R01AN0769JJ0101
Rev.1.01
2012.06.15

SH7216グループ

MMC を使用したユーザプログラムモード Flash 書き換えによるプログラムアップデート例

要旨

本アプリケーションノートは、SH7216 ユーザプログラムモードでのフラッシュメモリ（内蔵フラッシュ）書き換えによるプログラムアップデート例について説明します。

本アプリケーションノートで実現するプログラムアップデート例の特長を以下に示します。

- MMC に保存したモトローラ S フォーマット形式のアップデート用プログラムファイルデータを使用して、内蔵フラッシュ領域のプログラムを書き換えます。
- 書き換え（アップデート）後に再起動し、アップデートした新しいプログラムを実行します。
- 意図せず書き換えが中断するなど正常に書き換えできなかった場合の対策として、チェックサムによるエラー制御を実装しています。

対象デバイス

SH7216

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 仕様

本応用例では、ユーザプログラムモードを使用して、内蔵フラッシュ書き換えによるプログラムアップデートを行います。

MMC に保存したモトローラ S フォーマット形式のアップデート用プログラムファイルデータを読み出して、内蔵フラッシュ領域のプログラムを書き換えます。書き換え後はウォッチドッグタイマによるリセットを行い、書き換え後の新しいプログラムを実行します。

表 1に本応用例で使用する周辺機能と用途を、また 図 1にシステム構成図を示します。

表 1　本応用例で使用する周辺機能と用途

周辺機能	用途
フラッシュメモリ（内蔵フラッシュ）	プログラム格納領域
内蔵フラッシュ専用シーケンサ（FCU）	内蔵フラッシュ書き換え
FIFO 内蔵シリアルコミュニケーションインタフェース（SCIF）	メッセージ通信
ルネサスシリアルペリフェラルインタフェース（RSPI）	MMC からの読み出し
ウォッチドッグタイマ（WDT）	書き換え後のシステム再起動

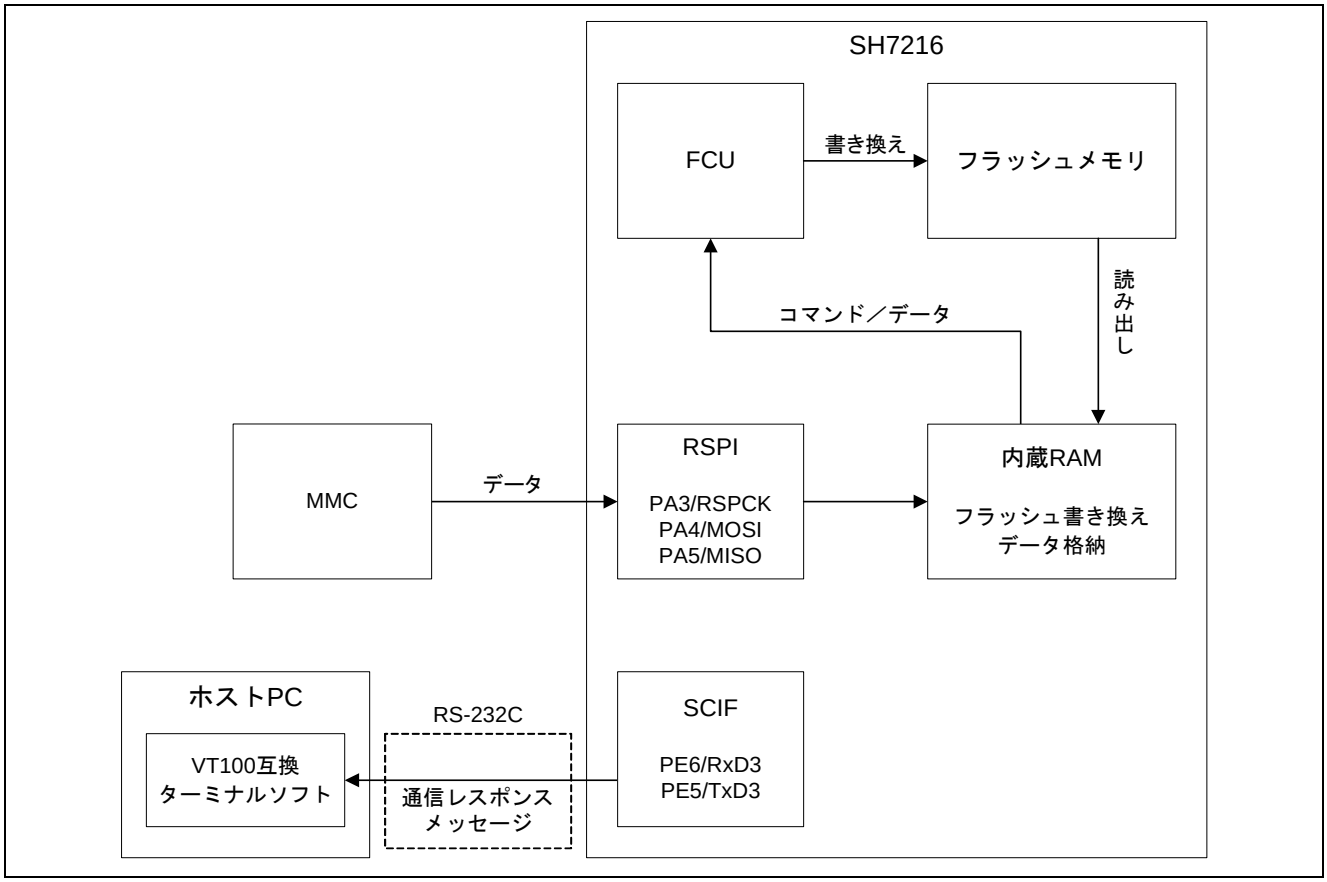


図 1　システム構成図

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2　動作確認条件

項目	内容
使用マイコン	SH7216
動作周波数	内部クロック（Iφ）：200MHz バスクロック（Bφ）：50MHz 周辺クロック（Pφ）：50MHz
動作電圧	3.3V（V_{CC}）
統合開発環境	ルネサス エレクトロニクス製 High-performance Embedded Workshop Ver.4.08.00
C コンパイラ	ルネサス エレクトロニクス製 SuperH RISC engine ファミリ C/C++コンパイラパッケージ Ver.9.03 Release 00
	コンパイルオプション -cpu=sh2afpu -fpu=single -include="$(WORKSPDIR)¥inc","$(WORKSPDIR)¥src¥mmc" -object="$(CONFIGDIR)¥$(FILELEAF).obj" -debug -ifunc -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo
動作モード	ユーザプログラムモード
ターミナルソフトの通信設定	• 9600bps • データ長 8 ビット • パリティ無し • 1 ストップビット • フロー制御無し
サンプルコードのバージョン	1.00
使用ボード	R0K572167C001BR（CPU ボード）* R0K572167B000BR（拡張ボード） 【注】 RS-232C コネクタ（J8）を使用するために JN2-20pin と JP2-4pin、JN2-21pin と JP3-4pin をジャンパー線などで接続する必要があります。
使用ツール	VT100 互換ターミナルソフト
使用メディア	HITACHI 社製 MultiMediaCard™（32MB）

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- SH7216 グループ　初期設定例（RJJ06B1073）
- SH ファミリ　SH-2, SH-2A 用シンプルフラッシュ API（R01AN0285JJ）
- SH7216 グループ　ユーザプログラムモード Flash 書き換えによるプログラムアップデート例（R01AN0686JJ）
- SH7216 グループ　ルネサスシリアルペリフェラルインタフェース マルチメディアカードのアクセス例（R01AN0039JJ）

4. 周辺機能説明

本アプリケーションノートで使用する周辺機能を紹介します。基本的な内容はハードウェアマニュアルに記載しています。

4.1 フラッシュメモリ

SH-2, SH-2A 用シンプルフラッシュ API を使用し、MMC から読み出したデータを内蔵フラッシュに書き込みます。

SH7216 は、品種別に 1M バイト／768K バイト／512K バイトのコード格納用フラッシュメモリ（ユーザマット）を内蔵しています。

ユーザマットは 8K バイト、64K バイト、128K バイトのブロックに分割されています。ユーザプログラムモードでユーザマットを消去する場合、このブロック単位で行います。

内蔵フラッシュへの書き込みは、消去状態の領域に対してのみ可能で、256 バイト単位で行います。また、書き込み開始アドレスには 256 バイト境界のアドレスを使用します。

本応用例では、ブロック 04（H'0000 8000～H'0000 9FFF）をアップデート用の書き換え領域とし、同時に予備のプログラムをブロック 05（H'0000 A000～H'0000 BFFF）に格納しています。また、ブロック 04 の最後尾 256 バイト領域（H'0000 9F00～H'0000 9FFF）を、チェックサム判定用の領域として使用しています。

