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+

8ビット・シングルチップ・マイクロコントローラ

EEPROM®エミュレーション

μPD78F9200
μPD78F9201
μPD78F9202
μPD78F9210
μPD78F9211
μPD78F9212
μPD78F9221
μPD78F9222
μPD78F9224
μPD78F9232
μPD78F9234
μPD78F9500
μPD78F9501
μPD78F9502
μPD78F9510
μPD78F9511
μPD78F9512

資料番号 U17379JJ3V0AN00（第3版）
発行年月 September 2009 NS

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

(1) 入力端子の印加波形：入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOSデバイスの入力がノイズなどに起因して，VIL（MAX.）からVIH（MIN.）までの領域にとどまるような場合は，誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん，VIL（MAX.）からVIH（MIN.）までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

(2) 未使用入力の処理：CMOSデバイスの未使用端子の入力レベルは固定してください。未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してVDDまたはGNDに接続することが有効です。資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

(3) 静電気対策：MOSデバイス取り扱いの際は静電気防止を心がけてください。MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，当社が出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。また，MOSデバイスを実装したボードについても同様の扱いをしてください。

(4) 初期化以前の状態　電源投入時，MOSデバイスの初期状態は不定です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

(5) 電源投入切断順序　内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

(6) 電源OFF時における入力信号　当該デバイスの電源がOFF状態の時に，入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により，誤動作を引き起こしたり，異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

注意：本製品は，Silicon Storage Technology, Inc.**からライセンスを受けた**SuperFlash **を使用しています。**

- **本資料に記載されている内容は**2009**年**9**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，ソフトウエ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。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。当社製品をお客様の機器にご使用の際には，当社製品の不具合の結果として，生命，身体および財産に対する損害や社会的損害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。**

注2. **本事項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，製造製品をいう。**

（M8E0909J**）**

はじめに

対 象 者　このアプリケーション・ノートは，78K0S/Kx1+のフラッシュ・メモリ内蔵品の機能を理解し，それを用いたアプリケーション・システムを設計するユーザを対象としています。

目　　的　このアプリケーション・ノートは，78K0S/Kx1+のフラッシュ・メモリ・セルフ・プログラミング機能を使用し，フラッシュ・メモリをEEPROMエミュレーションでデータを格納する方法（アプリケーションによる定数データ書き込み）をユーザに理解していただくことを目的としています。

構　　成　このマニュアルは，大きく分けて次の内容で構成しています。

・EEPROMエミュレーション機能

・EEPROMエミュレーション・プログラム

読 み 方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコントローラの一般知識を必要とします。

□78K0S/Kx1+のハードウエア機能を知りたいとき

→78K0S/Kx1+ 各製品のユーザーズ・マニュアルを参照してください。

□78K0S/Kx1+のフラッシュ・メモリ・セルフ・プログラミング機能を知りたいとき

→78K0S/Kx1+ 各製品のユーザーズ・マニュアルにあるフラッシュ・メモリの章を参照してください。

□一通りの機能を理解しようとするとき

→目次に従って読んでください。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。

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，信号名称に上線）
注	：本文中につけた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文の補足説明
数の表記	：2進数…××××または××××B 10進数…×××× 16進数…××××H

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

資　料　名	資料番号	
	和　　文	英　　文
78K0S/KB1+ ユーザーズ・マニュアル	U17446J	U17446E
78K0S/KA1+ ユーザーズ・マニュアル	U16898J	U16898E
78K0S/KY1+ ユーザーズ・マニュアル	U16994J	U16994E
78K0S/KU1+ ユーザーズ・マニュアル	U18172J	U18172E
78K0S/Kx1+　EEPROMエミュレーション　アプリケーション・ノート	このマニュアル	U17379E
78K/0Sシリーズ　ユーザーズ・マニュアル　命令編	U11047J	U11047E

目　　次

第1章　フラッシュ・メモリ・セルフ・プログラミング概要

78K0S/Kx1+はアプリケーション・プログラムからフラッシュ・メモリの書き換え（フラッシュ・メモリ・セルフ・プログラミング）が可能です。

このアプリケーション・ノートでは，セルフ・プログラミング機能を使用し，フラッシュ・メモリに任意のデータを格納する方法（EEPROMのように読み出し／書き込み）を解説します。

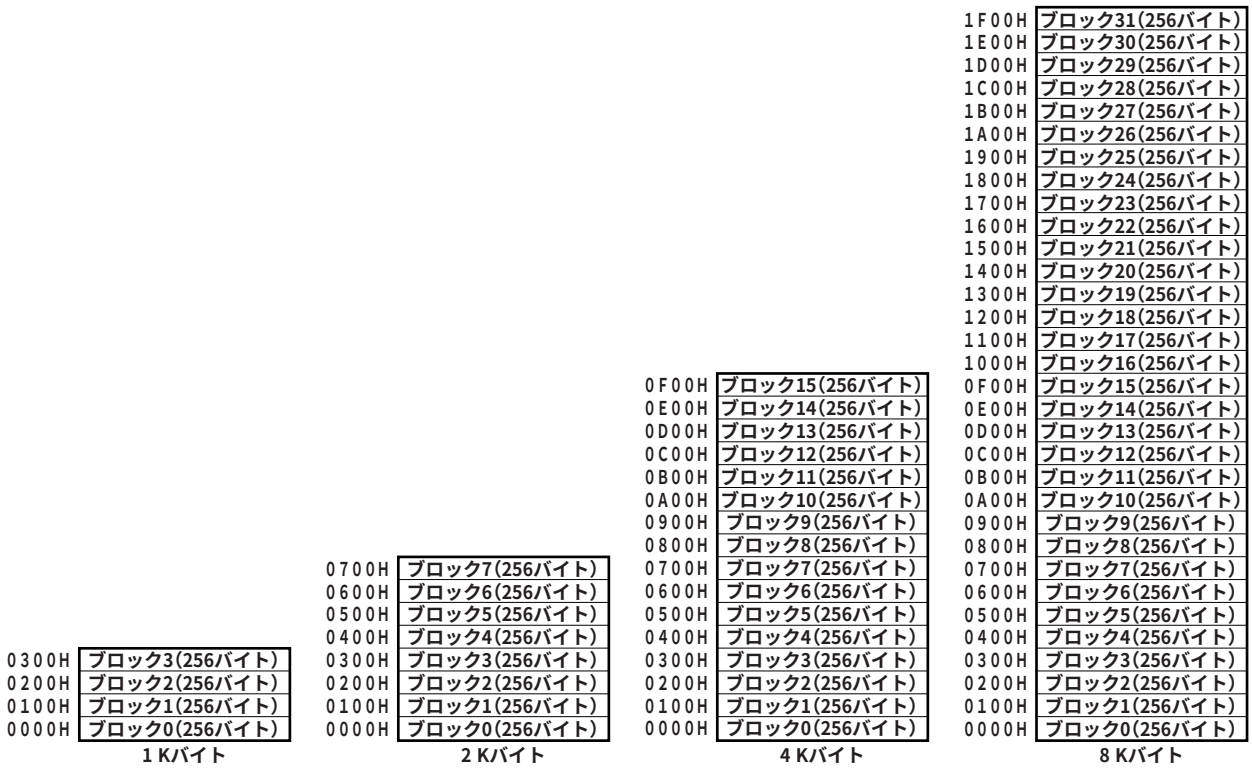
備考　フラッシュ・メモリ・セルフ・プログラミングの詳細については，78K0S/Kx1+**各製品のユーザーズ・マニュアルにあるフラッシュ・メモリの章**を参照してください。

1.1　セルフ・プログラミング可能なフラッシュ・メモリ領域

フラッシュ・メモリの制御では，消去，ブランク・チェック，ベリファイを行う領域をブロック単位で指定します。図1－1に，指定できるブロック番号の一覧を示します。

注意　製品が内蔵するフラッシュ・メモリ以外の領域へのアクセスはできません。

図1－1　ブロック番号の割り付け



第2章　EEPROMエミュレーション機能（固定長単一データ方式）

2. 1　EEPROMエミュレーションの基本仕様

EEPROMエミュレーションとは，フラッシュ・メモリのセルフ・プログラミング機能を使用することで，フラッシュ・メモリの一部を書き換え可能なデータROMとして使用するための機能です。サンプル・プログラムとユーザ・プログラムを組み合わせることでEEPROMのように読み出し／書き込み処理を実行できます。

なお，内蔵のフラッシュ・メモリの書き換え回数には制限があるので，データ長と書き換え回数に制約があります。次にサンプル・プログラムの基本仕様，書き換え回数の計算方法を示します。

サンプル・プログラムの基本仕様，書き換え回数の計算方法

・保存するデータ・フォーマット

データ（2バイト）	デリミタ（1バイト）
↑	↑
任意のデータ	書き込み状態を示す。データ検索に使用。

備考　データのサイズは，1バイトから設定可能です。上限は使用できるRAMサイズに依存します。

・フラッシュ・メモリ・ブロックに書き換え可能な回数

（256 － 2）÷ 3 ＝ 84 回（小数点以下切り捨て）

- データ・サイズ
- 有効／無効のフラグ（2. 1. 3参照）
- 1ブロックのメモリ・サイズ

・EEPROM領域として使用するブロック数

2ブロック（MIN.）を使用

備考　ブロック消去中に電源切断／電源瞬断などのトラブルによるデータ損失を防ぐために必要です。

★　・1ブロックの消去回数

1000回

★　・最大書き換え回数

84 × 2 × 1000 ＝ 168000 回

- ブロック消去回数
- 使用するブロック数
- 1ブロックで書き換え可能な回数

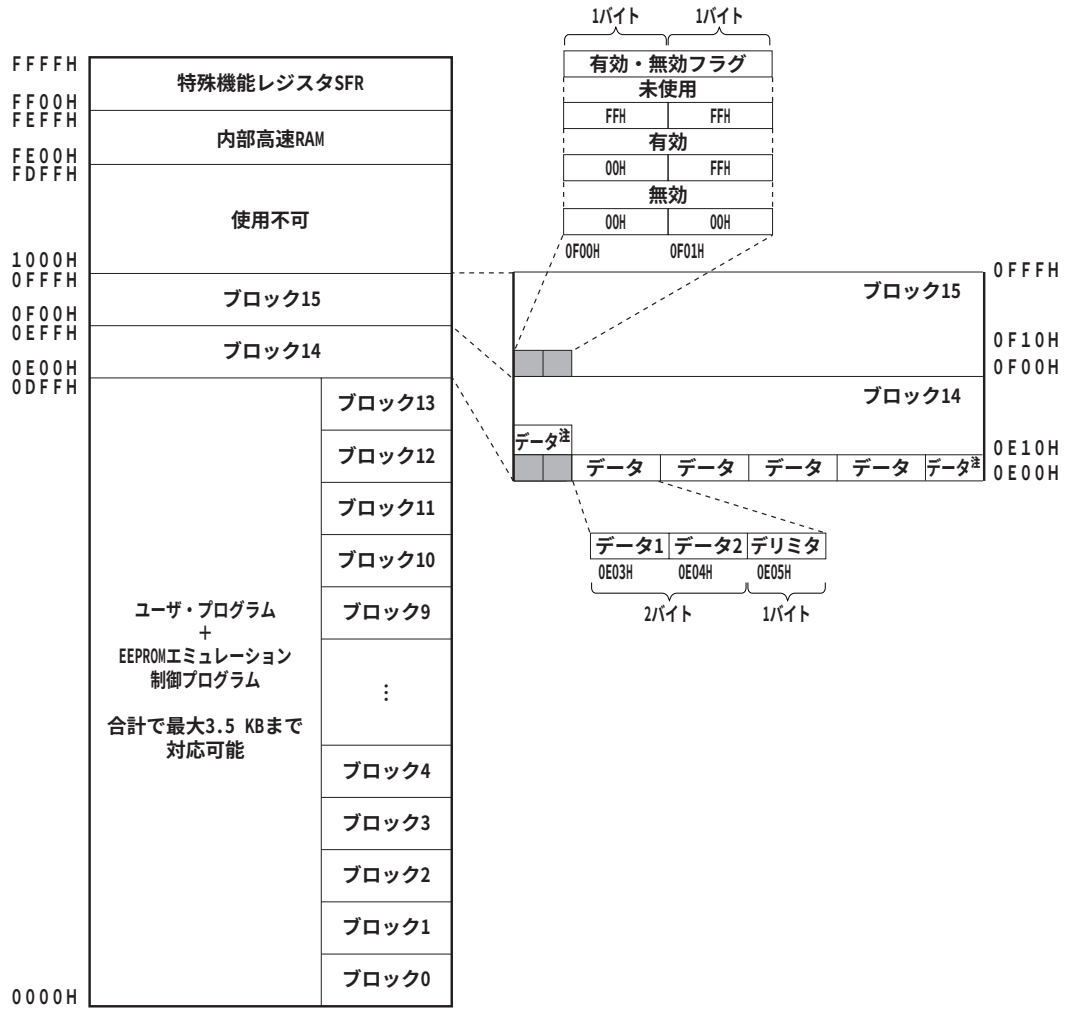
サンプル・プログラムでは，2バイト単位のデータを扱っており，データの終わりを指名すデリミタ（1バイト）との組み合わせで1ブロック（256バイト）当たり84回の書き換えが可能です。また，予期しない電源電圧降下などによる損失を防ぐためには連続した2ブロック以上の領域が必要なので，2ブロックを使用した場合，合計168
★ 回の書き換えが可能です。さらにサンプル・プログラムでは，1ブロックの消去回数を1000回まで許可しているので，2ブロックを使用した場合は，最大168000回のデータ書き換えが可能になります。

また，データを格納するためのフラッシュ・メモリの配置は，連続した2ブロック以上を確保する必要がありますが，これらのブロックはユーザにて自由に設定することができます。

図2－1にメモリ・マップとデータ構造例（プログラム・サイズが3.5 KBで，EEPROMエミュレーション用としてブロック14, 15を設定した場合）を示します。

図2－1　メモリ・マップとデータ構造例

（プログラム・サイズが3.5 KBで，EEPROMエミュレーションのデータ領域としてブロック14, 15を設定した場合）



注　データは続けて格納されます。

備考　図中に示したデータ構造はサンプル・プログラムを使用した場合の例です。

2. 1. 1　EEPROMエミュレーション・データ・ブロック

EEPROMエミュレーション・プログラムは，データを格納するためのブロックが連続2ブロック以上必要です。

使用するブロックは，ユーザ・プログラム領域，EEPROMエミュレーション制御プログラム領域と重ならなければ，ユーザによって自由に設定できます。

2. 1. 2　データ構造

EEPROMエミュレーションにより格納されるデータは，データ（1バイト単位），デリミタ（1バイト）で構成されています。

図2－2　データ構造

1バイト単位	1バイト
データ	デリミタ

(1) データ

00H-FFHの任意の値を設定できます。サイズは，1バイトから設定可能ですが，サイズが大きくなるにつれて書き換え可能な回数は減りますので，ご注意ください。

(2) デリミタ

デリミタは，00Hの固定値です。デリミタは，データ書き込み中に電源断などのトラブルにより，データが正常に書き込めなかった場合を検出するために，書き込まれます。最後にデリミタを書き込むことにより，そのデータの書き込みが正常に終了したかを判定します。デリミタ（00H）が正しく読み出せない場合は，データ書き込み中のトラブル発生が予想されるため，そのデータを使用しません。

