

CMOS 8-BIT SINGLE CHIP MICROCOMPUTER

# **S5U1C88000C Manual II**

**(S1C88 Family 統合ツールパッケージ)**

ワークベンチ/Development Tools/旧アセンブラパッケージ

本資料のご使用につきましては、次の点にご留意願います。

---

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

本書に記載のCコンパイラ、アセンブラおよびその他のツールはTASKING社が開発した製品です。

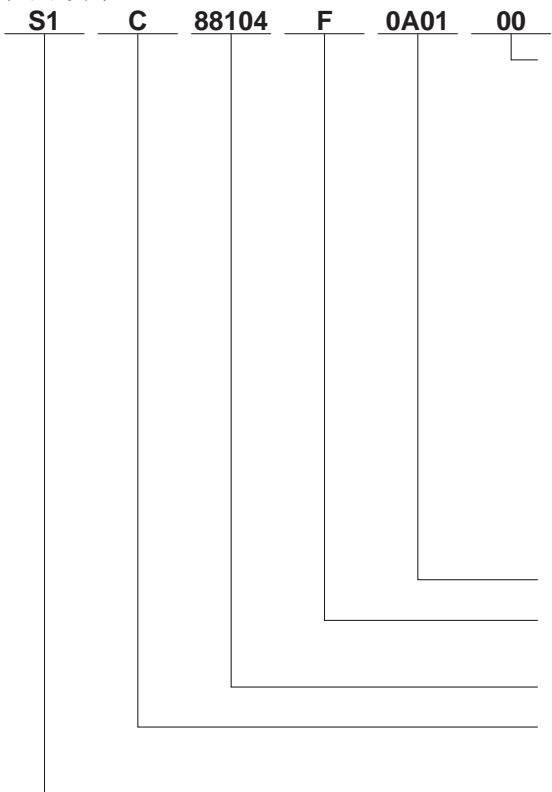
Windows 2000およびWindows XPは米国マイクロソフト社の登録商標です。

PC/ATおよびIBMは米国International Business Machines社の登録商標です。

その他のブランド名または製品名は、それらの所有者の商標もしくは登録商標です。

製品型番体系

デバイス



梱包仕様

- 00 : テープ&リール以外
- 0A : TCP BL 2方向
- 0B : テープ&リール BACK
- 0C : TCP BR 2方向
- 0D : TCP BT 2方向
- 0E : TCP BD 2方向
- 0F : テープ&リール FRONT
- 0G : TCP BT 4方向
- 0H : TCP BD 4方向
- 0J : TCP SL 2方向
- 0K : TCP SR 2方向
- 0L : テープ&リール LEFT
- 0M : TCP ST 2方向
- 0N : TCP SD 2方向
- 0P : TCP ST 4方向
- 0Q : TCP SD 4方向
- 0R : テープ&リール RIGHT
- 99 : 梱包仕様未定

仕様

形状

[D: ペアチップ、F: QFP、B: BGA]

機種番号

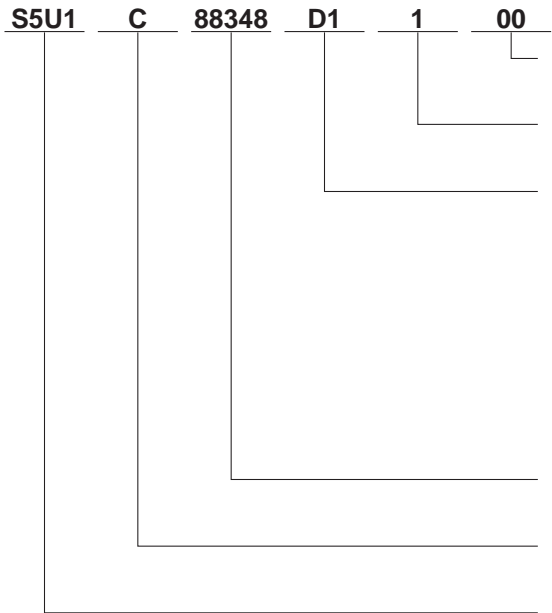
機種名称

[C: マイコン、デジタル製品]

製品分類

[S1: 半導体]

開発ツール



梱包仕様

[00: 標準梱包]

バージョン

[1: Version 1]

ツール種類

- Hx : ICE
- Ex : EVAボード
- Px : ペリフェラルボード
- Wx : FLASHマイコン用ROMライター
- Xx : ROMライター周辺ボード
- Cx : Cコンパイラパッケージ
- Ax : アセンブラパッケージ
- Dx : 機種別ユーティリティツール
- Qx : ソフトシミュレータ

対応機種番号

[88348: S1C88348用]

ツール分類

[C: マイコン用]

製品分類

[S5U1: 半導体用開発ツール]



## マニュアルの構成

S1C88 Family統合ツールパッケージには、S1C88 Familyマイクロコンピュータのソフトウェア開発に必要なツールが含まれています。S5U1C88000C Manual( S1C88 Family統合ツールパッケージ )は、これらのツールの機能と使用方法を説明します。マニュアルは次のとおり2冊で構成されています。

### I. Cコンパイラ/アセンブラ/リンカ

Cコンパイラを中心とした、ツールチェーン( 次ページの図の[Main Tool Chain] )を解説しています。

### II. ワークベンチ/Development Tools/旧アセンブラパッケージ( 本書 )

統合開発環境を提供するワークベンチ、アドバンスドロケータ、マスクデータ作成用ツール( 次ページの図の[Development Tool Chain] )、デバッガ、構造化アセンブラ( 次ページの図の[Sub Tool Chain] )を解説しています。

このマニュアルは、読者にCおよびアセンブリ言語の知識があることを前提として書かれています。

S1C88 Familyマイクロコンピュータの開発時は、必要に応じて以下のマニュアルも参照してください。

#### S1C88xxxテクニカルマニュアル

デバイスの仕様、Flash EEPROMプログラミング方法等を解説しています。

#### S5U1C88000Q Manual

シミュレータパッケージに含まれるツールの操作方法を解説しています。

#### S5U1C88000H5 Manual

ICE( S5U1C88000H5 )の取り扱い方法を解説しています。

#### S5U1C88xxxP Manual

ICEに装着する周辺回路ボードの取り扱い方法を解説しています。



## - 目 次 -

1	概要 .....	1
1.1	特長 .....	1
1.2	S1C88 Family統合開発環境 .....	2
2	インストール .....	5
2.1	パッケージの内容 .....	5
2.2	動作環境 .....	5
2.3	インストール方法 .....	6
2.4	インストール後のディレクトリとファイル構成 .....	7
2.5	環境設定 .....	8
3	ワークベンチ .....	9
3.1	特長 .....	9
3.2	起動および終了方法 .....	9
3.3	ワークベンチウィンドウ .....	10
3.4	ツールバーとボタン .....	12
3.5	メニュー .....	13
3.5.1	[File]メニュー .....	13
3.5.2	[View]メニュー .....	13
3.5.3	[Source]メニュー .....	14
3.5.4	[Build]メニュー .....	14
3.5.5	[Debug]メニュー .....	14
3.5.6	[Tools]メニュー .....	14
3.5.7	[Help]メニュー .....	15
3.6	プロジェクトとワークスペース .....	16
3.6.1	プロジェクトの新規作成 .....	16
3.6.2	プロジェクトへのソースファイル/ヘッダファイルの登録 .....	17
3.6.3	プロジェクトからのファイルの削除 .....	17
3.6.4	プロジェクトビュー .....	17
3.6.5	プロジェクトのオープン/クローズ .....	18
3.6.6	プロジェクトの保存 .....	18
3.7	ソースファイルの作成/編集 .....	19
3.7.1	エディタの指定 .....	19
3.7.2	ソースファイル/ヘッダファイルの新規作成 .....	20
3.7.3	ファイルの編集 .....	20
3.7.4	タグジャンプ機能 .....	21
3.8	ビルド .....	22
3.8.1	ビルドの準備 .....	22
3.8.2	実行形式オブジェクトのビルド .....	22
3.8.3	コンパイラ/アセンブラのみの実行 .....	23
3.9	ツールオプションの設定 .....	24
3.9.1	コンパイラオプション .....	25
3.9.2	アセンブラオプション .....	27
3.9.3	リンカオプション .....	29
3.9.4	ロケータオプション .....	30
3.9.5	セクションエディタ .....	32

3.10 デバッグ .....	38
3.10.1 シミュレータ .....	38
3.10.2 インサーキットエミュレータ( S5U1C88000H5 )とデバッグ .....	40
3.11 その他のツールの実行 .....	41
3.12 ファイル一覧 .....	42
3.13 エラーメッセージ .....	43
4 メインツールチェーンの概要 .....	44
5 アドバンスドロケータ<alc88> .....	45
5.1 alc88の機能 .....	45
5.2 入出力ファイル .....	46
5.3 操作方法 .....	47
5.4 エラーメッセージ .....	47
5.5 注意事項 .....	47
6 デベロップメントツールの概要 .....	48
7 内蔵ROM未使用領域FF詰めユーティリティ<fil88xxx> .....	50
7.1 fil88xxxの概要 .....	50
7.2 入出力ファイル .....	50
7.3 操作方法 .....	51
7.4 エラーメッセージ .....	52
7.5 入出力ファイル例 .....	53
8 ファンクションオプションジェネレータ<winfog> .....	54
8.1 winfogの概要 .....	54
8.2 入出力ファイル .....	54
8.3 操作方法 .....	55
8.3.1 起動方法 .....	55
8.3.2 ウィンドウ .....	56
8.3.3 メニューとツールバーボタン .....	57
8.3.4 操作手順 .....	58
8.4 エラーメッセージ .....	61
8.5 出力ファイル例 .....	62
9 セグメントオプションジェネレータ<winsog> .....	63
9.1 winsogの概要 .....	63
9.2 入出力ファイル .....	63
9.3 操作方法 .....	64
9.3.1 起動方法 .....	64
9.3.2 ウィンドウ .....	66
9.3.3 メニューとツールバーボタン .....	67
9.3.4 オプション選択用ボタン .....	68
9.3.5 操作手順 .....	68
9.4 エラーメッセージ .....	74
9.5 出力ファイル例 .....	75
10 マスクデータチェッカ<winmdc> .....	76
10.1 winmdcの概要 .....	76
10.2 入出力ファイル .....	76



10.3 操作方法 .....	77
10.3.1 起動方法 .....	77
10.3.2 メニューとツールバーボタン .....	78
10.3.3 操作手順 .....	79
10.4 エラーメッセージ .....	82
10.5 出力ファイル例 .....	83
11 自己診断プログラム<t88xxx> .....	84
11.1 t88xxxの概要 .....	84
11.2 ファイル構成 .....	84
11.3 使用方法 .....	84
12 88xxx.parファイル .....	85
12.1 88xxx.parファイル内容 .....	85
12.2 88xxx.parファイル内容説明 .....	86
12.3 エミュレーションメモリについて .....	86
13 S1C88 Familyデバッガ .....	87
13.1 概要 .....	87
13.2 入出力ファイル .....	87
13.3 起動と終了 .....	88
13.3.1 起動方法 .....	88
13.3.2 終了方法 .....	89
13.4 ウィンドウ .....	90
13.4.1 ウィンドウの基本構成 .....	90
13.4.2 [Command]ウィンドウ .....	91
13.4.3 [Source]ウィンドウ .....	93
13.4.4 [Dump]ウィンドウ .....	98
13.4.5 [Register]ウィンドウ .....	99
13.4.6 [Symbol]ウィンドウ .....	99
13.4.7 [Watch]ウィンドウ .....	99
13.4.8 [Trace]ウィンドウ .....	100
13.4.9 [Coverage]ウィンドウ .....	100
13.5 メニュー .....	101
13.6 ツールバー .....	105
13.7 コマンド実行方法 .....	106
13.7.1 コマンドのキーボード入力 .....	106
13.7.2 メニュー、ツールバーからの実行 .....	108
13.7.3 コマンドファイルによる実行 .....	109
13.7.4 ログファイル .....	110
13.8 デバッグ機能 .....	111
13.8.1 ファイルの読み込み .....	111
13.8.2 ソース表示およびシンボリックデバッグ機能 .....	112
13.8.3 メモリデータ、レジスタの表示と変更 .....	114
13.8.4 プログラムの実行 .....	116
13.8.5 ブレーク機能 .....	120
13.8.6 トレース機能 .....	127
13.8.7 カバレッジ .....	131
13.8.8 標準ペリフェラルボードのFPGAデータ書き込み .....	133
13.8.9 システムオプション .....	134

13.9 コマンドリファレンス .....	135
13.9.1 コマンド一覧 .....	135
13.9.2 各コマンド説明の見方 .....	136
13.9.3 メモリ操作コマンド .....	137
dd (data dump) .....	137
de (data enter) .....	140
df (data fill) .....	142
dm (data move) .....	143
ds (data search) .....	144
13.9.4 レジスタ操作コマンド .....	145
rd (register display) .....	145
rs (register set) .....	146
13.9.5 プログラム実行コマンド .....	148
g (go) .....	148
gr (go after reset CPU) .....	150
s (step) .....	151
n (next) .....	153
se (step exit) .....	154
13.9.6 CPUリセットコマンド .....	155
rst (reset CPU) .....	155
13.9.7 ブレーク設定コマンド .....	156
bp (software break point set) .....	156
bpa (software area break point set) .....	158
bpr / bc / bpc (software break point clear) .....	160
bas (sequential break setting) .....	161
ba (hardware break point set) .....	162
bar (hardware break point clear) .....	164
bd (hardware data break point set) .....	165
bdr (hardware data break point clear) .....	167
bl (break point list) .....	168
bac (break all clear) .....	169
13.9.8 プログラム表示コマンド .....	170
u (unassemble) .....	170
sc (source code) .....	172
m (mix) .....	174
13.9.9 シンボル情報表示コマンド .....	176
sy (symbol list) .....	176
w (symbol watch) .....	177
13.9.10 ファイル読み込みコマンド .....	178
lf (load file) .....	178
par (load parameter file) .....	179
13.9.11 トレースコマンド .....	180
td (trace data display) .....	180
ts (trace search) .....	183
tf (trace file) .....	185
13.9.12 カバレッジコマンド .....	186
cv (coverage) .....	186
cvc (coverage clear) .....	188

13.9.13 コマンドファイル実行コマンド .....	189
com           (execute command file) .....	189
cmw           (execute command file with wait) .....	190
rec           (record commands to file) .....	191
13.9.14 ログコマンド .....	192
log           (log) .....	192
13.9.15 マップ情報表示コマンド .....	193
ma           (map information) .....	193
13.9.16 FPGA操作コマンド .....	194
xfer          (xilinx fpga data erase) .....	194
xfwr          (xilinx fpga data write) .....	195
xfcp          (xilinx fpga data compare) .....	196
xdp          (xilinx fpga data dump) .....	197
13.9.17 終了コマンド .....	198
q            (quit) .....	198
13.9.18 ヘルプコマンド .....	199
?            (help) .....	199
13.10 エラーメッセージ .....	200
Appendix A アセンブラ (サブツールチェーン) .....	203
A.1 パッケージの概要 .....	203
A.1.1 はじめに .....	203
A.1.2 ソフトウェアツールの概要 .....	203
A.2 プログラム開発手順 .....	205
A.2.1 開発フロー .....	205
A.2.2 ソースファイルの作成 .....	207
A.2.3 アセンブル .....	210
A.2.3.1 構造化プリプロセッサ(sap88) .....	210
A.2.3.2 クロスアセンブラ(asm88) .....	210
A.2.3.3 sap88、asm88の起動方法 .....	212
A.2.3.4 リロケータブルアセンブルの一括処理(ra88.bat) .....	213
A.2.3.5 リロケータブルオブジェクトファイル .....	218
A.2.3.6 アセンブリリストファイル .....	218
A.2.3.7 クロスリファレンスリスト .....	219
A.2.3.8 エラーリスト .....	220
A.2.3.9 アセンブルの実行例 .....	220
A.2.4 リンク .....	221
A.2.4.1 モジュールのリンク .....	221
A.2.4.2 セクション管理 .....	221
A.2.4.3 モジュールのアロケーション情報 .....	223
A.2.4.4 link88の起動方法 .....	224
A.2.4.5 リンクの一括処理(lk88.bat) .....	224
A.2.4.6 アブソリュートオブジェクトファイル .....	229
A.2.4.7 リンクの実行例 .....	229
A.2.5 プログラムデータHEXファイルの生成 .....	230
A.2.5.1 プログラムデータHEXファイル .....	230
A.2.5.2 hex88によるプログラムデータHEXファイルの生成方法 .....	230
A.2.5.3 モトローラS2フォーマットについて .....	231

A.2.6 シンボル情報 .....	232
A.2.6.1 シンボル情報の生成(rel88) .....	232
A.2.6.2 シンボリックテーブルファイルの生成(sym88) .....	234
Appendix B アセンブリソースファイルの作成方法 (サブツールチェーン) .....	235
B.1 概要 .....	235
B.1.1 ファイル名 .....	235
B.1.2 sap88、asm88におけるソースファイルの相違点 .....	235
B.1.3 マクロ命令 .....	235
B.2 ソースファイルの一般形式 .....	236
B.2.1 シンボル .....	237
B.2.2 ニーモニック .....	237
B.2.3 オペランド .....	237
B.2.4 コメント .....	237
B.2.5 数値表現 .....	238
B.2.6 文字 .....	238
B.2.7 ASCIIキャラクタセット .....	238
B.2.8 式 .....	239
B.2.9 演算子 .....	240
B.2.10 命令セット .....	241
B.2.11 レジスタ名 .....	241
B.2.12 アドレッシングモード .....	242
B.2.13 ニーモニックの表記例 .....	243
B.3 擬似命令 .....	244
B.3.1 領域設定擬似命令 .....	245
B.3.2 データ定義擬似命令 .....	247
B.3.3 シンボル定義擬似命令 .....	251
B.3.4 ロケーションカウンタ制御擬似命令 .....	253
B.3.5 外部定義・外部参照擬似命令 .....	254
B.3.6 ソースファイル挿入擬似命令 [sap88 only] .....	255
B.3.7 アセンブル終了擬似命令 .....	256
B.3.8 マクロ関係擬似命令 [sap88 only] .....	257
B.3.9 条件アセンブル擬似命令 [sap88 only] .....	266
B.3.10 出力リスト制御擬似命令 .....	270
Appendix C アセンブルツールリファレンス (サブツールチェーン) .....	273
C.1 構造化プリプロセッサ <sap88> .....	274
C.2 クロスアセンブラ <asm88> .....	276
C.3 リンカ <link88> .....	281
C.4 シンボル情報生成ユーティリティ <rel88> .....	285
C.5 シンボリックテーブルファイル生成ユーティリティ <sym88> .....	288
C.6 パイナリ/HEXコンバータ <hex88> .....	290
クイックリファレンス .....	293

# 1 概要

## 1.1 特長

S1C88 Family統合ツールパッケージはS1C88 Familyの全機種に共通のソフトウェア開発ツールで、ソースプログラムのコンパイル/アセンブルからデバッグまでの効率良い作業環境を提供します。

主な特長を以下に示します。

- ・統合化作業環境

Windows GUIアプリケーション《ワークベンチwb88》により、全ファイルの一元管理、makeの実行、ユーザ指定のエディタを含めたツールの起動など、統合化作業環境を実現しています。

- ・C言語、S1C88 Familyアセンブリ言語による開発をサポート

Cコンパイラツールチェーンに加え、従来よりの構造化アセンブラツールチェーンも含まれています。

- ・シミュレータ、自動評価システム、ICEによるデバッグをサポート

S1C88 Family開発ツールとして別途用意されているICE( S5U1C88000H5 )およびシミュレータを自動生成されるコマンドファイルで起動して、ビルド後のオブジェクトをすぐにデバッグできます。

- ・S1C88 Familyの全機種に対応

本パッケージのツールは、パラメータファイルや機種情報定義ファイルから機種固有の情報を読み込むことによって、S1C88 Familyの全機種に対応します。



本パッケージに含まれる主要なソフトウェアツールの概要を以下に示します。

## 統合開発環境

ワークベンチ( wb88.exe )

Windows GUIベースのアプリケーションで、統合開発環境を提供します。エディタを使用したソースの作成や編集、ファイルの選択、Cコンパイラツールチェーン( Main tool chain )の起動時オプションの選択、各ツールの起動やビルド処理などが、ワークベンチ上で可能です。

## Main tool chain

Cコンパイラ( c88.exe )

Cソースファイルをコンパイルしてアセンブリソースファイルを生成します。

アセンブラ( as88.exe )

コンパイラが生成したアセンブリソースファイルをアセンブルして、リロケータブルオブジェクトファイルにします。

リンカ( lk88.exe )

複数のリロケータブルオブジェクトファイルとライブラリをリンクし、1つのリロケータブルオブジェクトファイルにまとめます。

ロケータ( lc88.exe )

リンカが生成したリロケータブルオブジェクトファイルを再配置し、絶対アドレスを持つロードモジュールを生成します。このファイルでデバッグやマスクデータ作成を行います。

アドバンスドロケータ( alc88.exe )

ロケータが持つ再配置機能をDELFEEDによる記述ファイルを使用せずに実現します。また、分岐最適化機能が追加されています。アドバンスドロケータの詳細については、本書の5章を参照してください。

アドバンスドロケータを除き、Main tool chainの詳細については別冊の"S5U1C88000C Manual I"を参照してください。

## Sub tool chain

プリプロセッサ( sap88.exe )

アセンブリソースファイル内のプリプロセッサ命令等をアセンブル可能なソースコードに展開します。

アセンブラ( asm88.exe )

プリプロセッサで処理されたアセンブリソースファイルをアセンブルして、リロケータブルオブジェクトファイルにします。

リンカ( link88.exe )

アセンブラが生成したリロケータブルオブジェクトを再配置し、アブソリュートオブジェクトファイルを生成します。

HEXコンバータ( hx88.exe )

リンカが生成したアブソリュートオブジェクトファイルをモトローラS形式のHEXデータに変換します。このファイルでデバッグやマスクデータ作成を行います。

その他のツールも含め、Sub tool chainの詳細については本書のAppendixを参照してください。

## Development tool chain

ファンクションオプションジェネレータ( winfog.exe )

winfogはS1C88xxxのマスクオプションを選択し、ICEのファンクションオプション設定ファイルとICマスクパターンを生成するためのファンクションオプションドキュメントファイルを作成するツールです。

セグメントオプションジェネレータ( winsog.exe )

winsogはS1C88xxxのLCDセグメントオプションを設定し、ICEのセグメントオプション設定ファイルとICマスクパターンを生成するためのセグメントオプションドキュメントファイルを作成します。winsogはセグメントオプションを持つ機種以外では使用しません。

内蔵ROM未使用領域FF詰めユーティリティ( fil88xxx.exe )

プログラムデータHEXファイルから内蔵ROM領域を切り出し、その未使用領域にFFHを埋め込みます。また、システム予約領域にシステムコードを設定します。この作業は、ICEでプログラムをデバッグする前、およびwinmdcでマスクデータを作成する前に行う必要があります。

マスクデータチェッカ( winmdc.exe )

winmdcは開発が終了したプログラムファイル、オプションドキュメントファイルのデータをチェックし、セイコーエプソンへ提出するためのマスクデータファイルを作成するツールです。

Development tool chainの詳細については本書の6～12章を参照してください。

## デバッグツール

db88デバッガ( db88.exe )

ハードウェアツールとして用意されているICE( S5U1C88000H5 )を制御してデバッグを行うソフトウェアです。ブレークやステップ実行など、頻繁に使用するコマンドはツールバーに登録されており、キーボード操作の量を抑えています。また、ソースやレジスタ内容、コマンド実行結果がマルチウィンドウ上に表示できるため、デバッグ作業が効率良く行えます。db88デバッガの詳細については、本書の13章を参照してください。

ice88urデバッガ( ice88ur.exe )

ハードウェアツールとして用意されているICE( S5U1C88000H5 )を制御してデバッグを行うソフトウェアです。操作方法はWindowsのヘルプファイル( .hlp )に説明されており、スタートメニューから開くことができます( 英文ヘルプファイルはice88urのメニュー/ツールバーからも開けます )。

## PROM書き込みツール

ROMライターコントロールソフトウェア

専用PROMライターを制御してFlash EEPROM内蔵マイコンのPROMにデータを書き込むソフトウェアです。ツールとファームウェアが機種およびPROMライタの種類別に用意されています。PROMライターおよびデータ書き込み方法については各Flash EEPROM内蔵マイコンのテクニカルマニュアルを参照してください。



## 2 インストール

### 2.1 パッケージの内容

S1C88 Family統合ツールパッケージの内容を以下に示します。開梱時にすべてそろっていることを確認してください。

- |                                  |       |
|----------------------------------|-------|
| 1. CD-ROM( ツール、マニュアルPDF含む )..... | 1枚    |
| 2. 保証書.....                      | 和英各1通 |
| 3. 保証登録カード .....                 | 和英各1通 |

### 2.2 動作環境

S1C88 Family統合ツールパッケージを使用するには、次の動作環境が必要です。

#### パーソナルコンピュータ

IBM PC/ATまたは完全互換機が必要です。Pentium 200MHz以上のCPUと64MB以上のRAMを搭載した機種を推奨します。

別売のICE( S5U1C88000H5 )を使用するには、USBポートを搭載しWindows 2000またはWindows XPが動作する機種が必要です。

#### ディスプレイ

800×600ドット以上のディスプレイが必要です。

#### ハードディスクおよびCD-ROMドライブ

S1C88 Family統合ツールパッケージをインストールするには、CD-ROMドライブとハードディスク ( 50MB以上の空き容量 )が必要です。

#### システムソフトウェアについて

S1C88 Family統合ツールパッケージはMicrosoft Windows 2000( 日本語版、英語版 )およびWindows XP( 日本語版、英語版 )に対応しています。

別売のICE( S5U1C88000H5 )を使用するには、Windows 2000またはWindows XPが必要です。

#### その他

デバッグにインサーキットエミュレータを使用する場合、本パッケージとは別売のICE( S5U1C88000H5 )および周辺回路ボード( S5U1C88xxxP )が必要です。

## 2.3 インストール方法

ツールのインストールは添付のCD-ROMに収められたインストーラ( Setup.exe )によって行います。

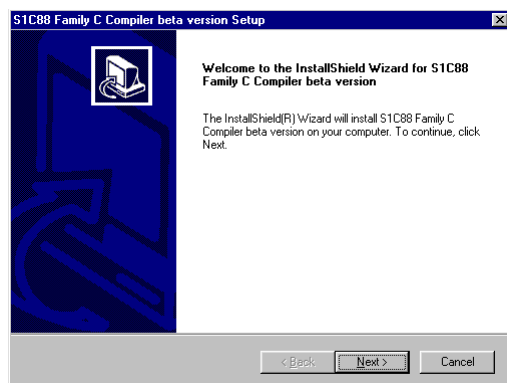
ツールをインストールするには



- (1) Windows 2000またはWindows XPを起動させてください。  
すでに起動している場合は、開いているプログラムをすべて終了させてください。

Setup.exe (2) CD-ROMをドライブに挿入し、その内容を表示させてください。

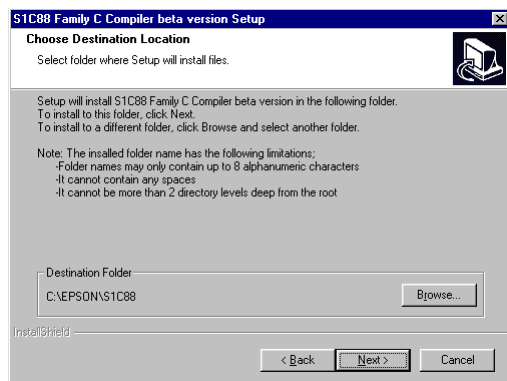
- (3) Setup.exeをダブルクリックして起動させてください。  
古いバージョンのツールがインストールされている場合、インストーラはワーニングメッセージを表示してインストールを中止します。古いバージョンをアンインストールし、再度インストーラを起動してください。



Welcom to ...

インストールウィザードのスタート画面が表示されます。

- (4) [Next >] ボタンをクリックして次に進めてください。

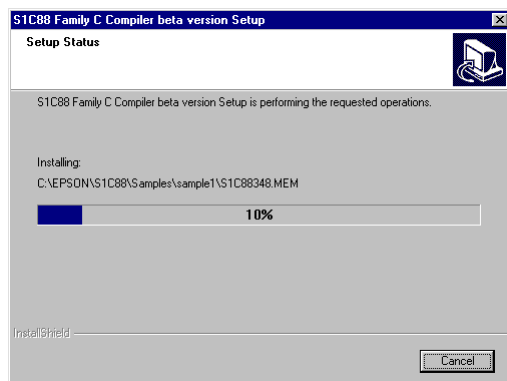


Choose Destination Location

インストールするフォルダを指定するダイアログボックスが表示されます。

- (5) デフォルト設定を変更しない場合は、[Next >] ボタンをクリックしてインストールを実行させてください。

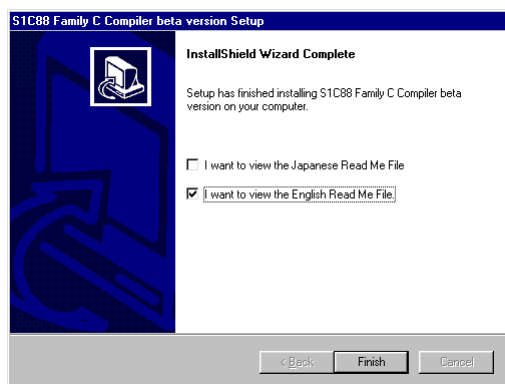
他のフォルダにインストールするには  
[Browse...] をクリックしてフォルダ選択用のダイアログボックスを表示させ、パスを入力するか、インストールするフォルダを選択します。[OK] ボタンをクリックして選択を終了し、[Next >] ボタンをクリックしてください。



注: デフォルト以外のフォルダにインストールする場合、選択可能なフォルダには以下の制限がありますので注意してください。

- フォルダ名が8文字以内であること
- フォルダ名にスペースが含まれていないこと
- サブディレクトリを選択する場合、ルートディレクトリから2レベル以内であること

この後、インストールが始まります。



### InstallShield Wizard Complete

(6) [Finish]をクリックしてインストーラを終了させてください。

終了後、場合によっては401Comupd.exeが実行されることがあります。

インストールを途中で中止するには

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

アンインストールするには

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

## 2.4 インストール後のディレクトリとファイル構成

インストーラは、指定ディレクトリ(デフォルトは"C:\EPSON\¥S1C88¥")に以下のファイルをコピーします。

[EPSON¥S1C88]	...	ReadMeテキストファイル(日本語)
README_J.TXT	...	ReadMeテキストファイル(英語)
README_E.TXT	...	環境設定用バッチファイル
ADDPATH.BAT	...	S1C88 Family C Compiler Tools
[¥BIN]	...	ワークベンチ
WB88.EXE	...	Cコンパイラ
C88.EXE	...	アセンブラ
AS88.EXE	...	リンカ
LK88.EXE	...	ロケータ
LC88.EXE	...	アドバンスドロケータ
ALC88.EXE	...	コントロールプログラム
CC88.EXE	...	makeプログラム
MK88.EXE	...	ライブラリメンテナ
AR88.EXE	...	オブジェクトリーダ
PR88.EXE	...	シンボリックテールファイルジェネレータ
SY88.EXE	...	S5U1C88000Hコントロールソフトウェア
ICE88UR.EXE	...	S5U1C88000Hヘルプファイル
ICE88UR.HLP	...	その他関連ファイル
...	...	
[¥SAP]	...	S1C88 Family Structured Assembler Tools
SAP88.EXE	...	プリプロセッサ
ASM88.EXE	...	アセンブラ
LINK88.EXE	...	リンカ
HEX88.EXE	...	HEXコンバータ
REL88.EXE	...	シンボル情報生成ユーティリティ
SYM88.EXE	...	シンボリックテールファイル生成ユーティリティ
[¥DB88]	...	DB88デバッグディレクトリ
DB88.EXE	...	DB88デバッグ
DEFAULT.PAR	...	デフォルトパラメータファイル
...	...	その他関連ファイル
[¥DEV]	...	
[¥BIN]	...	S1C88 Family Development Tool for Windows
WINFOG.EXE	...	ファンクションオプションジェネレータ
WINSOG.EXE	...	セグメントオプションジェネレータ
WINMDC.EXE	...	マスクデータチェッカ
[¥88xxx]	...	機種別ファイル
S1C88xxx.CPU	...	ロケータ記述ファイル
S1C88xxx.DSC	...	
S1C88xxx.MEM	...	
FIL88xxx.EXE	...	内蔵ROM未使用領域FF詰めユーティリティ
S1C88xxx.ini	...	機種情報定義ファイル
88xxx.PAR	...	ICEパラメータファイル
t88xxx.psa	...	ICE自己診断プログラム
t88xxx.fsa	...	
t88xxx.fdc	...	

```

[¥DOC]
  [¥JAPANESE]      ... ドキュメントフォルダ(日本語)
    REL_XXXX_J.TXT ... ツールのリリースノート
    TBD_J.PDF       ... マニュアル(PDF形式)
    TBD_J.PDF       ... クイックリファレンス(PDF形式)
  [¥HARD]           ... ハードウェアツールドキュメントフォルダ(日本語)
    PRC_COMMON_J.PDF ... 標準ペリフェラルボードマニュアル(PDF形式)
    ICE88UR_SETUP_J.PDF ... ICEマニュアル(PDF形式)

  [¥ENGLISH]        ... ドキュメントフォルダ(英語)
    REL_XXXX_E.TXT ... ツールのリリースノート
    TBD_E.PDF       ... マニュアル(PDF形式)
    TBD_E.PDF       ... クイックリファレンス(PDF形式)
  [¥HARD]           ... ハードウェアツールドキュメントフォルダ(英語)
    PRC_COMMON_E.PDF ... 標準ペリフェラルボードマニュアル(PDF形式)
    ICE88UR_SETUP_E.PDF ... ICEマニュアル(PDF形式)

[¥ETC]              ... デフォルトロケータ記述ファイル
  S1C88.DSC
  MK88.MK
  S1C88.CPU
  S1C88.MEM

[¥ICE]
  [¥FPGA]
    C88xxx.MOT      ... 標準ペリフェラルボード用FPGAデータ

[¥INCLUDE]          ... Cコンパイラ用ヘッダファイル

[¥LIB]              ... Cコンパイラ用ライブラリファイル
  [¥LIBCC]           ... コンパクトコードモデルライブラリオブジェクト
  [¥LIBCD]           ... コンパクトデータモデルライブラリオブジェクト
  [¥LIBCL]           ... ラージモデルライブラリオブジェクト
  [¥IBCS]            ... スモールモデルライブラリオブジェクト
  [¥SRC]             ... ライブラリソースファイル

[¥SAMPLES]          ... サンプルプログラムソース
                        サンプルプログラムの内容については、本フォルダ内のApplicationNote_J(E)
                        .PDFを参照してください。

[¥WRITER]
  [¥8xxxx] (Flashマイコン機種名)
    [¥URW2]
      RW8xxxx.EXE ... Universal ROM Writer IIコントロールソフトウェア
      8xxxx.FRM   ... ファームウェア
    [¥OBPW]
      OBW8xxxx.EXE ... オンボードプログラミングROMライタコントロールソフトウェア
      RW8xxxx.INI ... 機種情報設定用ファイル
    [¥MPRW]
      G8xxxxxx.EXE ... Multiple-Programming ROM Writerコントロールソフトウェア
  :
  * ROMライタおよびPROMプログラミングの詳細については、テクニカルマニュアルを参照してください。

```

## PDF形式のオンラインマニュアル

添付のオンラインマニュアルはPDF形式で作成されており、参照するにはAdobe Acrobat ReaderのVer. 4.0以降が必要です。

## 今後リリースされる機種のファイル

今後リリースされる機種用のファイルは、セイコーエプソンのマイクロコンピュータユーザーズサイトで提供いたします。インストールにつきましては、それぞれのReadmeファイルを参照してください。

## 2.5 環境設定

本パッケージのツールを使用するためには、以下の環境変数を設定しておく必要があります。

```

SET PATH=C:¥EPSON¥S1C88¥BIN;%PATH%
SET C88INC=C:¥EPSON¥S1C88¥INCLUDE
SET C88LIB=C:¥EPSON¥S1C88¥LIB

```

addpath.batには上記のコマンドが記述されていますので、ツールを使用する前に実行してください。デフォルト以外のディレクトリにインストールした場合は、上記の"EPSON¥S1C88¥"がインストール先に変更されます。

なお、wb88は起動時に環境変数を自動設定しますので、wb88から各ツールを起動する場合はaddpath.batを実行する必要はありません。

## 3 ワークベンチ

この章では、ワークベンチwb88の機能と操作方法を説明します。

### 3.1 特長

ワークベンチwb88は、ソースの作成/編集からデバッグまでの統合操作環境を実現するソフトウェアです。機能と特長を次にまとめます。

- ユーザの指定するエディタによる、エラーメッセージからのタグジャンプに対応したソース編集機能
- 必要なファイルやツールの設定などの情報をプロジェクトとして容易に管理可能
- 更新が必要なファイルのみを処理するmake処理による実行形式オブジェクトのビルド
- S1C88 Family Cコンパイラツールチェーンの実行、オプションの設定が可能
- 容易な操作環境を提供するWindows GUIインタフェース

### 3.2 起動および終了方法

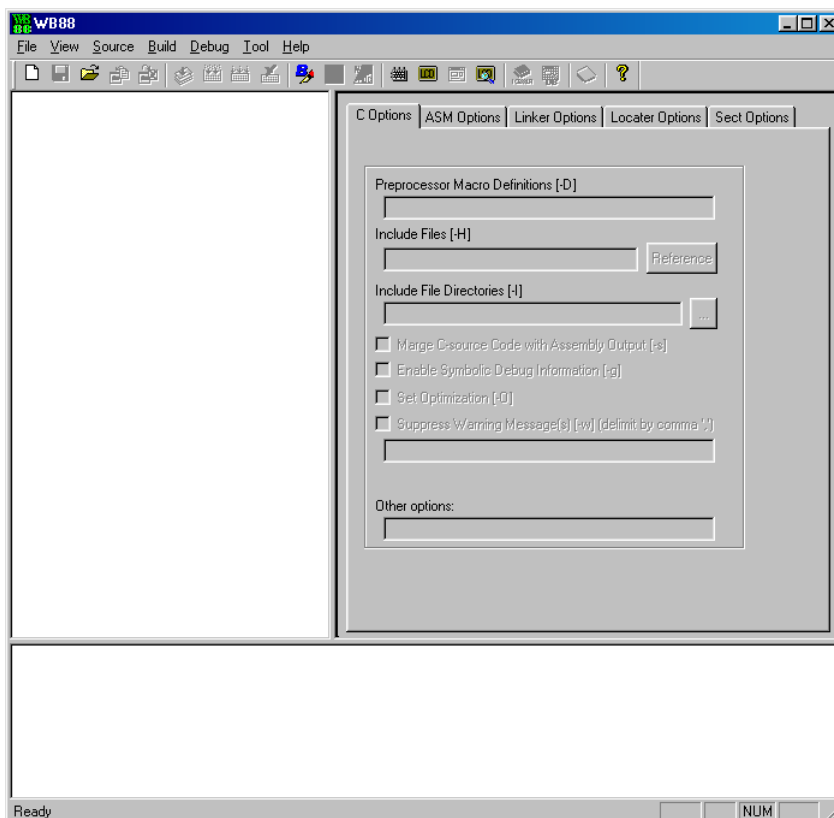
ワークベンチを起動するには



wb88.exe

wb88.exeのアイコンをダブルクリックして起動させてください。

ワークベンチが起動すると次の実行ウィンドウが表示されます。

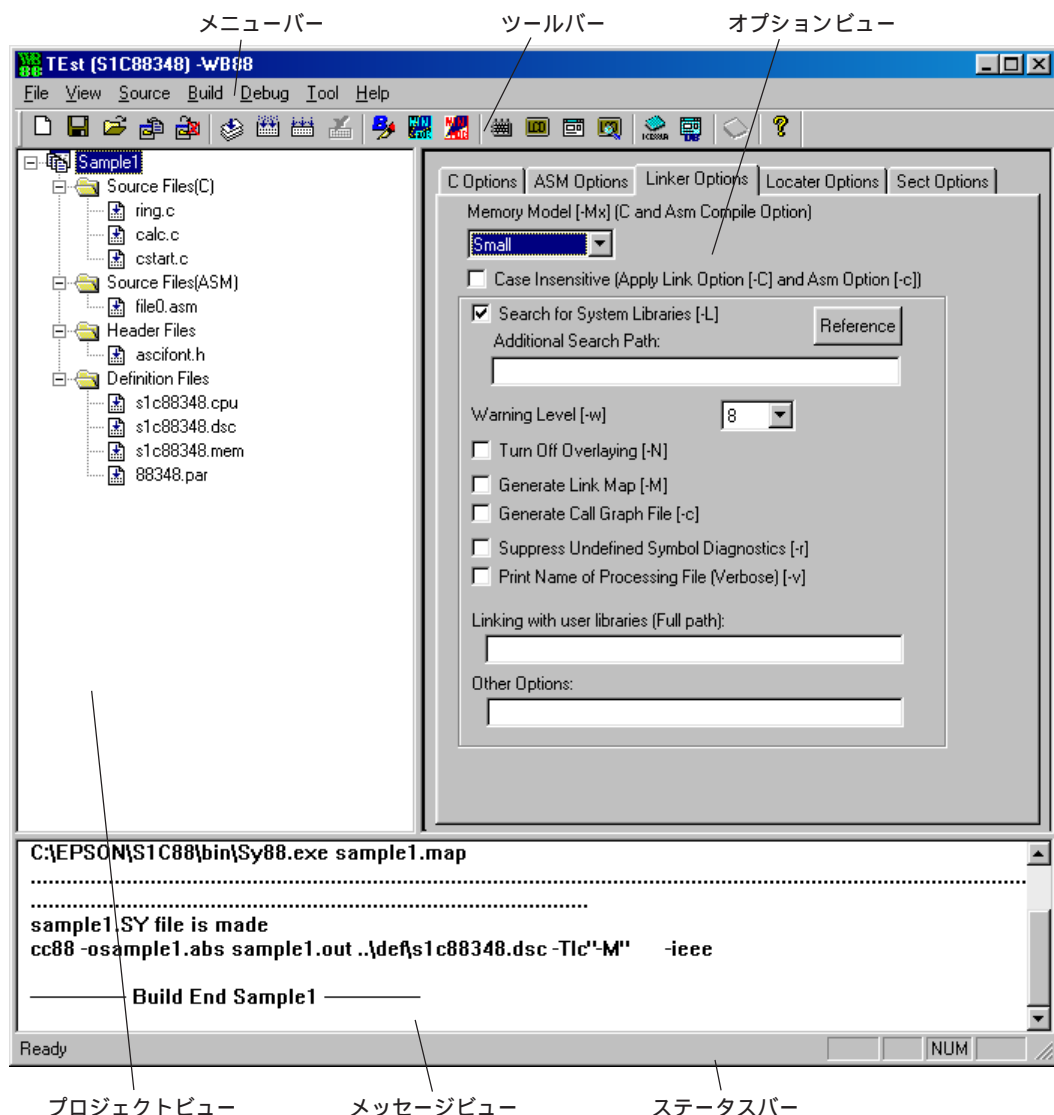


ワークベンチを終了するには

[File]メニューから[Exit]を選択してください。

### 3.3 ワークベンチウィンドウ

ワークベンチはプロジェクトビュー、オプションビューおよびメッセージビューで構成されます。



各ビューは、境界をドラッグすることによってサイズを変更することができます。

プロジェクトビューとメッセージビューの表示内容がウィンドウサイズに収まらない場合は標準のスクロールバーが表示されますので、それを使用して内容をスクロールさせてください。矢印キーも使用可能です。

#### プロジェクトビュー

現在開いているワークスペースフォルダとプロジェクト中のユーザが編集可能なファイルを、Windows エクスプローラと同様に表示します。ファイルは5つのノードに分類されて表示されます。

- ・ Project                      プロジェクト名(ワークスペースフォルダ名)
- ・ Source Files (C)            Cソースファイル(.c)
- ・ Source Files (ASM)        アセンブリソースファイル(.asm)
- ・ Header Files                ヘッドファイル(.h/.inc)
- ・ Definition Files            ユーザ編集可能な各種機種情報定義ファイル(.cpu/.dsc/.mem/.par)

リストされているソースファイルのアイコンをダブルクリックすると、指定のエディタがそのファイルを開き、編集可能となります。Definition Filesはセクションエディタの[Disable Making DELFEE]チェックボックスをONにした場合にのみ表示されます。

## オプションビュー

Cコンパイラ、アセンブラ、リンカ、ロケータのオプション、およびセグメントエディタを表示します。ここで、オプションの選択も行います。タグをクリックしてツールを選択する以外に、プロジェクトビュー内のノードあるいはファイルの選択に従って表示が切り替わります。詳細については、3.9項を参照してください。

## メッセージビュー

ビルドやコンパイル等の処理で実行されるツールが出力するメッセージを表示します。ここに表示されるソース行番号を含む文法エラーなどのエラーメッセージをダブルクリックすると、エラーが発生したソースファイルが指定のエディタにより開かれ、該当する行を表示します( wb88で実行可能なタグジャンプ機能を持つエディタを使用している場合のみ )。

## メニューバー

メニューの内容については3.5項を参照してください。

## ツールバー

ボタンについては3.4項を参照してください。

ツールバーの表示/非表示は、[View]メニューの[Tool Bar]で切り替えることができます。

ツールバーはウィンドウの左端または右端にドラッグすることで縦位置に変更することができます。また、ツールバー領域外へドラッグすると、ウィンドウ形式に変更されます。

## ステータスバー

マウスカーソルをメニュー項目やボタンに重ねると、その機能を示す簡易ヘルプが表示されます。

ステータスバーの表示/非表示は、[View]メニューの[Status Bar]で切り替えることができます。

### 3.4 ツールバーとボタン

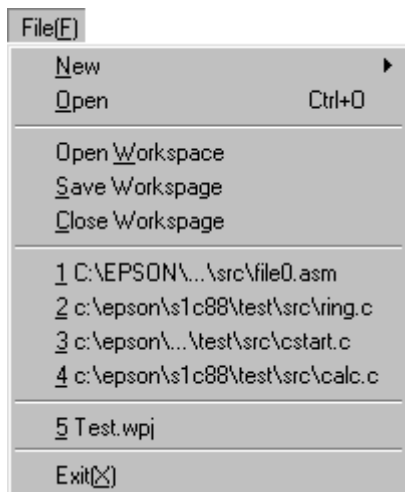
ワークベンチのツールバーには以下のボタンが登録されています。

-  [New Project]ボタン  
プロジェクトを新規作成します。
-  [Save Project]ボタン  
編集中のプロジェクトをファイルにセーブします。ファイルは上書きされます。このボタンは、プロジェクトを開いていない場合は無効になります。
-  [Insert a file]ボタン  
プロジェクトにソース/ヘッダファイルを追加します。このボタンは、プロジェクトを開いていない場合は無効になります。
-  [Remove a file]ボタン  
選択されているファイルをプロジェクトから削除します。
-  [Open]ボタン  
ファイルを選択するダイアログボックスが表示されます。ソースファイルやヘッダファイルを選択した場合は、指定のエディタが起動してファイルを開きます。
-  [Compile/Assemble]ボタン  
プロジェクトビュー内で選択されているファイルを、ソースの種類に応じてコンパイルまたはアセンブルします。
-  [Build]ボタン  
現在開いているプロジェクトのmake処理を行い、実行形式オブジェクトをビルドします。
-  [Rebuild]ボタン  
現在開いているプロジェクトのビルドを行います。ファイルの更新状況にかかわらず、すべてのソースのコンパイル/アセンブルから処理されます。
-  [Stop Build]ボタン  
実行中のビルド処理を中止します。
-  [BMPUtil]ボタン  
ビットマップユーティリティ BmpUtil を起動します。
-  [WinFOG]ボタン  
ファンクションオプションジェネレータ Winfog を起動します。
-  [WinMDC]ボタン  
マスクデータチェッカ Winmdc を起動します。
-  [PrtUtil]ボタン  
ポート設定ユーティリティ PrtUtil を起動します。
-  [LCDUtil]ボタン  
LCDパネルカスタマイズユーティリティ LCDUtil を起動します。
-  [Sim88]ボタン  
シミュレータ Sim88 を起動します。
-  [AutoEva]ボタン  
自動評価システム AutoEva を起動します。
-  [ICE88UR]ボタン  
ice88ur デバッガを起動します。
-  [DB88]ボタン  
db88 デバッガを起動します。
-  [ROM Writer]ボタン  
オンボードROMライターコントロールソフトウェアを起動します。
-  [About]ボタン  
wb88 のバージョンを表示します。



## 3.5 メニュー

### 3.5.1 [File]メニュー



このメニューにリストされているファイル名は最近開いたソースとプロジェクトファイルです。ここからの選択でも、そのファイルを開くことができます

#### [New - C Source File]

Cソースファイルを新規作成します。ファイル名を設定するダイアログボックスが表示され、その入力後に指定のエディタが起動して新規書類を開きます。作成したソースファイルはプロジェクト(プロジェクトビューのSource Files (C) ノード)に追加されます。

#### [New - Asm Source File]

アセンブリソースファイルを新規作成します。ファイル名を設定するダイアログボックスが表示され、その入力後に指定のエディタが起動して新規書類を開きます。作成したソースファイルはプロジェクト(プロジェクトビューのSource Files (ASM) ノード)に追加されます。

#### [New - Header File]

ヘッダファイルを新規作成します。ファイル名を設定するダイアログボックスが表示され、その入力後に指定のエディタが起動して新規書類を開きます。作成したヘッダファイルはプロジェクト(プロジェクトビューのHeader Files ノード)に追加されます。

#### [New - Project]

プロジェクトを新規作成します。

#### [Open] ([Ctrl]+[O])

ソースファイル、ヘッダファイル、プロジェクトを開きます。ファイルを選択するダイアログボックスが表示されます。ソースファイルやヘッダファイルを選択した場合は、指定のエディタが起動してファイルを開きます。

#### [Open Workspace]

プロジェクトを開きます。プロジェクトを選択するダイアログボックスが表示されます。

#### [Save Workspace]

編集中のプロジェクトをセーブします。

#### [Close Workspace]

現在開いているプロジェクトを閉じます。

#### [Exit]

wb88を終了します。

### 3.5.2 [View]メニュー



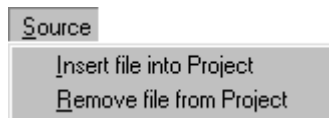
#### [Tool Bar]

ツールバーの表示/非表示を切り替えます。

#### [Status Bar]

ステータスバーの表示/非表示を切り替えます。

## 3.5.3 [Source]メニュー



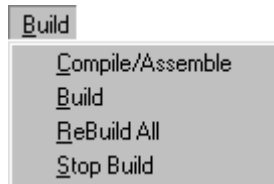
## [Insert file into Project]

指定のソースファイルを現在開いているプロジェクトに追加します。ソースファイルを選択するダイアログボックスが表示されます。

## [Remove file from Project]

プロジェクトビュー内で選択されているファイルをプロジェクトから削除します。実際のファイルが消去されることはありません。

## 3.5.4 [Build]メニュー



## [Compile/Assemble]

プロジェクトビュー内で選択されているファイルを、ソースの種類に応じてコンパイルまたはアセンブルします。

## [Build]

現在開いているプロジェクトのmake処理を行い、実行形式オブジェクトをビルドします。

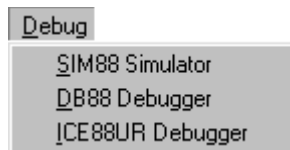
## [ReBuild All]

現在開いているプロジェクトのビルドを行います。ファイルの更新状況にかかわらず、すべてのソースのコンパイル/アセンブルから処理されます。

## [Stop Build]

実行中のビルド処理を中止します。

## 3.5.5 [Debug]メニュー



## [SIM88 Simulator]

シミュレータSim88を起動します。

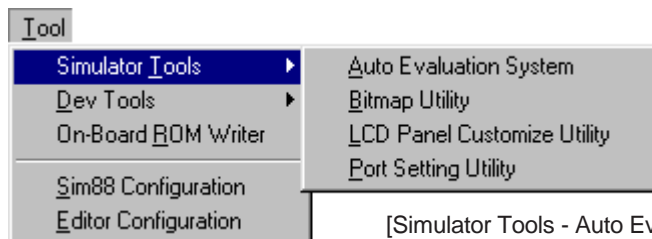
## [DB88 Debugger]

db88デバッガを起動します。

## [ICE88UR Debugger]

ice88urデバッガを起動します。

## 3.5.6 [Tools]メニュー



## [Simulator Tools - Auto Evaluation System]

自動評価システムAutoEvaを起動します。

## [Simulator Tools - Bitmap Utility]

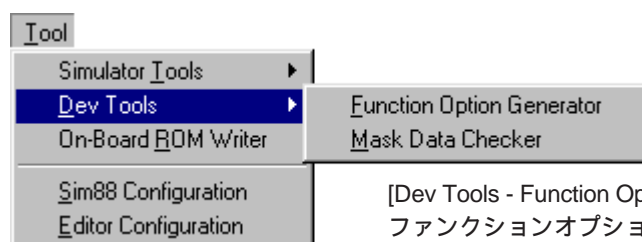
ビットマップユーティリティBmpUtilを起動します。

## [Simulator Tools - LCD Panel Customize Utility]

LCDパネルカスタマイズユーティリティLCDUtilを起動します。

## [Simulator Tools - Port Setting Utility]

ポート設定ユーティリティPrtUtilを起動します。



[Dev Tools - Function Option Generator]  
ファンクションオプションジェネレータWinfogを起動します。

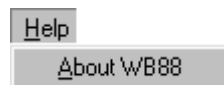
[Dev Tools - Mask Data Checker]  
マスクデータチェッカWinmdcを起動します。

[On-Board ROM Writer]  
オンボードROMライターコントロールソフトウェアを起動します。

[Sim88 Configuration]  
シミュレータSim88.exeのパスを設定します。

[Editor Configuration]  
エディタのパスとコマンドラインオプションを設定します。

### 3.5.7 [Help]メニュー



[About WB88]  
ワークベンチのバージョン情報を表示します。

## 3.6 プロジェクトとワークスペース

ワークベンチは、プログラム開発のタスクをワークスペースフォルダと、ファイル情報やツールの起動に必要な情報などを含むプロジェクトファイルで管理します。

### 3.6.1 プロジェクトの新規作成

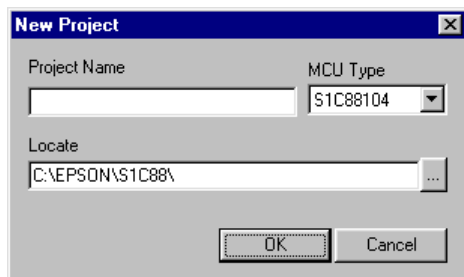
プロジェクトファイルは以下の手順で作成できます。

1. [File]メニューから[New Project]を選択するか、[New Project]ボタンをクリックします。

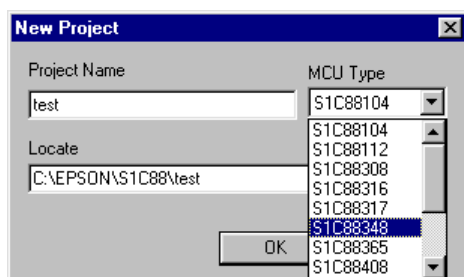


[New Project]ボタン

[New Project]ダイアログボックスが表示されます。



2. プロジェクト名を入力し、機種名とプロジェクトを保存するディレクトリを選択後、[OK]をクリックします。



\* [MCU Type]プルダウンリストには、"Dev"ディレクトリ内に存在する機種名がリストされます。

ワークベンチはワークスペースとして、[Locate]で指定したフォルダ(ディレクトリ)を作成し、その中にプロジェクトファイル(<プロジェクト名>.wpj)と下記のフォルダを作成します。

指定位置にすでに同じ名前のフォルダが存在する場合は、そのフォルダをそのままワークスペースとして使用します。

ここで指定したプロジェクト名は、ビルドにより生成されるアブソリュートオブジェクトファイルやその他のファイル名としても使用されます。

#### ワークスペース内のフォルダ

- def: アドバンスドロケータ定義ファイルなど、各種定義ファイルが格納されます。  
プロジェクトの新規作成によりテンプレートとして使用可能な定義ファイルがコピーされますので、これを必要に応じて修正して使用します。
- obj: ビルド時に生成される中間ファイルが格納されます。
- src: wb88から新規作成したソースファイルやヘッダファイルが格納されます。(他のフォルダにあるソースファイル等をプロジェクトに追加しても、ここにはコピーされません。)
- tmp: ビルド時や各ツール実行時に一時的に作成されるファイルが格納されます。

各フォルダに置かれるファイルの種類については、3.12項の"ファイル一覧"を参照してください。

### 3.6.2 プロジェクトへのソースファイル/ヘッダファイルの登録

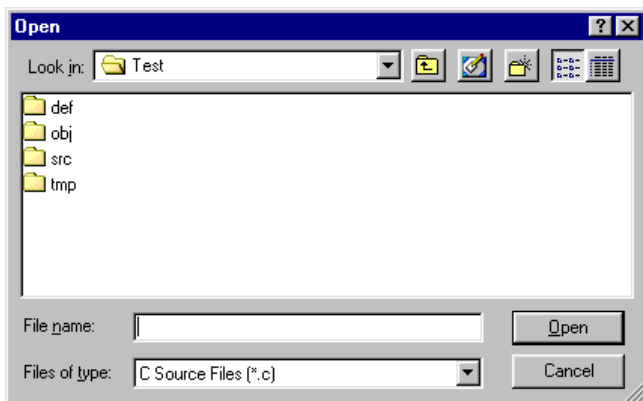
作成したソースファイルはプロジェクトに登録しておく必要があります。  
これには、以下に示す2つの方法のいずれかを使用してください。

