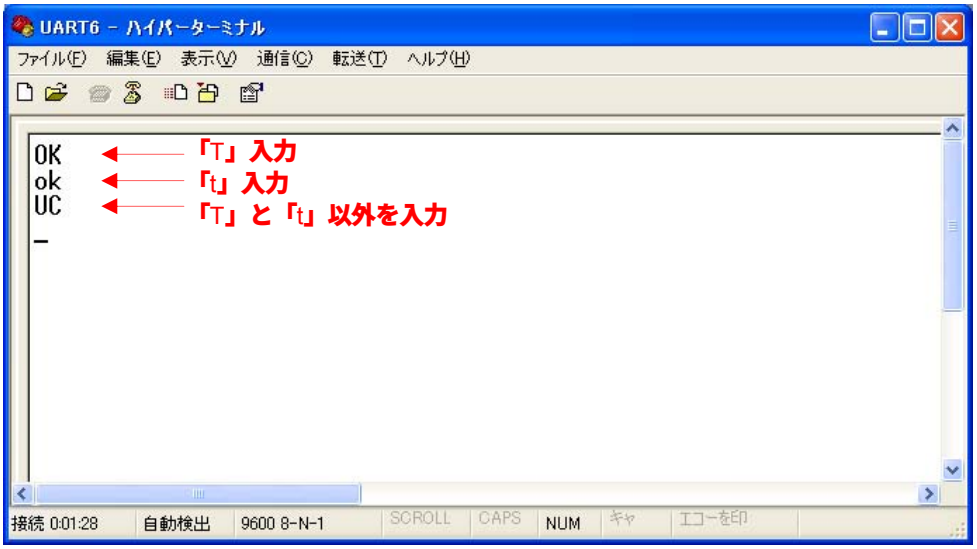


(4)　［xxxxのプロパティ］画面（xxxx：（3）で設定したポート名，下記の画面例では［COM3のプロパティ］）に切り替わります。ポートの通信プロトコルを下記の画面例のように設定して，［OK］ボタンをクリックしてください。



(5)　［yyyy – ハイパーターミナル］画面（yyyy：（2）で設定した通信名，下記の画面例では［UART6 – ハイパーターミナル］画面）に切り替わります。キーボードから入力する文字に応じて，画面に表示される文字が次のようになります。

キーボード入力	画面表示
T	OK＋ ”改行”
t	ok＋ ”改行”
上記以外	UC＋ ”改行”



第6章　関連資料

資　料　名		和文／英文
78K0S/KA1+　ユーザーズ・マニュアル		PDF
78K0S/KB1+　ユーザーズ・マニュアル		PDF
78K/0Sシリーズ　命令編　ユーザーズ・マニュアル		PDF
RA78K0S　アセンブラ・パッケージ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
CC78K0S　Cコンパイラ	言語編	PDF
ユーザーズ・マニュアル	操作編	PDF
PM+　プロジェクト・マネージャ　ユーザーズ・マニュアル		PDF
SM+　システム・シミュレータ　操作編　ユーザーズ・マニュアル		PDF
★ フラッシュ書き込み簡単マニュアル（MINICUBE2編）	78K0S/KA1+	PDF
インフォメーション	78K0S/KB1+	PDF
78K0S/Kx1+	サンプル・プログラム　スタートアップ・ガイド	PDF
アプリケーション・	サンプル・プログラム（初期設定）　LED点灯のスイッチ制御編	PDF
ノート	サンプル・プログラム（割り込み）　スイッチ入力による外部割り込み編	PDF
	サンプル・プログラム（低電圧検出）　2.7 V未満検出時リセット発生編	PDF

