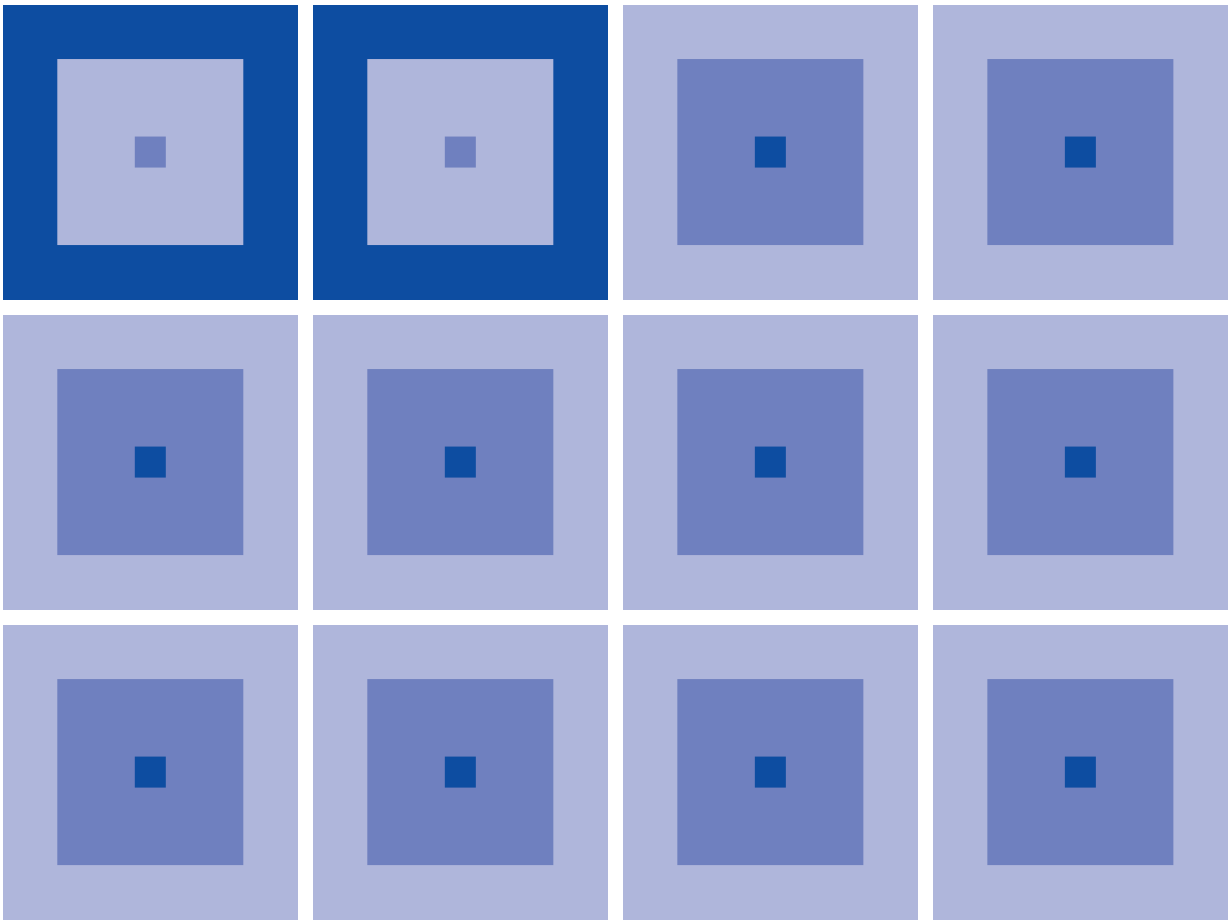
EPSON
EXCEED YOUR VISION

CMOS 8-BIT SINGLE CHIP MICROCOMPUTER

S5U1C8F626Y4 Manual

(S1C8F626自己プログラミングライブラリ)



セイコーエプソン株式会社

本資料のご使用につきましては、次の点にご留意願います。

1. 本資料の内容については、予告なく変更することがあります。
2. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りします。
3. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これらに起因する第三者の権利(工業所有権を含む)侵害あるいは損害の発生に対し、弊社は如何なる保証を行うものではありません。また、本資料によって第三者または弊社の工業所有権の実施権の許諾を行うものではありません。
4. 特性表の数値の大小は、数直線上の大小関係で表しています。
5. 本資料に掲載されている製品のうち、「外国為替及び外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。
6. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本(当該)製品をこれらの用途に用いた場合の如何なる責任についても負いかねます。

製品型番体系

●デバイス

S1　C　88104　F　0A01　00

- 00
  - ■ 梱包仕様
    - 00: テープ&リール以外
    - 0A: TCP BL　2方向
    - 0B: テープ&リール BACK
    - 0C: TCP BR　2方向
    - 0D: TCP BT　2方向
    - 0E: TCP BD　2方向
    - 0F: テープ&リール FRONT
    - 0G: TCP BT　4方向
    - 0H: TCP BD　4方向
    - 0J: TCP SL　2方向
    - 0K: TCP SR　2方向
    - 0L: テープ&リール LEFT
    - 0M: TCP ST　2方向
    - 0N: TCP SD　2方向
    - 0P: TCP ST　4方向
    - 0Q: TCP SD　4方向
    - 0R: テープ&リール RIGHT
    - 99: 梱包仕様未定
- 0A01
  - ■ 仕様
- F
  - ■ 形状
    - [D: ベアチップ、F: QFP、B: BGA]
- 88104
  - ■ 機種番号
- C
  - ■ 機種名称
    - [C: マイコン、デジタル製品]
- S1
  - ■ 製品分類
    - [S1: 半導体]

●開発ツール

S5U1　C　88348　D1　1　00

- 00
  - ■ 梱包仕様
    - [00: 標準梱包]
- 1
  - ■ バージョン
    - [1: Version 1]
- D1
  - ■ ツール種類
    - Hx: ICE
    - Ex: EVAボード
    - Px: ペリフェラルボード
    - Wx: FLASHマイコン用ROMライタ
    - Xx: ROMライタ周辺ボード
    - Cx: Cコンパイラパッケージ
    - Ax: アセンブラパッケージ
    - Dx: 機種別ユーティリティツール
    - Qx: ソフトシミュレータ
- 88348
  - ■ 対応機種番号
    - [88348: S1C88348用]
- C
  - ■ ツール分類
    - [C: マイコン用]
- S5U1
  - ■ 製品分類
    - [S5U1: 半導体用開発ツール]

- 目　次 -

1 概要

S5U1C8F626Y4はセイコーエプソン8ビットマイクロコンピュータS1C8F626用のプログラムライブラリで、S1C8F626に搭載されているFlash EEPROM内のプログラムコードやデータをアプリケーションプログラムから書き換えるためのプログラムモジュールを提供します。このライブラリをアプリケーションプログラムにリンクすることにより、セクタ消去、書き込み、ベリファイ、ブランクチェックの処理が関数コールで実行できます。これにより、S1C8F626組み込みアプリケーションに自己プログラミング機能を容易に実装することができます。

2 インストール

2.1 パッケージの内容

S1C8F626自己プログラミングライブラリのファイル、インストーラおよびPDFマニュアルは、CD-ROM 1枚で提供されます。

2.2 動作環境

S1C8F626自己プログラミングライブラリを使用するには以下の環境が必要です。

●パーソナルコンピュータ

IBM PC/ATまたは完全互換機が必要です。Pentium以降の200MHz CPUと64MB RAMが最低動作条件です。1GHz以上のCPUと256MB以上のRAMを搭載した機種を推奨します。

●ハードディスクおよびCD-ROMドライブ

S1C8F626自己プログラミングライブラリをインストールするには、CD-ROMドライブとハードディスク(10MB以上の空き容量)が必要です。

●ディスプレイ

SVGA(800×600ピクセル)以上のディスプレイが必要です。

●システムソフトウェアについて

Microsoft Windows 2000またはWindows XP(日本語版と英語版)に対応しています。

●開発用ソフトウェアツール

S5U1C88000C1(S1C88 Family統合ツールパッケージ)が必要です。

●開発用ハードウェアツール

S5U1C88000H5、S5U1C88000P1、S5U1C88655P2、S5U1C8F626F1、およびS5U1C8F626D4が必要です。

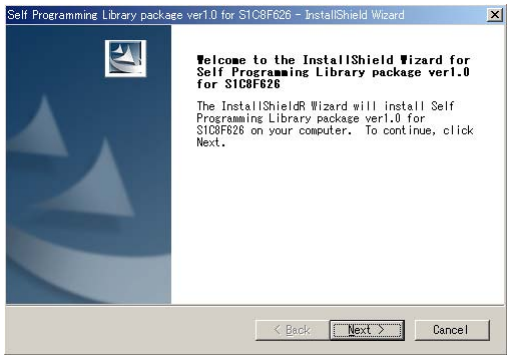
2.3 インストール方法

ライブラリは、添付のCD-ROMに収められたインストーラ(Setup.exe)を使用し、以下の手順で行います。S1C8F626自己プログラミングライブラリをインストールする前に、S5U1C88000C1(S1C88 Family統合ツールパッケージ)をインストールしておいてください。

(1) Windowsを起動します。既に起動している場合は、開いているプログラムをすべて終了させてください。

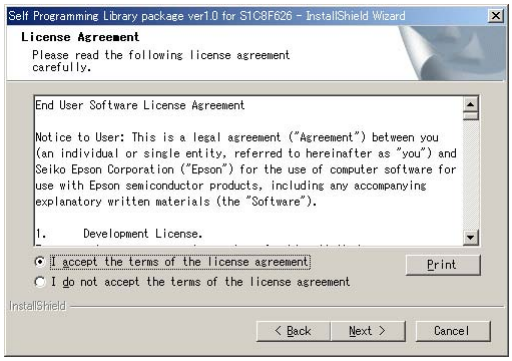
(2) CD-ROMをドライブに挿入し、その内容を表示させます。

(3) “Setup.exe”をダブルクリックして起動させます。



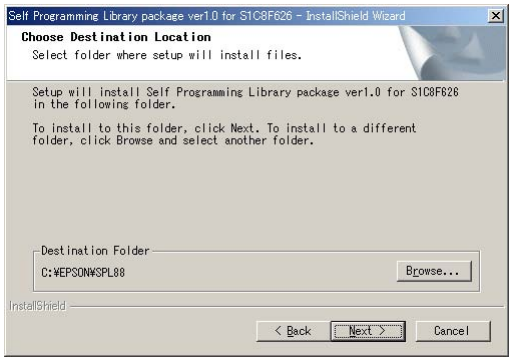
インストールウィザードのスタート画面が表示されます。

(4) [Next >]ボタンをクリックして次に進めてください。



エンドユーザソフトウェアライセンス契約画面が表示されます。表示されたライセンス契約内容は必ずお読みください。

(5) ライセンス契約に同意される場合は“I accept the terms of the license agreement”を選択して、[Next >]ボタンをクリックしてください。
同意できない場合は[Cancel]ボタンをクリックしてインストーラを終了させてください。

