EPSON
EXCEED YOUR VISION

FSA2.0
プログラム記述規約

セイコーエプソン株式会社

Rev.1.0

評価ボード・キット、開発ツールご使用上の注意事項

1. 本評価ボード・キット、開発ツールは、お客様での技術的評価、動作の確認および開発のみに用いられることを想定し設計されています。それらの技術評価・開発等の目的以外には使用しないで下さい。本品は、完成品に対する設計品質に適合していません。

2. 本評価ボード・キット、開発ツールは、電子エンジニア向けであり、消費者向け製品ではありません。お客様において、適切な使用と安全に配慮願います。弊社は、本品を用いることで発生する損害や火災に対し、いかなる責も負いかねます。通常の使用においても、異常がある場合は使用を中止して下さい。

3. 本評価ボード・キット、開発ツールに用いられる部品は、予告無く変更されることがあります。

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。

本資料の内容については、予告無く変更することがあります。

1. 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いたします。

2. 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知的財産権およびその他の権利侵害あるいは損害の発生に対し、弊社はいか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知的財産権およびその他の権利の実施権の許諾を行うものではありません。

3. 特性値の数値の大小は、数直線上の大小関係で表しています。

4. 製品および弊社が提供する技術を輸出等するにあたっては「外国為替および外国貿易法」を遵守し、当該法令の定める手続きが必要です。大量破壊兵器の開発等およびその他の軍事用途に使用する目的をもって製品および弊社が提供する技術を費消、再販売または輸出等しないでください。

5. 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いかなる責任についても負いかねます。

6. 本資料に掲載されている会社名、商品名は、各社の商標または登録商標です。

目　次

1. はじめに

FSA2.0（Flexible Signal Processing Accelerator）は、マイクロコードによって柔軟に処理内容を変更できる新しいタイプのアクセラレータである（以下、FSA と呼ぶ)。積和演算をベースとしたデジタル信号処理全般に適用が可能である。詳細は FSA2.0 仕様書を参照されたい。

1.1 目的

FSA はアクセラレータであることから､搭載されるホスト CPU 等はさまざまであることが予想される。そこで、FSA を用いたソフトウェアを作成するにあたっては、そのポータビリティ性を確保することが重要である。

本書は、FSA2.0 上で動作するソフトウェアの記述において、ポータビリティ性を確保するための記述規約を定義する。

1.2 用語

本規約で使用する用語の定義は以下の通りである。

- FSA 下位レベル API： FSA のレジスタを操作する API 関数。
- FSA インライン・プログラム： 後述する `fsa` 関数によって記述された FSA の 1 つの命令。
- FSA ユニット・プログラム： `FSASetPC` 関数から `_EXIT` 命令で処理を終了するまでの FSA のプログラムの実行単位。
- FSA ライブラリ関数： 少なくとも 1 つ以上の FSA ユニット・プログラムを含む関数。
- FSA モジュール： 複数の FSA ライブラリ関数で構成される特定の機能を実現する単位。
- ワークメモリ： FSA ライブラリ関数に閉じて一時的な作業領域として使用されるメモリ。
- ホスト CPU： アクセラレータである FSA を搭載する CPU。
- エミュレーション： FSA プログラムを PC 上で模擬的に実行すること。

下図に FSA を用いた DMA 転送プログラムを例として説明する。具体的な FSA の命令セット等の詳細は FSA の仕様書を参照されたい。

下図において、FSA から始まる青字の関数が FSA 下位レベル API と呼ぶものである。`fsa` から始まる赤字の関数がそれぞれ FSA インライン･プログラムと呼ぶものであり､FSA 上で動作する部分である。逆に赤字の部分以外はホスト CPU 上で動作する｡また､`FSASetPC` から始まる青字と赤字の部分を FSA ユニット・プログラムと呼ぶ。

下図の全体（`FsaDMA` 関数そのもの）を FSA ライブラリ関数と呼ぶ。このような FSA ライブラリ関数で構成される特定の機能を実現する単位を FSA モジュールと呼ぶ。

