

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ひ申し上げます。

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のですが、誤りが無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、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と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。

お客様各位

資料中の「三菱電機」、「三菱XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って株式会社日立製作所及び三菱電機株式会社のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ(フラッシュメモリ・SRAM等)を含む半導体事業は株式会社ルネサス テクノロジに承継されました。

従いまして、本資料中には「三菱電機」、「三菱電機株式会社」、「三菱半導体」、「三菱XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

注:「高周波・光素子事業、パワーデバイス事業については三菱電機にて引き続き事業運営を行います。」

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

M16C/80 グループ

ブートローダの説明

1.0 要約

この資料は M16C/80 グループ ブートローダの通信プロトコル仕様およびユーザで使用する書き換えプログラムについて説明しています。

2.0 はじめに

この資料で説明する応用例は次のマイコン、条件での利用に適用されます。

- ・マイコン：M16C/80 グループ

3.0 応用例の説明

3.1 ブートローダ概要

ブートローダ内蔵 M16C/80 グループ ROM 外付け版(以下ブートローダ内蔵 M16C/80 と称す)は、外付けフラッシュメモリを書き換えるためのブートプログラム(以下書き換えプログラムと称す)をマイコンにダウンロードするファームウェア(以下ブートローダと称す)を内蔵しています。表 3.1.1 にブートローダ内蔵 M16C/80 製品一覧を示します。

ユーザは、シリアルライターまたはパソコンを用いて、ブートローダ内蔵 M16C/80 とシリアル通信を行うことで、書き換えプログラムをマイコンの内部 RAM にダウンロードし、それを実行することができます。また、ブートローダは、ダウンロード機能の他にオプション機能として特定の外付けフラッシュメモリ(*1)に対して書き換えできるフラッシュメモリ制御機能を持っています。

ダウンロード、フラッシュメモリ制御の各機能については、「[3.2 ブートローダモード1\(クロック同期形\)機能概要](#)」および「[3.3 ブートローダモード2\(クロック非同期形\)機能概要](#)」で説明します。

*1 三菱製フラッシュメモリ M5M29GB/T160BVP、M5M29GB/T320BVP または同フラッシュメモリを混載した MCM に限る。

3.1.1 ブートローダモード

ブートローダ内蔵 M16C/80 は、CNVss 端子に"H"を印加してリセットを解除した場合、CPU はマイクロプロセッサモードで動作します。CNVss 端子に"L"を印加してリセットを解除した場合には、ブートローダプログラムで動作を開始します。このモードをブートローダモードと呼びます。

表 3.1.1.ブートローダ内蔵 M16C/80 製品一覧表

2001年10月現在

型名	ROM 容量	RAM 容量	パッケージ	備考
M30800SFP-BL		10 K バイト	100P6S-A	ブートローダ内蔵 ROM 外付け版
M30800SGP-BL			100P6Q-A	
M30802SGP-BL			144P6Q-A	
M30803SFP-BL	24 K バイト		100P6S-A	
M30803SGP-BL			100P6Q-A	
M30805SGP-BL			144P6Q-A	

3.1.2 ブートローダモード機能概要

ブートローダモードには、

- ・ブートローダモード1(クロック同期形)
- ・ブートローダモード2(クロック非同期形)

があり、シリアルライタ(*1)を用いて外部との通信を行います。

ブートローダモードは、CNVss に"L"を印加してリセットを解除することで起動します。シリアルデータの入出力は、UART1 を使い 8 ビット単位で転送します。リセット解除時の SCLK 端子の状態によってブートローダモード1(クロック同期形)とブートローダモード2(クロック非同期形)を切り替えます。

ブートローダモード1(クロック同期形)を使用する場合は、SCLK 端子に"H"を印加してリセットを解除してください。UART1 の端子 CLK1、RxD1、TxD1、RTS1 の4本を使用します。CLK1 端子は転送クロックの入力端子 SCLK となり、外部から転送クロックを入力します。TxD1 端子は、TxD となります。この端子は、CMOS 出力です。RTS1 端子は BUSY 出力となり、受信準備が完了すれば"L"を出力し、受信動作を開始すると"H"を出力します。

ブートローダモード2(クロック非同期形)を使用する場合は、SCLK 端子に"L"を印加してリセットを解除してください。UART1 の端子 RxD1、TxD1 の2本を RxD、TxD として使用します。

リセット解除時の BUSY 端子の状態によって内蔵のプルアップ機能を有効にするか無効にするかを切り替えます。リセット解除直後、BUSY 端子に"L"が印加されていると内蔵のプルアップ機能は無効に、"H"が印加されていると有効になります。表 3.1.2 に端子の機能説明を、また図 3.1.1 ~ 図 3.1.3 にブートローダモード時の端子結線図を示します。

- *1 ブートローダモード1(クロック同期形)は、PC カード型フラッシュメモリプログラム(M3A-0655G01/G02)および株式会社サニー技研製シリアルライタ Multi Flash Write で使用できます。また、ブートローダモード2(クロック非同期形)は、M16C Flash Starter(M3A-0806)で使用できます。
なお、通常シリアルライタは書き換えプログラムとともにユーザで開発する必要があります。

表 3.1.2.端子の機能説明

端子名	名称	入出力	機能
V _{CC}	電源入力		V _{CC} 端子には 4.2 (2.7) V ~ 5.5V (*) を印加してください。
V _{SS}			V _{SS} 端子には 0V を印加してください。
CNV _{SS}	モード選択 CNV _{SS}	入力	V _{SS} に接続してください。
RESET	リセット入力	入力	リセット入力端子です。 リセットが “L” の間、X _{IN} 端子には 20 サイクル以上のクロックが必要です。
X _{IN}	クロック入力	入力	X _{IN} 端子と X _{OUT} 端子の間にはセラミック共振子、または水晶振動子を接続してください。
X _{OUT}	クロック出力	出力	外部で生成したクロックを入力するときは、X _{IN} 端子から入力し、X _{OUT} 端子は開放してください。
BYTE	BYTE 入力	入力	V _{SS} または V _{CC} に接続してください。
AV _{CC}	アナログ電源入力	入力	AV _{CC} は V _{CC} に接続してください。
AV _{SS}			AV _{SS} は V _{SS} に接続してください。
V _{REF}	基準電源入力	入力	AD 変換器の基準電圧入力端子です。
P0 ₀ ~ P0 ₇	入力ポート P0	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P1 ₀ ~ P1 ₇	入力ポート P1	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P2 ₀ ~ P2 ₇	入力ポート P2	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P3 ₀ ~ P3 ₇	入力ポート P3	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P4 ₀ ~ P4 ₇	入力ポート P4	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P5 ₀ ~ P5 ₂	入力ポート P5	入出力	メモリとの接続、“H” を入力、“L” を入力、または開放してください。
P5 ₃ ~ P5 ₄	入力ポート P5	入力	“H” を入力、“L” を入力、または開放してください。
P5 ₅	HOLD 入力	入力	“H” を入力してください。
P5 ₆	入力ポート P5	入力	“H” を入力、“L” を入力、または開放してください。
P5 ₇	RDY 入力	入力	“H” を入力してください。
P6 ₀ ~ P6 ₃	入力ポート P6	入力	“H” を入力、“L” を入力、または開放してください。
P6 ₄ /RTS1	BUSY 出力 (*2)	出力	ブートローダモード 1 : BUSY 信号の出力端子です。 ブートローダモード 2 : プログラム動作チェック用モニタ。
P6 ₅ /CLK1	SCLK 入力	入力	ブートローダモード 1 : シリアルクロックの入力端子です。 ブートローダモード 2 : “L” を入力してください。
P6 ₆ /RxD1	データ入力 RxD	入力	シリアルデータの入力端子です。
P6 ₇ /TxD1	データ出力 TxD	出力	シリアルデータの出力端子です。
P7 ₀ ~ P7 ₇	入力ポート P7	入力	“H” を入力、“L” を入力、または開放してください。
P8 ₀ ~ P8 ₄ P8 ₆ , P8 ₇	入力ポート P8	入力	“H” を入力、“L” を入力、または開放してください。
P8 ₅	NMI 入力	入力	V _{CC} に接続してください。
P9 ₀ ~ P9 ₇	入力ポート P9	入力	“H” を入力、“L” を入力、または開放してください。
P10 ₀ ~ P10 ₇	入力ポート P10	入力	“H” を入力、“L” を入力、または開放してください。
P11 ₀ ~ P11 ₄	入力ポート P11	入力	“H” を入力、“L” を入力、または開放してください。
P12 ₀ ~ P12 ₇	入力ポート P12	入力	“H” を入力、“L” を入力、または開放してください。
P13 ₀ ~ P13 ₇	入力ポート P13	入力	“H” を入力、“L” を入力、または開放してください。
P14 ₀ ~ P14 ₆	入力ポート P14	入力	“H” を入力、“L” を入力、または開放してください。
P15 ₀ ~ P15 ₇	入力ポート P15	入力	“H” を入力、“L” を入力、または開放してください。

*1 4.2V 以下で使用する場合の最高動作周波数は、10MHz になります。

*2 詳細は、「[3.1.3 BUSY 端子機能について](#)」を参照してください。

・ 網掛け部分は、ブートローダモードで使用する端子です。

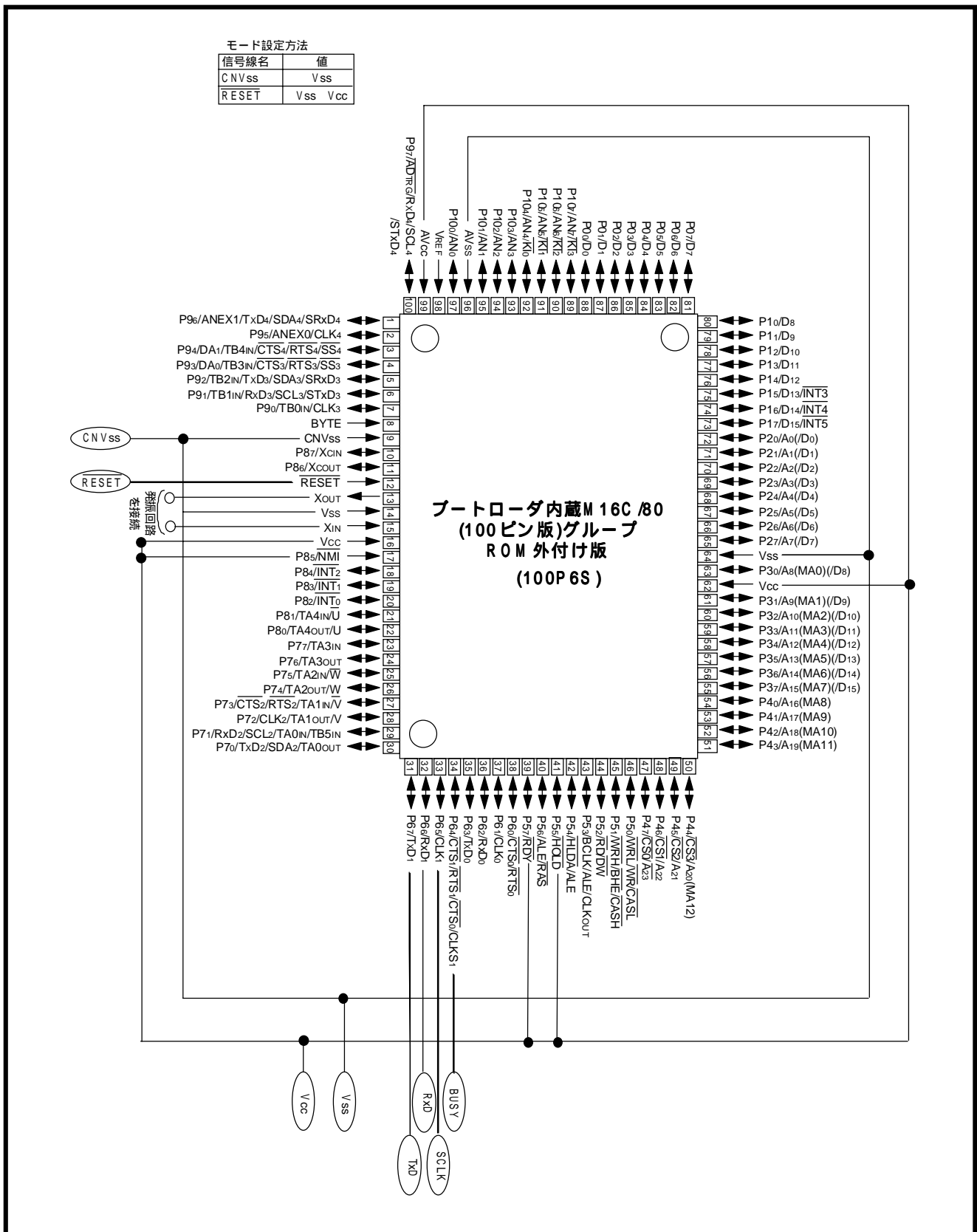


図 3.1.1.ブートローダモード時の端子結線図(100P6S)

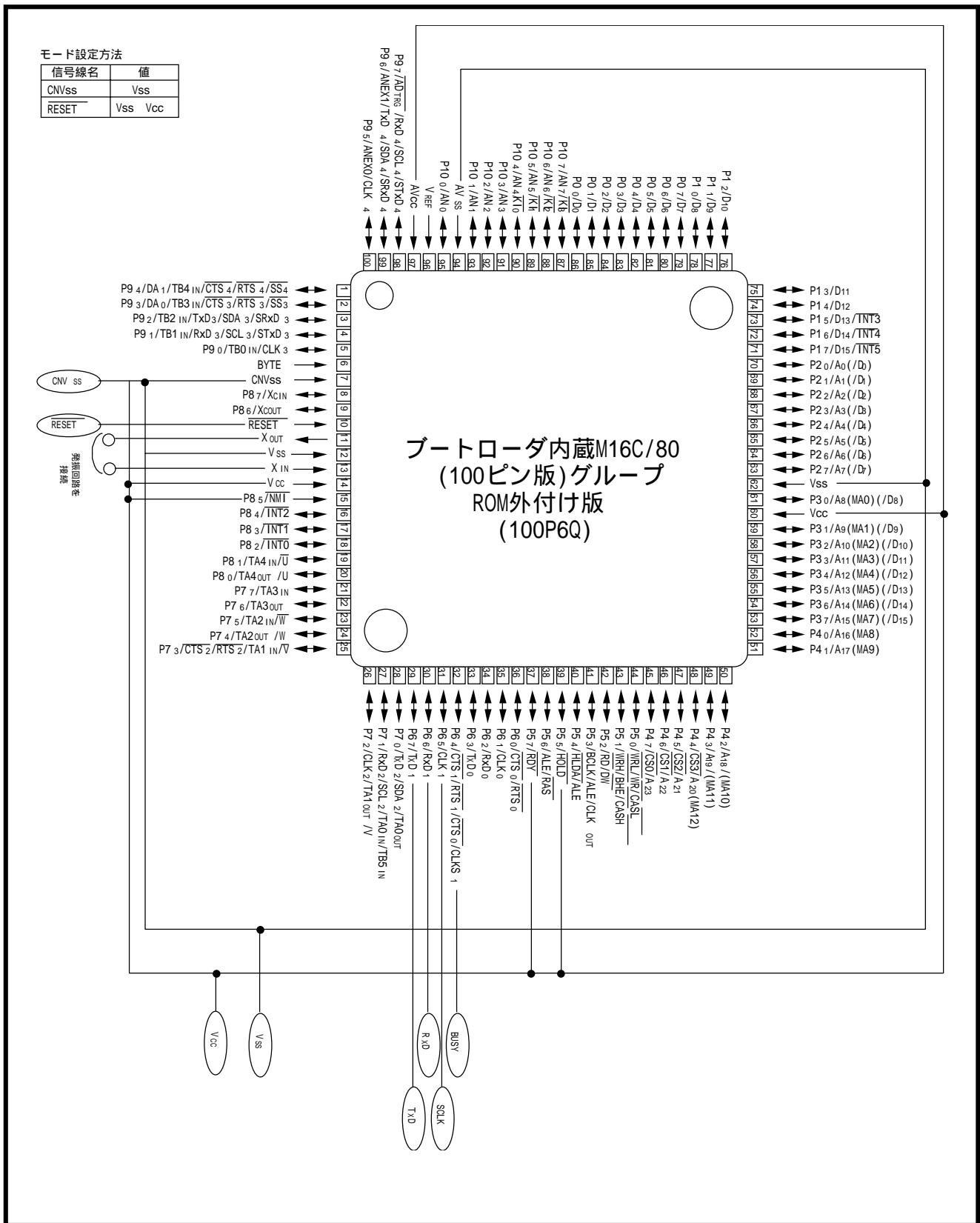


図 3.1.2.ブートローダモード時の端子結線図(100P6Q)

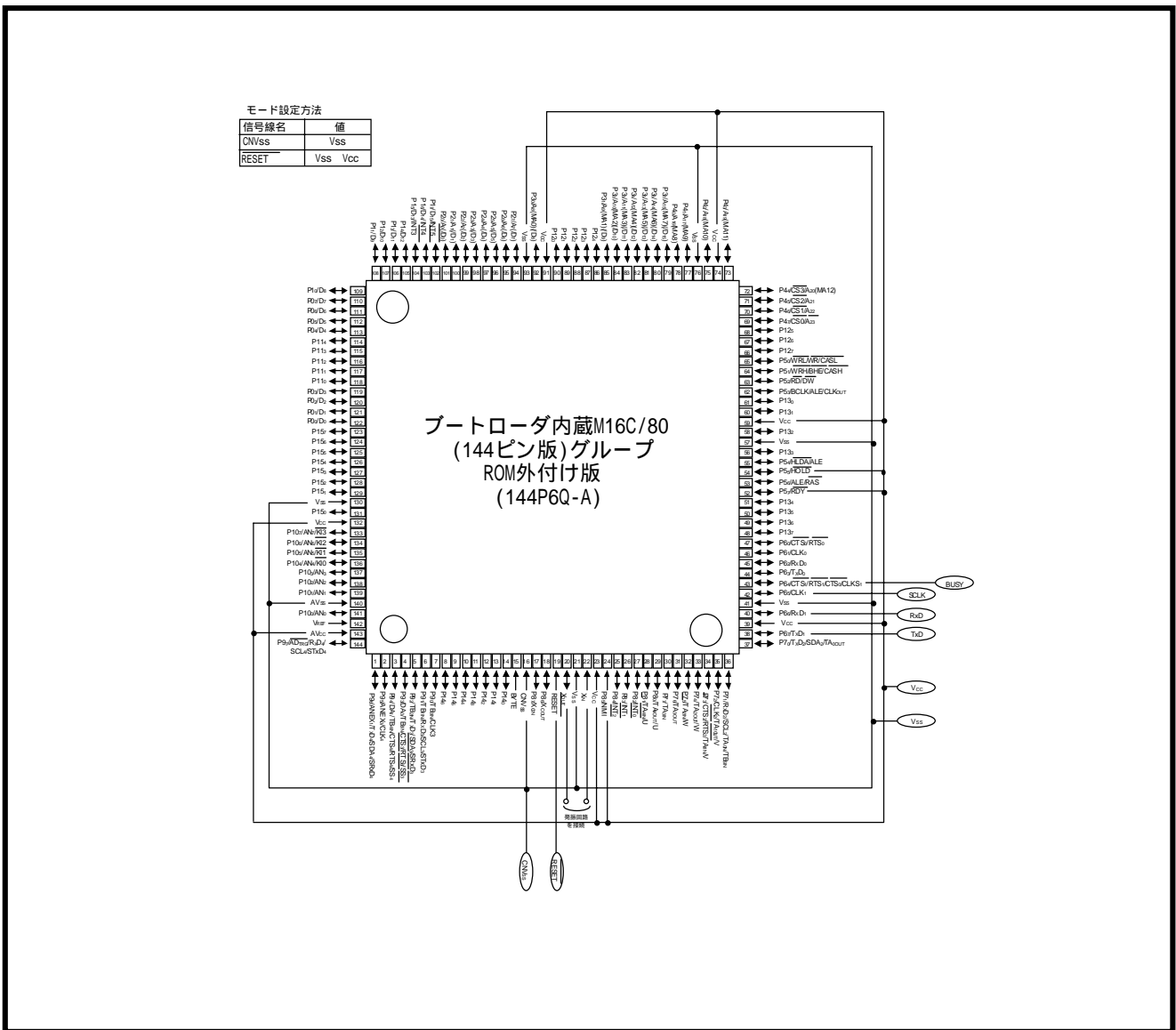


図 3.1.3.ブートローダモード時の端子結線図(144P6Q)

3.1.3 BUSY 端子機能について

BUSY(P6_4/RTS1)端子はリセット解除直後、入力として動作します。ブートローダは、このときの端子の状態により各ポート端子の内蔵プルアップ機能を有効にするか無効にするかを選択します。(ユーザーターゲットボード上では、BUSY 端子をプルダウンまたはプルアップすることで、内蔵プルアップ機能を無効にするか有効にするか選択してください。)リセット解除直後、BUSY 端子に"L"が印加されていると内蔵プルアップ機能が無効となり、"H"が印加されていると有効になります。選択後、BUSY 端子は出力となります。表 3.1.3 に内蔵プルアップ機能有効時のプルアップ端子の一覧を示します。

表 3.1.3.内蔵プルアップ機能有効時のプルアップ端子の一覧

プルアップ端子	プルアップ制御レジスタの設定
P0 ~ P3 (注 1)	PUR0 = 0FF ₁₆
P4、P5 (注 1)	PUR1 = 0F ₁₆
P6 ~ P9 (ただし、P8_5 を除く)	PUR2 = 0FF ₁₆
P10 ~ P13	PUR3 = 0FF ₁₆
P14、P15	PUR4 = 0F ₁₆

注 1: マイクロプロセッサモードに変更する前に、バスとして機能する P0 ~ P5 のプルアップ制御レジスタの値を"0"(内蔵プルアップ機能無効)にしてください。

3.2 ブートローダモード 1 (クロック同期形) 機能概要

ブートローダモード 1 では、4 線式クロック同期形のシリアル I/O (UART1) を用いてシリアルライタ(*1) との間でソフトウェアコマンド、アドレス、データ等の入出力を行います。SCLK 端子に"H"を印加してリセットを解除するとブートローダモード 1 になります。

受信時、SCLK 端子に入力する転送クロックの立ち上がりに同期して RxD 端子からソフトウェアコマンド、アドレスおよびプログラムデータが内部に取り込まれます。送信時には、転送クロックの立ち下がりに同期して、TxD 端子からリードデータおよびステータスが外部に出力されます。

TxD 端子は、CMOS 出力です。転送は 8 ビット単位、LSB ファーストで行います。送信、受信およびソフトウェアコマンド実行中等のビジー期間中には、BUSY 端子が"H"となります。したがって、次の転送は、BUSY 端子が"L"となった後に開始してください。図 3.2.1 に入出力タイミングを示します。

ブートローダモード 1 では、ダウンロード機能とフラッシュメモリ制御機能をサポートしています。

以下、これらの機能について説明します。

***1 シリアルライタとして、MAEC 製 PC カード型フラッシュメモリプログラマおよび株式会社サニー技研製シリアルライタ Muluti Flash Write を使用できます。**

マイコンから見た入出力タイミング

- (1) ターゲットボード電源ON
- (2) フラッシュ書き込み制御開始
- (3) リセット解除によりモードエントリ
- (4) フラッシュメモリへの消去・書込・読出
- (5) フラッシュ書き込み制御終了

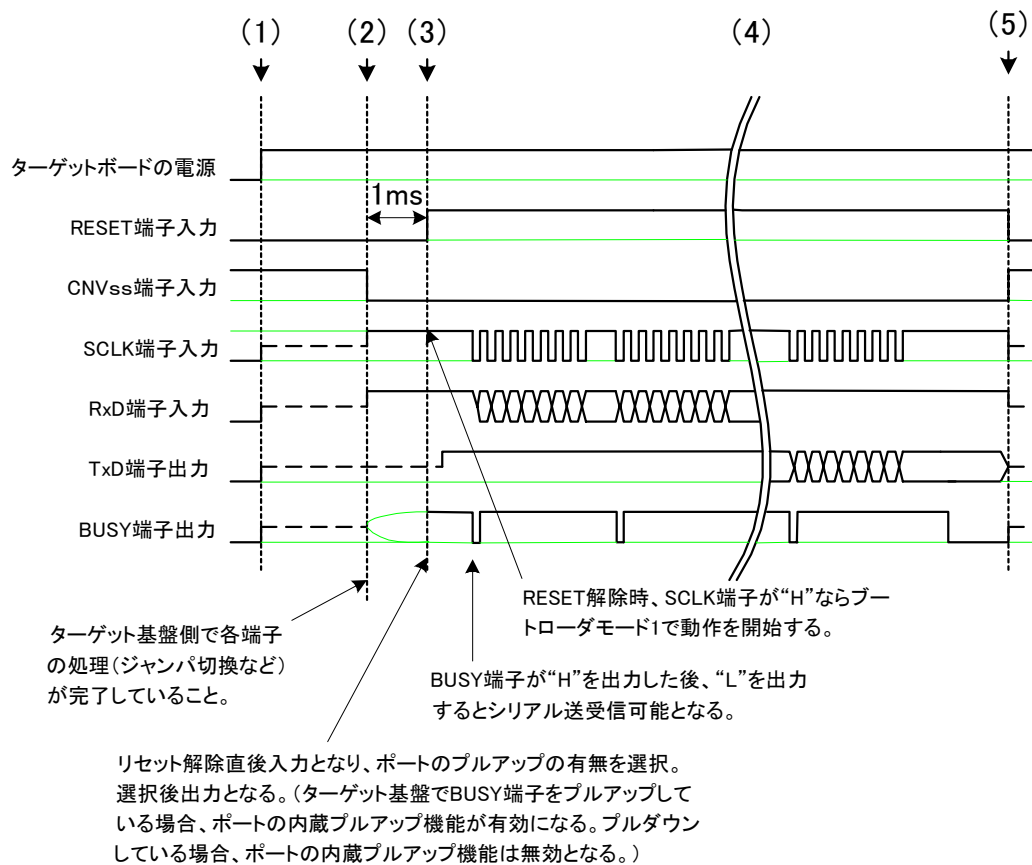


図 3.2.1.入力タイミング図

3.2.1 ダウンロード機能

機能概要

ブートローダのダウンロード機能は、シリアル通信を使って、書き換えプログラム(*1)をマイコンの内部RAMにダウンロードし、ダウンロードしたRAM上の番地へ飛びます。

- *1 書き換えプログラムは、ユーザで作成してください。
書き換えプログラムには、外付けフラッシュメモリに対する制御機能(書き込み、消去、読み出しなど)および、シリアルライタとの通信機能を有している必要があります。
書き換えプログラムでスタックを使用する場合、そのプログラム内でスタックポインタを設定してください。
ダウンロード完了後、マイコンはシングルチップモードで動作します。外付けフラッシュメモリへの書き込みや消去などの制御を開始する前に、書き換えプログラムでプロセッサモードをシングルチップモードからマイクロプロセッサモードに変更してください。
書き換えプログラムでは、割り込みは使用しないでください。
ダウンロードエリアについては、「[3.6 メモリマップ](#)」をご参照ください。

ソフトウェアコマンド

表 3.2.1. にブートローダモード 1 用ソフトウェアコマンドを示します。

表 3.2.1. ブートローダモード 1 用ソフトウェアコマンド

	制御コマンド名	1バイト目の転送	2バイト目	3バイト目	4バイト目	5バイト目	6バイト目	~
1	ダウンロード	FA ₁₆	サイズ(下位)	サイズ(上位)	チェックサム	データ入力	~ 必要回数	
2	ダウンロード結果出力	FA ₁₆	データ出力					
3	バージョン情報出力	FB ₁₆	バージョンデータ出力	バージョンデータ出力	バージョンデータ出力	バージョンデータ出力	バージョンデータ出力	~9バイト目 バージョンデータ出力

- ・ 網掛けは、マイコン シリアルライタへの転送
それ以外は、シリアルライタ マイコンへの転送

ダウンロード

内部 RAM に書き換えプログラムをダウンロードするコマンドです。ダウンロードされたプログラムは内部 RAM の 600₁₆ 番地以降へ格納されます。

ダウンロード後にリセットしても、内部 RAM に転送されたプログラムは保持されます。

ダウンロードは、以下の手順で実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "FA₁₆" を転送します。
- (2) 2 バイト目のシリアル転送でプログラムサイズの下位、3 バイト目の転送でプログラムサイズの上位を転送します。
- (3) 4 バイト目のシリアル転送でチェックサムを転送します。チェックサムは、5 バイト目以降に転送するデータを全て加算したものです。
- (4) 5 バイト目以降から書き換えプログラムを転送します。転送可能なプログラムの容量は、内部の RAM の容量によって異なります。(「[3.6 メモリマップ](#)」)

全データの転送が完了すると、マイコンは自動的にダウンロード結果出力コマンドを実行します。

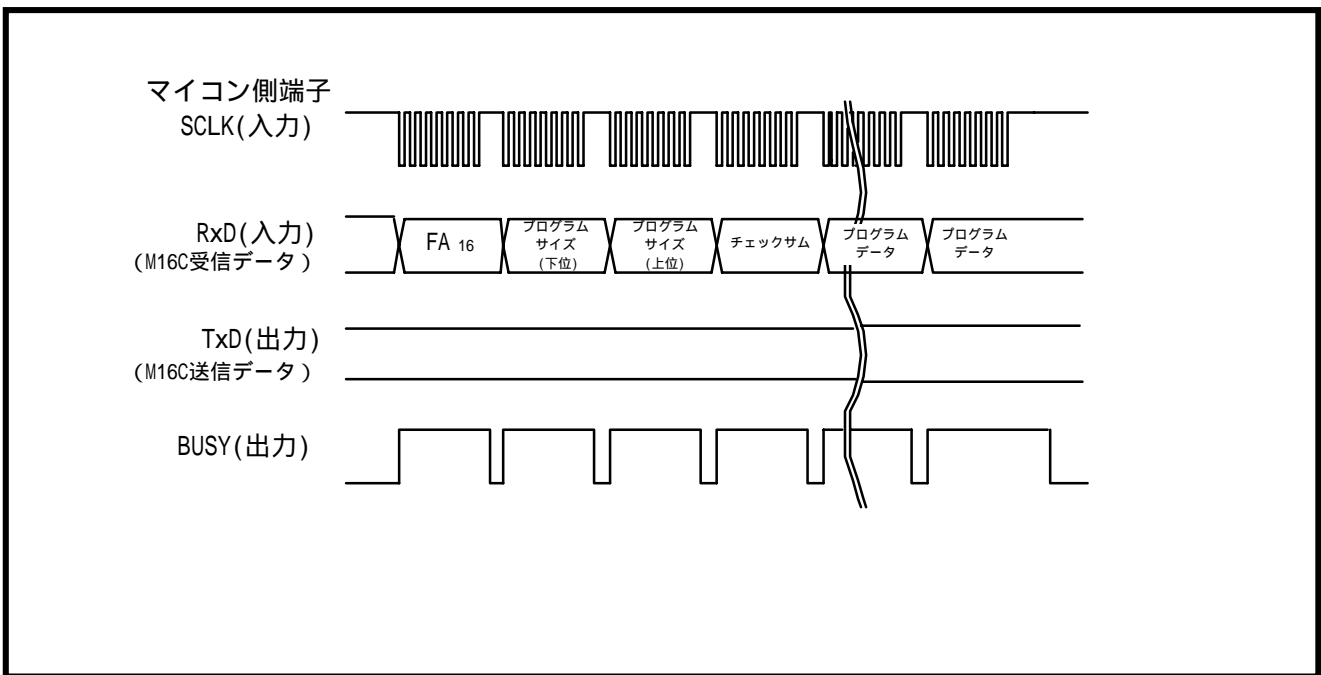


図 3.2.2.ダウンロードコマンド時のタイミング

ダウンロード結果出力

ダウンロードコマンド実行後、シリアルライターから転送されたチェックサム値とマイコンが受信したデータから求めたチェックサム値を比較します。チェックサム値が一致すれば、マイコンは"FA16"、"0016"（成功）を返送してから内部 RAM にダウンロードされたプログラムの先頭にジャンプし実行します。チェックサム値が不一致の場合、マイコンは"FA16"、"0116"（失敗）を返送してからマイコンに格納されているブートプログラムを改めて内部 RAM に転送し、これを実行します。（リセット後の元の状態に戻す。）