付録A　プログラム・リスト

プログラム・リスト例として，78K0S/KB1+マイクロコントローラのソース・プログラムを次に示します。

● main.asm （アセンブリ言語版）

```
;*******************************************************************************
;
;     NEC Electronics     78K0S/KB1+シリーズ
;
;*******************************************************************************
;     78K0S/KB1+シリーズ       サンプル･プログラム
;*******************************************************************************
;     シリアル・インタフェースUART6
;*******************************************************************************
;【履歴】
;     2007.8.--        新規作成
;*******************************************************************************
;
;【概要】
;
;本サンプルプログラムは、シリアル・インタフェースUART6の使用例を示すものである。
;システム・クロック・ソースとして8MHzの水晶／セラミック発振クロックを使用し、ボ
;ー・レート9600bpsでシリアル通信を行う。ASCIIコードの送受信を想定しており、1キャ
;ラクタのデータ受信に応じて4キャラクタのデータ送信を行う。受信エラー時にも、エラ
;ー内容に応じて4キャラクタのデータを送信する。
;
;
;  ＜主な設定内容＞
;
;  ・ウォッチドッグ・タイマの動作停止
;  ・低電圧検出電圧VLVIを4.3V±0.2Vに設定
;  ・VDD≧VLVIとなった後にVDD＜VLVIとなった場合、内部リセット信号発生(低電圧検出回路)
;  ・CPUクロックを8MHzに設定
;  ・周辺ハードウェアへの供給クロックを8MHzに設定
;  ・シリアル・インタフェースUART6の設定
;  ・DEレジスタ、HLレジスタを割り込み処理で使用する(=グローバル変数のように使用する)
;
;
;  ＜シリアル通信プロトコル＞
;
```

```
;  ・ボー・レート:              9600bps
;  ・データのキャラクタ長:      7bit
;  ・パリティ指定:              偶数パリティ
;  ・ストップ・ビット数:        1bit
;  ・先頭ビットの指定:          LSBファースト
;
;
;  ＜受信データについて＞
;
;  ASCIIコードの受信を想定しているので、データのキャラクタ長は7bitに指定し、LSB
;  ファーストに設定している。したがって、受信データはRXB6レジスタのbit0-6に転送
;  され、bit7(MSB)は必ず0になる。また、受信エラー発生時は受信エラー情報のbit7に
;  1をセットし、受信データと同じバッファに格納する。これにより、バッファ読み出し
;  時は、bit7で受信データか受信エラー情報かを判断している。
;
;
;  ＜連続受信について＞
;
;  受信データは割り込みを利用してバッファに格納するため、連続受信が可能であり、
;  バッファの先頭から順次累積していく。また、バッファはリングバッファとしており
;  バッファ最後尾に達した後は再びバッファの先頭から格納していく。このとき、読み
;  出し済みのバッファであれば受信データを格納するが、まだ読み出していないバッファ
;  であれば(読み出していないデータがバッファサイズに達している場合)、受信データ
;  は格納せずに破棄する。バッファのデータが読み出され次第、受信データを格納する。
;  バッファサイズはCBUFFSIZEで定義しており、デフォルト50バイトである。
;
;
;  ＜コマンド仕様＞
;
;  ・通常受信時
;  +-------------------------------------------------+
;  |   受信データ   |        4キャラクタ送信データ      |
;  | (16進数データ) |           (16進数データ)          |
;  |-------------------------------------------------|
;  |       T       |   O   |   K   | "CR"  | "LF"  |
;  |     (54H)     | (4FH) | (4BH) | (0DH) | (0AH) |
;  |-------------------------------------------------|
;  |       t       |   o   |   k   | "CR"  | "LF"  |
;  |     (74H)     | (6FH) | (6BH) | (0DH) | (0AH) |
;  |-------------------------------------------------|
;  | その他のデータ |   U   |   C   | "CR"  | "LF"  |
;  |               | (55H) | (43H) | (0DH) | (0AH) |
;  +-------------------------------------------------+
```

```
;  #"CR"+"LF"は改行コード
;
;  ・エラー受信時
;  +-------------------------------------------------+
;  | エラー受信情報 |        4キャラクタ送信データ      |
;  |                |           (16進数データ)          |
;  |-------------------------------------------------|
;  | パリティ・     |    P   |    E   |  "CR"  |  "LF"  |
;  | エラー         | (50H) | (45H) | (0DH) | (0AH) |
;  |-------------------------------------------------|
;  | フレーミング・ |    F   |    E   |  "CR"  |  "LF"  |
;  | エラー         | (46H) | (45H) | (0DH) | (0AH) |
;  |-------------------------------------------------|
;  | オーバラン・   |    O   |    E   |  "CR"  |  "LF"  |
;  | エラー         | (4FH) | (45H) | (0DH) | (0AH) |
;  +-------------------------------------------------+
;  #"CR"+"LF"は改行コード
;
;
;【ポート入出力の設定】
;
;  入力ポート：P44
;  出力ポート：P00-P03，P20-P23，P30-P33，P40-P43，P45-P47，P120-P123，P130
;  ※未使用のポートは全て出力ポートに設定しておく
;
;*******************************************************************************



;===============================================================================
;
;     シンボル定義
;
;===============================================================================
CBUFFSIZE     EQU     50              ; 受信データのバッファサイズ

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;     マクロの定義
;
; UART送信処理は記述を簡単にするためにマクロを定義する。
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
_SEROUT       MACRO   RTXDATA
```

```
        PUSH    AX                      ; AXレジスタのデータをスタックへ退避
        MOV     A,      RTXDATA         ; 仮パラメータRTXDATAをAレジスタに格納
        CALLT   [ZTXSUB]                ; UART送信用サブルーチンを実行
        POP     AX                      ; AXレジスタのデータを復帰

        ENDM

;=============================================================================
;
;       ベクタ・テーブルの設定
;
;=============================================================================
XVCT    CSEG    AT      0000H
        DW      IRESET                  ;(00)   RESET
        DW      IRESET                  ;(02)   --
        DW      IRESET                  ;(04)   --
        DW      IRESET                  ;(06)   INTLVI
        DW      IRESET                  ;(08)   INTP0
        DW      IRESET                  ;(0A)   INTP1
        DW      IRESET                  ;(0C)   INTTMH1
        DW      IRESET                  ;(0E)   INTTM000
        DW      IRESET                  ;(10)   INTTM010
        DW      IRESET                  ;(12)   INTAD
        DW      IRESET                  ;(14)   --
        DW      IRESET                  ;(16)   INTP2
        DW      IRESET                  ;(18)   INTP3
        DW      IRESET                  ;(1A)   INTTM80
        DW      IINTSRE6                ;(1C)   INTSRE6
        DW      IINTSR6                 ;(1E)   INTSR6
        DW      IRESET                  ;(20)   INTST6

;=============================================================================
;
;       CALLT・テーブルの設定
;
;   頻繁に呼び出すサブルーチンは、1バイトのコール命令であるCALLT命令を使用すると
;命令コードを短くすることが可能である。ここでは、UART送信のサブルーチンSTXSUBを
;CALLT命令で使用できるように、テーブルにアドレスをDW擬似命令で登録する。そのアド