これ以降、消去単位のブロックは "EB" と称します。（例えば、ブロック 04 の場合は "EB04" と表記します。）

図 2にユーザマットのブロック構成を示します。

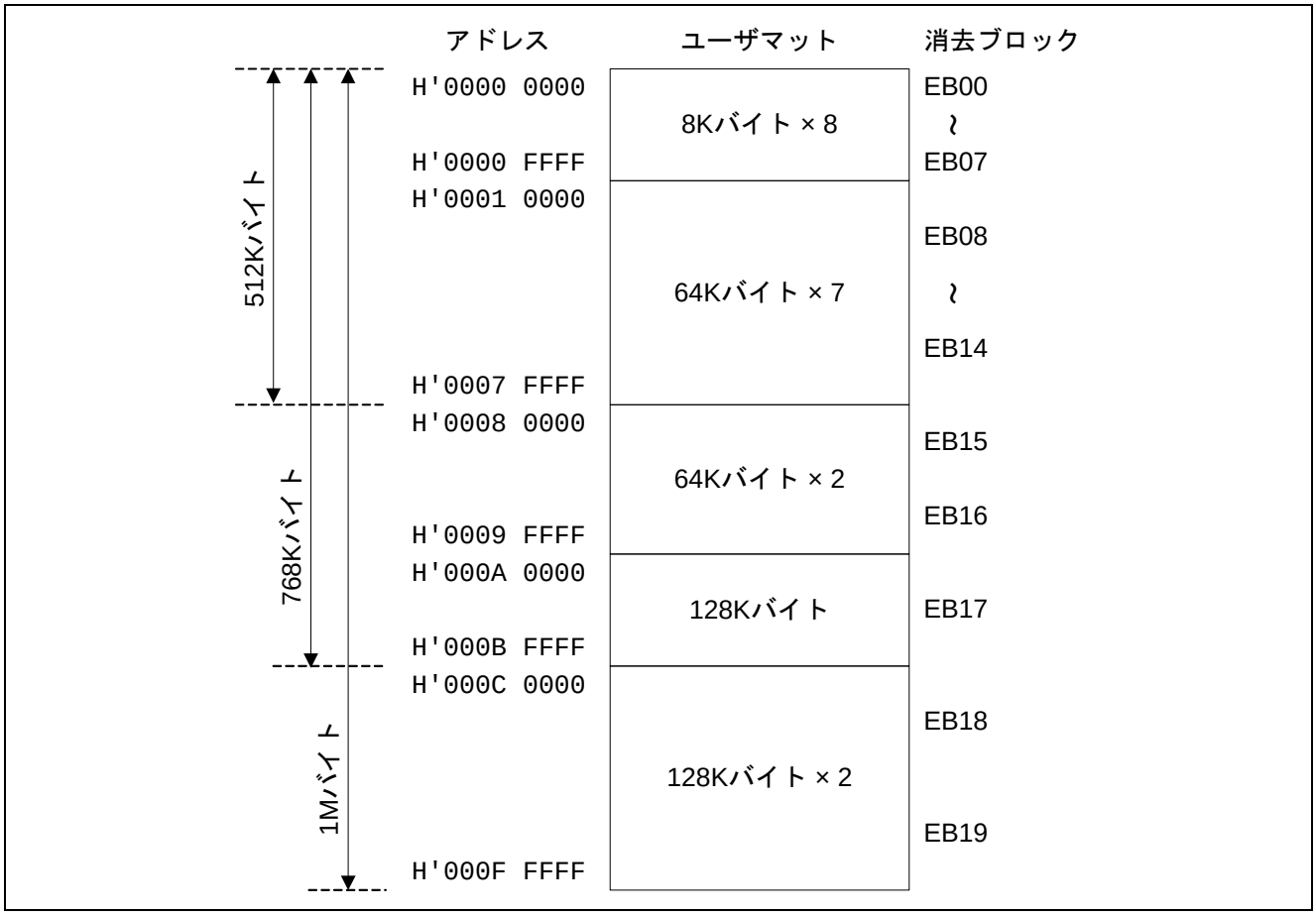


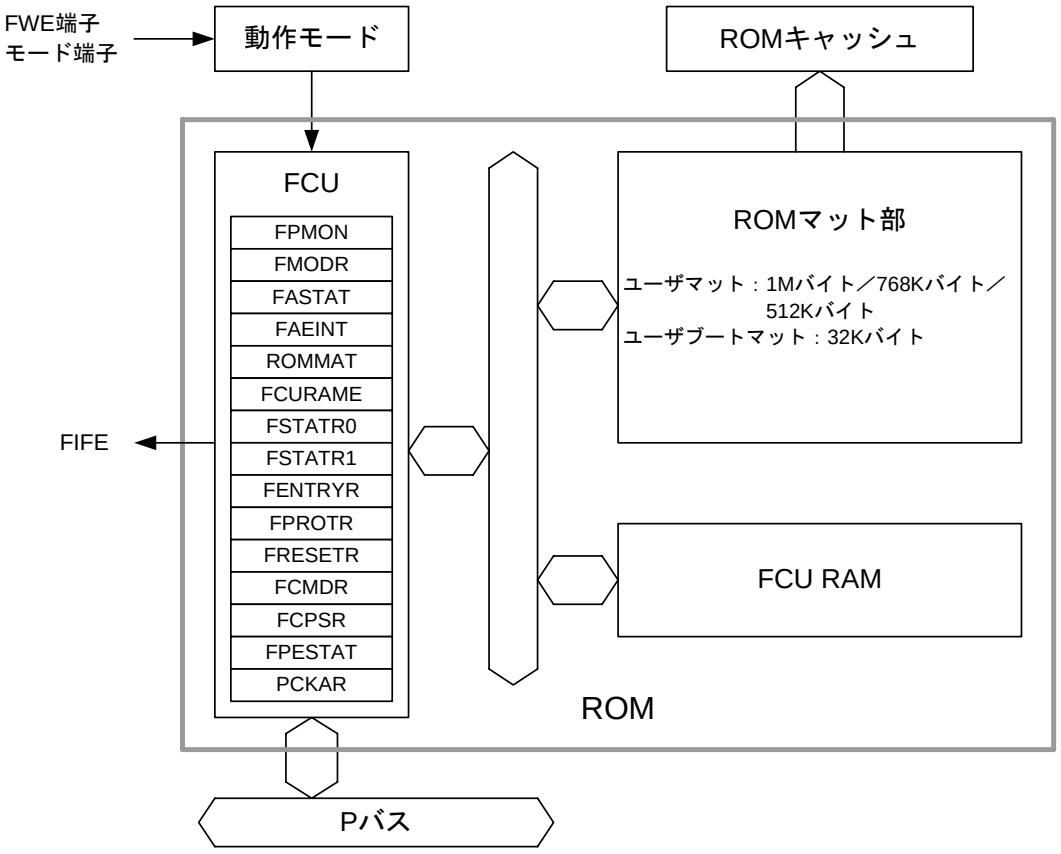
図 2　ユーザマットのブロック構成

4.2　内蔵フラッシュ専用シーケンサ（FCU）

SH-2, SH-2A 用シンプルフラッシュ API がデータの書き込み／消去のために使用します。

FCU を使用するためには、FCU 用のファームウェア（FCU ファーム）を FCURAM 領域に転送する必要があります。FCU ファーム転送後、FCU コマンドを発行することで、内蔵フラッシュへの書き込み／消去を実行することができます。

図 3に内蔵フラッシュのブロック図を示します。



【記号説明】

FPMON：	フラッシュ端子モニタレジスタ
FMODR：	フラッシュモードレジスタ
FASTAT：	フラッシュアクセスステータスレジスタ
FAEINT：	フラッシュアクセスエラー割り込み許可レジスタ
ROMMAT：	ROMマット選択レジスタ
FCURAME：	FCU RAMイネーブルレジスタ
FSTATR0、1：	フラッシュステータスレジスタ0、1
FENTRYR：	フラッシュP/Eモードエントリレジスタ
FPROTR：	フラッシュプロテクトレジスタ
FRESETR：	フラッシュリセットレジスタ
FCMDR：	FCUコマンドレジスタ
FCPSR：	FCU処理切り替えレジスタ
FPESTAT：	フラッシュP/Eステータスレジスタ
PCKAR：	周辺クロック通知レジスタ
FIFE：	フラッシュインタフェースエラー割り込み

図 3　内蔵フラッシュのブロック図

4.3　FIFO 内蔵シリアルコミュニケーションインタフェース（SCIF）

SCIF は、調歩同期式とクロック同期式の 2 方式のシリアル通信が可能なモジュールです。また、各チャネルとも独立に送信／受信用に 16 段の FIFO レジスタを内蔵し、効率的かつ高速な連続通信を可能にしています。

詳細は「SH7214 グループ、SH7216 グループ　ユーザーズマニュアル ハードウェア編　17. FIFO 内蔵シリアルコミュニケーションインタフェース（SCIF）」を参照してください。

4.4　ウォッチドッグタイマ（WDT）

本応用例では、サンプルコードのフラッシュ書き換え終了後のリセットにウォッチドッグタイマを使用しています。

詳細は「SH7214 グループ、SH7216 グループ　ユーザーズマニュアル ハードウェア編　15. ウォッチドッグタイマ（WDT）」を参照してください。

4.5　ルネサスシリアルペリフェラルインタフェース（RSPI）

本応用例では、SPI 通信を使用して MMC にアクセスし、SPI 通信は RSPI で制御します。RSPI は、全二重同期式のシリアル通信ができるモジュールであり、複数のプロセッサや周辺デバイスとの高速なシリアル通信機能を備えています。

詳細は「SH7214 グループ、SH7216 グループ　ユーザーズマニュアル ハードウェア編　18. ルネサスシリアルペリフェラルインタフェース（RSPI）」を参照してください。

5. ハードウェア説明

5.1 使用端子一覧

表 3に使用端子と機能の一覧を示します。

表 3　使用端子と機能

端子名	入出力	内容
PA3/RSPCK	出力	RSPI のクロック出力
PA4/MOSI	出力	RSPI のマスタ送出データ
PA5/MISO	入力	RSPI のスレーブ送出データ
PE5/TxD3	出力	SCIF のシリアルデータ出力
PE6/RxD3	入力	SCIF のシリアルデータ入力

6. ソフトウェア説明

6.1 動作概要

本応用例では、MMC から読み出したモトローラ S フォーマット形式のアップデート用プログラムファイルデータを使用して内蔵フラッシュ領域のプログラムを書き換えます。ここでは、その動作概要について説明します。

6.1.1 セクション配置設定

まず、内蔵フラッシュ書き換え中は内蔵フラッシュへのアクセスが禁止されているため、内蔵フラッシュ書き換え中に使用するプログラムは全て内蔵フラッシュ以外の領域に転送する必要があります。本応用例では、内蔵フラッシュ書き換え中に使用するプログラムは全て内蔵 RAM に転送するよう、セクション配置を設定しています。なお、セクション配置設定の詳細は、関連アプリケーションノート「SH-2, SH-2A 用シンプルフラッシュ API　3.4　SH7216 グループ／SH7239 グループ／SH7231 グループ」を参照してください。

表 4に、本応用例にて内蔵フラッシュ書き換え中に使用するプログラムとセクションを示します。各プログラム処理内容の詳細は、「6.6　関数一覧」、「6.7　関数仕様」、および「6.8　フローチャート」を参照してください。

表 4　内蔵フラッシュ書き換え中に使用するプログラムとセクション

プログラム	関数名	ROM セクション名	RAM セクション名
フラッシュ書き換え処理	rom_write Enter_PE_Mode Exit_PE_Mode R_FlashErase R_FlashWrite	PFRAM	RPFRAM

また、本応用例では、意図せず書き換え処理が中断するなど正常に書き換え（アップデート）できなかった場合の対策として、予備プログラム格納用のセクション領域を別途割り当てています。データ受信前（初期）の書き換え領域と予備領域には、それぞれ同じ処理内容のプログラムを格納しています。表 5にその対応を示します。

表 5　書き換え領域と予備領域の対応

項目	先頭番地（ブロック）	格納関数名	ROM セクション名
書き換え領域	H’0000 8000（EB04）	flash_write_sample	MasterPRG
予備領域	H’0000 A000（EB05）	flash_write_spare	SparePRG