ダウンロード実行後、実行結果をブートローダ（マイコン）が以下の手順で出力します。

- (1)ダウンロード実行後、1 バイト目のシリアル転送で"FA16"を出力します。
- (2)2 バイト目のシリアル転送で、ダウンロードの実行結果により"0016"(成功)または"0116"(失敗)を出力します。

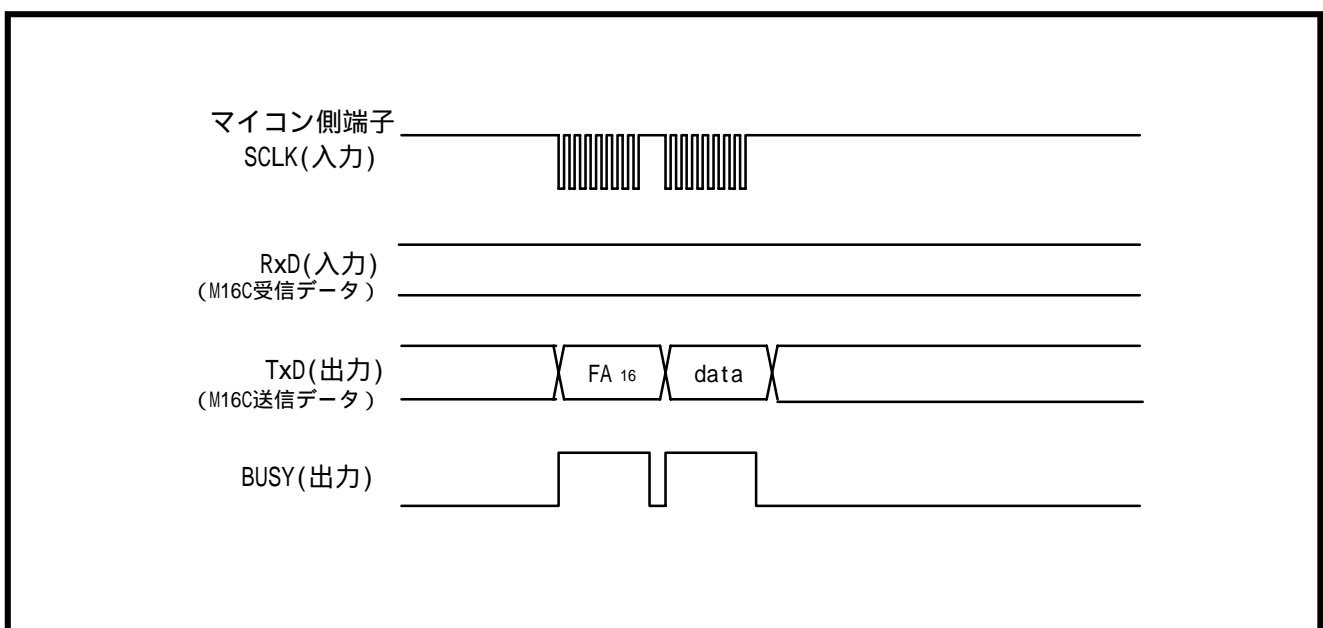


図 3.2.3.ダウンロード結果出力時のタイミング

バージョン情報出力

ブートローダのバージョン情報を出力するコマンドです。以下の手順でバージョン情報出力のコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "FB16" を転送します。
- (2) 2 バイト目から 9 バイト目のシリアル転送でバージョン情報を出力します。バージョン情報は、ASCII コード 8 文字で構成(*1)されています。

*1 バージョン情報のデータフォーマットは ASCII コード 8 文字で、
" VER . X . XX " (X ; 数字)
のような構成となっており、'V'から出力します。

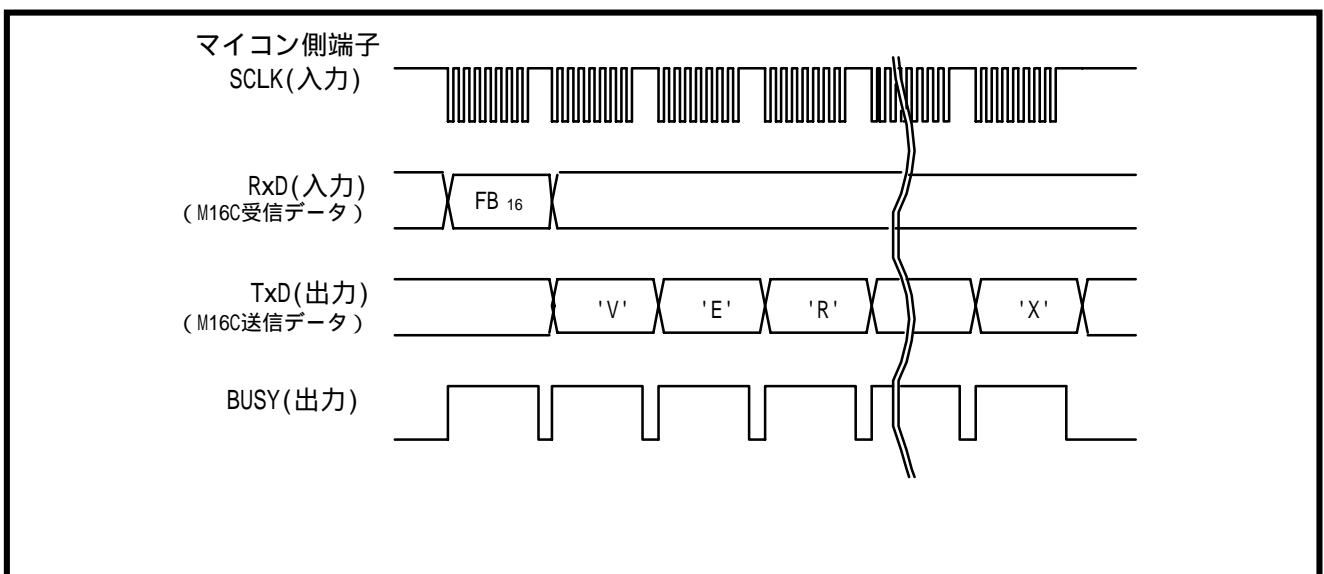


図 3.2.4.バージョン情報出力コマンド時のタイミング

3.2.2 フラッシュメモリ制御機能

機能概要

ブートローダは、外付けフラッシュメモリが三菱製フラッシュメモリ M5M29GB/T160BVP、M5M29GB/T320BVP または、同フラッシュメモリを混載する MCM である場合には、そのフラッシュメモリに対して書き換えプログラムなしで、書き込み、消去などを行うことができ、消去などを行うことができます。
(接続例を「[3.7 ブートローダ起動時の接続例](#)」に示します。)

フラッシュメモリへの書き込みや消去は、フラッシュメモリ制御用ソフトウェアコマンドやデータをシリアルライタと送受信することで行います。

ソフトウェアコマンド

表 3.2.2 にブートローダモード1フラッシュメモリ制御用ソフトウェアコマンドを示します。

これらのコマンドは、外付けフラッシュメモリが M5M29GB/T160BVP、M5M29GB/T320BVP または、同フラッシュメモリを混載する MCM のときだけに使用できるコマンドです。

表 3.2.2.ブートローダモード1 フラッシュメモリ制御用ソフトウェアコマンド

	制御コマンド名	1バイト目の転送	2バイト目	3バイト目	4バイト目	5バイト目	6バイト目	~
1	ヘジリット	FA ₁₆	アドレス (中位)	アドレス (上位)	データ出力	データ出力	データ出力	~259バイト目 データ出力
2	ヘジプログラム	41 ₁₆	アドレス (中位)	アドレス (上位)	データ入力	データ入力	データ入力	~259バイト目 データ入力
3	ブロックイレズ	20 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
4	イレズ全アンロックロック	A7 ₁₆	DO ₁₆					
5	リードステータスレジスタ	70 ₁₆	SRD 出力	SRD1 出力				
6	クリアステータスレジスタ	50 ₁₆						
7	リードロックビットステータス	71 ₁₆	アドレス (中位)	アドレス (上位)	ロックビット データ出力			
8	ロックビットプログラム	77 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
9	リードチェックデータ	FD ₁₆	データ出力 (下位)	データ出力 (上位)				

- ・ 網掛けは、マイコン シリアルライタへの転送
それ以外は、シリアルライタ マイコンへの転送
- ・ SRD はステータスレジスタデータ、SRD1 はステータスレジスタ1データ。

ページリード

フラッシュメモリの指定されたページ（256 バイト）を 1 バイトずつ順番に読み出すコマンドです。読み出す領域は上位アドレス（A16～A23）と中位アドレス（A8～A15）で設定し、xxxx0016～xxxxFF16 番地の 256 バイトが対象となります。（図 3.2.5 参照）

以下の手順でページリードコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード"FF16"を転送します。
- (2) 読み出す領域の A8～A15 を 2 バイト目で、A16～A23 を 3 バイト目で転送します。
- (3) 4 バイト目以降にクロックの立ち下がりに同期してアドレス A8～A23 で指定したページ（256 バイト）のデータ（D0～D7）を最小のアドレスから順番に出力します。

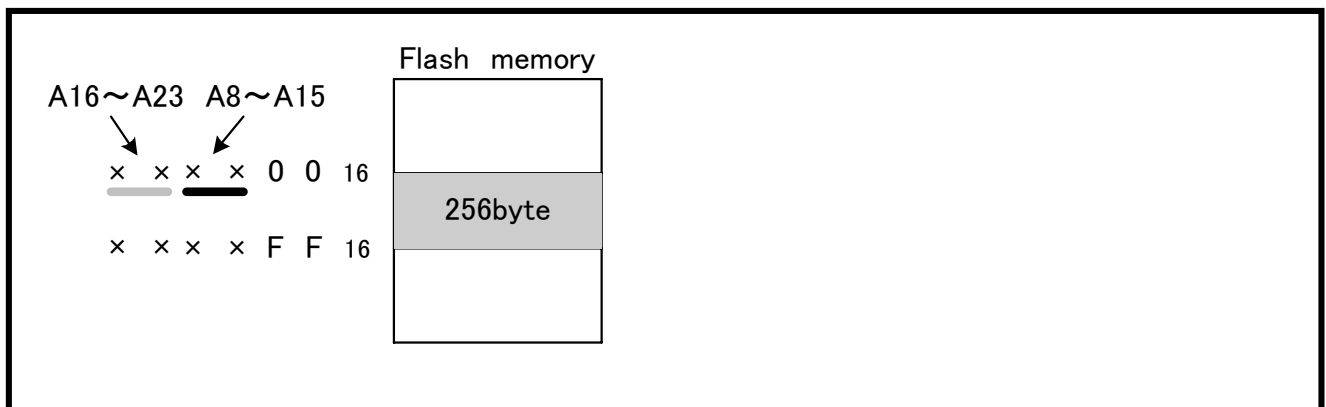


図 3.2.5.アドレスの指定とコマンド対象領域

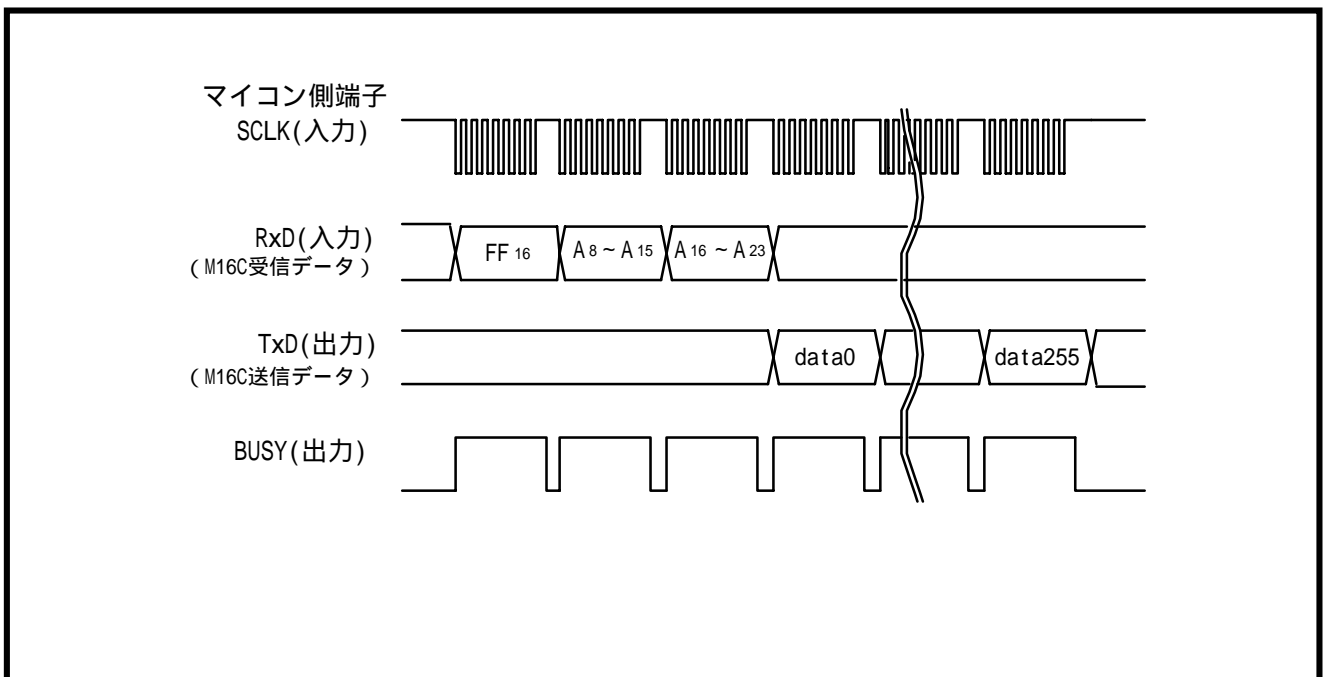


図 3.2.6.ページリードコマンド時のタイミング

ページプログラム

フラッシュメモリの指定されたページ (256 バイト) を 1 バイトずつ順番に書き込むコマンドです。書き込む領域は上位アドレス (A16 ~ A23) と中位アドレス (A8 ~ A15) で設定し、xxxx0016 ~ xxxxFF16 番地の 1 ページが対象となります。

以下の手順でページプログラムコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "4116" を転送します。
- (2) 書き込む領域の A8 ~ A15 を 2 バイト目で、A16 ~ A23 を 3 バイト目で転送します。
- (3) 4 バイト目以降、ライトデータ (D0 ~ D7) を、指定したページの最小アドレスから順番に 256 バイト入力すると、自動的に指定したページに対し書き込み動作を開始します。

次の 256 バイトの受信準備が完了すれば、BUSY 信号が "H" から "L" に変化します。ステータスレジスタを読み出すことにより、ページプログラムの結果を知ることができます。(「[リードステータスレジスタ](#)」参照)

なお、各ブロックはロックビットにより、書き込みをプロテクトすることが可能です。(「[ロックビットプログラム](#)」参照) 既にプログラムされたページには、再度プログラムを行うことはできません。

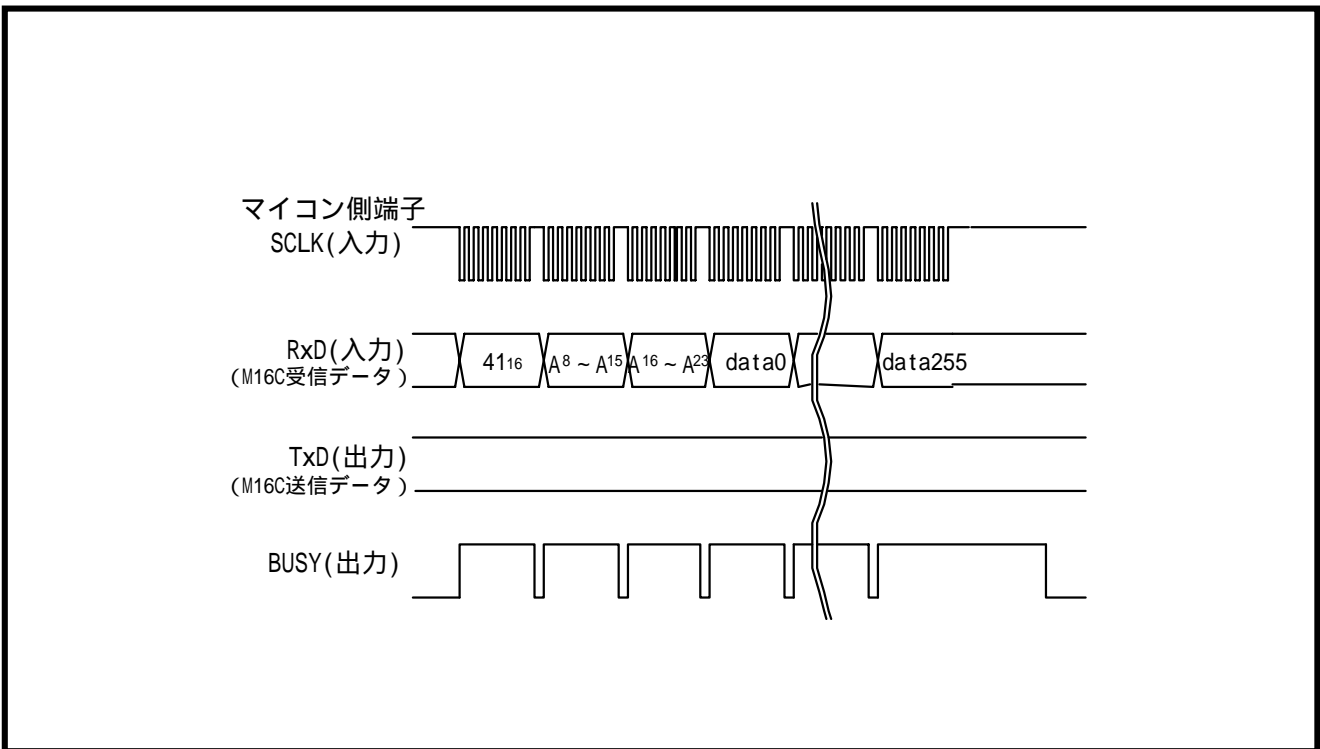


図 3.2.7. ページプログラムコマンド時のタイミング

ブロックイレーズ

フラッシュメモリの指定されたブロック内のデータを消去するコマンドです。以下の手順でブロックイレーズコマンドを実行してください。A8 ~ A23 のアドレスは、指定するブロックの最上位のアドレスとしてください。

- (1) 1 バイト目のシリアル転送でコマンドコード"20₁₆"を転送します。
- (2) 指定するブロックの最上位アドレスの A8 ~ A15 を 2 バイト目で、A16 ~ A23 を 3 バイト目で転送します。
- (3) 4 バイト目の転送で確認コマンドコード"D0₁₆"を転送すると、フラッシュメモリの指定ブロックに対する消去動作を開始します。

ブロックイレーズが終了すると BUSY 信号が"H"から"L"に変化します。ブロックイレーズを終了後、ステータスレジスタを読み出すことにより、ブロックイレーズの結果を知ることができます。（「[リードステータスレジスタ](#)」参照）

なお、各ブロックはロックビットにより、消去をプロテクトすることが可能です。（「[ロックビットプログラム](#)」参照）

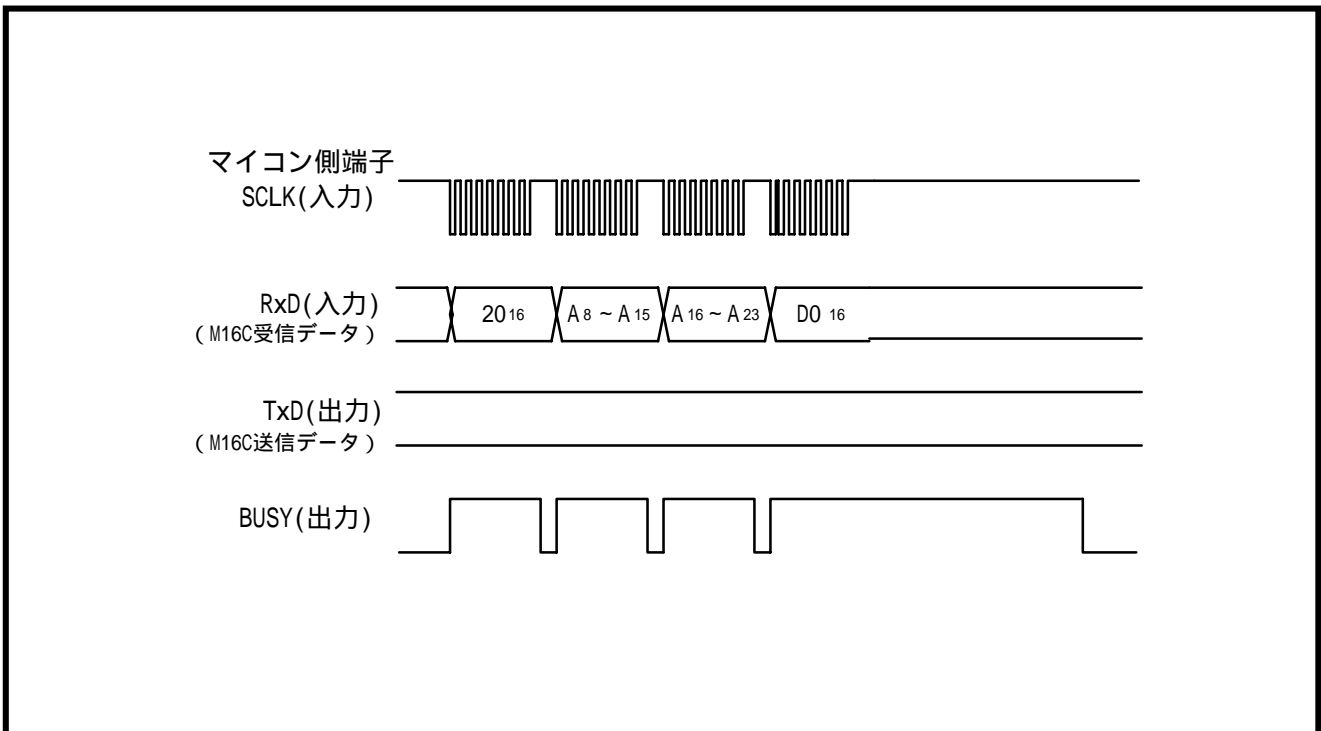


図 3.2.8. ブロックイレーズコマンド時のタイミング

イレース全アンロックブロック

フラッシュメモリの全ブロックを消去するコマンドです。以下の手順でイレース全アンロックブロックコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "A716" を転送します。
- (2) 2 バイト目の転送で確認コマンドコード "D016" を転送すると、全ブロックに対し、連続的にブロックイレース動作を開始します。

イレース全アンロックブロックイレースが終了すると BUSY 信号が "H" から "L" に変化します。ステータスレジスタを読み出すことにより、イレースの結果を知ることができます。（「[リードステータスレジスタ](#)」参照）

なお、各ブロックはロックビットにより、消去をプロテクトすることが可能です。（「[ロックビットプログラム](#)」参照）

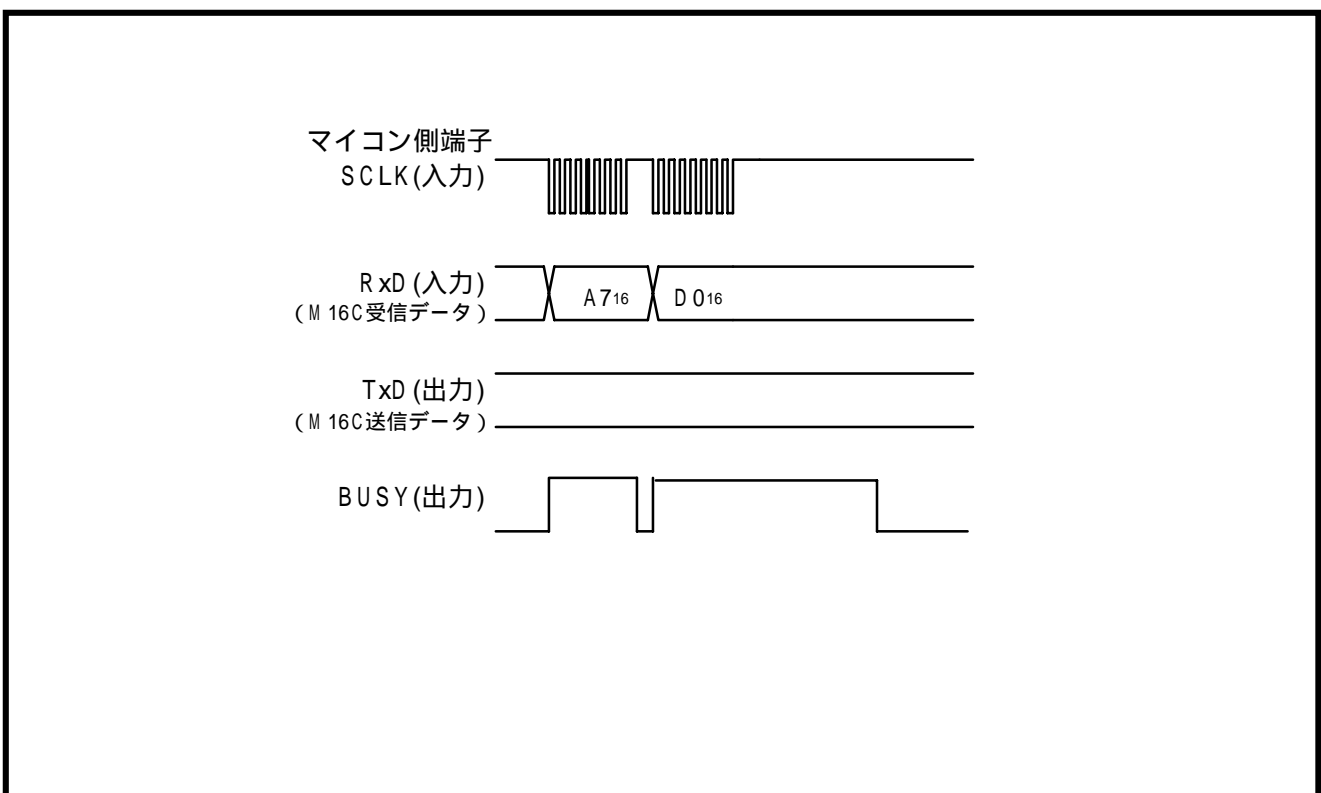


図 3.2.9.イレース全アンロックブロックコマンド時のタイミング

リードステータスレジスタ

ステータス情報を読み出すコマンドです。以下の手順でリードステータスレジスタコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード"70₁₆"を転送します。
- (2) 2 バイト目の転送でステータスレジスタ (SRD)、3 バイト目の転送でステータスレジスタ 1 (SRD1) の内容を出力します。

ステータスレジスタの内容については、ステータスレジスタおよびステータスレジスタ 1 を参照してください。

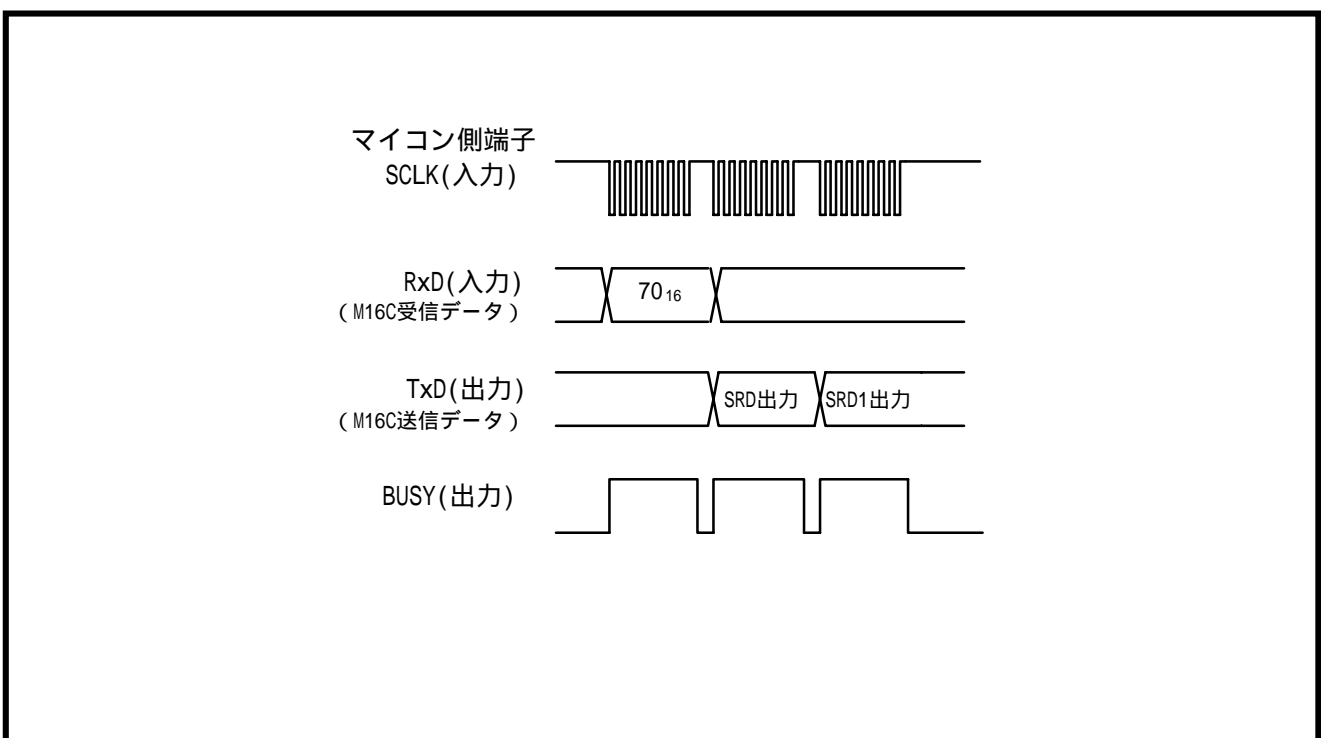


図 3.2.10.リードステータスレジスタコマンド時のタイミング

ステータスレジスタ (SRD)