;レス(ZTXSUB)を使って、CALLT [ZTXSUB]と記述することで使用することができる。
;
;=============================================================================
XCALT CSEG    CALLT0
```

```
ZTXSUB:             DW      STXSUB          ; UART送信用サブルーチン

;=============================================================================
;
;     RAMの定義
;
;=============================================================================
DRAM1 DSEG    UNIT
RRXBUFTOP:    DS      CBUFFSIZE       ; 受信データ・バッファ
RRXBUFEND:

DRAM2 DSEG    SADDR
RRXCNT:             DS      1               ; 受信カウント用変数

;=============================================================================
;
;     スタック領域の確保
;
;=============================================================================
DSTK  DSEG    AT      0FEE0H
RSTACKEND:    DS      20H             ; スタック領域を32バイト確保
RSTACKTOP:                            ; スタック領域の先頭アドレス = FF00H

;*****************************************************************************
;
;     リセット解除後の初期化処理
;
;*****************************************************************************
XMAIN CSEG    UNIT
IRESET:
;-----------------------------------------------------------------------------
;     スタック・ポインタの設定
;-----------------------------------------------------------------------------
      MOVW    AX,     #RSTACKTOP
      MOVW    SP,     AX              ; スタック・ポインタを設定

;-----------------------------------------------------------------------------
;     ウォッチドッグ・タイマの設定
;-----------------------------------------------------------------------------
      MOV     WDTM,   #01110111B      ; ウォッチドッグ・タイマ動作停止

;-----------------------------------------------------------------------------
;     低電圧検出 + クロックの設定
```

```
;-------------------------------------------------------------------------

;-----  クロックの設定1  -----
      MOV     PCC,    #00000000B      ; CPUクロックfcpu = fxp (= fx/4 = 2MHz)
      MOV     LSRCM,  #00000001B      ; 低速内蔵発振器の発振を停止

;-----  リセット要因の確認  -----
      MOV     A,      RESF            ; リセット要因の読み出し
      BT      A.0,    $HRST300        ; LVIリセット時は以降のLVI関連処理を省略し、SET_CLOCKへ

;-----  低電圧検出の設定  -----
      MOV     LVIS,   #00000000B      ; 低電圧検出レベルVLVI = 4.3V±0.2Vに設定
      SET1    LVION                   ; 低電圧検出回路の動作許可

      MOV     A,      #40             ; 200usウェイト用のカウント値を代入
;-----  200usウェイト  -----
HRST100:
      DEC     A
      BNZ     $HRST100                ; 0.5[us/clk]×10[clk]×40[count] = 200[us]

;-----  VDD≧VLVI待ち処理  -----
HRST200:
      NOP
      BT      LVIF,   $HRST200        ; VDD＜VLVIなら分岐

      SET1    LVIMD                   ; VDD＜VLVI時に内部リセット信号が発生するように設定

;-----  クロックの設定2  -----
HRST300:
      MOV     PPCC,   #00000000B      ; 周辺ハードウェアへの供給クロックfxp = fx (= 8MHz)
                                      ; -> CPUクロックfcpu = fxp = 8MHz

;-------------------------------------------------------------------------
;     ポート0の設定
;-------------------------------------------------------------------------
      MOV     P0,     #00000000B      ; P00-P03の出力ラッチLow
      MOV     PM0,    #11110000B      ; P00-P03を出力ポートに設定

;-------------------------------------------------------------------------
;     ポート2の設定
;-------------------------------------------------------------------------
      MOV     P2,     #00000000B      ; P20-P23の出力ラッチLow
      MOV     PM2,    #11110000B      ; P20-P23を出力ポートに設定
```

```
;-------------------------------------------------------------------------------
;     ポート3の設定
;-------------------------------------------------------------------------------
      MOV     P3,     #00000000B      ; P30-P33の出力ラッチLow
      MOV     PM3,    #11110000B      ; P30-P33を出力ポートに設定

;-------------------------------------------------------------------------------
;     ポート4の設定
;-------------------------------------------------------------------------------
      MOV     P4,     #00001000B      ; P40-P42,P44-P47の出力ラッチLow、P43の出力ラッチHigh(シリアル
送信用の設定)
      MOV     PM4,    #00010000B      ; P40-P43,P45-P47を出力ポートに、P44を入力ポートに設定

;-------------------------------------------------------------------------------
;     ポート12の設定
;-------------------------------------------------------------------------------
      MOV     P12,    #00000000B      ; P120-P123の出力ラッチLow
      MOV     PM12,   #11110000B      ; P120-P123を出力ポートに設定

;-------------------------------------------------------------------------------
;     ポート13の設定
;-------------------------------------------------------------------------------
      MOV     P13,    #00000001B      ; P130の出力High

;-------------------------------------------------------------------------------
;     UART6の設定
;-------------------------------------------------------------------------------
      MOV     CKSR6,  #1              ; ボー・レートを9600bpsに設定
      MOV     BRGC6,  #208            ; (同上)
      MOV     ASIM6,  #00011000B      ; 偶数パリティ出力、キャラクタ長7bit、ストップビット数1
                                      ; エラー発生時の割り込みにINTSRE6が発生
      SET1    POWER6                  ; 内部動作クロックの動作許可
      SET1    TXE6                    ; 送信動作許可
      SET1    RXE6                    ; 受信動作許可

;*******************************************************************************
;
;     メイン・ループ
;
;*******************************************************************************
MMAINLOOP:
```

```
;-----  RAM、汎用レジスタの初期化  -----
      MOV     RRXCNT, #0              ; 受信カウント = 0
      MOVW    HL,     #RRXBUFTOP      ; 書き込みアドレスをバッファ先頭に初期化
      MOVW    DE,     #RRXBUFTOP      ; 読み出しアドレスをバッファ先頭に初期化

;-----  割り込みの設定  -----
      MOV     IF0,    #00H            ; 無効割り込み要求をクリアしておく