1. [Source | Insert file into Project]メニューコマンドまたは[Insert a file]ボタン



[Insert a file]ボタン

このメニューコマンドを選択、またはボタンをクリックすると、ダイアログボックスが表示されます。



ファイル形式 (Cソース、アセンブリソース、ヘッダファイル) を指定後、ファイルを選択し、[Open]ボタンをクリックします。

選択したファイルはプロジェクトに追加され、プロジェクトビュー内に表示されます。

注: プロジェクトには選択したファイルの参照情報が登録されます。ワークスペース内にファイルがコピーされる訳ではありませんので、プロジェクトに追加後はファイルを移動しないでください。移動した場合は、一度そのファイルをプロジェクトから削除し (次項参照) 再度追加の操作を行ってください。

2. [File | New]メニューコマンド

このメニューコマンドでソースファイルまたはヘッダファイルを新規作成すると、そのファイルは自動的に現在開いているプロジェクトに追加されます。ソースファイル/ヘッダファイルの新規作成については3.7.2項を参照してください。

新規作成したファイルはプロジェクトに追加され、プロジェクトビュー内に表示されます。

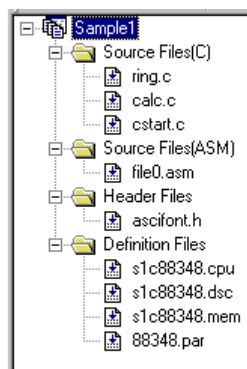
### 3.6.3 プロジェクトからのファイルの削除

ソースファイルやヘッダファイルをプロジェクトから削除するには、プロジェクトビュー上でそのファイル名を選択し、[Source]メニューから[Remove file from Project]を選択するか[Remove a file]ボタンをクリック、あるいは[Delete]キーを押します。この操作は、プロジェクトからファイル情報を削除するのみで、実際のファイルには影響を与えません。



[Remove a file]ボタン

### 3.6.4 プロジェクトビュー



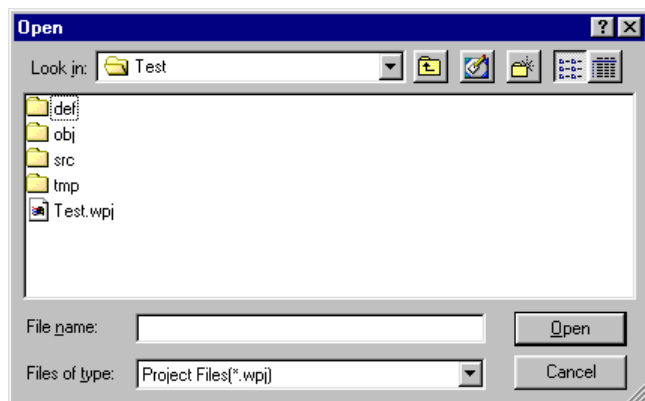
プロジェクトビューはワークスペースフォルダと、現在開いているプロジェクトに含まれるソースファイル、ヘッダファイル、ユーザ編集可能な各種定義ファイルの一覧を表示します。

表示されているファイルのアイコンまたはファイル名をダブルクリックすると、指定のエディタが起動してそのファイルを開きます。エディタはデフォルトでWindowsのメモ帳に設定されていますが、[Tool]メニューの[Editor Configuration]で変更可能です。

注: プロジェクトビューのリストは、実際のディレクトリ構造を表している訳ではありません。

### 3.6.5 プロジェクトのオープン/クローズ

プロジェクトを開くには、[File]メニューから[Open Workspace]を選択してください。  
プロジェクトファイルを選択するダイアログボックスが表示されます。



ワークベンチは、複数のプロジェクトを同時に開くことができません。プロジェクトが開いている状態で他のプロジェクトをオープンすると、前のプロジェクトは閉じられます。このとき、前のプロジェクトが修正後に保存されていない場合は、プロジェクトを保存するかどうかを選択するダイアログボックスが表示されます。

プロジェクトファイルは[File]メニューの[Open]または[Open]ボタンによっても開くことができます。



[Open]ボタン

この場合は、ファイルオープンダイアログボックス内のファイル形式をProject Files( \*.wpj )にしてプロジェクトファイルを選択してください。

現在開いているプロジェクトを閉じるには、[File]メニューの[Close Workspace]を選択してください。プロジェクトが修正後に保存されていない場合は、プロジェクトを保存するかどうかを選択するダイアログボックスが表示されます。[Yes]を選択すると、ファイル構成、ツール設定情報を含むすべての変更項目が保存されます。

### 3.6.6 プロジェクトの保存

編集中のプロジェクトを保存するには、[File]メニューから[Save Workspace]を選択するか、[Save Project]ボタンをクリックしてください。



[Save Project]ボタン

このほか、プロジェクトが保存されていない状態で以下の操作を行うと保存を確認するダイアログボックスが表示されますので、そこで保存することもできます。

- ・プロジェクトのオープン( [File]メニューの[Open Workspace]または[Open]でプロジェクトを選択 )
- ・プロジェクトのクローズ( [File]メニューの[Close Workspace] )
- ・プロジェクトの新規作成( [File]メニューの[New Project] )
- ・コンパイル/アセンブル( [Build]メニューの[Compile/Assemble] )
- ・ビルド( [Build]メニューの[Build] )
- ・リビルド( [Build]メニューの[ReBuild All] )

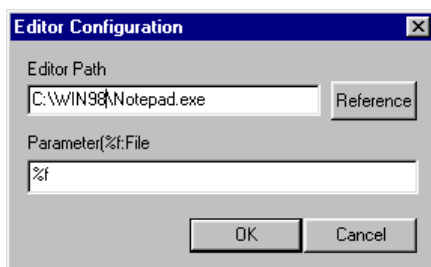
### 3.7 ソースファイルの作成/編集

ワークベンチ自体にはソースエディタ機能はありませんが、指定のエディタを起動させるとともにファイル情報や行番号情報をエディタに渡す機能があります。これにより、ソースの作成と編集、エラーメッセージからのタグジャンプが行えます。

#### 3.7.1 エディタの指定

ワークベンチは、ソースファイル/ヘッダファイルの新規作成、オープンが選択された場合、あるいはプロジェクトビューにリストされているファイル名がダブルクリックされた場合に、エディタを起動してファイル情報を渡します。デフォルトのエディタはWindowsのメモ帳に設定されています。これを他のエディタに変更するには、

1. [Tool]メニューから[Editor Configuration]を選択します。  
[Editor Configuration]ダイアログボックスが表示されます。



ここに、以下の内容を入力します。

##### [Editor Path]

使用するエディタのパスを入力するか、[Reference]ボタンにより表示されるファイル選択ダイアログボックスでエディタを選択します。

##### [Parameter]

エディタの起動時にファイル名と行番号(タグジャンプ用)を指定するコマンドラインオプションの正規表現を入力します。

"%f"はファイル名に、"%l"は行番号に置き換えられてエディタに送られます。

上記のデフォルト設定の例では、

C:\¥Win98¥Notepad.exe <指定ファイル名>

をコマンドラインとしてメモ帳を起動します。

たとえば、ファイル名はメモ帳と同様、タグジャンプは"/j<行番号>"のオプションをファイル名の前に指定するエディタの場合は次のように設定します。

/j%l %f

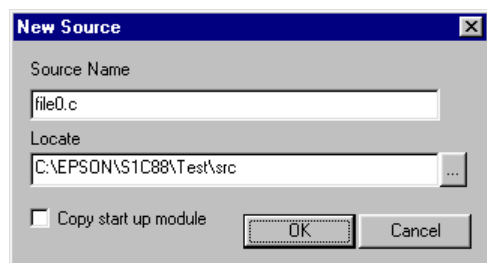
注: デフォルトのメモ帳では、タグジャンプの機能は使用できません。

2. [OK]ボタンをクリックします。  
使用するエディタが変更されます。

### 3.7.2 ソースファイル/ヘッダファイルの新規作成

ソースファイル/ヘッダファイルを新規作成するには、

1. [File]メニューから[New | C Source File]、[New | Asm Source File]または[New | Header File]を選択します。  
[New Source]ダイアログボックスが表示されます。



Cソースの例

[Source Name]

ソースファイル名を入力します。ソースの種類に従って以下の拡張子を使用してください。

- .c Cソースファイル
- .asm アセンブリソースファイル
- .h ヘッダファイル
- .inc インクルードファイル

[Locate]

ソースファイルを作成するディレクトリを入力します。[...]ボタンで表示されるダイアログで選択することもできます。

デフォルト位置としてワークスペース内のsrcフォルダが表示されますので、特に他を選択する必要がなければ、そのまま使用してください。

[Copy start up module]

このチェックボックスは、Cソースファイルを選択した場合にのみ表示されます。ここをチェックしておくと、新規作成するCソースファイルにCスタートアップモジュールの雛形からコードがコピーされます。雛形のファイルは¥EPSON¥S1C88¥LIB¥SRCフォルダにあるcstart.cです。

2. [OK]ボタンをクリックします。  
指定したソースファイルが作成され、設定されているエディタが起動してそのファイルを開きます。  
作成されたファイルはプロジェクトビューに表示されるプロジェクトツリーにも追加されます。
3. エディタ上でソースコードを入力し保存します。

### 3.7.3 ファイルの編集

ソースファイルの修正や印刷なども設定されているエディタで行います。

ソースファイルは、次の2つの方法のいずれかで開いてください。

1. [File]メニューから[Open]を選択するか、[Open]ボタンをクリックします。



[Open]ボタン

[Open]ダイアログボックスが表示されますので、ファイル形式(Cソース、アセンブリソース、ヘッダファイル)を指定後、ファイルを選択し、[Open]ボタンをクリックします。

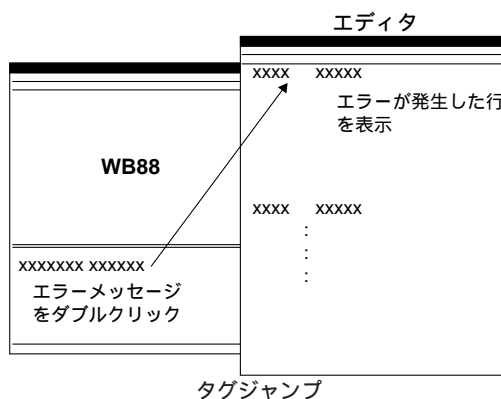
2. プロジェクトビューに表示されているファイル名をダブルクリックします。  
[Definition Files]にリストされている定義ファイルを開くこともできます。

1と2のいずれかの操作によりエディタが起動し、選択したファイルを開きます。エディタで必要な作業を行ってください。



### 3.7.4 タグジャンプ機能

コンパイルやアセンブルで文法エラーが発生すると、そのエラーメッセージがメッセージビューに表示されます。エラーメッセージがソース行番号を含んでいる場合、このメッセージをダブルクリックすることでエディタが起動して該当するソースファイルを開き、エラーが発生したソース行にジャンプすることができます。



注: タグジャンプ機能を使用するには、エディタがコマンドラインによるタグジャンプをサポートし、[Tool | Editor Configuration]でコマンドラインオプションが正しく設定されている必要があります(デフォルト設定のメモ帳では使用できません)。

## 3.8 ビルド

[Build]メニューまたはツールバーボタンにより、Cコンパイラツールチェーンによるビルド(ソースファイルから実行形式のオブジェクトファイルを生成)、コンパイル/アセンブルをワークベンチから実行することができます。

各ツールの詳細については、Cコンパイラのマニュアルを参照してください。

### 3.8.1 ビルドの準備

ビルドを実行する前に、必要なソースファイルを作成し、各ツールのオプションを設定しておきます。

1. プロジェクトを新規作成(3.6.1項参照)
2. ソースファイルの作成とプロジェクトへの登録(3.7項、3.6.2項参照)
3. alc88使用時: セクションエディタによるアドバンスドロケータ定義ファイルの編集(3.9.5項参照)  
lc88使用時: ロケータ記述ファイルの編集(3.7.3項、Cコンパイラのマニュアル参照)
4. ツールオプションの選択(3.9項参照)

### 3.8.2 実行形式オブジェクトのビルド

実行形式のオブジェクトファイルを作成するには、

1. プロジェクトファイルを開きます。
2. [Build]メニューから[Build]を選択するか、[Build]ボタンをクリックします。



[Build]ボタン

ワークベンチは、プロジェクト内のソースファイル構成とユーザが設定したツールオプションに従ってmakeファイルを作成します。このファイルは各ツールの実行を制御するために使用されます。

make処理は最初にコンパイラを起動し、各Cソースファイルをコンパイルします。すでに最新のアセンブリソースファイルが作成されている場合は、処理時間を短縮するため、そのCソースはコンパイルされません。

同様にアセンブラを実行し、リロケータブルオブジェクトを作成します。

次に、リンカを起動してリンカオブジェクトファイルを作成します。

最後にアドバンスドロケータまたはロケータを実行し、実行形式のオブジェクトファイルを作成します。

最新のアセンブリソースやリロケータブルオブジェクトファイルを含め、すべてのファイルを対象に再度ビルドを行う場合は、[Build]メニューの[ReBuild All]を選択するか、[Rebuild]ボタンをクリックします。



[Rebuild]ボタン

ビルドの開始後に実行を中止するには、[Build]メニューから[Stop Build]を選択するか、[Stop Build]ボタンをクリックしてください。



[Stop Build]ボタン

#### アドバンスドロケータalc88とロケータlc88の選択

リンク後のリロケータブルオブジェクトを絶対アドレスに再配置する処理は、アドバンスドロケータalc88またはロケータlc88が行います。どちらを使用するかは、ロケータオプション([Locator Options]タブの画面)の[Disable branch optimize]チェックボックスで選択できます。

[Disable branch optimize] = OFF(デフォルト)      alc88を実行

[Disable branch optimize] = ON      lc88を実行

alc88とlc88の違いは次のとおりです。

表3.8.2.1 alc88とlc88の相違点

項目	アドバンスドロケータalc88	ロケータlc88
定義ファイル	アドバンスドロケータ定義ファイル(.inf)	ロケータ記述ファイル(DELFEЕ)(.dec, .mem, .cpu)
定義ファイルの作成方法	wb88のセクションエディタを使用(DELFEЕの修得は不要)	wb88のセクションエディタを使用またはDELFEЕ言語でユーザが作成
CARL命令の分岐最適化機能	あり	なし

基本的には、アプリケーションのバージョンアップなど、既存のロケータ記述ファイルを使用する必要がある場合を除き、分岐最適化機能を持つalc88の使用を推奨します。  
定義ファイルの作成等、詳細については"3.9.5 セクションエディタ"を参照してください。

### 3.8.3 コンパイラ/アセンブラのみの実行

ソースファイルを個々にコンパイルまたはアセンブルすることもできます。  
コンパイラまたはアセンブラのみを起動するには、コンパイル/アセンブルするソースをプロジェクトビューから選択し、[Build]メニューの[Compile/Assemble]を選択するか、[Compile/Assemble]ボタンをクリックします。

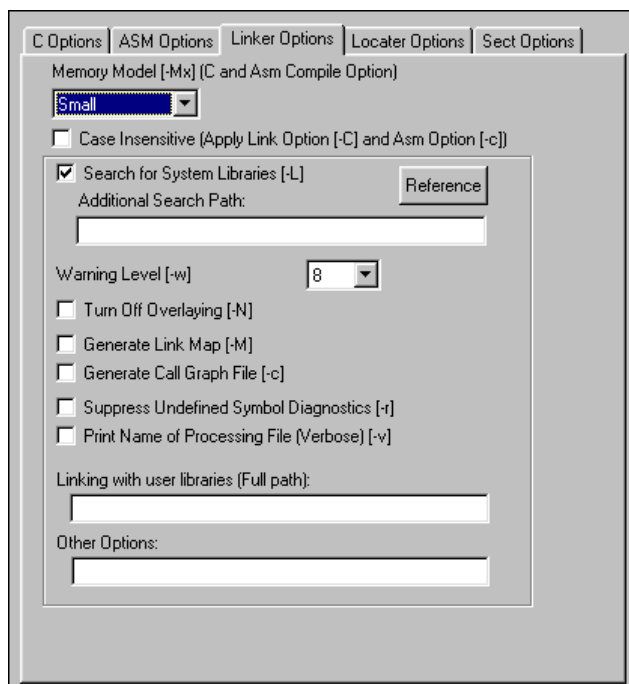


[Compile/Assemble]ボタン

選択したファイルの種類により、コンパイラまたはアセンブラのどちらかが起動してファイルを処理します。

### 3.9 ツールオプションの設定

ビルドで実行される各ツールは起動時に指定可能なオプションを持っています。  
ワークベンチでは、オプションビュー上で、それらのオプションを選択可能です。



オプションビュー

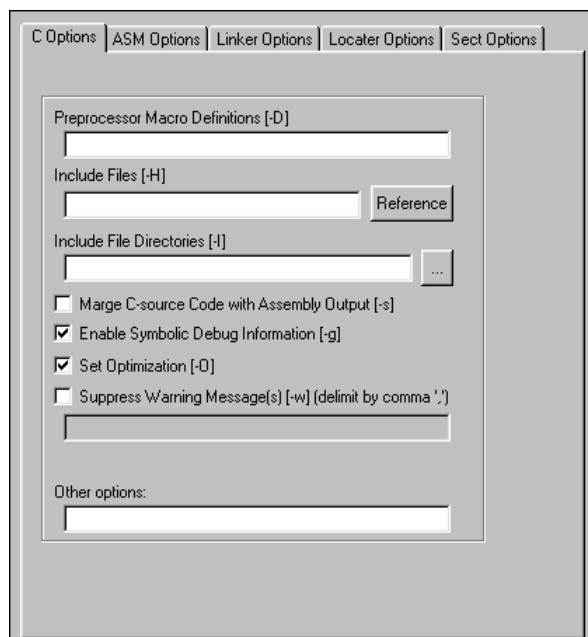
各ツールのオプションは、ツール名のタブをクリックすると表示されます。

また、プロジェクトビュー上の選択によっても表示されるツールオプションが次のように変わります。

- |                            |   |
|----------------------------|---|
| 1. プロジェクト名を選択              | リンカオプションを表示します。                               |
| 2. [Source Files (C)]を選択   | デフォルトコンパイルオプション(全Cソースに適用される)を表示します。           |
| 3. Cソースファイルを選択             | ローカルコンパイルオプション(選択しているCソースにのみ適用される)を表示します。     |
| 4. [Source Files (ASM)]を選択 | デフォルトアセンブルオプション(全アセンブリソースに適用される)を表示します。       |
| 5. アセンブリソースファイルを選択         | ローカルアセンブルオプション(選択しているアセンブリソースにのみ適用される)を表示します。 |

オプションビューで選択した各ツールのオプションは、次のツールの実行時から有効になります。

### 3.9.1 コンパイラオプション



この画面では、以下のコンパイラオプションが選択できます。

**Preprocessor Macro Definitions**                      c88の"-Dmacro[=def]"オプション  
 プリプロセッサマクロを定義します。テキストボックスには次の形式で入力してください。  
 マクロ名   または   マクロ名=定義内容

**Include Files**    c88の"-H file"オプション  
 コンパイル前にインクルードするファイル名を指定します。  
 [Reference]ボタンにより表示されるダイアログボックスでインクルードするファイルを選択することもできます。

**Include File Directories**                              c88の"-Idirectory"オプション  
 パスが指定されていないインクルードファイルを検索するディレクトリを指定します。  
 [...]ボタンにより表示されるダイアログボックスでフォルダを選択することもできます。

**Merge C-source Code with Assembly Output**      c88の"-s"オプション  
 このオプションを選択すると、Cソースコードをアセンブラ出力とマージして出力します。

**Enable Symbolic Debug Information**              c88の"-g"オプション  
 このオプションを選択すると、出力ファイルにシンボリックデバッグ情報を含めます。

**Set Optimization**                                      c88の"-O"オプション  
 このオプションを選択すると"-O1"が指定され、最適化処理を行います。チェックを外すと"-O0"が指定され最適化は行いません。

**Suppress Warning Message (s)**                      c88の"-w[num]"オプション  
 このオプションを選択すると、警告メッセージが抑制されます。  
 すべての警告メッセージを抑制する場合はテキストボックスを空白のままにしてください。  
 抑制する警告メッセージを指定する場合は、メッセージ番号をテキストボックスに入力します。複数の番号を入力する場合は、それぞれをカンマ(,)で区切ってください。

**Other options**  
 上記以外のオプションを指定したい場合(上記オプションも可)、ここにコマンドラインの書式でオプションを入力しておくことができます。

## コンパイラオプション指定における注意事項

-gオプション( Enable Symbolic Debug Information )と-O1オプション( Set Optimization )の両方を選択した場合、コンパイル時に-W555のワーニングメッセージが出力されます。

-O1オプションを指定すると、ソースに記述されているシンボルがコードの最適化によって実際には使用されない場合が発生します。この場合、-gオプションを指定してもそのシンボルのデバッグ情報は.absファイルに出力されません。

例: `int x,y,xy;`

`x = GLOBAL_X * 100;`

`y = GLOBAL_Y * 100;`

`xy = x * y;`

この例では最適化により変数xyは存在しなくなりますので、デバッグ時にxyの内容を参照することはできません。

-O0オプション( 最適化OFF )を指定して作成した実行ファイルを評価後、-O1オプション( 最適化ON )を指定して作成し直した場合は動作保証できませんので、再検証を行ってください。

## 表示されないオプションについて

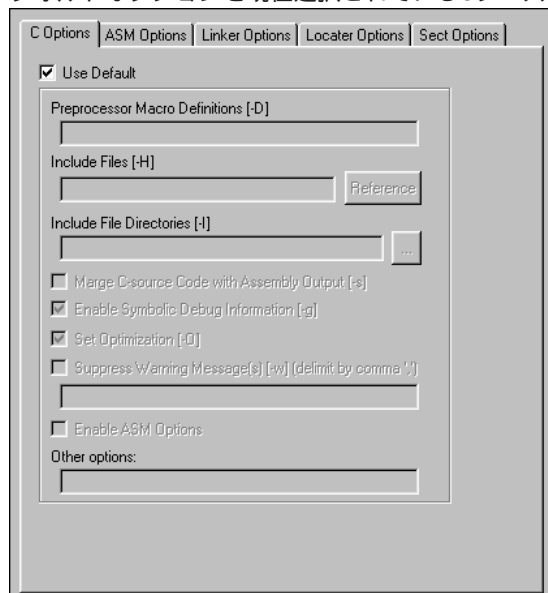
オプションビューに表示されていないCコンパイラのオプションの扱いについては以下のとおりです。

- e 内部処理で使用します。
- err Cコンパイラメッセージはメッセージウィンドウに表示されるとともに、エラーログファイルにも出力されます。
- f file 内部処理と競合するため使用不可。
- o file ソースファイル名を出力ファイルにも使用します。
- V wb88では使用しません。
- M{src|dll} リンカオプション設定画面で指定します。

## デフォルトオプションとローカルオプション

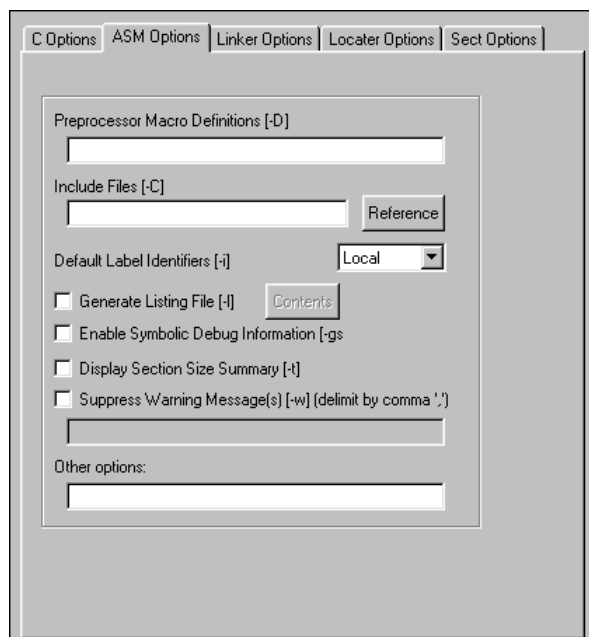
プロジェクトビューで個別のCソースファイルを選択している場合、このオプション設定画面はそのCソースファイルにのみ適用されるローカルオプションを表示します。プロジェクトビュー上で特に何も選択していない場合、および個別のCソースファイル以外を選択している場合は、すべてのCソースに適用されるデフォルトオプションを表示します。

ローカルオプションを表示している場合は、次の画面の例のように[Use Default]ボタンが追加され、デフォルトオプションを現在選択されているCソースファイルに適用するか指定できます。



Cソースごとにコンパイルオプションを変えたい場合は、[Use Default]ボタンのチェックを外し、各オプションを設定し直してください。

### 3.9.2 アセンブラオプション



この画面では、以下のアセンブラオプションが選択できます。

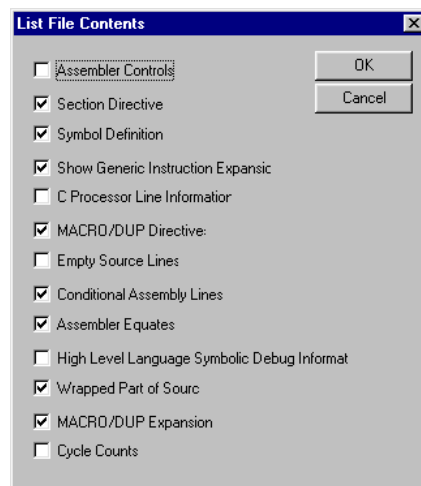
**Preprocessor Macro Definitions** as88の"-Dmacro[=def]"オプション  
 プリプロセッサマクロを定義します。テキストボックスには次の形式で入力してください。  
 マクロ名 または マクロ名=定義内容

**Include Files** as88の"-C file"オプション  
 ソースの前にインクルードするファイル名を指定します。  
 [Reference]ボタンにより表示されるダイアログボックスでインクルードするファイルを選択することもできます。

**Default Label Identifiers** as88の"-i[! ! g]"オプション  
 デフォルトのラベルスタイルをローカルまたはグローバルとして指定します。プルダウンリストから選択してください。

**Generate Listing File** as88の"-l"オプション  
 このオプションを選択すると、リストファイルを生成します。

**Contents** as88の"-L"オプション  
 このボタンは[Generate Listing File]を選択するとアクティブになります。ボタンをクリックすると次のダイアログボックスが表示され、リストファイルから削除するソース行の種類を選択することができます。デフォルトのオプション設定内容は"-LcDEIMnPQsWXy"です。



Enable Symbolic Debug Information as88の"-gs"オプション

このオプションを選択すると、出力ファイルにシンボリックデバッグ情報を含めます。

Display Section Size Summary as88の"-t"オプション

このオプションを選択すると、アセンブル時にセクションの要約をメッセージビューに表示します。

Suppress Warning Message (s) as88の"-w[num]"オプション

このオプションを選択すると、警告メッセージが抑制されます。

すべての警告メッセージを抑制する場合はテキストボックスを空白のままにしてください。

抑制する警告メッセージを指定する場合は、メッセージ番号をテキストボックスに入力します。複数の番号を入力する場合は、それぞれをカンマ(,)で区切ってください。

#### Other options

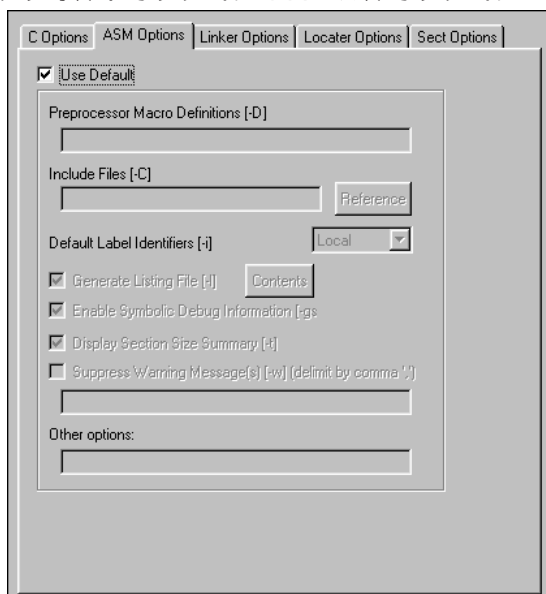
上記以外のオプションを指定したい場合(上記オプションも可) ここにコマンドラインの書式でオプションを入力しておくことができます。

#### 表示されないオプションについて

オプションビューに表示されていないアセンブラのオプションの扱いについては以下のとおりです。

- e 内部処理で使用します。
- err アセンブラメッセージはメッセージウィンドウに表示されるとともに、エラーログファイルにも出力されます。
- f file 内部処理と競合するため使用不可。
- o file ソースファイル名を出力ファイルにも使用します。
- V wb88では使用しません。
- v wb88では使用しません。
- c リンカオプション設定画面で指定します。
- M{s;c;d;l} リンカオプション設定画面で指定します。

#### デフォルトオプションとローカルオプション



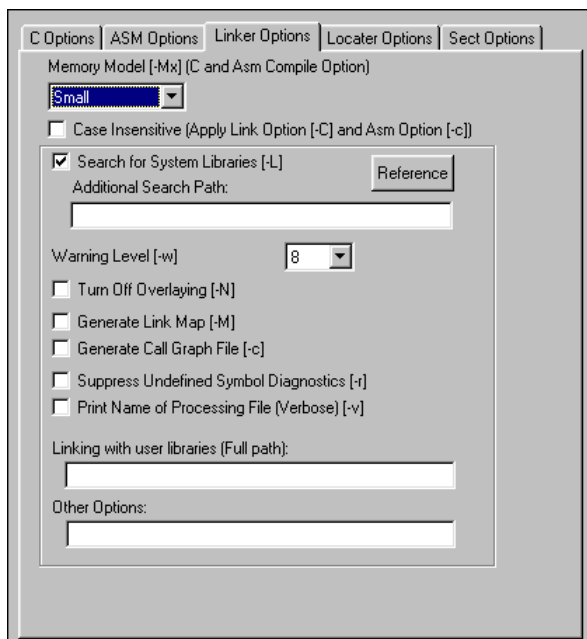
プロジェクトビューで個別のアセンブリソースファイルを選択している場合、このオプション設定画面はそのアセンブリソースファイルにのみ適用されるローカルオプションを表示します。プロジェクトビュー上で特に何も選択していない場合、および個別のアセンブリソースファイル以外を選択している場合は、すべてのアセンブリソースに適用されるデフォルトオプションを表示します。

ローカルオプションを表示している場合は、次の画面の例のように[Use Default]ボタンが追加され、デフォルトオプションを現在選択されているアセンブリソースファイルに適用するか指定できます。

アセンブリソースごとにアセンブラオプションを変えたい場合は、[Use Default]ボタンのチェックを外し、各オプションを設定し直してください。



### 3.9.3 リンカオプション



この画面では、以下のオプションが選択できます。

- Memory Model** c88/as88の"-M{s|c|d|i}"オプション  
メモリモデルをスモール、コンパクトコード、コンパクトデータ、ラージから選択します。この設定は、コンパイルおよびアセンブルの際に使用されます。
- Case Insensitive** as88の"-c"オプションおよびlk88の"-C"オプション  
このオプションを選択すると、大文字と小文字を区別しないでアセンブルおよびリンクを行います。
- Search for System Libraries** lk88の"-L"オプション  
このオプションを選択すると、システムライブラリの検索を行います。チェックを外すとシステムライブラリの検索をスキップします。  
このオプションを選択し、[Additional Search Path]を空白のままにした場合は、環境変数C88LIBで指定したディレクトリ内のみを検索します。それ以外のディレクトリも検索させる場合は、[Additional Search Path]にパスを入力するか、[Reference]ボタンでディレクトリを選択してください。
- Warning Level** lk88の"-w n"オプション  
警告メッセージを抑制するレベルを指定します。プルダウンリストで0~9のレベル(デフォルトは8)が選択可能で、選択した数値より大きなレベルの警告メッセージは表示されなくなります。
- Turn Off Overlaying** lk88の"-N"オプション  
このオプションを選択すると、重ね書きをオフします。
- Generate Link Map** lk88の"-M"オプション  
このオプションを選択すると、リンクマップファイルを生成します。
- Generate Call Graph File** lk88の"-c"オプション  
このオプションを選択すると、独立したコールグラフファイルを生成します。
- Suppress Undefined Symbol Diagnostics** lk88の"-r"オプション  
このオプションを選択すると、定義されていないシンボルの診断を抑制します。
- Print Name of Processing File (Verbose)** lk88の"-v"オプション  
このオプションを選択すると、リンク時に処理中のファイル名を表示します。

## Linking with user libraries

リンクするユーザライブラリがある場合は、ここにファイル名を入力します。複数のファイルを指定する場合はカンマ(,)で区切って入力してください。

## Other Options

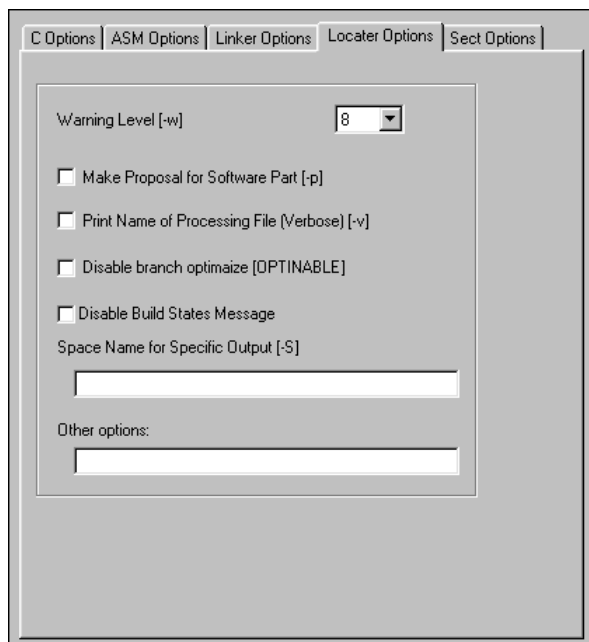
上記以外のオプションを指定したい場合(上記オプションも可)ここにコマンドラインの書式でオプションを入力しておくことができます。

## 表示されないオプションについて

オプションビューに表示されていないリンクカのオプションの扱いについては以下のとおりです。

- e 内部処理で使います。
- err リンカメッセージはメッセージウィンドウに表示されるとともに、エラーログファイルにも出力されます。
- f file 内部処理と競合するため使用不可。
- l x メモリモデル設定、システムライブラリ検索設定に連動し内部で自動処理されます。
- O file プロジェクト名に固定されます。
- o file プロジェクト名に固定されます。
- u symbol 指定する場合は[Other Options]に入力します。
- V wb88では使用しません。

## 3.9.4 ロケータオプション



この画面では、以下のオプションが選択できます。

## Warning Level

lc88の"-w n"オプション

警告メッセージを抑制するレベルを指定します。プルダウンリストで0~9のレベル(デフォルトは8)が選択可能で、選択した数値より大きなレベルの警告メッセージは表示されなくなります。

## Make Proposal for Software Part

lc88の"-p"オプション

このオプションを選択すると、ロケータ記述ファイルのソフトウェア部分のプロポーザルを表示します。

## Print Name of Processing File (Verbose)

lc88の"-v"オプション

このオプションを選択すると、処理中のファイル名を表示します。

### Disable branch optimize

lc88を使用する場合にこのオプションを選択します。チェックボックスがOFFの場合(デフォルト) 実行形式オブジェクトファイルの生成にはalc88が使用されます。

### Disable Build States Message

このチェックボックスがOFFの場合(デフォルト) ビルド、リビルド開始時に次のダイアログボックスが表示されます。



このダイアログボックスはalc88とlc88のどちらが使用されるか(ロケータオプションの[Disable branch optimize]チェックボックスの選択状態) およびDELFEED言語のロケータ記述ファイルがセクションエディタによって編集されるか(セクションエディタの[Disable Making DELFEE]チェックボックスの選択状態)を示します。

誤った選択がされている場合は、ダイアログボックスの[キャンセル]ボタンによってビルド(リビルド)を中止することができます。

このダイアログボックスの表示が不要な場合は、[Disable Build States Message]チェックボックスをONにしてください。

### Space Name for Specific Output

lc88の"-S space"オプション

ここにスペース名を入力すると、指定した空間に対応する特定の出力ファイルを生成します。

### Other options

上記以外のオプションを指定したい場合(上記オプションも可) ここにコマンドラインの書式でオプションを入力しておくことができます。

#### 表示されないオプションについて

オプションビューに表示されていないロケータのオプションの扱いについては以下のとおりです。

- d file 常にdscファイルが指定されます。
- e 内部処理で使用します。
- err ロケータメッセージはメッセージウィンドウに表示されるとともに、エラーログファイルにも出力されます。
- f file 内部処理と競合するため使用不可。
- f format 常にIEEE695標準(.abs)およびモトローラ(.s)ファイルが生成されます。
- M 常にロケータマップファイルを生成します。
- o file プロジェクト名に固定されます。
- V wb88では使用しません。

## 3.9.5 セクションエディタ

この画面でセクション、シンボル、外部メモリの配置を指定します。

wb88は、ここで指定された絶対アドレス情報を元にアドバンスドロケータ定義ファイルおよびDELFEEによるロケータ記述ファイルを生成し、ビルド実行時にアドバンスドロケータalc88またはロケータlc88の入力ファイルとして使用します。

Chip Mode

チップをMCUまたはMPUどちらのモードで使用するかをプルダウンリストから選択します。MCUモードは内蔵ROMを使用する場合に選択します。MPUモードは内蔵ROM領域を外部メモリに解放する(内蔵ROMを使用しない)場合に選択します。

Start Symbol

スタートシンボルを設定します。この設定内容は、ロケータ記述ファイル(.dsc)の"load\_mod start="パラメータとなります。

デフォルトは\_\_STARTで、cstart.cから始まるときはそのままかまいません。他のCルーチンから始まるときは、頭に"\_ "をつけた関数名を、他のアセンブラルーチンから始まるときは、そのシンボル名を設定してください。

例: 1. アセンブラルーチン

```
GLOBAL _main
_main:          から始まるとき      _main
```

2. Cルーチン

```
void main( )    から始まるとき      _main
```

Add Symbol( Rom )

ROMに配置するセクション、ベクタテーブル、ラベルの名称とアドレスを設定します。各行の設定項目は以下のとおりです。

Addr セクションまたはベクタテーブルの開始アドレス、あるいはラベルを割り付けるアドレスを入力します。

セクションを連続的に配置する場合は先頭セクションの開始アドレスのみが必要で、2番目以降のセクションのアドレスは空白のままにしておきます。ただし、コンパイラが生成した種類の異なるセクションを連続的に配置する場合は、セクションごとにアドレスを指定する必要があります(詳細は後述)。

Name セクション、ベクタテーブル、ラベルの名称(シンボル名)を入力します。

Kind 配置する項目の種類をプルダウンリストから選択します。

Vect ベクタテーブル

Label ラベル

Sect セクション

#### Add Symbol( Ram )

RAMに配置するセクション、ラベルの名称とアドレスを設定します。

各行の設定項目は以下のとおりです。

Addr セクションの開始アドレス、あるいはラベルを割り付けるアドレスを入力します。  
セクションを連続的に配置する場合は先頭セクションの開始アドレスのみが必要で、2番目以降のセクションのアドレスは空白のままにしておきます。ただし、コンパイラが生成した種類の異なるセクションを連続的に配置する場合は、セクションごとにアドレスを指定する必要があります(詳細は後述)。

Name セクションまたはラベルの名称(シンボル名)を入力します。

Kind 配置する項目の種類をプルダウンリストから選択します。

Label ラベル

Sect セクション

#### Add External Memory

外部バスに接続するメモリやデバイスのアドレスとサイズを設定します。

各行の設定項目は以下のとおりです。

Addr 外部メモリ/デバイスの開始アドレスを入力します。

Mem 外部メモリの種類をプルダウンリストから選択します。

Rom ROM

Ram RAM

Dev メモリマップされる各種デバイス(LCDコントローラなど)

Size 外部メモリの容量またはデバイスのマップサイズをバイト数で入力します。

#### Disable Making DELFEE

セクションエディタにより、DELFE言語のロケータ記述ファイルを生成するかしないか選択します。

チェックボックスがOFFの場合(デフォルト)

セクションエディタは、この画面で設定した内容からalc88用のアドバンスドロケータ定義ファイルおよびlc88用のロケータ記述ファイルを生成します。

チェックボックスがONの場合

セクションエディタは、lc88用のロケータ記述ファイルを生成しません。ユーザが作成した既存のロケータ記述ファイルを使用する場合に、このチェックを外します。この場合でも、alc88用のアドバンスドロケータ定義ファイルはこの画面の設定内容に従って生成されます。

#### Heap Size

malloc()などで確保するヒープエリアのサイズを指定します。ただし、この設定は実際にmalloc()などを使用してヒープエリアが必要になったときのみ有効となります。

### アドバンスドロケータalc88を使用する場合

alc88を使用する場合は、ロケータオプションとセクションエディタで以下の設定を行ってください。

1. [Locater Options]タブの画面で[Disable branch optimize]チェックボックスをOFFにします。  
以下の設定はセクションエディタで行います。
2. [Disable Making DELFEE]チェックボックスをOFFにします。
3. [Chip Mode]のリストから使用するモード( MCUまたはMPUモード )を選択します。
4. [Add Symbol (Rom)]と[Add Symbol (Ram)]ボックスにセクションなどの配置アドレスを入力します。  
(入力方法は後述)
5. 外部メモリ/デバイスを使用する場合は、[Add External Memory]ボックスに、その情報を入力します。  
(入力方法は後述)

ロケータオプションの[Disable branch optimize]をOFFしたことで、ビルド時はalc88が起動します。

### ロケータlc88を使用する場合1( セクションエディタで生成するロケータ記述ファイルを使用 )

既存のロケータ記述ファイルを使用する必要がない場合は、alc88の使用を推奨します。lc88を使用する必要がある場合は、以下の設定を行ってください。

1. [Disable Making DELFEE]チェックボックスをOFFにします。  
このチェックボックスがONのまま変わらない場合は、[Locater Options]タブの画面で[Disable branch optimize]チェックボックスをOFFにした後で、この操作を行ってください。
2. [Chip Mode]のリストから使用するモード( MCUまたはMPUモード )を選択します。
3. 必要に応じ、[Start Symbol]にスタートシンボル名を入力します。(通常は\_\_STARTのまま)
4. [Add Symbol (Rom)]と[Add Symbol (Ram)]ボックスにセクションなどの配置アドレスを入力します。  
(入力方法は後述)
5. 外部メモリ/デバイスを使用する場合は、[Add External Memory]ボックスに、その情報を入力します。  
(入力方法は後述)
6. [Locater Options]タブの画面で[Disable branch optimize]チェックボックスをONにします。

ロケータオプションの[Disable branch optimize]をONしたことで、ビルド時はlc88が起動します。

### ロケータlc88を使用する場合2( 既存のロケータ記述ファイルを使用 )

アプリケーションの改定など、既存のロケータ記述ファイルを使用する場合は、以下の設定を行ってください。

1. [Disable Making DELFEE]チェックボックスをONにします。  
この操作により、ロケータオプションの[Disable branch optimize]チェックボックスが自動的にONになります。
2. プロジェクトビューに[Definition Files]フォルダ内のファイル一覧が表示されますので、必要に応じてロケータ記述ファイルを修正します。

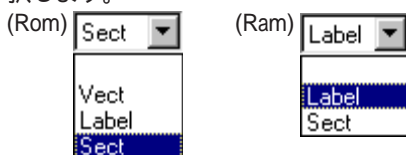
ロケータオプションの[Disable branch optimize]がONとなったことで、ビルド時はlc88が起動します。

注: 既存のロケータ記述ファイルを使用する場合、セクションエディタ上での配置アドレス等の入力は不要です。ただし、その状態でも内容が不完全なアドバンスドロケータ定義ファイルが作成されます(ロケータ記述ファイルの内容は反映されません)。alc88による処理に変更する場合には正しいアドバンスドロケータ定義ファイルを作成し直してください。

### [Add Symbol (Rom/Ram)] - シンボルの定義と削除

[Add Symbol (Rom/Ram)]へのシンボル定義は以下の手順で行います。

1. 空白行の[Addr]のセルをクリックし、アドレスを入力します。
2. [Name]にシンボルを入力します。
3. [Kind]のセルをクリックすると次のプルダウンリストが表示されますので、配置する項目の種類を選択します。



4. 3つの項目が埋まった状態で[Enter]キーが押されると、その下に空白の行が追加されます。
5. 上記を繰り返して、必要な割り付けをすべて行います。

同種のセクションを、アドレスを連続させて配置する場合、[Addr]は先頭のセクションのみ指定することで、2番目以降は省略できます。[Name]と[Kind]は省略できません。直前に設定したセクションと種類が異なる場合、[Addr]を入力しないとその行の設定は無効で、次の行に移ることもできません。特にコンパイラが生成するセクションは種類の違いに注意する必要があります。

アドレスは降順/昇順に入力する必要はありません。

入力/選択内容による定義ファイルの更新はビルド(リビルド)開始時、プロジェクトを保存またはwb88を終了した時点で行われます。

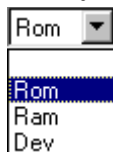
[Add Symbol (Rom/Ram)]に設定済みのアドレスを削除するには、

1. 削除するアドレスの行の[Addr]、[Name]、[Kind]の内容をすべて削除([Back Space]キーまたは[Delete]キーを使用、[Kind]は空白を選択)します。
2. 3つの項目が空白になった状態で[Enter]キーを押します。  
その行が削除され、続く行が繰り上がります。

#### [Add External Memory] - 外部メモリの定義と削除

外部バスにROMやRAM、あるいはLCDコントローラなどの外部デバイスを接続するシステムでは、[Add External Memory]にアドレス割り当てやデバイスのサイズを設定します。

1. 空白行の[Addr]のセルをクリックし、アドレスを入力します。
2. [Mem]のセルをクリックすると次のプルダウンリストが表示されますので、外部メモリの種類を選択します。



3. [Size]に外部メモリのサイズを入力します。
4. 3つの項目が埋まった状態で[Enter]キーが押されると、その下に空白の行が追加されます。
5. 上記を繰り返して、必要な定義をすべて行います。

アドレスは降順/昇順に入力する必要はありません。

入力/選択内容による定義ファイルの更新はビルド(リビルド)開始時、プロジェクトを保存またはwb88を終了した時点で行われます。

[Add External Memory]に設定済みの外部メモリ定義を削除するには、

1. 削除する行の[Addr]、[Mem]、[Size]の内容をすべて削除([Back Space]キーまたは[Delete]キーを使用、[Mem]は空白を選択)します。
2. 3つの項目が空白になった状態で[Enter]キーを押します。  
その行が削除され、続く行が繰り上がります。



## 注意事項

### 1. 入力内容の制限

最大入力行数と文字数は以下のとおりです。

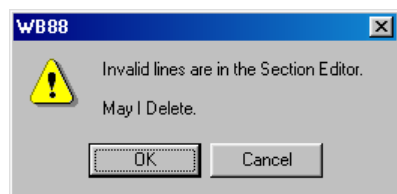
最大入力行数	[Add Symbol (Rom/Ram)]	.....	100行
	[Add External Memory]	.....	20行
最大入力文字数	[Addr]	.....	8桁
	[Name]	.....	30文字
	[Size]	.....	8桁

### 2. 入力データのチェック

wb88はビルド(リビルド)開始時、プロジェクトの保存または終了時にセクションエディタの必要項目の入力もれがないかチェックします。

問題がない場合は処理を継続または終了します。

シンボルや外部メモリ定義で、3項目の入力が必要な部分に2項目しか入力されていないといった不具合が見つかった場合は、次のダイアログボックスを表示します。



[OK]をクリックすると不正な行は削除され、処理を継続または終了します。

[キャンセル]をクリックした場合は、ビルド(リビルド)プロジェクトの保存または終了の処理を中止します。

wb88は、入力したアドレス値が実装メモリの領域内かどうか、シンボル名が重複していないかなど、入力内容のチェックは行いません。これらの不具合は、alc88またはlc88によってチェックされます。

### 3. コンパイラが生成するセクションについて

[Add Symbol (Rom/Ram)]にユーザ定義のセクションを連続的に指定する場合は、先頭セクションのアドレスのみを指定するだけで、以降のセクションのアドレス指定は省略できます。ここにはコンパイラが生成したセクションの配置も指定できます。ただし、コンパイラが生成するセクションには種別があり、セクションを連続的に配置する場合でも、種別が変わった場合はそのセクションで新たにアドレスの指定が必要となります。

以下にコンパイラが生成するセクションの種別を示します。

#### ROMエリア

```
code_short
    .comm
    .startup

code
    .text
    .text_xxxxxxxx
    table ..... アドレスは指定できません。

data_short
    .nrdata

data
    .frdata
```

#### RAMエリア

```
data_short
    .nbss
    .ndata
    .nbssnc

data
    .fdata
    .fbss
    .fbssnc
    stack ..... アドレスは指定できません。
    xvwbuffer ..... アドレスは指定できません。
```



## 4. ベクタ/ラベルについて

lc88の機能に合わせ、[Add Symbol (Rom/Ram)]にはベクタ/ラベルが定義できます。ユーザはプログラムの中で、\_\_lc\_u\_xxxxxという名称の外部 (extern) ベクタ/ラベルをアクセスすることができ、そのアドレスをセクションエディタで定義できます。

[Add Symbol (Rom/Ram)]に定義するときは、xxxxxの名前の部分のみを入力してください。

## 5. [Disable Making DELFEE]チェックボックスをONからOFFに戻す操作

このチェックボックスをONにすると、ロケータオプションの[Disable branch optimize]チェックボックスが自動的にONとなり、その状態ではこのチェックボックスをOFFに戻すことができなくなります。ONからOFFに戻すには、先にロケータオプションの[Disable branch optimize]チェックボックスをOFFにしてください。

## 6. 特殊セクションについて

以下の4種類のセクションはセクションエディタで指定できません。指定した場合は、セーブ/ビルド時に削除されます。

"heap"、"stack"、"table"、"xvwbuffer"

### 3.10 デバッグ

ワークベンチからシミュレータまたはインサーキットエミュレータを起動してデバッグを行うことができます。

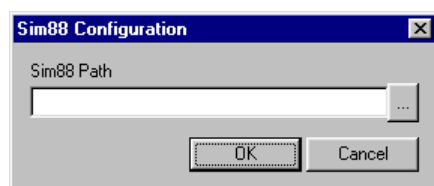
#### 3.10.1 シミュレータ

ここでは、ワークベンチからシミュレータsim88を起動する方法を説明します。シミュレータの機能や操作方法についてはシミュレータのマニュアルを参照してください。

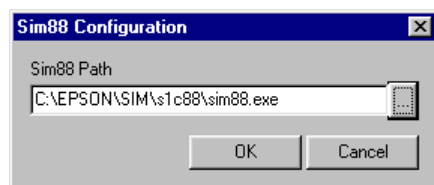
##### シミュレータのパス設定

シミュレータsim88を起動するには、パスを設定しておく必要があります。

パスを設定するには[Tool]メニューから[Sim88 Configuration]を選択して、次のダイアログボックスを表示させます。



[...]ボタンで表示されるファイル選択ダイアログボックスでsim88.exeを選択するか、テキストボックスにパスを直接入力してください。

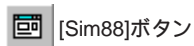


一度設定しておく、次回以降再設定する必要はありません。

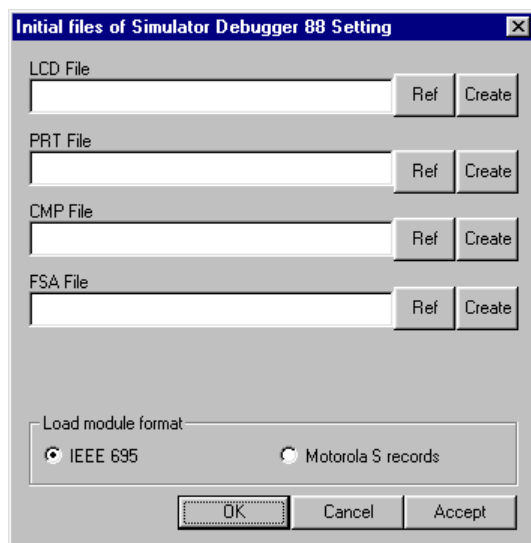
##### シミュレータの起動

シミュレータを起動するには、

1. [Debug]メニューから[Sim88 Simulator]を選択するか、[Sim88]ボタンをクリックします。



次のダイアログボックスが表示されます。



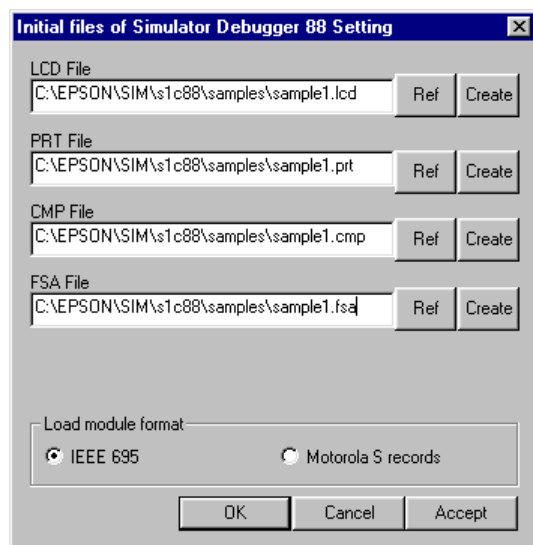
2. シミュレータの起動に必要な以下のファイルを指定します。各ファイルを、それぞれの[Ref]ボタンで表示されるファイル選択ダイアログボックスで選択するか、テキストボックスにパスを直接入力してください。

LCD File: LCDパネル定義ファイル

PRT File: ポート設定ファイル

CMP File: コンポーネントマッピングファイル

FSA File: ファンクションオプションHEXファイル



LCDパネル定義ファイル、ポート設定ファイル、コンポーネントマッピングファイルの詳細については、シミュレータのマニュアルを参照してください。

[Create]ボタンは各ファイルの作成ツールを起動します。

LCD File: LCDパネルカスタマイズユーティリティ LcdUtil

PRT File: ポート設定ユーティリティ PrtUtil

CMP File: エディタ ([Tool] Editor Configuration)で指定)

FSA File: ファンクションオプションジェネレータ winfog (8章参照)

LCDパネルカスタマイズユーティリティ、ポート設定ユーティリティについては、シミュレータのマニュアルを参照してください。

これらのツールは[Tool]メニューやボタンでも起動できます。

3. [Load module format]のラジオボタンで、シミュレータにロードするオブジェクトファイルの形式 (IEEE695またはモトローラS)を選択します。

4. [OK]ボタンをクリックすると、ダイアログボックスが閉じてシミュレータを起動します。  
ワークベンチは、入力されたファイル情報からシミュレータプロジェクトファイル(.spj)と必要なファイルをロードするためのコマンドファイルを生成し、シミュレータに渡します。したがって、シミュレータが起動した時点ですぐにデバッグを開始できる状態になります。

[Accept]ボタンは上記のファイルを生成するのみでダイアログボックスは閉じません。したがって、シミュレータも起動しません。

### 3.10.2 インサーキットエミュレータ(S5U1C88000H5)とデバッガ

ここでは、ワークベンチからS5U1C88000H5を使用するデバッグシステムを起動する方法を説明します。db88デバッガについては本書の13章を、ICEとice88urデバッガの機能や操作方法についてはS5U1C88000H5のマニュアルを参照してください。

S5U1C88000H5システムを起動するには、

1. ICEがパーソナルコンピュータに接続され、電源がONしていることを確認してください。
2. ワークベンチを起動します。
3. db88デバッガを起動する場合は、[Debug]メニューから[DB88 Debugger]を選択するか、[DB88]ボタンをクリックします。



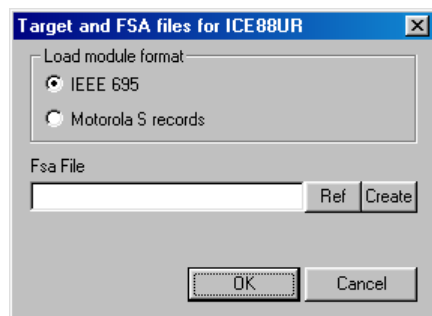
[DB88]ボタン

ice88urデバッガを起動する場合は、[Debug]メニューから[ICE88UR Debugger]を選択するか、[ICE88UR]ボタンをクリックします。



[ICE88UR]ボタン

次のダイアログボックスが表示されます。










4. [Load module format]のラジオボタンでアプソリュートオブジェクトのファイル形式(IEEE695またはモトローラS)を選択します。
5. [Fsa File]でファンクションオプションHEXファイルを指定します。[Ref]ボタンで表示されるファイル選択ダイアログボックスで選択するか、テキストボックスにパスを直接入力してください。[Create]ボタンは、ファンクションオプションHEXファイルを生成するファンクションオプションジェネレータwinfogを起動します。
6. [OK]ボタンをクリックすると、ダイアログボックスが閉じてデバッガを起動します。ワークベンチは、入力された情報から必要なファイルをロードするためのコマンドファイルを生成し、デバッガに渡します。したがって、デバッガが起動した時点ですぐにデバッグを開始できる状態になります。

### 3.11 その他のツールの実行

[Tool]メニューまたはツールバーボタンで、以下のツールを起動することができます。

表3.11.1 wb88から起動可能なツール

ツール	メニューコマンド	ボタン
1. 自動評価システム	[Tool   Simulator Tools   Auto Evaluation System]	
2. ビットマップユーティリティ	[Tool   Simulator Tools   Bitmap Utility]	
3. LCDパネルカスタマイズユーティリティ	[Tool   Simulator Tools   LCD Panel Customize Utility]	
4. ポート設定ユーティリティ	[Tool   Simulator Tools   Port Setting Utility]	
5. ファンクションオプションジェネレータ	[Tool   Dev Tools   Function Option Generator]	
6. マスクデータチェッカ	[Tool   Dev Tools   Mask Data Checker]	
7. オンボードROMライター コントロールソフトウェア	[Tool   On-Board ROM Writer]	

各ツールの操作方法については、1～4はシミュレータのマニュアルを、5～6は本書の各ツールの章を、7はFlash EEPROM内蔵マイコンのテクニカルマニュアルを参照してください。