6.1.2　内蔵フラッシュ書き換え動作概要

図 4に、本応用例における内蔵フラッシュ書き換え動作概要図を示します。

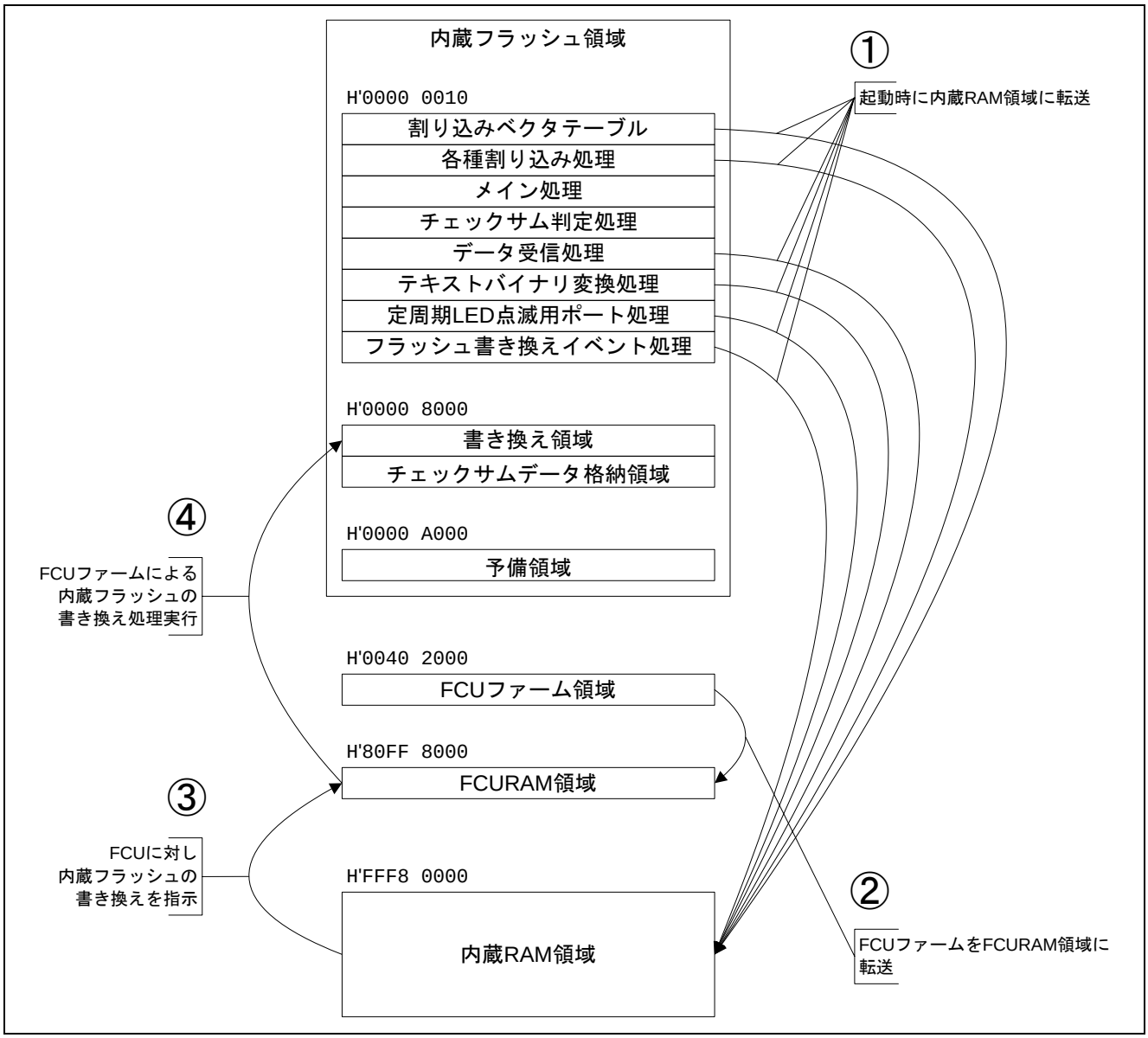


図 4　内蔵フラッシュ書き換え動作概要図

① リセット解除後、dbsct.c で指定したプログラム（ROM セクション）を内蔵 RAM 領域に転送します。

② 内蔵フラッシュの書き換えに使用する内蔵フラッシュ専用シーケンサ（FCU）のファームを FCU ファーム領域から FCURAM 領域に転送します。

③ FCURAM 領域に転送した FCU ファームに対し、内蔵フラッシュ書き換え処理の実行を指示します。（本応用例では、シンプルフラッシュ API をコールすることで、FCU ファームに対し書き換え処理の実行を指示しています。）

④ FCU ファームにより、内蔵フラッシュの書き換え処理が実行されます。

6.1.3　起動から通常動作まで

システム起動後、メイン処理にて各種初期化処理を行った後、ホスト PC に対し "Generate IRQ6 interrupt for transition to flash programming event." というメッセージを送信します。そして、「チェックサム判定処理」関数をコールして、書き換え領域（EB04）のプログラムコードに問題がないか、チェックサムにより判定します。

本応用例のチェックサムは、「プログラムコードサイズ」、およびプログラムを 1 バイトずつ加算した「チェックサムデータ」の 2 つを使用します。チェックサム判定処理関数では、書き換え領域の先頭番地（H'0000 8000）から 1 バイトずつ、プログラムサイズの回数分加算します。その加算結果を、データ受信時に算出したチェックサム判定データ（EB04 の最後尾 256 バイト領域に格納／詳細は次節参照）と比較し、一致していれば書き換え領域のプログラムを、一致していなければ予備領域のプログラムを実行します。（ただし、初回起動時のチェックサムは必ず一致するように、あらかじめダミーのチェックサム判定データを格納しています。）

初回起動時に格納しているプログラムはコンペアマッチタイマ（CMT）を使用して LED を点滅させるもので、100ms 周期で発生するコンペアマッチ割り込み内で「定周期 LED 点滅用ポート処理」関数をコールして、LED の点滅表示パターンを更新しています。

6.1.4　IRQ6 入力によるフラッシュ書き換えイベント処理

ホスト PC に送信したメッセージにしたがって、IRQ6 入力（立ち下がりエッジ検出／ボード上の IRQ6 スイッチ押下）により IRQ6 割り込みが発生すると、フラッシュ書き換えイベント処理に移行します。

フラッシュ書き換えイベント処理では、最初にホスト PC に "--> IRQ6 detected!" というメッセージを送信し、書き換え領域の EB04 を消去します。その後、ホスト PC に "Send subroutine code to update program in Motorola S-record format. Check card insert" というメッセージを送信し、MMC の挿入を待ちます。

MMC の挿入を検出すると、ルートディレクトリに保存されているモトローラ S フォーマット形式のアップデート用プログラムファイルデータを読み出します。セクタ単位で読み出した後、1 バイトずつ後述のデータ解析処理を行い、書き込みデータ格納バッファ（書き込みバッファ）にデータを格納します。1 セクタ分の解析が終わると書き込みバッファのデータを内蔵フラッシュに書き込み、最終データを内蔵フラッシュに書き込むまでこの処理を繰り返します。なお、書き込みバッファは二重構造になっており、内蔵フラッシュへの書き込みはバッファ単位で行っています。

6.1.5　MMC に保存したアップデート用プログラムファイルの読み出し

FAT ファイルシステムでフォーマットした MMC から規定ファイル（a.mot）を検索して読み出します。ただし、本応用例のサンプルコードは FAT ライブラリを実装していないため、以下の制限事項があります。

表 6　MMC に関する制限事項

項目	内容	説明
FAT タイプ	FAT16	FAT12 および FAT32 には対応していません。
セクタサイズ	512 バイト	
ファイル名	a.mot	アップデート用プログラムファイルは必ずこの名称で保存してください。
ファイルパス	H:¥¥a.mot	“H”はドライブ番号のためホスト PC の環境に応じて異なります。ルートディレクトリにファイルを保存してください。
ファイルサイズ	1 クラスタサイズ以内	複数クラスタにまたがるファイルは読み出せません。
ルートディレクトリの検索対象エントリ	1 セクタ内	ルートディレクトリの先頭セクタしか検索しないため、アップデート用プログラムファイル以外にもファイルが多数保存されていると検索が正しく行われない可能性があります。

6.1.6　データ解析処理

MMC から 1 セクタ分のデータを読み出すと、1 バイトずつ解析用バッファに格納します。改行コードを検出すると、それまで解析用バッファに格納したデータを 1 行分のレコードデータと判断し、以下に説明するデータ解析処理を行って、アップデートに必要な書き込みデータを抽出します。

まず､解析用バッファ内の 1 文字目のデータが "S" であるかを判定し､"S" ならばモトローラ S フォーマット形式とみなして 2 文字目の判定を行います。1 文字目が "S" でなければ、そのレコードデータは無効となり、再び解析用バッファの先頭から次のデータを格納します。

以下、図 5に示すモトローラSフォーマット形式データ例を参考に、データ解析処理について説明します。（図 5に示すデータは、機能によって色分けしています。）

```
S00E000073616D706C6520206D6F74DF
S1138000 7FFC04E0420AE50001E00014E725664332
S1138010 76F6E065E23F315903F4345100762670 78
S1138020 8061242084642F62C90FCB30806466F2 9F
S1138030 8465C90FCB30806506E042008466C90F B1
S1138040 CB3080668466C9F0CB0380668467C90F 31
S1138050 CB3080678467C9F0CB0380670000BEBB 68
S1138060 816C00005F5D816D06E0420000005F5D 91
                    ≀
S10B82C0 00090009000B7F0412
S9030000FC
```

図 5　モトローラ S フォーマット形式データ例

- 1 行目は 2 文字目（赤）が "0" になっています。この "0" はヘッダレコードを示します。ヘッダレコードにはプログラムデータはありませんので、ヘッダレコードを読み出した場合は次のレコード（2 行目）が揃うまで、再び解析データ待ち状態に戻ります。
- 2 行目のレコードデータが揃うと、1 文字目（黒）の "S" でモトローラ S フォーマット形式と判定し、2 文字目（赤）の "1"で、2 行目はデータレコードと判定します。このようにモトローラ S フォーマット形式では、各レコードの先頭から 2 文字目（赤）の数値により、レコードの種類が判別できるようになっています。
- レコードの 3 文字目と 4 文字目（青）はレコードサイズを示す 1 バイト分の 16 進数を示し、5 文字目から 8 文字目（緑）までの 4 文字は、レコードの先頭データ格納アドレスの下位 2 バイトを示します。
- レコードの 9 文字目（橙）以降が、それぞれ 2 文字単位で 1 バイトを示すデータ部になっています。データ解析処理では、この 9 文字目（橙）以降を 2 文字単位でバイナリデータに変換（「テキストバイナリ変換処理」関数をコール）し、変換後の 1 バイトデータを書き込みバッファに順次格納します。また、このとき、書き換え後のチェックサム判定用にその 1 バイトデータを加算（チェックサムデータ）し、さらにそのデータ数をプログラムコードサイズとしてカウントします。これらの処理をレコードの最後の 2 文字（黒）手前まで繰り返したら、データ解析処理を終了します。
- レコードデータの 2 文字目が "9" の場合は終了レコードを示します。（図 5では一番下の行に相当します。）終了レコードを判定したら、解析データの格納処理は行わずにデータ解析処理を終了します。ただし、この時点で書き込みバッファ内のデータサイズが（フラッシュ書き込み単位の）256 バイトに満たない場合、バッファサイズが 256 バイトになるようにH'FFを付加します。