検索により，最新のデータがデリミタ異常であるとわかった場合は，その前に書かれた，デリミタが正常なデータが最新データとして読み出されます。

(3) データの格納および検索動作の通常の流れ

データの格納および検索動作の通常の流れについて，次に示します（EEPROMエミュレーション指定ブロックが14, 15ブロックの場合）。

（状態1）ブロック14を有効ブロックにする。

ブロック14	＋0	
有効フラグ	00H	0E00H
無効フラグ	FFH	0E01H
データ	FFH	0E02H

（状態2）データ（11H, 22H）を書き込む。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データ	11H	22H	00H	0E02H
⋮	FFH			0E06H

（状態3）　データ（22H, 33H）を書き込む。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データ	11H	22H	00H	0E02H
データ	22H	33H	00H	0E06H
⋮	FFH			0E0AH

（状態4）　データ（20H, 30H）を書き込む。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データ	11H	22H	00H	0E02H
データ	22H	33H	00H	0E06H
データ	20H	30H	00H	0E0AH
⋮	FFH			0E0EH

（状態5）　データ（20H, 30H）の読み出し。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データa	11H	22H	00H	0E02H
データb	22H	33H	00H	0E06H
データc	20H	30H	00H	0E0AH
データd	FFH			0E0EH
	・ ・ ・			
消去状態	FFH			0EFEH

読み出し方法

① データaは，00H（正常値）なので，最新データとして格納し，次のデータを確認します。
② データbは，00H（正常値）なので，最新データとして更新し，次のデータを確認します。
③ データcは，00H（正常値）なので，最新データとして更新し，次のデータを確認します。
④ ブロックの最後まで消去状態か確認します。
⑤ このとき格納されている最新データcを読み出し値とします。

データ格納時に電源断などのトラブルが発生したときの流れについて，次に示します（EEPROMエミュレーション指定ブロックが14, 15ブロックの場合）。

（状態1）ブロック14を有効ブロックにする。

ブロック14	+0	
有効フラグ	00H	0E00H
無効フラグ	FFH	0E01H
データ	FFH	0E02H

（状態2）データ番号1（データ：11H, 22H）を書き込む。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データ	11H	22H	00H	0E02H
⋮	FFH			0E06H

（状態3）データ番号1（データ22H, 33H）の書き込み途中に電源断が発生し，デリミタが正しく書き込めない（00H以外のデリミタが書き込まれた）

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データ	11H	22H	00H	0E02H
データ	22H	33H	01H	0E06H
⋮	FFH			0E0AH

（状態4）データ番号1の読み出し。

ブロック14	+0	+1	+2	
有効フラグ	00H			0E00H
無効フラグ	FFH			0E01H
データa	11H	22H	00H	0E02H
データb	22H	33H	01H	0E06H
データc	FFH			0E0AH
	•			
	•			
	•			
消去状態	FFH			0EFEH

読み出し方法

① データaは，データ番号が一致するのでデリミタを確認し，デリミタが00H（正常）なので，データ2バイトを最新データとして格納し，次のデータに移ります。

② データbは，データ番号が一致するのでデリミタを確認し，デリミタが01H（異常）なので，次のデータに移ります。

③ ブロックの最後まで消去状態か確認します。

④ このとき格納されている最新データaを読み出し値とします。

2. 1. 3　有効／無効フラグ

有効／無効フラグは，ブロックの先頭に，1バイトずつ計2バイトのデータとして配置されています。有効／無効フラグは，そのブロックに格納されている各データの有効／無効状態を示しています。

有効／無効フラグの値がそれぞれ，有効フラグ = 00Hかつ無効フラグ = FFHの場合，そのブロックは有効となります。それ以外の場合，そのブロックは無効となります。

有効となっているブロックに順次データを格納しますが，そのブロックが満杯になった場合，次にデータを格納するブロックを有効にし，これまで有効だったブロックを無効にします。この手順により，ブロックが満杯となり次のブロックにデータを転送している最中に，電源断などのトラブルが発生した場合でも，それまでのデータは確保され，データの消失は起こらない構造となっています。

有効／無効フラグ動作の流れについて，次に示します。

（状態1）初期状態

	ブロックn		ブロックn＋1
有効フラグ	FFH	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	FFH	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態2）ブロックnの有効フラグに00Hを書き込み

	ブロックn		ブロックn＋1
有効フラグ	00H	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	FFH	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態3）ブロックnにデータ書き込み

	ブロックn		ブロックn＋1
有効フラグ	00H	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	Data	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態4）ブロックnのデータが満杯

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	FFH
無効フラグ	FFH
データ	FFH
⋮	⋮
データ	FFH

（状態5）ブロックn＋1に最新データを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	FFH
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態6）ブロックn＋1の有効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態7）ブロックnの無効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態8）ブロックn＋1のデータが満杯

	ブロックn
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態9）ブロックnの消去

	ブロックn
有効フラグ	FFH
無効フラグ	FFH
データ	FFH
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態10）ブロックnに最新データを書き込み

	ブロックn
有効フラグ	FFH
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態11）ブロックnの有効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態12）ブロックn＋1の無効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

2. 2　EEPROMエミュレーション・プログラム実行条件

EEPROMエミュレーション・プログラムを実行する前に，必ず表2－1の条件をすべて満たしてください。

表2－1　EEPROMエミュレーション動作時の条件

項　目	内　容
スタック領域の確保 （アセンブリ言語：22バイト）	EEPROMエミュレーション・プログラム動作時，ユーザ・プログラムが使用しているスタックを継承して使用します。EEPROMエミュレーション・プログラム実行開始時のスタック・アドレスから，さらに左記のスタック領域が必要です。スタックの内訳は3. 2 EEPROMエミュレーション・プログラム使用リソースを参照してください。
EEPROMエミュレーション・プログラム用RAM （アセンブリ言語：8バイト）	EEPROMエミュレーション専用のRAMとして，読み出し／書き込みデータを一時的に保持する領域が必要です。このデータ・バッファとして内部高速RAM上に左記の領域を確保してください。
ウォッチドック・タイマ（WDT）の動作	EEPROMエミュレーション・プログラムの実行時，フラッシュ・メモリ制御処理を実行している間は命令を実行できないため，フラッシュ・メモリ制御処理ではWDTのカウンタをクリアしています。その際，WDTによるオーバフローが発生しないように，オーバフロー時間を10 ms以上に設定してください。
リセットの禁止	EEPROMエミュレーション・プログラム動作時に，このマイコンをリセットしないでください。リセット発生時にアクセスしているフラッシュ・メモリのデータが不定となります。
電源切断／電源瞬断の禁止	EEPROMエミュレーション・プログラム動作時は，安定した電圧をマイコンに供給してください。電源切断／電源瞬断時にアクセスしているフラッシュ・メモリのデータが不定となります。

注意1. EEPROMエミュレーション・プログラムの書き込み処理中は，すべての割り込みが禁止されています。また，EEPROMエミュレーション・プログラムの書き込み処理後の割り込みのマスク状態は，書き込み処理前の状態に戻り，割り込みが許可されています。

★ 2. オンチップ・デバッグ機能使用時は，デバッグ用モニタ・プログラムが配置される領域に，EEPROMエミュレーションのデータ領域などを配置しないでください。

例．フラッシュ・メモリが4 Kバイトの製品で，2ブロックのデータ領域を配置する場合
ブロック14, 15は，オンチップ・デバッグ機能で使用するため，ブロック14, 15以外のフラッシュ・メモリ領域にデータ領域を配置してください。

備考　ウォッチドッグ・タイマの動作，割り込みのマスクおよび電源切断／電源瞬断の禁止については，78K0S/Kx1+各製品のユーザーズ・マニュアルにあるフラッシュ・メモリの章のセルフ・プログラミング機能の注意事項を参照してください。

★ オンチップ・デバッグ機能については，78K0S/Kx1+各製品のユーザーズ・マニュアルにあるオンチップ・デバッグ機能の章を参照してください。

★ 2. 3　サンプル・プログラムの入手方法

サンプル・プログラムは，下記URLから入手してください。

http://www.necel.com/micro/ja/designsupports/sampleprogram/78k0s/low_pin_count/index.html

第3章　EEPROMエミュレーション・プログラム（固定長単一データ方式，アセンブリ言語）

このプログラムは，フラッシュ・メモリのセルフ・プログラミング機能を使用することにより，フラッシュ・メモリをEEPROMとしてデータの格納などに使用するためのアプリケーション・プログラムです。

3. 1　EEPROMエミュレーション・プログラム構成

このプログラムのファイル構成を表3－1に示します。

表3－1　ファイル構成

ファイル名	機能	種別
EEPROM.asm	EEPROMエミュレーション処理 EEPROMエミュレーションのための読み出し／書き込みだけでなく，データの検索，ブロックの移動などの処理を行います。	アセンブラ・ソース

3. 2　EEPROMエミュレーション・プログラム使用リソース

このプログラムで使用するリソースを表3－2に示します。

表3－2　リソース

リソース	内　容	
RAM	EEPROMエミュレーション用RAM	8バイト
	スタック	22バイト
	EEPROM書き込み処理	22バイト
	EEPROM読み出し処理	12バイト
ROM	EEPROMエミュレーション処理	295バイト
★	フラッシュ・メモリ制御処理	177バイト
★	合計	472バイト

3. 3　EEPROMエミュレーション・プログラム使用方法

この章で説明するEEPROMエミュレーションは，最低2ブロック以上のフラッシュ・メモリを使用し，EEPROMエミュレーションにより，2バイトのデータ更新，参照を行うことができます。

このEEPROMエミュレーション・プログラムをユーザ・アプリケーションに組み込み，実行条件を満たし（2. 2 EEPROMエミュレーション・プログラム実行条件を参照），所定のプログラムを実行することにより，EEPROMエミュレーションを実現することができます。

EEPROMエミュレーションを行う方法として，固定単一データ長方式，アセンブリ言語のプログラム使用方法を次に示します。

3. 3. 1　ユーザ設定初期値

EEPROMエミュレーション・プログラムの初期設定値として，次の項目をユーザにて必ず設定してください。

・EEPROMとして使用する先頭ブロック番号

・EEPROMとして使用するブロック数

上記の初期設定項目はEEPROM.asmにあります。

（1）EEPROMとして使用するブロック番号

EEPROMエミュレーションで使用するブロック番号を指定してください。設定するブロックは，2ブロック以上連続している必要があります。ユーザ・プログラムの領域と重ならないように，ブロックを設定してください。

EEPROMとして使用するブロック数を増やすことにより，EEPROMとしての書き換え回数を増やすことが可能です。EEPROMエミュレーションにて使用するデータ数にかかわらず，プログラム領域として使用していない領域は，すべてEEPROMエミュレーションに設定することをお勧めします。

例1.　ブロック14, 15の2ブロックをEEPROM用として使用する場合

```
EEPROM_BLOCK EQU (14)
EEPROM_BLOCK_NO EQU (2)
```

2.　ブロック12, 13, 14, 15の4ブロックをEEPROM用として使用する場合

```
EEPROM_BLOCK EQU (12)
EEPROM_BLOCK_NO EQU (4)
```

（2）ユーザ使用データ長

ユーザにてEEPROMに格納したいデータ長を設定します。

EEPROMエミュレーションはデリミタが必要ですので，データ・サイズ＋デリミタ（1バイト）で設定してください。

例　格納するデータが2バイトの場合

```
LENG  EQU  (3)      ; データ長（デリミタ込み）
```

★　**（3）消去リトライ回数**

フラッシュ・メモリのブロック消去回数時間（MAX.値）に合わせてリトライ回数を設定します。

このマニュアルでのサンプル・プログラムでは，T_A =−40～＋85℃**，**4.5 V≦V_{DD}≦5.5 V**，**N_{ERASE}≦1000**回の設定（**4.9 s**）となります。**

他の条件で設定する場合，ブロック消去時間÷8.5 ms**を上回るリトライ回数に変更してください。**1**回の消去時間は**8.5 ms**です。**

3. 3. 2　EEPROMエミュレーション・ユーザ処理呼び出し方法

ユーザ・プログラムよりEEPROM**エミュレーションを行うための処理として，**EEPROM**読み出し／書き込みの**2**種類の処理を用意しています。**

＜EEPROM**読み出し／書き込み処理**＞

EEPROM**読み出し／書き込み処理は，データ・アドレスの所定の引数を設定して呼び出すことにより，容易に**EEPROM**エミュレーションを行うことができます。**

EEPROM**読み出し／書き込み処理のアセンブラ版は**main.asm**，**C**言語版は**main.c**に含まれています。**

EEPROM**エミュレーションは，読み出し／書き込み共通で下記の変数／構造体（**RAM**）を用います。**

main.asm**（アセンブラ版）の場合，下記に定義した変数**EEPROM_DATA**を用いてください。**

```
EEPROM_DATA:       DS   2    ; データ
EEPROM_DELIMITER:  DS   1    ; デリミタ
```

main.c**（C言語版）の場合，下記に定義した構造体**eeprom_data**を用いてください。**

```
Struct eeprom_data{
  unsigned char uc_eeprom_data[2]    ; データ
  unsigned char uc_delimiter         ; デリミタ
};
```

（1）EEPROM**読み出し処理（**__eeprom_read**）：設定されたサイズのデータを**EEPROM**領域から読み出します。**

main.asm**（アセンブラ版）の場合**

- **引数**

  ：EEPROM_DATA**のアドレスを**AX**レジスタに格納して**__eeprom_read**関数をサブルーチン・コールしてください。**

- **戻り値（**CY**フラグ）**

  ：データ読み出し正常終了（CY = 0**）または，データ読み出し異常終了（**CY = 1**）が戻ります。異常終了の場合は，指定したデータが一度も書き込まれていない場合も異常となります。**

main.c（C 言語版）の場合

・引数

：eeprom_data構造体へのアドレスを引数として_eeprom_read関数を実行してください。

・戻り値（エラー・フラグ）

：データ読み出し正常終了（戻り値 = 0）または，データ読み出し異常終了（戻り値 = 1）が戻ります。
異常終了の場合は，指定したデータが一度も書き込まれていない場合も異常となります。

(2) EEPROM書き込み処理（__eeprom_write）：設定されたサイズのデータをEEPROM領域に書き込みます。

main.asm（アセンブラ版）の場合

・引数

：書き込みたいデータをEEPROM_DATAに，デリミタをEEPROM_DELIMITERにそれぞれ設定後，
EEPROM_DATA のアドレスをAXレジスタに格納して__eeprom_ write関数を実行してください。

・戻り値（CYフラグ）

：データ書き込み正常終了（CY = 0）または，データ書き込み異常終了（CY = 1）が戻ります。異常
終了の場合は，指定したデータが一度も書き込まれていない場合も異常となります。

main.c（C 言語版）の場合

・引数

：eeprom_data構造体へのアドレスを引数として_eeprom_write関数を実行してください。

・戻り値（エラー・フラグ）

：データ書き込み正常終了（戻り値 = 0）または，データ書き込み異常終了（戻り値 = 1）が戻ります。

3. 4　EEPROMエミュレーション・プログラム説明

3. 4. 1　EEPROMエミュレーション・ユーザ・アクセス処理

EEPROMエミュレーションにて読み出し／書き込みを行うために使用する，ユーザからのアクセス処理を表3－3，表3－4に示します。

表3－3　EEPROM読み出し処理

（a）アセンブラ版

処理名	__eeprom_read（ユーザ・アクセス関数）
ROM容量	29バイト
スタック容量	5レベル（10バイト）
入力	AX ：変数のアドレス
戻り値	**正常終了：**CY = 0 **異常終了：**CY = 1
動作説明	**最新のデータを格納アドレスに読み出す。** 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックから最新データのアドレスを検索する。 3：検索アドレスから最新データを読み出す。