## 3.12 ファイル一覧

ワークベンチが扱うファイルの種類と格納位置を以下に示します。

表3.12.1 ファイル一覧

ファイル種別	ファイル名	拡張子	作成者/ツール	フォルダパス(Default)
<b>Cコンパイラ関連</b>				
Cソースファイル	任意	.c	ユーザ/テキストエディタ	任意(<project>%src)
Cヘッダファイル	任意	.h	ユーザ/テキストエディタ	任意(<project>%src)
Cスタートアップルーチン	cstartup	.c	wb88	<project>%def%
アセンブリソース(ユーザ作成)	任意	.asm	ユーザ/テキストエディタ	任意(<project>%src)
アセンブリヘッダファイル	任意	.inc	ユーザ/テキストエディタ	任意(<project>%src)
ビットマップファイル	任意	.bmp	ユーザ/ビットマップエディタ	任意
ビットマップ定義ファイル	任意	.bmu	ユーザ/BmpUtil	任意
データテーブル	任意	.txt	ユーザ/BmpUtil	任意
プロジェクト管理ファイル	プロジェクト名	.wpj	wb88	<project>%
Makeファイル	makefile	-	wb88	<project>%tmp%
エラーログファイル	プロジェクト名	.err	wb88/cc88	<project>%tmp%
中間アセンブリソースファイル	[ソース名引用]	.src	wb88/c88	<project>%obj%
アセンブリリストファイル	[ソース名引用]	.lst	wb88/as88	<project>%obj%
オブジェクトファイル	[ソース名引用]	.obj	wb88/as88	<project>%obj%
オブジェクトライブラリファイル	任意	.a	ユーザ/ar88	任意
リンクオブジェクトファイル	プロジェクト名	.out	wb88/lk88	<project>%obj%
リンクマップファイル	プロジェクト名	.lnl	wb88/lk88	<project>%obj%
コールグラフファイル	プロジェクト名	.cal	wb88/lk88	<project>%obj%
アドバンスドロケータ定義ファイル	機種名	.inf	wb88	EPSON%\$1C88%Dev%
ロケータ定義ファイル	機種名	.dsc	ユーザ/テキストエディタ	<project>%def%
CPU定義ファイル	機種名	.cpu	ユーザ/テキストエディタ	<project>%def%
メモリ定義ファイル	機種名	.mem	ユーザ/テキストエディタ	<project>%def%
ロケータマップファイル	プロジェクト名	.map	wb88/lc88	<project>%obj%
アブソリュートロードモジュール	プロジェクト名	.abs	wb88/lc88	<project>%obj%
モトローラSモジュール	プロジェクト名	.sa	wb88/lc88	<project>%obj%
シンボリックテーブルファイル	プロジェクト名	.sy	wb88/sy88	<project>%obj%
プログラムデータHEXファイル	プロジェクト名	.psa	wb88/fil88xxx	<project>%obj%
<b>Development Tool関連</b>				
機種情報定義ファイル	機種名	.ini	セイコーエプソン	EPSON%\$1C88%Dev%
ファンクションオプションHEXファイル	任意	.fsa	ユーザ/WinFOG	任意
ファンクションオプションドキュメントファイル	任意	.fdc	ユーザ/WinFOG	任意
マスクデータファイル	任意	.paN	ユーザ/WinMDC	任意
<b>自動評価システム関連</b>				
コマンドファイル	任意	.txt	ユーザ	任意
リファレンスデータファイル	任意	.mXX	ユーザ	任意
結果データファイル	任意	.aXX	ユーザ	任意
チェックシートファイル	任意	.csv	ユーザ/AutoEva	任意
<b>シミュレータ関連</b>				
LCDパネル定義ファイル	任意	.ldc	ユーザ/LCDUtil	任意
ポート設定ファイル	任意	.prt	ユーザ/PrtUtil	任意
シミュレータプロジェクトファイル	sim88	.spj	wb88	<project>%tmp%
コマンドファイル	debug	.cmd	wb88	<project>%tmp%
コンポーネントマップファイル	任意	.cmp	ユーザ/テキストエディタ	任意
<b>ICE関連</b>				
ICEパラメータファイル	機種名	.par	ユーザ/テキストエディタ	<project>%def%
周辺回路ボード用FPGAデータファイル	機種名	.mot	セイコーエプソン	
ICE用INIファイル	ice88ur	.ini	wb88	<project>%tmp%

### 3.13 エラーメッセージ

ワークベンチのエラーメッセージを以下に示します。

表3.13.1 システムエラーメッセージ

メッセージ	説 明
not enough memory	wb88 を実行するために十分なメモリがありません。

表3.13.2 プロジェクト生成時のエラーメッセージ

メッセージ	説 明
The file is not a WB88 project file.( <b>&lt;filename&gt;</b> )	<b>&lt;filename&gt;</b> はwb88のプロジェクトファイルではありません。
The version of the project file is not supported.( <b>&lt;filename&gt;</b> )	そのプロジェクトファイル <b>&lt;filename&gt;</b> のバージョンは、サポートしていません。
Unable to create a project : cannot access. <b>&lt;filename&gt;</b>	<b>&lt;filename&gt;</b> に正しくアクセスできなかったため、プロジェクトを生成できません。
Unable to create a project : Unable to copy DEF file.( <b>&lt;filename&gt;</b> )	定義ファイル <b>&lt;filename&gt;</b> のコピーに失敗したため、プロジェクトを生成できません。
The project is already existed.( <b>&lt;filename&gt;</b> )	<b>&lt;filename&gt;</b> は既に存在するため、プロジェクトを作成できません。同じフォルダに同じ名前のプロジェクトを二つ以上作成することはできません。
Unable to create a project : Dev Directory of S1C88 family package does not exist.	DEVディレクトリが存在しないため、プロジェクトを作成できません。 パッケージのDEVディレクトリにはビルドに必要な各種定義ファイルが納められているため、このディレクトリがないと、プロジェクトを構築できません。

表3.13.3 プロジェクトへのファイル追加時のエラーメッセージ

メッセージ	説 明
The file cannot be added to the project. It is not a C file.( <b>&lt;filename&gt;</b> )	Cソースファイルでないため、 <b>&lt;filename&gt;</b> をプロジェクトに追加できません。
The file cannot be added to the project. It is not an ASM file.( <b>&lt;filename&gt;</b> )	アセンブリソースファイルでないため、 <b>&lt;filename&gt;</b> をプロジェクトに追加できません。
The file cannot be added to the project. It is not a header file.( <b>&lt;filename&gt;</b> )	ヘッダファイルでないため、 <b>&lt;filename&gt;</b> をプロジェクトに追加できません。
The file is already existed in the project. It cannot be added in the project.( <b>&lt;filename&gt;</b> )	<b>&lt;filename&gt;</b> は既にプロジェクトに存在するため、追加できません。
WB88 does not support such source file type.( <b>&lt;filename&gt;</b> )	wb88がサポートしていないソースファイルです。

表3.13.4 ファイルエラーメッセージ

メッセージ	説 明
Failed to access the file.( <b>&lt;filename&gt;</b> )	<b>&lt;filename&gt;</b> の操作に失敗しました。
Unable to open the file.( <b>&lt;filename&gt;</b> )	<b>&lt;filename&gt;</b> のオープンに失敗しました。

表3.13.5 ツール起動時のエラーメッセージ

メッセージ	説 明
Unable to execute ICE88UR.exe : Unable to access <b>&lt;filename&gt;</b> .	<b>&lt;filename&gt;</b> のアクセスに失敗したため、S5U1C88000H5を起動できません。
Unable to execute Sim88 : Unable to access the DEF file.( <b>&lt;filename&gt;</b> )	定義ファイルのアクセスに失敗したため、Sim88を起動できません。
Unable to execute <b>&lt;toolname&gt;</b> .	<b>&lt;toolname&gt;</b> の起動に失敗しました。

表3.13.6 ビルド時のエラーメッセージ

メッセージ	説 明
Select a C or an ASM file.	Cソースもしくはアセンブリソースファイルを選択してください。コンパイルするときは、対象ファイルをツリービューから選ぶ必要があります。
Build Command needs an active project.	ビルドするには、プロジェクトが必要です。
No target file is found in the project.	ビルドターゲットファイルがプロジェクト内にありません。ビルドするには、ソースファイルをプロジェクトに登録する必要があります。

表3.13.7 その他のエラーメッセージ

メッセージ	説 明
The command needs an active project.	そのコマンドには、プロジェクトが必要です。プロジェクトが作成されていない場合に、その存在が必須な機能を実行すると表示されます。

## 4 メインツールチェーンの概要

メインツールチェーンはCコンパイラを中心とした、以下のツールで構成されています。

### 1. Cコンパイラ <c88.exe>

Cソースファイルをコンパイルしてas88で処理可能なアセンブリソースファイルを生成します。c88はANSI C準拠のCコンパイラです。特殊な文法はサポートしていませんので、他機種用のプログラムの移植なども容易に行えます。また、S1C88アーキテクチャをCレベルで効率的に使用でき、コンパクトなコードを生成しますので、組み込み用アプリケーションの開発に最適です。プリプロセッサ、S1C88 Cフロントエンド、コードジェネレータが単一のプログラムに統合され、中間ファイルが不要なワンパスコンパイラとして高速に動作します。

### 2. アセンブラ <as88.exe>

c88が出力するアセンブリソースファイルをアセンブルし、ソースファイルのニーモニックをS1C88のオブジェクト(機械語)コードに変換します。結果はlk88でリンク可能なIEEE-695形式のリロケータブルオブジェクトファイルとして出力されます。

### 3. リンカ <lk88.exe>

as88が生成した複数のリロケータブルオブジェクトファイルとライブラリモジュールを結合して、1つの新しいリロケータブルオブジェクトファイルを生成します。

### 4. ロケータ <lc88.exe>

lk88が作成したリロケータブルオブジェクトを絶対アドレスに再配置し、実行可能なロードイメージファイルを生成します。再配置情報は、ロケータが読み込むロケータ記述ファイルにDELFEY言語で記述しておきます。

lc88は既存のロケータ記述ファイルを使用して開発を行うために残されています。新規開発の場合は、lc88の機能に加え分岐最適化機能を持つアドバンスドロケータalc88がS5U1C88000C Ver.3から追加されましたので、そちらの使用を推奨します。lc88とalc88のどちらを使用するかについては、wb88で選択します。

### 5. アドバンスドロケータ <alc88.exe>

lc88が持つ再配置機能をDELFEYによる記述ファイルを使用せずに実現します。また、64Kバイト以上のコード領域を持つメモリモデルの場合、コール命令(CALL)の直前にバンク指定用の拡張命令(LD NB, xxxx)がアセンブラによって付加されますが、バンク内コールに付加された不要な拡張命令を削除する機能がalc88には追加されています。

1~4のツールの詳細については"S5U1C88000C Manual I"を参照してください。5のalc88は本書で説明します。なお、上記のツールはすべてwb88のビルド機能により実行されるため、ツールを個々に操作する必要はありません。



## 5 アドバンスドロケータ<alc88>

### 5.1 alc88の機能

アドバンスドロケータ<alc88>は、リンカ<lk88>が作成したリロケータブルオブジェクトを絶対アドレスに再配置し、実行可能なロードイメージファイルを生成します。これに加え、分岐先最適化機能も持っています。これは、64Kバイト以上のコード領域を持つメモリモデル(Compact-DataまたはLarge)の場合にアセンブラによってコール命令(CARL)の直前に無条件に付加されるバンク指定用の拡張命令(LD NB,xxxx)を、バンク内コールの場合は削除する機能です。

これにより、メインツールチェーンで従来より使用していたロケータ<lc88>よりもコンパクトな実行形式のオブジェクトファイルが生成されます。

また、lc88に再配置情報を提供していた、DELFEY言語によるロケータ記述ファイルはalc88では必要ありません。代わりにアドバンスドロケータ定義ファイル(.inf)を使用しますが、wb88のセクションエディタ機能で簡単に、特に意識せずに生成できます。

したがって、ロケータ記述ファイルも含め、従来の資産を使用してアプリケーションを開発する場合はlc88を、新規の開発または既存のロケータ記述ファイルが特に必要ない場合はalc88を使用してください。どちらのツールを使用するかを選択はwb88で行えます。

注: 分岐最適化は同一バンク(32Kバイト領域)内に分岐している次の形式のCARL命令のみが対象です。ジャンプ命令等、他の分岐命令の前に付加されている不要な拡張命令は削除されません。また、オブジェクトを入力した時点ですでにアドレスが確定している拡張命令と分岐命令については、最適化の対象であっても拡張命令を削除しません。

```
LD    NB,xxxx
CARL  yyyy
```

yyyyがCARL命令と同一バンク内にある場合: 直前の"LD NB,xxxx"は削除されます。

yyyyがCARL命令とは異なるバンクにある場合: 直前の"LD NB,xxxx"は削除されません。

## 5.2 入出力ファイル

図5.2.1にalc88の入出力ファイルを示します。

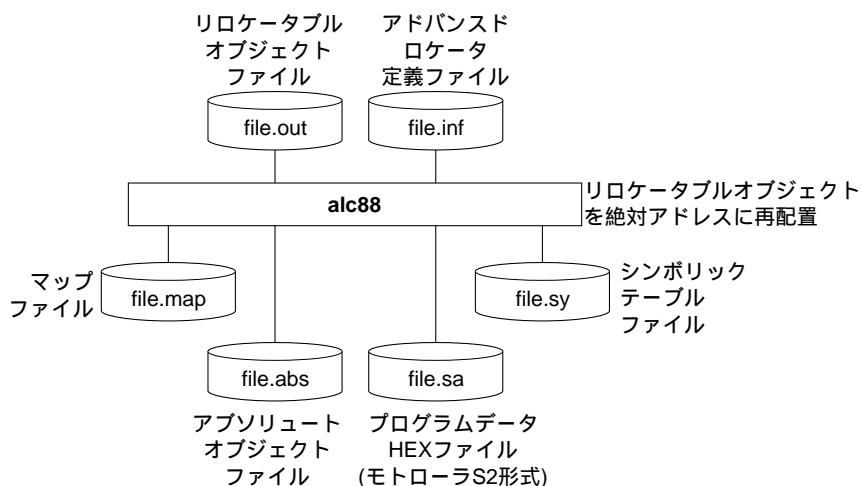


図5.2.1 alc88の入出力ファイル

リロケータブルオブジェクトファイル( file.out )

リンカ<lk88>が出力したIEEE-695形式のリロケータブルオブジェクトファイルです。

アドバンスドロケータ定義ファイル( file.inf )

リロケータブルオブジェクトを絶対アドレスに再配置するための情報が記述されたファイルです。wb88のセクションエディタを使用して作成します。

絶対オブジェクトファイル( file.abs )

alc88に入力したリロケータブルオブジェクトを絶対アドレスに再配置して出力される実行可能なオブジェクトファイルです。IEEE-695形式で、入力ファイル内のデバッグ情報も含まれます。

プログラムデータHEXファイル( file.sa )

絶対オブジェクトをモトローラS2形式に変換して出力されるHEXファイルです。内蔵ROM未使用領域FF詰めユーティリティ<fil88xxx>の入力ファイルとなります。

マップファイル( file.map )

セクションやラベルなどが配置された絶対アドレスの一覧を記録したファイルです。

シンボリックテーブルファイル( file.sy )


入力ファイルのデバッグ情報から抽出したシンボル情報が出力されます。このファイルはデバッガやシミュレータでシンボリックデバッグを行うのに必要です。

### 5.3 操作方法

アドバンスドロケータ定義ファイルの作成も含め、通常はすべての操作をwb88で行います。alc88はwb88によるビルド処理の中で自動的に実行されますので、ユーザがalc88を起動させる必要はありません。アドバンスドロケータ定義ファイルは、wb88のセクションエディタを使用して作成します。ビルドやセクションエディタの操作方法については、"3 ワークベンチ"を参照してください。

alc88を単独に実行するには、MS-DOSプロンプトから次のコマンドを実行します。

**>alc88 <project\_path> <file.out> <file.inf>** 

 リターンキーの入力を表します。  
 <project\_path> プロジェクトファイル(.wpj)へのパスを指定します。  
 <file.out> 入力するオブジェクトファイル名を指定します。  
 <file.inf> 入力するアドバンスドロケータ定義ファイルを指定します。

例) C:\¥epson¥s1c88¥app1 app1.out app1.inf

alc88は処理が終了すると、正常に終了したかどうかにかかわらず、次のメッセージを表示(stdoutに出力)します。

ALC88 Version x.xx

### 5.4 エラーメッセージ

以下に、alc88のエラーメッセージを示します。

表5.4.1 エラーメッセージ

メッセージ	説明
Illegal Inf File	アドバンスドロケータ定義ファイル(.inf)が不正です。
Duplicate Memory -- 0xn timer ~ 0xn timer & 0xn timer ~ 0xn timer	0xn timer ~ 0xn timerと0xn timer ~ 0xn timerでメモリ割り当てが重複しています。
No physical memory available for xxxx	シンボルxxxxを割り当てるべき、指定されたアドレスが存在しません。
Duplicate Symbol Name -- xxxx	シンボル名xxxxが重複しています。
Cannot find 0xn timer bytes for xxxx section	セクションxxxxの割り当てに必要な0xn timerバイトの領域がありません。
Found unresolved external -- xxxx	外部シンボル(Extern)xxxxの情報がありません。
There is no stack area	内蔵RAMエリアが足りないため、スタックエリアが確保できませんでした。
Absolute address 0xn timer occupied	0xn timerで始まる絶対アドレスセクションのエリアが、既に他のエリアに占有されています。
Value out of range to label xx at address 0xnn	ラベルxxはショート分岐命令(JRS、CARS)の分岐可能範囲(-128 ~ 127)を越えています。

### 5.5 注意事項

alc88には以下の制約がありますので注意してください。

- (1) lc88で対応しているラベルのうち、alc88が対応しているのは\_\_lc\_cp、\_\_lc\_es、\_\_lc\_u\_xxxx、\_\_lc\_b\_xxxx、\_\_lc\_e\_xxxxとなります。一方、\_\_lc\_bs、\_\_lc\_ub\_xxx、\_\_lc\_ue\_xxxをソース内で使用しても、alc88は対応していません。"S5U1C88000C Manual I"の"4.9 ロケータのラベル"を参照してください。
- (2) alc88によって分岐の最適化が行われても、as88が作成するリストファイルには、リローケータブル/アプソリュートに関わらず、その結果は反映されません。

## 6 デベロップメントツールの概要

S1C88 Family統合ツールパッケージには以下に示す、マスクオプションファイルやマスクデータファイルの作成を行うツール、機種ごとの設定情報を記述したファイルが含まれています。

1～3に示す各ツールは、Windows 2000またはWindows XPに対応したWindows GUIアプリケーションです。

### 1. ファンクションオプションジェネレータ <winfog.exe>

winfogはS1C88xxxのマスクオプションを選択し、ICE(S5U1C88000H5)のファンクションオプション設定ファイルとICマスクパターンを生成するためのファンクションオプションドキュメントファイルを作成するツールです。ウィンドウに表示された選択項目をチェックボックスで選択するだけでファンクションオプションデータを作成することができます。

### 2. セグメントオプションジェネレータ <winsog.exe>

winsogはS1C88xxxのLCDセグメントオプションを設定し、ICEのセグメントオプション設定ファイルとICマスクパターンを生成するためのセグメントオプションドキュメントファイルを作成するツールです。ウィンドウに表示された表示メモリマップとセグメントデコードテーブルをマウスでクリックするだけで、セグメント割り付けデータを作成することができます。

### 3. マスクデータチェッカ <winmdc.exe>

winmdcは開発が終了した内蔵ROMファイル、オプションドキュメントファイルのデータをチェックし、セイコーエプソンへ提出するためのマスクデータファイルを作成するツールです。

### 4. 機種情報定義ファイル <s1c88xxx.ini>

上記の3ツールにオプションの構成など、各機種の情報を設定するファイルです。各ツールを実行するためには必須のファイルです。

### 5. ICE用パラメータファイル <t88xxx.par>

ICEを各機種に対応させるためのパラメータファイルで、ICEを起動するのに必要です。

### 6. 内蔵ROM未使用領域FF詰めユーティリティ <fil88xxx.exe>

プログラムデータHEXファイルから内蔵ROM領域を切り出し、その未使用領域にFFHを埋め込みます。また、システム予約領域にシステムコードを設定します。この作業は、ICEでプログラムをデバッグする前、およびwinmdcでマスクデータを作成する前に行う必要があります。本ユーティリティはMS-DOSプロンプトから実行します。

### 7. 自己診断プログラム <t88xxx.psa, t88xxx.fsa, t88xxx.fdc, t88xxx.ssa, t88xxx.sdc, readme.txt>

ICEおよびS5U1C88xxxPのハードウェアを診断するための、自己診断プログラムとファンクションオプションデータです。ICEにダウンロードして自己診断を行います。t88xxx.ssaとt88xxx.sdcはセグメントオプションが設定された機種のパッケージにのみ含まれます。readme.txtには、自己診断プログラムによる動作確認のためのS5U1C88xxxPのLED点灯状態が記述されています。

注: • 本マニュアルは全機種に共通です。そのため、機種名をS1C88xxxとして説明を行っています。掲載されている画面サンプルの内容は機種により異なります。また、winsog、自己診断プログラムのt88xxx.ssaとt88xxx.sdcはセグメントオプションの設定されている機種にのみ適用されます。

• ICEにはS5U1C88000H5の他にS5U1C88000H3(旧型番: ICE88R)があります。

### 新ツール( S5U1C88000P対応版 )と既存ツールとの相違点

旧ペリフェラルボード( S5U1C88316PおよびS5U1C88348P )が標準ペリフェラルボード( S5U1C88000P )に置き換わっています。新旧のペリフェラルボードとデベロップメントツールの組み合わせにより、一部の動作と機能が変わりますので注意が必要です。

S1C88 Family統合ツールパッケージに含まれる以下の機種のデベロップメントツールは標準ペリフェラルボード( S5U1C88000P )に対応した新ツールです。

S1C88104, S1C88112, S1C88308, S1C88316, S1C88317, S1C88348, S1C88832, S1C88862

表6.1 S1C88316/348系ペリフェラルボードとデベロップメントツールの組み合わせによる相違

組み合わせ	機能	S1C88832/862の BZ(R51)、TOUT(R26)出力	OSC1/3発振周波数の可変 (OSC1はCR発振、OSC3はCR/セラミック発振用)
			(OSC1は32.768kHz、OSC3は4.9152MHzに固定のため、外部からのクロック入力に対応)
旧ペリフェラルボード +旧デベロップメントツール	不可		不可
新ペリフェラルボード +新デベロップメントツール	可能		可能
旧ペリフェラルボード +新デベロップメントツール	不可 特に弊害なし		不可(OSC1は水晶選択時32.768kHz、CR選択時約32kHz、OSC3はセラミック選択時8MHz、CR選択時約8MHzに固定のため、外部からのクロック入力に対応) 特に弊害なし
新ペリフェラルボード +旧デベロップメントツール	不可 特に弊害なし		不可(OSC1は32.768kHz、OSC3は4.9152MHzに固定のため、外部からの入力に対応) 特に弊害なし

## 7 内蔵ROM未使用領域FF詰め ユーティリティ <fil88xxx>

### 7.1 fil88xxxの概要

内蔵ROM未使用領域FF詰めユーティリティ <fil88xxx>は、モトローラS2フォーマットのプログラムデータHEXファイルを入力し、未使用領域にFFHを埋め込んだ内蔵ROM( 000000H ~ 00EFFFH )用のHEXファイルを生成します。この出力ファイルを使用してICE( S5U1C88000H5 )によるデバッグを行います。ICEでデバッグする場合、このファイルをホストマシンからダウンロードします。

また、マスクデータチェッカ<winmdc>を用いてセイコーエプソンに提出していただくマスクデータを作成する際のプログラムデータにもなります。

### 7.2 入出力ファイル

図7.2.1にfil88xxxの入出力ファイルを示します。

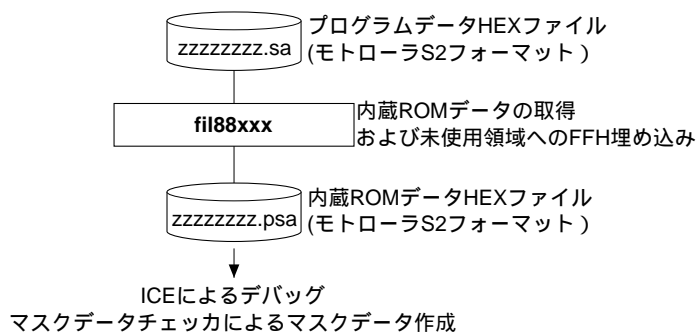


図7.2.1 fil88xxxの入出力ファイル

#### プログラムデータHEXファイル( zzzzzzzzz.sa )

HEXコンバータ<hex88>またはサードパーティー製のソフトウェアツールが出力したモトローラS2フォーマットのプログラムデータHEXファイルです。

#### 内蔵ROMデータHEXファイル( zzzzzzzzz.psa )

入力したプログラムデータHEXファイルから内蔵ROM領域のデータを切り出したモトローラS2フォーマットのファイルです。内蔵ROMの未使用領域にはFFHのコードが埋め込まれます。また、S1C88xxxのシステム予約領域( テクニカルマニュアルのベクタテーブル参照 )へのシステムコードの埋め込みもあわせて行います。

ICEでデバッグを行う場合、ICEのコマンドによってこのファイルをICE上にダウンロードします。

このファイルは完成した他のオプションファイルと共に、マスクデータチェッカ<winmdc>によって1つのファイルにパックし、マスクデータファイルとしてセイコーエプソンに提出していただきます。セイコーエプソンは、そのマスクデータファイルからICのマスクパターンを作成します。

\*1 ファイル名の"xxx"は機種名です。"zzzzzzzz"の部分には任意の名前を付けてください。

\*2 ICEへの内蔵ROMデータHEXファイルのダウンロード方法については、ICEのマニュアルを参照してください。

## 7.3 操作方法

### (1) 起動方法

fil88xxxを起動させるには、MS-DOSプロンプトから次のコマンドを実行します。

>fil88xxx <file name> ☐

☐はリターンキーの入力を表します。

<file name>には入力するモトローラS2フォーマットのプログラムデータHEXファイルを指定します。  
パスも指定できます。

例) C:¥S1C88¥DEV88¥DEV88xxx\_V1>fil88xxx d:¥test¥c8xxx0a0.sa

### (2) 起動メッセージ

fil88xxxが起動すると次のメッセージが表示されます。

```
FIL88xxx Unused Area Filling Utility Version X.XX
Copyright (C) SEIKO EPSON CORP. xxxx
```

### (3) 終了メッセージ

fil88xxxの処理が正常に終了すると、以下のメッセージを表示します。

正常に終了した場合

```
..... ←実行中の進行状況
Unused Area Filling Completed
System Area Data Set Completed
```

出力ファイル(.psa)は入力ファイルと同じフォルダに生成されます。

エラーが発生した場合

```
C8xxx0A0.SA 5:          File Format Error          ←エラーメッセージの例
```

fil88xxx実行中にエラーが発生した場合、エラーが発生したファイル名、行番号およびエラーメッセージを表示してfil88xxxを終了します。エラーの場合は、出力ファイル(.psa)は生成されません。ワーニングメッセージの場合、出力ファイルは生成されます。

### (4) 強制終了する場合

fil88xxxの実行を強制的に終了させたい場合、"CTRL"+"C"を入力します。

## 7.4 エラーメッセージ

以下に、fil88xxxのエラーメッセージおよびワーニングメッセージの一覧を示します。

表7.4.1 エラーメッセージ

メッセージ	説 明
Can't Find File	指定した入力ファイルが存在しない。
Syntax Error: Input File	入力ファイル名の指定がない。
File Format Error	入力ファイルのフォーマットが不正。(*1)
Can't Open File	入力ファイルをオープンできない。
Not S Record	入力ファイルがSレコード以外のレコードになっている。
Data Length	1行のデータ長が不足している。
Too Many Data In One Line	1行のデータが多すぎる。
Not 3Byte Address	アドレス長が3バイト以外。(含むS1, S3, S7, S9レコード)
Check Sum Error	チェックサムが合わない。
Duplicate Error	同一アドレスに2重定義されています。
Can't Use Vector xxH System Reserve	物理アドレス0000xxHはS1C88xxxシステム予約領域のため、ベクタとして使用できない。
Insufficient disk space	ディスクの容量が足りない。
Write Error	ファイルに書き込みができない。

\*1 ファイルのフォーマットエラーには次のことが含まれます。

- ・ S8レコードの後ろにレコードが存在する。
- ・ 文字数が12以外、またはバイトカウントが04以外のS8レコード。
- ・ 16進数以外がある。
- ・ S4, S5, S6レコードを含む場合。
- ・ 1行の文字数が12以下である。
- ・ S8レコードが存在しない。

表7.4.2 ワーニングメッセージ

メッセージ	説 明
Warning: No 00H Address	アドレス000000Hが存在しません。

注: 物理アドレス 000000Hにデータが存在しない場合はワーニングメッセージを出力し、データFFHを埋め込みます。



## 7.5 入出力ファイル例

### 入力ファイル例

```
S2240000000001235000502350235023502350235023502350235023502350235040
S2080000202350015013
S224000100CF6E00F6B4FFDD0030DD0100D94004C700F0C40000CFDCC30200D700F8E7F7F262
S2240001209300D94004B0FFB104C543F8C700F8CFEB7093CF3BE7FBC10001C20001CFEED725

:      :      :      :      :      :      :      :      :

S224007F001818000055AA000101000001000100010002000000401011121314151617181919
S20E007F201A1B1C1D1E1F2F3F0F3FEB
S804000000FB
```

### 出力ファイル例

S1C88xxxのシステム予約領域 (例: アドレス000024Hおよび000025H) にシステムコード (例: F1H、FFH) を設定

```
S2240000000001235000502350235023502350235023502350235023502350235040
S22400002023500150F1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF21
S224000040FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBB
S224000060FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9B
S224000080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF7B
S2240000A0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5B
S2240000C0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3B
S2240000E0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1B
S224000100CF6E00F6B4FFDD0030DD0100D94004C700F0C40000CFDCC30200D700F8E7F7F262
S2240001209300D94004B0FFB104C543F8C700F8CFEB7093CF3BE7FBC10001C20001CFEED725
S22400014000FEE7EEC500F8C600F8CFEE1255F5DAB000F23A04DD2003D94009DD22019C3F7C
S224000160B001CED400F0D94004F27A00F29000F2A600F2BC00F2DD00F21703F23F03F2BD28
S22400018003F2F203CED084F1803204E703B000CED484F1CED003F1803214E703B000CED462
S2240001A003F1CEAECED006F332FFE7F7B000CED406F3F1B3D97560CED0007F7810CED00143
S2240001C07F7811D97801CED0027F7844CED0037F7845DD62FFDD6000DD63F5DD613FD9768C
S2240001E010DD4008F8A2A0C60E7FB100CED084F1CF40464C02CEB0FC297802A8AAF8CED0CC
S22400020084F13203E608F2E503B000F106F2D403B0FFCED407F4F8A2A0C6127FB100CED0CB

:      :      :      :      :      :      :      :      :

S22400EFA0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C
S22400EFC0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4C
S22400EFE0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF2C
S804000000FB
```

## 8 ファンクションオプションジェネレータ<winfog>

### 8.1 winfogの概要

S1C88チップは、I/Oポート機能など、いくつかのハードウェア仕様をマスクオプションとして選択できるようになっています。これにより、開発する製品の仕様に合わせてS1C88チップのマスクパターンを変更し、ハードウェアを構成することができます。

ファンクションオプションジェネレータ<winfog>は、マスクパターン生成のためのファイルを作成するソフトウェアツールで、マスクオプションがGUIにより容易に選択できます。このwinfogで作成されたファイルから、セイコーエプソンはS1C88チップのマスクパターンを生成します。

また、ICE(S5U1C88000H5)を用いてデバッグを行う際に必要なマスクオプション設定用ファイル(モトローラS2フォーマットデータ)と同時に作成できます。ICEでデバッグする場合、このファイルをホストマシンからダウンロードすることによって実ICと同等のオプション機能がICE上で実現できます。

### 8.2 入出力ファイル

図8.2.1にwinfogの入出力ファイルを示します。

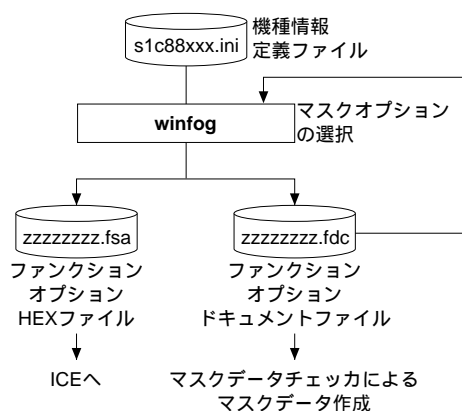


図8.2.1 winfogの入出力ファイル

機種情報定義ファイル(s1c88xxx.ini)

各機種のオプションリストやその他の情報が記録されています。必ずセイコーエプソンが提供するファイルを使用してください。このファイルはファイル名で示される機種にのみ有効です。ファイルの内容を修正したり、他の機種で使用しないでください。

ファンクションオプションドキュメントファイル(zzzzzzzz.fdc)

マスクオプションの選択内容が記録されるテキスト形式のファイルです。このファイルをwinfogに読み込ませて選択済みのオプション設定を修正することもできます。このファイルは完成した他のプログラム/データファイルと共に、マスクデータチェッカ<winmdc>によって1つのファイルにパックし、マスクデータファイルとしてセイコーエプソンに提出していただきます。セイコーエプソンは、そのマスクデータファイルからICのマスクパターンを作成します。

ファンクションオプションHEXファイル(zzzzzzzz.fsa)

選択したマスクオプションをICEに設定するための、モトローラS2フォーマットのファイルです。ICEでデバッグを行う場合、ICEのコマンドによって、このファイルをICE上にダウンロードします。

\*1 ファイル名の"xxx"は機種名です。"zzzzzzzz"の部分には任意の名前を付けてください。

\*2 ICEへのマスクオプションのダウンロード方法については、ICEのマニュアルを参照してください。

## 8.3 操作方法

### 8.3.1 起動方法

#### エクスプローラからの起動



winfog.exeアイコンをダブルクリックするか、スタートメニューからwinfogを選択してください。

前回の実行時に機種情報定義ファイル(s1c88xxx.ini)を読み込んでいる場合は、winfog起動時に同じファイルを自動的に読み込みます。

また、機種情報定義ファイルのアイコンをwinfog.exeアイコンにドラッグすることによってwinfogが起動し、その機種情報定義ファイルを読み込みます。

#### コマンド入力による起動

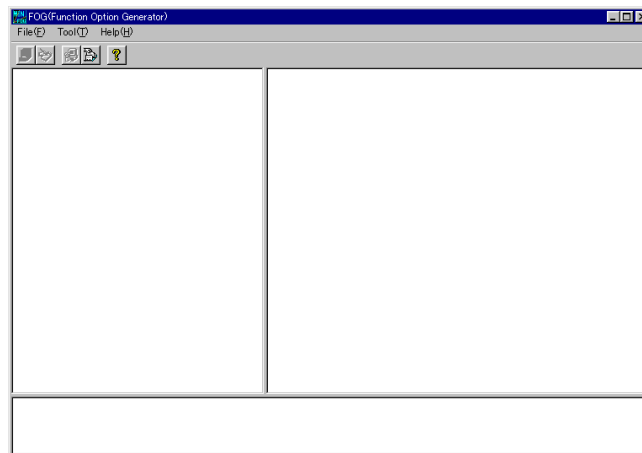
winfogはMS-DOSプロンプトからも次のコマンドで起動可能です。

**>winfog [s1c88xxx.ini]**

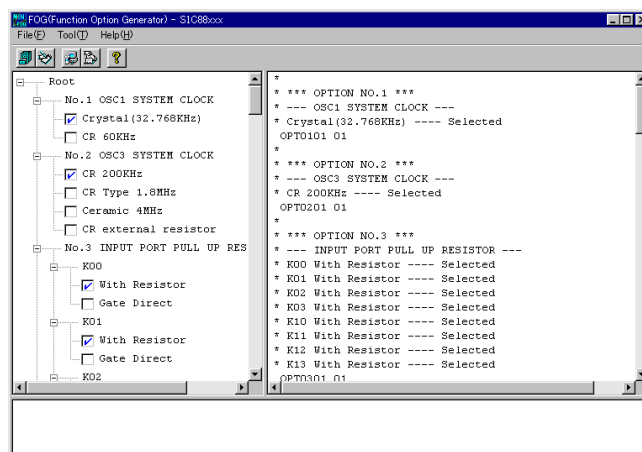
[ ]はリターンキーの入力を表します。

コマンドオプションとして機種情報定義ファイル(s1c88xxx.ini)が指定できます(パスも指定可能)。ここで指定すると、winfog起動時に機種情報定義ファイルが読み込まれます。この指定は省略可能です。

起動すると[FOG]ウィンドウを表示します。以下に機種情報定義ファイルを読み込まなかった場合と読み込んだ場合のウィンドウの表示例を示します。

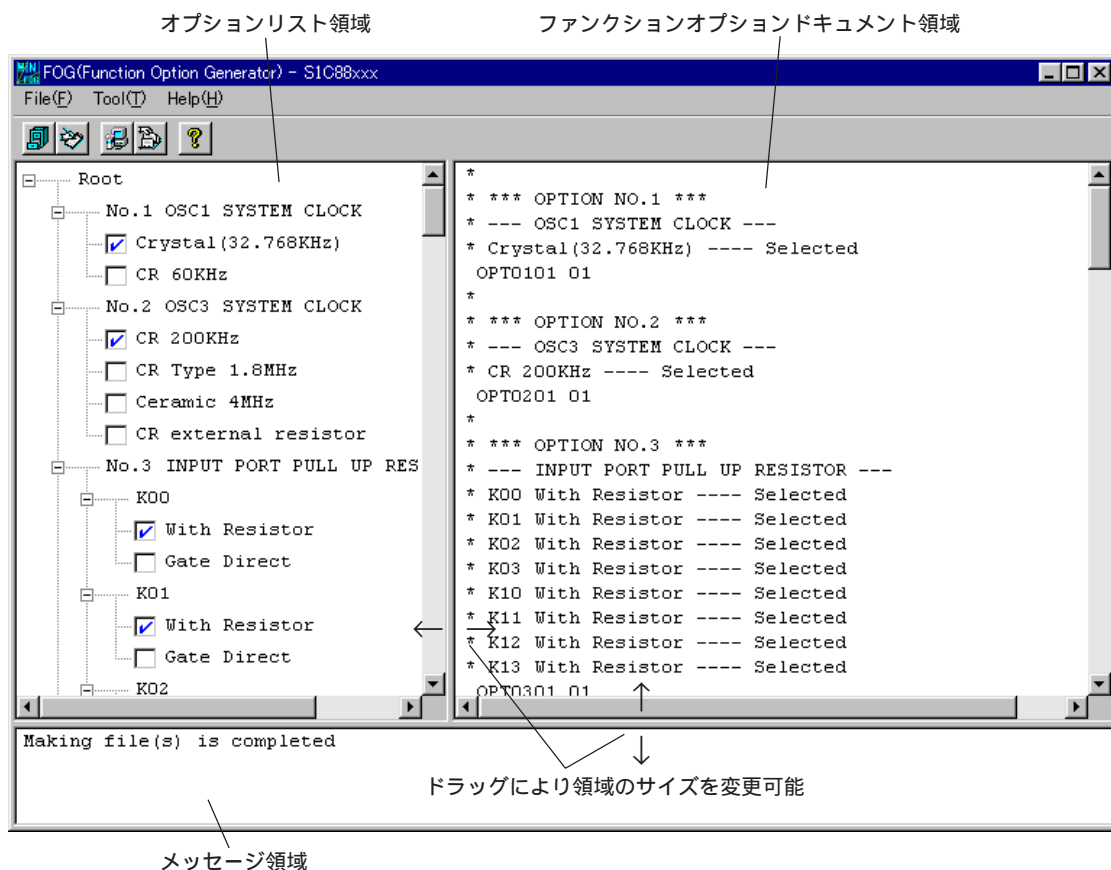


[FOG]ウィンドウ(初期画面)



[FOG]ウィンドウ(機種情報定義ファイル読み込み後)

## 8.3.2 ウィンドウ



- \* タイトルバーの機種名は、読み込んだ機種情報定義ファイルのファイル名(パスと拡張子を除く)です。
- \* オプションリストとファンクションオプションドキュメントの内容は機種により異なります。

図8.3.2.1 ウィンドウの構成

[FOG]ウィンドウは図に示すとおり、3つの領域に分割されています。

## オプションリスト領域

機種情報定義ファイル(s1c88xxx.ini)で設定される、マスクオプションの一覧です。チェックボックスを使用して、各オプションを選択します。チェックマーク(✓)は現在選択されているオプションを示します。

## ファンクションオプションドキュメント領域

オプションの選択内容がファンクションオプションドキュメントの形式で表示されます。ファンクションオプションドキュメントファイルには、この領域の表示内容が出力されます。オプションリスト領域で選択項目を変更すると、表示が即時更新されます。

## メッセージ領域

[Tool]メニューから[Generate]を選択、あるいは[Generate]ボタンをクリックしてファイルを作成した際に、その結果を示すメッセージを表示します。

### 8.3.3 メニューとツールバーボタン

以下、各メニュー項目と、ツールバーボタンについて説明します。

#### [File]メニュー



##### Open

ファンクションオプションドキュメントファイルを開きます。既存のファイルを修正する場合などに使用します。[Open]ボタンも同機能です。

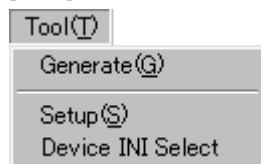


[Open]ボタン

##### End

winfogを終了します。

#### [Tool]メニュー



##### Generate

オプションリストの選択内容でファイルを作成します。[Generate]ボタンも同機能です。



[Generate]ボタン

##### Setup

作成日や出力ファイル名、ファンクションオプションドキュメントファイルに含めるコメントなどを設定します。[Setup]ボタンも同機能です。



[Setup]ボタン

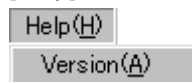
##### Device INI Select

機種情報定義ファイル(s1c88xxx.ini)をロードします。[Device INI Select]ボタンも同機能です。このファイルのロードは最初に行っておく必要があります。



[Device INI Select]ボタン

#### [Help]メニュー



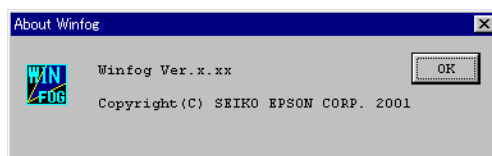
##### Version

winfogのバージョンを表示します。[Help]ボタンも同機能です。



[Help]ボタン

次のダイアログボックスが表示されます。閉じるには[OK]をクリックしてください。



### 8.3.4 操作手順

基本的な操作手順を以下に示します。

#### (1) 機種情報定義ファイルのロード

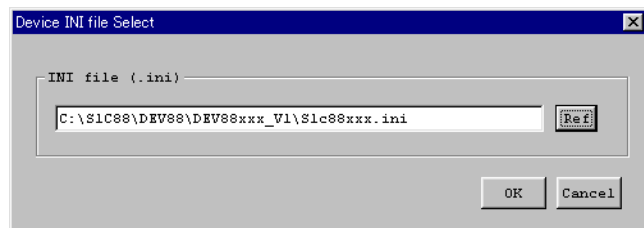
最初に機種情報定義ファイル( s1c88xxx.ini )を選択してロードします。

[Tool]メニューから[Device INI Select]を選択するか、[Device INI Select]ボタンをクリックします。



[Device INI Select]ボタン

次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、デフォルト設定のオプションリストとファンクションオプションドキュメントがそれぞれの領域に表示されます。ファイルのロードを中止するには [Cancel] をクリックします。

一度、機種情報定義ファイルを選択すると、次の起動時は同じファイルが自動的にロードされます。

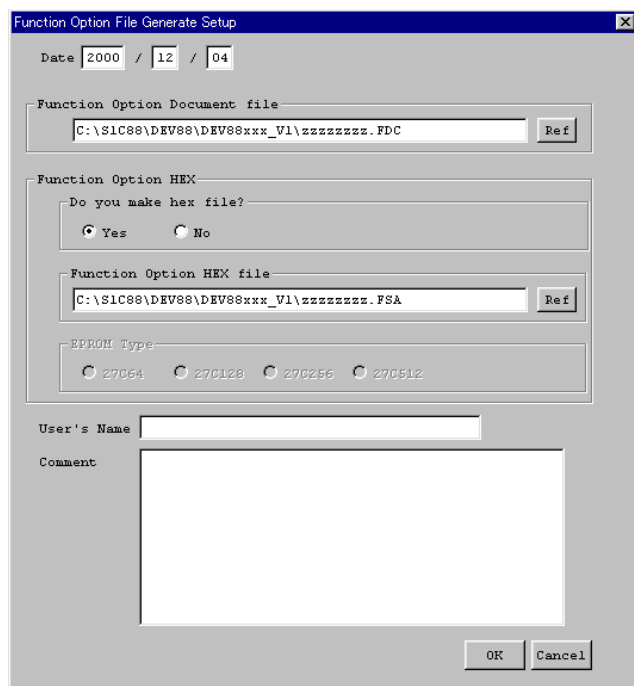
注: オプションをすでに設定している状態で機種情報定義ファイルをロードすると、設定がすべてデフォルトの状態に戻ります。

#### (2) セットアップ

[Tool]メニューから[Setup]を選択するか、[Setup]ボタンをクリックして[Setup]ダイアログボックスを表示させ、必要な選択と入力を行います。



[Setup]ボタン



##### Date

現在の日付が表示されます。必要に応じて変更してください。

##### Function Option Document file

作成するファンクションオプションドキュメントファイル名を、ここで指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンで他のフォルダも参照できます。

##### Function Option HEX

Do you make hex file?

ファンクションオプションHEXファイルを作成するか選択します。ICEを使用したデバッグを行う場合は作成してください。

##### Function Option HEX file

ファンクションオプションHEXファイルを作成する場合に、そのファイル名をここで指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンで他のフォルダも参照できます。

### EPROM Type

これはS1C88 Familyでは選択できません。

### User's Name

お客さまの会社名を入力します。最大40文字まで入力することができ、それを越えた場合は無視されます。英文字、数字、記号およびスペースが入力可能です。

ここに入力した内容は、ファンクションオプションドキュメントファイルのUSER'S NAMEフィールドに記録されます。

### Comment

コメントを入力します。1行に入力可能な文字数は50文字まで、最大10行まで入力することができます。英文字、数字、記号およびスペースが入力可能です。また、[Enter]キーで改行できます。

なお、コメントには、次のような内容を含めるようにお願いします。

- ・ 事業所、所属
- ・ 所在地、電話番号、FAX番号
- ・ その他、技術情報など

ここに入力した内容は、ファンクションオプションドキュメントファイルのCOMMENTフィールドに記録されます。

上記の必要な項目を入力後、[OK]をクリックすると設定内容が保存され、ダイアログボックスが閉じます。設定内容は即時有効となります。

[Cancel]をクリックした場合、現在の設定は変更されずにダイアログボックスが閉じます。

注: • ファイル名の指定には以下の制限があります。

1. パスを含めたファイル名指定の文字数は最大2048文字です。
2. ファイル名( 拡張子を除く )は最大15文字、拡張子は最大3文字です。
3. ファイル名の先頭にハイフン( - )は使用できません。また、ディレクトリ名( フォルダ名 )、ファイル名、拡張子に、以下の記号の使用を禁止します。  
/ : , ; \* ? " < > |

- User's NameとCommentに以下の記号は使用できません。

\$ ¥ ! `

## (3) オプションの選択

オプションリストのチェックボックスをクリックして必要なオプションを選択します。オプションリスト領域で選択項目を変更すると、ファンクションオプションドキュメント領域の表示が更新されます。

なお、オプションリストは、機種情報定義ファイルをロードした時点でデフォルトの選択状態になります。

オプション仕様については、各機種のテクニカルマニュアルを参照してください。

## (4) ファイルの作成

オプションの選択が終了後、[Tool]メニューから[Generate]を選択するか、[Generate]ボタンをクリックしてファイルを作成します。



[Generate]ボタン

[Setup]ダイアログボックスで指定したファンクションオプションドキュメントファイルとファンクションオプションHEXファイル( 指定時のみ )が作成されます。

ファイル作成が正常に終了した場合は、"Making file(s) is completed"がメッセージ領域に表示されます。

エラーが発生した場合は、エラーメッセージが表示されます。

## (5) 既存ドキュメントファイルの修正

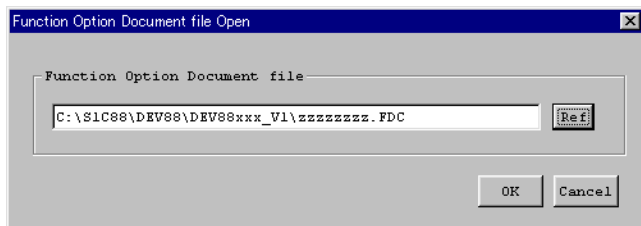
既存のファンクションオプションドキュメントファイルを読み込んで、必要箇所を修正することもできます。

ファイルを読み込むには、[File]メニューから[Open]を選択するか、[Open]ボタンをクリックします。



[Open]ボタン

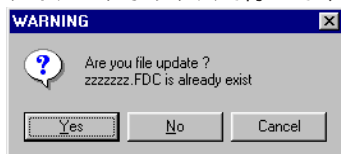
次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、オプションリストとファンクションオプションドキュメント領域がファイルの内容に更新されます。ファイルのロードを中止するには[Cancel]をクリックします。

(2)~(4)の作業を行い、ファイルを更新してください。

ファイル名を変更せずに[Generate]を選択すると、上書きを確認する次のメッセージが表示され、[Yes]をクリックして書き込み、[No]または[Cancel]をクリックして中止できます。ファイル名の変更は[Setup]ダイアログボックスで行ってください。



注: ファンクションオプションドキュメントファイルの読み込みは、機種情報定義ファイルがロードされている場合にのみ行えます。

## (6) 終了

winfogを終了するには、[File]メニューから[End]を選択してください。



## 8.4 エラーメッセージ

winfogのエラーメッセージの一覧を示します。表示の"Dialog"はダイアログボックスに表示されるメッセージを、"Message"は[FOG]ウィンドウのメッセージ領域に表示されるメッセージを示します。

表8.4.1 winfogエラーメッセージ一覧

メッセージ	説 明	表示
File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。	Dialog
Illegal character	入力禁止文字が入力された。	Dialog
Please input file name	ファイル名が未入力。	Dialog
Can't open File : xxxx	ファイル(xxxx)がオープンできない。	Dialog
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。	Dialog
INI file does not include FOG information	指定した機種情報定義ファイル(.ini)にファンクションオプション情報が含まれていない。	Dialog
Function Option document file is not found	指定したファンクションオプションドキュメントファイルが存在しない。	Dialog
Function Option document file does not match INI file	指定したファンクションオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。	Dialog
A lot of parameter	コマンドラインの引数が多すぎる。	Dialog
Making file(s) is completed [xxxx is no data exist]	ファイル作成完了。ただし、作成したファイル(xxxx)にはデータが含まれていない。	Message
Can't open File: xxxx	Generate実行時、ファイル(xxxx)がオープンできない。	Message
Making file(s) is not completed		
Can't write File: xxxx	Generate実行時、ファイル(xxxx)に書き込みができない。	Message
Making file(s) is not completed		

表8.4.2 winfogワーニングメッセージ

メッセージ	説 明	表示
Are you file update? xxxx is already exist	上書き確認メッセージ (指定したファイルは既に存在する。)	Dialog

## 8.5 出力ファイル例

注: オプションの構成等は、機種により異なります。

### ファンクションオプションドキュメントファイル例

```
* S1C88xxx FUNCTION OPTION DOCUMENT Vx.xx      ←バージョン
*
* FILE NAME      zzzzzzzzz.FDC                  ←ファイル名( [Setup]で指定 )
* USER'S NAME    SEIKO EPSON CORPORATION        ←ユーザ名( [Setup]で指定 )
* INPUT DATE      yyyy/mm/dd                    ←作成年月日( [Setup]で指定 )
* COMMENT         SAMPLE DATA                  ←コメント( [Setup]で指定 )
*
* *** OPTION NO.1 ***                          ←オプション番号
* --- OSC1 SYSTEM CLOCK ---                    ←オプション名
* Crystal(32.768KHz) ---- Selected             ←選択した仕様
OPT0101 01                                     ←マスクデータ
*
* *** OPTION NO.2 ***
* --- OSC3 SYSTEM CLOCK ---
* CR 200KHz ---- Selected
OPT0201 01
*
* *** OPTION NO.3 ***
* --- INPUT PORT PULL UP RESISTOR ---
* K00 With Resistor ---- Selected
* K01 With Resistor ---- Selected
* K02 With Resistor ---- Selected
* K03 With Resistor ---- Selected
* K04 With Resistor ---- Selected
* K05 With Resistor ---- Selected
* K06 With Resistor ---- Selected
* K07 With Resistor ---- Selected
OPT0301 01
OPT0302 01
OPT0303 01
OPT0304 01
OPT0305 01
OPT0306 01
OPT0307 01
OPT0308 01
*
* *** OPTION NO.4 ***
* --- OUTPUT PORT OUTPUT SPECIFICATION ---
* R00 Complementary ---- Selected
* R01 Complementary ---- Selected
* R02 Complementary ---- Selected
* R03 Complementary ---- Selected
OPT0401 01
OPT0402 01
OPT0403 01
OPT0404 01
*
*                                     :
*
* *** OPTION NO.8 ***
* --- SOUND GENERATOR POLARITY ---
* NEGATIVE ---- Selected
OPT0801 01
*EOF                                         ←エンドマーク
```

### ファンクションオプションHEXファイル例(モトローラS2フォーマット)

```
S22400000022FF0200FFFFFFFFFFFFFFFFFFFFFFFF00000000000000FFFFFFFFFFFFFFFFFCD
S804000000FB
```

モトローラS2フォーマットについては、"A.2.5.3 モトローラS2フォーマットについて"を参照してください。

## 9 セグメントオプションジェネレータ<winsog>

### 9.1 winsogの概要

S1C88 Familyの一部の機種はLCD出力端子の出力仕様、表示メモリとLCD出力端子の割り付けをハードウェアオプションで設定できるようになっており、オプション設定に従ってICのマスクパターンが作成されます。セグメントオプションジェネレータ<winsog>は、マスクパターン生成のためのファイルを作成するソフトウェアツールで、セグメントオプションがGUIにより容易に設定できます。

また、ICE (S5U1C88000H5)を用いてデバッグを行う際に必要なマスクオプション設定用ファイル(モトローラS2フォーマットデータ)も同時に作成できます。ICEでデバッグする場合、このファイルをホストマシンからダウンロードすることによって実ICと同等のオプション機能がICE上で実現できます。

注: セグメントオプションジェネレータ<winsog>は、セグメントオプションが設定されていない機種では使用しません。

### 9.2 入出力ファイル

図9.2.1にwinsogの入出力ファイルを示します。

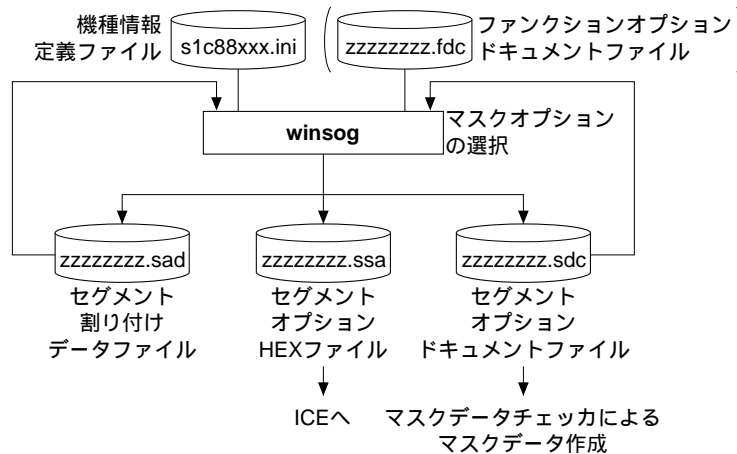


図9.2.1 winsogの入出力ファイル

機種情報定義ファイル( s1c88xxx.ini )

各機種のオプションリストやその他の情報が記録されています。必ずセイコーエプソンが提供するファイルを使用してください。このファイルはファイル名で示される機種にのみ有効です。ファイルの内容を修正したり、他の機種で使用しないでください。

ファンクションオプションドキュメントファイル( zzzzzzzz.fdc )

ファンクションオプションジェネレータ<winfog>で作成した、マスクオプションの選択内容が記録されているテキスト形式のファイルです。このファイルは、セグメントオプションの設定条件がwinfogのマスクオプションの選択によって決定する機種にのみ必要となります。

セグメントオプションドキュメントファイル( zzzzzzzz.sdc )

セグメントオプションの設定内容が記録されるテキスト形式のファイルです。このファイルをwinsogに読み込ませてオプション設定を修正することもできます。このファイルは完成した他のプログラム/データファイルと共に、マスクデータチェック<winmdc>によって1つのファイルにパックし、マスクデータファイルとしてセイコーエプソンに提出していただきます。セイコーエプソンは、そのマスクデータファイルからICのマスクパターンを作成します。

## セグメントオプションHEXファイル( zzzzzzzz.ssa )

選択したセグメントオプションをICEに設定するための、モトローラS2フォーマットのファイルです。  
ICEでデバッグを行う場合、ICEのコマンドによって、このファイルをICE上にダウンロードします

## セグメント割り付けデータファイル( zzzzzzzz.sad )

割り付け途中のセグメントオプションを記録しておくためのテキスト形式のファイルです。作業途中で winsog を終了する場合などにこのファイルを作成しておきます。今回はこのファイルを winsog に読み込ませることで続きのオプション設定が行えます。

- \*1 ファイル名の"xxx"は機種名です。"zzzzzzzz"の部分には任意の名前を付けてください。
- \*2 ICEへのマスクオプションのダウンロード方法については、ICEのマニュアルを参照してください。

## 9.3 操作方法

### 9.3.1 起動方法

#### エクスプローラからの起動



winsog.exe アイコンをダブルクリックするか、スタートメニューから winsog を選択してください。

前回の実行時に機種情報定義ファイル( s1c88xxx.ini )を読み込んでいる場合は、winsog 起動時に同じファイルを自動的に読み込みます。

また、機種情報定義ファイルのアイコンを winsog.exe アイコンにドラッグすることによっても winsog が起動し、その機種情報定義ファイルを読み込みます。

ファンクションオプションドキュメントファイルが必要な機種では、そのファイル名を入力するダイアログボックスが表示されますので、テキストボックスにパスも含め入力してください。あるいは、[Ref] ボタンをクリックしてファイルを選択してください。