本応用例では、書き込みバッファを 256 バイトサイズの二重構造としており、書き込みバッファ内の格納データが 256 バイトで満杯になるたびに、データ解析処理の中で格納先をもう一方の書き込みバッファに切り替えます。一方の書き込みバッファが満杯になるとフラッシュ書き込み可能状態となり、データ解析処理終了後にそのバッファデータを内蔵フラッシュに書き込みます。

6.1.7　データ解析／書き換え後の処理

データ解析処理にて終了レコードを判定し、全てのデータを内蔵フラッシュに書き込むと、次はデータ解析時に算出したチェックサム判定用のデータ（プログラムコードサイズおよびチェックサムデータ／それぞれ 2 バイト分）を内蔵フラッシュに書き込みます。本応用例では、チェックサム用のデータを、書き換え領域（EB04）の最後尾 256 バイト領域（H'0000 7F00～H'0000 7FFF）に格納します。（ただし、実際に使用するのはこの内 4 バイト分で、H'0000 9F00～H'0000 9F01 にプログラムコードサイズを、H'0000 9F02～H'0000 9F03 にチェックサムデータをそれぞれ格納します。）

チェックサム判定用のデータ書き込み後はウォッチドッグタイマを設定し、タイマカウンタオーバフローによるリセット待ち状態となります。

6.1.8 通信制御シーケンス

図 6に、本応用例における通信制御シーケンスを示します。

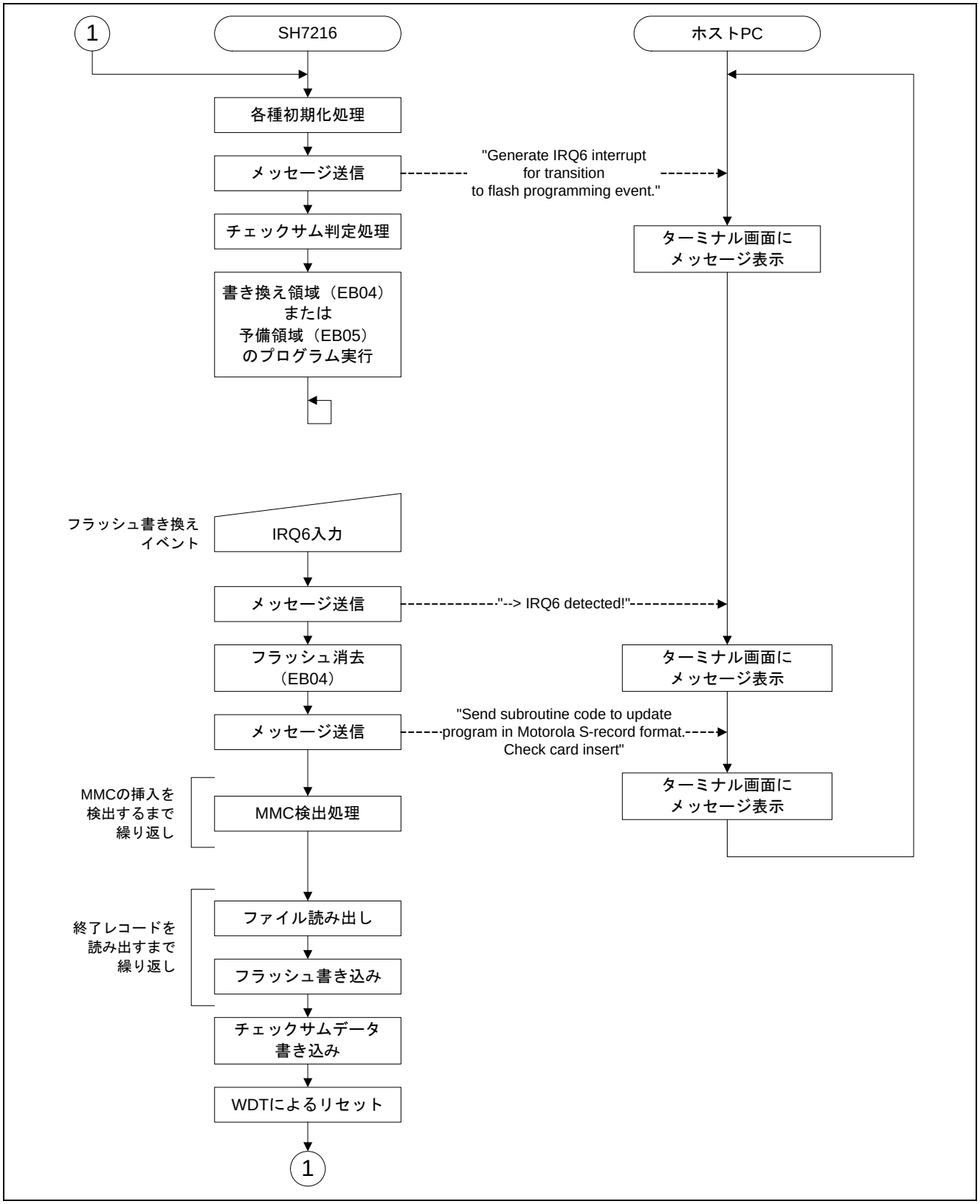


図 6　通信制御シーケンス

6.2　ファイル構成

表 7にファイル構成を示します。なお、統合開発環境で自動生成されるファイルは除きます。

表 7　ファイル構成

ファイル名	概要	備考
main.c	メイン処理、チェックサム判定処理、フラッシュ書き換えイベント処理、データ受信処理、定周期 LED 点滅用ポート処理	割り込みでコールされる関数も配置
intprg.c	各種割り込み処理	IRQ6 入力、コンペアマッチによりそれぞれの割り込み処理関数をコール
vecttbl.c	割り込みベクタテーブル	—
Flash_API_SH7216.c	シンプルフラッシュ API	FCU ファームによる内蔵フラッシュの書き換えに使用
Flash_API_SH7216.h	シンプルフラッシュ API 用ヘッダファイル	同上
siochar.c lowsrc.c	SCIF 制御処理	RS-232C 通信に使用
port_LED_sample.c	アップデート用サンプルプログラム	書き換え領域と予備領域に格納する初期プログラム
mmc_api.c	MMC ドライバ API	MMC ドライバを使用するための API を配置
mmc.h	MMC ドライバ用ヘッダファイル	同上
mmc_cmd.c	MMC のコマンド処理	MMC のコマンド処理を行うための関数を配置
mmc_cmd.h	MMC のコマンド処理用ヘッダファイル	同上
mmc_spi_bus.c	MMC の SPI モード処理	MMC を SPI モードで制御するための関数を配置
mmc_spi.h	MMC の SPI モード処理用ヘッダファイル	同上
mmc_time.c	ソフトウェアタイマ処理	—
io_rspi_sh7216.c	RSPI 制御処理	SPI 通信を行うための関数を配置

6.3　定数一覧

表 8にサンプルコードで使用する定数を示します。

表 8　サンプルコードで使用する定数

定数名	設定値	内容
SLOT0	0	MMC のスロット番号
BLOCK_4	4	書き換え領域のブロック番号
MAX_SCT_SIZE	512	MMC のセクタデータ読み出し用バッファサイズ
BUFFER_SIZE	256	フラッシュデータ書き込み用バッファサイズ
LED_PATTERN_NUM	10	LED 点滅表示パターン数

6.4　構造体/共用体一覧

図 7エラー! 参照元が見つかりません。にサンプルコードで使用する構造体/共用体一覧を示します。

```
/* ---- FAT 情報管理用構造体 ---- */
typedef struct{
        unsigned char       fat_type;       /* 4/6:FAT16, 11:FAT32 */
        unsigned long       mmc_bsr_area;   /* BSR領域の先頭セクタ */
        unsigned long       mmc_fat_area;   /* FAT領域の先頭セクタ*/
        unsigned long       mmc_rdir_area;  /* ルートディレクトリの先頭セクタ */
        unsigned long       mmc_data_area;  /* データ領域の先頭セクタ */
        unsigned short      sctsz;          /* セクタサイズ（バイト数） */
        unsigned char       clussz;         /* クラスタサイズ（セクタ数） */
        unsigned short      fatsz;          /* FAT領域のサイズ（セクタ数） */
        unsigned char       fatnum;         /* FAT領域の個数 */
        unsigned short      rdirnum;        /* ルートディレクトリのエントリ数 */
        unsigned char       f_entry[32];    /* ターゲットファイルのエントリ情報 */
        unsigned short      f_clusno;       /* ターゲットフェイルの先頭クラスタ番号 */
        unsigned long       f_filesz;       /* ターゲットファイルのファイルサイズ */
}MMC_FSYS;
```

図 7　構造体/共用体一覧

6.5　変数一覧

表 9にグローバル変数を、表 10にconst型変数を示します。

表 9　グローバル変数

型	変数名	内容	使用関数
MMC_FSYS	mmc_fsys	読み出した FAT ファイルシステムの情報を保存する構造体	int_fcu_flash_write smpl_fopen search_file
unsigned char	mmc_ReadDataBuff0	解析データ格納バッファ 0（配列）	analyze_read_data
unsigned char	mmc_ReadDataBuff1	解析データ格納バッファ 1（配列）	analyze_read_data
unsigned char	sct_buff	MMC のセクタ読み出しバッファ（配列）	int_fcu_flash_write smpl_fopen
int	f_WriteDataBuff0_Full	書き込みバッファ 0 フルフラグ	analyze_read_data int_fcu_flash_write
int	f_WriteDataBuff1_Full	書き込みバッファ 1 フルフラグ	analyze_read_data int_fcu_flash_write
int	f_status_EndRecord	終了レコード読み出しフラグ	analyze_read_data int_fcu_flash_write
int	cnt_store_ReadData	解析データ格納カウンタ	analyze_read_data
int	cnt_store_WriteDataBuff	書き込みデータ格納カウンタ	analyze_read_data
int	cnt_led_wink	LED 点滅表示パターンカウンタ	int_cmt_led_control
unsigned short	chksm_size	書き込みデータのプログラムコードサイズ	analyze_read_data int_fcu_flash_write
unsigned short	chksm_data	書き込みデータのチェックサムデータ	analyze_read_data int_fcu_flash_write
unsigned short	chksm_Mem	初回起動時の書き換え領域チェックサム判定用ダミーデータ（配列）	—
unsigned short	cmt_LedPattern	初回起動時の書き換え領域または予備領域格納プログラム向け LED 点滅表示パターンデータ	int_cmt_led_control

表 10　const 型変数

型	変数名	内容	使用関数
const char	msg_VT100_ClearScreen	VT100 互換 ESC シーケンス	main
const char	msg_StartComment	IRQ6 入力要求メッセージ	main
const char	msg_IRQ6_Detected	IRQ6 入力検出メッセージ	int_fcu_flash_write
const char	msg_Motorola_format	モトローラ S フォーマット形式ファイル送信要求メッセージ	int_fcu_flash_write
const char	msg_MMC_Attach	MMC カード挿入要求メッセージ	int_fcu_flash_write