（b）C言語版

処理名	_eeprom_read（ユーザ・アクセス関数）
ROM容量	29バイト
スタック容量	5レベル（10バイト）
入力	AX ：構造体のポインタ
戻り値	**正常終了：エラー・フラグ** = 0 **異常終了：エラー・フラグ** = 1
動作説明	**最新のデータを格納アドレスに読み出す。** 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックから最新データのアドレスを検索する。 3：検索アドレスから最新データを読み出す。

表3－4　EEPROM書き込み処理

（a）アセンブラ版

処理名	__eeprom_write（ユーザ・アクセス関数）
ROM容量	58バイト
スタック容量	11レベル（22バイト）
入力	AX ：変数のアドレス
戻り値	正常終了：CY = 0 異常終了：CY = 1
動作説明	格納アドレスからEEPROMにデータを書き込む。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックがない場合，最初の指定ブロックを有効に設定する。 3：有効ブロックへの書き込み可能アドレスを検索する。 4：有効ブロックが満杯で書き込めない場合は，次のブロックへの移行操作を行う。 5：書き込みデータを作成する。 6：有効ブロックへ書き込む。

（b）C言語版

処理名	_eeprom_write（ユーザ・アクセス関数）
ROM容量	58バイト
スタック容量	11レベル（22バイト）
入力	AX ：構造体のポインタ
戻り値	正常終了：エラー・フラグ = 0 異常終了：エラー・フラグ = 1
動作説明	格納アドレスからEEPROMにデータを書き込む。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックがない場合，最初の指定ブロックを有効に設定する。 3：有効ブロックへの書き込み可能アドレスを検索する。 4：有効ブロックが満杯で書き込めない場合は，次のブロックへの移行操作を行う。 5：書き込みデータを作成する。 6：有効ブロックへ書き込む。

3. 4. 2　EEPROM エミュレーション制御処理（内部処理用）

EEPROMエミュレーションにてエミュレーション制御を行うための処理を表3－5から表3－9に示します。

表3－5　EEPROM使用中ブロック検索処理

処理名	EEPROMUseBlockSearch
ROM容量	28バイト
スタック容量	2レベル（4バイト）
入力	なし
出力	正常終了：CY = 0, A = ブロック・テーブル番号（01H - FEH） 異常終了：CY = 1, A = 最終ブロックの次
使用レジスタ	A
動作説明	EEPROMとして割り振られているフラッシュ・メモリの現在使用中のブロックを検索する。

表3－6　EEPROMブロックの初期化処理

処理名	EEPROMBlockInit
ROM容量	17バイト
スタック容量	6レベル（12バイト）
入力	なし
出力	正常終了：CY = 0 異常終了：CY = 1
使用レジスタ	A, X, B, C, D, E
動作説明	EEPROM対象ブロックのいずれも有効ブロックでない場合，最初の指定ブロックを使用中（有効）に設定する。 正常に確保できれば，CY = 0で戻る。 正常に確保できなければ，CY = 1で戻る。

表3－7　EEPROM使用ブロックの変更処理

処理名	EEPROMUseBlockChange
ROM容量	59バイト
スタック容量	6レベル（12バイト）
入力	A = 使用中ブロック番号
出力	正常終了：CY = 0，C = ブロック・テーブル番号（02H - FEH），ゼロ・フラグ（Z）= 1 異常終了：CY = 1，Z（ZEROフラグ）= 0
使用レジスタ	A, X, B, C, D, E
動作説明	使用中ブロックのデータが満杯時に，次に使用するブロックの検索およびデータのコピーを行う。 1：次に使用するブロックの設定をする。 2：次に使用するブロックを消去する。 3：有効ブロックから，次のブロックに最新データを転送する。 4：次に使用するブロックを有効に設定する。 5：現在有効になっているブロックを無効に設定する。 6：新しいブロック番号「CurrentB_No」へ格納する。

表3－8　EEPROM使用ブロックのデータ書き込み先頭検索処理

処理名	EEPROMWriteTopSearch
ROM容量	44バイト
スタック容量	3レベル（6バイト）
入力	A：検索ブロック・テーブル番号
出力	検索成功：CY = 0，AXに書き込みアドレスをセットする 検索失敗：CY = 1
使用レジスタ	A, X, B, D, E
動作説明	指定ブロックの書き込み可能なアドレスを検索する。 データ領域が0FFHでブロック内に収まる場合のみ正常終了となる。

表3－9　EEPROM最新データ検索処理

処理名	EEPROMDataSearch
ROM容量	33バイト
スタック容量	3レベル（6バイト）
入力	A：使用中ブロック・テーブル番号
出力	正常終了：CY = 0，DE = 最新データのアドレス 異常終了：CY = 1，E = 0
使用レジスタ	A, X, D, E
動作説明	最新データの格納アドレスを読み出す。

3. 4. 3　フラッシュ・メモリ制御処理

EEPROMエミュレーションにてフラッシュ・メモリ制御を行うための処理を表3－10から表3－17に示します。

表3－10　ブロック消去処理

処理名	SelfFlashBlockErase
ROM容量	29バイト
スタック容量	3レベル（6バイト）
入力	A：消去ブロック番号
出力	正常終了：CY = 0 異常終了：CY = 1
使用レジスタ	B
動作説明	指定ブロックの消去およびブランク・チェックを行う。

★

表3－11　セルフ・プログラミング・モードから通常モードへの遷移処理

処理名	SelfFlashModeOff
ROM容量	31バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	なし
使用レジスタ	A, X
動作説明	セルフ・プログラミング・モードを解除する。

表3－12　通常モードからセルフ・プログラミング・モードへの遷移処理

処理名	SelfFlashModeOn
ROM容量	35バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	なし
使用レジスタ	A, X
動作説明	セルフ・プログラミング・モードに設定する。

表3－13　ブロック消去処理

処理名	FlashBlockErase
ROM容量	15バイト
スタック容量	1レベル（2バイト）
入力	A：ブロック番号
出力	正常終了：ゼロ・フラグ（Z）　= 1 異常終了：ゼロ・フラグ（Z）　= 0
使用レジスタ	A
動作説明	指定ブロックを消去する。

表3－14　フラッシュ・セルフ・プログラミング機能呼び出し処理

処理名	SubFlashSelfPrg
ROM容量	12バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	正常終了：ゼロ・フラグ（Z）　= 1 異常終了：ゼロ・フラグ（Z）　= 0
使用レジスタ	A
動作説明	フラッシュ・セルフ・プログラミング機能呼び出し

表3－15　ブロック・ブランク・チェック処理

処理名	FlashBlockBlankCheck
ROM容量	17バイト
スタック容量	1レベル（2バイト）
入力	A：対象ブロック番号
出力	正常終了：ゼロ・フラグ（Z） = 1 異常終了：ゼロ・フラグ（Z） = 0
使用レジスタ	A
動作説明	指定ブロックのブランク・チェックを行う。

表3－16　ブロックの有効設定処理

処理名	SetValid
ROM容量	9バイト
スタック容量	1レベル（2バイト）
入力	A：ブロック番号
出力	正常終了：ゼロ・フラグ（Z） = 1 異常終了：ゼロ・フラグ（Z） = 0
使用レジスタ	A, X, B, C, D, E
動作説明	使用ブロックを有効に設定する。

表3－17　EEPROMデータ書き込み処理

処理名	FlashEEPROMWrite
ROM容量	56バイト
スタック容量	5レベル（10バイト）
入力	DE：書き込み先頭アドレス C ：書き込みデータ数 AX：書き込みデータ格納アドレス
出力	正常終了：ゼロ・フラグ（Z） = 1 異常終了：ゼロ・フラグ（Z） = 0
使用レジスタ	D, E
動作説明	EEPROMでの書き込みおよび内部ベリファイを行う。

★

3. 5　EEPROMエミュレーション・プログラムのフロー・チャート

3. 5. 1　EEPROM エミュレーション・アクセス処理のフロー・チャート

EEPROM エミュレーションにて，読み出し／書き込みを行うための，ユーザからのアクセス処理のフロー・チャートを図 3－1，図 3－2 に示します。

図 3－1　EEPROM 読み出し処理フロー・チャート

［概　　要］

構造体で定義されたデータを指定された格納アドレスに読み出す

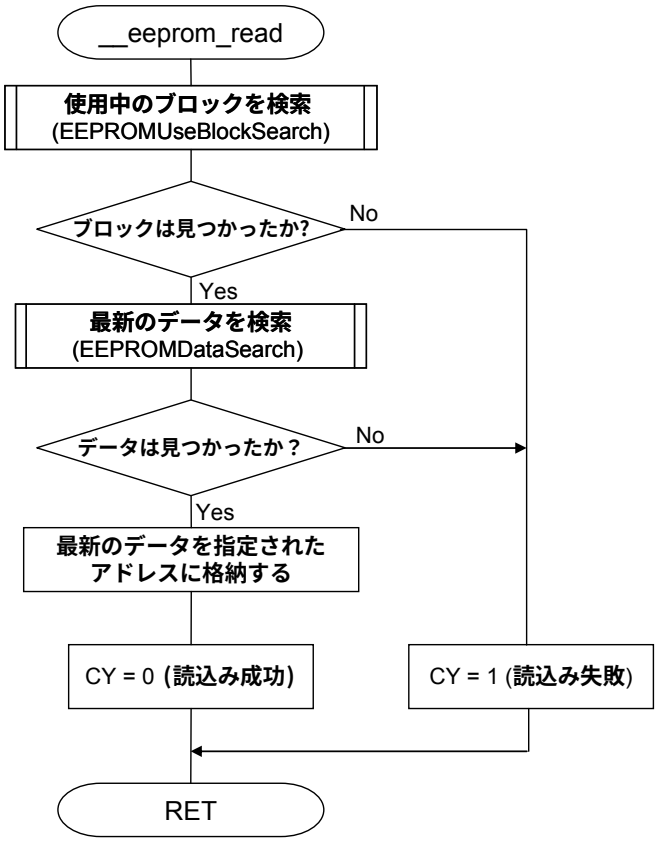
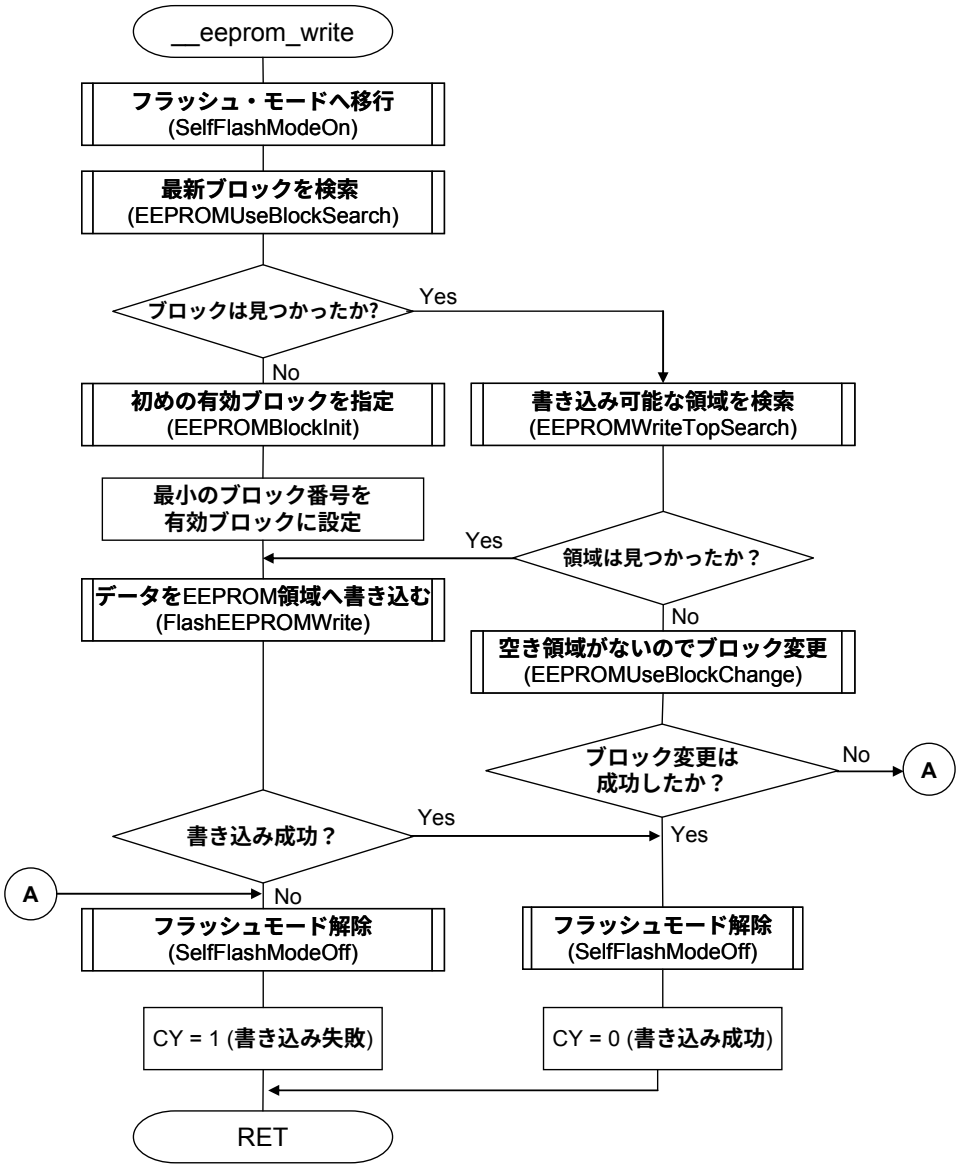


図 3－2　EEPROM 書き込み処理フロー・チャート

［概　　要］

有効ブロックに指定番号のデータを格納アドレスから書き込む



3. 5. 2　EEPROM エミュレーション制御処理のフロー・チャート

EEPROM エミュレーションにて，エミュレーション制御を行うための制御処理のフロー・チャートを図 3－3～図 3－7 に示します。

図 3－3　EEPROM 使用中ブロック検索処理フロー・チャート

［概　　要］

EEPROM として割り振られているフラッシュ・メモリの現在使用中のブロックを検索する

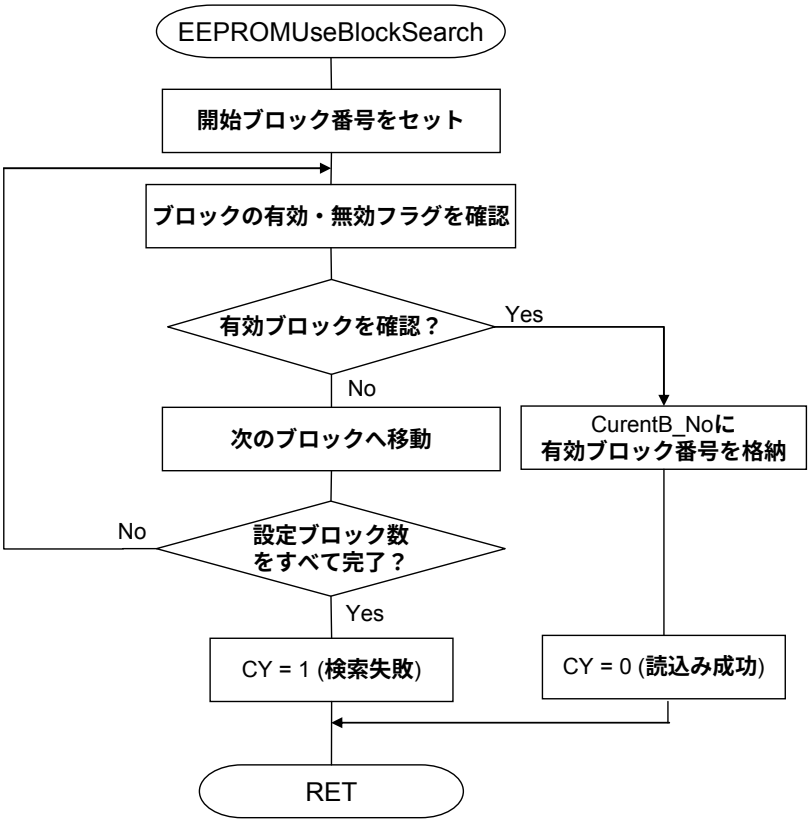


図 3－4　EEPROM **ブロックの初期化処理フロー・チャート**

[概　　要]