#### コマンド入力による起動

winsog は MS-DOS プロンプトからも次のコマンドで起動可能です。

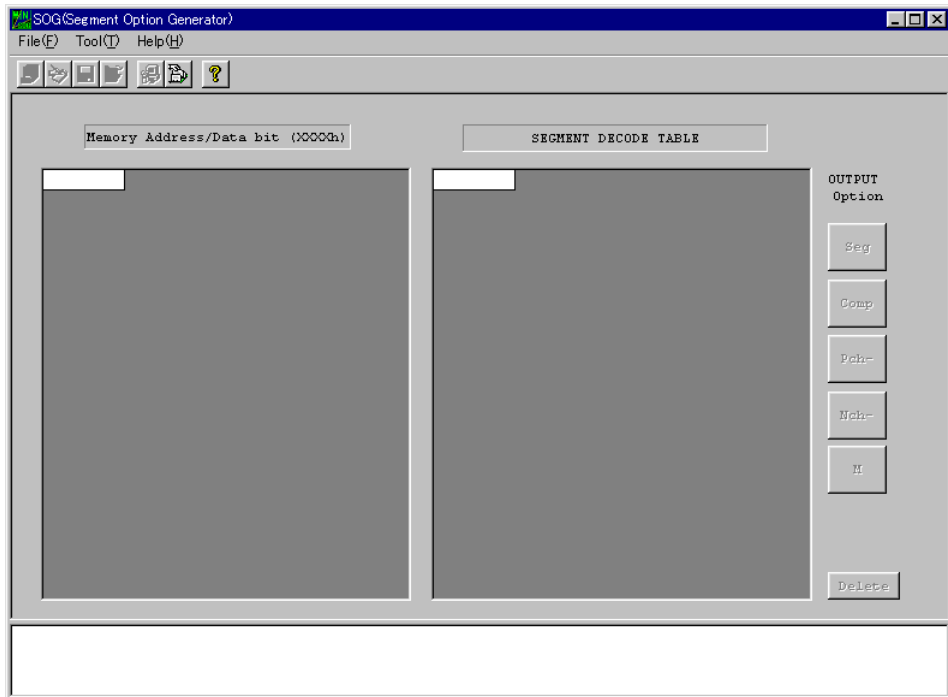
**>winsog [s1c88xxx.ini]**

はリターンキーの入力を表します。

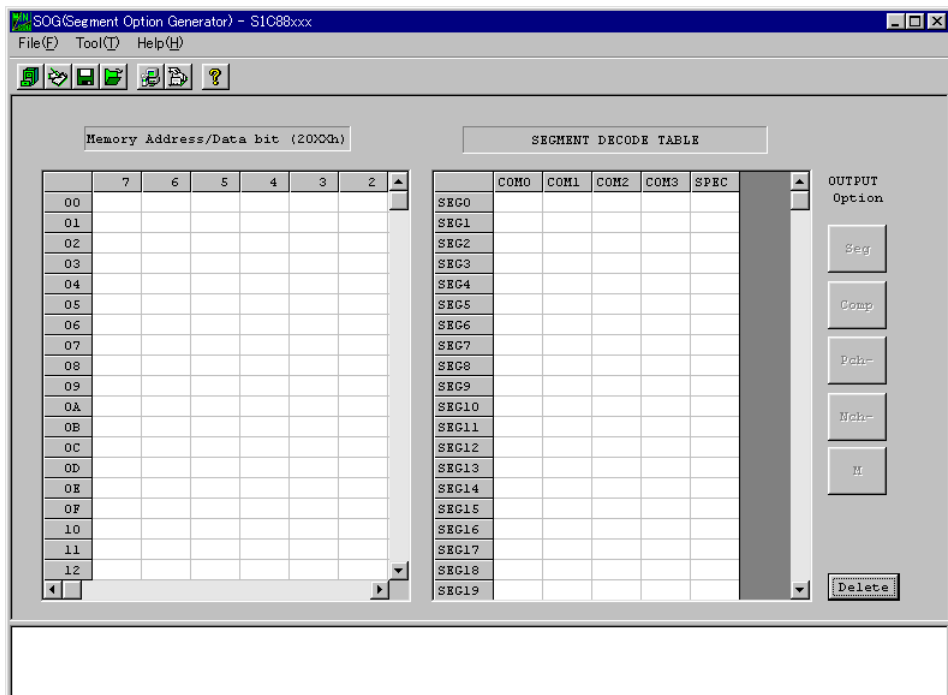
コマンドオプションとして機種情報定義ファイル( s1c88xxx.ini )が指定できます(パスも指定可能)。ここで指定すると、winsog 起動時に機種情報定義ファイルが読み込まれます。ファンクションオプションドキュメントファイルが必要な機種の場合は、そのファイルが s1c88xxx.ini および winsog.exe と同じフォルダに用意されている状態でコマンドを入力してください。コマンド実行後、ファンクションオプションドキュメントファイル名を入力するダイアログボックスが表示されますので、テキストボックスにパスも含め入力してください。あるいは、[Ref] ボタンをクリックしてファイルを選択してください。

機種情報定義ファイルの指定は省略可能です。

起動すると[SOG]ウィンドウを表示します。以下に機種情報定義ファイルを読み込まなかった場合と読み込んだ場合のウィンドウの表示例を示します。

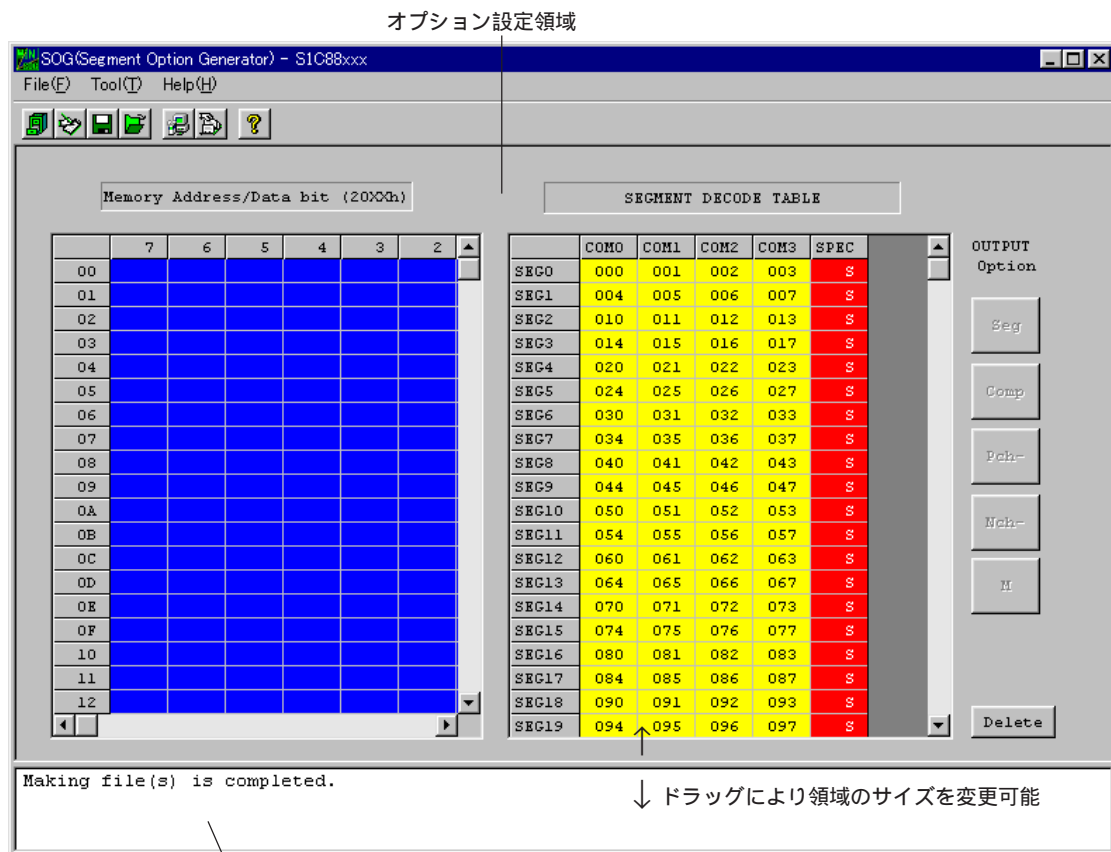


[SOG]ウィンドウ(初期画面)



[SOG]ウィンドウ(機種情報定義ファイル読み込み後)

## 9.3.2 ウィンドウ



- \* タイトルバーの機種名は、読み込んだ機種情報定義ファイルのファイル名(パスと拡張子を除く)です。
- \* 表示メモリアドレスとセグメントの構成は機種により異なります。

図9.3.2.1 ウィンドウの構成

[SOG]ウィンドウは図に示すとおり、2つの領域に分割されています。

## オプション設定領域

表示メモリマップとセグメントデコードテーブル、端子の仕様を選択するボタンで構成されています。表示メモリマップとセグメントデコードテーブルのセルをクリックすることで、表示メモリアドレス/ビットの割り付けが行えます。

## メッセージ領域

[Tool]メニューから[Generate]を選択、あるいは[Generate]ボタンをクリックしてファイルを作成した際に、その結果を示すメッセージを表示します。

### 9.3.3 メニューとツールバーボタン

以下、各メニュー項目と、ツールバーボタンについて説明します。

#### [File]メニュー



##### Open

セグメントオプションドキュメントファイルを開きます。既存のファイルを修正する場合などに使用します。[Open]ボタンも同機能です。



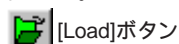
##### Record - Save

現在のオプション設定内容をファイル(セグメント割り付けデータファイル)に保存します。[Save]ボタンも同機能です。



##### Record - Load

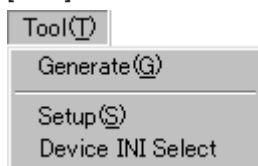
セグメント割り付けデータファイルを読み込みます。[Load]ボタンも同機能です。



##### End

winsogを終了します。

#### [Tool]メニュー



##### Generate

設定したセグメントオプションの内容でファイルを作成します。[Generate]ボタンも同機能です。



##### Setup

作成日や出力ファイル名、セグメントオプションドキュメントファイルに含めるコメントなどを設定します。[Setup]ボタンも同機能です。

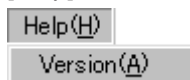


##### Device INI Select

機種情報定義ファイル(s1c88xxx.ini)をロードします。[Device INI Select]ボタンも同機能です。このファイルのロードは最初に行っておく必要があります。

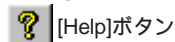


#### [Help]メニュー

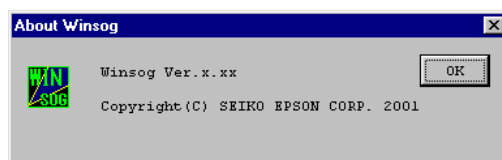


##### Version

winsogのバージョンを表示します。[Help]ボタンも同機能です。



次のダイアログボックスが表示されます。閉じるには[OK]をクリックしてください。








### 9.3.4 オプション選択用ボタン

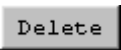
オプション設定領域には以下のボタンが用意されています。

#### OUTPUT Optionボタン

SEG端子の出力仕様を選択するボタンです。[SEGMENT DECODE TABLE]のSPECのセルをクリックして選択した場合に有効となります。

	LCDセグメント出力を選択します。
	DC-コンプリメンタリ出力を選択します。
	DC-Pchオープンドレイン出力を選択します。
	DC-Nchオープンドレイン出力を選択します。
	セグメント/コモン共有出力を選択します。

#### [Delete]ボタン

	選択したセグメント割り付けをクリアします。[Delete]キーも同機能です。
---	--

### 9.3.5 操作手順

基本的な操作手順を以下に示します。

#### (1) 機種情報定義ファイルのロード

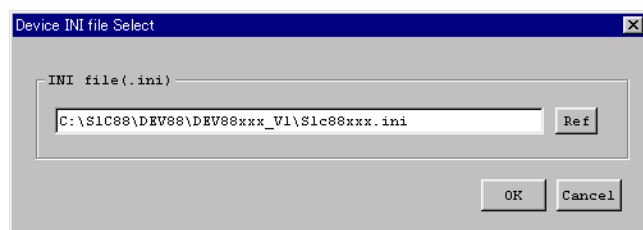
最初に機種情報定義ファイル(s1c88xxx.ini)を選択してロードします。

[Tool]メニューから[Device INI Select]を選択するか、[Device INI Select]ボタンをクリックします。



[Device INI Select]ボタン

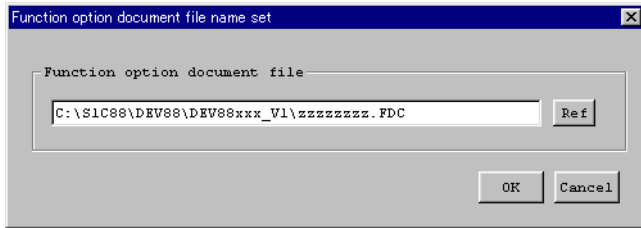
次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、読み込まれた機種情報によりwinsog内の各種設定が初期化されます。ファイルのロードを中止するには[Cancel]をクリックします。



一度、機種情報定義ファイルを選択すると、次の起動時は同じファイルが自動的にロードされます。機種情報定義ファイルをロード後、ファンクションオプションドキュメントファイルが必要な機種ではそのファイル名を入力するダイアログボックスが表示されますので、テキストボックスにパスも含め入力してください。あるいは、[Ref]ボタンをクリックしてファイルを選択してください。



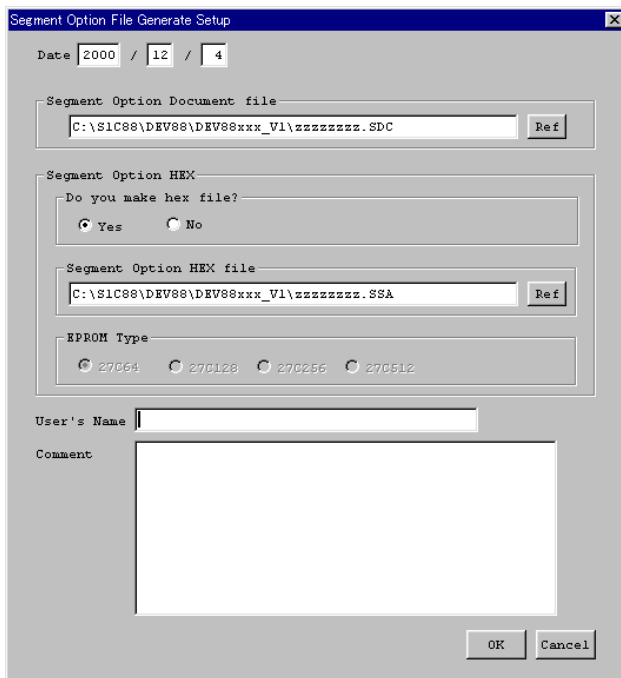
注: オプションをすでに設定している状態で機種情報定義ファイルをロードすると、設定がすべてクリアされます。

## (2) セットアップ

[Tool]メニューから[Setup]を選択するか、[Setup]ボタンをクリックして[Setup]ダイアログボックスを表示させ、必要な選択と入力を行います。



[Setup]ボタン



### Date

現在の日付が表示されます。必要に応じて変更してください。

### Segment Option Document file

作成するセグメントオプションドキュメントファイル名を、ここで指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンで他のフォルダも参照できます。

### Segment Option HEX

#### Do you make hex file?

セグメントオプションHEXファイルを作成するか選択します。ICEを使用してデバッグを行う場合は作成してください。

### Segment Option HEX file

セグメントオプションHEXファイルを作成する場合に、そのファイル名をここで指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンで他のフォルダも参照できます。

## EPROM Type

これはS1C88 Familyでは選択できません。

## User's Name

お客さまの会社名を入力します。最大40文字まで入力することができます。英文字、数字、記号およびスペースが入力可能です。

ここに入力した内容は、セグメントオプションドキュメントファイルのUSER'S NAMEフィールドに記録されます。

## Comment

コメントを入力します。1行に入力可能な文字数は50文字まで、最大10行まで入力することができます。英文字、数字、記号およびスペースが入力可能です。また、[Enter]キーで改行できます。

なお、コメントには、次のような内容を含めるようにお願いします。

- ・ 事業所、所属
- ・ 所在地、電話番号、FAX番号
- ・ その他、技術情報など

ここに入力した内容は、セグメントオプションドキュメントファイルのCOMMENTフィールドに記録されます。

上記の必要な項目を入力後、[OK]をクリックすると設定内容が保存され、ダイアログボックスが閉じます。設定内容は即時有効となります。

[Cancel]をクリックした場合、現在の設定は変更されずにダイアログボックスが閉じます。

注: ファイル名の指定には以下の制限があります。

1. パスを含めたファイル名指定の文字数は最大2048文字です。
2. ファイル名( 拡張子を除く )は最大15文字、拡張子は最大3文字です。
3. ファイル名の先頭にハイフン( - )は使用できません。また、ディレクトリ名( フォルダ名 )、ファイル名、拡張子に、以下の記号の使用を禁止します。

/ : , ; \* ? " < > |

- ・ User's NameとCommentに以下の記号は使用できません。

\$ ¥ ! `

## (3) セグメント出力の設定

S1C88 Familyでセグメントオプションが設定されているLCD駆動回路は、通常、2端子ごと( 機種によっては端子個々 )にセグメント出力とDC出力の選択が可能です。LCDパネルの駆動に用いる場合はセグメント出力を選択します。

セグメント出力ポートはセグメントデコードを内蔵しており、表示メモリ領域の任意のアドレス、データビットを任意のセグメントに割り付けることができます。このセグメントメモリのビットを1に設定すると割り付けられたセグメントが点灯し、0にすると消灯します。セグメントと表示メモリビットは1対1に対応しており、複数のセグメントに同一の表示メモリビットを重複して設定することはできません。したがって、セグメントをすべて異なるアドレス、データビットにする必要があります。

表示メモリマップおよびセグメント割り付けの詳細については、各機種のテクニカルマニュアルを参照してください。

以下の説明は、コモン端子がCOM0～COM3の4本あるものとして行います。

セグメントの割り付けは次のように行います。

1. [Memory Address/Data bit]のテーブルから、割り付けるメモリアドレス/データビットのセルをクリックして選択します。セルが青色に変わります。  
間違えたセルを選択してしまった場合は、正しいセルを選択し直してください。  
テーブルの横の行が表示メモリアドレスに対応します。Memory Address/Data bitのタイトルの横に表示されている16進数が表示メモリのベースアドレスで、テーブル内の各行にはアドレスの下位バイトのみが表示されます。テーブルの縦の列はデータビットに対応します。
2. [SEGMENT DECODE TABLE]から、1で選択したメモリアドレス/データビットを割り付けるSEG端子/COM端子のセルをクリックして選択します。セルにアドレス(上位2桁)とデータビット(下位1桁)を示す3桁の数値が表示され、セルは黄色に変わります。

選択例:

	7	6	5	4	3	2	▲		COM0	COM1	COM2	COM3	SPEC		▲
00								SEG0	007						
01								SEG1							

間違えたセルを選択してしまった場合は、[Delete]ボタンをクリックしてその割り付けをクリアし、再度1から指定し直してください。セルを範囲選択し、[Delete]ボタンでクリアすることも可能です。  
[SEGMENT DECODE TABLE]のセルを選択する前に、必ず[Memory Address/Data bit]のセルを選択してください。

3. 2で選択したセグメントのSPECのセルをクリックし、[Seg]ボタンをクリックします。  
セルはSを表示して赤色に変わります。これにより、そのセグメントがLCDセグメント出力端子に設定されます。  
セグメント出力とDC出力の選択が2端子ごとの場合は、ペアとなるもう一方の端子の仕様も同じに設定されます。

選択例:

	7	6	5	4	3	2	▲		COM0	COM1	COM2	COM3	SPEC		▲
00								SEG0	007	006	005	004	S		
01								SEG1	017	016	015	014	S		

4. 1~2をLCD出力に使用するすべてのセグメントについて行います。なお、3の仕様の選択は、後から行ってもかまいません。

1つのSEG端子の中で使用しないCOMのセルは空白のままにしておいてください。

選択例:

08								SEG8	087	086	085		S		
09								SEG9	097	096	095		S		

#### (4) DC出力の設定

SEG端子を汎用DC出力として使用する場合も、"(3)セグメント出力の設定"のステップ1と2の手順でセグメント割り付けを行います。ただし、出力制御はCOM0に割り付けられた表示メモリが有効となり、COM1~COM3に割り付けられた表示メモリは無効となります。したがって、メモリアドレス/データビットはCOM0のセルにのみ設定し、COM1~COM3のセルは空白にしておきます。

DC出力の場合は、出力仕様としてコンプリメンタリ出力とNch(またはPch)オープンドレイン出力のどちらかを選択できます。

SPECのセルで、以下のボタンを使用して選択してください。

[Comp]ボタン: コンプリメンタリ出力(C)

[Nch-]ボタン: Nchオープンドレイン出力(N)

[Pch-]ボタン: Pchオープンドレイン出力(P)

選択が2端子ごとの場合は、ペアとなるもう一方の端子の仕様も同じに設定されます。

選択例:

02								SEG2	027				C		
03								SEG3	037				C		
04								SEG4	047				P		
05								SEG5	057				P		

選択可能な出力仕様については、各機種のテクニカルマニュアルを参照してください。

#### (5) セグメント/コモン共有端子の出力設定

セグメント出力とコモン出力を共有する端子は、ファンクションオプションの選択によって出力内容が決まります。

SEG端子として使用する場合は前記のように割り付けを行い、使用しないCOMのセルは空白にしておきます。COM端子として使用する場合は割り付けを行わずに、出力仕様にセグメント/コモン共有出力[M]ボタンを選択します。

注: この設定はセグメント/コモン共有端子がある機種にのみ必要です。

## (6) 未使用SEG端子の設定

LCD出力およびDC出力のどちらにも使用しないISEG端子は、[SEGMENT DECODE TABLE]のCOM0～COM3のセルを空白にしておきます。ただし、SPECのセルは空白が許されませんので、セグメント出力(S)を選択してください。

選択例:

SEG6					S	
SEG7					S	

## (7) ファイルの作成

オプションの選択が終了後、[Tool]メニューから[Generate]を選択するか、[Generate]ボタンをクリックしてファイルを作成します。



[Generate]ボタン

[Setup]ダイアログボックスで指定したセグメントオプションドキュメントファイルとセグメントオプションHEXファイル(指定時のみ)が作成されます。

ファイル作成が正常に終了した場合は、"Making file(s) is completed"がメッセージ領域に表示されます。エラーが発生した場合は、エラーメッセージが表示されます。

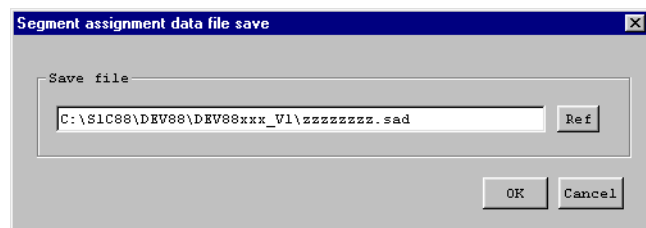
## (8) 割り付け途中のセグメントオプションデータの保存

セグメントオプションの割り付け途中でデータを保存することができます。データを保存するには、[File]メニューから[Record - Save]を選択するか、[Save]ボタンをクリックします。



[Save]ボタン

次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックして保存先のフォルダを選択し、ファイルの選択またはファイル名の入力を行ってください。



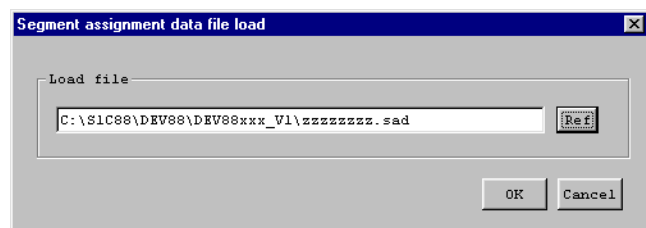
[OK]をクリックすると、割り付けデータを指定したファイルに保存します。ファイルの保存を中止するには[Cancel]をクリックします。

保存したセグメント割り付けデータファイルをロードするには、[File]メニューから[Record - Load]を選択するか、[Load]ボタンをクリックします。



[Load]ボタン

次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、保存した割り付け内容が表示されますので、割り付けの続きを行うことができます。ファイルのロードを中止するには[Cancel]をクリックします。

- 注: • セグメント割り付けデータファイルの読み込みは、機種情報定義ファイルがロードされている場合にのみ行えます。
- ファンクションオプションドキュメントファイルが必要な機種では、起動時に読み込んだファンクションオプションドキュメントファイルにより設定条件が変わりますので、異なる内容があるセグメント割り付けデータファイルを読み込むことはできません。

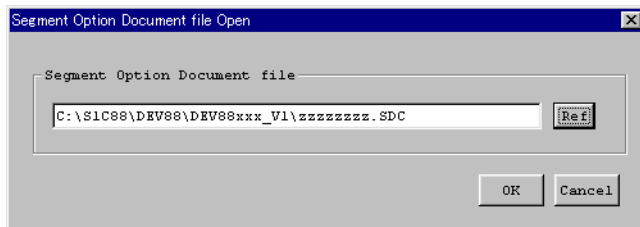
### (9) 既存ドキュメントファイルの修正

既存のセグメントオプションドキュメントファイルを読み込んで、必要箇所を修正することもできます。ファイルを読み込むには、[File]メニューから[Open]を選択するか、[Open]ボタンをクリックします。



[Open]ボタン

次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、[Memory Address/Data bit]と[SEGMENT DECODE TABLE]がファイルの内容に更新されます。ファイルのロードを中止するには[Cancel]をクリックします。

割り付けアドレスを変更する場合は、一度そのセルの割り付けを[Delete]ボタンでクリアしてから再度割り付けを行ってください。出力仕様を変更する場合も、一度SPECのセルを選択して[Delete]ボタンでクリアしてから再度選択してください。セルを範囲選択し、[Delete]ボタンでクリアすることも可能です。ファイル名を変更せずに[Generate]を選択すると、上書きを確認するダイアログボックスが表示され、[Yes]をクリックして書き込み、[No]または[Cancel]をクリックして中止できます。ファイル名の変更は[Setup]ダイアログボックスで行ってください。

- 注: • セグメントオプションドキュメントファイルの読み込みは、機種情報定義ファイルがロードされている場合にのみ行えます。
- ファンクションオプションドキュメントファイルが必要な機種では、起動時に読み込んだファンクションオプションドキュメントファイルにより設定条件が変わりますので、異なる内容があるセグメントオプションドキュメントファイルを読み込むことはできません。

### (10) 終了

winsogを終了するには、[File]メニューから[End]を選択してください。

## 9.4 エラーメッセージ

winsogのエラーメッセージの一覧を示します。表示の"Dialog"はダイアログボックスに表示されるメッセージを、"Message"は[SOG]ウィンドウのメッセージ領域に表示されるメッセージを示します。

表9.4.1 winsogエラーメッセージ一覧

メッセージ	説明	表示
File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。	Dialog
Illegal character	入力禁止文字が入力された。	Dialog
Please input file name	ファイル名が未入力。	Dialog
Can't open File : xxxx	ファイル(xxxx)がオープンできない。	Dialog
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。	Dialog
INI file does not include SOG information	指定した機種情報定義ファイル(.ini)にセグメントオプション情報が含まれていない。	Dialog
Function Option document file is not found	指定したファンクションオプションドキュメントファイルが存在しない。	Dialog
Function Option document file does not match INI file	指定したファンクションオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。	Dialog
Segment Option document file is not found	指定したセグメントオプションドキュメントファイルが存在しない。	Dialog
Segment Option document file does not match INI file	指定したセグメントオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。	Dialog
Segment assignment data file is not found	指定したセグメント割り付けデータファイルが存在しない。	Dialog
Segment assignment data file does not match INI file	指定したセグメント割り付けデータファイルの内容が機種情報定義ファイル(.ini)と異なる。	Dialog
Can't open File: xxxx Making file(s) is not completed	Generate実行時、ファイル(xxxx)がオープンできない。	Message
Can't write File: xxxx Making file(s) is not completed	Generate実行時、ファイル(xxxx)に書き込みができない。	Message
ERROR: SPEC is not set Making file(s) is not completed	空白のSPECセルがある状態でGenerateを実行した。	Message

表9.4.2 winsogワーニングメッセージ

メッセージ	説明	表示
Are you file update? xxxx is already exist	上書き確認メッセージ (指定したファイルは既に存在する。)	Dialog

## 9.5 出力ファイル例

注: 表示メモリアドレス、SEG/COM端子数および出力仕様は機種により異なります。

### セグメントオプションドキュメントファイル例

```
* S1C88xxx SEGMENT OPTION DOCUMENT Vx.xx      ←バージョン
*
* FILE NAME      zzzzzzzz.SDC                  ←ファイル名( [Setup]で指定 )
* USER'S NAME    SEIKO EPSON CORPORATION       ←ユーザ名( [Setup]で指定 )
* INPUT DATE      yyyy/mm/dd                   ←作成年月日( [Setup]で指定 )
* COMMENT         SAMPLE DATA                 ←コメント( [Setup]で指定 )
*
*
* OPTION NO.xx                                     ←オプション番号( 機種により異なる )
*
* < LCD SEGMENT DECODE TABLE >
*
* SEG COM0 COM1 COM2 COM3 SPEC
*
*   0  163  162  161  1F3  S                    ←セグメントデコードテーブル
*   1  170  172  171  160  S
*   2  143  142  141  1E1  S
*   3  150  152  151  140  S
*       :
*   xx  3B0  3B1  3B2  3B3  S
* *EOF                                           ←エンドマーク
```

### セグメント割り付けデータファイル例

```
* S1C88xxx SEGMENT OPTION DOCUMENT Vx.xx      ←バージョン
*
* FILE NAME      zzzzzzzz.SAD                  ←ファイル名
* USER'S NAME    SEIKO EPSON CORPORATION       ←ユーザ名( [Setup]で指定 )
* INPUT DATE      yyyy/mm/dd                   ←作成年月日( [Setup]で指定 )
* COMMENT         SAMPLE DATA                 ←コメント( [Setup]で指定 )
*
*
* OPTION NO.xx                                     ←オプション番号( 機種により異なる )
*
* < LCD SEGMENT DECODE TABLE >
*
* SEG COM0 COM1 COM2 COM3 SPEC
*
*   0  163  162  161  1F3  S                    ←割り付け済みセグメントデータ
*   1  170  172  171  160  S
*   2  143  142  141  1E1  S
*       :
*   mm  FRE  FRE  FRE  FRE  X                    ←FRE: セグメントアドレス/データビット未割り付け
*   nn  FRE  FRE  FRE  FRE  X                    ←X:   出力仕様未設定
*   oo  FRE  FRE  FRE  FRE  X
* *EOF                                           ←エンドマーク
```

### セグメントオプションHEXファイル例( モトローラS2フォーマット )

```
S2240000001603160216011F03FFFFFFFFFFFFFFFF1700170217011600FFFFFFFFFFFFFFFF23
S2240000201403140214011E01FFFFFFFFFFFFFFFF1500150215011400FFFFFFFFFFFFFFFF14
:
S2240010E0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0B
S804000000FB
```

モトローラS2フォーマットについては、"A.2.5.3 モトローラS2フォーマットについて"を参照してください。



## 10 マスクデータチェッカ<winmdc>

### 10.1 winmdcの概要

マスクデータチェッカ<winmdc>は、内蔵ROM未使用領域FF詰めユーティリティ <fil88xxx>によって生成された内蔵ROMデータHEXファイル、ファンクションオプションジェネレータ<winfog>によって生成されたファンクションオプションドキュメントファイル、セグメントオプションジェネレータ<winsog>によって生成されたセグメントオプションドキュメントファイルの各フォーマットをチェックし、マスクパターン生成のためのデータファイルを作成するソフトウェアツールです。

また、作成されたマスクデータファイルを元のファイル形式に復元する機能も持っています。

### 10.2 入出力ファイル

図10.2.1にwinmdcの入出力ファイルを示します。

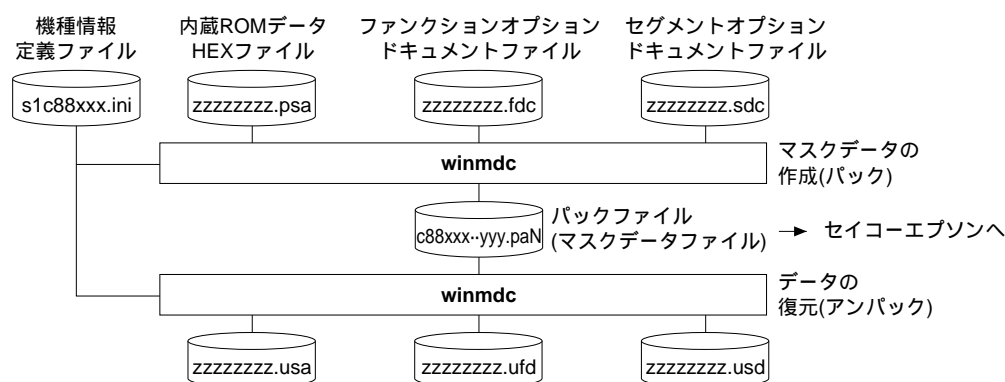


図10.2.1 winmdcの入出力ファイル

#### 機種情報定義ファイル( s1c88xxx.ini )

各機種のオプションリストやその他の情報が記録されています。必ずセイコーエプソンが提供するファイルを使用してください。このファイルはファイル名で示される機種にのみ有効です。ファイルの内容を修正したり、他の機種で使用しないでください。

#### 内蔵ROMデータHEXファイル( zzzzzzzz.psa )

内蔵ROM未使用領域FF詰めユーティリティ <fil88xxx>で生成した、モトローラS2フォーマットの内蔵ROMデータファイルです。内蔵ROMの未使用領域にはFFHのコードが埋め込まれています。また、S1C88xxxのシステム予約領域にはシステムコードが埋め込まれています。

#### ファンクションオプションドキュメントファイル( zzzzzzzz.fdc )

ファンクションオプションの選択内容が記録されたテキスト形式のファイルです。ファンクションオプションジェネレータ<winfog>で作成します。

#### セグメントオプションドキュメントファイル( zzzzzzzz.sdc )

セグメントオプションの設定内容が記録されるテキスト形式のファイルです。セグメントオプションジェネレータ<winsog>で作成します。このファイルはセグメントオプションの設定されている機種にのみ存在します。

#### バックファイル( c88xxx-yyy.paN, N=0 ~ )

上記のデータファイルを1つにまとめたテキスト形式のファイルです。これをマスクデータファイルとしてセイコーエプソンに提出していただきます。セイコーエプソンは、そのマスクデータファイルからICのマスクパターンを作成します。

- \* ファイル名の"xxx-"は機種名です。ファイル名が"yyy"の部分は、お客さまのカスタムコードが入りますので、セイコーエプソンより提示されるコードを入れてください。"zzzzzzzz"の部分には任意の名前を付けてください。



## 10.3 操作方法

### 10.3.1 起動方法

#### エクスプローラからの起動



winmdc.exeアイコンをダブルクリックするか、スタートメニューからwinmdcを選択してください。

前回の実行時に機種情報定義ファイル(s1c88xxx.ini)を読み込んでいる場合は、winmdc起動時に同じファイルを自動的に読み込みます。

また、機種情報定義ファイルのアイコンをwinmdc.exeアイコンにドラッグすることによってもwinmdcが起動し、その機種情報定義ファイルを読み込みます。

#### コマンド入力による起動

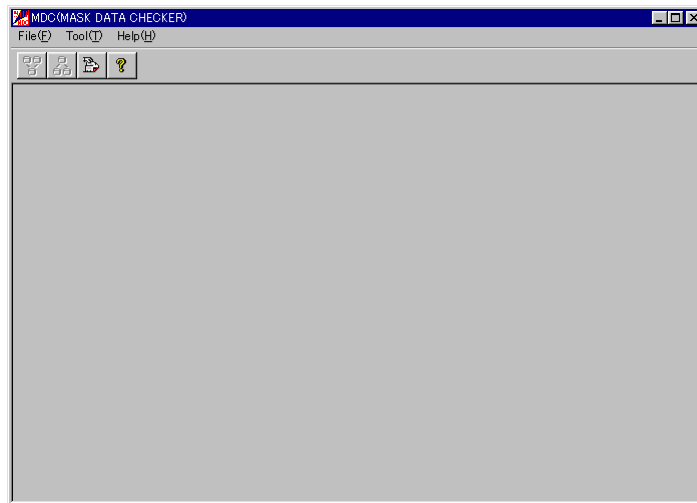
winmdcはMS-DOSプロンプトからも次のコマンドで起動可能です。

```
>winmdc [s1c88xxx.ini] [ ]
```

[ ]はリターンキーの入力を表します。

コマンドオプションとして機種情報定義ファイル(s1c88xxx.ini)が指定できます(パスも指定可能)。ここで指定すると、winmdc起動時に機種情報定義ファイルが読み込まれます。この指定は省略可能です。

起動すると[MDC]ウィンドウを表示します。



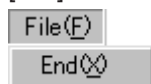
[MDC]ウィンドウ(初期画面)

- \* タイトルバーの機種名は、読み込んだ機種情報定義ファイルのファイル名(パスと拡張子を除く)です。
- \* ツールバーの[Pack]と[Unpack]ボタンは、機種情報定義ファイルが読み込まれると有効になります。

### 10.3.2 メニューとツールバーボタン

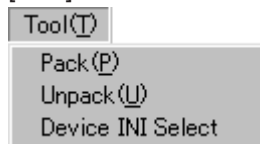
以下、各メニュー項目と、ツールバーボタンについて説明します。

#### [File]メニュー



End  
winmdcを終了します。

#### [Tool]メニュー



Pack  
ROMデータファイルとオプションドキュメントファイルをパックして、提出用のマスクデータファイルを作成します。[Pack]ボタンも同機能です。



[Pack]ボタン

Unpack  
パック後のファイルから元の形式のファイルを復元します。[Unpack]ボタンも同機能です。



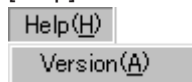
[Unpack]ボタン

Device INI Select  
機種情報定義ファイル(s1c88xxx.ini)をロードします。[Device INI Select]ボタンも同機能です。このファイルのロードは最初に行っておく必要があります。



[Device INI Select]ボタン

#### [Help]メニュー

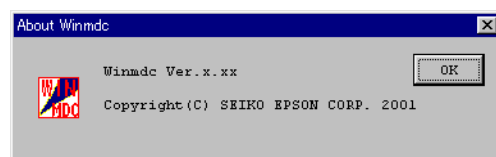


Version  
winmdcのバージョンを表示します。[Help]ボタンも同機能です。



[Help]ボタン

次のダイアログボックスが表示されます。閉じるには[OK]をクリックしてください。



### 10.3.3 操作手順

基本的な操作手順を以下に示します。

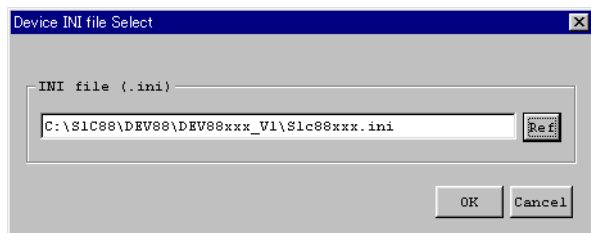
#### (1) 機種情報定義ファイルのロード

最初に機種情報定義ファイル(s1c88xxx.ini)を選択してロードします。

[Tool]メニューから[Device INI Select]を選択するか、[Device INI Select]ボタンをクリックします。

 [Device INI Select]ボタン

次のダイアログが表示されますので、テキストボックスにパスを含むファイル名を入力するか、[Ref]ボタンをクリックしてファイルの選択を行ってください。



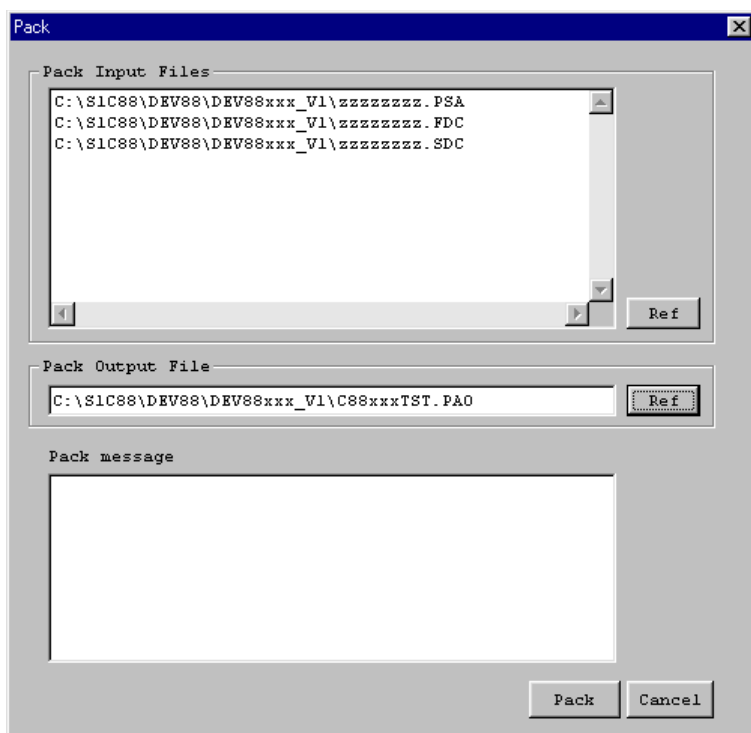
[OK]をクリックすると、ファイルをロードします。指定したファイルが存在し、内容に問題がなければ、読み込まれた機種情報によりwinmdc内の各種設定が初期化されます。ファイルのロードを中止するには[Cancel]をクリックします。

一度、機種情報定義ファイルを選択すると、次回の起動時は同じファイルが自動的にロードされます。

#### (2) パック

1. [Tool]メニューから[Pack]を選択するか、ツールバーの[Pack]ボタンをクリックして[Pack]ダイアログボックスを表示させます。

 [Pack]ボタン



## 2. 入力するファイルを選択します。

[Pack Input Files]には、機種情報定義ファイルで指定される種類のファイルがデフォルトのファイル名でリストされます。

入力するデータファイルをリストと異なる名前で用意してある場合は、次の手順でファイル名を置き換えてください。

- a. リストボックス内の変更するファイル名をクリックして選択します。
- b. [Ref]ボタンをクリックし、入力するデータファイルを選択してください。

これを、リストされているすべてのファイルについて行います。

置き換える場合は、ファイルを間違えないように注意してください。入力ファイルが不正な場合、バック時にエラーとなります。

## 3. 出力ファイル名を設定します。

[Pack Output File]テキストボックスで、出力するバックファイル名を指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンで他のフォルダも参照できます。

出力ファイル名の拡張子は".pa0"としてください。一度セイコーエプソンにデータを提出された後、プログラム等の不具合により再提出される場合は、最後の数値を1つ増やして入力します。たとえば、二度目の提出ファイルは、"c88xxx・yyy.pa1"とします。

注: ファイル名の指定には以下の制限があります。

1. バスを含めたファイル名指定の文字数は最大2048文字です。
2. ファイル名(拡張子を除く)は最大15文字、拡張子は最大3文字です。
3. ファイル名の先頭にハイフン(-)は使用できません。また、ディレクトリ名(フォルダ名)、ファイル名、拡張子に、以下の記号の使用を禁止します。  
/ : ; , \* ? " < > |

## 4. [Pack]ボタンをクリックしてバックを実行します。

正常に終了した場合は、[Pack message]テキストボックスに"Pack completed !"が表示されます。

エラーが発生した場合は、エラーメッセージが表示されます。

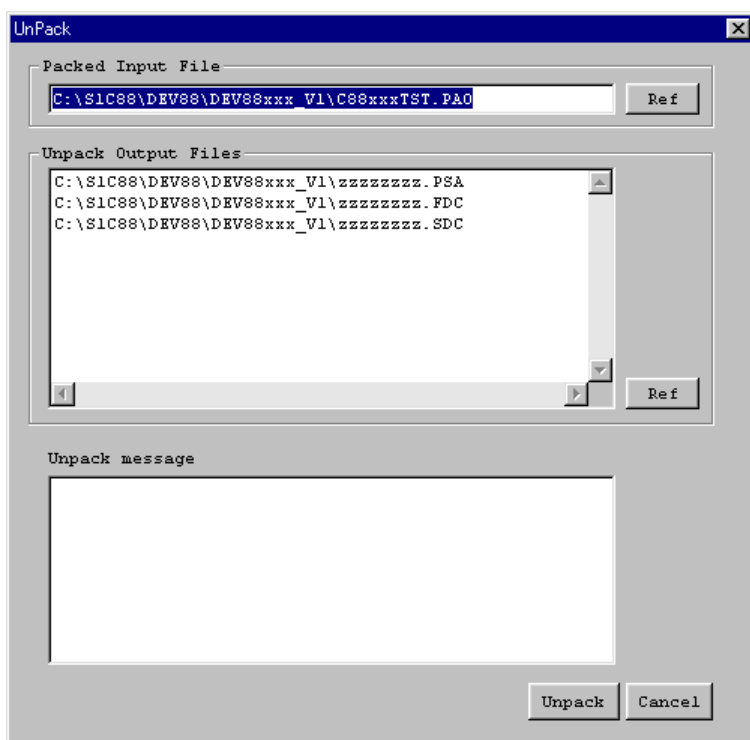
## 5. [Cancel]ボタンをクリックしてダイアログボックスを閉じます。

バック実行前に[Cancel]ボタンをクリックして終了することもできます。

### (3) アンパック

1. [Tool]メニューから[Unpack]を選択するか、ツールバーの[Unpack]ボタンをクリックして[Unpack]ダイアログボックスを表示させます。

 [Unpack]ボタン



2. アンパックするファイルを選択します。  
[Packed Input File]テキストボックスで、入力するパックファイル名を指定します。デフォルトで表示される名前を修正して使用してください。[Ref]ボタンでファイルを選択することもできます。
3. 出力ファイル名を設定します。  
[Unpack Output Files]には、機種情報定義ファイルで指定される種類のファイルがデフォルトのファイル名でリストされます。デフォルトで表示されるファイル名を次の手順で修正してください。
  - a. リストボックス内の変更するファイル名をクリックして選択します。
  - b. [Ref]ボタンをクリックし、出力するフォルダを選択してファイル名の入力を行ってください。これを、リストされているすべてのファイルについて行います。拡張子は変更できません。
4. [Unpack]ボタンをクリックしてアンパックを実行します。  
正常に終了した場合は、[Unpack message]テキストボックスに"Unpack completed !"が表示されます。エラーが発生した場合は、エラーメッセージが表示されます。
5. [Cancel]ボタンをクリックしてダイアログボックスを閉じます。  
アンパック実行前に[Cancel]ボタンをクリックして終了することもできます。

### (4) 終了

winmdcを終了するには、[File]メニューから[End]を選択してください。

## 10.4 エラーメッセージ

winmdcのエラーメッセージの一覧を示します。表示の"Dialog"はダイアログボックスに表示されるメッセージを、"Message"は[Pack]または[Unpack]ダイアログボックスのメッセージ領域に表示されるメッセージを示します。

表10.4.1 I/Oエラーメッセージ一覧

メッセージ	説明	表示
File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。	Dialog
Illegal character	入力禁止文字が入力された。	Dialog
Please input file name	ファイル名が入力されていない。	Dialog
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。	Dialog
INI file does not include MDC information	指定した機種情報定義ファイル(.ini)にMDC情報が含まれていない。	Dialog
Can't open file : xxxx	ファイル(xxxx)がオープンできない。	Dialog
Can't write file: xxxx	ファイル(xxxx)に書き込みができない。	Dialog

表10.4.2 ROMデータエラーメッセージ一覧

メッセージ	説明	表示
Hex data error: Not S record.	データが"S"で始まっていない。	Message
Hex data error: Data is not sequential.	データが昇順に並んでいない。	Message
Hex data error: Illegal data.	不当なキャラクタがある。	Message
Hex data error: Too many data in one line.	1行中のデータ数が多すぎる。	Message
Hex data error: Check sum error.	チェックサムが合わない。	Message
Hex data error: ROM capacity over.	データ容量が大きすぎる。(データサイズ>ROMサイズ)	Message
Hex data error: Not enough the ROM data.	データ容量が少ない。(データサイズ<ROMサイズ)	Message
Hex data error: Illegal start mark.	スタートマークが不当である。	Message
Hex data error: Illegal end mark.	エンドマークが不当である。	Message
Hex data error: Illegal comment.	データの最初の機種名表示が不当である。	Message

表10.4.3 ファンクションオプションデータエラーメッセージ一覧

メッセージ	説明	表示
Option data error : Illegal model name.	機種名が不当である。	Message
Option data error : Illegal version.	バージョンが不当である。	Message
Option data error : Illegal option number.	オプションNo.が不当である。	Message
Option data error : Illegal select number.	選択肢No.が不当である。	Message
Option data error : Mask data is not enough.	マスクデータが充分でない。	Message
Option data error : Illegal start mark.	スタートマークが不当である。	Message
Option data error : Illegal end mark.	エンドマークが不当である。	Message

表10.4.4 セグメントオプションデータエラーメッセージ一覧

メッセージ	説明	表示
LCD segment data error : Illegal model name.	機種名が不当である。	Message
LCD segment data error : Illegal version.	バージョンが不当である。	Message
LCD segment data error : Illegal segment No.	セグメントNo.が不当である。	Message
LCD segment data error : Illegal segment area.	表示メモリのアドレスが範囲外である。	Message
LCD segment data error : Illegal segment output specification.	出力仕様が不当である。	Message
LCD segment data error : Illegal data in this line.	16進数と出力仕様以外の記述がある。	Message
LCD segment data error : Data is not enough.	セグメントデータが充分でない。	Message
LCD segment data error : Illegal start mark.	スタートマークが不当である。	Message
LCD segment data error : Illegal end mark.	エンドマークが不当である。	Message

## 10.5 出力ファイル例

注: データの構成および内容は機種により異なります。

### バックファイル(マスクデータファイル)例

```

*
* S1C88xxx MASK DATA VER x.xx
*
¥FROM1
S1C88xxxxyy PROGRAM ROM
S224000000.....
:      :      :      :      :
S804000000FB
S224000000.....
:      :      :      :      :
S804000000FB
¥END
¥OPTION1
* S1C88xxx FUNCTION OPTION DOCUMENT V x.xx
*
* FILE NAME      zzzzzzzz.FDC
* USER'S NAME    SEIKO EPSON CORPORATION
* INPUT DATE      yyyy/mm/dd
* COMMENT         SAMPLE DATA
*
* *** OPTION NO.1 ***
* --- OSC1 SYSTEM CLOCK ---
* Crystal(32.768KHz) ---- Selected
OPT0101 01
:      :      :      :      :
OPTnn01 01
*EOF
¥END
¥SEGMENT1
* S1C88xxx SEGMENT OPTION DOCUMENT Vx.xx
*
* FILE NAME      zzzzzzzz.SDC
* USER'S NAME    SEIKO EPSON CORPORATION
* INPUT DATE      yyyy/mm/dd
* COMMENT         SAMPLE DATA
*
*
* OPTION NO.xx
*
* < LCD SEGMENT DECODE TABLE >
*
* SEG COM0 COM1 COM2 COM3 SPEC
*
*   0  163  162  161  1F3  S
*   1  170  172  171  160  S
*      :
*  xx  3B0  3B1  3B2  3B3  S
*EOF
¥END

```

←バージョン

←内蔵ROM HEXデータスタートマーク  
←機種名

"zzzzzzzz.psa"

←内蔵ROM HEXデータエンドマーク  
←ファンクションオプションスタートマーク  
←機種名/バージョン

"zzzzzzzz.fdc"

←ファンクションオプションエンドマーク  
←セグメントオプションスタートマーク  
←機種名/バージョン

"zzzzzzzz.sdc"

←セグメントオプションエンドマーク

# 11 自己診断プログラム<t88xxx>

## 11.1 t88xxxの概要

t88xxxは、S1C88 Familyのプログラムデバッグに使用するハードウェアツールICE( S5U1C88000H5 )およびS5U1C88xxxPの動作を診断する、自己診断プログラムです。ICEおよびS5U1C88xxxPについては、デバッグで使用する際本プログラムにより定期的な自己診断を行ってください。

## 11.2 ファイル構成

### ( 1 )プログラムデータHEXファイル( t88xxx.psa )

fil88xxxにより生成された、内蔵ROM未使用領域にFFHを埋め込まれ、またS1C88xxxのシステム予約領域に対しシステムコードが設定された自己診断プログラム本体です。

### ( 2 )ファンクションオプションHEXファイル( t88xxx.fsa )

winfogにより生成され、自己診断時に使用するICEおよびS5U1C88xxxPにファンクションオプションを設定するためのファイルです。

### ( 3 )ファンクションオプションドキュメントファイル(t88xxx.fdc)

上記ファンクションオプションHEXファイルの設定内容を記述したwinfogにより生成されるドキュメントファイルです。

### ( 4 )セグメントオプションHEXファイル( t88xxx.ssa )

winsogにより生成され、自己診断時に使用するICEおよびS5U1C88xxxPにセグメントオプションを設定するためのファイルです。

### ( 5 )セグメントオプションドキュメントファイル(t88xxx.sdc)

上記セグメントオプションHEXファイルの設定内容を記述したwinsogにより生成されるドキュメントファイルです。

( 4 )と( 5 )のセグメントオプション用のファイルは、セグメントオプションが設定された機種にのみ用意されます。

### ( 6 )readme.txt

自己診断時に使用するS5U1C88xxxPのLEDの点灯状態が記述されています。

## 11.3 使用方法

ICEにS5U1C88xxxPを装着後、以下の動作試験によりICEおよびS5U1C88xxxPの自己診断が行えます。

なお、以下の動作試験には自己診断用プログラム( t88xxx.psa )、ファンクションオプションHEXデータ( t88xxx.fsa )を使用します。セグメントオプションが設定されている機種の試験にはセグメントオプションHEXデータ( t88xxx.ssa )も必要です。

操作手順は以下のとおりです。

### ( 1 )自己診断用プログラム( t88xxx.psa )、ファンクションオプションHEXデータ( t88xxx.fsa )、セグメントオプションHEXデータ( t88xxx.ssa )をICEに読み込み、実行させます。

読み込み、実行方法に関する説明は、ICEのマニュアルを参照してください。

### ( 2 )S5U1C88xxxPのLED点灯状態を確認します。システムリセット後、readme.txtに示す点灯状態が確認できれば正常です。readme.txtに記述されている"サイクルカウント"は1秒間隔を表し、1秒毎にLEDの点灯状態が変化します。



# 12 88xxx.parファイル

88xxx.parファイルは、機種個別の定義情報を持ったマクロファイルです。ICE( S5U1C88000H5 )は本パラメータファイルに記述された内容にしたがって動作環境設定を行います。また、このファイルが存在しない場合は、ICEは起動しません。

## 12.1 88xxx.parファイル内容

出荷時の88xxx.parファイルの一例を以下に示します。

```
[Options]
Prccclksel=0          ...(1)
Vdddown=0             ...(2)
CC=0                  ...(3)
DIAG=0                ...(4)

[MAP Config]
;Slc88xxx MAP Configuration Setting
; 000000-00FFFF:Define 1 byte unit
; 010000-FFFFFF:Define 256 bytes unit
;
;syntax:<Start address> <End address> [E][I][U][S][W]
;      E:Emulation memory
;      I:I/O (PRC Board) memory
;      U:User memory
;      S:Stack area
;      W:Write protect (Default does not protect)

;Internal ROM
Map0=000000 00EFFF E W    ...(5)

;Internal RAM
Map1=00F000 00F3FF E

;Stack area
Map2=00F400 00F5FF E S

;Display memory
Map3=00F800 00F828 I
Map4=00F833 00F842 I
Map5=00F900 00F928 I
Map6=00F933 00F942 I
Map7=00FA00 00FA28 I
Map8=00FA33 00FA42 I
Map9=00FB00 00FB28 I
Map10=00FB33 00FB42 I
Map11=00FC00 00FC28 I
Map12=00FC33 00FC42 I
Map13=00FD00 00FD28 I
Map14=00FD33 00FD42 I

;I/O memory
Map15=00FF00 00FF02 I
Map16=00FF10 00FF12 I
Map17=00FF20 00FF25 I
Map18=00FF30 00FF34 I
Map19=00FF35 00FF36 I W
Map20=00FF40 00FF40 I
Map21=00FF41 00FF41 I W
Map22=00FF42 00FF42 I
Map23=00FF43 00FF43 I W
Map24=00FF44 00FF45 I
Map25=00FF48 00FF4A I
Map26=00FF50 00FF53 I
Map27=00FF54 00FF55 I W
Map28=00FF61 00FF61 I
Map29=00FF63 00FF63 I
Map30=00FF70 00FF71 I
Map31=00FF75 00FF75 I
Map32=00FF78 00FF78 I
```

## 12.2 88xxx.parファイル内容説明

- (1)~(4)はシステム予約となっていますので変更しないでください。  
 (5)以降の設定によりメモリのアロケーションおよびコンディションの設定を行います。

Map<Serial number> = <Start address> <End address> <Switch>

### シリアルナンバーの記述

Mapの後ろに0~1023の範囲でシリアルナンバーを記述してください。シリアルナンバーは順不同です。  
 番号が重複したときは、先に記述された設定が有効となり後に記述された内容は無効となります。

### アドレス設定

アドレス000000~00FFFFは1バイト単位の設定が可能です。アドレス010000以降は256バイト単位  
 (\*\*\*\*00~\*\*\*\*FF)の設定となります。

### スイッチ設定

メモリのアロケーション(E、I、Uスイッチ)

Iを指定すると、そのエリアはS5U1C88xxxP上のメモリに割り付けられます。  
 Eを指定すると、そのエリアはICE上のエミュレーションメモリに割り付けられます。  
 Uを指定すると、そのエリアはターゲットボード上のユーザメモリに割り付けられます。

スタックエリアの設定(Sスイッチ)

Sを指定すると、そのエリアはスタック領域として設定されます。

ライトプロテクト/ROM指定(Wスイッチ)