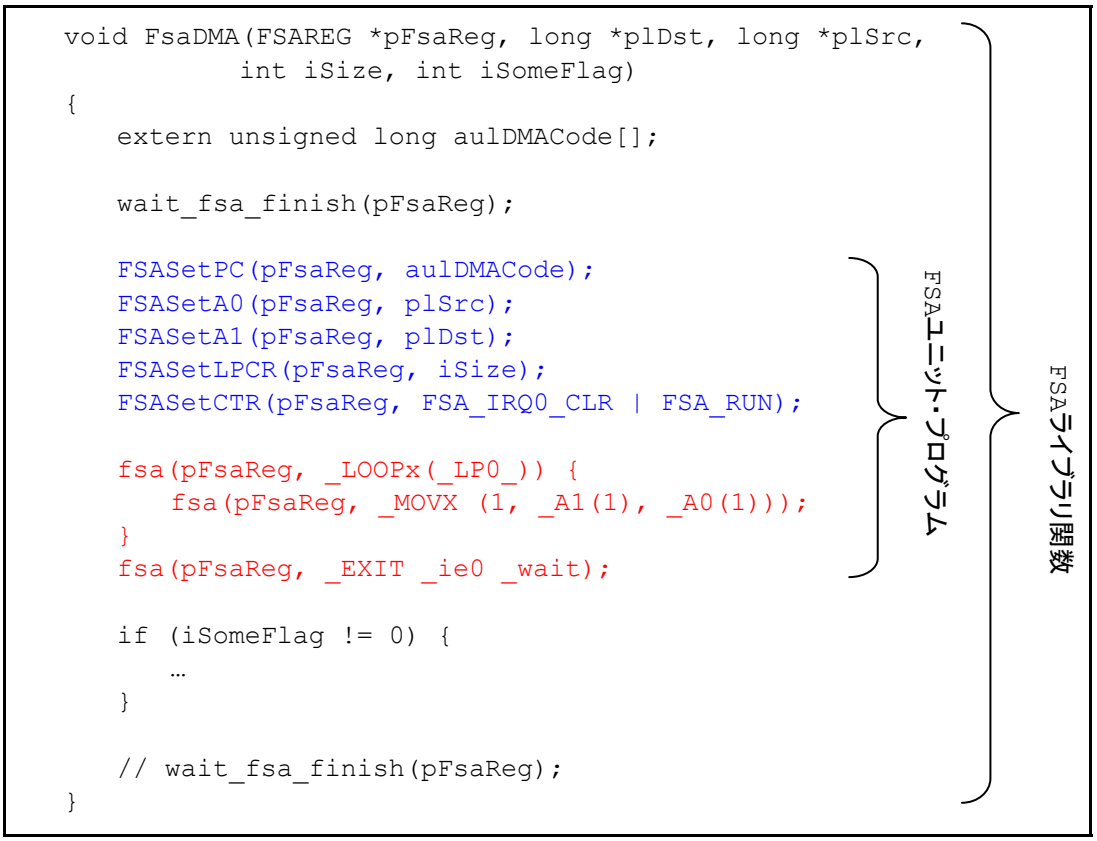

図 1.1　FSA を用いた DMA 転送処理

1.3　適用範囲

本規約は、FSA を使用したソフトウェアおよびそのドキュメントに適用する。

2. C 言語に関する一般的な規約

本規約は、FSA を使用するソフトウェアのポータビリティ性を高めることを目的とするため、一般的な規約はここでは十分に網羅しない。最低限必要な項目のみ列挙する。

2.1 ファイルに関する規約

(1) ファイル名に、日本語等の 2 バイト文字は使用してはならない。

(2) ファイルに関するコメントを、各ファイルの先頭に記述する。この記述は、プログラムファイル、ヘッダファイルの両方に適用する。

2.2 ソースコードに関する規約

(1) コメントに、日本語等の 2 バイト文字は使用してはならない。

(2) 関数に関するコメントを、各関数の先頭に記述する。

(3) インクルード宣言には相対および絶対パスを指定せず、インクルードパスを Makefile などで、渡すように作成する。

－ 悪い例 －

```
#include "c:¥temp¥head1.h"
#include "..¥header¥head1.h"
```

(4) 二重インクルードを避けるために、ヘッダファイルには必ず以下のインクルードガードを入れる。

例：

```
#ifndef HEADFILE_H
#define HEADFILE_H
・・・
#endif // HEADFILE_H
```

(5) C 言語と C++言語が混在する場合を考慮し、C ソースのヘッダファイルには以下の記述をする。

ファイル先頭のインクルードガードの直後：

```
#ifdef __cplusplus
extern "c" {
#endif
```

ファイル末尾のインクルードガードの#endif の直前：

```
#ifdef __cplusplus
}
#endif
```

(6) その他の記法

関数名、変数名の記法や、インデンテーションの記法は同一の FSA モジュール/FSA ライブラリ関数の中で統一する。タブ幅や改行コードも統一する。

3. FSA に関する規約

3.1 FSA インライン・プログラムに関する規約

(1) FSA インライン・プログラムは fsa 関数を用いてホスト CPU のソースコードにインライン形式で記述する。それ以外の方法で記述することを禁止する。

(2) _SETAD 命令でグローバルな配列のアドレスをセットする場合は、_FSA_GLOBAL_()マクロ関数を用いて、間接的に配列アドレスを渡す。
例：
```
fsa(pFsaReg, _SETAD(_A0_,  _FSA_GLOBAL_(someConstantTable), _long));
```