      CLR1    SRMK6                   ; INTSR6(シリアル受信)割り込み許可
      CLR1    SREMK6                  ; INTSRE6(受信エラー)割り込み許可
      EI                              ; ベクタ割り込み許可

;-----  受信割り込み待ち  -----
LMLP100:
      CMP     RRXCNT, #0
      BZ      $LMLP100                ; 受信カウント = 0なら分岐

      MOV     A, [DE]                 ; データを読み出し
      DEC     RRXCNT                  ; 受信カウント-1

      BT      A.7,    $LMLP300        ; ビット7が1なら受信エラー時の処理へ分岐

;-----  通常受信時の処理  -----
LMLP200:
      CMP     A, #'T'                 ; 受信データをT(54H)と比較
      BNZ     $LMLP210                ; 受信データがT(54H)でなければ分岐
      _SEROUT #'O'                    ; O(4FH)送信
      _SEROUT #'K'                    ; K(4BH)送信
      _SEROUT #0DH                    ; 改行コード"CR"送信
      _SEROUT #0AH                    ; 改行コード"LF"送信
      BR      !LMLP400                ; LMLP400へ分岐

LMLP210:
      CMP     A, #'t'                 ; 受信データをt(74H)と比較
      BNZ     $LMLP220                ; 受信データがt(74H)でなければ分岐
      _SEROUT #'o'                    ; o(6FH)送信
      _SEROUT #'k'                    ; k(6BH)送信
      _SEROUT #0DH                    ; 改行コード"CR"送信
      _SEROUT #0AH                    ; 改行コード"LF"送信
      BR      !LMLP400                ; LMLP400へ分岐

LMLP220:
      _SEROUT #'U'                    ; U(55H)送信
      _SEROUT #'C'                    ; C(43H)送信
```

```
        _SEROUT #0DH                    ; 改行コード"CR"送信
        _SEROUT #0AH                    ; 改行コード"LF"送信
        BR      !LMLP400                ; LMLP400へ分岐

;-----  受信エラー時の処理  -----
LMLP300:
        BF      A.2, $LMLP310           ; パリティ・エラーでなければ分岐
        _SEROUT #'P'                    ; P(50H)送信
        _SEROUT #'E'                    ; E(45H)送信
        _SEROUT #0DH                    ; 改行コード"CR"送信
        _SEROUT #0AH                    ; 改行コード"LF"送信

LMLP310:
        BF      A.1, $LMLP320           ; フレーミング・エラーでなければ分岐
        _SEROUT #'F'                    ; F(46H)送信
        _SEROUT #'E'                    ; E(45H)送信
        _SEROUT #0DH                    ; 改行コード"CR"送信
        _SEROUT #0AH                    ; 改行コード"LF"送信

LMLP320:
        BF      A.0, $LMLP400           ; オーバラン・エラーでなければ分岐
        _SEROUT #'O'                    ; O(4FH)送信
        _SEROUT #'E'                    ; E(45H)送信
        _SEROUT #0DH                    ; 改行コード"CR"送信
        _SEROUT #0AH                    ; 改行コード"LF"送信

;-----  読み出しアドレスの更新  -----
LMLP400:
        INCW    DE                      ; 読み出しアドレス+1
        MOVW    AX,     DE
        CMPW    AX,     #RRXBUFEND
        BC      $LMLP450                ; 読み出しアドレスがバッファ内なら分岐
        MOVW    DE,     #RRXBUFTOP      ; 読み出しアドレスをバッファ先頭に初期化
LMLP450:
        BR      !LMLP100                ; LMLP100へ分岐

;*******************************************************************************
;
;       シリアル受信割り込みINTSR6
;
;*******************************************************************************
IINTSR6:
        PUSH    AX                      ; AXレジスタのデータをスタックへ退避
```

```
;-----  受信データの読み出し  -----
      MOV     A, RXB6                 ; シリアル受信データを読み出し

;-----  バッファの空き容量を確認  -----
      CMP     RRXCNT, #CBUFFSIZE      ; 受信カウントをバッファサイズと比較
      BNC     $HSR100                 ; バッファに空きがなければデータを格納しない
      INC     RRXCNT                  ; 受信カウント+1

;-----  データの保存と書き込みアドレスの更新  -----
      MOV     [HL],   A               ; 受信データを格納
      INCW    HL                      ; 書き込みアドレス+1
      MOVW    AX,     HL
      CMPW    AX,     #RRXBUFEND
      BC      $HSR100                 ; 書き込みアドレスがバッファ内なら分岐
      MOVW    HL,     #RRXBUFTOP      ; 書き込みアドレスをバッファ先頭に初期化

HSR100:
      POP     AX                      ; AXレジスタのデータを復帰
      RETI                            ; 割り込み処理から復帰

;*************************************************************************
;
;     受信エラー割り込みINTSRE6
;
;*************************************************************************
IINTSRE6:
      PUSH    AX                      ; AXレジスタのデータをスタックへ退避

;-----  エラー・ステータスの読み出し  -----
      MOV     A, ASIS6                ; エラー・ステータスを読み出し
      SET1    A.7                     ; ビット7に受信エラー・フラグをセット
      XCH     A,      X               ; エラー情報を退避
      MOV     A, RXB6                 ; シリアル受信データを読み出し(破棄)
      XCH     A,      X               ; エラー情報を復帰

;-----  バッファの空き容量を確認  -----
      CMP     RRXCNT, #CBUFFSIZE      ; 受信カウントをバッファサイズと比較
      BNC     $HSRE100                ; バッファに空きがなければデータを格納しない
      INC     RRXCNT                  ; 受信カウント+1

;-----  データの保存と書き込みアドレスの更新  -----
      MOV     [HL],   A               ; エラー・ステータスを格納
```

```
        INCW    HL                      ; 書き込みアドレス+1
        MOVW    AX,     HL
        CMPW    AX,     #RRXBUFEND
        BC      $HSR100                 ; 書き込みアドレスがバッファ内なら分岐
        MOVW    HL,     #RRXBUFTOP      ; 書き込みアドレスをバッファ先頭に初期化

HSRE100:
        POP     AX                      ; AXレジスタのデータを復帰
        RETI

;===============================================================================
;       シリアル・データ送信用サブルーチン
;
; シリアル・データを送信するためのサブルーチン。
; 使い方としては、以下のようにすることで、#DATAで示した1バイトデータを送信可能。
;
;       MOV     A,      #DATA           ; DATAをAレジスタに格納
;       CALL    !STXSUB                 ; DATAを送信
;===============================================================================
STXSUB:
        XCH     A,      X               ; 送信データをXレジスタへ退避

;-----  送信許可待ち  -----
JTXS100:
        MOV     A, ASIF6
        BT      A.1, $JTXS100           ; 送信許可待ち

;-----  データの送信  -----
        XCH     A,      X               ; 送信データをXレジスタから復帰
        MOV     TXB6, A                 ; シリアル送信

        RET                             ; サブルーチンから復帰

end
```

