User's Manual

SMS アセンブラ

ユーザーズマニュアル

ルネサス エレクトロニクス
www.renesas.com

Rev.1.02　2025.12

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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。
10. 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。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

目次

1. はじめに

1.1. 製品について

SMSアセンブラの概要について説明します。

1.1.1. 製品の位置づけ

本製品は、ルネサス製マイコン上で動作するシーケンサ「Snooze Mode Sequencer」(SMS)用のアセンブラです。SMSアセンブラは、ルネサス製統合開発環境の1モジュールとして、他モジュールと連携して動作します。

1.1.2. 概要

本製品はアセンブリ言語で記述されたソース・ファイルを入力として、C 言語の配列型の初期値の形式へアセンブルする機能を備えます。

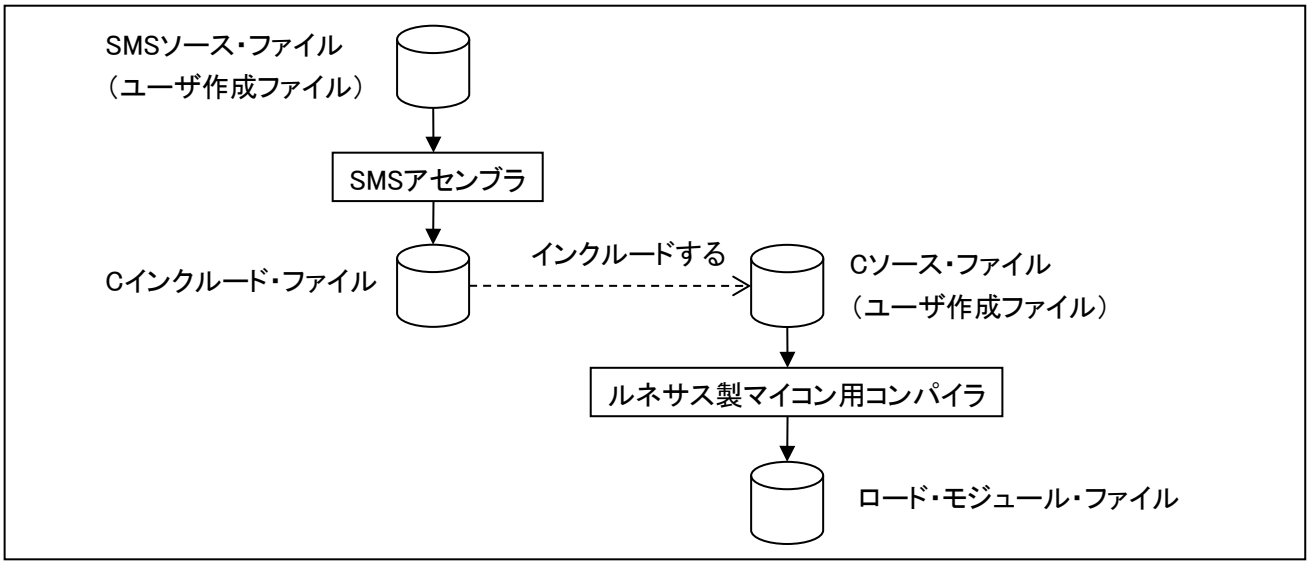

図 1-1 SMSアセンブラの使用例

1.1.3. サポート環境

本製品は、以下の動作環境において動作します。

【ハードウエア環境】

プロセッサ : 1GHz以上（ハイパー・スレッディング、マルチコアCPUに対応）
メイン・メモリ : 1Gバイト以上（Windows 10, および64ビット版の Windows は 2Gバイト以上）, 推奨2Gバイト以上

【OS 環境】

- Windows 10（32ビット版, 64ビット版）
- Windows 11（64ビット版）

1.1.4. 入出力

アセンブラの入出力ファイルは次のとおりです。

表 1.1 アセンブラの入出力ファイル

入出力	名称	説明	デフォルトの拡張子
入力	アセンブリ・ソース・ファイル	アセンブリ言語で記述されたユーザ作成ファイル	任意 （推奨は、smsasm）
	インクルード・ファイル	アセンブリ・ソース・ファイルから取り込むための、アセンブリ言語で記述されたユーザ作成ファイル	任意 （推奨は、inc）
	サブコマンド・ファイル	コマンド・ラインのオプション、入力ファイル指定を記述し、ファイル経由で指定するためのファイル	任意
出力	出力 C インクルード・ファイル	アセンブル結果の C 言語インクルード・ファイル	任意 （推奨は、h）
	エラー・メッセージ・ファイル	エラー・メッセージを出力するファイル	任意
	依存関係ファイル	$include 制御命令による依存関係を出力するファイル	任意 （推奨は、dep）

1.2.　本書の表記について

本書では、次の表記法に従って説明します。

…	同一の形式連続
[ ]	かっこ内を省略可能
{ }	かっこ内の|で区切られた項目から 1 つを選択
△	1 個以上の空白またはタブ
″ ″	″ ″内で囲まれた文字そのもの
CR	復帰(キャリッジ・リターン)
LF	改行(ライン・フィード)
HT	水平タブ
	復改(CRLF)、復帰(CR)、または改行(LF)

【注】メッセージ、文字列は″ ″で囲みます

2. 入力規約

コマンド・ラインからの入力については「3.1 コマンド入力形式」 を参照してください。
アセンブリ記述については「4.1.2 記述方法」を参照してください。

3. 操作方法

3.1. コマンド入力形式

コマンド・ラインからは、次のように入力します。

```
smsasm[△オプション]…△ファイル名[△オプション]…
```

1 度のアセンブラ実行につき、指定できる入力ファイルは 1 つのみです。ファイル名に空白が含まれる場合は″ ″で囲んでください。

3.2. オプションの種類と機能

本節では、アセンブラのオプションについて説明します。

オプションの大文字、小文字は区別します。一覧にないオプションを指定した場合はエラーになります。同じオプションを複数回指定した場合は、すべて有効になるか、最後の指定が有効になります。詳細は「3.2.2 オプション詳細」を参照してください。

3.2.1. オプション一覧

オプションの説明を以下に示します。

表 3.1 オプション一覧

オプション	説明
-V	smsasmのバージョン情報を表示します。
-help	smsasmのオプションの説明を表示します。
-o	出力Cインクルード・ファイル名を指定します。
-D	シンボルを定義します。
-U	-Dオプションによるシンボルの定義を解除します。
-I	インクルード・ファイルを検索するフォルダを指定します。
-character_set	日本語／中国語の文字コードを指定します。
-error_file	エラー・メッセージをファイルに出力します。
-no_warning	指定した警告メッセージの出力を抑止します。
@	サブコマンド・ファイルを指定します。
-MM -MMD -MP -MT -MF	依存関係ファイルに$include制御命令による依存関係を出力します。

3.2.2. オプション詳細

3.2.2.1. -V

smsasmのバージョン情報を表示します。

[指定形式]

-V

[詳細説明]

smsasmのバージョン情報を標準エラー出力に出力します。アセンブルおよび依存関係ファイル出力は行いません。

[使用例]

smsasm -V

3.2.2.2. -help

smsasmのオプションの説明を表示します。

[指定形式]

-help

[詳細説明]

smsasmのオプションの説明を標準エラー出力に出力します。アセンブルおよび依存関係ファイル出力は行いません。

[使用例]

smsasm -help

3.2.2.3. -o

出力Cインクルード・ファイル名を指定します。

[指定形式]

-o[△]*file*

[詳細説明]

- 出力 C インクルード・ファイル名を *file* に指定します。
- *file* がすでに存在する場合は、そのファイルを上書きします。
- 本オプションを指定しても、アセンブル処理でエラーが発生した場合は出力 C インクルード・ファイルの出力は行いません。
- -MM オプション指定時以外は本オプションを省略できません。省略した場合エラーになります。
- 出力 C インクルード・ファイル名を複数指定した場合は後から指定した方が有効になります。
- *file* を省略した場合はエラーとなります。

[使用例]

出力Cインクルード・ファイル名をsample.hに指定します。

```
smsasm smsmain.smsasm -o sample.h
```

次の例は上記の例と同等です。

```
smsasm smsmain.smsasm -osample.h
```

3.2.2.4. -D

シンボルを定義します。

[指定形式]

-D[△]*name*[=*def*][,*name*[=*def*]]...