(3) _SETAD 命令では、特別な理由がない限りグローバルな配列シンボルでアドレスを指定する。（アドレスの値を直接記述しない。）

(4) BAR レジスタはライトしない。

(5) FSA のデータ・ワード長が 32 ビットあるいは 24 ビットを前提として作成するプログラムは、ワード型指定に _long を使用し、_int は使用しない。
これは一般的な long と int の違いによる C 言語の規約や注意事項と同様の意図である。つまり、_int は FSA のコンフィギュレーションによっては 16 ビットになりうることから、16 ビットを超える演算が必要な場合には、_long を使用する必要があるためである。

(6) LOOP 命令の中括弧“{”は省略せず、かつ同一行に記述し、ループの対象範囲は次の行から記述する。
例：
```
fsa(pstFsaReg, _LOOP   (8)) {
        fsa(pstFsaReg, _LOOP   (8)) {
                fsa(pstFsaReg, …);
                …
        }
}
```

3.2 FSA ユニット・プログラムに関する規約

(1) FSA のレジスタにアクセスする場合は、提供される下位レベル API 関数越しに行う。
例：
```
FSASetPC(pFsaReg, aulFunctionName);
FSASetA0(pFsaReg, pInput1);
FSASetA1(pFsaReg, pInput2);
FSASetCTR(pFsaReg, FSA_IRQ0_CLR | FSA_RUN);
```

(2) FSASetPC 関数は、他のレジスタ設定より先に（1 番最初に）行う。

(3) FSASetCTR 関数で FSA を起動する際、同時に IRQ0 の Raw Status をクリアする。
例：
```
FSASetCTR(pFsaReg, FSA_IRQ0_CLR | FSA_RUN);
```

(4) _EXIT 命令は、_ie0 オプションを付ける。_ie1 オプションは付けない。
例：
```
fsa(pFsaReg, _EXIT    _ie0 _wait);
```

(5) FSA ユニット・プログラム全体は、対応する FSASetPC 関数と同一の関数内に記述する。
FSA ユニット・プログラムは、C の関数を超えた形でも記述できてしまうが、それは禁止する。

(6) FSA ユニット・プログラム内に、デバッグ用のプログラムを除いて、ホスト側の処理を記述しない。

－ 悪い例 －

```
fsa(pFsaReg, _LOOPx(_LP0_)) {
      printf("……");          // NG
      fsa(pFsaReg, _MOVX (1, _A1(1), _A0(1)));
}
if (iSize > 512) {            // NG
      iSize = 512;            // NG
}                             // NG
fsa(pFsaReg, _EXIT _ie0 _wait);
```

(7) 指定されたデータハザードの回避に必要な遅延スロットを設ける。

3.3 FSA モジュール/FSA ライブラリ関数に関する規約

(1) FSA モジュール/FSA ライブラリ関数は、原則的に C 言語で記述する。(C++言語を使用しない。)

(2) 複数の FSA による動作を考慮して、FSA ライブラリ関数の引数には FSA のインスタンス・ハンドラは第一引数に必須とする。
図 1.1 の例では pFsaReg という仮引数の名前で第一引数に定義している。

(3) FSASetPC 関数で FSA にセットするアドレスは、FSASetPC が実行される関数内で、unsigned long 型の extern の配列として宣言する。
図 1.1 の例では aulDMACode[]として定義している。

(4) FSA の処理終了待ちは、wait_fsa_finish 関数を使用する。
wait_fsa_finish マクロ関数は、fsasys.h に定義されている。ポータビリティ性を確保するために、CPU に依存するようなコード(起床待ちやポーリング等)を直接記述せず、wait_fsa_finish マクロ関数を記述する。

(5) 原則的に、FSASetPC 関数の直前に、wait_fsa_finish 関数を実行する。
wait_fsa_finish マクロ関数を実行して、FSA がレディ状態であることを確認してから、新たな FSA プログラムをキックする。但し、FSA ライブラリ関数内で、複数回 FSA をキックするようなケースで、すでに wait_fsa_finish マクロ関数が実行済みであれば不要である。

(6) FSA の処理終了を待つ wait_fsa_finish 関数は特に必要のない限り FSA ライブラリ関数の最後に記述しない。
図 1.1 の例では分かりやすくするために、関数の最後に wait_fsa_finish をコメントアウトしている。FSA の処理(図 1.1 の場合 DMA 転送)が完了する前に FSA ライブラリ関数を抜けてコール元に戻る。これによって、ホスト CPU は FSA の処理とは依存関係のない処理を FSA と並列に行うことができる。FSA の処理の完了を待つ必要がある場合は、必要に応じてコール元で wait_fsa_finish によって完了を待つ。
ただし、例えば FSA ライブラリ関数が FSA によって計算(除算等)した結果を戻り値として返す仕様の場合には、当然のことながら FSA の処理の完了を待つ必要がある。このような場合は、wait_fsa_finish を FSA ライブラリ関数の最後に記述することを許可する。