6.6　関数一覧

表 11に関数一覧を示します。

表 11　関数

関数名	概要
main	メイン処理
io_init_irq6	IRQ6 初期化処理
check_sum_check	書き換え領域のチェックサム判定処理
hex2bin	テキストバイナリ変換処理
INT_IRQ6	IRQ6 割り込み処理
INT_CMT_CMI0	CMT（チャネル 0）コンペアマッチ割り込み処理
analyze_read_data	データ解析処理
int_cmt_led_control	定周期 LED 点滅用ポート処理
int_fcu_flash_write	フラッシュ書き換えイベント処理
smpl_fopen	MMC のファイル情報取得処理
search_file	ディレクトリエントリ検索処理
swapl	4 バイトスワップ処理
swapw	2 バイトスワップ処理
io_output_msg	メッセージ出力処理
flash_write_sample*	100ms 定周期タイマ用 CMT 初期化処理
flash_write_spare	flash_write_sample* と同一のプログラム（予備領域に格納）
mmc_init_driver	（MMC ドライバ）MMC 初期化処理
mmc_attach	（MMC ドライバ）アタッチ確認処理
mmc_read_data	（MMC ドライバ）データリード処理
R_FlashErase	（シンプルフラッシュ API）イレース処理
R_FlashWrite	（シンプルフラッシュ API）データライト処理

【注】*　初回起動時の被アップデート用サンプルプログラム（書き換え領域に格納）。

6.7　関数仕様

サンプルコードの関数仕様を示します。

main

概要	メイン処理
ヘッダ	なし
宣言	void main(void)
説明	各種初期化処理を行った後、ホスト PC に対し IRQ6 入力要求メッセージを送信し、チェックサム判定処理を行います。チェックサムの判定結果により、書き換え領域または予備領域に配置したプログラムを実行します。
引数	なし
リターン値	なし
備考	

io_init_irq6

概要	IRQ6 初期化処理
ヘッダ	なし
宣言	void io_init_irq6(void)
説明	IRQ6 の初期化処理を行います。PA20 の端子機能を IRQ6 入力に設定した後、割り込みコントローラにて割り込み要求を IRQ6 入力の立ち下がりエッジで検出するよう設定します。その後、IRQ6 の割り込み優先レベルを設定します。
引数	なし
リターン値	なし
備考	

check_sum_check

概要	書き換え領域のチェックサム判定処理
ヘッダ	なし
宣言	int check_sum_check(void)
説明	書き換え領域の最後尾 256 バイト領域（H'0000 9F00～H'0000 9FFF）に格納したプログラムコードサイズおよびチェックサムデータをもとに、書き換え領域の先頭番地（H'0000 8000）からサム値を算出し、チェックサムデータとの一致判定を行います。
引数	なし
リターン値	• 0：チェックサム一致 • 1：チェックサム不一致
備考	本応用例では、初回起動時のチェックサム判定が必ず一致するよう、書き換え領域の最後尾 256 バイト領域（H'0000 9F00～H'0000 9FFF）にあらかじめダミーのチェックサム判定データ（chksm_Mem）を格納しています。

hex2bin

概要	テキストバイナリ変換処理
ヘッダ	なし
宣言	int hex2bin(unsigned char upper, unsigned char lower)
説明	テキストデータ（2 文字）を 1 バイトのバイナリデータに変換します。 引数に与えたデータが "0"～"9" または "A"～"F" のテキストデータであれば有効データとみなし、H'0～H'F のバイナリデータに変換します。 第一引数（upper）の変換結果を 4 ビット左シフトし、第二引数（lower）の変換結果との論理和をとった後、1 バイトのバイナリデータとして返却します。
引数	• 第一引数：upper　上位 4 ビット用のテキストデータ • 第二引数：lower　下位 4 ビット用のテキストデータ
リターン値	• 0～255：1 バイトのバイナリデータ • -1：入力データ不正
備考	

INT_IRQ6

概要	IRQ6 割り込み処理
ヘッダ	なし
宣言	void INT_IRQ6(void)
説明	フラッシュ書き換えイベント処理（int_fcu_flash_write 関数）を実行します。
引数	なし
リターン値	なし
備考	

INT_CMT_CMI0

概要	CMT（チャネル 0）コンペアマッチ割り込み処理
ヘッダ	なし
宣言	void INT_CMT_CMI0(void)
説明	定周期 LED 点滅用ポート処理（int_cmt_led_control 関数）を実行します。
引数	なし
リターン値	なし
備考	サンプルコードで LED 点滅用に使用しています。

analyze_read_data

概要	データ解析処理	
ヘッダ	なし	
宣言	void analyze_read_data(unsigned char ch)	
説明	引数データを解析データ格納バッファ（解析用バッファ）に格納し、アップデートに必要な書き込みデータを抽出します。 改行コード（"¥r" または "¥n"）を受信したら、それまで解析用バッファに格納したデータを 1 行分のレコードデータと判定し、さらにそのレコードデータがモトローラ S フォーマット形式であれば、その中から書き込みデータを抽出します。 書き込みデータ抽出時は、テキスト形式の書き込みデータ部を 2 文字単位でバイナリデータ（1 バイト）に変換し、書き込みデータ格納バッファ（書き込みバッファ／二重構造）に格納します。書き込みバッファへのデータ格納サイズが 256 バイトになるごとに、格納する書き込みバッファを切り替えます。 また、フラッシュ書き込み後のチェックサム判定用に、バイナリ変換後の書き込みデータを加算し、さらに加算した書き込みデータ数をカウントします。 モトローラ S フォーマットの終了レコードを受信すると、その時点の書き込みバッファ内の格納データ数が 256 バイトに満たない場合、256 バイトサイズになるように H'FF を付加します。	
引数	unsigned char ch	解析するデータ（文字データ）
リターン値	なし	
備考		

int_cmt_led_control

概要	定周期 LED 点滅用ポート処理	
ヘッダ	なし	
宣言	void int_cmt_led_control(void)	
説明	CMT（チャネル 0）コンペアマッチ割り込み要因フラグをクリアし、定周期 LED 点滅用に PE9, PE11～15 のポート出力パターンを切り替えます。	
引数	なし	
リターン値	なし	
備考		

int_fcu_flash_write

概要	フラッシュ書き換えイベント処理
ヘッダ	なし
宣言	void int_fcu_flash_write(void)
説明	ホスト PC にメッセージをシリアル送信した後、書き換え領域（EB04）を消去します。 書き換え領域消去後は､MMC の挿入待ち状態になります｡MMC の挿入を検出するとアップデート用プログラムファイルを読み出し、データを解析します。 データ解析処理で書き込みデータ格納バッファが満杯になって書き込み可能であることを検出すると、そのバッファデータを内蔵フラッシュに書き込みます。また、データ解析処理にて終了レコードを検出し、内蔵フラッシュ書き込みが終了すると、次はデータ解析処理時に算出したチェックサム判定用のデータ（プログラムコードサイズおよびチェックサムデータ）を書き換え領域（EB04）の最後尾 256 バイト領域（H'0000 7F00～H'0000 7FFF）に書き込みます。 チェックサム判定用のデータ書き込み後、ウォッチドッグタイマを設定し、タイマカウンタオーバフローによるリセットを待ちます。
引数	なし
リターン値	なし
備考	

smpl_fopen

概要	アップデートプログラムのオープン	
ヘッダ	なし	
宣言	int smpl_fopen(const char *fname, const char *mode)	
説明	MMC の FAT ファイルシステム情報を検索して、アップデートプログラムが格納された領域を検出します。検出した情報は、構造体変数 mmc_fsys に格納します。	
引数	const char *fname const char *mode	ファイル名（ショートファイル名のみ対応） ファイルオープン指定（無効）
リターン値	-1 : エラー 0 ：正常終了	
備考		

search_file

概要	ディレクトリエントリの検索	
ヘッダ	なし	
宣言	int search_file(unsigned char *buff, unsigned char *fname)	
説明	ディレクトリエントリからアップデートプログラムを検索します。	
引数	unsigned char *buff unsigned char *fname	ディレクトリエントリをリードしてあるバッファ 検索するファイル名
リターン値	-1 ：エラー 0 以上 ：エントリ番号	
備考		

swapl, swapw

概要	エンディアン変換（ロングワード／ワード）	
ヘッダ	なし	
宣言	unsigned long swapl(unsigned char *addr) unsigned short swapw(unsigned char *addr)	
説明	リトルエンディアンデータをビッグエンディアンデータとしてリードします。	
引数	unsigned char *addr	変換前データの先頭アドレス
リターン値	変換したデータ	
備考		

io_output_msg

概要	メッセージ出力	
ヘッダ	なし	
宣言	void io_output_msg(char *msg_string)	
説明	メッセージをホスト PC に出力します。	
引数	char *msg_string	出力するメッセージ
リターン値	なし	
備考		

flash_write_sample, flash_write_spare

概要	100ms 定周期タイマ用 CMT 初期化処理
ヘッダ	なし
宣言	void flash_write_sample(void), void flash_write_spare(void)
説明	100ms 定周期タイマ用に CMT（チャネル 0）の初期化処理を行います。 また、定周期 LED 点滅用に PE9, PE11～15 の端子機能をポート出力に設定し、コンペアマッチ割り込みを許可に設定します。
引数	なし
リターン値	なし
備考	

mmc_init_driver, mmc_attach, mmc_read_data の詳細は、アプリケーションノート「SH7216 グループ　ルネサスシリアルペリフェラルインタフェース マルチメディアカードのアクセス例（R01AN0039JJ）」を参照してください。

R_FlashErase, R_FlashWrite の詳細は、アプリケーションノート「SH ファミリ　SH-2, SH-2A 用シンプルフラッシュ API（R01AN0285JJ）」を参照してください。