[詳細説明]

- シンボルとして *name* を定義します。
- アセンブリ・ソース・プログラムの先頭で ″.DEFINE *name def*″ を記述するのと同等です。
- *name* を省略した場合はエラーとなります。
- =*def* を省略した場合は *def* を 1 とみなします。
- 本オプションは複数回指定することができます。
- 本オプション同士、または本オプションと-U オプションで同じ *name* を同時に指定した場合は、後から指定した方が有効になります。

[使用例]

シンボルとしてZERO=0を定義します。

```
smsasm smsmain.smsasm -o sample.h -D ZERO=0
```

次の例は上記の例と同等です。

```
smsasm smsmain.smsasm -o sample.h -DZERO=0
```

3.2.2.5. -U

-D オプションによるシンボルの定義を解除します。

[指定形式]

-U[△]*name*[,*name*]...

[詳細説明]

- -D オプションによるユーザ定義シンボル *name* の定義を解除します。
- *name* を省略した場合はエラーとなります。
- 本オプションは複数回指定することができます。
- 本オプションと-D オプションで同じ *name* を同時に指定した場合は、後から指定した方が有効になります。

[使用例]

-Dオプションによるシンボル ZERO の定義を解除します。

```
smsasm smsmain.smsasm -o sample.h -D ZERO=0 -U ZERO
```

次の例は上記の例と同等です。

```
smsasm smsmain.smsasm -o sample.h -D ZERO=0 -UZERO
```

3.2.2.6. -I

インクルード・ファイルを検索するフォルダを指定します。

[指定形式]

-I[△]*path*[,*path*]...

[詳細説明]

- $include 制御命令でインクルード・ファイル名をファイル名のみまたは相対パスで指定した場合に、インクルード・ファイルを検索するフォルダを *path* に指定します。インクルード・ファイルの検索は以下の順番で行います。

  <1> 本オプションで指定したフォルダ
  <2> $include制御命令が記述されたアセンブリ・ソース・ファイルまたはインクルード・ファイルのあるフォルダ
  <3> カレント・フォルダ（smsasmを起動したフォルダ）

- 本オプションを省略した場合、インクルード・ファイルを検索するフォルダは、アセンブリ・ソース・ファイルまたはインクルード・ファイルのあるフォルダとカレント・フォルダのみとなります。
- *path* を省略した場合はエラーとなります。
- 本オプションを複数回指定した場合、すべての指定が有効になります。

[使用例]

インクルード・ファイルをD:¥include, D:¥src, カレント・フォルダの順で検索します。

```
smsasm D:¥src¥smsmain.smsasm -o sample.h -I D:¥include
```

次の例は上記の例と同等です。

```
smsasm D:¥src¥smsmain.smsasm -o sample.h -ID:¥include
```

3.2.2.7. -character_set

日本語／中国語の文字コードを指定します。

[指定形式]

```
-character_set={none|sjis|euc_jp|utf8|big5|gb2312}
```

[詳細説明]

- 入力ファイル中の日本語／中国語のコメントおよび文字列に対して、使用する文字コードを指定します。
- 指定できるパラメータを以下に示します。これ以外を指定した場合はエラーとなります。

パラメータ	説明
none	日本語／中国語の文字コードを処理しません
sjis	Shift-JIS
euc_jp	EUC（日本語）
utf8	UTF-8
big5	繁体字中国語
gb2312	簡体字中国語

- 入力ファイル中で使用している文字コードと異なるものを指定した場合、動作は保証しません。
- 本オプションを省略した場合、文字コードを UTF-8 として扱います。
- 本オプションを複数回指定した場合は最後の指定が有効になります。

[使用例]

入力ファイル中の日本語のコメントおよび文字列に対して使用する文字コードにEUCを指定します。

```
smsasm smsmain.smsasm -o sample.h -character_set=euc_jp
```

3.2.2.8. -error_file

エラー・メッセージをファイルに出力します。

[指定形式]

-error_file=*file*

[詳細説明]

- エラー・メッセージを標準エラー出力およびファイル *file* に出力します。
- *file* がすでに存在する場合は、そのファイルを上書きします。
- *file* を省略した場合はエラーとなります。
- 本オプションを複数回指定した場合は最後の指定が有効になります。

[使用例]

エラー・メッセージを標準エラー出力およびファイルerrに出力します

```
smsasm smsmain.smsasm -o sample.h -error_file=err
```

3.2.2.9. -no_warning

指定した警告メッセージの出力を抑止します。

[指定形式]

-no_warning={*num*|*num1*-*num2*}[, ...]

[詳細説明]

- 指定した警告メッセージの出力を抑止します。
- *num, num1, num2* には警告メッセージ番号を指定します。存在しない警告メッセージ番号を指定した場合は無視します。
- *num* または *num1* および *num2* を省略した場合はエラーとなります。
- *num1*-*num2* の形式で指定すると、その範囲に含まれる警告メッセージ番号を指定したものとみなします。
- 本オプションで指定できる警告メッセージ番号は、W に続く 7 桁の数字のうち下位 4 桁です。
- 本オプションを複数回指定した場合はすべての指定が有効になります。

[使用例]

警告メッセージW0580304、W0580306の出力を抑止します。

```
smsasm smsmain.smsasm -o sample.h -no_warning=304,306
```

3.2.2.10. @

サブコマンド・ファイルを指定します。
サブコマンド・ファイルとは、アセンブラに対して指定するオプションやファイル名を記述したファイルです。
サブコマンド・ファイルの内容はコマンド・ラインの引数のように扱われます。

[指定形式]

@*file*

[詳細説明]

- *file* をサブコマンド・ファイルとして扱います。
- *file* が存在しない場合はエラーとなります。
- *file* を省略した場合はエラーとなります。
- 本オプションを複数回指定した場合はすべての指定が有効になります。
- サブコマンド・ファイルの記述については、「3.3 サブコマンド・ファイルの記述に関する注意事項」を参照ください。

[使用例]

command.txtをサブコマンド・ファイルとして扱います。

```
smsasm smsmain.smsasm -o sample.h @command.txt
```

3.2.2.11. -MM / -MMD / -MP / -MT / -MF

依存関係ファイルに$include制御命令による依存関係を出力します。

[指定形式]

-MM
-MMD
-MP
-MT=*name*
-MF=*file*

[詳細説明]