ステータスレジスタは、フラッシュメモリの動作状態やイレーズ、プログラムの正常終了やエラー終了等の状態を示すレジスタで、リードステータスレジスタコマンド (70₁₆) を実行することで読み出すことができます。また、ステータスレジスタはクリアステータスレジスタコマンド (50₁₆) を実行することで"80₁₆"になります。

リセット後、リードステータスレジスタコマンドを実行すると、"80₁₆"を出力します。

表 3.2.3 にステータスレジスタを示します。また、各ビットを次に説明します。

表 3.2.3.ステータスレジスタ (SRD)

SRDの 各ビット	ステータス名	定義	
		"1"	"0"
SR7 (bit7)	ライトステートマシンステータス	レディ	ビジー
SR6 (bit6)	リザーブ	-	-
SR5 (bit5)	イレーズステータス	エラー終了	正常終了
SR4 (bit4)	プログラムステータス	エラー終了	正常終了
SR3 (bit3)	ブロックステータスアフタプログラム	エラー終了	正常終了
SR2 (bit2)	リザーブ	-	-
SR1 (bit1)	リザーブ	-	-
SR0 (bit0)	リザーブ	-	-

ライトステートマシンステータス (SR7)

ライトステートマシンステータスは、フラッシュメモリの動作状況を知らせるもので電源投入時"1" (レディ) になります。

書き込みや消去の動作中は"0" (ビジー) になりますが、これらの動作終了とともに"1"になります。

イレーズステータス (SR5)

イレーズステータスは、消去の動作状況を知らせるもので、消去エラーが発生すると"1"になります。イレーズステータスは、クリアステータスレジスタコマンドを実行すると"0"になります。

プログラムステータス (SR4)

プログラムステータスは、書き込みの動作状況を知らせるもので、書き込みエラーが発生すると"1"になります。プログラムステータスは、クリアステータスレジスタコマンドを実行すると"0"になります。

ブロックステータスアフタプログラム (SR3)

ブロックステータスアフタプログラムは、ページ書き込み完了時、過剰書き込みが発生した場合に"1"になります。ブロックステータスアフタプログラムは、クリアステータスレジスタコマンドを実行すると"0"になります。

SR5、SR4、SR3 のいずれかが、"1"の状態では、ページプログラム、ブロックイレーズ、イレーズ全アンロックブロック、ロックビットプログラムコマンドを受け付けません。これらのコマンドを実行する前にクリアステータスレジスタコマンド (50₁₆) を実行してください。

ステータスレジスタ 1 (SRD1)

ステータスレジスタ 1 は、シリアル通信の状態、チェックサム比較の結果等を示すレジスタで、リードステータスレジスタコマンド (70₁₆) を実行したとき、SRD に続いて読み出すことができます。また、ステータスレジスタ 1 はクリアステータスレジスタコマンド (50₁₆) を実行すると SR9 が"0"になります。表 3.2.4 にステータスレジスタ 1 を各ビットの定義を以下に示します。

表 3.2.4.ステータスレジスタ 1 (SRD1)

SRD1の 各ビット	ステータス名	定義	
		"1"	"0"
SR15 (bit7)	ブート更新済みビット	更新	未更新
SR14 (bit6)	リザーブ	-	-
SR13 (bit5)	リザーブ	-	-
SR12 (bit4)	チェックサム一致ビット	一致	不一致
SR11 (bit3)	リザーブ	-	-
SR10 (bit2)	リザーブ	-	-
SR9 (bit1)	データ受信タイムアウト	タイムアウト	正常動作
SR8 (bit0)	リザーブ	-	-

ブート更新済みビット (SR15)

ダウンロード機能を使用して書き換えプログラムを内部 RAM にダウンロードしたかどうかを示すフラグです。ダウンロード機能を用いてシリアル通信で書き換えプログラムを転送した後、このビットが"1"になります。

チェックサム一致ビット (SR12)

ダウンロード機能を使用して書き換えプログラムをダウンロードしたとき、チェックサムが一致したかどうかを示すフラグです。

データ受信タイムアウトビット (SR9)

データ受信中のタイムアウトエラーの発生を示すフラグです。データ受信中にこのフラグが立つと、受信したデータを破棄し、コマンド待ち状態に戻ります。

クリアステータスレジスタ

ステータスレジスタとステータスレジスタ 1 のエラー終了を示すビット(SR3 ~ SR5, SR9)が"1"になった後、これらを"0"にするためのコマンドです。1 バイト目のシリアル転送でコマンドコード"50₁₆"を転送すると、上記のビットを"0"にします。クリアステータスレジスタが終了すると、BUSY 信号は"H"から"L"に変化します。

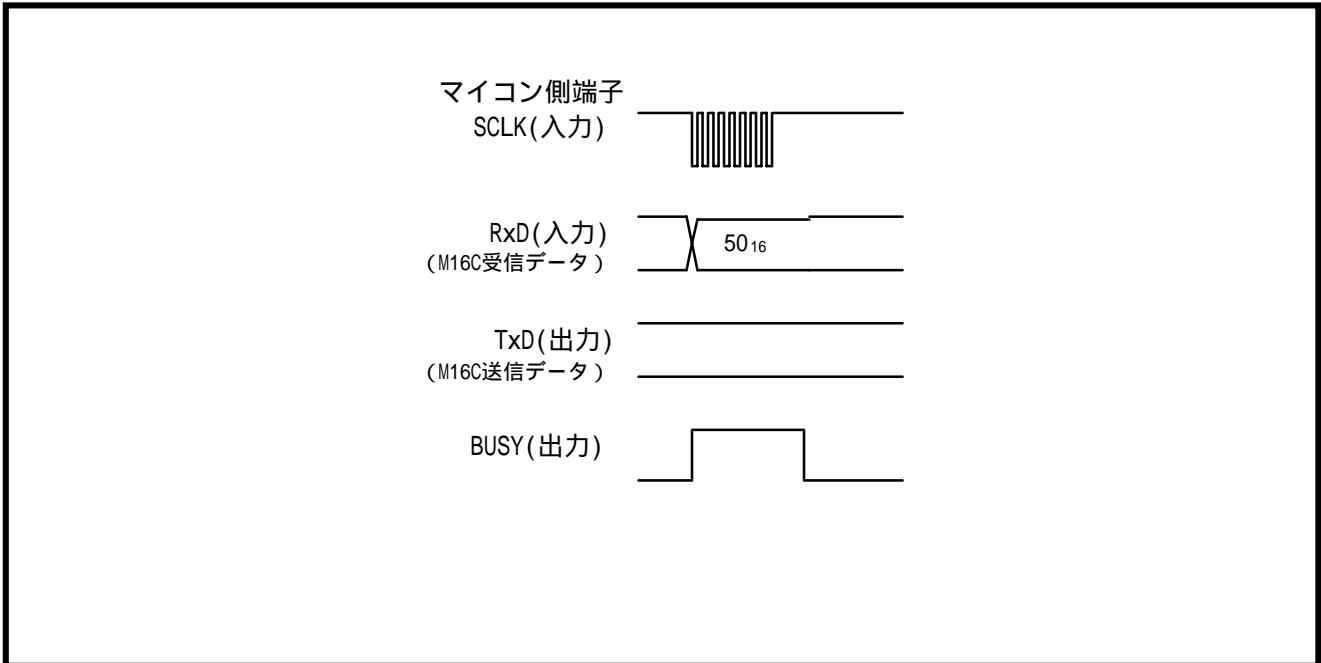


図 3.2.11.クアリアステータスレジスタ時のタイミング

リードロックビットステータス

指定したブロックのロックビットの状態を読み出すコマンドです。以下の手順でリードロックビットステータスコマンドを実行してください。A8～A23のアドレスは、指定するブロックの最上位のアドレスとしてください。ブロックの状態はロック、アンロックの状態があります。

- ロック・・・消去、書き込み禁止
- アンロック・・・消去、書き込み許可

- (1) 1バイト目のシリアル転送でコマンドコード"71₁₆"を転送します。
- (2) 指定するブロックの最上位アドレスのA8～A15を2バイト目で、A16～A23を3バイト目で転送します。
- (3) 4バイト目の転送でロックビットデータの内容を出力します。出力したデータの第6ビット目が状態を示します。"1"のときブロックアンロック、"0"のときブロックロックを示します。

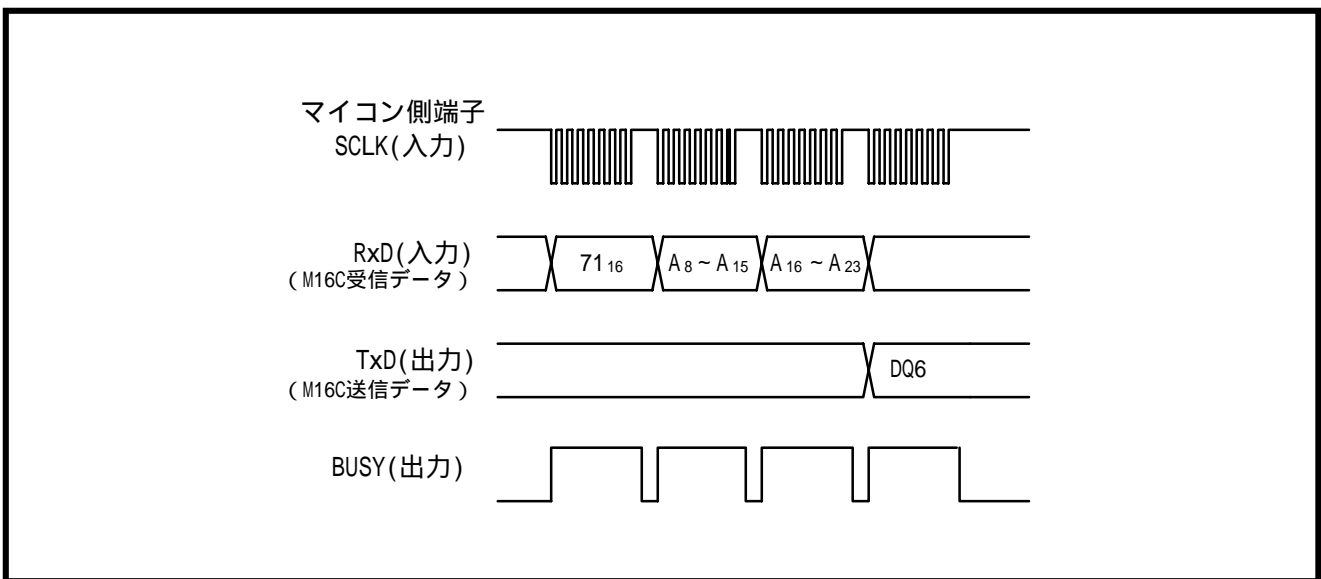


図 3.2.12.リードロックビットステータス時のタイミング

ロックビットプログラム

指定したブロックのロックビットに"0" (ロック状態) を書き込むコマンドです。以下の手順でロックビットプログラムコマンドを実行してください。A8 ~ A23 のアドレスは、指定するブロックの最上位のアドレスとしてください。

- (1) 1 バイト目のシリアル転送でコマンドコード "77₁₆" を転送します。
- (2) 指定するブロックの最上位アドレスの A8 ~ A15 を 2 バイト目で、A16 ~ A23 を 3 バイト目で転送します。
- (3) 4 バイト目の転送で確認コマンドコード "D0₁₆" を転送すると、指定ブロックのロックビットに"0"が書き込まれます。

書き込みが終了すると BUSY 信号は"H"から"L"に変化します。ロックビットの状態はリードロックビットステータスコマンドで読み出すことができます。

なお、ロックビットは、フラッシュメモリのライトプロテクトピンに"L"を入力すると有効に、"H"を入力すると無効になります。(M5M29GB/T160BVP、M5M29GB/T320BVP のデータシート参照)

ロックビットを"1"(アンロック状態)に戻すには、フラッシュメモリのライトプロテクトピンを"H"にして、ブロックイレーズまたはイレーズ全アンロックブロックコマンドを実行してください。

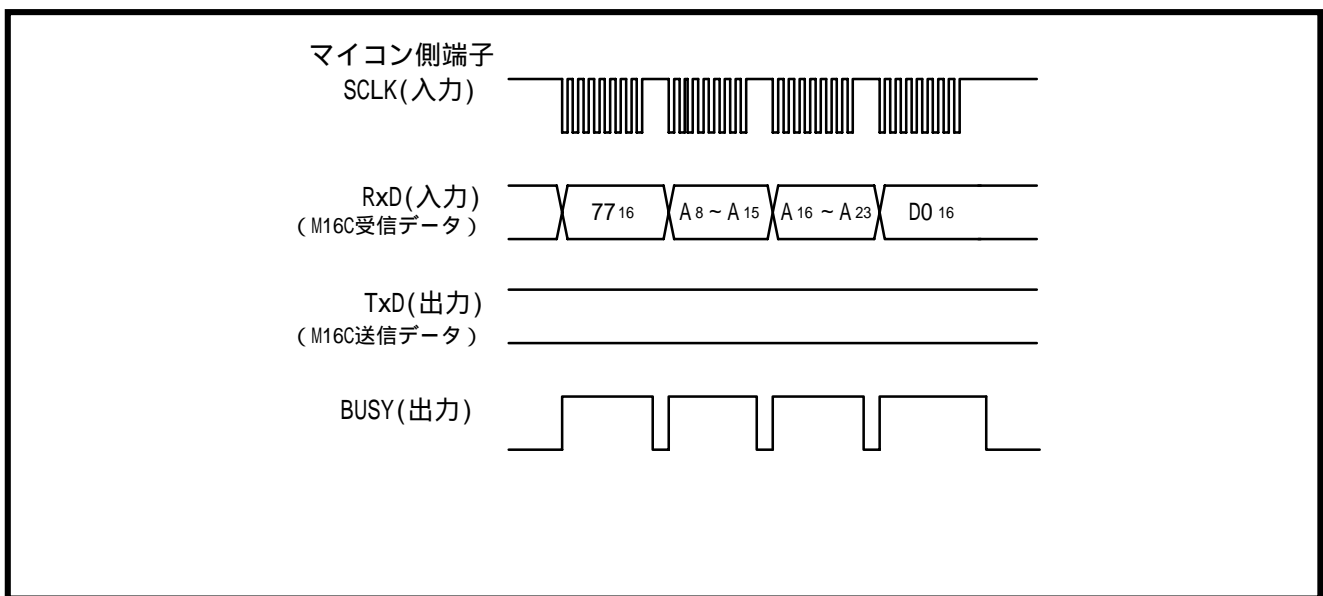


図 3.2.13.ロックビットプログラム時のタイミング

リードチェックデータ

ページプログラムコマンドでシリアルライタが送信した書き込みデータに対し、マイコンが正しく受信したことを確認するためのチェックデータを読み出すコマンドです。読み出されるチェックデータは 2 バイトで、このデータを読み出した後、チェックデータは"0000₁₆"になります。以下の手順で、リードチェックデータコマンドを実行してください。

表 3.2.5.チェックデータ方式

チェックデータ方式	計算方法
CRC 演算	M 16C のCRC 演算回路を用いて、CRC コードを算出する。

- (1)1 バイト目のシリアル転送でコマンドコード"FD₁₆"を転送します。
- (2)2 バイト目の転送でチェックデータ(下位)、3 バイト目の転送でチェックデータ(上位)を出力します。

このリードチェックデータコマンドを使用する場合、まず最初にこのコマンドを実行し、チェックデータを"0000₁₆"にしてください。次にページプログラムコマンドを必要回数実行してください。その後、再びリードチェックデータコマンドを実行すると、この間に実行したページプログラムコマンドで送信した書き込みデータ全てのチェックデータが読み出せます。

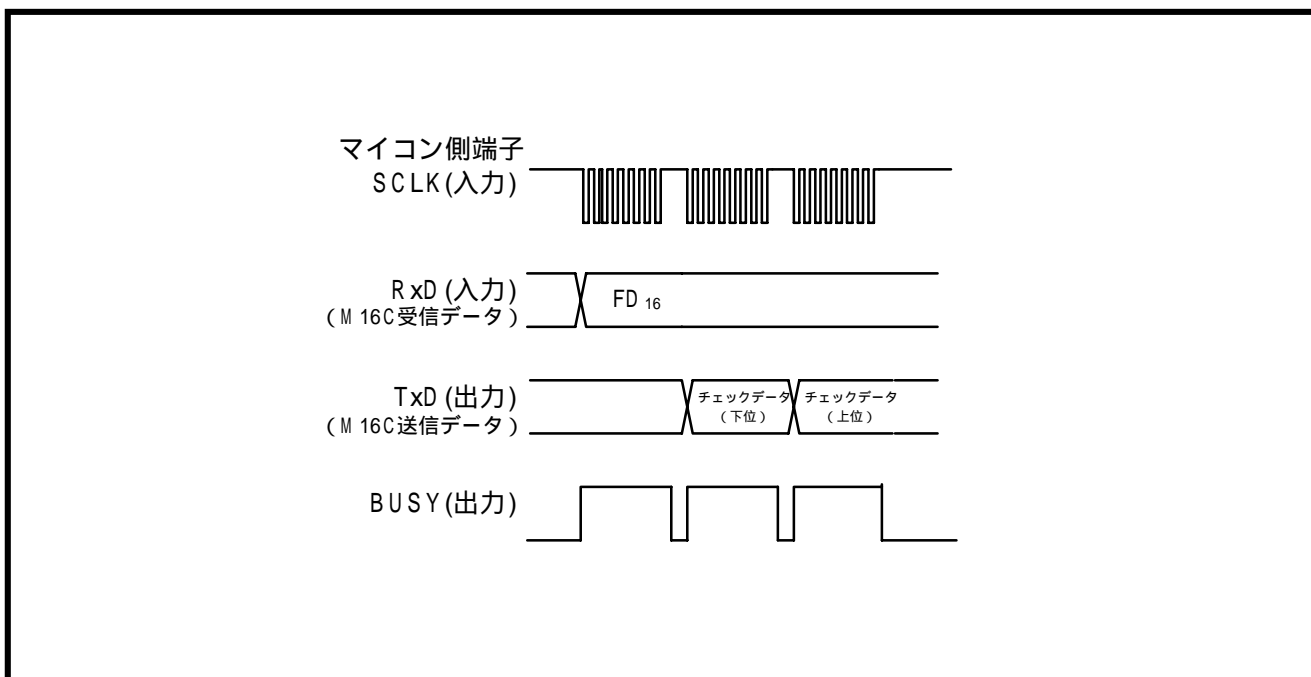


図 3.2.14.リードチェックデータコマンド時のタイミング

3.3 ブートローダモード 2 (クロック非同期形) 機能概要

ブートローダモード 2 では、2 線式クロック非同期形のシリアル I/O (UART1)を用いてシリアルライタ(*1)との間でソフトウェアコマンド、アドレス、データ等の入出力を行います。このモードを使用する場合、メインクロックの入力発振周波数は 2MHz 以上 20MHz 以下にしてください。P65 端子に"L"を印加してリセットを解除すると、ブートローダモード 2 になります。

TxD 端子は、CMOS 出力です。データ転送は、8 ビット単位、LSB ファースト、1 ストップビット、パリティ禁止で行います。

リセット後、転送速度 9600bps で接続が可能になり、その後 9600bps、19200bps、38400bps、57600bps、115200bps に変更できます。

以下、シリアルライタとの初期通信、周波数判定方法、およびブートローダモード 2 でサポートするダウンロード機能とフラッシュメモリ制御機能について説明します。

*1 シリアルライタとして、M16C Flash Starter を使用することができます。

3.3.1 シリアルライタとの初期通信について

リセット後、シリアルライタと初期通信を行い、マイコン側の転送速度を 9600bps に調整します。次の手順で初期通信を行ってください。

(1)シリアルライタの転送速度を 9600bps にして、シリアルライタから"0016"を 16 回転送します。このとき転送間隔を 15ms 以上あけてください。(マイコンは、"0016"が正しく受信できるように転送速度レジスタを設定します。)

(2)マイコンは、確認コード"B016"を出力し、初期通信を終了します。(*2)

図 3. 3.1 にシリアルライタとの初期通信の手順を、図 3.3.2 に初期通信の入出力タイミングを示します。

*2 シリアルライタが"B016"を正しく受信できない場合は、マイコンのメインクロック入力発振周波数を変更してください。

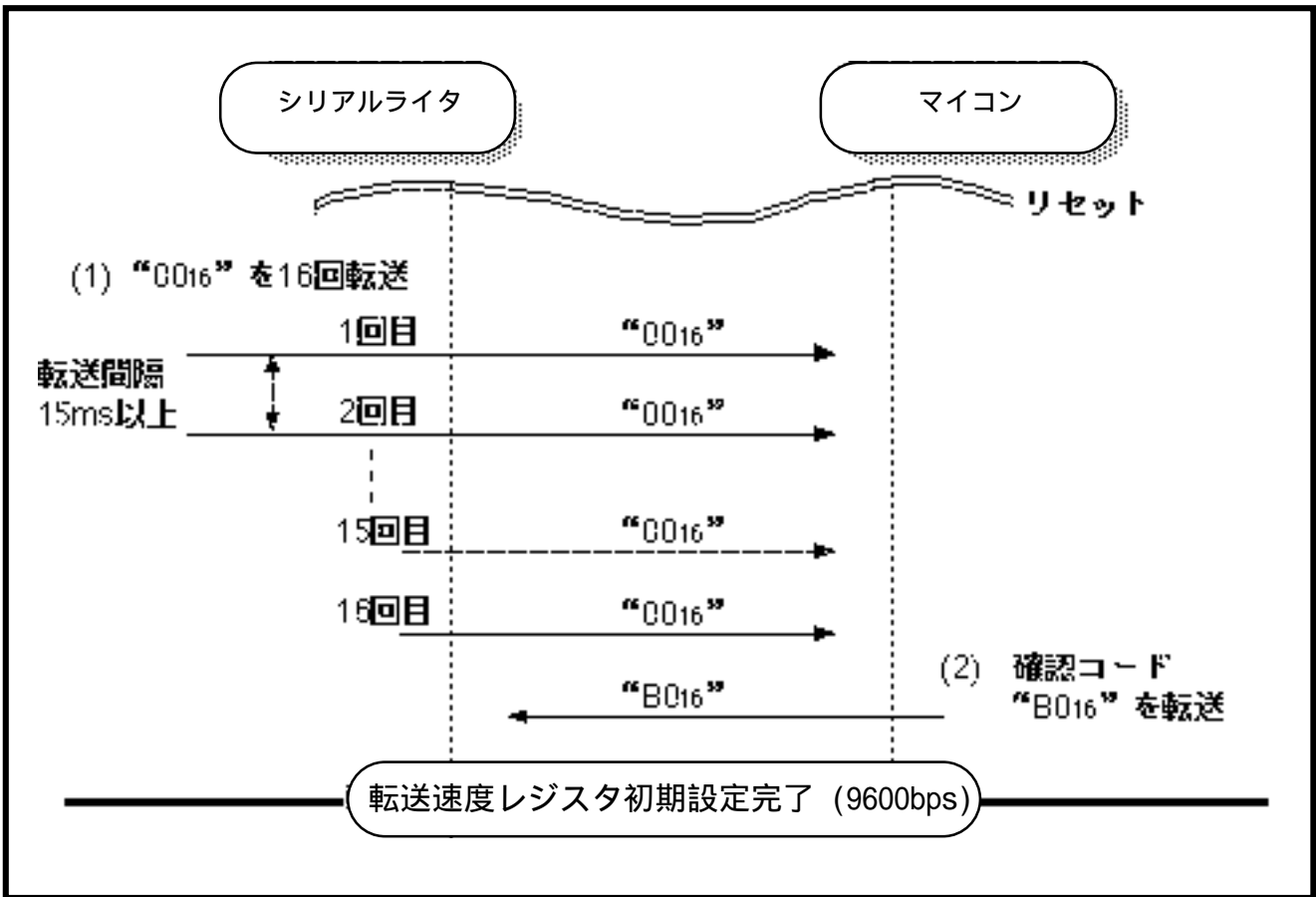


図 3.3.1.シ リアルライターとの初期通信の手順

マイコンから見た、初期通信の入出力タイミング

- (1) ターゲット基板電源ON
- (2) フラッシュ書き込み制御開始
- (3) リセット解除によりモードエントリ
- (4) シリアルライター電源ON
- (5) 初期通信開始
- (6) 初期通信完了

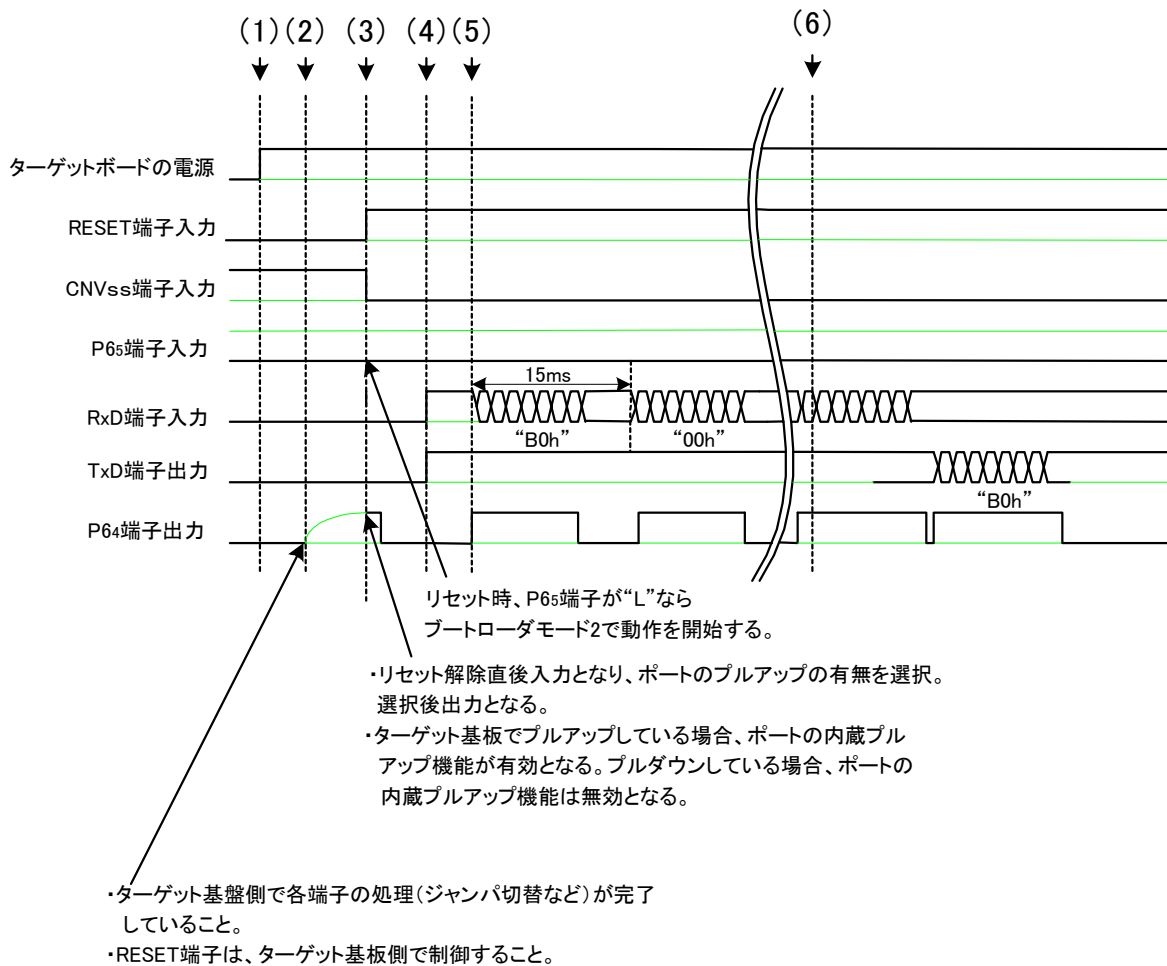


図 3.3.2.初期通信の入出力タイミング

3.3.2 メインクロック入力発振周波数と転送速度

メインクロック入力発振周波数によっては、希望する転送速度にならない場合があります。表 3.3.1 にメインクロック入力発振周波数と転送速度を示します。

表 3.3.1.メインクロック入力発振周波数と転送速度

メインクロック 入力発振周波数 (MHz)	転送速度 9600bps	転送速度 19200bps	転送速度 38400bps	転送速度 57600bps	転送速度 115200bps
20MHz					
16MHz					
12MHz					×
10MHz					×
8MHz					×
7.3728MHz					
6MHz				×	×
5MHz				×	×
4.5MHz					×
4.194304MHz				×	×
4MHz			×	×	×
3.58MHz					
2MHz		×	×	×	×

： 通信可能

×： 通信不可

3.3.3 ダウンロード機能

機能概要

ブートローダのダウンロード機能は、シリアル通信を使って書き換えプログラム(*1)をマイコンの内部 RAM にダウンロードし、ダウンロードした RAM 上の番地へ飛ばします。

*1 書き換えプログラムは、ユーザで作成してください。

書き換えプログラムには、外付けフラッシュメモリに対する制御機能（書き込み、消去、読み出しなど）および、シリアルライタとの通信機能を有している必要があります。

書き換えプログラムでスタックを使用する場合、そのプログラム内でスタックポインタを設定してください。

ダウンロード完了後、マイコンはシングルチップモードで動作します。外付けフラッシュメモリへの書き込みや消去などの制御を開始する前に、書き換えプログラムでプロセッサモードをシングルチップモードからマイクロプロセッサモードに変更してください。

書き換えプログラムでは、割り込みは使用しないでください。

ダウンロードエリアについては、「[3.6 メモリマップ](#)」を参照してください。

ソフトウェアコマンド

表 3.3.2 に、ブートローダモード 2 用ソフトウェアコマンドを示します。

表 3.3.2.ブートローダモード 2 用ソフトウェアコマンド

	制御コマンド名	1 バイト目 の転送	2 バイト目	3 バイト目	4 バイト目	5 バイト目	6 バイト目	~
1	ダウンロード	FA ₁₆	サイズ (下位)	サイズ (上位)	チェックサム	データ入力	~ 必要回数	
2	ダウンロード 結果出力	FA ₁₆	データ入力					
3	バージョン情報出力	FB ₁₆	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	~9 バイト目 バージョンデータ 出力
4	ポインタ 9600	B0 ₁₆	B0 ₁₆					
5	ポインタ 19200	B1 ₁₆	B1 ₁₆					
6	ポインタ 38400	B2 ₁₆	B2 ₁₆					
7	ポインタ 57600	B3 ₁₆	B3 ₁₆					
8	ポインタ 115200	B4 ₁₆	B4 ₁₆					