6.8 フローチャート

6.8.1 メイン処理

図 8にメイン処理のフローチャートを示します。

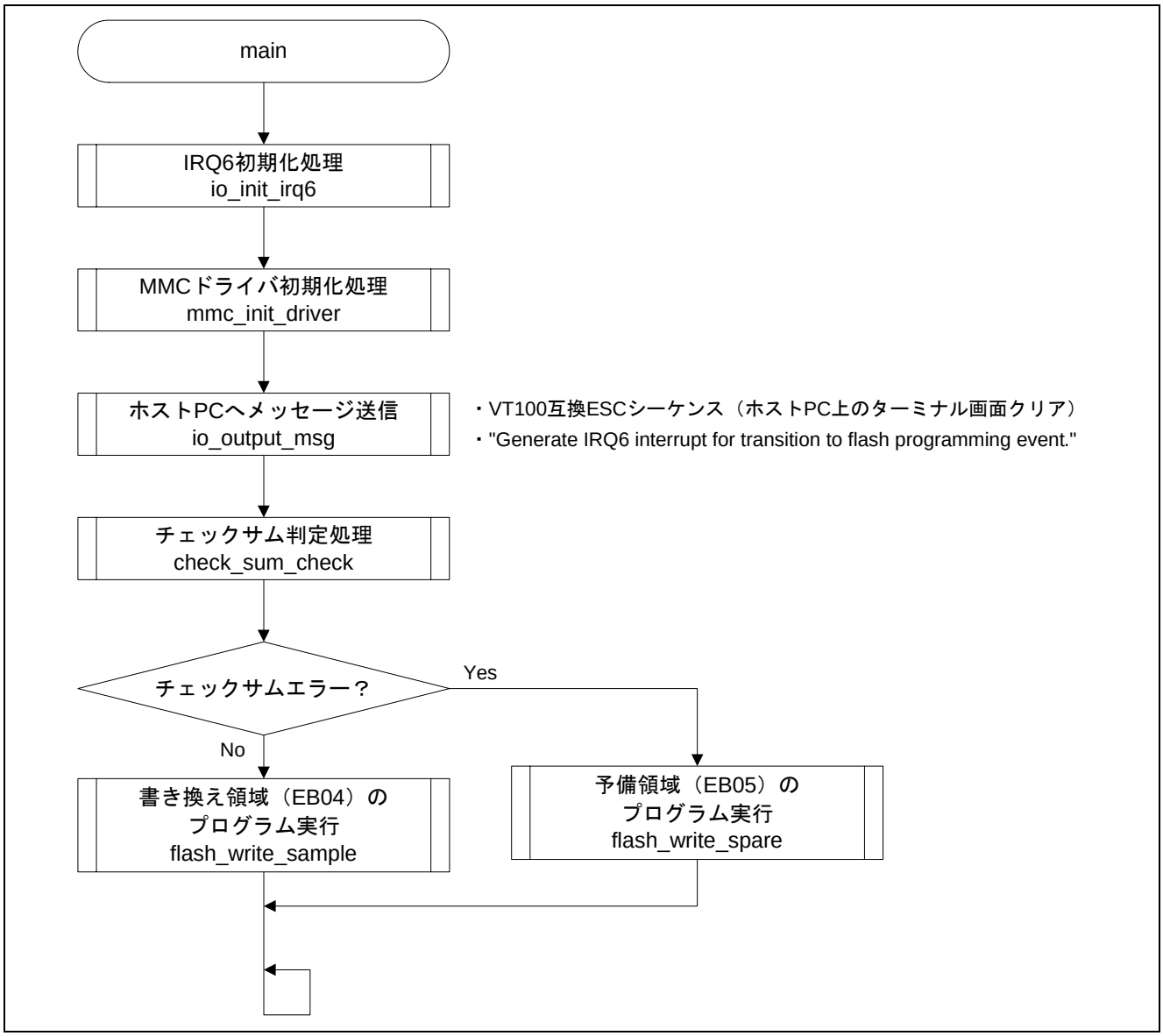


図 8　メイン処理フロー

6.8.2 フラッシュ書き換えイベント処理

図 9および図 10にフラッシュ書き換えイベント処理のフローチャートを示します。

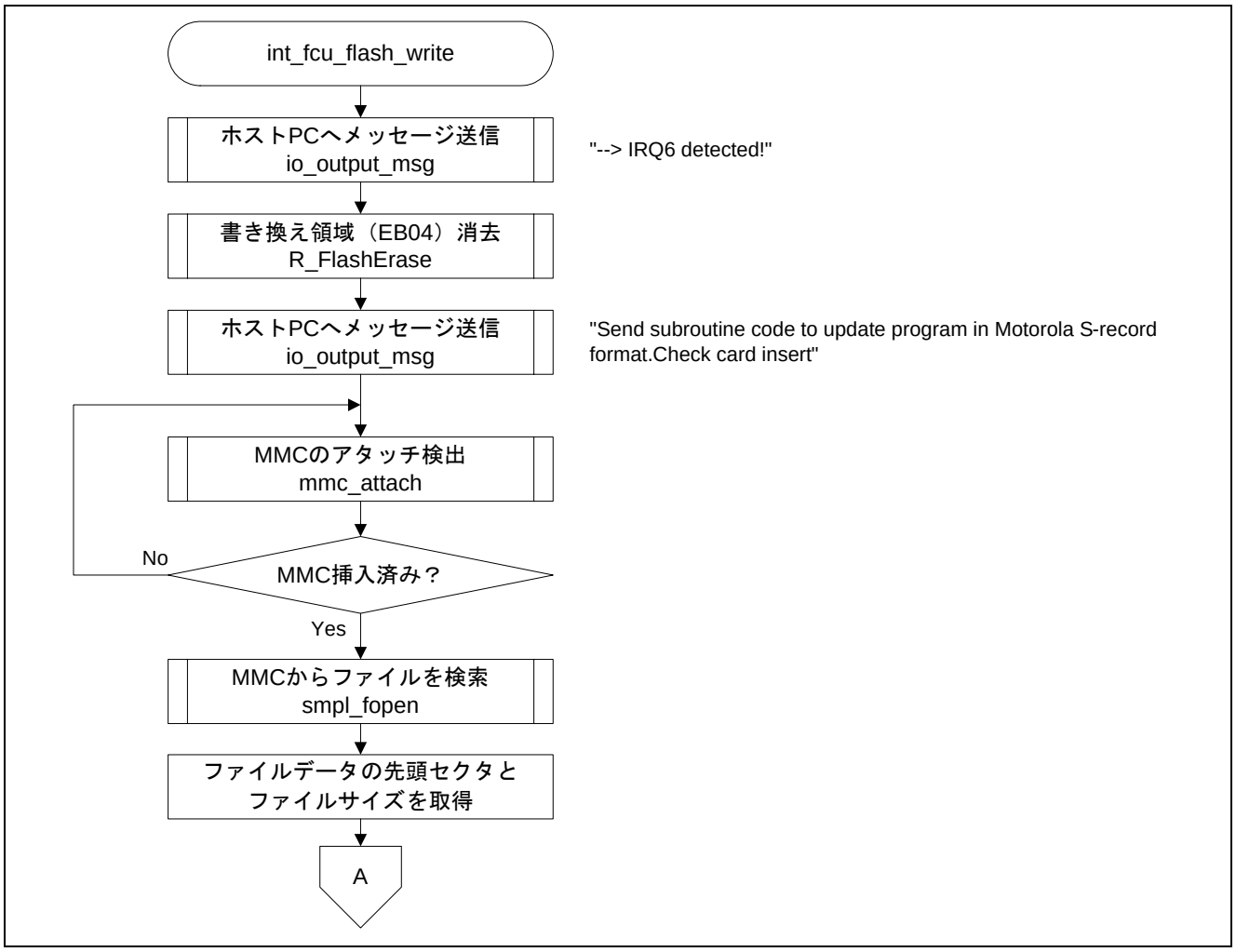


図 9　フラッシュ書き換えイベント処理フロー（1）

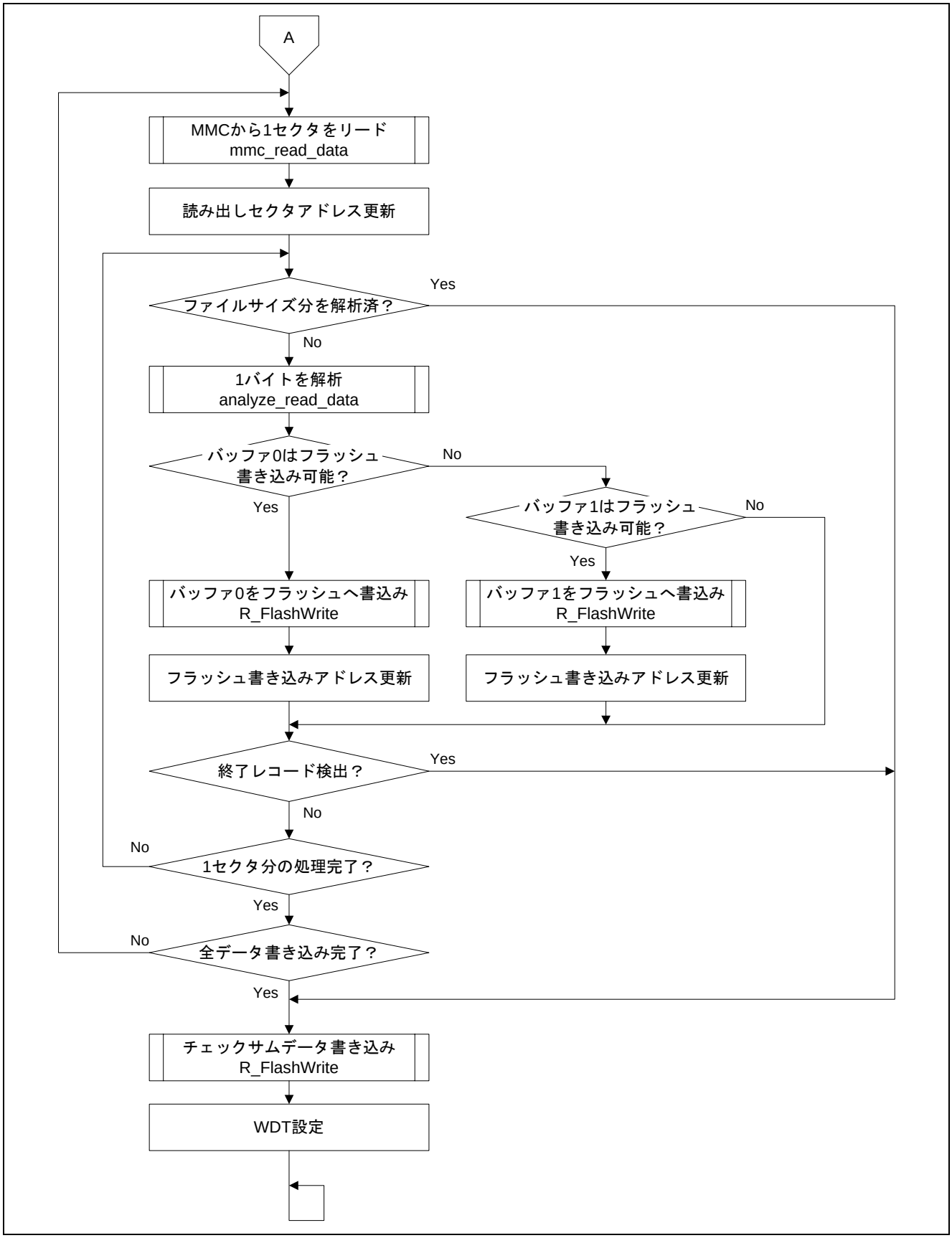


図 10　フラッシュ書き換えイベント処理フロー（2）

6.8.3 データ解析処理

図 11にデータ解析処理のフローチャートを示します。

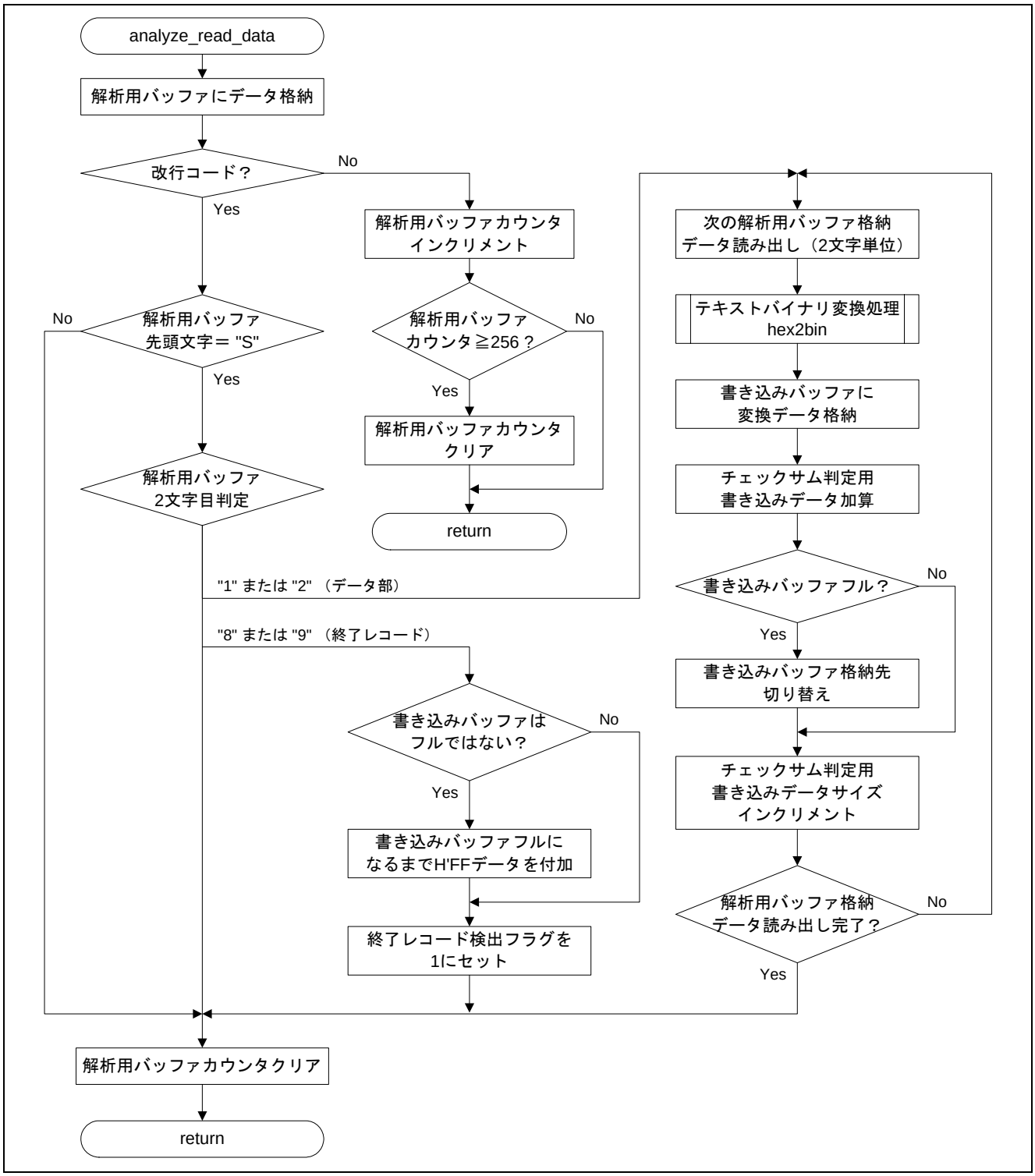