- -MM オプションを指定すると、$include 制御命令による依存関係を依存関係ファイルに出力して、処理を中断します。C インクルード・ファイルは出力しません。-o オプションは無視します。
- -MMD オプションを指定すると、$include 制御命令による依存関係を依存関係ファイルに出力します。アセンブル処理は継続し、C インクルード・ファイルも出力します。
- -MM / -MMD オプションを複数回指定した場合は、最後の指定が有効になります。
- -MM / -MMD オプションを指定せずに-MP, -MT, -MF オプションを指定した場合は、-MP, -MT, -MF オプション指定を無視します。
- -MP オプションを指定すると、依存関係にあるインクルード・ファイルパスをターゲットとして依存関係ファイルに出力します。
  複数回指定した場合は、1 回指定した場合と同じ意味になります。このとき、警告を出力しません。
- -MT オプションを指定すると、依存関係ファイルに出力するターゲット名が-MT オプションで指定した文字列になります。指定したターゲット名は-MP オプションを指定した際に出力されるターゲット名には影響しません。
  指定しない場合、以下の順番でターゲット名を決定します。
  <1> -o オプションで指定されたファイル名
  <2> 入力ファイル名の拡張子を.h に変えた名前

  複数回指定した場合は、各オプションで指定した文字列を、空白 1 個を挟んで連結した文字列をターゲット名とします。
- *name* を省略した場合はエラーとなります。
- -MF オプションを指定すると、依存関係ファイル名が-MF オプションで指定したファイル名になります。
  指定しない場合、入力ファイル名の拡張子を.dep に変えた名前を依存関係ファイル名とします。
  複数回指定した場合は、最後の指定が有効になります。
- *file* に実在するフォルダのパスを指定した場合、指定したフォルダに入力ファイル名の拡張子を.dep に変えたファイル名で出力します。
- *file* に実在しないフォルダのパスを指定した場合、最下層フォルダ名をファイル名とします。

[使用例]

入力ファイル
sms_main.smsasm
$include “a.inc”
a.inc
$include “b.inc”
b.inc
($includeなし)

依存関係ファイル sms_main.dep を出力します。sample.h は出力しません。

```
smsasm D:¥src¥sms_main.smsasm -o sample.h -MM
```

sms_main.dep

```
sample.h: sms_main.smsasm
sample.h: a.inc
sample.h: b.inc
```

インクルード・ファイル名をターゲットとして依存関係ファイル sms_main.dep に出力します。

```
smsasm D:¥src¥sms_main.smsasm -o sample.h -MM -MP
```

sms_main.dep

```
sample.h: sms_main.smsasm
sample.h: a.inc
a.inc
sample.h: b.inc
b.inc
```

すべての-MTオプションで指定した文字列を空白で連結した文字列をターゲット名とします。

```
smsasm D:¥src¥sms_main.smsasm -o sample.h -MM -MT=t1 -MT=t2 -MT=t3
```

sms_main.dep

```
t1△t2△t3: sms_main.smsasm
t1△t2△t3: a.inc
t1△t2△t3: b.inc
```

dependencies.dep というファイル名で依存関係ファイルを出力します。

```
smsasm D:¥src¥sms_main.smsasm -o sample.h -MM -MF=dependencies.dep
```

dependencies.dep

```
sample.h: sms_main.smsasm
sample.h: a.inc
sample.h: b.inc
```

3.3. サブコマンド・ファイルの記述に関する注意事項

- 指定する引数は複数行に分けて記述することができます。ただしオプション指定やファイル名の途中で改行することはできません。
- サブコマンド・ファイル内でサブコマンド・オプションを指定する場合、同名のサブコマンド・ファイルを指定することはできません。
- 以下の文字は特殊文字として扱います。

″（ダブルクォーテーション）	次のダブルクォーテーションまでの空白を含む文字列を連続した文字列として扱います。
#（シャープ）または ；（セミコロン）	行末までをコメントとして扱います。
¥（バックスラッシュ）	ダブルクォーテーションの直前につけるとダブルクォーテーションを特殊文字として扱いません。

4. アセンブリ言語仕様

この章では、SMS アセンブラの言語仕様について説明します。

4.1. ソースの基本記述

この節では、ソースの記述方法、式と演算子などについて説明します。

4.1.1. 基本構成

ソース・プログラムの構成について説明します。

ソース・プログラムは、セクションと呼ばれるブロックで構成されます。セクションには2つの種類があります。

1つはSMSプログラムを構成するセクションです。アセンブラがセクション内に書かれたアセンブリ言語をアセンブルし、出力Cインクルード・ファイルにアセンブル後の機械語命令列を出力します。

もう1つは、ソース・ファイルに記述した内容をそのまま出力するセクションです。SMSプログラムと組み合わせて使うC言語の処理などを記述します。

ユーザの基本的な使用手順は次のようになります。

1. SMS 用プログラムをアセンブリ言語で記述する。
2. SMS アセンブラで、1.のプログラムをアセンブルし、C 言語の出力ファイルを得る。
3. RL78 用のプログラムを C 言語で記述し、その中に 2.のファイルをインクルードする。また、インクルードした SMS 用プログラムを命令レジスタ、汎用レジスタに書き込む処理（初期化処理）と、SMS を起動する処理を記述する。
4. 3.の RL78 用プログラムをコンパイルし、実行形式ファイルを出力する。

入力例：example.smsasm

```
.psection   ; SMS 汎用レジスタの初期値部（RL78 用 C ソース部分）
#if !defined(EXAMPLE_SMSASM)
#define EXAMPLE_SMSASM

extern unsigned short var1[2], var2[3];
struct bitsfr { unsigned char b0:1, b1:1, b2:1, b3:1, b4:1, b5:1, b6:1, b7:1 };
#define P1 (*(volatile struct bitsfr __near *)0xff01)

#define sms_greg1_initializer ¥
   0x0000, /* smsg1 */ ¥
   &var1,   /* smsg2 */ ¥
   0x0000, /* smsg3 */ ¥
   &var2,   /* smsg4 */ ¥
   0x0000, /* smsg5 */ ¥
   0x0000, /* smsg6 */ ¥
   0x0000, /* smsg7 */ ¥
   &P1     /* smsg8 */

.section sms_program1   ; SMS 用プログラム（SMS 用 ASM ソース部分）
            movw smsg1, [smsg2 + 0]
            movw smsg3, [smsg4 + 4]
            subw smsg1, smsg3
            bc $L1
            while1 [smsg8 + 0].7
L1:
            movw [smsg2 + 2], smsg1
            finish

.psection   ; ファイルの終端部（RL78 用 C ソース部分）
#endif /* EXAMPLE_SMSASM */
```

出力例：example_output.h

```
/*
 *   SMSASM V1.00.00.01 [18 Mar 2020]
 *   assembled at Wed Mar 18 12:34:56 2020
 */

/* psection */
#if !defined(EXAMPLE_SMSASM)
#define EXAMPLE_SMSASM

extern unsigned short var1[2], var2[3];
struct bitsfr { unsigned char b0:1, b1:1, b2:1, b3:1, b4:1, b5:1, b6:1, b7:1 };
#define P1 (*(volatile struct bitsfr __near *)0xff01)

#define sms_greg1_initializer ¥
   0x0000, /* smsg1 */ ¥
   &var1,   /* smsg2 */ ¥
   0x0000, /* smsg3 */ ¥
   &var2,   /* smsg4 */ ¥
   0x0000, /* smsg5 */ ¥
   0x0000, /* smsg6 */ ¥
   0x0000, /* smsg7 */ ¥
   &P1     /* smsg8 */

/* end psection */

#define SIZEOF_sms_program1 (7)

#define sms_program1 ¥
          0x3210, /*   0: MOVW      SMSG1, [SMSG2+0] */ ¥
          0x3434, /*   1: MOVW      SMSG3, [SMSG4+4] */ ¥
          0x7131, /*   2: SUBW      SMSG1, SMSG3 */ ¥
          0x8020, /*   3: BC        $2 */ ¥
          0xa870, /*   4: WHILE1    [SMSG8+0].7 */ ¥
          0x2212, /*   5: MOVW      [SMSG2+2], SMSG1 */ ¥
          0xf000   /*   6: FINISH */

/* psection */
#endif /* EXAMPLE_SMSASM */

/* end psection */
```

