

お客様各位

---

## カタログ等資料中の旧社名の扱いについて

---

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願ひ申し上げます。

ルネサスエレクトロニクス ホームページ (<http://www.renesas.com>)

2010年4月1日

ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社 (<http://www.renesas.com>)

【問い合わせ先】 <http://japan.renesas.com/inquiry>

## ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りが無いことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。  
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット  
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）  
特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社とその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。



お客様各位

---

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

---

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

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

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

2003年4月1日  
株式会社ルネサス テクノロジ  
カスタマサポート部

# AS79 V.4.10

ユーザーズマニュアル

7900 シリーズ用リロケータブルアセンブラ

#### 安全設計に関するお願い

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

#### 本資料ご利用に際しての留意事項

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

#### 製品内容及び本書についてのお問い合わせ先

インストーラが生成する以下のテキストファイルに必要事項を記入の上、ツール技術サポート窓口 [support\\_tool@renesas.com](mailto:support_tool@renesas.com) まで送信ください。

¥SUPPORT¥製品名¥SUPPORT.TXT

株式会社ルネサス ソリューションズ マイコンツール部

ツール技術サポート窓口	<a href="mailto:support_tool@renesas.com">support_tool@renesas.com</a>
ユーザ登録窓口	<a href="mailto:regist_tool@renesas.com">regist_tool@renesas.com</a>
ホームページ	<a href="http://www.renesas.com/jp/tools">http://www.renesas.com/jp/tools</a>

# AS79 ユーザーズマニュアル目次

第1章 AS79 の仕様 .....	10
1.1 AS79制限値一覧 .....	10
1.2 仕様の詳細 .....	11
第2章 AS79 の概要 .....	13
2.1 AS79の構成 .....	13
2.2 as79の構成 .....	14
2.3 as79の機能概要 .....	15
2.4 ln79の機能概要 .....	16
2.5 lmc79の機能概要 .....	16
2.6 lb79の機能概要 .....	17
2.7 xrf79の機能概要 .....	18
2.8 abs79の機能概要 .....	18
第3章 AS79 の特徴 .....	19
3.1 リロケータブルプログラミング .....	19
3.2 シンボリックプログラミング .....	25
3.3 データのサイズ指定 .....	28
3.4 ファイルのインクルード .....	30
3.5 ライブラリファイル .....	31
3.6 ソースレベルデバッグ対応 .....	32
3.7 IEEE-695フォーマット準拠 .....	32
3.8 条件アセンブル機能 .....	32
3.9 マクロ命令の定義と参照 .....	33
3.10 繰り返しマクロ機能 .....	34
3.11 プリデファインドマクロ機能 .....	34
3.12 行の連結機能 .....	35
3.13 環境変数の参照 .....	36
3.14 OSへの戻り値 .....	39

# 目次

第4章 AS79 操作方法 .....	40
4.1 コマンド入力時の注意事項 .....	40
4.2 コマンド行の構成 .....	41
第5章 as79 の使用方法 .....	42
5.1 as79コマンドオプション .....	42
5.2 as79コマンドパラメータの指定規則 .....	43
5.3 as79コマンドオプション .....	44
- .....	45
-A .....	46
-B .....	47
-C .....	48
-D .....	49
-finfo .....	50
-F .....	51
-G1 .....	52
-G2 .....	53
-H .....	54
-I .....	55
-JF .....	56
-JN .....	57
-L .....	58
-M6 .....	59
-M8 .....	60
-N .....	61
-O .....	62
-P .....	63
-Q .....	64
-S[M] .....	65
-T .....	66
-V .....	67
-W .....	68
-X .....	69
5.4 as79エラーメッセージ .....	70
5.5 as79ワーニングメッセージ .....	81

## 目次

第6章 ln79 の操作方法 .....	83
6.1 ln79コマンドパラメータ .....	83
6.2 ln79コマンドパラメータの指定規則 .....	84
6.3 コマンドファイル .....	85
6.4 ln79コマンドオプション .....	85
- .....	86
-ABLX .....	87
-C .....	88
-E .....	89
-G .....	90
-L .....	91
-LD .....	92
-M .....	93
-MS/-MSL .....	94
-NOSTOP .....	95
-O .....	96
-ORDER .....	97
-T .....	98
-V .....	99
@ .....	100
6.5 ln79エラーメッセージ .....	101
6.6 ln79ワーニングメッセージ .....	104
第7章 lmc79 の操作方法 .....	107
7.1 lmc79コマンドパラメータ .....	107
7.2 lmc79コマンドパラメータの指定規則 .....	107
7.3 lmc79コマンドオプション .....	108
- .....	109
-A .....	110
-E .....	111
-F .....	112
-H .....	113
-L .....	114
-O .....	115
-V .....	116
7.4 lmc79エラーメッセージ .....	117
7.5 lmc79ワーニングメッセージ .....	118



## 目次

第8章 lb79 の操作方法 .....	119
8.1 lb79コマンドパラメータ .....	119
8.2 lb79コマンドパラメータの指定規則 .....	120
8.3 lb79コマンドファイル.....	120
8.4 lb79コマンドオプション .....	120
-.....	121
-A .....	122
-C .....	123
-D .....	124
-L.....	125
-R .....	126
-U .....	127
-V .....	128
-X .....	129
@ .....	130
8.4 lb79エラーメッセージ.....	131
7.5 lb79ワーニングメッセージ .....	133
第9章 xrf79 の操作方法 .....	134
9.1 xrf79コマンドパラメータ .....	134
9.2 xrf79コマンドパラメータの指定規則 .....	134
9.3 xrf79コマンドファイル .....	135
9.4 xrf79コマンドオプション .....	135
-.....	136
-N .....	137
-O .....	138
-V .....	139
@ .....	140
9.5 xrf79エラーメッセージ .....	141
第10章 abs79 の操作方法 .....	142
10.1 abs79使用上の注意事項.....	142
10.2 abs79コマンドパラメータ .....	142
10.3 abs79コマンドパラメータの指定規則 .....	143
10.4 abs79コマンドオプション .....	143
-.....	144
-D .....	145
-O .....	146
-V .....	147
10.5 abs79エラーメッセージ.....	148
10.6 abs79ワーニングメッセージ .....	149

## 目次

第 11 章 入出力ファイル .....	150
11.1 入出力ファイル一覧 .....	150
11.2 ソースファイル .....	151
11.3 リロケータブルモジュールファイル .....	151
11.4 アセンブラリストファイル .....	152
11.5 アセンブラエラータグファイル .....	160
11.6 アブソリュートモジュールファイル .....	161
11.7 マップファイル .....	162
11.7 リンクエラータグファイル .....	164
11.8 モトローラ S フォーマットファイル .....	165
11.9 インテル HEX フォーマットファイル .....	165
11.10 ライブラリファイル .....	166
11.11 ライブラリリストファイル .....	166
11.12 クロスリファレンスファイル .....	168
11.13 アブソリュートリストファイル .....	169
第 12 章 プログラムの記述規則 .....	170
12.1 プログラム記述上の注意事項 .....	170
12.2 文字セット .....	171
12.3 予約語 .....	171
12.4 名前 .....	172
12.5 行の記述規則 .....	177
12.6 オペランド .....	183
12.7 演算子 .....	186
第 13 章 7900 のアドレッシングモード .....	190
13.1 ダイレクトアドレッシングモード .....	190
13.2 アブソリュートアドレッシングモード .....	191
13.3 アブソリュートロングアドレッシングモード .....	191
第 14 章 アドレッシングモードの指定と最適化 .....	192
14.1 アドレッシングモードの指定方法 .....	192
14.2 ダイレクトアドレッシングモードを指定する .....	194
14.3 アブソリュートアドレッシングモードを指定する .....	197
14.4 アブソリュートロングアドレッシングモードを指定する .....	199
14.5 アドレッシングモード指定子を記述可能なアドレッシングモード ..	200
14.6 アドレッシングモードの指定例とアセンブル結果 .....	201
14.7 アセンブラによるアドレッシングモードの自動選択 .....	203
14.8 特定のアドレッシングモードの選択を禁止する .....	205

## 目次

第 15 章 指示命令 .....	207
15.1 指示命令一覽 .....	208
15.2 指示命令記述方法 .....	213
..FILE .....	214
..MACPARA .....	215
..MACREP ..MACREPZ .....	216
.ADDR .....	217
.ALIGN .....	218
.ASSERT .....	220
.BLKA .....	221
.BLKB .....	222
.BLKD .....	223
.BLKDF .....	224
.BLKF .....	225
.BLKW .....	226
.BTEQU .....	227
.BYTE .....	228
.CALL .....	229
.DATA .....	230
.DEFINE .....	231
.DOUBLE .....	232
.DP[0!1!2!3] .....	233
.DP[0!1!2!3]SYM .....	234
.DT .....	235
.DTSYM .....	236
.DWORD .....	237
.EINSF .....	238
.ELIF .....	239
.ELSE .....	240
.END .....	241
.ENDIF .....	242
.ENDM .....	243
.ENDR .....	244
.EQU .....	245
.EXITM .....	246
.FLOAT .....	247
.FORM .....	248
.GLB .....	249
.IF .....	250
.INCLUDE .....	252
.INDEX .....	253
.INSF .....	254
.INSTR .....	255
.LEN .....	257
.LENGTH .....	259
.LGSYM .....	260
.LIST .....	261
.LOCAL .....	262

## 目次

.MACRO .....	263
.MREPEAT .....	266
.ORG .....	267
.PAGE .....	269
.SECTION .....	270
.STK .....	271
.SUBSTR .....	272
.VER .....	274
.WORD .....	275
? .....	276
@ .....	277
第 16 章 構造化記述 .....	278
16.1 AS79の構造化記述 .....	278
16.2 構造化記述の演算子 .....	280
16.3 構造化記述行の記述規則 .....	281
16.4 分岐距離の指定 .....	284
16.5 サイズ指定による式のサイズ決定 .....	286
16.6 構造化記述命令 .....	290
IF .....	291
FOR-STEP .....	292
FOR-NEXT .....	293
DO .....	294
SWITCH .....	295
BREAK .....	296
CONTINUE .....	298
FOREVER .....	299
GOTO .....	300
CALL .....	301
RETURN .....	302
代入文 .....	303
16.7 構造化記述内でのみ有効となる指示命令 .....	304
.FAR .....	305
.GLBB / .GLBW / .GLBD .....	306
.NEAR .....	307
16.7 構造化記述命令構文図 .....	308
第 17 章 RASM77 用アセンブリプログラムとの互換性について .....	320
17.1 算術演算子 .....	320
17.2 アドレッシング指定子とEQUシンボル（オフセット計算） .....	321
17.3 ロケーションシンボル .....	321
17.4 アセンブル指示命令 .....	322
17.5 マクロ命令に関する相違点 .....	324
第 18 章 7900 命令一覧 .....	327

# 第1章 AS79の仕様

---

AS79は以下に示す仕様に基づいて設計しています。この仕様の範囲内でご使用願います。

## 1.1 AS79制限値一覧

項目	仕様
同時にオープンするファイル数	9
ファイル名文字数	パソコン上では128バイト（文字） ワークステーション上では512バイト（文字）
コマンドライン文字数	パソコン上では128バイト（文字） ワークステーション上では512バイト（文字）
環境変数に設定できる文字数	256バイト（文字）
名前の文字数	32バイト（文字）
名前の総数	処理を行うホストマシンのメモリ容量に依存
マクロ定義の数	0 ~ 65535

## 1.2 仕様の詳細

### 1.2.1 文字セット

AS79に含まれる各プログラムの起動時に、コマンド行に使用可能な文字及びソースプログラム内で記述可能な文字セットを次に示します。

#### 英大文字

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

#### 英小文字

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

#### 数字

0 1 2 3 4 5 6 7 8 9

#### 特殊文字

" # \$ % & ' ( ) \* + , - . / : ; [ ] \ ^ \_ | ~

#### 注意事項

使用するホストによって'¥'は、'¥'となります。

#### 空白文字

(スペース) (タブ)

#### 改行文字

(リターン) (ラインフィード)

#### 注意事項

漢字などの多バイト文字は使用できません。

### 1.2.2 コマンドラインの文字数

- ・ コマンド行に指定できる文字数は、パソコン版では128文字 (バイト)、ワークステーション版では、512文字 (バイト) 以下です。

#### 注意事項

AS79の使用環境 (OSの種類) によっては、上記以下の文字数に制限される場合もあります。

### 1.2.3 ファイル名の入力規則

- ・ ファイル名の長さは、パソコン版ではディレクトリ指定を含めて128文字（バイト）、ワークステーション版ではディレクトリ指定を含めて512文字（バイト）以下です。ただし、起動するプログラム名やコマンドオプションを含め、コマンド行の文字数が上記の範囲内で指定してください。
- ・ ファイル名の記述規則は、上記の他にパソコン及びワークステーションのOSによって規定されます。詳しくは、それぞれのOSの説明書を参照してください。

#### 注意事項

---

ワークステーション上では、ファイル名に二箇所以上のピリオド(.)で区切ったものを使用できますが、AS79ではファイル名に含めることのできるピリオド(.)は、一箇所だけです。また、ファイルの拡張子（ピリオド以降の文字）についても、規定されているものがありますので、AS79のそれぞれのプログラムの起動方法を確認してください。

---

## 第2章 AS79の概要

---

### 2.1 AS79の構成

AS79は、次に示すプログラムで構成されています。

#### 2.1.1 アセンブラドライバ(as79)

マクロプロセッサ、構造化プリプロセッサ及びアセンブラプロセッサを連続して起動するプログラムです。アセンブラドライバは、複数のソースファイル进行处理できます。

##### マクロプロセッサ

アセンブリソース中のマクロ指示命令及びアセンブラプロセッサのための前処理を行い、ソースファイルを生成します。

##### 注意事項

---

マクロプロセッサが生成したソースファイルは、アセンブラプロセッサの処理終了後に消去されます。ユーザが記述したソースファイルは変更されません。

---

##### 構造化プリプロセッサ

アセンブリソース中の構造化記述命令及びアセンブラプロセッサのための前処理を行い、ソースファイルを生成します。

##### 注意事項

---

構造化プリプロセッサが生成したソースファイルは、アセンブラプロセッサの処理終了後に消去されます。ユーザが記述したソースファイルは変更されません。

---

##### アセンブラプロセッサ

マクロプロセッサ及び構造化プリプロセッサが前処理を行ったソースファイルをリロケータブルモジュールファイルに変換します。



### 2.1.2 リンケージエディタ(ln79)

アセンブラプロセッサの生成したリロケータブルモジュールファイルをリンクし、アブソリュートモジュールファイルを生成します。

### 2.1.3 ライブラリアン(lb79)

リロケータブルモジュールファイルを読み込み、ライブラリファイルを生成、管理します。

### 2.1.4 ロードモジュールコンバータ(lmc79)

リンケージエディタの生成したアブソリュートモジュールファイルをROM化可能な機械語ファイル（モトローラSファイル、インテルHEXファイル）に変換します。

### 2.1.5 クロスリファレンサ(xrf79)

ユーザーの作成したソースファイル中の各種シンボル及びラベルの定義情報を格納したクロスリファレンスファイルを生成します。

### 2.1.6 アブソリュートリスタ(abs79)

アブソリュートモジュールファイルのアドレス情報を基に、プリントアウト可能なアブソリュートリストファイルを生成します。

## 2.2 as79の構成

as79は、マクロ記述を処理するマクロプロセッサ、構造化記述命令を処理する構造化プリプロセッサ及びソースファイルをオブジェクトファイルに変換するアセンブラプロセッサで構成されます。as79は、これら3つのプログラムを制御するプログラムの名称です。以降、「as79」と表記した場合は、マクロプロセッサ、構造化プリプロセッサ、アセンブラプロセッサ及びas79の全てを示します。

### 注意事項

---

AS79では、アセンブラドライバがマクロプロセッサ、構造化プリプロセッサ及びアセンブラプロセッサをコントロールします。したがって、マクロプロセッサ、構造化プリプロセッサ及びアセンブラプロセッサを直接起動しないでください。これらのプログラムを直接起動した結果については保証しません。

---

## 2.3 as79の機能概要

### 2.3.1 リロケータブルモジュールファイルの生成

- ・ ソースファイルをリンケージエディタで読み込み可能なリロケータブルモジュールファイルに変換します。
- ・ ソースファイルから、複数ファイルをリンクするために必要なリロケータブル情報を、リロケータブルオブジェクトファイルに出力します。
- ・ 「7900シリーズ用 Cコンパイラ NC79」、as79の「マクロ記述」及び「構造化記述命令」のソースデバッグを実現するために必要な情報を、リロケータブルモジュールファイルに出力します。

### 2.3.2 アセンブラリストファイルの生成

- ・ プリントまたは画面に出力して、デバッグ時に参照可能なリストファイルを生成します。
- ・ リストファイルには、ソース行の他にリロケータブルアドレス及びリロケータブルなオブジェクトコードを出力します。

### 2.3.3 アセンブラエラータグファイルの生成

- ・ アセンブルエラーの発生した行番号と行の記述内容を含むアセンブラエラータグファイルを生成します。

### 2.3.4 シンボル定義

- ・ as79は、プログラムの起動時にコマンドオプション(-D)によって、シンボルの定義ができます。
- ・ シンボル定義の機能は、条件アセンブル機能などと組み合わせて使用できます。

## 2.4 In79の機能概要

### 2.4.1 アブソリュートモジュールファイルの生成

- ・ 1つ以上のリロケータブルモジュールファイルから、1つのアブソリュートモジュールファイルを生成します。リロケータブル値をアブソリュート値に変換します。
- ・ リロケータブルモジュールファイルをセクション単位で実アドレスに配置し、7900シリーズ用デバッガでデバッグ可能な、アブソリュートモジュールファイルに変換します。

### 2.4.2 ライブラリファイルの参照

- ・ ライブラリファイルに登録されている任意のリロケータブルモジュールファイルを利用できます。

## 2.5 lmc79の機能概要

### 2.5.1 モトローラSフォーマットファイルの生成

- ・ 7900シリーズチップの内蔵ROMやEPROMなどに書き込み可能なモトローラSフォーマットファイルを生成します。

### 2.5.2 インテルHEXフォーマットファイルの生成

- ・ コマンドオプション(-H)を指定した場合のみ、7900シリーズチップの内蔵ROMやEPROMなどに書き込み可能なインテルHEXフォーマットファイルを生成します。

## 2.6 lb79の機能概要

### 2.6.1 ライブラリファイルの新規作成

- ・ 指定した名前のライブラリファイルがない場合に、ライブラリファイルを新規に作成します。
- ・ ライブラリファイルへのリロケータブルモジュールの登録は、コマンド行で指定された順に登録します。

### 2.6.2 リロケータブルモジュールの追加

- ・ ライブラリファイルにリロケータブルモジュールを追加します。
- ・ ライブラリファイルの内容が更新されます。
- ・ ライブラリファイルに登録されているリロケータブルモジュールを、そのリロケータブルモジュールファイルが生成された日時を基準に管理します。

### 2.6.3 リロケータブルモジュールの更新

- ・ ライブラリファイルに登録されているリロケータブルモジュールの内容を更新します。
- ・ ライブラリファイルの内容が更新されます。
- ・ ライブラリファイルに登録されているリロケータブルモジュールを、そのリロケータブルモジュールファイルが生成された日時を基準に管理します。

### 2.6.4 リロケータブルモジュールの削除

- ・ ライブラリファイル内の、不要になったリロケータブルモジュールを削除します。
- ・ ライブラリファイルの内容が更新されます。

### 2.6.5 リロケータブルモジュールの抽出

- ・ ライブラリファイルから、リロケータブルモジュールファイルを抽出します。
- ・ ライブラリファイルの内容は更新されません。
- ・ 抽出されたりロケータブルモジュールファイルは、as79が出力したりロケータブルモジュールファイルと、同一の内容になります。
- ・ 抽出されたりロケータブルモジュールファイルのファイル情報は、抽出された日時の情報を持ちますが、リロケータブルファイル内のファイル生成情報は、as79がそのリロケータブルモジュールファイルを生成した日時となります。

### 2.6.6 ライブラリリストファイルの生成

- ・ ライブラリファイルに登録されているリロケータブルモジュールの情報を、ライブラリリストファイルに出力します。
- ・ ライブラリファイルの内容は更新されません。
- ・ ライブラリリストファイルに出力するリロケータブルモジュールの作成日時は、すべてリロケータブルモジュールファイルの生成日時となります。

## 2.7 xrf79の機能概要

### 2.7.1 クロスリファレンスファイル生成

- ・ ソースファイル又はアセンブラリストファイルを参照し、ラベル及びシンボルの定義を一覧できるクロスリファレンスファイルを生成します。
- ・ システムラベル情報の出力を制御できます。

## 2.8 abs79の機能概要

### 2.8.1 アブソリュートリストファイルの生成

- ・ アブソリュートモジュールファイルの、実アドレス情報及びコード情報をもつ、アブソリュートリストファイルを生成します。
- ・ abs79は、アブソリュートモジュールファイルに含まれる、リロケータブルモジュール毎に、アブソリュートリストファイルを生成します。
- ・ abs79は、アブソリュートモジュールファイルとアセンブラリストファイルを参照してファイルを生成します。

## 第3章 AS79の特徴

---

AS79は、7900シリーズに対応したアセンブリ言語で記述されたソースプログラムを機械語ファイルに変換するほかに、次のような特徴的な機能を持っています。

- リロケータブルプログラミング
- シンボリックプログラミング
- データのサイズ指定
- ファイルのインクルード
- ライブラリファイル
- ソースレベルデバッグ対応
- IEEE-695フォーマット準拠
- 条件アセンブル
- マクロの定義と参照
- 行の連結機能
- 環境変数の参照
- OSへの戻り値

### 3.1 リロケータブルプログラミング

1つまたは複数のファイルに記述したソースプログラムを、セクション毎に任意のアドレスに配置することができます。

これにより、マイクロコンピュータのメモリ空間を意識することなくプログラムを記述できます。また、複数人でのプログラム開発などに適しています。

実際には、マイクロコンピュータのRAM、ROMサイズを超えないように予め各サイズを確認しておいてください。

### 3.1.1 セクション

リロケータブルプログラミングでは、セクションを最小の単位としてソースプログラムをアドレスに配置します。

1つのソースファイルには一つ以上のセクションが必要です。例外として、シンボル定義のみを記述したファイルはセクションの定義がないことがあります。

#### セクションの定義

指示命令.SECTIONで、新しいセクションの開始とセクション名を定義します。

- ・ 一つのファイルに複数のセクションを定義できます。
- ・ 同じ名前のセクションを複数個定義することもできます。

#### 注意事項

---

セクションをネストして定義することはできません。

---

#### セクションの開始と終了

セクションを定義した行から始まり、ソースファイルの終わり（指示命令.ENDを記述した行）または、別のセクションが定義されるまでを1セクションとします。

```

        .SECTIONram,DATA      ; start of ram section
work:   .BLKB    10          ; end of ram section
        .SECTIONprogram,CODE ; start of program section
        JSR     sub1        ; end of program section
        .SECTIONsubroutine  ; start of subroutine section
sub1:   NOP
        MOVM.W  work
        RTS                ; end of subroutine section
        .END

```

## AS79の特徴

### セクションのタイプ

セクションには3つのタイプがあります。セクションを定義すると同時にタイプを指定できます。

- ・ セクションのタイプは、セクションを配置する際にROM領域とRAM領域が混在しないように予め指定しておきます。
- ・ アセンブラは、ソースプログラムの中にセクションのタイプと矛盾する指示命令の記述があるとエラーメッセージを出力します。

### CODE (プログラム領域)

主にソースプログラム (機械語) を格納する領域で、マイクロコンピュータのROMに配置します。

```
.SECTION main-p, CODE
```

### DATA (可変データ領域)

プログラム動作中にメモリの内容が書き変わる領域で、マイクロコンピュータのRAMに配置します。

```
.SECTION work-area, DATA
```

### ROMDATA (固定データ領域)

ソースプログラム以外の固定のデータを格納する領域で、マイクロコンピュータのROMに配置します。

```
.SECTION disp-data, ROMDATA
```

### 注意事項

---

セクション定義の際にタイプを指定しない場合はCODEタイプとして処理されます。

---



### 3.1.2 ソース記述時にセクションの開始アドレスを決める

ソースプログラムの記述時にセクション内のアドレスを決めることができます。スペシャルファンクションレジスタ領域などはソース記述時にアドレスを決めておくことがあります。

#### 絶対属性セクション

- ・ ソース記述時にセクションの開始アドレスを指定したセクションを絶対属性セクションといいます。
- ・ セクションを定義した直後の行で指示命令.ORGを使用してセクションの開始アドレスを宣言します。

```
.SECTION sfr-area, RAM
.ORG    00H
:
```

#### 相対属性セクション

- ・ 開始アドレスが指定されていないセクションは相対属性セクションといいます。
- ・ アセンブラは相対属性セクションの開始アドレスを0番地からと仮定してコードを生成します。

### 3.1.3 セクションの開始アドレスを偶数番地にそろえる

ワードデータまたはダブルワードデータをメモリに配置する場合、データを偶数番地から配置することでメモリへのアクセス効率が向上します。

- ・ セクション定義行と指示命令.ALIGNを使って指定します。

```
.SECTION program, CODE, ALIGN=2
.ALIGN    2
```

#### 注意事項

相対属性セクションにたいしてのみ指定できます。

### 3.1.4 リンク時にセクションの開始アドレスを決める

相対属性セクションはリンク時にセクション内のアドレスを決めます。セクションの開始アドレスを指定する場合はリンクコマンドln79のコマンドオプション(-order)で指定します。

```
>ln79 sample -order program=0e000
```

### 3.1.5 セクションの配置規則

#### 同一名セクションの配置

同じ名前のセクションは連続して配置されます。

- ・ 同じ名前タイプが異なるセクションに対してリンクはワーニングを出力します。
- ・ 同じ名前属性が異なるセクションに対してリンクはワーニングを出力します。
- ・ 同じ名前のセクションで、相対属性セクションの後に絶対属性セクションを配置しようとした場合は、エラーとなります。

#### 複数ファイルのセクション配置

リンクコマンドで指定されたファイルの順にセクションを検索し、最初に見つかったセクションから順に配置します。

- ・ 最初に見つかったセクションの開始アドレスが指定されていない場合は0番地から配置されます。

#### 異なるセクション名の配置

相対属性セクションは、リンクコマンドで指定しない限り、前のセクションと連続するアドレスに配置されます。

#### セクションのオーバーラップ

セクションの配置結果がオーバーラップする場合は次のように処理されます。

- ・ セクションタイプが"DATA"で、アドレスがオーバーラップする場合は、ワーニングが出力されます。セクションはオーバーラップして配置されます。
- ・ セクションタイプが"CODE"又は"ROMDATA"で、アドレスがオーバーラップする場合は、エラーとなります。
- ・ 絶対属性セクションの後に、絶対属性セクションを配置しようとした場合は、ワーニングが出力されます。

#### 注意事項

**ワーニングが出力された場合も機械語ファイルは生成されます。**

#### セクションの配置結果とアドレッシングモード

ln79でセクションを配置した結果、アセンブラが生成したアドレッシングモードでアクセス可能な範囲外にアドレスが決定された場合はワーニングが出力されます。

## 3.1.6 セクションの配置例

3つのファイルをリンクしたときの各セクションの配置結果を示します。

file1.a79のセクション定義

```
.SECTION    program
:
.SECTION    subroutine
.ORG       10000H
:
.END
```

file2.a79のセクション定義

```
.SECTION    subroutine
:
.SECTION    interrupt
:
.END
```

file3.a79のセクション定義

```
.SECTION    interrupt
:
.SECTION    program
:
.END
```

リンクコマンドでセクションの開始アドレスを指定しなかった場合の配置結果

ln79コマンド入力

```
>ln79 file1 file2 file3
```

セクション配置結果

program	REL CODE	000000	000003	f1
	REL CODE	000003	000003	f3
subroutine	ABS CODE	001000	000003	f1
	REL CODE	001003	000002	f2
interrupt	REL CODE	001005	000002	f2
	REL CODE	001007	000003	f3

リンクコマンドでセクションの開始アドレスを指定した場合の配置結果

ln79コマンド入力

```
>ln79 file1 file2 file3 -order interrupt=0f000
```

セクション配置結果

interrupt	REL CODE	00F000	000002	file2
	REL CODE	00F002	000003	file3
program	REL CODE	00F005	000003	file1
	REL CODE	00F008	000003	file3
subroutine	ABS CODE	001000	000003	file1
	REL CODE	001003	000002	file2

## 3.2 シンボリックプログラミング

数値に名前をつけてプログラムの判読性を向上させることができます。数値の種類によって名前を「ラベル」または「シンボル」と区別しています。

### 3.2.1 ラベル

マイクロコンピュータのメモリの番地（アドレス）に付けられた名前をラベルといいます。

#### ラベルの定義

ソース行の先頭に名前を記述するか、メモリのエリアを確保する指示命令またはデータを格納する指示命令でラベル名を定義できます。

#### ラベルを定義する指示命令

<code>.BLKB</code>	<code>.BLKW</code>	<code>.BLKA</code>	<code>.BLKD</code>	<code>.BLKF</code>
<code>.BYTE</code>	<code>.WORD</code>	<code>.ADDR</code>	<code>.DWORD</code>	<code>.FLOAT</code>
<code>.DOUBLE</code>				

#### ラベルの参照

オペランドにラベルを記述します。

#### ラベルの外部参照

別のファイル内で定義されているラベルを参照（外部参照）することができます。外部参照したラベルをグローバルラベルといいます。

- ラベルが示す番地はファイルをリンクした時に決まります。したがって、アセンブラは番地を仮の値（リロケータブル値）でコードを生成します。
- ラベルを外部参照できるようにするには、指示命令`.GLB`でグローバル宣言しておきます。グローバル宣言は、ラベルを定義したファイルとラベルを参照するファイルの両方に記述してください。

### 3.2.2 シンボル

定数値に付けられた名前をシンボルといいます。ただし、オペランドへの記述のしかたによっては、シンボル値がアドレスとして扱われることがあります。

#### シンボルの定義

指示命令.EQUで定義します。シンボルに定義する数値はアセンブル実行時に確定する値である必要があります。詳細は.EQUの説明を参照してください。

#### シンボルの再定義

1つのファイルの中で同じシンボル名に違う値を定義できます。

#### シンボルの参照

オペランドにシンボルを記述します。シンボルの前方参照（シンボル定義された行より前の行でシンボルを参照）はできません。

#### シンボルの外部参照

シンボルを外部参照すると、値はアドレス値として扱われます。

- ・ 定数を記述すべきオペランドに外部参照シンボルを記述するとアセンブラはエラーを出力します。
- ・ イミディエイトアドレッシングで記述された命令のオペランドに外部参照シンボルを記述した場合は、シンボル値を定数として扱います。

### 3.2.3 ビットシンボル

ビットシンボルはあるメモリのビット位置を示す名前です。

#### ビットシンボルの定義

指示命令.BTEQUで定義します。ビット位置を示す数値はアセンブル実行時に確定する必要があります。

- ・ メモリの番地はラベルで指定することができ、グローバルラベルも指定できます。

#### ビットシンボルの参照

ビット操作命令のオペランドにビットシンボルを記述します。

#### 注意事項

---

**ビットシンボルの外部参照はできません。**

---

### 3.2.4 ロケーションシンボル

ロケーションシンボル(\$)が記述されている行の命令が配置される番地を指します。ロケーションシンボルを使用すれば、ラベルを定義しなくてもオペランドに番地を指定できます。

- ・ 命令のオペランドに記述してください。
- ・ ロケーションシンボルを使った式を記述できます。
- ・ ロケーションシンボルを分岐命令のオペランドの式に記述する場合は、分岐先アドレスまでの全ての命令に対して最適化が行われないように記述してください。このような記述を行った場合、アセンブル実行時にワーニングを出力します。

記述例 (リストファイル)

1		.DT	00H	
2		.SECTION		program
3	008000	.ORG	8000H	
4	008000	112C008000	ADD	A,\$
5	008005	1C058000	LDA	A,\$

### 3.2.5 グローバルとローカル

#### グローバル

指示命令.GLBで宣言されているシンボルおよびラベルをグローバルといいます。

#### ローカル

グローバル宣言していないラベルおよびシンボルをローカルといいます。

- ・ ローカルな名前は、異なるファイルであれば同一の名前で異なる値のラベルなどを使用できます。

## 3.3 データのサイズ指定

命令のオペランドに記述するデータのサイズを指定することで、プログラムサイズを節約したり、大きな値の計算結果を得ることができます。

- ・ サイズ指定しない場合は、すべてのデータはワード（16ビット）長で扱われます。

指定できるデータのサイズは、次の三つです。

- ・ バイト（8ビット長）
- ・ ワード（16ビット長）
- ・ ダブルワード（32ビット長）

データサイズを指定するには3つの記述方法があります。アセンブラは命令の記述内容に従ってコードを生成します。サイズ指定の記述に矛盾がある場合は、以下の優先順位に従います。

- 1 サイズ指定子による指定方法
- 2 指示命令による指定方法
- 3 データ長およびインデックスレジスタ長の宣言

### 3.3.1 サイズ指定子による指定方法

命令にサイズ指定子を続けて記述します。構造化記述行でも指定できます。

- ・ 構造化記述行で指定するには、演算の対象となる変数にサイズ指定子を続けて記述します。

サイズ指定子	データサイズ
.B	バイト
.W	ワード
.D	ダブルワード

- ・ 命令が扱えないサイズは指定できません。
- ・ 分岐命令、リターン命令、フラグ命令等には記述できません。

#### 記述例

```
LDA.B      A, #0
MOVW.W    SYM1, SYM2
[LAB1].W = 10
E         = [WORK].D
```

### 3.3.2 指示命令による指定方法

領域を確保する指示命令およびデータを格納する指示命令は、同時にそのデータサイズを指定することになります。

データサイズ	指示命令
バイト	.BLKB、.BYTE、.GLBB
ワード	.BLKW、.WORD、.GLBW
ダブルワード	.BLKD、.DWORD、.GLBD

#### 注意事項

.GLBB、.GLBW、.GLBDは構造化記述内でのみ有効となります。

#### 記述例

```
.SECTION    ramarea
workb:     .BLKB      1
workw:     .BLKW      1
workd:     .BLKD      1

.SECTION    sample
LDA.B      A,workb
LDA.W      A,workw
LDA.D      E,workd
```

### 3.3.3 データ長およびインデックスレジスタ長の宣言

データ長の宣言は、マクロコンピュータのデータ長選択フラグ(m)の状態をアセンブラに対して宣言するものです。インデックスレジスタの宣言はインデックスレジスタ長選択フラグ(x)の状態を宣言するものです。

- ・ 指示命令でそれぞれのデータ長を指定します。

指示命令	内容
.DATA	データ長の宣言を行います。
.INDEX	インデックスレジスタ長の宣言を行います。
.LENGTH	データ長及びインデックスレジスタ長の宣言を一度に行います。

#### 注意事項

指示命令による宣言と各フラグの状態が矛盾しないように、必ず同時に各フラグを設定する命令 (SEP,CLPなど) を記述してください。

#### 記述例

```
.DATA      8
.INDEX     8
SEP        m,x
LDA        A,sym
LDX        0

.LENGTH    16,16
CLP        m,x
MOVM       sym1,sym2
STX        sym1
```



## 3.4 ファイルのインクルード

as79は、ソースプログラムの任意の行で、インクルードファイルを読み込むことができます。ソースプログラムの可読性の向上などに利用できます。

### 3.4.1 インクルードファイルの記述規則

インクルードファイルの記述は、ソースプログラムの記述規則に従って記述してください。

#### 注意事項

---

指示命令".END"は、インクルードファイル内に記述できません。

---

### 3.4.2 インクルードファイルの読み込み

指示命令".INCLUDE"のオペランドに読み込みたいファイル名を記述します。

- ・ この行の位置にインクルードファイルの内容が全て読み込まれます。
- ・ 読み込まれた位置の連続したアドレスに配置されます。

## 3.5 ライブラリファイル

### 3.5.1 リロケータブルファイルのライブラリ化

ライブラリファイルを作成するにはライブラリアンlb79を使用します。lb79は、ライブラリファイルの新規作成、リロケータブルモジュールの追加 / 削除、ライブラリリストファイルの生成などを行います。

詳細はlb79の使用方法を参照してください。

### 3.5.2 ライブラリファイルの参照

次の条件の全てを満たした場合に、ln79はライブラリファイルに登録されているリロケータブルファイルをリンクします。

コマンド行でライブラリファイル参照を指定した場合  
指定された全てのリロケータブルファイルを配置した結果、値の決定しなかったグローバルラベルが残っている場合

#### 注意事項

---

ln79は、必要なグローバルラベルの定義を行っているリロケータブルファイル全体をリンクします。

---

#### ライブラリファイルの参照規則

ln79は、次の順序でリンクするリロケータブルファイルを決定します。

- 1 ライブラリファイルに登録されているリロケータブルファイルのグローバルラベル情報を検索します。リロケータブルファイルの参照は、ライブラリファイルに登録されている順に行います。
- 2 ライブラリファイルから検索したラベルと、値の決まっていないラベルとを比較し、一致するものがあれば、ライブラリファイル内のリロケータブルファイルをリンクします。
- 3 ライブラリファイル内のリロケータブルファイルを一巡した結果、値の決定していないグローバルラベルが残った場合（ライブラリファイルに登録されているリロケータブルファイルに外部参照ラベルが存在した場合）、再度ライブラリファイルを登録順に検索します。

## 3.6 ソースレベルデバッグ対応

7900シリーズ対応の高級言語で開発したソースプログラムに対して、ソースレベルでのデバッグを可能にする高級言語デバッグ情報を出力します。

## 3.7 IEEE-695フォーマット準拠

AS79が生成するバイナリ形式のファイルはIEEE-695フォーマットで出力します。IEEE-695フォーマットに準拠したフォーマットを採用している他の7900シリーズ用開発ツールで利用されます。

IEEE(Institute of Electrical and Electronics Engineers:アメリカ電気電子技術者協会)

## 3.8 条件アセンブル機能

指示命令`.IF`, `.ELIF`, `.ELSE`, `.ENDIF`を使ってアセンブラ実行時に不要な部分をアセンブルしないような制御が可能です。

- ・ 指示命令`.IF`および`.ELIF`のオペランドに条件を記述します。
- ・ 条件アセンブルはネスト記述が可能です。

## 3.9 マクロ命令の定義と参照

マクロ命令とは、一つ以上の命令（ニーモニック）のブロックを一つの命令で置き換えたものをいいます。

### 3.9.1 マクロ命令の定義

マクロ命令は、指示命令.MACRO～.ENDMで定義します。詳細は指示命令.MACROの説明を参照してください。

- ・ マクロ定義には、命令のオペランドに相当する仮引数を定義できます。
- ・ マクロ命令は再帰的な定義が可能です。
- ・ 同一名のマクロの再定義ができます。
- ・ マクロ命令を定義している所では、コードの生成はされません。したがって、セクションの範囲外に記述できます。
- ・ マクロボディには、ソースプログラムに記述可能な全ての命令を記述できます。
- ・ マクロ命令の定義内で、マクロローカルラベルを使用できます。

### 3.9.2 マクロローカルラベル

マクロ命令の定義の際にマクロローカルラベルを使用できます。

- ・ マクロ定義内で、指示命令".LOCAL"で宣言したラベルがマクロローカルラベルとなります。
- ・ マクロローカルラベルは、マクロ定義内でのみ使用できます。
- ・ 異なるマクロ命令の定義であれば、同じ名前のマクロローカルラベルを記述できません。
- ・ マクロローカルラベルとして使用するラベルは、そのラベルを定義する以前に、マクロローカルラベルであることを宣言してください。

### 3.9.3 マクロの呼び出し

マクロ命令の定義より後の行であれば、任意の箇所でマクロの呼び出しを行えます。

- ・ 指示命令.MACROを記述した行で定義したマクロ名を記述します。
- ・ マクロ呼び出しを行った場所にコードが生成されます。
- ・ マクロ命令の前方参照（マクロ呼び出し行よりも後の行で定義されているマクロ名を記述すること）はできません。必ずマクロ定義は、呼び出し行よりも前の行に記述してください。
- ・ マクロ命令の外部参照（別のファイルで定義されているマクロ名を記述すること）はできません。複数のファイルから同一のマクロを呼び出す場合は、インクルードファイル内にマクロを定義し、そのファイルをインクルードしてください。

## 3.10 繰り返しマクロ機能

指示命令.MREPEAT～.ENDRで繰り返しマクロ機能を使用できます。

- ・ 指示命令".MREPEAT"と".ENDR"で囲まれた命令を、指定した回数分コード生成します。
- ・ 繰り返しマクロは、記述した行でコード生成されます。
- ・ 繰り返しマクロの定義行にはラベルを記述できます。

### 注意事項

---

このラベルは、マクロ名ではありません。繰り返しマクロのマクロ呼び出しはありません。

---

## 3.11 プリデファインドマクロ機能

ANDM、ANDMB、ORAM、ORAMB命令を用いてメモリの任意ビットのクリア、セットを行うプリデファインドマクロ機能 "CLB"、"CLBB"、"SEB"、"SEBB" を使用できます。

- ・ CLBは7900命令"ANDM"命令を用いてメモリの任意ビットをクリアします。
- ・ CLBBは7900命令"ANDMB"命令を用いてメモリの任意ビットをクリアします。
- ・ SEBは7900命令"ORAM"命令を用いてメモリの任意ビットをセットします。
- ・ SEBBは7900命令"ORAMB"命令を用いてメモリの任意ビットをセットします。
- ・ プリデファインドマクロは記述した行でコード生成されます。
- ・ プリデファインドマクロの記述行にはラベルを記述できます。

## 3.12 行の連結機能

AS79では、行の連結機能を使って一つの命令を複数行にわたって記述することができます。

- ・ 行末に'¥¥'を記述した次の行を、'¥¥'を記述した位置に連結します。
- ・ '¥¥'を記述した行にコメントを記述できます。ただし、連結結果にはコメントは出力されません。
- ・ '¥¥'を記述した行でエラーが発生した場合、連結される最終行に対して出力されます。

### 注意事項

連結された結果の行の最大文字数は512文字です。ただし、連結される行の先頭のスペース及びタブは文字数に含まれません。

2バイトコード文字の直後に'¥'を記述した場合、'¥¥'と認識される場合がありますのでご注意ください。

#### 行連結の記述例1

```
.BYTE 1, \\
      2,  \\
      3    \\
      ,4
```

#### 連結結果1

```
.BYTE 1, 2, 3, 4
```

#### 行連結の記述例2

```
.BYTE 1, \\ ;commennt
      2, ;Comment  \\
      3 ;COMMENT
```

#### 連結結果2

```
.BYTE 1, 2, ;Comment
      3 ;COMMENT
```

#### 行連結の記述例3

```
.BYTE 1, \\
      2, \\
      3,  \\
      4
```

#### 連結結果3

```
.BYTE 1, 2, 3, 4
```

## 3.13 環境変数の参照

AS79に含まれる各プログラムは次の環境変数を参照します。

環境変数名	環境変数を参照するプログラム
AS79COM	as79
BIN79	as79
INC79	as79
LIB79	ln79
TMP79	as79,ln79,lb79

### 3.13.1 AS79COM

as79のコマンドオプションを指定します。本環境変数にコマンドオプションを指定しておくことで、コマンド行の入力を減らすことができます。

本環境変数に設定できるコマンドオプションを次に示します。コマンドオプションの詳細については「as79の操作方法」を参照してください。

オプション	機能
-A	アブソリュートアドレッシングを選択する。
-B	64Kバイトモードで分岐最適化を行う。
-C	mac79,pre79及びasp79の起動コマンドラインを表示する。
-F	..FILE展開のファイル名をソースファイル名に固定する。
-G1	指示命令に依存しないで構造化記述命令の展開を実行する。
-G2	構造化記述命令の代入文の展開コードを制御する。
-H	リストファイルのヘッダを出力しない。
-JF	外部シンボルをFARで処理する。
-JN	外部シンボルをNEARで処理する。
-L	リストファイルを生成する。
-LC	行連結をそのままにリストファイルに出力する。
-LD	.DEFINEを置き換える以前の情報をリストファイルに出力
-LI	条件アセンブルの条件が偽の部分リストファイルに出力する。
-LM	マクロ展開行をリストファイルに出力する。
-LS	構造化記述命令の展開行をリストファイルに出力する。
-M6	DPRレジスタを6ビットとしてコードを生成する。
-M8	DPRレジスタを8ビットとしてコードを生成する。
-N	高級言語デバッグ情報を出力しない。
-P	構造化記述命令を変換する。
-Q	アドレッシングモードの不一致を検出する。
-S	ローカルシンボル情報を出力する。
-SM	システムラベルを含むローカルシンボル情報を出力する。
-T	タグファイルを生成する。

## AS79の特徴

### AS79COMの設定方法

#### パソコン版

```
SET AS79COM=-L -N -S -T
```

#### ワークステーション版

```
setenv AS79COM '-L -N -S -T'
```

#### 注意事項

ワークステーション上で、スペースを含む文字列を環境変数に設定する場合は、必ずクォーテーションで囲って指定してください。

### AS79COMの設定解除方法

#### パソコン版

```
SET AS79COM=
```

#### ワークステーション版

```
unsetenv AS79COM
```

### AS79COMの使用例

as79は、AS79COMが設定されている場合、次の順序でコマンドオプションを設定します。

- 1 AS79COMに設定されているコマンドオプションを設定します。
- 2 コマンド行から入力されたコマンドオプションを設定します。

次に、AS79COMにオプションを設定した例、コマンド行からコマンドオプションを入力した例及び有効となるコマンドオプションの例を示します。



## AS79の特徴

### AS79COMの設定例

```
SET AS79COM=-L -N -S -T
```

#### コマンド入力例(1)

```
>as79 -Dsym=0 --N
```

#### as79実行時に有効となるオプション(1)

```
-Dsym=0 -L -S -T
```

#### コマンド入力例(2)

```
>as79 -O¥tmp --T -SM -LM
```

#### as79実行時に有効となるオプション(2)

```
-O¥tmp -N -SM -LM
```

### 3.13.2 BIN79

マクロプロセッサ、構造化プリプロセッサ及びアセンブルプロセッサを格納したディレクトリを設定します。

- ・ アセンブラドライバas79が本環境変数を参照します。
- ・ 本環境変数を設定した場合は、マクロプロセッサと構造化プリプロセッサ及びアセンブルプロセッサを必ず、記述したディレクトリに格納してください。
- ・ 複数のディレクトリを指定できます。複数のディレクトリが指定されている場合は左から順にディレクトリを検索します。
- ・ 環境変数が設定されていない場合は、次の順序でプログラムを検索します。
  - 1 カレントディレクトリ
  - 2 コマンドパスディレクトリ

### 3.13.3 INC79

ソースファイルに記述されているインクルードファイルを検索するディレクトリを設定します。

- ・ as79が本環境変数を参照します。
- ・ 複数のディレクトリを指定できます。複数のディレクトリが指定されている場合は左から順にディレクトリを検索します。

### 3.13.4 LIB79

ライブラリファイルを検索するディレクトリを設定します。

- ・ In79が本環境変数を参照します。
- ・ 複数のディレクトリを指定できます。複数のディレクトリが指定されている場合は左から順にディレクトリを検索します。

### 3.13.5 TMP79

プログラムの実行に必要なテンポラリファイルを生成するディレクトリを設定します。

- ・ as79、In79及びlb79が本環境変数を参照します。
- ・ 作業ファイルは各プログラムの処理終了時に消去されます。

### 3.13.6 環境変数設定例

#### PC版

```
>SET INC79=C:¥COMMON;C:¥PROJECT
```

複数のディレクトリを設定する場合、ディレクトリ名をセミコロンで区切って記述してください。

#### EWS版

```
>setenv INC79 /usr/common:/usr/project
```

複数のディレクトリを設定する場合、ディレクトリ名をコロンで区切って記述してください。

## 3.14 OSへの戻り値

AS79の各プログラムは実行を終了する際に、終了時の状態を数値でOSに戻します。

戻り値	内容
0	プログラムは正常に終了しました。
1	コントロールCの入力によって、強制的にプログラムを終了しました。
2	OSのファイルシステムまたはメモリシステムに関するエラーが発生しました。
3	処理対象のファイルに原因があるエラーが発生しました。
4	コマンド行の入力に関するエラーが発生しました。

## 第4章 AS79操作方法

---

AS79に含まれる各プログラムの基本的な操作方法について説明します。

AS79に含まれるプログラムの操作は、全てパソコン又はワークステーションのプロンプトからコマンドを入力することで行います。

### 4.1 コマンド入力時の注意事項

Windows版では、MS-DOSプロンプト上で操作を行ってください。

ワークステーション上では、起動プログラム名は必ず英子文字で入力してください。

ワークステーション上では、ファイル名の大文字小文字が区別されます。

コマンドオプションは、パソコン及びワークステーションのいずれにおいても英大文字と英小文字は区別しません。

起動プログラム名とファイル名の間には必ずスペースを記述してください。

ファイル名と各コマンドオプションの間には必ずスペースを記述してください。

コマンドオプションを指定する際には、必ず"-"（ハイフン）を添付して記述してください。

ワークステーション上では、ファイル名に二箇所以上のピリオド(.)で区切ったものを使用できますが、AS79を使用する上ではファイル名に含めることのできるピリオド(.)は、一箇所だけです。

ファイルの拡張子名（ピリオド以降の文字）があらかじめ決められているものがありますので、AS79のそれぞれのプログラムの起動方法を確認してください。

## 4.2 コマンド行の構成

コマンド行では、次の情報を入力してください。

### 4.2.1 プログラム名

使用するプログラムの名前です。

#### 注意事項

---

ワークステーション上では、必ずプログラム名を小文字で入力してください。

---

### 4.2.2 コマンドパラメータ

プログラムを正しく実行するために必要な情報全てをコマンドパラメータと呼びます。コマンドパラメータには、起動するプログラムの処理対象となるファイル名や、プログラムの機能を記号で示したコマンドオプションが含まれます。

コマンドパラメータは次の情報を含みます。

#### ファイル名

プログラムの処理対象となるファイルの名前です。

#### 注意事項

---

ワークステーション上では、大文字、小文字を正しく入力してください。

---

#### コマンドオプション

各プログラムの機能を利用するために、コマンドオプションを指定してください。

## 第5章 as79の使用方法

as79の機能を利用するための操作方法について説明します。as79の基本機能は、ソースファイルから、リロケートブルモジュールファイルを生成します。

### 5.1 as79コマンドオプション

オプション	機能
-.	メッセージを画面に出力しない。
-A	アブソリュートアドレッシングを選択する。
-B	64Kバイトモードで分岐最適化を行う。
-C	mac79,pre79及びasp79の起動コマンドラインを表示する。
-D	シンボルに定数を設定する。
-finfo	インスペクタ情報を生成する。
-F	..FILE展開のファイル名をソースファイル名に固定する。
-G1	指示命令による演算サイズを無視して構造化命令を展開する。
-G2	構造化記述命令の代入文の展開コードを制御する。
-H	リストファイルのヘッダを出力しない。
-I	インクルードファイルの検索ディレクトリを指定する。
-JF	外部シンボルをFARで処理する。
-JN	外部シンボルをNEARで処理する。
-L	リストファイルを生成する。
-M6	DPRレジスタを6ビットとしてコードを生成する。
-M8	DPRレジスタを8ビットとしてコードを生成する。
-N	高級言語デバッグ情報を出力しない。
-O	ファイルの生成ディレクトリを指定する。
-P	構造化記述命令を変換する。
-Q	アドレッシングモードの不一致を検出する。
-S[M]	ローカルシンボル情報を出力する。
-T	タグファイルを生成する。
-V	プログラムのバージョンを表示する。
-W	ダブルワード命令の出力を抑止する。
-Xプログラム名	プログラム名で指定されたアプリケーションを起動する。

## 5.2 as79コマンドパラメータの指定規則

as79のコマンドパラメータは、次の規則に従って指定してください。

### 5.2.1 コマンドパラメータの指定順序

- ・ 任意の順序で指定できます。

```
>as79 file ... [option ...]
>as79 [option ...] file ...
```

file=ソースファイル名

option=コマンドオプション

#### ソースファイル名 (必須)

- ・ 必ず、1つ以上のソースファイル名を指定してください。
- ・ ソースファイル名には、パスの指定ができます。
- ・ ソースファイル名は、80個まで指定できます。

#### 注意事項

複数指定されたソースファイルのうち、エラーを含むソースファイル以降に指定されたファイルは処理しません。

- ・ 拡張子が".a79"であるファイルは、拡張子を省略できます。

#### コマンドオプション

- ・ コマンドオプションは、省略できます。
- ・ コマンドオプションは、複数指定できます。
- ・ コマンドオプションには、文字列や数値が指定できるものがあります。

#### 注意事項

コマンドオプションと文字列又は数値の間には、スペース又はタブを記述しないでください。

- ・ 後に続くコマンドオプションを無効にするには、"--"を添付してコマンドオプションを記述してください。

#### 注意事項

コマンドオプションを無効にできるのは、as79のコマンドオプションだけです。その他のプログラムを起動する場合には使用できません。

例)

```
>as79 sample -L
```

オプション"L"が有効です。

```
>as79 sample -S
```

オプション"S"が有効です。

```
>as79 sample -L -S --L
```

オプション"S"だけが有効です。

```
>as79 sample -S -L --S
```

オプション"L"だけが有効です。

### 数値の指定方法

- ・ 数値は、必ず16進数で指定してください。
- ・ 数値の先頭がアルファベットになる場合は、必ず0を付加して指定してください。

例)

```
55  
5A  
0A5
```

### 5.2.2 インクルードファイルの検索ディレクトリ

- ・ インクルードファイル名は、指示命令".INCLUDE"で指定します。したがって、指示命令".INCLUDE"のオペランドにパスが記述されていれば、そのディレクトリを検索します。
- ・ 指示命令のオペランドにパスの指定が無い場合は、カレントディレクトリを検索します。このとき、カレントディレクトリに該当ファイルが無く、環境変数"INC79"が設定されている場合は、INC79の設定ディレクトリも検索します。
- ・ 複数のディレクトリが環境変数に設定されている場合は、左から順にディレクトリを検索します。

## 5.3 as79コマンドオプション

以降に、コマンドオプションの指定規則を説明します。

-.