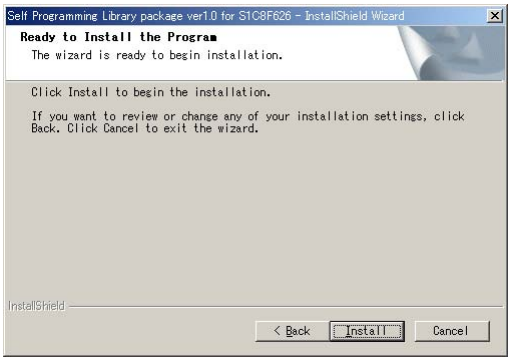


S1C8F626自己プログラミングライブラリをインストールするフォルダを指定する画面が表示されます。

(6) 表示されたインストール先を確認します。
インストール先を変更する場合は、[Browse...]ボタンでフォルダ選択ダイアログボックスを表示させます。インストールするフォルダをダイアログボックス上のリストから選択して[OK]ボタンをクリックしてください。

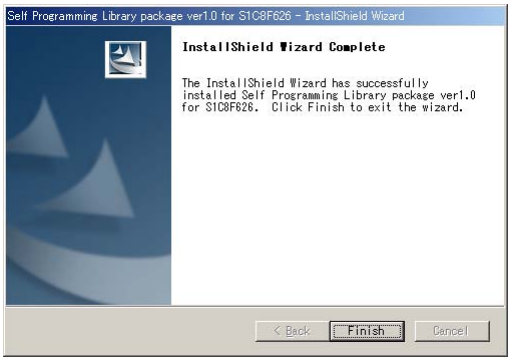
古いバージョンのライブラリがインストールされているフォルダを指定した場合、そのライブラリをアンインストールするか、フォルダを変更することを促すワーニングメッセージが表示されます。異なるフォルダを指定すれば、新旧のライブラリを共存させることができます。

(7) [Next >]ボタンをクリックします。



インストールのスタート画面が表示されます。

(8) [Install]ボタンをクリックしてインストールを開始してください。



インストールが完了するとコンプリート画面が表示されます。

(9) [Finish]ボタンをクリックしてインストーラを終了させてください。

インストールを途中で中止するには

インストール中に表示されるダイアログボックスはすべて[Cancel]ボタンを持っています。中止するにはダイアログボックスが表示されたところで[Cancel]ボタンをクリックしてください。

アンインストールするには

ライブラリをアンインストールするには、コントロールパネルの[アプリケーションの追加と削除]を使用してください。

2.4 インストールされるファイルの構成

コピー後のディレクトリとファイルの構成を以下に示します。

¥EPSON¥SPL88

ReadMe.txt	readmeファイル(最初にお読みください。)
¥lib	
¥Large	ラージモデル用ライブラリディレクトリ
selfprog.obj	自己プログラミングオブジェクトファイル
spl88_def.inc	外部宣言定義ファイル(アセンブラ用)
spl88_def.h	外部宣言定義ファイル(C用)
¥CompactData	コンパクトデータモデル用ライブラリディレクトリ
selfprog.obj	自己プログラミングオブジェクトファイル
spl88_def.inc	外部宣言定義ファイル(アセンブラ用)
spl88_def.h	外部宣言定義ファイル(C用)
¥CompactCode	コンパクトコードモデル用ライブラリディレクトリ
selfprog.obj	自己プログラミングオブジェクトファイル
spl88_def.inc	外部宣言定義ファイル(アセンブラ用)
spl88_def.h	外部宣言定義ファイル(C用)
¥Small	スモールモデル用ライブラリディレクトリ
selfprog.obj	自己プログラミングオブジェクトファイル
spl88_def.inc	外部宣言定義ファイル(アセンブラ用)
spl88_def.h	外部宣言定義ファイル(C用)
¥sample	
¥ASM	アセンブラサンプルディレクトリ
¥Large	自己プログラミングサンプル(ラージモデル用)
¥CompactData	自己プログラミングサンプル(コンパクトデータモデル用)
¥CompactCode	自己プログラミングサンプル(コンパクトコードモデル用)
¥Small	自己プログラミングサンプル(スモールモデル用)
¥C	Cサンプルディレクトリ
¥Large	自己プログラミングサンプル(ラージモデル用)
¥CompactData	自己プログラミングサンプル(コンパクトデータモデル用)
¥CompactCode	自己プログラミングサンプル(コンパクトコードモデル用)
¥Small	自己プログラミングサンプル(スモールモデル用)
¥doc	
¥english	英語ドキュメントディレクトリ
manual_e.pdf	マニュアル
rel_note_e.txt	リリースノート
¥japanese	日本語ドキュメントディレクトリ
manual_j.pdf	マニュアル
rel_note_j.txt	リリースノート

3 ライブラリの機能

本ライブラリは、S1C8F626の内蔵Flash EEPROMの自己プログラミングに必要な処理が記述されたオブジェクトファイルと各種の定義が記述されたヘッダファイルを提供します。

※使用条件

1. 本ライブラリはEPSON 8ビットマイクロコンピュータS1C8F626専用です。
2. 本ライブラリはS5U1C88000C1(S1C88 Family統合ツールパッケージ)によるプログラム開発用に作成されています。
3. 本ライブラリ関数実行中はS1C8F626の電源電圧を2.7V以上にする必要があります(“S1C8F626テクニカルマニュアル”参照)。

3.1 ライブラリのファイル構成

selfprog.obj: オブジェクトファイル

Flash消去や書き込みなどを処理する機能別の関数が含まれています。このファイルをアプリケーションプログラムにリンクして、自己プログラミング機能を実装します。

spl88_def.inc: アセンブラソース用外部宣言定義ファイル

アセンブラソースからの関数コールに使用する各種シンボルがEXTERN宣言されています。
アプリケーションの自己プログラミングモジュールをアセンブラソースとして作成する場合にインクルードしてください。

spl88_def.h: Cソース用外部宣言定義ファイル

Cソースからの関数コールに使用する各種シンボルが定義されています。
アプリケーションの自己プログラミングモジュールをCソースとして作成する場合にインクルードしてください。

オブジェクトファイルはメモリモデル(ラージ、コンパクトコード、コンパクトデータ、スモール)別に用意されており、上記2つの外部宣言定義ファイルと共にメモリモデル別のディレクトリ(2.4節参照)にインストールされます。アプリケーションのメモリ構成に合ったオブジェクトファイルを使用してください。

3.2 ライブラリ機能一覧

オブジェクトファイルが提供する機能を以下に示します。

(1)セクタ消去機能(関数名: spl88_erase)

S1C8F626内蔵Flash EEPROMの指定セクタ(4096バイト)を消去します。

(2)書き込み機能(関数名: spl88_write)

RAM内のデータを、Flash EEPROMの指定セクタに指定サイズ分書き込みます。書き込みサイズは、1バイトから4096バイトまで指定可能です。

(3)ベリファイ機能(関数名: spl88_verify)

Flash EEPROMの指定セクタに書き込まれている指定サイズ分のデータをRAM内のデータと比較して検証します。ベリファイサイズは、1バイトから4096バイトまで指定可能です。

(4)ブランクチェック機能(関数名: spl88_blank)

Flash EEPROMの指定セクタをブランクチェックします。

各関数の詳細については、“6 ライブラリ関数”を参照してください。

3.3 ライブラリサイズと処理サイクル数

表3.3.1 ライブラリサイズと処理サイクル数

サイズ/サイクル数		ライブラリメモリモデル	
		スモール, コンパクトコード	コンパクトデータ, ラージ
オブジェクトファイルサイズ		834バイト	981バイト
ライブラリワーク領域サイズ(RAM)	スタック	34バイト	34バイト
	エラー構造体	6バイト	6バイト
	データバッファ	最大4,096バイト	最大4,096バイト
コマンド処理サイクル(1セクタ)	書き込み	151,836サイクル	168,250サイクル
	消去	227サイクル	249サイクル
	ベリファイ	73,999サイクル	106,797サイクル
	ブランク	61,626サイクル	94,416サイクル

4 ライブラリの使用方法

4.1 プロジェクトへのファイルの追加

アプリケーションプログラムからS1C8F626自己プログラミングライブラリを使用するには、ライブラリのファイルをプロジェクトに追加する必要があります。

(1)ファイルのコピー

ライブラリのオブジェクトファイルとヘッダファイルをそれぞれ以下のフォルダにコピーします。

selfprog.obj プロジェクトのOBJフォルダ(¥<*プロジェクト名*>¥OBJ)にコピーします。
アプリケーションのメモリモデルに合った“selfprog.obj”を使用してください。

spl88_def.inc プロジェクトのSRCフォルダ(¥<*プロジェクト名*>¥SRC)にコピーします。(自己プログラミングモジュールをアセンブラソースとして作成する場合)

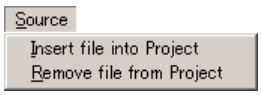
spl88_def.h プロジェクトのSRCフォルダ(¥<*プロジェクト名*>¥SRC)にコピーします。(自己プログラミングモジュールをCソースとして作成する場合)

(2)インクルードファイルの指定

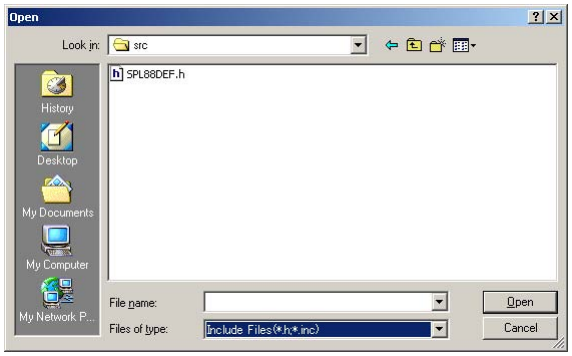
ヘッダファイル(spl88_def.incまたはspl88_def.h)はソースファイルに直接インクルードすることもできますが、ワークベンチ(WB88)で以下のとおりインクルードファイルとして指定することができます。

プロジェクトにヘッダファイルをインクルードする場合

1. [Source]メニューから[Insert file into Project]を選択します。



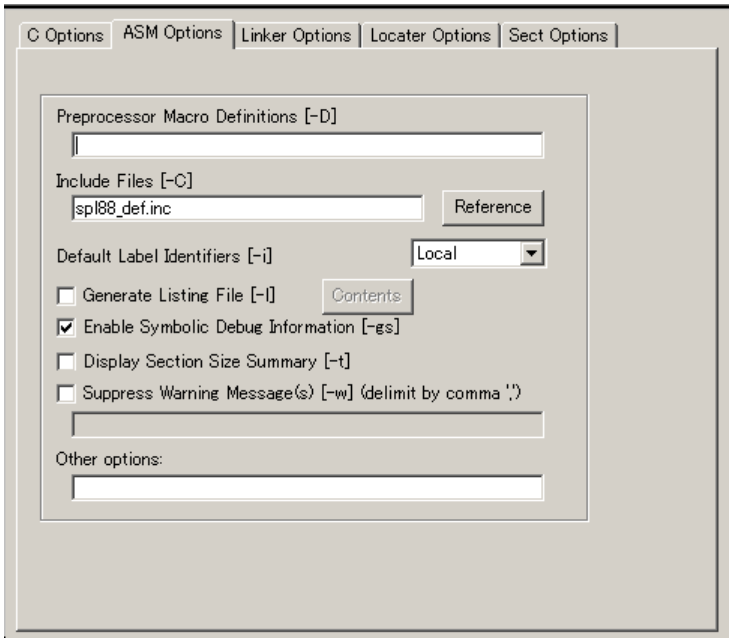
インクルードファイルを選択するためのダイアログボックスが表示されます。



2. ファイルの種類(Files of type:)を“Include Files(*.h, *.inc)”に変更し、インクルードするファイルを選択します。
[開く(Open)]をクリックすることで、選択したファイルがプロジェクトにインクルードされます。