出力プログラムの利用例：rl78main.c

```
#include "example_output.h"

unsigned short sms_greg1[] = {
    sms_greg1_initializer
};

void sms_program1_start(void) {
    static const unsigned short iregs[SIZEOF_sms_program1] = {
        sms_program1
    };
    int i;
    unsigned short* dst = 0x0380; /* SMSI0 のアドレス */
    SMSEN = 1;
    for (i = 0; i < SIZEOF_sms_program1 ; ++i) {
        *dst = iregs[i];
        ++dst;
    }
    dst = 0x03c2; /* SMSG1 のアドレス */
    for (i = 0; i < sizeof(sms_greg1)/sizeof(sms_greg1[0]); ++i) {
        *dst = sms_greg1[i];
        ++dst;
    }
    SMSC.SMSSTART = 1;
    /* SMS へトリガ送信 */
    /* CPU を間欠状態に遷移させる */
}
```

【注】アドレスや初期化手順は一例です。実際の仕様や手順についてはデバイスのユーザーズ・マニュアルを参照してください。

4.1.2.　記述方法

アセンブリ言語のソース・プログラムは、文で構成されます。
1つの文は、「文字セット」に含まれる文字を使って記述します。アセンブリ言語文は、”シンボル”、”ニモニック”、”オペランド”、および”コメント”から構成されます。

[シンボル][:]　[ニモニック]　[オペランド], [オペランド]　;[コメント] ⏎

各欄は空白、タブ、コロン(:)、またはセミコロン(;)で区切ります。1行の最大文字数は4,294,967,294 (=0xFFFFFFFE)(理論値)です。ただし、実際にはメモリ量により制限されます。

文の記述方法は自由記述形式で、シンボル欄、ニモニック欄、オペランド欄、コメント欄の順序が正しければ任意のカラムから記述することができます。ただし、1つの文は1行以内で記述しなければなりません。

シンボル欄にシンボルを記述する場合は、コロン、1つ以上の空白、またはタブで区切ります。
ただし、コロンか空白かタブかはニモニックで記述する命令によります。また、コロンの前後には任意の数の空白またはタブを記述することができます。

オペランド欄の記述が必要な場合は、1つ以上の空白またはタブで区切ります。コメント欄にコメントを記述する場合は、セミコロンまたはシャープで区切ります。セミコロンまたはシャープの前後には任意の数の空白またはタブを記述することができます。

アセンブリ言語文の最後は改行(リターン)します。

4.1.2.1.　文字セット

アセンブラがサポートするソース・プログラムで使用できる文字は次の2つから構成されます。

- 言語文字
- 文字データ

a)　言語文字
ソース上で命令を記述するために使用する文字です。言語文字を機能別に細分類すると下表になります。

表 4.1 言語文字の文字セット

細分類の総称名		文字
数字		0 1 2 3 4 5 6 7 8 9
英字	大文字	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
	小文字	a b c d e f g h i j k l m n o p q r s t u v w x y z
英字相当文字		_(アンダースコア)
特殊文字	特殊文字 1	. , : ; * / + − < > ( ) $ = ! & # [ ] % << >> | ^ ~ △
	特殊文字 2	LF, CR LF, HT

【注】英字(英字相当文字も英字に含める)と数字をまとめて英数字といいます。
【注】予約語または数値定数に記述した英小文字は、対応する英大文字として解釈されます。
【注】英小文字をユーザ定義シンボルの中に使った場合、英大文字、英小文字の区別をして解釈されます。
【注】エスケープ・シーケンスは非対応です。

特殊文字 1 の用途を以下に示します。
以下の使用方法以外で文字データおよびコメント部以外のソース・プログラム中に記述したときはエラーとなります。

表 4.2 特殊文字1とその用途

特殊文字 1	用途
.(ピリオド)	ビット位置指定子 疑似命令の開始記号
,(カンマ)	オペランドの区切り
:(コロン)	ラベルの区切り
;(セミコロン)	コメントの開始
*	乗算演算子
/	除算演算子
+	正符号 加算演算子
-(ハイフン)	負符号 減算演算子
<	比較演算子 シフト演算子
>	比較演算子 シフト演算子
( )	演算順序の指定
$	制御命令の開始記号 相対アドレッシング開始記号
=	比較演算子
!	比較演算子
&	ビット論理演算子 論理演算子
#	コメントの開始
[ ]	インダイレクト表示記号
%	剰余演算子
|	ビット論理演算子 論理演算子
^	ビット論理演算子
~	ビット論理演算子
△(空白またはタブ)	各欄の区切り記号

特殊文字 2 の用途を以下に示します。

表 4.3 特殊文字2とその用途

特殊文字 2	用途
CR LF	ソース・プログラムの改行およびカラム位置を行頭に移動させる
LF	ソース・プログラムの改行
HT	ソース・プログラム文のカラム位置を移動させる

b）　文字データ
文字データはコメントや制御命令部を記述するために使用する文字です。英小文字と英大文字は区別して扱います。

4.1.2.2.　定数

数値定数として10進数および16進数を記述可能です。

数値定数の種類	表記方法	表記例
10 進数	数値をそのまま記述	128
16 進数	数値の前に"0x"または"0X"を記述 10 進数字(0-9)から始まり、数値の後に"h"または"H"を記述	0xA6, 0XA6 6Ah, 6AH

【注】16進数のプレフィックス(0x/0X)とサフィックス(h/H)はどちらか片方のみ記述できます。

4.1.2.3. シンボル

シンボルとは、数値データやアドレスなどに付けた名前のことです。シンボルを使用することにより、ソースの内容がわかりやすくなります。シンボルを参照した場合、そのシンボルに対して定義した値を指定したものとして扱われます。

本アセンブラでのシンボルは、次の種類に分けられます。

- ラベル
  行の先頭からコロン(:)までの部分に記述したシンボルです。このシンボルはアドレス値を持ちます。アドレス値の範囲は0 ～ 31(0x00 ～ 0x1F)です。
- セクション名
  セクション定義疑似命令のシンボル欄に記述したシンボルです。
  このシンボルは値を持ちません。
- マクロ名
  マクロ定義疑似命令で定義したシンボルです。アセンブル開始前に、定義した文字列に置き換えられます。

(a) シンボルの種類

シンボルはその使用目的、定義方法によって次に示す種類に分けられます。

シンボル欄に複数のシンボルを記述することはできません。また、1行にいずれかのシンボルを1個しか定義できません。

シンボルの種類	使用目的	定義方法
ラベル	ラベルの位置のアドレスを参照するために使用します。 なお、疑似命令に付加されたラベルはその疑似命令の直前のセクションに含まれるとみなされます。	シンボルのあとにコロン(:)を付けることにより定義します。
セクション名	出力 C インクルード・ファイル中でマクロ名として使用します。	セクション定義疑似命令のシンボル欄に定義します。
マクロ名	アセンブリ・ソース中で定数や文字列への置き換えに使用します。	マクロ定義疑似命令のシンボル欄に記述します。 シンボル名と置換文字列は 1 個以上の空白またはタブで区切ります。 複数行にまたがる場合は行末に¥記号が必要です。