網掛けは、マイコン シリアルライタへの転送
それ以外は、シリアルライタ マイコンへの転送

ダウンロード

内部 RAM に書き換えプログラムをダウンロードするコマンドです。ダウンロードされたプログラムは内部 RAM の 600₁₆ 番地以降へ格納されます。

ダウンロード後にリセットしても、内部 RAM に転送されたプログラムは保持されます。

ダウンロードは、以下の手順で実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード"FA₁₆"を転送します。
- (2) 2 バイト目のシリアル転送でプログラムサイズの下位、3 バイト目の転送でプログラムサイズの上位を転送します。
- (3) 4 バイト目のシリアル転送でチェックサムを転送します。チェックサムは、5 バイト目以降に転送するデータを全て加算したものです。
- (4) 5 バイト目以降から書き換えプログラムを転送します。転送可能なプログラムの容量は、内部の RAM の容量によって異なります。(「[3.6 メモリマップ](#)」参照)

全データの転送が完了すると、マイコンは自動的にダウンロード結果出力コマンドを実行します。

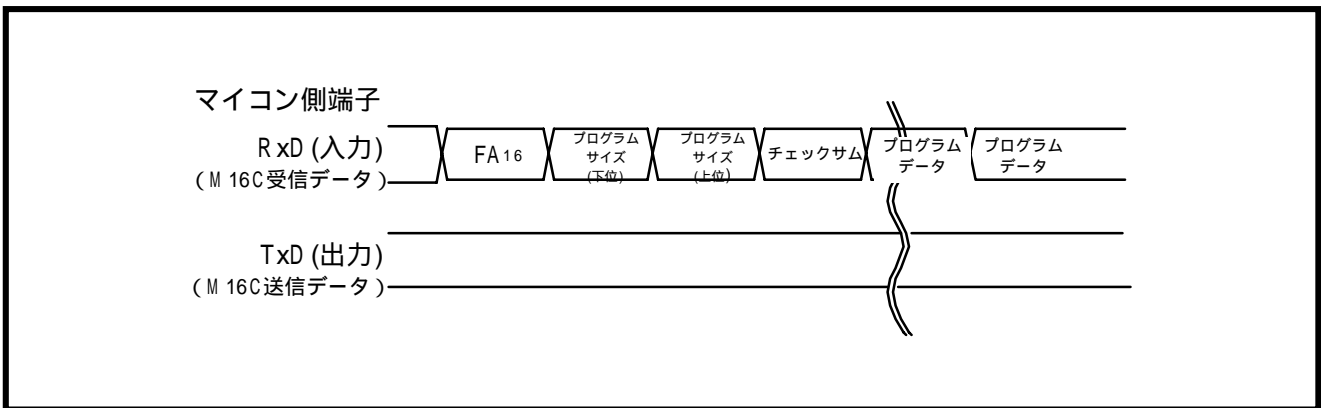


図 3.3.3.ダウンロードコマンド時のタイミング

ダウンロード結果出力

ダウンロードコマンド実行後、シリアルライタから転送されたチェックサム値とマイコンが受信したデータから求めたチェックサム値を比較します。チェックサム値が一致すれば、マイコンは"FA16"、"0016"（成功）を返送してから内部 RAM にダウンロードされたプログラムの先頭にジャンプし実行します。チェックサム値が不一致の場合、マイコンは"FA16"、"0116"（失敗）を返送してからマイコンに格納されているブートプログラムを改めて内部 RAM に転送し、これを実行します。（リセット後の元の状態に戻す。）

ダウンロード実行後、実行結果をブートローダ（マイコン）が以下の手順で出力します。

- (1)ダウンロード実行後、1 バイト目のシリアル転送で"FA16"を出力します。
- (2)2 バイト目のシリアル転送で、ダウンロードの実行結果により"0016"（成功）または"0116"（失敗）を出力します。

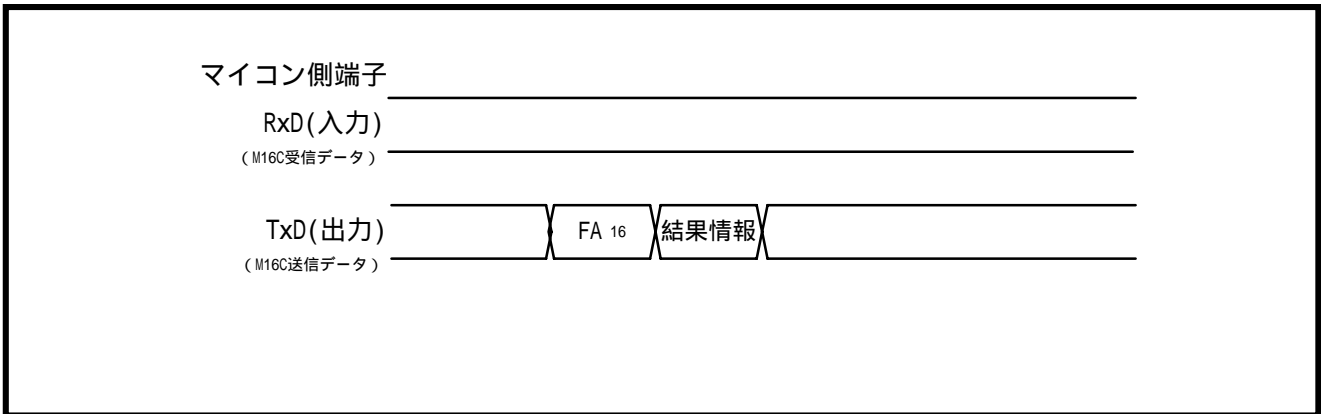


図 3.3.4.ダウンロード結果出力時のタイミング

バージョン情報出力

ブートローダのバージョン情報を出力するコマンドです。以下の手順でバージョン情報出力のコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード"FB16"を転送します。
- (2) 2 バイト目から 9 バイト目のシリアル転送でバージョン情報を出力します。バージョン情報は、ASCII コード 8 文字で構成(*1)されています。

*1 バージョン情報のデータフォーマットは ASCII コード 8 文字で、
"VER.X.XX" (X; 数字)
のような構成となっており、'V'から出力します。

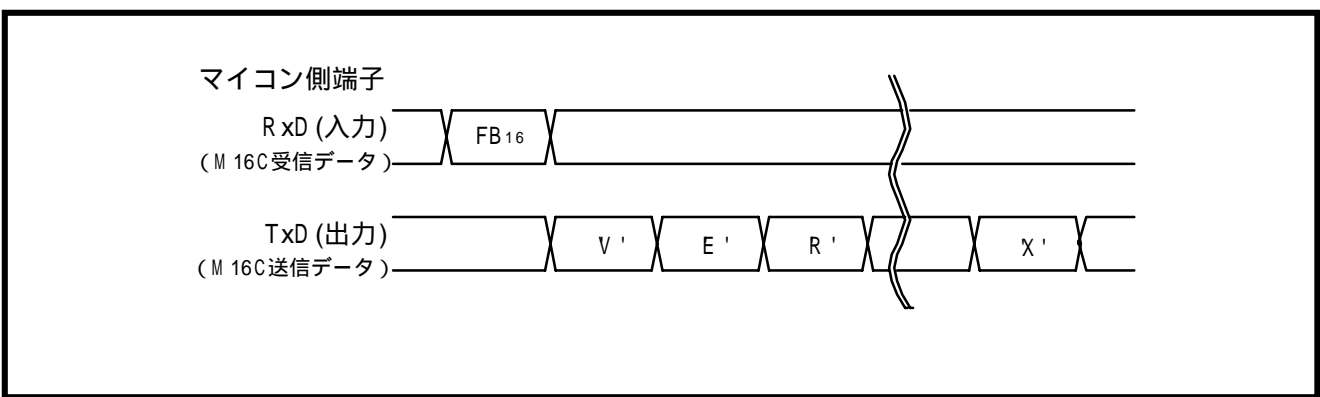


図 3.3.5.バージョン情報出力コマンド時のタイミング

ボーレート9600

転送速度を 9600bps に変更するコマンドです。以下の手順でコマンドを実行してください。

- (1)1 バイト目のシリアル転送でコマンドコード"B016"を転送します。
- (2)2 バイト目の転送で確認コード"B016"を出力した後、転送速度が 9600bps に変わります。

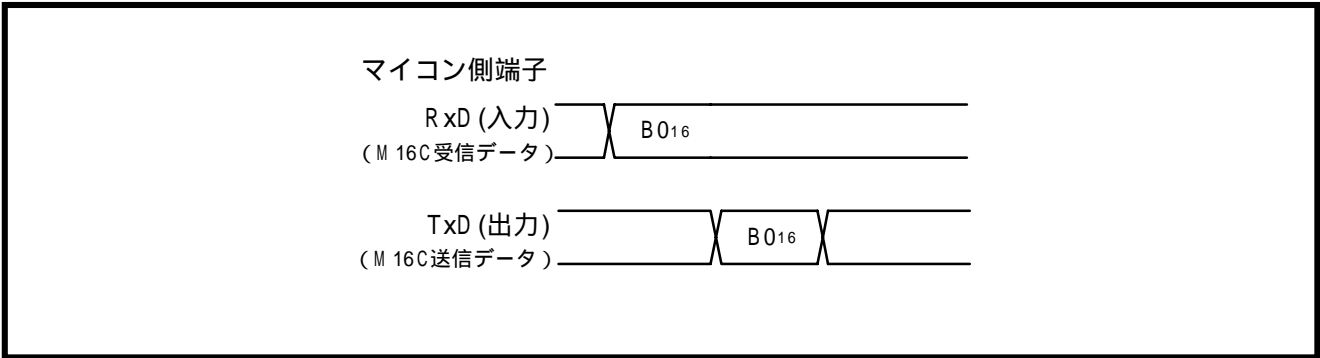


図 3.3.6.ボーレート9600のタイミング

ボーレート19200

転送速度を 19200bps に変更するコマンドです。以下の手順でコマンドを実行してください。

- (1)1 バイト目のシリアル転送でコマンドコード "B116"を転送します。
- (2)2 バイト目の転送で確認コード"B116"を出力した後、転送速度が 19200bps に変わります。

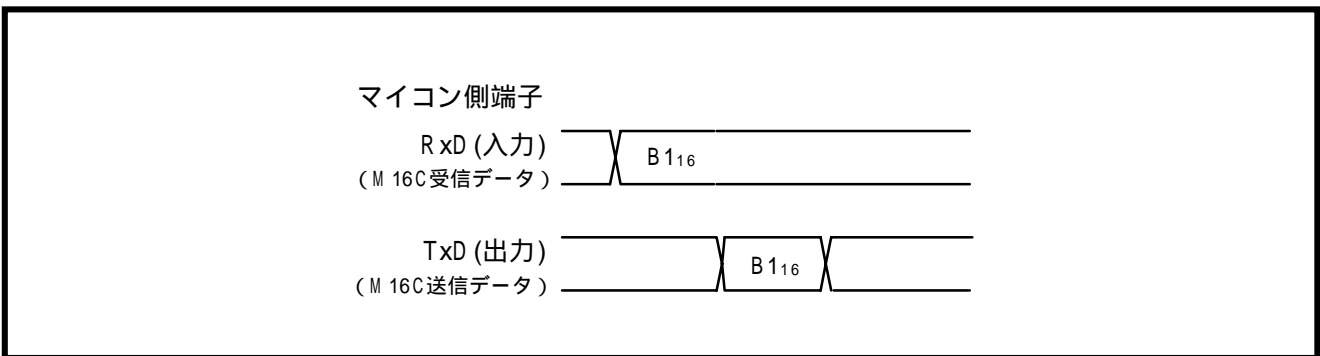


図 3.3.7.ボーレート19200のタイミング

ボーレート38400

転送速度を 38400bps に変更するコマンドです。以下の手順でコマンドを実行してください。

- (1)1 バイト目のシリアル転送でコマンドコード"B216"を転送します。
- (2)2 バイト目の転送で確認コード"B216"を出力した後、転送速度が 38400bps に変わります。

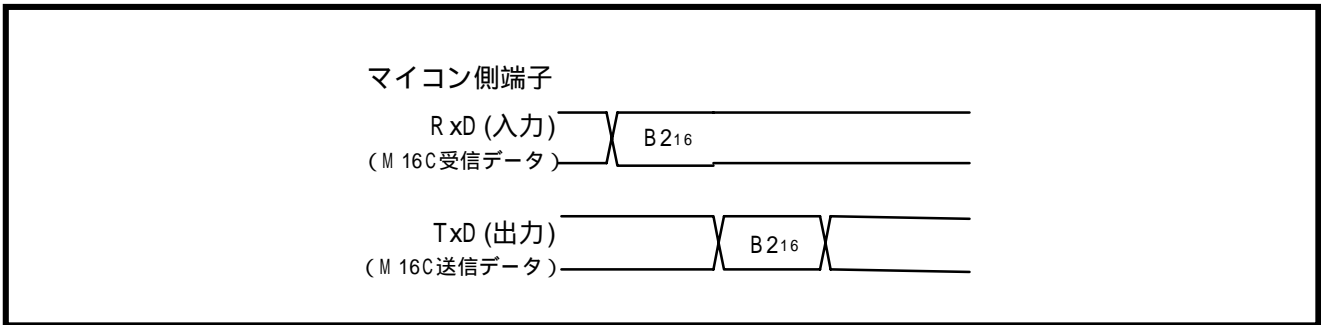


図 3.3.8.ボーレート38400のタイミング

ボーレート57600

転送速度を 57600bps に変更するコマンドです。以下の手順でコマンドを実行してください。

- (1)1 バイト目のシリアル転送でコマンドコード"B316"を転送します。
- (2)2 バイト目の転送で確認コード"B316"を出力した後、転送速度が 57600bps に変わります。

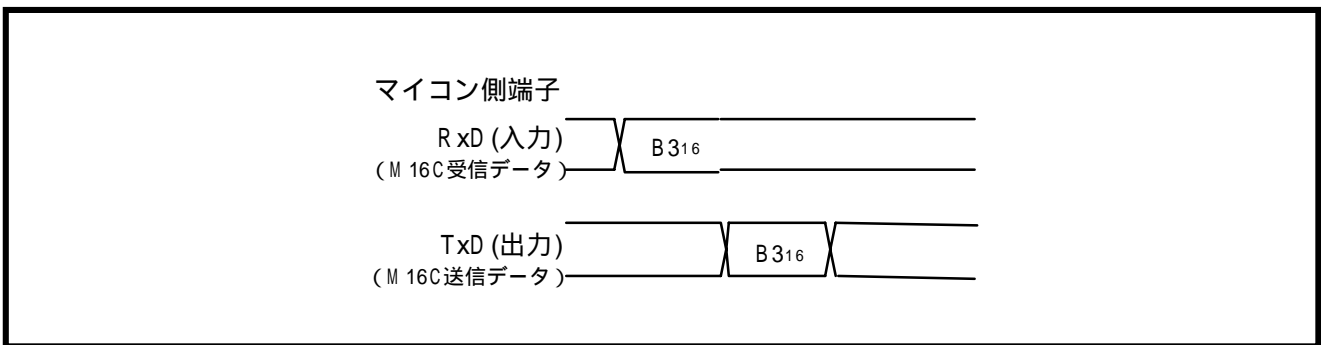


図 3.3.9.ボーレート57600のタイミング

ボーレート115200

転送速度を 115200bps に変更するコマンドです。以下の手順でコマンドを実行してください。

- (1)1 バイト目のシリアル転送でコマンドコード"B416"を転送します。
- (2)2 バイト目の転送で確認コード"B416"を出力した後、転送速度が 115200bps に変わります。

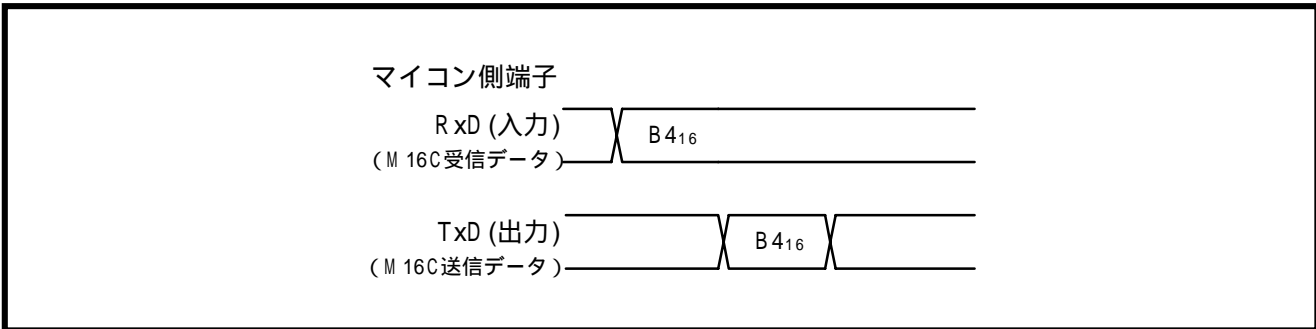


図 3.3.10.ボーレート115200のタイミング

3.3.4 フラッシュメモリ制御機能

機能概要

ブートローダは、外付けフラッシュメモリが三菱製フラッシュメモリ M5M29GB/T160BVP、M5M29GB/T320BVP または、同フラッシュを混載する MCM である場合には、そのフラッシュメモリに対して書き換えプログラムなしで、書き込みや消去などを行うことができます。（接続例を「[3.7 ブートローダ起動時の接続例](#)」に示します。）

フラッシュメモリへの書き込みや消去は、フラッシュメモリ制御用ソフトウェアコマンドやデータをシリアルライタと送受信することで行います。

ソフトウェアコマンド

表 3.3.3 に、ブートローダモード2 フラッシュメモリ制御用ソフトウェアコマンドを示します。

これらのコマンドは、外付けフラッシュメモリが M5M29GB/T160BVP、M5M29GB/T320BVP または同フラッシュメモリを混載する MCM のときだけに使用できるコマンドです。 これらのコマンドについては、前節の「[ブートローダモード2のダウンロード機能](#)」を参照してください。

表 3.3.3.ブートローダモード2 フラッシュメモリ制御用ソフトウェアコマンド

	制御コマンド名	1バイト目の転送	2バイト目	3バイト目	4バイト目	5バイト目	6バイト目	～
1	ページリード	FA ₁₆	アドレス (中位)	アドレス (上位)	データ出力	データ出力	データ出力	～259バイト目 データ出力
2	ページプログラム	41 ₁₆	アドレス (中位)	アドレス (上位)	データ入力	データ入力	データ入力	～259バイト目 データ入力
3	ブロックイレース	20 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
4	イレース全アンロックロック	A7 ₁₆	DO ₁₆					
5	リードステータスレジスタ	70 ₁₆	SRD 出力	SRD1 出力				
6	クリアステータスレジスタ	50 ₁₆						
7	リードロックビットステータス	71 ₁₆	アドレス (中位)	アドレス (上位)	ロックビット データ出力			
8	ロックビットプログラム	77 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
9	リードチェックデータ	FD ₁₆	データ出力 (下位)	データ出力 (上位)				
10	ホールド9600	B0 ₁₆	B0 ₁₆					
11	ホールド19200	B1 ₁₆	B1 ₁₆					
12	ホールド38400	B2 ₁₆	B2 ₁₆					
13	ホールド57600	B3 ₁₆	B3 ₁₆					
14	ホールド115200	B4 ₁₆	B4 ₁₆					

- ・ 網掛けは、マイコン シリアルライタへの転送
それ以外は、シリアルライタ マイコンへの転送
- ・ SRD はステータスレジスタデータ、SRD1 はステータスレジスタ1データ。

ページリード

フラッシュメモリの指定されたページ (256 バイト) を 1 バイトずつ順番に読み出すコマンドです。読み出す領域は上位アドレス (A16 ~ A23) と中位アドレス (A8 ~ A15) で設定し、xxxx0016 ~ xxxxFF16 番地の 256 バイトが対象となります。(図 3.3.11 参照)

以下の手順でページリードコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "FF16" を転送します。
- (2) 読み出す領域の A8 ~ A15 を 2 バイト目で、A16 ~ A23 を 3 バイト目で転送します。
- (3) 4 バイト目以降にクロックの立ち下がりに同期してアドレス A8 ~ A23 で指定したページ (256 バイト) のデータ (D0 ~ D7) を最小のアドレスから順番に出力します。

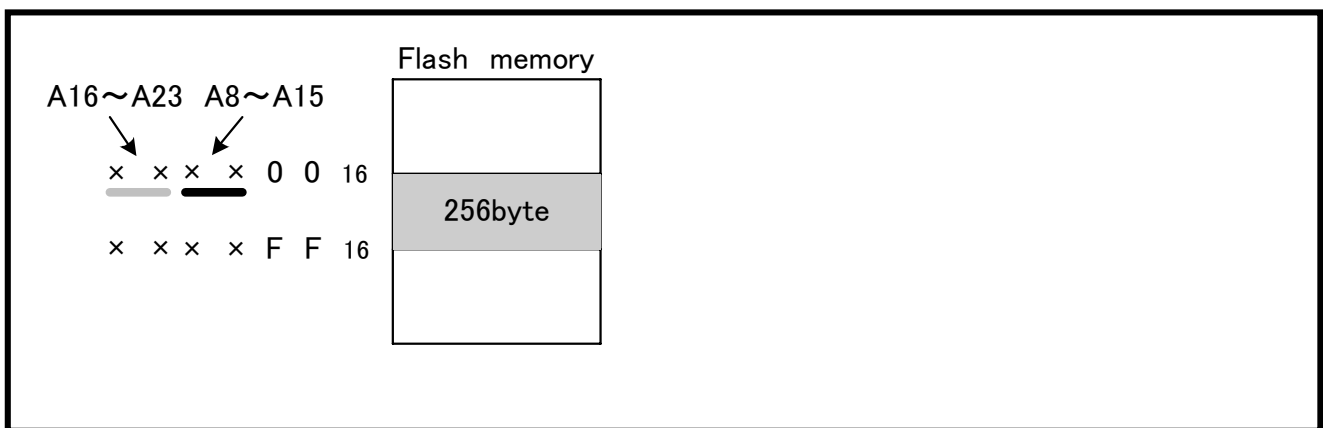


図 3.3.11.アドレスの指定とコマンド対象領域

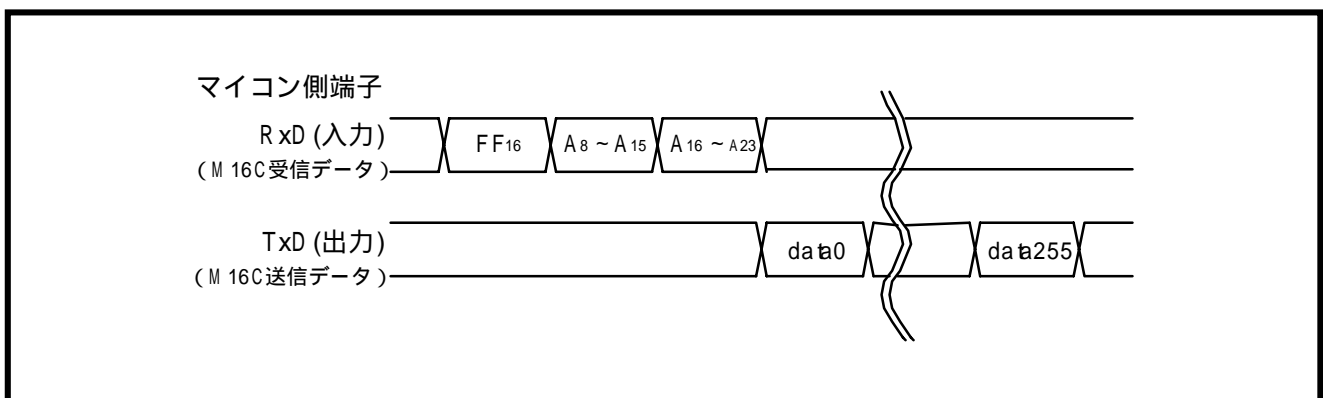


図 3.3.12.ページリードコマンド時のタイミング

ページプログラム

フラッシュメモリの指定されたページ（256 バイト）を 1 バイトずつ順番に書き込むコマンドです。書き込む領域は上位アドレス（A16～A23）と中位アドレス（A8～A15）で設定し、xxxx0016～xxxxFF16 番地の 1 ページが対象となります。

以下の手順でページプログラムコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード"4116"を転送します。
- (2) 書き込む領域の A8～A15 を 2 バイト目で、A16～A23 を 3 バイト目で転送します。
- (3) 4 バイト目以降、ライトデータ（D0～D7）を、指定したページの最小アドレスから順番に 256 バイト入力すると、自動的に指定したページに対し書き込み動作を開始します。

ステータスレジスタを読み出すことにより、ページプログラムの結果を知ることができます。（「[リードステータスレジスタ](#)」参照）

なお、各ブロックはロックビットにより、書き込みをプロテクトすることが可能です。（「[ロックビットプログラム](#)」参照）既にプログラムされたページには、再度プログラムを行うことはできません。

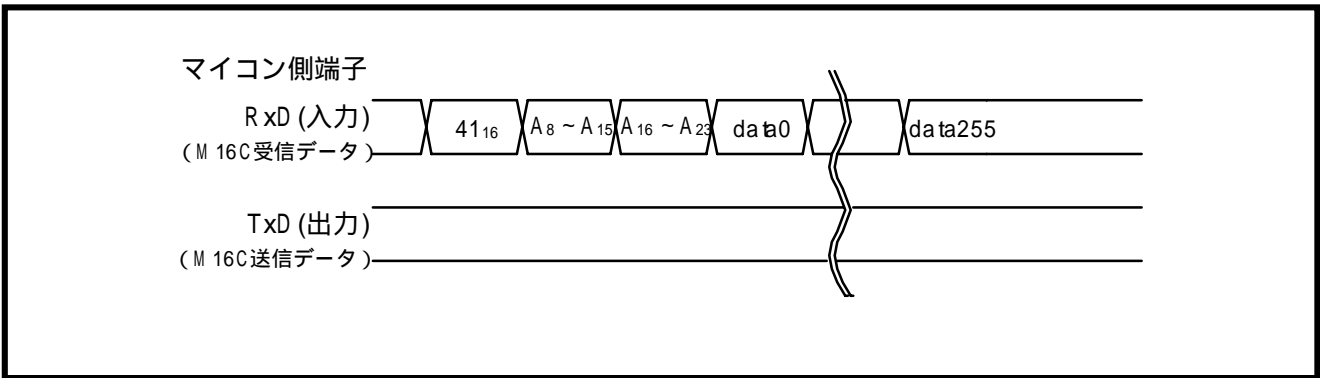


図 3.3.13. ページプログラムコマンド時のタイミング

ブロックイレーズ

フラッシュメモリの指定されたブロック内のデータを消去するコマンドです。以下の手順でブロックイレーズコマンドを実行してください。A8～A23のアドレスは、指定するブロックの最上位のアドレスとしてください。

- (1) 1 バイト目のシリアル転送でコマンドコード"20₁₆"を転送します。
- (2) 指定するブロックの最上位アドレスのA8～A15を2バイト目で、A16～A23を3バイト目で転送します。
- (3) 4 バイト目の転送で確認コマンドコード"D0₁₆"を転送すると、フラッシュメモリの指定ブロックに対する消去動作を開始します。

ブロックイレーズを終了後、ステータスレジスタを読み出すことにより、ブロックイレーズの結果を知ることができます。（「[リードステータスレジスタ](#)」参照）

なお、各ブロックはロックビットにより、消去をプロテクトすることが可能です。（「[ロックビットプログラム](#)」参照）

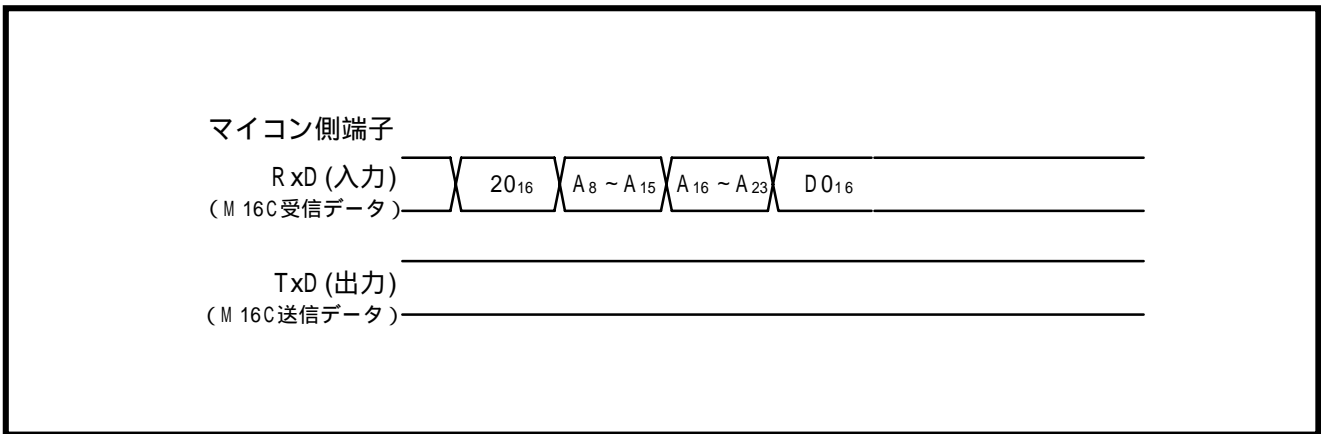


図 3.3.14. ブロックイレーズコマンド時のタイミング

イレース全アンロックブロック

フラッシュメモリの全ブロックを消去するコマンドです。以下の手順でイレース全アンロックブロックコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "A7₁₆" を転送します。
- (2) 2 バイト目の転送で確認コマンドコード "D0₁₆" を転送すると、全ブロックに対し、連続的にブロックイレース動作を開始します。