● main.c （C言語版）

```
/************************************************************************************

      NEC Electronics      78K0S/KB1+シリーズ

************************************************************************************
      78K0S/KB1+シリーズ        サンプル･プログラム
************************************************************************************
      シリアル・インタフェースUART6
************************************************************************************
【履歴】
      2007.8.--         新規作成
************************************************************************************

【概要】

本サンプルプログラムは、シリアル・インタフェースUART6の使用例を示すものである。
システム・クロック・ソースとして8MHzの水晶／セラミック発振クロックを使用し、ボ
ー・レート9600bpsでシリアル通信を行う。ASCIIコードの送受信を想定しており、1キャ
ラクタのデータ受信に応じて4キャラクタのデータ送信を行う。受信エラー時にも、エラ
ー内容に応じて4キャラクタのデータを送信する。


  ＜主な設定内容＞

  ・割り込みで起動される関数の宣言: INTSR6 -> fn_intsr6()
  ・割り込みで起動される関数の宣言: INTSRE6 -> fn_intsre6()
  ・ウォッチドッグ・タイマの動作停止
  ・低電圧検出電圧VLVIを4.3V±0.2Vに設定
  ・VDD≧VLVIとなった後にVDD＜VLVIとなった場合、内部リセット信号発生(低電圧検出回路)
  ・CPUクロックを8MHzに設定
  ・周辺ハードウェアへの供給クロックを8MHzに設定
  ・シリアル・インタフェースUART6の設定


  ＜シリアル通信プロトコル＞

  ・ボー・レート:           9600bps
  ・データのキャラクタ長:   7bit
  ・パリティ指定:           偶数パリティ
  ・ストップ・ビット数:     1bit
  ・先頭ビットの指定:       LSBファースト
```