EEPROM **対象ブロックのいずれも有効ブロックでないときに，最初の指定ブロックを有効に設定する。**

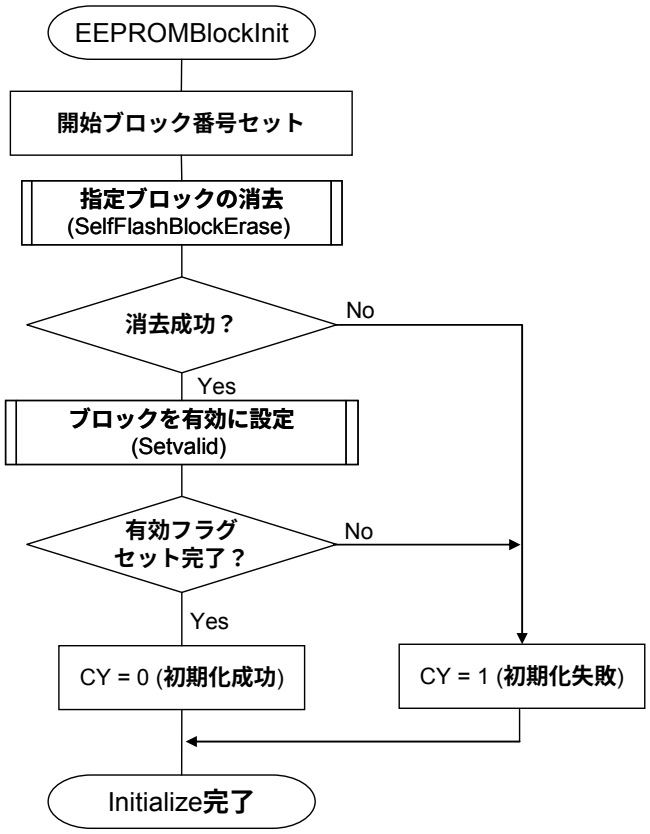


図 3－5　EEPROM 使用ブロックの変更処理フロー・チャート

[概　　要]

使用中ブロックのデータが満杯時，次に使用するブロックの検索および新ブロックへのデータ・コピーを行う。

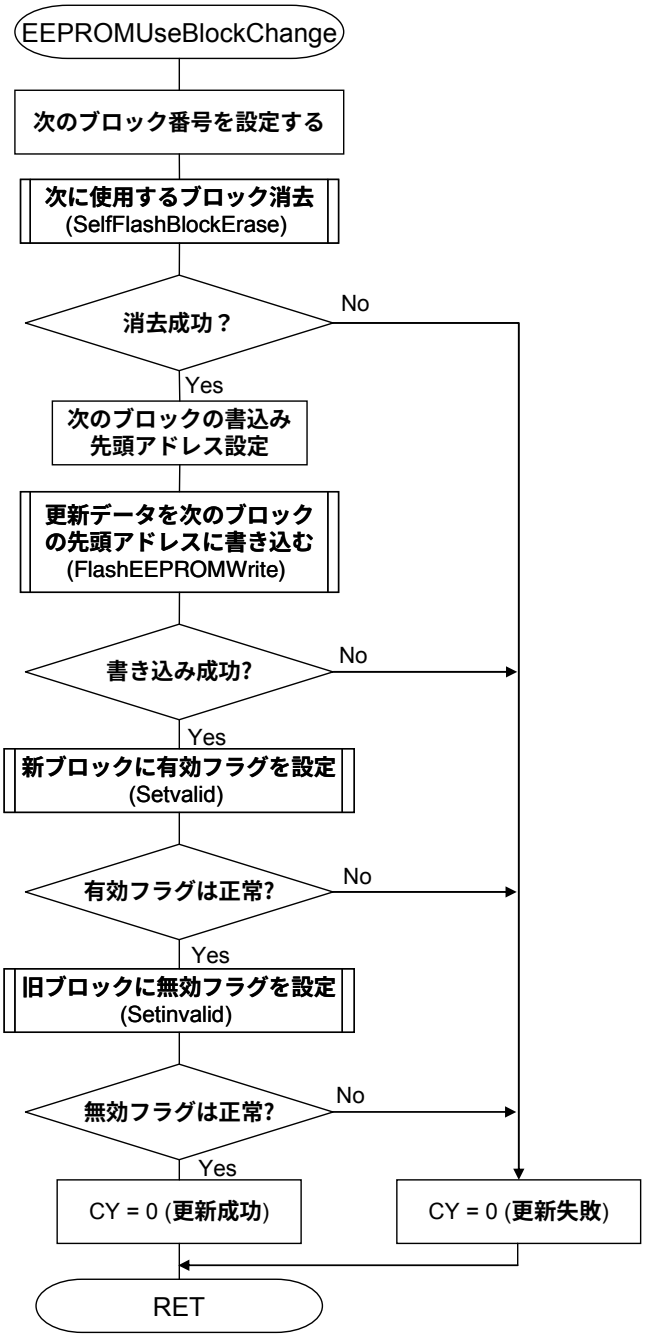


図 3－6　EEPROM **使用ブロックのデータ書き込み先頭検索処理フロー・チャート**

[概　　要]

指定ブロックの書込み可能な領域を探す。

データ領域が 0xFF **で，ブロック内に次データが収まる場合のみ正常終了となる。**

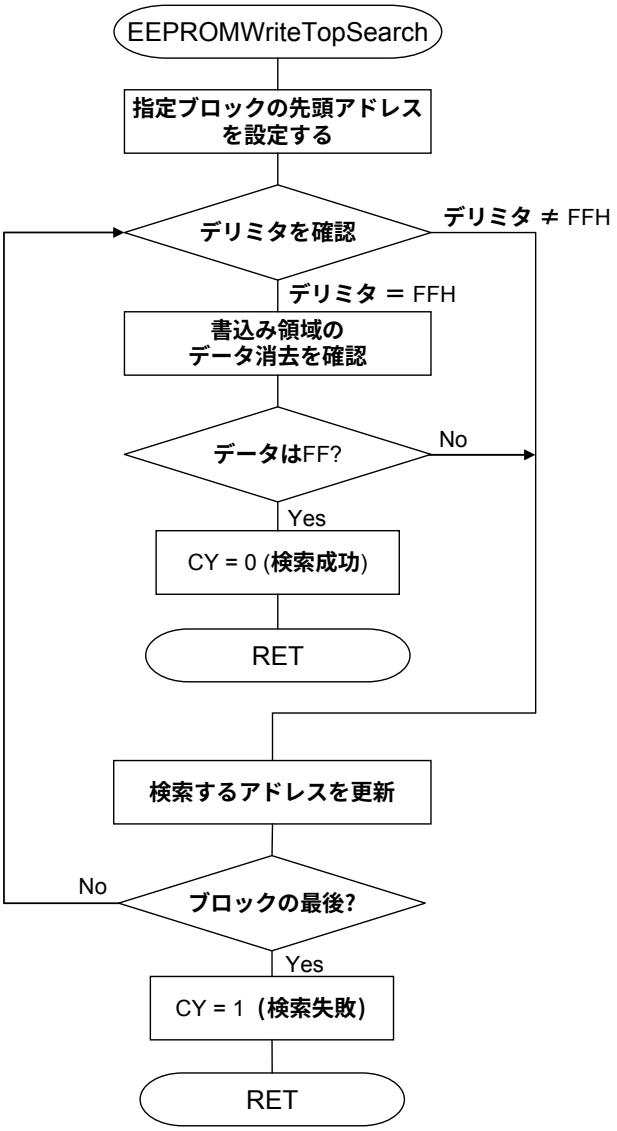
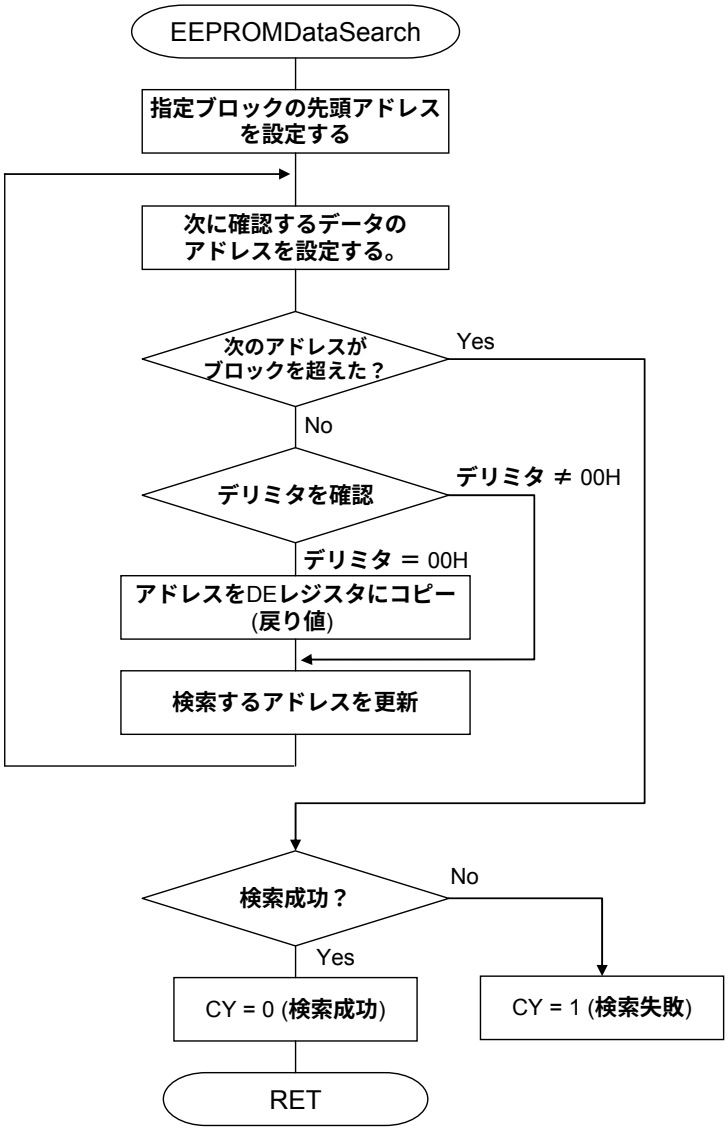


図 3－7　EEPROM 最新データ検索処理フロー・チャート

［概　　要］

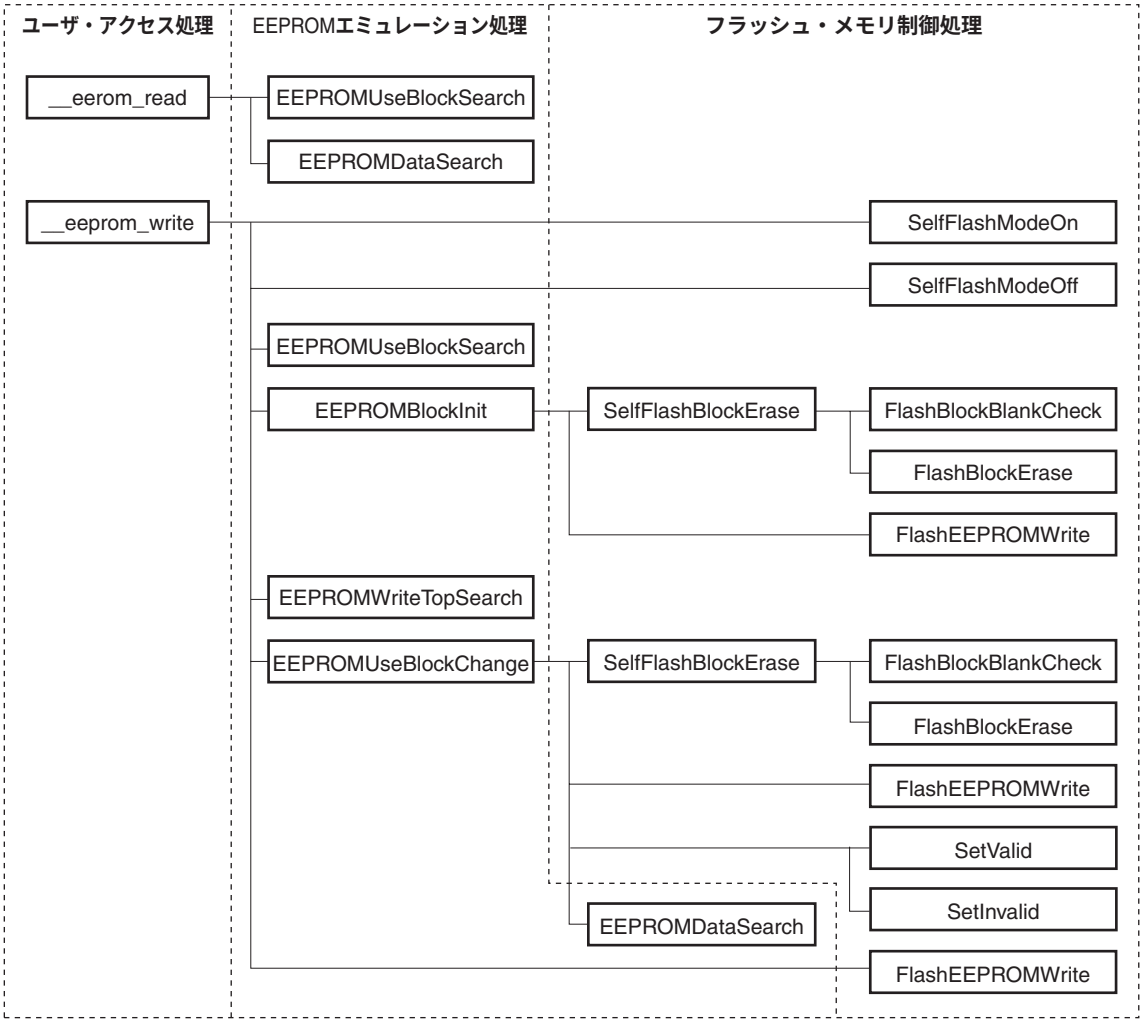
最新データの格納アドレスを読み出す。



3. 6　EEPROM エミュレーションの処理一覧

EEPROM エミュレーション処理のコール・ツリーを次に示します。

図 3－8　**コール・ツリー**



第4章　EEPROMエミュレーション機能（固定長複数データ方式）

4. 1　EEPROMエミュレーションの基本仕様

EEPROMエミュレーションとは，フラッシュ・メモリのセルフ・プログラミング機能を使用することで，フラッシュ・メモリの一部を書き換え可能なデータROMとして使用するための機能です。サンプル・プログラムとユーザ・プログラムを組み合わせることでEEPROMのように読み出し／書き込み処理を実行できます。

なお，内蔵のフラッシュ・メモリの書き換え回数には制限があるので，データ長と書き換え回数に制約があります。次にサンプル・プログラムの基本仕様，書き換え回数の計算方法を示します。