ステータスレジスタを読み出すことにより、イレースの結果を知ることができます。（「[リードステータスレジスタ](#)」参照）

なお、各ブロックはロックビットにより、消去をプロテクトすることが可能です。（「[ロックビットプログラム](#)」参照）

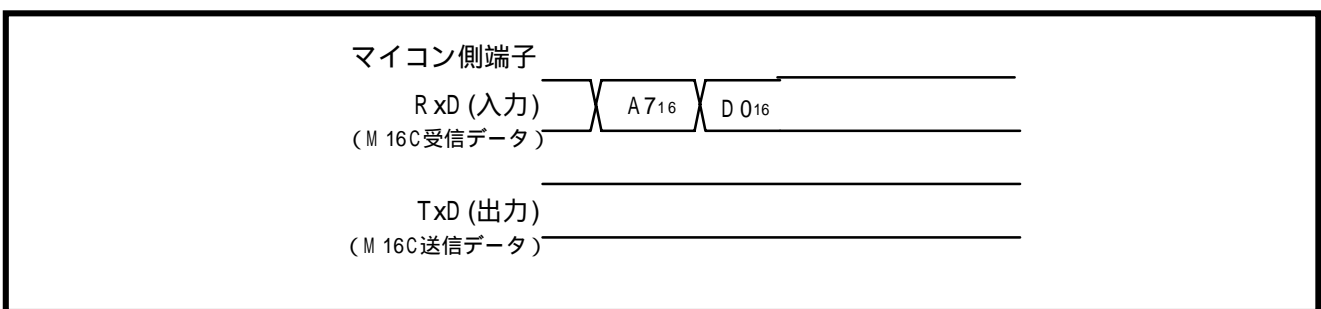


図 3.3.15.イレース全アンロックブロックコマンド時のタイミング

リードステータスレジスタ

ステータス情報を読み出すコマンドです。以下の手順でリードステータスレジスタコマンドを実行してください。

- (1) 1 バイト目のシリアル転送でコマンドコード "70₁₆" を転送します。
- (2) 2 バイト目の転送でステータスレジスタ (SRD)、3 バイト目の転送でステータスレジスタ 1 (SRD1) の内容を出力します。

ステータスレジスタの内容については、ブートローダモード 1 の「[ステータスレジスタ](#)」および「[ステータスレジスタ 1](#)」を参照してください。

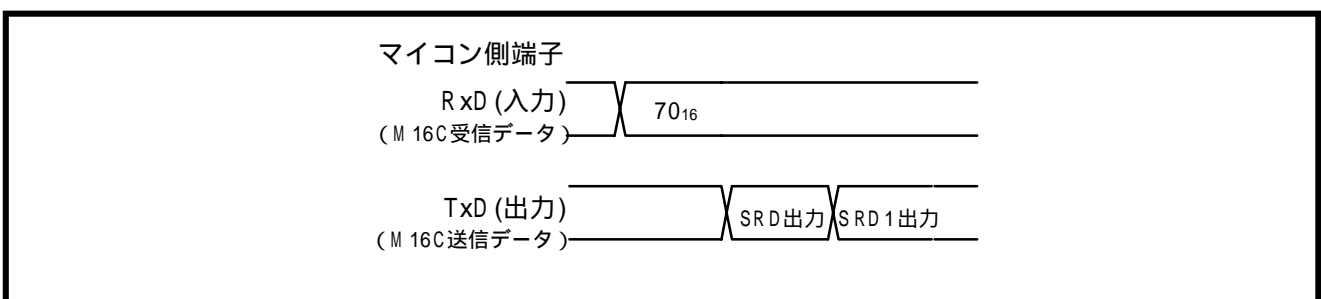


図 3.3.16.リードステータスレジスタコマンド時のタイミング

クリアステータスレジスタ

ステータスレジスタとステータスレジスタ 1 のエラー終了を示すビット(SR3 ~ SR5, SR9)が"1"になった後、これらを"0"にするためのコマンドです。1 バイト目のシリアル転送でコマンドコード"50₁₆"を転送すると、上記のビットを"0"にします。

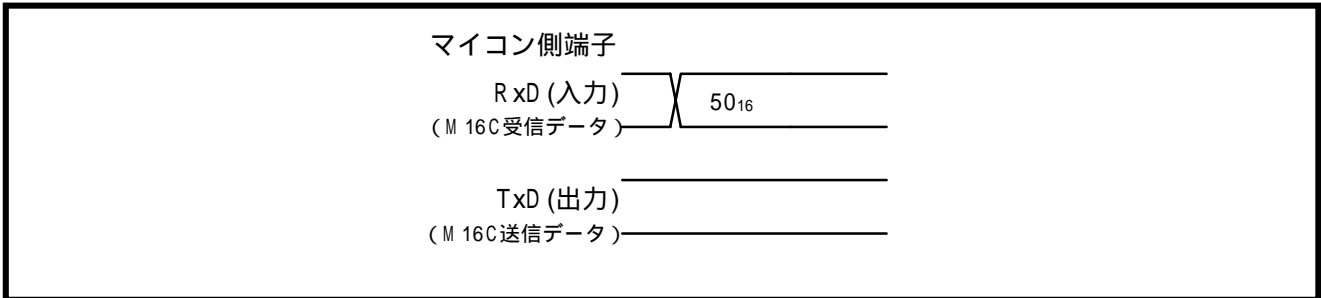


図 3.3.17.クリアステータスレジスタ時のタイミング

リードロックビットステータス

指定したブロックのロックビットの状態を読み出すコマンドです。以下の手順でリードロックビットステータスコマンドを実行してください。A8 ~ A23 のアドレスは、指定するブロックの最上位のアドレスとしてください。ブロックの状態はロック、アンロックの状態があります。

- ロック・・・消去、書き込み禁止
- アンロック・・・消去、書き込み許可

- (1)1 バイト目のシリアル転送でコマンドコード "71₁₆"を転送します。
- (2)指定するブロックの最上位アドレスの A8 ~ A15 を 2 バイト目で、A16 ~ A23 を 3 バイト目で転送します。
- (3)4 バイト目の転送でロックビットデータの内容を出力します。出力したデータの第 6 ビット目が状態を示します。"1"のときブロックアンロック、"0"のときブロックロックを示します。

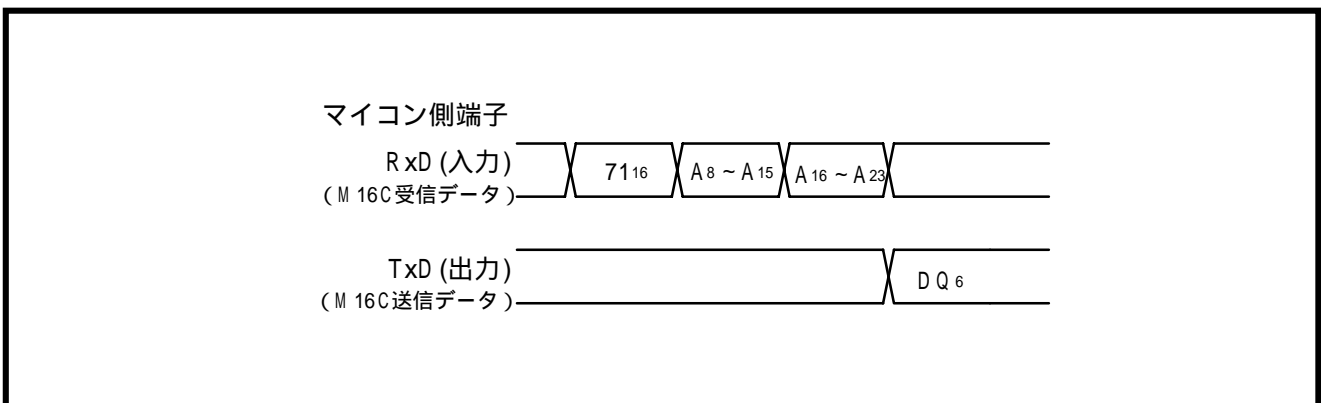


図 3.3.18.リードロックビットステータス時のタイミング

ロックビットプログラム

指定したブロックのロックビットに"0"（ロック状態）を書き込むコマンドです。以下の手順でロックビットプログラムコマンドを実行してください。A8～A23のアドレスは、指定するブロックの最上位のアドレスとしてください。

- (1) 1バイト目のシリアル転送でコマンドコード"77₁₆"を転送します。
- (2) 指定するブロックの最上位アドレスのA8～A15を2バイト目で、A16～A23を3バイト目で転送します。
- (3) 4バイト目の転送で確認コマンドコード"D0₁₆"を転送すると、指定ブロックのロックビットに"0"が書き込まれます。

ロックビットの状態はリードロックビットステータスコマンドで読み出すことができます。

なお、ロックビットは、フラッシュメモリのライトプロテクトピンに"L"を入力すると有効に、"H"を入力すると無効になります。（M5M29GB/T160BVP、M5M29GB/T320BVPのデータシートを参照）ロックビットを"1"（アンロック状態）に戻すには、フラッシュメモリのライトプロテクトピンを"H"にして、ブロックイレーズまたはイレーズ全アンロックブロックコマンドを実行してください。

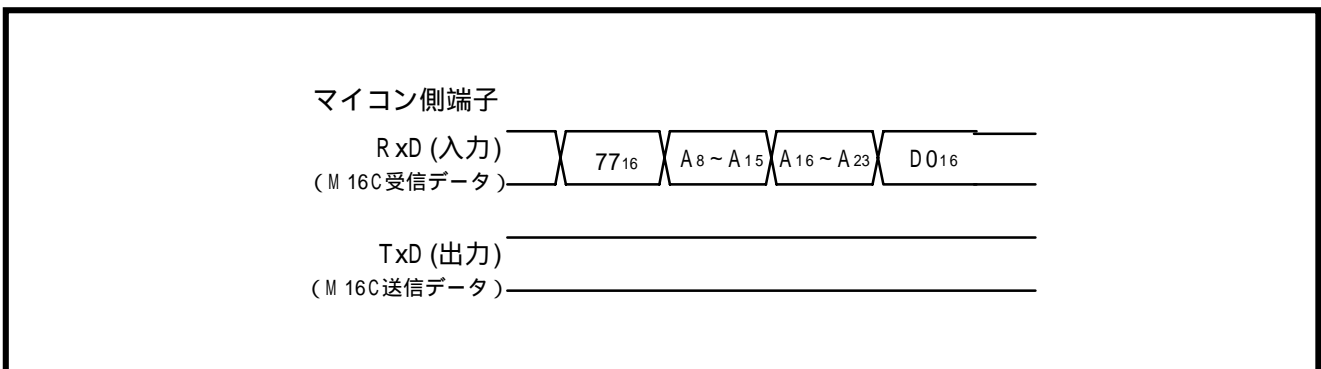


図 3.3.19.ロックビットプログラム時のタイミング

リードチェックデータ

ページプログラムコマンドでシリアルライターが送信した書き込みデータに対し、マイコンが正しく受信したことを確認するためのチェックデータを読み出すコマンドです。読み出されるチェックデータは 2 バイトで、このデータを読み出した後、チェックデータは"0000₁₆"になります。以下の手順で、リードチェックデータコマンドを実行してください。

表 3.3.4.チェックデータ方式

チェックデータ方式	計算方法
CRC 演算	M 16C のCRC 演算回路を用いて、CRC コードを算出する。

- (1)1 バイト目のシリアル転送でコマンドコード"FD₁₆"を転送します。
- (2)2 バイト目の転送でチェックデータ(下位)、3 バイト目の転送でチェックデータ(上位)を出力します。

このリードチェックデータコマンドを使用する場合、まず最初にこのコマンドを実行し、チェックデータを"0000₁₆"にしてください。次にページプログラムコマンドを必要回数実行してください。その後、再びリードチェックデータコマンドを実行すると、この間に実行したページプログラムコマンドで送信した書き込みデータ全てのチェックデータが読み出せます。

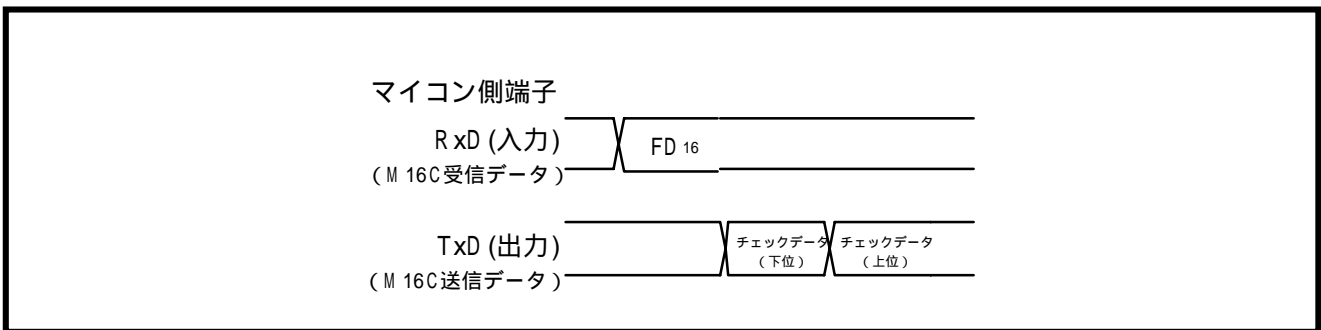


図 3.3.20.リードチェックデータコマンド時のタイミング

3.4 ブートローダモード 1 使用例

ブートローダモード 1 では、MAEC 製(*1)PC カード型フラッシュメモリプログラマまたは、株式会社サニー技研製 Muluti Flash Write(以下 MFW- 1 を称す)を使用して書き換えプログラムをダウンロードすることができます。ここでは、MFW- 1 を使用する場合の書き換えプログラムについて説明します。表 3.4.1 に MFW- 1 を使用する場合のコマンドを示します。また、図 3.4.1 に MFW- 1 使用時の書き換えサンプルプログラムフローチャートを示します。

プログラム全体については、「[3.8 プログラムリスト](#)」を参照してください。

*1 MAEC=三菱電機セミコンダクタ・アプリケーション・エンジニアリング株式会社の略称

表 3.4.1.MFW- 1 を使用する場合のコマンド

	制御コマンド名	1 バイト目 の転送	2 バイト目	3 バイト目	4 バイト目	5 バイト目	6 バイト目	~
1	ヘッダリード	FA ₁₆	アドレス (中位)	アドレス (上位)	データ出力	データ出力	データ出力	~ 259 バイト目 データ出力
2	ヘッダプログラム	41 ₁₆	アドレス (中位)	アドレス (上位)	データ入力	データ入力	データ入力	~ 259 バイト目 データ入力
3	ブロックイレース	20 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
4	イレース全アンロックロック	A7 ₁₆	DO ₁₆					
5	リードステータスレジスタ	70 ₁₆	SRD 出力	SRD1 出力				
6	クリアステータスレジスタ	50 ₁₆						
7	リードロックビットステータス	71 ₁₆	アドレス (中位)	アドレス (上位)	ロックビット データ出力			
8	ロックビットプログラム	77 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
9	リードチェックデータ	FD ₁₆	データ出力 (下位)	データ出力 (上位)				
10	ダウンロード	FA ₁₆	サイズ (下位)	サイズ (上位)	チェックサム	データ入力	~ 必要回数	
11	ダウンロード結果出力	FA ₁₆	データ出力					
12	バージョン情報出力	FB ₁₆	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	~ 9 バイト目 バージョンデータ 出力

- ・ 網掛けは、マイコン シリアルライターへの転送。
それ以外は、シリアルライター マイコンへの転送。
- ・ SRD はステータスレジスタデータ。SRD1 はステータスレジスタ 1 データ。
- ・ コマンド No.9 は、MFW- 1 では未使用のコマンド。

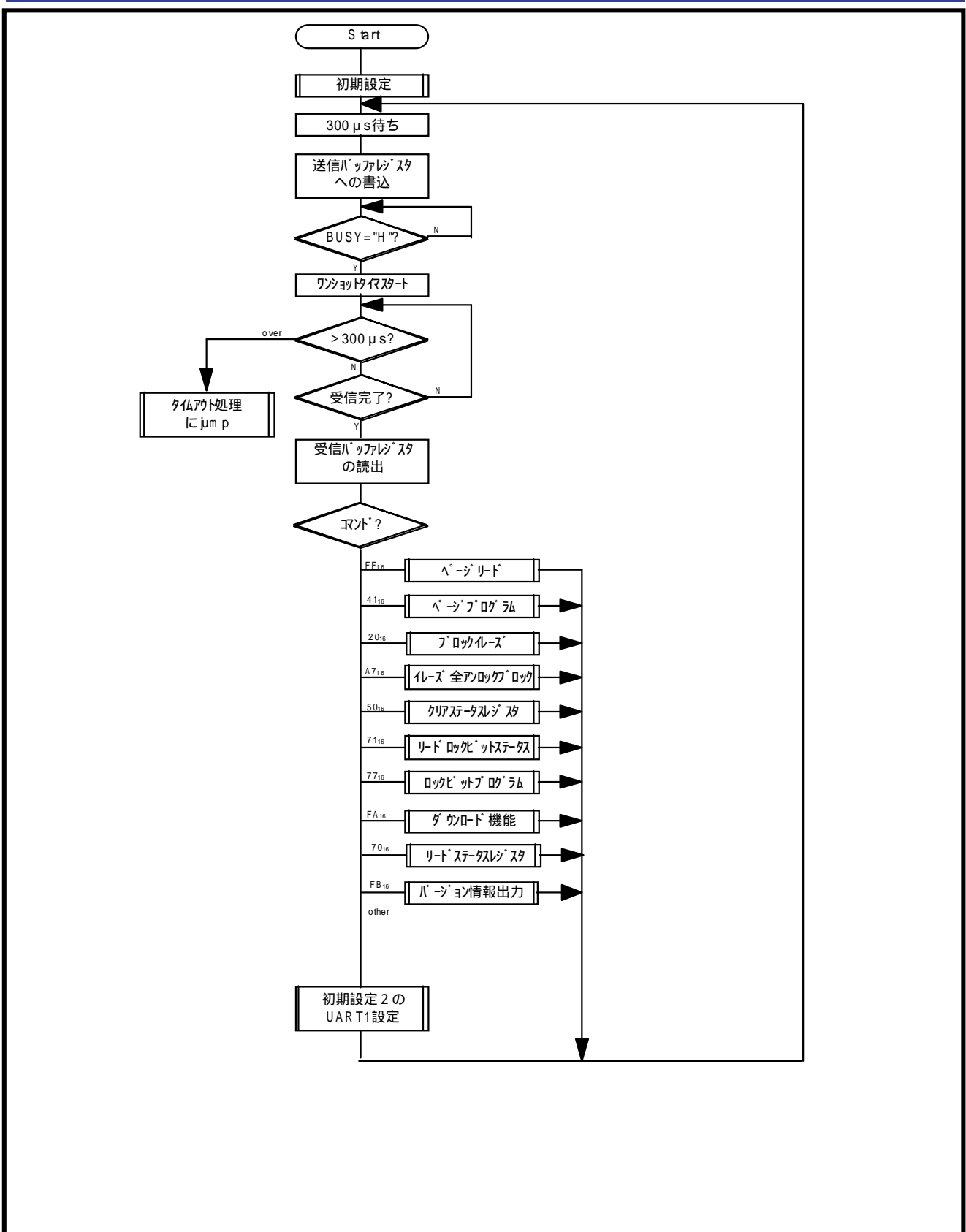


図 3.4.1.MFW- 1 使用時の書き換えサンプルプログラムフローチャート

```

;+++++
;+      Include file                                +
;+++++
        .list          off
        .include      sfr800.inc                    ; SFR header include
        .include      bl80.inc                      ; Bootloader definition include
        .list          on
;
;
;+++++
;+      Version table                                +
;+++++
;
        .section      rom,code
        .org          OFFE000h                     ; Download address
        .byte         'VER.1.01'                  ; Version information
;
;
=====
;+      Boot program start                            +
=====
Program_start:
;-----
;+      Initialize_1                                +
;-----
        ldc          #lstack,ISP                   ; stack pointer set
;
;-----
;+      Processor mode register                      +
;+      & System clock control register            +
;-----
CPU_set:
mov.b   #3,prcr                ; Protect off
mov.w   #00000011b,pm0         ; wait off, micro processor mode
mov.b   #02h,mcd               ; f2
mov.b   #20h,cm1
mov.b   #08h,cm0
mov.b   #00001111b,ds         ; data bus width 16bit
mov.b   #10101010b,wcr       ; all 2 wait
mov.b   #0,prcr              ; Protect on
    
```

) (1-1)

) (1-2)

) (2)

) (3)

図 3.4.2.初期設定

(1)ファイルインクルード、書き換えプログラム先頭アドレスおよびバージョン情報の設定

(1-1)定義ファイルのインクルード

M16C/80 グループの SFR 定義ファイル(sfr80.inc)とサンプルプログラムで使用する RAM データ宣言およびシンボル定義のためのファイル(bl80.inc)をインクルードしています。

(1-2)書き換えプログラム先頭アドレスおよびバージョン情報の設定

ブートローダでダウンロードする書き換えプログラムは、600₁₆ 番地から配置します。プログラムサイズについては、「[3.6.4 メモリマップ 4 \(MFW-1 使用時\)](#)」を参照してください。

ブートローダでダウンロードする書き換えプログラムでは、600₁₆ 番地を先頭にして 8 バイトのバージョン情報を設定する必要があります。バージョン情報をユーザで使わない場合でも、必ずバージョン情報を設定してください。

(2)スタックポインタの設定

書き換えプログラムでは、まずスタックポインタ(ISP)の設定を行います。設定値は、書き換えプログラムと重ならない内部 RAM エリアに設定してください。(書き換えプログラムは、600₁₆ 番地を先頭に転送されます。「[3.6 メモリマップ](#)」参照。)

(3)CPU 関連レジスタの設定

PM0 の変更：

書き換えプログラムがダウンロードされた時点では、CPU はシングルチップモードで動作していますので、マイクロプロセッサモードに変更します。

MCD および WCR の設定：

外付けフラッシュメモリとのアクセスタイミングに合わせてメインクロックの分周およびソフトウェアウエイトを設定します。(M16C/80 グループのアクセスタイミングについては、M16C/80 グループデータシートを参照してください。)

DS の設定：

外付けフラッシュメモリの接続状態に合ったデータバス幅を設定してください。

```

;+++++
;+      Main flow - clock synchronous serial I/O mode -      +
;+++++
Main:
    jsr          Initialize_2          ; clock synchronous serial I/O mode ) (4)
;
Loop_main:
    bset        ta0os
    mov.b       #0,ta0ic
Loop_main1:
    btst        ir_ta0ic              ; 300 usec ?
    jz          Loop_main1
    mov.b       #0,ta0ic
    mov.b       #0ffh,r1l             ; #ffh --> r1l (transfer dummy data)
    mov.b       r1l,u1tb             ; transfer data --> transfer buffer

    bclr        busy_d               ; busy input
?:
    btst        busy                 ; Reception start?
    jz          ?-

    bset        ta0os                ; 300 usec timer start
?:
    btst        ir_ta0ic              ; 300 usec ?
    jc          Time_out             ; jump Time_out at time out
    btst        ri_u1c1              ; receive complete ?
    jz          ?-
    mov.w       u1rb,r0              ; receive data --> r0
;
Command_check:
    cmp.b       #0ffh,r0l             ; Read          (ffh)
    jeq         Read
    cmp.b       #041h,r0l             ; Program       (41h)
    jeq         Program
    cmp.b       #020h,r0l             ; Erase         (20h)
    jeq         Erase
    cmp.b       #0a7h,r0l             ; All erase (a7h)
    jeq         All_erase
    cmp.b       #050h,r0l             ; Clear SRD (50h)
    jeq         Clear_SRD
    cmp.b       #071h,r0l             ; Read LBS     (71h)
    jeq         Read_LB
    cmp.b       #077h,r0l             ; LB program(77h)
    jeq         Program_LB
    cmp.b       #0fah,r0l             ; Download     (fah)
    jeq         Download

    cmp.b       #070h,r0l             ; Read SRD     (70h)
    jeq         Read_SRD
    cmp.b       #0fbh,r0l             ; Version out  (fbh)
    jeq         Ver_output
Command_err:
    jsr          Initialize_21        ; command error,UART1 reset
    jmp         Loop_main            ; command error,jump Loop_main
    
```

図 3.4.3.メインルーチン

(4)通信初期設定処理

通信初期設定処理部にサブルーチンジャンプし、シリアル通信の初期設定を行います。

(5)コマンド受信、判定処理

まず、コマンド受信準備（300 μ sec（20MHz 動作時）待った後、BUSY 端子(*1)が"H"になるまで待つ）をします。BUSY 端子が"H"になったら、コマンド受信を行います。コマンド受信時、タイムアウトが発生した場合は、タイムアウトエラー処理に分岐します。タイムアウトせずにコマンドデータを1バイト受信した場合は、続けてコマンドの判定を行います。コマンド判定により、各コマンド処理に分岐します。

*1 BUSY 端子は、受信準備が完了すれば"L"となり、受信動作を開始すれば"H"を出力します。

```

;-----
;+      Read      +
;-----
Read:
    mov.w  #0,r3          ; receive number
    mov.b  #0,addr_l     ; addr_l = 0
;
Read_loop:
    mov.b  r1l,u1tb      ; data transfer
    bset   ta0os         ; ta0 start
?:
    btst  ir_ta0ic       ; time out error ?
    jc    Time_out      ; jump Time_out at time out
    btst  ri_u1c1        ; receive complete ?
    jnc   ?-             ;
    mov.w  u1rb,r0       ; receive data read --> r0
;
    add.w  #1,r3         ; r3 +1 increment
    cmp.w  #2,r3         ; r3 = 2 ?
    jgtu          Read_data ; jump Read_data at r3>3
    mov.w  r3,a0         ; r3 --> a0
    mov.b  r0l,addr_l[a0] ; Store address
    cmp.w  #2,r3         ; r3 = 2 ?
    jltu          Read_loop ; jump Read_loop at r3<2
;
    mov.w  addr_l,a0     ; addr_l,m --> a0
    mov.b  addr_h,a1     ; addr_h --> a1
    sha.l  #16,a1        ;
    add.l  a0,a1         ; get read address
;
Read_data:
;
; Flash memory read & store to r1l
;
    add.l  #1,a1         ; address increment
    cmp.w  #258,r3      ; r3 = 258 ?
    jne          Read_loop ; jump Read_loop at r<260
;
    jmp    Loop_main    ; jump Loop_main
;

```

(6_1)

図 3.4.4.Read コマンド処理

(6-1)Read コマンド(FF16)処理

MFV-1 の Blank,Read,Verify,Program/Verify ボタンが押されると送信されてくるコマンドです。

- ・ アドレス情報 (2 バイト目、3 バイト目) を受信します。
- ・ 外付けフラッシュメモリから 1 バイトデータを読み出し、r1l へ書き込みます。(ユーザで追加してください。)
- ・ MFV-1 に読み出したデータを送信します。
- ・ データの読み出し、書き込み、送信を 256 回繰り返します。

```

;-----
;+      Program      +
;-----
Program:
    mov.w  #0,r3          ; receive number
    mov.b  #0,addr_l     ; addr_l = 0
Program_bop_1:
    mov.b  r1l,u1tb      ; data transfer
    bset   ta0os         ; ta0 start
    mov.b  #0,ta0ic      ; clear time out
?:
    btst   ir_ta0ic      ; time out error ?
    jc     Time_out      ; jump Time_out at time out
    btst   ri_u1c1       ; receive complete ?
    jnc    ?-
    mov.w  u1rb,r0        ; receive data read --> r0
    add.w  #1,r3          ; r3 +1 increment
    mov.w  r3,a0          ; r3 --> a0
    mov.b  r0l,addr_l[a0] ; Store address
    cmp.w  #258,r3        ; r3 = 258 ?
    jltu   Program_bop_1 ; jump Program_loop_1 at r3<258
;
    mov.w  #0,r3          ; writing number (r3=0)
Program_bop_2:
    mov.b  addr_h,a1      ; addr_h --> a1
    sha.l  #16,a1
    mov.w  r3,a0          ; r3 --> a0
    mov.w  data[a0],r1    ; data --> r1
    mov.w  addr_l,a0      ; addr_l,m --> a0
    add.l  a0,a1
;
; data write
;
    add.w  #2,addr_l      ; address +2 increment
    add.w  #2,r3          ; writing number +2 increment
    cmp.w  #255,r3        ; r3 = 255 ?
    jltu   Program_bop_2 ; jump Program_loop_2 at r3<255
Program_end:
    jmp    Loop_main     ; jump Loop_main
;

```

(6_2)

図 3.4.5.Program マンド処理

(6-2)Program コマンド(4116)処理

MFV-1 の Program,Program/Verify ボタンが押されると送信されてくるコマンドです。

- ・アドレス情報(2 バイト目、 3 バイト目)を受信し、続けてプログラムデータ(256 バイト)を受信します。
- ・外付けフラッシュメモリへ 256 バイトデータを書き込みます。(ユーザで追加してください。)
サンプルプログラムでは、ワード単位でデータを書き込む場合を想定してアドレス(addr_l)および書き込み回数(r3)の加算値を"+2"としています。


```

;-----
;+      Block erase      +
;-----
Erase:
    mov.w  #1,r3          ; receive number (r3=1)
Erase_loop:
    mov.b  r1l,u1tb      ; data transfer
    bset   ta0os         ; ta0 start
    mov.b  #0,ta0ic      ; clear time out
?:
    btst   ir_ta0ic     ; time out error ?
    jc     Time_out     ; jump Time_out at time out
    btst   ri_u1c1      ; receive complete ?
    jnc    ?-           ;
    mov.w  u1rb,r0       ; receive data read --> r0
    mov.w  r3,a0         ; r3 --> a0
    mov.b  r0l,addr_[a0] ; Store address
    add.w  #1,r3         ; r3+1 increment
    cmp.w  #4,r3         ; r3=4 ?
    jltu   Erase_loop   ; jump Erase_loop at r3<4
;
    cmp.b  #0d0h,data   ; Confirm command check
    jne    Erase_end    ; jump Erase_end at Confirm command error
;
; Block Erase
;
Erase_end:
    jmp    Loop_main    ; jump Loop_main
;

```

(6_3)

図 3.4.6.BlockErase コマンド処理

(6_3)BlockErase コマンド(2016)処理

MFW-1 の Erase,Program ボタンが押されると送信されてくるコマンドです。消去エリアが全ブロックでない場合に送信されます。消去エリアが全ブロックの場合は、「(6_4)」の ALLErase コマンド(A716)が送信されてきます。

- ・アドレス情報(2バイト目、3バイト目)を受信し、続けて確認コマンド(4バイト目)を受信します。
- ・4バイト目で受信した確認コマンド(D016)のチェックを行います。
- ・外付けフラッシュメモリに対し、指定されたブロックのデータ消去を行います。(ユーザで追加してください。)

```

;-----
;+   All erase ( unlock block )   +
;-----
All_erase:
    mov.b  r1l,u1tb                ; data transfer
    bset   ta0os                    ; ta0 start
    mov.b  #0,ta0ic                ; clear time out
?:
    btst   ir_ta0ic                ; time out error ?
    jc     Time_out                ; jump Time_out at time out
    btst   ri_u1c1                 ; receive complete ?
    jnc    ?-                      ;
    mov.w  u1rb,r0                 ; receive data read --> r0
;
    cmp.b  #0d0h,r0l              ; Confirm command check
    jne    All_erase_end          ; jump All_erase_end at Confirm command error
;
; All Erase
;
All_erase_end:
    jmp    Loop_main              ; jump Loop_main
;