＜受信データについて＞

ASCIIコードの受信を想定しているので、データのキャラクタ長は7bitに指定し、LSBファーストに設定している。したがって、受信データはRXB6レジスタのbit0-6に転送され、bit7(MSB)は必ず0になる。また、受信エラー発生時は受信エラー情報のbit7に1をセットし、受信データと同じバッファに格納する。これにより、バッファ読み出し時は、bit7で受信データか受信エラー情報かを判断している。

＜連続受信について＞

受信データは割り込みを利用してバッファに格納するため、連続受信が可能であり、バッファの先頭から順次累積していく。また、バッファはリングバッファとしておりバッファ最後尾に達した後は再びバッファの先頭から格納していく。このとき、読み出し済みのバッファであれば受信データを格納するが、まだ読み出していないバッファであれば(読み出していないデータがバッファサイズに達している場合)、受信データは格納せずに破棄する。バッファのデータが読み出され次第、受信データを格納する。バッファサイズはBUFF_SIZEで定義しており、デフォルト50バイトである。

＜コマンド仕様＞

・通常受信時

受信データ (16進数データ)	4キャラクタ送信データ (16進数データ)			
T (54H)	O (4FH)	K (4BH)	"CR" (0DH)	"LF" (0AH)
t (74H)	o (6FH)	k (6BH)	"CR" (0DH)	"LF" (0AH)
その他のデータ	U (55H)	C (43H)	"CR" (0DH)	"LF" (0AH)