(7) 一つの FSA ライブラリ関数が、複数の FSA ユニット・プログラムで構成される場合、FSA のレジスタは破壊されることを想定する必要はない。
FSA ライブラリ関数が、複数の FSA ユニット・プログラムで構成されるケースは少なくない。こうした場合、FSA ユニット・プログラム間で、全く別の FSA ライブラリ関数が動作することを考えると、一般的には FSA のレジスタは破壊されてしまう。つまり FSA のレジスタの値を一旦メモリに退避・復帰する必要があるが、性能の低下に繋がる。そこで、このような FSA の排他処理は上位の OS 等に委ねることを前提とし、FSA ライブラリ関数や FSA モジュール側では考慮しない。

(8) FSA が直接アクセスする定数配列は、グローバルな配列として記述する。その際、__FSA_CONSTANTS__キーワードを付与する。
例：
```
const long someConstantTable[] __FSA_CONSTANTS__ = {
        .....
};
```

(9) FSA ライブラリ関数の内部に閉じて使用する一時メモリは WA レジスタから取得する。
最終的には WA レジスタは固定のアドレスとなる。つまり固定のアドレスを一時メモリとして使用可能ではあるが、汎用性を考慮して必ず WA レジスタから取得することとする。

(10) FSA がアクセスする定数配列は、特に理由のない限り、const 宣言する。

(11) FSA 関連の API 関数の使用に当っては、fsa.h のみをインクルードする。(fsabhv.h、fsamac.h、fsasys.h はインクルードしない。)

ポータビリティ性を確保するため、ホスト CPU が変更されても動作するように配慮されている必要があることから、以下の規約を設ける。

(12) インラインアセンブラは使用してはならない。

(13) #pragma は使用してはならない。

3.4 その他の部分に関する規約

(1) FSA がアクセスする定数配列に対して、FSABhvConstMap 関数を用いてアドレス割り当てを行う関数を用意する。これはエミュレーションの際に必要な処理であり、_FSABHV マクロ定数が有効な場合にのみコンパイルされるようにしておく。
例：
```
#ifdef _FSABHV
unsigned long FSABhvFFTMemMap(unsigned long ulAddr)
{
  ulAddr = FSABhvConstMap(aFFTConst,     ulAddr, sizeof(aFFTConst)     );
  ulAddr = FSABhvConstMap(a512pRFFTConst, ulAddr, sizeof(a512pRFFTConst));

  return ulAddr;
}
#endif   // !_FSABHV
```

(2) ホスト CPU に依存する関数定義等は、fsasys.h に記述する。(他のヘッダファイルは編集しない。)

4. ドキュメントに対する規約

以下では、FSA モジュールまたは FSA ライブラリ関数の作成に伴って必要なドキュメントに書くべき項目を記述する。

4.1 処理内容、引数、戻り値

FSA モジュール/FSA ライブラリ関数の処理の内容と、API 関数の引数および戻り値の型と意味を記載する。

4.2 使用リソース

FSA モジュール/FSA ライブラリ関数が使用する以下のリソースについて記載する。

- 定数テーブル
  定数テーブルを使用する場合、その名前とサイズを記載する。
- 入出力メモリ
  入出力メモリのサイズを記載する。また、そのアドレスが 128 バイト境界にないと動作しないなど、必要なアライメントがあれば記載する。
- ワークメモリ
  ワークメモリを使用する場合、その最大サイズを記載する。
- FSA のレジスタ
  内部で使用されるレジスタのリストを記載する。

4.3 マクロ等

FSA モジュール/FSA ライブラリ関数のソースコードに、マクロなどが定義され、それを変更することで動作等が変わるようにコーディングされている場合、そのマクロとその効果を記載する。

4.4 FSA のコンフィギュレーション

FSA モジュールまたは FSA ライブラリ関数が前提としている FSA のコンフィギュレーションを記載する。以下に一例を示す。

データビット幅	32bit
乗算器構成	32×32bit
アキュミュレータビット幅	72bit
飽和処理	モード 0：8 ビット符号無し飽和 モード 1：16 ビット符号付き飽和 モード 2：32 ビット符号付き飽和
VerInfo レジスタ (Version Information Register)	0x20000000
fsabhv.lib のバージョン	Ver.2.77
特別な変更を前提としたコンフィギュレーション	なし

改訂履歴表

付－1

Rev. No.	日付	ページ	種別	改訂内容（旧内容を含む） および改訂理由
Rev 1.0	2015/01/09	全ページ	新規	新規制定

セイコーエプソン株式会社
マイクロデバイス事業部　デバイス営業部

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

ドキュメントコード：412863800
2015 年 1 月 作成Ⓥ