(b)　シンボル記述上の規則

シンボルは次の規則に基づいて記述します。

- シンボルは英数字および英字相当文字(_)で構成します。ただし、先頭文字に数字(0 ～ 9)は使用できません。
- シンボルの最大文字数は 4,294,967,294(=0xFFFFFFFE)(理論値)です。ただし、実際には利用可能なメモリ量に依存します。
- シンボルとして予約語は使用できません。予約語については「4.1.3 予約語」を参照してください。
- 同一シンボルを二度以上定義することはできません。
- ラベルを記述する場合は、ラベルの直後にコロン(:)を記述します。それ以外は空白、またはタブでニモニック欄と区切ります。

正しいシンボルの例を以下に示します。

```
.SECTION  CODE01                    ;“CODE01” はセクション名
.DEFINE  VAR01 0x10                 ;“VAR01” はマクロ名
LAB01:  MOV [SMSG1 + 1], SMSG2      ;“LAB01” はラベル
```

誤ったシンボルの例を以下に示します。

```
.DEFINE  1ABC 0x3                   ;  先頭文字に数字は使用できません。
LAB  MOV SMSG1, [SMSG10 + 2]        ;“LAB” ラベルです。ニモニック欄とコロン(:)で区切ります。
.DEFINE  FLAG: 0x10                 ;  マクロ名にはコロン(:)が必要ありません。
```

シンボルのみで構成される文の例を以下に示します。

```
ABCD:                               ; ABCD  がラベルとして定義されます。
```

(c)　シンボルに関する注意事項

アセンブラ生成シンボル(「4.1.4 アセンブラ生成シンボル」を参照)を記述する場合、多重定義でエラーとなる可能性があるのでアセンブラ生成シンボルは使用しないでください。

4.1.2.4.　コメント

ソース行中の、セミコロン(;)またはシャープ(#)から行末までをコメントとして扱います。コメント欄の記述は、アセンブル処理の対象とはなりません。

例

```
# これはコメント  です
LABEL1:  MOV SMSG1, [SMSG2 + 2]    # これはコメントです

LABEL2:  MOV [SMSG3 + 1], SMSG4    ;  これはコメントです
;
; BEGIN LOOP HERE
;
```

4.1.3.　予約語

SMS アセンブラの予約語は次のとおりです。

- 命令（MOV, MOVW, ADDW など）
- 疑似命令
- 制御命令
- レジスタ名

4.1.4.　アセンブラ生成シンボル

アセンブラが生成するシンボルについては、「4.10.1　セクション定義疑似命令」を参照してください。

4.2.　式と演算子

式とはシンボル、数値定数、前述の 2 つに演算子を付加したものまたは演算子で結合したものです。

【注】式の要素として指定可能なシンボルは-D オプションまたは.DEFINE 疑似命令で定義した数値定数マクロのみです。

式を構成する演算子以外の要素を項といい、記述された左側から順に第1項、第2項、… と呼びます。

演算子には「表 4.4 演算子の種類」に示すものがあり、演算実行上の優先順位が「表 4.5 演算子の優先順位」のように決められています。演算の順序を変更するにはかっこ "( )" を使用します。

表 4.4 演算子の種類

演算子の種類	演算子
算術演算子	+ − * / % +符号 −符号
ビット論理演算子	~ & | ^
シフト演算子	>> <<
比較演算子	== != > >= < <=
論理演算子	&& ||
その他の演算子	( )

表 4.5 演算子の優先順位

優先度	優先順位	演算子
高い	1	( )
	2	+符号 −符号 ~
	3	* / %
	4	+ −
	5	>> <<
	6	> >= < <=
	7	== !=
	8	&
	9	^
	10	|
	11	&&
低い	12	||

式の演算は次の規則に従います。

- 演算の順序は演算子の優先順位に従います。
  同一順位の場合は、左から右に演算されます。単項演算子の場合は、右から左に演算されます。
- かっこ "( )" の中の演算は、かっこの外の演算に先立って行われます。
- 式の演算は 32 ビットで行います。
  式中に記述した定数、式の評価途中、および評価結果が 32 ビットを越えた場合は、下位 32 ビットを有効とします。その場合エラーは出力されません。
- 演算の各項は符号なし整数として扱われますが、以下の場合は符号付き整数として扱われます。

  乗算、除算、剰余算、論理シフトの第 2 項

- 除数がゼロの場合はエラーとなります。
- 負の値は 2 の補数形式となります。

4.3.　算術演算子

算術演算子には次のものがあります。

演算子	概要
+	第 1 項と第 2 項の値の加算
−	第 1 項と第 2 項の値の減算
*	第 1 項と第 2 項の値の乗算
/	第 1 項と第 2 項の値の剰余算を行い、整数部を求める
%	第 1 項と第 2 項の値の剰余算を行い、余りを求める
+符号	項の値をそのまま返す
−符号	項の値の 2 の補数を求める

4.4.　ビット論理演算子

ビット論理演算子には次のものがあります。

演算子	概要
~	項のビットごとの論理否定を求める
&	第 1 項の値と第 2 項の値のビットごとの論理積を求める
|	第 1 項の値と第 2 項の値のビットごとの論理和を求める
^	第 1 項の値と第 2 項の値のビットごとの排他的論理和を求める

4.5.　シフト演算子

シフト演算子には次のものがあります。

演算子	概要
>>	第 1 項の値を第 2 項で示す値分だけ右シフトした値を求める
<<	第 1 項の値を第 2 項で示す値分だけ左シフトした値を求める

4.6.　比較演算子

比較演算子には次のものがあります。

演算子	概要
==	第 1 項の値と第 2 項の値が等しいかどうか比較
!=	第 1 項の値と第 2 項の値が等しくないかどうか比較
>	第 1 項の値が第 2 項の値より大きいかどうか比較
>=	第 1 項の値が第 2 項の値より大きい、または等しいかどうか比較
<	第 1 項の値が第 2 項の値より小さいかどうか比較
<=	第 1 項の値が第 2 項の値より小さい、または等しいかどうか比較

4.7.　論理演算子

論理演算子には次のものがあります。

演算子	概要
&&	第 1 項の値と第 2 項の値の論理積を求める
||	第 1 項の値と第 2 項の値の論理和を求める

4.8. その他の演算子

その他の演算子には次のものがあります。

演算子	概要
( )	( ) 内の演算を優先して行う

4.9.　命令の記述方法

命令には機械命令、疑似命令、制御命令の3種類があります。
機械命令については、デバイスのユーザーズ・マニュアルを参照してください。

(1)　オペランドの値のサイズとアドレス範囲
命令のオペランドとして記述可能な数値／ラベルの値のサイズとアドレス範囲には条件があります。機械命令の場合は、各機械命令のオペランドの表現形式により、記述可能なオペランドのサイズとアドレス範囲に条件があります。これらの条件を次に示します。

表 4.6 機械命令のオペランド値の範囲

オペランドの表現形式	値の範囲
Byte	3 ビット値： 0x0 ～ 0x7
addr5	8 ビット値： 0x00 ～ 0x1F, 0xE1 ～ 0xFF (−31 ～ 31)
bit	3 ビット値： 0 ～ 7
IM1	8 ビット値： 0x00 ～ 0xFF
IM2	3 ビット値： 0x0 ～ 0x7
SMSGn, SMSGm	n,m：4 ビット値： 0 ～ 15

(2)　命令の要求するオペランドのサイズ
命令には機械命令と疑似命令がありますが、オペランドとしてイミディエイト・データまたはシンボルを要求する命令については、各命令により要求するオペランドのサイズが異なります。したがって、命令の要求するオペランドのサイズ以上のデータを記述すると、エラーとなります。
また、式の評価は計算結果および途中も含めて 32 ビットで行うため、オーバフローした場合でも 32 ビット値として扱います。