#"CR"+"LF"は改行コード

・エラー受信時

エラー受信情報	4キャラクタ送信データ (16進数データ)

```
  |-------------------------------------------------|
  | パリティ・     |    P   |    E   | "CR"  | "LF"  |
  | エラー         | (50H) | (45H) | (0DH) | (0AH) |
  |-------------------------------------------------|
  | フレーミング・ |    F   |    E   | "CR"  | "LF"  |
  | エラー         | (46H) | (45H) | (0DH) | (0AH) |
  |-------------------------------------------------|
  | オーバラン・   |    O   |    E   | "CR"  | "LF"  |
  | エラー         | (4FH) | (45H) | (0DH) | (0AH) |
  +-------------------------------------------------+
  #"CR"+"LF"は改行コード


【ポート入出力の設定】

  入力ポート：P44
  出力ポート：P00-P03, P20-P23, P30-P33, P40-P43, P45-P47, P120-P123, P130
  ※未使用のポートは全て出力ポートに設定しておく

*************************************************************************/


/*=======================================================================

      前処理指令（#pragma指令）

=======================================================================*/
#pragma       SFR                              /* 特殊機能レジスタ(SFR)名を記述可能にする */
#pragma       EI                               /* EI命令を記述可能にする */
#pragma       NOP                              /* NOP命令を記述可能にする */
#pragma interrupt INTSR6 fn_intsr6    /* 割り込み関数宣言:INTSR6 */
#pragma interrupt INTSRE6 fn_intsre6  /* 割り込み関数宣言:INTSRE6 */

#define BUFF_SIZE 50                   /* 受信データのバッファサイズ */

/*=======================================================================

      関数プロトタイプ宣言

=======================================================================*/
void fn_uart_send(unsigned char ucTxData);    /* シリアル・データ送信用関数 */

/*=======================================================================
```

```
        グローバル変数の定義

================================================================================*/
static unsigned char g_ucRxBuff[BUFF_SIZE];    /* 受信データ・バッファ用テーブル */
sreg unsigned char g_ucRxCnt;          /* 受信カウント用8ビット変数 */
sreg unsigned char g_ucStoreAddr;      /* 書き込みアドレス用8ビット変数 */
sreg unsigned char g_ucReadAddr;       /* 読み出しアドレス用8ビット変数 */
sreg unsigned char g_ucRxData;                 /* 受信データ判別用8ビット変数 */
sreg unsigned char g_ucAsif6;          /* 送信ステータス判別用8ビット変数 */

/*******************************************************************************

        リセット解除後の初期化処理

*******************************************************************************/
void hdwinit(void){
        unsigned char ucCnt200us;       /* 200usウェイト用8ビット変数 */

/*------------------------------------------------------------------------------
        ウォッチドッグ・タイマの設定 + 低電圧検出 + クロックの設定
------------------------------------------------------------------------------*/
        /* ウォッチドッグ・タイマの設定 */
        WDTM  = 0b01110111;             /* ウォッチドッグ・タイマ動作停止 */

        /* クロックの設定1 */
        PCC   = 0b00000000;             /* CPUクロックfcpu = fxp (= fx/4 = 2MHz) */
        LSRCM = 0b00000001;             /* 低速内蔵発振器の発振を停止 */

        /* リセット要因の確認 */
        if (!(RESF & 0b00000001)){      /* LVIリセット時は以降のLVI関連処理を省略 */

                /* 低電圧検出の設定 */
                LVIS  = 0b00000000;     /* 低電圧検出レベルVLVI = 4.3V±0.2Vに設定 */
                LVION = 1;              /* 低電圧検出回路の動作許可 */

                for (ucCnt200us = 0; ucCnt200us < 9; ucCnt200us++){     /* 約200usウェイト */
                        NOP();
                }

                while (LVIF){           /* VDD≧VLVI待ち */
                        NOP();
                }
```

```
            LVIMD = 1;              /* VDD＜VLVI時に内部リセット信号が発生するように設定 */
        }

        /* クロックの設定2 */
        PPCC  = 0b00000000;         /* 周辺ハードウェアへの供給クロックfxp = fx (= 8MHz)
                                       -> CPUクロックfcpu = fxp = 8MHz */

    /*-------------------------------------------------------------------------
        ポート0の設定
    -------------------------------------------------------------------------*/
        P0    = 0b00000000;         /* P00-P03の出力ラッチLow */
        PM0   = 0b11110000;         /* P00-P03を出力ポートに設定 */

    /*-------------------------------------------------------------------------
        ポート2の設定
    -------------------------------------------------------------------------*/
        P2    = 0b00000000;         /* P20-P23の出力ラッチLow */
        PM2   = 0b11110000;         /* P20-P23を出力ポートに設定 */

    /*-------------------------------------------------------------------------
        ポート3の設定
    -------------------------------------------------------------------------*/
        P3    = 0b00000000;         /* P30-P33の出力ラッチLow */
        PM3   = 0b11110000;         /* P30-P33を出力ポートに設定 */

    /*-------------------------------------------------------------------------
        ポート4の設定
    -------------------------------------------------------------------------*/
        P4    = 0b00001000;         /* P40-P42,P44-P47の出力ラッチLow、P43の出力ラッチHigh(シリアル
送信用の設定) */
        PM4   = 0b00010000;         /* P40-P43,P45-P47を出力ポートに、P44を入力ポートに設定 */

    /*-------------------------------------------------------------------------
        ポート12の設定
    -------------------------------------------------------------------------*/
        P12   = 0b00000000;         /* P120-P123の出力ラッチLow */
        PM12  = 0b11110000;         /* P120-P123を出力ポートに設定 */

    /*-------------------------------------------------------------------------
        ポート13の設定
    -------------------------------------------------------------------------*/
        P13   = 0b00000001;         /* P130の出力High */
```

```
/*---------------------------------------------------------------------------
      UART6の設定
---------------------------------------------------------------------------*/
      CKSR6 = 1;                    /* ボー・レートを9600bpsに設定 */
      BRGC6 = 208;                  /* (同上) */
      ASIM6 = 0b00011000;           /* 偶数パリティ出力、キャラクタ長7bit、ストップビット数1 */
                                    /* エラー発生時の割り込みにINTSRE6が発生 */
      POWER6 = 1;                   /* 内部動作クロックの動作許可 */
      TXE6  = 1;                    /* 送信動作許可 */
      RXE6  = 1;                    /* 受信動作許可 */

      return;
}

/*****************************************************************************

      メイン・ループ

*****************************************************************************/
void main(void)
{
      g_ucRxCnt = 0;                /* 受信カウント = 0 */