サンプル・プログラムの基本仕様，書き換え回数の計算方法

・保存するデータ・フォーマット

データ番号（1バイト）	データ（2バイト）	デリミタ（1バイト）

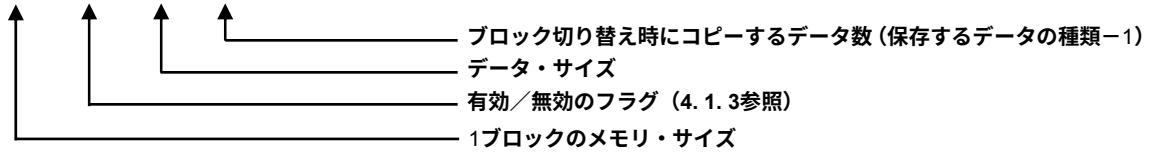
↑

保存するデータを区別する番号（サンプル・プログラムでは2種類のデータ番号を使用）

備考　データのサイズは，1バイトから設定可能です。上限は使用できるRAMサイズに依存します。

・フラッシュ・メモリ・ブロックに書き換え可能な回数

（256 － 2）÷ 4 － 1 ＝ 62 回（小数点以下切り捨て）



・EEPROM領域として使用するブロック数

2ブロック（MIN.）を使用

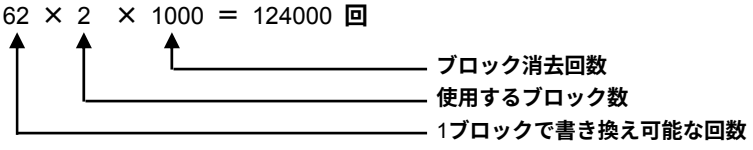
備考　ブロック消去中に電源切断／電源瞬断などのトラブルによるデータ損失を防ぐために必要です。

★　・1ブロックの消去回数

1000回

★　・最大書き換え回数

62 × 2 × 1000 ＝ 124000 回



サンプル・プログラムでは，2バイト単位のデータを扱っており，データの終わりを示すデリミタ（1バイト）との組み合わせで1ブロック（256バイト）当たり62回の書き換えが可能です。また，予期しない電源電圧降下などによる損失を防ぐためには連続した2ブロック以上の領域が必要なので，2ブロックを使用した場合，合計124
★ 回の書き換えが可能です。さらにサンプル・プログラムでは，1ブロックの消去回数を1000回まで許可しているので，2ブロックを使用した場合は，最大124000回の書き換えが可能になります。

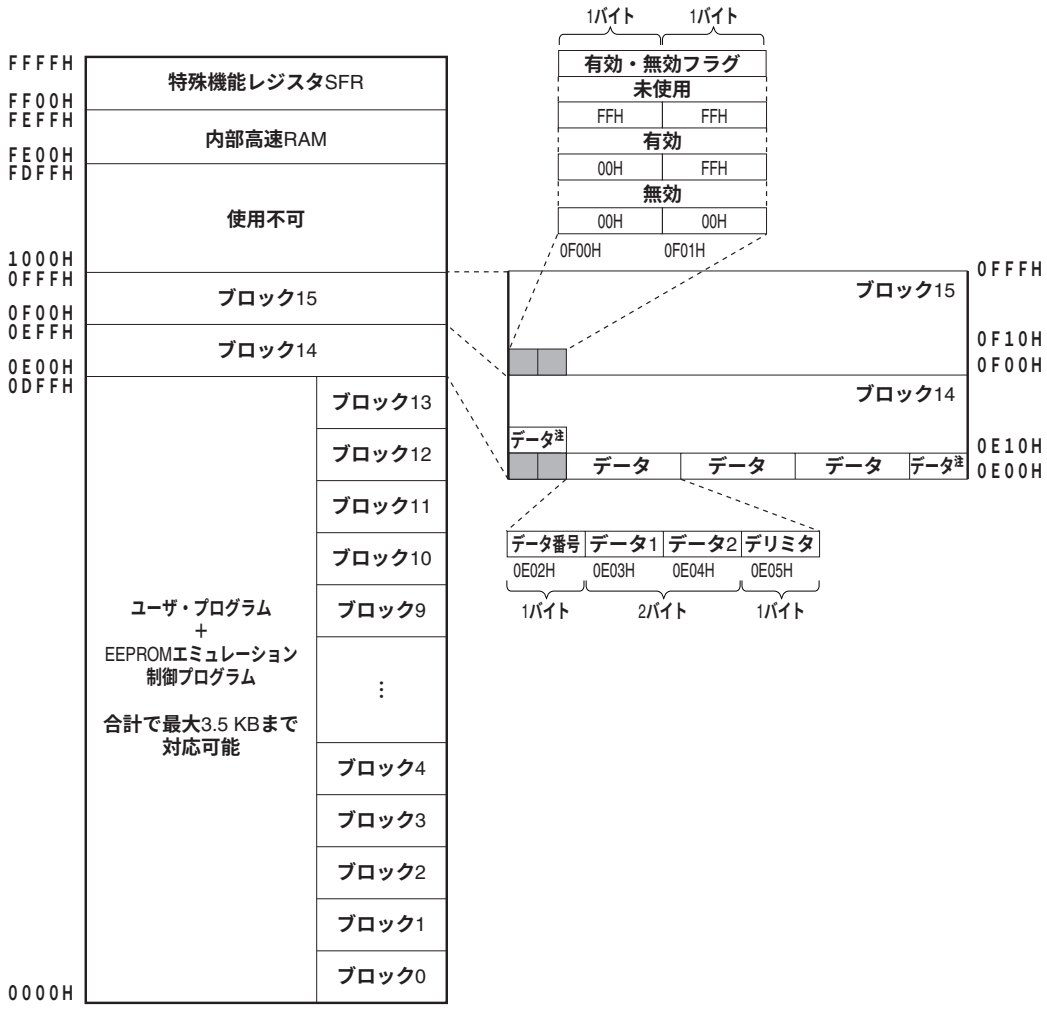
ただし，EEPROM格納データ（データ＋デリミタ）の全種類の合計バイト数が格納ブロックのサイズに対して十分余裕がない（データの合計バイト数がブロックの1/2のサイズより大きい）場合，EEPROMエミュレーションとして正しく機能しないことがあります。具体的には，データのブロック間転送が多発し，データの書き換え回数が著しく低下する可能性があります。

また，データを格納するためのフラッシュ・メモリの配置は，連続した2ブロック以上を確保する必要がありますが，これらのブロックはユーザにて自由に設定することができます。

図4－1にメモリ・マップとデータ構造例（ユーザ・プログラム・サイズが3.5 KBで，EEPROMエミュレーション用としてブロック14, 15を設定した場合）を示します。

図4－1　メモリ・マップとデータ構造例

（ユーザ・プログラム・サイズが3.5 KBで，EEPROMエミュレーションのデータ領域としてブロック14, 15を設定した場合）



注　データは続けて格納されます。

備考　図中に示したデータ構造はサンプル・プログラムを使用した場合の例です。

4. 1. 1　EEPROMエミュレーション・データ・ブロック

EEPROMエミュレーション・プログラムは，データを格納するためのブロックが連続2ブロック以上必要です。

使用するブロックは，ユーザ・プログラム領域，EEPROMエミュレーション制御プログラム領域と重ならなければ，ユーザによって自由に設定できます。

4. 1. 2　データ構造

EEPROMエミュレーションにより格納されるデータは，データ番号（1バイト），データ（1バイト単位），デリミタ（1バイト）で構成されています。このデータ構造が基本で，EEPROMエミュレーション・プログラムで使用する構造体／変数もこのデータ構造で用いられます（5. 3. 2　EEPROMエミュレーション・ユーザ処理呼び出し方法参照）。

図4－2　データ構造

1バイト	1バイト単位	1バイト
データ番号	データ	デリミタ

(1) データ番号

データ番号は，データの読み出し／書き込みを行う際にデータを区別するための番号です。データ番号は00H～FEHが有効です。

データごとに，それぞれデータ番号をユーザにて割り振って使用してください。

データは，基本的に書き込みプログラムが実行されるたびに，ブロックの先頭から順番に書き込まれます。

最新のデータを読み出すためには，ブロックの先頭から検索を行い，同一データ番号かどうかを検索します。同一データ番号のデータが複数見つかった場合は，データの終端に近いデータが最新のデータになります。

データ番号がFFHの場合，データの終端と判定します。フラッシュ・メモリは消去後，全データがFFHとなるため，データ番号の位置でFFHが読み出された場合にデータの終端と判定し，検索を終了します。したがって，データ番号として00H～FEHが有効となります。

備考　サンプル・プログラムでは2種類のデータ番号を使用しています。

(2) データ

保存する任意の値で，00H-FFHを設定することができます。サイズは，1バイトから設定可能ですが，上限は使用できるRAMサイズに依存します。また，サイズが大きくなるにつれて書き換え可能な回数は減りますので，ご注意ください。サンプル・プログラムでは2バイト（データ長（LENG）を4に定義）に指定してあります。

(3) デリミタ

デリミタは，00Hの固定値です。デリミタは，データ書き込み中に電源断などのトラブルにより，データが正常に書き込めなかった場合を検出するために，書き込まれます。最後に書き込まれる領域にデリミタを書き込むことにより，そのデータの書き込みが正常に終了したかを判定します。デリミタ（00H）が正しく読み出せない場合は，データ書き込み中のトラブル発生が予想されるため，そのデータを使用しません。

検索により，最新のデータがデリミタ異常であるとわかった場合は，その前に書かれた，デミリタが正

常なデータ（次に終端に近いデータ）が最新データとして読み出されます。

(4) データの格納および検索動作の通常の流れ

データの格納および検索動作の通常の流れについて，次に示します（EEPROMエミュレーション指定ブロックが14, 15ブロックの場合）。

（状態1）ブロック14を有効ブロックにする。

ブロック14	+0	
有効フラグ	00H	0E00H
無効フラグ	FFH	0E01H
データ	FFH	0E02H

（状態2）データ番号1（データ：11H, 22H）を書き込む。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データ	01H	11H	22H	00H	0E02H
⋮	FFH				0E06H

（状態3）　データ番号2（データ：22H, 33H）を書き込む。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データ	01H	11H	22H	00H	0E02H
データ	02H	22H	33H	00H	0E06H
⋮	FFH				0E0AH

（状態4）　データ番号2（データ：20H, 30H）を書き込む。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データ	01H	11H	22H	00H	0E02H
データ	02H	22H	33H	00H	0E06H
データ	02H	20H	30H	00H	0E0AH
⋮	FFH				0E0EH

（状態5）　データ番号2の読み出し。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データa	01H	11H	22H	00H	0E02H
データb	02H	22H	33H	00H	0E06H
データc	02H	20H	30H	00H	0E0AH
データd	FFH				0E0EH

読み出し方法

① データaは，データ番号が異なるので，次のデータに移ります。

② データbは，データ番号が一致するのでデリミタを確認し，デリミタが00H（正常）なので，

データ2バイトを最新データとして格納し，次のデータに移ります。

③　データcは，データ番号が一致するのでデリミタを確認し，デリミタが00H（正常）なので，データ2バイトを最新データとして格納し，次のデータに移ります。

④　データdは，データ番号がFFH（終端）なので，読み出しを終了します。

⑤　格納されている最新データcを読み出し値とします。

データ格納時に電源断などのトラブルが発生したときの流れについて，次に示します（EEPROMエミュレーション指定ブロックが14, 15ブロックの場合）。

（状態1）ブロック14を有効ブロックにする。

ブロック14	+0	
有効フラグ	00H	0E00H
無効フラグ	FFH	0E01H
データ	FFH	0E02H

（状態2）データ番号1（データ：11H, 22H）を書き込む。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データ	01H	11H	22H	00H	0E02H
⋮	FFH				0E06H

（状態3）データ番号1（データ22H, 33H）の書き込み途中に電源断が発生し，デリミタが正しく書き込めない（00H以外のデリミタが書き込まれた）

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データ	01H	11H	22H	00H	0E02H
データ	01H	22H	33H	01H	0E06H
⋮	FFH				0E0AH

（状態4）データ番号1の読み出し。

ブロック14	+0	+1	+2	+3	
有効フラグ	00H				0E00H
無効フラグ	FFH				0E01H
データa	01H	11H	22H	00H	0E02H
データb	01H	22H	33H	01H	0E06H
データc	FFH				0E0AH

読み出し方法

①　データaは，データ番号が一致するのでデリミタを確認し，デリミタが00H（正常）なので，データ2バイトを最新データとして格納し，次のデータに移ります。

②　データbは，データ番号が一致するのでデリミタを確認し，デリミタが01H（異常）なので，次のデータに移ります。

③　データcは，データ番号がFFH（終端）なので，読み出しを終了します。

④　格納されている有効な最新データaを読み出し値とします。

4.1.3　有効／無効フラグ

有効／無効フラグは，ブロックの先頭に，1バイトずつ計2バイトのデータとして配置されています。有効／無効フラグは，そのブロックに格納されている各データの有効／無効状態を示しています。

有効／無効フラグの値がそれぞれ，有効フラグ = 00Hかつ無効フラグ = FFHの場合，そのブロックは有効となります。それ以外の場合，そのブロックは無効となります。

有効となっているブロックに順次データを格納しますが，そのブロックが満杯になった場合，次にデータを格納するブロックを検索し（このために最低でも2ブロック必要），次のブロックにデータ（データ番号ごとの最新データのみ）を転送します。転送が完了した時点で，有効／無効フラグの設定により，次のブロックを有効にし，これまで有効だったブロックを無効にします。この手順により，ブロックが満杯となり次のブロックにデータを転送している最中に，電源断などのトラブルが発生した場合でも，それまでのデータは確保され，データの消失は起こらない構造となっています。

有効／無効フラグ動作の流れについて，次に示します。

（状態1）初期状態

	ブロックn		ブロックn＋1
有効フラグ	FFH	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	FFH	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態2）ブロックnの有効フラグに00Hを書き込み

	ブロックn		ブロックn＋1
有効フラグ	00H	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	FFH	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態3）ブロックnにデータ書き込み

	ブロックn		ブロックn＋1
有効フラグ	00H	有効フラグ	FFH
無効フラグ	FFH	無効フラグ	FFH
データ	Data	データ	FFH
⋮	⋮	⋮	⋮
データ	FFH	データ	FFH

（状態4）ブロックnのデータが満杯

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	FFH
無効フラグ	FFH
データ	FFH
⋮	⋮
データ	FFH

（状態5）ブロックn＋1に更新されていないデータ番号の最新データを転送し，その後，更新する最新データを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	FFH
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態6）ブロックn＋1の有効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態7）ブロックnの無効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

（状態8）ブロックn＋1のデータが満杯

	ブロックn
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態9）ブロックnの消去

	ブロックn
有効フラグ	FFH
無効フラグ	FFH
データ	FFH
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態10）ブロックnに更新されていないデータ番号の最新データを転送し，その後，更新する最新データを書き込み

	ブロックn
有効フラグ	FFH
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態11）ブロックnの有効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	Data