Wを指定するとライトプロテクト(ROMエリア)となります。Wが指定されない場合、ICEはRAMエリアと認識します。

### コメントの記述

各行の先頭にセミコロン(;)が記述されているとき、ICEはその行をコメント行として認識します。パラメータ設定の後ろにコメントを記述することはできません。

例)

```
;Internal ROM ...OK
Map0=000000 00FFFF E W ;internal ROM ...NG
```

## 12.3 エミュレーションメモリについて

ICEはアドレス000000~00FFFF番地のメモリとして使用可能な64Kバイトのエミュレーションメモリ、および010000番地以降のメモリとして使用可能なエミュレーションメモリを、S5U1C88000H5は512Kバイト、S5U1C88000H3は256Kバイト内蔵しています。

外部メモリを必要とするシステム構成において、外部メモリに割り付けられるメモリ空間は、このエミュレーションメモリを用いることが可能となります。そこで、お客さまはターゲットボード上にユーザメモリを搭載することなくプログラム開発およびデバッグが行えます。

なお、010000番地以降に割り付け可能なエミュレーションメモリはMax.512Kバイトとなります。512Kバイト以上の外部メモリを必要とするときは、ターゲットボード上に別途ご用意ください。

### 注意

- Windowsシステムのフォルダにあるice88\*.ini(\*=rまたはur)ファイル内のpathを変更してください。ただし、88xxx.parファイルがice88\*.exeと同じフォルダに存在する場合は、ファイル名のみでpathの記述は不要です。  
 ICE88\* for Windowsインストール直後は、ice88\*.exeと同一ディレクトリにdefault.parが配置され、ice88\*.iniファイル内のpath設定はdefault.parを参照する設定となっています。
- (1)~(4)の内容は[Options]以降に記述してください。(5)以降の内容は[MAP Config]以降に記述してください。また、[Options]および[MAP Config]は削除しないでください。

# 13 S1C88 Familyデバッガ

## 13.1 概要

デバッガdb88はS1C88 Familyの開発ツールです。このデバッガにより、S1C88 Family統合ツール( Cコンパイラ、アセンブラ等 )で作成したプログラムをICE( S5U1C88000H5 )を使用してデバッグできます。

デバッガの特長を以下に示します。

- マルチウィンドウにより各種のデータを一度に参照可能
- 使用頻度の高いコマンドはツールバーおよびメニューからマウス操作で実行可能
- Cソース表示、逆アセンブルによるプログラムコード表示およびシンボル表示機能
- プログラムの連続実行と3種類のステップ実行が可能
- 3種類のブレーク機能
- トレースおよびカバレッジ機能
- コマンドファイルによるコマンドの自動実行機能

## 13.2 入出力ファイル

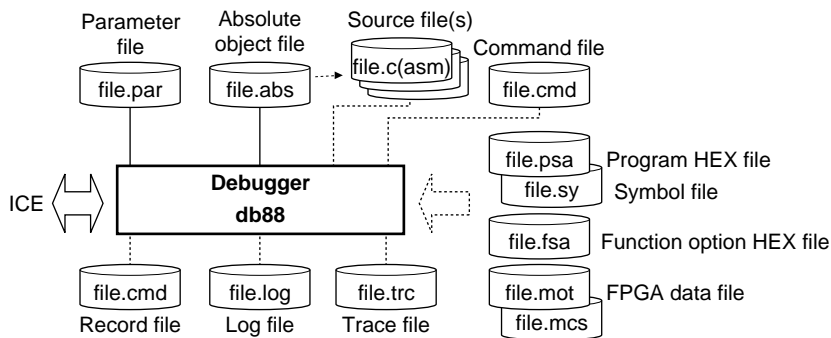


図13.2.1 入出力ファイル

パラメータファイル( file\_name.par )

各種種のメモリ情報などが記録されたテキストファイルで、ICEのメモリマップ設定に使用されます。内容については、"12 88xxx.parファイル"を参照してください。

アブソリュートオブジェクトファイル( file\_name.abs )

アドバンスドロケータまたはロケータで生成したIEEE-695形式のオブジェクトファイルです。デバッグ情報を含んだIEEE-695形式のファイルを読み込むことで、Cソース表示およびシンボリックデバッグが行えます。

ソースファイル( file\_name.c, file\_name.asm )

上記オブジェクトファイルのソースファイルで、ソース表示を行うときに読み込まれます。

内蔵ROMデータHEXファイル( file\_name.psa )

内蔵ROM未使用領域FF詰めユーティリティ( fil88xxx )で生成した、モトローラS2フォーマットの内蔵ROMデータファイルです。内蔵ROMの未使用領域にはFFHのコードが埋め込まれています。また、S1C88xxxのシステム予約領域にはシステムコードが埋め込まれています。

シンボル情報ファイル( file\_name.sy )

シンボルテーブルファイルジェネレータで生成したシンボル情報ファイルです。上記のROMデータHEXファイルと同じ名称で同じディレクトリに用意しておくことにより、ROMデータHEXファイルのロード時に自動的に読み込まれます。これによりソースで定義したシンボルの表示等が行えるようになります。

ファンクションオプションHEXファイル( file\_name.fsa )

ファンクションオプションジェネレータで生成した、モトローラS2フォーマットのマスクオプション設定用ファイルです。

FPGAデータファイル( file\_name.mot, file\_name.mcs )

ペリフェラルボードS5U1C88000P上のFPGAを各機種用に設定するためのデータファイルです。".mot"はモトローラS2フォーマット、".mcs"はIntel HEXフォーマットのファイルです。

コマンドファイル( file\_name.cmd )

連続して実行させるデバッグコマンドを記述したテキストファイルです。頻繁に使用する一連のコマンドを書き込んでおくことで、キーボードからのコマンド入力の手間を省くことができます。このファイルはcomコマンドにより読み込み、実行させます。

ログファイル( file\_name.log )

実行したコマンドと実行結果が出力されます。このファイル出力はlogコマンドによって制御できます。

レコードファイル( file\_name.cmd )

実行したコマンドがテキスト形式で出力されます。このファイル出力はrecコマンドによって制御できます。出力されたファイルはそのままコマンドファイルとして使用できます。

トレースファイル( file\_name.trc )

トレースデータの指定範囲が出力されます。このファイル出力はtfコマンドによって制御できます。

## 13.3 起動と終了

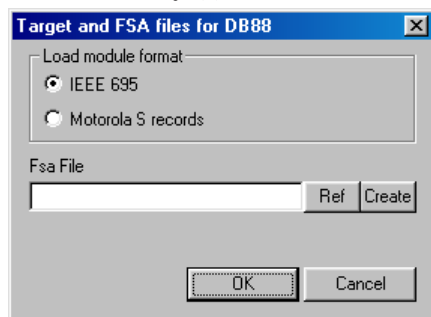
### 13.3.1 起動方法

デバッグを起動する前に、ICE( S5U1C88000H5 )をパーソナルコンピュータに接続し、電源をONしておく必要があります。

デバッグは以下の方法により起動します。

#### ワークベンチからの起動

プロジェクトのビルドを終了後、[Debug]メニューから[DB88 Debugger]を選択、または[DB88]ボタンをクリックします。次のダイアログボックスが表示されます。



デバッグにロードするオブジェクトファイルの種類( IEEE-695またはモトローラS2 )をラジオボタンで選択します。また、ファンクションオプションファイルを[Ref]ボタンで選択するか、[Fsa File]テキストボックスに名称を直接入力します。[Create]ボタンはファンクションオプションジェネレータを起動しますので、オプションファイルの新規作成や修正が行えます。上記の選択/入力後、[OK]ボタンをクリックするとデバッグが起動します。

#### エクスプローラからの起動



DB88.exe

このアイコンをダブルクリックすると、デバッグが起動します。

#### MS-DOSプロンプトからの起動

MS-DOSプロンプトからは次のコマンドでデバッグを起動することができます。

```
db88 ^ [<パラメータファイル名>] ^ [<コマンドファイル名>]
```

^はスペースを示します。

[ ]は省略可能なことを示します。

例: c:\¥epson¥s1c88¥¥db88¥db88 par88xxx.par startup.cmd

注: パラメータファイルとコマンドファイルは拡張子".par"、".cmd"によって認識されます。したがって、コマンドラインのファイル名は拡張子を含めて指定してください。

デバッグが起動すると次のメッセージを[Command]ウィンドウに出力します。

```
DB88 Ver x.xx
Copyright SEIKO EPSON CORP. 2001

Parameter file: xxxxxxxx.par
Initialize..... OK
>
```

各種チェックおよび初期化が終了するとOKを表示してコマンド入力待ち状態になります。ワークベンチから起動した場合は、チェック終了後にオブジェクトファイルのロードも行います。デバッグの各ウィンドウは、前回終了時の位置と大きさで開きます。

注: ICEが自己診断中( DIAGスイッチをONにして起動した場合 )の場合は、診断が終了するまでOKが表示されません。自己診断は開始から終了まで40秒程度かかります。

NGとなった場合は以下の点を確認し、再度デバッグを起動してください。

- USBケーブルは正しく接続されているか
- ICE用USBドライバはインストールされているか
- ペリフェラルボードが正しく装着されているか
- ICEの電源はONになっているか
- ICEがリセット状態になったままではないか

### 13.3.2 終了方法

デバッグを終了するには[File]メニューから[Exit]を選択してください。

[Command]ウィンドウ上でqコマンドを入力することによっても終了します。

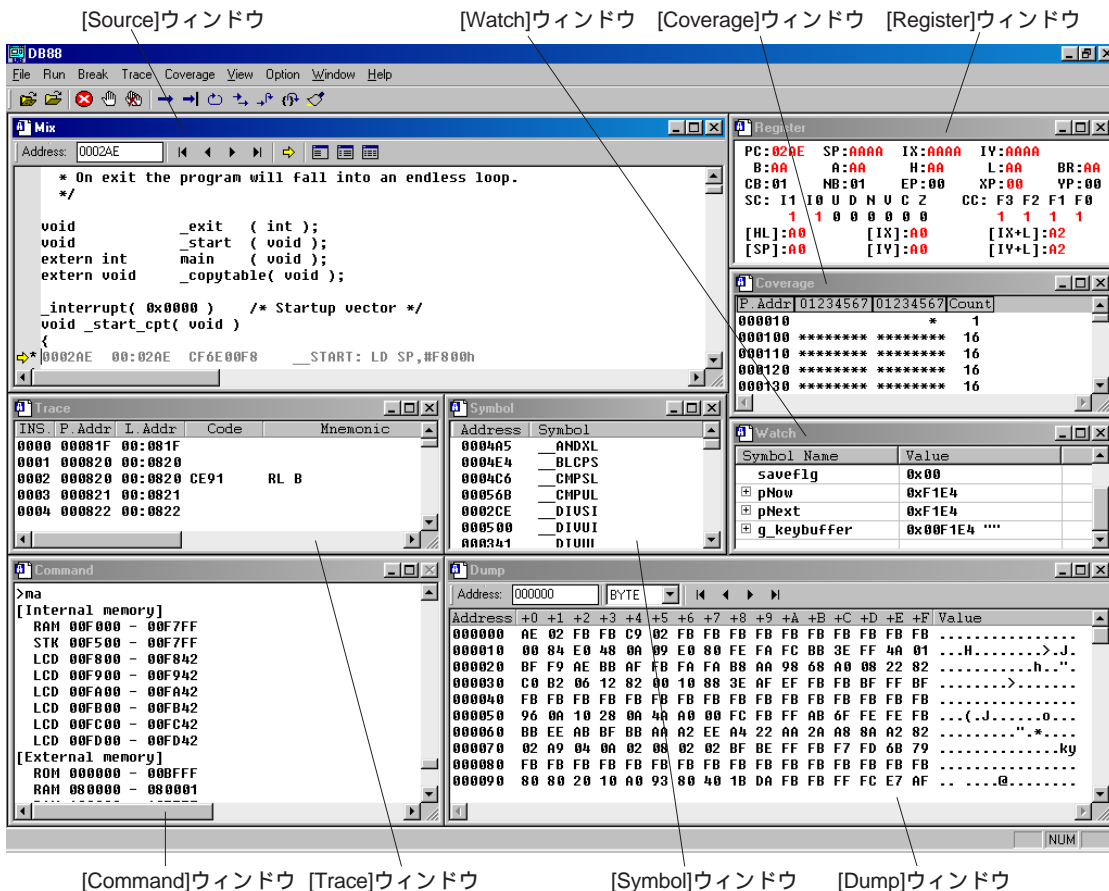
```
>q
```

## 13.4 ウィンドウ

ここでは、デバッガで使用するウィンドウの種類を説明します。

### 13.4.1 ウィンドウの基本構成

デバッガのウィンドウ構成は次のとおりです。



#### 全ウィンドウの共通な操作

##### (1) ウィンドウのオープン/クローズとアクティブ化

[Command]以外のウィンドウはすべて閉じることと開くことができます。

ウィンドウを開くには、[View]メニューからそのウィンドウ名を選択してください。結果を特定のウィンドウに表示するデバッグコマンドを実行した場合も、対応するウィンドウが開きます。

ウィンドウを閉じるには、ウィンドウの[閉じる]ボタンをクリックしてください。

開いているウィンドウは[Window]メニューにリストされます。そこからウィンドウ名を選択することで、そのウィンドウがアクティブになります。ウィンドウ上をクリックすることでも同様です。また、[Ctrl]+[Tab]のキー操作によってもアクティブウィンドウの切り替えが行えます。

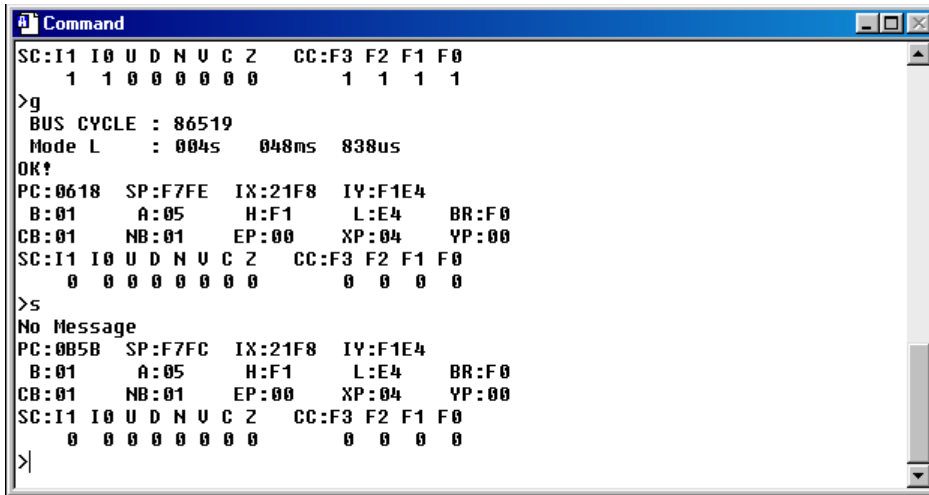
##### (2) サイズの変更と移動

それぞれのウィンドウサイズは、ウィンドウの境界をドラッグすることによって任意の大きさに変更できます。[最小化]ボタン、[最大化]ボタン等も一般のWindowsアプリケーションと同様です。各ウィンドウはタイトルバーをドラッグすることによって、表示位置を変更できます。ただし、サイズ変更、移動ともに、アプリケーションウィンドウの範囲内に限られます。

##### (3) その他

開いているウィンドウは、[Window]メニューの[Cascade]または[Tile]を選択することで整列させることができます。

## 13.4.2 [Command]ウィンドウ



[Command]ウィンドウは以下の目的に使用します。

(1) デバッグコマンドの入力

[Command]ウィンドウにプロンプト">"が表示され、キーボードからのコマンド入力を受け付けます。

(2) メニュー/ツールバーから選択したコマンドの表示

メニューやツールバーからデバッグコマンドを選択して実行させた場合は、そのコマンドラインが[Command]ウィンドウに表示されます。

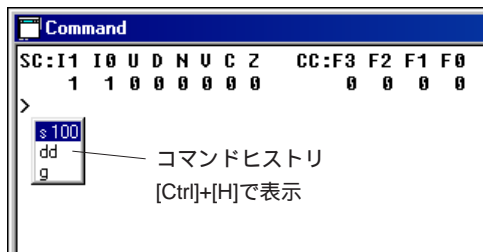
(3) コマンド実行結果の表示

コマンドの実行結果を表示します。ただし、コマンド実行結果の中には、他のウィンドウに表示される内容もあります。これらの内容は対応するウィンドウが開いていれば、その中に表示されます。ウィンドウが閉じている場合は、[Command]ウィンドウに表示されます。

ログファイルへの書き込みを設定中は、その書き込み内容が表示されます。(logコマンド参照)

(4) コマンド履歴の表示

db88はコマンド履歴として、起動後から現在までに実行しているコマンドの中で最新の32個を記憶します(まったく同じコマンドが複数回実行されていた場合はそれらを1つとして数えます)。記憶しているコマンドは、[Command]ウィンドウがアクティブになっている状態で[Ctrl]+[H]キーの入力によって呼び出すことができます。





- 単に[Ctrl]+[H]を入力すると、コマンド履歴がポップアップリストの形式で表示されます。再実行するコマンドをマウスでダブルクリックするか、上下矢印キーで選択して[Enter]を押すと、そのコマンドがプロンプト位置にペーストされ、さらに[Enter]キーを押すことで実行できます。コマンド履歴にコマンドが1個のみ登録されている場合、ポップアップリストは表示されずに、プロンプト位置に直接ペーストされます。
- 文字入力に続けて[Ctrl]+[H]を入力した場合は、以下のいずれかの動作をします。
  - コマンド履歴にその文字(列)で始まるコマンドが複数登録されている場合、それらのコマンドがリストされます。その状態でさらに文字(列)を入力すると、表示されたコマンドの中で、その文字(列)を含む最近実行したコマンドが選択(ハイライト)状態となります。
  - コマンド履歴にその文字(列)で始まるコマンドが1個だけ登録されている場合、そのコマンドが直接プロンプト位置にペーストされます。
  - コマンド履歴にその文字(列)で始まるコマンドが1個もない場合は何も行いません。

たとえば、dd、sy、sの3つのコマンドがコマンド履歴に登録されている場合、

- sを入力後、[Ctrl]+[H]を入力するとsとsyがリスト表示されます。この段階では、最近実行したsコマンドが上に表示され、ハイライト状態となります。
- 上記の操作に続けてyを入力すると、syがハイライト表示されます。
- dを入力後、[Ctrl]+[H]を入力するとプロンプト位置にddがペーストされます。

注: [Command]ウィンドウを閉じることはできません。



### 13.4.3 [Source]ウィンドウ

[Source]ウィンドウはプログラムコードを表示します。以下に示す3種類の表示形式に対応しています。

#### 1. 逆アセンブル表示モード

ロードしたオブジェクトを逆アセンブルしてアドレス、コード、ニーモニックを表示します。[Source]ウィンドウをこの表示モードで開くには[View]メニューから[Source ⇄ Disassemble]を選択します。他のモードで表示中に、逆アセンブル表示モードにするには、上記メニューを選択するか[Source]ウィンドウ上の[Disassemble]ボタンをクリック、あるいはuコマンドを実行します。[Source]ウィンドウがこの表示モードになると、タイトルバーには"Disassemble"と表示されます。この表示モードは読み込んだオブジェクトファイルの種類にかかわらず選択可能です。



[Disassemble]ボタン

#### 2. ソース表示モード

現在のプログラムカウンタアドレスを含むオブジェクトに対応するソースを表示します。ただし、このモードは、ソース表示用のデバッグ情報を含むIEEE-695形式のアブソリュートオブジェクトファイル(.abs)を読み込んだ場合にのみ選択可能です。[Source]ウィンドウをこの表示モードで開くには[View]メニューから[Source ⇄ Source]を選択します。他のモードで表示中に、ソース表示モードにするには、上記メニューを選択するか[Source]ウィンドウ上の[Source]ボタンをクリック、あるいはscコマンドを実行します。また、[Source]ウィンドウが表示されている状態で、Cソースデバッグ情報を含むアブソリュートオブジェクトファイル(.abs)を読み込むと、[Source]ウィンドウは自動的にこのモードになります。この表示モードになると、タイトルバーにはソースファイル名が表示されます。



[Source]ボタン

#### 3. ミックス表示モード

ソースと逆アセンブル内容(アドレス、コード、ニーモニック)を上下に対応させて表示します。ただし、このモードは、ソース表示用のデバッグ情報を含むIEEE-695形式のアブソリュートオブジェクトファイル(.abs)を読み込んだ場合にのみ選択可能です。[Source]ウィンドウをこの表示モードで開くには[View]メニューから[Source ⇄ Mix]を選択します。他のモードで表示中に、ソース表示モードにするには、上記メニューを選択するか[Source]ウィンドウ上の[Mix]ボタンをクリック、あるいはmコマンドを実行します。[Source]ウィンドウがこの表示モードになると、タイトルバーには"Mix"と表示されます。



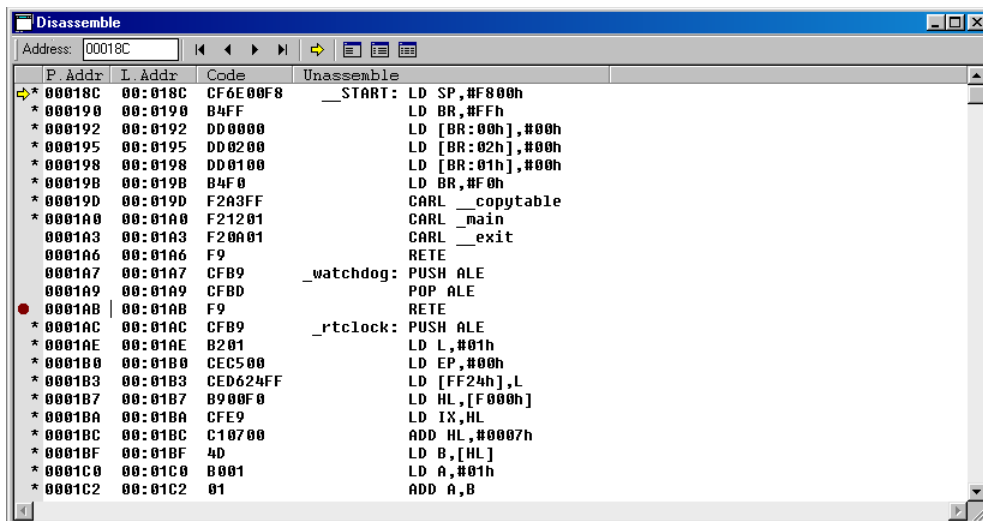
[Mix]ボタン

#### ソースの表示

ソースは、ソース表示用のデバッグ情報を含むIEEE-695形式のアブソリュートオブジェクトファイルを読み込んだ場合にのみ表示可能です。

また、オブジェクトファイルのデバッグ情報(ソースファイルの相対パス情報)からソースファイルを探して読み込みますので、ソースファイルを削除あるいは移動した(オブジェクトファイルからの相対位置が変わった)場合、ソースは表示されません。この場合、ソース表示モードではウィンドウが空白となり、ミックス表示モードでは逆アセンブル内容のみが表示されます。

## 逆アセンブル表示モード



逆アセンブル表示モード時の[Source]ウィンドウの機能を以下に示します。

## (1) プログラムコードの表示

物理/論理アドレス、オブジェクトコード、逆アセンブル内容を表示します。

プログラムの表示箇所は、スクロール以外に以下の方法で変更できます。

- [Address]テキストボックスにアドレスを入力またはuコマンドでアドレスを指定します。そのアドレスを先頭に表示します。



プログラムの先頭、または最後を表示します。

現在のウィンドウサイズで1ページ前、または後を表示します。

現在のPCアドレスから表示します。

注: S1C88 Familyプロセッサのニーモニックは可変長のため、上方向にスクロールした場合、実際のコードと異なる逆アセンブルを行うことがあります。

## 表示の更新

プログラムをロードして実行(g、gr、s、n、se、rstコマンド)するか、プログラムメモリの内容を変更(de、df、dmコマンド)すると表示内容が更新されます。この場合、現在のPCが示すアドレスがウィンドウ内に表示されるように表示が更新されます。

## (2) カレントPCの表示

現在のPC(プログラムカウンタ)が示すアドレスの行は、先頭に黄色の矢印を表示します。

## (3) PCブレークポイントの表示

ブレークポイントに設定されたアドレスの行は、先頭に赤のマークを表示します。

## (4) カバレッジ情報の表示

カバレッジ機能により、実行したアドレスの先頭に\*が表示されます。

## (5) カーソル位置でブレーク設定

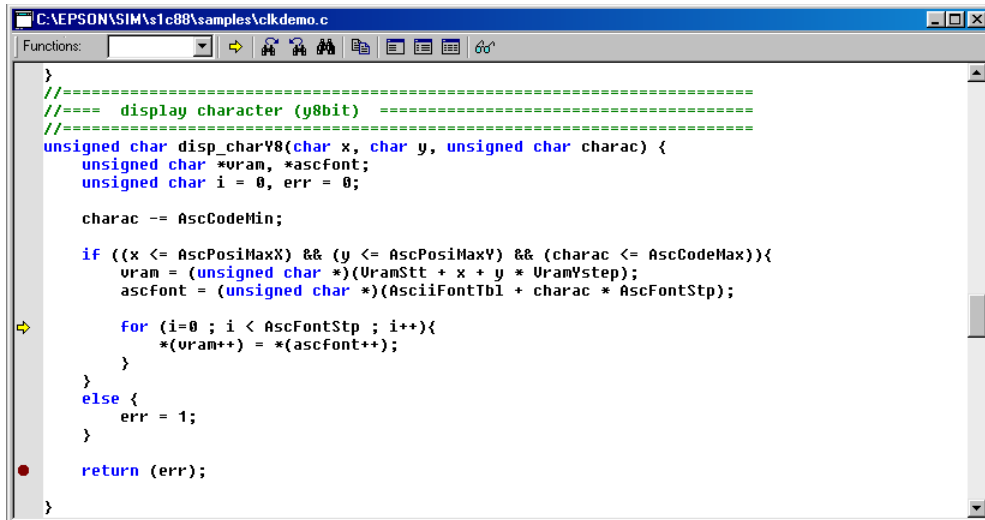


ブレークポイントを設定したいアドレスの行にカーソルを置きます。そこで、[Break]ボタンをクリックすると、そのアドレスがPCブレークポイントに設定されます(行内をダブルクリックすることによっても設定可能)。PCブレークポイントに設定されたアドレスの行で同じ操作をすると、そのブレークポイントは解除されます。ブレークポイントは複数のアドレスに設定可能です。



[Go to Cursor]ボタンをクリックすると、プログラムが現在のPCから実行を開始し、カーソルのある行の実行後にブレークします。

## ソース表示モード



ソース表示モード時の[Source]ウィンドウの機能を以下に示します。

## (1) プログラムコードの表示

ソースを表示します。自動的に表示されるのは、現在のPC プログラムカウンタ が示すアドレスを含むソースです。

コメントは緑、予約語は青、その他は黒で表示されます。

タブ幅は4文字に固定です。

プログラムの表示箇所は、スクロール以外に以下の方法で変更できます。



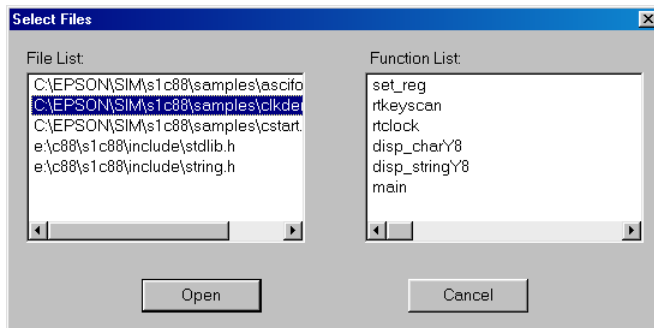
- ・ [Functions]プルダウンリストから関数名を選択します。  
その関数の先頭から表示します。



- ・ [Current PC]ボタンをクリックします。現在のPCアドレスから表示します。



- ・ 他のソースファイルを表示させるには、[Source Files]ボタンをクリックして次のダイアログボックスを表示させ、リストされているソースの中から表示させるものを選択します。



## 表示の更新

プログラムをロードして実行 (g、gr、s、n、se、rstコマンド) 後、実行を中断すると表示内容が更新されます。この場合、現在のPCアドレスを含むソースがウィンドウ内に表示されます。このとき、対応するソースが見つからなかった場合は上図の[Select Files]ダイアログボックスが表示され、表示するソースの選択が必要になります。

## (2) カレントPCの表示

現在のPC(プログラムカウンタ)が示すアドレスを含むソース行は、先頭に黄色の矢印を表示します。

## (3) PCブレークポイントの表示

ブレークポイントに設定されたアドレスを含むソース行は、先頭に赤のマークを表示します。

## (4) カーソル位置でブレーク設定



ブレークポイントを設定したいソースの行にカーソルを置きます。そこで、[Break]ボタンをクリックすると、そのソース行(ソースに対応する有効なオブジェクトコードの先頭のアドレス)がブレークポイントに設定されます(行内をダブルクリックすることによっても設定可能)。PCブレークポイントに設定されたソース行で同じ操作をすると、そのブレークポイントは解除されます。ブレークポイントはソース行単位で複数設定可能です。ただし、実コードを持たないソース行にはブレークポイントを設定することはできません。また、通常はコードが生成されるCステートメントでも、Cコンパイラの最適化によりコードが生成されないこともあります。ブレークポイントに設定できない場合はミックス表示モードに切り換えて確認してください。



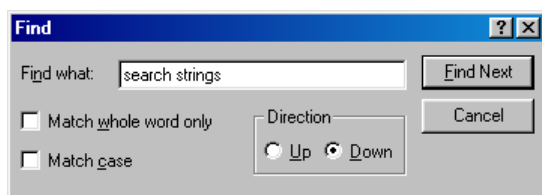
[Go to Cursor]ボタンをクリックすると、プログラムが現在のPCから実行を開始し、カーソルのある行でブレークします。この場合も、カーソルを実コードを持つソース行に置く必要があります。実コードがない場合、[Go to Cursor]の操作は無効です。

## (5) 文字列の検索

ソース表示モードでは、[Source]ウィンドウに以下の検索用のボタンが表示され、文字列の検索を行うことができます。



[Find]ボタンをクリックすると、検索文字列を指定するダイアログボックスが表示されます。



[Find what]エディットボックスに検索文字列を入力して[Find Next]ボタンをクリックすると、現在のカーソル位置から[Source]ウィンドウの下方向(プログラムの後方)に検索を行います。指定文字列が[Source]ウィンドウ内で見つかったら、その文字列を選択状態にします。

そこでさらに[Find Next]ボタンをクリックすると、その位置から次の検索を開始します。ウィンドウの上方向(プログラムの前方)に検索したい場合は、[Direction]の[Up]ボタンを選択してください。指定文字列に完全に一致するもののみを検索する場合は[Match whole word only]チェックボックスを、大文字と小文字を区別して検索する場合は[Match case]チェックボックスを、[Find Next]ボタンをクリックする前に選択してください。



[Source]ウィンドウ内で文字列をドラッグして選択し、[Source]ウィンドウ上の[Find Next]ボタンをクリックすると、その選択位置から[Source]ウィンドウの下方向(プログラムの後方)にその文字列の検索を行います。文字列が見つかったら、新たに見つかった方を選択状態にします。そこでさらに[Find Next]ボタンをクリックすると、その位置から次の検索を開始します。この検索においては、大文字と小文字は区別されません。また完全に一致しない文字列も検索の対象です。



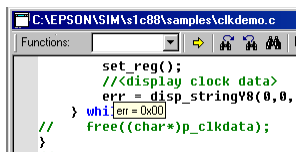
[Find Previous]ボタンは、検索方向がウィンドウの上側(プログラムの前方)に変わる以外、上記の[Find Next]ボタンと同じ機能です。

## (6) [Watch]ウィンドウへのシンボルの登録



ウィンドウ内のシンボル名をドラッグして選択(反転表示)し[Watch]ボタンをクリックすると、そのシンボルが[Watch]ウィンドウのシンボルリストに登録されます。その後、[Watch]ウィンドウでそのシンボルの値を確認することができます。

## (7) 変数の値を表示



表示されているソース中の変数名の上にマウスカーソルを置く(クリックは不要) 変数の値(ポインタ変数の場合はアドレス)が表示されます。(signed/unsigned)int/long/short は10進数で、アドレス、構造体、共用体は16進数で表示されます。構造体のメンバの値を表示するには、変数名をマウスで選択する必要があります。配列の要素の場合もマウスで選択してください。スコープを外れた変数は表示されません。

## ミックス表示モード

```

//=====
//==== display_string(y8bit) =====
//=====
unsigned char disp_stringY8(char x, char y, unsigned char *string) {
*0003B5 00:03B5 CF6A0400 _disp_stringY8: SUB SP,#0004h
*0003B9 00:03B9 48                                LD B,A
*0003BA 00:03BA CEC600                                LD XP,#00h
*0003BD 00:03BD CFFA                                LD IX,SP
*0003BF 00:03BF CE5402                                LD [IX+02h],L
  unsigned char err = 0;
*0003C2 00:03C2 B200                                LD L,#00h

  while ((*string != NULL) || err !=0) {
*0003C4 00:03C4 F133                                JRS 33h
    err = disp_charY8(x,y,*string);
*0003C6 00:03C6 CF7700                                LD [SP+00h],IV
*0003C9 00:03C9 CEC700                                LD VP,#00h
*0003CC 00:03CC 5F                                LD H,[IV]
*0003CD 00:03CD CEC600                                LD XP,#00h
*0003D0 00:03D0 CFFA                                LD IX,SP
*0003D2 00:03D2 CE4C03                                LD [IX+03h],B
*0003D5 00:03D5 CEC700                                LD VP,#00h
*0003D8 00:03D8 CFFE                                LD IV,SP
*0003DA 00:03DA CE5102                                LD L,[IV+02h]
  }
}

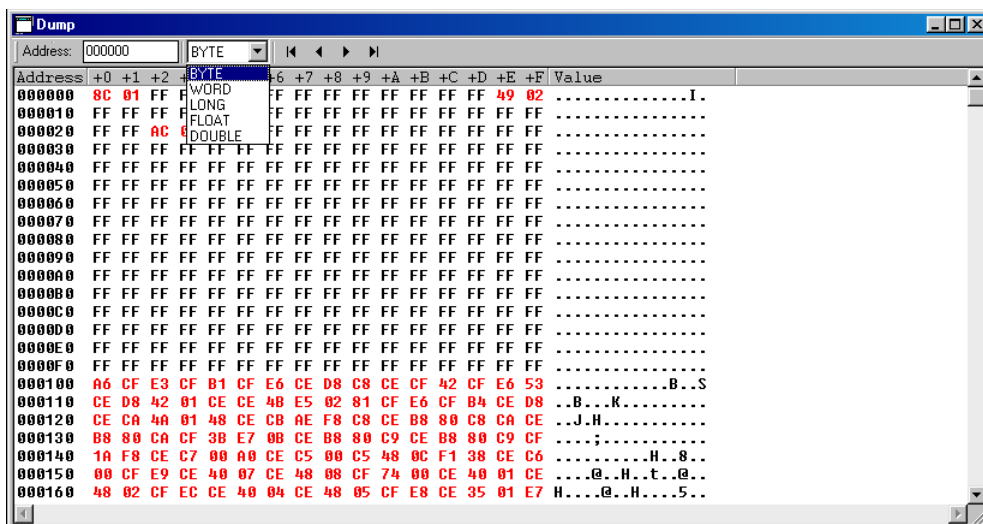
```

ミックス表示モードの機能は逆アセンブル表示モードと同様です。違いは各ソース行と対応するオブジェクトコードの逆アセンブル内容(物理/論理アドレス、オブジェクトコード、ニーモニック)が上下に並んで表示されることのみです。ただし、ミックス表示モードはデバッグ情報を含むIEEE-695形式のアブソリュートオブジェクトファイル(.abs)を読み込んだ場合のみ選択可能です。

表示されたソース行に対しては、ブレーク設定等の操作は一切行えません。各種表示、操作は逆アセンブル表示内容に対してのみ有効です。ミックス表示モードが対応している機能については、逆アセンブル表示モードの説明を参照してください。

ソース行は黒、逆アセンブル内容はグレーで表示されます。

## 13.4.4 [Dump]ウィンドウ



## (1) メモリ内容の表示

データメモリのダンプ結果を16進数で表示します。

デフォルトではバイト単位に表示されますが、プルダウンボックスでその他のサイズにも変更可能です。

メモリの表示箇所は、スクロール以外に以下の方法で変更できます。

- [Address]テキストボックスにアドレスを入力またはddコマンドでアドレスを指定  
そのアドレスを先頭に表示します。



メモリ領域の先頭、または最後を表示します。

現在のウィンドウサイズで1ページ前、または後を表示します。

## 表示の更新

メモリ内容をコマンド (de、df、dmコマンド) で変更すると、[Dump]ウィンドウの表示内容が更新されます。また、プログラムを実行 (g、gr、s、n、se、rstコマンド) した場合も更新されます。

これ以外で、最新の内容を表示させるには、ddコマンドを実行するか、垂直スクロールバーをクリックしてください。

プログラムの実行を中断すると、実行前から変更された値は赤で表示されます。

## (2) データメモリ内容の直接変更

[Dump]ウィンドウ上で、データメモリを直接変更することができます。変更するデータの直前にカーソルを置くか、データをダブルクリック後、16進の数値 (0~9、a~f) を入力してください。そのアドレスのデータが変更されます。カーソルは次のアドレスのデータに移動し、連続的なデータ変更を可能にしています。

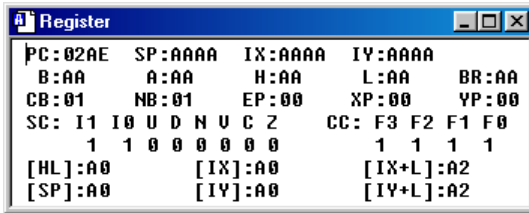
## (3) 10進データの表示

[BYTE]、[WORD]、[LONG]で表示中にマウスカーソルをデータの上に重ねると (クリックは不要)、10進データ (signed int/unsigned int) が表示されます。[BYTE]の場合はビットデータも表示されます。

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8
000100	CE	BD	00	E6	08	CE	80	CE	91
000110	CF	B1	CF	E6	CE	D8	C8	CE	CF
000120	01	CE	CE	4B	E6	11001110	CF	E6	
000130	01	48	CE	CB	uint -50	C7	00		
000140	F1	38	CE	C6	00	CF	E9	CE	40



### 13.4.5 [Register]ウィンドウ



#### (1) レジスタ内容の表示

S1C88 CPU内の全レジスタおよびコンディションフラグの内容、[HL]、[SP]、[IX]、[IY]、[IX+L]、[IY+L]で指定されるメモリの内容を表示します。

#### 表示の更新

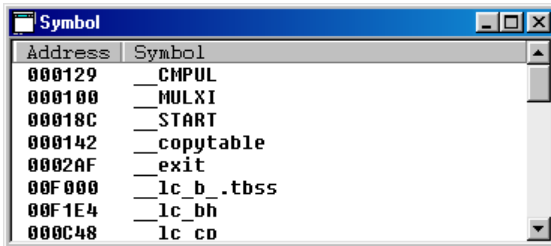
レジスタダンプ時 (rdコマンド)、レジスタデータ変更時 (rsコマンド)、CPUリセット時 (rstコマンド)、プログラムの実行 (g、gr、s、n、seコマンド) 終了後に更新されます。

プログラムの実行を中断すると、実行前から変更された値は赤で表示されます。

#### (2) レジスタ内容の直接変更

[Register]ウィンドウ上で、レジスタの内容を直接変更することができます。変更するデータを選択 (ハイライト) 後、16進の数値 (0~9、a~f) を入力し [Enter] キーを押してください。そのレジスタの内容が変更されます。

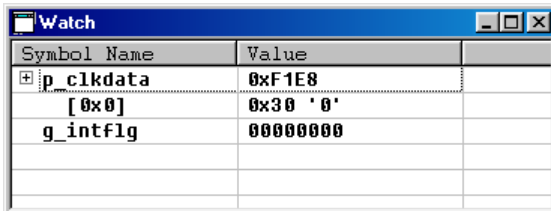
### 13.4.6 [Symbol]ウィンドウ



シンボル情報が読み込まれている場合に、シンボルの一覧を表示します。デフォルトではアルファベット順に表示されます。"sy /a" コマンドにより、アドレス順に表示させることもできます。

\* モトローラS2形式のプログラムファイルをロードした場合、シンボルファイル (.sy) が自動的に読み込まれます。ただし、そのためには、ターゲットプログラムと同じ名称のシンボルファイルをターゲットプログラムと同じディレクトリに用意しておく必要があります。IEEE-695形式のプログラムファイルの場合、シンボルファイルは読み込まれません。

### 13.4.7 [Watch]ウィンドウ



wコマンドまたは[Source]ウィンドウの[Watch]ボタンで登録したシンボルの名称と現在の値を表示します。値はwコマンドで指定した形式で表示されます。シンボルが配列、構造体、共用体の場合、+アイコンが表示されます。それをクリックすることにより、メンバーが階層表示されます。

また、右クリックにより表示されるメニューで、登録したシンボルの削除や表示形式変換 (16進数、10進数など) ができます。ただし、変換できるのはint、char、long、shortなどの型のみで、アドレスは16進数表記固定です。なお、このウィンドウを使用したシンボル表示は、指定シンボルの情報を含むIEEE-695形式のアブソリュートオブジェクトファイル (.abs) を読み込んだ場合のみ可能です。

注: コンパイル時に-O1オプションを指定すると、コードの最適化により不要なシンボルが削除され、シンボル情報が生成されない場合があります。そのようなシンボルは、[Watch]ウィンドウに登録することはできません。

#### 表示の更新

プログラムの実行 (g、gr、s、n、seコマンド) 終了後に更新されます (デフォルト)。このウォッチ機能については[Run | Setting...]メニューコマンドで表示されるダイアログボックスによって、プログラム実行中の表示更新を設定することができます ("13.8.4 プログラムの実行" 参照)。

### 13.4.8 [Trace]ウィンドウ

INS	P Addr	I Addr	Code	Mnemonic	BA	HL	IX	IY	SP	BR	EP	XP	YP	SC	CC	Memory
0217	000499	01:0499	93	INC IX	3E84	F828	F828	F060	F7F3	F0	00	00	00	00	-N-C-	0000
0218	00049A	01:049A	92	INC IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	-N-C-	0000
0219	00049B	01:049B	CF7601	LD [SP+01h],IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	-N-C-	0000 MW:[00F7F4]=29 MW:[00F7
0220	00049E	01:049E	8001	LD A,#01h	3E01	F828	F829	F060	F7F3	F0	00	00	00	00	-N-C-	0000
0221	0004A0	01:04A0	A6	PUSH IP	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	-N-C-	0000 MW:[00F7F2]=00 MW:[00F7
0222	0004A1	01:04A1	CEC600	LD XP,#00h	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	-N-C-	0000
0223	0004A4	01:04A4	CFFA	LD IX,SP	3E01	F828	F7F1	F060	F7F1	F0	00	00	00	00	-N-C-	0000
0224	0004A6	01:04A6	CE402	LD B,[IX+02h]	0401	F828	F7F1	F060	F7F1	F0	00	00	00	00	-N-C-	0000 MR:[00F7F3]=04 MR:[00F7
0225	0004A9	01:04A9	AE	POP IP	0401	F828	F7F1	F060	F7F3	F0	00	00	00	00	-N-C-	0000 MR:[00F7F1]=00 MR:[00F7
0226	0004AA	01:04AA	01	ADD A,B	0405	F828	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0227	0004AB	01:04AB	50	LD L,A	0405	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0228	0004AC	01:04AC	B105	LD B,#05h	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0229	0004AE	01:04AE	42	LD A,L	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0230	0004AF	01:04AF	A6	PUSH IP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F2]=00 MW:[00F7
0231	0004B0	01:04B0	CEC600	LD XP,#00h	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0232	0004B3	01:04B3	CFFA	LD IX,SP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0233	0004B5	01:04B5	CE402	LD [IX+02h],A	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F3]=05
0234	0004B8	01:04B8	AE	POP IP	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000 MR:[00F7F1]=00 MR:[00F7
0235	0004B9	01:04B9	3A80	XOR A,#00h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-N----	0000
0236	0004BB	01:04BB	CEB800	XOR B,#00h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-N----	0000
0237	0004BE	01:04BE	31	CP A,B	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0238	0004BF	01:04BF	CEE0CB	JRS LT,CBh	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0239	0004C2	01:04C2	F105	JRS 05h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000

mdコマンドでトレース機能をONに設定すると、それ以降のプログラム実行時にトレース情報を採取します。トレース用のバッファは8192命令分の容量があり(容量を越えた分は先頭から上書き) この中に記録した情報を[Trace]ウィンドウに表示することができます。

表示されるトレース内容は、次のとおりです。

- ・実行命令番号
- ・フェッチコードと逆アセンブル内容
- ・レジスタおよびコンディションフラグの内容
- ・メモリのアクセス内容(R/W、アドレス、データ)

[Trace]ウィンドウは、tsコマンドによるトレースデータの検索結果の表示にも使用します。

#### 表示の更新

ターゲットプログラムの実行により、[Trace]ウィンドウの内容はクリアされます。プログラム実行を中断すると、[Trace]ウィンドウはトレースバッファの内容を表示します。

### 13.4.9 [Coverage]ウィンドウ

P Addr	01234567	01234567	Count
000010	*		1
000100	*****	*****	16
000110	*****	*****	16
000120	*****	*****	16
000130	*****	*****	16
000140	*****	*****	16
000150	*****	*****	16
000160	*****	*****	16

ICEが取得したカバレレッジ情報(実行アドレス情報)を読み出して表示します。

表示内容は16バイト/行のメモリマップです。各行先頭の数値は物理アドレス(16進数)で、そのアドレスから始まる16バイトの中で実行したアドレスがアステリクス(\*)で示されます。Countの数値は16バイトの中で実行したアドレスの数です。

[Coverage]ウィンドウはプログラム実行後も自動的に更新されません。[Coverage]メニューから[Coverage]を選択するか、cvコマンドを実行する必要があります。また[Coverage]メニューから[Coverage Clear]を選択すると、[Coverage]ウィンドウの表示もクリアされます。

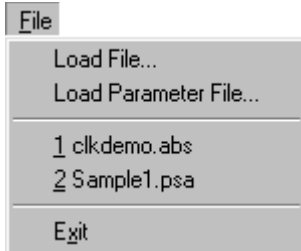


## 13.5 メニュー

ここでは、メニューバーの概要を説明します。

デバッグのメニューバーには9つのメニュー項目があり、頻繁に使用するコマンドが設定されています。

### [File]メニュー



リストされているファイル名は最近ロードしたファイルです。ここからの選択でも、そのファイルを開くことができます。

#### [Load File...]

IEEE-695形式のアブソリュートオブジェクトファイル、モトローラS2形式のプログラムファイルまたはファンクションオプションファイルを読み込みます。この選択はlfコマンドを実行するのと同じ働きがあります。

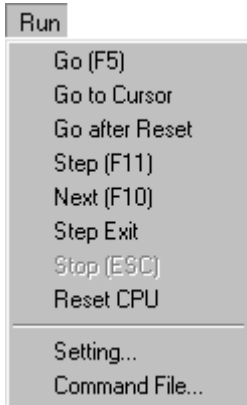
#### [Load Parameter File...]

パラメータファイルを読み込みます。この選択はparコマンドを実行するのと同じ働きがあります。

#### [Exit]

デバッグを終了します。この選択はqコマンドを実行するのと同じ働きがあります。

### [Run]メニュー



#### [Go]

現在のPQ(プログラムカウンタ)からターゲットプログラムを実行します。[F5]キーでも実行可能です。この選択はgコマンドを実行するのと同じ働きがあります。

#### [Go to Cursor]

現在のPCから、[Source]ウィンドウのカーソル位置(その行のアドレス)までターゲットプログラムを実行します。このメニュー項目を選択するには、[Source]ウィンドウを開き、ブレークするアドレスの行をクリックしておく必要があります。

#### [Go after Reset]

CPUをリセット後、リセットベクタをフェッチしてターゲットプログラムを実行します。この選択はgrコマンドを実行するのと同じ働きがあります。

#### [Step]

現在のPCからターゲットプログラムを1ステップ実行します。[F11]キーでも実行可能です。この選択はsコマンドを実行するのと同じ働きがあります。

#### [Next]

現在のPCからターゲットプログラムを1ステップ実行します。実行命令がcars、carl、call、int命令の場合は、次のアドレスにリターンするまでを1ステップとみなし、それらのサブルーチンのステップをすべて実行します。[F10]キーでも実行可能です。この選択はnコマンドを実行するのと同じ働きがあります。

#### [Step Exit]

現在のPCからターゲットプログラムを実行します。開始位置がサブルーチン内の場合、親ルーチンにリターンしたところで実行を中断します。この選択はseコマンドを実行するのと同じ働きがあります。

#### [Stop]

実行中のプログラムを強制的に中断します。[ESC]キーも同様です。

**[Reset CPU]**

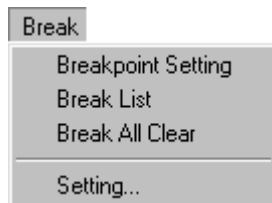
CPUをリセットします。この選択はrstコマンドを実行するのと同じ働きがあります。

**[Setting...]**

実行関連オプション(実行モニタ間隔、シングルステップ時の割り込みモード、ウォッチ更新モード、実行時間測定単位)をダイアログボックスを使用して設定します。

**[Command File...]**

コマンドファイルを読み込み、記述されているコマンドを実行します。この選択はcom、cmwコマンドを実行するのと同じ働きがあります。

**[Break]メニュー****[Breakpoint Setting]**

PCブレークポイントやデータブレーク条件をダイアログボックスを使用して設定/解除します。この選択はbp、bpa、ba、bdコマンドを実行するのと同じ働きがあります。

**[Break List]**

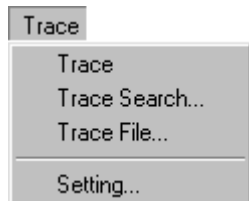
設定されているすべてのブレーク条件を表示します。この選択はblコマンドを実行するのと同じ働きがあります。

**[Break All Clear]**

すべてのブレーク条件を解除します。この選択はbacコマンドを実行するのと同じ働きがあります。

**[Setting...]**

ソフトウェアブレーク有効領域やシーケンシャルブレークモードをダイアログボックスを使用して設定します。

**[Trace]メニュー****[Trace]**

[Trace]ウィンドウをアクティブにして、トレースデータバッファ内のトレース情報を表示します。この選択はtdコマンドを実行するのと同じ働きがあります。

**[Trace Search...]**

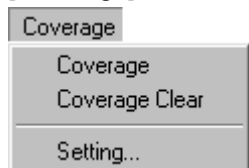
トレースデータバッファ内のトレース情報を検索します。検索条件はダイアログボックスで指定します。この選択はtsコマンドを実行するのと同じ働きがあります。

**[Trace File...]**

[Trace]ウィンドウに表示したトレース情報の指定範囲をファイルに保存します。この選択はtfコマンドを実行するのと同じ働きがあります。

**[Setting...]**

トレースモードをダイアログボックスを使用して選択します。

**[Coverage]メニュー****[Coverage]**

[Coverage]ウィンドウをアクティブにして、ICEに取得されているカバレッジ情報を表示します。この選択はcvコマンドを実行するのと同じ働きがあります。

**[Coverage Clear]**

ICE内のカバレッジ情報と[Coverage]ウィンドウ内の表示をクリアします。この選択はcvcコマンドを実行するのと同じ働きがあります。

**[Setting...]**

カバレッジオプション(取得エリア、取得対象)をダイアログボックスを使用して選択します。

## [View]メニュー

View	[Command]	
Command	[Command]ウィンドウをアクティブにします。	
Source	[Source - Disassemble]	
Dump	[Source]ウィンドウを開いてアクティブにします。[Source]ウィンドウは逆アセンブル表示モードでプログラムを表示します。	
Register	[Source - Source]	
Trace	[Source]ウィンドウを開いてアクティブにします。[Source]ウィンドウはソース表示モードでプログラムを表示します。	
Coverage	[Source - Mix]	
Symbol	[Source]ウィンドウを開いてアクティブにします。[Source]ウィンドウはミックス表示モードでプログラムを表示します。	
Watch		
✓ Toolbar		
✓ Status Bar		

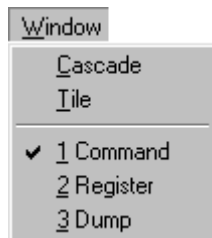
  

[Dump]	[Dump]ウィンドウを開いてアクティブにします。[Dump]ウィンドウはメモリの内容をメモリの先頭から表示します。
[Register]	[Register]ウィンドウを開いてアクティブにします。[Register]ウィンドウは各レジスタの内容を表示します。
[Trace]	[Trace]ウィンドウを開いてアクティブにします。[Trace]ウィンドウはトレースデータバッファの内容を表示します。
[Coverage]	[Coverage]ウィンドウを開いてアクティブにします。[Coverage]ウィンドウはICEに取得されているカバレッジ情報を表示します。
[Symbol]	[Symbol]ウィンドウを開いてアクティブにします。シンボル情報が読み込まれていれば、その内容を表示します。
[Watch]	[Watch]ウィンドウを開いてアクティブにします。シンボルが登録されていれば、その内容を表示します。
[Toolbar]	ツールバーの表示/非表示を切り換えます。
[Status Bar]	ステータスバーの表示/非表示を切り換えます。

## [Option]メニュー

Option	[Log...]
Log...	ログ出力のON/OFFを切り換えます。この選択はlogコマンドを実行するのと同じ働きがあります。
Record...	[Record...]
Setting...	実行コマンドのファイルへの記録を制御します。この選択はrecコマンドを実行するのと同じ働きがあります。
	[Setting...]
	システムオプション( エミュレーションクロック、ファームウェアクロック、自己書き換えチェック機能、cmwのウェイト時間 )をダイアログボックスを使用して設定します。

## [Window]メニュー



### [Cascade]

開いているウィンドウを斜めに整列させます。

### [Tile]

開いているウィンドウを縦に整列させます。

このメニューには、現在開いているウィンドウ名が表示されます。いずれかを選択すると、そのウィンドウがアクティブになります。

## [Help]メニュー



### [About DB88...]

デバッガのアバウトダイアログボックスを表示します。

## 13.6 ツールバー

ここでは、ツールバーの概要を説明します。

デバッグのツールバーには12個のボタンがあり、頻繁に使用するコマンドが設定されています。



クリックすることにより、指定の機能を実行します。



[Load File]ボタン

IEEE-695形式のアブソリュートオブジェクトファイル、モトローラS2形式のプログラムファイルまたはファンクションオプションファイルを読み込みます。この選択はlfコマンドを実行するのと同じ働きがあります。



[Load Parameter]ボタン

パラメータファイルを読み込みます。この選択はparコマンドを実行するのと同じ働きがあります。



[Key Break]ボタン

ターゲットプログラムの実行を強制的にブレークします。この機能はプログラムが永久ループ状態になった場合などにブレークさせることができます。



[Break]ボタン

[Source]ウィンドウ上のカーソルが置かれた行のアドレスに対し、ブレークポイントの設定と解除を行うのに使用します。[Source]ウィンドウが開いている時のみ有効です。



[Break All Clear]ボタン

すべてのブレーク条件を解除します。この選択はbacコマンドを実行するのと同じ働きがあります。



[Go]ボタン

現在のPC(プログラムカウンタ)からターゲットプログラムを実行します。この選択はgコマンドを実行するのと同じ働きがあります。



[Go to Cursor]ボタン

現在のPCから、[Source]ウィンドウのカーソル位置(その行のアドレス)までターゲットプログラムを実行します。このボタンを選択するには、[Source]ウィンドウを開き、ブレークするアドレスの行をクリックしておく必要があります。



[Go after Reset]ボタン

CPUをリセット後、リセットベクタをフェッチしてターゲットプログラムを実行します。この選択はgrコマンドを実行するのと同じ働きがあります。



[Step]ボタン

現在のPCからターゲットプログラムを1ステップ実行します。この選択はsコマンドを実行するのと同じ働きがあります。



[Next]ボタン

現在のPCからターゲットプログラムを1ステップ実行します。実行命令がcars、carl、call、int命令の場合は、次のアドレスにリターンするまでを1ステップとみなし、それらのサブルーチンのステップをすべて実行します。この選択はnコマンドを実行するのと同じ働きがあります。



[Step Exit]ボタン

現在のPCからターゲットプログラムを実行します。開始位置がサブルーチン内の場合、親ルーチンにリターンしたところで実行を中断します。この選択はseコマンドを実行するのと同じ働きがあります。



[Reset CPU]ボタン

CPUをリセットします。この選択はrstコマンドを実行するのと同じ働きがあります。

## 13.7 コマンド実行方法

デバッグ機能はすべてデバッグコマンドによって実行できます。ここでは、コマンドを実行させる方法を説明します。

### 13.7.1 コマンドのキーボード入力

[Command]ウィンドウを選択してください([Command]ウィンドウ上をクリック)。その中の最終行にプロンプト">"が表示され、その後にカーソルが点滅していればコマンドが入力できる状態にあります。そこに、デバッグコマンドを入力してください。コマンドは大文字でも、小文字でも受け付けます。

#### コマンド入力の一般形

> コマンド [パラメータ [パラメータ... パラメータ]] ↵

- ・コマンドとパラメータ間にはスペースが必要です。
- ・パラメータ間にはスペースが必要です。

入力ミスの修正には矢印キー、[Back Space]キー、[Delete]キーが使用できます。

最後に[Enter]キーを入力すると、そのコマンドを実行します(ガイダンス付きのコマンドは、表示に従って必要なデータを入力した時点で実行されます)。

入力例:

>g↵ (コマンドのみの入力)

>lf test.abs↵ (コマンドとパラメータの入力)

#### ガイダンス付きのコマンド入力

パラメータが指定されないと実行できないコマンドや、既存のデータを変更するコマンドはコマンドのみの入力ではガイダンスモードとなります。ガイダンスが表示されますので、そこにパラメータを入力してください。

入力例:

>cmw↵

File name ? :test.cmd↵ ...ガイダンスに従ってデータ(アンダーライン部)を入力  
:

#### ・パラメータ入力が必要なコマンド

上記例のcmwコマンドはコマンドファイルを読み込むコマンドです。このような、パラメータの入力が必要なコマンドはパラメータを入力後、[Enter]キーを押すと実行されます。複数のパラメータを持つコマンドでは、次のガイダンスが表示されますので、最後のパラメータまで順次入力してください。いずれかのガイダンスで[Enter]キーのみを入力すると、そのコマンドはキャンセルされ実行されません。