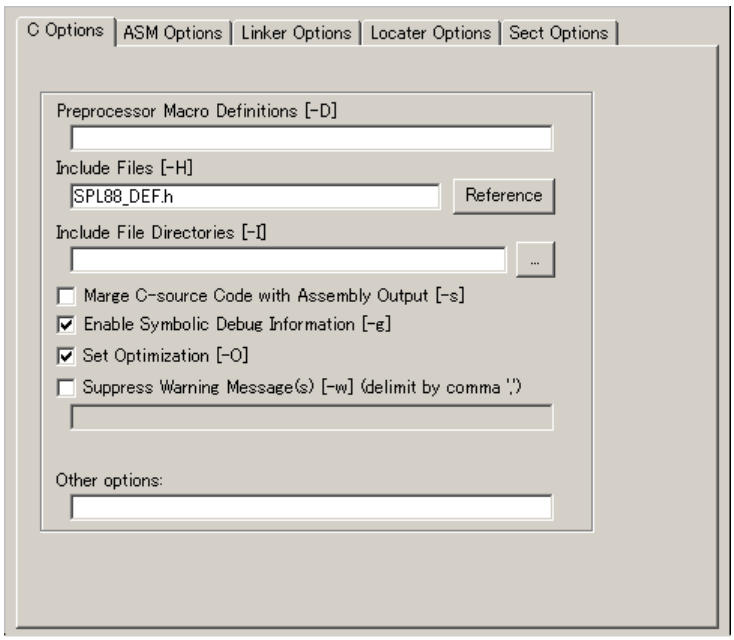
ライブラリ関数をアセンブラソースからコールする場合

1. オプションビューで[ASM Options]のページを表示させます。
2. [Reference]ボタンでファイル選択ダイアログボックスを表示させ、上記(1)でコピーした“spl88_def.inc”を選択して[Include Files]に設定します。あるいは、[Include Files]にファイル名を直接入力します。



ライブラリ関数をCソースからコールする場合

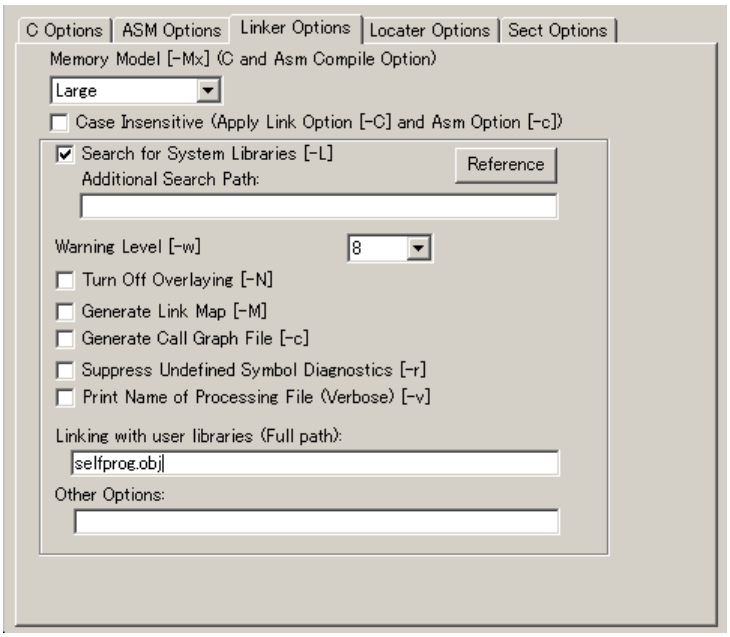
1. オプションビューで[C Options]のページを表示させます。
2. [Reference]ボタンでファイル選択ダイアログボックスを表示させ、上記(1)でコピーした“spl88_def.h”を選択して[Include Files]に設定します。あるいは、[Include Files]にファイル名を直接入力します。



(3)リンカオプションの指定

コピーしたオブジェクトファイル(selfprog.obj)がリンクされるようにリンカオプションを設定します。WB88による指定方法は以下のとおりです。

1. オプションビューで[Linker Options]のページを表示させます。
2. [Memory Model]を使用するメモリモデルに変更します。
3. [Linking with user libraries]に“selfprog.obj”を入力します。



4.2 メモリへのライブラリの配置

オブジェクトの配置はロケータ記述ファイル(*.dsc)に定義します。以下、WB88を使用してロケータ記述ファイルに定義する方法を示します。

例: S1C8F626自己プログラミングライブラリモジュールを1000H番地から配置する場合

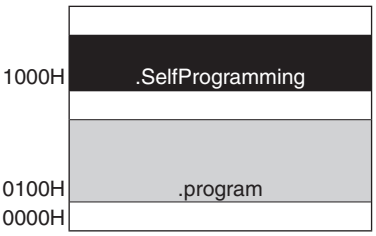
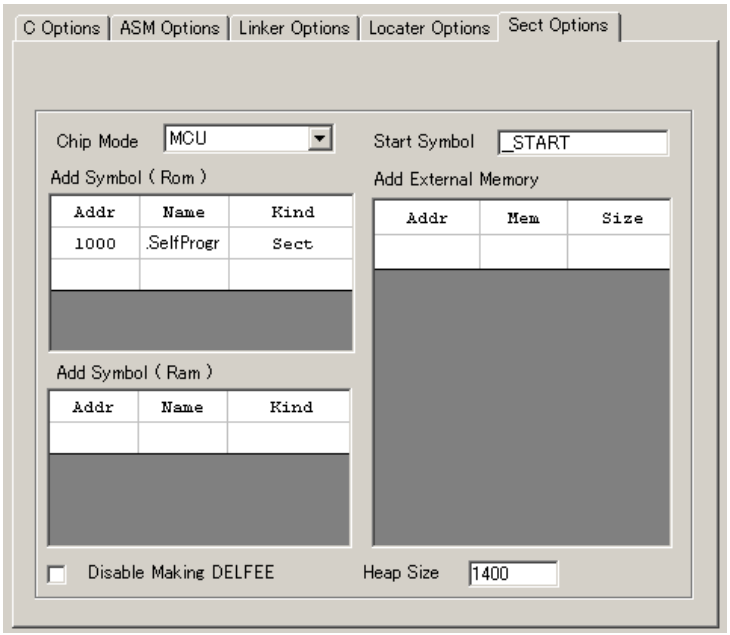


図4.2.1 メモリ配置例

1. WB88のオプションビューで[Sect Options]のページを表示させます。
2. [Add Symbol (Rom)]内の空白行の[Addr]のセルをクリックし、アドレス(例: 1000)を入力します。
3. [Name]に".SelfProgramming"を入力します。
4. [Kind]のセルをクリックし、表示されるプルダウンリストから"Sect"を選択します。
5. アプリケーションプログラムに従って、その他のシンボルを入力します。



WB88はここに設定された内容でロケータ記述ファイルを生成し、ロケータに渡します。

自己プログラミングライブラリモジュールは、S1C8F626内蔵メモリの任意のアドレスに配置可能です。ただし、使用するCPUモードによって配置可能な領域が制限されますので注意してください。詳細については、"5.5 プログラミング上の注意事項"を参照してください。

5 プログラムの作成

5.1 自己プログラミング処理フロー

図5.1.1に、アプリケーションプログラム内に作成する自己プログラミングルーチンのフローチャートを示します。
実際のプログラム例については、本ライブラリに添付のサンプルプログラム(¥SPL88¥sample)とAppendixの説明を参照してください。

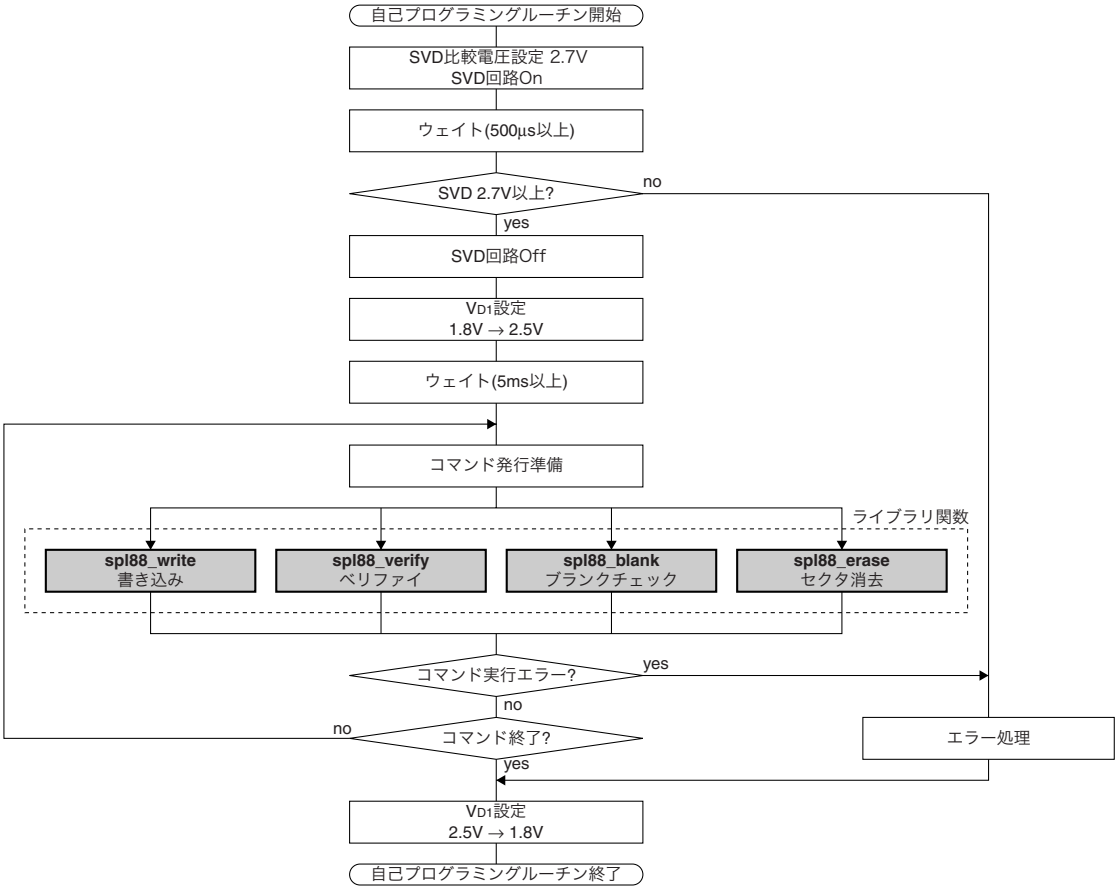


図5.1.1 自己プログラミング処理フロー

5.2 データバッファ

Flash EEPROMに書き込むコード、データをライブラリ関数に渡すためのデータバッファ（最大4,096バイト）をアプリケーションプログラムでRAM内に用意します。ベリファイの際も、Flash EEPROMのデータがここに設定したデータと比較されます。RAM内の任意の領域が使用可能で、その先頭アドレスをライブラリ関数呼び出し時に引数として渡します。