（状態12）ブロックn＋1の無効フラグに00Hを書き込み

	ブロックn
有効フラグ	00H
無効フラグ	FFH
データ	Data
⋮	⋮
データ	FFH

	ブロックn＋1
有効フラグ	00H
無効フラグ	00H
データ	Data
⋮	⋮
データ	Data

4. 2　EEPROMエミュレーション・プログラム実行条件

EEPROMエミュレーション・プログラムを実行する前に，必ず表4－1の条件をすべて満たしてください。

表4－1　EEPROMエミュレーション動作時の条件

項　目	内　容
スタック領域の確保 （アセンブリ言語：22バイト）	EEPROMエミュレーション・プログラム動作時，ユーザ・プログラムが使用しているスタックを継承して使用します。EEPROMエミュレーション・プログラム実行開始時のスタック・アドレスから，さらに左記のスタック領域が必要です。スタックの内訳は5. 2　EEPROMエミュレーション・プログラム使用リソースを参照してください。
EEPROMエミュレーション・プログラム用RAM （アセンブリ言語：11バイト）	EEPROMエミュレーション専用のRAMとして，読み出し／書き込みデータを一時的に保持する領域が必要です。このデータ・バッファとして内部高速RAM上に左記の領域を確保してください。EEPROMエミュレーション・プログラムでは，左記のプログラム用RAM以外は，すべてスタックを使用します。
ウォッチドック・タイマ（WDT）の動作	EEPROMエミュレーション・プログラムの実行時，フラッシュ・メモリ制御処理を実行している間は命令を実行できないため，フラッシュ・メモリ制御処理ではWDTのカウンタをクリアしています。その際，WDTによるオーバフローが発生しないように，オーバフロー時間を10 ms以上に設定してください。
リセットの禁止	EEPROMエミュレーション・プログラム動作時に，このマイコンをリセットしないでください。リセット発生時にアクセスしているフラッシュ・メモリのデータが不定となります。
電源切断／電源瞬断の禁止	EEPROMエミュレーション・プログラム動作時は，安定した電圧をマイコンに供給してください。電源切断／電源瞬断時にアクセスしているフラッシュ・メモリのデータが不定となります。

注意1. EEPROMエミュレーション・プログラムの書き込み処理中は，すべての割り込みが禁止されています。また，EEPROMエミュレーション・プログラムの書き込み処理後の割り込みのマスク状態は，書き込み処理前の状態に戻り，割り込みが許可されています。

★　2. オンチップ・デバッグ機能使用時は，デバッグ用モニタ・プログラムが配置される領域に，EEPROMエミュレーションのデータ領域などを配置しないでください。

例．フラッシュ・メモリが4 Kバイトの製品で，2ブロックのデータ領域を配置する場合
ブロック14, 15は，オンチップ・デバッグ機能で使用するため，ブロック14, 15以外のフラッシュ・メモリ領域にデータ領域を配置してください。

備考　ウォッチドッグ・タイマの動作，割り込みのマスクおよび電源切断／電源瞬断の禁止については，78K0S/Kx1+各製品のユーザーズ・マニュアルにあるフラッシュ・メモリの章のセルフ・プログラミング機能の注意事項を参照してください。

★　オンチップ・デバッグ機能については，78K0S/Kx1+各製品のユーザーズ・マニュアルにあるオンチップ・デバッグ機能の章を参照してください。

★ 4. 3　サンプル・プログラムの入手方法

サンプル・プログラムは，下記URLから入手してください。

http://www.necel.com/micro/ja/designsupports/sampleprogram/78k0s/low_pin_count/index.html

第5章　EEPROMエミュレーション・プログラム（固定長複数データ方式，アセンブリ言語）

このプログラムは，フラッシュ・メモリのセルフ・プログラミング機能を使用することにより，フラッシュ・メモリをEEPROMとしてデータの格納などに使用するためのアプリケーション・プログラムです。

5. 1　EEPROMエミュレーション・プログラム構成

このプログラムのファイル構成を表5－1に示します。

表5－1　ファイル構成

ファイル名	機能	種別
EEPROM.asm	EEPROMエミュレーション処理 EEPROMエミュレーションのための読み出し／書き込みだけでなく，データの検索，ブロックの移動などの処理を行います。	アセンブラ・ソース

5. 2　EEPROMエミュレーション・プログラム使用リソース

このプログラムで使用するリソースを表5－2に示します。

表5－2　リソース

リソース	内　容	
RAM	EEPROMエミュレーション用RAM	11バイト
	スタック	22バイト
	EEPROM書き込み処理	22バイト
	EEPROM読み出し処理	12バイト
ROM	EEPROMエミュレーション処理	303バイト
★	フラッシュ・メモリ制御処理	177バイト
★	合計	480バイト

5. 3　EEPROMエミュレーション・プログラム使用方法

この章で説明するEEPROMエミュレーションは，最低2ブロック以上のフラッシュ・メモリを使用し，EEPROMエミュレーションにより，データ更新，参照を行うことができます。

このEEPROMエミュレーション・プログラムをユーザ・アプリケーションに組み込み，実行条件を満たし（4. 2 EEPROMエミュレーション・プログラム実行条件を参照），所定のプログラムを実行することにより，EEPROMエミュレーションを実現することができます。

EEPROMエミュレーションを行う方法として，固定複数データ長方式，アセンブリ言語のプログラム使用方法を次に示します。

5. 3. 1　ユーザ設定初期値

EEPROMエミュレーション・プログラムの初期設定値として，次の項目をユーザにて必ず設定してください。

・EEPROMとして使用する先頭ブロック番号（定数EEPROM_BLOCKとして定義）

・EEPROMとして使用するブロック数（定数EEPROM_BLOCK_NOとして定義）

・ユーザ使用データ数，データ長（それぞれ定数DATA_NO_MAX, LENGとして定義）

上記の初期設定項目はEEPROM.asmにあります。

（1）EEPROMとして使用するブロック番号，ブロック数

EEPROMエミュレーションで使用するブロック番号を指定してください。設定するブロックは，2ブロック以上連続している必要があります。ユーザ・プログラムの領域と重ならないように，ブロックを設定してください。

EEPROMとして使用するブロック数を増やすことにより，EEPROMとしての書き換え回数を増やすことが可能です。EEPROMエミュレーションにて使用するデータ数にかかわらず，プログラム領域として使用していない領域は，すべてEEPROMエミュレーションに設定することをお勧めします。

例1.　ブロック14, 15の2ブロックをEEPROM用として使用する場合

```
EEPROM_BLOCK EQU (14)
EEPROM_BLOCK_NO EQU (2)
```

2.　ブロック12, 13, 14, 15の4ブロックをEEPROM用として使用する場合

```
EEPROM_BLOCK EQU (12)
EEPROM_BLOCK_NO EQU (4)
```

（2）ユーザ使用データ数，データ長

ユーザにてEEPROMに格納したいデータの数とデータ長を設定します。

EEPROMエミュレーションはデータ番号とデリミタが必要ですので，データ・サイズ＋データ番号（1バイト）＋デリミタ（1バイト）で設定してください。

例　2種類の2バイト・データを使用する場合

```
DATA_NO_MAX  EQU  (2)    ; データ番号0～1の2個のデータを使う場合
LENG  EQU  (4)           ; データ長（データ番号，デリミタ（計2バイト）を含んだサイズ）
```

★　**(3) 消去リトライ回数**

フラッシュ・メモリのブロック消去回数時間（MAX.値）に合わせてリトライ回数を設定します。

このマニュアルでのサンプル・プログラムでは，T_A =−40〜＋85℃，4.5 V≦V_{DD}≦5.5 V，N_{ERASE}≦1000回の設定（4.9 s）となります。

他の条件で設定する場合，ブロック消去時間÷8.5 msを上回るリトライ回数に変更してください。1回の消去時間は8.5 msです。

5. 3. 2　EEPROMエミュレーション・ユーザ処理呼び出し方法

ユーザ・プログラムよりEEPROMエミュレーションを行うための処理として，EEPROM読み出し／書き込みの2種類の処理を用意しています。

＜EEPROM読み出し／書き込み処理＞

EEPROM読み出し／書き込み処理は，変数／構造体のアドレスを所定の引数として呼び出すことにより，容易にEEPROMエミュレーションを行うことができます。

EEPROM読み出し／書き込み処理のアセンブラ版はmain.asm**，C言語版は**main.c**に含まれています。**

EEPROMエミュレーションは，読み出し／書き込み共通で下記の変数／構造体（RAM）を用います。

main.asm**（アセンブラ版）の場合，下記に定義した変数**EEPROM_DATA**を用いてください。**

```
EEPROM_NO:         DS   1    ; データ番号
EEPROM_DATA:       DS   2    ; データ
EEPROM_DELIMITER:  DS   1    ; デリミタ
```

main.c**（C言語版）の場合，下記に定義した構造体**eeprom_data**を用いてください。**

```
Struct eeprom_data{
  unsigned char uc_data_no           ; データ番号
  unsigned char uc_eeprom_data[2]    ; データ
  unsigned char uc_delimiter         ; デリミタ
};
```

(1)EEPROM**読み出し処理（**__eeprom_read**）：設定されたサイズのデータを**EEPROM**領域から読み出します。**

main.asm**（アセンブラ版）の場合**

- **引数**

  ：読み出したいデータのデータ番号をEEPROM_NO**に設定後，**EEPROM_NO**のアドレスを**AX**レジスタに格納して**__eeprom_read**関数をサブルーチン・コールしてください。**

- **戻り値（**CY**フラグ）**

  ：データ読み出し正常終了（CY = 0**）または，データ読み出し異常終了（**CY = 1**）が戻ります。異常終了の場合は，データ番号が設定した範囲内にあるかを確認してください。また，指定したデータ番号のデータが一度も書き込まれていない場合も異常となります。**

main.c（C 言語版）の場合

・引数

：読み出したいデータのデータ番号をeeprom_data構造体のuc_data_noに設定後，eeprom_data構造体へのアドレスを引数として_eeprom_read関数を実行してください。

・戻り値（エラー・フラグ）

：データ読み出し正常終了（戻り値 = 0）または，データ読み出し異常終了（戻り値 = 1）が戻ります。異常終了の場合は，データ番号が設定した範囲内にあるかを確認してください。また，指定したデータ番号のデータが一度も書き込まれていない場合も異常となります。

(2) EEPROM書き込み処理（__eeprom_write）：設定されたサイズのデータをEEPROM領域に書き込みます。

main.asm（アセンブラ版）の場合

・引数

：書き込みたいデータのデータ番号をEEPROM_NOに，データをEEPROM_DATAに，デリミタをEEPROM_DELIMITERにそれぞれ設定後，EEPROM_NOのアドレスをAXレジスタに格納して__eeprom_ write関数を実行してください。

・戻り値（CYフラグ）

：データ書き込み正常終了（CY = 0）または，データ書き込み異常終了（CY = 1）が戻ります。異常終了の場合は，データ番号が設定した範囲内にあるかを確認してください。また，指定したデータ番号のデータが一度も書き込まれていない場合も異常となります。

main.c（C 言語版）の場合

・引数

：書き込みたいデータのデータ番号をeeprom_data構造体のuc_data_noに，データをuc_eeprom_dataに，デリミタをuc_delimiterにそれぞれ設定後，eeprom_data構造体へのアドレスを引数として_eeprom_write関数を実行してください。

・戻り値（エラー・フラグ）

：データ書き込み正常終了（戻り値 = 0）または，データ書き込み異常終了（戻り値 = 1）が戻ります。

5. 4　EEPROMエミュレーション・プログラム説明

5. 4. 1　EEPROMエミュレーション・ユーザ・アクセス処理

EEPROMエミュレーションにて読み出し／書き込みを行うために使用する，ユーザからのアクセス処理を表5－3，表5－4に示します。

表5－3　EEPROM読み出し処理

（a）アセンブラ版

処理名	__eeprom_read（ユーザ・アクセス関数）
ROM容量	31バイト
スタック容量	6レベル（12バイト）
入力	AX ：変数のアドレス
戻り値	正常終了：CY = 0 異常終了：CY = 1
動作説明	指定したデータ番号の最新データを格納アドレスに読み出す。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックから最新データのアドレスを検索する。 3：検索アドレスから最新データを読み出す。

（b）C言語版

処理名	_eeprom_read（ユーザ・アクセス関数）
ROM容量	31バイト
スタック容量	6レベル（12バイト）
入力	AX ：構造体のポインタ
戻り値	正常終了：エラー・フラグ = 0 異常終了：エラー・フラグ = 1
動作説明	指定したデータ番号の最新データを格納アドレスに読み出す。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックから最新データのアドレスを検索する。 3：検索アドレスから最新データを読み出す。

表5－4　EEPROM書き込み処理（1/2）

（a）アセンブラ版

処理名	__eeprom_write（ユーザ・アクセス関数）
ROM容量	63バイト
スタック容量	11レベル（22バイト）
入力	AX ：変数のアドレス
戻り値	正常終了：CY = 0 異常終了：CY = 1
動作説明	EEPROMに指定番号のデータを，格納アドレスから書き込む。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックがない場合，最初の指定ブロックを有効に設定する。 3：有効ブロックへの書き込み可能アドレスを検索する。 4：有効ブロックが満杯で書き込めない場合は，次のブロックへの移行操作を行う。 5：書き込みデータを作成する。 6：有効ブロックへ書き込む。

表5－4　EEPROM書き込み処理（2/2）

（b）C言語版

処理名	_eeprom_write（ユーザ・アクセス関数）
ROM容量	63バイト
スタック容量	11レベル（22バイト）
入力	AX ：構造体のポインタ
戻り値	正常終了：エラー・フラグ = 0 異常終了：エラー・フラグ = 1
動作説明	EEPROMに指定番号のデータを，格納アドレスから書き込む。 1：EEPROMとして使用しているブロックを検索する。 2：有効ブロックがない場合，最初の指定ブロックを有効に設定する。 3：有効ブロックへの書き込み可能アドレスを検索する。 4：有効ブロックが満杯で書き込めない場合は，次のブロックへの移行操作を行う。 5：書き込みデータを作成する。 6：有効ブロックへ書き込む。

5. 4. 2　EEPROM エミュレーション制御処理（内部処理用）

EEPROMエミュレーションにてエミュレーション制御を行うための処理を表5－5から表5－10に示します。

表5－5　EEPROMパラメータ取得処理

処理名	getpara
ROM容量	8バイト
スタック容量	1レベル（2バイト）
入力	AX ：構造体のポインタ
戻り値	A = データ番号，RQDATA_Noにもコピー HL = データ・アドレス
動作説明	ユーザからの関数呼び出しの引数（ポインタ）からその構造体の中身を読み出して必要なパラメータを得る。

表5－6　EEPROM使用中ブロック検索処理

処理名	EEPROMUseBlockSearch
ROM容量	27バイト
スタック容量	2レベル（4バイト）
入力	なし
出力	正常終了：CY = 0, A = ブロック・テーブル番号（01H - FEH） 異常終了：CY = 1, A = 最終ブロックの次
使用レジスタ	A
動作説明	EEPROMとして割り振られているフラッシュ・メモリの現在使用中のブロックを検索する。

表5－7　EEPROMブロックの初期化処理

処理名	EEPROMBlockInit
ROM容量	19**バイト**
スタック容量	6**レベル（**12**バイト）**
入力	**なし**
出力	**正常終了：**CY = 0 **異常終了：**CY = 1
使用レジスタ	A, X, B, C, D, E
動作説明	EEPROM**対象ブロックのいずれも有効ブロックでない場合，最初の指定ブロックを使用中（有効）に設定する。** **正常に確保できれば，**CY = 0**で戻る。** **正常に確保できなければ，**CY = 1**で戻る。**