---

## 画面へのメッセージ出力停止

---

### 機能

- ・ as79が処理を行う際のメッセージを出力しません。
- ・ エラーメッセージ、ワーニングメッセージ及び指示命令".ASSERT"によるメッセージは出力されます。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>as79 -. sample
```

sampleを処理した結果エラーが発生した場合のような出力結果が得られません。

```
>as79 -. sample
```

```
sample.a79 2 Error (as79): Section type is not appropriate
```



-A

---

## アブソリュートアドレッシングモード選択

---

### 機能

- ・ アドレッシングモード指定されていないシンボルのアクセスに対し、アブソリュートアドレッシングモードを選択します。
- ・ 本コマンドオプション省略時は上記に対してアブソリュートロングアドレッシングが選択されます。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>as79 -A sample
```

アブソリュートアドレッシングモードを選択します。

```
>as79 sample
```

アブソリュートロングアドレッシングモードを選択します。

## -B

---

### 無条件分岐命令の最適化

---

#### 機能

- ・ コードサイズを64Kバイト以下とみなして、構造化記述命令"GOTO"にたいして無条件分岐命令"BRA","JMP"のいずれかを生成します。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -B sample
```

"BRA"又は"JMP"命令にてコード展開します。

```
>as79 sample
```

"BRA"、"BRAL"又は"JMPL"にてコード展開します。

## -C

### コマンド起動行表示

#### 機能

- ・ 本オプションを指定することによって、as79がマクロプロセッサ、構造化プリプロセッサ、アセンブラプロセッサを起動する際に各プログラムに渡したコマンドオプションを画面上で確認することができます。環境変数で設定されているコマンドオプションも含めて表示します。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -C -N sample
```

AS79COMに'-L -T'が設定されている場合に、次のような出力が得られます。

as79の正常起動時に出力される All Rights Reserved. の次の行から示します。

```
>as79 -C -N sample

( sample.a79 )
mac79 -L -T sample.a79
macro prossesing now

pre79 -L -T sample.m79
structured prossesing now

asp79 -L -T sample.p79
assembler prossesing now
TOTAL ERROR(S)      00000
:
```

```
>as79 -. -C -N sample
```

画面へのメッセージ出力停止オプションと組み合わせた場合は、次のような出力が得られます。

```
>as79 -. -C sample
mac79 -L -T sample.a79
pre79 -L -T sample.m79
asp79 -L -T sample.p79
```

## -D

# シンボル定数設定

### 機能

- ・ シンボルに値を設定します。
- ・ 値は絶対値として扱います。

### 注意事項

本オプションで定義したシンボルは、ソースファイル中の先頭箇所にシンボル定義を行った場合と同様の処理になります。ただし、アセンブラリストファイルには出力されません。

- ・ 本オプションで定義したシンボルは、ソースファイル内に記述したシンボル定義と同じに扱われます。つまり、ソースファイル内に同一名のシンボル定義がされている場合は、ファイル内に記述しているシンボル定義行で再定義したことになります。
- ・ コマンド行で複数のファイルを指定した場合、本オプションで定義したシンボルは、全てのファイル内で定義されることになります。

### 記述規則

- ・ `-D (シンボル名) = (数値)` のように指定してください。
- ・ コマンド行の任意の位置に記述できます。
- ・ コマンドオプションとシンボル名の間には、スペース又はタブを記述しないでください。
- ・ 複数のシンボルに値を定義できます。複数のシンボル定義を行う場合は、次のように、コロンで区切って続けて記述してください。  
`-D (シンボル名) = (数値) : (シンボル名) = (数値)`
- ・ コロンの前後に、スペース又はタブは記述できません。

### 記述例

```
>as79 -Dname=1 sample
```

nameというシンボルに1を設定します。

```
>as79 -Dname=1:symbol=1 sample
```

name及びsymbolというシンボルに1を設定します。

```
>as79 -Dname=1 sample1 sample2
```

sample1 と sample2 のファイルに対して、name というシンボルを定義します。

## -finfo

---

### インスペクタ情報を生成

---

#### 機能

- ・ NC79の '- finfo'オプションで生成された各情報、またはアセンブラ指示命令で記述されたインスペクタ情報をリロケータブルモジュールファイルに出力します。

#### 注意事項

---

TMをご使用の場合、本オプションはデフォルトで指定されます。

---

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。
- ・ 文字の大文字と小文字を区別しますので、すべて小文字で指定してください。

#### 記述例

```
>as79 -L -S -finfo sample
```

## -F

---

### ..**FILE**展開制御

---

#### 機能

- ・ 指示命令..**FILE**が展開するファイル名を、コマンド行から指定されたソースファイル名に固定します。

#### 記述規則

- ・ コマンド行の任意の位置に記述できます。

#### 記述例

```
>as79 -F sample
```

sample.a79ソースファイル内でインクルードされているファイル"include.inc"内に記述されている指示命令..**FILE**が展開するファイル名は、"sample"となります。

本オプションが指定されていない場合は、..**FILE**が展開するファイル名は、"include.inc"となります。

## -G1

---

### 指示命令による演算サイズを無視

---

#### 機能

- ・ 構造化記述命令に記述されているシンボルを展開する場合、次の指示命令で定義または宣言されたシンボルを、指示命令によって宣言されるデータサイズで扱いません。

```
.GLBB      .BLKB      .BYTE  
.GLBW .BLKW  .WORD  
.GLBD      .BLKD      .DWORD
```

#### 記述規則

- ・ コマンド行の任意の位置に記述できます。
- ・ コマンドオプション-Pと同時に指定してください。

#### 記述例

```
>as79 -P -G1 sample
```

## -G2

---

### 構造化命令の代入文の展開命令を制御

---

#### 機能

- ・ アキュムレータAにアキュムレータBの内容を代入する文 ( A = B ) に対して生成されるアセンブリ命令を制御します。
- ・ 本オプションを指定するとアキュムレータAとBの内容を交換するアセンブリ命令 ( XAB ) が生成されます。

#### 記述規則

- ・ コマンド行の任意の位置に記述できます。
- ・ コマンドオプション-Pと同時に指定してください。

#### 記述例

```
>as79 -P -G2 sample
```



## -H

---

# アセンブラリストファイルのヘッダ出力停止

---

### 機能

- ・ アセンブラリストファイルのヘッダ情報が出力されません。

### 注意事項

---

abs79が処理するアセンブラリストファイルを生成する場合は、本オプションを指定しないでください。abs79が正しくアセンブラリストファイルを処理できません。

---

### 記述規則

- ・ コマンド行の任意の位置に記述できます。
- ・ コマンドオプション'-L'と同時に指定してください。

### 記述例

```
>as79 -L -H sample
```

sample.lstファイルにはヘッダ情報が出力されません。

-I

---

## インクルードファイル検索ディレクトリ指定

---

### 機能

- ・ 指定されたディレクトリからソースファイルに記述されている".INCLUDE"で指定されたインクルードファイルを検索します。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。
- ・ "-I"に続けてディレクトリパス名を指定してください。
- ・ オプションとディレクトリパス名の間にはスペース又はタブは記述できません。

### 記述例

```
>as79 -I¥work¥include
```

¥work¥includeディレクトリから指示命令".INCLUDE"のオペランドに記述されているインクルードファイルを検索します。

## -JF

---

### ラベルシンボル属性制御

---

#### 機能

- ・ 構造化記述命令を変換する際に、.NEAR/.FAR宣言していないラベルシンボルの属性をオプション(-JF)指定時に.FAR属性にします。

#### 記述規則

- ・ コマンド行の任意の位置に記述できます。
- ・ 本オプションはコマンドオプション-Pを指定した場合のみに指定してください。

#### 記述例

```
>as79 -P -JF sample
```

分岐距離が指定されていないラベルシンボルがFAR属性になります。

```
>as79 -P sample
```

分岐距離が指定されていないラベルシンボルがFAR属性になります。

## -JN

---

### ラベルシンボル属性制御

---

#### 機能

- ・ 構造化記述命令を変換する際に、.NEAR/.FAR宣言していないラベルシンボルの属性をオプション(-JN)指定時に.NEAR属性にします。
- ・ 本オプション省略時は.FAR属性になります。

#### 記述規則

- ・ コマンド行の任意の位置に記述できます。
- ・ 本オプションはコマンドオプション-Pを指定した場合のみに指定してください。

#### 記述例

```
>as79-P -JN sample
```

分岐距離が指定されていないラベルシンボルがNEAR属性になります。

```
>as79 -P sample
```

分岐距離が指定されていないラベルシンボルがFAR属性になります。

## -L

## アセンブラリストファイル生成

## 機能

- ・ リロケータブルモジュールファイルの他にアセンブラリストファイルを生成します。
- ・ 生成したリストファイルの拡張子は、".lst"となります。
- ・ コマンドオプション"-O"でディレクトリを指定している場合は、指定したディレクトリにアセンブラリストファイルを生成します。

## 記述規則

- ・ コマンド行の任意の位置に指定できます。
- ・ 本オプションには、'C','D','I','M','S'のファイルフォーマット指定子を指定できます。
- ・ ファイルフォーマット指定子と-Lの間に、スペース又はタブは記述できません。
- ・ ファイルフォーマット指定子は、同時に複数の指定が可能です。
- ・ ファイルフォーマット指定子の指定順序は任意です。
- ・ 本オプションは、環境変数"AS79COM"に設定できます。
- ・ 次にそれぞれの指定子の機能を示します。

指定子	内容
C	行連結をそのままリストファイルに出力する。
D	.DEFINEを置き換える以前の情報をリストファイルに出力する。
I	条件アセンブルの条件が偽の部分をリストファイルに出力する。
M	マクロ展開行をリストファイルに出力する。
S	構造化記述命令の展開行をリストファイルに出力する。

## 記述例

```
>as79 -LIMS sample
>as79 -LSMI sample
```

## -M6

---

### CPU動作モード選択

---

#### 機能

- ・ オプション(-M6)指定時は拡張ダイレクトアドレッシングモード（4本のダイレクトページレジスタを用いたアドレッシングが使用可能）になります。
- ・ オプションを指定しない時は6ビットオフセットモードで処理されます。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -M6 sample
```

6ビットオフセットモード（4本のダイレクトページレジスタが使用可能）になります。

```
>as79 sample
```

6ビットオフセットモード（4本のダイレクトページレジスタが使用可能）になります。

## -M8

---

### CPU動作モード選択

---

#### 機能

- ・ オプション(-M8)指定時はダイレクトアドレッシングモード（1本のダイレクトページレジスタを用いたアドレッシングが使用可能）になります。
- ・ オプションを指定しない時は6ビットオフセットモードで処理されます。

#### 注意事項

---

8ビットオフセットモードを使用する場合は必ず本オプションを指定してください。

---

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -M8 sample
```

8ビットオフセットモード（1本のダイレクトページレジスタが使用可能）になります。

```
>as79 sample
```

6ビットオフセットモード（4本のダイレクトページレジスタが使用可能）になります。

-N

---

## 行情報の出力停止

---

### 機能

- ・ ソース行情報をリロケータブルモジュールファイルに出力しません。
- ・ リロケータブルモジュールファイルのサイズを縮小できます。

### 注意事項

---

本オプションを指定して生成したリロケータブルモジュールファイルから作成したアブソリュートモジュールファイルでは、ソースレベルでのデバッグはできません。

---

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>as79 -N sample
```



## -O

### 生成ファイルの出力先ディレクトリ指定

#### 機能

- ・ アセンブラが生成するリロケータブルモジュールファイル、アセンブラリストファイル及びアセンブラエラータグファイルの出力先ディレクトリを指定します。
- ・ ディレクトリ名には、ドライブ名を含めて指定できます。また、相対パスによる指定も可能です。

#### 記述規則

- ・ -O (ディレクトリ名) のように記述してください。
- ・ 本オプションとディレクトリ名の間には、スペース又はタブは記述できません。

#### 記述例

```
>as79 -Oc:¥work¥asmout sample
```

リロケータブルモジュールファイルを、Cドライブの¥work¥asmoutディレクトリに生成します。

```
>as79 sample -O..¥tmp
```

リロケータブルモジュールファイルを、カレントディレクトリの親ディレクトリに属す、tmpディレクトリに生成します。

```
>as79 -Oc:¥work¥asmout sample -L -T
```

リロケータブルモジュールファイル、アセンブラエラータグファイル及びアセンブラリストファイルを、Cドライブの¥work¥asmoutディレクトリに生成します。

-P

---

## pre79起動制御

---

### 機能

- ・ pre79を起動し、構造化記述をアセンブリ言語に変換します。

### 記述規則

- ・ コマンド行の任意の位置に指定出来ます。

### 記述例

```
>as79 -P sample
```

pre79を起動します。

## -Q

---

### アドレッシングモードチェック

---

#### 機能

- ・ アドレッシング指定指示命令(.DPnSYM, .DTSTM, .LGSYM)で指定されたラベルに対する命令オペランド中のアドレッシング指定子が一致していない場合、ワーニングを出力します。

#### 記述規則

- ・ コマンド行の任意の位置に指定出来ます。

#### 記述例

```
>as79 -Q sample
```

アドレッシング指定指示命令とアドレッシング指定子のモード指定が一致していない場合、ワーニングを出力します。

```
.DTSYM LAB  
STA A,DP:LAB;Warning is output
```

## -S[M]

---

### ローカルシンボル情報の出力指定

---

#### 機能

- ・ ローカルシンボル情報をリロケータブルモジュールファイルに出力します。
- ・ 本オプションに'M'を付加することで、システムラベル情報もリロケータブルモジュールファイルに出力します。
- ・ 本オプションを指定して生成したりロケータブルモジュールファイルから作成したアブソリュートモジュールファイルでは、ローカルシンボルを用いたシンボリックデバッグが可能になります。

#### 注意事項

---

シンボリックデバッグ可能なシンボル及びラベルの情報については、ln79が出力するマップファイル(.map)で確認できます。

本オプションを省略して生成したりロケータブルモジュールファイルから作成したアブソリュートモジュールファイルでは、ローカルシンボル情報を含むデバッグはできません。

---

#### 記述規則

- ・ システムラベル情報とローカルラベル情報を同時に出力するには、"-SM"と入力してください。
- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -S sample
```

sample.a79のローカルシンボル情報をsample.r79に出力します。

```
>as79 -SM sample
```

sample.a79のシステムラベル情報及びローカルシンボル情報をsample.r79に出力します。

-T

---

## アセンブラエラータグファイル生成

---

### 機能

- ・ アセンブラエラーが発生した場合に、アセンブラエラータグファイルを生成します。
- ・ エディタのタグジャンプ機能を利用可能なフォーマットでファイルを出力します。
- ・ 本オプションを指定しても、エラーがゼロの場合はファイルを生成しません。
- ・ エラーが発生した場合は、リロケータブルモジュールファイルは生成しません。ワーニングのみの発生の場合は、リロケータブルモジュールファイルを生成します。
- ・ エラータグファイル名は、ソースファイル名の拡張子を".atg"に変更したものになります。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>as79 -T sample
```

エラーが発生した場合、"sample.atg"ファイルを生成します。

-V

---

## バージョン表示

---

### 機能

- ・ 本オプションは、as79に含まれる全てのプログラムのバージョン番号を表示して、処理を終了します。

### 注意事項

---

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・ 本オプションのみを指定してください。

### 記述例

```
>as79 -V
```

-W

---

## ダブルワード命令出力抑止

---

### 機能

- ・ 本オプションを指定した場合、構造化記述を使用している箇所において指示命令".BLKD",".DWORD"及び".GLBD"により宣言されたシンボルを記述している場合でもワード命令を使用し展開します。
- ・ サイズ指定子".D"によりサイズ指定を行っている場合は、ダブルワード命令を使用して展開します。
- ・ 本オプションを省略した場合、ダブルワード命令を使用し展開します。

### 注意事項

本オプションは、構造化記述行に対してのみに有効です。アセンブラニーモニック命令で記述された行には無効です。

---

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>as79 -W sample
```

## -X

---

### 外部プログラムを起動

---

#### 機能

- ・ エラータグファイルを生成し、'-X'に続けて指定した実行プログラムを起動します。
- ・ 本オプションを指定した場合、'-T'の指定の有無に関わらずエラーが発生したときは、エラータグファイルを生成します。
- ・ コマンド行の任意の位置に指定できます。

#### 記述規則

- ・ -X(プログラム名)のように入力してください。プログラム名にはディレクトリパスが指定できます。
- ・ 本オプションとプログラム名の間、スペース又はタブは記述できません。
- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>as79 -Xedit sample
```



## 5.4 as79エラーメッセージ

'#' is missing

? '#'の記述がありません。

! 本オペランドには、イミディエイト値を記述してください。

')' is missing

? ')'の記述がありません。

! '('に対応する')を記述してください。

'(' is missing

? '('の記述がありません。

! ')'に対応する'('を記述してください。

'(' or ') ' is missing

? 括弧の数が一致していません。

! プログラム記述を確認してください。

',' is missing

? ','の記述がありません。

! オペランドの区切りには、カンマを記述してください。

'.EINSF' is missing for '.INSF'

? .INSFに対する.EINSFがありません。

! .INSFの記述位置を確認してください。

'.IF' is missing for '.ELIF'

? .ELIF に対する .IF がありません。

! .ELIF の記述位置を確認してください。

'.IF' is missing for '.ELSE'

? .ELSE に対する .IF がありません。

! .ELSE の記述位置を確認してください。

'.IF' is missing for '.ENDIF'

? .ENDIF に対する .IF がありません。

! .ENDIF の記述位置を確認してください。

'.INSF' is missing for '.EINSF'

? .EINSFに対する .INSFがありません。

! .EINSFの記述位置を確認してください。

'.MACRO' is missing for '.ENDM'

? .ENDM に対する .MACRO がありません。

! .ENDM の記述位置を確認してください。

'.MACRO' is missing for '.LOCAL'

? .LOCAL に対する .MACRO がありません。

! .LOCAL の記述位置を確認してください。 .LOCAL は、マクロブロック内にしか記述できません。

## as79の使用方法

'`.MACRO`' or '`.MREPEAT`' is missing for '`.EXITM`'

? `.EXITM` に対する `.MACRO` 又は `.MREPEAT` がありません。

! `.EXITM` の記述位置を確認してください。

'`.MREPEAT`' is missing for '`.ENDR`'

? `.ENDR` に対する `.MREPEAT` がありません。

! `.ENDR` の記述位置を確認してください。

'`.VER`' is duplicated

? '`.VER`' が重複指定されています。

! '`.VER`' は、1つのファイルに1回だけ記述できます。余分な'`.VER`'を削除してください。

'`:`' is missing

? '`:`'の記述がありません。

! ラベルの最後に、'`:`'を記述してください。

'`ALIGN`' is multiple specified in '`.SECTION`'

? `.SECTION` 定義行に複数の '`ALIGN`' 指定があります。

! 余分な '`ALIGN`' 指定を削除してください。

'`]`' is missing

? '`]`' の記述がありません。

! '[' に対応する '`]`' を記述してください。

'`{`' is missing

? '`{`'の記述がありません。

! '`{`'を記述してください。

Absolute addressing is not avail

? `.アブソリュートアドレッシング`にはできません。

! アドレッシングモードを確認して記述し直してください。

Addressing mode specifier is not appropriate

? アドレッシングモード指定子の記述に間違いがあります。

! アドレッシングモード指定子の記述方法を確認してください。

Already had DEFAULT statement

? 一つの `SWITCH` ブロック内に複数の `DEFAULT` 文が記述されています。

! `DEFAULT` 文を一つにしてください。

BREAK not inside structured statement

? 構造化ボディ外で `break` の記述を行っています。

! 構造化ボディ内に `break` を記述してください。

Bit-symbol is in expression

? 式中にビットシンボルがあります。

! ビットシンボルは式に記述できません。シンボル名を確認してください。

CASE not associates with SWITCH

- ? 構造化ボディ外でcaseの記述を行っています。
- ! 構造化ボディ内にcaseを記述してください。

CONTINUE not inside structured statement

- ? 構造化ボディ外でcontinueの記述を行っています。
- ! 構造化ボディ内にcontinueを記述してください。

Can't create Temporary file

- ? テンポラリファイルが生成できません。
- ! カレントディレクトリ以外にテンポラリファイルを作成するように、環境変数 'TMP79'にディレクトリを指定してください。

Can't create file 'XX'

- ? 'XX'ファイルが生成できません。
- ! ディレクトリ容量を確認してください。

Can't open '.ASSERT' message file 'XX'

- ? .ASSERT の出力ファイルをオープンできません。
- ! ファイル名を確認してください。

Can't open file 'XX'

- ? 'XX'ファイルがオープンできません。
- ! ファイル名を確認してください。

Can't open include file 'XX'

- ? インクルードファイルをオープンできません。
- ! インクルードファイル名を確認してください。インクルードファイルの格納ディレクトリを確認してください。

Can't read input file

- ? 入力ファイルへのアクセスができません。
- ! ファイルのパーミッションを確認してください。

Can't write '.ASSERT' message file 'XX'

- ? .ASSERT の出力ファイルに書き込みできません。
- ! ファイル名等を確認してください。

Can't write file 'XX'

- ? 'XX'ファイルに書き込むことができません。
- ! ファイル名等を確認してください。

Characters exist in expression

- ? 命令又は式中に余分な文字があります。
- ! 式の記述規則を確認してください。

Command line is too long

- ? コマンド行の文字数が多すぎます。
- ! コマンドを入力し直してください。

## as79の使用方法

Command option 'XX' cannot be used at the same time.

- ? 同時に指定できないコマンドオプションを指定しています。
- ! コマンドオプションの機能を確認してください。

DEFAULT not associates with SWITCH

- ? 構造化ボディ外でdefaultの記述を行っています。
- ! 構造化ボディ内にdefaultを記述してください。

Division by zero

- ? 0 除算が行われています。
- ! 式を記述し直してください。

ELIF not associates with IF

- ? elif記述に対応したifの記述がありません。
- ! ifの記述位置を確認してください。ifを記述してください。

ELSE not associates with IF

- ? else記述に対応したifの記述がありません。
- ! ifの記述位置を確認してください。ifを記述してください。

ENDIF not associates with IF

- ? endif記述に対応したifの記述がありません。
- ! ifの記述位置を確認してください。ifを記述してください。

ENDS not associates with SWITCH

- ? ends記述に対応したswitchの記述がありません。
- ! switchの記述位置を確認してください。switchを記述してください。

Error occured in executing 'XX'

- ? コマンド実行時にエラーが発生しました。
- ! 本エラーが発生した場合は、お手数ですが「ツールサポート窓口」までご連絡頂きますようお願い致します。

Function information is not defined

- ? インспекタ情報の関数情報が定義されていません。
- ! 関数情報を定義してください。

Ignore option 'XX'

- ? 不正なコマンドオプションが指定されました。
- ! 正しいコマンドオプションを指定してください。

Illegal directive command is used

- ? 不正な指示命令を記述しています。
- ! 正しい指示命令に記述し直してください。

Illegal file name 'XX'

- ? ファイル名が不正です。
- ! ファイル名の記述規則に従ったファイル名を指定してください。

### Illegal macro parameter

- ? マクロ引数に不正な記述があります。
- ! マクロ引数の記述内容を確認してください。

### Illegal macro statements

- ? '.IF'とネストが交差しています。
- ! '.IF'とネストが交差しないようにしてください。

### Illegal operand is used

- ? オペランドが間違っています。
- ! オペランドの記述方法を確認して、記述し直してください。

### Include nesting over

- ? インクルードのネスティングが多すぎます。
- ! インクルードレベルが9以下になるように記述し直してください。。

### Including the include file in itself

- ? インクルードファイル内で、自身をインクルードしています。
- ! インクルードファイル名を確認して、記述し直してください。

### Invalid bit-symbol exist

- ? 無効なビットシンボルの記述があります。
- ! ビットシンボルの定義を記述し直してください。

### Invalid chip mode

- ? コマンドオプション指定によるダイレクトアドレッシング選択モードとソースファイル中の記述モードが一致していません。
- ! 使用するダイレクトアドレッシングモードを確認してください。

### Invalid label definition

- ? 無効なラベル記述をしています。
- ! ラベル定義を記述し直してください。

### Invalid operand(s) exist in instruction

- ? 命令に無効なオペランドがあります。
- ! 命令のオペランドの記述方法を確認して、記述し直してください。

### Invalid option 'XX' is used

- ? コマンドオプションの指定が誤っています。
- ! コマンドオプションの指定方法を確認してください。

### Invalid option 'XX' is in environment data

- ? 環境変数に指定されているオプションの指定に誤りがあります。
- ! 環境変数'AS79COM'を設定し直してください。

### Invalid reserved word exist in operand

- ? オペランド中に予約語が記述されています。
- ! 予約語はオペランドに記述できません。オペランドを記述し直してください

Invalid symbol definition

- ? 無効なシンボル記述をしています。
- ! シンボルの定義を記述し直してください。

NEXT not associates with FOR

- ? next記述に対応したforの記述がありません。
- ! forの記述位置を確認してください。forを記述してください。

No '.END' statement

- ? .END の記述がありません。
- ! ソースプログラムの最後の行に.ENDを記述してください。

No '.ENDM' statement

- ? .ENDM 記述がありません。
- ! .ENDM の記述位置を確認してください。 .ENDM を記述してください。

No '.ENDR' statement

- ? .ENDR 記述がありません。
- ! .ENDR の記述位置を確認してください。 .ENDR を記述してください。

No '.SECTION' statement

- ? '.SECTION' の記述がありません。
- ! ソースプログラムには、必ず1つ以上の .SECTION を記述してください。
- ' .SECTION '記述より前にメモリに関する命令を記述していないか確認してください。

No ';' at the top of comment

- ? コメント先頭に ; が記述されていません。
- ! コメントの先頭には、セミコロンを記述してください。ニーモニック又はオペランドの記述に誤りがないか確認してください。

No .END statement

- ? .END の記述がありません。
- ! ソースプログラムの最後の行に.ENDを記述してください。

No .ENDIF statement

- ? .ENDIF 記述がありません。
- ! .ENDIF の記述位置を確認してください。 .ENDIF を記述してください。

No endif statement

- ? if構文に対応したendifの記述がありません。
- ! endifの記述位置を確認してください。 endifを記述してください。

No ends statement

- ? switch構文に対応したendsの記述がありません。
- ! endsの記述位置を確認してください。 endsを記述してください。

No input files specified

- ? 入力ファイルの指定がありません。
- ! 入力ファイルを指定してください。

No macro name

- ? マクロ名がありません。
- ! マクロ定義には、マクロ名を記述してください。

No next statement

- ? for構文に対応したnextの記述がありません。
- ! nextの記述位置を確認してください。nextを記述してください。

No space after mnemonic or directive

- ? ニーモニック、アセンブル指示命令の直後に空白文字がありません。
- ! 命令とオペランドの間に、空白文字を記述してください。

Not enough memory

- ? メモリが足りません。
- ! ファイルを分割して実行し直してください。又はメモリを増設してください。

No while statement

- ? do構文に対応したwhileの記述がありません。
- ! whileの記述位置を確認してください。whileを記述してください。

Not support

- ? サポートしていない構文記述があります。
- ! 構文を確認して、記述し直してください。

Operand expression is not completed

- ? オペランド記述に不足があります。
- ! オペランドの記述方法を確認して、記述し直してください。

Operand number is not enough

- ? オペランドが不足しています。
- ! オペランドの記述方法を確認して、記述し直してください。

Operand size is not appropriate

- ? オペランドのサイズが間違っています。
- ! オペランドの記述方法を確認してください。

Operand type is not appropriate

- ? オペランドの種類が間違っています。
- ! オペランドの記述方法を確認して、記述し直してください。

Operand value is not defined

- ? オペランドの値が未定義です。
- ! オペランドには確定値を記述してください。

Option 'XX' is not appropriate

- ? コマンドオプションXXの記述が間違っています。
- ! コマンドオプションを指定し直してください。

Questionable syntax

- ? 命令の記述に間違いがあります。
- ! 命令を記述し直してください。



Quote is missing

- ? 文字列に対する引用符の記述がありません。
- ! 文字列は引用符で囲って記述してください。

Reference to undefined symbol

- ? 未定義シンボルを参照しています。
- ! シンボルを定義してください。

Reserved word is used as label or symbol

- ? 予約語をラベル又はシンボルに用いています。
- ! ラベル又はシンボル名を記述し直してください。

Reference to undefined symbol

- ? 未定義のシンボルを参照しています。
- ! 定義しているシンボルを参照してください。

Right quote is missing

- ? 右側の引用符がありません。
- ! 引用符を記述してください。

SYMBOL is missing

- ? シンボルの記述がありません。
- ! シンボル名を記述してください。

Same items are multiple specified

- ? オペランドの同一項目を複数指定しています。
- ! オペランドの記述方法を確認して記述し直してください。

Same kind items are multiple specified

- ? オペランドの同種の項目を複数指定しています。
- ! オペランドの記述方法を確認して記述し直してください。

Section attribute is not defined

- ? セクションの属性が未定義です。このセクション内では指示命令".ALIGN"は記述できません。
- ! 指示命令".ALIGN"は、絶対属性セクション又はALIGN指定のある相対属性セクション内に記述してください。

Section has already determined as attribute

- ? セクションは既に相対属性に確定しています。指示命令".ORG"は記述できません。
- ! セクションの属性を確認してください。

Section name is missing

- ? セクション名がありません。
- ! オペランドにセクション名を記述してください。

Section type is multiple specified

- ? セクション定義行でセクションタイプの指定が重複しています。
- ! セクション定義行には"CODE","DATA","ROMDATA"の指定は1つだけ記述してください。



Section type is not appropriate

- ? セクションタイプの記述が間違っています。
- ! セクションタイプを記述し直してください。

Size error

- ? 式のサイズに矛盾が生じています。
- ! 各項のサイズを確認して、記述し直してください。

Size specifier is missing

- ? サイズ指定子がありません。
- ! サイズ指定子を記述してください。

Size specifier is not appropriate

- ? サイズ指定子の記述に間違いがあります。
- ! サイズ指定子を記述し直してください。

Source files number exceeds 80

- ? 指定したソースファイル数が多すぎます。
- ! 指定するソースファイル数を80以下にしてください。

Source line is too long

- ? 行連結子の処理後またはマクロ引数の変換後ソース行が512文字を越えています。
- ! 行連結子の処理後またはマクロ引数の変換後ソース行が512文字を越えないようにしてください。

String value exist in expression

- ? 式中に文字列式が記述されています。
- ! 式を記述し直してください。

Symbol definition is not appropriate

- ? シンボルの定義に間違いがあります。
- ! シンボル定義方法を確認して記述し直してください。

Symbol has already defined as another type

- ? シンボルは既に同一名で異なる指示命令で定義されています。指示命令".EQU"と".BTEQU"で同一のシンボル名を定義できません。
- ! シンボル名を変更してください。

Symbol is undefined

- ? シンボルが定義されていません。
- ! シンボル名を確認してください。また、シンボルの前方参照もできません。

Symbol was already defined as the same type

- ? シンボルは、すでにビットシンボルとして定義されています。ビットシンボルは再定義できません。
- ! シンボル名を変更してください。

Symbol is expected

- ? シンボルが不足しています。
- ! シンボルの数を確認してください。

Symbol is missing

- ? シンボルの記述がありません。
- ! シンボルを記述してください。

Symbol is multiple defined

- ? シンボルが二重定義です。マクロ名と他の名前が重複しています。
- ! 名前を変更してください。

Symbol 'XX' is not defined ( regarded as 0 )

- ? シンボルが未定義です。
- ! 未定義のシンボル名は使用できません。前方参照となるシンボル名は記述できません。シンボル名を確認してください。

Symbol name is missing

- ? 'EQU'、'BTEQU'で定義されるシンボル名の記述がありません。
- ! シンボル名を記述してください。

Syntax error in expression

- ? 式の記述に間違いがあります。
- ! 式の記述方法を確認して、記述し直してください。

Syntax error in indexed addressing expression

- ? 間接アドレッシングモードの記述に間違いがあります。
- ! アドレッシングモードを確認して、記述し直してください

Temporary label is undefined

- ? テンポラリラベルが未定義です。
- ! テンポラリラベルの定義を行ってください。

The value is not constant

- ? 値がアセンブル時確定値ではありません。
- ! アセンブル時に確定するような、式、シンボル名又はラベル名を記述してください。

Too many formal parameter

- ? マクロの仮引数の定義数が多すぎます。
- ! マクロの仮引数の数を80以下にしてください。

Too many macro local label definition

- ? マクロローカルラベルの定義が多すぎます。
- ! マクロローカルラベル数を1ファイル内に65535個以下にしてください。

Too many macro nesting

- ? マクロのネスティングが多すぎます。
- ! マクロのネスティングレベルを65535レベル以下にしてください。ソース記述を確認してください。

Too many nesting level of condition assemble

- ? 条件アセンブルのネスティングが多すぎます。
- ! 条件アセンブルの記述を確認してください。

Too many operand

- ? オペランドが余分にあります。
- ! オペランドの記述内容を確認してください。

Too many operand data

- ? オペランドのデータが多すぎます。
- ! オペランドに記述されているデータ数が、一行に記述できる範囲を超えています。命令を複数に分けて記述してください。

Too many source files

- ? 指定したファイルの数が多すぎます。
- ! ファイル数を80以下にしてください。複数回にわけてアセンブルを実行してください。

Too many structured label definition

- ? 構造化記述命令に対応するラベル数が多すぎます。
- ! ソースファイルを分割してアセンブルしてください。

Too many temporary label

- ? テンポラリラベルの個数が多すぎます。
- ! テンポラリラベルをラベル名に置き換えて記述してください。

Undefined symbol exist

- ? 未定義のシンボルがあります。
- ! シンボルを定義してください。

Value error

- ? 値が不正です。
- ! 値を記述し直してください。

Value is out of range

- ? 値が範囲外です。
- ! レジスタなどのビット長に合った値を記述してください。

## 5.5 as79ワーニングメッセージ

'`.ALIGN`' with not '`ALIGN`' specified relocatable section

? `ALIGN`指定がないセクション内に指示命令"`.ALIGN`"が記述されています。

! 指示命令"`.ALIGN`"の記述位置を確認してください。指示命令"`.ALIGN`"を記述するセクションのセクション定義行に`ALIGN`指定を記述してください。

'`.END`' statement is in include file

? インクルードファイルに `.END` 記述があります。

! インクルードファイル内には、`.END`は記述できません。記述を削除してください。`.END`を無視して処理します。

Actual macro parameters are not enough

? マクロ実引数の数がマクロ仮引数の数より少なくなっています。

! 該当する実引数のない仮引数は無効となります。

Destination address may be changed

? 分岐命令が最適選択されるため、分岐先が期待するものと異なる位置になる可能性があります。

! 分岐命令のオペランドを記述し直してください。

Addressing mode warning

? 異なった複数のアドレッシングモードが指定されています。

! 一つのアドレッシングモードを指定してください。

Floating point value is out of range

? 浮動小数点数が範囲外です。

! 浮動小数点数の記述を確認してください。範囲を超えた分は考慮しません。

I flag is specified.

? `SEP`命令でIフラグを操作しています。

! `SEI`命令で操作するように変更してください。

Location counter exceed '`0FFFFFFH`'

? ロケーションカウンタが`0FFFFFFH`を越えています。

! ロケーションカウンタが`0FFFFFFH`を越えないようにしてください。

Mismatch data size

? 展開した`m/x`フラグ依存命令と、現在のデータ又はインデックスサイズが異なります。

! データ又はインデックスサイズを合わせてください。

Not CASE values for SWITCH statement

? `SWITCH`文のなかに`CASE`文が記述されていません。

! プログラムを確認してください。

Source line exceeds 512 characters

? 1行の文字数が512文字を越えています。越えた部分は無視します。

! ソースファイルを確認してください。

String 'XX' is too long

- ? 文字列が長すぎます。
- ! 文字列を短くしてください。

Symbol 'XX' is not defined ( regard as 0 )

- ? 未定義シンボルが使用されています。0として処理します。
- ! シンボルを定義してください。

Too many actual macro parameters

- ? マクロ実引数の数が多すぎます。
- ! 余分な実引数は無視されます。

Unnecossary ':' is found

- ? マクロ名の後ろにコロンが付けられています。
- ! マクロ名の後ろのコロンを削除してください。
- ! コマンドオプション'- I'により、コロンを無視することができます。

Unnecossary BREAK is found

- ? 無効なBREAK文が記述されています。
- ! BREAK文を削除してください。

Value warning

- ? 値が範囲外です。
- ! レジスタ等のビット長に合った値を記述してください。

## 第6章 In79の操作方法

In79の機能を利用するための操作方法について説明します。In79の基本機能は、一つ以上のリロケータブルモジュールファイルから一つのアブソリュートモジュールファイルを生成します。

### 6.1 In79コマンドパラメータ

In79のコマンドパラメータ一覧を次に示します。

パラメータ名	機能
ファイル名	In79の処理対象となるリロケータブルモジュールファイル名
-.	メッセージを画面に出力しない。
-ABLX	アドレッシングモードABL,Xを評価する。
-C	特定のニーモニックがバンク境界にある場合にワーニング出力。
-E	アブソリュートモジュールファイルの開始アドレスを指定。
-G	ソースデバッグ情報をアブソリュートモジュールファイルに出力する。
-L	参照するライブラリファイルを指定する。
-LD	ライブラリファイルを検索するディレクトリを指定する。
-M	マップファイルを生成する。
-MS	マップファイルにシンボル情報を出力する。
-MSL	16文字を越えるシンボルをそのままマップファイルに出力します。
-NOSTOP	発生したエラー全てのメッセージを画面に出力する。
-O	生成するアブソリュートモジュールファイル名を指定する。
-ORDER	セクションの配置順序及び先頭アドレスを指定する。
-T	エラータグファイルを生成する。
-V	In79のバージョン番号を表示する。
@ファイル名	ファイル名で指定されたコマンドファイルを参照する。

## 6.2 In79コマンドパラメータの指定規則

In79のコマンドパラメータは次の規則に従って指定してください。

### コマンドパラメータの指定順序

- ・リロケータブルモジュールファイル名とコマンドオプションの指定順序は任意です。

```
>ln79 [option ...] file ...
>ln79 file ... [option ...]
```

file=リロケータブルモジュールファイル名

option=コマンドオプション

### リロケータブルモジュールファイル名 (必須)

- ・必ず、一つ以上のリロケータブルファイルを指定してください。
- ・ファイル名にはパスが指定できます。
- ・複数のリロケータブルファイルを指定する場合は、ファイル名の間、必ずスペースかタブを挿入してください。

### アブソリュートモジュールファイル名

- ・通常In79は、リロケータブルモジュールファイル名のうち一番目に指定されたファイル名をアブソリュートモジュールファイル名として生成します。
- ・アブソリュートモジュールファイル名を指定する場合は、コマンドオプション(-O)で指定してください。

### ライブラリファイル名

- ・参照するライブラリファイルを指定する場合は、コマンドオプション(-L)で指定してください。ファイル名には、パスが指定できます。
- ・ライブラリファイルは、環境変数(LIB79)が設定されていれば、そのディレクトリから検索し、該当するファイルが無い場合は、カレントディレクトリを検索します。または、コマンドオプション(-LD)で指定されたディレクトリから検索し、該当するファイルが無い場合は、カレントディレクトリを検索します。

### コマンドオプション

- ・コマンドオプションを指定する場合は、コマンドオプションとその他の指定の間には、必ずスペースかタブを挿入してください。

### アドレス指定

- ・In79は、セクション単位で絶対アドレスを決定し、アブソリュートモジュールファイルを生成します。
- ・In79を起動する際に、コマンド行からセクションの開始アドレスを指定することができます。
- ・アドレス値を指定する際には、16進数で指定してください。なお、数値の先頭がアルファベット文字になる場合は、先頭に0を付けて指定してください。

例)

```
7fff
64
0a57
```

## 6.3 コマンドファイル

ln79は、コマンドパラメータをファイルに記述し、そのファイルを読み込んでプログラムを実行できます。

### コマンドファイルの指定方法

- ・ コマンドファイル名の先頭に@を付けて指定してください。

例)

```
>ln79 @cmdfile
```

- ・ コマンドファイル名には、ディレクトリパスを指定できます。
- ・ 指定したディレクトリパスにファイルが存在しない場合は、エラーとなります。

### コマンドファイルの記述規則

ln79,lb79及びxrf79で処理可能なコマンドファイルの記述規則を説明します。

- ・ コマンドファイル内に、コマンドファイル自身の名前は記述できません。
- ・ コマンドファイルにはコマンドパラメータを複数行にわたって記述できます。
- ・ コマンドファイルに記述する行の行頭及び行末には、"," (カンマ) の記述はできません。
- ・ セクション配置指定を複数行にわたって記述したい場合は、改行した後で "-ORDER" オプションを行の先頭に記述してください。
- ・ ファイルの一行に記述可能な文字数は、255文字までです。255文字を越えた場合はエラーとなります。
- ・ コマンドファイルにはコメントを記述することができます。コメントを記述するには、コメントの先頭に"# "を記述してください。#以降改行文字までをコメントとして処理します。

### コマンドファイルの記述例

```
sample1 sample2
sample3
-ORDER ram=80
-ORDER prog,sub,data
-M
```

## 6.4 ln79コマンドオプション

以降に、コマンドオプションの指定方法を説明します。



-.

---

## 画面へのメッセージ出力を停止

---

### 機能

- ・ In79が処理を行う際のメッセージを出力しません。
- ・ エラーメッセージ及びワーニングメッセージは出力されます。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>ln79 -. smaple1 sample2
```

## -ABLX

---

### アドレッシングモードABL,Xの評価

---

#### 機能

- ・ アドレッシングモードABL,X (アブソリュートロングインデクストX) を指定されている命令が、バンク境界に配置された場合にワーニング"`ABL,X' addressing exists in the bank boundary value`"を出力します。

#### 記述規則

- ・ コマンド行の任意の位置で指定できます。

#### 記述例

```
>ln79 sample1 sample2 -ABLX
```

## -C

## 特定ニーモニックに対するワーニング出力

### 機能

- ・次に示す分岐命令が各バンクの最上位番地または、バンクにまたがって配置される場合に、ワーニングを出力します。

命令	アドレッシングモード	バイト数	命令コード
RTS	インプライド	1	84H
JMP	アブソリュート	3	9CH
	アブソリュート・インダイレクト	3	315CH
	アブソリュート・インデクストX・インダイレクト	3	BCH
	アブソリュート・インダイレクトロング	3	315DH
JSR	アブソリュート	3	9DH
	アブソリュート・インデクストX・インダイレクト	3	BDH

- ・該当するニーモニックが存在しない場合はワーニングは出力されません。
- ・プログラムで指定したアドレスには分岐せず、次のバンク内のアドレスに分岐します。

### 記述規則

- ・コマンド行の任意の位置で指定できます。

### 記述例

```
>ln79 sample1 sample2 -C
>ln79 -C sample1 sample2
```

## -E

---

### 実行開始アドレス指定

---

#### 機能

- ・ エントリーアドレスを設定します。エントリーアドレスは、デバッガにスタートアドレスを示すためのアドレスです。
- ・ アドレス値の指定には数値又はラベル名が記述できます。ただし、ローカルラベル名は指定できません。

#### 記述規則

- ・ -E (数値又はラベル名) のように入力してください。
- ・ 本オプションと数値又はラベル名の間には、必ずスペースを入力してください。
- ・ 数値は必ず16進数で指定してください。
- ・ 数値の先頭が英文字 (a,b,c,d,e,f) になる場合は、必ず先頭に0を記述してください。
- ・ コマンド行の任意の位置に記述できます。

#### 記述例

```
>ln79 sample1 sample2 -E num
```

"sample1.x79"のエントリーアドレスに、グローバルラベル"num"が持っているアドレス値を指定

```
>ln79 sample1 sample2 -E 0f0000
```

"sample1.x79"のエントリーアドレスに、f0000を指定

## -G

---

### ソースデバッグ情報出力

---

#### 機能

- ・ C言語やマクロ記述のソース行情報をアブソリュートモジュールファイルに出力します。
- ・ 本オプションを指定しないで生成したアブソリュートモジュールファイルでは、ソース行レベルでのデバッグはできません。

#### 注意事項

---

as79実行時に行情報の出力停止オプション (-N) を指定して生成したリロケータブルモジュールファイルをリンクした場合は、本オプション(-G)を指定してもソース行レベルでのデバッグはできません。

---

- ・ ソースデバッグ情報をアブソリュートモジュールファイルに出力します。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>ln79 -G sample1 sample2
```

## ライブラリファイル名指定

### 機能

- ・リンク実行時に参照するライブラリファイル名を指定します。
- ・ln79は、指定したライブラリファイル内から、グローバルシンボル情報を読み込んで、必要なりロケータブルモジュールをリンクします。

### 記述規則

- ・-L (ライブラリファイル名) のように入力してください。
- ・本オプションとファイル名の間には、必ずスペースを入力してください。
- ・コマンド行の任意の位置に指定できます。
- ・ライブラリファイル名にはパスが指定できます。
- ・ライブラリファイルは複数個指定できます。ライブラリファイルを複数指定する場合は、ファイル名をカンマで区切って指定してください。このとき、カンマの前後にスペース又はタブは記述できません。

### 注意事項

複数のライブラリファイルが指定されたとき、リンクは指定された順にライブラリファイルを参照します。したがって、次の条件を満たす場合にシンボル未定義エラーが発生します。

- (1)リロケータブルファイル(sample.r79)がライブラリファイル(A.LIB)に登録されているグローバルシンボルを参照している。
- (2)(1)でリンク対象となったライブラリファイル(A.LIB)内のリロケータブルモジュールが別のライブラリファイル(B.LIB)に登録されているグローバルシンボルを参照している。
- (3)-Lオプションで、上記(1)のライブラリファイル(A.LIB)よりも先に(2)の別のライブラリファイル(B.LIB)が指定されている(例1)。

例1 :

```
>ln79 sample.r79 -L B.LIB A.LIB
```

ライブラリファイルを次のように指定することでシンボル未定義エラーは発生しません(例2)。

例2 :

```
>ln79 sample.r79 -L A.LIB B.LIB
```

### 記述例

```
>ln79 sample1 sample2 -L lib1
```

カレントディレクトリ又は環境変数(LIB79)で指定されているディレクトリ内のlib1.libファイルを必要に応じて参照します。

```
>ln79 sample1 sample2 -L work¥lib1
```

カレントディレクトリまたは環境変数(LIB79)で指定されているディレクトリの下workディレクトリ内のlib1.libファイルを必要に応じて参照します。

```
>ln79 sample1 sample2 -L lib1,lib2
```

カレントディレクトリ又は環境変数(LIB79)で指定されているディレクトリのlib1.lib及びlib2.libファイルを必要に応じて参照します。

## -LD

---

### ライブラリファイル検索ディレクトリ指定

---

#### 機能

- ・ ライブラリファイルを参照するディレクトリ名を指定します。
- ・ 本オプションを指定した場合も、ライブラリファイル名は指定してください。
- ・ 本オプションで指定したディレクトリ名は、次に本オプションで指定し直すまで有効です。
- ・ ライブラリファイル名にパスを指定した場合は、本オプションで指定したディレクトリに、ライブラリファイルパスを連結したディレクトリのライブラリファイルが処理対象となります。

#### 記述規則

- ・ -LD (ディレクトリ名) のように入力してください。
- ・ 本オプションとディレクトリ名の間には、必ずスペースを入力してください。
- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>ln79 sample1 sample2 -LD ¥work¥lib -L lib1
```

¥work¥lib¥lib1.libファイルを参照

```
>ln79 sample1 sample2 -LD ¥work¥lib -L lib1 -LD ¥work¥tmp -L lib2
```

¥work¥lib¥lib1.lib,¥work¥tmp¥lib2.libファイルを参照

```
>ln79 sample1 -LD ¥work -L lib¥lib1
```

¥work¥lib¥lib1.libファイルを参照

-M

---

## マップファイル生成

---

### 機能

- ・ アドレスマッピング情報を格納したマップファイルを生成します。
- ・ 生成するマップファイルは、アブソリュートモジュールファイルの拡張子を".map"に変更したファイル名になります。

### 記述規則

- ・ コマンド行の任意の位置に指定できます。

### 記述例

```
>ln79 -M sample1 sample2  
sample1.x79とsample1.mapファイルを生成
```



## -MS/-MSL

---

### マップファイルにシンボル情報出力

---

#### 機能

- ・ アドレスマッピング情報及びシンボル情報を格納したマップファイルを生成します。
- ・ -MSを指定した場合、16文字を越えるシンボルは16文字までしかマップファイルに出力されません。
- ・ -MSLを指定した場合、16文字を越えるシンボルをそのまま出力します。
- ・ 生成するマップファイルは、アブソリュートモジュールファイルの拡張子を".map"に変更したファイル名になります。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>ln79 sample1 sample2 -MS  
sample1.x79とsample1.mapファイルを生成
```

## -NOSTOP

---

### 全エラー出力指定

---

#### 機能

- ・ 発生したリンクエラーを全て画面に出力します。
- ・ 本指示命令を指定しない場合は、最大20個までのエラーを画面に出力します。

#### 記述規則

- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>ln79 sample1 sample2 -NOSTOP
```

## -O

# アブソリュートモジュールファイル名指定

### 機能

- ln79が生成するアブソリュートモジュールファイル名を任意の名前に設定できます。
- 本オプションで、アブソリュートモジュールファイル名を指定しない場合は、コマンド行で一番目に指定されている、リロケータブルモジュールファイル名の拡張子を".x79"としたものをアブソリュートモジュールファイル名とします。

### 記述規則

- -O (ファイル名) のように入力してください。
- オプションとファイル名の間には、必ずスペースを入力してください。
- ファイル名の拡張子は省略できます。省略した場合の拡張子は、".x79"となります。
- ファイル名には、パスを指定できます。

### 記述例

```
>ln79 sample1 sample2 -O abssmp  
"abssmp.x79"ファイルを生成
```

```
>ln79 -O ¥work¥absfile¥abssmp sample1 sample2  
ディレクトリ"¥work¥absfile"に"abssmp.x79"を生成
```

## -ORDER

---

### セクションアドレス/配置順序指定

---

#### 機能

- ・ セクションの配置順序と、セクションの開始アドレスを指定します。

#### 注意事項

---

絶対セクションに対して、開始アドレスを指定した場合はエラーとなります。

---

- ・ 開始アドレスを指定しない場合は、0 からアドレスを配置します。
- ・ 同一名のセクションが指定したりロケータブルファイルに存在するときは、指定したファイル順にセクションを配置します。このとき、相対属性を持つセクションの後に絶対属性を持つものが配置されるとエラーとなります。

#### 記述規則

- ・ -ORDER (セクション名), (セクション名) 又は -ORDER (セクション名) = (スタートアドレス) のように入力してください。
- ・ オプションとセクション名の間には、必ずスペースを入力してください。
- ・ セクション名とセクション名又はアドレス値とセクション名は、カンマで区切って指定してください。このとき、カンマの前後にスペース又はタブは入力できません。
- ・ コマンド行の任意の位置に指定できます。

#### 記述例

```
>ln79 sample1 sample2 -ORDER main,sub,dat
```

main,sub,datの順にアドレス0Hからセクションを配置します。

```
>ln79 sample1 sample2 -ORDER main=0f0000,sub,dat
```

main,sub,datの順にアドレス0f000Hからセクションを配置します。

-T

---

## リンクエラータグファイル生成

---

### 機能

- ・リンクエラーが発生した場合、リンクエラータグファイルを生成します。
- ・エディタのタグジャンプ機能を利用可能なフォーマットでファイルを出力します。
- ・本オプションを指定してもエラーが発生しなければ、ファイルは生成しません。
- ・エラータグファイル名は、先頭に指定したリロケータブルモジュールファイル名の拡張子を".ltg"に変更したものになります。
- ・リンクエラータグファイルのエラー情報は、アセンブリソース行番号で出力されます。

### 記述規則

- ・コマンド行の任意の位置に指定できます。

### 記述例

```
>ln79 sample1 sample2 -T
```

エラーが発生した場合、"sample1.ltg"ファイルを生成

-V

---

## バージョン表示

---

### 機能

- ・ ln79のバージョン番号を表示します。

### 注意事項

---

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・ 本オプションのみを記述してください。

### 記述例

```
>ln79 -V
```

@

---

## コマンドファイル参照

---

### 機能

- ・ 指定したファイルの内容をコマンドパラメータとしてIn79を起動します。

### 記述規則

- ・ @ (ファイル名) のように入力してください。
- ・ 本オプションとファイル名の間にはスペースを記述しないでください。
- ・ 他のパラメータはコマンド行に記述できません。

### 記述例

```
>ln79 @cmdfile
```

## 6.5 In79エラーメッセージ

- '-order' section 'XX' is multiple defined  
? -order で指定されたセクション名が二重定義です。  
! セクションは、一度だけ定義してください。
- '-order' section 'XX' is not found  
? -order で指定されたセクションが見つかりません。  
! セクション名を確認して実行し直してください。
- 'CODE' section 'section-1' is overlapped on the 'section-2'  
? CODEのセクション'section-1'と'section-2'がオーバーラップしています。  
! セクションがオーバーラップしないように配置し直してください。
- 'ROMDATA' section 'section-1' is overlapped on the 'section-2'  
? ROMDATAタイプのセクション'section-1'と'section-2'がオーバーラップしています。  
! セクションがオーバーラップしないように配置し直してください。
- 'SWITCH' statement is crossing bank.  
? テーブルジャンプコードがバンク境界をまたがっています。  
! nc79の-fswitch\_table[-fST]オプションをはずしてください。
- 'XX' is written after the same name of relocatable section  
? 相対属性セクションの後に同名の絶対属性セクション'XX'を連結しています。  
! 絶対属性の後に相対属性を配置してください。
- 'XX' is multiple defined  
? シンボルが二重定義されています。  
! 外部シンボル名を確認してください。
- 'XX' value is undefined  
? シンボル'XX'の値が未定義です。  
! 値を0として処理します。シンボル値を確認してください。
- Absolute section 'XX' is relocated  
? 絶対セクション'XX'を再配置しています。  
! セクションの配置指定をし直してください。
- Address is overlapped in 'CODE' section 'XX'  
? CODEタイプのセクション'XX'内でアドレスがオーバーラップしています。  
! セクションがオーバーラップしないように配置し直してください。
- Address is overlapped in 'ROMDATA' section 'XX'  
? ROMDATAタイプのセクション'XX'内でアドレスがオーバーラップしています。  
! アドレスがオーバーラップしないように配置し直してください。
- Can't close file 'XX'  
? 'XX'ファイルがクローズできません。  
! ディレクトリ情報を確認してください。



Can't close temporary file

- ? テンポラリファイルがクローズできません。
- ! ディスクの残り容量を確認してください。

Can't create file 'XX'

- ? 'XX'ファイルが生成できません。
- ! ディレクトリ情報を確認してください。

Can't create temporary file

- ? テンポラリファイルが生成できません。
- ! ディレクトリが書き込み禁止になっていないか確認してください。

Can't open file 'XX'

- ? 'XX'ファイルがオープンできません。
- ! ファイル名を確認してください。

Can't open temporary file

- ? テンポラリファイルがオープンできません。
- ! ディレクトリ情報を確認してください。

Can't remove file 'XX'

- ? 'XX'ファイルが削除できません。
- ! ファイル名等を確認してください。

Can't remove temporary file

- ? テンポラリファイルが削除できません。
- ! ファイル名等を確認してください。

Can't registered symbol in the list

- ? シンボルをリストに登録できません。
- ! 本エラーが発生した場合は、お手数ですが「ツールサポート窓口」までご連絡いただきますようお願いいたします。

Command-file line characters exceed 255

- ? コマンドファイルの1行の文字数が255文字を越えています。
- ! コマンドファイルの内容を確認してください。

Command line is too long

- ? コマンド行の文字数が多すぎます。
- ! コマンドファイルを作成してください。

Illegal file extension '.XX' is used

- ? ファイルの拡張子'.XX'に間違いがあります。
- ! ファイルの拡張子を指定し直してください。

Illegal format 'XX'

- ? 'XX'ファイルのフォーマットに間違いがあります。
- ! リロケータブルファイルがas79で生成されたものであることを確認してください。

## In79の操作方法

Illegal format 'XX' :expression error occurred

? 'XX'ファイルのフォーマットに間違いがあります。

! リロケータブルファイルがas79で生成されたものであることを確認してください。

Illegal format 'XX' :it's not library file

? 'XX'ファイルのフォーマットに間違いがあります。ライブラリファイルではありません。

! ライブラリファイルがlb79で生成されたものであることを確認してください。

Illegal format 'XX' :it's not relocatable file

? 'XX'ファイルのフォーマットに間違いがあります。リロケータブルファイルではありません。

! リロケータブルファイルがas79で生成されたものであることを確認してください。

Invalid option 'XX' is used

? 無効なオプション'XX'を使用しています。

! オプションを指定し直してください。

MCU information mismatch in file 'XX'

? 'XX'ファイルのMCU情報が一致していません。

! リロケータブルファイルがas79で生成されたものであることを確認してください。

No input files specified

? 入力ファイルが指定されていません。

! ファイル名を指定してください。

Not enough memory

? メモリが足りません。

! メモリを増設してください。

Option 'XX' is not appropriate

? オプションの使用方法が間違っています。

! オプションの使用方法を確認して、指定し直してください。

Option parameter address exceed 0FFFFFFFH

? オプションで指定したアドレスが 0FFFFFFFHを越えています。

! コマンドを入力し直してください。

Symbol type of floating point is not supported

? シンボルタイプの浮動小数点はサポートしていません。

! 本エラーが発生した場合は、お手数ですが「ツールサポート窓口」までご連絡いただきますようお願いいたします。

Zero division exists in the expression

? リロケーションデータの演算に0除算があります。

! 式を記述し直してください。

## 6.6 In79ワーニングメッセージ

'-e' option parameter 'XX' is undefined

? -e で指定されたシンボル'XX'が未定義です。

! ソースプログラム内で、'XX'を定義してください。値を0として処理します。

'ABL,X' addressing exists in the bank boundary value

? アドレッシングモード ABL,X (アブソリュートロング・インデクストX)がバンク境界に配置されています。

! NOP命令等でずらすなど、アドレッシングモード ABL,X がバンク境界値に配置されないようにして下さい。

'CODE' section 'section-1' is overlapped on the 'section-2'.