(3)　シンボルの参照方向
シンボルの参照方向には、後方参照と前方参照があります。

- 後方参照：オペランドとして参照するシンボルがそれ以前の行で定義されています。
- 前方参照：オペランドとして参照するシンボルがそれ以降の行で定義されています。

例を以下に示します。

例

```
.section sms_prog
L1:                    ← 後方参照
BC   $L1
BC   $L2
L2:                    ← 前方参照
```

条件分岐命令で参照するラベルシンボルは、後方参照、前方参照とも可能です。
.DEFINE疑似命令によるマクロシンボルは、後方参照のみ可能です。

4.10. 疑似命令

疑似命令とは、アセンブラが一連の処理を行う際に必要な各種の指示を行うものです。
機械命令はアセンブルの結果、機械語に変換されますが、疑似命令は原則として機械語に変換されません。

疑似命令は主に次の機能を持ちます。

- ソースの記述を容易にします。
- アセンブラが処理を行うために必要となる情報を与えます。

以下に疑似命令の種類を示します。

表 4.7 疑似命令一覧

種類	疑似命令
セクション定義疑似命令	.SECTION , .PSECTION
マクロ定義疑似命令	.DEFINE , .UNDEF

4.10.1. セクション定義疑似命令

セクション定義疑似命令は、セクションの開始、および終了を指示するための疑似命令です。
セクションは、SMSアセンブラにおけるSMSプログラムの単位です。
同名のセクションを複数記述することはできません。

例

```
.SECTION    program1
      :
.SECTION    program2
      :
.SECTION    program3
      :
```

セクション定義疑似命令には、次のものがあります。

表 4.8 セクション定義疑似命令

疑似命令	概要
.SECTION	SMSプログラム用セクションの開始を指示します
.PSECTION	出力Cインクルード・ファイルにそのまま出力されるセクションの開始を指示します

4.10.1.1. .SECTION

[指定形式]

.SECTION　name

[詳細説明]

- 1 つの SMS プログラムを格納する領域を定義します。
- 本疑似命令は、次にセクション定義疑似命令が記述されるまで有効です。
- 本疑似命令を記述する前に機械命令を記述すると、エラーになります。
- 本疑似命令で指定したセクション名は、出力 C インクルード・ファイル中でマクロ名として使います。
  アセンブル後の機械語命令列を表現した、2 バイト整数型配列の初期値列を持つマクロになります。
  RL78 用の C ソース・プログラム中で、SMS の命令レジスタ列へ書き込む値として使用できます。
- また、"SIZEOF_セクション名"というマクロ名で、出力した初期値の数を示すマクロを出力します。

[使用例]

入力

```
.SECTION　prog1
MOV SMSG1, [SMSG2+3]
MOV [SMSG4+5], SMSG6
```

出力

```
#define SIZEOF_prog1 (2)

#define prog1 ¥
        0x3213, /* 0: MOV SMSG1, [SMSG2+3] */ ¥
        0x4650 /* 1: MOV [SMSG4+5], SMSG6 */
```

4.10.1.2. .PSECTION

[指定形式]

.PSECTION

[詳細説明]

- 出力 C インクルード・ファイルにそのまま出力する領域を定義します。
  セクション内に書かれた文字列を、出力ファイル中にそのまま出力します。
- 出力内容の上下に、/* psection */、/* end psection */ という C ソース・コメントを付加します。
- 本疑似命令は、次にセクション定義疑似命令が記述されるまで有効です。
- 本疑似命令ではセクション名は指定できません。

[使用例]

入力

```
.PSECTION

unsigned short ireg_initializer[SIZEOF_prog1] = {
    prog1
};
```

出力

```
/* psection */

unsigned short ireg_initializer[SIZEOF_prog1] = {
    prog1
};
/* end psection */
```

4.10.2. マクロ定義疑似命令

マクロ定義疑似命令は、アセンブルを開始する前にソース中の文字列を指定した文字列に置き換える機能を持ちます。

マクロ定義疑似命令には、次のものがあります。

表 4.9 マクロ定義疑似命令

疑似命令	概要
.DEFINE	マクロを定義
.UNDEF	マクロ定義を解除

4.10.2.1. .DEFINE

[指定形式]

.DEFINE name def
.DEFINE name(arg[, ...]) def

[詳細説明]

- アセンブル開始前に *name* を *def* に置換するマクロを定義します。*def* は任意の文字の集合です。
- *arg* を指定すると引数付きマクロを定義することができます。
- *name* に指定した文字列が-D オプションのシンボルと重複したり、すでに本疑似命令で指定されていたりすると、エラーとなります。

[使用例]

定数マクロの定義

```
.DEFINE    SYM1        1
MOV        SMSG1, [SMSG2+SYM1]                    ; "MOV     SMSG1, [SMSG2+1]" に置換されます

.DEFINE    BITSYM      [SMSG10+0].3
SET1       BITSYM                                 ; "SET1    [SMSG10+0].3" に置換されます
```

引数付きマクロの定義

```
.DEFINE    BITSYM(BIT)          [SMSG10+0].BIT
SET1       BITSYM(3)                              ; "SET1    [SMSG10+0].3" に置換されます
```

4.10.2.2. .UNDEF

[指定形式]

.UNDEF name

[詳細説明]

- -D オプションまたは.DEFINE 疑似命令による *name* の定義を解除します。

[使用例]

マクロ定義の解除		
.DEFINE	SYM1	1
	MOV	SMSG1, [SMSG2 + SYM1]
.UNDEF	SYM1	
.DEFINE	SYM1	2

4.11. 制御命令

制御命令はアセンブラの動作に対し細かい指示を与えるもので、ソース中に記述します。
次に制御命令の種類を示します。

表 4.10 制御命令一覧

種類	制御命令
ファイル入力制御命令	$include
条件アセンブル制御命令	$ifdef , $ifndef , $if , $elseif , $else , $endif

制御命令は疑似命令と同様にソース中に記述します。

4.11.1. $include

インクルード・ファイルを展開します。

[指定形式]

$include (ファイル名)
$include "ファイル名"

[機能]

- 指定されたファイルの内容を指定された行以降に挿入展開し、アセンブルします。

[詳細説明]

- -I オプションでインクルード・ファイルのサーチ・パスを指定することができます。

  インクルード・ファイルの読み込みパスのサーチの順番は、次のとおりです。
  (a) -I オプションで指定されたフォルダ
  (b) $include 制御命令が記述されたアセンブリ・ソース・ファイルまたはインクルード・ファイルのあるフォルダ
  (c) カレント・フォルダ(smsasm を起動したフォルダ)

- インクルード・ファイルはネスティングすることが可能です(ネスティングとはインクルード・ファイル中で別のインクルード・ファイルを指定することです)。
- インクルード・ファイルのネスト・レベルの最大値は 4,294,967,294(=0xFFFFFFFE)(理論値)です。ただし実際には利用可能なメモリ量に依存します。
- インクルード・ファイルがオープンできない場合、メッセージが出力されアセンブルが中止されます。
- 1 つのインクルード・ファイル中に条件アセンブル制御命令のような開始から終了までのブロックがあるものは、その対応が取れた状態で閉じなければなりません。対応が取れてない、あるいは閉じていない場合はエラーとします。

4.11.2. $ifdef

シンボルによる制御（シンボルが定義されているときアセンブル）を行います。

[指定形式]