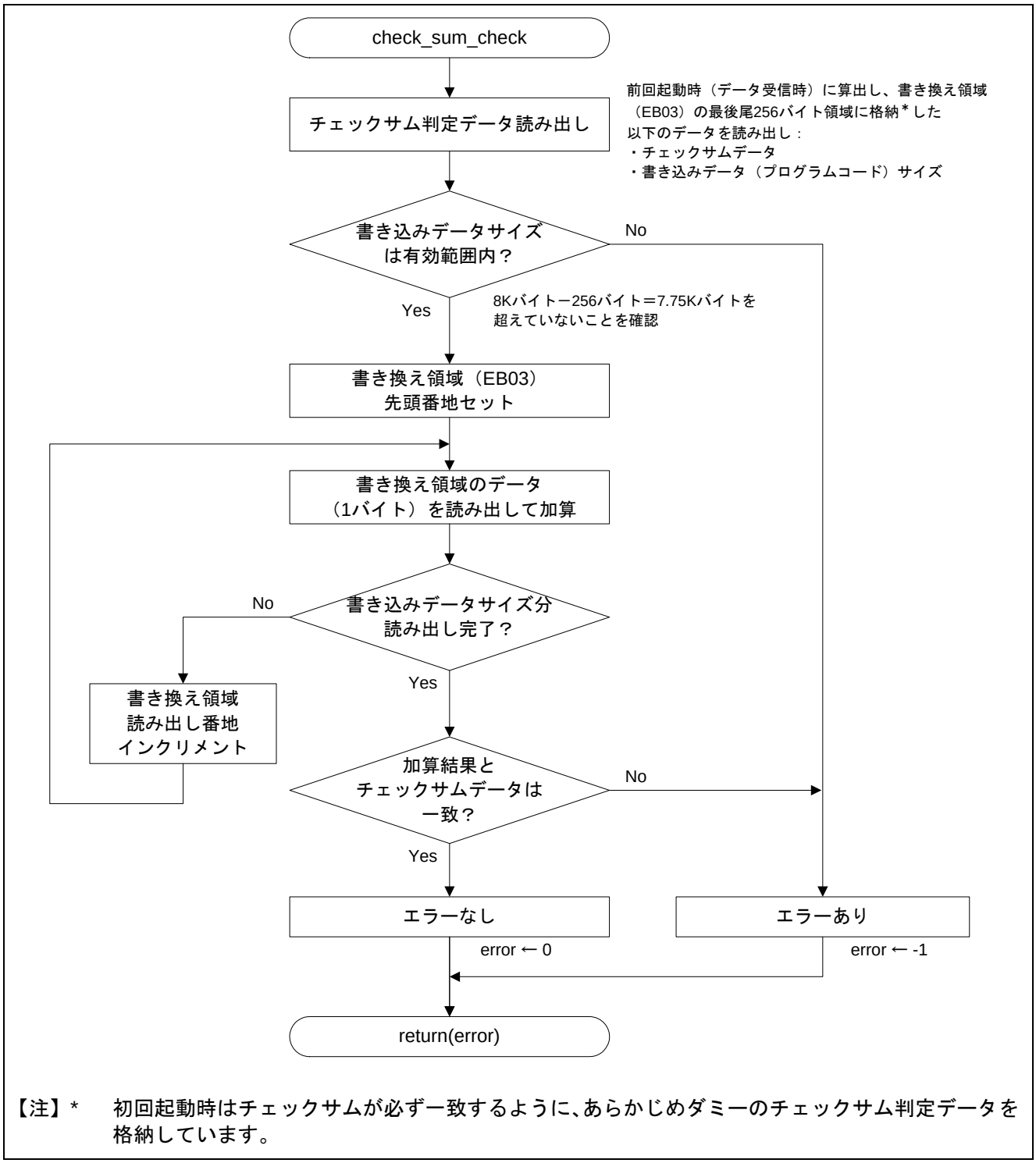


図 11　データ解析処理フロー

6.8.4　チェックサム判定処理

図 12にチェックサム判定処理のフローチャートを示します。



【注】*　初回起動時はチェックサムが必ず一致するように、あらかじめダミーのチェックサム判定データを格納しています。

図 12　チェックサム判定処理フロー

6.8.5 テキストバイナリ変換処理

図 13にテキストバイナリ変換処理のフローチャートを示します。

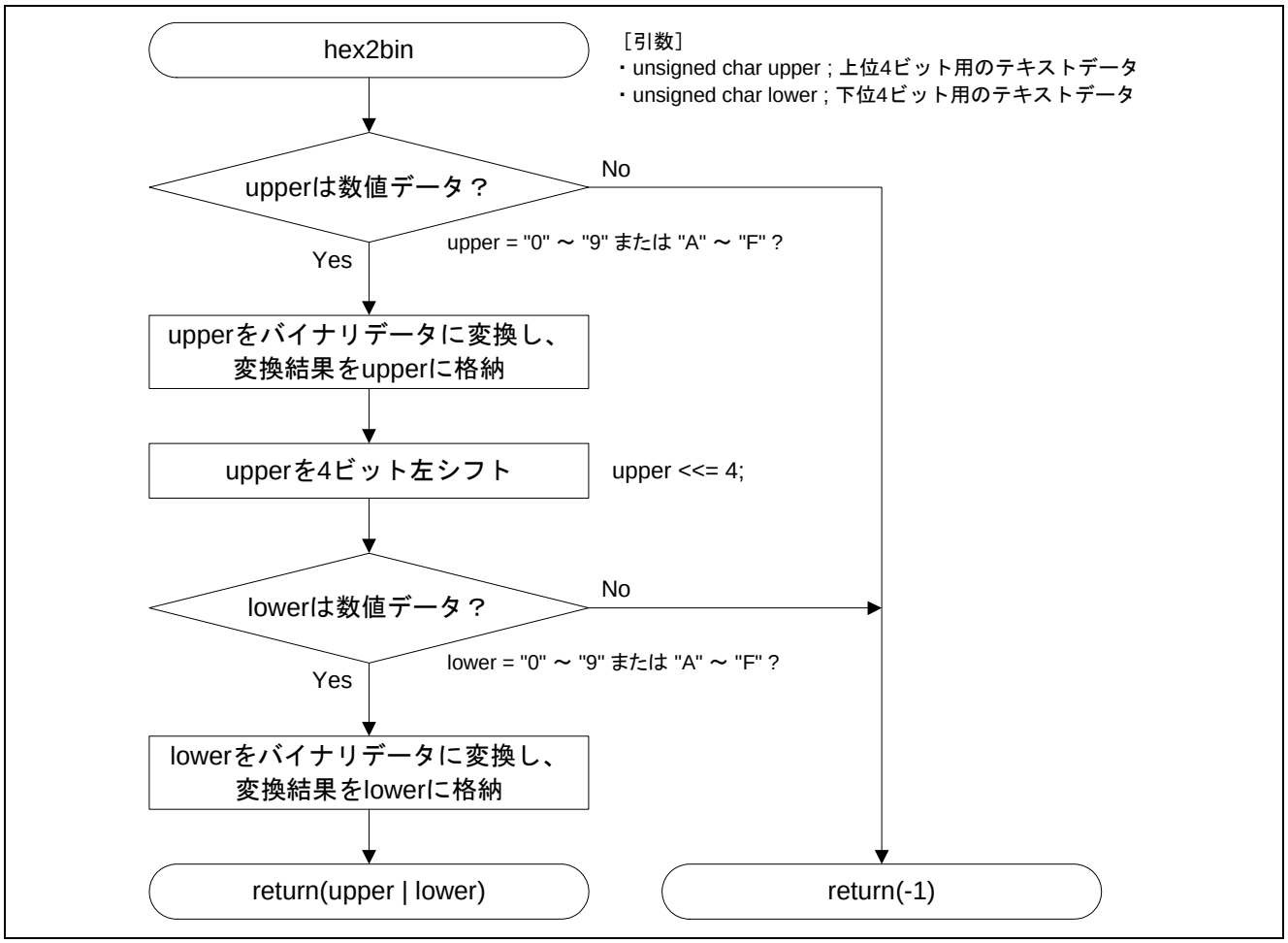


図 13　テキストバイナリ変換処理フロー

7.　操作概要

本応用例では、ホスト PC 上で VT100 互換ターミナルソフトを使用します。ここでは Microsoft Windows シリーズに標準搭載されているハイパーターミナルを使った操作例を示します。