```

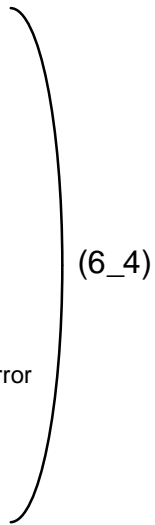


図 3.4.7.All Erase マンド処理

(6_4)All Erase コマンド(A716)処理

MFW- 1 の Erase,Program ボタンが押されると送信されてくるコマンドです。消去エリアが全ブロックの場合に送信されます。消去エリアが全ブロックでない場合は、「(6_3)」の BlockErase コマンド(2016)が送信されてきます。

- ・確認コマンド (2 バイト目) を受信します。
- ・2 バイト目で受信した確認コマンド(D016)のチェックを行います。
- ・外付けフラッシュメモリに対し、全ブロックデータ消去を行います。(ユーザで追加してください。)

```

;-----
;+      Read SRD          +
;-----
Read_SRD:
    mov.w  #0,r3          ; receive number (r3=0)
    mov.b  #80h,r1l      ; dummy SRD set
;
Read_SRD_loop:
    mov.b  r1l,u1tb      ; data transfer
    bset   ta0os         ; ta0 start
    mov.b  #0,ta0ic      ; clear time out
?:
    btst   ir_ta0ic      ; time out error ?
    jc     Time_out      ; jump Time_out at time out
    btst   ri_u1c1       ; receive complete ?
    jnc    ?-            ;
    mov.w  u1rb,r0       ; receive data read --> r0
    mov.b  SRD1,r1l      ; SRD1 data --> r1l
    add.w  #1,r3         ; r3 +1 increment
    cmp.w  #2,r3         ; r3=2 ?
    jltu   Read_SRD_loop ; jump Read_SRD_loop at r3<2
;
    jmp    Loop_main     ; jump Loop_main
;

```

(6_5)

図 3.4.8.リードステータスコマンド処理

(6-5)リードステータスコマンド(70₁₆)処理

MF_W-1 との通信制御で使用されるコマンドです。

- ・ SRD データとして"80₁₆" (2 バイト目) を送信します。
- ・ SRD1 データ (3 バイト目) を送信します。

```

;-----
;+      Clear SRD        +
;-----
Clear_SRD:
;
    and.b  #10011100b,SRD1 ; SRD1 clear
;
    jmp    Loop_main       ; jump Loop_main
;

```

(6_6)

図 3.4.9.クリアステータスコマンド処理

(6-6)クリアステータスコマンド(50₁₆)処理

MF_W-1 との通信制御で使用されるコマンドです。

- ・ SRD1 データをクリアします。

```

;-----
;+      Read Lock Bit      +
;-----
Read_LB:
    mov.w  #1,r3           ; receive number (r3=1)
Read_LB_loop:
    mov.b  r1l,u1tb       ; data transfer
    bset   ta0os          ; ta0 start
    mov.b  #0,ta0ic       ; clear time out
?:
    btst   ir_ta0ic       ; time out error ?
    jc     Time_out       ; jump Time_out at time out
    btst   ri_u1c1        ; receive complete ?
    jnc    ?-             ;
    mov.w  u1rb,r0        ; receive data read --> r0
    mov.w  r3,a0          ; r3 --> a0
    mov.b  r0l,addr_[a0] ; Store address
    add.w  #1,r3          ; r3 +1 increment
    cmp.w  #3,r3         ; r3=3 ?
    jltu   Read_LB_loop   ; jump Read_LB_loop at r3<3
    jgtu   Read_LB_end    ; jump Read_LB_end at r3>3
;
    mov.w  #00aah,r1      ; dummy read LB status set
;
    jmp    Read_LB_loop   ; jump Read_LB_loop
;
Read_LB_end:
    jmp    Loop_main     ; jump Loop_main
;

```

(6_7)

図 3.4.10.リードロックビットコマンド処理

(6-7)リードロックビットコマンド(7116)処理

ユーザが MFW-1 上で「No Change」を選択した状態で、MFW-1 の Erase,Program ボタンが押されると送信されてくるコマンドです。

- ・アドレス情報(2 バイト目、3 バイト目)を受信します。
- ・ロックビットデータとして"AA16"(4 バイト目)を送信します。

```

;-----
;+      Program Lock Bit      +
;-----
Program_LB:
    mov.w  #1,r3                ; receive number (r3=1)
Program_LB_loop:
    mov.b  r11,u1tb             ; data transfer
    bset   ta0os                ; ta0 start
    mov.b  #0,ta0ic            ; clear time out
?:
    btst   ir_ta0ic            ; time out error ?
    jc     Time_out            ; jump Time_out at time out
    btst   ri_u1c1             ; receive complete ?
    jnc    ?-
    mov.w  u1rb,r0              ; receive data read --> r0
    mov.w  r3,a0                ; r3 --> a0
    mov.b  r0l,addr_l[a0]      ; Store address
    add.w  #1,r3                ; r3 +1 increment
    cmp.w  #4,r3                ; r3=4 ?
    jltu   Program_LB_loop     ; jump Program_LB_loop at r3<4
    cmp.b  #0d0h,data          ; Confirm command check
    jne    Program_LB_end      ;
;
Program_LB_end:
    jmp    Loop_main           ; jump Loop_main
;
    
```

(6_8)

図 3.4.11.ロックビットプログラムコマンド処理

(6-8)ロックビットプログラムコマンド(7716)処理

MFW-1 の Erase,Program ボタンが押されると送信されてくるコマンドです。

- ・ アドレス情報(2バイト目、3バイト目)を受信し、続けて確認コマンド(4バイト目)を受信します。
- ・ 確認コマンド(D016)のチェックを行います。

```

;-----
;+      Version output      +
;-----
Ver_output:
    mov.w  #0,a0                ; Version address offset (a0=0)
Ver_output_loop:
    mov.b  ver[a0],u1tb         ; Version data transfer
    bset   ta0os                 ; ta0 start
    mov.b  #0,ta0ic             ; clear time out
?:
    btst  ir_ta0ic              ; time out error ?
    jc    Time_out              ; jump Time_out at time out
    btst  ri_u1c1               ; receive complete ?
    jnc   ?-                    ;
    mov.w  u1rb,r0              ; receive data read --> r0
    add.w  #1,a0                ; a0 +1 increment
    cmp.w  #8,a0                ; a0=8 ?
    jltu  Ver_output_loop       ; jump Ver_output_loop at a0<8
Ver_output_end:
    jmp   Loop_main             ; jump Loop_main
;
    
```

図 3.4.12.バージョン出力コマンド処理

(6-9)バージョン出力コマンド(FB16)処理

MFW-1 との通信制御で使用されるコマンドです。

- ・バージョン情報（2 バイト目 ~ 9 バイト目）を送信します。

```

;-----
;+      Download      +
;-----
Download:
    mov.b  #3,prcr              ; Protect off
    mov.w  #0000h,pm0           ; wait off, single chip mode
    mov.b  #02h,mcd             ; f2
    mov.b  #20h,cm1             ;
    mov.b  #08h,cm0            ;
    mov.b  #0,prcr              ; Protect on

    jmp.a  Download_program     ; jump Download_program
;
    
```

図 3.4.13.Download コマンド処理

(6_10)Download コマンド(FA16)処理

ユーザがダウンロード選択時に、MFW-1 が起動してブートローダと通信を開始する時に送信されてくるコマンドです。

- ・プロセッサモードをシングルチップモードに変更します。
- ・マイコンの内部 ROM 上のブートローダの特定の番地（ダウンロード処理部）にジャンプします。

3.5 ブートローダモード 2 使用例

ブートローダモード 2 では、MAEC 製 M16C Flash Starter を使用して書き換えプログラムをダウンロードすることができます。ここでは、M16C Flash Starter を使用する場合の書き換えプログラムについて説明します。表 3.5.1 に M16C Flash Starter を使用する場合のコマンドを示します。また、図 3.5.1 に M16C Flash Starter 使用時の書き換えサンプルプログラムフローチャートを示します。

プログラム全体については、「[3.8 プログラムリスト](#)」を参照してください。

表 3.5.1.M16C Flash Starter を使用する場合のコマンド

	制御コマンド名	1 バイト目 の転送	2 バイト目	3 バイト目	4 バイト目	5 バイト目	6 バイト目	~
1	ヘジリット	FF ₁₆	アドレス (中位)	アドレス (上位)	データ出力	データ出力	データ出力	~ 259 バイト目 データ出力
2	ヘジプログラム	41 ₁₆	アドレス (中位)	アドレス (上位)	データ入力	データ入力	データ入力	~ 259 バイト目 データ入力
3	プログラクセス	20 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
4	レス全アンロックロック	A7 ₁₆	DO ₁₆					
5	リットステータスレジスタ	70 ₁₆	SRD 出力	SRD1 出力				
6	クリアステータスレジスタ	50 ₁₆						
7	リットロックビットステータス	71 ₁₆	アドレス (中位)	アドレス (上位)	ロックビット データ出力			
8	ロックビットプログラム	77 ₁₆	アドレス (中位)	アドレス (上位)	DO ₁₆			
9	リットチェックデータ	FD ₁₆	データ出力 (下位)	データ出力 (上位)				
10	ダウンロード	FA ₁₆	サイズ (下位)	サイズ (上位)	チェックサム	データ入力	~ 必要回数	
11	ダウンロード結果出力	FA ₁₆	データ出力					
12	バージョン情報出力	FB ₁₆	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	バージョン データ出力	~ 9 バイト目 バージョンデータ 出力
13	ホレート 9600	B0 ₁₆	B0 ₁₆					
14	ホレート 19200	B1 ₁₆	B1 ₁₆					
15	ホレート 38400	B2 ₁₆	B2 ₁₆					
16	ホレート 57600	B3 ₁₆	B3 ₁₆					
17	ホレート 115200	B4 ₁₆	B4 ₁₆					

- ・ 網掛けは、マイコン シリアルライターへの転送。
それ以外は、シリアルライター マイコンへの転送。
- ・ SRD はステータスレジスタデータ。SRD1 はステータスレジスタ 1 データ。
- ・ コマンド No.3、No.7~No.9 は、M16C FlashStarter では未使用のコマンド。

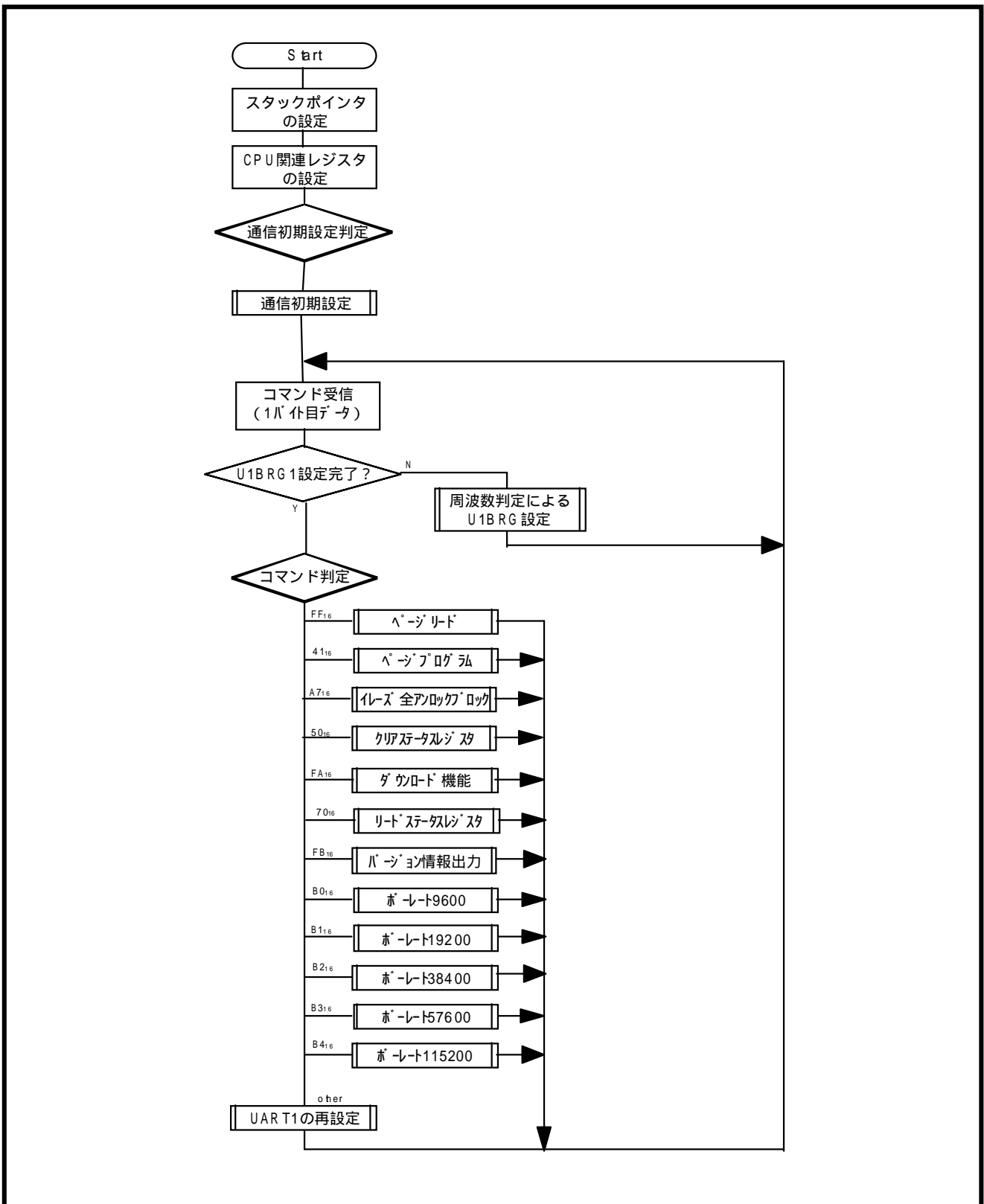


図 3.5.1.M16C Flash Starter 使用時の書き換えサンプルプログラムフローチャート


```

;+++++
;+   Include file                                     +
;+++++
        .list      off
        .include   sfr800.inc           ; SFR header include
        .include   bl80.inc            ; Bootloader definition include
        .list      on
;
;+++++
;+   Version table                                   +
;+++++
;
        .section  rom,code
        .org      0600h                ; Download address
        .byte     'VER.0.01'           ; Version information
;
;=====
;+   Boot program start                             +
;=====
Program_start:
;-----
;+   Initialize_1                                   +
;-----
        ldc       #Istack,ISP          ; stack pointer set
;
;-----
;+   Processor mode register                       +
;+   & System clock control register             +
;-----
CPU_set:
        mov.b    #3,prcr                ; Protect off
        mov.w    #00000011b,pm0         ; wait off, micro processor mode
        mov.b    #02h,mcd               ; f2
        mov.b    #20h,cm1
        mov.b    #08h,cm0
        mov.b    #00001111b,ds         ; data bus width 16bit
        mov.b    #10101010b,wcr        ; all 2 wait
        mov.b    #0,prcr                ; Protect on
    
```

図 3.5.2.初期設定

(1)インクルードファイル、書き換えプログラム先頭アドレスおよびバージョン情報の設定

(1-1)定義ファイルのインクルード

M16C/80 グループの SFR 定義ファイル(sfr80.inc)とサンプルプログラムで使用する RAM データ宣言およびシンボル定義のためのファイル(bl80.inc)をインクルードしています。

(1-2)書き換えプログラム先頭アドレスおよびバージョン情報の設定

ブートローダでダウンロードする書き換えプログラムは、600₁₆ 番地から配置します。プログラムサイズについては、「[3.6.3 メモリマップ 3 \(M16C Flash Starter 使用時\)](#)」を参照してください。

ブートローダでダウンロードする書き換えプログラムでは、600₁₆ 番地を先頭にして 8 バイトのバージョン情報を設定する必要があります。バージョン情報をユーザで使用しない場合でも、必ずバージョン情報を設定してください。

(2)スタックポインタの設定

書き換えプログラムでは、まずスタックポインタ(ISP)の設定を行います。設定値は、書き換えプログラムと重ならない内部 RAM エリアに設定してください。(書き換えプログラムは、600₁₆ 番地を先頭に転送されます。「[3.6 メモリマップ](#)」参照。)

(3)CPU 関連レジスタの設定

PM0 の変更：

書き換えプログラムがダウンロードされた時点では、CPU はシングルチップモードで動作していますので、マイクロプロセッサモードに変更します。

MCD および WCR の設定：

外付けフラッシュメモリとのアクセスタイミングに合わせてメインクロックの分周およびソフトウェアウェイトを設定します。(M16C/80 グループのアクセスタイミングについては、M16C/80 グループデータシートを参照してください。)

DS の設定：

外付けフラッシュメモリの接続状態に合ったデータバス幅を設定してください。

```

;+++++
;+      Main flow - UART mode -      +
;+++++
U_Main:
    btst  updata_f          ;
    bmltu updata_f          ; if "C"flag is "0", updata_f set "1"
    jc    U_Main1          ; if "C"flag is "1", initialize execute(jump U_Main1)
    jmp   U_Loop_main      ;
) (4)

U_Main1:
    bclr  updata_f          ;
    bclr  freq_set1        ; freq set flag clear
    bclr  freq_set2        ;
    mov.b #01111111b,data   ; Initialize Baud rate
    jsr   Initialize_3     ; UART mode Initialize
    mov.b #01000000b,r1l    ; counter1,2 reset
    mov.b #10000000b,r1h    ;
    mov.w u1rb,r0          ; receive data --> r0
) (5)

;
U_Loop_main:
    bclr  te_u1c1          ; Transmission disabled
    bset  re_u1c1          ; Reception enabled
? :
    btst  ri_u1c1          ; receive complete ?
    jz    ?-               ;
    mov.w u1rb,r0          ; receive data --> r0
    btst  freq_set2        ;
    jz    U_Freq_check    ;
;
U_Command_check:
    cmp.b #0ffh,r0l        ; Read      (ffh)
    jeq   U_Read           ;
    cmp.b #041h,r0l        ; Program   (41h)
    jeq   U_Program        ;
    cmp.b #020h,r0l        ; Erase     (20h)
    jeq   U_Erase          ;
    cmp.b #0a7h,r0l        ; All erase (a7h)
    jeq   U_All_erase      ;
    cmp.b #050h,r0l        ; Clear SRD (50h)
    jeq   U_Clear_SRD      ;
    cmp.b #071h,r0l        ; Read LBS  (71h)
    jeq   U_Read_LB        ;
    cmp.b #077h,r0l        ; LB program (77h)
    jeq   U_Program_LB     ;
    cmp.b #0fah,r0l        ; Download  (fah)
    jeq   U_Download       ;
    cmp.b #0fdh,r0l        ; Read check (fdh)
    jeq   U_Read_check     ;
) (6)

    cmp.b #070h,r0l        ; Read SRD  (70h)
    jeq   U_Read_SRD       ;
    cmp.b #0fbh,r0l        ; Version out (fbh)
    jeq   U_Ver_output     ;
    cmp.b #0b0h,r0l        ; Baud rate 9600bps (b0h)
    jeq   U_BPS_B0         ;
    cmp.b #0b1h,r0l        ; Baud rate 19200bps (b1h)
    jeq   U_BPS_B1         ;
    cmp.b #0b2h,r0l        ; Baud rate 38400bps (b2h)
    jeq   U_BPS_B2         ;
    cmp.b #0b3h,r0l        ; Baud rate 57600bps (b3h)
    jeq   U_BPS_B3         ;
    cmp.b #0b4h,r0l        ; Baud rate 115200bps (b4h)
    jeq   U_BPS_B4         ;
    jsr   U_Initialize_31  ; command error, UART mode Initialize
    jmp   U_Loop_main      ; jump U_Loop_main
    
```

図 3.5.3.メインルーチン

(4)通信初期設定判定処理

書き換えプログラムのダウンロード完了後にリセットが実行された場合、コマンド判定処理部へ分岐しないで、通信初期設定処理部へ分岐します。

(5)通信初期設定処理

書き換えプログラムのダウンロード完了後、リセットされた場合に実行されます。

(6)コマンド判定処理

M16C Flash Starter からコマンドデータを1バイト受信し、転送速度レジスタ設定完了判定もしくは、コマンドの判定を行います。コマンド判定により、各コマンド処理に分岐します。

```

;-----
;+      Read - UART mode -      +
;-----
U_Read:
    mov.w #0,r3                ; receive number
    mov.b #0,addr_l           ; addr_l =0
?:
    btst ri_u1c1              ; receive complete ?
    jnc      ?-
;
    mov.w u1rb,r0             ; receive data read --> r0
    add.w #1,r3               ; r3 +1 increment
    mov.w r3,a0              ; r3 --> a0
    mov.b r0l,addr_l[a0]     ; Store address
    cmp.w #2,r3              ; r3 = 2 ?
    jltu      ?-             ; jump Read_loop at r3<2
;
    mov.w addr_l,a0          ; addr_l,m --> a0
    mov.b addr_h,a1          ; addr_h --> a1
    sha.l #16,a1             ;
    add.l a0,a1              ; a1 is address-data
;
    bclr re_u1c1             ; Reception disabled
    bset te_u1c1             ; Transmission enabled
U_Read_data:
    cmp.w #258,r3            ; r3 = 258 ?
    jz      U_Read_end
;
; Flash memory read
;
    mov.b r1l,u1tb           ; r1l --> transmit buffer register
?:
    btst ti_u1c1             ; transmit buffer empty ?
    jnc      ?-
    add.l #1,a1              ; address increment
    add.w #1,r3              ; counter increment
    jmp      U_Read_data     ; jump U_Read_data
;
U_Read_end:
    btst txept_u1c0          ; Transmit register empty ?
    jnc      U_Read_end
    jmp      U_Loop_main
;

```

(7-1)

図 3.5.4.Read コマンド処理

(7-1)Read コマンド(FF16)処理

M16C Flash Starter の Blank,Read(B.P.R.,E.P.R.)ボタンが押されると送信されてくるコマンドです。

- ・ アドレス情報 (2 バイト目、 3 バイト目) を受信します。
- ・ 外付けフラッシュメモリから 1 バイトデータを読み出し、 r1l へ書き込みます。(ユーザで追加してください。)
- ・ M16C Flash Starter に読み出したデータを送信します。
- ・ データの読み出し、書き込み、送信を 256 回繰り返します。

```

;-----
;+      Program - UART mode -      +
;-----
U_Program:
    mov.w  #0,r3                ; receive number
    mov.b  #0,addr_l           ; addr_l = 0
    mov.w  sum,crcd            ; for Read check command
U_Program_loop:
    btst   ri_u1c1              ; receive complete ?
    jnc    U_Program_loop
    mov.w  u1rb,r0              ; receive data read --> r0
    add.w  #1,r3                ; r3 + 1 increment
    mov.w  r3,a0                ; r3 --> a0
    mov.b  r0l,addr_l[a0]       ; Store address
    cmp.w  #258,r3              ; r3 = 258 ?
    jltu   U_Program_loop      ; jump U_Program_loop at r3<258
;
    mov.w  #0,r3                ; writing number (r3=0)
U_Program_loop_2:
    mov.b  addr_h,a1            ; addr_h --> a1
    sha.l  #16,a1
    mov.w  r3,a0                ; r3 --> a0
    mov.w  data[a0],r1          ; data --> r1
    mov.w  addr_l,a0            ; addr_l,m --> a0
    add.l  a0,a1
;
; data write
;
    mov.b  r1l,crcin            ; for Read check command
    mov.b  r1h,crcin
;
    add.w  #2,addr_l            ; address +2 increment
    add.w  #2,r3                ; writing number +2 increment
    cmp.w  #255,r3              ; r3 = 255 ?
    jltu   U_Program_loop_2    ; jump U_Program_loop_2 at r3<255
U_Program_end:
    mov.w  crcd,sum             ; for Read check command
    jmp    U_Loop_main          ; jump U_Loop_main
;

```

(7-2)

図 3.5.5.Program コマンド

(7-2)Program コマンド(4116)処理

- M16C FlashStarter の Program(B.P.R.,E.P.R.)ボタンが押されると送信されてくるコマンドです。
- ・ アドレス情報 (2 バイト目、 3 バイト目) を受信し、続けてプログラムデータ(256 バイト)を受信します。
 - ・ 外付けフラッシュメモリへ 256 バイトデータを書き込みます。(ユーザで追加してください。)

サンプルプログラムでは、ワード単位でデータを書き込む場合を想定してアドレス(addr_l)および書き込み回数(r3)の加算値を"+2"としています。

```

;-----
;+           All erase ( unlock block ) - UART mode - +
;-----
U_All_erase:
    btst    ri_u1c1                ; receive complete ?
    jnc     U_All_erase
;
    mov.w   u1rb,r0                ; receive data read --> r0
    cmp.b   #0d0h,r0l              ; Confirm command check
    jne     U_All_erase_end ; jump U_All_erase_end at Confirm command error
;
; All erase
;
U_All_erase_end:
    jmp     U_Loop_main            ; jump U_Loop_main
;
    
```

(7-3)

図 3.5.6.All Erase コマンド処理

(7-3)All Erase コマンド(A716)処理

M16C Flash Starter の Erase(E.P.R.)ボタンが押されると送信されてくるコマンドです。

- ・ 確認コマンド (2 バイト目) を受信します。
- ・ 2 バイト目で受信した確認コマンド(D016)のチェックを行います。
- ・ 外付けフラッシュメモリに対し、全ブロックデータ消去を行います。(ユーザで追加します。)

```

;-----
;+      Read SRD - UART mode      +
;-----
U_Read_SRD:
    bclr    re_u1c1                ; Reception disabled
;
    mov.w  #0,r3                  ; receive number (r3=0)
;
    mov.b  #80h,r1l               ; dummy SRD set
    bset   te_u1c1                ; Transmission enabled
U_Read_SRD_loop:
    mov.b  r1l,u1tb               ; r1l --> transmit buffer register
?:
    btst   ti_u1c1                ; transmit buffer empty ?
    jnc    ?-                     ;
    mov.b  SRD1,r1l               ; SRD1 data --> r1l
    add.w  #1,r3                  ; r3 +1 increment
    cmp.w  #2,r3                  ; r3=2 ?
    jltu          U_Read_SRD_loop ; jump U_Read_SRD_loop at r3<2
U_Read_SRD_end:
    btst   txept_u1c0             ; Transmit register empty ?
    jnc    U_Read_SRD_end
;
    jmp    U_Loop_main           ; jump U_Loop_main
;

```

図 3.5.7.リードステータスコマンド処理

(7-4)リードステータスコマンド(70₁₆)処理

M16C Flash Starter との通信制御で使用されるコマンドです。

- ・ SRD データとして"80₁₆" (2 バイト目) を送信します。
- ・ SRD1 データ (3 バイト目) を送信します。

```

;-----
;+      Clear SRD - UART mode     +
;-----
U_Clear_SRD:
    and.b  #10010000b,SRD1        ; SRD1 clear
;
    jmp    U_Loop_main           ; jump U_Loop_main
;

```

図 3.5.8.クリアステータスコマンド処理

(7-5)クリアステータスコマンド(50₁₆)処理

M16C Flash Starter との通信制御で使用されるコマンドです。

- ・ SRD1 データをクリアします。


```

;-----
;+      Version output - UART mode -      +
;-----
U_Ver_output:
    mov.w  #0,a0                ; Version address offset (a0=0)
    bclr  re_u1c1                ; Reception disabled
    bset  te_u1c1                ; Transmission enabled
U_Ver_loop:
    mov.b  ver[a0],u1tb         ; Version data transfer
?:
    btst  ti_u1c1                ; transmit buffer empty ?
    jnc   ?-                    ;
    add.w  #1,a0                ; a0 +1 increment
    cmp.w  #8,a0                ; a0=8 ?
    jltu          U_Ver_loop    ; jump U_Ver_loop at a0<8
U_Ver_end:
    btst  txept_u1c0            ; Transmit register empty ?
    jnc   U_Ver_end            ;
    jmp   U_Loop_main          ; jump U_Loop_main
;
    
```

(7-6)

図 3.5.9.バージョン出力コマンド処理

(7-6)バージョン出力コマンド(FB16)処理

M16C Flash Starter との通信制御で使用されるコマンドです。

- ・バージョン情報（2 バイト目～9 バイト目）を送信します。

```

;-----
;+      Download - UART mode -      +
;-----
U_Download:
    mov.b  #3,prcr                ; Protect off
    mov.w  #0000h,pm0            ; wait off, single chip mode
    mov.b  #02h,mcd              ; f2
    mov.b  #20h,cm1              ;
    mov.b  #08h,cm0              ;
    mov.b  #0,prcr                ; Protect on
    jmp.a  U_Download_program    ; jump U_Download_program
;
    
```

(7-7)

図 3.5.10.Download コマンド処理

(7-7)Download コマンド(FA16)処理

M16C Flash Starter の Download ボタンが押されると送信されてくるコマンドです。

プロセッサモードをシングルチップモードに変更します。

- ・マイコンの内部 ROM 上のブートローダの特定の番地（ダウンロード処理部）にジャンプします。

```

;-----
;+      Baud rate change - UART mode  +
;-----
U_BPS_B0:
    mov.b  baud,data                ; Baud rate 9600bps
    jmp    U_BPS_SET
U_BPS_B1:
    mov.b  baud+1,data              ; Baud rate 19200bps
    jmp    U_BPS_SET
U_BPS_B2:
    mov.b  baud+2,data              ; Baud rate 38400bps
    jmp    U_BPS_SET
U_BPS_B3:
    mov.b  baud+3,data              ; Baud rate 57600bps
    jmp    U_BPS_SET
U_BPS_B4:
    mov.b  baud+4,data              ; Baud rate 115200bps
U_BPS_SET:
    bclr   re_u1c1                  ; Reception disabled
    bset   te_u1c1                  ; Transmission enabled
    mov.b  r0l,u1tb                 ; r0l--> transmit buffer register
?:
    btst   ti_u1c1                  ; transmit buffer empty ?
    jnc    ?-
?:
    btst   txept_u1c0
    jnc    ?-
    bclr   te_u1c1                  ; Transmission disabled
    jsr    U_blank_end              ; UART mode Initialize
    jmp    U_Loop_main              ; jump U_Loo_main
;
    
```

(7-8)