? CODEセクションの'section-1'が'section-2'にオーバーラップしています。オーバーラップして配置しました。

! セクションがオーバーラップ可能か確認してください。

'DATA' section 'section-1' is overlapped on the 'section-2'

? DATAセクションの'section-1'が'section-2'にオーバーラップしています。オーバーラップして配置しました。

! セクションがオーバーラップ可能か確認してください。

'ROMDATA' section 'section-1' is overlapped on the 'section-2'.

? ROMDATAセクションの'section-1'が'section-2'にオーバーラップしています。オーバーラップして配置しました。

! セクションがオーバーラップ可能か確認してください。

'XX' value exceed 0FFFFFFFH

? ラベル'label'の値が0FFFFFFFH を越えています。

! セクションの配置アドレスを確認してください。ラベルの定義を確認してください。

'XX' data exceed 0FFFFFFFH

? セクションのデータが0FFFFFFFH番地を越えています。

! セクションの配置アドレスを確認してください。

16-bits signed value is out of range -32768 -- 32767 address='XX'

? リロケーションデータの演算結果が -32768 から +32767 の範囲を越えています。

! オーバーフロー分は、無視した結果を扱います。

16-bits unsigned value is out of range 0 -- 65535 address='XX'

? リロケーションデータの演算結果が 0 から 65535 の範囲を越えています。

! オーバーフロー分は、無視した結果を扱います。

16-bits value is out of range -32768 -- 65535 address='XX'

? リロケーションデータの演算結果が -32768 から 65535 の範囲を越えています。

! オーバーフロー分は、無視した結果を扱います。

## In79の操作方法

- 24-bits signed value is out of range -8388608 -- 8388607 address='XX'  
? リロケーションデータの演算結果が -8388608 から 8388607 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 24-bits unsigned value is out of range 0 -- 16777215 address='XX'  
? リロケーションデータの演算結果が 0 から 16777215 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 24-bits value is out of range -8388608 -- 16777215 address='XX'  
? リロケーションデータの演算結果が -8388608 から 16777215 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 4-bits signed value is out of range -8 -- 7 address='XX'  
? リロケーションデータの演算結果が -8 から 7 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 6-bits unsigned value is out of range 0 -- 63 address='XX'  
? リロケーションデータの演算結果が 0 から 63 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 8-bits signed value is out of range -128 -- 127 address='XX'  
? リロケーションデータの演算結果が -128 から 127 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 8-bits unsigned value is out of range 0 -- 255 address='XX'  
? リロケーションデータの演算結果が 0 から 255 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- 8-bits value is out of range -128 -- 255 address='XX'  
? リロケーションデータの演算結果が -128 から 255 の範囲を越えています。  
! オーバーフロー分は、無視した結果を扱います。
- Absolute-section is written after the absolute-section 'XX'  
? 絶対属性セクション'XX'の後に同名の絶対属性を連結しています。アドレスが不連続に配置される可能性があります。  
! ソース記述を確認してください。
- Absolute-section is written before the absolute-section 'XX'  
? 絶対属性セクション'XX'の前に絶対属性を連結しています。  
! 連結を実行します。ソースプログラムのアドレス指定を確認してください。
- Address information mismatch in file 'file'  
? リロケータブルファイル'file'ファイルのアドレス情報が一致していません。  
! リロケータブルファイルがas79で生成されたものであることを確認してください。

Address is overlapped in the same 'DATA' section 'XX'

? 同一名のDATAセクション'XX'内でアドレスがオーバーラップしています。オーバーラップして配置しました。

! アドレスがオーバーラップ可能か確認してください。

JMP instruction exists at end of bank

? JMP命令がバンク境界にあります。

! 命令がバンク境界に配置されないように記述してください。

JSR instruction exists at end of bank

? JSR命令がバンク境界にあります。

! 命令がバンク境界に配置されないように記述してください。

Object format version mismatch in file 'XX'

? リロケータブルファイル又はライブラリファイルのバージョン情報が一致していません。

! リロケータブルファイル又はライブラリファイルがAS79プログラムによって生成されたものであることを確認してください。ファイルを生成し直してください。本エラーが発生した場合は、お手数ですが「ツールサポート窓口」までご連絡いただきますようお願いいたします。

RTS instruction exists at end of bank

? RTS命令がバンク境界にあります。

! 命令がバンク境界に配置されないように記述してください。

Section type mismatch 'XX'

? 同一名セクション'XX'のセクションタイプが異なります。

! ソースのセクションタイプを確認してください。

## 第7章 lmc79の操作方法

lmc79の操作方法を説明します。lmc79の主な機能は、アブソリュートモジュールファイルから、モトローラSフォーマットの機械語ファイルを生成します。

### 7.1 lmc79コマンドパラメータ

lmc79のコマンドパラメータ一覧を次に示します。

パラメータ名	機能
ファイル名	lmc79の処理対象となるアブソリュートモジュールファイル名。
-.	メッセージを画面に出力しない。
-A	出力データのアドレス範囲を指定する。
-E	実行開始アドレスを設定する。
-F	空き領域データを設定する。
-H	インテルHEXフォーマットファイルを生成する。
-L	データレコード長を選択する。
-O	生成するファイル名を指定する。
-V	lmc79のバージョン番号を表示する。

### 7.2 lmc79コマンドパラメータの指定規則

コマンドパラメータは必ず次の順序で指定してください。

1 コマンドオプション

2 アブソリュートモジュールファイル名 (必須)

```
>lmc79 [option ...] file
```

option=コマンドオプション

file=アブソリュートモジュールファイル名

#### アブソリュートモジュールファイル名 (必須)

- ・ lmc79が生成したアブソリュートモジュールファイルを指定してください。
- ・ アブソリュートモジュールファイル名は一つだけ指定してください。
- ・ ファイルの拡張子(.x79)は省略できます。
- ・ 拡張子が".x79"以外のファイル名は指定できません。

#### コマンドオプション

- ・ コマンドオプションは必要に応じて指定してください。
- ・ 複数のコマンドオプションを指定できます。ただし、-Eと-Hは同時に指定できません。
- ・ 複数のコマンドオプションを指定する場合のコマンドオプションの指定順序は任意です。

## 7.3 lmc79コマンドオプション

lmc79のコマンドオプションの指定方法について詳しく説明します。

-.

---

## 画面へのメッセージ出力を停止

---

### 機能

- ・ lmc79が処理を行う際のメッセージを出力しません。
- ・ エラーメッセージ及びワーニングメッセージは出力されます。

### 記述規則

- ・ 必ず、ファイル名より前に本オプションを指定してください。

### 記述例

```
>lmc79 -. debug
```



## -A

---

## 出力データのアドレス範囲指定

---

### 機能

- ・ 生成するファイルに出力する機械語データのアドレス範囲を指定します。
- ・ 本オプションは以下の2種類の指定が可能です。
  1. 出力の開始アドレスと終了アドレスを指定
  2. 出力の開始アドレスのみ指定

### 記述規則

- ・ “-A (開始アドレス値:終了アドレス値)” または “-A (開始アドレス値)” のように指定してください。
- ・ 本オプションと開始アドレス値の間には、必ずスペースを指定してください。
- ・ アドレス値は、必ず16進数値で指定してください。
- ・ ファイル名を指定する前に本オプションを指定してください。
- ・ 出力の開始アドレス値のみ指定した場合、アブソリュートモジュールファイルに登録されているデータの最大アドレスが終了アドレス値となります。
- ・ 開始アドレス値が終了アドレス値よりも大きい場合はエラーとなります。
- ・ 指定されたアドレス範囲内でデータが存在しないアドレスに対しては、空き領域データ設定オプション'-F'が指定されている場合は指定されたデータを出力し、指定されていない場合は何も出力されません。

### 記述例

```
>lmc79 -A 1000:11FF sample
```

アドレス範囲指定の開始アドレス値を1000H、終了アドレス値を11FFHとします。

```
>lmc79 -A 1000 sample
```

アドレス範囲指定の開始アドレス値を1000Hとします。

## -E

---

### 実行開始アドレス指定

---

#### 機能

- ・ 実行開始アドレスを設定します。
- ・ 設定したアドレスをモトローラSフォーマットファイルに出力します。

#### 記述規則

- ・ -E (アドレス値) のように入力してください。
- ・ 本オプションと数値の間には、必ずスペースを入力してください。
- ・ アドレス値は、必ず16進数で設定してください。
- ・ アドレス値の先頭の値が、'a' ~ 'f'の場合は先頭に必ず'0'を付けてください。

#### 注意事項

---

本オプションは、"-H"と同時に指定できません。

---

#### 記述例

```
>lmc79 -E 0f0000 debug  
>lmc79 -E 8000 debug
```

## -F

---

### 空き領域データの設定

---

#### 機能

- ・ 指定されたアブソリュートモジュールファイル内のデータ登録されていないアドレスに対して任意データを出力します。

#### 記述規則

- ・ “-F (データ)” のように指定してください。
- ・ 本オプションとデータの間には、必ずスペースを指定してください。
- ・ アドレス値は、必ず16進数値で指定してください。
- ・ ファイル名を指定する前に本オプションを指定してください。

#### 記述例

```
>lmc79 -F FF sample
```

空き領域アドレスに0FFHを出力します。

## -H

---

# インテルHEXファイル生成

---

### 機能

- ・ 生成する機械語ファイルのフォーマットを拡張インテルHEXフォーマットに設定します。

### 記述規則

- ・ ファイル名を指定する前に本オプションをしてしてください。
- ・ 本オプションは、"-E"オプションと同時に設定できません。

### 記述例

```
>lmc79 -H debug
```

-L

---

## データレコード長の選択

---

### 機能

- ・ モトローラSフォーマットのデータレコード長を32バイトに設定します。
- ・ インテルHEXフォーマットのデータレコード長を32バイトに設定します。

### 記述規則

- ・ ファイル名を指定する前に本オプションを指定してください。

### 記述例

```
>lmc79 -L debug
```

## -O

---

### 出力ファイル名指定

---

#### 機能

- ・ lmc79が生成する機械語ファイルのファイル名を指定します。
- ・ ファイル名にはパスの指定ができます。
- ・ ファイル名の拡張子を指定することができます。拡張子の指定を省略した場合、モトローラSフォーマットは".mot"、インテルHEXフォーマットは".hex"で生成されます。
- ・ 出力ファイルは、指定されたアブソリュートモジュールファイルと同じディレクトリに出力されます。

#### 記述規則

- ・ -O (ファイル名) のように指定してください。
- ・ 本オプションとファイル名の間には、必ずスペースを入力してください。
- ・ ファイル名を指定する前に本オプションを指定してください。

#### 記述例

```
>lmc79 -O test debug
```

"test.mot"ファイルを生成します。

```
>lmc79 -O tmp¥test debug
```

"tmp"ディレクトリに"test.mot"ファイルを生成します。

-V

---

## バージョン表示

---

### 機能

- ・ロードモジュールコンバータのバージョン番号を画面に表示します。

### 注意事項

---

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・本オプションのみを指定してください。

### 記述例

```
>lmc79 -V
```

## 7.4 lmc79エラーメッセージ

'-e' option is too long

- ? -e オプションの引数の並びが長すぎます。
- ! オプションの指定方法を確認して指定し直してください。

'XX' option multiple specified

- ? オプション 'XX' を複数指定しています。
- ! オプションの指定方法を確認して指定し直してください。

Address specified by '-e' option exceed 0FFFFFFFH

- ? -eオプションで指定したアドレスが0FFFFFFFHを越えました。
- ! アドレス値を指定し直してください。

Can't close file 'XX'

- ? 'XX'ファイルをクローズできません。
- ! ディレクトリの情報を確認してください。

Can't create file 'XX'

- ? 'XX'ファイルが作成できません。
- ! ディレクトリの情報を確認してください。

Can't open file 'XX'

- ? 'XX'ファイルがオープンできません。
- ! ファイル名を確認してください。

Illegal file format 'XX' is used

- ? ファイルのフォーマットが間違っています。
- ! ファイル名を確認してください。ファイルを生成し直してください。

Invalid option 'XX' is used

- ? 無効なオプション'XX'を指定しています。
- ! オプションを指定し直してください。

Not enough memory

- ? メモリが足りません。
- ! メモリを増設してください。

Option 'XX' is not appropriate

- ? オプションの使用方法が間違っています。
- ! オプションの指定方法を確認して指定し直してください。

Unknown file extension '.XX' is specified

- ? 指定したファイルの拡張子'.XX'に間違いがあります。
- ! ファイル名を確認してください。



## 7.5 lmc79ワーニングメッセージ

'XX' does not contain object data

- ? 指定したファイルにオブジェクトデータがありません。
- ! ファイル名を確認してください。

Address exceed 0FFFFFFFH

- ? アドレスが0FFFFFFFHを越えました。
- ! ソースの記述内容を確認してください。セクションの配置指定を確認してください。

Original HEX format for mitsubishi microcomputers is generated

- ? 三菱専用HEXファイルが生成されました。
- ! 三菱専用HEXファイルで問題ないか確認下さい。

## 第8章 lb79の操作方法

---

lb79の機能を使用するための操作方法を説明します。lb79の機能は、複数のリロケータブルモジュールファイルを一つのライブラリファイルとして管理します。

### 8.1 lb79コマンドパラメータ

lb79のコマンドパラメータ一覧を次に示します。

パラメータ名	機能
ファイル名	lb79の処理対象となるファイル名。
-.	メッセージを画面に出力しない。
-A	ライブラリファイルにモジュールを追加する。
-C	ライブラリファイルを新規生成する。
-D	モジュールを削除する。
-L	ライブラリリストファイルを生成する。
-R	モジュールを置き換える。
-U	モジュールを更新する。
-V	lb79のバージョン番号を表示する。
-X	モジュールをリロケータブルモジュールファイルとして抽出する。
@ファイル名	ファイル名で指定されたファイルを参照する。

## 8.2 lb79コマンドパラメータの指定規則

lb79のコマンドパラメータは必ず次の順序で指定してください。指定順序が間違っている場合には、正しくlb79の処理が行われません。

- 1 コマンドオプション
- 2 ライブラリファイル名
- 3 リロケータブルモジュールファイル名

```
>lb79 [option ...] LIB_file R79_file ...
```

option=コマンドオプション

LIB\_file=ライブラリファイル名

R79\_file=リロケータブルモジュールファイル名

### ライブラリファイル名

- ・ライブラリファイル名は必ず指定してください。
- ・ファイル名には、ディレクトリパスを指定できます。
- ・コマンド行では、拡張子(lib)は省略できます。

### リロケータブルモジュールファイル名 (リロケータブルモジュール名)

- ・リロケータブルモジュールファイル名は必ず指定してください。
- ・リロケータブルモジュールファイル名の拡張子は'.r79'です。コマンド行では、拡張子を省略できます。
- ・リロケータブルファイルは、複数指定できます。このとき、ファイル名とファイル名の間には、必ずスペースを入力してください。
- ・ファイル名にはディレクトリパスを指定できます。ディレクトリの指定がない場合は、カレントディレクトリにあるファイルを処理します。

### コマンドオプション

- ・コマンドオプションの大文字小文字は区別しません。
- ・コマンドオプションのうち、'-A','-C','-D','-L','-R','-U','-X'は、ライブラリアン実行時に必ず一つを指定してください。コマンド行の指定に、次に示すコマンドオプションが一つも指定していなかったり、二つ以上同時に指定がある場合は、エラーとなります。

## 8.3 lb79コマンドファイル

- ・入力パラメータを記述したコマンドファイル名を指定できます。
- ・コマンドファイルの記述方法はln79と同様です。ln79の操作方法を参照してください。

## 8.4 lb79コマンドオプション

以降に、コマンドオプションの指定規則を説明します。

-.

---

## 画面へのメッセージ出力を停止

---

### 機能

- ・ lb79が処理を行う際のメッセージを出力しません。
- ・ エラーメッセージ及びワーニングメッセージは出力されます。

### 記述規則

- ・ 本オプションのみ他のオプションとの組み合わせが可能です。
- ・ 他のオプションとの指定順序は任意です。

### 記述例

```
>lb79 -. -A new sample2
```

-A

---

## モジュール追加

---

### 機能

- ・ 既にあるライブラリファイルにリロケータブルモジュールを追加登録します。
- ・ 指定したライブラリファイルが存在しない場合は、新しく作成します。
- ・ 追加しようとしたリロケータブルモジュール名と同一のモジュールが登録されている場合は、エラーとなります。
- ・ 追加しようとしたリロケータブルモジュールファイル内に、既にライブラリファイルに登録されているモジュール内で同じグローバルシンボル名の定義が存在する場合はエラーとなります。

### 記述規則

- ・ -A (ライブラリファイル名) (リロケータブルモジュールファイル名) のように入力してください。
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。

### 記述例

```
>lb79 -A new.lib sample3.r79  
new.libファイルに、sample3を追加
```

## -C

---

### ライブラリファイル新規作成

---

#### 機能

- ・新しくライブラリファイルを生成します。

#### 注意事項

本コマンドオプションで指定したライブラリファイル名と同一のライブラリファイルが既に存在している場合は、古いライブラリファイルの内容は、新しく作成したライブラリファイルの内容に書き変わります。

---

#### 記述規則

- ・ -C (ライブラリファイル名) (リロケータブルモジュールファイル名) のように入力してください。
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。

#### 記述例

```
>lb79 -C new sample1 sample2
```

sample1とsample2を登録した、new.libファイルを新しく作成

## -D

---

### モジュール削除

---

#### 機能

- ・ ライブラリファイルに登録されているリロケータブルモジュールを削除します。
- ・ 削除したモジュールは元には戻りません。

#### 記述規則

- ・ -D (ライブラリファイル名) (リロケータブルモジュール名)
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュール名との間には、必ずスペースを入力してください。
- ・ 削除するリロケータブルモジュールを複数指定できます。このとき、モジュール名の間には、必ずスペースを入力してください。

#### 記述例

```
>lb79 -D new sample2
```

new.libというライブラリファイルに登録されている、sample2というリロケータブルモジュールを削除します。

---

-L

---

## ライブラリリストファイル生成

---

### 機能

- ・ 指定したライブラリファイルの情報を格納したライブラリリストファイルを生成します。生成するライブラリリストファイルのファイルの拡張子は、'.lls'です。
- ・ ライブラリファイル内の必要なモジュールの情報だけを格納したライブラリリストファイルを生成できます。
- ・ 同じ名前のライブラリリストファイルが既に存在している場合は、新しいライブラリリストファイルの内容が上書きされます。

### 記述規則

- ・ -L (ライブラリファイル名) [(リロケータブルモジュール名)]のように入力してください。
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- ・ リロケータブルモジュール名は複数指定できます。このとき、リロケータブルモジュール名の間には、必ずスペースを入力してください。

### 記述例

```
>lb79 -L new
```

new.libというライブラリファイルに登録されている全てのモジュールの情報をnew.llsというライブラリリストファイルに出力します。

```
>lb79 -L new sample1
```

new.libに登録されているsample1というモジュールの情報をnew.llsに出力します。

```
>lb79 -L new.lib sample1 sample3
```

new.libに登録されているsample1,sample3というモジュールの情報をnew.llsに出力します。



## -R

---

### モジュール置き換え

---

#### 機能

- ・ ライブラリファイルに登録されているリロケータブルモジュールの内容を指定したリロケータブルモジュールファイルの内容に更新します。更新されるモジュールは、指定したリロケータブルモジュールファイル名と同一の名前のモジュールです。

#### 記述規則

- ・ -R (ライブラリファイル名) (リロケータブルモジュールファイル名) のように指定してください。
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- ・ リロケータブルモジュールファイル名は複数指定できます。このとき、リロケータブルモジュールファイル名の間には、必ずスペースを入力してください。

#### 記述例

```
>lb79 -R new sample1
```

new.libファイルに登録されているsample1の内容を、同じ名前のsample1.r79ファイルの内容に置き換えます。

## -U

---

### モジュール更新

---

#### 機能

- ・ ライブラリファイルに登録されているリロケータブルモジュールの作成日付と、更新しようとするリロケータブルモジュールファイルの作成日付を比較し、リロケータブルモジュールファイルの日付が新しい場合にのみ、モジュールを更新します。

#### 記述規則

- ・ -U (ライブラリファイル名) (リロケータブルモジュールファイル名) のように入力してください。
- ・ 本オプションとライブラリファイル名の間及びライブラリファイル名とリロケータブルモジュールファイル名との間には、必ずスペースを入力してください。
- ・ リロケータブルモジュール名は複数指定できます。このとき、リロケータブルモジュール名の間には、必ずスペースを入力してください。

#### 記述例

```
>lb79 -U new sample1
```

new.libファイルに登録されているsample1の作成日が、同じ名前のsample1.r79ファイルの作成日より古い場合だけ、登録されているsample1の内容を、sample1.r79ファイルの内容に更新します。

-V

---

## バージョン表示

---

### 機能

- ・ lb79のバージョン番号を画面に出力します。

### 注意事項

---

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・ 本オプションのみを入力してください。

### 記述例

```
>lb79 -V
```

-X

---

## モジュール抽出

---

### 機能

- ・ ライブラリファイルに登録されているリロケータブルモジュールをリロケータブルモジュールファイルとして抽出します。
- ・ ライブラリファイルは変更されません。
- ・ 抽出したリロケータブルモジュールファイルの作成日時は、抽出した日時となります。
- ・ 抽出したリロケータブルモジュールファイル名と同一名のファイルが存在する場合、上書きします。

### 記述規則

- ・ 本オプションとライブラリファイル名の間には、必ずスペースを入力してください。

### 記述例

```
>lb79 -X new sample3
```

new.libファイルに登録されている、sample3から、sample3.r79というリロケータブルモジュールファイルを生成

@

---

## コマンドファイル参照

---

### 機能

- ・ 指定したファイルの内容をコマンドパラメータとしてlb79を起動します。

### 記述規則

- ・ @ (ファイル名) のように入力してください。
- ・ 本オプションとファイル名の間、スペース又はタブは記述できません。
- ・ 他のパラメータはコマンド行に記述できません。

### 記述例

```
>lb79 @cmdfile
```

## 8.5 lb79エラーメッセージ

'XX' is not library file

? 'XX'ファイルはライブラリファイルではありません。

! ファイル名を確認してください。ファイルがlb79で生成されたものであることを確認してください。

'XX' is not relocatable file

? 'XX'ファイルがリロケータブルファイルではありません。

! ファイル名を確認してください。ファイルがas79で生成されたものであることを確認してください。

'module' already registered in 'filename'

? モジュール'module'はライブラリ'filename'に登録済みです。

! ライブラリファイル名及びリロケータブルファイル名を確認してください。

'module' does not match with 'filename'

? モジュール名'module'とリロケータブルファイル名'filename'が異なります。モジュール名が変更されています。

! リロケータブルファイル名を確認してください。

'module' is multiple specified

? 同一のモジュール名'module'を複数指定しています。

! モジュール名を指定し直してください。

'module' is not registered in 'filename'

? モジュール'module'がライブラリファイル'filename'に登録されていません。指定された処理(モジュールの削除又は更新)はできません。

! モジュール名を確認してください。

'symbol' is multiple defined at 'module1' and 'module2' in 'filename'

? 同名外部定義シンボル'symbol'がライブラリ'filename'中の'module1'と'module2'に二重定義されています。

! リロケータブルファイル名を確認してください。

'symbol' is multiple defined in 'filename'

? シンボル'symbol'は'filename'ファイルに二重定義されています。

! 本エラーが発生した場合は、お手数ですが「ツールサポート窓口」までご連絡いただきますようお願いいたします。

'symbol' is multiple defined in 'module1' and 'module2'

? 同名外部定義シンボル'symbol'が'module1'と'module2'に二重定義されています。

! リロケータブルファイル名を確認してください。

'XX' and 'XX' are used

? 'XXx'オプションと'XX'オプションを同時に使用しています。

! オプションは、同時に指定できません。コマンドを入力し直してください。

Can't close file 'XX'

? 'XX'ファイルをクローズできません。  
! ディレクトリ情報を確認してください。

Can't close temporary file

? テンポラリファイルがクローズできません。  
! ディレクトリ情報を確認してください。

Can't create file 'XX'

? 'XX'ファイルが作成できません。  
! ディレクトリ情報を確認してください。

Can't create temporary file

? テンポラリファイルが生成できません。  
! ディレクトリ情報を確認してください。

Can't open file 'XX'

? 'XX'ファイルがオープンできません。  
! ファイル名を確認してください。

Can't open temporary file

? テンポラリファイルがオープンできません。  
! ディレクトリ情報を確認してください。

Can't write in file 'XX'

? 'XX'ファイルの書き込みができません。メモリが不足しています。  
! メモリを増設してください。

Command-file is included in itself

? コマンドファイル自身をインクルードしています。  
! コマンドファイルの記述内容を確認してください。

Command-file line characters exceed 255

? コマンドファイルの行が255文字を越えました。  
! コマンドファイルの内容を確認してください。

Illegal file format 'XX'

? 'XX'ファイルのフォーマットが間違っています。  
! ファイル名を確認してください。

Invalid option 'XX' is used

? 無効なオプション'XX'を指定しています。  
! オプションを指定し直してください。

No public symbol is in 'XX'

? ファイル'XX'に外部定義シンボルがありません。  
! リロケータブルファイルの内容を確認してください。

Not enough memory

- ? メモリが足りません。
- ! メモリを増設してください。

Symbol-name characters exceed 500

- ? シンボル名が500文字をこえました。
- ! ライブラリファイルを複数に分割してください。

Too many modules

- ? 登録モジュール数が多すぎます。
- ! ライブラリファイルを複数に分割してください。

Unknown file extension '.XX' is used

- ? ファイル拡張子'.XX'が間違っています。
- ! ファイル名を確認してください。

## 8.6 lb79ワーニングメッセージ

'XX' is not registered in library

- ? モジュール'XX'がライブラリに登録されていません。該当するモジュールは抽出しませんでした。
- ! モジュール名を確認してください。

'XX' is not registered in library, can't output list-file

- ? モジュール'XX'がライブラリに登録されていません。リストファイルに情報を出力しませんでした。
- ! モジュール名を確認してください。

'XX' was created in the current directory

- ? モジュール'XX'をカレントディレクトリに生成しました。
- ! 指定したディレクトリ名を確認してください。

Can't replace, 'XX' is older than module in library

- ? モジュール'XX'の作成日時がライブラリ中のモジュールより古いいため置換しませんでした。
- ! リロケータブルファイルの生成日時を確認してください。



## 第9章 xrf79の操作方法

xrf79の機能を使用するための操作方法を説明します。xrf79の機能は、指定したソースファイルとアセンブラリストファイルから、分岐命令及びサブルーチン呼び出し命令の参照リスト（クロスリファレンス）ファイルを生成します。

### 9.1 xrf79コマンドパラメータ

xrf79のコマンドパラメータ一覧を次に示します。

パラメータ名	機能
ファイル名	xrf79の処理対象となるファイル名。
-.	メッセージを画面に出力しない。
-N	システムラベル情報をクロスリファレンスファイルに出力する。
-O	クロスリファレンスファイルを出力するディレクトリを指定する。
-V	xrf79のバージョン番号を表示する。
@ファイル名	ファイル名で指定されたファイルを参照する。

### 9.2 xrf79コマンドパラメータの指定規則

xrf79のコマンドパラメータは任意の順序で指定できます。

```
>xrf79 file ... [option ...]
>xrf79 [option ...] file ...
```

file=ソースファイル名又はアセンブラリストファイル名

option=コマンドオプション

#### ソースファイル名又はアセンブラリストファイル名

- ・一つ以上のファイル名を必ず指定してください。
- ・ファイル名には、パスが指定できます。
- ・最大600個までのファイルを指定できます。
- ・ファイル拡張子は必ず記述してください。
- ・必ず、ファイル拡張子が".lst"のアセンブラリストファイルを指定してください。
- ・複数のファイルを指定する場合は、ファイル名をスペース又はタブで区切って指定してください。

#### コマンドオプション

- ・複数のコマンドオプションを指定できます。

## 9.3 xrf79コマンドファイル

- ・ 入力パラメータを記述したコマンドファイル名を指定できます。
- ・ コマンドファイルの記述方法はIn79と同様です。In79の操作方法を参照してください。

## 9.4 xrf79コマンドオプション

以降に、コマンドオプションの指定規則を説明します。

-.

---

## 画面へのメッセージ出力停止

---

### 機能

- ・ xrf79が処理を行う際のメッセージを出力しません。
- ・ エラーメッセージ及びワーニングメッセージは出力されます。

### 記述規則

- ・ 本オプションは、コマンド行の任意の位置に指定できます。

### 記述例

```
>xrf79 -. sample.a79
```

-N

---

## システムラベル情報出力指定

---

### 機能

- ・ as79が出力するシステムラベルについての情報もクロスリファレンスファイルに出力します。システムラベルは、ピリオド2つ(..)で始まるラベルです。

### 記述規則

- ・ 本オプションは、コマンド行の任意の位置に指定できます。

### 記述例

```
>xrf79 -N sample.lst
```

sample.lstファイルから、sample.xrfファイルを生成します。

```
>xrf79 -N sample.a79
```

sample.a79ファイルから、sample.xrfファイルを生成します。

-O

---

## ファイル出力ディレクトリ指定

---

### 機能

- ・ クロスリファレンスファイルを出力するディレクトリを指定します。

### 記述規則

- ・ -O (ディレクトリ名) のように入力してください。
- ・ 本オプションとディレクトリ名の間、スペース又はタブは記述できません。
- ・ 本オプションは、コマンド行の任意の位置に指定できます。

### 記述例

```
>xrf79 -O¥work¥list sample.a79
```

¥work¥listディレクトリにsample.xrfファイルを生成

```
>xrf79 -O¥work¥list sample.lst
```

-V

---

## バージョン表示

---

### 機能

- ・ クロスリファレンサのバージョンを表示します。

### 注意事項

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・ 本オプションのみを指定してください。

### 記述例

```
>xrf79 -V
```

@

---

## コマンドファイル参照

---

### 機能

- ・ 指定したファイルの内容をコマンドパラメータとして、xrf79を起動します。

### 記述規則

- ・ 本オプションとファイル名の間、スペース又はタブは記述できません。
- ・ 他のパラメータはコマンド行に入力できません。

### 記述例

```
>xrf79 @cmdfile
```

## 9.5 xrf79エラーメッセージ

Can't create temporary file

- ? テンポラリファイルの生成ができません。
- ! ディレクトリ情報を確認してください。

Can't open file 'XX'

- ? XX ファイルがオープンできません。
- ! ファイル名を確認してください。

Command-file is included in itself

- ? コマンドファイル自身をインクルードしています。
- ! コマンドファイルの記述内容を確認してください。

Command-file line characters exceed 255

- ? コマンドファイルの1行の文字数が255文字を越えています。
- ! コマンドファイルの内容を確認してください。

Command line is too long

- ? コマンド行の文字列が長すぎます。
- ! コマンドファイルを作成してください。

Input files exceed 80

- ? 入力ファイルの数が80を越えました。
- ! コマンドを入力し直してください。コマンドファイルの内容を分割してください。

Invalid option 'XX' is used

- ? 無効なコマンドオプション'XX'の指定があります。
- ! コマンドオプションを指定し直してください。

No input files specified

- ? 入力ファイルの指定がありません。
- ! ファイル名を指定してください。

Not enough memory

- ? メモリが足りません。
- ! メモリを増設してください。

Option 'XX' is not appropriate

- ? コマンドオプションの指定が間違っています。
- ! コマンドオプションの指定方法を確認して指定し直してください。



## 第10章 abs79の操作方法

abs79の機能を使用するための操作方法を説明します。abs79の機能は、指定したアセンブラリストファイルからアブソリュートリストファイルを生成します。

### 10.1 abs79使用上の注意事項

- ・ マクロ機能及び構造化記述命令を使用しているソースファイル进行处理する場合は、必ずコマンドオプション-LMSを付加してアセンブラリストファイルを生成してください。
- ・ abs79の処理対象となるアセンブラリストファイルを生成する場合は、abs79のコマンドオプション'-H'を付加しないでください。コマンドオプション'-H'を指定するとアセンブラリストファイルにヘッダ情報が出力されません。abs79は、ヘッダ情報が出力されていないアセンブラリストファイルを正しく処理できません。
- ・ abs79の処理対象となるアブソリュートモジュールファイルを生成する際に、必ずln79のコマンドオプション'-G'を付けて下さい。
- ・ 1つのソースファイル内に、同一名のセクションが複数個定義されており、そのセクションが指示命令'.LIST OFF'で、アセンブラリストファイルに出力されていない場合には、正しい実アドレスが出力されない場合があります。

### 10.2 abs79コマンドパラメータ

abs79のコマンドパラメータ一覧を次に示します。

パラメータ名	機能
ファイル名	abs79の処理対象となるファイル名
-.	メッセージを画面に出力しない。
-D	アセンブラリストファイルを検索するディレクトリ名を指定する。
-O	アブソリュートリストファイルを生成するディレクトリを指定する。
-V	abs79のバージョン番号を表示する。

## 10.3 abs79コマンドパラメータの指定規則

コマンドパラメータは、必ず次の順序に従って指定してください。

- 1 コマンドオプション
- 2 アブソリュートモジュールファイル名
- 3 アセンブラリストファイル名

```
>abs79 [option ...] X79_file [LST_file ...]
```

option=コマンドオプション

X79\_file=アブソリュートモジュールファイル名

LST\_file=アセンブラリストファイル名

### アブソリュートモジュールファイル名 (必須)

- ・ アブソリュートモジュールファイル名は必ず指定してください。
- ・ アブソリュートモジュールファイル名にはパスが指定できます。
- ・ 拡張子(.x79)は省略できます。

### アセンブラリストファイル名

- ・ アセンブラリストファイルは、スペースで区切って複数指定できます。
- ・ アセンブラリストファイル名にはパスが指定できます。
- ・ ファイル属性は省略できます。
- ・ アセンブラリストファイル名は省略できます。省略した場合は、アブソリュートモジュールファイルに含まれているリロケータブルモジュールファイルに対応するアセンブラリストファイルを処理対象とします。

### アブソリュートリストファイル名

- ・ アブソリュートリストファイル名は、アセンブラリストファイルの拡張子を".als"に変更したものになります。(sample.lst sample.als)

### コマンドオプション

- ・ コマンドオプションの大文字小文字は区別しません。
- ・ コマンドオプションとその引数の間には、スペース又はタブを必ず入力してください。

## 10.4 abs79コマンドオプション

以降に、コマンドオプションの指定規則を示します。

-.

---

## 画面へのメッセージ出力停止

---

### 機能

- ・ エラーメッセージを除く全てのメッセージの出力を停止します。

### 記述規則

- ・ 本オプションは、コマンド行の任意の位置に指定できます。

### 記述例

```
>abs79 -. sample
```

## -D

---

### ファイル検索ディレクトリ指定

---

#### 機能

- ・ アセンブラリストファイルの参照先ディレクトリを指定します。
- ・ 本指定がない場合は、カレントディレクトリからアセンブラリストファイルを検索します。

#### 記述規則

- ・ -D (ディレクトリ名) のように入力してください。
- ・ 本オプションとディレクトリ名の間には、スペース又はタブは記述できません。

#### 記述例

```
>abs79 sample -Ddir
```

カレントディレクトリの下での"dir"内のアセンブラリストファイルを検索します。

```
>abs79 sample -Ddir list1
```

カレントディレクトリの下での"dir"内の"list1.lst"を検索します。

-O

---

## ファイル出力ディレクトリ指定

---

### 機能

- ・ アブソリュートリストファイルの生成ディレクトリを指定します。
- ・ 本指定がない場合は、カレントディレクトリにファイルを生成します。

### 記述規則

- ・ -O (ディレクトリ名) のように入力してください。
- ・ 本オプションとディレクトリ名の間には、スペース又はタブは記述できません。

### 記述例

```
>abs79 sample -Oabslist
```

カレントディレクトリの下の"abslist"ディレクトリにアブソリュートリストファイルを生成します。

-V

---

## バージョン表示

---

### 機能

- ・ アブソリュートリスタのバージョンを画面に表示します。

### 注意事項

---

本オプションを指定した場合は、コマンド行の他のパラメータは全て無視されます。

---

### 記述規則

- ・ 本オプションのみを指定してください。

### 記述例

```
>abs79 -V
```

## 10.5 abs79エラーメッセージ

Can't create file 'XX'

- ? 'XX'が生成できません。
- ! ディレクトリ情報を確認してください。

Can't open file 'XX'

- ? 'XX'がオープンできません。
- ! ファイル名を確認してください。

Can't write in file 'XX'

- ? 'XX'に書き込みできません。
- ! ファイル名等を確認してください。

Command line is too long

- ? コマンド行の文字数が多すぎます。
- ! コマンドを入力し直してください。

Error information is in 'XX'

- ? 'XX'はエラー情報を含んでいます。
- ! アセンブラリストファイルを生成し直してください。

Illegal file format 'XX'

- ? 'XX'のフォーマットが正しくありません。
- ! ファイル名を確認してください。

Input files number exceed 80

- ? 入力ファイル数が80を越えています。
- ! コマンドを入力し直してください。

Not enough disk space

- ? ディスク容量が不足です。
- ! ディスク情報を確認してください。

Not enough memory

- ? メモリ容量が不足です。
- ! メモリを増設してください。

Section information is not appropriate in 'XX'

- ? 'XX'内のセクション情報が正しくありません。
- ! ファイル名を確認してください。

## 10.6 abs79ワーニングメッセージ

Address area exceed 0FFFFFFFH

- ? アドレス範囲0FFFFFFFHを越えています。
- ! アブソリュートモジュールファイル名を確認してください。

File 'l-filename' is missing corresponding to module in 'a-filename'

- ? 'a-filename'が持つモジュールに相当する'l-filename'がありません。該当するアブソリュートリストファイルは生成しませんでした。
- ! アセンブラリストファイルを生成し直してください。アセンブラリストファイルのあるディレクトリを確認してください。

Lines 'num-num' are relocatable address in 'filename'

- ? 'filename'の'num-num'行はリロケートブルアドレスのままです。
- ! アセンブリソースファイルに指示命令".LIST OFF"が記述されていることを確認してください。

No information of 'l-filenam' in 'a-filename'

- ? 'a-filename'は、'l-filename'の情報を持っていません。
- ! ファイル名を確認してください。

No section information of l-name in x-name

- ? x-nameは、l-nameのセクション情報を持っていません。
- ! l-nameからアブソリュートリストファイルは生成できません。

Overwrite in 'XX'

- ? 'XX'に上書きします。
- ! 古いファイルの内容は保存されません。

Symbol 'XX' is too long in XX

- ? シンボル名の文字数が1行に記述できる文字数を超えています。
- ! シンボル名を変更してください。



# 第11章 入出力ファイル

## 11.1 入出力ファイル一覧

本システムが扱うファイルの一覧を次に示します。ユーザが作成するソースファイル及びインクルードファイルの拡張子は任意につけることができます。ただし、プログラムを起動する際にファイル拡張子を省略した場合には、表の【】に示す拡張子を付加して処理します。

### as79の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
ソースファイル【.a79】	リロケータブルモジュールファイル【.r79】
インクルードファイル	アセンブラリストファイル【.lst】
	アセンブラエラータグファイル【.atg】

### ln79の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
リロケータブルモジュールファイル【.r79】	オブジェクトモジュールファイル【.x79】
ライブラリファイル【.lib】	マップファイル【.map】
	リンクエラータグファイル【.ltg】

### lmc79の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
オブジェクトモジュールファイル【.x79】	モトローSフォーマットファイル【.mot】
	インテルHEXフォーマットファイル【.hex】

### lb79の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
リロケータブルモジュールファイル【.r79】	ライブラリファイル【.lib】
ライブラリファイル【.lib】	リロケータブルモジュールファイル【.r79】
	ライブラリリストファイル【.lls】

### xrf79 の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
ソースファイル【.a79】	クロスリファレンスファイル【.xrf】
	アセンブラリストファイル【.lst】

### abs79の入出力ファイル

入力ファイル名【拡張子】	出力ファイル名【拡張子】
オブジェクトモジュールファイル【.x79】	オブジェクトリストファイル【.als】
	アセンブラリストファイル【.lst】

### 11.2 ソースファイル

#### ソースファイル名

任意のファイル名を指定してください。本アセンブラシステムでは、ソースファイルの拡張子はデフォルトで'.a79'です。他の拡張子名を定義した場合は、アセンブラを起動する際にファイル名をフルネームで指定してください。

#### ソースファイルの生成ディレクトリ

任意のディレクトリに生成してください。

#### ソースファイルのフォーマット

「プログラムの記述規則」に従って記述されたファイルを示します。テキストエディタで記述されたテキストデータです。

### 11.3 リロケータブルモジュールファイル

as79は、ソースファイルを読み込んでリロケータブルモジュールファイル（以降'R79ファイル'と記述します）を生成します。1つのソースファイルに対して1つのR79ファイルを生成します。

#### R79ファイル名

ソースファイルの拡張子（デフォルトで".a79"）を".r79"に変更したものが、リロケータブルモジュールファイルのファイル名になります。(sample.a79 sample.r79)

#### R79ファイルの生成ディレクトリ

コマンドオプション(-O)でディレクトリを指定した場合は、指定されたディレクトリに生成します。

指定のない場合は、ソースファイルのあるディレクトリに生成します。

#### R79ファイルフォーマット

as79は、IEEE-695に準拠したフォーマットのリロケータブルモジュールファイルを生成します。

#### 注意事項

---

このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。このファイルをエディタでオープンしたり、編集した場合の以降の処理は正常に行われませんのでご注意ください。

---

### 11.4 アセンブラリストファイル

as79は、コマンドオプション(-L[!M!S])を指定した場合のみ、ソース行情報及びリロケータブル情報を出力可能なテキスト形式のアセンブラリストファイル（以降'LSTファイル'と記述します）を生成します。

#### LSTファイル名

ソースファイルの拡張子（デフォルトでは".a79"）を".lst"に変更したものが、LSTファイルのファイル名になります。（sample.a79 sample.lst）

#### LSTファイルの生成ディレクトリ

コマンドオプション(-O)でディレクトリを指定した場合は、指定されたディレクトリに生成します。

指定のない場合は、ソースファイルのあるディレクトリに生成します。

#### LSTファイルフォーマット

LSTファイルの出力フォーマットをLSTファイル例 1 に示します。

##### ヘッダ

ヘッダ情報として、アセンブラリストファイルの1ページ毎に、次に示す情報を出力します。

1行目 アセンブル実行日時、ページ番号。

2行目 （空行）

3行目 行番号、ロケーション、オブジェクトコード、行の種類、桁数を示す記号。

4行目 （空行）

##### 行番号 (SEQ.)

アセンブラリストの行番号を出力します。

##### ロケーション (LOC.)

アセンブル時に決定できる範囲のオブジェクトコードのロケーションアドレスを出力します。

##### オブジェクトコード (OBJ.)

命令に対応するオブジェクトコードを出力します。

## 入出力ファイル

### 行の種類 (OXMSJA)

as79がソース行を処理した結果の情報を出力します。次の情報が出力されます。

行記号	内容
0	X M S J A
0~9	インクルードファイルのネストレベルを示す。
X	条件アセンブルでアセンブルされなかった行を示す。
M	マクロの展開行を示す。
-	.DEFINEを置き換える以前の行を示す。
S	構造化記述命令の展開行を示す。
J	分岐最適化が行われたことを示す。
A	アドレッシングモード最適化が行われたことを示す。

### 桁数 ( ....\*....SOURCE STATEMENT....7....\*....8.)

ソースプログラムの桁数を示します。この桁には、アセンブリソース行を出力します。

## 入出力ファイル

### LSTファイル例1

LSTファイルフォーマットは、ソースプログラムの行毎にLSTファイルへの出力形式が異なります。また、LSTファイル生成のコマンドオプションによっても出力形式が異なります。

- ・ リストファイル例1に、コマンドオプション"-LCDMIS"を指定して生成したLSTファイルを示します。
- ・ 行番号は、リストファイルに記述されている行番号を示しています。