5.3 エラー構造体spl88_err_str

ベリファイまたはブランクチェックでエラーが発生した場合に、エラーの情報がライブラリ関数によってエラー構造体spl88_err_strに書き込まれます。構造体のメンバは以下のとおりです。

```
struct spl88_err_str{
   unsigned long  spl88_err_adr;        エラーの起きたアドレス
   unsigned char  spl88_org_dat;        比較元のデータ
   unsigned char  spl88_err_dat;        エラーとなったデータ
};
```

5.4 定数定義

インクルードファイルに以下の定数が定義されており、ユーザプログラムで使用可能です。

関数の戻り値

ライブラリ関数は実行結果をunsigned char型の戻り値として返します。アセンブラプログラムでは、Aレジスタから読み出せます。
ライブラリ関数の戻り値は、以下のとおり定義されています。

表5.4.1 関数戻り値一覧

定義名称	値	内容
SPL88_ERR_NON	0	正常終了
SPL88_ERR_SCTNUM	1	セクタ番号エラー 指定されたセクタ番号が0CH、0DH、0EH、0FH、または40H以上です。
SPL88_ERR_DATSIZ	2	データサイズエラー 指定されたデータサイズが0、または1001H以上です。
SPL88_ERR_BLANK	3	ブランクエラー ブランクチェックでエラーが発生しました。
SPL88_ERR_VERIFY	4	ベリファイエラー ベリファイチェックでエラーが発生しました。
SPL88_ERR_VD1	5	V_{D1}エラー V_{D1}が1.8Vに設定されています。

セクタ指定用定数

書き込み/ベリファイサイズとセクタ範囲のデフォルト値が以下のとおり定義されています。

表5.4.2 セクタ指定デフォルト値

定義名称	値	内容
SPL88_DAT_SIZ	1000H	書き込みサイズ、またはベリファイサイズ
SPL88_START_SCTNUM	4	消去、ブランクチェック、書き込み、ベリファイの開始セクタ番号
SPL88_END_SCTNUM	5	消去、ブランクチェック、書き込み、ベリファイの終了セクタ番号

5.5 プログラミング上の注意事項

自己プログラミングルーチンは以下の内容を考慮して作成してください。

(1)予約語

S1C8F626自己プログラミングライブラリは以下のセクション名とグローバルラベル/関数名を使用します。これらの名称をユーザプログラム内で使用することはできません。

セクション名: .SelfProgramming
グローバルラベル名(アセンブラ): _spl88_erase、_spl88_write、_spl88_verify、_spl88_blank
グローバル関数名(C): spl88_erase、spl88_write、spl88_verify、spl88_blank

(2)コード効率

Cコンパイラが生成するアセンブラコードは、アセンブラで開発した場合のコードと比較して、コードサイズ、実行速度とも、おおよそ2倍(スモールモデル)～4倍(ラージモデル)となります。処理速度が要求されるような場合や、コンパクトなコードサイズが要求されるような場合は、アセンブラでのプログラム開発をお勧めします。(コードサイズ比は処理内容によって異なり、上記コードサイズはあくまで目安です。)

(3)コンパイラメモリモデルとCPUモードの組み合わせ

S1C88ではCPUモードやバスモードにより、アクセス可能なコード用メモリやデータ用メモリのサイズが異なります。これに対応するため、Cコンパイラには4つのメモリモデルが用意されています。

表5.5.1 コンパイラメモリモデル

コンパイラメモリモデル	コードサイズ	データサイズ	CPUモード	バスモード
スモールモデル	コード < 64Kバイト	データ < 64Kバイト	ミニマム	シングルチップモード(MCU) 拡張64Kモード(MPU)
コンパクトコードモデル	コード < 64Kバイト	データ ≧ 64Kバイト	ミニマム	拡張512Kミニマムモード
コンパクトデータモデル	コード ≧ 64Kバイト	データ < 64Kバイト	マキシマム	拡張512Kマキシマムモード
ラージモデル	コード ≧ 64Kバイト	データ ≧ 64Kバイト	マキシマム	拡張512Kマキシマムモード

CPUモードがミニマムモードの場合、たとえばCARL命令でスタックに退避させる戻りアドレスは2バイトで済みますが、マキシマムモードの場合は3バイトになります。
このような制限がありますので、使用するCPUモードやデータメモリサイズに合ったコンパイラのメモリモデルを選択する必要があります。表に示した組み合わせ以外では使用しないでください。

本ライブラリにもコンパイラの各メモリモデルに対応する4種類のオブジェクトファイルが用意されていますので、必ずシステムに合ったものを使用してください。アセンブラでプログラムを作成する場合も、システムに合ったメモリモデルのライブラリオブジェクトを使用してください。

(4)ライブラリ配置可能領域と関数呼び出し可能領域

コンパイラメモリモデルによって、ライブラリを配置可能なページとライブラリを呼び出し可能なページが異なります。スモール/コンパクトコードモデルの場合、ページ0以外にプログラムコードを配置することはできません。

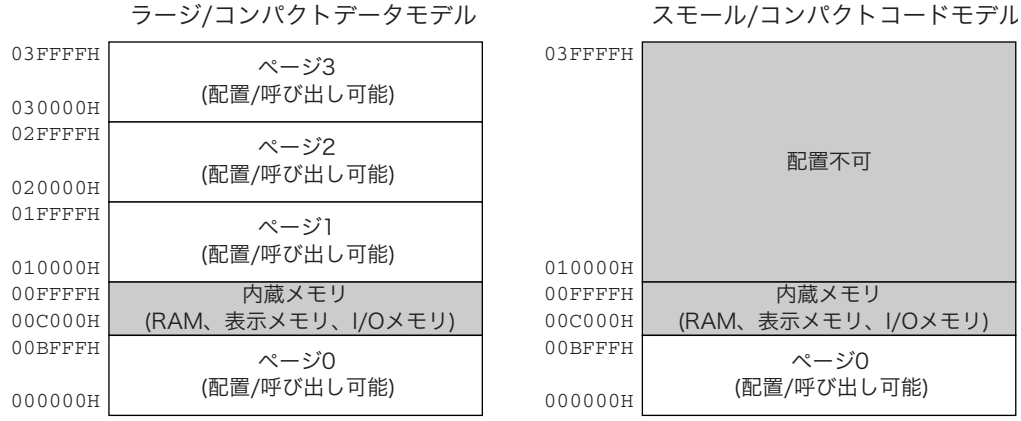


図5.5.1 ライブラリ配置可能領域

(5) セクタ番号とアドレスの対応

表5.5.2にセクタ番号とアドレスの対応を示します。セクタ番号0CH～0FH(0C000H～0FFFFH)と40H以上(40000H～)は指定できません。

表5.5.2 セクタ番号とアドレスの対応

セクタ番号	アドレス	セクタ番号	アドレス
00H	00000H～00FFFH	20H	20000H～20FFFH
01H	01000H～01FFFH	21H	21000H～21FFFH
02H	02000H～02FFFH	22H	22000H～22FFFH
03H	03000H～03FFFH	23H	23000H～23FFFH
04H	04000H～04FFFH	24H	24000H～24FFFH
05H	05000H～05FFFH	25H	25000H～25FFFH
06H	06000H～06FFFH	26H	26000H～26FFFH
07H	07000H～07FFFH	27H	27000H～27FFFH
08H	08000H～08FFFH	28H	28000H～28FFFH
09H	09000H～09FFFH	29H	29000H～29FFFH
0AH	0A000H～0AFFFH	2AH	2A000H～2AFFFH
0BH	0B000H～0BFFFH	2BH	2B000H～2BFFFH
(0CH)*	0C000H～0CFFFH	2CH	2C000H～2CFFFH
(0DH)*	0D000H～0DFFFH	2DH	2D000H～2DFFFH
(0EH)*	0E000H～0EFFFH	2EH	2E000H～2EFFFH
(0FH)*	0F000H～0FFFFH	2FH	2F000H～2FFFFH
10H	10000H～10FFFH	30H	30000H～30FFFH
11H	11000H～11FFFH	31H	31000H～31FFFH
12H	12000H～12FFFH	32H	32000H～32FFFH
13H	13000H～13FFFH	33H	33000H～33FFFH
14H	14000H～14FFFH	34H	34000H～34FFFH
15H	15000H～15FFFH	35H	35000H～35FFFH
16H	16000H～16FFFH	36H	36000H～36FFFH
17H	17000H～17FFFH	37H	37000H～37FFFH
18H	18000H～18FFFH	38H	38000H～38FFFH
19H	19000H～19FFFH	39H	39000H～39FFFH
1AH	1A000H～1AFFFH	3AH	3A000H～3AFFFH
1BH	1B000H～1BFFFH	3BH	3B000H～3BFFFH
1CH	1C000H～1CFFFH	3CH	3C000H～3CFFFH
1DH	1D000H～1DFFFH	3DH	3D000H～3DFFFH
1EH	1E000H～1EFFFH	3EH	3E000H～3EFFFH
1FH	1F000H～1FFFFH	3FH	3F000H～3FFFFH

＊ 指定不可

6 ライブラリ関数

ここでは、ライブラリ関数とその機能を個々に説明します。

注: 本章は関数名をCの形式で記載しています。アセンブラソースの場合、関数名の前に'_'が必要です。
例: C

```
stat = spl88_erase(sectornum);
```

アセンブラ

```
CARL   _spl88_erase
```

6.1 セクタ消去関数(spl88_erase)

関　　数	unsigned char spl88_erase(unsigned int sector_num);	
機　　能	S1C8F626 Flash EEPROMの指定のセクタを消去します。 本関数を実行中は、ウォッチドックタイマが停止し、すべての割り込みが禁止されます。	
引　　数	BAレジスタ (unsigned int sector_num)	セクタ番号(表5.5.2参照) 00H～0BH、10H～3FH
戻　り　値	Aレジスタ (unsigned char)	ステータス(表5.4.1参照) SPL88_ERR_NON:　正常終了 SPL88_ERR_SCTNUM: セクタ番号エラー SPL88_ERR_VD1:　V_{D1}エラー
出力データ	なし	

使　用　例

[ASM]

```
LD      BA,#001H                ; 消去セクタ番号を設定(BA設定) セクタ番号 = 1
CARL    _spl88_erase            ; セクタ消去関数をコール
CP      A,#000H                 ; ステータスをチェック
```