図 3.5.11.ボーレート変更コマンド処理

(7-8)ボーレート変更コマンド(B016,B116,B216,B316,B416)処理

M16C Flash Starter との通信制御で使用されるコマンドです。

- ・ボーレート変更データ作成
- ・2バイト目データ送信（受信した1バイト目データと同じ内容のデータを送信）
- ・ボーレート変更（UART の再初期化）

```

;+++++ Freq check - UART mode - ++++++
;+      Freq check - UART mode -      +
;+++++
U_Freq_check:
    bclr    re_u1c1                ; Reception disabled
    btst    0,r1h                 ; counter = 8 times
    jc      U_Freq_check_4
;
    btst    freq_set1
    jc      U_Freq_check_1
    btst    5,r0h                 ; fer_u1rb
    jz      U_Freq_check_3
    jmp     U_Freq_check_2
U_Freq_check_1:
    cmp.b   #00h,r0l              ; "00h"?
    jeq     U_Freq_check_3
U_Freq_check_2:
    or.b    r1h,r1l              ; r1l= counter1 or counter2
U_Freq_check_3:
    xor.b   data,r1l              ; Baud = Baud xor r1l
    mov.b   r1l,data              ; data set
    mov.b   r1h,r1l
    rot.b   #-1,r1l
    rot.b   #-1,r1h              ; counter sift
    rot.b   #-1,r1l
    jmp     U_Freq_check_6
;
U_Freq_check_4:
    btst    freq_set1              ; Baud get ?
    jc      U_Freq_set_1          ; Yes , finished
    bset    freq_set1
    btst    5,r0l                 ; fer_u1rb
    jz      U_Freq_check_5
    xor.b   data,r1h
    mov.b   r1h,data
U_Freq_check_5:
    mov.b   data,data+1           ; Min Baud --> data+1
    mov.b   #01000000b,r1l        ; counter reset
    mov.b   #10000000b,r1h
    mov.b   #10000000b,data       ; Reset
U_Freq_check_6:
    jsr     U_blank_end           ; UART mode Initialize
?:
    btst    p6_6
    jz      ?-
    jmp     U_Loop_main
;
U_Freq_set_1:
    cmp.b   #00h,r0l              ; "00h"?
    jeq     U_Freq_set_2
    xor.b   data,r1h
    mov.b   r1h,data
U_Freq_set_2:
    bset    freq_set2
    mov.b   data,r1l              ; Max Baud --> data
    sub.b   data+1,r1l
    shl.b   #-1,r1l
    add.b   data+1,r1l
;
    mov.b   r1l,baud              ; 9600bps
    shl.b   #-1,r1l               ; 19200bps
    mov.b   r1l,baud+1
    shl.b   #-1,r1l              ; 38400bps
    mov.b   r1l,baud+2
    mov.b   baud,r0l              ; 57600bps
    mov.b   #0,r0h
    divu.b  #6
    mov.b   r0l,baud+3
    mov.b   baud+3,r0l            ; 115200bps
    shl.b   #-1,r0l
    mov.b   r0l,baud+4
    mov.b   baud,data
    mov.b   #0b0h,r0l            ; "B0h" set
    jsr     U_blank_end           ; UART mode Initialize
    jmp     U_BPS_SET
;
    
```

図 3.5.12.転送レジスタ設定処理

(8)転送速度レジスタ設定処理

(6)のコマンド判定処理の転送速度レジスタ設定完了チェック処理で、設定完了フラグ(freq_set2)が未完了(=0)となっている場合に分岐してきます。M16C Flash Starter から転送速度 9600bps で"0016"を 16 回受信する事で、(8)の転送速度レジスタ設定処理では、メインクロック入力発振周波数(2MHz~20MHz)に合った転送速度レジスタの値を設定します。最初の 8 回で転送速度レジスタの Min 値を、次の 8 回で Max 値を求め、その値から 9600bps 時の値を計算します。

(9)UART1 初期化処理

UART1 に関連する各レジスタの初期化を行います。この処理部は、(5)の通信初期設定処理、(7-8)のポーレート変更コマンド処理、(8)の転送速度レジスタ設定処理から呼ばれます。

```

;+++++ Subroutine : Initialize_3 - UART mode +
;+++++
Initialize_3:
U_blank_end:
;
;-----
;+      UART1      +
;-----
;---- UART nit rate generator 1
;
;      mov.w  data,u1brg
;
;
;-----
U_Initialize_31:
;
;---- Function select register B0
;
;      mov.b  #00000000b,psl0
;
;---- Function select register A0
;
;      mov.b  #10010000b,ps0      ; When you hope busy output OFF, set
"#10000000b"
;
;---- UART1 transmit/receive mode register
;
;      mov.b  #0,u1c1      ; transmit/receive disable
;      mov.b  #0,u1mr      ; u1mr reset
;      mov.b  #0000101b,u1mr
;
;      |||| +----- transfer data 8 bit long
;      |||+----- Internal clock
;      |||+----- one stop bit
;      ||+----- parity disabled
;      |+----- sleep mode deselected
;
;---- UART1 transmit/receive control register 0
;
;      mov.b  #00000100b,u1c0
;
;      |||| +----- f1 select
;      |||+----- RTS select
;      ||+----- CRT/RTS enabled
;      |+----- CMOS output(TxD)
;      ++----- Must always be "0"
;
;---- UART transmit/receive control register 2
;
;      mov.b  #00000000b,ucon
;
;      |||| +----- Transmit buffer empty
;      |||+----- Invalid
;      |+----- Must always be "0"
;      |+----- CTS/RTS shared
;      +----- fixed
;
;---- UART1 transmit/received control register 1
;
;      mov.b  #00000000b,u1c1
;
;      |||||+----- Transmission disabled
;      |||||+----- Transmission enabled
;      |||+----- Reception disabled
;      |||+----- Reception enabled
;      ++++----- fixed
;
;
;      rts
;

```

(9)

図 3.5.13.UART1 初期化处理

3.6 メモリマップ

3.6.1 RAM=10K 品

(M30800SFP-BL、 M30800SGP-BL、 M30802SGP-BL)

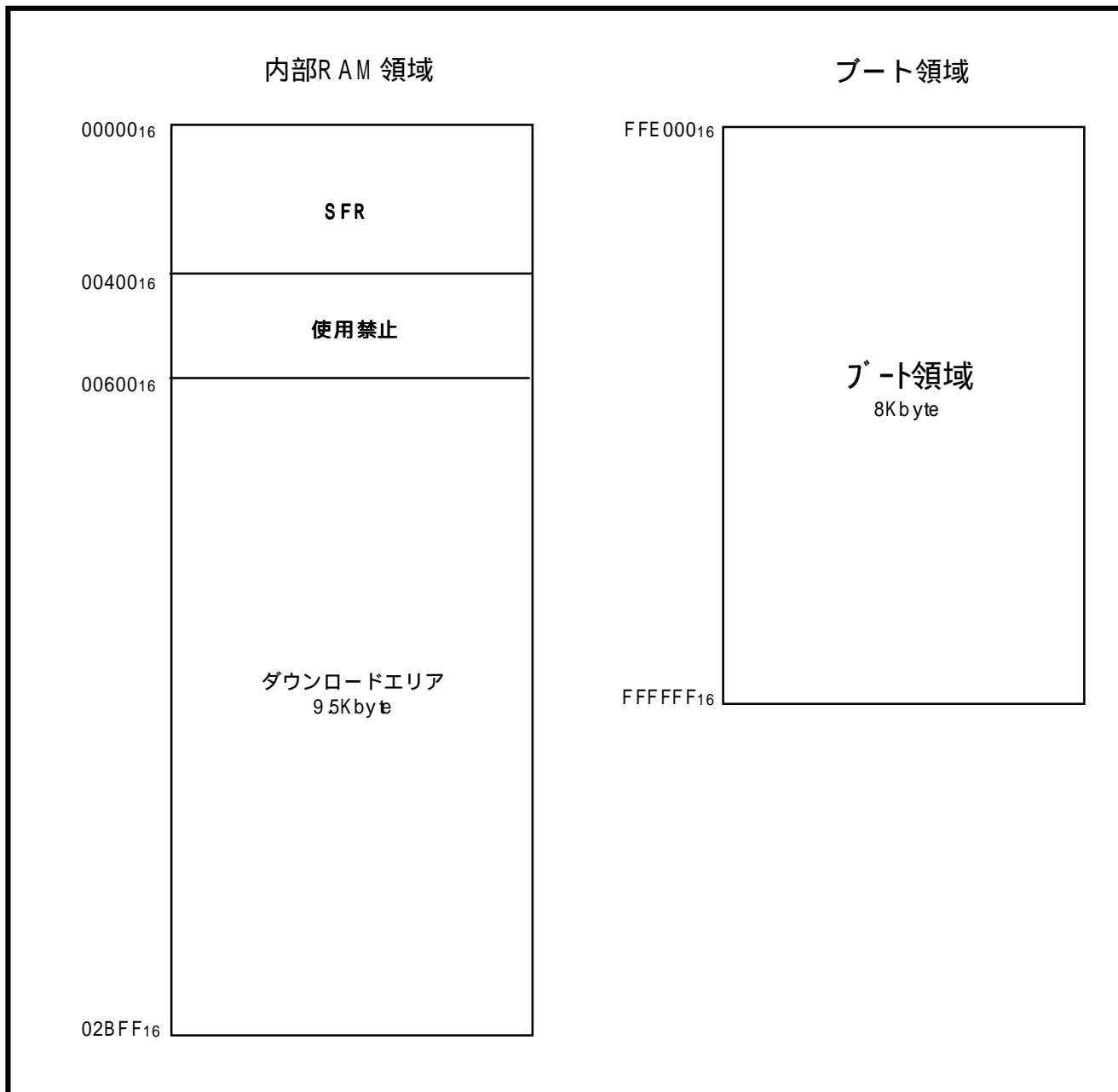


図 3.6.1.メモリマップ 1 (RAM = 10K 品)

3.6.2 RAM=24K 品

(M30803SFP-BL、 M30803SGP-BL、 M30805SGP-BL)

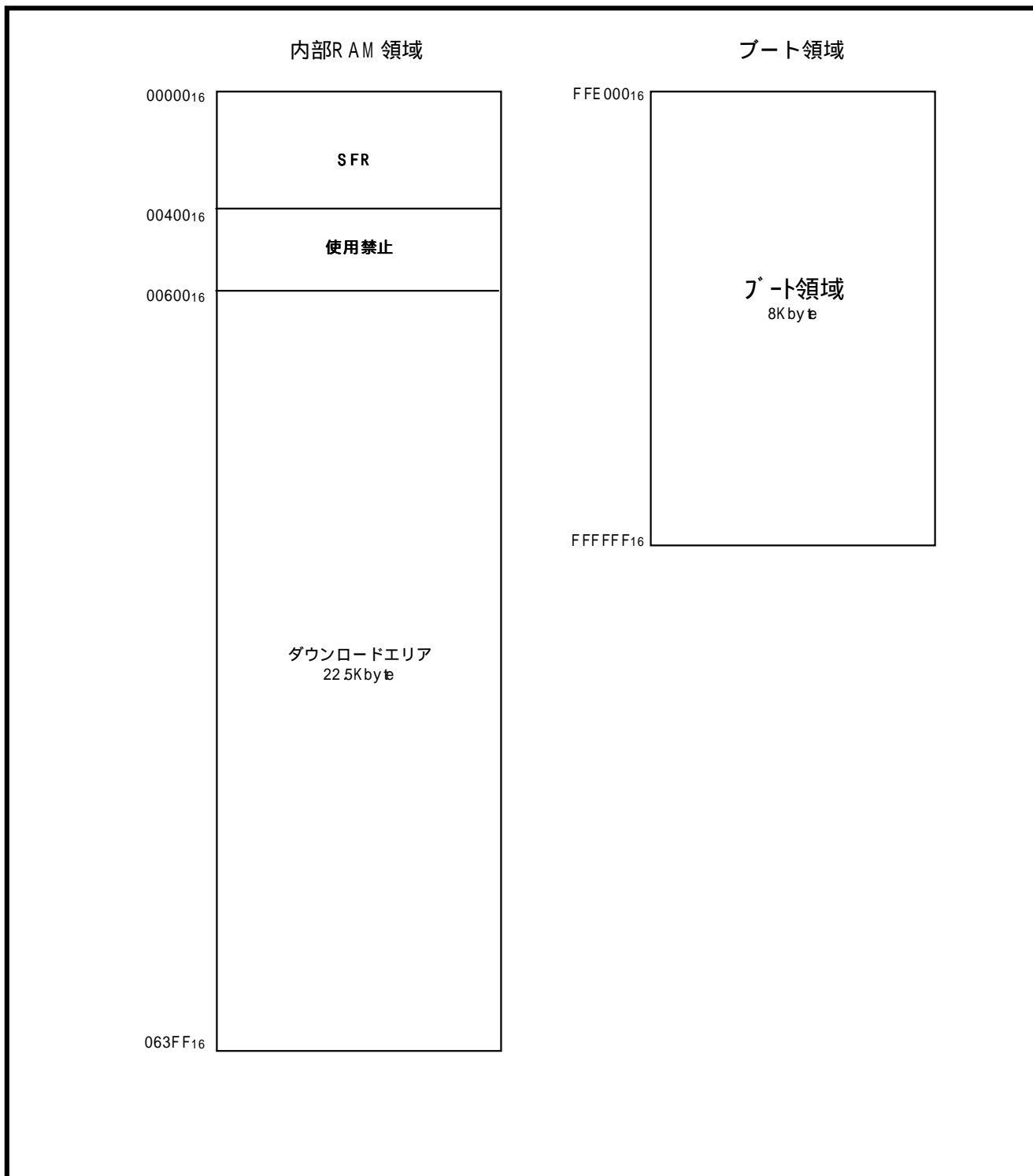


図 3.6.2.メモリマップ 2 (RAM = 24K 品)

3.6.3 M16C FlashStarter 使用時

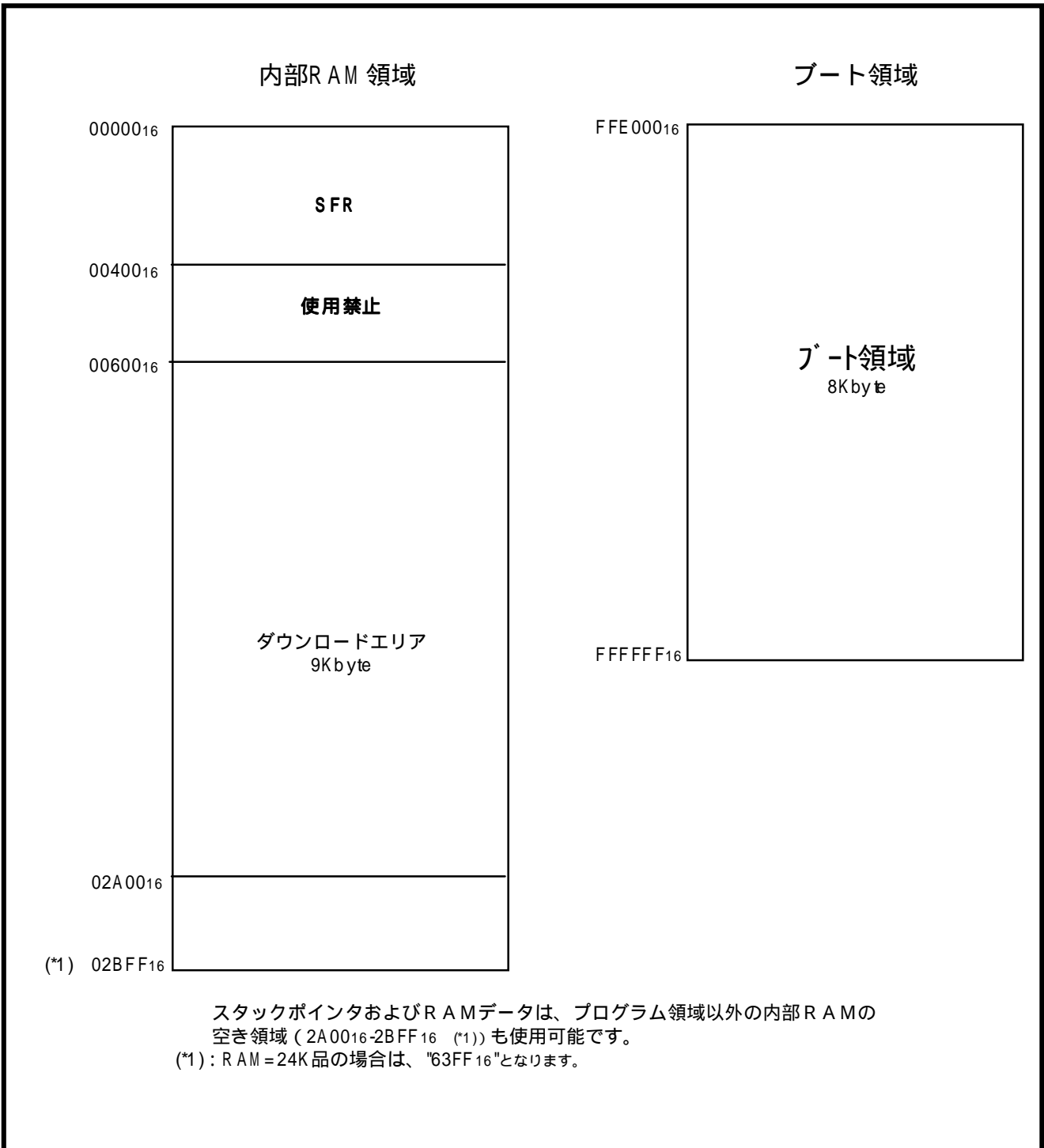


図 3.6.3.メモリマップ 3 (M16C FlashStarter 使用時)

3.6.4 MFW -1 使用時

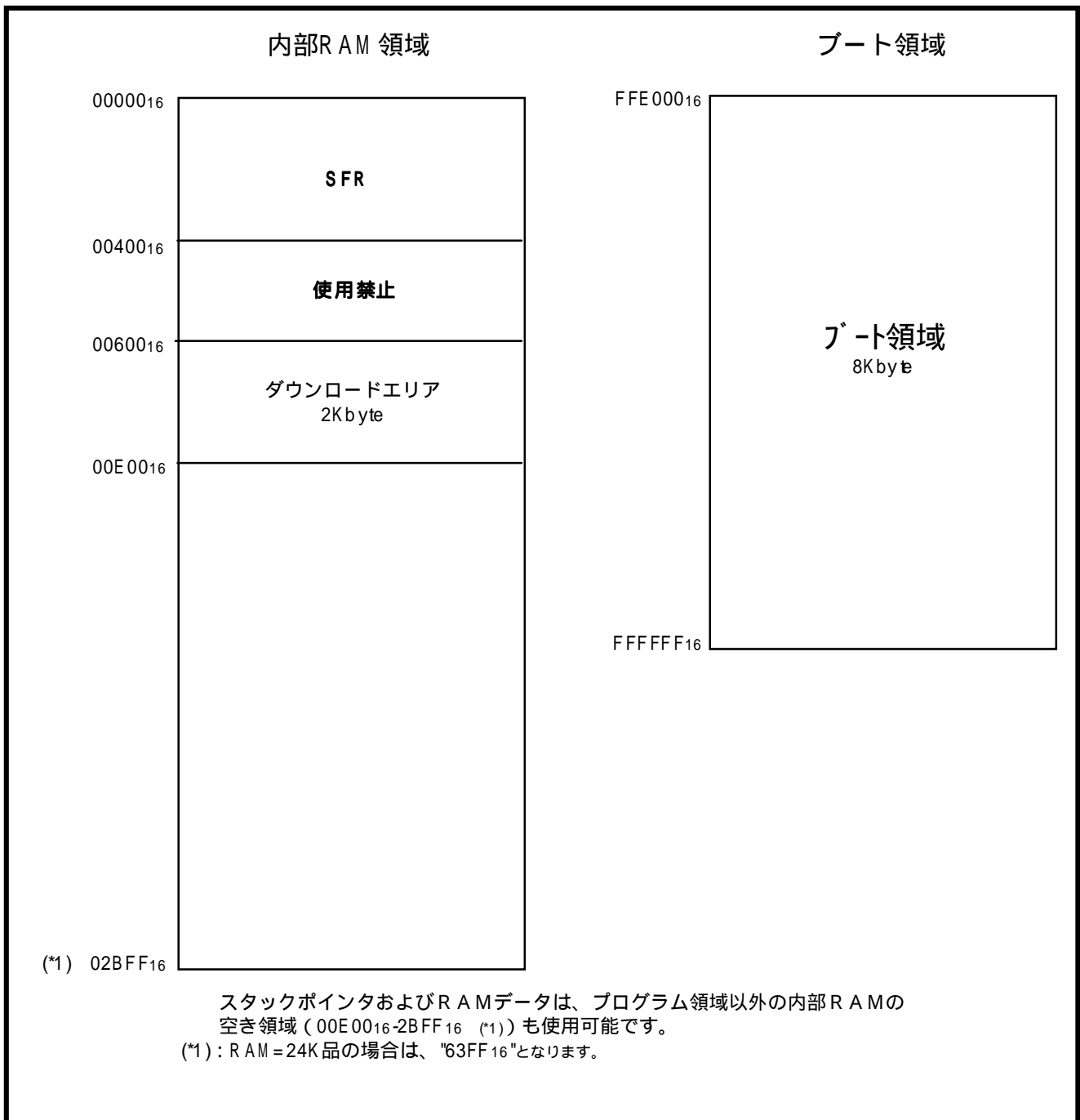


図 3.6.4.メモリマップ 4 (MFW-1 使用時)

3.7 ブートローダ起動時の接続例

3.7.1 ブートローダモード1 の場合の接続例

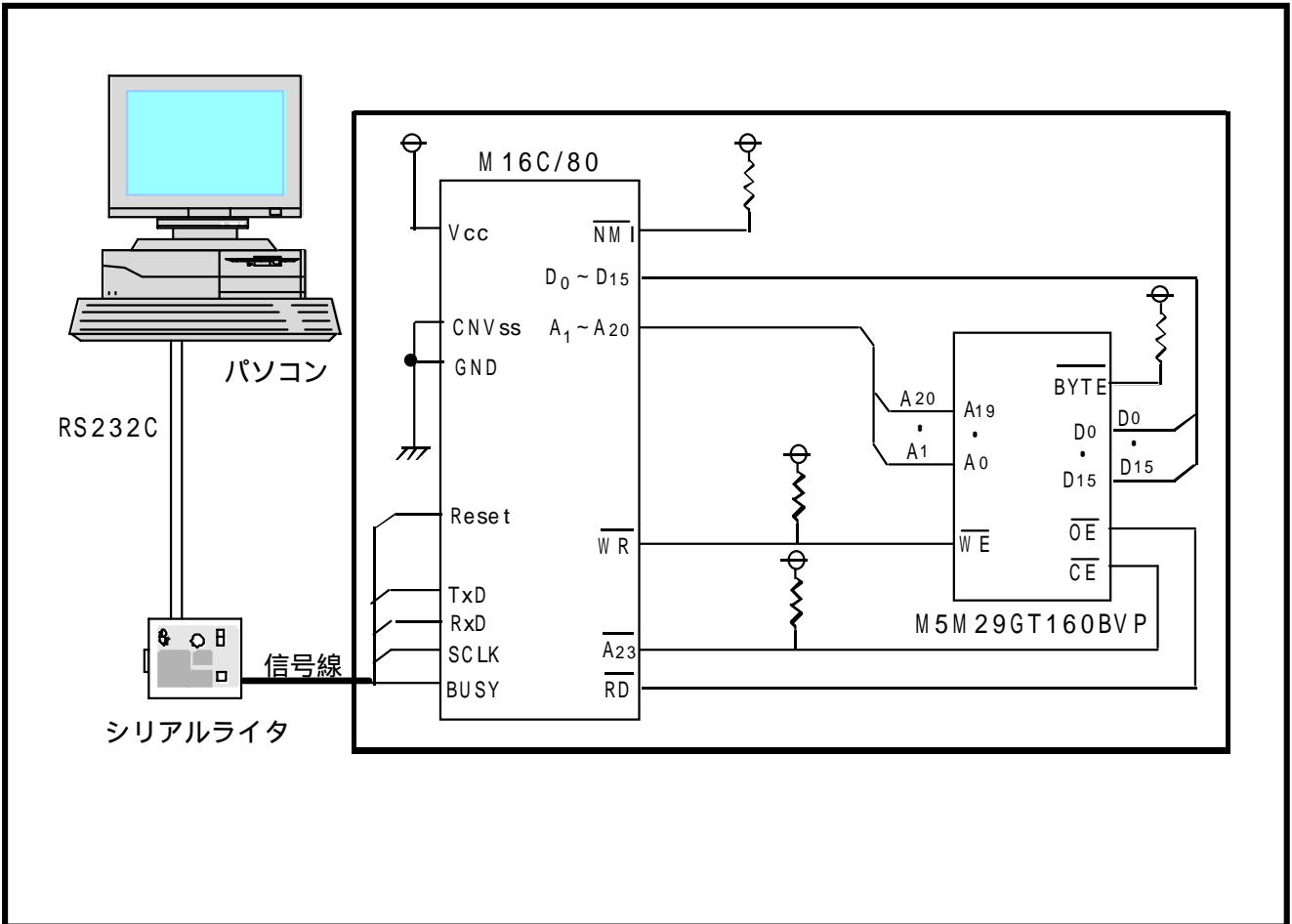


図 3.7.1.ブートローダモード1 の場合の接続例

3.7.2 ブートローダモード 2 の場合の接続例

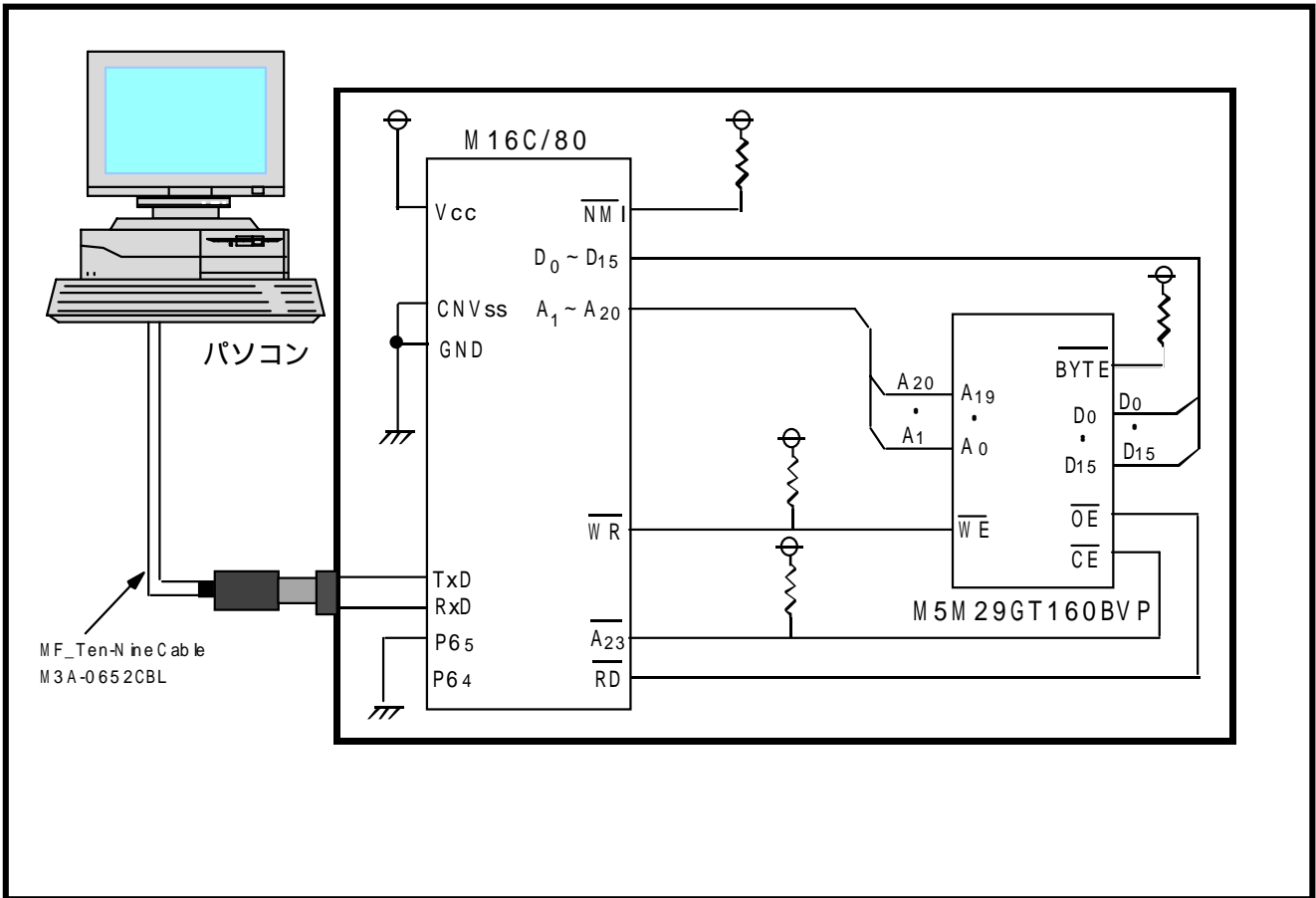


図 3.7.2.ブートローダモード 2 の場合の接続例

3.8 プログラムリスト

3.8.1 プログラムリスト 1

(MFW-1 を使用する場合の書き換えプログラムサンプル)

```

;*****
;
;*      System Name      : Rewrite Program for M16C/80 BootLoader      *
;*      File Name       : sample_Sync.a30                             *
;*      MCU             : M3080xSGP-BL                               *
;*      Xin             : 2M-20MHz (for Sync mode )                   *
;-----*
;*      Copyright,2001 MITSUBISHI ELECTRIC CORPORATION                *
;*      AND MITSUBISHI SEMICONDUCTOR SYSTEM CORPORATION              *
;*****
;
;
;
;+++++
;+      Include file                                             +
;+++++
;      .list          off
;      .include      sfr80.inc          ; SFR header include
;      .include      bl80.inc          ; Bootloader definition include
;      .list          on
;
;
;+++++
;+      Version table                                           +
;+++++
;
;      .section      rom,code
;      .org          0600h          ; Download address
;      .byte         'VER.1.01'    ; Version infomation
;
;
;=====
;+      Boot program start                                       +
;=====
Program_start:
;-----
;+      Initialize_1                                           +
;-----
;      ldc          #lstack,ISP          ; stack pointer set
;
;
;-----
;+      Processor mode register                                 +
;+      & System clock control register                       +
;-----