行番号	内容
6	指示命令".ORG"でロケーションアドレスを定義した行です。
7~11	指示命令".BLKB",".BLKW",".BLKA",".BLKD",".BLKF",".BLKDF"で領域定義を行った行です。
12	指示命令".EQU"でシンボル定義を行った行です。
13	指示命令".BTEQU"でビットシンボル定義を行った行です。
14	指示命令".DEFINE"で文字列をシンボルに定義した行です。
17,18	指示命令.BYTE,.WORD,.ADDR,.DWORD,.FLOAT,.DOUBLEでROM領域にデータを設定した行です。
19	ラベルのみを記述した行です。「ラベル定義行」になります。
22~24	読み込んだインクルードファイルを表示した行です。
25	条件アセンブルで条件が偽となった行です。コマンドオプション(-LI)を指定した場合のみ出力されます。
28~30	マクロ定義を行っている行です。
33,34	7900シリーズのニーモニックを記述した行です。
35~37	行連結指定子を記述した行です。
39~41	マクロを展開した結果のアセンブリソース行を出力します。コマンドオプション(-LM)を指定した場合のみLSTファイルに出力されます。
42	文字列定義によるシンボルが使用されている行のソース行です。コマンドオプション(-LD)を指定した場合のみLSTファイルに出力されます。
45~65	構造化記述命令を記述した行です。コマンドオプション(-LS)を指定した場合のみ構造化記述命令を展開した結果がLSTファイルに出力されます。
Information List	
アセンブルを行った結果の全エラー数、全ワーニング数及び全リスト行数を出力します。	
Section List	
セクションのタイプ、セクションサイズ及びセクション名をリストにして出力します。	

# 入出力ファイル

\* M37950 ASSEMBLER ASP79 \* SOURCE LIST Tue Jan 21 11:18:44 1997 PAGE 001

```

SEQ. LOC.      OBJ.          OXMSJA  ....*....SOURCE STATEMENT....7....*....8

 1              .FORM      43,80
 2              .VER       "sample program file"
 3              .GLB       sub1,sub2
 4              .GLB       sym1,sym2,sym3
 5              .SECTION   ram,DATA
 6 000800      .ORG       800H
 7 000800(000001H) work1:  .BLKB   1
 8 000801(000001H) mem1:  .BLKB   1
 9 000802(000001H) mem2:  .BLKB   1
10 000803(000001H) mem3:  .BLKB   1
11 000804(000001H) flags:  .BLKB   1
12 00000000h   sym0     .EQU    0
13 0,00000804h btsym   .BTEQU  0,flags
14             membit   .DEFINE #01,mem3
15             .SECTION   static,ROMDATA
16 00F000      .ORG       0f000H
17 00F000 636F6465 .BYTE   "code"
18 00F004 141E28   .BYTE   20,30,40
19 00F007      cond:
20             .IF       sym4
21             .INCLUDE   data1.inc
22 00F007      1        data1:
23 00F007 5468697320697320 1 .BYTE   "This is sample1 program
           73616D706C653120 1
           70726F6772616D2E 1
24             .ELSE
25             X         .INCLUDE   data2.inc
26             .ENDIF
27             .SECTION   macrodef
28             ldm      .MACRO   p1,p2
29             MOV      p2,p1
30             .ENDM
31             .SECTION   program
32 000000      main:
33 000000 166400     LDA     A,#100
34 000003 DE0008     STA     A,work1
35             MOV      #3,mem1,#sym1,\\
36             mem2,#sym2,\\ ;This is
37             mem3,#sym3

```

# 入出力ファイル

\* M37950 ASSEMBLER ASP79 \* SOURCE LIST Tue Jan 21 11:18:44 1997 PAGE 002

```
SEQ. LOC.      OBJ.          OXMSJA  ....*....SOURCE STATEMENT....7....*....8

38 000014                      mcall:
39                                ldm    #05,work1
40 000014 9605000008          M      MOVMM  work1,#05
41                                M      .ENDM
42                                -      CLB    membit
43 000019 51670308FEFF          CLB    #01,mem3
44 00001F                      struc:
45                                A = 0
46 00001F 54                    S      clr   a
47                                [work1] = 0
48 000020 44                    S      clrb  a
49 000021 CE0008                S A    stab  a, work1
50                                [mem1] = 0
51 000024 44                    S      clrb  a
52 000025 CE0108                S A    stab  a, mem1
53                                FOR    X = 0 TO 10 STEP 1
54 000028 E4                    S      clrX
55 000029                      S      ..fr0000:
56 000029 E60A00                S      cpx   #10
57 00002C 8008                  S      bgt   ..fr0002
58                                [work1,X] = 0
59 00002E 44                    S      clrb  a
60 00002F CF0008                S A    stab  a, work1,x
61                                NEXT
62 000032                      S      ..fr0001:
63 000032 0101                  S      addx  #1
64 000034 20F3                  S      bra   ..fr0000
65 000036                      S      ..fr0002:
66                                .END
```

## Information List

```
TOTAL ERROR(S)    00000
TOTAL WARNING(S)  00000
TOTAL LINE(S)     00066  LINES
SOURCE LINE(S)    00046  LINES
INCLUDED LINE(S)  00002  LINES
```

## Section List

Attr	Size	Name
DATA	0000005(00005H)	ram
ROMDATA	0000031(0001FH)	static
CODE	0000000(00000H)	macrodef
CODE	0000054(00036H)	program

# 入出力ファイル

## LSTファイル例2

- ・ コマンドオプション"-L"を指定して生成したLSTファイル例を示します。  
行番号は、リストファイルに記述されている行番号を示しています。

行番号	内容
-----	----

34	行連結指定子によって連結された結果の行です。
----	------------------------

```
* M37950 ASSEMBLER ASP79 * SOURCE LIST Tue Jan 21 11:19:34 1997 PAGE 001

SEQ. LOC. OBJ. OXMSJA ....*....SOURCE STATEMENT....7....*....8

1 .FORM 43,80
2 .VER "sample program file"
3 .GLB sub1,sub2
4 .GLB sym1,sym2,sym3
5 .SECTION ram,DATA
6 000800 .ORG 800H
7 000800(000001H) work1: .BLKB 1
8 000801(000001H) mem1: .BLKB 1
9 000802(000001H) mem2: .BLKB 1
10 000803(000001H) mem3: .BLKB 1
11 000804(000001H) flags: .BLKB 1
12 00000000h sym0 .EQU 0
13 0,00000804h btsym .BTEQU 0,flags
14 membit .DEFINE #01,mem3
15 .SECTION static,ROMDATA
16 00F000 .ORG 0f000H
17 00F000 636F6465 .BYTE "code"
18 00F004 141E28 .BYTE 20,30,40
19 00F007 cond:
20 .IF sym4
21 .INCLUDE data1.inc
22 00F007 1 data1:
23 00F007 5468697320697320 1 .BYTE "This is sample1 program
73616D706C653120 1
70726F6772616D2E 1
24 .ELSE
25 .ENDIF
26 .SECTION macrodef
27 ldm .MACRO p1,p2
28 MOVN p2,p1
29 .ENDM
30 .SECTION program
31 000000 main:
32 000000 166400 LDA A,#100
33 000003 DE0008 A STA A,work1
34 000006 6133 MOVR #3,mem1,#sym1,mem2,#sym2
0000r0108
0000r0208
0000r0308
```



# 入出力ファイル

\* M37950 ASSEMBLER ASP79 \* SOURCE LIST Tue Jan 21 11:19:34 1997 PAGE 002

SEQ. LOC. OBJ. OXMSJA ....\*....SOURCE STATEMENT....7....\*....8

```
35 000014          mcall:
36                  ldm    #05,work1
37 000019 51670308FEFF  CLB    #01,mem3
38 00001F          struc:
39                  A = 0
40                  [work1] = 0
41                  [mem1] = 0
42                  FOR    X = 0 TO 10 STEP 1
43                  [work1,X] = 0
44                  NEXT
45                  .END
```

## Information List

TOTAL ERROR(S)	00000	
TOTAL WARNING(S)	00000	
TOTAL LINE(S)	00045	LINES
SOURCE LINE(S)	00046	LINES
INCLUDED LINE(S)	00002	LINES

## Section List

Attr	Size	Name
DATA	0000005(00005H)	ram
ROMDATA	0000031(0001FH)	static
CODE	0000000(00000H)	macrodef
CODE	0000054(00036H)	program

## 入出力ファイル

### アセンブラソースリスト

リストファイル例1および2のソースプログラムです。

```
.FORM 40,81
.VER "sample program file"
.GLB sub1,sub2
.GLB sym1,sym2,sym3
.SECTION ram,DATA
.ORG 800H
work1: .BLKB 1
mem1: .BLKB 1
mem2: .BLKB 1
mem3: .BLKB 1
flags: .BLKB 1
sym0 .EQU 0
btsym .BTEQU 0,flags
membit .DEFINE #01,mem3
.SECTION static,ROMDATA
.ORG 0f000H
.BYTE "code"
.BYTE 20,30,40
cond:
. IF sym0 == 0
. INCLUDE data1.inc
. ELSE
. INCLUDE data2.inc
. ENDF
. SECTION macrodef
ldm .MACRO p1,p2
MOVW p2,p1
. ENDM
. SECTION program
main:
LDA A,#100
STA A,work1
MOVR #3,mem1,#sym1,\\ ;3lines connect
mem2,#sym2,\\ ;This is \\ all comment
mem3,#sym3
mcall:
ldm #05,work1
CLB membit
struc:
A = 0
[work1] = 0
[mem1] = 0
FOR X = 0 TO 10 STEP 1
[work1,X] = 0
NEXT
.END
```

data1.inc

```
DATA1:
.BYTE "This is sample1 program."
```

data2.inc

```
DATA2:
.BYTE "This is sample2 program."
```

### 11.5 アセンブラエラータグファイル

as79は、コマンドオプション(-T/-X)を指定した場合のみソースファイルをアセンブルする際に発生したエラーをアセンブラエラータグファイル（以降'ATGファイル'と記述します）に出力します。

このファイルをタグジャンプ機能を持つエディタで処理することでエラーを簡単に修正できます。

#### ATGファイル名

ソースファイルの拡張子（デフォルトでは".a79"）を".atg"に変更したものが、ATGファイルのファイル名になります。（sample.a79 sample.atg）

#### ATGファイルの生成ディレクトリ

コマンドオプション(-O)でディレクトリを指定した場合は、そのディレクトリに生成します。

指定のない場合は、ソースファイルのあるディレクトリに生成します。

#### ATGファイルのフォーマット

エディタのタグジャンプ機能を使用できるフォーマットになっています。

以下に示すように、ソースファイル名、エラー行番号、エラーメッセージの順に出力します。

```
sample.a79 21 Error (asp79) : Operand value is not defined
sample.a79 72 Error (asp79) : Undefined symbol exist "work2"
```

### 11.6 アブソリュートモジュールファイル

In79は、複数のR79ファイルから、一つのアブソリュートモジュールファイル（以降'X79ファイル'と記述します）を生成します。

#### X79ファイル名

通常は、コマンド行から入力された、R79ファイル名のうちの1番目のファイル名の拡張子".r79"を".x79"に変更したものが、X79ファイル名になります。

(sample.r79 sample.x79)

コマンドオプション(-O)で、ファイル名を指定した場合は、指定した名前になります。

#### X79ファイルの生成ディレクトリ

通常は、カレントディレクトリに生成します。

コマンドオプション(-O)のファイル名にパスが指定されている場合は、そのパスのディレクトリに生成します。

#### X79ファイルのフォーマット

このファイルは、IEEE-695に準拠したフォーマットになっています。

#### 注意事項

---

このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。このファイルをエディタでオープンしたり、編集した場合の以降の処理は正常に行われませんのでご注意ください。

---

### 11.7 マップファイル

In79は、コマンドオプション(-M/MS)を指定した場合のみ、リンク情報、セクションの最終配置アドレス情報、シンボル情報などをマップファイル（以降'MAPファイル'と記述します）に出力します。シンボル情報は、コマンドオプション(-MS)を指定した場合だけMAPファイルに出力されます。

#### MAPファイル名

X79ファイルの拡張子(.x79)を".map"に変更したものが、MAPファイルのファイル名になります。(sample.x79 sample.map)

#### MAPファイルの生成ディレクトリ

X79ファイルと同じディレクトリに生成します。

#### MAPファイルのフォーマット

次に示す情報を順にリスト形式でマップファイルに出力します。MAPファイルの出力フォーマットを《マップファイル例》に示します。

##### (1)リンク情報

コマンド行情報、R79ファイル名及びR79ファイルの生成日時情報を出力します。

##### (2)セクション情報

再配置されたセクション名、属性、タイプ、スタートアドレス、セクションサイズ、セクション整列の有無及びモジュール名（R79ファイル名）の情報を出力します。

##### (3)グローバルラベル情報

グローバルラベル名とアドレス情報を出力します。コマンドオプション"-MS"を指定した場合のみ出力されます。

##### (4)グローバルシンボル情報

グローバルシンボル名と数値情報を出力します。コマンドオプション"-MS"を指定した場合のみ出力されます。

##### (5)ローカルラベル情報

モジュール名（R79ファイル名）、ローカルラベル名及びアドレス情報を出力します。コマンドオプション"-MS"を指定した場合のみ出力されます。

##### (6)ローカルシンボル情報

モジュール名（R79ファイル名）、ローカルシンボル名及び数値情報を出力します。コマンドオプション"-MS"を指定した場合のみ出力されます。

##### (7)ローカルビットシンボル情報

モジュール名（R79ファイル名）、ローカルビットシンボル名、ビット位置及びメモリアドレス情報を出力します。コマンドオプション"-MS"を指定した場合のみ出力されます。

## 入出力ファイル

```
#####
# (1) LINK INFORMATION #
#####
ln79 -ms smplst smplink

# LINK FILE INFORMATION
smplst (smplst.r79)
    May 29 15:36:21 1997
    .VER "sample program file"
smplink (smplink.r79)
    May 29 15:34:21 1997

#####
# (2) SECTION INFORMATION #
#####
# SECTION          ATR TYPE      START  LENGTH ALIGN  MODULENAME
ram                ABS DATA    000800 000005      smplst
static             ABS ROMDATA  00F000 00001F      smplst
macrodef           REL CODE     00F01F 000000      smplst
program            REL CODE     00F01F 000036      smplst
subroutine         REL CODE     00F055 000002      smplink

#####
# (3) GLOBAL LABEL INFORMATION #
#####
sub1                00f055  sub2                00f056

#####
# (4) GLOBAL EQU SYMBOL INFORMATION #
#####
sym1                00000001  sym2                00000002  sym3                00000003

#####
# (5) LOCAL LABEL INFORMATION #
#####
@ smplst ( smplst.r79 )
..fr0000            00f048  ..fr0001            00f051  ..fr0002            00f055
cond                00f007  data1                00f007  flags                000804
main                00f01f  mcall                00f033  mem1                 000801
mem2                000802  mem3                 000803  struc                00f03e
work1               000800
@ smplink ( smplink.r79 )
```

```
#####  
# (6) LOCAL EQU SYMBOL INFORMATION      #  
#####  
@ smp1st ( smp1st.r79 )  
sym0          00000000  
@ smp1ink ( smp1ink.r79 )  
  
#####  
# (7) LOCAL EQU BIT-SYMBOL INFORMATION  #  
#####  
@ smp1st ( smp1st.r79 )  
btsym         0 000804  
@ smp1ink ( smp1ink.r79 )
```

## 11.7 リンクエラータグファイル

ln79は、コマンドオプション(-T)を指定した場合のみ、リンクエラー情報をリンクエラータグファイル（以降'LTGファイル'と記述します）に出力します。このとき、エラーが発生した箇所をアセンブリソース行で出力します。

このファイルをタグジャンプ機能を持つエディタで処理することでエラーを簡単に修正できます。

### LTGファイル名

X79ファイルの拡張子(.x79)を".ltg"に変更したものがLTGファイルのファイル名になります。(sample.x79 sample.ltg)

### LTGファイルの生成ディレクトリ

X79ファイルと同じディレクトリに生成します。

### LTGファイルのフォーマット

フォーマットは、ATGファイルと同じです。エディタのタグジャンプ機能が使用できます。

次に示すように、ソースファイル名、エラー行番号、エラーメッセージの順に出力します。

```
smp.inc 2 Warning (ln79) : smp2.r79 : Absolute-section is  
written agter the absolute-section 'ppp'  
smp.inc 2 Error (ln79) : smp3.r79 : Address is overlapped in  
'CODE' section 'ppp'
```

### 11.8 モトローラSフォーマットファイル

lmc79は、X79ファイルから一つの本トローラSフォーマットファイル（以降'MOTファイル'と記述します）を生成します。

#### MOTファイル名

X79ファイルの拡張子(.x79)を".mot"に変更したものがMOTファイルのファイル名になります。(sample.x79 sample.mot)

#### MOTファイルの生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

#### MOTファイルのフォーマット

7900シリーズチップの内蔵ROMやEPROMなどに書き込み可能な本トローラSフォーマットです。

MOTファイルを生成する際に次のデータを指定できます。

- ・ 0H ~ 0FFFFFFH番地までのデータ領域を設定できます。
- ・ 1データレコード長を16バイト又は32バイトのいずれかに設定できます。
- ・ 実行開始アドレスを設定できます。

### 11.9 インテルHEXフォーマットファイル

lmc79は、X79ファイルから一つの本テルHEXフォーマットファイル（以降HEXファイルと記述します）を生成します。

#### HEXファイル名

X79ファイルの拡張子(.x79)を".hex"に変更したものがHEXファイルのファイル名になります。(sample.x79 sample.hex)

#### HEXファイルの生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

#### HEXファイルのフォーマット

コマンドオプション(-H)を指定した場合のみ、7900シリーズチップの内蔵ROMやEPROMなどに書き込み可能な本テルHEXフォーマットです。

HEXファイルを生成する際に次のデータを指定できます。

- ・ 0H ~ 0FFFFFFH番地までのデータ領域を設定できます。
- ・ 1データレコード長を16バイト又は32バイトのいずれかに設定できます。

#### 注意事項

---

1Mbyte空間(0H ~ 0FFFFFFH)を越えるアプソリュートモジュールファイルの場合は、三菱専用HEXフォーマットで出力します。

---



### 11.10 ライブラリファイル

lb79は、as79が生成したR79ファイルをモジュールとして、一つのファイルにまとめたライブラリファイル（以降'LIBファイル'と記述します）を生成します。

#### LIBファイル名

コマンド行で指定した名前のLIBファイルが生成されます。拡張子は".lib"のみです。コマンド行で、ファイル名を省略できません。

#### LIBファイルの生成ディレクトリ

コマンド行でパスを指定した場合は、そのディレクトリに生成されます。パスの指定が無い場合はカレントディレクトリに生成されます。

#### LIBファイルのフォーマット

このファイルは、IEEE-695フォーマットに準拠しています。

#### 注意事項

---

このファイルは、バイナリ形式のため、画面やプリンタに出力したり、編集したりできません。このファイルをエディタでオープンしたり、編集した場合の以降の処理は正常に行われませんのでご注意ください。

---

### 11.11 ライブラリリストファイル

lb79は、LIBファイル及びLIBファイルに登録されている任意のモジュールの情報をライブラリリストファイル（以降'LLSファイル'と記述します）に出力します。

#### LLSファイル名

LIBファイルの拡張子".lib"を".lls"に変換したものがLLSファイル名になります。

#### LLSファイルの生成ディレクトリ

ファイルはカレントディレクトリに生成されます。

#### LLSファイルのフォーマット

画面やプリンタに出力可能なテキスト形式です。このファイルを参照することで、LIBファイルに登録されているモジュールの概要を確認することができます。LLSファイルのフォーマットを《ライブラリリストファイル例》に示します。

次に、LLSファイルに出力される情報を示します。

## 入出力ファイル

### (1)ライブラリ情報

1つのライブラリファイルに、1回出力されます。次に示す情報を出力します。

ライブラリファイル名

LIBファイル名を出力します。

ファイル更新日時

LIBファイルの最新の更新日時を出力します。

モジュール数

LIBファイルに登録されているモジュールの総数を出力します。

グローバルシンボル数

LIBファイルに登録されているグローバルラベル及びグローバルシンボルの総数を出力します。

### (2)モジュール情報

登録されているモジュール毎に、1回出力されます。次に示す情報を出力します。

モジュール名

LIBファイルに登録されているモジュール名を出力します。

バージョン情報

指示命令".VER"で指定された文字列情報を出力します。

登録日時

モジュールをLIBファイルに登録した日時を示します。

モジュールサイズ

登録されているモジュールのコード及びデータサイズを出力します。

### 注意事項

**モジュールサイズは、R79ファイルのファイルサイズとは異なります。**

グローバルシンボル名

モジュール内で定義されているグローバルシンボル及びグローバルラベル名を出力します。

外部参照シンボル名

モジュールが外部参照しているグローバルシンボル及びグローバルラベル名を出力します。

```
Librarian (lb79) for M37950 Series Version 1.00.00
Library file name:      libsmp.lib
Last update time:      1996-Mar-1 15:44
Number of module(s):   1
Number of global symbol(s): 12

Module name:           sample
.Ver:                  .VER           "sample program file"
Date:                  1996-Mar-1 15:43
Size:                  00894H
Global symbol(s):     btsym5 btsym6 btsym7
                      btsym8 btsym9 sub1
                      sub2 sym5 sym6
                      sym7 sym8 sym9
```

## 11.12 クロスリファレンスファイル

xrf79は、ソースファイルを基に、シンボル及びラベルの定義と参照の情報をまとめたクロスリファレンスファイル（以降'XRFファイル'と記述します）を生成します。

### XRFファイル名

LSTファイル名又はソースファイル名の拡張子(.a79または.lst)を".xrf"に変更したものがXRFファイル名になります。(sample.lst sample.xrf;sample.a79 sample.xrf)

### XRFファイルの生成ディレクトリ

コマンド行で、パスを指定した場合は、そのディレクトリにファイルを生成します。

コマンドオプション(-O)でディレクトリを指定した場合は、そのディレクトリにファイルを生成します。

上記のディレクトリ指定が無い場合は、カレントディレクトリにファイルを生成します。

### XRFファイルフォーマット

このファイルは、画面やプリンタに出力可能なフォーマットになっています。従って、デバッグなどの際にプリントアウトして、シンボルを定義したソースファイル内の位置を確認することができます。XRFファイルのフォーマットを《クロスリファレンスファイル例》に示します。

次にXRFファイルに出力される情報を示します。

#### (1)ラベル名

ラベル名を出力します。

#### (2)ファイル名

上記ラベルの記述されているファイル名を出力します。

#### (3)参照及び宣言されている行番号とその区別を示す記号

行番号に次に示す記号を添付して出力します。

:d 定義行

:j 分岐命令による参照行

:s サブルーチン呼び出し命令による参照行

```
btsym0
  sample.a79
    00023:d
btsym1
  sample.a79
    00024:d
btsym2
  sample.a79
    00025:d
btsym20
  sample.a79
    00033:d
```

### 11.13 アブソリュートリストファイル

abs79は、画面やプリンタに出力可能なフォーマットのアブソリュートリストファイル（以降'ALSファイル'と記述します）を生成します。

#### ALSファイル名

LSTファイルの拡張子(.lst)を".als"に変更したものがALSファイル名になります。  
(sample.lst sample.als)

#### ALSファイルの生成ディレクトリ

コマンドオプション(-O)が指定されている場合は、そのディレクトリにファイルを生成します。

上記以外の場合は、カレントディレクトリにファイルを生成します。

#### ALSファイルのフォーマット

ALSファイルのフォーマットは、ロケーション情報が絶対アドレス情報に変換される点を除いてはLSTファイルと同一です。

## 第12章 プログラムの記述規則

---

AS79アセンブラに対応のソースプログラムを記述するための基本規則を示します。

### 12.1 プログラム記述上の注意事項

命令及びオペランドの記述は必ず、全て半角文字で記述してください。コメント以外には、多バイト文字（漢字など）は使用できません。

予約語は、ソースプログラム中の名前に使用しないでください。

AS79の指示命令からピリオド及びアンダーラインをとった文字列については、名前に使用してもエラーとなりません。しかし、AS79の処理に影響する文字列もありますので使用しないでください。

システムラベル（"."で始まる文字列）については、ユーザーがソースプログラム内に記述した場合でもエラーは出力しませんが、AS79の拡張用に用いられる可能性がありますので使用しないでください。

構造化記述でメモリ・メモリビット変数を指定する際は、"[ ]"で囲んで指定して下さい。

構造化記述を使用する場合、一時領域としてアキュムレータ(A,B,E)を使用しますのでご注意下さい。

構造化記述の代入文は、乗除算及び符号・ゼロ拡張を除き、同一サイズ間演算のみ有効です。

as79及びln79は、実際の7900シリーズの機種毎のROM及びRAMの物理的なアドレス情報を持っていません。したがって、リンクの結果によって、DATAタイプのセクションがチップのROM領域に配置されてしまうこともあります。リンクを実行する場合は、実際のチップのアドレスを確認して、セクションを配置するようにしてください。

## 12.2 文字セット

AS79対応のアセンブラプログラムは、AS79の仕様で示した「文字セット」を使用して記述できます。

### 注意事項

---

命令及びオペランドの記述は必ず、全て半角文字で記述してください。コメント以外には、多バイト文字（漢字など）は使用できません。

---

## 12.3 予約語

AS79が予約語としている名前を次に示します。次に示す予約語は、大文字と小文字を区別しません。"abs","ABS","Abs","ABs","AbS","abS","aBs","aBS"は、全て予約語になります。

### 12.3.1 予約語の種類

#### アセンブル指示命令

本マニュアルで説明している全てのアセンブル指示命令と指示命令に対するキーワードおよびピリオド一つで始まる文字列の全ては予約語です。

#### 演算子

本マニュアルで説明している全ての演算子及び構造化記述命令の演算子は全て予約語です。

#### 構造化記述命令

本マニュアルで説明している構造化記述命令は全て予約語です。

#### システムラベル

アセンブラが生成するラベルをシステムラベルといいます。ピリオド二つで始まる名前は、全てシステムラベルとして扱います。

#### ニーモニック

7900シリーズのニーモニック全てが予約語です。

#### レジスタ/フラグ名

7900シリーズのレジスタ名及びフラグ名は全てが予約語です。

## 12.4 名前

### 12.4.1 名前の記述規則

#### 名前の長さ

名前として記述できる文字列の長さは、255文字までです。ただし、構造化記述の変数名として記述できる文字列の長さは、32文字までです。

#### 名前の判別

名前は、大文字と小文字を区別して扱います。"LAB","Lab"は異なる名前として扱います。

#### 注意事項

---

予約語と同一の名前を使用することはできません。万一使用した場合のプログラムの動作については保証いたしません。

---

#### 名前の種類

名前は、ソースプログラムの中で、任意に定義し使用できます。

名前は、次の種類に分けられ、それぞれ記述できる範囲が異なります。

- ・ ラベル名
- ・ シンボル名
- ・ ビットシンボル名

## 12.4.2 ラベル

### 機能

- ・ CPUがアクセス可能な範囲の、特定のアドレスについて付けられる名前。

### 記述規則

- ・ 名前には英数字、アンダーライン及びクエスチョンが使用できます。
- ・ 名前の先頭には、数字は使用できません。
- ・ 定義の際には、名前の最後に必ずコロン(:)を付けてください。
- ・ 一つのファイル中に同一のラベル名を再定義することはできません。

### 定義方法

- ・ ラベルを定義するには次の二つの方法があります。

- 1 指示命令で、領域を確保する。

例)

```
flags:  .BLKB  1
work:   .BLKD  1
```

- 2 ソース行の先頭に名前を記述する。

例)

```
name1:
name:
sym_name:
```

### 参照方法

例)

```
JMP sym_name
[work] = 10
```

### 注意事項

構造化記述命令でラベルを参照する際は、"[ ]"囲んで記述してください。



## 12.4.3 シンボル

### 機能

- ・ 定数値について付けられる名前。

### 記述規則

- ・ 数値は、アセンブル実行時に確定しなければなりません。
- ・ 名前には英数字、アンダーライン及びクエスチョンが使用できます。
- ・ 名前の先頭には、数字は使用できません。
- ・ セクションの範囲外でも定義できます。
- ・ 名前の最後に、コロン(:)は必要ありません。
- ・ 同一名のシンボルを再定義した場合は、後で定義したシンボル値が有効となります。

### 定義方法

- ・ シンボルを定義するには、数値定義の指示命令を使用します。

例)

```
value1 .EQU1  
value2 .EQU2
```

### 参照方法

- ・ 命令のオペランドにシンボルを記述します。

例)

```
MOVM.W work,#value1  
[work] = value2  
value3 .EQUvalue2+1
```

## 12.4.4 ビットシンボル

### 機能

- ・ 特定のメモリの特定のビット位置について付けられる名前。
- ・ ビット操作命令のオペランドに記述できます。
- ・ ビットシンボルは、グローバル指定できません。

### 注意事項

ビットシンボルはローカルシンボルとしてのみ使用可能です。

### 記述規則

- ・ オペランドはカンマ(,)で区切って記述してください。
- ・ 第1オペランドにビット位置を示す数値を、第2オペランドにメモリ名を記述してください。

### 注意事項

オペランドが複数記述できる場合は、左から"第1オペランド"、"第2オペランド"のように記述します。

- ・ ビット位置を示す数値は、アセンブル実行時に確定しなければなりません。
- ・ ビット位置を示す数値は、データ長宣言が8ビットの場合0～7の範囲の整数値が、16ビットの場合は0～15の整数値が記述できます。
- ・ 名前には英数字、アンダーライン及びクエスチョンが使用できます。
- ・ 名前の先頭には、数字は使用できません。
- ・ セクションの範囲外でも定義できます。
- ・ 同一名のビットシンボルを再定義することはできません。

### 定義方法

- ・ ビットシンボルを定義するには、ビットシンボル定義の指示命令(.BTEQU)を使用します。

例)

```
flag1    .BTEQU    1,flags
flag2    .BTEQU    2,flags
```

### 参照方法

- ・ ビット操作命令のオペランドに記述できます。

例)

```
BSC flag1,LAB1
BSS flag2,LAB2
```

### 12.4.5 ロケーションシンボル

#### 機能

- ・ 記述した行のアドレスを示します。
- ・ ドルマーク(\$)をオペランドに記述することで、記述した行のオペコードの1バイト目のアドレスを示します。

#### 記述規則

- ・ ニーモニックのオペランドに記述してください。
- ・ 名前や、予約語の先頭に'\$'は、記述できません。
- ・ ロケーションシンボルを式の項に記述できます。

#### 記述例

```
JMP $+5
```

#### 注意事項

ロケーションシンボルをオフセットとするアドレスを分岐命令のニーモニックに記述する場合は、分岐先アドレスまでの全ての命令に対して、最適化が行われないように記述してください。警告としてアセンブル時に、ワーニングを発生します。

## 12.5 行の記述規則

### 注意事項

---

as79では、指示命令、ニーモニック及び構造化記述命令を同じ一行に記述することはできません。また、1行に複数の命令を記述することもできません。

---

### 12.5.1 行の種類

as79はソースプログラムを行単位で処理します。行に記述されている内容によって以降に示す種類に分けられます。

#### 指示命令行

- ・ AS79の指示命令を記述した行です。
- ・ 指示命令は、一行に一つのみ記述できます。
- ・ 指示命令行には、コメントを記述できます。

#### アセンブリソース行

- ・ ニーモニックを記述した行です。
- ・ アセンブリソース行には、コメントを記述できます。
- ・ アセンブリソース行には、先頭にラベル名を記述できます。

#### 構造化記述行

- ・ 構造化記述命令を記述した行です。
- ・ 構造化記述行には、コメントを記述できます。
- ・ 構造化記述行には、先頭にラベル名を記述できます。
- ・ 対応するニーモニックに置き換えられます。

#### ラベル定義行

- ・ ラベル名だけを記述した行です。

#### コメント行

- ・ コメントだけを記述している行です。

#### 空行

- ・ スペース、タブ又は改行コードだけを含む行です。

## 12.5.2 行の記述規則

### 行の区切り

- ・ 改行文字で区切られ、改行文字の直後の文字から、次の改行文字までを 1 行とします。

### 行の長さ

- ・ 一行に記述可能な最大文字数は512文字です。512文字を越えた文字については、as79は処理しません。

以降に各行の記述規則を説明します。

## 12.5.3 指示命令行

### 機能

- ・ アセンブラ指示命令を記述できます。

### 記述規則

- ・ 指示命令とそのオペランドの間に、必ずスペース又はタブを記述してください。
- ・ オペランドを複数個記述する場合は、オペランドとオペランドの間に、必ずカンマ(,)を記述してください。
- ・ オペランドとカンマの間には、スペース又はタブを記述できます。
- ・ 指示命令によっては、オペランドを記述しないものがあります。
- ・ 指示命令は行の先頭から記述できます。
- ・ 指示命令行の先頭には、スペース又はタブを記述できます。
- ・ 指示命令行にコメントを記述する場合は、指示命令とオペランドのつぎに、セミコロン(;)を記述し、それ以降の桁にコメントを記述してください。
- ・ コメント行は、アセンブラリストファイルに出力されます。

### 注意事項

AS79は、セミコロン(;)以降の桁に記述した内容は全てコメントとして処理を行います。セミコロン以降の桁に記述した指示命令について、アセンブラはコード生成を行いません。セミコロン(;)の記述位置には注意してください。AS79は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

- ・ 指示命令のオペランドとコメントの間にはスペース又はタブを記述できます。

### 記述例

```

SECTION      program,DATA,ALIGN=2
.ORG         00H
sym          .EQU         0
work:       .BLKB        1
            .ALIGN       2
            .PAGE        "newpage"
            .ALIGN       2           ;comment
    
```

## 12.5.4 アセンブリソース行

### 機能

- ・ 7900シリーズ用のニーモニックを記述できます。

### 記述規則

- ・ ニーモニックと、そのオペランドの間には必ずスペース又はタブを記述してください。
- ・ オペランドを複数個記述する場合は、オペランドとオペランドの間には必ずカンマ(,)を記述してください。
- ・ オペランドとカンマの間には、スペース又はタブを記述できます。
- ・ ニーモニックによっては、オペランドを記述しないものがあります。
- ・ ニーモニックは行の先頭から記述できます。
- ・ アセンブリソース行の先頭には、スペース又はタブを記述できます。
- ・ アセンブリソース行でラベルを定義する場合は、必ずニーモニックよりも前の桁にラベル名を記述してください。
- ・ ラベル定義のラベル名の直後には必ずコロンを記述してください。
- ・ ラベル名とニーモニックの間には、スペース又はタブを記述できます。
- ・ アセンブリソース行にコメントを記述する場合は、ニーモニックとオペランドのつぎに、セミコロン(;)を記述し、それ以降の桁にコメントを記述してください。
- ・ コメント行は、アセンブラリストファイルに出力されます。

### 注意事項

AS79は、セミコロン(;)以降の桁に記述したニーモニックについて、コードを生成しません。セミコロン(;)の記述位置には注意してください。AS79は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

- ・ ニーモニックのオペランドとコメントの間にはスペース又はタブを記述できます。

### 記述例

```
      MOVW.W   sym,#0
      RTS
main:  MOVW.W   sym,#0
      RTS
```

## 12.5.5 構造化記述行

### 機能

- ・ 7900シリーズ用の構造化記述命令を記述できます。

### 記述規則

- ・ 構造化記述命令と、その条件式の間には必ずスペース又はタブを記述してください。
- ・ 構造化記述命令によっては、オペランドを記述しないものがあります。
- ・ 構造化記述命令は行の先頭から記述できます。
- ・ 構造化記述命令行の先頭には、スペース又はタブを記述できます。
- ・ 構造化記述命令行でラベルを定義する場合は、必ず構造化記述命令よりも前の桁にラベル名を記述してください。
- ・ ラベル定義のラベル名の直後には必ずコロンを記述してください。
- ・ ラベル名と構造化記述命令の間には、スペース又はタブを記述できます。
- ・ 構造化記述命令行にコメントを記述する場合は、構造化記述命令と条件式のつぎに、セミコロン(;)を記述し、それ以降の桁にコメントを記述してください。
- ・ コメント行は、アセンブラリストファイルに出力されます。

### 注意事項

AS79は、セミコロン(;)以降の桁に記述した構造化記述命令について、コードを生成しません。セミコロン(;)の記述位置には注意してください。AS79は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

- ・ 構造化記述命令の条件式とコメントの間にはスペース又はタブを記述できます。

### 記述例

```
        IF  [sym] == 10
        :
        ENDIF
LAB:    IF  [sym] == 10
        :
        ENDIF
```

## 12.5.6 ラベル定義行

### 機能

- ・ 任意の名前を記述できます。

### 記述規則

- ・ ラベル名の直後には必ずコロン(:)を記述してください。
- ・ ラベル名とコロン(:)の間には、何も記述しないでください。
- ・ ラベル名は行の先頭から記述できます。
- ・ 行の先頭にはスペース又はタブを記述できます。
- ・ ラベル定義行にコメントを記述する場合は、コロン(:)以降の桁にセミコロン(;)を記述し、それ以降の桁にコメントを記述してください。
- ・ コメントは、アセンブラリストファイルに出力されます。

### 注意事項

---

AS79は、ダブルクォーテーション(")又はシングルクォーテーション(')で囲まれたセミコロン(;)は、コメントの先頭文字と判断しません。

---

- ・ ラベルとコメントの間にはスペース又はタブを記述できます。

### 記述例

```
start:
rabel:      .BLKB    1
main:       nop
loop:
```



## 12.5.7 コメント行

### 機能

- ・ 任意の文字列を記述できます。

### 記述規則

- ・ コメントの先頭には必ずセミコロン(;)を記述してください。
- ・ コメント行の先頭にはスペース又はタブを記述できます。
- ・ コメントには、全ての文字を記述できます。

### 記述例

```
    ;Comment line
    MOVW.W    sym,#0 ;Comment
```

## 12.5.8 空行

### 機能

- ・ 見かけ上なにも記述していない行です。

### 記述規則

- ・ ソースプログラムの可読性を向上するなどの必要に応じて、文字を含まない行を記述できます。
- ・ 空行には、スペース、タブ、リターン及びラインフィード文字以外は記述できません。

### 記述例

```
loop:
:
    JMP loop

    JSR sub1
```

## 12.6 オペランド

ニーモニック、構造化記述命令及び指示命令には、その命令の制御の対象を示す、オペランドが記述できます。オペランドには、次に示す種類があります。

### 注意事項

---

オペランドを持たない命令もあります。オペランドの有無については、各命令の記述規則を参照してください。

---

### 数値

数値には、整数と、浮動小数点数が含まれます。

### 名前

ラベル名及びシンボル名が記述できます。

### 式

数値及び名前を項に持つ式が記述できます。

### 文字列

文字又は文字列をASCIIコードとして扱えます。

### 12.6.1 オペランドの記述位置

オペランドと、オペランドをもつ命令との間に、必ずスペース又はタブを記述してください。

各オペランドの記述規則を説明します。

### 12.6.2 数値の記述規則

数値には、整数と浮動小数点数が含まれます。

## 12.6.3 整数

整数は、10進数、16進数、2進数及び8進数で記述できます。それぞれの記述方法と記述例を次に示します。

### 2進数

0と1で記述し末尾にBまたはbを記述します。

```
1001001B  
1001001b
```

### 8進数

0～7の数字で記述し末尾にOまたはoを記述します。

```
60702O  
60702o
```

### 10進数

0～9までの数字で記述します。

```
9423
```

### 16進数

0～9、a～fまたはA～Fの数字とアルファベットで記述し、末尾にHまたはhを記述します。ただし、アルファベットで始まる数値の場合は、先頭に0（ゼロ）を記述します。

```
0A5FH  
5FH  
0a5fh
```

## 12.6.4 浮動小数点数

浮動小数点数で表される次の範囲の値を記述できます。

FLOAT(32ビット長)

$$1.17549435 \times 10^{-38} \sim 3.40282347 \times 10^{38}$$

DOUBLE(64ビット長)

$$2.2250738585072014 \times 10^{-308} \sim 1.7976931348623157 \times 10^{308}$$

ソース記述	演算式
3.4E35	$3.4 \times 10^{35}$
3.4e-35	$3.4 \times 10^{-35}$
-.5E20	$-0.5 \times 10^{20}$
5e-20	$5.0 \times 10^{-20}$

### 注意事項

浮動小数点数は、指示命令".DOUBLE",".FLOAT"のオペランドだけに記述できます。

## 12.6.5 式の記述規則

数値、名前及び演算子を組み合わせた式を記述できます。

- ・ 演算子と数値の間には、スペース又はタブを記述できます。
- ・ 演算子は複数組み合わせて記述できます。
- ・ シンボル値として式を記述する場合は、式の値がアセンブル時に確定するように式を記述してください。
- ・ 式の演算結果の値の範囲は、 - 2147483648 ~ 2147483647となります。

### 注意事項

浮動小数点数は、式に記述できません。  
式の項に文字定数は使用できません。

## 12.7 演算子

### 12.7.1 演算子一覧

#### 単項演算子

演算子	機能
+	続く値を正の値として扱います。
-	続く値を負の値として扱います。
~	続く値の論理否定値を扱います。
sizeof	オペランドに指定したセクションのサイズ(バイト数)を値として扱います。
offsetof	オペランドに指定したセクションの開始アドレスを値として扱います。
BANK	ラベルの上位8ビットまたはシンボルの上位17~24ビットを切り出します。
OFFSET	ラベルまたはシンボルの下位16ビットを切り出します。

#### 二項演算子

演算子	機能
+	左辺値と右辺値を加算します。
-	左辺値から右辺値を減算します。
*	左辺値と右辺値を乗算します。
/	左辺値を右辺値で除算します。
%	左辺値を右辺値で割った余りを扱います。
>>	左辺値を右辺値回右へビットシフトします。
<<	左辺値を右辺値回左へビットシフトします。
&	左辺値と右辺値のビット毎の論理積値を扱います。
	左辺値と右辺値のビット毎の論理和値を扱います。
^	左辺値と右辺値のビット毎の排他的論理和値を扱います。

## プログラムの記述規則

### 条件演算子

演算子	機能
>	左辺値が右辺値より大きいことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。
<	右辺値が左辺値より大きいことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。
> =	左辺値が右辺値より大きいか等しいことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。
< =	右辺値が左辺値より大きいか等しいことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。
= =	左辺値と右辺値が等しいことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。
! =	左辺値と右辺値が等しくないことを評価します（指示命令 <code>.IF</code> 、 <code>.ELIF</code> のオペランドだけに記述できます）。

### 演算優先順位変更演算子

演算子	機能
( )	( ) で囲った演算を最優先で行います。一つの式に複数の ( ) が記述されている場合は、左が優先になります。( ) はネストした記述ができます。

### 注意事項

演算子`"SIZEOF"`、`"STARTOF"`及び`"BANK"`、`"OFFSET"`は、オペランドとの間に、必ずスペース又はタブを記述してください。

## 12.7.2 式の演算優先順位

AS79は、記述されている式について、次に示す優先順位に従って、演算を行った結果の数値をオペランドの値として扱います。また、構造化記述命令の条件式も以下の優先順位に従って処理します。構造化記述命令については後述します。

- 1 演算子が持つ優先順位の高いものから演算します。演算子の優先順位を次の表に示します。表の優先順位の欄の値の小さいものほど優先順位は高くなります。
- 2 同一の優先順位を持つ演算子は、左から順に演算を行います。
- 3 ( ) で囲むことで演算の優先順位を変更できます。

### 注意事項

( ) は、インダイレクトアドレッシングモード指定時の括弧と混同しないようご注意ください。

優先順位	演算子の種類	演算子
1	演算順位変更演算子	(, )
2	単項演算子	+, -, ~,   (絶対値), + +, SIZEOF, STARTOF, BANK, OFFSET
3	二項演算子1	*, /, %
4	二項演算子2	+, -
5	二項演算子3	> >, < <, > > .A, > > .R, < < .R
6	二項演算子4	&
7	二項演算子5	, ^
8	二項演算子6	- =
9	条件演算子	>, <, > =, < =, = =, ! =
10	論理積	& &
11	論理和	
12	代入演算子	=, = .S, = .Z

### 12.7.3 式の演算結果

式の記述例と、AS79が演算を行った結果の値について、次に例を示します。

式	演算結果
$2+6/2$	5
$(2+6)/2$	4
$1<<3+1$	16
$(1<<3)+1$	9
$3*2\%4/2$	1
$(3*2)\%(4/2)$	0
$8!4/2$	10
$(8!4)/2$	6
$8\&8/2$	0
$(8\&8)/2$	4
$6*-3$	-18
$-(6*-3)$	18
$-6*-3$	18

### 12.7.4 文字列の記述規則

一部の指示命令のオペランドに文字列が記述できます。文字列には、7ビット長ASCIIコードの文字が記述できます。

オペランドに文字列を記述する際には、特に指定のある場合を除いて、シングル又はダブルクォーテーションで囲って記述してください。

例)

```
"string"
'string'
```



## 第13章 7900のアドレッシングモード

---

7900シリーズの命令は1つ以上のアドレッシングモードをもっています。アドレッシングモードを使い分けることによってメモリ効率を向上することができます。

7900シリーズのアドレッシングモードを大きく分類すると次の3つになります。

- ダイレクトアドレッシングモード
- アブソリュートアドレッシングモード
- アブソリュートロングアドレッシングモード

### 注意事項

---

アドレッシングモードの詳細については、「7900シリーズ ソフトウェアマニュアル」を参照してください。

---

### 13.1 ダイレクトアドレッシングモード

アクセスするメモリの番地をマイクロコンピュータのダイレクトページレジスタ(DPR)の値を基準として、その番地からの距離(オフセット)を8ビットまたは6ビットの値で指定します。

- バンク0(0000H - 0FFFFH)の範囲でのみ使用できます。

#### 13.1.1 ダイレクトページレジスタのモード

7900のダイレクトページレジスタには2つのモードがあります。アプリケーションプログラムの実行中に、2つのモードを切り替えて使用することはできません。したがって、プログラムの記述を始める前にダイレクトページレジスタをどちらのモードで使用するかを決めておきます。

##### 8ビットオフセットモード

ダイレクトページレジスタのDPR0だけを使用します。DPR0に設定されている値を基準の番地として、その番地からの距離(オフセット)を指定します。

- 指定できるオフセットの範囲は0~0FFH(8ビット)です。

##### 6ビットオフセットモード

ダイレクトページレジスタ、DPR0、DPR1、DPR2、DPR3の4本の各ダイレクトページレジスタに設定されている値を基準の番地として、その番地からの距離(オフセット)を指定します。

- 指定できるオフセットの範囲は0~3FH(6ビット)です。

### 注意事項

---

アセンブラは、デフォルトでは6ビットオフセットモードを選択して処理します。8ビットオフセットモードを使用する場合は、必ずas79コマンドのオプション-M8を指定してください。

---

## 13.2 アブソリュートアドレッシングモード

データバンクレジスタ(DT)を基準としてアクセスするメモリまでの距離(オフセット値)を指定します。

- ・ DTの値は番地の上位8ビットを表します。DTの値が00Hならば000000H番地、DTの値が01Hならば010000H番地が基準となります。
- ・ オペランドに記述できる値の範囲は0～0FFFFFFHです。

## 13.3 アブソリュートロングアドレッシングモード

アクセスする番地を24ビット値で直接指定します。全メモリ空間に対して使用できます。

- ・ オペランドに記述できる値の範囲は、0～0FFFFFFHです。

# 第14章 アドレッシングモードの指定と最適化

## 14.1 アドレッシングモードの指定方法

ソースプログラム中でアドレッシングモードを指定するには、次の方法があります。

命令のオペランド毎にアドレッシングモードを明記する方法

シンボル名、ラベル名毎にアドレッシングモードを指定する方法

### 14.1.1 命令のオペランド毎にアドレッシングモードを明記する方法

命令のオペランドに以下に示すアドレッシングモード指定子を記述します。

- ・ この方法で指定したアドレッシングモードで必ずコードが生成されます。

アドレッシングモード	
指定子	機能
DP:	8ビットオフセットモードでデータレクトアドレッシングモードを指定
DP0:	6ビットオフセットモードでデータレクトアドレッシングモードを指定
DP1:	
DP2:	
DP3:	
DT:	アドレス絶対アドレッシングモードを指定
LG:	アドレス絶対ロングアドレッシングモードを指定

#### アドレッシングモード指定子の記述例

```
LDA A,DP:label
LDA A,DT:label
LDA A,LG:label
```

### 14.1.2 シンボル名、ラベル名毎にアドレッシングモードを指定する方法

シンボル名またはラベル名毎にアドレッシングモードを宣言します。

- ・ 指示命令で宣言したアドレッシングモードとアドレッシングモード指定子によるアドレッシングモードの指定に矛盾がある場合は、アドレッシングモード指定子による指定が優先となります。

指示命令	機能
.DPSYM	8ビットオフセットモードでデータアドレッシングモードを指定
.DP0SYM	6ビットオフセットモードでデータアドレッシングモードを指定
.DP1SYM	
.DP2SYM	
.DP3SYM	
.DTSYM	アドレスアドレッシングモードを指定
.LGSYM	アドレスロングアドレッシングモードを指定

#### アドレッシングモードの宣言例

```
.DPSYM label  
.DTSYM alabel  
.LGSYM allabel
```

## 14.2 ダイレクトアドレッシングモードを指定する

ダイレクトアドレッシングモードを指定するには次の方法があります。

- アドレッシングモード指定子を記述する
- 指示命令でダイレクトアドレッシングモードを宣言する

### 14.2.1 アドレッシングモード指定子を記述する

命令のオペランドにアドレッシングモード指定子を付加して記述します。

- ・ オフセット値はアセンブラが計算します。命令のオペランドにはアクセスしたいメモリの番地をそのまま記述してください。

#### 注意事項

オフセット値がマイナスとなるような指定はできません。

指定子	機能
DP:	8ビットオフセットモード
DP0:	DPR0を基準にした6ビットオフセットモード
DP1:	DPR1を基準にした6ビットオフセットモード
DP2:	DPR2を基準にした6ビットオフセットモード
DP3:	DPR3を基準にした6ビットオフセットモード

#### 注意事項

ダイレクトアドレッシングモードの6ビットオフセットモードと8ビットオフセットモードを一つのアプリケーションプログラムで混在して使用することはできません。

#### 記述例

8ビットオフセットモードでDPRの値が80Hのとき、88H番地の内容をアキュムレータAにロードする。

```
.DP      80H
.SECTION program, CODE
LDA     A, DP:88H
```

### 14.2.2 指示命令でダイレクトアドレッシングモードを宣言する

ラベルまたはシンボルに対してダイレクトアドレッシングモードでコードを生成することを予め宣言しておきます。

- ・ 指示命令、.DPSYM, .DP0SYM, .DP1SYM, .DP2SYM, .DP3SYMで宣言します。
- ・ 宣言されたラベルまたはシンボルがオペランドに記述されていると、アセンブラはダイレクトアドレッシングモードでコードを生成します。オペランドのDP:などの指定子を省略できます。
- ・ 指示命令による宣言とアドレッシングモード指定子による宣言に矛盾がある場合は、後者が優先されます。

#### .注意事項

---

DPSYMによる宣言とその他の指示命令 (.DP0SYM ~ .DP3SYM) による宣言を混在させることはできません。

---

#### 記述例

8ビットオフセットモードでDPRの値が80Hのとき、88H番地の内容をアキュムレータAにロードする。

```
.SECTIONmem,DATA
.ORG      80H
work1:   .BLKB      8
work2:   .BLKB      1
.DPSYM   work1,work2
.DP      80H
.SECTIONprogram,CODE
LDA      A,work2
```

### 14.2.3 オペランドにオフセット値を記述する

ダイレクトアドレッシングモードのオフセット値を直接オペランドに記述する方法があります。

オフセット値の前に次の指定子を記述します。アセンブラはこれらのオペランドに対してダイレクトページレジスタからのオフセット値を計算しません。

指定子	機能
DP+:	8ビットオフセットモード
DP0+:	DPR0を基準にした6ビットオフセットモード
DP1+:	DPR1を基準にした6ビットオフセットモード
DP2+:	DPR2を基準にした6ビットオフセットモード
DP3+:	DPR3を基準にした6ビットオフセットモード

#### 記述例

8ビットオフセットモードでDPRの値が80Hのとき、88H番地の内容をアキュムレータAにロードする

```
.DP      80H
.SECTION program, CODE
LDA     A, DP+:08H
```

#### 注意事項

アセンブラは、デフォルトでは6ビットオフセットモードを選択して処理します。8ビットオフセットモードを使用する場合は、必ずas79コマンドのオプション-M8を指定してください。

## 14.3 アブソリュートアドレッシングモードを指定する

アブソリュートアドレッシングモードを指定するには次の方法があります。

アドレッシングモード指定子を記述する

指示命令でアブソリュートアドレッシングモードを宣言する

### 14.3.1 アドレッシングモード指定子を記述する

命令のオペランドにアドレッシングモード指定子(DT:)を付加して記述します。

- ・ オフセット値はアセンブラが計算します。命令のオペランドにはアクセスしたいメモリの番地をそのまま記述してください。

#### 注意事項

オフセット値がマイナスとなるような指定はできません。

#### 記述例

DTの値が01Hのとき、010080H番地の内容をアキュムレータAにロードする。

```
.DT      01H
.SECTION program, CODE
LDA      A, DT:10080H
```