表5－8　EEPROM使用ブロックの変更処理

処理名	EEPROMUseBlockChange
ROM容量	88**バイト**
スタック容量	6**レベル（**12**バイト）**
入力	A = **使用中ブロック番号**
出力	**正常終了：**CY = 0 **異常終了：**CY = 1
使用レジスタ	A, X, B, C, D, E
動作説明	**使用中ブロックのデータが満杯時に，次に使用するブロックの検索およびデータのコピーを行う。** 1**：次に使用するブロックの設定をする。** 2**：次に使用するブロックを消去する。** 3**：有効ブロックから，次のブロックに最新データを転送する。** 4**：次に使用するブロックを有効に設定する。** 5**：現在有効になっているブロックを無効に設定する。** 6**：新しいブロック番号「**CurrentB_No**」へ格納する。**

表5－9　EEPROM使用ブロックのデータ書き込み先頭検索処理

処理名	EEPROMWriteTopSearch
ROM容量	26**バイト**
スタック容量	3**レベル（**6**バイト）**
入力	A**：検索ブロック・テーブル番号**
出力	**検索成功：**CY = 0**，**AX**に書き込みアドレスをセットする** **検索失敗：**CY = 1
使用レジスタ	A, X
動作説明	**指定ブロックの書き込み可能なアドレスを検索する。** **データ領域が**0FFH**でブロック内に収まる場合のみ正常終了となる。**

表5－10　EEPROM最新データ検索処理

処理名	EEPROMDataSearch
ROM容量	41バイト
スタック容量	2レベル（4バイト）
入力	A：使用中ブロック・テーブル番号，X：検索データ番号5
出力	正常終了：CY = 0，DE = 最新データのアドレス 異常終了：CY = 1，E = 0
使用レジスタ	A , D, E
動作説明	最新データの格納アドレスを読み出す。

5. 4. 3　フラッシュ・メモリ制御処理

EEPROMエミュレーションにてフラッシュ・メモリ制御を行うための処理を表5－11から表5－18に示します。

表5－11　ブロック消去処理

処理名	SelfFlashBlockErase
ROM容量	29バイト
スタック容量	3レベル（6バイト）
入力	A：消去ブロック番号
出力	正常終了：CY = 0 異常終了：CY = 1
使用レジスタ	B
動作説明	指定ブロックの消去およびブランク・チェックを行う。

★

表5－12　セルフ・プログラミング・モードから通常モードへの遷移処理

処理名	SelfFlashModeOff
ROM容量	31バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	なし
使用レジスタ	A, X
動作説明	セルフ・プログラミング・モードを解除する。

表5－13　通常モードからセルフ・プログラミング・モードへの遷移処理

処理名	SelfFlashModeOn
ROM容量	35バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	なし
使用レジスタ	A, X
動作説明	セルフ・プログラミング・モードに設定する。

表5－14　ブロック消去処理

処理名	FlashBlockErase
ROM容量	15バイト
スタック容量	1レベル（2バイト）
入力	A：ブロック番号
出力	正常終了：ゼロ・フラグ（Z）　= 1 異常終了：ゼロ・フラグ（Z）　= 0
使用レジスタ	A
動作説明	指定ブロックを消去する。

表5－15　フラッシュ・セルフ・プログラミング機能呼び出し処理

処理名	SubFlashSelfPrg
ROM容量	12バイト
スタック容量	1レベル（2バイト）
入力	なし
出力	正常終了：ゼロ・フラグ（Z）　= 1 異常終了：ゼロ・フラグ（Z）　= 0
使用レジスタ	A
動作説明	フラッシュ・セルフ・プログラミング機能呼び出し

表5－16　ブロック・ブランク・チェック処理

処理名	FlashBlockBlankCheck
ROM容量	17バイト
スタック容量	1レベル（2バイト）
入力	A：対象ブロック番号
出力	正常終了：ゼロ・フラグ（Z）　= 1 異常終了：ゼロ・フラグ（Z）　= 0
使用レジスタ	A
動作説明	指定ブロックのブランク・チェックを行う。

表5－17　ブロックの有効設定処理

処理名	SetValid
ROM容量	9バイト
スタック容量	1レベル（2バイト）
入力	A：ブロック番号
出力	正常終了：ゼロ・フラグ（Z） = 1 異常終了：ゼロ・フラグ（Z） = 0
使用レジスタ	A, X, C, D, E
動作説明	使用ブロックを有効に設定する。

表5－18　EEPROMデータ書き込み処理

処理名	FlashEEPROMWrite
ROM容量	56バイト
スタック容量	5レベル（10バイト）
入力	DE：書き込み先頭アドレス C ：書き込みデータ数 AX：書き込みデータ格納アドレス
出力	正常終了：ゼロ・フラグ（Z） = 1 異常終了：ゼロ・フラグ（Z） = 0
使用レジスタ	D, E
動作説明	EEPROMでの書き込みおよび内部ベリファイを行う。

★

5. 5　EEPROMエミュレーション・プログラムのフロー・チャート

5. 5. 1　EEPROM エミュレーション・アクセス処理のフロー・チャート

EEPROM エミュレーションにて，読み出し／書き込みを行うための，ユーザからのアクセス処理のフロー・チャートを図 5－1，図 5－2 に示します。

図 5－1　EEPROM 読み出し処理フロー・チャート

[概　　要]

構造体で定義されたデータを指定された格納アドレスに読み出す

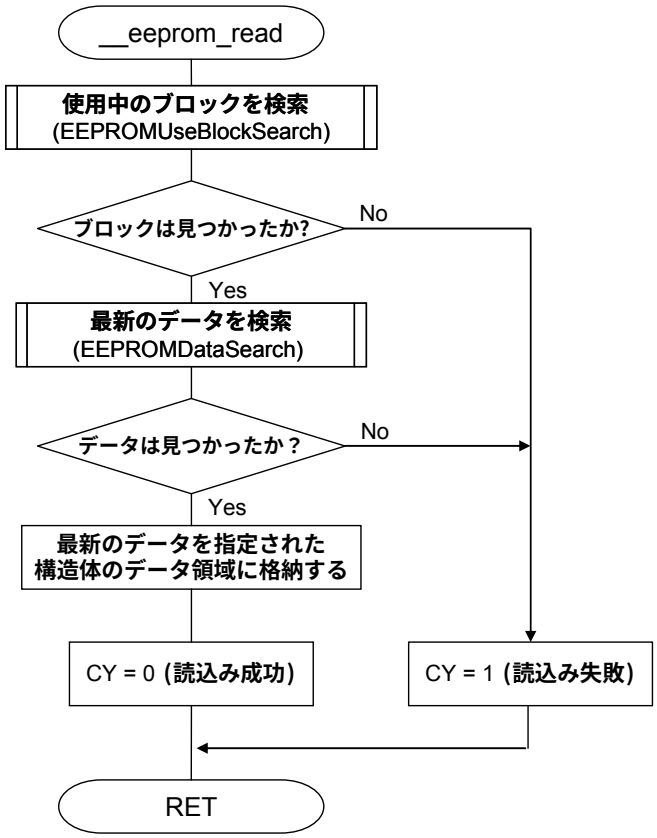
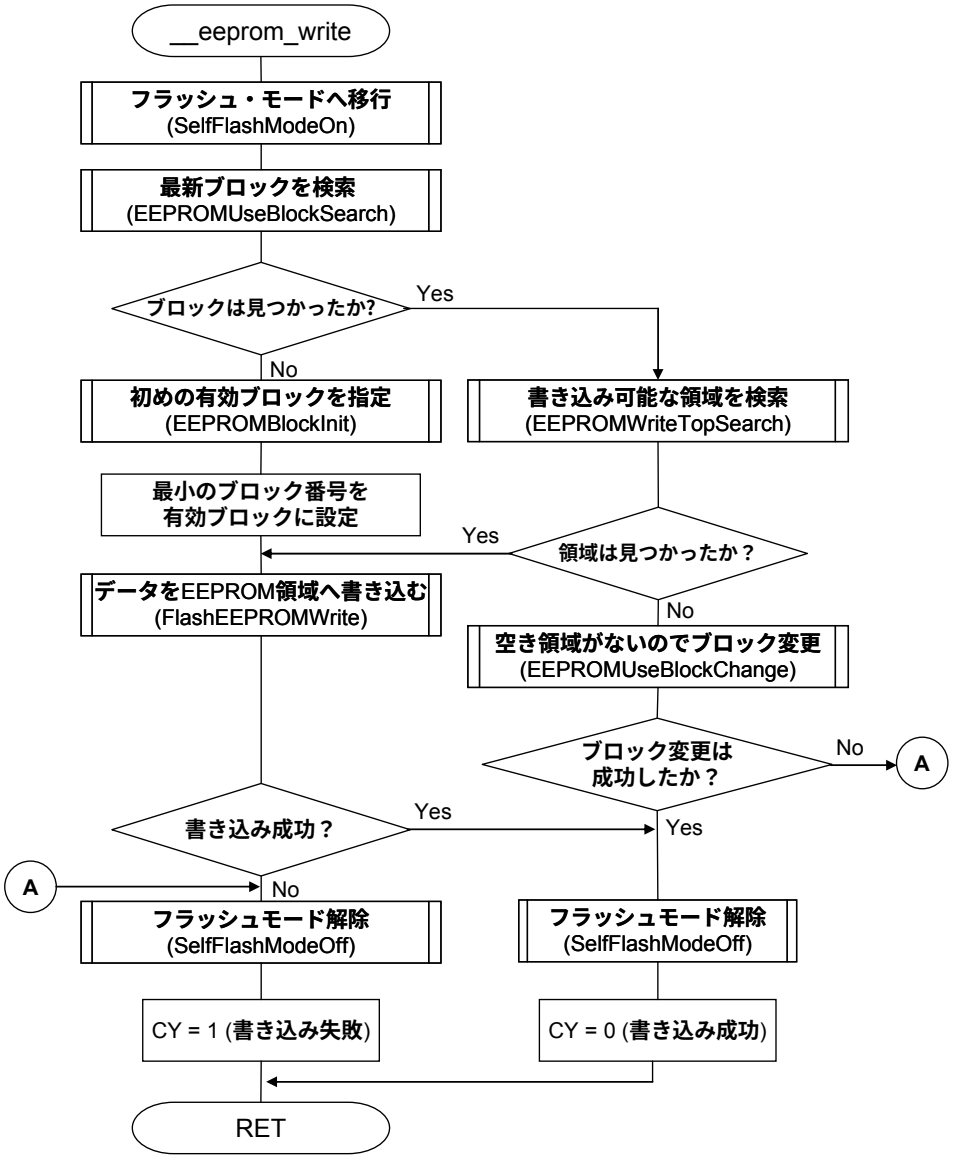


図 5－2　EEPROM 書き込み処理フロー・チャート

［概　　要］

有効ブロックに指定番号のデータを格納アドレスから書き込む



5. 5. 2　EEPROM エミュレーション制御処理のフロー・チャート

EEPROM エミュレーションにて，エミュレーション制御を行うための制御処理のフロー・チャートを図 5－3～図 5－7 に示します。

図 5－3　EEPROM 使用中ブロック検索処理フロー・チャート

［概　　要］

EEPROM として割り振られているフラッシュ・メモリの現在使用中のブロックを検索する

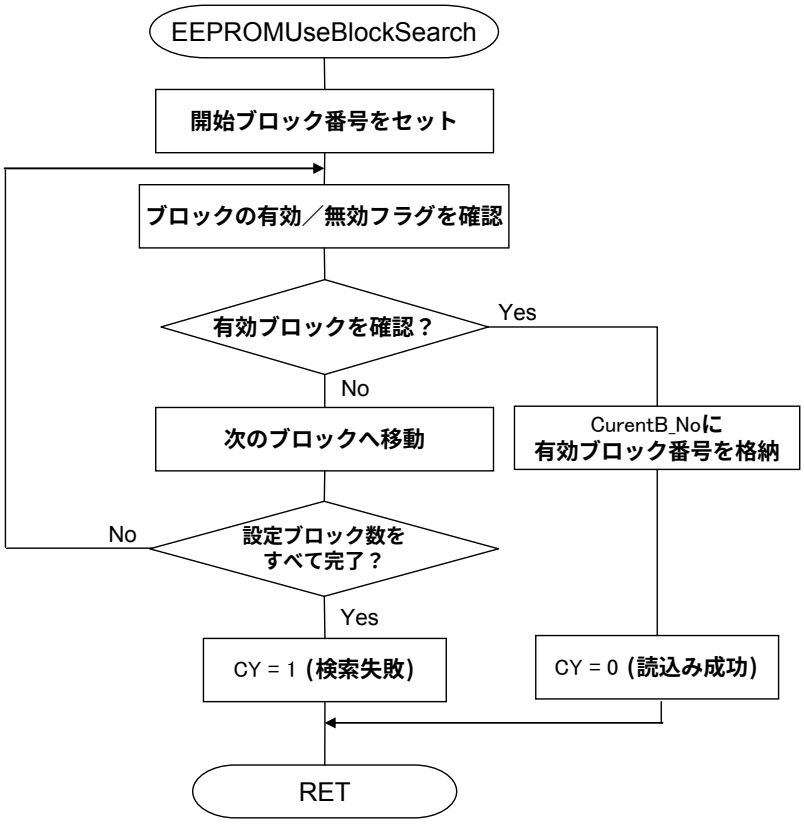


図 5－4　EEPROM ブロックの初期化処理フロー・チャート

[概　　要]

EEPROM 対象ブロックのいずれも有効ブロックでないときに，最初の指定ブロックを有効に設定する。

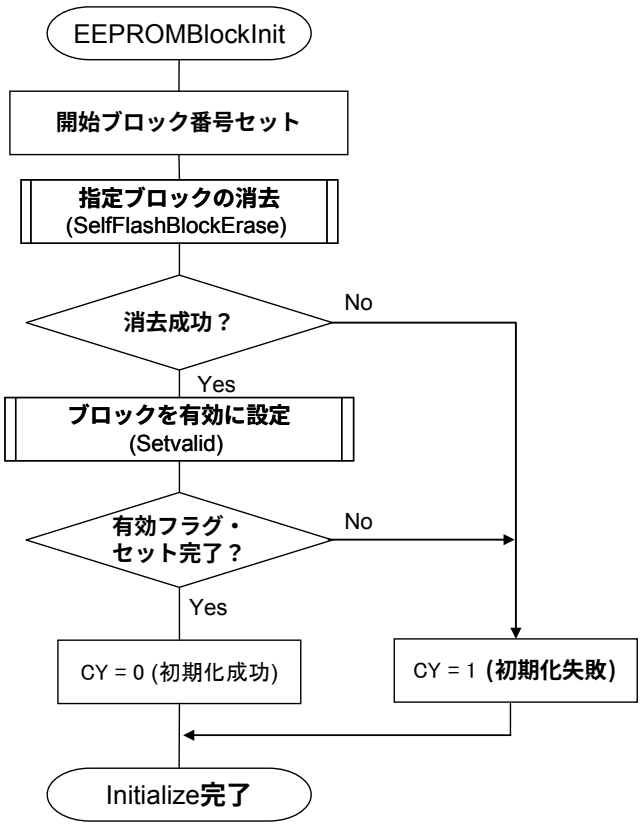


図 5ー5　EEPROM **使用ブロックの変更処理フロー・チャート**（1/2）

[概　　要]