$ifdef シンボル名

[機能]

- 指定したシンボルが定義されている場合

  (a) 本制御命令と本制御命令に対応する elseif 制御命令、または else 制御命令が存在する場合は、本制御命令とその制御命令とで囲まれる文（ブロック）をアセンブルします。

  (b) それらの制御命令が存在しない場合は、本制御命令とその制御命令に対応する endif 制御命令とで囲まれるブロックをアセンブルします。

- 指定したシンボルが定義されていない場合

  本制御命令に対応するelseif 制御命令、else 制御命令、またはendif 制御命令までスキップします。

[詳細説明]

- 条件アセンブル制御命令のネスト・レベルの最大値は 4,294,967,294（=0xFFFFFFFE）（理論値）です。ただし、実際には利用可能なメモリ量に依存します。

4.11.3. $ifndef

シンボルによる制御（シンボルが定義されていないときアセンブル）を行います。

[指定形式]

$ifndef シンボル名

[機能]

- 指定したシンボルが定義されている場合

  本制御命令に対応するelseif 制御命令、else 制御命令、またはendif 制御命令までスキップします。

- 指定したシンボルが定義されていない場合

  (a) 本制御命令と本制御命令に対応する elseif 制御命令、または else 制御命令が存在する場合は、本制御命令とその制御命令とで囲まれるブロックをアセンブルします。

  (b) それらの制御命令が存在しない場合は、本制御命令とその制御命令に対応する endif 制御命令とで囲まれるブロックをアセンブルします。

[詳細説明]

- 条件アセンブル制御命令のネスト・レベルの最大値は 4,294,967,294（=0xFFFFFFFE）（理論値）です。ただし、実際には利用可能なメモリ量に依存します。

4.11.4. $if

式による制御（式が真のときアセンブル）を行います。

[指定形式]

$if 式

[機能]

- 指定した式が真（≠0）に評価された場合

  (a) 本制御命令と本制御命令に対応する elseif 制御命令、または else 制御命令が存在する場合は、本制御命令とその制御命令とで囲まれるブロックをアセンブルします。

  (b) それらの制御命令が存在しない場合は、本制御命令とその制御命令に対応する endif 制御命令とで囲まれるブロックをアセンブルします。

- 指定した式が偽（=0）に評価された場合

  本制御命令に対応する elseif 制御命令、else 制御命令、または endif 制御命令までスキップします。

[詳細説明]

- 条件アセンブル制御命令のネスト・レベルの最大値は 4,294,967,294（=0xFFFFFFFE）（理論値）です。ただし、実際には利用可能なメモリ量に依存します。

4.11.5. $elseif

式による制御（式が真のときアセンブル）を行います。

[指定形式]

$elseif 式

[機能]

- 指定した式が真（≠0）に評価された場合

  (a) 本制御命令と本制御命令に対応する elseif 制御命令、または else 制御命令が存在する場合は、本制御命令とその制御命令とで囲まれるブロックをアセンブルします。

  (b) それらの制御命令が存在しない場合は、本制御命令とその制御命令に対応する endif 制御命令とで囲まれるブロックをアセンブルします。

- 指定した式が偽（=0）に評価された場合

  本制御命令に対応する elseif 制御命令、else 制御命令、または endif 制御命令までスキップします。

[詳細説明]

- 条件アセンブル制御命令のネスト・レベルの最大値は 4,294,967,294（=0xFFFFFFFE）（理論値）です。ただし、実際には利用可能なメモリ量に依存します。

4.11.6. $else

シンボルまたは式による制御を行います。

[指定形式]

$else

[機能]

- ifdef 制御命令においてシンボルが定義されていない場合、if 制御命令または elseif 制御命令において、式が偽(＝0)に評価された場合、本制御命令と本制御命令に対応する endif 制御命令とで囲まれるブロックをアセンブルします。

4.11.7. $endif

制御範囲の終わりを示します。

[指定形式]

$endif

[機能]

条件アセンブル制御命令による制御の範囲の終わりを示します。

4.12. マクロ

マクロ機能の使い方について説明します。

マクロ機能とは、.DEFINE疑似命令によりマクロ・ボディとして定義された一連の命令群をマクロ参照している箇所に展開する機能です。アセンブラはマクロ参照を検出するとマクロ・ボディを展開し、マクロ・ボディの仮パラメータを参照時の実パラメータに置き換えながら命令群を機械語に変換します。

マクロはパラメータを記述することができます。例えば、処理手順は同じであるがオペランドに記述するデータだけが異なる命令群がある場合、そのデータに仮パラメータを割り当ててマクロを定義します。マクロ参照時にはマクロ名と実パラメータを記述することにより、記述の一部分だけが異なる種々の命令群に対処することができます。

4.12.1. マクロの利用

マクロとは一連の決まった手順パターンを登録し、それを利用して記述するものです。マクロはユーザが定義します。マクロを複数行で定義する場合は、行末に"¥"を付加します。

マクロの定義

```
.DEFINE  ADDMEM(reg1, reg2)¥        ; 次の4つの文がマクロ本体
   MOV    SMSG1, [reg1+0]¥
   MOV    SMSG2, [reg2+0]¥
   ADDW  SMSG1, SMSG2¥
   MOV    [reg1+0], SMSG1              ; この行までがマクロ
```

上記を定義したあと、次のように記述した場合、「SMSG3とSMSG4が指すアドレスの内容を加算する」というコードに置き換えられます。

```
ADDMEM(SMSG3, SMSG4)
```

したがって、次のようなコードに展開されます。

```
MOV    SMSG1, [SMSG3+0]
MOV    SMSG2, [SMSG4+0]
ADDW  SMSG1, SMSG2
MOV    [SMSG3+0], SMSG1
```

5. メッセージ

メッセージの出力形式と、SMSアセンブラが出力する各種メッセージについて説明します。

5.1. メッセージの形式

メッセージの種別は次のとおりです。

表 5.1 メッセージ種別

メッセージ種別	説明
C	内部エラー：処理を中止します。
E	エラー：一定数以上発生した場合、処理を中止します。
F	致命的エラー：処理を中止します。
W	ワーニング：処理を続行します。

メッセージは以下の形式で出力されます。
メッセージ番号はメッセージ種別、"058"に続けて、4桁の10進数字で表されます。

(1) ソース位置情報を出力しない場合

```
メッセージ番号:メッセージ
```

(2) ソース位置情報のうち、行番号のみを出力する場合

```
ソース・ファイル名(ソース行番号):メッセージ番号:メッセージ
```

(3) ソース位置情報のうち、行番号と列番号を出力する場合

```
ソース・ファイル名(ソース行番号:ソース列番号):メッセージ番号:メッセージ
ソース行のイメージ
ソース列位置を示すキャレット記号
```