### 14.3.2 指示命令でアブソリュートアドレッシングモードを宣言する

ラベルまたはシンボルに対してアブソリュートアドレッシングモードでコードを生成することを予め宣言しておきます。

- ・ 指示命令、.DTSYMで宣言します。
- ・ 宣言されたラベルまたはシンボルがオペランドに記述されていると、アセンブラはアブソリュートアドレッシングモードでコードを生成します。オペランドのDT:指定子を省略できます。
- ・ 指示命令による宣言とアドレッシングモード指定子による宣言に矛盾がある場合は、後者が優先されます。

#### 記述例

DTの値が01Hのとき、010080H番地の内容をアキュムレータAにロードする。

```
addr     .EQU     10080H
.DTSYM   addr
.DT      01H
.SECTION program, CODE
LDA      A, addr
```



### 14.3.3 オペランドにオフセット値を記述する

アブソリュートアドレッシングモードのオフセット値を直接オペランドに記述する方法があります。

オペランドの前にDT+:指定子を記述します。アセンブラはこれらのオペランドに対してDTからのオフセット値を計算しません。

#### 記述例

DTの値が01Hのとき、010080H番地の内容をアキュムレータAにロードする。

```
.DT      01H
.SECTION program, CODE
LDA      A, DT+:80H
```

## 14.4 アブソリュートロングアドレッシングモードを指定する

アブソリュートロングアドレッシングモードを指定するには次の方法があります。

- アドレッシングモード指定子を記述する
- 指示命令でアブソリュートロングアドレッシングモードを宣言する

### 14.4.1 アドレッシングモード指定子を記述する

命令のオペランドにアドレッシングモード指定子(LG:)を付加して記述します。

#### 記述例

010080H番地の内容をアキュムレータAにロードする。

```
.DT      01H
.SECTIONprogram, CODE
LDA      A, LG:10080H
```

### 14.4.2 指示命令でアブソリュートロングアドレッシングモードを宣言する

ラベルまたはシンボルに対してアブソリュートロングアドレッシングモードでコードを生成することを予め宣言しておきます。

- ・ 指示命令、.LGSYMで宣言します。
- ・ 宣言されたラベルまたはシンボルがオペランドに記述されていると、アセンブラはアブソリュートアドレッシングモードでコードを生成します。オペランドのLG:指定子を省略できます。
- ・ 指示命令による宣言とアドレッシングモード指定子による宣言に矛盾がある場合は、後者が優先されます。

#### 記述例

010080H番地の内容をアキュムレータAにロードする。

```
addr     .EQU     10080H
.LGSYM   addr
.DT      01H
.SECTIONprogram, CODE
LDA      A, addr
```

## 14.5 アドレッシングモード指定子を記述可能なアドレッシングモード

AS79アセンブラは、以下のアドレッシングモードに対してアドレッシングモード指定子を記述できます。

アドレッシングモード	指定子
ダイレクト	DP:
ダイレクト・インデクストX	
ダイレクト・インデクストY	
ダイレクト・インダイレクト	
ダイレクト・インデクストX・インダイレクト	
ダイレクト・インダイレクト・インデクストY	
ダイレクト・インダイレクトロング	
ダイレクト・インダイレクトロング・インデクストY	
アブソリュート	DT:
アブソリュート・インデクストX	
アブソリュート・インデクストY	
アブソリュートロング	LG:
アブソリュートロング・インデクストX	
ダイレクト・ビット・レラティブ	DP:
アブソリュート・ビット・レラティブ	DT:

## 14.6 アドレッシングモードの指定例とアセンブル結果

アドレッシングモードを指定して記述したプログラムをアセンブルした結果の例をリストファイルで示します。

- ・ リストファイルを生成するには、as79コマンドのオプション-lを指定します。

### 14.6.1 アドレッシングモード指定子の使用例(LIST1)

```

1          .DT      00H
2          .DP      10H
3 00000030h  sym     .EQU     30H
4          .SECTION          program
5 000000 2A20  ADD     A,DP:sym
6 000002 2B20  ADD     A,DP:sym,x
7 000004 112020 ADD     A,(DP:sym)
8 000007 112120 ADD     A,(DP:sym,x)
9 00000A 112820 ADD     A,(DP:sym),y
10 00000D 112220 ADD     A,L(DP:sym)
11 000010 112920 ADD     A,L(DP:sym),y
12 000013 2E3000 ADD     A,DT:sym
13 000016 2F3000 ADD     A,DT:sym,x
14 000019 11263000 ADD     A,DT:sym,y
15 00001D 112C300000 ADD     A,LG:sym
16 000022 112D300000 ADD     A,LG:sym,x
17 000027 414A30080007 BBS     #8,DP:40H,label
18 00002D 414E4000040000 BBS     #4,DT:40H,label
19 000034          label:
20          .END

```

## アドレッシングモードの指定と最適化

### 14.6.2 指示命令でアドレッシングモードを宣言したプログラム記述例(LIST2)

```
1          .DT      00H
2          .DP      10H
3  00000030h      sym      .EQU      30H
4  00000500h      sym_a    .EQU      500H
5  00010000h      sym_al   .EQU      10000H
6
7              .DPSYM  sym
8              .DTSYM  sym_a
9              .LGSYM  sym_al
10             .SECTION      program
11  000000  2A20      A  ADD      A, sym
12  000002  2B20      A  ADD      A, sym, x
13  000004  112020      ADD      A, (sym)
14  000007  112120      ADD      A, (sym, x)
15  00000A  112820      ADD      A, (sym), y
16  00000D  112220      ADD      A, L(sym)
17  000010  112920      ADD      A, L(sym), y
18  000013  2E0005      A  ADD      A, sym_a
19  000016  2F0005      A  ADD      A, sym_a, x
20  000019  11260005      ADD      A, sym_a, y
21  00001D  112C000001      ADD      A, sym_al
22  000022  112D000001      ADD      A, sym_al, x
23  000027  414A20080006      A  BBS      #8, sym, label
24  00002D  414A20040000      A  BBS      #4, sym, label
25  000033          label:
                .END
```

## 14.7 アセンブラによるアドレッシングモードの自動選択

アセンブラは、ソースプログラムにアドレッシングモードの指定がない場合は、命令が記述されている場所のダイレクトページレジスタおよびデータバンクレジスタの値を考慮して、プログラムサイズが小さくなるようにアドレッシングモードを選択します。

- ・アセンブラがアドレッシングモードを自動選択した場合は、リストファイルのソースの先頭に'A'を出力します。

### 14.7.1 アドレッシングモード指定のないソース記述例(LIST3)

```

1          .DT      00H
2          .DP      10H
3          .GLB     gsym
4 00000030h  sym     .EQU    30H
5          .SECTION          program
6 000000 2A20      A  ADD     A, sym
7 000002 2B20      A  ADD     A, sym, x
8 000004 112020    ADD     A, (sym)
9 000007 112120    ADD     A, (sym, x)
10 00000A 112820    ADD     A, (sym), y
11 00000D 112220    ADD     A, L(sym)
12 000010 112920    ADD     A, L(sym), y
13 000013 2A20      A  ADD     A, sym
14 000015 2B20      A  ADD     A, sym, x
15 000017 11263000  ADD     A, sym, y
16 00001B 2A20      A  ADD     A, sym
17 00001D 2B20      A  ADD     A, sym, x
18 00001F 414A3008000B  A  BBS     #8, 40H, label
19 000025 414A30040005  A  BBS     #4, 40H, label
20 00002B 112C000000r  ADD     A, gsym
21 000030          label:
22          .END

```

### 14.7.2 コマンドオプション-Aをつけてアセンブルした結果(LIST4)

```

1          .DT      00H
2          .DP      10H
3          .GLB     gsym
4 00000030h sym      .EQU    30H
5          .SECTION          program
6 000000 2A20      A  ADD    A, sym
7 000002 2B20      A  ADD    A, sym, x
8 000004 112020      ADD    A, (sym)
9 000007 112120      ADD    A, (sym, x)
10 00000A 112820     ADD    A, (sym), y
11 00000D 112220     ADD    A, L(sym)
12 000010 112920     ADD    A, L(sym), y
13 000013 2A20      A  ADD    A, sym
14 000015 2B20      A  ADD    A, sym, x
15 000017 11263000   ADD    A, sym, y
16 00001B 2A20      A  ADD    A, sym
17 00001D 2B20      A  ADD    A, sym, x
18 00001F 414A30080009 A  BBS    #8, 40H, label
19 000025 414A30040003 A  BBS    #4, 40H, label
20 00002B 2E0000r    A  ADD    A, gsym
21 00002E          label:
22          .END

```

アドレッシングモード指定のないソースプログラム（LIST3で示したのと同じソースプログラム）をas79コマンドのオプション-Aをつけてアセンブルすると、リロケータブルな外部参照ラベルに対してアブソリュートアドレッシングモードでコードを生成します（リストファイルの20行目）。

ただし、リンクした結果アブソリュートアドレッシングモードでアクセスできないアドレスに決定された場合は、リンクエラーが発生します。

## 14.8 特定のアドレッシングモードの選択を禁止する

アドレッシングモードが指定されていない命令に対してアセンブラが実行するアドレッシングモードの自動選択を禁止できます。

- ・ダイレクトアドレッシングモードとアブソリュートアドレッシングモードが禁止できます。
- ・アドレッシングモードが禁止されている区間であっても、アドレッシングモード指定子を記述した命令については、指定されたアドレッシングモードでコードを生成します。ただし、オペランドデータの値はオフセット値としてコードを生成します。

### 14.8.1 アドレッシングモード禁止の指定方法

ソース記述	機能
.DP OFF	ダイレクトアドレッシングモードの選択を禁止
.DP0 OFF	
.DP1 OFF	
.DP2 OFF	
.DP3 OFF	
.DT OFF	アブソリュートアドレッシングモードの選択を禁止

### 14.8.2 禁止したアドレッシングモードを有効にする

禁止したアドレッシングモードを同じソースファイル内で再度有効にするには、指示命令.DP, .DP0, .DP1, .DP2, .DP3, .DTでレジスタ値を宣言してください。



14.8.3 アドレッシングモードの禁止例(LIST5)

```

1          .DT      00H
2          .DP      30H
3          .GLB     gsym
4 00000030h sym      .EQU    30H
5          .SECTION      program
6          .DP      OFF
7 000000 2E3000      A ADD    A, sym
8 000003 2A30          ADD    A, DP: sym
9 000005 2F3000      A ADD    A, sym, x
10         .DT      OFF
11 000008 112C300000 ADD    A, sym
12 00000D 2E3000      ADD    A, DT: sym
13 000010 112D300000 ADD    A, sym, x
14         .DP      30H
15         .DT      00H
16 000015 2A00          A ADD    A, sym
17 000017 2B00          A ADD    A, sym, x
18 000019 11263000      ADD    A, sym, y
19 00001D 414A1008000B A BBS    #8, 40H, label
20 000023 414A10040005 A BBS    #4, 40H, label
21 000029 112C000000r      ADD    A, gsym
22 00002E          label:
23         .END

```

6行目

ダイレクトアドレッシングモードの選択を禁止

8行目

オフセット計算をしないでダイレクトアドレッシングモードでコード生成

10行目

アブソリュートアドレッシングモードの選択を禁止

12行目

symの値をオフセット値としてアブソリュートアドレッシングモードで  
コード生成

14,15行目

ダイレクトアドレッシングモード、アブソリュートアドレッシングモード  
選択許可

## 第15章 指示命令

---

AS79対応のソースプログラムには、7900シリーズのニーモニック、構造化記述命令以外に指示命令が記述できます。指示命令には、次の種類があります。

### アドレス制御指示命令

アセンブル実行時にアドレスを決定するための指示をします。

### アセンブル制御指示命令

as79の実行について指示をします。

### リンク制御指示命令

アドレス再配置制御のための情報を定義します。

### リスト制御指示命令

as79が生成するリストファイルのフォーマットを制御します。

### 条件アセンブル制御指示命令

アセンブル実行時に設定した条件によって、コード生成するブロックを選択します。

### 拡張機能指示命令

上記以外の制御を行う指示命令です。

### インスペクタ情報の出力制御指示命令

インスペクタ情報の出力を制御する指示命令です。

### マクロ指示命令

マクロ機能の定義及び展開を指示します。

### 7900シリーズ用ツールソフトウェアが出力する指示命令

これらの指示命令及びオペランドについては、全て7900シリーズ用ツールソフトウェアが出力します。

### 注意事項

---

7900シリーズ用ツールソフトウェアが出力する指示命令（ピリオドで始まる文字列のすべて）は、ユーザーはソースプログラムに記述できません。記述した場合の動作については保証しません。

---

## 15.1 指示命令一覧

指示命令の機能の概要を示します。

### アドレス制御

AS79がアドレスの更新を行う場合の指示をします。

#### 注意事項

---

絶対属性セクション内のアドレスをのぞいて、AS79が制御を行うアドレスはリロケータブル値です。

---

#### .ORG

本指示命令を記述した行以降の行の生成コードのアドレス値を指定します。本指示命令を記述したセクションは、絶対属性セクションとなります。

#### .BLKB

1 バイト単位でRAM領域を確保します。

#### .BLKW

2 バイト単位でRAM領域を確保します。

#### .BLKA

3 バイト単位でRAM領域を確保します。

#### .BLKD

4 バイト単位でRAM領域を確保します。

#### .BLKF

4 バイト単位でRAM領域を確保します。

#### .BLKDF

8 バイト単位でRAM領域を確保します。

#### .BYTE

1 バイト長のデータをROM領域に格納します。

#### .WORD

2 バイト長のデータをROM領域に格納します。

#### .ADDR

3 バイト長のデータをROM領域に格納します。

## AS79指示命令

### .DWORD

4 バイト長のデータをROM領域に格納します。

### .FLOAT

4 バイトで表される浮動小数点数データをROM領域に格納します。

### .DOUBLE

8 バイトで表される浮動小数点数データをROM領域に格納します。

### .ALIGN

ロケーションカウンタをワードまたはダブルワードアライメントに変換することを指示します。

## アセンブル制御

指示命令自身はデータを生成しません。命令に対する機械語コードの生成を制御する命令です。アドレスの更新は行いません。

### .EQU

シンボルを設定します。

### .BTEQU

ビットシンボルを設定します。

### .DEFINE

シンボルに文字列を定義します。

### .END

ソースプログラムの終了を指定します。

### .DP

ダイレクトページレジスタ値を宣言します。

### .DP0、.DP1、.DP2、.DP3

拡張ダイレクトページレジスタ値を宣言します。

### .DPSYM

オペランドに指定したシンボルをダイレクトアドレッシングモード（8ビットオフセットモード）としてコード生成します。

### .DP0SYM、.DP1SYM、.DP2SYM、.DP3SYM

オペランドに指定したシンボルをダイレクトアドレッシングモード（6ビットオフセットモード）としてコード生成します。

### .DT

データバンクレジスタ値を宣言します。

## AS79指示命令

### .DTSYM

オペランドに指定したシンボルをアブソリュートアドレッシングモードとしてコード生成します。

### .LGSYM

オペランドに指定したシンボルをアブソリュートロングアドレッシングモードとしてコード生成します。

### .INCLUDE

本指示命令を記述した位置に、指定したファイルの内容を読み込みます。

### .DATA

データ長選択フラグ(m)の既定値を宣言します。

### .INDEX

インデックスレジスタ長選択フラグ(x)の既定値を宣言します。

### .LENGTH

データ長選択フラグ(m)及びインデックスレジスタ長選択フラグ(x)の既定値を同時に宣言します。

## リンク制御

プログラムを複数のファイルに分割して記述するリロケータブルアセンブルを実行するための指示命令です。

### .SECTION

アドレスを再配置するための最小の単位となる領域を定義します。as79及びln79はセクション情報を基に、コード生成及びリロケータブルファイルの結合とセクションの再配置を行います。

### .GLB

シンボルが全てのファイルから参照可能（グローバル、外部シンボル）であることを宣言します。この宣言がされているシンボルは、任意の1つのファイルでその定義（.EQUによる値の指定や任意のセクションにそのラベルを記述する）がされていなければなりません。

一般に外部シンボルについては、as79はアドレスを保留します。このとき、その値はln79が決定します。

### .VER

オペランドに記述した文字列をln79が生成するマップファイルに、バージョン情報として出力します。この情報を利用して、リンクの制御ができます。

## AS79指示命令

### リスト制御

リストファイルに出力する情報や、リストファイルのフォーマットの制御を行います。コード生成には影響しません。

#### .LIST

リストファイルを生成する際に、ソースファイルの行単位でリストファイルへの出力を行うか行わないかを制御します。

#### .PAGE

リストファイルを生成する際に、ソースファイルの任意の位置でリストを改ページします。同時に任意のメッセージをヘッダ部分に出力します。

#### .FORM

リストファイルの1ページに出力する行数及び桁数を設定します。

### 条件アセンブル制御

AS79は、条件アセンブル指示命令を使って、指定した範囲の行のアセンブルを行うか、行わないかを指定できます。

#### .IF

条件アセンブルブロックの始まりを示します。条件の判定を行います。

#### .ELIF

二つ以上の条件ブロックを記述する場合に、二つ目以降の条件を判定します。

#### .ELSE

全ての条件が偽である場合に、アセンブルを行うブロックの始まりを示します。

#### .ENDIF

条件アセンブルブロックの終了を示します。

### 拡張機能指示命令

これらの指示命令は、コードを生成しません。

#### .ASSERT

オペランドに記述した文字列を標準エラー出力又はファイルに出力します。

#### ?

テンポラリラベルの定義と参照を指定します。

#### ..FILE

AS79が処理を行っているソースファイル名を示します。

#### @

@の前後の文字列を連結し、一つの文字列として扱います。

### インスペクタ情報出力制御指示命令

インスペクタ情報の出力を制御します。

INSF

インスペクタ情報の関数（サブルーチン）開始情報を定義します。

.EINSF

インスペクタ情報の関数（サブルーチン）終了情報を定義します。

.CALL

インスペクタ情報の関数（サブルーチン）呼び出し先情報を定義します。

.STK

インスペクタ情報のスタック情報を定義します。

### マクロ指示命令

AS79のマクロ命令には次のものがあります。

.MACRO

マクロ名を定義し、マクロボディの始まりを示します。

.EXITM

マクロボディの展開を中止します。

.LOCAL

マクロ内ローカルラベルを宣言します。

.ENDM

マクロボディの終了を示します。

.MREPEAT

繰り返しマクロボディの始まりを示します。

.ENDR

繰り返しマクロボディの終了を示します。

..MACPARA

マクロ呼び出しの実引数の個数を持ちます。

..MACREP

繰り返しマクロボディの展開数を持ちます。

..MACREPZ

.MACREPから1を引いた値を持ちます。

.LEN

指定した文字列の文字列長を示します。

.INSTR

指定した文字列のなかの指定した文字列の開始位置を示します。

.SUBSTR

指定した文字列の指定した位置から指定した文字数を切り出します。

## 15.2 指示命令記述方法

指示命令の記述方法を各命令毎に示します。指示命令はアルファベット順に並べられています。



## ..**FILE**

---

### 機能

- ・ AS79が処理中のファイル名に展開されます（ソースファイル又はインクルードファイル）。

### 注意事項

本指示命令で読み込まれるファイル名は、ファイルの拡張子及びパスを除いた部分です。

コマンドオプション"-F"を指定すると、"..FILE"は、コマンド行で指定したソースファイル名に固定されます。オプションを指定しない場合は、"..FILE"が記述されているファイル名を示します。

### 記述形式

..**FILE**

### 記述規則

- ・ 指示命令".ASSERT"及び指示命令".INCLUDE"のオペランドに記述できます。

### 記述例

```
.ASSERT "sample" > ..FILE
```

ソースファイル名が"sample.a79" の場合、"sample" ファイルにメッセージを出力します。

```
.INCLUDE ..FILE@.inc
```

ソースファイル名が"sample.a79" の場合、"sample.inc" ファイルをインクルードします。

```
.INCLUDE "sample" > ..FILE@.mes
```

上記の行が、"sample.a79" ファイルでインクルードしている"incl.inc" 内に記述されている場合、通常、"incl.mes" に文字列を出力します。

コマンドオプション(-F) を指定している場合は、"sample.mes" ファイルに文字列を出力します。

## ..

### 機能

- ・ マクロ呼び出しの実引数の個数を示します。
- ・ ".MACRO"によるマクロ定義のボディ内に記述できます。

### 注意事項

マクロボディの外には記述できません。

### 記述形式

```
..
```

### 記述規則

- ・ 本指示命令は式の項として記述できます。

### 記述例

- ・ マクロ実引数の数を判断して、条件アセンブルを行う。

#### マクロ定義

```
.GLB      mem
name      .MACRO  f1,f2
.IF      ..MACPARA == 2
        ADDM    f1,f2
.ELSE
        ADD     A,f1
.ENDIF
        .ENDM
```

#### マクロ呼び出し

```
name      mem
```

#### マクロ展開

```
.ELSE
        ADD     A,mem
.ENDIF
.ENDM
```

## ..

### 機能

- ・ 繰り返しマクロが展開されている回数を示します。
- ・ ..MACREPZは、..- ・ ".MREPEAT"によるマクロ定義のボディ内に記述できます。

### 注意事項

マクロボディの外には記述できません。

- ・ 条件アセンブルのオペランドに記述できます。

### 記述形式

```
..

```

### 記述規則

- ・ 本指示命令は式の項として記述できます。

### 記述例

#### マクロ定義

```
mclr      .GLB      mem
          .MACRO   value,name
          .MREPEATvalue
          MOVW.W   name+..

```

#### マクロ呼び出し

```
mclr      3,mem
```

#### マクロ展開

```
.MREPEAT 3
MOVW.W   mem+1,#0
MOVW.W   mem+2,#0
MOVW.W   mem+3,#0
. ENDR
. ENDM
```

## .ADDR

### 機能

- ・ 3バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。

### 記述形式

```
.ADDR      ( 数値 )
( 名前: ) .ADDR      ( 数値 )
```

### 記述規則

- ・ オペランドに整数値を記述してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ 複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
- ・ オペランドにはクォーテーション(')又は、ダブルクォーテーション(")で囲って、文字又は、文字列を記述できます。このとき格納されるデータは、文字のASCIIコードになります。

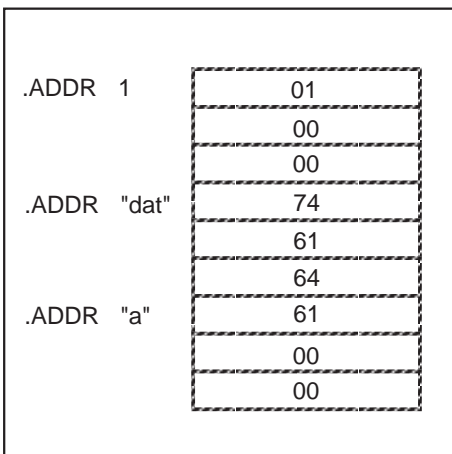
### 注意事項

**オペランドに文字列を記述する場合、有効な文字数は3文字です。3文字を超える場合はワーニングとなります。**

- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```
.SECTIONname,ROMDATA
.ADDR      1
.ADDR      "dat", "a"
.ADDR      symbol
.ADDR      symbol+1
.ADDR      1,2,3,4,5
.END
```



## .ALIGN

### 機能

- ・ 本指示命令を記述した直後の行のコードを格納するアドレスをワードまたはダブルワードアライメントに補正します。
- ・ セクションタイプがCODE又は、ROMDATAの場合は、アドレスを補正した結果、空になったところにNOPのコード(74H)を書き込みます。
- ・ セクションタイプがDATAの場合は、アドレス値を+nします。

.SECTION .ALIGN指示 動作内容  
による指定 命令の指定

なし	なし	セクションのアライメント補正は行わない。
なし	2	.ALIGN指定行でワードアライメントに補正される。
なし	4	.ALIGN指定行でダブルワードアライメントに補正される。
ALIGN=2	なし	セクションの先頭がワードアライメントにマッピングされる。
ALIGN=2	2	セクションの先頭がワードアライメントにマッピングされ、各.ALIGN 指定行でワードアライメントに補正される。
ALIGN=4	なし	セクションの先頭がダブルワードアライメントにマッピングされる。
ALIGN=4	4	セクションの先頭がダブルワードアライメントにマッピングされ、各.ALIGN 指定行でダブルワードアライメントに補正される。

### 記述形式

.ALIGN [2|4]

### 記述規則

- ・ 本指示命令は、次の条件に当てはまるセクション内に記述できます。
- 1 セクション定義の際にアドレス補正を指示している相対属性セクション。

```
.SECTION program, CODE, ALIGN
```

#### 2 絶対属性セクション

```
.SECTION program, CODE  
.ORG 0e000H
```

- ・ 相対属性セクションで、.SECTION指示命令行でALIGN指定のされていないセクションに本指示命令を記述した場合は、ワーニングが出力されます。

## AS79指示命令

### 記述例1

```
.SECTION area, DATA, ALIGN=2
MEM1: .BLKW 2
MEM2: .BLKB 1
      .ALIGN 2
      .END
```

### 記述例2

```
.SECTION program, CODE, ALIGN=2
.BYTE 01H
.ALIGN 2
.END
```

ソース	アドレス,コード
.SECTION count,ROMDATA,ALIGN=2	
.ADDR 1	00000 010000
.ALIGN 2	00003 74 NOPコードを挿入
.SECTION ram,DATA,ALIGN=2	
.BLKA 1	00000
.ALIGN 2	00003 アドレスを+1
.BLKB 1	00004
.END	

## .ASSERT

---

### 機能

- ・ オペランドに記述した文字列をアセンブル実行時に、標準エラー出力に出力します。
- ・ ファイル名を指定した場合は、オペランドに記述した文字列をファイルに出力します。
- ・ ファイル名にディレクトリ指定がない場合はカレントディレクトリにファイルを生成します。

### 記述形式

```
.ASSERT "(文字列)"  
.ASSERT "(文字列)" > (ファイル名)  
.ASSERT "(文字列)" >> (ファイル名)
```

### 記述規則

- ・ オペランドと指示命令の間には、必ずスペース又はタブを記述してください。
- ・ オペランドの文字列は必ずダブルクォーテーションで囲ってください。
- ・ 文字列をファイルに出力するときは、">"又は">>"に続けてファイル名を指定してください。
- ・ > は、新規にファイルを生成して、そのファイルにメッセージを出力します。以前に同名のファイルがある場合は、そのファイルに上書きされます。
- ・ >> は、ファイルの内容に追加して、メッセージを出力します。指定したファイルが存在しない場合は、新しくファイルを生成します。
- ・ ">"又は">>"の前後には、スペース又はタブを記述できます。
- ・ ファイル名に指示命令"..FILE"を記述できます。

### 記述例

```
.ASSERT "string" > sample.dat
```

sample.dat ファイルにメッセージを出力します。

```
.ASSERT "string" >> sample.dat
```

sample.dat ファイルにメッセージを追加します。

```
.ASSERT "string" > ..FILE
```

現在処理中のファイルと同じ名前でも拡張子を除くファイル名のファイルにメッセージを出力します。

## .BLKA

---

### 機能

- ・ 3バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。

### 記述形式

```
                .BLKA    ( 数値 )  
(名前:) .BLKA    ( 数値 )
```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロンの(:)を記述してください。

### 記述例

```
symbol      .EQU      1  
            .SECTIONarea,DATA  
work1:      .BLKA      1  
work2:      .BLKA      symbol  
            .BLKA      symbol+1
```



## .BLKB

### 機能

- ・ 1バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。
- ・ 本指示命令で定義されたラベルを構造化記述命令の式中に記述した場合、式のサイズをバイトサイズに指定します。

### 記述形式

```

                .BLKB    ( 数値 )
(名前:) .BLKB    ( 数値 )

```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

### 記述例

```

symbol      .EQU      1
            .SECTIONarea,DATA
work1:      .BLKB      1
work2:      .BLKB      symbol
            .BLKB      symbol+1

```

## .BLKD

### 機能

- ・ 4バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。
- ・ 本指示命令にて定義されたラベルを構造化記述命令の式中に記述した場合、式のサイズをダブルワードサイズに指定します。

### 記述形式

```

                .BLKD    ( 数値 )
(名前:) .BLKD    ( 数値 )

```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

### 記述例

```

symbol      .EQU      1
             .SECTIONarea,DATA
work1:      .BLKD      1
work2:      .BLKD      symbol
             .BLKD      symbol+1

```

## .BLKDF

---

### 機能

- ・ 8バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。
- ・ 本指示命令は、倍精度浮動小数点を用いる場合に使用します。

### 記述形式

```
.BLKDF    ( 数値 )  
( 名前: ) .BLKDF    ( 数値 )
```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

### 記述例

```
symbol    .EQU    1  
          .SECTIONarea,DATA  
work1:    .BLKDF  1  
work2:    .BLKDF  symbol  
          .BLKDF  symbol+1
```

## .BLKF

### 機能

- ・ 4バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。
- ・ 本指示命令は、単精度浮動小数点を用いる場合に使用します。

### 記述形式

```
.BLKF      ( 数値 )
( 名前: ) .BLKF      ( 数値 )
```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

### 記述例

```
symbol      .EQU      1
             .SECTIONarea,DATA
work1:      .BLKF      1
work2:      .BLKF      symbol
             .BLKF      symbol+1
```

## .BLKW

### 機能

- ・ 2バイト単位で、指定したバイト数のRAM領域を確保します。
- ・ 確保したRAMのアドレスに、ラベル名を定義することもできます。
- ・ 本指示命令にて定義されたラベルを構造化記述命令の式中に記述した場合、式のサイズをワードサイズに指定します。

### 記述形式

```

                .BLKW    ( 数値 )
(名前:) .BLKW    ( 数値 )

```

### 記述規則

- ・ 本指示命令は必ず、DATAタイプのセクション内に記述してください。セクション定義の際に、セクション名に続けて",DATA"を記述することでセクションタイプがDATAタイプとなります。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに整数値を記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ オペランドの式の値は、アセンブル実行時に確定しなければなりません。
- ・ 領域にラベル名を定義する場合は、指示命令の前にラベル名を記述してください。ラベル名には、必ずコロン(:)を記述してください。

### 記述例

```

symbol      .EQU      1
             .SECTIONarea,DATA
work1:      .BLKW      1
work2:      .BLKW      symbol
             .BLKW      symbol+1

```

## .BTEQU

### 機能

- ・ ビット位置と配置アドレスを定義します。本指示命令で定義したシンボルをビットシンボルと呼びます。
- ・ 本指示命令でビットシンボルを定義することで、ビット命令のオペランドにビットシンボルを記述できます。

### 注意事項

**ビットシンボルの前方参照はできません。**

- ・ シンボリックデバッグでビットシンボルを使用できます。
- ・ ビットシンボルは、グローバル指定できません。従ってローカルシンボルとしてのみ使用可能です。

### 記述形式

(名前) .BTEQU (ビット位置), (配置アドレス値)

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ ビット位置と、そのビットのメモリアドレスをカンマで区切って記述してください。
- ・ 必ず、ビット位置を第一オペランドに記述してください。
- ・ ビット位置を示す数値は、データ長宣言が8ビットの場合は0~7の範囲の整数値が、16ビットの場合は0~15の範囲の整数値が記述できます。
- ・ ビット位置は、必ずアセンブル実行時に確定する値を指定してください。
- ・ オペランドにはシンボルを記述できます。

### 注意事項

**前方参照となるシンボル名は記述できません。**

- ・ ビットシンボルの再定義はできません。
- ・ ビットシンボルを使用する際は、本指示命令で定義してから参照してください。

### 記述例

```
bit0    .btequ  0,0
bit1    .btequ  1,flag
bit2    .btequ  one,flag
```

## .BYTE

### 機能

- ・ 1バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。
- ・ 本指示命令で定義されたラベルを構造化記述式中に記述した場合、式のサイズをバイトサイズに指定します。

### 記述形式

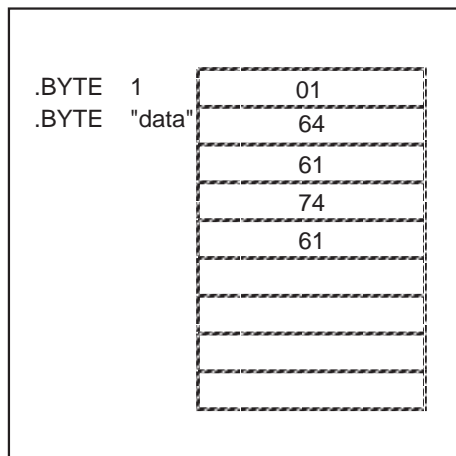
```
.BYTE      ( 数値 )
( 名前: ) .BYTE      ( 数値 )
```

### 記述規則

- ・ オペランドに整数値を記述してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ 複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
- ・ オペランドにはクォーテーション(')又は、ダブルクォーテーション(")で囲って、文字又は、文字列を記述できます。このとき格納されるデータは、文字のASCIIコードになります。
- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```
.SECTION name , ROMDATA
.BYTE 1
.BYTE "data"
.BYTE symbol
.BYTE symbol+1
.BYTE 1,2,3,4,5
.END
```



## .CALL

---

### 機能

- ・ インспекタ情報の関数（サブルーチン）呼び出し先情報を定義します。

### 記述形式

.CALL （呼び出し先関数（サブルーチン）名），（記憶クラス）

### 記述規則

- ・ 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- ・ 呼び出し先関数（サブルーチン）名および記憶クラスは必ず記述してください。
- ・ 記憶クラスを記述する場合は、カンマで区切って記述してください。
- ・ 記憶クラスは 'G（グローバルラベル）'、'S（ローカルラベル）' のいずれかを記述してください。

### 注意事項

---

本指示命令は、インспекタ情報の関数開始情報と関数終了情報の範囲内で記述してください。

本指示命令は、コマンドオプション"- finfo" が指定された場合に有効となります。

---

### 記述例

```
.INSF  glbfunc, G, 0
      :
jsr   glbsub
      .CALL  glbsub, G
      :
jsr   locsub
      .CALL  locsub, S
      :
.EINSF
```



## .DATA

---

### 機能

- CPU内部のデータ長(8または16)を示します。
- 本指示命令は、下記の事項に影響を及ぼします。
  - (1)mフラグの状態により生成コードの内容が異なる命令に対するアセンブル処理
  - (2)構造化記述命令に対するコード生成

### 記述形式

.DATA (数値)

### 記述規則

- 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- オペランドの数値が8の時は8ビット長、16の時は16ビット長を示します。
- SEM,CLM命令等でデータ長を変更する場合は、本指示命令の宣言も同時に行ってください。
- 他のデータ長宣言(サイズの決定方法参照)と本指示命令による指定が異なった場合は、他のデータ長指定に従ってアセンブルします。

### 注意事項

本指示命令はアセンブラに対してデータ長宣言をする命令であり、実際のデータ長を設定できるものではありません。実際にデータ長を設定するためには、本指示命令の直前または直後にデータ長操作命令を記述して下さい。

### 記述例

```
.DATA      16
CLM
ADC      A,#symbol    ;16bits length
```

## .DEFINE

---

### 機能

- ・ シンボルに文字列を定義します。
- ・ シンボルは再定義が可能です。

### 注意事項

本指示命令で定義されたシンボルは、外部参照指定ができません。

### 記述形式

- (シンボル名) .DEFINE (文字列)
- (シンボル名) .DEFINE '(文字列)'
- (シンボル名) .DEFINE "(文字列)"

### 記述規則

- ・ スペースまたはタブを含む文字列を定義する場合は、必ずシングルクォーテーション(')または、ダブルクォーテーション(")で囲って記述してください。

### 記述例

```
.SECTIONram,DATA
data1: .BLKB 1
flag .DEFINE "#01H,data1"
.SECTIONprogram
CLB flag
```

## .DOUBLE

### 機能

- ・ 8バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。

### 記述形式

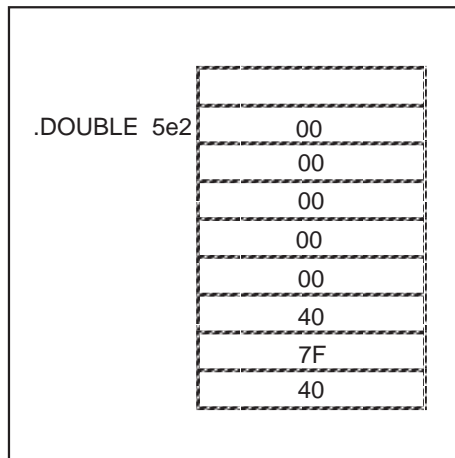
.DOUBLE (数値)  
 (名前:) .DOUBLE (数値)

### 記述規則

- ・ オペランドに浮動小数点数を記述してください。
- ・ 浮動小数点数の記述方法は、「オペランドの記述規則」を参照してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```
.DOUBLE 5E2  
constant: .DOUBLE 5e2
```



## .DP[0|1|2|3]

### 機能

- ・ダイレクトページレジスタ値を宣言します。
- ・アセンブル実行時にダイレクトページレジスタの値を本指示命令で定義した値であると判断し、以降のコードを生成します。
- ・以降の行で、ダイレクトページアドレッシングモードを指定できます。
- ・指示命令".DP[n]SYM"で指定されたシンボル名を用いた命令に対して、ダイレクトページアドレッシングモードでコードを生成します。その際（ラベル/シンボル値）-（.DP宣言値）でコード化します。

### 注意事項

本指示命令は、アセンブラに対してDPレジスタ値を宣言する命令であり、実際のDPレジスタ値に値を設定できるものではありません。実際にDPレジスタ値を設定するためには、本指示命令の直前又は直後にDPレジスタ操作命令を記述してください。

### 記述形式

- ・ダイレクトアドレッシングモード（8ビットオフセットモード）動作時  
.DP（数値,シンボル）
- ・拡張ダイレクトアドレッシングモード（6ビットオフセットモード）動作時  
.DP[0|1|2|3]（数値,シンボル）

### 記述規則

- ・指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ダイレクトアドレッシングモードを使用する前に、必ず本指示命令を記述してください。
- ・オペランドには、0～0FFFFHの範囲の整数値が記述できます。
- ・オペランドにはシンボルが記述できます。
- ・拡張ダイレクトアドレッシングモード（6ビットオフセットモード）時（DPレジスタ4本使用）は、.DP0、.DP1、.DP2、.DP3が記述できます。
- ・ダイレクトアドレッシングモード（8ビットオフセットモード）時（DPレジスタ1本のみ使用）は、.DPのみ記述できます。

### 注意事項

ダイレクトアドレッシングモードと拡張ダイレクトアドレッシングモードを混在して使用することはできません。

### 記述例

```
.DP0800H
DP0 = 800H
```

## .DP[0|1|2|3]SYM

### 機能

- ・ 本指示命令のオペランドに指定したシンボルに対して、ダイレクトアドレッシングモード又は拡張ダイレクトアドレッシングモードが選択されます。

### 注意事項

ダイレクトアドレッシングモードと拡張ダイレクトアドレッシングモードを混在して使用することはできません。

### 記述形式

- ・ ダイレクトアドレッシングモード（8ビットオフセットモード）動作時  
 .DPSYM                   (シンボル)  
 .DPSYM                   (シンボル) [, (シンボル) ...]
- ・ 拡張ダイレクトアドレッシングモード（6ビットオフセットモード）動作時  
 .DP[0|1|2|3]SYM       (シンボル)  
 .DP[0|1|2|3]SYM       (シンボル) [, (シンボル) ...]

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドには、シンボル及びラベルが記述できます。ただし、ビットシンボルは記述できません。
- ・ 本指示命令を記述する場合は、必ず指示命令".DP[0|1|2|3]"でダイレクトページレジスタ値を設定してください。
- ・ 本指示命令は、必ずセクション範囲内に記述してください。
- ・ 複数のシンボルを指定する場合は、シンボルをカンマで区切って記述してください。
- ・ 他の".DP[0|1|2|3]SYM"や".DTSYM"及び".LGSYM"で既に指定したシンボルを再定義した場合は、以降の記述では再定義した内容に従いアドレッシングモードが決定されます。

### 記述例

```
.DP0       800H
LDD       0,800H
.DP0SYM   sym1,sym2
```

## .DT

---

### 機能

- ・ データバンクレジスタ値を宣言します。
- ・ アセンブル実行時にデータバンクレジスタの値を本指示命令で定義した値であると判断し、以降のコードを生成します。
- ・ 以降の行で、アブソリュートアドレッシングモードを指定できます。
- ・ 指示命令".DTSYM"で指定されたシンボルを用いた命令に対して、アブソリュートアドレッシングモードでコードを生成します。その際、ラベルまたはシンボル値の下位16ビットアドレスをコード化します。

### 記述形式

.DT (数値)

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ アブソリュートアドレッシングモードを記述する前に、必ず本指示命令を記述してください。
- ・ オペランドには、0～0FHの範囲の整数値が記述できます。
- ・ オペランドにはシンボルが記述できます。

### 注意事項

本指示命令は、アセンブラに対してDTレジスタ値を宣言する命令であり、実際のDTレジスタ値に値を設定できるものではありません。実際にDTレジスタ値を設定するためには、本指示命令の直前又は直後にDTレジスタ操作命令を記述してください。

---

### 記述例

```
.DT 01H  
LDT #01H
```

## .DTSYM

---

### 機能

- ・ 本指示命令のオペランドに指定したシンボルに対して、アブソリュートアドレッシングモードが選択されます。

### 記述形式

```
.DTSYM      (シンボル)  
.DTSYM      (シンボル) [, (シンボル) ...]
```

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 本指示命令を記述する場合は、必ず指示命令".DT"でデータバンクレジスタ値を設定してください。
- ・ 本指示命令は、必ずセクション範囲内に記述してください。
- ・ オペランドには、シンボル及びラベルが記述できます。ただし、ビットシンボルは記述できません。
- ・ 複数のシンボルを指定する場合は、シンボルをカンマで区切って記述してください。
- ・ 他の".DP[0!1!2!3]SYM"や".DTSYM"及び".LGSYM"で既に指定したシンボルを再定義した場合は、以降の記述では再定義した内容に従いアドレッシングモードが決定されます。

### 記述例

```
.DT      02H  
LDT      #02H  
.DTSYM   sym1 , sym2
```

## .DWORD

### 機能

- ・ 4バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。
- ・ 本指示命令で定義されたラベルを構造化記述命令の式中に記述した場合、式のサイズをダブルワードサイズに指定します。

### 記述形式

```
.DWORD (数値)
(名前:) .DWORD (数値)
```

### 記述規則

- ・ オペランドに整数値を記述してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ 複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
- ・ オペランドにはクォーテーション(')又は、ダブルクォーテーション(")で囲って、文字又は、文字列を記述できます。このとき格納されるデータは、文字のASCIIコードになります。

### 注意事項

**オペランドに文字列を記述する場合、有効な文字数は4文字です。4文字を超える場合はワーニングとなります。**

- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```
.SECTIONname,ROMDATA
.DWORD 1
.DWORD "data"
.DWORD symbol
.DWORD symbol+1
.DWORD 1,2,3,4,5
.END
```



## .EINSF

---

### 機能

- ・ インспекタ情報の関数（サブルーチン）終了情報を定義します。
- ・ ".INSF"から、関数（サブルーチン）終了情報までを1つの関数（サブルーチン）情報として定義します。

### 記述形式

```
.EINSF
```

### 記述規則

- ・ 本指示命令を記述した場合、必ず指示命令 ".INSF"を記述してください。
- ・ 本指示命令はアセンブラ言語記述専用の指示命令であり、NC79のasm関数にて本指示命令を記述した場合、エラーとなります。
- ・ 本指示命令は、コマンドオプション "- finfo" が指定された場合に有効となります。

### 記述例

```
.INSF  glbfunc, G, 0  
:  
.EINSF
```

## .ELIF

---

### 機能

- ・ 複数の条件で条件アセンブルを行いたい場合に、".IF"と組み合わせて条件を記述します。
- ・ オペランドに記述した条件を判定し、真であれば以降に続くボディをアセンブルします。
- ・ 条件が真である場合にアセンブルされる行は、指示命令".ELIF",".ELSE"及び".ENDIF"行の前までです。

### 記述形式

```
.IF 条件式
    ボディ
.ELIF 条件式
    ボディ
.ENDIF
```

### 記述規則

- ・ 本指示命令のオペランドには、必ず条件式を記述してください。
- ・ 本指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 本指示命令は、1つの条件アセンブルブロック内に複数記述できます。

### 記述例

```
.IF TYPE==0
    .byte    "Proto Type Mode"
.ELIF  TYPE>0
    .byte    "Mass Production Mode"
.ELSE
    .byte    "Debug Mode"
.ENDIF
```

## .ELSE

---

### 機能

- 全ての条件が偽である場合に、アセンブルを実行する行の始まりを示します。
- 指示命令".ENDIF"の前の行までをアセンブルします。

### 記述形式

```
.IF    条件式  
    ボディ  
.ELSE  
    ボディ  
.ENDIF
```

```
.IF    条件式  
    ボディ  
.ELIF  条件式  
    ボディ  
.ELSE  
    ボディ  
.ENDIF
```

### 記述規則

- 本指示命令は、条件アセンブルブロック内に一つ以下記述できます。
- 本指示命令にオペランドはありません。

### 記述例

```
.IF TYPE==0  
    .byte    "Proto Type Mode"  
.ELIF TYPE>0  
    .byte    "Mass Production Mode"  
.ELSE  
    .byte    "Debug Mode"  
.ENDIF
```

## .END

---

### 機能

- ・ ソースプログラムの終了を宣言します。
- ・ 本指示命令を記述した行以降の記述内容に対する処理は一切行いません。従ってリストファイルの出力やコード生成などの処理は行いません。

### 記述形式

.END

### 記述規則

- ・ 本指示命令は、一つのアセンブリソースファイルに必ず一つ以上記述する必要があります。

### 注意事項

AS79は、本指示命令以降の行については、エラーの検出もしません。

### 記述例

```
.END
```

## .ENDIF

---

### 機能

- ・ 条件アセンブルブロックの終了を示します。

### 記述形式

```
.IF    条件式  
    ボディ  
.ENDIF
```

### 記述規則

- ・ 本指示命令は、条件アセンブルブロックに必ず一つ記述してください。
- ・ 本指示命令にオペランドはありません。

### 記述例

```
.IF    TYPE==0  
.byte  "Proto Type Mode"  
.ELIF  TYPE>0  
.byte  "Mass Production Mode"  
.ELSE  
.byte  "Debug Mode"  
.ENDIF
```

## .ENDM

---

### 機能

- ・ 一つのマクロ定義のボディが終了する事を示します。

### 記述規則

- ・ 必ず、指示命令".MACRO"に対応させて記述してください。

### 記述形式

```
(マクロ名)  .MACRO      (仮引数)
              ボディ
              .ENDM
```

### 記述例

#### マクロ定義

```
ldm .MACRO  p1,p2
      MOVW.W  p2,p1
      .ENDM
```

#### マクロ呼び出し

```
ldm #0,mem
```

#### マクロ展開

```
MOVW.W  mem,#0
```

## .ENDR

---

### 機能

- ・ 繰り返しマクロの終了を示します。

### 記述形式

```
[(ラベル):] .MREPEAT (数値)  
           ボディ  
           .ENDR
```

### 記述規則

- ・ 必ず指示命令".MREPEAT"に対応させて記述してください。

### 記述例

#### マクロ定義

```
rep .MACRO num  
    .MREPEATnum  
    .IF num > 49  
        .EXITM  
    .ENDIF  
    nop  
    .ENDR  
    .ENDM
```

#### マクロ呼び出し

```
rep 3
```

#### マクロ展開

```
nop  
nop  
nop
```

## .EQU

---

### 機能

- ・ シンボルに32ビット符号付き整数値の範囲の値を定義します。

### 記述形式

(名前) .EQU (数値)

### 記述規則

- ・ シンボルに定義できる値は、アセンブル実行時に確定しなければなりません。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ シンボル定義のオペランドには、シンボルを記述できます。

### 注意事項

---

**前方参照となるシンボル名は記述できません。**

---

- ・ シンボル定義のオペランドには式を記述できます。
- ・ シンボルはグローバル指定ができます。
- ・ 同一シンボルの再定義を行うことが可能です。
- ・ 再定義を行った場合にはそれ以降の行で再定義した値が有効になります。

### 記述例

```
symbol .equ1  
symbol1 .equsymbol+symbol  
symbol2 .equ2
```



## .EXITM

---

### 機能

- ・ マクロボディの展開を中止し、最も近い".ENDM"に制御を渡します。

### 記述形式

```
(マクロ名)  .MACRO          (仮引数)
             ボディ
             .EXITM
             ボディ
             .ENDM
```

### 記述規則

- ・ マクロ定義のボディ内に記述してください。

### 記述例

#### マクロ定義

```
data1 .MACRO value
      .IF value == 0
        .EXITM
      .ELSE
        .BLKB value
      .ENDIF
      .ENDM
```

#### マクロ呼び出し

```
data1 0
```

#### マクロ展開

```
.IF 0 == 0
  .EXITM
.ENDIF
.ENDM
```

## .FLOAT

### 機能

- ・ 4バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。

### 記述形式

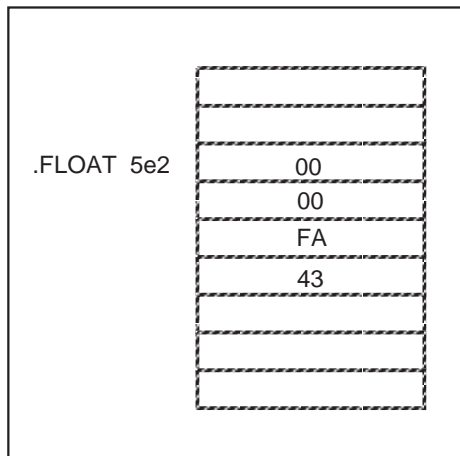
```
.FLOAT    (数値)
(名前:) .FLOAT    (数値)
```

### 記述規則

- ・ オペランドに浮動小数点数を記述してください。
- ・ 浮動小数点数の記述方法は、「オペランドの記述規則」を参照してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```
constant: .FLOAT    5E2
           .FLOAT    5e2
```



## .FORM

---

### 機能

- ・ リストファイルの1ページの行数を20～255行の範囲で指定します。
- ・ リストファイルの1ページの桁数を80～295桁の範囲で指定します。
- ・ 本指示命令を記述した次のページから記述内容が有効になります。ただし、本指示命令をアセンブリソースファイルの1行目に記述した場合は、1ページ目から、指定内容が有効になります。
- ・ 本指示命令を指定しない場合は、66行、140桁で出力します。

### 記述形式

```
.FORM (行数),(桁数)  
.FORM (行数)  
.FORM ,(桁数)
```

### 記述規則

- ・ 1つのソースファイルに複数回記述できます。
- ・ 行数及び桁数にはシンボルを記述できます。

#### 注意事項

---

**前方参照となるシンボルは記述できません。**

---

- ・ 行数及び桁数には式を記述できます。
- ・ オペランドに桁数のみを指定する場合は、数値の直前に必ずカンマ(,)を記述してください。

### 記述例

```
.FORM 20,80  
.FORM 60  
.FORM ,100  
.FORM line,culmn
```

## .GLB

---

### 機能

- ・ オペランドに指定したラベル及びシンボルが、グローバルであることを宣言します。
- ・ オペランドに指定したラベル及びシンボルで、ファイル内で定義されているものは、外部から参照できるように処理します。
- ・ オペランドに指定したラベル及びシンボルで、ファイル内で定義されていないものは、外部のファイルで定義されているものとして処理します。

### 記述形式

.GLB (名前) [, (名前) ...]

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドにグローバルラベルとするラベル名を記述します。
- ・ オペランドにグローバルシンボルとするシンボル名を記述します。
- ・ オペランドに複数のラベル及びシンボル名を記述する場合は、カンマ(,)で区切って記述してください。
- ・ オペランドにビットシンボルは記述できません。

### 記述例

```
.GLB name1,name2,name3
.SECTION program
ADC A,name1
[name1] = [name4] ; Byte size
```

## .IF

### 機能

- ・ 条件アセンブルブロックの始まりを示します。
- ・ オペランドに記述した条件を判定し、真であれば以降に続くボディをアセンブルします。
- ・ 条件が真である場合にアセンブルされる行は、指示命令".ELIF",".ELSE"及び".ENDIF"行の前までです。
- ・ 条件アセンブルボディ内には、AS79のソースプログラムに記述可能な全ての命令を記述できます。

### 記述形式