ハイパーターミナルを起動して、「接続の設定」ウィンドウを立ち上げます。接続の名前を入力してアイコンを選択してください。図 14の左側のウィンドウが表示されます。このウィンドウ上の「接続方法」欄にシリアルポート番号を設定すると、図 14の右側に示すプロパティ画面が立ち上がります。この画面に 表 12 に示す値を設定してください。

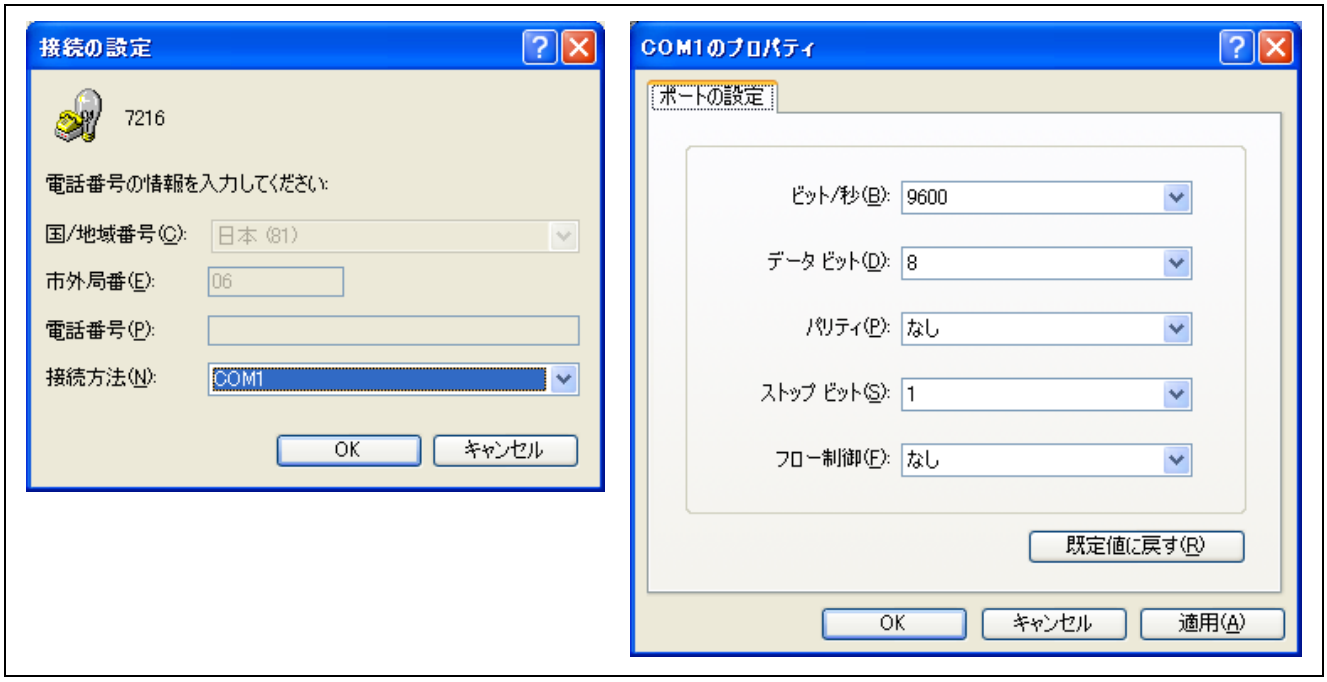


図 14　ハイパーターミナル接続設定とポートの設定

表 12　ポートの設定

項目	設定値
ビット／秒	9600
データビット	8
パリティ	なし
ストップビット	1
フロー制御	なし

上記の設定が完了したら、ホスト PC と SH7216 ボードをシリアルケーブル（RS-232C）で接続します。

ホストPC上で前ページ記載の設定を行った後、SH7216 ボード側でサンプルコードを起動すると、SH7216 は最初にESCシーケンスを送信し、ホストPC上のターミナルソフト画面表示がいったんクリアされます。続けて、SH7216 はホストPCに対し "Generate IRQ6 interrupt for transition to flash programming event." というメッセージを送信します（図 15）。

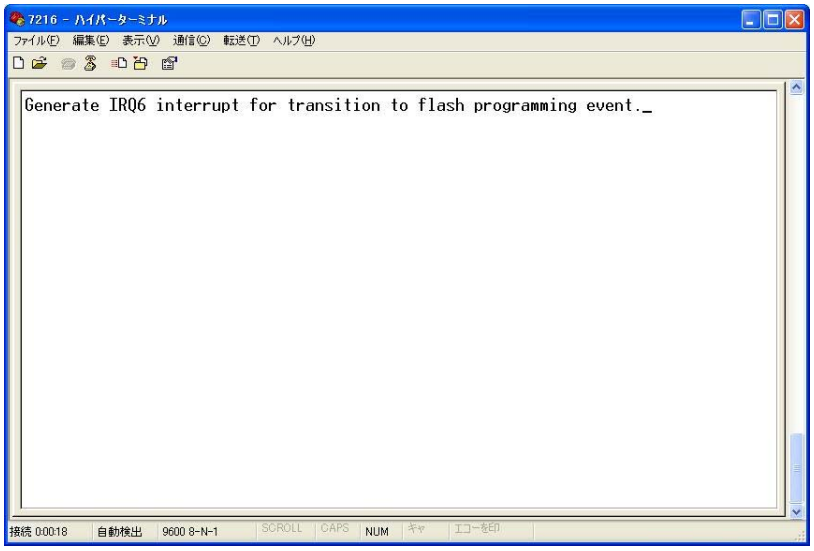


図 15　サンプルコード起動時のターミナルソフト画面

SH7216 はその後、アップデート領域（EB04）に格納したプログラムを実行し、ボード上の LED を一定周期（100ms）でパターン点滅させます。

この状態でボード上のIRQ6 スイッチを押すと、SH7216 はホストPCに対し "--> IRQ6 detected!" というメッセージを送信します。IRQ6 割り込みが発生すると、SH7216 はフラッシュ書き換えイベント処理に入り、最初にアップデート領域（EB04）をいったん消去します。消去が完了すると、SH7216 はホストPCに対し "Send subroutine code to update program in Motorola S-record format. Check card insert" というメッセージを送信し、MMCカードスロットにMMCが挿入されるまで待ち状態に入ります（図 16）。

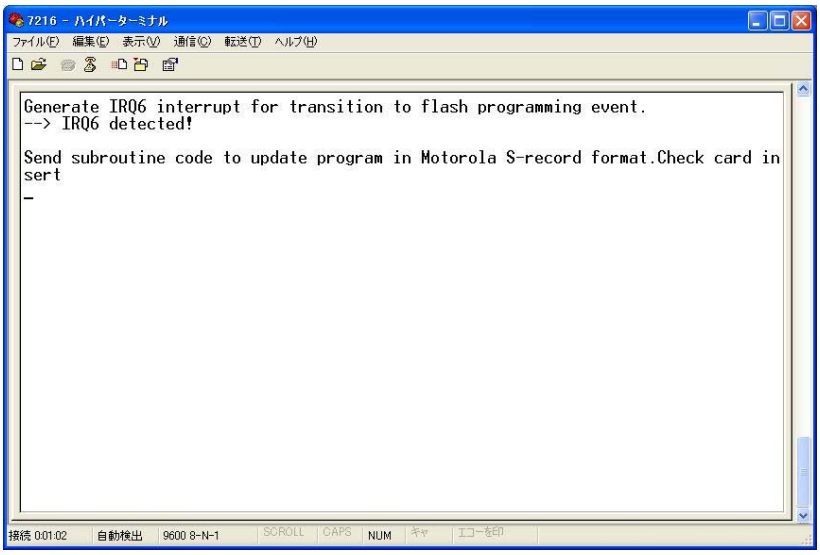


図 16　MMC 挿入待ち時のターミナルソフト画面

MMC をカードスロットに挿入すると、SH7216 はその中から有効データ（プログラムコード）を抽出し、アップデート領域（EB04）に書き込んでいきます。データ読み出しが完了し、プログラムコードの書き込み（アップデート）が完了すると、SH7216 はデータ読み出し中に算出したチェックサム判定用のデータを内蔵フラッシュに書き込みます。そして最後にウォッチドッグタイマを設定し、SH7216 はタイマカウンタオーバフローによるリセット待ち状態になります。

ウォッチドッグタイマによって再起動すると、SH7216 はアップデートした新しいプログラムを実行します。このとき、ボード上の LED は前回と異なるパターンで点滅します。

再起動前のデータ読み出し／フラッシュ書き換え（アップデート）が正常に実施できなかった場合、SH7216 はリセット入力による（再）起動時にチェックサムエラーと判定します。この場合、SH7216 は予備領域（EB04）のプログラムを実行します。

8. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

9. 参考ドキュメント

- ハードウェアマニュアル
  SH7214 グループ、SH7216 グループ　ユーザーズマニュアル ハードウェア編　Rev.3.00
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- 開発環境マニュアル
  SuperH C/C++コンパイラパッケージ V.9.04　ユーザーズマニュアル　Rev.1.01
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

ホームページとサポート窓口

- ルネサス エレクトロニクスホームページ
  http://japan.renesas.com/
- お問合せ先
  http://japan.renesas.com/contact/

改訂記録	SH7216 グループ　アプリケーションノート　MMC を使用したユーザプログラムモード Flash 書き換えによるプログラムアップデート例

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.09.16	—	初版発行
1.01	2012.06.15	—	サンプルコード（シンプルフラッシュ API）更新

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】電源投入時は，製品の状態は不定です。

   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。

   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

4. クロックについて

   【注意】リセット時は、クロックが安定した後、リセットを解除してください。

   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。

   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

ご注意書き

1. 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・システムの設計において、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て、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。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等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。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   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、産業用ロボット等
   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、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている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・システムに使用することはできません。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/