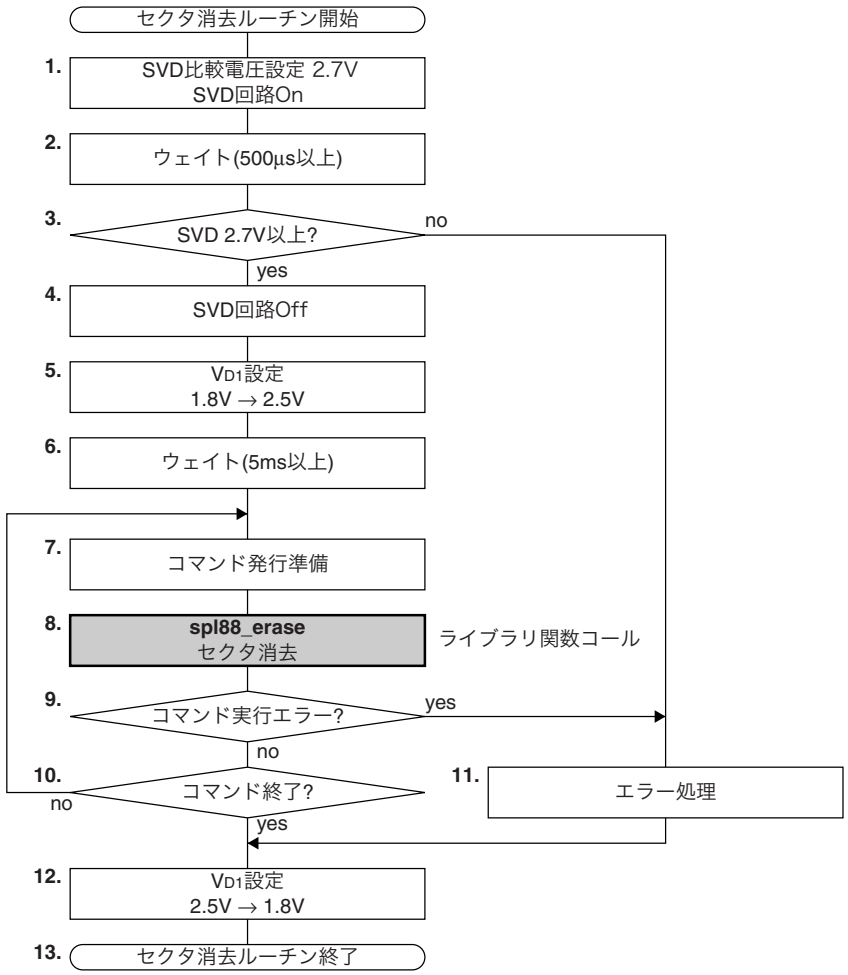
[C]

```
unsigned char stat;             // ステータス(= A)
unsigned int sectornum;         // セクタ番号(= BA)

sectornum = 0x1;                // 消去セクタ番号を設定(セクタ番号 = 1)
stat = spl88_erase(sectornum);  // セクタ消去関数をコール
if(stat != 0){                  // ステータスをチェック
```

セクタ消去処理フロー

セクタ消去処理の手順は以下のとおりです。



1. SVDのチェックを行う必要があるため、比較電圧を2.7Vに設定し、SVD回路をOnします。
2. 500µs以上のウェイトを挿入します。
3. SVDにより電源電圧が2.7V以上あるか確認します。2.7V以上の場合は4に、2.7V未満の場合は11に分岐します。
4. SVD回路をOffします。
5. V_{D1}を1.8Vから2.5Vに変更します。
6. V_{D1}の設定変更から`spl88_erase`関数コールまで5ms以上のウェイトを挿入します。
7. アセンブラソースの場合は消去するセクタ番号をBAレジスタにセットします。
   Cソースの場合は`(unsigned int)sectornum`変数を宣言し、消去するセクタ番号を代入します。
8. アセンブラソースの場合は`_spl88_erase`をコールします。
   Cソースの場合は`sectornum`を引数として、`spl88_erase`関数をコールします。
   関数がコールされるとセクタ消去処理を開始します。
9. アセンブラソースの場合はセクタ消去処理結果をAレジスタで確認します。
   Cソースの場合は`spl88_erase`関数の戻り値を確認します。
   正常の場合は10に、エラーの場合は11に分岐します。
10. コマンド処理を終了する場合は12に、続ける場合は7に分岐します。
11. エラー処理をします。
12. V_{D1}を2.5Vから1.8Vに変更します。
13. セクタ消去処理ルーチンを終了します。

6.2 書き込み関数(spl88_write)

関　　数	unsigned char spl88_write(unsigned char* pdata, unsigned int sector_num, unsigned int size);	
機　　能	S1C8F626 Flash EEPROMの指定セクタの先頭から指定サイズ分の領域に、ポインタで示されるデータを書き込みます。書き込みサイズは1バイトから4096バイトまで指定可能です。 本関数を実行中は、ウォッチドックタイマが停止し、すべての割り込みが禁止されます。	
引　　数	YP-IYレジスタ (unsigned char* pdata)	書き込みデータへのポインタ(RAM) YP: アドレスの上位1バイト, IY: アドレスの下位2バイト
	BAレジスタ (unsigned int sector_num)	セクタ番号(表5.5.2参照) 00H～0BH、10H～3FH
	HLレジスタ (unsigned int size)	書き込みサイズ 1～4096
戻　り　値	Aレジスタ (unsigned char)	ステータス(表5.4.1参照) SPL88_ERR_NON:　正常終了 SPL88_ERR_SCTNUM:　セクタ番号エラー SPL88_ERR_DATSIZ:　データサイズエラー SPL88_ERR_VD1:　V_{D1}エラー
出力データ	なし	

使　用　例

[ASM]

```
LD      YP,#@DPAG(spl88_rxp_dat) ; 書き込みデータポインタの設定(YP設定) 上位1バイト
LD      IY,#@DOFF(spl88_rxp_dat) ; 書き込みデータポインタの設定(IY設定) 下位2バイト
LD      BA,#001H                 ; 書き込みセクタ番号の設定(BA設定) セクタ番号 = 1
LD      HL,#01000H               ; 書き込みサイズの設定(HL設定) 4096バイト
CARL    _spl88_write             ; 書き込み関数をコール
CP      A,#000H                  ; ステータスをチェック
```

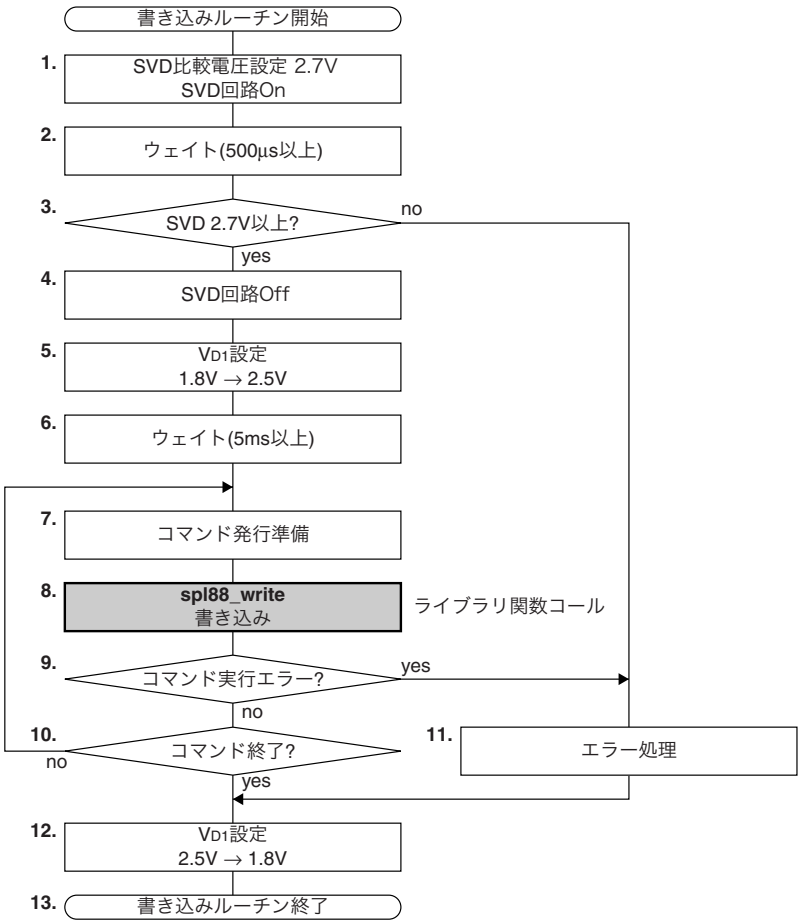
[C]

```
unsigned char stat;                                  // ステータス(= A)
unsigned char* pdat;                                 // 書き込みデータポインタ(= YP-IY)
unsigned int sectornum;                              // 書き込みセクタ番号(= BA)
unsigned int datasize;                               // 書き込みサイズ(= HL)

pdat = (unsigned char*)malloc(0x1000);               // 書き込みデータ領域の確保
  ...                                                // 書き込みデータの設定
sectornum = 0x1;                                     // 書き込みセクタ番号(1)の設定
datasize = 0x1000;                                   // 書き込みサイズの設定
stat = spl88_write(pdat, sectornum, datasize);       // 書き込み関数をコール
if(stat != 0){                                       // ステータスをチェック
  ...                                                // エラー処理
}
free(pdat);                                          // 書き込みデータ領域の解放
```

書き込み処理フロー

書き込み処理の手順は以下のとおりです。



1. SVDのチェックを行う必要があるため、比較電圧を2.7Vに設定し、SVD回路をOnします。
2. 500µs以上のウェイトを挿入します。
3. SVDにより電源電圧が2.7V以上あるか確認します。2.7V以上の場合は4に、2.7V未満の場合は11に分岐します。
4. SVD回路をOffします。
5. VD1を1.8Vから2.5Vに変更します。
6. VD1の設定変更からspl88_write関数コールまで5ms以上のウェイトを挿入します。
7. アセンブラソースの場合は、書き込みデータへのポインタ(先頭アドレス)をYPとIYレジスタ、書き込みセクタ番号をBAレジスタ、書き込みサイズをHLレジスタにそれぞれセットします。
   Cソースの場合は(unsigned char*)pdat、(unsigned int)sectornum、(unsigned int)datasizeの変数に、書き込みデータ、書き込みセクタ番号、書き込みサイズをそれぞれ設定します。
8. アセンブラソースの場合は_spl88_writeをコールします。
   Cソースの場合はpdat、sectornum、datasizeを引数として、spl88_write関数をコールします。関数がコールされると書き込み処理を開始します。
9. アセンブラソースの場合は書き込み処理結果をAレジスタで確認します。
   Cソースの場合はspl88_write関数の戻り値を確認します。
   正常の場合は10に、エラーの場合は11に分岐します。
10. コマンド処理を終了する場合は12に、続ける場合は7に分岐します。
11. エラー処理をします。
12. VD1を2.5Vから1.8Vに変更します。
13. 書き込み処理ルーチンを終了します。