```
.IF 条件式
ボディ
.ENDIF
```

### 記述規則

- ・ 本指示命令のオペランドには、必ず条件式を記述してください。
- ・ 本指示命令とオペランドの間には、必ずスペース又はタブを記述してください。

### 条件式の機能

- ・ 条件式の結果によって、条件アセンブルが行われます。

### 条件式の記述規則

- ・ 条件式は、指示命令のオペランドに一つだけ記述できます。
- ・ 条件式には、必ず条件演算子を記述してください。
- ・ 次に示す条件演算子が記述できます。

条件演算子	内容
>	左辺値が右辺値より大きい場合に真
<	右辺値が左辺値より大きい場合に真
> =	左辺値が右辺値より大きいか等しい場合に真
< =	右辺値が左辺値より大きいか等しい場合に真
= =	左辺値と右辺値が等しい場合に真
! =	左辺値と右辺値が等しくない場合に真

- ・ 条件式の演算は符号付き32ビットで演算します。

#### 注意事項

演算結果のオーバフロー及びアンダーフローは判断しません。

- ・ 条件演算子の左辺及び右辺には、シンボルが記述できます。

#### 注意事項

シンボルの、前方参照（本指示命令行より後に定義されているシンボルを参照）はできません。前方参照のシンボルや、未定義のシンボルを記述した場合は、値を0として式を判定します。

## AS79指示命令

- 条件演算子の左辺及び右辺には、式が記述できます。式は「式の記述規則」に従って記述してください。
- 条件演算子の左辺及び右辺には、文字列が記述できます。文字列は、必ずシングルクォーテーション(')又はダブルクォーテーション(")で囲って記述してください。このとき、文字列の大小は、文字コードの値で判定されます。  
"ABC"<"CBA"      414243 < 434241で真となります。  
"C" < "A"      43 < 41で偽となります。
- 条件演算子の前後には、スペース又はタブが記述できます。
- 条件式は、指示命令".IF"及び".ELIF"のオペランドに記述できます。

### 条件式の記述例

```
sym<1  
sym < 1  
sym+2 < data1  
sym+2 < data1+2  
'smp1'==name
```

### 記述例

```
.IF TYPE==0  
    .byte    "Proto Type Mode"  
.ELIF  TYPE>0  
    .byte    "Mass Production Mode"  
.ELSE  
    .byte    "Debug Mode"  
.ENDIF
```

## .INCLUDE

---

### 機能

- ・ ソースプログラムの行に、他のファイルの内容を全て読み込みます。
- ・ 本指示命令で読み込まれたファイルの内容は、読み込んだファイル内に記述した場合と、同じ一つのファイルとして処理されます。
- ・ インクルードファイル名に絶対パスを記述した場合は、記述したディレクトリ内のファイルを検索します。ファイルが見つからない場合はエラーとなります。
- ・ インクルードファイル名に絶対または相対パスを記述した場合は、記述したディレクトリ内のファイルを検索します。ファイルが見つからない場合はエラーとなります。
- ・ インクルードファイル名にパスがない場合は次に示す順序でファイルを検索します。
  1. AS79起動時にコマンド行で指定したファイル名にディレクトリ指定がない場合は、インクルード指示命令で指定されたファイル名を検索します。AS79起動時にコマンド行で指定したファイル名にディレクトリ指定がある場合は、インクルード指示命令で指定されたファイル名にコマンド行で指定されたディレクトリ名を付加して検索します。
  2. コマンドオプション-Iで指定されたディレクトリを検索
  3. 環境変数INC79に設定されているディレクトリを検索

### 記述形式

.INCLUDE (ファイル名)

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドのファイル名には、必ずファイル拡張子を記述してください。
- ・ オペランドには、指示命令"..FILE"や"@ "を含む文字列が記述できます。
- ・ インクルードファイルのネスティングレベルは9レベルまでです。

### 注意事項

**インクルードファイル内で、自分自身をインクルード指定しないでください。**

### 記述例

```
.INCLUDE initial.a79
.INCLUDE ..FILE@.inc
```

## .INDEX

---

### 機能

- CPU内部のインデックス長(8または16)を示します。
- 本指示命令は、下記の事項に影響を及ぼします。
  - 1 xフラグの状態により生成コードの内容が異なる命令に対するアセンブル処理
  - 2 構造化記述命令に対するコード生成

### 記述形式

.INDEX (数値)

### 記述規則

- 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- オペランドの数値が8の時は8ビット長、16の時は16ビット長を示します。
- CLP X,SEP X命令等でインデックス長を変更する場合は、本指示命令の宣言も同時に行ってください。

### 注意事項

---

本指示命令はアセンブラに対してインデックス長宣言をする命令であり、実際のインデックス長を設定できるものではありません。実際にインデックス長を設定するためには、本指示命令の直前または直後にインデックス長操作命令を記述して下さい。

---

### 記述例

```
.INDEX 16
CLP X
LDX #symbol ;16bits length
```



## .INSF

---

### 機能

- ・ インспекタ情報の関数（サブルーチン）開始情報を定義します。
- ・ 関数（サブルーチン）開始情報から、指示命令 ".EINSF"までを1つの関数（サブルーチン）情報として定義します。

### 記述形式

.INSF （関数（サブルーチン）開始ラベル名），（記憶クラス），（フレームサイズ）

### 記述規則

- ・ 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- ・ 関数（サブルーチン）開始ラベル名、記憶クラスおよびフレームサイズは必ず記述してください。
- ・ 記憶クラスおよびフレームサイズを記述する場合は、カンマで区切って記述してください。
- ・ 記憶クラスは 'G（グローバルラベル）'、'S（ローカルラベル）' のいずれかを記述してください。
- ・ フレームサイズは整数値を記述してください。

### 注意事項

---

本指示命令を記述した場合、必ず指示命令 ".EINSF"を記述してください。  
本指示命令はアセンブラ言語記述専用の指示命令であり、NC79のasm関数にて本指示命令を記述した場合、エラーとなります。  
本指示命令は、コマンドオプション "- finfo" が指定された場合に有効となります。

---

### 記述例

```
glbfunc:  
  .INSF  glbfunc, G, 0  
  :  
  .EINSF  
  
locfunc:  
  .INSF  locfunc, S, 0  
  :  
  .EINSF
```

## .INSTR

---

### 機能

- ・ オペランドで指定した文字列のなかで、検出文字列が始まる位置を示します。
- ・ 文字列の検索を開始する位置を指定できます。

#### 注意事項

---

文字列よりも、検索文字列が長い場合の値は0となります。文字列のなかに、検索文字列が含まれていなかった場合の値は0となります。文字列の長さよりも、検索開始位置の値が大きかった場合の値は0となります。

---

### 記述形式

```
.INSTR {" (文字列) ", " (検出文字列) ", (検出開始位置) }
.INSTR { (文字列) ', (検出文字列) ', (検出開始位置) }
```

### 記述規則

- ・ オペランドは、必ず{}で囲ってください。
- ・ 文字列、検出文字列及び検索開始位置は、必ず記述してください。
- ・ 文字列、検出文字列及び検索開始位置は、カンマで区切って記述してください。
- ・ カンマの前後には、スペース及びタブは記述できません。
- ・ 検索開始位置は、シンボルを記述できます。
- ・ 検索開始位置を1とした場合は、文字列の先頭を示します。
- ・ 文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。

#### 注意事項

---

漢字などの8ビットコードについては、正しく処理されませんが、AS79はエラーの検出を行いません。

---

- ・ 文字列は、必ずクォーテーションで囲って記述してください。

#### 注意事項

---

マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。ダブルクォーテーションで囲って記述した文字列は文字列そのものが展開されます。

---

- ・ 本指示命令は、式の項に記述できます。

## 記述例

## マクロ定義

```
top      .EQU1

point_set .MACRO  source,dest,top
point    .EQU     .INSTR{'source','dest',top}
        .ENDM
```

## マクロ呼び出し

```
point_set  japanese,se,1
```

## マクロ展開

```
point      .EQU7
```

- ・ 指定した文字列(japanese)の先頭(top)からの、"se"文字列の位置(7)を取り出しています。

## .LEN

---

### 機能

- ・ オペランドに記述した文字列の文字列長を示します。

### 記述形式

.LEN {" (文字列) }

.LEN {' (文字列) }

### 記述規則

- ・ オペランドは、必ず{}で囲ってください。
- ・ 本指示命令とオペランドの間にスペース又はタブが記述できます。
- ・ 文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。

#### 注意事項

---

漢字などの8ビットコードについては、正しく処理されませんが、AS79はエラーの検出を行いません。

---

- ・ 文字列は、必ずクォーテーションで囲って記述してください。

#### 注意事項

---

マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。ダブルクォーテーションで囲って記述した文字列は文字列そのものが展開されます。

---

- ・ 本指示命令を式の項に記述できます。

## 記述例

## マクロ定義1

```
bufset .MACRO f1,f2
buffer@f1: .BLKB .LEN{'f2'}
.ENDM
```

## マクロ呼び出し1

```
bufset 1,Printout_data
bufset 2,Sample
```

## マクロ展開1

```
buffer1: .BLKB 13
buffer2: .BLKB 6
```

## マクロ定義例2

```
buf .MACRO f1
buffer: .BLKB .LEN{"f1"}
.ENDM
```

## マクロ呼び出し2 : dataは展開されません

```
buf 1,data
```

## マクロ展開2

```
buffer: .BLKB 2
```

## .LENGTH

---

### 機能

- CPU内部のデータ長及びインデックス長を同時に指定します。
- 本指示命令は、mおよびxフラグと関連する下記の事項に影響を及ぼします。
  - 1 mフラグ、xフラグの状態により生成コードの内容が異なる命令に対するアセンブル処理
  - 2 構造化記述命令に対するコード生成アドレッシングモードのデータ長

### 記述形式

```
.LENGTH (データ長), (インデックス長)
.LENGTH [8:16],[8:16]
```

### 記述規則

- 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- 第一オペランドと第二オペランドの間には、必ずカンマを記述してください。
- 第一オペランドでデータ長を指定し、第二オペランドでインデックス長を指定します。数値が8の時は8ビット長、16の時は16ビット長を示します。

### 注意事項

---

本指示命令はアセンブラに対してデータ長およびインデックス長宣言をする命令であり、実際のデータ長およびインデックス長を設定できるものではありません。実際にデータ長およびインデックス長を設定するためには、本指示命令の直前または直後にデータ長およびインデックス長操作命令を記述して下さい。

---

### 記述例

```
.LENGTH 16,16
CLP      M,X
ADC      A,#symbol ;16 bits length
LDX      #symbol ;16 bits length
```

## .LGSYM

---

### 機能

- ・ 本指示命令のオペランドに指定したシンボルに対して、アブソリュートロングアドレッシングモードが選択されます。

### 記述形式

```
.LGSYM      (シンボル)  
.LGSYM      (シンボル) [, (シンボル) ...]
```

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 本指示命令は、必ずセクション範囲内に記述してください。
- ・ オペランドには、シンボル及びラベルが記述できます。ただし、ビットシンボルは記述できません。
- ・ 複数のシンボルを指定する場合は、シンボルをカンマで区切って記述してください。
- ・ 他の".DP[0!1!2!3]SYM"や".DTSYM"及び".LGSYM"で既に指定したシンボルを再定義した場合は、以降の記述では再定義した内容に従いアドレッシングモードが決定されます。

### 記述例

```
.LGSYM  sym1 , sym2
```

## .LIST

---

### 機能

- ・ リストファイルへの行の出力を開始(ON)することができます。
- ・ リストファイルへの行の出力を停止(OFF)することができます。
- ・ エラーが発生した場合は、リストへの行の出力を停止している範囲内でもエラーメッセージ及びエラー発生行をリストファイルに出力します。
- ・ 本指示命令を指定しない場合は、全ての行をリストファイルに出力します。

### 記述形式

```
.LIST [ON|OFF]
```

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 行の出力を停止する場合は、オペランドに'OFF'を記述してください。
- ・ 行の出力を開始する場合は、オペランドに'ON'を記述してください。

### 記述例

```
.LIST ON  
.LIST OFF
```



## .LOCAL

### 機能

- ・ オペランドに記述されたラベルがマクロローカルラベルであることを宣言します。
- ・ マクロローカルラベルは、異なるマクロ定義及びマクロ定義外であれば、同一の名前を複数個記述できます。

### 注意事項

マクロ定義がネストしている場合は、マクロ定義内で定義を行っているマクロ内のマクロローカルラベルは、同一名を使用できません。

### 記述規則

`.LOCAL (ラベル名) [, (ラベル名) ...]`

### 記述規則

- ・ 本指示命令は、必ずマクロボディ内に記述してください。
- ・ 本指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 本指示命令によるマクロローカルラベル宣言は、ラベル名を定義するより前に記述してください。
- ・ マクロローカルラベル名の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
- ・ 本指示命令のオペランドは、カンマで区切って複数のラベルを記述できます。このときの最大ラベル数は100個までです。

### 注意事項

インクルードファイルの内容を含む、一つのアセンブリソースファイルに記述できるマクロローカルラベルは65535個までです。

### 記述例

```
name      .MACRO
          .LOCAL      m1 ; m1 is local label
m1:
          nop
          jmp          m1
          .ENDM
```

## .MACRO

---

### 機能

- ・ マクロ名を定義します。
- ・ マクロ定義の始まりを示します。

### 記述形式

#### マクロ定義

```
(マクロ名) .MACRO [(仮引数) [, (仮引数) ...]]
    ボディ
.ENDM
```

#### マクロ呼び出し

```
(マクロ名) [(実引数) [, (実引数) ...]]
```

### 記述規則

- ・ マクロ名は必ず記述してください。
- ・ マクロ名の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
- ・ オペランドには、仮引数が定義できます。
- ・ 本指示命令とマクロ仮引数の間には、必ずスペース又はタブを記述してください。
- ・ 本指示命令とマクロ名の間には、スペース又はタブを記述できます。
- ・ マクロのネスティングレベルは65535レベルまでです。

### 仮引数の記述規則

- ・ マクロ仮引数の名前の記述は、「プログラムの記述規則」の「名前の記述規則」に従ってください。
- ・ マクロ仮引数の名前は、ネストしているマクロ定義を含めて、異なる名前で定義してください。
- ・ 仮引数を複数定義する場合は、仮引数をカンマ(,)で区切って記述してください。
- ・ 指示命令".MACRO"のオペランドに記述した仮引数は、必ずマクロボディ内に記述してください。

#### 注意事項

---

ダブルクォーテーションで囲った文字列は、全てその文字列そのものを示します。仮引数をダブルクォーテーションで囲わないでください。

---

- ・ 仮引数は80個まで記述できます。

#### 注意事項

---

1行に記述できる文字数の範囲内で最大80個まで記述できます。

---

## 実引数の記述規則

- ・ マクロ名と実引数の間には、必ずスペース又はタブを記述してください。
- ・ 実引数は、マクロ呼び出しの際に仮引数に対応させて記述してください。
- ・ 特殊文字を実引数に記述する場合は、ダブルクォーテーションで囲って記述してください。
- ・ 実引数には、ラベル、グローバルラベル及びシンボルが記述できます。
- ・ 実引数には式が記述できます。

## 実引数の展開

- ・ 仮引数と実引数は、左から記述されている順に置き換えられます。
- ・ 仮引数が定義されていて、マクロ呼び出しで実引数の記述が無い場合は、仮引数にあたる部分のコードは出力されません。
- ・ 仮引数の数が、実引数の数より多い場合は、対応する実引数がない仮引数にあたる部分のコードは出力されません。
- ・ ボディに記述した仮引数をシングルクォーテーション(')で囲った場合は、対応する実引数をシングルクォーテーションで囲って出力されます。
- ・ 1つの実引数がカンマ(,)を含む場合に、括弧(())で囲った場合は、括弧を含めて変換されます。
- ・ 実引数の数が、仮引数の数より多い場合は、対応する仮引数がない実引数については処理されません。

### 注意事項

実引数と仮引数の数が合わない場合は、AS79はワーニングメッセージを出力します。

## 実引数の展開例

### マクロ定義例

```
name .MACRO string
    .BYTE 'string'
.ENDM
```

### マクロ呼び出し例1

```
name "name, address"
```

### 展開例1

```
.BYTE 'name, address'
```

### マクロ呼び出し例2

```
name (name, address)
```

### 展開例2

```
.BYTE '(name, address)'
```

## 記述例

## マクロ定義

```

mac .MACRO  p1,p2,p3
  .IF  ..MACPARA == 3
    .IF  'p1' == 'byte'
      MOVMB.B  p2,#p3
    .ELSE
      MOVM.W   p2,#p3
    .ENDIF
  .ELIF  ..MACPARA == 2
    .IF  'p1' == 'byte'
      LDABA,p2
    .ELSE
      LDA.W   A,p2
    .ENDIF
  .ENDIF
  .ENDM

```

## マクロ呼び出し

```

mac word,SYM,10

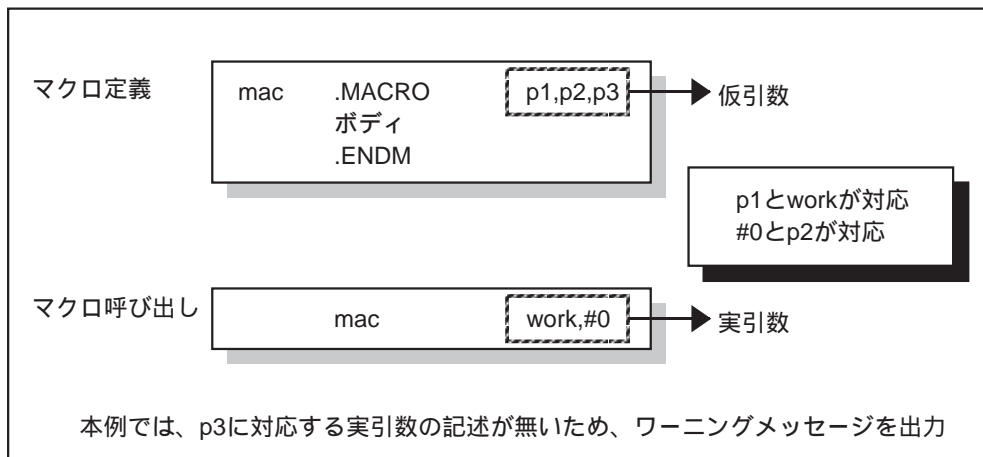
```

## マクロ展開

```

  .IF  3=3
  .ELSE
    MOVM.W   SYM,#10
  .ENDIF
  .ENDIF
  .ENDM

```



## .MREPEAT

### 機能

- ・ 繰り返しマクロの始まりを示します。
- ・ ボディを指定した数値回、繰り返して展開します。
- ・ 繰り返し回数は、最大65535回まで指定できます。
- ・ 65535レベルまでのネストができます。
- ・ 本指示命令を記述した場所に、マクロボディを展開します。

### 記述形式

```
[ (ラベル) : ] .MREPEAT    (数値)
                ボディ
                .ENDR
```

### 記述規則

- ・ オペランドは必ず記述してください。
- ・ 本指示命令とオペランドの間に必ず、スペース又はタブを記述してください。
- ・ 本指示命令行の先頭にラベルを記述できます。
- ・ オペランドには、シンボルを記述できます。

### 注意事項

**前方参照となるシンボルは記述できません。**

- ・ オペランドには、式が記述できます。
- ・ ボディには、マクロ定義及びマクロ呼び出しが記述できます。
- ・ ボディ内に指示命令".EXITM"を記述できます。

### 記述例

#### マクロ定義

```
rep .MACRO  num
    .MREPEATnum
    .IF num > 49
    .EXITM
    .ENDIF
nop
.ENDR
.ENDM
```

#### マクロ呼び出し

```
rep 3
```

#### マクロ展開

```
nop
nop
nop
```

## .ORG

---

### 機能

- ・ セクションを配置する絶対アドレスを指定します。
- ・ 本指示命令を記述することにより、該当セクションは絶対属性を持ちますので、リンク時にアドレスの再配置は出来ません。
- ・ 本指示命令を記述した直後の行から記述したニーモニックのコードが格納されるアドレスを決定します。
- ・ 本指示命令の直後の行から記述した領域確保指示命令で、確保されるメモリのアドレスを決定します。

#### 注意事項

---

絶対属性セクションは、リンク時にアドレスの再配置ができません。

---

### 記述形式

.ORG (数値)

### 記述規則

- ・ 本指示命令は、必ず、セクション指示命令の直後に記述してください。

#### 注意事項

---

".SECTION"を記述した直後の行に".ORG"の記述が無い場合は、そのセクションは相対属性セクションとなります。

---

- ・ 相対属性セクション内には、本指示命令は記述できません。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに記述できる値は、0～0FFFFFFHの範囲の数値です。値は10進数、16進数のいずれかを記述できます。
- ・ オペランドには式を記述できます。ただし、式の値がアセンブル実行時に確定する値でなければなりません。
- ・ オペランドにはシンボルを記述できます。ただし、シンボルの値がアセンブル実行時に確定する値でなければなりません。

#### 注意事項

---

SECTION指示命令でALIGN指定を行った相対属性セクションでは本指示命令で絶対アドレスを指定することはできません。

---

- ・ 絶対属性セクション内であれば複数回記述できます。

### 記述例

```
.SECTIONname ,ROMDATA
.ORG      0FF00H
.BYTE     "abcdefghijklmnopqrstuvwxyz"
.ORG      0FF80H
.BYTE     "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
.END
```

次のような記述はエラーとなります。

```
.SECTIONname ,ROMDATA
.BYTE     "abcdefghijklmnopqrstuvwxyz"
.ORG      0FF80H ; Error
.BYTE     "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
.END
```

## .PAGE

---

### 機能

- ・ リストファイルを改ページします。
- ・ オペランドに記述した文字列を改ページした際のヘッダ部分に出力します。

### 注意事項

ヘッダに出力できる最大文字数は（リストファイルの桁数） - 85文字です。  
リストファイルの桁数は、指示命令".FORM"で設定できます。

### 記述形式

```
.PAGE "(文字列)"  
.PAGE '(文字列)'
```

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドは、クォーテーション( )又はダブルクォーテーション(")で囲って記述してください。
- ・ オペランドは省略できます。

### 記述例

```
.PAGE  
.PAGE "strings"  
.PAGE 'strings'
```



## .SECTION

### 機能

- ・ セクション名を定義します。
- ・ セクションの始まりを定義します。一つのセクション指示命令から、次のセクション指示命令又は指示命令".END"までを一つのセクションとして定義します。
- ・ セクションタイプを定義します。
- ・ ALIGN=指定がある場合、In79がセクションの始まりを指定値に従って割り当てます。
- ・ ALIGN=指定をした相対属性セクションおよび絶対属性セクションに、指示命令".ALIGN"が記述できます。

### 記述形式

```
.SECTION (セクション名)
.SECTION (セクション名), (セクションタイプ)
.SECTION (セクション名), (セクションタイプ), ALIGN=[2!4]
.SECTION (セクション名), ALIGN=[2!4]
```

### 記述規則

- ・ セクション名は必ず記述してください。
- ・ メモリ領域を確保したり、メモリにデータを格納するアセンブリ指示命令を記述する場合や、ニーモニック及び構造化記述命令を記述する場合は必ず、本指示命令でセクションを定義してください。
- ・ セクションタイプとALIGNは、セクション名の後に記述してください。
- ・ セクションタイプ及び、ALIGN指定をする場合は、カンマで区切って記述してください。
- ・ セクションタイプとALIGNの記述順序は任意です。
- ・ セクションタイプは、'CODE','ROMDATA','DATA'のいずれかを記述できます。

セクションタイプ	内容
CODE	プログラム領域
ROMDATA	固定データ領域
DATA	可変データ領域

- ・ セクションタイプを省略した場合、as79は次の命令の種類に従ってセクションタイプをCODE又はDATAから選択します。。

### 記述例

```
.SECTION program, CODE
nop
.SECTION ram, DATA
.BLKB 10
.SECTION dname, ROMDATA
.BYTE "abcd"
.END
```

## .STK

---

### 機能

- ・ インспекタ情報のスタック情報を定義します。

### 記述形式

.STK スタックサイズ

### 記述規則

- ・ 本指示命令とオペランドの間には、必ずスペースまたはタブを記述してください。
- ・ スタックサイズは必ず記述してください。
- ・ スタックサイズは整数値を記述してください。

### 注意事項

---

本指示命令は、インспекタ情報の関数開始情報と関数終了情報の範囲内で記述してください。

本指示命令は、コマンドオプション"- finfo" が指定された場合に有効となります。

---

### 記述例

```
.INSF  glbfunc, G, 0
      :
      .STK  2          ;2バイトプッシュ
jsr  glbsub
      .STK  -2        ;2バイトポップ
      :
.EINSF
```

## .SUBSTR

---

### 機能

- ・ 文字列の指定した位置から、指定した文字数を取り出します。

### 注意事項

---

文字列の長さよりも切り出し開始位置の値が大きい場合の値は0となります。文字列の長さよりも切り出し文字数の値が大きい場合の値は0となります。切り出し文字数を0とした場合の値は0となります。

---

### 記述形式

.SUBSTR {" (文字列)", (切り出し開始位置), (切り出し文字数)}  
.SUBSTR {' (文字列)', (切り出し開始位置), (切り出し文字数)}

### 記述規則

- ・ オペランドは、必ず{}で囲ってください。
- ・ 文字列、切り出し開始位置及び切り出し文字数は、必ず記述してください。
- ・ 文字列、切り出し開始位置及び切り出し文字数は、カンマで区切って記述してください。
- ・ 切り出し開始位置及び切り出し文字数は、シンボルが記述できます。
- ・ 切り出し開始位置を1とした場合は、文字列の先頭を示します。
- ・ 文字列には、スペース及びタブを含む、7ビットアスキーコードの文字が記述できます。

### 注意事項

---

漢字などの8ビットコードについては、正しく処理されませんが、AS79はエラーの検出を行いません。

---

- ・ 文字列は、必ずクォーテーションで囲って記述してください。

### 注意事項

---

マクロの引数を文字列として展開したい場合は、引数名をシングルクォーテーションで囲って記述してください。ダブルクォーテーションで囲って記述した文字列は文字列そのものが展開されます。

---

## 記述例

## マクロ定義

```
name      .MACRO  data
          .MREPEAT .LEN{ 'data' }
          .BYTE    .SUBSTR{ 'data' , ..MACREP , 1 }
          .ENDR
          .ENDM
```

## マクロ呼び出し

```
nameABCD
```

## マクロ展開

```
.BYTE    "A"
.BYTE    "B"
.BYTE    "C"
.BYTE    "D"
```

- ・ マクロの実引数として与えられた文字列の長さを、".MREPEAT"のオペランドに与えます。
- ・ "..MACREP"は、". BYTE"の行を実行する毎に、1 2 3 4と増加します。したがって、マクロの実引数として与えられた文字列の先頭の文字から順に1文字ずつ、".BYTE"のオペランドに与えることになります。

## .VER

---

### 機能

- ・ 指定した文字列をIn79が生成するマップファイルへ出力するようにリロケートブルモジュールファイルに出力します。
- ・ マップファイルには指定した全ての文字列が出力されます。
- ・ リロケートブルモジュールファイル毎に、ユーザーの指定する情報をマップファイルに出力できます。

### 記述形式

```
.VER "(文字列)"
```

```
.VER '(文字列)'
```

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドに、出力させたい文字列をクォーテーション(')又はダブルクォーテーション(")で囲って記述してください。
- ・ オペランドは、一行の範囲内で記述してください。
- ・ 1つのアセンブリソースファイルに1度しか記述できません。。
- ・ 指示命令".END"以前であれば任意の行に記述できます。

### 記述例

```
.VER 'strings'  
.VER "strings"
```

## .WORD

### 機能

- ・ 2バイト長の固定データをROMに格納します。
- ・ データを格納したアドレスにラベルを定義することができます。
- ・ 本指示命令にて定義されたラベルを構造化記述命令の式中に記述した場合、式のサイズをワードサイズに指定します。

### 記述形式

```

        .WORD    ( 数値 )
(名前:) .WORD    ( 数値 )

```

### 記述規則

- ・ オペランドに整数値を記述してください。
- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ オペランドにはシンボルを記述できます。
- ・ オペランドには式を記述できます。
- ・ 複数のオペランドを記述するときは、カンマ(,)で区切って記述してください。
- ・ オペランドにはクォーテーション(")又は、ダブルクォーテーション(")で囲って、文字又は、文字列を記述できます。このとき格納されるデータは、文字のASCIIコードになります。

### 注意事項

**オペランドに文字列を記述する場合、有効な文字数は2文字です。2文字を超える場合はワーニングとなります。**

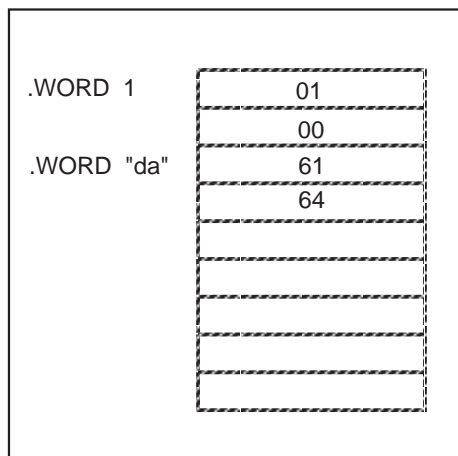
- ・ ラベルを定義する場合には、指示命令の前にラベル名を記述してください。
- ・ ラベル名には必ず、コロン(:)を記述してください。

### 記述例

```

.SECTIONname ,ROMDATA
.WORD      1
.WORD      "da" ,"ta"
.WORD      symbol
.WORD      symbol+1
.WORD      1,2,3,4,5
.END

```



?

## 機能

- ・ テンポラリラベルを定義します。
- ・ 直前又は直後に定義されたテンポラリラベルを参照します。

## 注意事項

参照できるラベルは、直前又は直後のラベルだけです。

- ・ 同一ファイル内で定義及び参照が可能です。ただし、デバッグ時にはテンポラリラベルを参照することはできません。
- ・ ファイル内に65535個までのテンポラリラベルが定義できます。このとき、ファイル内に".INCLUDE"が記述されている場合は、インクルードファイル内のテンポラリラベルを含み65535個までの記述ができます。
- ・ リストファイルには、テンポラリラベルとして変換された結果が出力されます。

## 注意事項

デバッグ時に、テンポラリラベルを参照することはできません。

## 記述形式

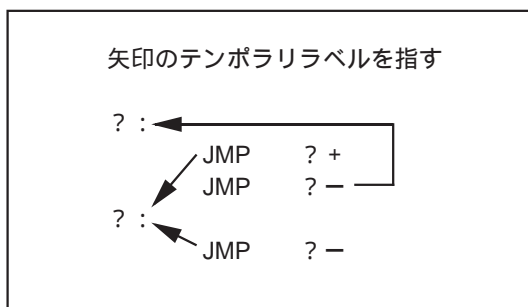
```
?:  
  (ニーモニック) ?+  
  (ニーモニック) ?-
```

## 記述規則

- ・ テンポラリラベルとして定義したい行に"?:"を記述してください。
- ・ 直前に定義したテンポラリラベルを参照したい場合は、命令のオペランドに"?-"を記述してください。
- ・ 直後に定義したテンポラリラベルを参照したい場合は、命令のオペランドに"?+"を記述してください。

## 記述例

```
?:  
  JMP ?+  
  JMP ?-  
?:  
  JMP ?-
```



## @

## 機能

- ・ マクロ引数、マクロ変数、予約シンボル、指示命令"..FILE"の展開ファイル名及び指定文字列を連結します。

## 記述形式

(文字列)@(文字列)  
(文字列)@(文字列)[@(文字列) ...]

## 記述規則

- ・ 本指示命令の前後に記述したスペース及びタブは、文字列として連結します。
- ・ 本指示命令の前後には、文字列が記述できます。
- ・ @を文字データ(40H)として記述する場合は、"(ダブルクォーテーション)で囲んでください。@を含む文字列をシングルクォーテーションで囲った場合は、@の前後の文字列を連結します。
- ・ 一行に複数回記述できます。

## 注意事項

連結した文字列を名前とする場合は、本指示命令の前後にスペース及びタブを記述しないでください。

## 記述例

```
.ASEERT "sample" > ..FILE@.dat
```

- ・ 現在処理中のファイル名がsample1.a79の場合、sample.datファイルにメッセージを出力します。



# 第16章 構造化記述

---

## 16.1 AS79の構造化記述

AS79では、構造化記述命令を用いて、C言語にみられるような構造化プログラミング記述ができます。

- ・ プログラムの構造化を行う命令文を構文と呼びます。
- ・ プログラムの流れを制御する命令文を制御文と呼びます。
- ・ 構文内に記述する文をボディと呼びます。
- ・ AS79の構造化記述は、65535レベルまでのネストした記述ができます。

### 注意事項

---

実際のネストレベルはホストマシンのメモリ容量に依存します。

---

#### IF構文

指定した条件により分岐し、成立した条件のボディを実行するコードを生成します。

#### FOR-STEP構文

指定した回数だけ、ボディを実行するコードを生成します。条件判定を行ってから、ボディを実行します。

#### FOR-NEXT構文

指定した条件が成立している間、ボディを実行するコードを生成します。条件判定を行ってから、ボディを実行します。

#### DO構文

指定した条件が成立している間、ボディを実行するコードを生成します。ボディを実行した後、条件判定を行います。

#### SWITCH構文

複数の条件のなかで、成立した条件のボディを実行するコードを生成します。

#### BREAK制御文

繰り返し処理、多岐選択処理を強制的に終了させます。

#### CONTINUE制御文

繰り返し処理の条件判定行へ、分岐させます。

### FOREVER制御文

繰り返し処理を永久ループさせます。

### GOTO制御文

無条件分岐します。

### CALL制御文

サブルーチンを呼び出します。

### RETURN制御文

サブルーチンの復帰をします。

### 代入文

右辺を左辺に代入します。

## 16.2 構造化記述の演算子

構造化記述行に記述できる演算子を以下に示します。演算子の演算優先順位については、「プログラムの記述規則」の章を参照してください。

### 単項演算子

演算子	機能
+	続く項を正の値として扱います。
-	続く項を負の値として扱います。
~	続く項の論理否定値を扱います。
	続く項の絶対値を扱います。
++	続く項のインクリメントを行います。
--	続く項のデクリメントを行います。
sizeof	オペランドに指定したセクションのサイズ(バイト数)を値として扱います。
startof	オペランドに指定したセクションの開始アドレスを値として扱います。
bank	ラベルの上位8ビット又はシンボルの上位17~24ビットを切り出します。
offset	ラベル又はシンボルの下位16ビットを切り出します。

### 二項演算子

演算子	機能
+ [.C]	左辺値と右辺値を加算します。
- [.C]	左辺値から右辺値を減算します。
* [.S]	左辺値と右辺値を乗算します。
/ [.S]	左辺値を右辺値で除算します。
% [.S]	左辺値を右辺値で割った余りを扱います。
>>	左辺値を右辺値回論理右ビットシフトします。
<<	左辺値を右辺値回算術左ビットシフトします。
>>.A	左辺値を右辺値回算術右ビットシフトします。
<<.R	左辺値を右辺値回左ビットローテートします。
>>.R	左辺値を右辺値回右ビットローテートします。
&	左辺値と右辺値のビット毎の論理積値を扱います。
	左辺値と右辺値のビット毎の論理和値を扱います。
^	左辺値と右辺値のビット毎の排他的論理和値を扱います。
- =	左辺値を右辺値だけ減算し左辺に代入します。

## 条件演算子

演算子	機能
> [.S]	左辺値が右辺値より大きいことを評価します。
< [.S]	右辺値が左辺値より大きいことを評価します。
> = [.S]	左辺値が右辺値より大きいか等しいことを評価します。
< = [.S]	右辺値が左辺値より大きいか等しいことを評価します。
= = [.S]	左辺値と右辺値が等しいことを評価します。
! = [.S]	左辺値と右辺値が等しくないことを評価します。

## 論理演算子

演算子	機能
& &	論理の積を行います。
	論理の和を行います。

## 代入演算子

演算子	機能
= 右	辺値を左辺に代入します。
= .S	右辺値を符号拡張して左辺に代入します。
= .Z	右辺値をゼロ拡張して左辺に代入します。

## 演算優先順位変更演算子

演算子	機能
( )	( ) で囲った演算を最優先で行います。一つの式に複数の ( ) が記述されている場合は、左が優先になります。( ) はネストした記述ができます。

### 注意事項

".S"は符号付き演算を、".C"はキャリー又はポロー付き演算を示します。

## 16.3 構造化記述行の記述規則

構造化記述行の記述方法及び用語を説明します。

### 16.3.1 変数

構造化記述行でマクロコンピュータのレジスタやメモリを変数と言います。変数には次の種類があります。

#### レジスタ変数

7900シリーズがもっているレジスタを示します。

#### フラグ変数

7900シリーズのプロセッサステータスレジスタに割り付けられているフラグを示します。

## レジスタビット変数

レジスタ変数（アキュムレータA及びアキュムレータBのみ）のビット位置を示します。

## メモリ変数

任意のラベルまたはシンボルを示します。

## メモリビット変数

任意のビットシンボルを示します。

## スタック変数

スタックポインタが指しているメモリ領域を示します。

### 16.3.2 予約変数

変数のうち、レジスタ変数、フラグ変数及びレジスタビット変数を予約変数といいます。

### 16.3.3 予約変数の記述規則

予約変数は、以下に示す変数名をそのままプログラムに記述してください。予約変数は大文字と小文字を区別しません。

#### レジスタ変数名

A AL B BL E X Y DP DP0 DP1 DP2 DP3 DT PS  
S PG

#### フラグ変数名

C Z I D XF M V N

#### レジスタビット変数名

BIT\_A0 BIT\_A1 BIT\_A2 BIT\_A3 BIT\_A4 BIT\_A5 BIT\_A6  
BIT\_A7 BIT\_A8 BIT\_A9 BIT\_A10 BIT\_A11 BIT\_A12 BIT\_A13  
BIT\_A14 BIT\_A15  
BIT\_B0 BIT\_B1 BIT\_B2 BIT\_B3 BIT\_B4 BIT\_B5 BIT\_B6  
BIT\_B7 BIT\_B8 BIT\_B9 BIT\_B10 BIT\_B11 BIT\_B12 BIT\_B13  
BIT\_B14 BIT\_B15

### 16.3.4 メモリ変数、メモリビット変数及びスタック変数の記述規則

- ・ シンボル、ラベル、ビットシンボル名及びスタック(S)を必ず、[]または{}で囲って記述してください。
- ・ 変数名と括弧の間にはスペースまたはタブが記述できます。
- ・ アドレッシングモードを指定する際は、アドレッシングモード指定子及びアドレッシングモード記述を含めて、[]または{}で囲って記述してください。
- ・ メモリ変数にサイズ指定子を記述する場合は、括弧の外にサイズ指定子を記述してください。

#### 記述例

```

        .SECTION memory, DATA
mem:      .BLKB    1
work:    .BLKB    1
bitsym   .EQU     0, work
        .SECTION program, CODE
        [mem]    = 0
        [work].B= 0

        IF [ mem[SB] ]
            :
        ELSE
            :
        ENDIF

        IF[ bitsym ]
            :
        ELSE
            :
        ENDIF

        [S] = A
        .END
    
```

## 16.4 分岐距離の指定

構造化記述命令の分岐命令(GOTO)及びサブルーチン呼び出し・復帰(CALL・RETURN)命令の分岐先への距離を指定します。指定方法には、.NEAR属性と.FAR属性の2種類があります。

次にそれぞれの指定方法及び展開コードを説明します。

### 16.4.1 NEAR属性ラベル

同一バンク内(64Kバイト)にあることを示し、本属性のラベルが構造化記述命令のオペランドに指定された場合、アブソリュートアドレッシングでコード展開します。

#### 機能

- ・ 同一バンク内(64Kバイト)にあることを示し、本属性のラベルが構造化記述命令のオペランドに指定された場合、アブソリュートアドレッシングでコード展開します。

#### 指定方法

- ・ 指示命令".NEAR"のオペランドにラベルを記述してください。  
.NEAR ラベル[,ラベル...]

#### 注意事項

記述方法の詳細については、「指示命令」の章を参照してください。

- ・ コマンドオプション'-JN'を指定して起動した場合、指示命令".FAR"にて宣言されていない全てのラベルを.NEAR属性とします。

#### 展開コード

- ・ .NEAR属性に指定されたラベルを以下の構造化命令のオペランドに指定した場合、同一バンク内にあるラベルとしてコード展開します。

構造化記述命令	展開コード
GOTO	JMP
CALL	BSR/JSR
RETURN	RTS

- ・ CALL命令で指定されたラベルが、同一ファイル中の同一セクション内にある場合、分岐最短命令である"BSR"命令にて分岐が可能な場合は、"BSR"命令にてコード展開します。

#### 注意事項

コマンドオプション'-JN'を指定して起動した場合、指示命令".FAR"で宣言されていない全てのラベルを.NEAR属性とします。

## 16.4.2 FAR属性ラベル

他のバンク内にあることを示し、本属性のラベルが構造化記述命令のオペランドに指定された場合、アブソリュートロングアドレッシングでコード展開します。

### 機能

- ・他のバンク内にあることを示し、本属性のラベルが構造化記述命令のオペランドに指定された場合、アブソリュートロングアドレッシングでコード展開します。

### 指定方法

- ・指示命令".FAR"のオペランドにラベルを記述してください。  
`.FAR ラベル[,ラベル...]`

### 注意事項

記述方法の詳細については、「指示命令」の章を参照してください。

- ・コマンドオプション'-JF'を指定して起動した場合、指示命令".NEAR"にて宣言されていない全てのラベルを.FAR属性とします。

### 展開コード

- ・.FAR属性に指定されたラベルを以下の構造化記述命令のオペランドに指定した場合、他のバンク内にあるラベルとしてコード展開します。

構造化記述命令	展開コード
GOTO	JMPL
CALL	JSRL
RETURN	RTL

### 注意事項

コマンドオプション'-JF'を指定して起動した場合、指示命令".NEAR"にて宣言されていない全てのラベルを.FAR属性とします。



### 16.4.3 NEAR/.FAR属性なしラベルのコード生成

分岐距離指定指示命令(.NEAR/.FAR)にて分岐距離の指定がされていないラベルについては以下のようにコード展開します。

#### 展開コード

- ・ 属性が指定されていないラベルを構造化記述命令のオペランドに指定した場合、.FAR属性と同様にコード展開します。

構造化記述命令	展開コード
GOTO	JMPL
CALL	JSRL
RETURN	RTL

- ・ 構造化記述命令"GOTO"に指定したラベルが同一ファイル内でかつ同一セクションの場合、ニーモニック"BRA","BRAL"及び"JMPL"の中で最適な命令を選択してコード展開します。

#### 注意事項

コマンドオプション'-B'を指定した場合はニーモニック"BRA"又は"JMP"の中で最適な命令を選択してコード展開します。

- ・ コマンドオプション'-JN','-JF'指定時は、それぞれ.NEAR属性、.FAR属性としてコード展開します。

## 16.5 サイズ指定による式のサイズ決定

構造化記述命令を用いて記述された式をアセンブラニーモニック命令に展開する際、式に指定されたサイズに従ってコード展開します。

- ・ 式のサイズは、その式に含まれる項のサイズにより決定します。
- ・ 各項のサイズ指定が矛盾する場合、サイズエラーとなります。
- ・ コード展開時にm/xフラグ依存命令を使用する時、決定した式のサイズとコード展開される場所のデータ長宣言が矛盾する場合はワーニングを出力します。

以下に式のサイズ決定方法について説明します。ただし、乗除算式のサイズ決定方法はその他の一般式と異なるため分けて説明します。

### 16.5.1 式のサイズ決定方法（乗除算式をのぞく）

#### サイズが確定されている項が含まれる場合

式中にサイズが確定されている項が含まれる場合、式のサイズを確定している項のサイズに合わせます。

サイズが確定している項には以下の3種類があります。

##### 1 サイズ固定のレジスタ

サイズ	レジスタ
バイトサイズ	DT PG
ワードサイズ	DP DP0 DP1 DP2 DP3 PS PC S
ダブルワードサイズ	E

##### 2 サイズ指定子でサイズ指定されているメモリ変数

##### 3 リンク制御/アドレス制御指示命令でサイズ指定されているメモリ変数

#### サイズ可変なレジスタが含まれる場合

式中にデータ長宣言によりサイズ可変なレジスタが含まれる場合、式のサイズを確定したレジスタのサイズに合わせます。

サイズ可変なレジスタには以下の4種類のレジスタがあります。

レジスタ名	データ長宣言が8の場合	データ長宣言が16の場合
A B	バイトサイズ	ワードサイズ
X Y	バイトサイズ	ワードサイズ

#### サイズ未定なメモリ変数が含まれる場合

式中にサイズ未定なメモリ変数がある場合、データ長宣言の指定に従ってメモリのサイズを決定し、そのメモリサイズに式のサイズを合わせます。

以下に指定可能な式の組み合わせを示します。表中の記号は次の通りです。

- B バイト
- W ワード
- D ダブルワード

代入文

宣言長	サイズ組み合わせ
バイト(8)長	B = B
	B = B [+ -] B
	D = D
	D = D [+ -] D
ワード(16)長	B = B
	B = B [+ -] B (イミディエイトのみ)
	W = W
	W = W [+ -] W
	D = D
	D = D [+ -] D

符号/ゼロ拡張文

宣言長	サイズ組み合わせ
バイト(8)長	D = [.S .Z] B
ワード(16)長	W = [.S .Z] B
	D = [.S .Z] B
	D = [.S .Z] W

式の記述例

```
.GLB      work,glabel
.GLBB     glabelb
.LENGTH  16,16
E = [work] + 10           ;Double word size
[work]   = [work] .B + 10 ;Byte size
[work]   = [glabelb] + 10 ;Byte size
[work]   = A + 10         ;Word size
[work]   = [glabel] + 10  ;Word size
```

## 16.5.2 乗除算式のサイズ決定方法

以下に乗除算式のサイズ決定手順を優先順位の高い順に示します。

### サイズが確定している項が含まれる場合

式中にサイズが確定している項が含まれる場合、式のサイズを確定している項のサイズに合わせます。

サイズが確定している項には以下の2種類があります。

- 1 サイズ指定子にてサイズ指定されているメモリ変数
- 2 リンク制御及びアドレス制御指示命令にてサイズ指定されているメモリ変数

### サイズ可変なレジスタが含まれる場合

式中にデータ長宣言によりサイズ可変なレジスタが含まれる場合、式のサイズを確定したレジスタのサイズに合わせます。

サイズ可変なレジスタには以下の3種類のレジスタがあります。

レジスタ名	データ長宣言が8の場合	データ長宣言が16の場合
A	バイトサイズ	ワードサイズ
B	バイトサイズ	ワードサイズ
E	ワードサイズ(BL,AL)	ダブルワードサイズ

### サイズ未定なメモリ変数が含まれる場合

式中にサイズ未定なメモリ変数がある場合、データ長宣言の指定に従ってメモリのサイズを決定し、式のサイズを決定したメモリサイズに合わせます。

以下に指定可能な式の組み合わせを示します。表中の記号は次の通りです。

- B バイト
- W ワード
- D ダブルワード
- ・ 乗算代入文

宣言長	サイズ組み合わせ
バイト(8)長	$B = B \times B$ $W = B \times B$
ワード(16)長	$W = W \times W$ $D = W \times W$

- ・ 除算代入文

宣言長	サイズ組み合わせ
バイト(8)長	$B = B / B$ $B = W / B$
ワード(16)長	$W = W / W$ $W = D / W$

## 乗除算式の記述例

コメント欄に決定した式のサイズを示します。

- B   バイトサイズ
- W   ワードサイズ
- D   ダブルワードサイズ

```

.GLB    work,label
.GLBB   glabelb
.GLBW   glabelw
.GLBD   glabeld

.LENGTH 16,16

[label] = [work] * 10           ;W = W × W
[label] = A * 10                ;W = W × W
E = [work] * 10                 ;D = W × W
[work].D = [label] * 10 ;D = W × W
[glabeld] = [label] * 10        ;D = W × W

[work] = [label] / 10           ;W = W / W
[work] = A / 10                 ;W = W / W
[work] = E / 10                 ;W = D / W
[work] = [label].D / 10 ;W = D / W
[work] = [glabeld] / 10 ;W = D / W

.LENGTH 8,8

[label] = [work] * 10           ;B = B × B
[label] = A * 10                 ;B = B × B
E = [work] * 10                 ;W(BL,AL) = B × B
[work].W = [label] * 10 ;W = B × B
[glabelw] = [label] * 10        ;W = B × B

[work] = [label] / 10           ;B = B / B
[work] = A / 10                 ;B = B / B
[glabelb] = [work] / 10 ;B = B / B
[work] = E / 10                 ;B = W(BL,AL) / B
[work] = [label].W / 10 ;B = W / B
[work] = [glabelw] / 10 ;B = W / B

.END

```

## 16.6 構造化記述命令

以降に構造化記述命令の記述規則を説明します。

## IF

### 記述形式

```
IF (条件式)
  ボディ1
[ ELIF (条件式)
  ボディ2 ]
[ ELSE
  ボディ3 ]
ENDIF
```

#### 条件式の記述形式

- ・ 式 [比較演算子 定数 メモリ]
- ・ ビット変数 [比較演算子 1 0]
- ・ 論理演算子を使用して、上記の式を組み合わせて記述できます（ただし4組以下）。

#### 注意事項

条件式の詳細については「構造化記述構文図」を参照してください。

### 機能

- ・ IFの条件が真であるときボディ1を実行し、IFの条件式が偽かつELIFの条件式が真であるときボディ2を、いずれでもない場合にはボディ3を実行する。

#### 注意事項

条件アセンブル指示命令と混同しないように注意してください。

### 記述規則

- ・ 構造化記述命令"ELIF"とボディ2及び、構造化記述命令"ELSE"とボディ3は省略することができます。
- ・ 条件式は、構造化記述命令"IF"又は"ELIF"のオペランドに記述してください。
- ・ 構造化記述命令"IF"又は"ELIF"と条件式の間には、必ずスペース又はタブを記述してください。
- ・ 一つのIF構文内には複数の"ELIF"を記述できます。
- ・ 条件式は必ず記述してください。

### 記述例

```
IF [sym1] == 10
  :
ELIF [sym2] != 10
  :
ELSE
  IF [sym3] == 10
    :
  ENDIF
ENDIF
```

## FOR-STEP

---

### 記述形式

```
FOR ( 繰り返し式 )
  ボディ
NEXT
```

#### 繰り返し式の記述形式

- ・ループカウンタ = 初期値 TO 最終値 [STEP 増分]

#### 注意事項

---

条件式の詳細については「構造化記述構文図」を参照してください。

---

### 機能

- ・ループカウンタに初期値を設定し増分だけループカウンタを更新して行き、最終値を越えるまでボディを処理します。
- ・増加分にマイナス '-' を付加した場合、ループカウンタをダウンカウントすることができます。
- ・ループカウンタにはインデックスレジスタ X・Y 及びメモリが使用できます。
- ・"STEP 増分" を省略した場合、STEP の値を 1 として処理を行います。
- ・ボディ内に制御文 "BREAK" を記述した場合、ループを抜けることができます。
- ・ボディ内に制御文 "CONTINUE" を記述した場合、増分の処理部に移行することができます。

#### 注意事項

---

ループカウンタをボディ内にて、変更した場合、正常に処理を終了することができません。

---

### 記述規則

- ・ループカウンタ、初期値、最終値、増分はそれぞれ 1 つのみ記述してください。
- ・構造化記述命令 "FOR" と繰り返し式の間には、必ずスペース又はタブを記述してください。
- ・繰り返し式の "TO" 及び "STEP" の前後には、必ずスペース又はタブを記述してください。
- ・ボディ内には "BREAK" 又は "CONTINUE" が記述できます。

### 記述例