5.2.　命令コマンド（シーケンサ・コマンド）

表 5.2 命令コマンド（シーケンサ・コマンド）エラー・メッセージ

番号	メッセージ	説明、対処法
E0580001	″SMSG%s″ cannot be specified as dst.	SMSG0、SMSG15 には値を書き込めません。SMSG0、SMSG15 以外を指定してください。
E0580002	″%s″ is out of range (%d–%d).	オペランドの値が指定可能な範囲ではありません。オペランドの値を確認してください。
E0580003	label cannot be specified as memory operand.	ラベルはメモリオペランドのオフセットに記述できません。オペランドの記述を確認してください。
E0580004	branch destination is out of program area.	分岐命令の分岐先がプログラムの外です。分岐命令の分岐距離を確認してください。
E0580005	invalid address format.	$addr には数値またはラベルのみを指定してください。
E0580006	unknown label ″%s″.	分岐命令の分岐先ラベルがみつかりません。オペランドの記述を確認してください。
E0580007	expecting ″%s″.	構文に誤りがあります。記述を確認してください。
E0580008	unexpected ″%s″.	構文に誤りがあります。記述を確認してください。
E0580009	″%s″ is already defined.	ラベルが多重に定義されています。ラベル名を確認してください。
E0580010	illegal operation (″%s″).	式の記述に誤りがあります。記述を確認してください。
E0580011	machine instructions cannot exceed 32 instructions.	プログラムの命令数が 32 個より多いです。プログラムを縮小してください。
E0580012	illegal symbol (″%s″).	シンボル名に不正な文字が使われています。シンボル名を確認してください。
C0580013	internal error. unexpected syntax error message.	特約店または弊社までご連絡ください。
C0580014	internal error. section ″%s″ is not found.	特約店または弊社までご連絡ください。
C0580015	internal error. function for ″%s″ is not found.	特約店または弊社までご連絡ください。
C0580016	internal error. additional bytes are invalid ″%s″.	特約店または弊社までご連絡ください。
E0580017	suffix and prefix cannot be specified together.	16 進数値のプレフィックスとサフィックスはどちらか片方のみを記述してください。

5.3. 疑似命令

表 5.3 疑似命令エラー・メッセージ

番号	メッセージ	説明、対処法
E0580100	specify .SECTION directive.	プログラムの先頭には.SECTION 疑似命令を記述してください。
F0580101	section name is not specified in the .SECTION directive.	.SECTION 疑似命令にはセクション名を指定してください。
F0580102	"%s" is already specified.	ファイル内に同名のセクションがあります。セクション名を確認してください。
F0580103	cannot specify a section name in the .PSECTION directive.	.PSECTION 疑似命令にはセクション名を記述しないでください。
E0580104	"%s" is already defined.	同名のマクロが複数あります。マクロ名を確認してください。
E0580105	illegal macro argument.	構文に誤りがあります。記述を確認してください。
E0580106	actual argument of macro is not matched.	マクロ参照の実パラメータの数が仮パラメータの数と一致していません。記述を確認してください。

5.4. 制御命令

表 5.4 制御命令エラー・メッセージ

番号	メッセージ	説明、対処法
E0580200	illegal syntax.	構文に誤りがあります。記述を確認してください。
E0580201	include file cannot nest over 4294967294 times.	$include 制御命令の入れ子の回数が上限を超えました。
E0580202	condition assembly directive cannot nest over 4294967294 times.	$ifdef 制御命令の入れ子の回数が上限を超えました。
E0580203	unexpected directive ″%s″.	″%s″に対応する$if,$ifdef 制御命令がありません。
E0580204	expecting directive ″%s″.	″%s″に対応する$endif 制御命令がありません。

5.5. コマンド・ライン

表 5.5 コマンド・ライン エラー・メッセージ

番号	メッセージ	説明、対処法
E0580300	"%s" is unrecognized.	不正なオプション名です。オプション指定を確認してください。
E0580301	specify the "-o" option.	-o オプションを指定してください。
E0580302	"%s" option requires an argument.	オプションにパラメータを指定してください。
E0580303	invalid argument for the "%s" option.	オプションのパラメータ指定が不正です。オプション指定を確認してください。
W0580304	"%s" option is specified more than once. The latter is valid.	同じオプションが複数回指定されています。最後の指定が有効になります。
E0580305	"%s" specified in the "%s" option is invalid name.	-D または-U オプションで指定したシンボル名が不正です。-D または-U オプションの指定を確認してください。
W0580306	the folder "%s" specified by the "-I" option is not found.	-I オプションで指定されたフォルダが見つかりません。-I オプションの指定を確認してください。
E0580307	cannot open subcommand file "%s".	サブコマンド・ファイルを開くことができません。サブコマンド・ファイルの状態を確認してください。
E0580308	cannot read subcommand file "%s".	サブコマンド・ファイルを読むことができません。サブコマンド・ファイルの状態を確認してください。
E0580309	subcommand file "%s" is read more than once.	同名のサブコマンド・ファイルが複数回指定されています。サブコマンド・ファイル指定を確認してください。
E0580310	missing double quotation.	2 重引用符が欠けています。記述を確認してください。
E0580311	specify the input file.	入力ファイルを 1 つ指定してください。
E0580312	too many input files.	入力ファイル指定は 1 つのみにしてください。
E0580313	the file "%s" specified by the "%s" option is a folder.	指定された出力先ファイルはフォルダです。出力ファイルオプションの指定を確認してください。
E0580314	the output folder "%s" specified by the "%s" option is not found.	出力先フォルダがみつかりません。出力ファイルオプションの指定を確認してください。
E0580315	cannot open input file "%s".	入力ファイルを開くことができません。入力ファイルの状態を確認してください。
E0580316	cannot read input file "%s".	入力ファイルを読むことができません。入力ファイルの状態を確認してください。
E0580317	cannot open output file "%s".	出力先ファイルを開くことができません。出力先ファイルの状態を確認してください。
E0580318	cannot write output file "%s".	出力先ファイルへの書き込みに失敗しました。出力先ファイルの状態を確認してください。
W0580319	-MM and -MMD option are both specified. The latter is valid.	-MM オプションと-MMD オプションの両方が指定されています。最後の指定が有効になります。
F0580399	too many errors.	エラーの数が多すぎるため、処理を中断しました。

5.6. その他

表 5.6 その他のエラー・メッセージ

番号	メッセージ	説明、対処法
C0580900	internal error. undefined message for "%s".	特約店または弊社までご連絡ください。
C0580901	internal error. failed to set signal handler.	特約店または弊社までご連絡ください。
C0580902	internal error. invalid memory access.	特約店または弊社までご連絡ください。
C0580903	internal error. invalid instruction execution.	特約店または弊社までご連絡ください。
C0580904	internal error. abnormal termination.	特約店または弊社までご連絡ください。
C0580905	internal error. illegal operation.	特約店または弊社までご連絡ください。
C0580906	internal error. unexpected signal received.	特約店または弊社までご連絡ください。
C0580907	internal error. unknown message type "%s".	特約店または弊社までご連絡ください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2020.03.19	ー	初版発行
1.02	2025.12.01	3	1.1.3. サポート環境 サポート OS から Windows8.1 を削除し、Windows11 を追加
		4	1.1.4. 入出力 出力ファイルに依存関係ファイルを追加
		7	3.2.1. オプション一覧 依存関係ファイル出力オプション -MM / -MMD / -MP / -MT / -MF を追加
		8,9	3.2.2.1. -V 3.2.2.2. -help 詳細説明に依存関係ファイルの出力は行わない旨を追記
		10	3.2.2.3. -o -MM オプション指定時以外は省略不可、に変更
		14	3.2.2.7. -character_set 詳細説明に複数回指定時は最後の指定が有効になる旨を追記
		18,19	3.2.2.11.-MM / -MMD / -MP / -MT / -MF 依存関係ファイル出力オプションの説明を追加
		57	5.5. コマンド・ライン 以下のワーニングメッセージを追加 W0580319：-MM and -MMD option are both specified. The latter is valid.

SMSアセンブラ ユーザーズマニュアル

発行年月日　2025年12月1日　Rev.1.02

発行　ルネサス エレクトロニクス株式会社
〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

SMS アセンブラ

ルネサス エレクトロニクス株式会社

R20UT4792JJ0102