6.3 ベリファイ関数（spl88_verify）

関　　数	unsigned char spl88_verify(unsigned char* pdata, unsigned int sector_num, unsigned int size, spl88_err_str* pSpl88_err_str);	
機　　能	S1C8F626 Flash EEPROMの指定セクタの先頭から指定サイズ分のデータを読み出し、ポインタで示されるデータと比較します。ベリファイサイズは1バイトから4096バイトまで指定可能です。 本関数を実行中は、ウォッチドックタイマが停止し、すべての割り込みが禁止されます。	
引　　数	YP-IYレジスタ （unsigned char* pdata）	比較元データへのポインタ（RAM） YP: アドレスの上位1バイト, IY: アドレスの下位2バイト
	BAレジスタ （unsigned int sector_num）	セクタ番号（表5.5.2参照） 00H～0BH、10H～3FH
	HLレジスタ （unsigned int size）	ベリファイサイズ 1～4096
	IXレジスタ （spl88_err_str* pSpl88_err_str）	エラー構造体（spl88_err_str）へのポインタ
戻　り　値	Aレジスタ （unsigned char）	ステータス（表5.4.1参照） SPL88_ERR_NON:　正常終了 SPL88_ERR_SCTNUM:　セクタ番号エラー SPL88_ERR_DATSIZ:　データサイズエラー SPL88_ERR_VERIFY:　ベリファイエラー SPL88_ERR_VD1:　V_{D1}エラー
出力データ	(unsigned long) spl88_err_str.spl88_err_adr	エラーが発生したアドレス（Flash EEPROM）
	(unsigned char) spl88_err_str.spl88_org_dat	比較元データ（RAM）
	(unsigned char) spl88_err_str.spl88_err_dat	エラーデータ（Flash EEPROM）
使　用　例	[ASM]	

```
LD    XP,#@DPAG(spl88_err_str) ; 構造体ポインタ設定 ページアドレス
LD    IX,#@DOFF(spl88_err_str) ; 構造体ポインタ設定 アドレス
PUSH  IX
LD    IX,SP                    ; スタックポインタ（構造体ポインタのスタック）
LD    YP,#@DPAG(spl88_rxp_dat) ; 比較元データポインタの設定（YP設定） 上位1バイト
LD    IY,#@DOFF(spl88_rxp_dat) ; 比較元データポインタの設定（IY設定） 下位2バイト
LD    BA,#001H                 ; ベリファイセクタ番号を設定（BA設定） セクタ番号 = 1
LD    HL,#01000H               ; ベリファイサイズの設定（HL設定）
CARL  _spl88_verify            ; ベリファイ関数をコール
CP    A,#000H                  ; ステータスをチェック
POP   IX
```

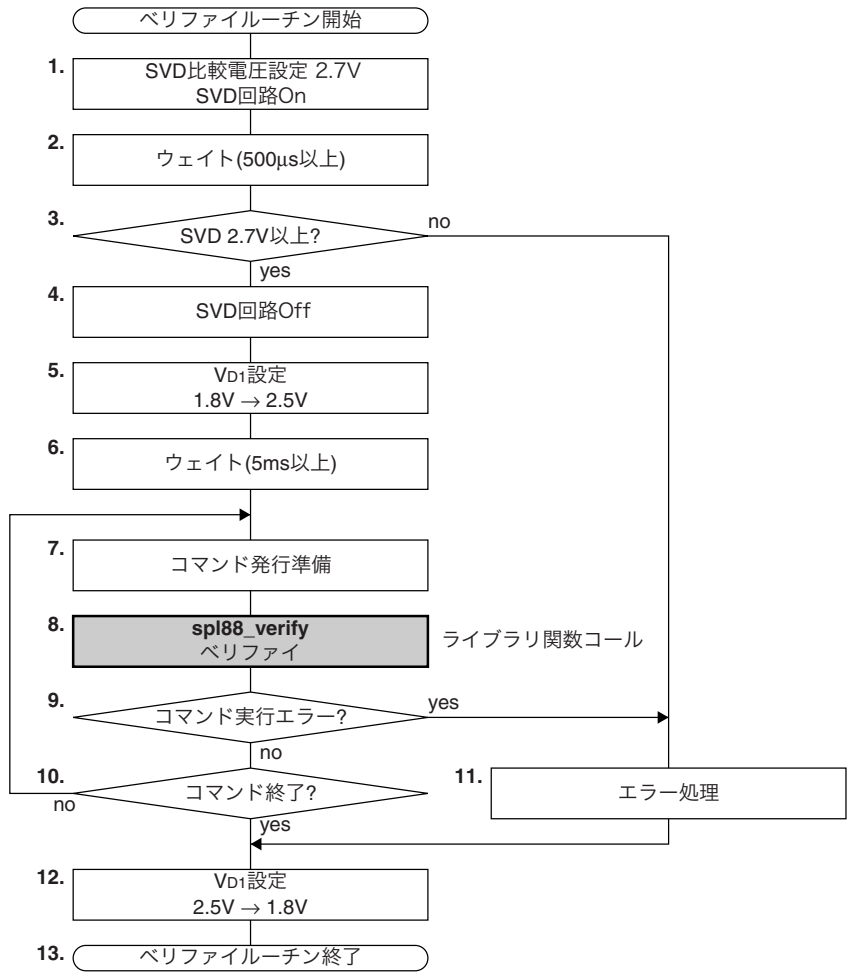
使　用　例

```
[C]
unsigned char stat;                                    // ステータス(= A)
unsigned char* pdat;                                   // 比較元データポインタ(= YP-IY)
unsigned int sectornum;                                // ベリファイセクタ番号(= BA)
unsigned int datasize;                                 // ベリファイサイズ(= HL)
spl88_err_str* pSpl88errstr;                           // エラー構造体(= IX)

pdat = (unsigned char*) malloc(0x1000);                // 比較元データ領域確保
  ...                                                  // 比較元データの設定
pSpl88errstr = (spl88_err_str*) malloc(sizeof(spl88_err_str));
                                                       // エラー構造体領域確保
sectornum = 0x1;                                       // ベリファイセクタ番号(1)の設定
datasize = 0x1000;                                     // ベリファイサイズの設定
stat = spl88_verify(pdat, sectornum, datasize, (spl88_err_str*) &pSpl88errstr);
                                                       // ベリファイ関数をコール
if(stat != 0){                                         // ステータスをチェック
  ...                                                  // エラー処理
}
free(pdat);                                            // 比較元データ領域の解放
free(pSpl88errstr);                                    // エラー構造体領域開放
```

ベリファイ処理フロー

ベリファイ処理の手順は以下のとおりです。



1. SVDのチェックを行う必要があるため、比較電圧を2.7Vに設定し、SVD回路をOnします。
2. 500µs以上のウェイトを挿入します。
3. SVDにより電源電圧が2.7V以上あるか確認します。2.7V以上の場合は4に、2.7V未満の場合は11に分岐します。
4. SVD回路をOffします。
5. V_{DI}を1.8Vから2.5Vに変更します。
6. V_{DI}の設定変更からspl88_verify関数コールまで5ms以上のウェイトを挿入します。
7. アセンブラソースの場合は、比較元データへのポインタ(先頭アドレス)をYPとIYレジスタ、ベリファイセクタ番号をBAレジスタ、ベリファイサイズをHLレジスタ、エラー構造体のポインタをIXレジスタにそれぞれセットします。
   Cソースの場合は(unsigned char*)pdat、(unsigned int)sectornum、(unsigned int)datasize、(spl88_err_str*)pSpl88errstrの変数に、比較元データ、ベリファイセクタ番号、ベリファイサイズ、エラー構造体のポインタをそれぞれ設定します。
8. アセンブラソースの場合は_spl88_verifyをコールします。
   Cソースの場合はpdat、sectornum、datasize、pSpl88errstrを引数として、spl88_verify関数をコールします。
   関数がコールされるとベリファイ処理を開始します。
9. アセンブラソースの場合はベリファイ処理結果をAレジスタで確認します。
   Cソースの場合はspl88_verify関数の戻り値を確認します。
   正常の場合は10に、エラーの場合は11に分岐します。
10. コマンド処理を終了する場合は12に、続ける場合は7に分岐します。
11. エラー処理をします。
12. V_{DI}を2.5Vから1.8Vに変更します。
13. ベリファイ処理ルーチンを終了します。

6.4 ブランクチェック関数(spl88_blank)

関　　数	unsigned char spl88_blank(unsigned int sector_num, spl88_err_str* pSpl88_err_str);	
機　　能	S1C8F626 Flash EEPROMの指定セクタのデータ(4096バイト)を読み出し、ブランクチェック(0FFHか否かをチェック)を行います。 本関数を実行中は、ウォッチドックタイマが停止し、すべての割り込みが禁止されます。	
引　　数	BAレジスタ (unsigned int sector_num)	セクタ番号(表5.5.2参照) 00H～0BH、10H～3FH
	IYレジスタ (spl88_err_str* pSpl88_err_str)	エラー構造体(spl88_err_str)へのポインタ
戻　り　値	Aレジスタ (unsigned char)	ステータス(表5.4.1参照) SPL88_ERR_NON: 正常終了 SPL88_ERR_SCTNUM: セクタ番号エラー SPL88_ERR_BLANK: ブランクエラー SPL88_ERR_VD1: V_{D1}エラー
出力データ	(unsigned long) spl88_err_str.spl88_err_adr	エラーが発生したアドレス(Flash EEPROM)
	(unsigned char) spl88_err_str.spl88_org_dat	オリジナルデータ(0FFH)
	(unsigned char) spl88_err_str.spl88_err_dat	エラーデータ(Flash EEPROM)

使　用　例

[ASM]

```
LD    YP,#@DPAG(spl88_err_str) ; 構造体ポインタ設定 ページアドレス
LD    IY,#@DOFF(spl88_err_str) ; 構造体ポインタ設定 アドレス
PUSH  IY
LD    IY,SP                    ; スタックポインタ(構造体ポインタのスタック)
LD    BA,#001H                 ; ブランクチェックセクタ番号を設定(BA設定) セクタ番号 = 1
CARL  _spl88_blank             ; ブランクチェック関数をコール
CP    A,#000H                  ; ステータスをチェック
POP   IY
```