```
FOR X = 0 TO 10 STEP 1
  :
NEXT
```

## FOR-NEXT

---

### 記述形式

```
FOR (条件式)
  ボディ
NEXT
```

### 条件式の記述形式

- ・式 [比較演算子 定数 メモリ]
- ・ビット変数 [比較演算子 1 0]
- ・FOREVER

### 注意事項

条件式の詳細については「構造化記述構文図」を参照してください。

### 機能

- ・条件式が真である間、ボディの内容を繰り返し処理します。
- ・条件式を判定した後、ボディを処理します。
- ・条件式に制御文"FOREVER"を指定した場合、無限ループとなります。
- ・ボディ内に制御文"BREAK"を記述した場合、ループを抜けることができます。
- ・ボディ内に制御文"CONTINUE"を記述した場合、ボディの直後に分岐することができます。

### 記述規則

- ・構造化記述命令"FOR"と条件式の間には、必ずスペース又はタブを記述してください。
- ・ボディ内には"BREAK"又は"CONTINUE"が記述できます。

### 記述例

```
FOR A <.S 10
  :
NEXT
```



## DO

### 記述形式

```
DO
  ボディ
WHILE (条件式)
```

#### 条件式の記述形式

- ・式 [比較演算子 定数 メモリ]
- ・ビット変数 [比較演算子 1 0]
- ・FOREVER

#### 注意事項

条件式の詳細については「構造化記述構文図」を参照してください。

### 機能

- ・条件式が真である間、ボディの内容を繰り返し処理します。
- ・ボディを処理した後、条件式を判定します。
- ・条件式に制御文"FOREVER"を指定した場合、無限ループとなります。
- ・ボディ内に制御文"BREAK"を記述した場合、ループを抜けることができます。
- ・ボディ内に制御文"CONTINUE"を記述した場合、ボディの直後に分岐することができます。

### 記述規則

- ・構造化記述命令"WHILE"と条件式の間には、必ずスペース又はタブを記述してください。
- ・ボディ内には"BREAK"又は"CONTINUE"が記述できます。

### 記述例

```
DO
:
WHILE C==1
```

# SWITCH

## 記述形式

```
SWITCH  比較対象データ
        CASE  比較データ
            ボディ1
        [CASE  比較データ
            ボディ2]
        [DEFAULT
            ボディ3]
ENDS
```

### 比較対象データの記述形式

- ・式

### 比較データの記述形式

- ・定数

### 注意事項

式及び定数の詳細については「構造化記述構文図」を参照してください。

## 機能

- ・比較対象データと比較データが一致した場合、直後のボディを処理します。
- ・比較データと一致しなかった場合、構造化記述命令"DEFAULT"の直後のボディを処理します。
- ・ボディ内に制御文"BREAK"を記述した場合、SWITCH構文から抜けることができます。

## 記述規則

- ・構造化記述命令"SWITCH"と比較対象データとの間には、必ずスペース又はタブを記述してください。
- ・構造化記述命令"CASE"と比較データとの間には、必ずスペース又はタブを記述してください。
- ・構造化記述命令"CASE"行は、複数記述することができます。
- ・構造化記述命令"DEFAULT"は、省略することができます。
- ・構造化記述命令"DEFAULT"は、1つのSWITCH構文に1回のみ記述できます。
- ・構造化記述命令"DEFAULT"はCASEボディの間には記述できません。必ず、ENDSの直前に記述してください。

## 記述例

```
SWITCH [mem]
    CASEmode1
        JSR sub1
    CASEmode2
        JSR sub2
    DEFAULT
        JSR sub3
ENDS
```

# BREAK

---

## 記述形式

### FOR構文の記述形式

```
FOR [条件式]
  ボディ
  IF [条件式]
    BREAK
  ENDF
  ボディ
NEXT
```

### DO構文の記述形式

```
DO
  ボディ
  IF [条件式]
    BREAK
  ENDF
  ボディ
WHILE [条件式]
```

### SWITCH構文の記述形式

```
SWITCH [比較対象データ]
  CASE [比較データ]
    ボディ
    BREAK
  CASE [比較データ]
    ボディ
    BREAK
ENDS
```

## 機能

- ・ 繰り返し構文(FOR構文,DO構文)及び多岐選択構文(SWITCH構文)の処理を強制的に終了させます。  
FOR           NEXTの次の行に分岐します。  
DO            WHILEの次の行に分岐します。  
SWITCH       ENDSの次の行に分岐します。

### 記述規則

- ・ 繰り返し構文(FOR構文,DO構文)及び多岐選択構文(SWITCH構文)のボディ内のみ記述できます。

### 記述例

```
DO
  :
  IF  C==1
    BREAK
  ENDIF
  :
WHILE  FOREVER
```

## CONTINUE

---

### 記述形式

#### FOR構文の記述形式

```
FOR [条件式]
  ボディ
  IF [条件式]
    CONTINUE
  ENDIF
  ボディ
NEXT
```

#### DO構文の記述形式

```
DO
  ボディ
  IF [条件式]
    CONTINUE
  ENDIF
  ボディ
WHILE [条件式]
```

### 機能

- ・ 繰り返し構文(FOR構文,DO構文)のボディの直後へ分岐します。  
FOR NEXTの行へ分岐します。  
DO WHILEの行へ分岐します。

### 記述規則

- ・ 繰り返し構文(FOR構文,DO構文)のボディ内のみ記述できます。

### 記述例

```
DO
  :
  IF C == 1
    CONTINUE
  ENDIF
  :
  WHILE [sym] == 10
```

# FOREVER

---

## 記述形式

### FOR構文の記述形式

```
FOR    FOREVER
      ボディ
NEXT
```

### DO構文の記述形式

```
DO
      ボディ
WHILE  FOREVER
```

## 機能

- ・ 繰り返し構文(FOR構文,DO構文)の条件式を、常に真とし無限ループ処理にします。

## 記述規則

- ・ 繰り返し構文(FOR構文,DO構文)の条件式部に記述してください。
- ・ 構造化記述命令"FOR,WHILE"と"FOREVER"の間には、必ずスペース又はタブを記述してください。

## 記述例

```
FOR FOREVER
  :
  IF == 1
    BREAK
  ENDIF
NEXT
```

# GOTO

---

## 記述形式

GOTO    ラベル

## 機能

- ・ 指定したラベルに分岐します。
- ・ 指示命令".NEAR"で宣言されたラベルを指定した場合、ニーモニック"JMP"命令を生成します。
- ・ 指示命令".FAR"で宣言されたラベルを指定した場合、ニーモニック"JMPL"命令を生成します。
- ・ 指示命令".FAR"及び".NEAR"で宣言されていない同一ファイル内であつ同一セクション内のラベルを指定した場合、ニーモニック"BRA","BRAL"及び"JMPL"の内で最適なコードを生成します。
- ・ 指示命令".FAR"及び".NEAR"で宣言されていない同一ファイル内の他セクション及び他ファイルのラベルを指定した場合、コマンドオプションの指定(-JN又は-JF)に応じて、それぞれ"JMP"又は"JMPL"命令にて分岐します。コマンドオプション省略時のデフォルトは"-JF"指定となります。

## 記述規則

- ・ 構造化記述命令"GOTO"とラベルの間には、必ずスペース又はタブを記述してください。

## 記述例

```
.NEAR  lab1
.FAR   lab2
GOTO   lab1; Generated JMP  command
      :
GOTO   lab2; Generated JMPL command
```

# CALL

## 記述形式

CALL    ラベル

## 機能

- ・ 指定したラベルにサブルーチン分岐します。
- ・ 指示命令".NEAR"で宣言された同一ファイル内でかつ同一セクション内のラベルを指定した場合、ニーモニック"BSR"か"JSR"のどちらか最適な命令を生成します。
- ・ 指示命令".NEAR"で宣言された同一ファイルの他セクション及び他ファイルのラベルを指定した場合、ニーモニック"JSR"命令を生成します。
- ・ 指示命令".FAR"で宣言されたラベルを指定した場合、ニーモニック"JSRL"命令を生成分岐します。
- ・ 指示命令".FAR"及び".NEAR"で宣言されていないラベルを指定した場合、コマンドオプションの指定(-JN又は-JF)に応じて、それぞれ"JSR"又は"JSRL"命令にて分岐します。コマンドオプション省略時のデフォルトは"-JF"指定となります。

## 記述規則

- ・ 構造化記述命令"CALL"とラベルの間には、必ずスペース又はタブを記述してください。

## 記述例

```
.NEAR    lab1
.FAR            lab2
CALL    lab1        ; BSR or JSR
      :
CALL    lab2        ; JSRL
```



# RETURN

---

## 記述形式

RETURN [ラベル]

## 機能

- ・ 指定したラベルのサブルーチンから復帰します。
- ・ 指示されているラベルが指示命令".NEAR"で宣言されたラベルの場合、ニーモニック"RTS"命令を生成します。
- ・ 指示されているラベルが指示命令".FAR"で宣言されたラベルの場合、ニーモニック"RTL"命令を生成します。
- ・ 指示命令".FAR"及び".NEAR"で宣言されていないラベルを指定した場合、コマンドオプションの指定(-JN又は-JF)に応じて、それぞれ"RTS"又は"RTL"命令を生成します。
- ・ ラベルを省略した場合、コマンドオプションの指定(-JN又は-JF)されたデフォルトの属性に応じて、それぞれ"RTS"又は"RTL"命令を生成します。コマンドオプション省略時のデフォルトは"-JF"指定となります。

## 記述規則

- ・ 構造化記述命令"RETURN"とラベルの間には、必ずスペース又はタブを記述してください。

## 記述例

```
.FAR    lab1
lab1:
        :
        RETURN    lab1
lab2:
        :
        RETURN
```

## 代入文

---

### 記述形式

左辺 = 右辺

注意事項

**注意事項**

右辺及び左辺の詳細については「構造化記述構文図」を参照してください。

### 機能

- ・ 右辺を左辺に代入します。

### 記述規則

- ・ 転送元を右辺に記述し、転送先を左辺に記述してください。
- ・ メモリビット変数・レジスタビット変数・フラグ変数への代入は、0又は1の値を持つもののみ記述できます。

### 記述例

```
C = 1
BIT_A = 1
[bitsym] = 1
X = Y
[lab] = 10
[wok1] = [wok2]
```

## 16.7 構造化記述内でのみ有効となる指示命令

AS79では、構造化記述内にのみ記述できる指示命令があります。次に、その概要を示します。

.GLBB、.GLBW、.GLBD

サイズ指定つきでグローバルラベルシンボルを定義します。

.NEAR

オペランドに指定したシンボルが、同一バンク内にあるものとしてコード生成します。

.FAR

オペランドに指定したシンボルが、他のバンクにあるものとしてコード生成します。

### 指示命令の詳細

以降に、構造化記述内でのみ有効となる指示命令の記述方法について示します。

## .FAR

### 機能

- ・ オペランドに指定したシンボルが、他のバンク内にあることを宣言します。
- ・ 本指示命令は、指示命令".GLB"の機能も同時に宣言します。
- ・ 指定されたシンボルを構造化記述命令"GOTO"や"CALL"及び"RETURN" のオペランドに記述した場合それぞれ以下のアブソリュートロングアドレッシング出力コードで展開します。

構造化記述命令	展開コード
GOTO	JMPL
CALL	JSRL
RETURN	RTL

- ・ コマンドオプション'-JF'を使用することにより、指示命令".NEAR"にて定義されていない全てのシンボルを.FAR属性にすることができます。但し、この場合にはグローバルの属性を持ちません。

### 記述形式

.FAR シンボル [ , シンボル . . . ]

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 複数のシンボルを指定する場合は、シンボルをカンマで区切って記述してください。

### 注意事項

本指示命令は、指示命令".GLB"機能も同時に宣言するため宣言されたシンボルはパブリックシンボル又は外部参照シンボルとなります。

### 記述例

```
.FAR      NSYM1 , NSYM2 , NSYM3
:
GOTO     NSYM1      ; JMPL
CALL     NSYM2      ; JSRL
MSYM2 :
RETURN   NSYM2      ; RTL
```

## .GLBB / .GLBW / .GLBD

### 機能

- ・オペランドに指定したラベル及びシンボルが、グローバルであることを宣言します。
- ・オペランドに指定したラベル及びシンボルで、ファイル内で定義されているものは、外部から参照できるように処理します。
- ・オペランドに指定したラベル及びシンボルで、ファイル内で定義されていないものは、外部のファイルで定義されているものとして処理します。
- ・本指示命令は、同時に以下のようなサイズ指定を行うことができます。

指示命令	指定サイズ
.GLBB	バイト
.GLBW	ワード
.GLBD	ダブルワード

### 注意事項

同一シンボルに対して複数のサイズ宣言を行った場合には、エラーとなります。

### 記述形式

```
.GLBB (名前) [, (名前) ...]
.GLBW (名前) [, (名前) ...]
.GLBD (名前) [, (名前) ...]
```

### 記述規則

- ・指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・オペランドにグローバルラベルとするラベル名を記述します。
- ・オペランドにグローバルシンボルとするシンボル名を記述します。
- ・オペランドに複数のラベル及びシンボル名を記述する場合は、カンマ(,)で区切って記述してください。
- ・オペランドにビットシンボルは記述できません。

### 記述例

```
.GLBB      name1
.GLBW      name2
.GLBD      name3
.GLBD      name4
.SECTION program
ADC        A, name1
[name4] = [name1] ;Byte size
[name4] = [name2] ;Word size
[name4] = [name3] ;Doubleword size
```

## .NEAR

### 機能

- ・ オペランドに指定したシンボルが、同一バンク内にあることを宣言します。
- ・ 本指示命令は、指示命令".GLB"の機能も同時に宣言します。
- ・ 指定されたシンボルを構造化記述命令"GOTO"や"CALL"及び"RETURN" のオペランドに記述した場合それぞれ以下のアプソリュートアドレッシング出力コードで展開します。

構造化記述命令	展開コード
GOTO	JMP
CALL	BSR/JSR
RETURN	RTS

- ・ コマンドオプション'-JN'を使用することにより、指示命令".FAR"にて定義されていない全てのシンボルを.NEAR属性にすることができます。但し、この場合にはグローバルの属性は持ちません。

### 記述形式

.NEAR シンボル [ , シンボル . . . ]

### 記述規則

- ・ 指示命令とオペランドの間には、必ずスペース又はタブを記述してください。
- ・ 複数のシンボルを指定する場合は、シンボルをカンマで区切って記述してください。

### 注意事項

本指示命令は、指示命令".GLB"機能も同時に宣言するため宣言されたシンボルはパブリックシンボル又は外部参照シンボルとなります。

### 記述例

```
.NEAR    NSYM1 , NSYM2 , NSYM3
:
GOTONSYM1    ; JMP
CALLNSYM2    ; BSR/JSR
CALLNSYM3    ; JSR
:
NSYM2:
:
RETURN@NSYM2    ; RTS
```

## 16.8 構造化記述命令構文図

AS79で記述可能な構造化記述命令の文法を構文図で示します。

### 構文図記号一覧

記号	内容
A	アキュムレータA
AL	アキュムレータAの下位8ビット
B	アキュムレータB
BL	アキュムレータBの下位8ビット
E	アキュムレータE
X	インデックスレジスタX
Y	インデックスレジスタY
DP	ダイレクトページレジスタ (拡張ダイレクトページレジスタとの併用はできません)
DP0 ~ 3	拡張ダイレクトページレジスタ (ダイレクトページレジスタとの併用はできません)
DT	データバンクレジスタ
PS	プロセッサステータスレジスタ
S	スタックポインタ
PG	プログラムバンクレジスタ
C	キャリーフラグ
Z	ゼロフラグ
I	割り込み禁止フラグ
D	10進演算モードフラグ
XF	インデックスレジスタ長選択フラグ
M	データ長選択フラグ
V	オーバーフローフラグ
N	ネガティブフラグ
BIT_A	アキュムレータAビット変数
BIT_B	アキュムレータBビット変数
.B	バイト演算サイズ指定子

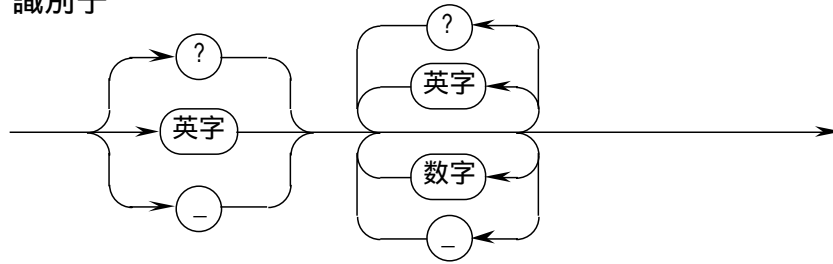
## 構造化記述

記号	内容
.W	ワード演算サイズ指定子
.D	ダブルワード演算サイズ指定子
.S	符号付き演算指定子
.C	キャリー付き演算指定子
.L	論理シフト指定子
.A	算術シフト指定子
.R	ローテート指定子
DP[+]:	ダイレクトアドレッシングモード指定子 ( DP0: ~ DP3:を含む ) ただし、8ビットオフセットモード(DP:) と6ビットオフセットモード(DP0 ~ 3:の併用はできません。
DT[+]:	アブソリュートアドレッシングモード指定子
LG[+]:	アブソリュートロングアドレッシングモード指定子

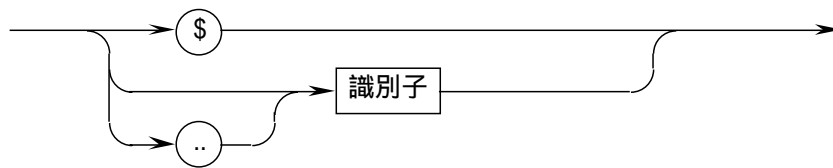


# 構造化構文図

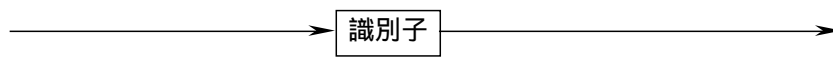
識別子



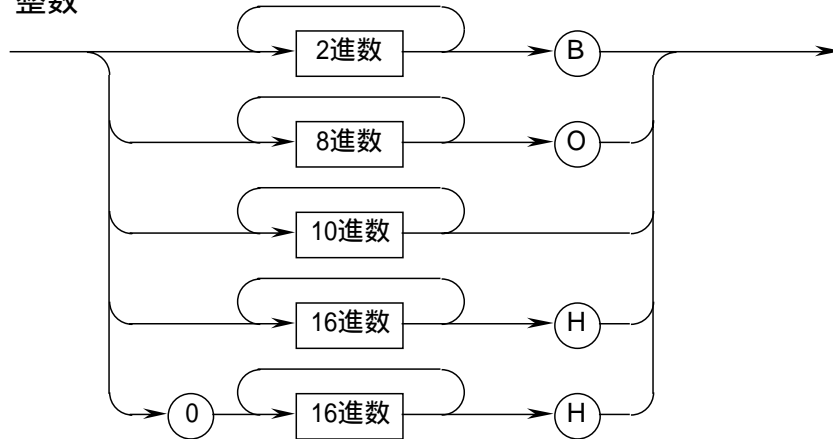
シンボル



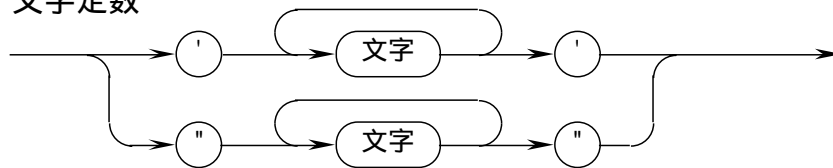
ビットシンボル



整数



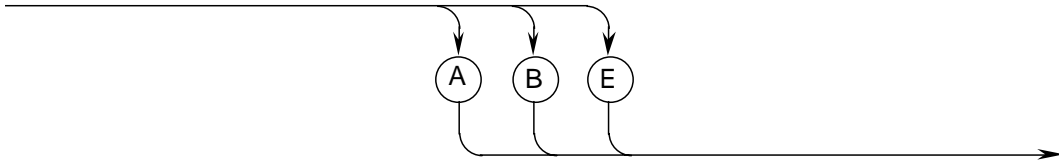
文字定数



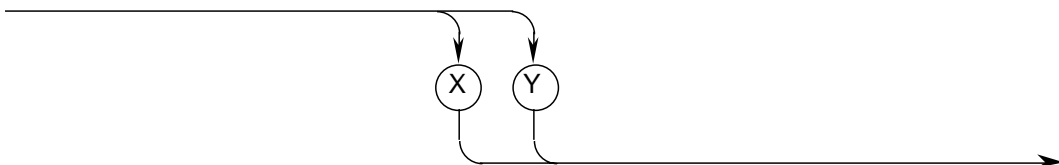


# 構造化構文図

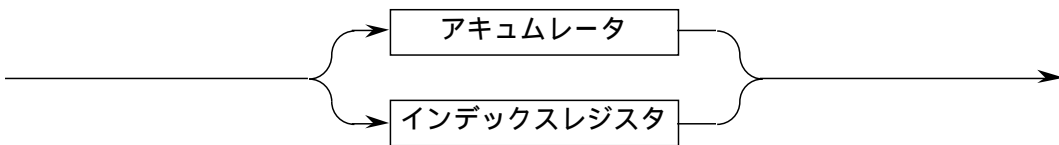
アキュムレータ



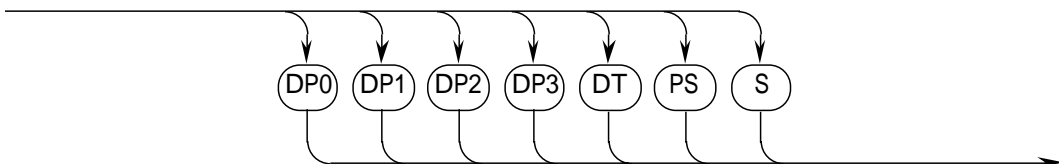
インデックスレジスタ



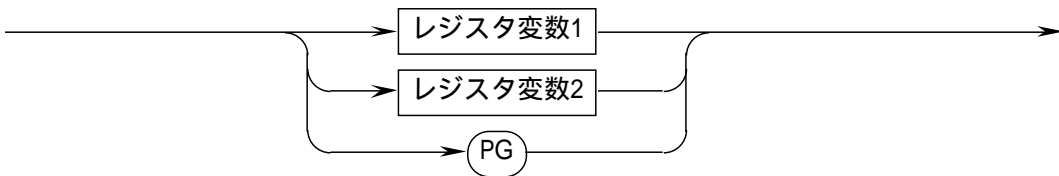
レジスタ変数1



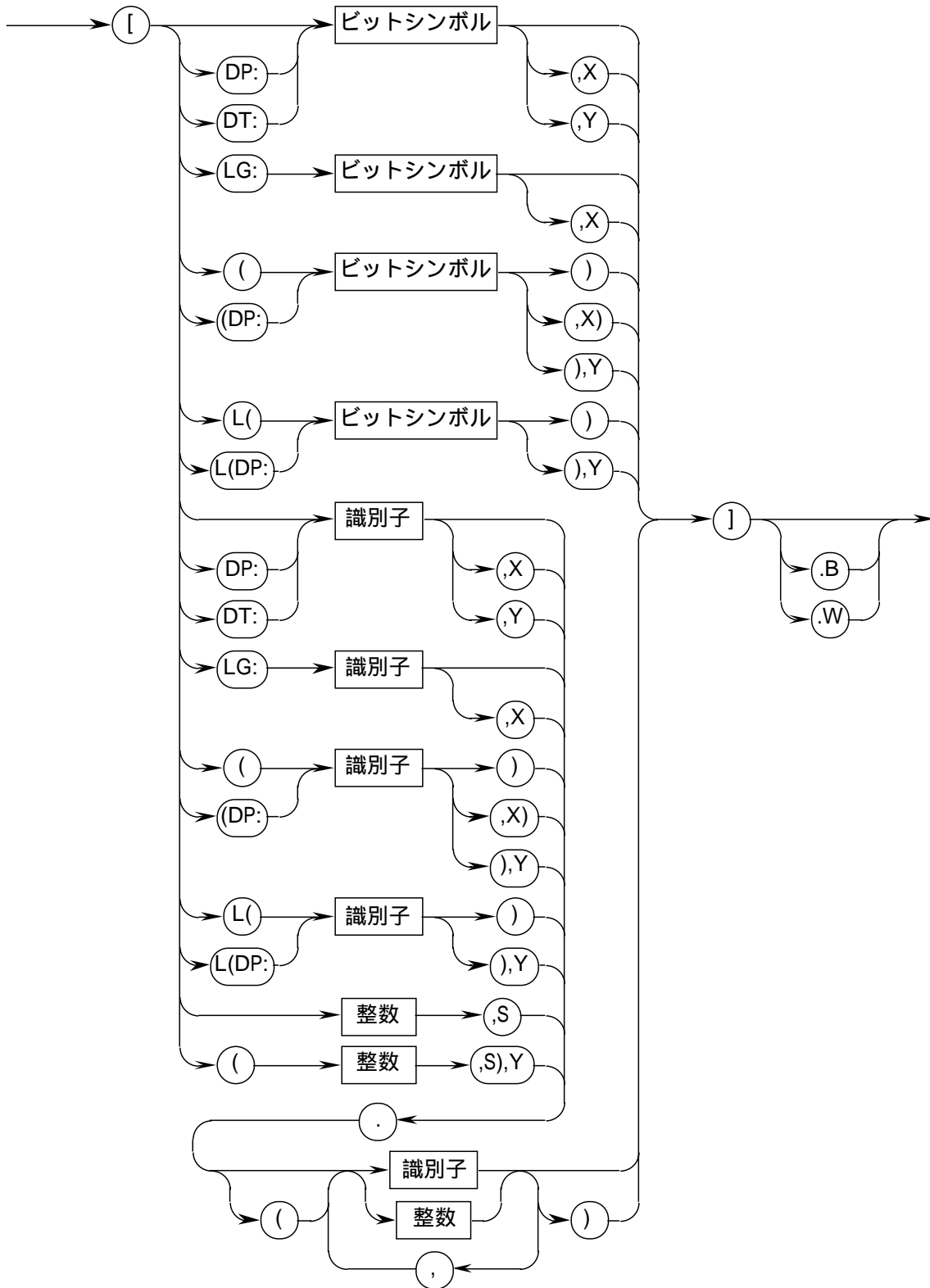
レジスタ変数2



レジスタ変数

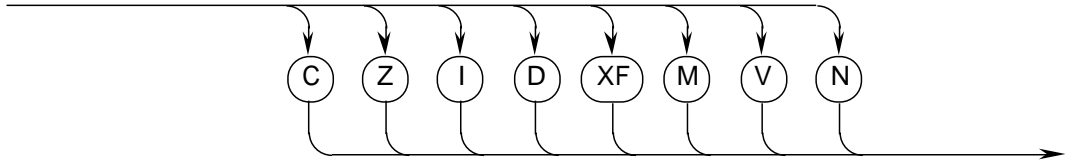


メモリビット変数

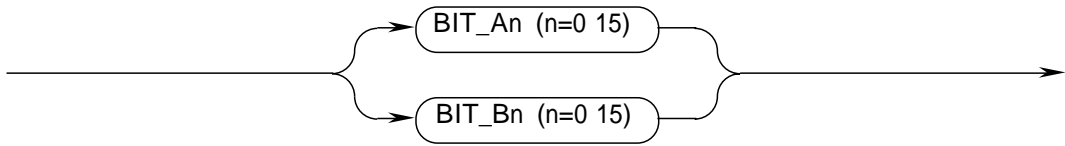


# 構造化構文図

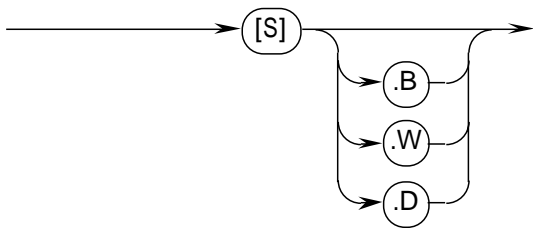
## フラグ変数



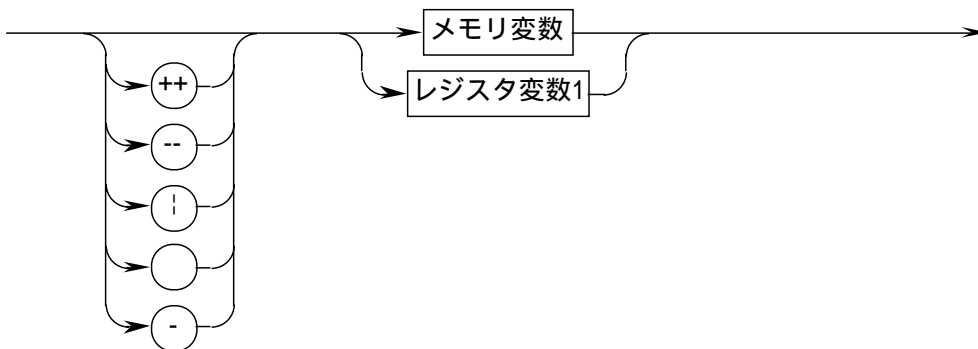
## レジスタビット変数



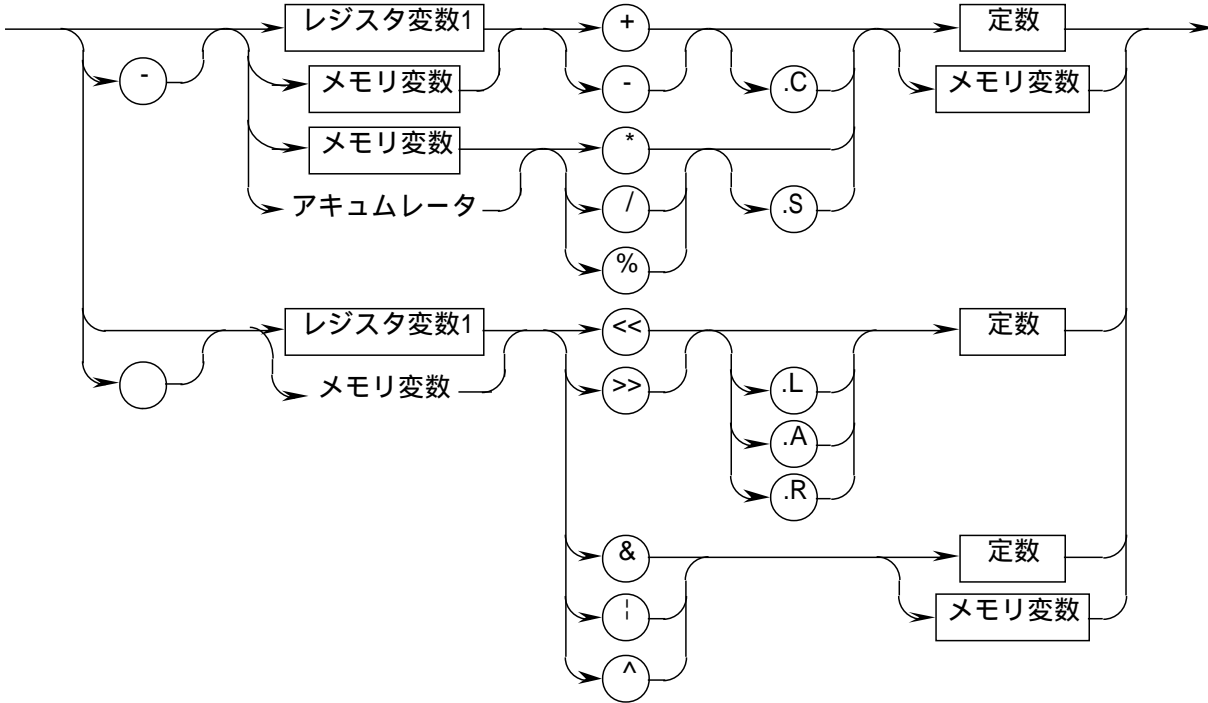
## スタック



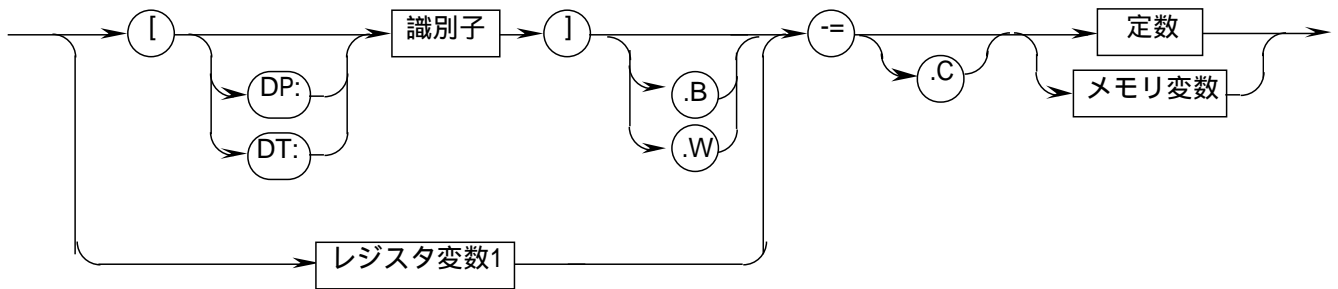
## 単項演算式



二項演算式

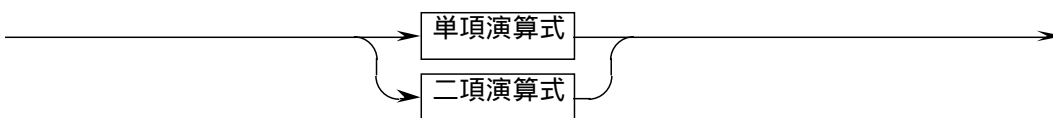


二項演算式2

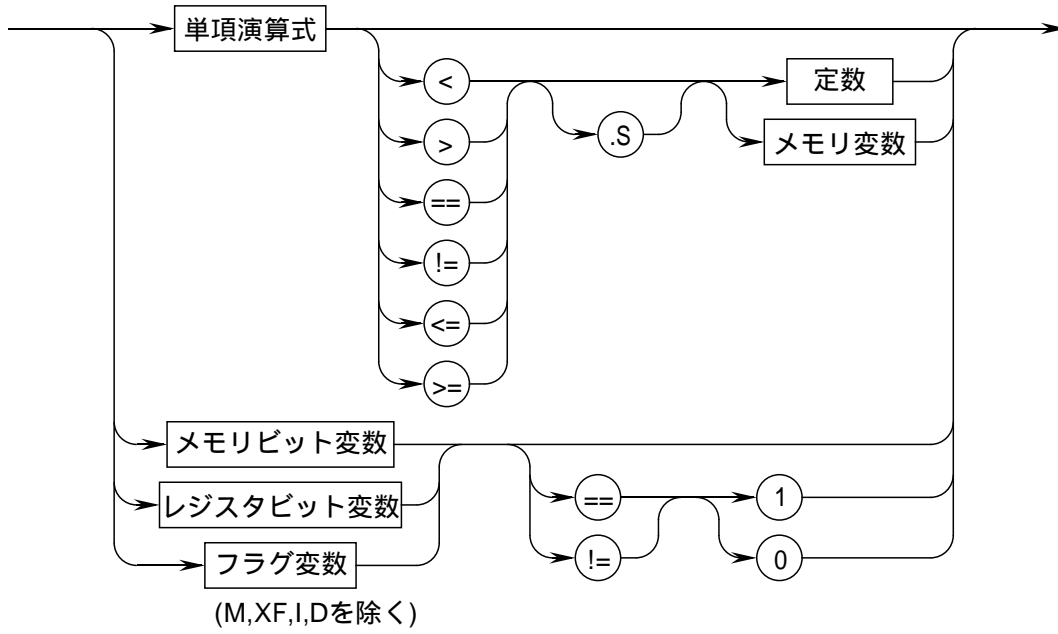


\* DP : DT : 省略時、識別子は DP SYM、DT SYM 宣言されている必要がある

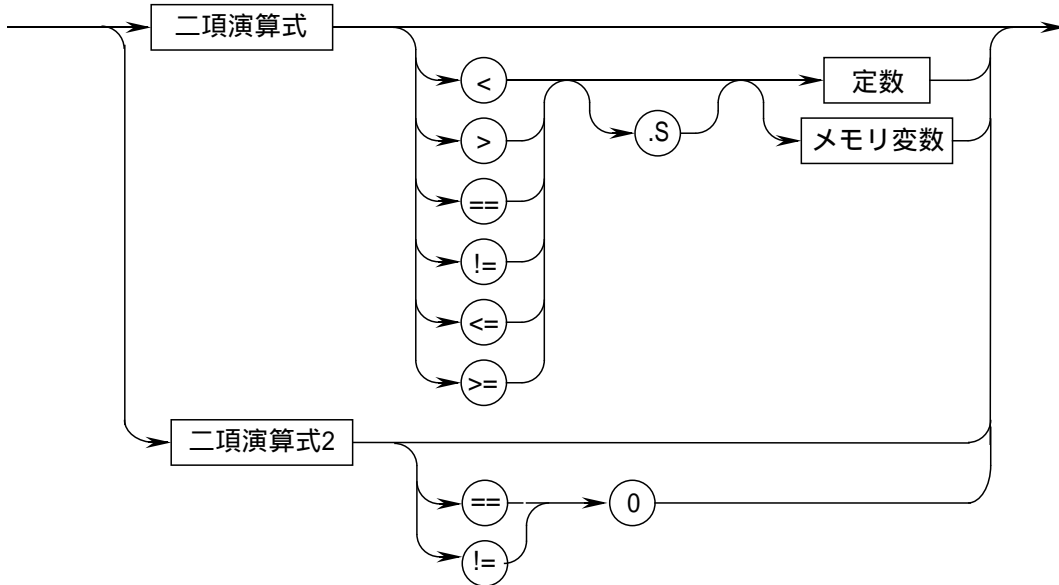
式



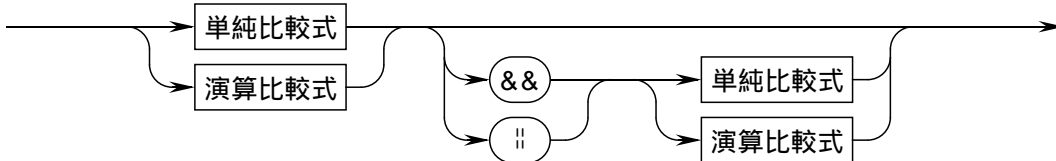
単純比較式



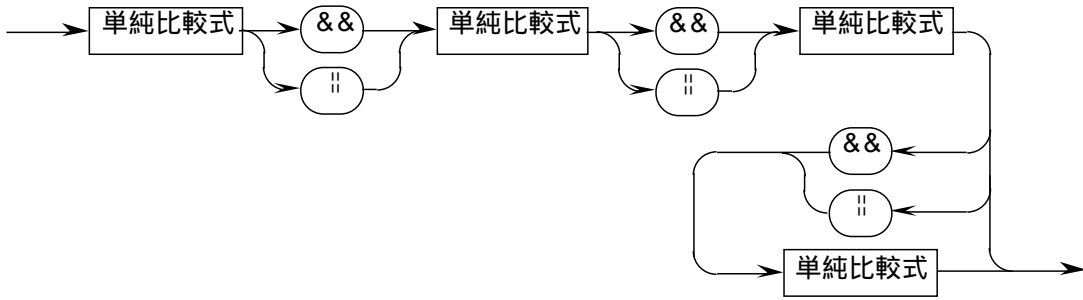
演算比較式



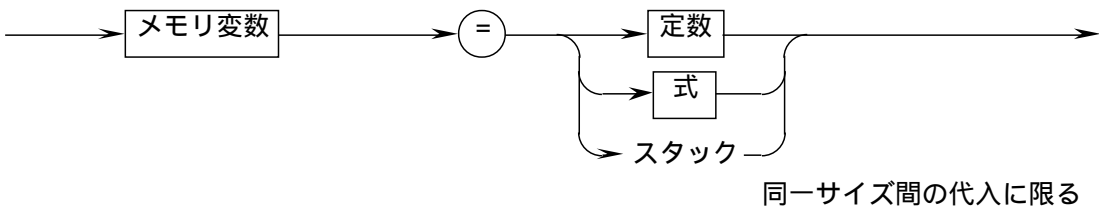
条件式1



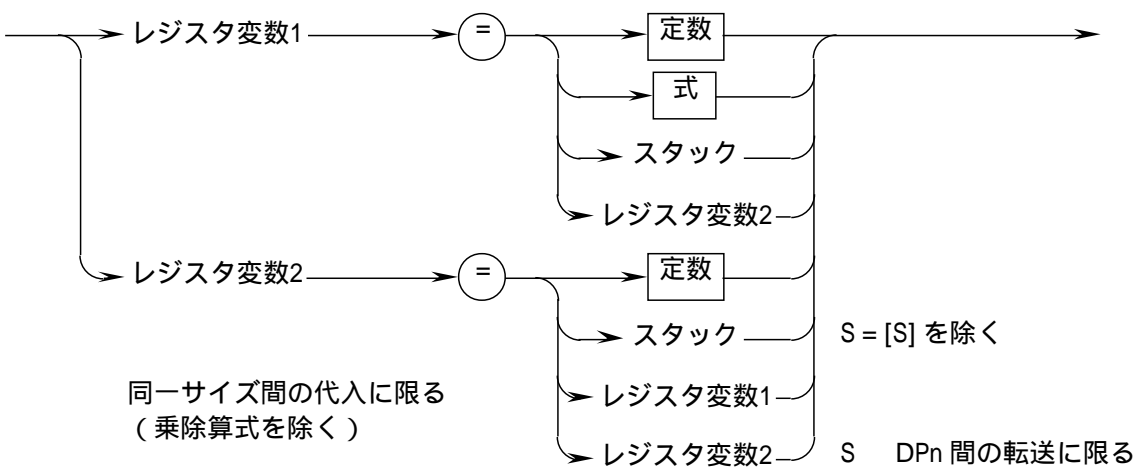
条件式2



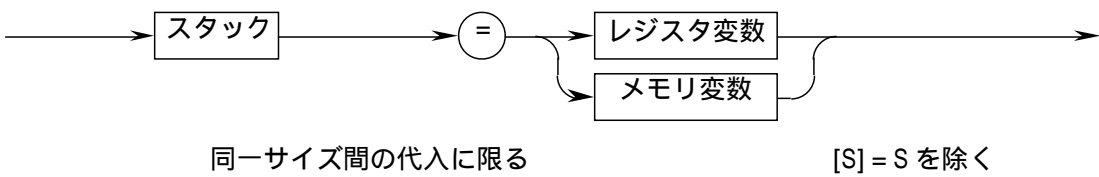
メモリ変数代入文



レジスタ変数代入文

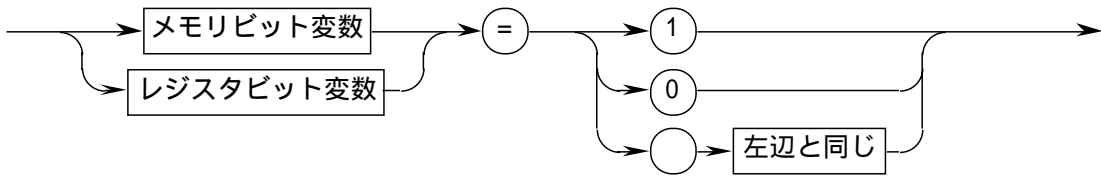


スタック代入文

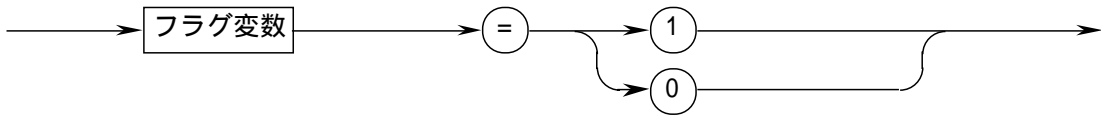




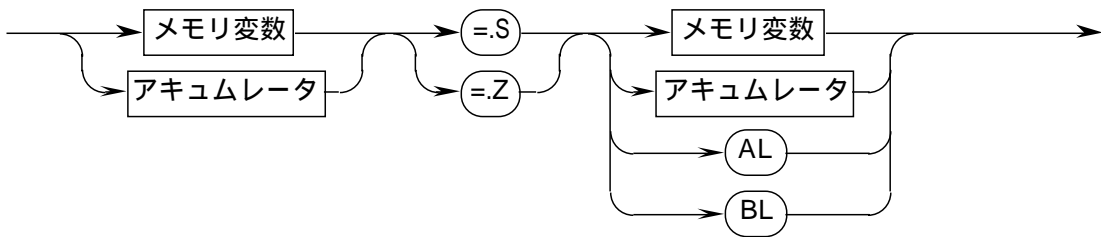
ビット変数代入文



フラグ変数代入文

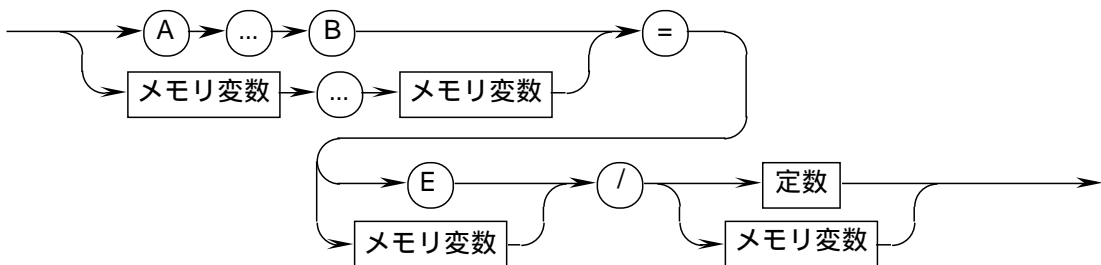


符号/ゼロ拡張文

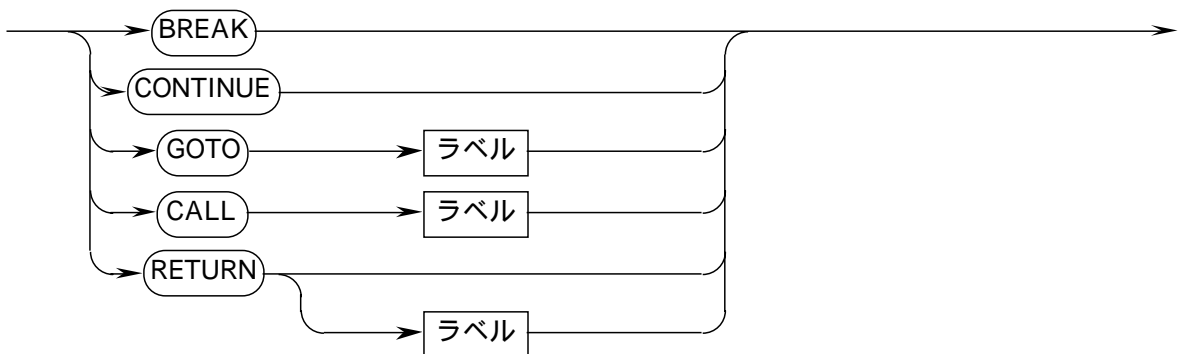


(ただし、左辺のサイズ > 右辺のサイズ)

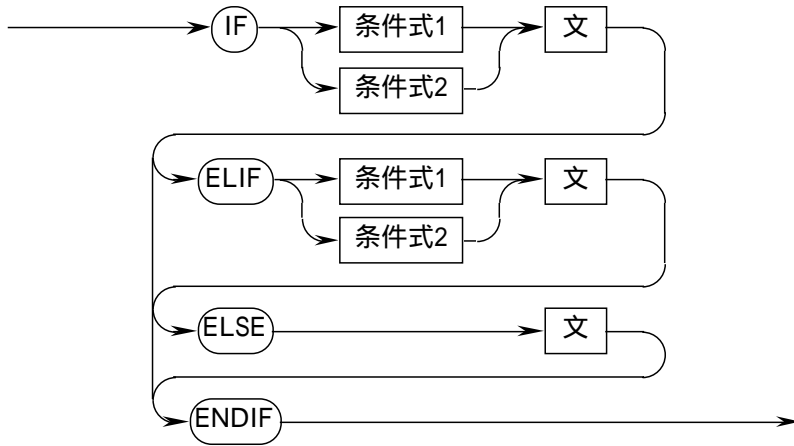
除算の商余代入文



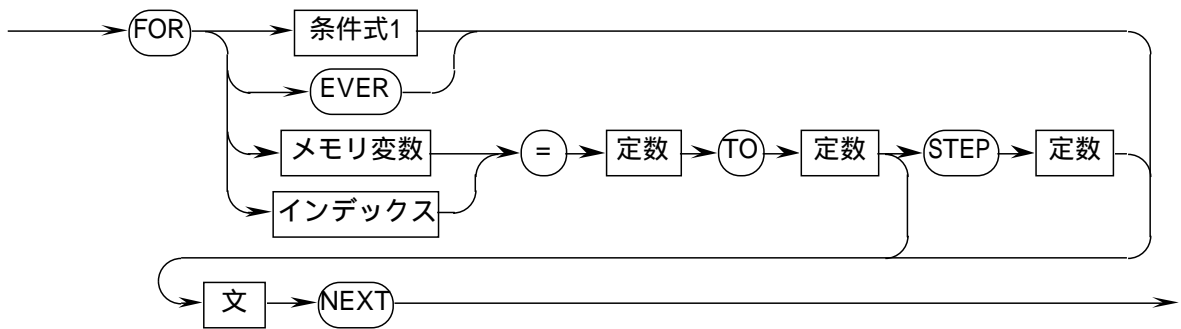
制御文



IF文



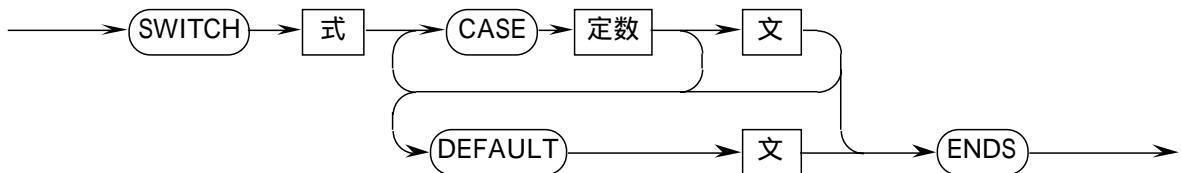
FOR文



DO文



SWITCH文



## 第17章 RASM77用アセンブリプログラムとの互換性について

---

7700ファミリ用アセンブラRASM77と7900シリーズ用アセンブラAS79には、以降に示すような仕様の相違があります。

7700ファミリのマイクロコンピュータ用に開発されたアプリケーションを7900シリーズのマイクロコンピュータのアプリケーションプログラムに流用される場合は、以降の内容をご参照のうえプログラムの必要な個所を変更してください。

それぞれの機能の詳細については、各アセンブラのユーザズマニュアルを参照してください。

### 17.1 算術演算子

演算の優先順位が異なります。したがって、式の記述内容によっては、演算結果が異なります。

#### AS79

二項演算子に一般の四則演算規則と同様の演算優先順位を設定しています。

#### RASM77

二項演算子に演算優先順位の設定はありません。複数の二項演算子が記述されている式の演算は左から順に行われます。

次に演算結果の異なる式の記述例を示します。

AS79	RASM77
3   4+5*6	3   4+5*6
=3   4+30	=7+5*6
=3   34	=12*6
=35	=72

## 17.2 アドレッシング指定子とEQUシンボル (オフセット計算)

ダイレクトアドレッシングモードまたは、アブソリュートアドレッシングモードが指定されている場合のオペランドの計算方法が異なります。

### AS79

特別に指定をしない限り、オペランドの種類に関わらず、オペランドとダイレクトページレジスタまたは、データページレジスタとのオフセット値を計算し、その結果をオペランドコードとして生成します。詳細はユーザーズマニュアルを参照してください。

### RASM77

オペランドが疑似命令'.EQU'で定義されたシンボルの場合、オペランドとダイレクトページレジスタ又は、データページレジスタとのオフセット計算を行いません。

次に、ダイレクトアドレッシングモードでオペランドが'.EQU'で定義されたシンボルを記述した場合のオペランド値の相違を示します。

AS79	RASM77
sym .EQU 34h	sym .EQU 34h
.DP 12h	.DP 12h
LDA A,DP:sym	LDA A,DP:sym
;CODE is 1A22	;CODE is A534

## 17.3 ロケーションシンボル

ロケーションシンボルが異なります。次にロケーションシンボルを使用したプログラムの記述例を示します。

### AS79