#### ・既存のデータを確認して置き換えるコマンド

メモリやレジスタを1つずつ書き換えるコマンドではガイダンスをスキップ(その内容を変更しない)1つ前のガイダンスに戻す、および途中で入力を終了することができます。

[Enter]キー 入力をスキップ

[^]キー 1つ前のガイダンスに戻す

[q]キー 入力を終了する

入力例:

```
>de↵ ...メモリを変更するコマンド
Data enter address ? :00ff00↵ ...開始アドレスを入力
00FF00 A:1↵ ...00ff00H番地を1に変更
00FF01 A:↵ ...1つ前のアドレスに戻す
00FF00 1:0↵ ...00ff00H番地の入力をし直す
00FF01 A:↵
00FF02 A:↵
00FF01 A:q↵ ...入力を終了
>
```

### パラメータの数値データ形式

パラメータとして入力する数値は、ほとんどのコマンドが16進数のみを受け付けます。  
ただし、一部のコマンドのパラメータは、10進数または2進数で指定します。  
数値として有効な文字は次のとおりです。

16進数: 0~9、a~f、A~F、\*

10進数: 0~9

2進数: 0、1、\*

( \*はデータパターン指定でマスクするビットに使用します。 )

### シンボルによる指定

シンボル情報を含んだIEEE-695形式のアブソリュートオブジェクトファイル(.abs) またはシンボルファイル(.sy) が読み込まれている場合、シンボルを使用してアドレスを指定することができます。

入力例:

>u Main┘ ...プログラムをMainというラベルから表示

- \* シンボルファイル(.sy) はターゲットプログラム( モトローラS2 )のロード時に自動的に読み込まれます。  
ただし、そのためには、ターゲットプログラムと同じ名称のシンボルファイルをターゲットプログラムと同じディレクトリに用意しておく必要があります。IEEE-695形式のプログラムファイルを指定した場合は読み込まれません。

注: ・ 指定したシンボルがない場合、db88はその指定文字列を16進数として扱います( ABCなど )。ただし、文字列が16進数の指定文字以外を含んでいる場合はエラーとなります。

- ・ Cソースのコンパイル時に-O1オプションを指定すると、ソースに記述されているシンボルがコードの最適化によって実際には使用されない場合が発生します。この場合、-gオプションを指定してもそのシンボルのデバッグ情報は.absファイルに出力されません。

例: 

```
int x,y,xy;
x = GLOBAL_X * 100;
y = GLOBAL_Y * 100;
xy = x * y;
```

この例では最適化により変数xyは存在しなくなりますので、デバッグ時にxyの内容を参照することはできません。

-O0オプション( 最適化OFF )を指定して作成した実行ファイルを評価後、-O1オプション( 最適化ON )を指定して作成し直した場合は動作保証できませんので、再検証を行ってください。

### [Enter]キーによる連続実行

以下のコマンドは一度実行すると、その後は[Enter]キーのみで連続して実行できます。プログラム実行コマンドは同じ動作を繰り返します。表示コマンドは前に表示した次の部分を連続して表示します。

実行コマンド: g, s, n, se, com

表示コマンド: u, dd, td,

この連続実行機能は他のコマンドを実行すると解除されます。

## 13.7.2 メニュー、ツールバーからの実行

13.5項、13.6項に示したように、メニューとツールバーには頻繁に使用するコマンドが登録されています。メニューコマンドを選択するか、ツールバーボタンをクリックするだけで、指定のコマンドを実行できます。表13.7.2.1に登録されているコマンドの一覧を示します。

表13.7.2.1 メニュー、ツールバーで指定可能なコマンド

コマンド	機能	メニュー	ボタン
lf	プログラムファイルのロード	[File   Load File...]	
par	パラメータファイルのロード	[File   Load Parameter File...]	
g	プログラムの連続実行	[Run   Go]	
—	プログラムをカーソル位置まで連続実行	[Run   Go to Cursor]	
gr	CPUリセット後、プログラムの連続実行	[Run   Go after Reset]	
s	ステップ実行	[Run   Step]	
n	ステップ&サブルーチンスキップ	[Run   Next]	
se	サブルーチンの終了	[Run   Step Exit]	
com	コマンドファイルのロード、実行	[Run   Command File...]	—
cmw	コマンドファイルウェイト付き実行	[Run   Command File...]	—
rst	CPUリセット	[Run   Reset CPU]	
bp, bpa, bpr, bc, bpc	ソフトウェアブレークポイント設定/解除	[Break   Breakpoint Setting]	
bas	シーケンシャルブレークモード設定	[Break   Setting]	—
ba, bar	ハードウェアブレークポイント設定/解除	[Break   Breakpoint Setting]	—
bd, bdc	データブレーク条件の設定/解除	[Break   Breakpoint Setting]	—
bl	ブレーク条件の表示	[Break   Break List]	—
bac	全ブレーク条件の解除	[Break   Break All Clear]	
td	トレース情報の表示	[View   Trace], [Trace   Trace]	—
ts	トレース情報の検索	[Trace   Trace Search...]	—
tf	トレース情報のファイルへの保存	[Trace   Trace File...]	—
cv	カバレッジ情報の表示	[Coverage   Coverage]	—
cvc	カバレッジ情報のクリア	[Coverage   Coverage Clear]	—
u	逆アセンブル表示	[View   Source   Disassemble]	 *
sc	ソース表示	[View   Source   Source]	 *
m	ミックス表示	[View   Source   Mix]	 *
dd	メモリダンプ	[View   Dump]	—
rd	レジスタ値の表示	[View   Register]	—
sy	シンボル一覧の表示	[View   Symbol]	—
w	シンボル情報の表示	[View   Watch]	—
	シンボルの登録	—	 *
log	ログ出力ON/OFF	[Option   Log...]	—
rec	実行コマンドの記録	[Option   Record...]	—

\* [Source]ウィンドウ上のボタン



### 13.7.3 コマンドファイルによる実行

一連のデバグコマンドを記述したコマンドファイルを読み込んで、それらのコマンドを実行させることができます。

#### コマンドファイルの作成

コマンドファイルはエディタでテキストファイルとして作成してください。

ファイル名の拡張子は".cmd"とします。

コマンドファイルは、recコマンドによっても作成できます。recコマンドを使用すると、デバグが実際に実行したコマンドをコマンドファイルに記録することができます。

#### コマンドファイル例

次の例は、プログラムファイルを読み込み、ブレークポイントを設定して実行させるためのコマンド群です。

例: ファイル名=start.cmd

```
lf test.abs
```

```
bp 0004d7
```

```
g
```

コマンドファイルにはガイダンスモードに対応した記述も可能です。その場合は、ガイダンス入力の各項目ごとに改行して記述してください。

#### コマンドファイルの読み込み/実行

コマンドファイルの実行用に、comコマンドとcmwコマンドが用意されています。

comコマンドは指定されたファイルを読み込み、その中のコマンドを記述順に指定された間隔(0~256秒)で実行します。cmwコマンドも同様ですが、個々のコマンドはmdコマンドで指定された間隔(1~256秒)で実行されます。

例: com start.cmd

```
cmw test.cmd
```

コマンドファイルに記述されたコマンドは[Command]ウィンドウに表示されます。

#### 制限事項

コマンドファイル内から別のコマンドファイルを読み込むことも可能です。ただし、最大5階層までに制限されます。6階層目のcom/cmwコマンドが現れるとエラーとなり、それ以後の実行を中止します。

### 13.7.4 ログファイル

実行したコマンドと実行結果を、テキスト形式のログファイルとして保存することができます。これによって、後からデバッグの手順と内容を確認することができます。

保存の対象となるのは[Command]ウィンドウに表示された内容です。

#### コマンド例

```
>log tst.log
```

logコマンドでログモードに設定後(出力開始後)は、logコマンドがトグル動作(ログモード/出力ON↔通常モード/出力OFF)となりますので、必要な部分のみの出力が簡単に行えます。

#### ログモードでの[Command]ウィンドウの表示

ログモードでは、[Command]ウィンドウに表示される内容が通常の場合と異なります。

##### (1) 各ウィンドウが開いている場合のコマンド実行時

(ウィンドウに結果が表示されるコマンドをそのウィンドウが開いている状態で実行した場合)

通常モード: 表示先のウィンドウの内容が更新されます。[Command]ウィンドウに実行結果は表示されません。

ログモード: ウィンドウに表示される情報と同等の内容が[Command]ウィンドウにも表示されます。ただし、ウィンドウのスクロール操作、ウィンドウを開いたことによる表示内容は、[Command]ウィンドウには表示されません。

##### (2) 各ウィンドウが閉じている場合のコマンド実行時

表示先のウィンドウが閉じている場合、ログモード/通常モードにかかわらず実行結果が[Command]ウィンドウに表示されます。





## 13.8 デバッグ機能

ここでは、デバッグ機能の概要を機能別に説明します。

### 13.8.1 ファイルの読み込み

表13.8.1.1にデバッグが読み込むファイルの一覧と読み込みコマンドを示します。

表13.8.1.1 ファイルと読み込みコマンド一覧

ファイル	拡張子	生成ツール	コマンド	メニュー	ボタン
1. パラメータファイル	.par	—	par	[File   Load Parameter File...]	
2. IEEE-695アブソリュートオブジェクトファイル	.abs	lc88	lf	[File   Load File...]	
3. モトローラS2プログラムファイル	.psa	fil88xxx	lf	[File   Load File...]	
4. ファンクションオプションファイル	.fsa	fog88xxx or winfog	lf	[File   Load File...]	
5. シンボルファイル	.sy	sy88, sym88	—	—	—
6. コマンドファイル	.cmd	—	com/cmw	[Run   Command File...]	—
7. FPGAデータファイル	.mot	—	xfwr ;S	—	—
	.mcs	—	xfwr ;H	—	—

パラメータファイルをロードすると、その時点でデバッグがリセットされます。パラメータファイルで設定されたメモリマップ情報はmaコマンドで表示させることができます。パラメータファイルについては、「12 88xxx.parファイル」を参照してください。

IfコマンドはIEEE-695形式のアブソリュートオブジェクトファイル(.abs)、モトローラS2形式のプログラムファイル(.psa)、ファンクションオプションHEXファイル(.fsa)をロードします。デバッグはこれらのファイルを指定された拡張子で区別します。ソースレベルデバッグを行うためには、IEEE-695形式でデバッグ情報を含んだファイルを読み込む必要があります。

シンボルファイルはモトローラS2形式のプログラムファイルをデバッグする際に、アドレスをソース作成時のシンボルを使用して指定するために必要で、ロードしなくてもデバッグは行えます。このファイルはIfコマンドによるモトローラS2形式プログラムファイルのロード時に同時に読み込まれます。ただし、プログラムファイルと同じ名称(拡張子は.sy)のシンボルファイルをプログラムファイルと同じディレクトリに用意しておく必要があります。シンボルファイルが読み込まれている場合、その内容は[Symbol]ウィンドウまたはsyコマンドで確認できます。

IEEE-695形式でシンボル情報を含んだアブソリュートオブジェクトファイルを読み込んだ場合は、そのファイルのみでシンボルが使用可能なため、シンボルファイルは読み込まれません。

コマンドファイルは13.7.3項で説明したとおりです。

FPGAデータファイルは、ペリフェラルボード(S5U1C88000P)上のFPGAを各機種用に構成するために必要です。一度FPGAに書き込むと、基本的にはその機種の開発が終了するまで再書き込みの必要はありません。




## 13.8.2 ソース表示およびシンボリックデバッグ機能

本デバッガは、Cソースを表示させたデバッグが可能です。また、シンボル名を使用してアドレスの指定も行えます。

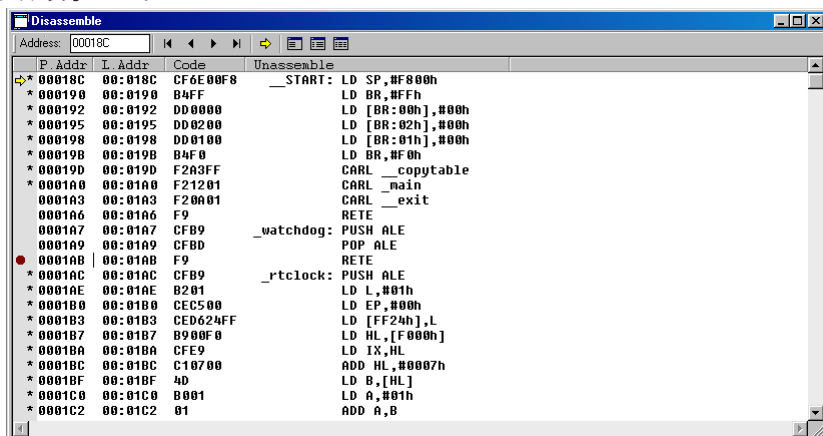
### プログラムコードの表示

[Source]ウィンドウは指定された表示モードでプログラムを表示します。表示モードは、逆アセンブル表示モード、ソース表示モード、ミックス表示モードの3種類から選択できます。

表13.8.2.1 表示モード切り換えコマンド

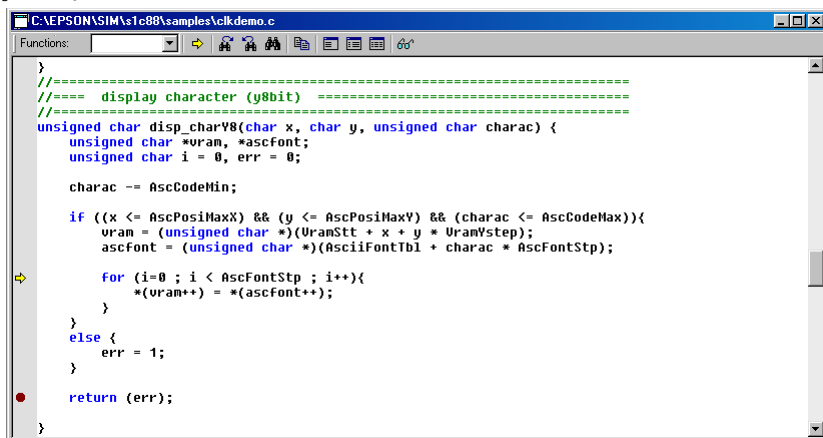
機能	コマンド	メニュー	ボタン
逆アセンブル表示モード	u	[View   Source   Disassemble]	
ソース表示モード	sc	[View   Source   Source]	
ミックス表示モード	m	[View   Source   Mix]	

#### (1) 逆アセンブル表示モード



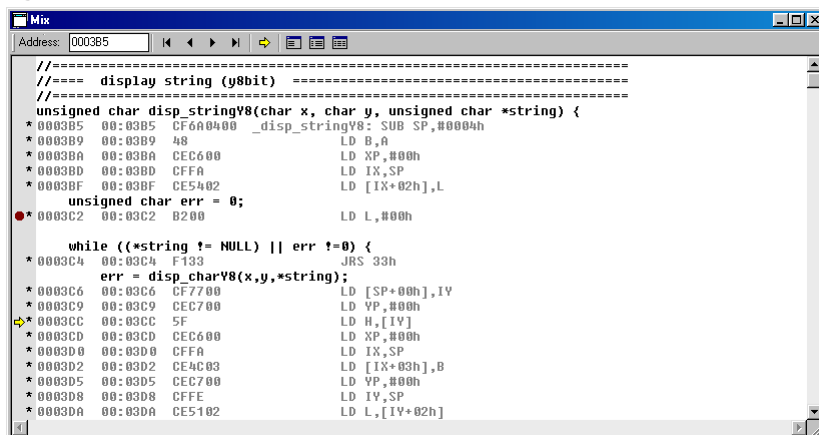
このモードでは、オブジェクトコードをニーモニックに逆アセンブルして表示します。

#### (2) ソース表示モード



このモードでは、現在のPCアドレス上のコードを含むソースが表示されます。このモードは、ソースデバッグ情報を含むIEEE-695形式アプソリュートオブジェクトファイルをロードしている場合にのみ指定可能です。

## (3)ミックス表示モード



このモードでは、プログラムソースとそれに対応するオブジェクトの逆アセンブル内容が表示されます。このモードは、ソースデバグ情報を含むIEEE-695形式のアブソリュートオブジェクトファイルをロードしている場合にのみ指定可能です。

表示内容とウィンドウ上の操作については、「13.4.3 [Source]ウィンドウ」を参照してください。

## シンボル参照

IEEE-695形式のオブジェクトファイル(.abs)を読み込んでデバグを行う場合、ソースファイルで定義されたシンボルを使用してアドレスを指定することができます。パラメータに<address>を持つコマンドを[Command]ウィンドウ上で入力する際、あるいはアドレスをダイアログで指定する際に使用することができます。ただし、オブジェクトファイルにデバグ情報が含まれている必要があります。

モトローラS2形式のプログラムファイル(.psa)を読み込んでシンボルを使用したデバグを行う場合は、プログラムファイルと同名のシンボルファイルを同じディレクトリに用意しておく必要があります。シンボルファイルはプログラムファイルのロード時に自動的に読み込まれます。

デバグ中のプログラムで使用しているシンボルと定義されたアドレスは、[Command]ウィンドウまたは[Symbol]ウィンドウに表示させることができます。

表13.8.2.2 シンボルリスト表示コマンド

機能	コマンド	メニュー	ボタン
シンボルリストの表示	sy	[View   Symbol]	—

### 13.8.3 メモリデータ、レジスタの表示と変更

デバッグはメモリ、レジスタに対する操作機能を持っています。メモリ領域はパラメータファイルで与えられるマップ情報に従ってデバッグに設定されます。

#### メモリの操作

メモリ(ROM領域、RAM領域、表示メモリ、I/Oメモリ)に対しては以下の操作が行えます。

表13.8.3.1 メモリ操作コマンド

機能	コマンド	メニュー	ボタン
メモリダンプ	dd	[View   Dump]	—
データの入力/変更	de	—	—
指定領域の書き換え	df	—	—
指定領域のコピー	dm	—	—
データの検索	ds	—	—

#### (1)メモリダンプ

メモリの内容を指定サイズ( Byte, Word, Long, Float, Double )の16進ダンプ形式で表示します。[Dump]ウィンドウが開いていれば[Dump]ウィンドウの内容を更新し、開いていなければ[Command]ウィンドウに表示します。

#### (2)データの入力/変更

16進データを入力して、指定アドレスのデータを書き換えます。[Dump]ウィンドウ上で直接変更することもできます。

#### (3)指定領域の書き換え

指定した領域を指定したデータですべて書き換えます。

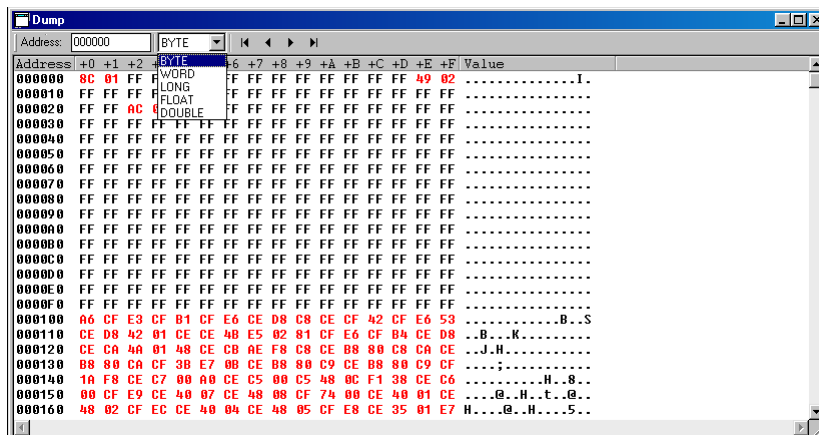
#### (4)指定領域のコピー

指定した領域の内容を、別の領域にコピーします。

#### (5)データの検索

指定の領域内で指定のデータを検索できます。検索結果は最大256個まで[Command]ウィンドウに表示できます。また、[Dump]ウィンドウが現在表示している範囲内で指定データが見つかった場合、[Dump]ウィンドウ上の該当データは緑色で表示されます。

[Dump]ウィンドウの表示内容とウィンドウ上の操作については、"13.4.4 [Dump]ウィンドウ"を参照してください。



## レジスタの操作

レジスタに対しては以下の操作が行えます。

表13.8.3.2 レジスタ操作コマンド

機能	コマンド	メニュー	ボタン
レジスタの表示	rd	[View   Register]	—
レジスタ値の変更	rs	—	—

## (1) レジスタの表示

レジスタおよびレジスタで間接アドレッシングされるメモリの内容を[Register]ウィンドウまたは[Command]ウィンドウに表示させることができます。

レジスタ: PC, SP, IX, IY, A, B, H, L, BR, CB, NB, EP, XP, YP, SC (I1, I0, U, D, N, V, C, Z), CC (F3, F2, F1, F0)

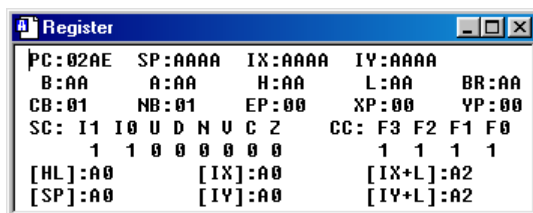
メモリ: [HL], [SP], [IX], [IY], [IX+L], [IY+L]

## (2) レジスタ値の変更

上記レジスタの内容を任意の値に設定できます。

レジスタ値は[Register]ウィンドウ上で直接変更することもできます。

[Register]ウィンドウの表示内容とウィンドウ上の操作については、"13.4.5 [Register]ウィンドウ"を参照してください。



### 13.8.4 プログラムの実行

デバッグにはターゲットプログラムを連続実行およびシングルステップ実行させる機能があります。




#### 連続実行

##### (1) 連続実行の種類

3種類の連続実行機能があります。

- ・現在のPCアドレスから連続実行
- ・現在のPCアドレスから[Source]ウィンドウのカーソル位置まで連続実行
- ・CPUをリセット後に連続実行

表13.8.4.1 連続実行コマンド

機能	コマンド	メニュー	ボタン
現在のPCアドレスから連続実行	g	[Run   Go]	
現在のPCアドレスからカーソル位置まで連続実行	-	[Run   Go to Cursor]	
CPUをリセット後に連続実行	gr	[Run   Go after Reset]	

##### (2) 連続実行の停止

テンポラリブレークアドレスを[Source]ウィンドウで指定することができます。[Source]ウィンドウ上のブレークさせる行にカーソルを置いて[Go to Cursor]ボタンをクリックすると、プログラムは現在のPCから実行を開始し、カーソル位置の命令を実行する直前にブレークします。

なお、ソース表示モードでCソースを表示している場合は、有効なオブジェクトコードに展開されているソース行にカーソルを置く必要があります。コメント行や宣言文など、オブジェクトコードにコンパイルされないソース行にカーソルがある場合、[Go to Cursor]ボタンをクリックしてもプログラムは実行されません(ソフトウェアブレークポイントの説明も参照してください)。

このテンポラリブレーク以外では、実行中のプログラムは次のいずれかの要因によってブレークするまで停止しません。

- ・ブレーク設定コマンドで設定したブレーク条件が成立
- ・ICEのBRKIN端子への入力
- ・[Key Break]ボタンのクリック、[Run | Stop]メニューコマンドの選択または[ESC]キー入力
- ・プログラム実行エラーの検出



[Key Break]ボタン

プログラムが停止しない場合は、このボタンで強制ブレークさせることができます。

注: Cソース表示モードでプログラムの実行を停止すると、停止したアドレスが含まれるオブジェクトのソースが表示されます。ただし、停止したアドレスにソースがない場合、[Source Files]ダイアログボックスが表示され、ソースファイルの選択が必要になります。

##### (3) 連続実行時の表示

連続実行により、各ウィンドウは次のように表示を更新します。

実行を停止すると、[Command]ウィンドウに実行サイクル数および実行時間が表示されます。

例: >g

```
BUS CYCLE : 428649 ... バスサイクル数
Mode L    : 00lmin 002s 543ms 468us ... 実行時間(デフォルト 1μs単位)
```

[Source]、[Register]、[Dump]ウィンドウの表示内容は、実行中は実行開始時の状態を保ち、実行停止後に更新されます。[Register]ウィンドウが閉じている場合は、実行停止後にレジスタの値が[Command]ウィンドウに表示されます。

[Trace]ウィンドウは実行中は表示がクリアされ、実行停止後に最新データを表示します。

[Watch]ウィンドウも、デフォルト設定では実行停止後に更新されます。ただし、このウォッチ機能については[Run | Setting...]メニューコマンドで表示されるダイアログボックスによって、表示更新間隔を設定することができます。

[Symbol]ウィンドウと[Coverage]ウィンドウは変更されません。



## シングルステップ実行



### (1) シングルステップ実行の種類

3種類のシングルステップ実行機能があります。

- ・ Cステートメント/命令をシングルステップ実行( STEP )  
Cソース表示モードではCソース行単位にシングルステップで実行します。  
逆アセンブル表示またはミックス表示モードでは命令単位にシングルステップで実行します。
- ・ 関数、サブルーチン以外をシングルステップ実行( NEXT )  
Cソース表示モードでは、関数呼び出しがあっても関数内には入らず、リターンするまでを1ステップとして実行します。関数呼び出し以外は通常のシングルステップ実行と同様です。  
逆アセンブル表示またはミックス表示モードでは、cars、carl、call、int命令を、リターン命令で次のステップに戻るまでを1ステップとして実行します。他の命令は、通常のシングルステップ実行と同様です。
- ・ 関数、サブルーチンの終了( STEP EXIT )  
Cソース表示モードでは、現在の関数から上位レベルの関数に戻るまでを連続実行し、リターン後に停止します。メイン関数内では実行しないでください。  
逆アセンブル表示またはミックス表示モードでは、現在のサブルーチンからリターン命令で上位レベルに戻るまでを連続実行し、リターン後に停止します。最上位レベルではgコマンドと同様の機能です。下位レベルのサブルーチンをコールしてリターンしても停止しません。

いずれの場合も現在のPCから実行します。

表13.8.4.2 シングルステップコマンド

機能	コマンド	メニュー	ボタン
シングルステップ実行	s	[Run   Step]	
関数/サブルーチン以外をシングルステップ実行	n	[Run   Next]	
関数/サブルーチンの終了	se	[Run   Step Exit]	

sおよびnコマンド入力では、実行するステップ数を最大65,535ステップまで指定することができます。メニューコマンド、ツールバーでは1ステップずつ実行します。

以下の場合には、シングルステップ実行が指定のステップ数の実行前に終了します。

- ・ [Key Break]ボタンのクリック、[Run | Stop]メニューコマンドの選択または[ESC]キー入力
- ・ プログラム実行エラーの検出

PCブレーク、データブレーク等のユーザ設定ブレークでは停止しません。



[Key Break]ボタン

プログラムが停止しない場合は、このボタンで強制ブレークさせることができます。

### (2) ステップ動作中の表示

デバッガの初期設定では、次のように表示を更新します。

[Source]、[Register]、[Dump]、[Trace]、[Watch]ウィンドウの表示内容は最終ステップを実行後に更新されます。[Register]ウィンドウが閉じている場合は、レジスタの値が最終ステップを実行後に[Command]ウィンドウに表示されます。

[Symbol]ウィンドウと[Coverage]ウィンドウは変更されません。

## (3) ステップ実行時の割り込み

halt命令またはslp命令を実行するとCPUはスタンバイモードとなり、その解除には割り込みが必要です。デバッガでは、シングルステップ動作用に外部割り込みの許可/禁止モードが設定されています。

表13.8.4.3 外部割り込みモード

	許可モード	禁止モード
外部割り込み	割り込みを処理する	割り込みを処理しない
halt、slp命令	halt命令として実行 外部割り込み、または[Key Break] ボタンで処理を継続	halt、slp命令をnop命令に置き換えて実行

デバッガの初期設定で、割り込み禁止モードに設定されます。[Run | Setting...]メニューコマンドで表示されるダイアログボックスによって、割り込み許可モードに設定することができます。

## (4) Cソースのシングルステップ実行に関する注意事項

Cソース表示モードでのシングルステップ実行は基本的にソース行単位になります。ただし、対応するオブジェクトコードのないソース行やユーザーソースのない箇所(インラインアセンブルやコンパイラが自動生成した関数など)は、その次の行まで実行されます。

これに伴い、Cステートメントの書き方でもステップ回数が変わります。

例: `for(x=0; x<10; x++) a[x]=x;` ... 1ステップで実行されます。

```
for(x=0; x<10; x++)
    a[x]=x;
```

... for文を抜けるのに20ステップの実行が必要です。

## 実行オプション

プログラムの実行に関連して4つのオプションが用意されています。その選択は、[Run]メニューの[Setting...]により表示されるダイアログボックスで行います。



## Run Monitor Interval

[Watch]ウィンドウの更新モードをショートブレイクモードに設定した場合の表示更新間隔を100ms単位で設定します。設定可能な範囲は1(100ms、デフォルト)~10(1秒)です。

## Single Step Mode

シングルステップ時に割り込みを許可するか禁止するか選択します(表13.8.4.3参照)。割り込みを許可する場合はチェックボックスをONしてください。

## Watch/Local repaint

[Watch]ウィンドウの更新モードを設定します。デフォルトのリアルタイムモード([Real Time]を選択)は、プログラムをリアルタイムに実行させるためのモードで、[Watch]ウィンドウはプログラム実行のブレイク後に更新されます。ショートブレイクモード([Short Break]を選択)は[Run Monitor Interval]で指定した時間間隔ごとに表示内容を更新します。ただし、このモードでは表示更新のためにプログラムを一時的に停止しています。したがって、プログラムはリアルタイムに実行できません。

## Run-time measurement base

ICEは31ビットの実行サイクルカウンタを内蔵しており、プログラムを連続実行した時間およびバスサイクル数を測定することができます。このときの時間測定の単位を1μs(デフォルト)とするか62.5nsとするか選択できます。バスサイクルは最大2,147,483,647サイクル(誤差±0)までカウント可能です。時間測定の最大値は次のとおりです。

1μs単位: 約35分50秒(誤差±1μs)

62.5ns単位: 約2分15秒(誤差±62.5ns)

測定結果は連続実行がブレイクすると、[Command]ウィンドウに次のように表示されます。

例: >g

```
BUS CYCLE : 428649 ... バスサイクル数
Mode L    : 001min 002s 543ms 468us ... 実行時間(1μs単位、デフォルト)
```

>g

```
BUS CYCLE : 35095 ... バスサイクル数
Mode L    : 003s 094ms 152us 0.0ns ... 実行時間(62.5ns単位)
```

カウンタの最大値を越えた場合、バスサイクル数は"Count overflow"を、実行時間は"Time over"を表示します。

カウンタはプログラムの連続実行開始時にリセットされます。

シングルステップ動作時は測定しません。

## CPUのリセット

表13.8.4.4 CPUリセットコマンド

機能	コマンド	メニュー	ボタン
CPUリセット	rst	[Run   Reset CPU]	
CPUをリセット後に連続実行	gr	[Run   Go after Reset]	

CPUはgrコマンドの実行時、およびrstコマンドの実行によりリセットされます。

CPUのリセットによる初期設定内容は以下のとおりです。

### (1) CPUの内部レジスタ、メモリ

イニシャルリセットによりICE CPUの内部レジスタは以下のように初期化されます。

PC: リセット例外処理によって、0バンクのメモリの先頭(000000H~000001H)に格納されている値がPCにロードされます。

SP, IX, IY: 0xAAAA

B, A, H, L, BR: 0xAA

CB, NB: 0x01

EP, XP, YP: 0x00

SC: 0b11000000

CC: 0b1111

内蔵RAMおよび外部RAMはイニシャルリセット時に初期化されません。

内蔵のI/Oメモリについては、それぞれの初期値に設定されます。

### (2) [Source]ウィンドウ、[Register]ウィンドウを再表示

PCが再設定されるため、[Source]ウィンドウはそのアドレスから再表示されます。

[Register]ウィンドウも上記の設定で再表示されます。

### 13.8.5 ブレーク機能

ターゲットプログラムは次の要因により実行を中断します。

- ・ブレーク設定コマンドの条件成立( 連続実行時のみ )
- ・ICEのBRKIN端子への入力( 連続実行時のみ )
- ・[Key Break]ボタンのクリック、[Run | Stop]メニューコマンドの選択または[ESC]キー入力
- ・プログラム実行エラーの検出

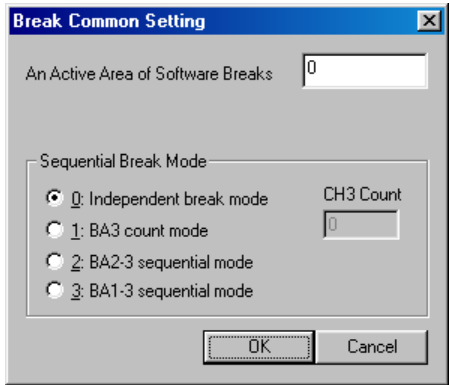
#### コマンドによるブレーク機能

デバッグはコマンドによってブレーク条件が設定できる3種類のブレーク機能を持っています。それぞれ、条件が成立すると実行中のプログラムはブレークします。

#### (1)ソフトウェアブレークポイント/ソフトウェアブレーク領域

PCが設定したアドレスに一致するとブレークする機能です。プログラムは、そのアドレスの命令をフェッチすると実行前にブレークします。ソフトウェアブレークポイントは最大64ヶ所の個別アドレスと、アドレス範囲を指定した1領域に設定できます。

ただし、これらのブレークポイントが有効になるのは、1MBのブレーク有効領域内のみです。この領域を外れたアドレスを指定した場合、無効なブレークポイントとしてアドレスは登録されますが、そこでブレークを発生させることはできません。ブレーク有効領域は8MBのコード空間を1MBずつ分割した8領域から1つを[Break | Setting...]メニューコマンドで表示される[Break Common Setting]ダイアログボックスで選択できるようになっています。デバッグの起動時には0x0 ~ 0x0ffffの1MBに設定されます。



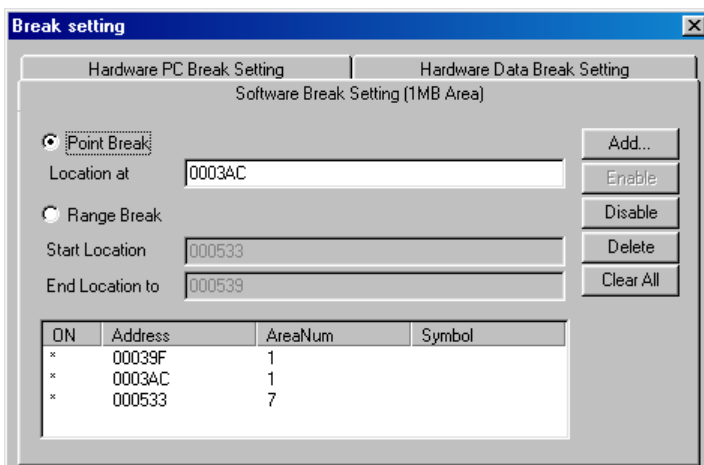
この領域の指定は[An Active Area of Software Breaks]テキストボックスで行います。0～7の数値を入力してください。

0: 0x000000 ~ 0x0fffff  
1: 0x100000 ~ 0x1fffff  
2: 0x200000 ~ 0x2fffff  
:  
7: 0x700000 ~ 0x7fffff

表13.8.5.1 ブレークポイント設定コマンド

機能	コマンド	メニュー	ボタン
ソフトウェアブレークポイントの設定	bp	[Break   Breakpoint Setting]	
ソフトウェアブレーク領域の設定	bpa	[Break   Breakpoint Setting]	-
ソフトウェアブレークポイントの解除	bpr bc (bpc)	[Break   Breakpoint Setting]	

[Break]メニューから[Breakpoint Setting]を選択すると、[Break setting]ダイアログボックスが現れ、[Software Break Setting (1MB Area)]タブの画面に、設定されているPCブレークポイントの一覧が表示されます。



ソフトウェアブレークポイントを設定するには、[Point Break]ラジオボタンを選択し[Location at]テキストボックスにアドレスを入力します。[Add]ボタンをクリックすると、有効なブレークポイントとして登録されます。64ヶ所まではリストに追加されますが、それを越えるとワーニングになります。その場合は先に不要なブレークポイントを削除してください。

ソフトウェアブレーク領域を設定するには、[Range Break]ラジオボタンを選択し[Start Location]テキストボックスに領域開始アドレスを、[End Location to]テキストボックスに領域終了アドレスを入力します。[Add]ボタンをクリックすると、ソフトウェアブレーク領域として登録されます。この領域内は、すべてのアドレスがブレークポイントに設定されたものと見なされます。リストのAddressには先頭アドレスが、AreaNumには領域のバイト数が表示されます。すでにソフトウェアブレーク領域が登録されている状態で新たな領域を設定するとワーニングになります。その場合は先に登録されているソフトウェアブレーク領域を削除してください。

ソフトウェアブレーク領域内の全アドレスを含め、すでにブレークポイントとして登録されているアドレス(またはそのアドレスを含む領域)を重複して設定することはできません。

有効なブレークポイント(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのアドレスを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレークポイントは無効になります。

無効なブレークポイントを有効にするには、ブレークポイントリストからそのアドレスを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレークポイントを解除するには、ブレークポイントリストからそのアドレスを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、ソフトウェアブレーク領域も含め、設定されているすべてのブレークポイントを解除します。

PCブレークポイントに設定されたアドレスは、[Source]ウィンドウ上の行の先頭に が表示されます。ただし、ソフトウェアブレーク領域に設定されたアドレスには表示されません。

Cソース表示の例

```

    }
    return (err);
}

```

逆アセンブル表示の例

```

0001A7 00:01A7 CFB9      _watchdog: PUSH ALE
0001A9 00:01A9 CFBD              POP ALE
0001AB 00:01AB F9              RETE
* 0001AC 00:01AC CFB9      _rtclock: PUSH ALE

```

[Break]ボタンを使用して簡単にソフトウェアブレークポイントの設定と解除を行うこともできます。



[Break]ボタン

[Source]ウィンドウ上で、ブレークポイントに設定する行をクリック(カーソルを移動)し、[Break]ボタンをクリックします。その行の先頭にマークが表示され、そのアドレスはブレークポイントリストに登録されます。で始まる行をクリックして[Break]ボタンをクリックすると、ブレーク設定が解除され、アドレスがブレークポイントリストから削除されます。

[Break]ボタンでソフトウェアブレーク領域の設定と解除は行えません。

#### ソース表示モード時のブレークポイント設定

ソース表示モードの[Source]ウィンドウ上では、ブレークポイントを設定できる行と、できない行があります。実コードが生成されていないソース行には設定できません。

```
例: 1      void func(void)           // NG
    2      {                       // OK
    3          int a;               // NG
    4          int x=0;             // OK
    5          a = x;              // OK
    6      }                       // OK
```

1は関数の宣言で、実コードがない(アセンブラのラベル宣言と同じです)ため設定できません。

3は変数の宣言で、実コードがないため設定できません。

4は変数の宣言ですが、初期化のコードが生成されるため設定できます。

2は設定できます。ただし、ブレークポイントは4(その関数の先頭の命令)に設定されます。

5は実コードのある有効な行ですので、設定できます。

6は関数の終了(ニーモニックのretと等価)ですので、設定できます。

ただし、コンパイル時に最適化を行うとブレークポイントが設定できなくなることがあります。上記例では、各行の実行により得られるものが何もない(ローカル変数の書き換えのみでグローバル変数の書き換えがない)ため、最適化により実コードがなくなる可能性があります。

[Go to Cursor]ボタンで停止できる行の条件も同じです。

## (2) シーケンシャルブレーク機能

シーケンシャルブレークは、ターゲットプログラムが指定のアドレスを指定のシーケンスに従って実行した場合にブレークを発生させる機能です。

シーケンシャルブレーク用に3つのチャンネル(BA1～BA3)が用意されています。それぞれのチャンネルには1つずつアドレスを設定しておくことができます。また、BA3はアドレスのほかに実行回数を指定することができます。

ここで設定するアドレスは、ブレーク有効領域の設定に関わらず、全コード空間で有効です。

使用するチャンネルにより、以下に4つのシーケンシャルブレークモードを設定することができます。

## 独立ブレークモード

このモードでは、各チャンネルが独立したブレークポイントとなります。

ブレークは、チャンネルに設定したアドレスの命令をフェッチすると、その命令の実行前に発生します。BA3に対する実行回数の指定は、無効となります。

## BA3カウントモード

このモードでは、BA3に設定したアドレスの命令を指定回数フェッチするとブレークします。

BA1とBA2の設定は無効となります。

## BA2-3シーケンシャルモード

このモードでは、BA2に設定したアドレスの命令を一度以上実行後、BA3に設定したアドレスの命令を指定回数フェッチするとブレークします。BA1の設定は無効となります。

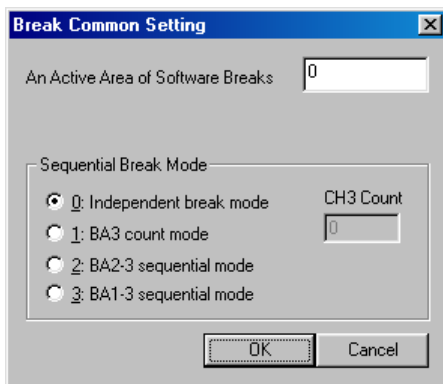
## BA1-3シーケンシャルモード

このモードでは、BA1、BA2の順にそれぞれに設定したアドレスの命令を一度以上実行後、BA3に設定したアドレスの命令を指定回数フェッチするとブレークします。

表13.8.5.2 シーケンシャルブレーク設定コマンド

機能	コマンド	メニュー	ボタン
シーケンシャルブレークモードの設定	bas	[Break   Setting...]	—
ハードウェアブレークポイントの設定	ba	[Break   Breakpoint Setting]	—
ハードウェアブレークポイントの解除	bar	[Break   Breakpoint Setting]	—

シーケンシャルブレークモードを設定するには、[Break]メニューから[Setting...]を選択します。

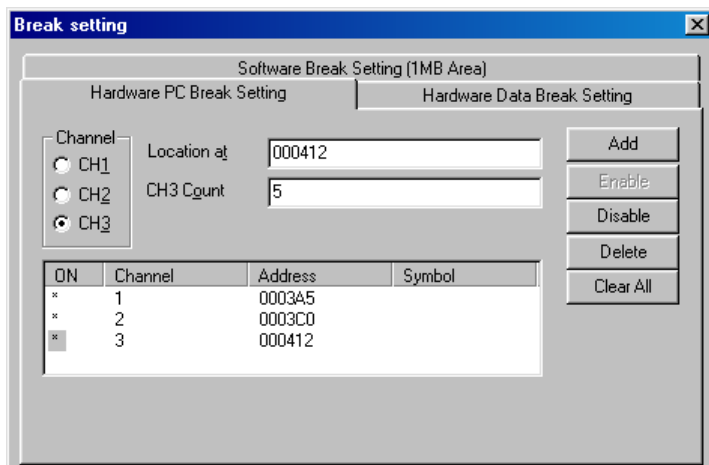


[Break Common Setting]ダイアログボックスが表示されますので、[Sequential Break Mode]のラジオボタンで設定するモードを選択します。

BA3カウンタを使用するモードのラジオボタンを選択すると[CH3 Count]テキストボックスがアクティブになりますので、BA3の実行回数を入力します。ここに設定した回数だけ、BA3アドレス上の命令がフェッチされるまで、ブレークは発生しません。



各チャンネルのアドレスは[Break]メニューから[Breakpoint Setting]を選択することによって表示される[Break setting]ダイアログボックスで行います。[Break setting]ダイアログボックスが表示されたら、[Hardware PC Break Setting]タブの画面に切り換えてください。



設定するチャンネルをラジオボタンで選択し、[Location at]テキストボックスにアドレスを入力します。BA3のカウンタ数を指定する場合は、[CH3 Count]テキストボックスに16進数で入力します。[Break Common Setting]ダイアログボックスでカウンタ数を設定した場合は、その数値がここに反映されます。[Add]ボタンをクリックすると、有効なブレークポイントとして登録されます。設定可能なアドレスは各チャンネルにつき1ヶ所のみで、設定済みチャンネルに新たな設定を行うと上書きされます。また、すでにハードウェアPCブレークポイントに設定してあるアドレスを指定した場合はワーニングになります。BA1-3シーケンシャルモードで上記の設定を行った場合、プログラムが次のようなシーケンスを実行するとブレークします。

1. 実行開始
  - ：
  2. アドレス0x0003A5の命令を1回以上実行
  - ：
  3. アドレス0x0003C0の命令を1回以上実行
  - ：
  4. アドレス0x000412の命令を4回実行
  - ：
  5. アドレス0x000412の命令を再度フェッチ
- 5の時点で、0x000412の命令を実行前にブレークが発生します。

有効なブレークポイント(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのアドレスを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレークポイントは無効になります。

無効なブレークポイントを有効にするには、ブレークポイントリストからそのアドレスを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレークポイントを解除するには、ブレークポイントリストからそのアドレスを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、設定されているすべてのブレークポイントを解除します。



## (3) データブレイク機能

データブレイクは、指定したメモリをアクセスした場合にブレイクを発生させる機能です。データブレイク用には、4つのチャンネル(CH0～CH3)が用意されており、チャンネルごとに次の3つの条件が指定できます。

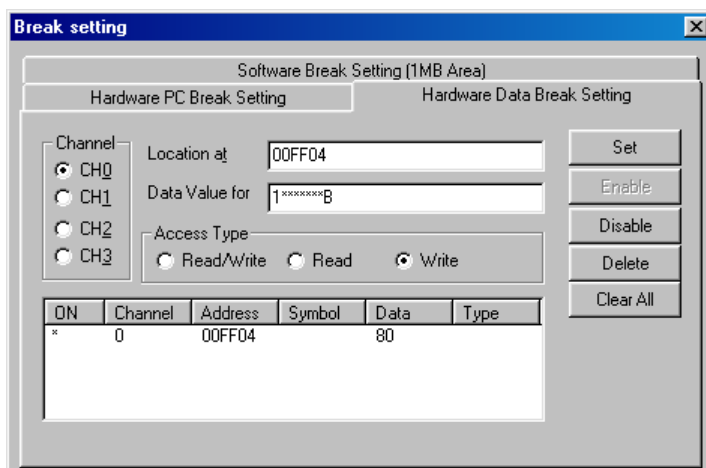
- アドレス**      アドレスを指定すると、ターゲットプログラムがそのアドレスにアクセスした場合にブレイクします。
- データ**          データを指定すると、ターゲットプログラムが指定のデータを書き込むか、読み出した場合にブレイクします。1バイトのデータを指定します。データビットはマスクが可能で、必要なビットの一致のみでブレイクさせることもできます。
- リード/ライト**    リードサイクルのみ、ライトサイクルのみ、またはどちらのサイクルでもブレイクするように指定できます。

この中の1つ以上の条件を指定します。複数の条件を指定した場合は、すべての条件を満たすメモリアクセスを実行後にブレイクします。

表13.8.5.3 データブレイク設定コマンド

機能	コマンド	メニュー	ボタン
データブレイク条件の設定	bd	[Break   Breakpoint Setting]	—
データブレイク条件の解除	bdr	[Break   Breakpoint Setting]	—

[Break]メニューから[Breakpoint Setting]を選択すると、[Break setting]ダイアログボックスが表示されます。ダイアログボックスが表示されたら、[Hardware Data Break Setting]タブの画面に切り換えてください。



設定するチャンネルをラジオボタンで選択し、[Location at]テキストボックスにアドレスを、[Data Value for]テキストボックスにデータを入力します(未指定も可能)。リード/ライト条件をラジオボタンで選択し、[Set]ボタンをクリックすると、有効なブレイク条件として登録されます。設定済みチャンネルに新たな設定を行うと上書きされます。

上記の例では、ターゲットプログラムがアドレス0x00ff04にMSBが1のデータを書き込むとブレイクします。

有効なブレイク条件(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのチャンネルを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレイク条件は無効になります。

無効なブレイク条件を有効にするには、リストからそのチャンネルを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレイクポイントは有効になります。

ブレイク条件を解除するには、リストからそのチャンネルを選択し、[Delete]ボタンをクリックします。[Clear All]ボタンは、設定されているすべてのブレイク条件を解除します。

## (4) その他のブレークコマンド

設定されている全ブレーク条件を[Command]ウィンドウに表示するコマンドと、全ブレーク条件を解除するコマンドが用意されています。

表13.8.5.4 その他のブレークコマンド

機能	コマンド	メニュー	ボタン
全ブレーク条件の表示	bl	[Break   Break List]	—
全ブレーク条件の解除	bac	[Break   Break All Clear]	

## 強制ブレーク

[Key Break]ボタンおよび[ESC]キーは実行中のプログラムを強制的に終了させます。[RUN]メニューから[Stop]を選択して強制終了させることもできます。



[Key Break]ボタン

## ICEのBRKIN端子へのLowレベル入力

ICEのBRKIN端子にLowレベルのパルスが入力されると、プログラムはその立ち上がりエッジでブレークします。

## プログラム実行エラーによるブレーク

ターゲットプログラムの実行時に以下の動作を検出するとブレークします。

- ・ ROM領域へのデータ書き込み
- ・ スタック領域外へのスタック操作
- ・ 未定義領域へのアクセス
- ・ 不当命令(その機種に使用できない命令)を実行

この判定は、パラメータファイルに記述されたメモリなどの情報をもとに行われます。

## 13.8.6 トレース機能

### トレースデータバッファとトレース情報

ICEはトレースデータバッファを持っています。プログラムを実行するとバスサイクルごとのトレース情報がこのバッファに取り込まれます。トレースデータバッファは8,192サイクル分の容量があり、トレース情報がこの容量を越えると、古いデータから上書きしていきます。したがって、トレースデータバッファには常に8,192サイクル以内のトレース情報が記録されています。プログラムの実行によりトレースデータバッファはクリアされ、新しい実行データをトレースします。

Trace																	Memory
INS	P Addr	L Addr	Code	Mnemonic	BA	HL	IX	IY	SP	BR	EP	XP	YP	SC	CC	Memory	
0217	000499	01:0499	93	INC IV	3E84	F828	F828	F060	F7F3	F0	00	00	00	00	00--N-C-	0000	
0218	00049A	01:049A	92	INC IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	00--N-C-	0000	
0219	00049B	01:049B	CF7601	LD [SP+01h],IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	00--N-C-	0000 MW:[00F7F4]=29 MW:[00F7	
0220	00049E	01:049E	B001	LD A,#01h	3E01	F828	F829	F060	F7F3	F0	00	00	00	00	00--N-C-	0000	
0221	0004A0	01:04A0	A6	PUSH IP	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	00--N-C-	0000 MW:[00F7F2]=00 MW:[00F7	
0222	0004A1	01:04A1	CEC600	LD XP,#00h	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	00--N-C-	0000	
0223	0004A4	01:04A4	CFFA	LD IX,SP	3E01	F828	F7F1	F060	F7F1	F0	00	00	00	00	00--N-C-	0000	
0224	0004A6	01:04A6	CE4802	LD B,[IX+02h]	0401	F828	F7F1	F060	F7F1	F0	00	00	00	00	00--N-C-	0000 MR:[00F7F3]=04	
0225	0004A9	01:04A9	AE	POP IP	0401	F828	F7F1	F060	F7F3	F0	00	00	00	00	00--N-C-	0000 MR:[00F7F1]=00 MR:[00F7	
0226	0004AA	01:04AA	01	ADD A,B	0405	F828	F7F1	F060	F7F3	F0	00	00	00	00	00-----	0000	
0227	0004AB	01:04AB	50	LD L,A	0405	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----	0000	
0228	0004AC	01:04AC	0105	LD B,#05h	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----	0000	
0229	0004AE	01:04AE	42	LD L,A	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----	0000	
0230	0004AF	01:04AF	A6	PUSH IP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	00-----	0000 MW:[00F7F2]=00 MW:[00F7	
0231	0004B0	01:04B0	CEC600	LD XP,#00h	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	00-----	0000	
0232	0004B3	01:04B3	CFFA	LD IX,SP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	00-----	0000	
0233	0004B5	01:04B5	CE4A02	LD [IX+02h],A	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	00-----	0000 MW:[00F7F3]=05	
0234	0004B8	01:04B8	AE	POP IP	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----	0000 MR:[00F7F1]=00 MR:[00F7	
0235	0004B9	01:04B9	3A80	XOR A,#80h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	00--N--	0000	
0236	0004BB	01:04BB	CEB800	XOR B,#80h	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	00--N--	0000	
0237	0004BE	01:04BE	31	CP A,B	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----Z	0000	
0238	0004BF	01:04BF	CEE0CB	JRS LT,CBh	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----Z	0000	
0239	0004C2	01:04C2	F105	JRS 05h	8585	F805	F7F1	F060	F7F3	F0	00	00	00	00	00-----Z	0000	

各命令の実行でトレースデータバッファに取り込まれるトレース情報は次のとおりです。[Trace]ウィンドウの表示に対応させて示します。

INS: 実行サイクル番号(0~8,191、10進数)0000が最も古いトレースデータです。  
P Addr: PCアドレス(16進物理アドレス)  
L Addr: PCアドレス(16進論理アドレス)  
Code: 命令コード(16進表示)  
Mnemonic: 逆アセンブルコード  
BA ~ YP: CPUレジスタ値(16進数)  
SC, CC: コンディションフラグの状態  
Memory: メモリのアクセス内容(コードフェッチを除く)  
MR: メモリリード  
MW: メモリライト  
[<address>] = <data>: アクセスしたアドレスとリード/ライトしたデータ(16進数)

## トレースモード

トレース情報の採取方法の違いにより、2種類のトレースモードが設定されています。

### 全トレースモード

実行したバスサイクルの情報をすべて記録します。このモードでは、常に最新のトレースデータ(最大8,192サイクル分)を得ることができます。

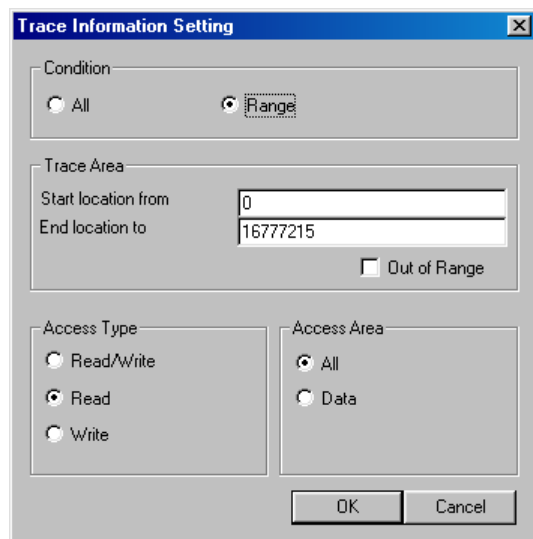
### 範囲指定トレースモード

このモードではメモリアクセス条件が指定でき、その条件と一致したバスサイクルの情報のみが記録されます。

指定するメモリアクセス条件は次のとおりです。

- ・ アドレス範囲、およびその範囲内をトレースするか/範囲外をトレースするか
- ・ プログラムフェッチサイクルおよびデータリード/ライトサイクル、あるいはデータリード/ライトサイクルのみのどちらをトレースするか
- ・ リードサイクルとライトサイクルのどちらをトレースするか(両方可)

トレースモードを設定するには、[Break]メニューから[Setting...]を選択します。



全トレースモードを設定するには、[All]ラジオボタンを選択して[OK]をクリックします。

範囲指定トレースモードを設定するには、[Range]ラジオボタンを選択します。[Start location from]テキストボックスに開始アドレスを、[End location to]テキストボックスに終了アドレスを10進数で入力し、アドレス範囲を指定します。この範囲の外側をトレースさせる場合は[Out of Range]ラジオボタンを選択してください。さらに、[Access Type]のラジオボタンでリード/ライト条件を選択します。[Access Area]のラジオボタンでは、すべてのアクセスをトレース対象とするか( All )、データのリード/ライトのみをトレースするか( Data )選択します。以上を選択し、[OK]ボタンをクリックします。

設定を中止する場合は[Cancel]ボタンをクリックしてください。

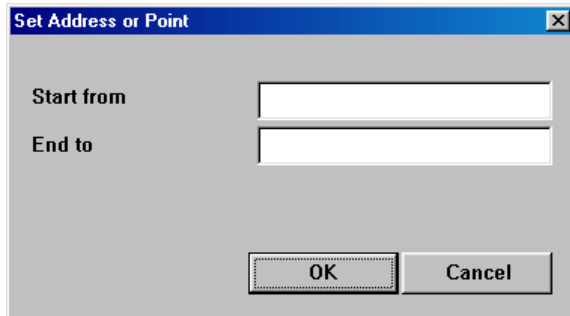
## トレース情報の表示と検索

採取したトレース情報はプログラムの実行を中断すると[Trace]ウィンドウに表示されます。[Trace]ウィンドウではスクロールによってトレースデータバッファの全データを見ることができます。コマンドにより指定したサイクルから表示させることもできます。表示内容は前記のとおりです。[Trace]ウィンドウが閉じている場合はコマンドで[Command]ウィンドウに表示することもできます。

表13.8.6.1 トレース情報表示コマンド

機能	コマンド	メニュー	ボタン
トレース情報の表示	td	[Trace   Trace]	—

[Trace]メニューから[Trace]を選択すると、次のダイアログボックスが表示されます。



[Start from]テキストボックスに表示を開始するサイクル番号を、[End to]テキストボックスに終了サイクル番号を16進数で入力して[OK]ボタンをクリックします。入力を省略した場合、開始サイクルは0、終了サイクルは0x1fff (8191)として処理されます。中止する場合は[Cancel]ボタンをクリックしてください。

検索条件を指定し、条件に合ったトレース情報のみを表示させることができます。

検索条件は次の3種類から選択できます。

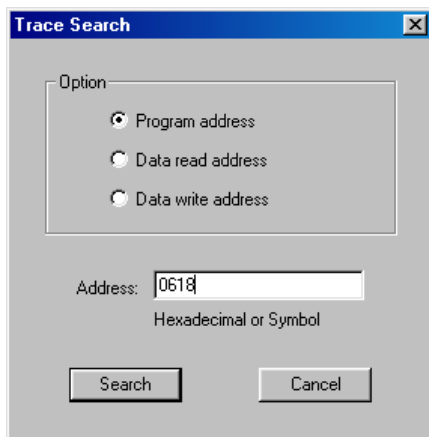
1. プログラムの実行アドレス
2. データを読み出したアドレス
3. データを書き込んだアドレス