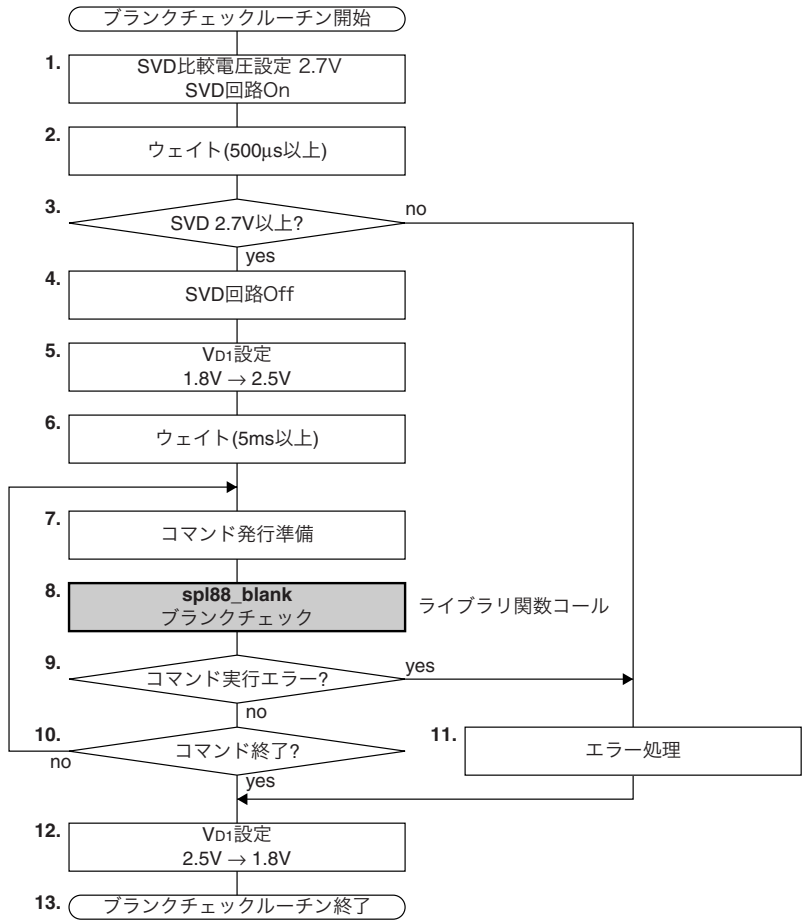
[C]

```
unsigned char stat;                     // ステータス(= A)
unsigned int sectornum;                 // ブランクチェックセクタ番号(= BA)
spl88_err_str* pSpl88errstr;            // エラー構造体(= IY)

pSpl88errstr = (spl88_err_str*) malloc(sizeof(spl88_err_str));
                                        // エラー構造体領域確保
sectornum = 0x1;                        // ブランクチェックセクタ番号(1)の設定
stat = spl88_blank(sectornum, (spl88_err_str*) &pSpl88errstr);
                                        // ブランクチェック関数をコール
if(stat != 0){                          // ステータスをチェック
  ...                                   // エラー処理
}
free(pSpl88errstr);                     // エラー構造体領域開放
```

ブランクチェック処理フロー

ブランク処理の手順は以下のとおりです。



1. SVDのチェックを行う必要があるため、比較電圧を2.7Vに設定し、SVD回路をOnします。
2. 500µs以上のウェイトを挿入します。
3. SVDにより電源電圧が2.7V以上あるか確認します。2.7V以上の場合は4に、2.7V未満の場合は11に分岐します。
4. SVD回路をOffします。
5. VD1を1.8Vから2.5Vに変更します。
6. VD1の設定変更からspl88_blank関数コールまで5ms以上のウェイトを挿入します。
7. アセンブラソースの場合は、Flash EEPROMのブランクチェックを行うセクタ番号をBAレジスタ、エラー構造体のポインタをIYレジスタにそれぞれセットします。
   Cソースの場合は(unsigned int)sectornumと(spl88_err_str*)pSpl88errstrの変数に、ブランクチェックセクタ番号とエラー構造体のポインタをそれぞれ設定します。
8. アセンブラソースの場合は_spl88_blankをコールします。
   Cソースの場合はsectornumとpSpl88errstrを引数として、spl88_blank関数をコールします。
   関数がコールされるとブランクチェック処理を開始します。
9. アセンブラソースの場合はブランクチェック処理結果をAレジスタで確認します。
   Cソースの場合はspl88_blank関数の戻り値を確認します。
   正常の場合は10に、エラーの場合は11に分岐します。
10. コマンド処理を終了する場合は12に、続ける場合は7に分岐します。
11. エラー処理をします。
12. VD1を2.5Vから1.8Vに変更します。
13. ブランクチェック処理ルーチンを終了します。

7 デバッグ時の注意事項

本ライブラリをリンクしたプログラムをデバッグする際は、以下の点に注意してください。

- デバッグの前に、パラメータファイル(8F626.par)の“Internal ROM”の指定を以下のとおり変更してください。
  `Map0=000000 00BFFF U W → Map0=000000 00BFFF U`
  `Map1=010000 03FFFF U W → Map1=010000 03FFFF U`
- 本ライブラリ関数によるセクタ消去と書き込みは、指定セクタが本ライブラリ、Cライブラリ、またはユーザコードが配置されている領域でも無条件に実行されますので、セクタ消去と書き込み関数をコールする際はセクタの指定に十分注意してください。
- 本ライブラリは電源電圧が2.7V以上なければ動作しません。ツールによるデバッグに加え、実機による評価も行ってください。
- 本ライブラリはスタック領域を34バイト使用します。このスタック領域を壊しますと、ライブラリ関数が正常に動作しませんので注意してください。
- 本ライブラリ関数の処理中は、すべての割り込み機能とウォッチドッグタイマが停止しますので、注意してください。

8 制限事項

- S1C8F626以外のFlash EEPROM内蔵プロセッサでは動作しませんので注意してください。
- S1C8F626のCPUモードと使用するライブラリのコンパイラメモリモデルの組み合わせが正しくないと、ライブラリ関数が正常に動作しない場合があります。

表8.1 コンパイラメモリモデルとCPUモードの対応

コンパイラメモリモデル	CPUモード	
	ミニマムモード	マキシマムモード
スモールモデル	○	×
コンパクトコードモデル	○	×
コンパクトデータモデル	×	○
ラージモデル	×	○

（○: 使用可、×: 使用不可）

- ライブラリ関数の実行中はウォッチドックタイマが停止します。
- ライブラリ関数の実行中はすべての割り込み機能が停止します。
- ライブラリ関数の実行には汎用レジスタが使用され、上書きされます。アセンブラで自己プログラミングモジュールを作成する場合、関数を呼び出す前に必ず汎用レジスタ値を退避させてください。
- セクタごとにV_{D1}を切り換える（1.8V→2.5V、2.5V→1.8V）ような処理は行わないでください。
- 本ライブラリ関数の実行時は電源電圧を2.7V以上にする必要があります。詳細は、“S1C8F626テクニカルマニュアル”を参照してください。
- 本ライブラリにより、RAM上のコードまたはデータをFlash EEPROMに書き込むことができますが、PCなどからRAM上にコードやデータを設定することはできません。PCからコードやデータを転送するプログラムや回路は、お客さまに用意していただく必要があります。

Appendix サンプルプログラム

S1C8F626自己プログラミングライブラリパッケージには、以下の処理を行うサンプルプログラムが含まれています。

1. SVDの制御（電源電圧が2.7V以上か確認）
2. V_{D1}電圧の制御（自己プログラミング中は2.5Vに設定）
3. 1セクタの消去（消去アドレス: 4000H～4FFFH）
4. 1セクタのブランクチェック（ブランクチェックアドレス: 4000H～4FFFH）
5. 1セクタのデータ書き込み（書き込みアドレス: 4000H～4FFFH、書き込みデータ: 04H）
6. 1セクタのベリファイチェック（ベリファイアドレス: 4000H～4FFFH）
7. 関数のエラー処理および終了処理

A.1 サンプルプログラム一覧

サンプルプログラムはライブラリのインストール時に、C:¥EPSON¥SPL88¥sampleディレクトリ（デフォルト）にコピーされます。sampleディレクトリ内には、以下のようにソース言語とメモリモデル別にサブディレクトリが作成されます。各サンプルプログラムの機能自体はすべて同じです。

C:¥EPSON	
¥SPL88	
¥sample	
¥ASM	アセンブラサンプルディレクトリ
¥Small	スモールモデル用アセンブラサンプルプログラム
¥SRC	
boot.asm	スタートアップルーチンソースファイル
sample.asm	メインルーチンソースファイル
spl88_def.inc	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥CompactCode	コンパクトコードモデル用アセンブラサンプルプログラム
¥SRC	
boot.asm	スタートアップルーチンソースファイル
sample.asm	メインルーチンソースファイル
spl88_def.inc	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥CompactData	コンパクトデータモデル用アセンブラサンプルプログラム
¥SRC	
boot.asm	スタートアップルーチンソースファイル
sample.asm	メインルーチンソースファイル
spl88_def.inc	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥Large	ラージモデル用アセンブラサンプルプログラム
¥SRC	
boot.asm	スタートアップルーチンソースファイル
sample.asm	メインルーチンソースファイル
spl88_def.inc	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル

¥C	Cサンプルディレクトリ
¥Small	スモールモデル用Cサンプルプログラム
¥SRC	
cstart.s	スタートアップルーチンソースファイル
sample.c	メインルーチンソースファイル
spl88_def.h	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥CompactCode	コンパクトコードモデル用Cサンプルプログラム
¥SRC	
cstart.s	スタートアップルーチンソースファイル
sample.c	メインルーチンソースファイル
spl88_def.h	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥CompactData	コンパクトデータモデル用Cサンプルプログラム
¥SRC	
cstart.s	スタートアップルーチンソースファイル
sample.c	メインルーチンソースファイル
spl88_def.h	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル
¥Large	ラージモデル用Cサンプルプログラム
¥SRC	
cstart.s	スタートアップルーチンソースファイル
sample.c	メインルーチンソースファイル
spl88_def.h	外部シンボル宣言定義ファイル
¥OBJ	
selfprog.obj	自己プログラミングライブラリオブジェクトファイル

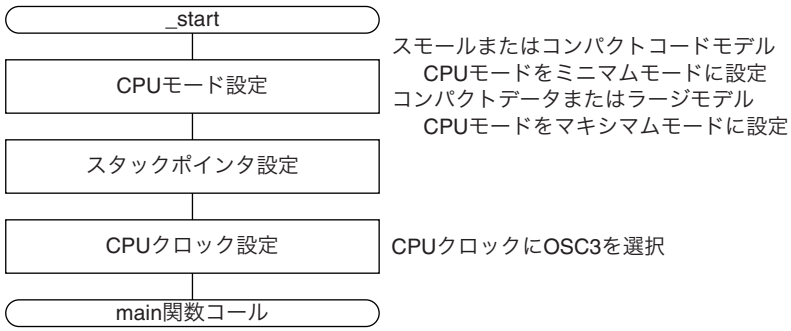
A.2 サンプルプログラム内の関数

サンプルプログラムには以下の関数が含まれています。

(1) _start　　初期化関数
(2) main　　メイン関数
(3) spl88_wait　　ウェイト関数
(4) spl88_setwritedat　　書き込みデータ設定関数
(5) spl88_finish_proc　　終了処理関数

A.2.1 _start(初期化関数)