```

```

CPU_set:
    mov.b  #3,prcr                ; Protect off
    mov.w  #00000011b,pm0        ; wait off, micro processor mode
    mov.b  #02h,mcd              ; f2
    mov.b  #20h,cm1              ;
    mov.b  #08h,cm0              ;
    mov.b  #00001111b,ds         ; data bus width 16bit
    mov.b  #10101010b,wcr        ; all 2wait
    mov.b  #0,prcr               ; Protect on
;
;+++++
;+          Main flow - clock synchronous serial I/O mode -      +
;+++++
Main:
    jsr    Initialize_2          ; clock synchronous serial I/O mode
;
Loop_main:
    bset   ta0os
    mov.b  #0,ta0ic
Loop_main1:
    btst   ir_ta0ic              ; 300 usec ?
    jz     Loop_main1
    mov.b  #0,ta0ic
    mov.b  #0ffh,r1l             ; #ffh --> r1l (transfer dummy data)
    mov.b  r1l,u1tb              ; transfer data --> transfer buffer

    bclr   4,pd6                 ; busy input
?:
    btst   4,p6                  ; Reception start?
    jz     ?-

    bset   ta0os                 ; 300 usec timer start
?:
    btst   ir_ta0ic              ; 300 usec ?
    jc     Time_out              ; jump Time_out at time out
    btst   ri_u1c1               ; receive complete ?
    jz     ?-
    mov.w  u1rb,r0               ; receive data --> r0
;
Command_check:
    cmp.b  #0ffh,r0l             ; Read          (ffh)
    jeq    Read
    cmp.b  #041h,r0l            ; Program      (41h)
    jeq    Program
    
```

```

    cmp.b #020h,r0l      ; Erase      (20h)
    jeq   Erase
    cmp.b #0a7h,r0l      ; All erase  (a7h)
    jeq   All_erase
    cmp.b #050h,r0l      ; Clear SRD (50h)
    jeq   Clear_SRD
    cmp.b #071h,r0l      ; Read LBS  (71h)
    jeq   Read_LB
    cmp.b #077h,r0l      ; LB program (77h)
    jeq   Program_LB
    cmp.b #0fah,r0l      ; Download  (fah)
    jeq   Download

    cmp.b #070h,r0l      ; Read SRD  (70h)
    jeq   Read_SRD
    cmp.b #0fbh,r0l      ; Version out (fbh)
    jeq   Ver_output
Command_err:
    jsr   Initialize_21  ; command error,UART1 reset
    jmp   Loop_main     ; command error,jump Loop_main

;
;-----
;+      Read      +
;-----
Read:
    mov.w #0,r3          ; receive number
    mov.b #0,addr_l     ; addr_l = 0
;
Read_loop:
    mov.b r1l,u1tb      ; data transfer
    bset  ta0os         ; ta0 start
    mov.b #0,ta0ic      ; clear time out
?:
    btst  ir_ta0ic      ; time out error ?
    jc   Time_out       ; jump Time_out at time out
    btst  ri_u1c1       ; receive complete ?
    jnc  ?-
    mov.w u1rb,r0       ; receive data read --> r0
;
    add.w #1,r3         ; r3 +1 increment
    cmp.w #2,r3         ; r3 = 2 ?
    jgtu Read_data     ; jump Read_data at r3>3
    mov.w r3,a0         ; r3 --> a0
    mov.b r0l,addr_l[a0] ; Store address

```

```

    cmp.w  #2,r3                ; r3 = 2 ?
    jltu   Read_loop           ; jump Read_loop at r3<2
;
    mov.w  addr_l,a0           ; addr_l,m --> a0
    mov.b  addr_h,a1           ; addr_h --> a1
    sha.l  #16,a1              ;
    add.l  a0,a1               ; a1 is address-data
;
Read_data:
;
; Flash memory read & store to r1l
;
    add.l  #1,a1               ; address increment
    cmp.w  #258,r3            ; r3 = 258 ?
    jne    Read_loop           ; jump Read_loop at r<260
;
    jmp    Loop_main           ; jump Loop_main
;
;-----
;+          Program          +
;-----
Program:
    mov.w  #0,r3                ; receive number
    mov.b  #0,addr_l           ; addr_l = 0
Program_loop_1:
    mov.b  r1l,u1tb            ; data transfer
    bset   ta0os                ; ta0 start
    mov.b  #0,ta0ic            ; clear time out
?:
    btst   ir_ta0ic            ; time out error ?
    jc     Time_out            ; jump Time_out at time out
    btst   ri_u1c1             ; receive complete ?
    jnc    ?-
    mov.w  u1rb,r0              ; receive data read --> r0
    add.w  #1,r3                ; r3 +1 increment
    mov.w  r3,a0                ; r3 --> a0
    mov.b  r0l,addr_l[a0]       ; Store address
    cmp.w  #258,r3             ; r3 = 258 ?
    jltu   Program_loop_1       ; jump Program_loop_1 at r3<258
;
    mov.w  #0,r3                ; writing number (r3=0)
Program_loop_2:
    mov.b  addr_h,a1           ; addr_h --> a1
    sha.l  #16,a1              ;
    mov.w  r3,a0                ; r3 --> a0
    mov.w  data[a0],r1         ; data --> r1
    mov.w  addr_l,a0           ; addr_l,m --> a0

```

```

    add.l   a0,a1
    ;
    ; data write
    ;
    add.w   #2,addr_l           ; address +2 increment
    add.w   #2,r3              ; writing number +2 increment
    cmp.w   #255,r3           ; r3 = 255 ?
    jltu   Program_loop_2     ; jump Program_loop_2 at r3<255
Program_end:
    jmp    Loop_main          ; jump Loop_main
;
;-----
;+           Block erase           +
;-----
Erase:
    mov.w   #1,r3             ; receive number (r3=1)
Erase_loop:
    mov.b   r1l,u1tb         ; data transfer
    bset    ta0os            ; ta0 start
    mov.b   #0,ta0ic         ; clear time out
?:
    btst    ir_ta0ic         ; time out error ?
    jc      Time_out         ; jump Time_out at time out
    btst    ri_u1c1          ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0          ; receive data read --> r0
    mov.w   r3,a0            ; r3 --> a0
    mov.b   r0l,addr_l[a0]   ; Store address
    add.w   #1,r3            ; r3 +1 increment
    cmp.w   #4,r3           ; r3=4 ?
    jltu   Erase_loop        ; jump Erase_loop at r3<4
;
    cmp.b   #0d0h,data       ; Confirm command check
    jne     Erase_end        ; jump Erase_end at Confirm command error
;
; Block Erase
;
Erase_end:
    jmp    Loop_main          ; jump Loop_main
;
;-----
;+           All erase ( unlock block )           +
;-----
All_erase:
    mov.b   r1l,u1tb         ; data transfer
    bset    ta0os            ; ta0 start
    mov.b   #0,ta0ic         ; clear time out

```



```

?:
    btst    ir_ta0ic                ; time out error ?
    jc      Time_out                ; jump Time_out at time out
    btst    ri_u1c1                ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0                ; receive data read --> r0
;
    cmp.b   #0d0h,r0l              ; Confirm command check
    jne     All_erase_end          ; jump All_erase_end at Confirm command error
;
;
; All Erase
;
All_erase_end:
    jmp     Loop_main              ; jump Loop_main
;
;-----
;+          Read SRD                +
;-----
Read_SRD:
    mov.w   #0,r3                  ; receive number (r3=0)
    mov.b   #80h,r1l              ; dummy SRD set
;
Read_SRD_loop:
    mov.b   r1l,u1tb              ; data transfer
    bset    ta0os                  ; ta0 start
    mov.b   #0,ta0ic              ; clear time out
?:
    btst    ir_ta0ic                ; time out error ?
    jc      Time_out                ; jump Time_out at time out
    btst    ri_u1c1                ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0                ; receive data read --> r0
    mov.b   SRD1,r1l              ; SRD1 data --> r1l
    add.w   #1,r3                  ; r3 +1 increment
    cmp.w   #2,r3                  ; r3=2 ?
    jltu   Read_SRD_loop          ; jump Read_SRD_loop at r3<2
;
    jmp     Loop_main              ; jump Loop_main
;
;-----
;+          Clear SRD                +
;-----

```

```

Clear_SRD:
;
;   and.b   #10011100b,SRD1           ; SRD1 clear
;
;   jmp     Loop_main                 ; jump Loop_main
;
;-----
;+       Read Lock Bit                +
;-----
Read_LB:
    mov.w   #1,r3                     ; receive number (r3=1)
Read_LB_loop:
    mov.b   r1l,u1tb                   ; data transfer
    bset    ta0os                       ; ta0 start
    mov.b   #0,ta0ic                   ; clear time out
?:
    btst    ir_ta0ic                   ; time out error ?
    jc      Time_out                   ; jump Time_out at time out
    btst    ri_u1c1                     ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0                     ; receive data read --> r0
    mov.w   r3,a0                       ; r3 --> a0
    mov.b   r0l,addr_l[a0]              ; Store address
    add.w   #1,r3                       ; r3 +1 increment
    cmp.w   #3,r3                       ; r3=3 ?
    jltu   Read_LB_loop                 ; jump Read_LB_loop at r3<3
    jgtu   Read_LB_end                 ; jump Read_LB_end   at r3>3
;
;   mov.w   #00aah,r1                 ; dummy read LB status set
;
;   jmp     Read_LB_loop               ; jump Read_LB_loop
;
Read_LB_end:
    jmp     Loop_main                   ; jump Loop_main
;
;-----
;+       Program Lock Bit             +
;-----
Program_LB:
    mov.w   #1,r3                     ; receive number (r3=1)
Program_LB_loop:
    mov.b   r1l,u1tb                   ; data transfer
    bset    ta0os                       ; ta0 start
    mov.b   #0,ta0ic                   ; clear time out
?:
    btst    ir_ta0ic                   ; time out error ?
    jc      Time_out                   ; jump Time_out at time out

```

```

    btst    ri_u1c1                ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0                ; receive data read --> r0
    mov.w   r3,a0                  ; r3 --> a0
    mov.b   r0l,addr_l[a0]         ; Store address
    add.w   #1,r3                  ; r3 +1 increment
    cmp.w   #4,r3                  ; r3=4 ?
    jltu   Program_LB_loop         ; jump Program_LB_loop at r3<4
    cmp.b   #0d0h,data             ; Confirm command check
    jne     Program_LB_end         ; jump Program_LB_end at Confirm command error
;
Program_LB_end:
    jmp     Loop_main              ; jump Loop_main
;
;-----
;+          Version output          +
;-----
Ver_output:
    mov.w   #0,a0                  ; Version address offset (a0=0)
Ver_output_loop:
    mov.b   ver[a0],u1tb           ; Version data transfer
    bset    ta0os                  ; ta0 start
    mov.b   #0,ta0ic              ; clear time out
?:
    btst    ir_ta0ic              ; time out error ?
    jc      Time_out              ; jump Time_out at time out
    btst    ri_u1c1                ; receive complete ?
    jnc     ?-
    mov.w   u1rb,r0                ; receive data read --> r0
    add.w   #1,a0                  ; a0 +1 increment
    cmp.w   #8,a0                  ; a0=8 ?
    jltu   Ver_output_loop         ; jump Ver_output_loop at a0<8
Ver_output_end:
    jmp     Loop_main              ; jump Loop_main
;
;-----
;+          Download                +
;-----
Download:
    mov.b   #3,prcr                ; Protect off
    mov.w   #0000h,pm0             ; wait off, single chip mode
    mov.b   #02h,mcd               ; f2
    mov.b   #20h,cm1              ;
    mov.b   #08h,cm0              ;
    mov.b   #0,prcr               ; Protect on

    jmp.a   Download_program       ; jump Download_program

```

```

;
;-----
;+      Time_out      +
;-----
Time_out:
    bset    sr9          ; SRD1 time out flag set
    jmp     Command_err ; jump Command_err at time out
;
;+++++
;+      Subroutine : Initialize_2      +
;+++++
Initialize_2:
    bset    sr10          ; check complete at r0=ffffh
    bset    sr11
    bset    blank        ; blank flag set
;
;-----
;+      UART1      +
;-----
Initialize_21:
;----- Function select register A0
;
    mov.b   #10010000b,ps0
;
;----- Function select register B0
;
    mov.b   #00000000b,ps10
;
;----- UART1 transmit/receive mode register
;
    mov.b   #0,u1c1          ; transmit/receive disable
    mov.b   #0,u1mr          ; u1mr reset
    mov.b   #00001001b,u1mr
;
    |||||+++----- clock synchronous SI/O
;
    |||||+----- external clock
;
    ++++----- fixed
;
;----- UART1 transmit/receive control register 0
;
    mov.b   #00000100b,u1c0
;
    |||| |++----- f1 select
;
    |||| +----- RTS select
;
    |||+----- CTS/RTS enabled
;
    |+----- CMOS output(TxD)
;
    |+----- falling edge select
;
    +----- LSB first
;
;

```

```

;----- UART transmit/receive control register 2
;
;      mov.b   #00000000b,ucon
;      |||||++----- Transmit buffer empty
;      |||++----- Continuous receive mode disabled
;      ||++----- CLK/CLKS normal
;      |+----- CTS/RTS shared
;      +----- fixed
;
;----- UART1 transmit/receive control register 1
;
;      mov.b   #00000101b,u1c1
;      ||| | +----- Transmission enabled
;      ||| +----- Reception enabled
;      +----- fixed
;
;-----
;+      Timer A0      +
;-----
;----- Timer A0 mode register
;
;      mov.b   #00000010b,ta0mr
;      ||| |++----- One-shot mode
;      ||| +----- Pulse not output
;      ||+----- One-shot start flag
;      |+----- fixed
;      ++----- f1 select
;
;::;   mov.b   #0,ta0ic      ; clear TA0 interrupt flag
;      mov.w   #6000-1,ta0   ; set 300 usec at 20 MHz
;      bset   ta0s
;      mov.b   #0,ta0ic      ; clear TA0 interrupt flag      changed 0629
;
;      rts
;
;
;
;
;      .end

```

3.8.2 プログラムリスト 2

(M16C FlashStarter を使用する場合の書き換えサンプルプログラム)

```

;*****
;
;*      System Name      : Rewrite Program for M16C/80 BootLoader      *
;*      File Name       : sample_UART.a30                             *
;*      MCU              : M3080xSGP-BL                               *
;*      Xin              : 2M-20MHz (for UART mode )                  *
;*-----*
;*      Copyright,2001 MITSUBISHI ELECTRIC CORPORATION                *
;*      AND MITSUBISHI SEMICONDUCTOR SYSTEM CORPORATION              *
;*****
;
;
;
;
;+++++
;+      Include file                                             +
;+++++
;      .list          off
;      .include      sfr80.inc          ; SFR header include
;      .include      bl80.inc          ; Bootloader definition include
;      .list          on
;
;
;+++++
;+      Version table                                           +
;+++++
;
;      .section      rom,code
;      .org          0600h          ; Download address
;      .byte         'VER.1.01'     ; Version infomation
;
;
;=====
;+      Boot program start                                       +
;=====
Program_start:
;-----
;+      Initialize_1                                           +
;-----
;      ldc          #lstack,ISP          ; stack pointer set
;
;
;-----
;+      Processor mode register                                 +
;+      & System clock control register                       +
;-----
;

```

```

CPU_set:
    mov.b    #3,prcr                ; Protect off
    mov.w    #00000011b,pm0        ; wait off, micro processor mode
    mov.b    #02h,mcd              ; f2
    mov.b    #20h,cm1              ;
    mov.b    #08h,cm0              ;
    mov.b    #00001111b,ds         ; data bus width 16bit
    mov.b    #10101010b,wcr        ; all 2wait
    mov.b    #0,prcr              ; Protect on
;
;-----
;=====
;+      Transfer Program -- UART mode      +
;+          (1) Main flow                  +
;+          (2) Communication program for flash memory control  +
;=====
;
;+++++
;+      Main flow - UART mode -          +
;+++++
U_Main:
    bst     updata_f                ;
    bmltu   updata_f                ; if "C"flag is "0", updata_f set "1"
    jc      U_Main1                ; if "C"flag is "1", initialize execute(jump
U_Main1)
    jmp     U_Loop_main            ;
U_Main1:
    bclr    updata_f                ;
    bclr    freq_set1              ; freq set flag clear
    bclr    freq_set2              ;
    mov.b   #01111111b,data        ; Initialize Baud rate
    jsr     Initialize_3           ; UART mode Initialize
    mov.b   #01000000b,r1l         ; counbter1,2 reset
    mov.b   #10000000b,r1h         ;
    mov.w   u1rb,r0                ; receive data --> r0
;
U_Loop_main:
    bclr    te_u1c1                ; Transmission disabled
    bset    re_u1c1                ; Reception enabled
?:
    bst     ri_u1c1                ; receive complete ?
    jz     ?-                      ;
    mov.w   u1rb,r0                ; receive data --> r0
    bst     freq_set2              ;
    jz     U_Freq_check            ;

```

```

;
U_Command_check:
    cmp.b    #0ffh,r0l                ; Read            (ffh)
    jeq     U_Read
    cmp.b    #041h,r0l                ; Program        (41h)
    jeq     U_Program
    cmp.b    #0a7h,r0l                ; All erase      (a7h)
    jeq     U_All_erase
    cmp.b    #050h,r0l                ; Clear SRD     (50h)
    jeq     U_Clear_SRD
    cmp.b    #0fah,r0l                ; Download      (fah)
    jeq     U_Download
    cmp.b    #070h,r0l                ; Read SRD      (70h)
    jeq     U_Read_SRD
    cmp.b    #0fbh,r0l                ; Version out   (fbh)
    jeq     U_Ver_output
    cmp.b    #0b0h,r0l                ; Baud rate 9600bps (b0h)
    jeq     U_BPS_B0
    cmp.b    #0b1h,r0l                ; Baud rate 19200bps (b1h)
    jeq     U_BPS_B1
    cmp.b    #0b2h,r0l                ; Baud rate 38400bps (b2h)
    jeq     U_BPS_B2
    cmp.b    #0b3h,r0l                ; Baud rate 57600bps (b3h)
    jeq     U_BPS_B3
    cmp.b    #0b4h,r0l                ; Baud rate 115200bps (b4h)
    jeq     U_BPS_B4
    jsr     U_Initialize_31           ; command error, UART mode Initialize
    jmp     U_Loop_main              ; jump U_Loop_main
;
;-----
;+          Read - UART mode -          +
;-----
U_Read:
    mov.w    #0,r3                    ; receive number
    mov.b    #0,addr_l                ; addr_l = 0
?:
    btst     ri_u1c1                  ; receive complete ?
    jnc     ?-
;
    mov.w    u1rb,r0                  ; receive data read --> r0
    add.w    #1,r3                    ; r3 +1 increment
    mov.w    r3,a0                    ; r3 --> a0
    mov.b    r0l,addr_l[a0]           ; Store address
    cmp.w    #2,r3                    ; r3 = 2 ?
    jltu    ?-                        ; jump Read_loop at r3<2
;

```



```

        mov.w  addr_l,a0          ; addr_l,m --> a0
        mov.b  addr_h,a1          ; addr_h  --> a1
        sha.l  #16,a1             ;
        add.l  a0,a1              ; a1 is address-data
;
        bclr   re_u1c1            ; Reception disabled
        bset   te_u1c1            ; Transmission enabled
U_Read_data:
        cmp.w  #258,r3            ; r3 = 258 ?
        jz     U_Read_end
        ;
        ; Flash memory read & store to r1l
        ;
        mov.b  r1l,u1tb           ; r1l --> transmit buffer register
?:
        btst   ti_u1c1            ; transmit buffer empty ?
        jnc    ?-
        add.l  #1,a1              ; address increment
        add.w  #1,r3              ; counter increment
        jmp    U_Read_data        ; jump U_Read_data
;
U_Read_end:
        btst   txept_u1c0         ; Transmit register empty ?
        jnc    U_Read_end
        jmp    U_Loop_main
;
-----
;+          Program - UART mode -          +
;-----
U_Program:
        mov.w  #0,r3              ; receive number
        mov.b  #0,addr_l          ; addr_l = 0
        mov.w  sum,crcd           ; for Read check command
U_Program_loop:
        btst   ri_u1c1            ; receive complete ?
        jnc    U_Program_loop
        mov.w  u1rb,r0            ; receive data read --> r0
        add.w  #1,r3              ; r3 +1 increment
        mov.w  r3,a0              ; r3 --> a0
        mov.b  r0l,addr_l[a0]     ; Store address
        cmp.w  #258,r3            ; r3 = 258 ?
        jltu  U_Program_loop      ; jump U_Program_loop at r3<258
;
        mov.w  #0,r3              ; writing number (r3=0)

U_Program_loop_2:
        mov.b  addr_h,a1          ; addr_h  --> a1

```

```

sha.l   #16,a1
mov.w   r3,a0                ; r3    --> a0

mov.w   data[a0],r1         ; data  --> r1
mov.w   addr_l,a0          ; addr_l,m --> a0
add.l   a0,a1
;
; data write
;
mov.b   r1l,crcin           ; for Read check command
mov.b   r1h,crcin
;
;
add.w   #2,addr_l          ; address +2 increment
add.w   #2,r3              ; writing number +2 increment
cmp.w   #255,r3            ; r3 = 255 ?
jltu   U_Program_loop_2   ; jump U_Program_loop_2 at r3<255
U_Program_end:
mov.w   crcd,sum           ; for Read check command
jmp     U_Loop_main        ; jump U_Loop_main
;
;-----
;+           All erase ( unlock block ) - UART mode - +
;-----
U_All_erase:
btst   ri_u1c1             ; receive complete ?
jnc    U_All_erase
;
mov.w   u1rb,r0            ; receive data read --> r0
cmp.b   #0d0h,r0l         ; Confirm command check
jne     U_All_erase_end    ; jump U_All_erase_end at Confirm command error
;
; All erase
;
U_All_erase_end:
jmp     U_Loop_main        ; jump U_Loop_main
;
;-----
;+           Read SRD - UART mode           +
;-----
U_Read_SRD:
bclr   re_u1c1             ; Reception disabled
;
mov.w   #0,r3              ; receive number (r3=0)
;
;

```

```

        mov.b  #80h, r1l          ; dummy SRD set
        bset   te_u1c1           ; Transmission enabled
U_Read_SRD_loop:
        mov.b  r1l, u1tb         ; r1l --> transmit buffer register
?:
        btst   ti_u1c1          ; transmit buffer empty ?

        jnc    ?-
        mov.b  SRD1, r1l        ; SRD1 data --> r1l
        add.w  #1, r3           ; r3 +1 increment
        cmp.w  #2, r3           ; r3=2 ?
        jltu  U_Read_SRD_loop   ; jump U_Read_SRD_loop at r3<2
U_Read_SRD_end:
        btst   txept_u1c0       ; Transmit register empty ?
        jnc    U_Read_SRD_end
;
        jmp    U_Loop_main      ; jump U_Loop_main
;
;-----
;+          Clear SRD - UART mode          +
;-----
U_Clear_SRD:
;
        and.b  #10010000b, SRD1 ; SRD1 clear
;
        jmp    U_Loop_main      ; jump U_Loop_main
;
;-----
;+          Version output - UART mode -   +
;-----
U_Ver_output:
        mov.w  #0, a0           ; Version address offset (a0=0)
        bclr   re_u1c1         ; Reception disabled
        bset   te_u1c1         ; Transmission enabled
U_Ver_loop:
        mov.b  ver[a0], u1tb    ; Version data transfer
?:
        btst   ti_u1c1         ; transmit buffer empty ?
        jnc    ?-
        add.w  #1, a0          ; a0 +1 increment
        cmp.w  #8, a0          ; a0=8 ?
        jltu  U_Ver_loop       ; jump U_Ver_loop at a0<8
U_Ver_end:
        btst   txept_u1c0       ; Transmit register empty ?
        jnc    U_Ver_end
        jmp    U_Loop_main      ; jump U_Loop_main
;

```

```

;-----
;+           Download - UART mode -           +
;-----
U_Download:
    mov.b    #3,prcr                ; Protect off
    mov.w    #0000h,pm0             ; wait off, single chip mode
    mov.b    #02h,mcd               ; f2

    mov.b    #20h,cm1              ;
    mov.b    #08h,cm0              ;
    mov.b    #0,prcr               ; Protect on

    jmp.a    U_Download_program     ; jump U_Download_program
;
;-----
;+           Baud rate change - UART mode     +
;-----
U_BPS_B0:
    mov.b    baud,data              ; Baud rate 9600bps
    jmp      U_BPS_SET
U_BPS_B1:
    mov.b    baud+1,data            ; Baud rate 19200bps
    jmp      U_BPS_SET
U_BPS_B2:
    mov.b    baud+2,data            ; Baud rate 38400bps
    jmp      U_BPS_SET
U_BPS_B3:
    mov.b    baud+3,data            ; Baud rate 57600bps
    jmp      U_BPS_SET
U_BPS_B4:
    mov.b    baud+4,data            ; Baud rate 115200bps
U_BPS_SET:
    bclr    re_u1c1                 ; Reception disabled
    bset    te_u1c1                 ; Transmission enabled
    mov.b   r0l,u1tb                ; r0l --> transmit buffer register
?:
    btst    ti_u1c1                 ; transmit buffer empty ?
    jnc     ?-
?:
    btst    txept_u1c0
    jnc     ?-
    bclr    te_u1c1                 ; Transmission disabled
    jsr     U_blank_end             ; UART mode Initialize
    jmp     U_Loop_main             ; jump U_Loo_main
;
;+++++
;+           Freq check - UART mode -         +
;+++++

```

```

U_Freq_check:
    bclr    re_u1c1                ; Reception disabled
    btst    0,r1h                 ; counter = 8 times
    jc      U_Freq_check_4
;
    btst    freq_set1
    jc      U_Freq_check_1

    btst    5,r0h                 ; fer_u1rb
    jz      U_Freq_check_3
    jmp     U_Freq_check_2
U_Freq_check_1:
    cmp.b   #00h,r0l              ; "00h"?
    jeq     U_Freq_check_3
U_Freq_check_2:
    or.b    r1h,r1l               ; r1l = counter1 or counter2
U_Freq_check_3:
    xor.b   data,r1l              ; Baud = Baud xor r1l
    mov.b   r1l,data              ; data set
    mov.b   r1h,r1l
    rot.b   #-1,r1l
    rot.b   #-1,r1h               ; counter sift
    rot.b   #-1,r1l
    jmp     U_Freq_check_6
;
U_Freq_check_4:
    btst    freq_set1             ; Baud get ?
    jc      U_Freq_set_1          ; Yes , finished
    bset    freq_set1
    btst    5,r0l                 ; fer_u1rb
    jz      U_Freq_check_5
    xor.b   data,r1h
    mov.b   r1h,data
U_Freq_check_5:
    mov.b   data,data+1           ; Min Baud --> data+1
    mov.b   #01000000b,r1l        ; counter reset
    mov.b   #10000000b,r1h
    mov.b   #10000000b,data       ; Reset
U_Freq_check_6:
    jsr     U_blank_end           ; UART mode Initialize
?:
    btst    p6_6
    jz      ?-
    jmp     U_Loop_main
;

```

```

U_Freq_set_1:
    cmp.b    #00h,r0l                ; "00h"?
    jeq     U_Freq_set_2
    xor.b    data,r1h
    mov.b    r1h,data
U_Freq_set_2:
    bset     freq_set2
    mov.b    data,r1l                ; Max Baud --> data
    sub.b    data+1,r1l
    shl.b    #-1,r1l

    add.b    data+1,r1l
;
    mov.b    r1l,baud                ; 9600bps
    shl.b    #-1,r1l                ; 19200bps
    mov.b    r1l,baud+1
    shl.b    #-1,r1l                ; 38400bps
    mov.b    r1l,baud+2
    mov.b    baud,r0l                ; 57600bps
    mov.b    #0,r0h
    divu.b   #6
    mov.b    r0l,baud+3
    mov.b    baud+3,r0l              ; 115200bps
    shl.b    #-1,r0l
    mov.b    r0l,baud+4
    mov.b    baud,data
    mov.b    #0b0h,r0l               ; "B0h" set
    jsr     U_blank_end              ; UART mode Initialize
    jmp     U_BPS_SET
;
;+++++
;+          Subroutine : Initialize_3 - UART mode          +
;+++++
Initialize_3:
U_blank_end:
;
;-----
;+          UART1          +
;-----
;----- UART nit rate generator 1
;
    mov.w    data,u1brg
;
U_Initialize_31:
;

```



```
    rts  
;  
    .end
```


3.8.3 プログラムリスト 3

(プログラムリスト 1 および 2 のためのインクルードファイルサンプル)

```

*****
;
;*
;*
;* file name : definition of Download sample program *
;*           for M16C/80 Bootloader *
;*
;*
;* Version   : 0.01 ( 2000- 8- 1 ) *
;*           for Bootloader Ver.1.00 *
*****
;
;
;-----
; define of symbols
;-----
Ram_TOP      .equ    000400h      ;;
Istack      .equ    002a00h      ;; Stack pointer
SB_base     .equ    000400h      ;; SB base
;
Download_program .equ    0ffe100h      ;; Download function top address(Boot loader
mode1 Sync)
U_Download_program .equ    0ffe200h      ;; Download function top address(Boot loader
mode2 UART)
;
;
;
Vector      .equ    0ffffdch
;
    .section    memory,data
    .org        Ram_TOP
;
SRD:        .blkb    1            ;; not use
SRD1:       .blkb    1            ;; SRD1
ver:        .blkb    10           ;; version infomation
SF:         .blkb    1            ;; status flag
unuse:      .blkb    4            ;;
addr_l:     .blkb    1            ;; address L
addr_m:     .blkb    1            ;; address M
addr_h:     .blkb    1            ;; address H
data:       .blkb    300          ;; data buffer
buff:       .blkb    20           ;;
ID_err:     .blkb    1            ;; not use
sum:        .blkb    2            ;;
baud:       .blkb    5            ;;
BY_sts:     .blkb    2            ;; not use
;

```

```

sr8          .btequ 0,SRD1      ;;
sr9          .btequ 1,SRD1      ;; Time out bit
sr10         .btequ 2,SRD1      ;; ID check(for Internal flash memory)
sr11         .btequ 3,SRD1      ;; ID check(for Internal flash memory)
sr12         .btequ 4,SRD1      ;; check sum bit
sr13         .btequ 5,SRD1      ;;
sr14         .btequ 6,SRD1      ;;
sr15         .btequ 7,SRD1      ;; download check bit
;
ram_check   .btequ 0,SF        ;; not use
blank       .btequ 1,SF        ;; not use
old_mode    .btequ 2,SF        ;; not use
freq_set1   .btequ 3,SF        ;;
freq_set2   .btequ 4,SF        ;;
updata_f    .btequ 5,SF        ;; download flag
;
;

```

安全設計に関するお願い

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

本資料ご利用に際しての留意事項

- ・ 本資料は、お客様が用途に応じた適切な三菱半導体製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について三菱電機が所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
- ・ 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、三菱電機は責任を負いません。
- ・ 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、三菱電機は、予告なしに、本資料に記載した製品または仕様を変更することがあります。三菱半導体製品のご購入に当たりましては、事前に三菱電機または特約店へ最新の情報をご確認頂きますとともに、三菱電機半導体情報ホームページ (<http://www.mitsubishielectric.co.jp/semiconductors>) などを通じて公開される情報に常にご注意ください。
- ・ 本資料に記載した情報は、正確を期すため、慎重に制作したのですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、三菱電機はその責任を負いません。
- ・ 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。三菱電機は、適用可否に対する責任を負いません。
- ・ 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、三菱電機または特約店へご照会ください。
- ・ 本資料の転載、複製については、文書による三菱電機の事前の承諾が必要です。
- ・ 本資料に関し詳細についてのお問い合わせ、その他お気づきの点がございましたら三菱電機または特約店までご照会ください。