      g_ucStoreAddr = 0;            /* 書き込みアドレスをバッファ先頭に初期化 */
      g_ucReadAddr = 0;             /* 読み出しアドレスをバッファ先頭に初期化 */
      IF0 = 0x00;                   /* 無効割り込み要求をクリアしておく */
      SRMK6 = 0;                    /* INTSR6(シリアル受信)割り込み許可 */
      SREMK6 = 0;                   /* INTSRE6(受信エラー)割り込み許可 */
      EI();                         /* ベクタ割り込み許可 */

      while (1)
      {
            while (g_ucRxCnt == 0)  /* 受信割り込み待ち */
            {
                  NOP();
            }

            while (g_ucRxCnt > 0)   /* 受信カウント＞0の場合の処理 */
            {
                  g_ucRxData = g_ucRxBuff[g_ucReadAddr];  /* データを読み出し */
                  g_ucRxCnt -= 1;                         /* 受信カウント-1 */

                  if (!g_ucRxData.7)      /* 通常受信時の処理 */
```

```
{
        switch (g_ucRxData)
        {
                case 'T' :      /* T(54H)受信時 */

                        fn_uart_send('O');      /* O(4FH)送信 */
                        fn_uart_send('K');      /* K(4BH)送信 */
                        fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                        fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
                        break;

                case 't' :      /* t(74H)受信時 */

                        fn_uart_send('o');      /* o(6FH)送信 */
                        fn_uart_send('k');      /* k(6BH)送信 */
                        fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                        fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
                        break;

                default  :      /* その他のデータ受信時 */

                        fn_uart_send('U');      /* U(55H)送信 */
                        fn_uart_send('C');      /* C(43H)送信 */
                        fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                        fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
                        break;
        }
}

else                    /* エラー受信時の処理 */
{
        if (g_ucRxData.2)       /* パリティ・エラー時 */
        {
                fn_uart_send('P');      /* P(50H)送信 */
                fn_uart_send('E');      /* E(45H)送信 */
                fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
        }

        if (g_ucRxData.1)       /* フレーミング・エラー時 */
        {
                fn_uart_send('F');      /* F(46H)送信 */
                fn_uart_send('E');      /* E(45H)送信 */
```

```
                        fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                        fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
                    }

                    if (g_ucRxData.0)       /* オーバラン・エラー時 */
                    {
                        fn_uart_send('O');      /* O(4FH)送信 */
                        fn_uart_send('E');      /* E(45H)送信 */
                        fn_uart_send(0x0D);     /* 改行コード"CR"送信 */
                        fn_uart_send(0x0A);     /* 改行コード"LF"送信 */
                    }
                }
                g_ucReadAddr += 1;              /* 読み出しアドレス+1 */

                if (g_ucReadAddr >= BUFF_SIZE)  /* 読み出しアドレスがバッファ外の場合 */
                {
                    g_ucReadAddr = 0;       /* 読み出しアドレスをバッファ先頭に初期化 */
                }
            }
        }
    }

    /************************************************************************

        シリアル受信割り込みINTSR6

    ************************************************************************/
    __interrupt void fn_intsr6()
    {
        unsigned char ucData;

        ucData = RXB6;                  /* シリアル受信データを読み出し */

        if (g_ucRxCnt < BUFF_SIZE)              /* 書き込みアドレスがバッファ内の場合 */
        {
            g_ucRxCnt += 1;                         /* 受信カウント+1 */
            g_ucRxBuff[g_ucStoreAddr] = ucData;     /* 受信データを保存 */

            g_ucStoreAddr += 1;                     /* 書き込みアドレス+1 */

            if (g_ucStoreAddr >= BUFF_SIZE)         /* 書き込みアドレスがバッファ外の場合 */
            {
                g_ucStoreAddr = 0;              /* 書き込みアドレスをバッファ先頭に初期化 */
```

```
		}
	}

	return;
}

/*****************************************************************************

	受信エラー割り込みINTSRE6

*****************************************************************************/
__interrupt void fn_intsre6()
{
	unsigned char ucData;
	unsigned char ucTemp;

	ucData = ASIS6 | 0b10000000;	/* bit7にエラー・フラグをセットしてエラー情報を格納 */
	ucTemp = RXB6;			/* シリアル受信データを読み出し(破棄) */

	if (g_ucRxCnt < BUFF_SIZE)		/* 書き込みアドレスがバッファ内の場合 */
	{
		g_ucRxCnt += 1;				/* 受信カウント+1 */
		g_ucRxBuff[g_ucStoreAddr] = ucData;	/* 受信データを保存 */

		g_ucStoreAddr += 1;			/* 書き込みアドレス+1 */

		if (g_ucStoreAddr >= BUFF_SIZE)		/* 書き込みアドレスがバッファ外の場合 */
		{
			g_ucStoreAddr = 0;		/* 書き込みアドレスをバッファ先頭に初期化 */
		}
	}

	return;
}

/*===========================================================================
	シリアル・データ送信用関数

 シリアル・データを送信するための関数。
 使い方としては、以下のようにすることで、Dataで示した1バイトデータを送信可能。

	fn_uart_send(Data);
===========================================================================*/
```

```
void fn_uart_send(unsigned char ucTxData)
{
      g_ucAsif6 = ASIF6;              /* 送信ステータス読み出し */

      while (g_ucAsif6.1)             /* 送信許可待ち */
      {
            g_ucAsif6 = ASIF6;        /* 送信ステータス読み出し */
      }
      TXB6 = ucTxData;                /* シリアル送信 */

      return;
}
```

● op.asm （アセンブリ言語版とC言語版共通）

```
;=============================================================================
;
;   オプション・バイトの設定
;
;=============================================================================
OPBT        CSEG    AT      0080H
        DB      10011000B       ; オプション・バイトの設定
;               || ||||
;               || |||+---------- 低速内蔵発振器はソフトウェアで停止可能
;               || |++---------- 水晶/セラミック発振クロックを使用
;               || +------------- P34/RESET端子をリセット端子として使用
;               ++---------------- 電源投入時またはリセット解除後の発振安定時間 = 2^10/fx

        DB      11111111B       ; プロテクト・バイトの設定(セルフプログラミング用)
;               ||||||||
;               ++++++++---------- 全てのブロックへの書き込み許可

end
```

付録B　改版履歴

本文欄外の★印は，本版で改訂された主な箇所を示しています。この"★"をPDF上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

版　数	発行年月	改版箇所	改版内容
第1版	October 2007	－	－
第2版	July 2008	pp.27-29	5. 1　**サンプル・プログラムのビルド**を変更
		p.33	**第6章　関連資料** **・フラッシュ書き込み簡単マニュアル**（MINICUBE2編）　**インフォメーション**を変更

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。