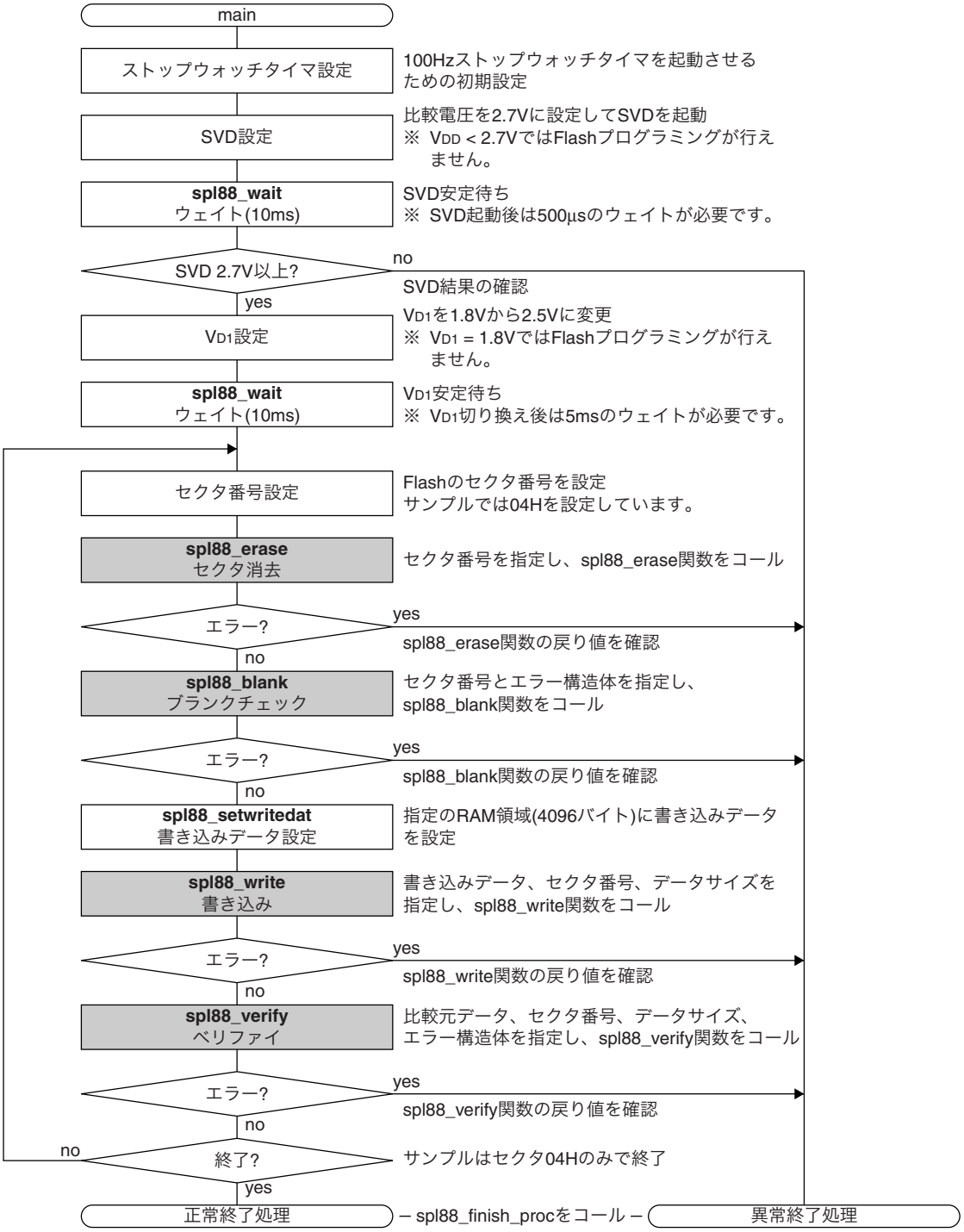
関　　数	void _start(void);
機　　能	イニシャルリセット後、CPUにより実行され、I/Oレジスタを初期化します。 まず、I/OアドレスFF00HでCPUモードを設定します。スモールまたはコンパクトコードモデル用のサンプルはCPUをミニマムモードに設定します。コンパクトデータまたはラージモデル用のサンプルはCPUをマキシマムモードに設定します。 次にI/OアドレスFF01Hでスタックページを0に設定後、スタックポインタを設定します。 最後にI/OアドレスFF02HでCPUクロックをOSC3に設定し、main関数をコールします。
引　　数	なし
戻　り　値	なし



図A.2.1.1 _startフローチャート

A.2.2 main(メイン関数)

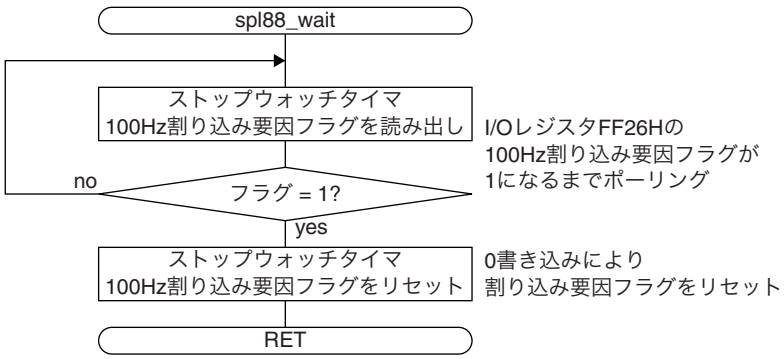
関　　数	void main(void);
機　　能	サンプルプログラムのメインルーチンで、ROM上で実行されます。 最初に、SVDやV_{D1}用のウェイト時間を生成するストップウォッチタイマを設定します。次にSVDを設定して電源電圧が2.7V以上あるか確認します。電源電圧が2.7V未満の場合は、spl88_finish_proc関数をコールして異常終了処理を行います。 電源電圧が2.7V以上ある場合は、V_{D1}をFlashプログラミング用の2.5Vに切り換えます。V_{D1}電圧の安定を待った後、Flashのセクタ4(4000H～4FFFH)に対して、消去、ブランクチェック、データ(04H)書き込み、ベリファイチェックを連続して行います。 処理中にエラーが発生した場合は、spl88_finish_proc関数をコールして異常終了処理を行います。 ベリファイチェックまで正常に終了した場合は、spl88_finish_proc関数をコールして正常終了処理を行います。
引　　数	なし
戻　り　値	なし



図A.2.2.1 mainフローチャート

A.2.3 spl88_wait（ウェイト関数）

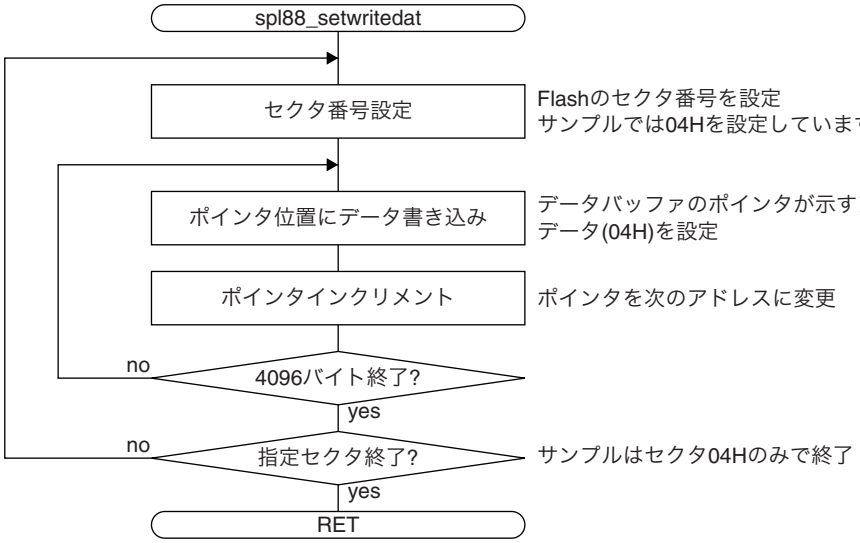
関　　数	void spl88_wait(void);
機　　能	メインルーチンから呼び出され、ストップウォッチタイマを使用してSVDやV_{D1}用の動作安定待ち時間を作ります。この関数は、100Hzストップウォッチタイマによる約10msのカウント終了後、呼び出し元にリターンします。
引　　数	なし
戻　り　値	なし
注　　意	• 本関数を使用するには、事前に100Hzストップウォッチタイマの設定が必要です（main関数のソースを参照してください。） • 待ち時間の生成にストップウォッチタイマを使用する必要はありません。ただし、SVD起動時やV_{D1}切り換え時はそれぞれに必要な安定時間以上の待ち時間を、アプリケーションで可能な方法で生成してください。



図A.2.3.1 spl88_waitフローチャート

A.2.4 spl88_setwritedat（書き込みデータ設定関数）

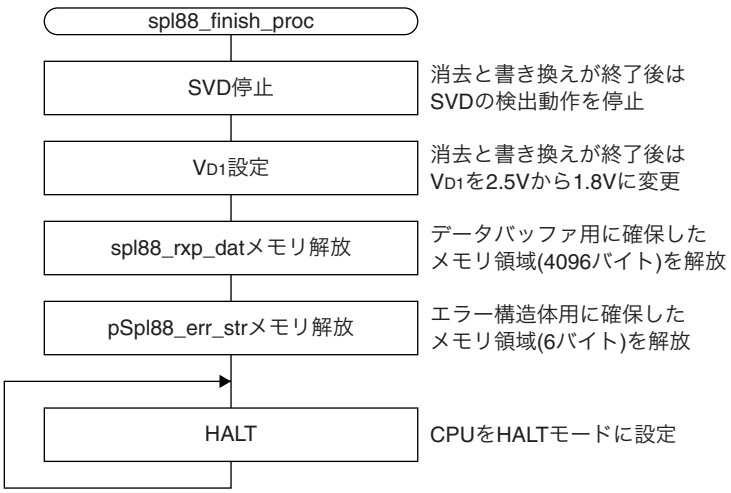
関　　数	void spl88_setwritedat(unsigned char* spl88_rxp_dat, unsigned int sectornum);	
機　　能	ポインタで指定される4096バイトのデータバッファ（RAM領域）に、データを設定します。サンプルでは04Hを4096バイト分設定しています。	
引　　数	YP-IYレジスタ （unsigned char* spl88_rxp_dat）	データバッファへのポインタ（RAM） YP: アドレスの上位1バイト, IY: アドレスの下位2バイト
	BAレジスタ （unsigned int sectornum）	セクタ番号
戻　り　値	なし	
注　　意	本関数はサンプルプログラム用に作成されており、指定RAM上への固定値の設定のみを行います。アプリケーションに使用可能な機能は実装されていません。	



図A.2.4.1 spl88_setwritedatフローチャート

A.2.5 spl88_finish_proc(終了処理関数)

関　　数	void spl88_finish_proc(unsigned char* spl88_rxp_dat, spl88_err_str* pSpl88_err_str);	
機　　能	ライブラリ関数実行後の終了処理を行います。 SVDを停止し、V_{D1}を通常モードの1.8Vに戻します。また、4096バイトのデータバッファとエラー構造体用に確保したメモリ領域を解放します。 最後にCPUをHALTモードにして終了します。	
引　　数	YP-IYレジスタ (unsigned char* spl88_rxp_dat)	データバッファへのポインタ(RAM) YP: アドレスの上位1バイト, IY: アドレスの下位2バイト
	XP-IXレジスタ (spl88_err_str* pSpl88_err_str)	エラー構造体(spl88_err_str)へのポインタ XP: アドレスの上位1バイト, IX: アドレスの下位2バイト
戻　り　値	なし	



図A.2.5.1 spl88_finish_procフローチャート

S5U1C8F626Y4 Manual
(S1C8F626自己プログラミングライブラリ)

セイコーエプソン株式会社
半導体事業部　IC営業部

〈IC国内営業グループ〉
東京　　〒191-8501　東京都日野市日野421-8
TEL（042）587-5313（直通）　FAX（042）587-5116

大阪　　〒541-0059　大阪市中央区博労町3-5-1　エプソン大阪ビル15F
TEL（06）6120-6000（代表）　FAX（06）6120-6100

インターネットによる電子デバイスのご紹介　http://www.epson.jp/device/semicon/

ドキュメントコード：411090100

2007年7月作成 ⓁA