使用中ブロックのデータが満杯時，次に使用するブロックの検索および新ブロックへのデータ・コピーを行う。

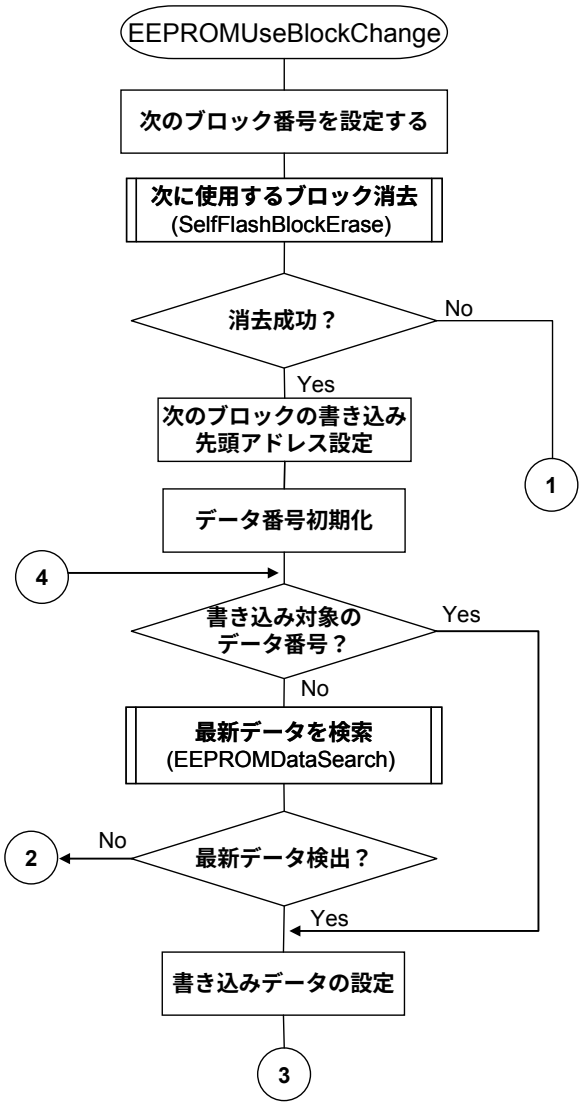


図 5－5　EEPROM 使用ブロックの変更処理フロー・チャート（2/2）

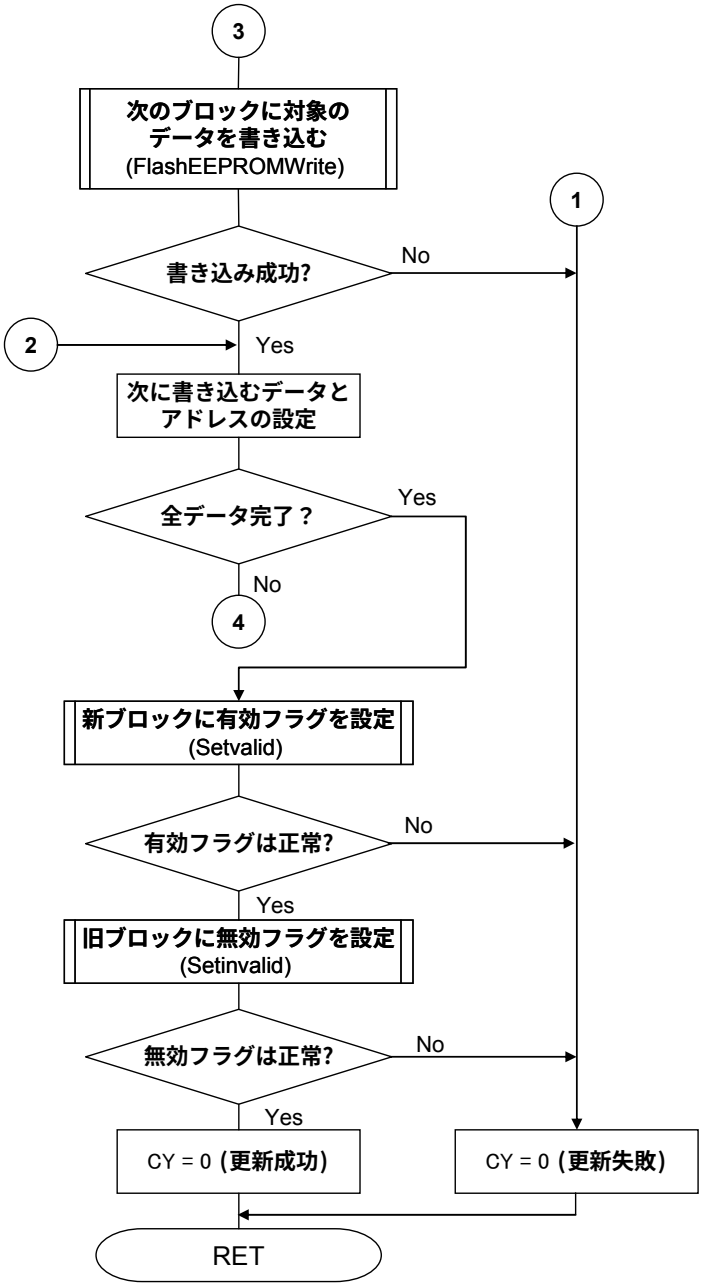


図 5－6　EEPROM **使用ブロックのデータ書き込み先頭検索処理フロー・チャート**

［概　　要］EEPROMWriteTopSearch

指定ブロックの書込み可能な領域を探す。

データ領域が 0xFF **で，ブロック内に次データが収まる場合のみ正常終了となる。**

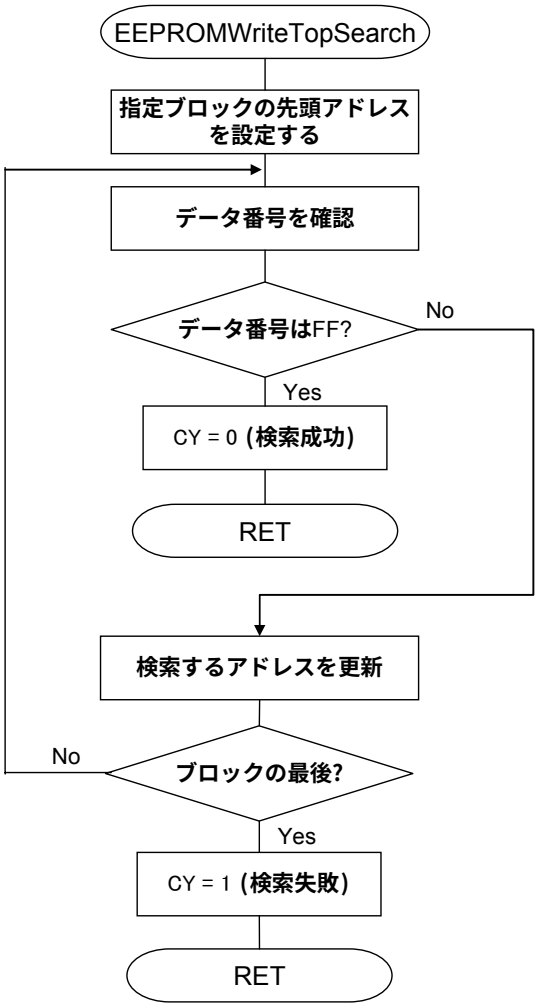
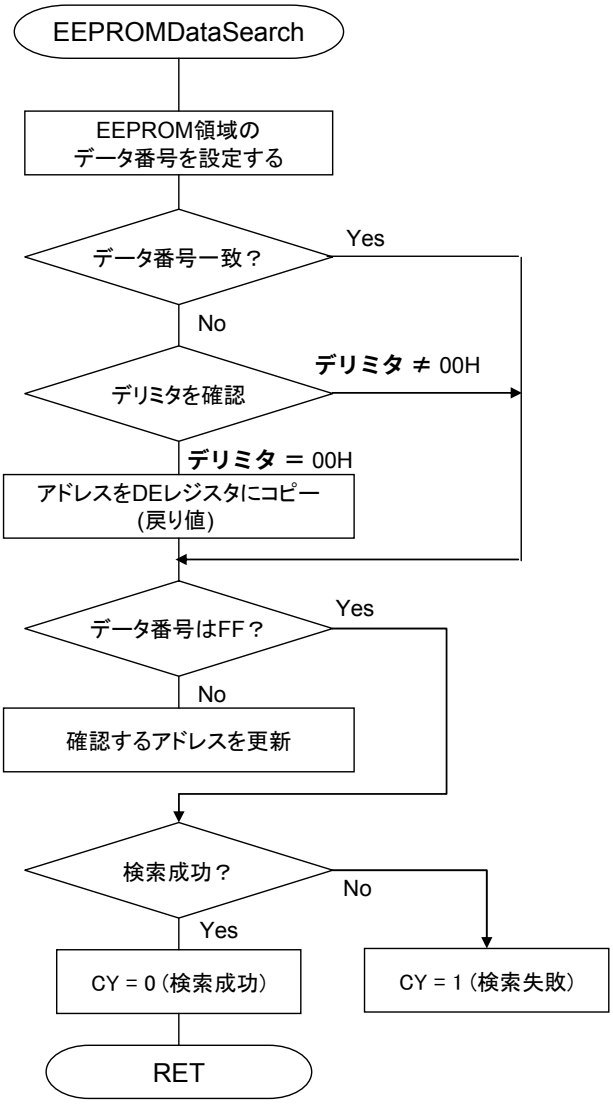


図 5－7　EEPROM 最新データ検索処理フロー・チャート

［概　　要］

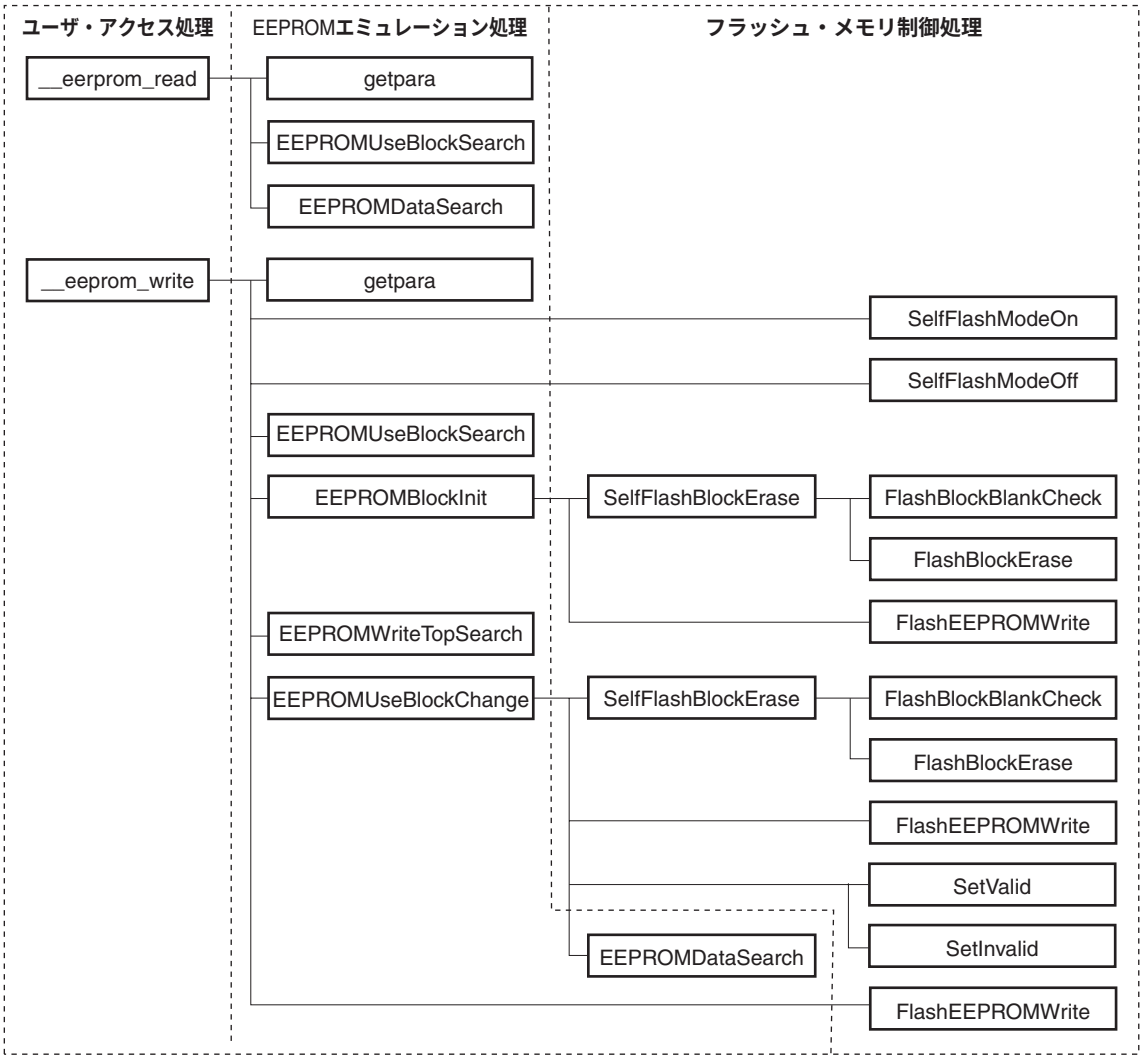
最新データの格納アドレスを読み出す。



5. 6　EEPROM エミュレーションの処理一覧

EEPROM エミュレーション処理のコール・ツリーを次に示します。

図 5－8　**コール・ツリー**



付録A　改版履歴

A. 1　本版で改訂された主な箇所

箇　所	内　容
旧版p.65	旧版の付録A　サンプル・プログラム・リスト（固定長単一データ方式）を削除
旧版p.85	旧版の付録B　サンプル・プログラム・リスト（固定長複数データ方式）を削除
第2章　EEPROMエミュレーション機能（固定長単一データ方式）	
pp.10, 11	2. 1　EEPROMエミュレーションの基本仕様の1ブロックの消去回数と最大書き換え回数を変更
p.18	表2－1　EEPROMエミュレーションの動作時の条件に注意2を追加，備考を変更
p.18	2. 3　サンプル・プログラムの入手方法を追加
第3章　EEPROMエミュレーション・プログラム（固定長単一データ方式，アセンブリ言語）	
p.19	表3－2　リソースを変更
p.21	3. 3. 1（3）消去リトライ回数を変更
pp.26, 28	表3－10　ブロック消去処理と表3－17　EEPROMデータ書き込み処理を変更
第4章　EEPROMエミュレーション機能（固定長複数データ方式）	
p.37, 38	4. 1　EEPROMエミュレーションの基本仕様の1ブロックの消去回数と最大書き換え回数を変更
p.45	表4－1　EEPROMエミュレーションの動作時の条件に注意2を追加，備考を変更
p.45	4. 3　サンプル・プログラムの入手方法を追加
第5章　EEPROMエミュレーション・プログラム（固定長複数データ方式，アセンブリ言語）	
p.46	表5－2　リソースを変更
p.48	5. 3. 1（3）消去リトライ回数を変更
pp.53, 55	表5－11　ブロック消去処理と表5－18　EEPROMデータ書き込み処理を変更
付録A　改版履歴	
p.65	A. 2　前版までの改版履歴を追加

★　A. 2　前版までの改版履歴

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版　数	内　容	適用箇所
第2版	章を全面変更	第2章　EEPROMエミュレーション機能（固定長単一データ方式）
		第3章　EEPROMエミュレーション・プログラム（固定長単一データ方式，アセンブリ言語）
	章を追加	第4章　EEPROMエミュレーション機能（固定長複数データ方式）
		第5章　EEPROMエミュレーション・プログラム（固定長複数データ方式，アセンブリ言語）
	章を全面変更	付録A　サンプル・プログラム・リスト（固定長単一データ方式）
	章を追加	付録B　サンプル・プログラム・リスト（固定長複数データ方式）
		付録C　改版履歴

【発行】NEC**エレクトロニクス株式会社（**http://www.necel.co.jp/**）**
【問い合わせ先】http://www.necel.com/contact/ja/