上記条件とアドレスを1ヶ所指定して検索を行います。条件に合ったトレース情報が見つかったら、件数を[Command]ウィンドウに表示します。検索データは[Trace]ウィンドウ(閉じている場合は[Command]ウィンドウ)に表示します。

表13.8.6.2 トレース情報検索コマンド

機能	コマンド	メニュー	ボタン
トレース情報の検索	ts	[Trace   Trace Search...]	—

[Trace]メニューから[Trace Search...]を選択すると、[Trace Search]ダイアログボックスが表示されます。



検索条件をラジオボタンで選択し、アドレスを入力して[Search]ボタンをクリックします。検索を中止する場合は[Cancel]ボタンをクリックしてください。

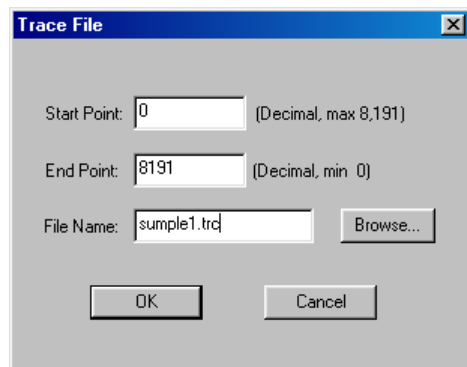
## トレース情報の保存

トレース情報の指定サイクル範囲をファイルに保存することができます。

表13.8.6.3 トレース情報保存コマンド

機能	コマンド	メニュー	ボタン
トレース情報の保存	tf	[Trace   Trace File...]	—

[Trace]メニューから[Trace File...]を選択すると、[Trace File]ダイアログボックスが表示されます。



[Start Point]テキストボックスに開始サイクルを、[End Point]テキストボックスに終了サイクルを入力して保存する範囲を指定します。

ファイル名は[File Name]テキストボックスに入力します。[Browse...]ボタンでフォルダ/ファイルを選択することもできます。

### 13.8.7 カバレッジ

ICEはアクセスしたメモリアドレスを記録するカバレッジ機能を持っています。

カバレッジ情報は、デバッガのカバレッジオプションで指定されている取得モードと取得範囲に従って記録されます。

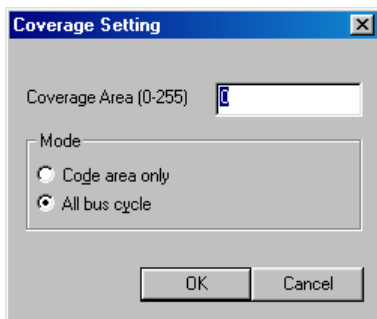
#### 取得モード

コード空間およびデータ空間の両方を対象としてカバレッジ情報を取得するか、コード空間のアクセスのみを対象とするか指定できます。デフォルトは両空間が対象です。

#### 取得範囲

ICEは16MBのアドレス空間を64KB×256の領域に分割し、個々の64KBを対象にカバレッジ情報を取得します。デフォルトは0x000000～0x00FFFFの64KBが対象です。したがって、他の領域のカバレッジ情報を取得するには、プログラム実行前にその領域を指定しておく必要があります。

カバレッジオプションを設定するには、[Coverage]メニューから[Setting...]を選択します。



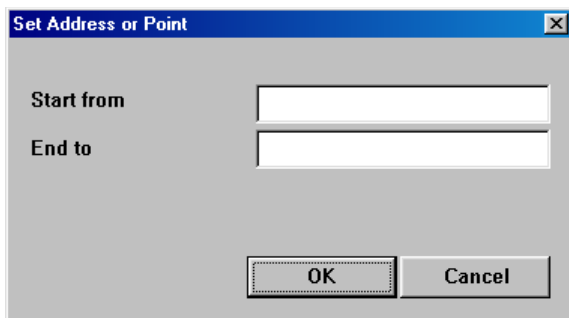
[Coverage Area (0-255)]テキストボックスに0～255の数値を入力して取得範囲を指定します。取得モードはラジオボタンで選択してください。[OK]ボタンをクリックすると設定されます。設定を中止する場合は[Cancel]ボタンをクリックしてください。

取得したカバレッジ情報は、[Coverage]ウィンドウに表示させることができます。

表13.8.7.1 カバレッジコマンド

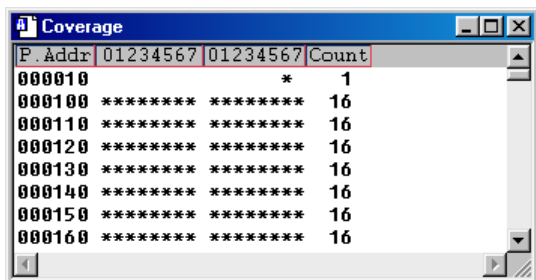
機能	コマンド	メニュー	ボタン
カバレッジ情報の表示	cv	[Coverage   Coverage]	–
カバレッジ情報のクリア	cvc	[Coverage   Coverage Clear]	–

[Coverage]メニューから[Coverage]を選択すると[Coverage]ウィンドウが開き、次のダイアログボックスが表示されます。



[Start from]テキストボックスに表示を開始するアドレスを16進数で入力して[OK]ボタンをクリックします。[Coverage]ウィンドウに表示させる場合、特に[End to]を入力する必要はありません。入力を省略した場合、開始アドレスは設定されている64KB領域の先頭アドレス、終了アドレスは64KB領域の終了アドレスとして処理されます。中止する場合は[Cancel]ボタンをクリックしてください。

[Coverage]ウィンドウには次のように表示されます。



P.Addr	01234567	01234567	Count
000010		*	1
000100	*****	*****	16
000110	*****	*****	16
000120	*****	*****	16
000130	*****	*****	16
000140	*****	*****	16
000150	*****	*****	16
000160	*****	*****	16

16バイト/行でカバレッジ情報を表示します。P.Addrは各行の先頭アドレス(物理アドレス)です。アクセスしたアドレスは"\*"で、未アクセスアドレスは" "(スペース)で示されます。Countの数値は、各行の16バイト中でアクセスしたアドレスの合計(バイト数)です。

[Coverage]ウィンドウとは別に、[Source]ウィンドウの実行したアドレスの前に\*が表示されます(ソース表示モードを除く)。

[Coverage]ウィンドウが閉じている状態でcvコマンドを実行した場合は、[Command]ウィンドウに次のように表示されます。

例: >cv 0  
 00001e  
 000100 - 00010f  
 :



### 13.8.8 標準ペリフェラルボードのFPGAデータ書き込み

標準ペリフェラルボードS5U1C88000Pは、ボード上のFPGAにデータを書き込むことによってサポートしている各機種の周辺機能が設定されます。標準ペリフェラルボードを最初に使用するとき、あるいは新機種の開発を始める前にはこのデータ書き込みが必要です。

デバッグは、ICEに装着した標準ペリフェラルボードのFPGAを消去したり、データを書き込む機能を持っています。

FPGAに対しては以下の操作が行えます。

(1) FPGAの消去

FPGAの内容をすべて消去します。

(2) FPGAへのデータ書き込み

指定したファイルのデータをFPGAに書き込みます。書き込みコマンドでFPGAの消去も行えます。

サポート機種のデータは"%epson%s1c88%ice%fpga"ディレクトリ(デフォルト)に"c88xxx.mot"ファイルとして用意されています。

(3) FPGAデータのコンペア

FPGAのデータと指定ファイルの内容を比較します。

(4) FPGAデータダンプ

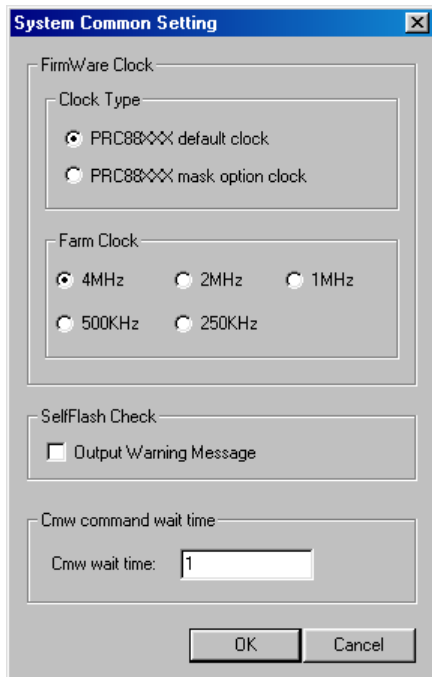
FPGAのデータを16進ダンプ形式で表示します。

表13.8.8.1 FPGAコマンド

機 能	コマンド	メニュー	ボタン
FPGAの消去	xfer	—	—
FPGAデータ書き込み	xfwr	—	—
FPGAデータコンペア	xfcp	—	—
FPGAデータダンプ	xdp	—	—

### 13.8.9 システムオプション

[Option]メニューから[Setting...]を選択して表示される[System Common Setting]ダイアログボックスは、ICEハードウェアにかかわる設定に使用します。



#### Clock Type

エミュレーションに使用するクロックを次の2種類から選択することができます。

- (1) ペリフェラルボードのデフォルトクロック(デフォルト)
- (2) ペリフェラルボードのマスクオプションクロック

ペリフェラルボード( PRC88XXX )のデフォルトクロックを選択すると、マスクオプションの設定にかかわらずペリフェラルボード上のクロックがエミュレーションクロックとして使用されます。機種によっては、このクロック選択ができません。

クロック周波数については、各機種のテクニカルマニュアルを参照してください。

#### Firm Clock

ICEのファームウェアクロックを次の5種類から選択することができます。

- (1) 4MHz( デフォルト設定 )
- (2) 2MHz
- (3) 1MHz
- (4) 500kHz
- (5) 250kHz

ファームウェアクロックは、ICEがデバッグ機能を実行するために使用します。たとえば、メモリダンプは、ファームウェアクロックを使用して行います。このため、ターゲットボードに低速デバイスや、データ出力に遅延の生じるデバイスなどを使用している場合、メモリダンプ内容と、プログラム実行により読み出した内容が同一とならないことが考えられます。このような場合は、ファームウェアクロックを低い周波数に設定してください。

#### SelfFlash Check

自己書き換えチェック機能のON/OFFを行います。自己書き換えチェック機能はパラメータファイル内の記述に従って設定されますが、ここでそれを切り換えることができます。

#### Cmw command wait time

cmwコマンドでコマンドファイルを読み込んで実行する際の、コマンド実行間隔を指定します。1～256秒の範囲( 1秒単位 )で設定できます。初期設定は1秒です。

## 13.9 コマンドリファレンス

### 13.9.1 コマンド一覧

表13.9.1.1にデバッグコマンドの一覧を示します。

表13.9.1.1 コマンド一覧表

分類	コマンド	機能	P
メモリ操作	<b>dd</b> [ <code>&lt;addr1&gt;</code> [ <code>&lt;addr2&gt;</code> ]] [{-B -W -L -F -D}] [ <code>&lt;addr1&gt;</code> <code>&lt;@size&gt;</code> ] [{-B -W -L -F -D}]	メモリダンプ	137
	<b>de</b> [ <code>&lt;addr&gt;</code> <code>&lt;data1&gt;</code> ] [ <code>..&lt;data16&gt;</code> ]	データの入力	140
	<b>df</b> [ <code>&lt;addr1&gt;</code> <code>&lt;addr2&gt;</code> <code>&lt;data&gt;</code> ]	領域のフィル	142
	<b>dm</b> [ <code>&lt;addr1&gt;</code> <code>&lt;addr2&gt;</code> <code>&lt;addr3&gt;</code> ] [ <code>&lt;addr1&gt;</code> <code>&lt;@size&gt;</code> <code>&lt;addr3&gt;</code> ]	領域のコピー	143
	<b>ds</b> <code>&lt;addr1&gt;</code> [ <code>&lt;addr2&gt;</code> ] <code>&lt;@byte&gt;</code> ... ...[" <code>&lt;str&gt;</code> " <code>&lt;data&gt;</code> [-{B W L}]] [ <code>S=&lt;step&gt;</code> ]	データの検索	144
レジスタ操作	<b>rd</b>	レジスタの表示	145
	<b>rs</b> [ <code>&lt;reg&gt;</code> <code>&lt;value&gt;</code> ]	レジスタ値の変更 reg={PC SP IX IY A B HL BR CB EP XP YP SC I1 I0 U D N V Z C}	146
プログラム実行	<b>g</b> [ <code>&lt;addr&gt;</code> ]	カレントPCから連続実行	148
	<b>gr</b> [ <code>&lt;addr&gt;</code> ]	CPUリセット後に連続実行	150
	<b>s</b> [ <code>&lt;step&gt;</code> ]	カレントPCからシングルステップ実行	151
	<b>n</b> [ <code>&lt;step&gt;</code> ]	関数/サブルーチン以外をシングルステップ実行	153
	<b>se</b>	関数/サブルーチン終了	154
CPUリセット	<b>rst</b>	CPUをリセット	155
ブレイク	<b>bp</b> {- + _} <code>&lt;addr&gt;</code>	ソフトウェアブレイクポイントの設定	156
	<b>bpa</b> <code>&lt;addr1&gt;</code> <code>&lt;addr2&gt;</code>	ソフトウェアブレイク領域の設定	158
	<b>bpr</b>	ソフトウェアブレイクポイントの解除	160
	<b>bc</b> [ <code>&lt;addr&gt;</code> ]		
	<b>bpc</b> [ <code>&lt;addr&gt;</code> ]		
	<b>bas</b> {0 1 2 3}	シーケンシャルブレイクモードの設定	161
	<b>ba</b> <code>&lt;ch&gt;</code> <code>&lt;addr&gt;</code> [ <code>&lt;count&gt;</code> ] <code>&lt;ch&gt;</code> {- + _}	ハードウェアブレイクポイントの設定	162
	<b>bar</b>	ハードウェアブレイクポイントの解除	164
	<b>bd</b> <code>&lt;ch&gt;</code> [ <code>A=&lt;addr&gt;</code> ][ <code>D=&lt;data&gt;</code> ][{R W}] <code>&lt;ch&gt;</code> {- + _}	ハードウェアデータブレイク条件の設定	165
	<b>bdr</b>	ハードウェアデータブレイク条件の解除	167
	<b>bl</b>	全ブレイク条件の表示	168
	<b>bac</b>	全ブレイク条件の解除	169
プログラム表示	<b>u</b> [ <code>&lt;addr&gt;</code> ]	逆アセンブル表示	170
	<b>sc</b> [ <code>&lt;addr&gt;</code> ]	ソース表示	172
	<b>m</b> [ <code>&lt;addr&gt;</code> ]	ミックス表示	174
シンボル情報	<b>sy</b> [/a]	シンボル一覧の表示	176
	<b>w</b> <code>&lt;symbol&gt;</code> [{:H D Q B}] [/A]	シンボル情報の表示	177
ファイルロード	<b>lf</b> [ <code>&lt;file&gt;</code> ]	プログラム/オプションHEXファイルのロード	178
	<b>par</b> [ <code>&lt;file&gt;</code> ]	パラメータファイルのロード	179
トレース	<b>td</b> [ <code>&lt;cycle&gt;</code> ]	トレース情報の表示	180
	<b>ts</b> [{pc dr dw} <code>&lt;addr&gt;</code> ]	トレース情報の検索	183
	<b>tf</b> [ <code>&lt;file&gt;</code> ] [ <code>&lt;cycle1&gt;</code> ] [ <code>&lt;cycle2&gt;</code> ]]	トレース情報の保存	185
カバレッジ	<b>cv</b> [ <code>&lt;addr1&gt;</code> <code>&lt;addr2&gt;</code> ]	カバレッジ情報の表示	186
	<b>cvc</b>	カバレッジ情報のクリア	188
コマンド ファイル	<b>com</b> [ <code>&lt;file&gt;</code> ] [ <code>&lt;interval&gt;</code> ]	コマンドファイル読み込み/実行	189
	<b>cmw</b> [ <code>&lt;file&gt;</code> ]	ウェイト付きコマンドファイル読み込み/実行	190
	<b>rec</b> [ <code>&lt;file&gt;</code> ]	実行コマンドの記録	191
ログ	<b>log</b> [ <code>&lt;file&gt;</code> ]	ログ出力	192
マップ情報	<b>ma</b>	マップ情報の表示	193
FPGA操作	<b>xfer</b>	FPGAの消去	194
	<b>xfwr</b> <code>&lt;file&gt;</code> ;{H S} [:N]	FPGAデータ書き込み	195
	<b>xfcp</b> <code>&lt;file&gt;</code> ;{H S}	FPGAデータコンペア	196
	<b>xdp</b> <code>&lt;addr1&gt;</code> [ <code>&lt;addr2&gt;</code> ]	FPGAデータダンプ	197
終了	<b>q</b>	デバッグ終了	198
ヘルプ	<b>?</b>	コマンドusageの表示	199

### 13.9.2 各コマンド説明の見方

次項よりすべてのデバッグコマンドを機能別に説明します。  
各コマンドの説明項目を以下に示します。

#### 機 能

コマンドの機能が記述されています。

#### 書 式

キーボードからのコマンド入力形式、およびパラメータの内容が記述されています。

#### 例

コマンドの実行例が記述されています。

#### 注

使用上の注意事項や補足説明が記述されています。

#### G U I

コマンドを実行可能なメニューやボタン、パラメータを指定するダイアログボックスを示します。

- 注:
- 書式の記述で、<>で囲まれたパラメータはユーザによって指定される内容です。[]で囲まれたパラメータは省略可能であることを示します。
  - ユーザ定義シンボルを除き、コマンドは大文字と小文字が区別されません。大文字、小文字のどちらでも、また混在した形でも入力できます。
  - 直接入力モードでコマンドを実行する場合、必要なパラメータがすべて入力されないエラーになります。

Error : Incorrect number of parameters

### 13.9.3 メモリ操作コマンド

## dd (data dump)

#### 機能

メモリの内容を16ワード/行の16進ダンプ形式で表示します。

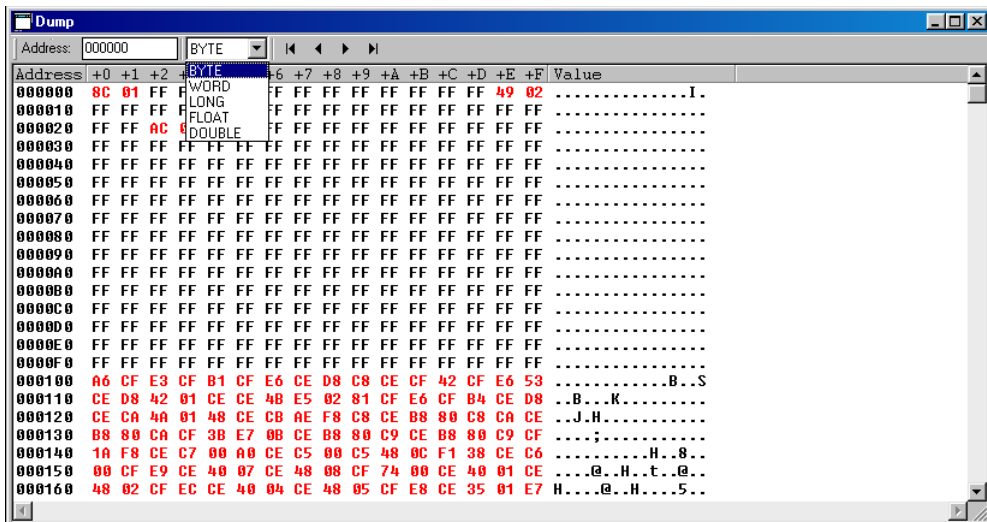
#### 書式

- (1) >dd [<address1> [<address2>]] [<option>];  
( 直接入力モード )
- (2) >dd [<address1> @<size>] [<option>];  
( 直接入力モード )
- <address1>: 表示開始アドレス      16進数、またはシンボル( IEEE-695形式のみ )  
<address2>: 表示終了アドレス      16進数、またはシンボル( IEEE-695形式のみ )  
<size>:      表示サイズ( バイト数 ) 16進数  
<option>:      表示形式オプション      次のシンボルで指定
- B      Byte形式 ( デフォルト )
  - W      Word形式
  - L      Long形式
  - F      Float形式
  - D      Double形式

条件:      0   address1   address2   0xffffffff、0   size   0xffffffff

#### 表示

- (1) [Dump]ウィンドウを開いている場合



<address1>と<address2>を省略した場合、[Dump]ウィンドウはアドレス0x000000からデータを表示します。

<address1>を指定すると、<address1>が最上部の行に表示されるように[Dump]ウィンドウを再表示します。<address1>が16アドレス/行の途中を指定していても、データはその行の先頭から表示されます。たとえば、<address1>に0x00ff08を指定した場合でもデータはアドレス0x00ff00から表示されます。ただし、<address1>に0xffffc0などのメモリ最上部付近のアドレスを指定した場合は、ウィンドウの最終行が0xfffff0となり、指定のアドレスがウィンドウの先頭部とはなりません。

[Dump]ウィンドウは、スクロールによってデータメモリ全体を表示可能なため、<address2>および@<size>の指定は特に意味を持ちません。<address1>のみ指定した場合も、<address1>と<address2>または@<size>の両方を指定した場合もまったく同じ表示となります。

## (2) [Dump]ウィンドウを閉じている場合

<address1>と<address2>を省略した場合、アドレス0x000000から256ワードのデータを[Command]ウィンドウに表示します。

```
>dd↓
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
000000 AE 02 F0 F0 C9 02 F0 F0 F0 F0 F0 F0 F0 F0 .....
000010 00 A4 E0 48 0A 08 E0 80 EE 6A FC BA 3E BA 4A 01 ...H....j..>.J.
:                                     :
0000F0 A6 A2 22 82 A0 0C 04 02 FE F7 BD 9E FE 7F BA FB ..".....
>
```

<address1>のみを指定した場合、<address1>から256ワードのデータを表示します。

```
>dd ff00↓
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
00FF00 30 00 00 FF FF FF FF FF FF FF FF FF FF FF FF 0.....
00FF10 00 00 1F 00 FF FF FF FF FF FF FF FF FF FF FF .....
:                                     :
00FFFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
>
```

<address1>と<address2>の両方を指定すると、<address1>から<address2>までのデータを表示します。

```
>dd ff00 ff1f↓
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
00FF00 30 00 00 FF FF FF FF FF FF FF FF FF FF FF FF 0.....
00FF10 00 00 1F 00 FF FF FF FF FF FF FF FF FF FF FF .....
>
```

<address2>の代わりに@<size>を指定すると、表示開始アドレスから指定バイト分のデータを表示します。

```
>dd ff00 @20↓
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
00FF00 30 00 00 FF FF FF FF FF FF FF FF FF FF FF FF 0.....
00FF10 00 00 1F 00 FF FF FF FF FF FF FF FF FF FF FF .....
>
```

## (3) 表示形式オプション

表示形式オプションは[Dump]ウィンドウのプルダウンリストによる選択と同様の機能を指定します。オプション指定を省略した場合はバイト形式で表示されます。

以下、各オプション選択による表示例を示します。

```
>dd -b↓ ... Byte形式(デフォルト)
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
000000 AE 02 F0 F0 C9 02 F0 F0 F0 F0 F0 F0 F0 F0 .....
:                                     :

>dd -w↓ ... Word形式
Address +0 +2 +4 +6 +8 +A +C +E Value
000000 02AE F0F0 02C9 F0F0 F0F0 F0F0 F0F0 F0F0 .....
:                                     :

>dd -l↓ ... Long形式
000000 F0F002AE F0F002C9 F0F0F0F0 F0F0F0F0 .....
:                                     :

>dd -f↓ ... Float形式
000000 AE 02 F0 F0 -5.942371e+029
000004 C9 02 F0 F0 -5.942382e+029
:                                     :

>dd -d↓ ... Double形式
000000 AE 02 F0 F0 C9 02 F0 F0 -1.018151011077231e+236
000008 F0 F0 F0 F0 F0 F0 F0 F0 -1.077308742674321e+236
:                                     :
>
```

## (4) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Dump]ウィンドウが開いている場合でも [Commad]ウィンドウにデータを表示し、その内容をログファイルにも出力します。

[Dump]ウィンドウが閉じている場合、[Command]ウィンドウへの表示は上記(2)と同様です。

[Dump]ウィンドウが開いていれば、上記(1)と同様にその再表示も行います。この場合、[Command]ウィンドウに表示される行数は [Dump]ウィンドウの表示行数と同じになります。

## (5) 連続表示機能

ddコマンドを一度実行すると、他のコマンドを実行するまでは[Enter]キーの入力のみでデータを連続して表示することができます。

[Enter]キーを入力すると、[Dump]ウィンドウは1画面分スクロールします。

[Commad]ウィンドウにデータを表示している場合は、前回表示したアドレスに続く16行分(ログ出力中は [Dump]ウィンドウと同じ行数)を表示します。

```
>dd,↓
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F Value
000000 AE 02 F0 F0 C9 02 F0 F0 F0 F0 F0 F0 F0 F0 .....
000010 00 A4 E0 48 0A 08 E0 80 EE 6A FC BA 3E BA 4A 01 ...H.....j..>.J.
      :                               :
0000F0 A6 A2 22 82 A0 0C 04 02 FE F7 BD 9E FE 7F BA FB ..".....
>↓
000100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
000110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
      :           :           :
0001F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
>
```

**注**

- アドレスは、各機種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 先頭アドレスが終了アドレスより大きい場合、エラーとなります。

**GUI**

[View ! Dump]メニューコマンド

このメニューコマンドを選択することによって、[Dump]ウィンドウがアクティブになり、現在のメモリの内容を表示します。

## de (data enter)

### 機能

16進データを入力してメモリの内容を書き換えます。指定したアドレスから連続してデータの書き込みが行えます。

### 書式

```
(1)>de <address> <data1> [<data2> [...<data16>]]  
                                     ( 直接入力モード )  
(2)>de  
Data enter address ? : <address>  
アドレス 現在のデータ : <data>  
.....  
>  
    <address>:   書き込み開始アドレス  16進数、またはシンボル( IEEE-695形式のみ )  
    <data( 1-16 )>: 書き込みデータ      16進数  
条件:          0 address 0xffffffff, 0 data 0xff
```

### 例

書式1)>de ff10 0 ...アドレス0xff10にデータ0を書き込み

```
書式2)>de  
Data enter address ? :ff10 ...開始アドレスを入力  
00FF10 00 : a ...データを入力  
00FF11 00 :  ...入力をスキップ  
00FF12 00 : q ...コマンドを終了  
>
```

### 注

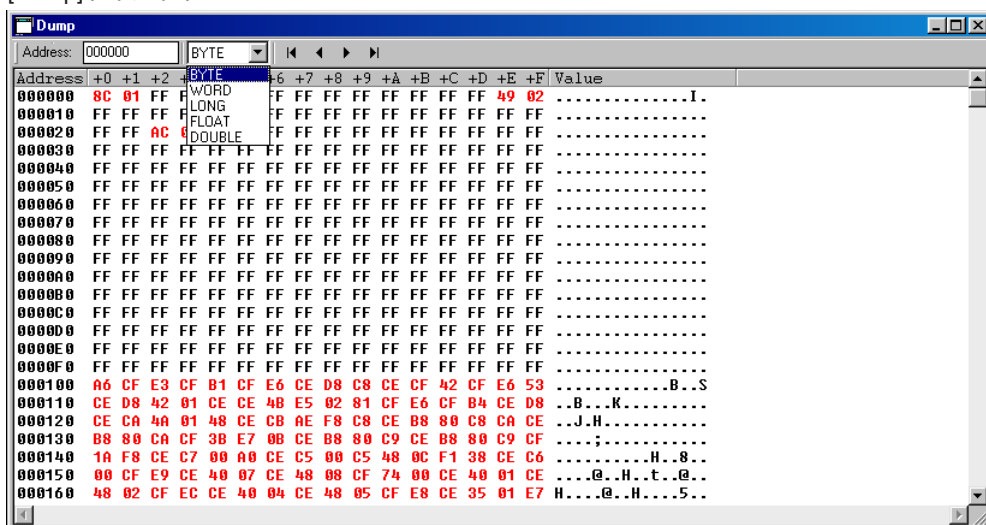
- 開始アドレスは、各機種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 未使用アドレスは、"\*"を表示します。"\*"が表示されたアドレスは、[Enter]キーでそのアドレスをスキップするか、コマンドを終了させてください。
- データは8ビット(0~0xff)の範囲内の16進数で入力してください。これを越えるとエラーとなります。
- deコマンドでメモリの内容を変更すると、[Dump]ウィンドウの表示内容は自動的に更新されます。
- ガイダンスモードでは、以下のキー操作も有効です。  
"q" ...コマンドを終了(入力を終了し、コマンド機能を実行)  
"h" ...直前のアドレスに戻る  
"j" ...入力をスキップ(現在の内容を保持)

メモリの最終アドレスに達した場合、"h"以外の有効な入力を行うと、コマンドは終了します。



## GUI

## [Dump]ウィンドウ



[Dump]ウィンドウ上で、データメモリを直接変更することができます。変更するデータの直前にカーソルを置くか、データをダブルクリック後、16進の数値(0~9, a~f)を入力してください。そのアドレスのデータが変更されます。カーソルは次のアドレスのデータに移動し、連続的なデータ変更を可能にしています。

## df (data fill)

### 機能

指定のメモリ領域の内容すべてを指定のデータに書き換えます。

### 書式

(1) >df <address1> <address2> <data>␣ (直接入力モード)

(2) >df␣ (ガイダンスモード)

Start address ? <address1>␣

End address ? <address2>␣

Data pattern ? <data>␣

>

<address1>: 指定範囲先頭アドレス 16進数、またはシンボル(IEEE-695形式のみ)

<address2>: 指定範囲終了アドレス 16進数、またはシンボル(IEEE-695形式のみ)

<data>: 書き込みデータ 16進数

条件: 0 address1 address2 0xffffffff, 0 data 0xff

### 例

書式1) >df ff200 ff2ff 0␣ ...アドレス0xff200 ~ 0xff2ffの範囲を0x0で書き換え

書式2) >df␣

Start address ? ff200␣ ...先頭アドレスを入力

End address ? ff2ff␣ ...終了アドレスを入力

Data pattern ? 0␣ ...書き込みデータを入力

>

[Enter]キーのみの入力でコマンドの実行を中止できます。

### 注

- アドレスは、各機種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 先頭アドレスが終了アドレスより大きい場合、エラーとなります。
- データは8ビット(0~0xff)の範囲内の16進数で入力してください。これを越えるとエラーとなります。
- I/O領域の読み出し専用アドレスに対して書き込みは行われません。
- 指定アドレス範囲に未使用領域が含まれている場合、未使用領域以外への書き込みは行われ、エラーとはなりません。
- dfコマンドでデータメモリの内容を変更すると、[Dump]ウィンドウの表示内容は自動的に更新されます。

### GUI

なし

## dm (data move)

### 機能

指定のデータメモリ領域の内容を別の領域にコピーします。

### 書式

- (1) >dm <address1> <address2> <address3>↵ (直接入力モード)
- (2) >dm <address1> @<size> <address3>↵ (直接入力モード)
- (3) >dm↵ (ガイダンスモード)
- Start address ? <address1>↵  
 End address ? <address2>↵  
 Destination address ? <address3>↵  
 >
- |                         |                              |
|-------------------------|------------------------------|
| <address1>: コピー元の先頭アドレス | 16進数、またはシンボル( IEEE-695形式のみ ) |
| <address2>: コピー元の終了アドレス | 16進数、またはシンボル( IEEE-695形式のみ ) |
| <address3>: コピー先のアドレス   | 16進数、またはシンボル( IEEE-695形式のみ ) |
| <size>: コピー元サイズ( バイト数 ) | 16進数                         |
- 条件: 0 address1 address2 0xffffffff、0 address3 0xffffffff、0 size 0xffffffff

### 例

- 書式1) >dm ff200 ff2ff ff280↵ ... アドレス0xff200 ~ 0xff2ffの範囲をアドレス0xff280から始まる領域にコピー
- 書式2) >dm ff200 @100 ff280↵ ... 書式1と同様
- 書式3) >dm↵
- |                              |                    |
|------------------------------|--------------------|
| Start address ? ff200↵       | ... コピー元の先頭アドレスを入力 |
| End address ? ff2ff↵         | ... コピー元の終了アドレスを入力 |
| Destination address ? ff280↵ | ... コピー先のアドレスを入力   |
- >

[Enter]キーのみの入力でコマンドの実行を中止できます。

### 注

- アドレスは、各機種種のメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- I/O領域の読み出し専用アドレスに対して書き込みは行われません。
- 書き込み専用アドレスのデータは読み出せません。コピー元の領域が書き込み専用アドレスを含んでいる場合、対応するコピー先アドレスには0が書き込まれます。コピー先の領域が読み出し専用アドレスを含んでいる場合、そのアドレスに対して書き込みは行われません。コピー元またはコピー先に書き込み専用ビットまたは読み出し専用ビットを含むアドレスがある場合、それらのビットに対応した書き込み動作を行います。
- dmコマンドでデータメモリの内容を変更すると、[Dump]ウィンドウの表示内容は自動的に更新されます。

### GUI

なし

## ds (data search)

### 機能

メモリ内の指定範囲から指定のデータまたは文字列を検索し、見つかった場合はそのアドレスを [Command] ウィンドウに表示します。また、[Dump] ウィンドウが表示している範囲に指定のデータが見つかったと、そのデータは緑で表示されます。

### 書式

```
>ds <address1> {<address2>|@<byte>} {"<string>"|<data>[:<size>]} [S=<step>],␣( 直接入力モード )
```

<address1>:	検索範囲先頭アドレス	16進数、またはシンボル( IEEE-695形式のみ )
<address2>:	検索範囲終了アドレス	16進数、またはシンボル( IEEE-695形式のみ )
<byte>:	検索範囲サイズ(バイト数)	16進数
<string>:	検索する文字列	最大4文字のASCIIキャラクタ
<data>:	検索するデータ	<size>で指定するサイズの16進数または2進数 データバイト/ビットを"*"でマスク可能
<size>:	データサイズ	次のシンボルで指定 B バイト( 1バイト ) ( デフォルト ) W ワード( 2バイト ) L ロング( 4バイト )
<step>:	検索するステップ幅( バイト単位 )	省略時はデータサイズ( <size>の指定 )
条件:	0 address1 address2 0xffffffff, address2 address1+0xffff, byte 0x10000	
	1 step 0xffff	

### 例

```
>ds f000 30:W S=10,␣
00F000 00F070
>
```

この例では、アドレス0x00f000から0x0030のワードデータを検索します。16バイトのステップ指定により、16バイト境界アドレス( アドレス0x00f000、0x00f010... )にあるワードデータのみをチェックします。たとえば、0xf002に0x0030のデータがあっても検索結果には現れません。

```
>ds f000 f0ff "ABC",␣
00F022
>
```

この例では、アドレス0x00f000から アドレス0x00f0ffの範囲にある文字列"ABC"( =0x41, 0x42, 0x43 )を検索します。文字列の検索は、バイトステップ( デフォルト )で行われます。

### 注

- アドレスは、各機種種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 検索範囲は64KB以内です。この範囲を越える指定はエラーとなります。

### GUI

なし

### 13.9.4 レジスタ操作コマンド

## rd (register display)

#### 機能

CPUレジスタの内容を表示します。

#### 書式

>rd␣ (直接入力モード)

#### 表示

##### (1) 表示内容

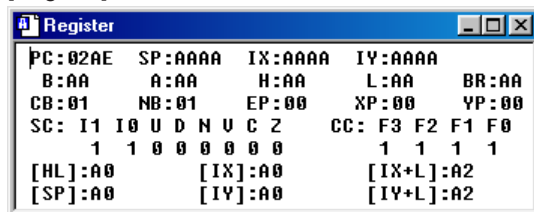
表示する内容は以下のレジスタとレジスタで間接指定されるメモリの値です。

レジスタ: PC, SP, IX, IY, B, A, H, L, BR, SC, CC

メモリ: [HL], [IX], [IX+L], [SP], [IY], [IY+L]

レジスタで指定されるメモリが未使用領域の場合、そのデータは"\*"で表示されます。

##### (2) [Register]ウィンドウを開いている場合



[Register]ウィンドウを開いている場合は、プログラムの実行終了後に上記の内容がすべて[Register]ウィンドウに表示されます。rdコマンドは[Register]ウィンドウの表示を更新します。

##### (3) [Register]ウィンドウが閉じている場合

[Command]ウィンドウに次のようにデータを表示します。

```

>rd␣
PC:02AE  SP:AAAA  IX:AAAA  IY:AAAA
B:AA     A:AA     H:AA     L:AA     BR:AA
CB:01     NB:01    EP:00     XP:00     YP:00
SC:I1 I0 U D N V C Z  CC:F3 F2 F1 F0
      1 1 0 0 0 0 0 0      1 1 1 1
>
  
```

##### (4) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Register]ウィンドウが開いている場合でも [Command]ウィンドウにデータを表示し、その内容をログファイルにも出力します。

#### GUI

[View | Register]メニューコマンド

このメニューコマンドを選択することによって、[Register]ウィンドウがアクティブになり、現在の各レジスタの内容を表示します。

## rs (register set)

### 機能

レジスタの値を変更します。

### 書式

(1) >rs <register> <value>↵ (直接入力モード)

(2) >rs↵ (ガイダンスモード)

PC = 現在の値 : <value>↵

SP = 現在の値 : <value>↵

IX = 現在の値 : <value>↵

IY = 現在の値 : <value>↵

A = 現在の値 : <value>↵

B = 現在の値 : <value>↵

I1 = 現在の値 : <value>↵

I0 = 現在の値 : <value>↵

U = 現在の値 : <value>↵

D = 現在の値 : <value>↵

N = 現在の値 : <value>↵

V = 現在の値 : <value>↵

C = 現在の値 : <value>↵

Z = 現在の値 : <value>↵

HL = 現在の値 : <value>↵

BR = 現在の値 : <value>↵

CB = 現在の値 : <value>↵

EP = 現在の値 : <value>↵

XP = 現在の値 : <value>↵

YP = 現在の値 : <value>↵

>

<register>: レジスタ名( PC, SP, IX, IY, A, B, HL, BR, CB, EP, XP, YP, SC, I1, I0, U, D, N, V, Z, C)

<value>: レジスタの設定値 16進数

### 例

書式1) >rs SC 0↵ ...SCレジスタの全フラグをクリア

書式2) >rs↵

PC=02ae : 180↵

SP=aaaa : f0ff↵

IX=aaaa : f000↵

IY=aaaa : f000↵

A= aa : 0↵

B= aa : 0↵

HL=aaaa : 0↵

BR= aa : 0↵

I1= 0 : 1↵

I0= 0 : 1↵

U= 0 : ↵

D= 0 : ↵

N= 0 : ↵

V= 0 : ↵

C= 0 : ↵

Z= 0 : ↵

CB= 01 : ↵

EP= 00 : ↵

XP= 00 : ↵

YP= 00 : ↵

>

レジスタを変更する場合、[Register]ウィンドウは入力した内容に更新されます。

"q↵"によって途中で入力を中止した場合、それまでに入力した内容は変更されません。

**注**

- レジスタのビット数を越える値を入力するとエラーとなります。
- 直接入力モードで、不正なレジスタ名を入力するとエラーとなります。
- ガイダンスモードでは、以下のキー操作も有効です。
  - "q↵" ...コマンドを終了( 入力を終了し、コマンド機能を実行 )
  - "^↵" ...直前のアドレスに戻る
  - "↵" ...入力をスキップ( 現在の内容を保持 )

**GUI**

[Register]ウィンドウ

[Register]ウィンドウ上で直接データを変更可能です。[Register]ウィンドウ内の変更するデータをダブルクリックして選択し、設定値を入力後に[Enter]キーを押してください。

## 13.9.5 プログラム実行コマンド

### g (go)

#### 機能

現在のPCアドレスまたは指定のアドレスからターゲットプログラムを実行します。

#### 書式

>g [<address>].J (直接入力モード)  
 <address>: ブレークアドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 条件: 0 address プログラムメモリ最終アドレス

#### 動作

##### (1) プログラムの実行

PCが示すアドレスからターゲットプログラムを実行します。プログラムの実行は次のいずれかの要因によってブレークするまで継続します。

- ・ブレーク設定コマンドで設定したブレーク条件が成立
- ・ICEのBRKIN端子への入力
- ・[Key Break]ボタンのクリック、[Run! Stop]メニューコマンドの選択または[ESC]キー入力
- ・プログラム実行エラーの検出

ブレークアドレスを指定すると、プログラムは指定アドレスの命令実行直前にブレークします。

>g 1a0.J ...プログラムは現在のPCアドレスから実行を開始し、アドレス0x1a0の命令を実行する直前に停止します。

プログラムの実行がブレークすると、実行サイクル/時間を表示後コマンド入力待ちとなります。ここで[Enter]キーを入力すると、ブレーク後のPCアドレスからプログラムの実行を再開します。ブレークアドレスの設定も有効です。

##### (2) プログラム実行によるウィンドウの表示

[Source]ウィンドウはブレーク後に、ブレークしたアドレスがウィンドウ内に表示されるように更新されます。

[Trace]ウィンドウが開いている場合は、プログラムの実行により表示内容がクリアされます。ブレーク後に、新しいトレース情報が表示されます。

[Dump]ウィンドウ、[Register]ウィンドウが開いている場合、表示内容はブレーク後に更新されます。

[Watch]ウィンドウを[Run! Setting...]でショートブレークモードに設定している場合、プログラム実行中は、[Watch]ウィンドウの表示が指定の周期で更新されます。

##### (3) ログモード時の表示

ログモードをONにしてプログラムを実行した場合、ブレーク時は実行サイクル/時間の後にrdコマンド実行時と同じ内容が[Command]ウィンドウに表示されます。

例: >g  
 BUS CYCLE : 86519  
 Mode L : 004s 036ms 943us  
 OK!  
 PC:0618 SP:F7FE IX:21F8 IY:F1E4  
 B:01 A:05 H:F1 L:E4 BR:F0  
 CB:01 NB:01 EP:00 XP:04 YP:00  
 SC:I1 I0 U D N V C Z CC:F3 F2 F1 F0  
 0 0 0 0 0 0 0 0 0 0 0 0  
 >

ブレーク時はrdコマンドによる場合と同じ表示を行います。

##### (4) 実行サイクルカウンタ

ターゲットプログラムが停止した後、[Command]ウィンドウに実行サイクル数および実行時間が表示されます。(詳細は13.8.4項参照)

実行サイクルカウンタは、gコマンド発行ごとにリセットされます。



**注**

- ブレーク条件が成立すると、プログラムは実行を中断します。このときのPCアドレスはブレークポイントのアドレスとなります。
- アドレスは、各機種種のプログラムメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。

**GUI**


[Run | Go]メニューコマンド、[Go]ボタン

このメニューコマンドまたはボタンを選択することによって、gコマンド(ブレークアドレス指定なし)が実行されます。

 [Go]ボタン

[Run | Go to Cursor]メニューコマンド、[Go to Cursor]ボタン

[Source]ウィンドウ内のブレークさせるアドレスの行にカーソルを置いてこのメニューコマンドまたはボタンを選択することによって、ブレークアドレス指定付きのgコマンドが実行されます。プログラムは、カーソル位置の命令を実行直前に中断します。

 [Go to Cursor]ボタン

## gr (go after reset CPU)

---

### 機能

CPUをリセットし、ブートアドレスからターゲットプログラムを実行します。

### 書式

>gr [<address>]↵ (直接入力モード)  
 <address>: ブレークアドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 条件: 0 address プログラムメモリ最終アドレス

### 動作

プログラムを実行する前にCPUをリセットします。これによりPCがブートアドレスに設定され、そこからプログラムの実行を開始します。

プログラム実行開始後の動作はgコマンドと同様です。ただし、[Enter]キーの入力による実行の再開は行えません。詳細については、gコマンドの説明を参照してください。

### 注

ブレーク条件が成立すると、プログラムは実行を中断します。このときのPCアドレスはブレークポイントのアドレスとなります。

### GUI

[Run | Go after Reset]メニューコマンド、[Go after Reset]ボタン  
 このメニューコマンドまたはボタンを選択することによって、grコマンドが実行されます。



[Go after Reset]ボタン

## s (step)

### 機能

現在のPCからターゲットプログラムをステップ実行します。

### 書式

```
>s [<step>].↓      ( 直接入力モード )
<step>: 実行ステップ数 10進数( デフォルト=1 )
条件: 0 step 65,535
```

### 動作

#### (1) ステップ実行

<step>の指定を省略すると、PCが示すアドレスのプログラムを1ステップ実行します。<step>を指定した場合は、PCが示すアドレスから指定したステップ分のプログラムをステップ実行します。

```
>s.↓              ...PCアドレスの1ステップを実行
>s 20.↓           ...PCアドレスから20ステップを実行
```

プログラムの実行は、次の要因によって指定ステップ数の実行途中でも終了します。

- [Key Break]ボタンのクリックまたは[ESC]キーの入力

指定ステップの実行を終了すると[Register]ウィンドウ内の表示が更新されます。[Register]ウィンドウが閉じている場合は、rdコマンドの実行時と同じ内容を[Command]ウィンドウに表示します。デバグはコマンド入力待ちとなり、ここで[Enter]キーを入力すると、続くアドレスから再度同じステップ実行を行います。

#### (2) HALT、SLEEP状態と割り込み

halt命令またはslp命令を実行するとCPUはスタンバイモードとなり、その解除には割り込みが必要です。デバグでは、シングルスステップ動作用に外部割り込みの許可/禁止モードが設定されています。

	許可モード	禁止モード
外部割り込み	割り込みを処理する	割り込みを処理しない
halt、slp命令	halt命令として実行 外部割り込み、または [Key Break]ボタンで処理を継続	halt、slp命令をnop命令 に置き換えて実行

デバグの初期設定で、割り込み禁止モードに設定されます。

[Run ! Setting...]により割り込み許可モードに設定することもできます。

#### (3) 実行サイクルカウンタ

指定ステップ数の実行終了後、[Command]ウィンドウに実行サイクル数および実行時間が表示されます。( 詳細は13.8.4項参照 )

実行サイクルカウンタは、sコマンド発行ごとにリセットされます。

#### (4) ログモード時

ログモードをONにしてステップ実行した場合、ステップ実行終了時にrdコマンドの実行時と同じ内容を[Command]ウィンドウに表示します。

**注**

- ステップ数は、0～65,535の範囲内で指定してください。これを越えた場合、エラーとなります。
- [Dump]ウィンドウが開いている場合、その表示内容はステップ実行終了後に更新されます。
- 本コマンド実行中は、コマンドで設定したブレーク条件が成立してもブレークしません。

**GUI**

[Run | Step]メニューコマンド、[Step]ボタン

このメニューコマンドまたはボタンを選択することによって、<step>指定なしのsコマンドが実行されます。



[Step]ボタン

## n (next)

---

### 機 能

現在のPCからターゲットプログラムをステップ実行します。

### 書 式

>n [<step>]↵ (直接入力モード)  
 <step>: 実行ステップ数 10進数(デフォルト=1)  
 条件: 0 step 65,535

### 動 作

基本的な動作はsコマンドと同様です。

ただし、コール命令は、次のアドレスに戻るまでのサブルーチンをすべて含めて1ステップとして実行します。

### 注

- ステップ数は、0～65,535の範囲内で指定してください。これを越えた場合、エラーとなります。
- [Dump]ウィンドウが開いている場合、その表示内容はステップ実行終了後に更新されます。
- 本コマンド実行中は、コマンドで設定したブレーク条件が成立してもブレークしません。

### G U I

[Run | Next]メニューコマンド、[Next]ボタン

このメニューコマンドまたはボタンを選択することによって、<step>指定なしのnコマンドが実行されます。



[Next]ボタン

## se (step exit)

---

### 機能

現在のPCからターゲットプログラムをステップ実行し、現在の関数またはサブルーチンからリターンしたところで停止します。

### 書式

>se↓ (直接入力モード)

### 動作

現在のPCからターゲットプログラムをステップ実行し、上位のルーチンにリターンしたところで停止します。

### 注

- 最上位のルーチンでは実行しないでください。
- [Dump]ウィンドウが開いている場合、その表示内容はステップ実行終了後に更新されます。
- シングルステップ動作中は、コマンドで設定したブレーク条件が成立してもブレークしません。

### GUI

[Run | Step Exit]メニューコマンド、[Step Exit]ボタン

このメニューコマンドまたはボタンを選択することによって、seコマンドが実行されます。



[Step Exit]ボタン

### 13.9.6 CPUリセットコマンド

#### rst (reset CPU)

---

##### 機 能

CPUをリセットします。

##### 書 式

>rst,␣ (直接入力モード)

##### 注

- 各レジスタとフラグは以下のように設定されます。  
 PC: リセット例外処理によって、0バンクのメモリの先頭(000000H~000001H)に格納されている値がPCにロードされます。  
 SP, IX, IY: 0xAAAA  
 B, A, H, L, BR: 0xAA  
 CB, NB: 0x01  
 EP, XP, YP: 0x00  
 SC: 0b11000000  
 CC: 0b1111  
 内蔵RAMおよび外部RAMはイニシャルリセット時に初期化されません。  
 内蔵のI/Oメモリについては、それぞれの初期値に設定されます。
- \* リセット例外処理によって、0バンクのメモリの先頭(000000H~000001H)に格納されている値がPCにロードされます。また、このとき同時にNBの初期値01HがCBにロードされます。
- [Source]ウィンドウが開いている場合はブートアドレスから再表示されます。[Register]ウィンドウが開いている場合は上記の内容で再表示されます。
- メモリ内容、ブレークやトレースなどのデバッグステータスはリセットされません。

##### GUI

[Run | Reset CPU]メニューコマンド、[Reset]ボタン

このメニューコマンドまたはボタンを選択することによって、rstコマンドが実行されます。



[Reset]ボタン

### 13.9.7 ブレーク設定コマンド

#### bp (software break point set)

##### 機能

プログラムを指定アドレスで停止させるソフトウェアブレークポイントの設定と解除を行います。1MBのブレーク有効領域内に設定されている有効なソフトウェアブレークポイントの命令をフェッチすると、その命令を実行する直前にブレークが発生します。

##### 書式

```
>bp [<condition>] <address>、 ( 直接入力モード )
<condition>: 設定の解除、有効化、無効化の指定
- ブレークポイントを解除
+ ブレークポイントを有効に設定( デフォルト )
_ ブレークポイントを無効に設定
<address>: ブレークアドレス 16進数、またはシンボル( IEEE-695形式のみ )
条件: 0 address プログラムメモリ最終アドレス( 0x7ffff )
```

##### 例

```
>bp 200、 ...ブレークポイントをアドレス0x200に設定
>bp _ 200、 ...アドレス0x200のブレークポイントを無効に設定
>bp - 200、 ...アドレス0x200のブレークポイントを解除
```

##### 注

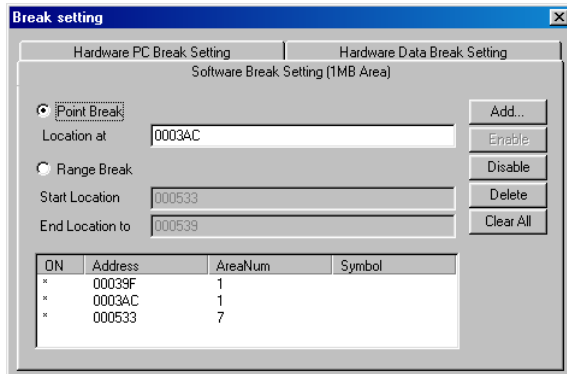
- デバッグの動作環境として設定されている1MBのブレーク有効領域を外れたアドレスを指定した場合、無効なブレークポイントとしてアドレスは登録されますが、そこでブレークが発生させることはできません。ブレーク有効領域は8MBのコード空間を1MBずつ分割した8領域から1つをブレークオプションとして選択できるようになっています( [Break | Setting...]で選択 )。デバッグの起動時には0x0 ~ 0xfffffの1MBに設定されます。
- 設定可能なブレークポイント数は最大64ヶ所です。これを越えた場合、ワーニングとなります。
- アドレスは、各機種種のプログラムメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- すでにブレークポイントに設定されているアドレスを再設定しようとするとワーニングになります。
- 設定されていないアドレスを解除しようとするとエラーになります。
- ブレークポイントには、命令の先頭アドレスを指定してください。途中のアドレスを指定するとブレークは発生しません。
- bpaコマンドで設定したソフトウェアブレーク領域内には、個別のブレークポイントを設定することはできません。指定位置が重複するとエラーになります。
- プログラム/パラメータファイルをロードすると、ブレークの設定内容がすべてクリアされます。



## GUI

## [Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。以下の操作は[Software Break Setting (1MB Area)]タブの画面を表示させて行ってください。



ソフトウェアブレークポイントを設定するには、[Point Break]ラジオボタンを選択し[Location at]テキストボックスにアドレスを入力します。[Add]ボタンをクリックすると、有効なブレークポイントとして登録されます。64ヶ所まではリストに追加されますが、それを越えるとワーニングになります。その場合は先に不要なブレークポイントを削除してください。

有効なブレークポイント(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのアドレスを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレークポイントは無効になります。

無効なブレークポイントを有効にするには、ブレークポイントリストからそのアドレスを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレークポイントを解除するには、ブレークポイントリストからそのアドレスを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、ソフトウェアブレーク領域も含め、設定されているすべてのブレークポイントを解除します。

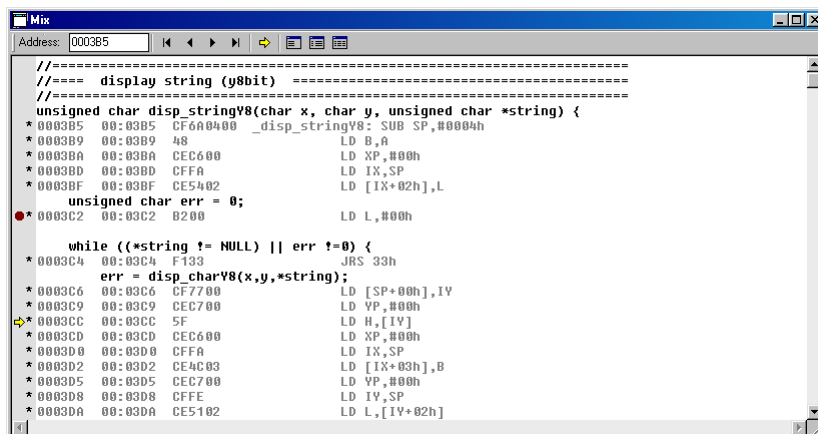
## [Break]ボタン

[Source]ウィンドウ内で、ブレークポイントを設定するアドレスの行をクリック(カーソルを移動)して[Break]ボタンをクリックしてください。その行がブレークポイントに設定されます。逆に、ブレークポイントに設定されている行をクリックして[Break]ボタンをクリックすると、そのブレークアドレスが解除されます。



[Break]ボタン

[Source]ウィンドウは、ブレークポイントに設定されたアドレスを行の先頭のマークで示します。



## bpa (software area break point set)

---

### 機能

プログラムを指定アドレス範囲で停止させるソフトウェアブレイク領域の設定を行います。  
1MBのブレイク有効領域内に設定されているソフトウェアブレイク領域内の命令をフェッチすると、その命令を実行する直前にブレイクが発生します。

### 書式

- ( 1 )>bpa <address1> <address2>↵ ( 直接入力モード )
- ( 2 )>bpa - <address1>↵ ( 直接入力モード )
- <address1>: ブレイク領域開始アドレス 16進数、またはシンボル( IEEE-695形式のみ )
- <address2>: ブレイク領域終了アドレス 16進数、またはシンボル( IEEE-695形式のみ )
- 条件: 0 address1 address2 プログラムメモリ最終アドレス( 0x7ffff )

### 例

書式1)>bpa 100 1ff↵ ...アドレス0x0100 ~ 0x01ffをブレイク領域に設定

書式2)>bpa - 100↵ ...上記ソフトウェアブレイク領域を解除

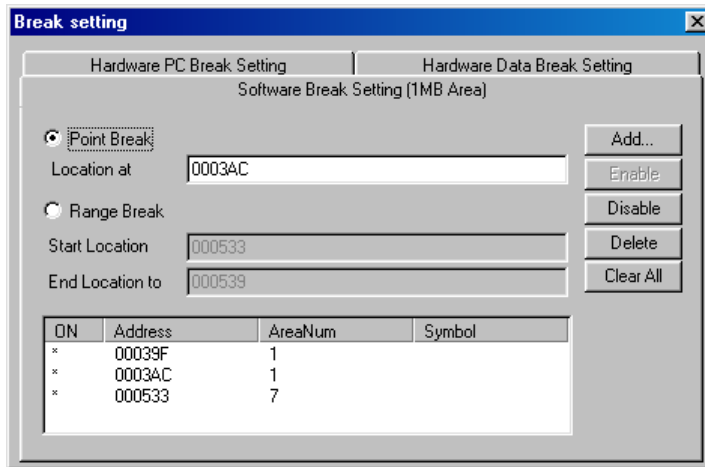
### 注

- デバッグの動作環境として設定されている1MBのブレイク有効領域を外れたアドレスを指定した場合、エラーとなります。ブレイク有効領域は8MBのコード空間を1MBずつ分割した8領域から1つをブレイクオプションとして選択できるようになっています( [Break | Setting...]で選択 )。デバッグの起動時には0x0 ~ 0x0ffffの1MBに設定されます。
- 設定可能なソフトウェアブレイク領域は1ヶ所のみです。新たに設定を行う場合は、前の設定を先にクリアしてください。
- アドレスは、各機種種のプログラムメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- すでに単独のブレイクポイントに設定されているアドレスを含む領域を設定しようとするとワーニングになります。同様に、bpaコマンドで設定したソフトウェアブレイク領域内には、個別のブレイクポイントを設定することはできません。
- ブレイク領域の先頭および終了アドレスには、命令の先頭アドレスを指定してください。命令途中のアドレスを指定すると、それらの命令ではブレイクしません。
- プログラム/パラメータファイルをロードすると、ブレイクの設定内容がすべてクリアされます。

## GUI

[Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。以下の操作は[Software Break Setting (1MB Area)]タブの画面を表示させて行ってください。