```
BCC $+3 ;If C flag is 0, branch to NOP
INX
NOP
```

### RASM77

```
BCC *+3 ;If C flag is 0, branch to NOP
INX
NOP
```

## 17.4 アセンブル指示命令

AS79では一部の指示命令がRASM77とは異なります。

### RASM77から削除された指示命令

.ERROR	.LIB	.OBJ	.LISTM	.NLISTM
.PROGNAME	.ENDIO	.ENDRAM	.IO	.RAM
.PROCINT	.PROCMAIN	.PROCSUB		

### RASM77とは機能が異なる指示命令

.EVEN	.PUB	.EXT	.DPEXT	.DTEXT	.COL
.LINE	.LIST	.NLIST			

次に、指示命令毎に相違内容を説明します。

#### .EVEN

アドレス補正命令'.EVEN'は、'.ALIGN'に置き換えてください。

AS79	RASM77
.ALIGN 2	.EVEN

#### .PUB

パブリック指定命令'.PUB'は、'.GLB'に置き換えてください。

AS79	RASM77
.GLB esym	.PUBesym

#### .EXT

外部参照指定命令'.EXT'は、'.GLB'に置き換えてください。

AS79	RASM77
.GLB esym	.EXTesym
.DPEXT	

ダイレクトアドレッシング外部参照指定命令'.DPEXT'は、'.GLB'及び'.DPSYM'に置き換えてください。

#### 注意事項

AS79では、「ダイレクトページ名ラベル(directpage\_name)」に相当する機能はありません。

AS79	RASM77
.GLB esym	.DPEXT esym
.DPSYM esym	

## RASM77用プログラムとの互換性について

### .DTEXT

アブソリュートアドレッシング外部参照指定'.DTEXT'は、'.GLB'及び'.DTSYM'に置き換えてください。

#### 注意事項

AS79では、「バンク名ラベル(databank\_name)」に相当する機能はありません。

AS79		RASM77	
.GLB	esym	.DTEXT	esym
.DTSYM	esym		

### .COL

リストファイルカラム数指定命令'.COL'は、'.FORM'に置き換えてください。  
.FORM命令でリストファイルのカラム数を指定する場合は、第2オペランドにカラム数を指定してください。

AS79		RASM77	
.FORM	,100	.COL	100

### .LINE

リストファイル行数指定命令'.LINE'は、'.FORM'に置き換えてください。  
.FORM命令でリストファイルの行数を指定する場合は、第1オペランドに行数を指定してください。

AS79		RASM77	
.FORM	60	.LINE	60

### .LIST/.NLIST

リスト出力制御'.LIST'及び'.NLIST'は、それぞれ".LIST ON"、".LIST OFF"に置き換えてください。

AS79		RASM77	
.LIST	OFF	.NLIST	
:		:	
.LIST	ON	.LIST	

## 17.5 マクロ命令に関する相違点

### マクロ名

マクロ名の直後のコロン(:)を削除してください。

AS79	RASM77
Fclr .MACRO	Fclr: .MACRO
CLP C,Z,I	CLP C,Z,I
.ENDM	.ENDM

### 繰り返しマクロ命令

繰り返しマクロ命令'.REPEAT ~ .ENDM'は、'.MREPEAT ~ .ENDR'に置き換えてください。

RASM77の繰り返しマクロ命令'.REPEATC'及び'.REPEATI'は、AS79では対応していません。

AS79	RASM77
.MREPEAT 3	.REPEAT 3
NOF	NOF
.ENDM	.ENDM

### 文字列連結演算子に関する相違点

文字列連結演算子'\$'は、'@'に置き換えてください。

AS79	RASM77
Obyte .MACRO	Obyte: .MACRO
.BYTE Oval	.BYTE Oval
.ENDM	.ENDM

### ビットクリア、ビットセット命令

7700シリーズのCLB、SEB命令は、7900シリーズのANDM、ORAMに包含されました。

AS79ではこのCLB、SEB命令の記述規則と同様の命令をプリデファインドマクロ機能にて搭載しています。

また、ANDM、ORAM命令は7900のmフラグの内容によって動作が異なりますが、mフラグの内容に依存しないANDMB、ORAMB命令を展開するCLBB、SEBB命令も使用できます。

以下にこれらプリデファインドマクロ命令の記述形式、および7900命令への展開形式を示します。

am=アドレッシングモード

命令名	データ長	am	マクロ命令記述形式 7900展開形式
CLB	.B/.W	DIR	CLB[.B/.W] #imm8/16,[DPn:]zz
			ANDM[.B/.W] [DPn:]zz,#imm8/16
			ABS CLB[.B/.W] #imm8/16,[DT:]mml1
			ANDM[.B/.W] [DT:]mml1,#imm/16
CLBB	.B	DIR	CLBB[.B] #imm8,[DPn:]zz
			ANDMB[.B] [DPn:]zz,#imm8
			ABS CLB[.B] #imm8,[DT:]mml1
			ANDMB[.B] [DT:]mml1,#imm8
SEB	.B/.W	DIR	SEB[.B/.W] #imm8/16,[DPn:]zz
			ORAM[.B/.W] [DPn:]zz,#imm8/16
			ABS SEB[.B/.W] #imm8/16,[DT:]mml1
			ORAM[.B/.W] [DT:]mml1,#imm/16
SEBB	.B	DIR	SEBB[.B] #imm8,[DPn:]zz
			ORAMB[.B] [DPn:]zz,#imm8
			ABS SEBB[.B] #imm8,[DT:]mml1
			ORAMB[.B] [DT:]mml1,#imm8

上記のimm8 / imm16にはビットクリア、ビットセットを行うビット位置に"1"を設定した値を記述します。CLB、CLBB命令に記述されたimm8 / imm16は指定された(1を設定した)ビットをクリアするようにANDM、ANDMB命令に展開する際にAS79が値を変換して展開します。

上記の命令オペランドにはビットシンボルの記述も可能です。

#### 注意事項

なお、上記命令はデバッガでの逆アセンブル表示では7900展開形式となり、デバッガのラインアセンブルでは使用できません。



### ダイレクトページレジスタ転送命令

コマンド オプション-M8を指定（8ビットオフセットモードを選択）してAS79を起動したとき、ダイレクトページレジスタとアキュムレータA、Bとの転送命令について7700シリーズ用の記述形式を許可しています。

この場合、ダイレクトページレジスタ0を指定したとしてAS79は処理します。

以下に7700用記述形式とAS79が展開する7900シリーズ用記述形式を示します。

7700用記述形式	AS79が展開する7900用記述形式
TAD	TAD 0
TBD	TBD 0
TDA	TDA 0
TDB	TDB 0

# 第18章 7900命令一覧

## 使用記号一覧

命令一覧表で使用している記号を説明します。

記号	内容
Acc	アキュムレータAもしくはアキュムレータB (アキュムレータアドレッシング)
A	アキュムレータA (アキュムレータアドレッシング)
B	アキュムレータB (アキュムレータアドレッシング)
E	アキュムレータE (アキュムレータアドレッシング)
X	インデックスレジスタX
Y	インデックスレジスタY
S	スタックポインタ
d8	8ビットのデータ
n	ダイレクトページ番号を指定
[ DPn: ] zz	イレクトアドレッシングモードでのダイレクトページレジスタからの相 対アドレス値
[ DT: ] mml	アブソリュートアドレッシングモードでの下位16ビットのアドレス値
[ LG ] hhml	アブソリュートロングアアドレッシングモードでの24ビットのアドレ ス値
i mm	即値データ
bit	ビット番号またはビットパターン
rr	- 128 ~ +127の範囲の相対アドレス
rrll	- 32768 ~ +32767の範囲の相対アドレス

## 7900命令一覧

記号	内容
.B	バイト演算サイズ指定子
.W	ワード演算サイズ指定子
.D	ダブルワード演算サイズ指定子
[...] ]	鍵括弧内は省略可能。
IMP	インプライド
IMM	イミディエイト
b	ビット
REL	レラティブ
DIR	ダイレクト
DIR,X	ダイレクト・インデクストX
DIR,Y	ダイレクト・インデクストY
(DIR)	ダイレクト・インダイレクト
(DIR,X)	ダイレクト・インデクストX・インダイレクト
(DIR),Y	ダイレクト・インダイレクト・インデクストY
L(DIR)	ダイレクト・インダイレクトロング
L(DIR),Y	ダイレクト・インダイレクトロング・インデクストY
ABS	アブソリュート
ABS,X	アブソリュート・インデクストX
ABS,Y	アブソリュート・インデクストY
ABL	アブソリュートロング
ABL,X	アブソリュートロング・インデクストX
(ABS)	アブソリュート・インダイレクト
L(ABS)	アブソリュート・インダイレクトロング
SR	スタックポインタ・レラティブ
(SR),Y	スタックポインタ・レラティブ・インダイレクト・インデクストY
BLK	ブロック転送

## 7900命令一覧

命令名	データ長	アドレスモード	記述形式
ABS	.B/.W	Acc	ABS[.B/.W] Acc
ABSD	.D	E	ABSD[.D] E
ADC	.B/.W	IMM	ADC[.B/.W] Acc,#imm8/16
		DIR	ADC[.B/.W] Acc,[DPn:]zz
		DIR,X	ADC[.B/.W] Acc,[DPn:]zz,X
		(DIR)	ADC[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	ADC[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	ADC[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	ADC[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	ADC[.B/.W] Acc,L([DPn:]zz),Y
		ABS	ADC[.B/.W] Acc,[DT:]mml1
		ABS,X	ADC[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	ADC[.B/.W] Acc,[DT:]mml1,Y
		ABL	ADC[.B/.W] Acc,[LG:]hhmml1
		ABL,X	ADC[.B/.W] Acc,[LG:]hhmml1,X
		SR	ADC[.B/.W] Acc,d8,S
(SR),Y	ADC[.B/.W] Acc,(d8,S),Y		
ADCB	.B	IMM	ADCB[.B] Acc,#imm8
ADCD	.D	IMM	ADCD[.D] E,#imm32
		DIR	ADCD[.D] E,[DPn:]zz
		DIR,X	ADCD[.D] E,[DPn:]zz,X
		(DIR)	ADCD[.D] E,([DPn:]zz)
		(DIR,X)	ADCD[.D] E,([DPn:]zz,X)
		(DIR),Y	ADCD[.D] E,([DPn:]zz),Y
		L(DIR)	ADCD[.D] E,L([DPn:]zz)
		L(DIR),Y	ADCD[.D] E,L([DPn:]zz),Y
		ABS	ADCD[.D] E,[DT:]mml1
		ABS,X	ADCD[.D] E,[DT:]mml1,X
		ABS,Y	ADCD[.D] E,[DT:]mml1,Y
		ABL	ADCD[.D] E,[LG:]hhmml1
		ABL,X	ADCD[.D] E,[LG:]hhmml1,X
		SR	ADCD[.D] E,d8,S
(SR),Y	ADCD[.D] E,(d8,S),Y		

## 7900命令一覧

命令名	データ長	アドレスモード	記述形式
ADD	.B/.W	IMM	ADD[.B/.W] Acc,#imm8/16
		DIR	ADD[.B/.W] Acc,[DPn:]zz
		DIR,X	ADD[.B/.W] Acc,[DPn:]zz,X
		(DIR)	ADD[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	ADD[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	ADD[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	ADD[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	ADD[.B/.W] Acc,L([DPn:]zz),Y
		ABS	ADD[.B/.W] Acc,[DT:]mml1
		ABS,X	ADD[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	ADD[.B/.W] Acc,[DT:]mml1,Y
		ABL	ADD[.B/.W] Acc,[LG:]hhmml1
		ABL,X	ADD[.B/.W] Acc,[LG:]hhmml1,X
		SR	ADD[.B/.W] Acc,d8,S
(SR),Y	ADD[.B/.W] Acc,(d8,S),Y		
ADDB	.B	IMM	ADDB[.B] Acc,#imm8
ADDD	.D	IMM	ADDD[.D] E,#imm32
		DIR	ADDD[.D] E,[DPn:]zz
		DIR,X	ADDD[.D] E,[DPn:]zz,X
		(DIR)	ADDD[.D] E,([DPn:]zz)
		(DIR,X)	ADDD[.D] E,([DPn:]zz,X)
		(DIR),Y	ADDD[.D] E,([DPn:]zz),Y
		L(DIR)	ADDD[.D] E,L([DPn:]zz)
		L(DIR),Y	ADDD[.D] E,L([DPn:]zz),Y
		ABS	ADDD[.D] E,[DT:]mml1
		ABS,X	ADDD[.D] E,[DT:]mml1,X
		ABS,Y	ADDD[.D] E,[DT:]mml1,Y
		ABL	ADDD[.D] E,[LG:]hhmml1
		ABL,X	ADDD[.D] E,[LG:]hhmml1,X
		SR	ADDD[.D] E,d8,S
(SR),Y	ADDD[.D] E,(d8,S),Y		

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
ADDM	.B/.W	DIR	ADDM[.B/.W] [DPn:]zz,#imm8/16
		ABS	ADDM[.B/.W] [DT:]mml1,#imm8/16
ADDMB	.B	DIR	ADDMB[.B] [DPn:]zz,#imm8
		ABS	ADDMB[.B] [DT:]mml1,#imm8
ADDMD	.D	DIR	ADDMD[.D] [DPn:]zz,#imm32
		ABS	ADDMD[.D] [DT:]mml1,#imm32
ADDS	.W	IMM	ADDS[.W] #imm8
ADDX	.B/.W	IMM	ADDX[.B/.W] #imm5
ADDY	.B/.W	IMM	ADDY[.B/.W] #imm5
AND	.B/.W	IMM	AND[.B/.W] Acc,#imm8/16
		DIR	AND[.B/.W] Acc,[DPn:]zz
		DIR,X	AND[.B/.W] Acc,[DPn:]zz,X
		(DIR)	AND[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	AND[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	AND[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	AND[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	AND[.B/.W] Acc,L([DPn:]zz),Y
		ABS	AND[.B/.W] Acc,[DT:]mml1
		ABS,X	AND[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	AND[.B/.W] Acc,[DT:]mml1,Y
		ABL	AND[.B/.W] Acc,[LG:]hhmml1
		ABL,X	AND[.B/.W] Acc,[LG:]hhmml1,X
		SR	AND[.B/.W] Acc,d8,S
(SR),Y	AND[.B/.W] Acc,(d8,S),Y		
ANDB	.B	IMM	ANDB[.B] Acc,#imm8
ANDM	.B/.W	DIR	ANDM[.B/.W] [DPn:]zz,#imm8/16
		ABS	ANDM[.B/.W] [DT:]mml1,#imm8/16
ANDMB	.B	DIR	ANDMB[.B] [DPn:]zz,#imm8
		ABS	ANDMB[.B] [DT:]mml1,#imm8
ANDMD	.D	DIR	ANDMD[.D] [DPn:]zz,#imm32
		ABS	ANDMD[.D] [DT:]mml1,#imm32

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
ASL	.B/.W	Acc	ASL[.B/.W] Acc
		DIR	ASL[.B/.W] [DPn:]zz
		DIR,X	ASL[.B/.W] [DPn:]zz,X
		ABS	ASL[.B/.W] [DT:]mml1
		ABS,X	ASL[.B/.W] [DT:]mml1,X
		A	ASL[.B/.W] A,#imm4
ASLD	.D	E	ASLD[.D] E,#imm5
ASR	.B/.W	Acc	ASR[.B/.W] Acc
		DIR	ASR[.B/.W] [DPn:]zz
		DIR,X	ASR[.B/.W] [DPn:]zz,X
		ABS	ASR[.B/.W] [DT:]mml1
		ABS,X	ASR[.B/.W] [DT:]mml1,X
		A	ASR[.B/.W] A,#imm4
ASRD	.D	E	ASRD[.D] E,#imm5
BBC	.B/.W	DIR,b,REL	BBC[.B/.W] #bit,[DPn:]zz,rr
		ABS,b,REL	BBC[.B/.W] #bit,[DT:]mml1,rr
BBCB	.B	DIR,b,REL	BBCB[.B] #bit,[DPn:]zz,rr
		ABS,b,REL	BBCB[.B] #bit,[DT:]mml1,rr
BBS	.B/.W	DIR,b,REL	BBS[.B/.W] #bit,[DPn:]zz,rr
		ABS,b,REL	BBS[.B/.W] #bit,[DT:]mml1,rr
BBSB	.B	DIR,b,REL	BBSB[.B] #bit,[DPn:]zz,rr
		ABS,b,REL	BBSB[.B] #bit,[DT:]mml1,rr
BCC		REL	BCC rr
BCS		REL	BCS rr

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
BEQ		REL	BEQ rr
BGE		REL	BGE rr
BGT		REL	BGT rr
BGTU		REL	BGTU rr
BLE		REL	BLE rr
BLEU		REL	BLEU rr
BLT		REL	BLT rr
BMI		REL	BMI rr
BNE		REL	BNE rr
BPL		REL	BPL rr
BRA		REL	BRA rr
BRAL		REL	BRAL rrl1
BRK		IMP	BRK
BSC	.B/.W	A	BSC[.B/.W] bit,A,rr
		DIR	BSC[.B/.W] bit,[DPn:]zz,rr
		ABS	BSC[.B/.W] bit,[DT:]mml1,rr
BSR		REL	BSR rr
BSS	.B/.W	A	BSS[.B/.W] bit,A,rr
		DIR	BSS[.B/.W] bit,[DPn:]zz,rr
		ABS	BSS[.B/.W] bit,[DT:]mml1,rr
BVC		REL	BVC rr
BVS		REL	BVS rr
CBEQ	.B/.W	Acc	CBEQ[.B/.W] Acc,#imm8/16,rr
		DIR	CBEQ[.B/.W][DPn:]zz,#imm8/16,rr
CBEQB	.B	Acc	CBEQB[.B] Acc,#imm8,rr
		DIR	CBEQB[.B] [DPn:]zz,#imm8,rr
CBNE	.B/.W	Acc	CBEQ[.B/.W] Acc,#imm8/16,rr
		DIR	CBEQ[.B/.W][DPn:]zz,#imm8/16,rr
CBNEB	.B	Acc	CBEQB[.B] Acc,#imm8,rr
		DIR	CBEQB[.B] [DPn:]zz,#imm8,rr



命令名	データ長	アドレッシングモード	記述形式
CLC		IMP	CLC
CLI		IMP	CLI
CLM		IMP	CLM
CLP		IMM	CLP #imm8
CLR	.B/.W	Acc	CLR[.B/.W] Acc
CLRB	.B	Acc	CLRB[.B] Acc
CLRM	.B/.W	DIR	CLRM[.B/.W] [DPn:]zz
		ABS	CLRM[.B/.W] [DT:]mml1
CLRMB	.B	DIR	CLRMB[.B] [DPn:]zz
		ABS	CLRMB[.B] [DT:]mml1
CLR X	.B/.W	IMP	CLR X[.B/.W]
CLR Y	.B/.W	IMP	CLR Y[.B/.W]
CLV		IMP	CLV
CMP	.B/.W	IMM	CMP[.B/.W] Acc,#imm8/16
		DIR	CMP[.B/.W] Acc,[DPn:]zz
		DIR,X	CMP[.B/.W] Acc,[DPn:]zz,X
		(DIR)	CMP[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	CMP[.B/.W] Acc,([DPn:]zz,X)
		[DIR),Y	CMP[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	CMP[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	CMP[.B/.W] Acc,L([DPn:]zz),Y
		ABS	CMP[.B/.W] Acc,[DT:]mml1
		ABS,X	CMP[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	CMP[.B/.W] Acc,[DT:]mml1,Y
		ABL	CMP[.B/.W] Acc,[LG:]hhmml1
		ABL,X	CMP[.B/.W] Acc,[LG:]hhmml1,X
		SR	CMP[.B/.W] Acc,d8,S
(SR),Y	CMP[.B/.W] Acc,(d8,S),Y		

CLP命令はオペランド欄にフラグ名を記述できます。

例) CLP C,Z,I,D,X,M,V,N

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
CMPB	.B	IMM	CMPB[.B] Acc, #imm8
CMPD	.D	IMM	CMPD[.D] E, #imm32
		DIR	CMPD[.D] E, [DPn:]zz
		DIR, X	CMPD[.D] E, [DPn:]zz, X
		(DIR)	CMPD[.D] E, ([DPn:]zz)
		(DIR, X)	CMPD[.D] E, ([DPn:]zz, X)
		(DIR), Y	CMPD[.D] E, ([DPn:]zz), Y
		L(DIR)	CMPD[.D] E, L([DPn:]zz)
		L(DIR), Y	CMPD[.D] E, L([DPn:]zz), Y
		ABS	CMPD[.D] E, [DT:]mml1
		ABS, X	CMPD[.D] E, [DT:]mml1, X
		ABS, Y	CMPD[.D] E, [DT:]mml1, Y
		ABL	CMPD[.D] E, [LG:]hhmml1
		ABL, X	CMPD[.D] E, [LG:]hhmml1, X
		SR	CMPD[.D] E, d8, S
(SR), Y	CMPD[.D] E, (d8, S), Y		
CMPM	.B/.W	DIR	CMPM[.B/.W] [DPn:]zz, #imm8/16
		ABS	CMPM[.B/.W] [DT:]mml1, #imm8/16
CMPMB	.B	DIR	CMPMB[.B] [DPn:]zz, #imm8
		ABS	CMPMB[.B] [DT:]mml1, #imm8
CMPMD	.D	DIR	CMPMD[.D] [DPn:]zz, #imm32
		ABS	CMPMD[.D] [DT:]mml1, #imm32
CPX	.B/.W	IMM	CPX[.B/.W] #imm8/16
		DIR	CPX[.B/.W] [DPn:]zz
		ABS	CPX[.B/.W] [DT:]mml1
CPY	.B/.W	IMM	CPY[.B/.W] #imm8/16
		DIR	CPY[.B/.W] [DPn:]zz
		ABS	CPY[.B/.W] [DT:]mml1
DEBNE	.B/.W	DIR	DEBNE[.B/.W] [DPn:]zz, #imm5, rr
		ABS	DEBNE[.B/.W] [DT:]mml1, #imm5, rr

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
DEC	.B/.W	Acc	DEC[.B/.W] Acc
		DIR	DEC[.B/.W] [DPn:]zz
		DIR,X	DEC[.B/.W] [DPn:]zz,X
		ABS	DEC[.B/.W] [DT:]mml1
		ABS,X	DEC[.B/.W] [DT:]mml1,X
DEX	.B/.W	IMP	DEX[.B/.W]
DEY	.B/.W	IMP	DEY[.B/.W]
DIV	.B/.W	IMM	DIV[.B/.W] #imm8/16
		DIR	DIV[.B/.W] [DPn:]zz
		DIR,X	DIV[.B/.W] [DPn:]zz,X
		(DIR)	DIV[.B/.W] ([DPn:]zz)
		(DIR,X)	DIV[.B/.W] ([DPn:]zz,X)
		(DIR),Y	DIV[.B/.W] ([DPn:]zz),Y
		L(DIR)	DIV[.B/.W] L([DPn:]zz)
		L(DIR),Y	DIV[.B/.W] L([DPn:]zz),Y
		ABS	DIV[.B/.W] [DT:]mml1
		ABS,X	DIV[.B/.W] [DT:]mml1,X
		ABS,Y	DIV[.B/.W] [DT:]mml1,Y
		ABL	DIV[.B/.W] [LG:]hhmml1
		ABL,X	DIV[.B/.W] [LG:]hhmml1,X
		SR	DIV[.B/.W] d8,S
		(SR),Y	DIV[.B/.W] (d8,S),Y

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
DIVS	.B/.W	IMM	DIVS[.B/.W] #imm8/16
		DIR	DIVS[.B/.W] [DPn:]zz
		DIR,X	DIVS[.B/.W] [DPn:]zz,X
		(DIR)	DIVS[.B/.W] ([DPn:]zz)
		(DIR,X)	DIVS[.B/.W] ([DPn:]zz,X)
		(DIR),Y	DIVS[.B/.W] ([DPn:]zz),Y
		L(DIR)	DIVS[.B/.W] L([DPn:]zz)
		L(DIR),Y	DIVS[.B/.W] L([DPn:]zz),Y
		ABS	DIVS[.B/.W] [DT:]mml1
		ABS,X	DIVS[.B/.W] [DT:]mml1,X
		ABS,Y	DIVS[.B/.W] [DT:]mml1,Y
		ABL	DIVS[.B/.W] [LG:]hhmml1
		ABL,X	DIVS[.B/.W] [LG:]hhmml1,X
		SR	DIVS[.B/.W] d8,S
		(SR),Y	DIVS[.B/.W] (d8,S),Y
		DXBNE	.B/.W
DYBNE	.B/.W	IMM	DYBNE[.B/.W] #imm5,rr
EOR	.B/.W	IMM	EOR[.B/.W] Acc,#imm8/16
		DIR	EOR[.B/.W] Acc,[DPn:]zz
		DIR,X	EOR[.B/.W] Acc,[DPn:]zz,X
		(DIR)	EOR[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	EOR[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	EOR[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	EOR[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	EOR[.B/.W] Acc,L([DPn:]zz),Y
		ABS	EOR[.B/.W] Acc,[DT:]mml1
		ABS,X	EOR[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	EOR[.B/.W] Acc,[DT:]mml1,Y
		ABL	EOR[.B/.W] Acc,[LG:]hhmml1
		ABL,X	EOR[.B/.W] Acc,[LG:]hhmml1,X
		SR	EOR[.B/.W] Acc,d8,S
		(SR),Y	EOR[.B/.W] Acc,(d8,S),Y

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
EORB	.B	IMM	EORB[.B] Acc, #imm8
EORM	.B/.W	DIR	EORM[.B/.W] [DPn:]zz, #imm8/16
		ABS	EORM[.B/.W] [DT:]mml1, #imm8/16
EORMB	.B	DIR	EORMB[.B] [DPn:]zz, #imm8
		ABS	EORMB[.B] [DT:]mml1, #imm8
EORMD	.D	DIR	EORMD[.D] [DPn:]zz, #imm32
		ABS	EORMD[.D] [DT:]mml1, #imm32
EXTS	.W	Acc	EXTS[.W] Acc
EXTSD	.D	E	EXTSD[.D] E
EXTZ	.W	Acc	EXTZ[.W] Acc
EXTZD	.D	E	EXTZD[.D] E
INC	.B/.W	Acc	INC[.B/.W] Acc
		DIR	INC[.B/.W] [DPn:]zz
		DIR, X	INC[.B/.W] [DPn:]zz, X
		ABS	INC[.B/.W] [DT:]mml1
		ABS, X	INC[.B/.W] [DT:]mml1, X
INX	.B/.W	IMP	INX[.B/.W]
INY	.B/.W	IMP	INY[.B/.W]
JMP		ABS	JMP mml1
		(ABS)	JMP (mml1)
		(ABS, X)	JMP (mml1, X)
JMPL		ABL	JMPL hhmm11
		L(ABS)	JMPL (mml1)
JSR		ABS	JSR mml1
		(ABS, X)	JSR (mml1, X)
JSRL		ABL	JSRL hhmm11

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式		
LDA	.B/.W	IMM	LDA[.B/.W] Acc,#imm8/16		
		DIR	LDA[.B/.W] Acc,[DPn:]zz		
		DIR,X	LDA[.B/.W] Acc,[DPn:]zz,X		
		(DIR)	LDA[.B/.W] Acc,([DPn:]zz)		
		(DIR,X)	LDA[.B/.W] Acc,([DPn:]zz,X)		
		(DIR),Y	LDA[.B/.W] Acc,([DPn:]zz),Y		
		L(DIR)	LDA[.B/.W] Acc,L([DPn:]zz)		
		L(DIR),Y	LDA[.B/.W] Acc,L([DPn:]zz),Y		
		ABS	LDA[.B/.W] Acc,[DT:]mml1		
		ABS,X	LDA[.B/.W] Acc,[DT:]mml1,X		
		ABS,Y	LDA[.B/.W] Acc,[DT:]mml1,Y		
		ABL	LDA[.B/.W] Acc,[LG:]hhmml1		
		ABL,X	LDA[.B/.W] Acc,[LG:]hhmml1,X		
		SR	LDA[.B/.W] Acc,d8,S		
		(SR),Y	LDA[.B/.W] Acc,(d8,S),Y		
		LDAB	.W	IMM	LDAB[.W] Acc,#imm8
				DIR	LDAB[.W] Acc,[DPn:]zz
DIR,X	LDAB[.W] Acc,[DPn:]zz,X				
(DIR)	LDAB[.W] Acc,([DPn:]zz)				
(DIR,X)	LDAB[.W] Acc,([DPn:]zz,X)				
(DIR),Y	LDAB[.W] Acc,([DPn:]zz),Y				
L(DIR)	LDAB[.W] Acc,L([DPn:]zz)				
L(DIR),Y	LDAB[.W] Acc,L([DPn:]zz),Y				
ABS	LDAB[.W] Acc,[DT:]mml1				
ABS,X	LDAB[.W] Acc,[DT:]mml1,X				
ABS,Y	LDAB[.W] Acc,[DT:]mml1,Y				
ABL	LDAB[.W] Acc,[LG:]hhmml1				
ABL,X	LDAB[.W] Acc,[LG:]hhmml1,X				
SR	LDAB[.W] Acc,d8,S				
(SR),Y	LDAB[.W] Acc,(d8,S),Y				

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
LDAD	.D	IMM	LDAD[.D] E, #imm32
		DIR	LDAD[.D] E, [DPn:]zz
		DIR, X	LDAD[.D] E, [DPn:]zz, X
		(DIR)	LDAD[.D] E, ([DPn:]zz)
		(DIR, X)	LDAD[.D] E, ([DPn:]zz, X)
		(DIR), Y	LDAD[.D] E, ([DPn:]zz), Y
		L(DIR)	LDAD[.D] E, L([DPn:]zz)
		L(DIR), Y	LDAD[.D] E, L([DPn:]zz), Y
		ABS	LDAD[.D] E, [DT:]mml1
		ABS, X	LDAD[.D] E, [DT:]mml1, X
		ABS, Y	LDAD[.D] E, [DT:]mml1, Y
		ABL	LDAD[.D] E, [LG:]hhmml1
		ABL, X	LDAD[.D] E, [LG:]hhmml1, X
		SR	LDAD[.D] E, d8, S
(SR), Y	LDAD[.D] E, (d8, S), Y		
LDD	.W	IMM	LDD[.W] bit, #imm16[, #imm16...]
LDT	.B	IMM	LDT[.B] #imm8
LDX	.B/.W	IMM	LDX[.B/.W] #imm8/16
		DIR	LDX[.B/.W] [DPn:]zz
		DIR, Y	LDX[.B/.W] [DPn:]zz, Y
		ABS	LDX[.B/.W] [DT:]mml1
		ABS, Y	LDX[.B/.W] [DT:]mml1, Y
LDXB	.W	IMM	LDXB[.W] #imm8
LDY	.B/.W	IMM	LDY[.B/.W] #imm8/16
		DIR	LDY[.B/.W] [DPn:]zz
		DIR, X	LDY[.B/.W] [DPn:]zz, X
		ABS	LDY[.B/.W] [DT:]mml1
		ABS, X	LDY[.B/.W] [DT:]mml1, X
LDYB	.W	IMM	LDYB[.W] #imm8

## 7900命令一覧

命令名	データ長	アドレスモード	記述形式		
LSR	.B/.W	Acc	LSR[.B/.W] Acc		
		DIR	LSR[.B/.W] [DPn:]zz		
		DIR,X	LSR[.B/.W] [DPn:]zz,X		
		ABS	LSR[.B/.W] [DT:]mml1		
		ABS,X	LSR[.B/.W] [DT:]mml1,X		
		A	LSR[.B/.W] A,#imm4		
LSRD	.D	E	LSRD[.D] E,#imm5		
MOVM	.B/.W	DIR < IMM	MOVM[.B/.W] [DPn:]zz,#imm8/16		
		DIR < DIR	MOVM[.B/.W] [DPn:]zz,[DPn:]zz		
		DIR < ABS	MOVM[.B/.W] [DPn:]zz,[DT:]mml1		
		DIR < ABS,X	MOVM[.B/.W] [DPn:]zz,[DT:]mml1,X		
		DIR,X < IMM	MOVM[.B/.W] [DPn:]zz,X,#imm8/16		
		ABS < IMM	MOVM[.B/.W] [DT:]mml1,#imm8/16		
		ABS < DIR	MOVM[.B/.W] [DT:]mml1,[DPn:]zz		
		ABS < DIR,X	MOVM[.B/.W] [DT:]mml1,[DPn:]zz,X		
		ABS < ABS	MOVM[.B/.W] [DT:]mml1,[DT:]mml1		
		ABS,X < IMM	MOVM[.B/.W] [DT:]mml1,X,#imm8/16		
		MOVMB	.B	DIR < IMM	MOVMB[.B] [DPn:]zz,#imm8
				DIR < DIR	MOVMB[.B] [DPn:]zz,[DPn:]zz
DIR < ABS	MOVMB[.B] [DPn:]zz,[DT:]mml1				
DIR < ABS,X	MOVMB[.B] [DPn:]zz,[DT:]mml1,X				
DIR,X < IMM	MOVMB[.B] [DPn:]zz,X,#imm8				
ABS < IMM	MOVMB[.B] [DT:]mml1,#imm8				
ABS < DIR	MOVMB[.B] [DT:]mml1,[DPn:]zz				
ABS < DIR,X	MOVMB[.B] [DT:]mml1,[DPn:]zz,X				
ABS < ABS	MOVMB[.B] [DT:]mml1,[DT:]mml1				
ABS,X < IMM	MOVMB[.B] [DT:]mml1,X,#imm8				



## 命令名 データ長 アドレスモード

## 記述形式

---

 MOVR .B/.W IMM,DIR « IMM

MOVR[.B/.W] #imm4,[DPn:]zz,#imm8/16[...[DPn:]zz,#imm8/16]

---

 IMM,DIR « DIR

MOVR[.B/.W] #imm4,[DPn:]zz,[DPn:]zz[...[DPn:]zz,[DPn:]zz]

---

 IMM,DIR « ABS

MOVR[.B/.W] #imm4,[DPn:]zz,[DT:]mml1[...[DPn:]zz,[DT:]mml1]

---

 IMM,DIR « ABS,X

MOVR[.B/.W] #imm4,[DPn:]zz,[DT:]mml1,X[...[DPn:]zz,[DT:]mml1,X]

---

 IMM,ABS « IMM

MOVR[.B/.W] #imm4,[DT:]mml1,#imm8/16[...[DT:]mml1,#imm8/16]

---

 IMM,ABS « DIR

MOVR[.B/.W] #imm4,[DT:]mml1,[DPn:]zz[...[DT:]mml1,[DPn:]zz]

---

 IMM,ABS « DIR,X

MOVR[.B/.W] #imm4,[DT:]mml1,[DPn:]zz,X[...[DT:]mml1,[DPn:]zz,X]

---

 IMM,ABS « ABS

MOVR[.B/.W] #imm4,[DT:]mml1,[DT:]mml1[...[DT:]mml1,[DT:]mml1]

---

 MOV RB .B IMM,DIR « IMM

MOV RB[.B] #imm4,[DPn:]zz,#imm8[...[DPn:]zz,#imm8]

---

 IMM,DIR « DIR

MOV RB[.B] #imm4,[DPn:]zz,[DPn:]zz[...[DPn:]zz,[DPn:]zz]

---

 IMM,DIR « ABS

MOV RB[.B] #imm4,[DPn:]zz,[DT:]mml1[...[DPn:]zz,[DT:]mml1]

---

 IMM,DIR « ABS,X

MOV RB[.B] #imm4,[DPn:]zz,[DT:]mml1,X[...[DPn:]zz,[DT:]mml1,X]

---

 IMM,ABS « IMM

MOV RB[.B] #imm4,[DT:]mml1,#imm8[...[DT:]mml1,#imm8]

---

 IMM,ABS « DIR

MOV RB[.B] #imm4,[DT:]mml1,[DPn:]zz[...[DT:]mml1,[DPn:]zz]

---

 IMM,ABS « DIR,X

MOV RB[.B] #imm4,[DT:]mml1,[DPn:]zz,X[...[DT:]mml1,[DPn:]zz,X]

---

 IMM,ABS « ABS

MOV RB[.B] #imm4,[DT:]mml1,[DT:]mml1[...[DT:]mml1,[DT:]mml1]

## 7900命令一覧

命令名	データ長	アドレス指定モード	記述形式
MPY	.B/.W	IMM	MPY[.B/.W] #imm8/16
		DIR	MPY[.B/.W] [DPn:]zz
		DIR,X	MPY[.B/.W] [DPn:]zz,X
		(DIR)	MPY[.B/.W] ([DPn:]zz)
		(DIR,X)	MPY[.B/.W] ([DPn:]zz,X)
		(DIR),Y	MPY[.B/.W] ([DPn:]zz),Y
		L(DIR)	MPY[.B/.W] L([DPn:]zz)
		L(DIR),Y	MPY[.B/.W] L([DPn:]zz),Y
		ABS	MPY[.B/.W] [DT:]mml1
		ABS,X	MPY[.B/.W] [DT:]mml1,X
		ABS,Y	MPY[.B/.W] [DT:]mml1,Y
		ABL	MPY[.B/.W] [LG:]hhmml1
		ABL,X	MPY[.B/.W] [LG:]hhmml1,X
		SR	MPY[.B/.W] d8,S
		(SR),Y	MPY[.B/.W] (d8,S),Y
MPYS	.B/.W	IMM	MPYS[.B/.W] #imm8/16
		DIR	MPYS[.B/.W] [DPn:]zz
		DIR,X	MPYS[.B/.W] [DPn:]zz,X
		(DIR)	MPYS[.B/.W] ([DPn:]zz)
		(DIR,X)	MPYS[.B/.W] ([DPn:]zz,X)
		(DIR),Y	MPYS[.B/.W] ([DPn:]zz),Y
		L(DIR)	MPYS[.B/.W] L([DPn:]zz)
		L(DIR),Y	MPYS[.B/.W] L([DPn:]zz),Y
		ABS	MPYS[.B/.W] [DT:]mml1
		ABS,X	MPYS[.B/.W] [DT:]mml1,X
		ABS,Y	MPYS[.B/.W] [DT:]mml1,Y
		ABL	MPYS[.B/.W] [LG:]hhmml1
		ABL,X	MPYS[.B/.W] [LG:]hhmml1,X
		SR	MPYS[.B/.W] d8,S
		(SR),Y	MPYS[.B/.W] (d8,S),Y

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
MVN		BLK	MVM d8, d8
MVP		BLK	MVP d8, d8
NEG	.B/.W	Acc	NEG[.B/.W] Acc
NEGD	.D	E	NEGD[.D] E
NOP		IMP	NOP
ORA	.B/.W	IMM	ORA[.B/.W] Acc, #imm8/16
		DIR	ORA[.B/.W] Acc, [DPn:]zz
		DIR, X	ORA[.B/.W] Acc, [DPn:]zz, X
		(DIR)	ORA[.B/.W] Acc, ([DPn:]zz)
		(DIR, X)	ORA[.B/.W] Acc, ([DPn:]zz, X)
		(DIR), Y	ORA[.B/.W] Acc, ([DPn:]zz), Y
		L(DIR)	ORA[.B/.W] Acc, L([DPn:]zz)
		L(DIR), Y	ORA[.B/.W] Acc, L([DPn:]zz), Y
		ABS	ORA[.B/.W] Acc, [DT:]mml1
		ABS, X	ORA[.B/.W] Acc, [DT:]mml1, X
		ABS, Y	ORA[.B/.W] Acc, [DT:]mml1, Y
		ABL	ORA[.B/.W] Acc, [LG:]hhmml1
		ABL, X	ORA[.B/.W] Acc, [LG:]hhmml1, X
		SR	ORA[.B/.W] Acc, d8, S
(SR), Y	ORA[.B/.W] Acc, (d8, S), Y		
ORAB	.B	IMM	ORAB[.B] Acc, #imm8
ORAM	.B/.W	DIR	ORAM[.B/.W] [DPn:]zz, #imm8/16
		ABS	ORAM[.B/.W] [DT:]mml1, #imm8/16
ORAMB	.B	DIR	ORAMB[.B] [DPn:]zz, #imm8
		ABS	ORAMB[.B] [DT:]mml1, #imm8
ORAMD	.D	DIR	ORAMD[.D] [DPn:]zz, #imm32
		ABS	ORAMD[.D] [DT:]mml1, #imm32
PEA	.W	STK	PEA[.W] #imm16
PEI	.W	STK	PEI[.W] [DPn:]zz
PER	.W	STK	PER[.W] #imm16

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
PHA	.B/.W	STK	PHA[.B/.W]
PHB	.B/.W	STK	PHB[.B/.W]
PHD	.W	STK	PHD[.W] [bit]
PHG	.B	STK	PHG[.B]
PHLD	.W	STK	PHLD[.W] bit, #imm16[, #imm16...]
PHP	.W	STK	PHP[.W]
PHT	.B	STK	PHT[.B]
PHX	.B/.W	STK	PHX[.B/.W]
PHY	.B/.W	STK	PHY[.B/.W]
PLA	.B/.W	STK	PLA[.B/.W]
PLB	.B/.W	STK	PLB[.B/.W]
PLD	.W	STK	PLD[.W] [bit]
PLP	.W	STK	PLP[.W]
PLT	.B	STK	PLT[.B]
PLX	.B/.W	STK	PLX[.B/.W]
PLY	.B/.W	STK	PLY[.B/.W]
PSH	.B/.W	STK	PSH bit
PUL	.B/.W	STK	PUL bit
RLA	.B/.W	IMM	PLA[.B/.W] #imm8/#imm16
RMPA	.B/.W	繰り返し積和	RMPA[.B/.W] #imm8
ROL	.B/.W	Acc	ROL[.B/.W] Acc
		DIR	ROL[.B/.W] [DPn:]zz
		DIR,X	ROL[.B/.W] [DPn:]zz,X
		ABS	ROL[.B/.W] [DT:]mml1
		ABS,X	ROL[.B/.W] [DT:]mml1,X
		A	ROL[.B/.W] A,#imm4
ROLD	.D	E	ROLD[.D] E,#imm5

PSH,PUL命令はオペランド欄にレジスタ名を記述できます。

例 1 ) PSH     A,B,X,Y,DP,DT,PG,PS

例 2 ) PUL     A,B,X,Y,DP,DT,PS

## 7900命令一覧

命令名	データ長	アドレスモード	記述形式
ROR	.B/.W	Acc	ROR[.B/.W] Acc
		DIR	ROR[.B/.W] [DPn:]zz
		DIR,X	ROR[.B/.W] [DPn:]zz,X
		ABS	ROR[.B/.W] [DT:]mml1
		ABS,X	ROR[.B/.W] [DT:]mml1,X
		A	ROR[.B/.W] A,#imm4
RORD	.D	E	RORD[.D] E,#imm5
RTI		IMP	RTI
RTL		IMP	RTL
RTLD	.W	STK	RTLD[.W] bit
RTS		IMP	RTS
RTSD	.W	STK	RTSD[.W] bit
SBC	.B/.W	IMM	SBC[.B/.W] Acc,#imm8/16
		DIR	SBC[.B/.W] Acc,[DPn:]zz
		DIR,X	SBC[.B/.W] Acc,[DPn:]zz,X
		(DIR)	SBC[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	SBC[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	SBC[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	SBC[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	SBC[.B/.W] Acc,L([DPn:]zz),Y
		ABS	SBC[.B/.W] Acc,[DT:]mml1
		ABS,X	SBC[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	SBC[.B/.W] Acc,[DT:]mml1,Y
		ABL	SBC[.B/.W] Acc,[LG:]hhmml1
		ABL,X	SBC[.B/.W] Acc,[LG:]hhmml1,X
		SR	SBC[.B/.W] Acc,d8,S
(SR),Y	SBC[.B/.W] Acc,(d8,S),Y		
SBCB	.B	IMM	SBCB[.B] E,#imm8

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
SBCD	.D	IMM	SBCD[.D] E,#imm32
		DIR	SBCD[.D] E,[DPn:]zz
		DIR,X	SBCD[.D] E,[DPn:]zz,X
		(DIR)	SBCD[.D] E,([DPn:]zz)
		(DIR,X)	SBCD[.D] E,([DPn:]zz,X)
		(DIR),Y	SBCD[.D] E,([DPn:]zz),Y
		L(DIR)	SBCD[.D] E,L([DPn:]zz)
		L(DIR),Y	SBCD[.D] E,L([DPn:]zz),Y
		ABS	SBCD[.D] E,[DT:]mml1
		ABS,X	SBCD[.D] E,[DT:]mml1,X
		ABS,Y	SBCD[.D] E,[DT:]mml1,Y
		ABL	SBCD[.D] E,[LG:]hhmml1
		ABL,X	SBCD[.D] E,[LG:]hhmml1,X
		SR	SBCD[.D] E,d8,S
		(SR),Y	SBCD[.D] E,(d8,S),Y
		SEC	
SEI		IMP	SEI
SEM		IMP	SEM
SEP		IMM	SEP #imm8
STA	.B/.W	DIR	STA[.B/.W] Acc,[DPn:]zz
		DIR,X	STA[.B/.W] Acc,[DPn:]zz,X
		(DIR)	STA[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	STA[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	STA[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	STA[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	STA[.B/.W] Acc,L([DPn:]zz),Y
		ABS	STA[.B/.W] Acc,[DT:]mml1
		ABS,X	STA[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	STA[.B/.W] Acc,[DT:]mml1,Y
		ABL	STA[.B/.W] Acc,[LG:]hhmml1
		ABL,X	STA[.B/.W] Acc,[LG:]hhmml1,X
		SR	STA[.B/.W] Acc,d8,S
		(SR),Y	STA[.B/.W] Acc,(d8,S),Y

SEP命令はオペランド欄にフラグ名を記述できます。

例) SEP C,Z,I,D,X,M,V,N

## 7900命令一覧

命令名	データ長	アドレス指定モード	記述形式
STAB	.B	DIR	STAB[.B] Acc, [DPn:]zz
		DIR,X	STAB[.B] Acc, [DPn:]zz,X
		(DIR)	STAB[.B] Acc, ([DPn:]zz)
		(DIR,X)	STAB[.B] Acc, ([DPn:]zz,X)
		(DIR),Y	STAB[.B] Acc, ([DPn:]zz),Y
		L(DIR)	STAB[.B] Acc,L([DPn:]zz)
		L(DIR),Y	STAB[.B] Acc,L([DPn:]zz),Y
		ABS	STAB[.B] Acc,[DT:]mml1
		ABS,X	STAB[.B] Acc,[DT:]mml1,X
		ABS,Y	STAB[.B] Acc,[DT:]mml1,Y
		ABL	STAB[.B] Acc,[LG:]hhmml1
		ABL,X	STAB[.B] Acc,[LG:]hhmml1,X
		SR	STAB[.B] Acc,d8,S
		(SR),Y	STAB[.B] Acc,(d8,S),Y
STAD	.D	DIR	STAD[.D] E,[DPn:]zz
		DIR,X	STAD[.D] E,[DPn:]zz,X
		(DIR)	STAD[.D] E,([DPn:]zz)
		(DIR,X)	STAD[.D] E,([DPn:]zz,X)
		(DIR),Y	STAD[.D] E,([DPn:]zz),Y
		L(DIR)	STAD[.D] E,L([DPn:]zz)
		L(DIR),Y	STAD[.D] E,L([DPn:]zz),Y
		ABS	STAD[.D] E,[DT:]mml1
		ABS,X	STAD[.D] E,[DT:]mml1,X
		ABS,Y	STAD[.D] E,[DT:]mml1,Y
		ABL	STAD[.D] E,[LG:]hhmml1
		ABL,X	STAD[.D] E,[LG:]hhmml1,X
		SR	STAD[.D] E,d8,S
		(SR),Y	STAD[.D] E,(d8,S),Y
STP		IMP	STP

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
STX	.B/.W	DIR	STX[.B/.W] [DPn:]zz
		DIR,Y	STX[.B/.W] [DPn:]zz,Y
		ABS	STX[.B/.W] [DT:]mml1
STY	.B/.W	DIR	STY[.B/.W] [DPn:]zz
		DIR,X	STY[.B/.W] [DPn:]zz,X
		ABS	STY[.B/.W] [DT:]mml1
SUB	.B/.W	IMM	SUB[.B/.W] Acc,#imm8/16
		DIR	SUB[.B/.W] Acc,[DPn:]zz
		DIR,X	SUB[.B/.W] Acc,[DPn:]zz,X
		(DIR)	SUB[.B/.W] Acc,([DPn:]zz)
		(DIR,X)	SUB[.B/.W] Acc,([DPn:]zz,X)
		(DIR),Y	SUB[.B/.W] Acc,([DPn:]zz),Y
		L(DIR)	SUB[.B/.W] Acc,L([DPn:]zz)
		L(DIR),Y	SUB[.B/.W] Acc,L([DPn:]zz),Y
		ABS	SUB[.B/.W] Acc,[DT:]mml1
		ABS,X	SUB[.B/.W] Acc,[DT:]mml1,X
		ABS,Y	SUB[.B/.W] Acc,[DT:]mml1,Y
		ABL	SUB[.B/.W] Acc,[LG:]hhmml1
		ABL,X	SUB[.B/.W] Acc,[LG:]hhmml1,X
		SR	SUB[.B/.W] Acc,d8,S
		(SR),Y	SUB[.B/.W] Acc,(d8,S),Y
SUBB	.B	IMM	SUBB[.B] Acc,#imm8



## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
SUBD	.D	IMM	SUBD[.D] E, #imm32
		DIR	SUBD[.D] E, [DPn:]zz
		DIR, X	SUBD[.D] E, [DPn:]zz, X
		(DIR)	SUBD[.D] E, ([DPn:]zz)
		(DIR, X)	SUBD[.D] E, ([DPn:]zz, X)
		(DIR), Y	SUBD[.D] E, ([DPn:]zz), Y
		L(DIR)	SUBD[.D] E, L([DPn:]zz)
		L(DIR), Y	SUBD[.D] E, L([DPn:]zz), Y
		ABS	SUBD[.D] E, [DT:]mml1
		ABS, X	SUBD[.D] E, [DT:]mml1, X
		ABS, Y	SUBD[.D] E, [DT:]mml1, Y
		ABL	SUBD[.D] E, [LG:]hhmml1
		ABL, X	SUBD[.D] E, [LG:]hhmml1, X
		SR	SUBD[.D] E, d8, S
		(SR), Y	SUBD[.D] E, (d8, S), Y
		SUBM	.B/.W
ABS	SUBM[.B/.W] [DT:]mml1, #imm8/16		
SUBMB	.B	DIR	SUBMB[.B] [DPn:]zz, #imm8
		ABS	SUBMB[.B] [DT:]mml1, #imm8
SUBMD	.D	DIR	SUBMD[.D] [DPn:]zz, #imm32
		ABS	SUBMD[.D] [DT:]mml1, #imm32
SUBS	.W	IMM	SUBS[.W] #imm8
SUBX	.B/.W	IMM	SUBX[.B/.W] #imm5
SUBY	.B/.W	IMM	SUBY[.B/.W] #imm5

## 7900命令一覧

命令名	データ長	アドレッシングモード	記述形式
TAD	.W	IMP	TAD[.W] n
TAS	.W	IMP	TAS[.W]
TAX	.B/.W	IMP	TAX[.B/.W]
TAY	.B/.W	IMP	TAY[.B/.W]
TBD	.W	IMP	TBD[.W] n
TBS	.W	IMP	TBS[.W]
TBX	.B/.W	IMP	TBX[.B/.W]
TBY	.B/.W	IMP	TBY[.B/.W]
TDA	.B/.W	IMP	TDA[.B/.W] n
TDB	.B/.W	IMP	TDB[.B/.W] n
TDS	.W	IMP	TDS[.W]
TSA	.B/.W	IMP	TSA[.B/.W]
TSB	.B/.W	IMP	TSB[.B/.W]
TSD	.W	IMP	TSD[.W]
TSX	.B/.W	IMP	TSX[.B/.W]
TXA	.B/.W	IMP	TXA[.B/.W]
TXB	.B/.W	IMP	TXB[.B/.W]
TXS	.B/.W	IMP	TXS[.W]
TXY	.B/.W	IMP	TXY[.B/.W]
TYA	.B/.W	IMP	TYA[.B/.W]
TYB	.B/.W	IMP	TYB[.B/.W]
TYX	.B/.W	IMP	TYX[.B/.W]
WIT		IMP	WIT
XAB	.B/.W	IMP	XAB[.B/.W]

# AS79 V.4.10 ユーザーズマニュアル

---

Rev. 1.00  
03.08.01  
RJJ10J0198-0100Z

COPYRIGHT ©2003 RENESAS TECHNOLOGY CORPORATION  
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

AS79 V.4.10  
ユーザーズマニュアル



ルネサスエレクトロニクス株式会社  
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ10J0198-0100Z