ソフトウェアブレーク領域を設定するには、[Range Break]ラジオボタンを選択し[Start Location]テキストボックスに領域開始アドレスを、[End Location to]テキストボックスに領域終了アドレスを入力します。[Add]ボタンをクリックすると、ソフトウェアブレーク領域として登録されます。この領域内は、すべてのアドレスがブレークポイントに設定されたものと見なされます。リストのAddressには先頭アドレスが、AreaNumには領域のバイト数が表示されます。すでにソフトウェアブレーク領域が登録されている状態で新たな領域を設定するとワーニングになります。その場合は先に登録されているソフトウェアブレーク領域を削除してください。また、すでにブレークポイントとして登録されているアドレスを含む領域を重複して設定することはできません。

有効なブレークポイント(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのアドレスを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレークポイントは無効になります。

無効なブレークポイントを有効にするには、ブレークポイントリストからそのアドレスを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレークポイントを解除するには、ブレークポイントリストからそのアドレスを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、ソフトウェアブレーク領域も含め、設定されているすべてのブレークポイントを解除します。

## bpr / bc / bpc (software break point clear)

### 機能

設定されているソフトウェアブレークポイントまたはソフトウェアブレークエリアを解除します。

### 書式

- (1) >bpr↵ (直接入力モード)  
 (2) >bc [<address>]↵ (直接入力モード)  
 (3) >bpc [<address>]↵ (直接入力モード)  
 <address>: ブレークアドレス 16進数、またはシンボル(IEEE-695形式のみ)

### 例

```
>bc 200↵ ... アドレス0x0200のブレークポイントを解除
            アドレス0x0200からブレーク領域が始まっている場合は、ブレーク領域が解
            除されます。

>bpr↵ ... すべてのブレークポイントおよびブレーク領域を解除

>bc↵ ... すべてのブレークポイントおよびブレーク領域を解除

>bpc↵ ... すべてのブレークポイントおよびブレーク領域を解除
```

### 注

- bcコマンドとbpcコマンドは同じ機能を持っています。
- bcコマンドとbpcコマンドでアドレスの指定を省略するとbprコマンドと同機能となり、設定されているブレーク領域とすべてのブレークポイントを解除します。
- ブレークポイントに設定されていないアドレスを指定するとエラーになります。

### GUI

[Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。(bpコマンドの説明を参照してください。)

[Break]ボタン

[Source]ウィンドウ内で、ブレークポイントに設定されているアドレスの行をクリック(カーソルを移動)して[Break]ボタンをクリックすると、そのブレークアドレスが解除されます。逆に、ブレークポイント以外のアドレスの行をクリックして[Break]ボタンをクリックすると、その行がブレークポイントに設定されます。



[Break]ボタン

## bas (sequential break setting)

### 機能

シーケンシャルブレークモードの設定を行います。

### 書式

```
>bas[<mode>].  
( 直接入力モード )  
<mode>: シーケンシャルブレークモード番号  
0 独立ブレークモード  
1 BA3カウントモード  
2 BA2&BA3シーケンシャルモード  
3 BA1-BA3シーケンシャルモード
```

### 例

```
>bas3...BA1-BA3シーケンシャルブレークモードを設定  
  
>bas...<mode>を省略すると、現在の設定内容を表示します。  
Independent Break Mode  
>
```

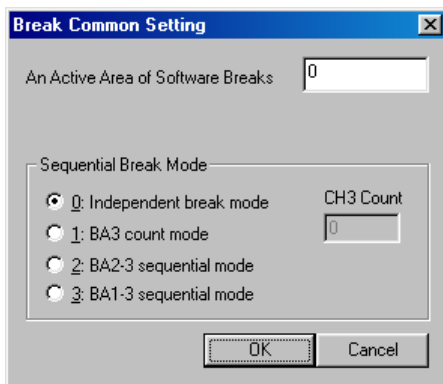
### 注

- "bas"と<mode>の間にはスペースを挿入しないでください。
- 各モードの動作と各チャンネルの設定方法については、baコマンドを参照してください。
- デバッガ起動時は独立ブレークモードに設定されます。
- プログラム/パラメータファイルをロードすると、ブレークの設定内容がすべてクリアされます。

### GUI

[Break | Setting...]メニューコマンド

このメニューコマンドを選択すると、ブレークオプションを選択するためのダイアログボックスが表示されます。



[Sequential Break Mode]のラジオボタンで設定するモードを選択します。

BA3カウンタを使用するモードのラジオボタンを選択すると[CH3 Count]テキストボックスがアクティブになりますので、BA3の実行回数を入力します。

## ba (hardware break point set)

### 機能

プログラムが指定のシーケンスを実行した場合に停止させるハードウェアブレークポイントの設定と解除を行います。各チャンネルに設定するブレークポイントとCH3に設定するカウント数は、basコマンドで設定したシーケンシャルブレークモードに従って有効または無効になります。

シーケンシャルブレークモードによるブレークの発生条件は次のとおりです。

#### (1) 独立ブレークモード (BAS0) (デフォルト)

このモードでは、各チャンネルに設定した個々のブレークポイントの命令をフェッチするとブレークが発生します。CH3 (BA3) のカウント数は無効です。

#### (2) BA3カウントモード (BAS1)

このモードでは、CH3 (BA3) のカウント機能が有効となります。ブレークは、CH3に設定したブレークポイントの命令を、設定したカウントの回数フェッチすると発生します。CH1とCH2に設定したブレークポイントは無効となります。

#### (3) BA2&BA3シーケンシャルモード (BAS2)

このモードでは、CH2に設定したブレークポイントの命令を一度以上実行した後、CH3に設定したブレークポイントの命令を設定したカウントの回数フェッチするとブレークが発生します。CH1に設定したブレークポイントは無効となります。

#### (4) BA1-BA3シーケンシャルモード (BAS3)

このモードでは、CH1、CH2の順に、それぞれに設定したブレークポイントの命令を一度以上実行した後、CH3に設定したブレークポイントの命令を、設定したカウントの回数フェッチするとブレークが発生します。

### 書式

- (1) >ba<channel> <address> [<count>].<sub>↓</sub> (直接入力モード)
- (2) >ba<channel> <condition>.<sub>↓</sub> (直接入力モード)
- <channel>: ブレークチャンネル番号(1~3)
- <address>: ブレークアドレス 16進数、またはシンボル(IEEE-695形式のみ)
- <count>: CH3のカウント数 10進数(デフォルト:1)
- <condition>: 設定の解除、有効化、無効化の指定
- ブレークポイントを解除
  - + ブレークポイントを有効に設定(デフォルト)
  - ブレークポイントを無効に設定
- 条件: 0 address プログラムメモリ最終アドレス(0x7ffff) 0 count 4095

### 例

```
>bas0.↓
>ba1 200.↓
>
```

この例は独立ブレークモードを設定し、CH1のブレークポイントをアドレス0x0200に設定しています。プログラムを実行後、0x0200の命令をフェッチするとブレークが発生します。このブレークポイントは、1MBのブレーク有効領域の外に設定されていても有効です。

```
>ba1 _.↓
>
```

この例はCH1のブレークポイントを無効に設定します。

```
>bas2.↓
>ba2 200.↓
>ba3 300 2.↓
>
```

この例はBA2&BA3シーケンシャルモードを設定し、CH2とCH3のブレークポイントをそれぞれアドレス0x0200、0x0300に設定しています。またCH3のカウントを2に設定しています。プログラムを実行後、0x0200の命令を1回以上実行してから、0x0300の命令を1回実行し、再度0x0300の命令をフェッチすると、その実行の前にブレークが発生します。このブレークポイントは、1MBのブレーク有効領域の外に設定されていても有効です。

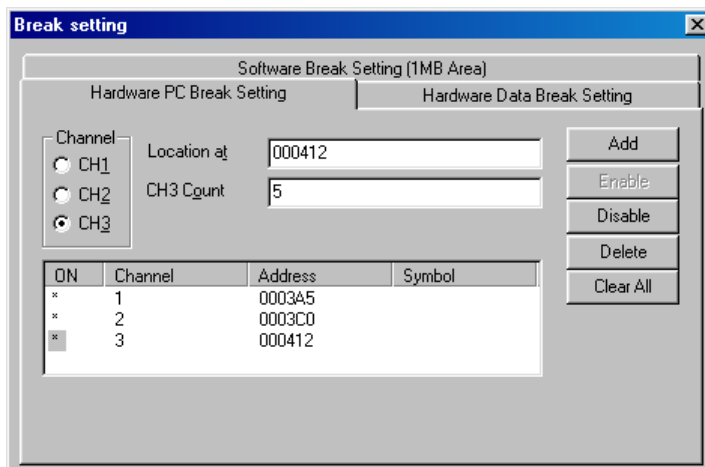
**注**

- "ba"と<channel>の間にはスペースを挿入しないでください。
- CH3の設定でカウント数の指定を省略すると、カウンタは1に設定されます。また0を指定するとカウンタは4096に設定されます。
- 独立ブレークモードの場合もCH3のカウント数を設定することはできますが機能しません。
- アドレスは、各機種のプログラムメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- すでにブレークポイントに設定されているアドレスを再設定しようとするとワーニングになります。
- 設定されていないチャンネルを解除しようとするとエラーになります。
- ブレークポイントには、命令の先頭アドレスを指定してください。途中のアドレスを指定するとブレークは発生しません。
- プログラム/パラメータファイルをロードすると、ブレークの設定内容がすべてクリアされます。

**GUI**

[Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。以下の操作は[Hardware PC Break Setting]タブの画面を表示させて行ってください。



設定するチャンネルをラジオボタンで選択し、[Location at]テキストボックスにアドレスを入力します。BA3のカウント数を指定する場合は、[CH3 Count]テキストボックスに16進数で入力します。[Break Common Setting]ダイアログボックスでカウント数を設定した場合は、その数値がここに反映されます。[Add]ボタンをクリックすると、有効なブレークポイントとして登録されます。設定可能なアドレスは各チャンネルにつき1ヶ所のみで、設定済みチャンネルに新たな設定を行うと上書きされます。また、すでにハードウェアPCブレークポイントに設定してあるアドレスを指定した場合はワーニングになります。有効なブレークポイント(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのアドレスを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレークポイントは無効になります。

無効なブレークポイントを有効にするには、ブレークポイントリストからそのアドレスを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレークポイントを解除するには、ブレークポイントリストからそのアドレスを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、設定されているすべてのブレークポイントを解除します。

## bar (hardware break point clear)

---

### 機 能

設定されているハードウェアブレークポイントおよびCH3のカウント数をすべてクリアします。

### 書 式

>bar,␣ ( 直接入力モード )

### 例

>bar,␣ ...全チャンネルのハードウェアブレークポイントを解除

### 注

ハードウェアブレークポイントが設定されていない場合はエラーになります。

### G U I

[Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。( baコマンドの説明を参照してください。 )



## bd (hardware data break point set)

### 機能

プログラムが指定条件のメモリアクセスを行った場合に停止させるハードウェアデータブレークの設定と解除を行います。

4つのチャンネルそれぞれにデータブレーク条件を設定できます。また、チャンネルごとに有効または無効の設定や解除ができます。

設定可能なデータブレーク条件は次のとおりです。

#### (1) アドレス条件

特定のアドレスをアクセスしたときにブレークさせる場合に指定します。

#### (2) データ条件

特定の1バイトデータをメモリから読み出したとき、あるいはメモリに書き込んだときにブレークさせる場合に指定します。データを10進数以外で指定する場合、任意のビットを"\*"で指定することによって、そのビットをマスクする(データ条件に含めない)ことができます。

#### (3) リード/ライト条件

ブレークをリードサイクルまたはライトサイクルのどちらで発生させるかを指定します。省略した場合は、どちらのサイクルでもブレークが発生します。

これら3つの条件は任意に組み合わせて指定可能です。その場合、設定した条件をすべて満たすメモリアクセスが行われるとブレークが発生します。

### 書式

(1) >bd<channel> [A=<address>] [D=<data>] [(R|W)]\_ (直接入力モード)

(2) >bd<channel> <condition>\_ (直接入力モード)

<channel>: データブレークチャンネル番号(0~3)

<address>: メモリアドレス 16進数、またはシンボル(IEEE-695形式のみ)

<data>: データパターン(1バイト)

10進数以外の指定では"\*"によるビットマスクが可能

R|W R リードサイクルでブレーク

W ライトサイクルでブレーク

省略時はリード/ライト両サイクルでブレーク

<condition>: 設定の解除、有効化、無効化の指定

- ブレーク条件を解除

+ ブレーク条件を有効に設定(デフォルト)

\_ ブレーク条件を無効に設定

条件: 0 address 0xffffffff, 0 data 0xff

### 例

```
>bd0 A=f100 D=1*****B R_
```

```
>
```

この例はデータブレークCH0を設定しています。プログラムを実行後、アドレス0xf100からMSBが1のデータを読み出すとブレークが発生します。このアドレスは、1MBのブレーク有効領域の外に設定されていても有効です。

```
>bd0 _
```

```
>
```

この例はCH0のブレーク条件を無効に設定します。

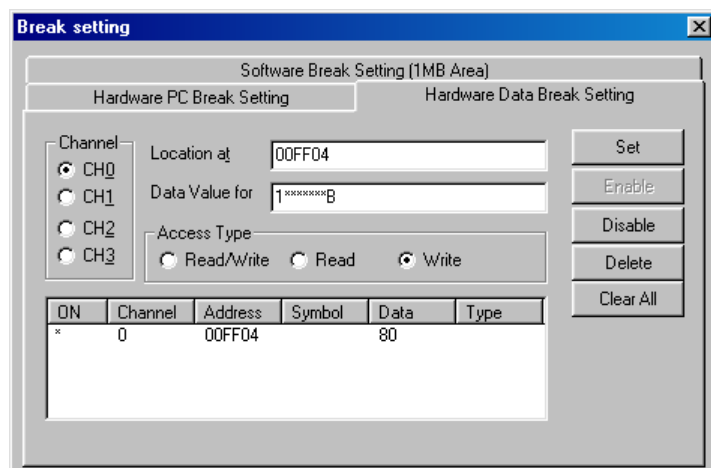
**注**

- "bd"と<channel>の間にはスペースを挿入しないでください。
- アドレスは、各機種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 設定されていないチャンネルを解除しようとするエラーになります。
- プログラム/パラメータファイルをロードすると、ブレークの設定内容がすべてクリアされます。

**GUI**

[Break | Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレークポイントを設定/解除するためのダイアログボックスが表示されます。以下の操作は[Hardware Data Break Setting]タブの画面を表示させて行ってください。



設定するチャンネルをラジオボタンで選択し、[Location at]テキストボックスにアドレスを、[Data Value for]テキストボックスにデータを入力します(未指定も可能)。リード/ライト条件をラジオボタンで選択し、[Set]ボタンをクリックすると、有効なブレーク条件として登録されます。設定済みチャンネルに新たな設定を行うと上書きされます。

有効なブレーク条件(リスト内で先頭に"\*"が付いているアドレス)を無効にするには、リストからそのチャンネルを選択(ON部分をクリック)し、[Disable]ボタンをクリックします。"\*"が消えてブレーク条件は無効になります。

無効なブレーク条件を有効にするには、リストからそのチャンネルを選択し、[Enable]ボタンをクリックします。"\*"が付いてブレークポイントは有効になります。

ブレーク条件を解除するには、リストからそのチャンネルを選択し、[Delete]ボタンをクリックします。

[Clear All]ボタンは、設定されているすべてのブレーク条件を解除します。

## bdr (hardware data break point clear)

---

### 機 能

設定されているハードウェアデータブレイク条件をすべて解除します。

### 書 式

>bdr↵ (直接入力モード)

### 例

>bdr↵ ...全チャンネルのハードウェアデータブレイク条件を解除

### 注

ハードウェアデータブレイク条件が設定されていない場合はエラーになります。

### G U I

[Break ! Breakpoint Setting]メニューコマンド

このメニューコマンドを選択すると、ブレイクポイントを設定/解除するためのダイアログボックスが表示されます。( bdコマンドの説明を参照してください。 )

## bl (break point list)

---

### 機能

すべてのブレーク条件を表示します。

### 書式

>bl␣ (直接入力モード)

### 例

```
>bl␣
PC break:
Software Break:
    1: 0005fa ENABLE
    2: 000618 ENABLE
    3: 00062d ENABLE
Area Break:
    000100 - 0001ff ENABLE
Hardware Break:
    1: CH1 000728 ENABLE
    2: CH2 000742 ENABLE
    3: CH3 000786 ENABLE
Sequential Break Mode:
    BA1 - BA3 Sequential Mode : Count(3)
Data break:
    CH0 DATA: 1***** R/W: R R/W AREA: 00F010 ENABLE
>
```

### GUI

[Break | Break List]メニューコマンド

このメニューコマンドを選択すると、blコマンドが実行されます。

## **bac (break all clear)**

---

### **機 能**

bp、bpa、bas、ba、bdコマンドで設定したブレーク条件をすべて解除します。

### **書 式**

>bac↵ (直接入力モード)

### **G U I**

[Break | Break All Clear]メニューコマンド、[Break All Clear]ボタン

このメニューコマンドまたはボタンを選択すると、bacコマンドが実行されます。



[Break All Clear]ボタン

## 13.9.8 プログラム表示コマンド

### u (unassemble)

#### 機能

プログラムを逆アセンブルして[Source]ウィンドウに表示します。表示内容は次のとおりです。

- 物理メモリアドレス
- 論理メモリアドレス
- オブジェクトコード
- プログラムの逆アセンブル内容

#### 書式

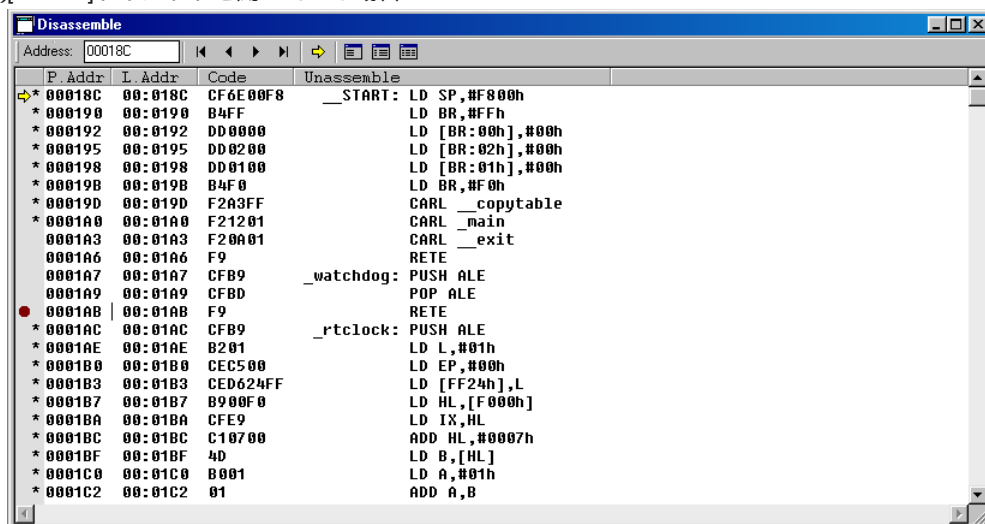
>u [<address>]␣ (直接入力モード)

<address>: 表示開始アドレス 16進数、またはシンボル(IEEE-695形式のみ)

条件: 0 address プログラムメモリ最終アドレス(0x7ffff)

#### 表示

(1) [Source]ウィンドウを開いている場合



<address>を省略した場合、[Source]ウィンドウの表示を逆アセンブル形式に変更します。<address>を指定すると、[Source]ウィンドウの表示を逆アセンブル形式に変更するとともに、<address>からコードの表示を行います。

(2) [Source]ウィンドウを開じている場合

[Command]ウィンドウに16命令分の逆アセンブル結果を表示して、コマンド入力待ちとなります。

<address>を省略した場合は現在のPCから、<address>を指定すると<address>から表示します。

>u␣

```

P. ADDR  L. ADDR  CODE          UNASSEMBLE
0002AE   00:02AE  CF6E00F8  __START: LD SP, #F800h
0002B2   00:02B2  B4FF          LD BR, #FFh
0002B4   00:02B4  DD0000        LD [BR:00h], #00h
0002B7   00:02B7  DD020C        LD [BR:02h], #0Ch
:         :         :
0002CF   00:02CF  B200          LD L, #00h
0002D1   00:02D1  C30000        ADD IY, #0000h
>
```

## (3) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Command]ウィンドウにコードを表示し、その内容をログファイルにも出力します。

[Source]ウィンドウが閉じている場合の表示は上記2)と同様です。

[Source]ウィンドウが開いていれば、その再表示も行います。この場合、[Command]ウィンドウに表示される行数は[Source]ウィンドウの表示行数と同じになります。

## (4) 連続表示機能

uコマンドをキーボードより入力して実行すると、他のコマンドを実行するまでは[Enter]キーの入力のみでコードを連続して表示することができます。

[Enter]キーを入力すると、[Source]ウィンドウは1画面分スクロールします。

[Command]ウィンドウにコードを表示している場合は、前回表示したアドレスに続く16行分(ログ出力中は[Source]ウィンドウと同じ行数)を表示します。

**注**

表示開始アドレスは、各機種のプログラムメモリ領域の範囲内で指定してください。

メモリの有効範囲を越えた場合、エラーとなります。

アドレスが16進数または有効なシンボル以外の場合もエラーとなります。

**GUI**

[View | Source | Disassemble]メニューコマンド、[Disassemble]ボタン

このメニューコマンドまたはボタンを選択すると、[Source]ウィンドウがアクティブとなり、プログラムを現在のPCアドレスから表示します。



[Disassemble]ボタン

## sc (source code)

### 機能

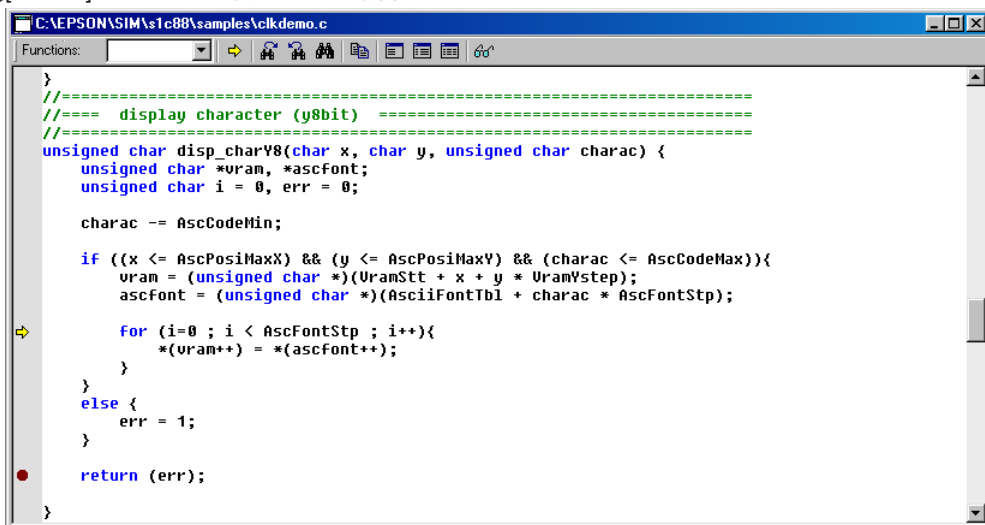
プログラムのソースファイルの内容を[Source]ウィンドウに表示します。

### 書式

>sc [<address>]␣ (直接入力モード)  
 <address>: 表示開始アドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 条件: 0 address プログラムメモリ最終アドレス(0x7ffff)

### 表示

(1) [Source]ウィンドウを開いている場合



<address>を省略した場合、[Source]ウィンドウの表示をソース形式に変更します。<address>を指定すると、[Source]ウィンドウの表示をソース形式に変更するとともに、<address>からコードの表示を行います。

(2) [Source]ウィンドウを閉じている場合

[Command]ウィンドウに17行分のソースを表示して、コマンド入力待ちとなります。

<address>を省略した場合は現在のPCから、<address>を指定すると<address>から表示します。

```

>sc␣
{
    #pragma asm

    GLOBAL    __START
    __START:

    ;=====
    ;===== system initialization =====
    ;=====

    LD    SP, #@DOFF(__lc_es)                ; stack pointer initialize
    LD    BR, #0FFh                          ; BR register initialize to I/O area
    ;----- bus mode setting -----
                                         ; MCU & MPU mode

    LD    [BR:00h], #0

                                         ; Single Chip mode
                                         ; /CE0,/CE2,/CE3,/CE1:disenabled
  
```



## (3) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Command]ウィンドウにコードを表示し、その内容をログファイルにも出力します。

[Source]ウィンドウが閉じている場合の表示は上記(2)と同様です。

[Source]ウィンドウが開いていれば、その再表示も行います。この場合、[Command]ウィンドウに表示される行数は[Source]ウィンドウの表示行数と同じになります。

## (4) 連続表示機能

scコマンドをキーボードより入力して実行すると、他のコマンドを実行するまでは[Enter]キーの入力のみでコードを連続して表示することができます。

[Enter]キーを入力すると、[Source]ウィンドウは1画面分スクロールします。

[Command]ウィンドウにコードを表示している場合は、前回表示したアドレスに続く17行分(ログ出力中は[Source]ウィンドウと同じ行数)を表示します。

**注**

- ソースは、ソースデバugg情報を含むアブソリュートオブジェクトファイルをロードした場合にのみ表示可能です。
- 表示開始アドレスは、各機種のプロプログラムメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。

**GUI**

[View | Source | Source]メニューコマンド、[Source]ボタン

このメニューコマンドまたはボタンを選択すると、[Source]ウィンドウがアクティブとなり、プログラムを現在のPCアドレスから表示します。



[Source]ボタン

## m (mix)

### 機能

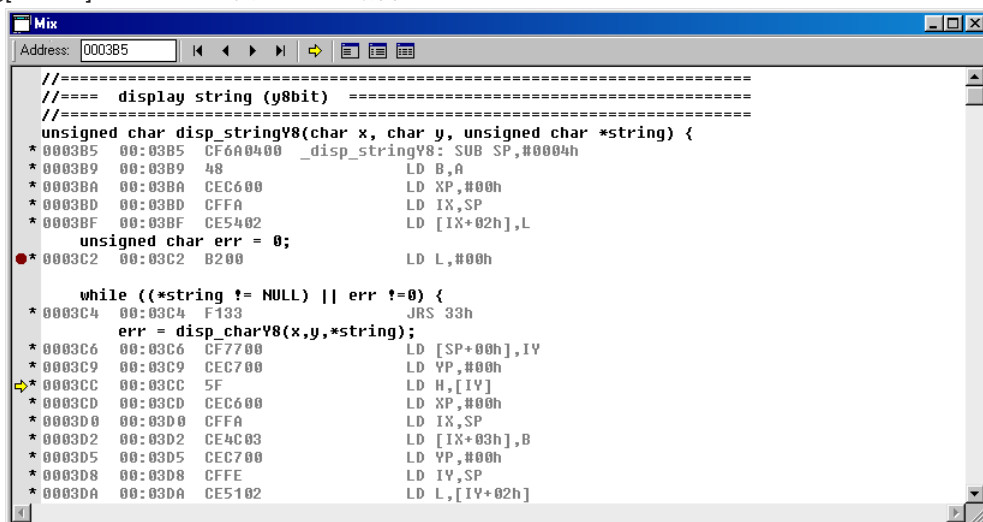
プログラムソースの各行とそれに対応するコードの逆アセンブル結果を[Source]ウィンドウに表示します。逆アセンブル表示の内容は逆アセンブル表示モードと同様です。

### 書式

>m [<address>]↵ (直接入力モード)  
 <address>: 表示開始アドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 条件: 0 address プログラムメモリ最終アドレス(0x7fffff)

### 表示

(1) [Source]ウィンドウを開いている場合



<address>を省略した場合、[Source]ウィンドウの表示をミックス形式(ソース&逆アセンブル)に変更します。<address>を指定すると、[Source]ウィンドウの表示をミックス形式に変更するとともに、<address>からコードの表示を行います。

(2) [Source]ウィンドウを閉じている場合

[Command]ウィンドウに16行分のミックス表示を行い、コマンド入力待ちとなります。

<address>を省略した場合は現在のPCから、<address>を指定すると<address>から表示します。

```
>m↵
_interrupt( 0x0000 )    /* Startup vector */
void _start_cpt( void )
{
0002AE  00:02AE  CF6E00F8    __START: LD SP, #F800h
0002B2  00:02B2  B4FF          LD BR, #FFh
0002B4  00:02B4  DD0000          LD [BR:00h], #00h
0002B7  00:02B7  DD020C          LD [BR:02h], #0Ch
0002BA  00:02BA  DD0100          LD [BR:01h], #00h
0002BD  00:02BD  B4F0          LD BR, #F0h
      :      :      :      :
      :      :      :      :
```

## (3) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Command]ウィンドウにコードを表示し、その内容をログファイルにも出力します。

[Source]ウィンドウが閉じている場合の表示は上記(2)と同様です。

[Source]ウィンドウが開いていれば、その再表示も行います。この場合、[Command]ウィンドウに表示される行数は[Source]ウィンドウの表示行数と同じになります。

## (4) 連続表示機能

mコマンドをキーボードより入力して実行すると、他のコマンドを実行するまでは[Enter]キーの入力のみでコードを連続して表示することができます。

[Enter]キーを入力すると、[Source]ウィンドウは1画面分スクロールします。

[Command]ウィンドウにコードを表示している場合は、前回表示したアドレスに続く16行分(ログ出力中は[Source]ウィンドウと同じ行数)を表示します。

**注**

- ソースは、ソースデバッグ情報を含むアブソリュートオブジェクトファイルをロードした場合にのみ表示可能です。
- 表示開始アドレスは、各機種のプログラムメモリ領域の範囲内で指定してください。メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。

**GUI**

[View ! Source ! Mix]メニューコマンド、[Mix]ボタン

このメニューコマンドまたはボタンを選択すると、[Source]ウィンドウがアクティブとなり、プログラムを現在のPCアドレスから表示します。



[Mix]ボタン

## 13.9.9 シンボル情報表示コマンド

**sy** (symbol list)**機 能**

定義されているすべてのシンボルのリストを[Command]ウィンドウに表示します。

**書 式**

>sy [/a]↵ (直接入力モード)

**例**

```
>sy↵
Address Symbol
0004A5  __ANDXL
0004E4  __BLCPS
0004C6  __CMPSL
00056B  __CMPUL
0002CE  __DIVSI
      :
000E48  __strtok
0002C9  __watchdog
```

>  
/aを省略すると、定義されているシンボルをアルファベット順に表示します。

```
>sy /a↵
Address Symbol
000100  __copytable
00014A  __rtclock
0002AE  __START
0002AE  __start_cpt
0002C9  __watchdog
      :
00F1F2  __ungetc
00F800  __lc_es
```

>  
/aを指定すると、アドレス順に表示します。

**注**

シンボルリストの表示は、IEEE-695形式のオブジェクトファイル(.abs)を読み込んでいる場合、またはプログラムHEXファイル(.psa)ロード時にシンボルファイル(.sy)が読み込まれている場合にのみ可能です。

**GUI**

なし

## w (symbol watch)

### 機能

シンボルの内容を表示します。

### 書式

(1) >w <symbol> [;<option>] [/a] (直接入力モード)

(2) >w (ガイダンスモード)

```
File name: <file name>
Function name: <function>
Symbol name: <symbol>
Format ? (B/Q/D/H) <option>
Display in watch window? (Y/N) {Y|N}
<symbol> = 現在の値
>
  <symbol>: シンボル名
  <option>: 表示形式オプション
             B   2進数
             Q   8進数
             D  10進数
             H  16進数 (デフォルト)
  <file name>: ソースファイル名
  <function>: 関数名
```

### 例

```
書式1) >w saveFlg ;B
saveFlg = 00000001 ... シンボルの値を表示
>w saveFlg ;B /a ... シンボルの値を[Watch]ウィンドウに表示
>w xxx
No such symbol exists. ... シンボルが見つからない場合
>
```

/aオプションを指定すると、[Watch]ウィンドウにシンボル名と値が表示されるとともにウォッチシンボルリストに登録され、[Watch]ウィンドウの更新モードに従って表示が自動的に更新されます。

```
書式2) >w
File name: calc.c
Function name: main
Symbol name: count
Format? (B/Q/D/H)H
Display in watch window? (Y/N)N
count = 0x00
>
```

グローバルシンボルを指定する場合、ファイル名と関数名は[Enter]キーのみを入力してスキップします。

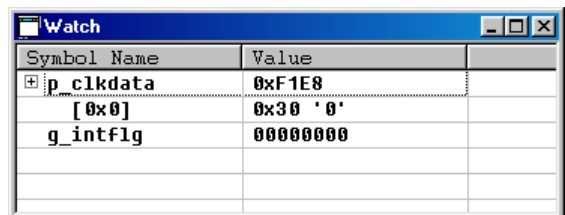
### 注

wコマンドによるシンボル情報の表示は、IEEE-695形式のオブジェクトファイル(.abs)を読み込んでいる場合にのみ可能です。

### GUI

[Watch]ボタン([Source]ウィンドウ上)  
[Source]ウィンドウ内のシンボル名をドラッグして選択(反転表示)し[Watch]ボタンをクリックすると、そのシンボルが[Watch]ウィンドウのシンボルリストに登録されます。その後は、[Watch]ウィンドウでそのシンボルの値を確認することができます。

 [Watch]ボタン



Symbol Name	Value
p_clkdata	0xF1E8
[0x0]	0x30 '0'
g_intflg	00000000

### 13.9.10 ファイル読み込みコマンド

## If (load file)

### 機能

プログラムファイル(.abs: IEEE-695形式、.psa: モトローラS2形式)およびファンクションオプションHEXファイル(.fsa: モトローラS2形式)を読み込みます。

### 書式

- (1) >If <file name>↓ (直接入力モード)  
 (2) >If↓ (ガイダンスモード)  
 Program object file name (.ABS/.PSA) ... ? <file name>↓  
 Function option file name (.FSA) ... ? <file name>↓  
 OK!  
 >  
 <file name>: 読み込みファイル名(パスも指定可能)

### 例

書式1) >lf test.abs↓  
 OK!  
 Symbol file is loaded. ... シンボル情報が読み込まれたことを示します。  
 >lf test.fsa↓  
 OK!  
 >

書式1ではオブジェクトファイルとファンクションオプションファイルを個別に指定する必要があります。

書式2) >lf↓  
 Program object file name(.ABS/.PSA) ... ? test.abs↓  
 Function option file name(.FSA) ... ? test.fsa↓  
 OK!  
 Symbol file is loaded.  
 >

書式2ではガイダンスに従ってオブジェクトファイルとファンクションオプションファイル名を入力し、2つのファイルを1回で読み込むことができます。[Enter]のみで一方のファイルの読み込みをスキップすることができます。

### 注

- デバッグは指定されたファイル名でファイルの種類を判断します。このため、上記の拡張子以外のファイルは読み込めません。指定した場合はエラーとなります。
- デバッグにソースの表示やシンボルを使用する場合、デバッグ情報を含んだIEEE-695形式のオブジェクトファイルを読み込む必要があります。
- ファイルの読み込み時に[Source]ウィンドウが開いていれば、その内容を更新します。プログラムの表示は現在のPCアドレスから行われます。
- ファイル読み込み中にエラーが発生した場合、すでに読み込まれた部分はエミュレーションメモリに残ります。
- プログラムファイルを読み込むと、設定済みのブレークポイント/条件がすべて解除され、取得されているトレース情報とカバレッジ情報はクリアされます。

### GUI

[File | Load File...]メニューコマンド、[Load File]ボタン

このメニューコマンドまたはボタンを選択すると、読み込むオブジェクトファイルを指定するダイアログボックスが表示されます。



[Load File]ボタン

## par (load parameter file)

---

### 機能

パラメータファイル(.par)を読み込み、メモリマップ情報を設定します。  
自己書き換え用プログラムアドレスの設定が必要な場合は、そのプログラムの最終アドレスにブレークを設定します。

### 書式

- (1) >par <file name>␣ (直接入力モード)  
(2) >par␣ (ガイドンスモード)  
File Name ...? <file name>␣  
>  
<file name>: パラメータファイル名(パスも指定可能)

### 例

書式1) >par 88xxx.par␣  
>

書式2) >par␣  
File name ? 88xxx.par␣  
>

### 注

- パラメータファイルを読み込むと、設定済みのブレークポイント/条件がすべて解除され、取得されているトレース情報とカバレッジ情報はクリアされます。
- 読み込んだパラメータファイルのマップ情報に異常があった場合、初期化処理に失敗してプログラムを実行できません。

### GUI

[File | Load Parameter File]メニューコマンド、[Load Parameter]ボタン

このメニューコマンドまたはボタンを選択すると、読み込むパラメータファイルを指定するダイアログボックスが表示されます。



[Load Parameter]ボタン

## 13.9.11 トレースコマンド

**td** (trace data display)**機 能**

ICEのトレースメモリに採取したトレース情報を表示します。

**書 式**

(1) >td [cycle]< ( 直接入力モード )

(2) >td< ( ガイダンスモード )

Start index (ENTER as 0)? : <cycle><

(トレース情報が表示されます)

>

<cycle>: トレースサイクル番号 10進数

条件: 0 cycle 8191

**表 示**

トレース情報の表示内容は次のとおりです。

INS: CPUサイクル数( 10進数 )

P. Addr: 物理アドレス( 16進数 )

L. Addr: 論理アドレス( 16進数 )

Code: オブジェクトコード( 16進表示 )

Mnemonic: 逆アセンブル内容

BA ~ YP: サイクル実行後の各レジスタ値( 16進数 ) 初期状態はxxxx

SC, CC: コンディションフラグの状態

Memory: メモリのアクセス内容( コードフェッチを除く )

MR: メモリリード

MW: メモリライト

[<address>] = <data>: アクセスしたアドレスとリード/ライトしたデータ( 16進数 )

(1) [Trace]ウィンドウが開いている場合

INS	P. Addr	L. Addr	Code	Mnemonic	BA	HL	IX	IY	SP	BR	EP	XP	YP	SC	CC	Memory
0217	000499	01:0499	93	INC IX	3E84	F828	F828	F060	F7F3	F0	00	00	00	00	N-C	0000
0218	00049A	01:049A	92	INC IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	N-C	0000
0219	00049B	01:049B	CF7601	LD [SP+01h], IX	3E84	F828	F829	F060	F7F3	F0	00	00	00	00	N-C	0000 MW:[00F7F4]=29 MW:[00F7
0220	00049E	01:049E	8001	LD A, #01h	3E01	F828	F829	F060	F7F3	F0	00	00	00	00	N-C	0000
0221	0004A0	01:04A0	A6	PUSH IP	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	N-C	0000 MW:[00F7F2]=00 MW:[00F7
0222	0004A1	01:04A1	CEC600	LD XP, #00h	3E01	F828	F829	F060	F7F1	F0	00	00	00	00	N-C	0000
0223	0004A4	01:04A4	CFFA	LD B, [IX+02h]	3E01	F828	F7F1	F060	F7F1	F0	00	00	00	00	N-C	0000
0224	0004A6	01:04A6	CE4802	LD B, [IX+02h]	0401	F828	F7F1	F060	F7F1	F0	00	00	00	00	N-C	0000 MR:[00F7F3]=04
0225	0004A9	01:04A9	AE	POP IP	0401	F828	F7F1	F060	F7F3	F0	00	00	00	00	N-C	0000 MR:[00F7F1]=00 MR:[00F7
0226	0004AA	01:04AA	01	ADD A, B	0405	F828	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0227	0004AB	01:04AB	50	LD L, A	0405	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0228	0004AC	01:04AC	B105	LD B, #05h	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0229	0004AE	01:04AE	42	LD A, L	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000
0230	0004AF	01:04AF	A6	PUSH IP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F2]=00 MW:[00F7
0231	0004B0	01:04B0	CEC600	LD XP, #00h	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0232	0004B3	01:04B3	CFFA	LD IX, SP	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000
0233	0004B5	01:04B5	CE4402	LD [IX+02h], A	0505	F805	F7F1	F060	F7F1	F0	00	00	00	00	-----	0000 MW:[00F7F3]=05
0234	0004B8	01:04B8	8E	POP IP	0505	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----	0000 MR:[00F7F1]=00 MR:[00F7
0235	0004B9	01:04B9	3080	XOR A, #80h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	N----	0000
0236	0004BB	01:04BB	CEB880	XOR B, #80h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	N----	0000
0237	0004BE	01:04BE	31	CP A, B	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0238	0004BF	01:04BF	CEC0CB	JRS LT, CBh	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000
0239	0004C2	01:04C2	F105	JRS 05h	0585	F805	F7F1	F060	F7F3	F0	00	00	00	00	-----Z	0000

<cycle>を省略してtdコマンドを実行すると、[Trace]ウィンドウは最新のデータを再表示します。

<cycle>を指定してtdコマンドを実行すると、指定したサイクルから表示されます。

[Trace]ウィンドウの内容は、ターゲットプログラムの実行後に更新されます。

スクロールによってすべてのトレースデータを表示させることができます。



## (2) [Trace]ウィンドウが開いている場合

<cycle>を省略してtdコマンドを実行すると、11行分の最新データを[Command]ウィンドウに表示します。  
 <cycle>を指定してtdコマンドを実行すると、指定したトレース番号から11行分のデータを[Command]ウィンドウに表示します。

```
>td
Start index (ENTER as 0)? :
Ins. P.Addr L.Addr Code Mnemonic BA HL IX IY SP BR EP XP YP SC CC Memory
0000 000179 00:0179 CF7000 LD BA,[SP+00h] xxxx xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0001 00017A 00:017A xxxx xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0002 00017B 00:017B xxxx xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0003 0077FC 00:F7FC xx00 xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0 MR:[00F7FC]=00
0004 00017C 00:017C xx00 xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0005 0077FD 00:F7FD 0100 xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0 MR:[00F7FD]=01
0006 00017C 00:017C 98 DEC BA 0100 xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0007 00017D 00:017D 0100 xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0008 00017D 00:017D CF7400 LD [SP+00h],BA 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0009 00017E 00:017E 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0010 00017F 00:017F 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
>td 11
Ins. P.Addr L.Addr Code Mnemonic BA HL IX IY SP BR EP XP YP SC CC Memory
0011 0077FC 00:F7FC 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0 MW:[00F7FC]=FF
0012 000180 00:0180 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0013 0077FD 00:F7FD 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0 MW:[00F7FD]=00
0014 000180 00:0180 E7EB JRS NZ,EBh 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0015 000181 00:0181 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0016 00016C 00:016C CE3501 CP [HL],#01h 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0017 00016D 00:016D 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0018 00016E 00:016E 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0
0019 000C53 00:0C53 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11----- 00C0 MR:[000C53]=01
0020 00016F 00:016F E706 JRS NZ,06h 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11-----Z 00C0
0021 000170 00:0170 00FF xxxx xxxx F0E4 xxxx xx xx xx xx 11-----Z 00C0
>
```

## (3) ログ出力中

logコマンドの指定によってコマンド実行結果をログファイルに出力している場合は、[Command]ウィンドウにトレースデータを表示し、その内容をログファイルにも出力します。

[Trace]ウィンドウが開いている場合の表示は上記(2)と同様です。

[Trace]ウィンドウが開いていれば、その再表示も行います。この場合、[Command]ウィンドウに表示される行数は[Trace]ウィンドウの表示行数と同じになります。

## (4) 連続表示機能

tdコマンドを実行すると、他のコマンドを実行するまでは[Enter]キーの入力のみでトレースデータを連続して表示することができます。

[Enter]キーを入力すると、[Trace]ウィンドウは1画面分前にスクロールします。

[Command]ウィンドウに表示している場合は、前回表示したサイクルの前の11行分(ログ出力中は[Trace]ウィンドウと同じ行数)を表示します。

表示方向は、[Enter]キーの入力ごとに古い実行サイクル(FORWARD)に向かいますが、[B]キーによってこの方向を逆(BACKWORD)にすることができます。表示方向を戻すには[F]キーを入力します。[Trace]ウィンドウが開いている場合は、そのスクロール方向も変更されます。

```
>td 100
...FORWORDで表示開始
( サイクルNo.100 ~ 110のデータを表示 )
>b
...BACKWORDに変更
( サイクルNo. 99 ~ 89のデータを表示 )
>
...BACKWORDで表示を継続
( サイクルNo.88 ~ 78のデータを表示 )
>f
...FORWORDに変更
( サイクルNo. 99 ~ 89のデータを表示 )
>
```

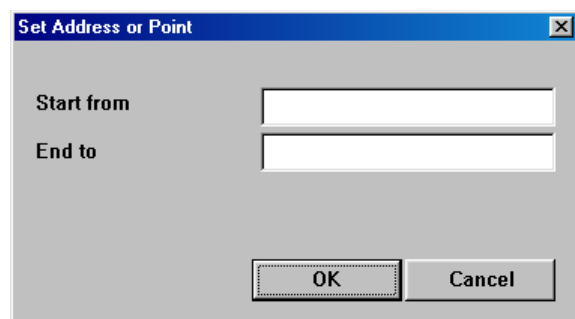
**注**

- サイクルNo.は0～0x1fff( 8,191 )の領域内で指定してください。これを越えた場合、エラーとなります。
- トレースメモリはプログラムの実行がブレイクするまで新しいデータを取り込みます。トレース情報がトレースメモリの容量を越えた場合は、古いデータから上書きされます。

**GUI****[Trace | Trace]メニューコマンド**

このメニューコマンドを選択すると、[Trace]ウィンドウが開き、最新のトレース情報を表示します。

このとき、次のダイアログが表示され、表示させるサイクルNo. を指定できます。



[Start from]テキストボックスに表示開始サイクル番号を、[End to]テキストボックスに表示終了サイクル番号を16進数で入力し、[OK]をクリックします。これらの入力は省略可能で、[Start from]を省略した場合はサイクル番号0から表示されます。

**[Trace | Setting...]メニューコマンド**

このメニューコマンドを選択すると、[Trace Information Setting]ダイアログボックスが開き、トレース条件が設定できます。詳細については、"13.8.6 トレース機能"を参照してください。

## ts (trace search)

### 機能

トレースメモリの中から指定した条件でトレース情報を検索します。  
検索条件を次の3種類から選択することができます。

1. 実行したアドレスによる検索  
プログラムメモリアドレスを指定して、そのアドレスを実行したサイクルを検索します。
2. 指定メモリの読み出しサイクルを検索  
データメモリアドレスを指定して、そのアドレスから読み出しを行ったサイクルを検索します。
3. 指定メモリへの書き込みサイクルを検索  
データメモリアドレスを指定して、そのアドレスへの書き込みを行ったサイクルを検索します。

### 書式

(1) >ts <option> <address>␣ (直接入力モード)

(2) >ts␣ (ガイダンスモード)

1. pc address 2. data read address 3. data write address ...? <1 | 2 | 3>␣

Search address?: <address>␣

(検索結果を表示)

>

<option>: 検索条件 pc(=実行アドレス) dr(=データ読み出し) dw(=データ書き込み)

<address>: 検索アドレス 16進数、またはシンボル(IEEE-695形式のみ)

条件: 0 address 0x7ffff( pcオプション指定時) 0 address 0xfffff( dr/dwオプション指定時)

### 例

検索結果は、[Trace]ウィンドウが開いていれば[Trace]ウィンドウに表示されます。[Trace]ウィンドウが閉じている場合は、tdコマンドと同様に[Command]ウィンドウに表示されます。

書式1) >ts pc 823␣

Searching trace data ... OK!

Ins.	P.Addr	L.Addr	Code	Mnemonic	BA	HL	IX	IY	...
0006	000823	00:0823			0006	xxxx	xxxx	xxxx	...
0007	000823	00:0823	E7FA	JRS NZ,FAh	0006	xx07	xxxx	xxxx	...

>

書式2) >ts␣

1.pc address 2.data read address 3.data write address ...? 1␣

Searching trace data ... OK!

Ins.	P.Addr	L.Addr	Code	Mnemonic	BA	HL	IX	IY	...
0006	000823	00:0823			0006	xxxx	xxxx	xxxx	...
0007	000823	00:0823	E7FA	JRS NZ,FAh	0006	xx07	xxxx	xxxx	...

>

logコマンドの指定によってコマンド実行結果をログファイルに出力しているときには、[Trace]ウィンドウが開いている場合でも、検索結果が[Command]ウィンドウに表示され、ログファイルに出力されます。

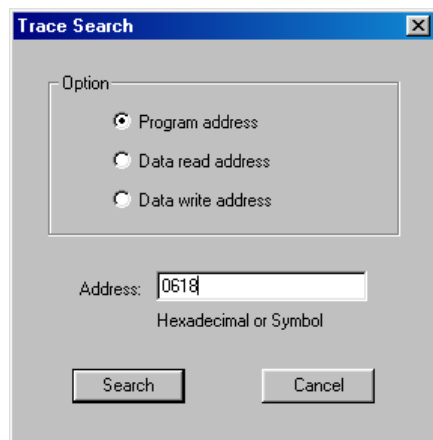
### 注

アドレスは、各機種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。

**GUI**

[Trace ! Trace Search...]メニューコマンド

このメニューコマンドを選択すると、検索条件を設定するダイアログボックスが表示されます。



ラジオボタンでオプションを選択し、アドレスをテキストボックスに入力して[Search]ボタンをクリックします。

## tf (trace file)

### 機能

td、tsコマンドの実行によって現在表示されているトレースデータの中から、指定範囲のデータをファイルに保存します。

### 書式

(1) >tf <file name> [<cycle1> [<cycle2>]]  
(直接入力モード)

(2) >tf.  
(ガイダンスモード)

Start index (min 0)? : <cycle1>.

End index (max 8191)? : <cycle2>.

File Name ? : <file name>.

>

<file name>: 出力ファイル名(パスも指定可能)

<cycle1>: 開始サイクル数 16進数(デフォルト0)

<cycle2>: 終了サイクル数 16進数(デフォルト0x1fff)

条件: 0 cycle1 cycle2 0x1fff

### 例

書式1) >tf trace.trc  
8191-8000  
8000-7000  
:  
1000- 1  
OK!  
>  
...tdコマンドで表示させたすべてのトレース情報を保存

書式2) >tf.  
Start index (min 0) ? :0.  
End index (max 8191) ? :100.  
File name ? :test.trc.  
1000- 1  
OK!  
>

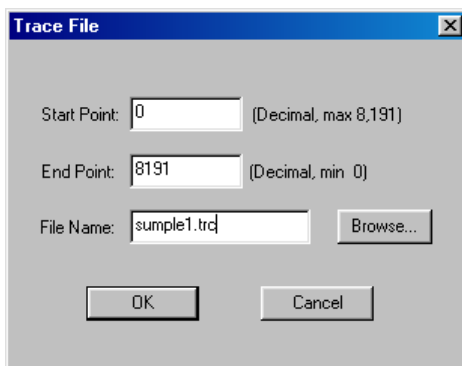
### 注

- 既存のファイルを指定すると、データを上書きします。
- <cycle1>のデフォルト値はサイクル番号0、<cycle2>のデフォルト値は0x1fff(8191)最新トレースデータです。

### GUI

[Trace ! Trace File...]メニューコマンド

このメニューコマンドを選択すると、パラメータを設定するダイアログボックスが表示されます。



開始サイクル数、終了サイクル数、ファイル名を入力し、[OK]ボタンをクリックします。  
すべてのトレース情報を保存するには、[Start Point]と[End Point]は空白のままにしてください。  
ファイル名は、[Browse...]ボタンで表示される標準ファイル選択ダイアログボックスによっても選択可能です。

## 13.9.12 カバレッジコマンド

**cv** (coverage)**機 能**

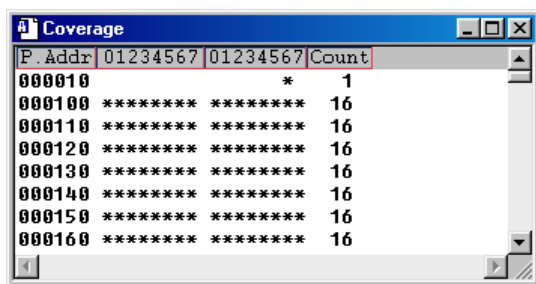
ターゲットプログラム実行中にICEが取得したカバレッジ情報(アクセスしたアドレス)を表示します。

**書 式**

>cv <address1> [<address2>].<sub>↓</sub> (直接入力モード)  
 <address1>: 開始アドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 <address2>: 終了アドレス 16進数、またはシンボル(IEEE-695形式のみ)  
 条件: 0 address1 address2 メモリ最終アドレス(0xffffffff)

**例**

(1) [Coverage]ウィンドウが開いている場合



P.Addr	01234567	01234567	Count
000010	*		1
000100	*****	*****	16
000110	*****	*****	16
000120	*****	*****	16
000130	*****	*****	16
000140	*****	*****	16
000150	*****	*****	16
000160	*****	*****	16

16バイト/行の形式で<address1>からカバレッジ情報を表示します。P.Addrは各行の先頭アドレス(物理アドレス)です。アクセスしたアドレスは"\*"で、未アクセスアドレスは" "(スペース)で示されます。Countの数値は、各行の16バイト中でアクセスしたアドレスの合計(バイト数)です。スクロールによって取得されているすべてのデータを表示させることができます。

(2) [Coverage]ウィンドウが閉じている場合

<address2>を省略してcvコマンドを実行すると、<address1>から最終アドレスまでのカバレッジ情報を[Command]ウィンドウに表示します。

<address2>を指定してcvコマンドを実行すると、<address1>から<address2>までの情報を表示します。

```
>cv 100.↓ ...0x000100以降で実行したアドレスを表示
000100 - 00020e
000233 - 0002c4
0004e4 - 0004e9
      :
00ff40
00ff54 - 00ff55
00ff61
00ff63

>cv 100 1ff.↓ ...0x000100 ~ 0x0001ffの範囲で実行したアドレスを表示
000100 - 0001ff
>
```

**注**

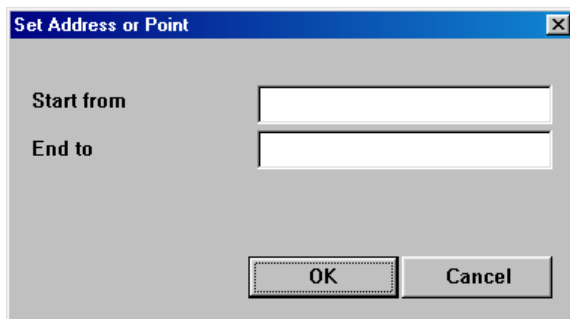
- カバレッジ情報は、デバグガのカバレッジオプションで指定されている取得モード(全アドレス空間またはデータ空間のみから取得)と取得範囲(指定の64KB領域)に従って記録されます。カバレッジオプションは、[Coverage]メニューから[Setting...]を選択すると表示されるダイアログボックスで設定します。詳細は"13.8.7 カバレッジ"を参照してください。
- アドレスは、各機種種のメモリ領域の範囲内で指定してください。  
メモリの有効範囲を越えた場合、エラーとなります。  
アドレスが16進数または有効なシンボル以外の場合もエラーとなります。
- 開始アドレスが終了アドレスより大きい場合、エラーとなります。

**GUI**

[Coverage | Coverage]メニューコマンド

このメニューコマンドを選択すると、[Coverage]ウィンドウが開きます。

このとき、次のダイアログが表示され、表示を開始させるアドレスを指定できます。



[Start from]テキストボックスに表示を開始するアドレスを16進数で入力して[OK]ボタンをクリックします。[Coverage]ウィンドウに表示させる場合、特に[End to]を入力する必要はありません。入力を省略した場合、開始アドレスは設定されている64KB領域の先頭アドレス、終了アドレスは64KB領域の終了アドレスとして処理されます。

## cvc (coverage clear)

---

### 機 能

カバレッジ情報をクリアします。

### 書 式

>cvc,」 ( 直接入力モード )

### G U I

[Coverage ! Coverage Clear]メニューコマンド

このメニューコマンドを選択すると、cvcコマンドが実行されます。



### 13.9.13 コマンドファイル実行コマンド

#### com (execute command file)

##### 機能

コマンドファイルを読み込み、その中に記述されたデバッグコマンドを連続実行します。各コマンドの実行間隔を0～256秒の範囲で1秒間隔に指定可能です。

##### 書式

- (1) >com <file name> [<interval>]↵ (直接入力モード)
- (2) >com↵ (ガイダンスモード)
- File name ? <file name>↵
- Execute commands 1. successively 2. with wait ...? <1 | 2>↵
- Interval (0 - 256 seconds) : <interval>↵ ("2. with wait"選択時にのみ表示)
- >(実行コマンドを表示)
- <file name>: コマンドファイル名(パスも指定可能)
- <interval>: コマンドの実行間隔(ウェイト時間) 10進数(0～256)

##### 例

書式1) >com batch1.cmd↵

>..... "...batch1.com"に記述されたコマンドを連続実行

書式2) >com↵

File name ? test.cmd↵

Execute commands 1. successively 2. with wait ...? 2↵

Wait time (0 - 256 seconds) : 2↵

>..... ...各コマンド実行後に2秒の待ち時間を挿入

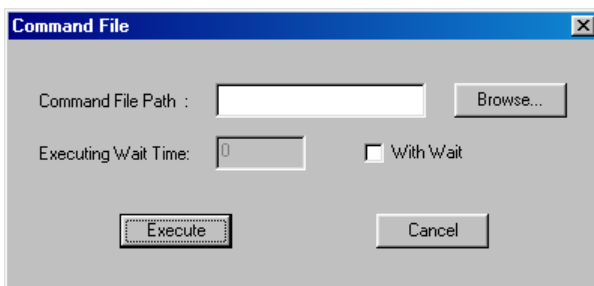
##### 注

- ・コマンドファイルにコマンド以外の内容は記述しないでください。
- ・指定のファイルが見つからない場合はエラーとなります。
- ・コマンドファイル内から別のコマンドファイルを読み込むことも可能です。ただし、最大5階層までに制限されます。6階層目のcom( cmw )コマンドが現れるとエラーとなり、そのcom( cmw )コマンドで指定されるコマンドファイルは実行されません。そのcom( cmw )コマンドをスキップし、それ以後の実行を継続します。
- ・実行間隔に256以上の数値を指定した場合、256秒が指定されたものとして処理されます。
- ・[Ctrl]+[Q]キーの入力により、コマンドファイルの実行を停止することができます。

##### GUI

[Run ! Command File...]メニューコマンド

このメニューコマンドを選択すると、コマンドファイルを選択するダイアログボックスが表示されます。



[Command File Path]にファイル名を入力し、[Execute]ボタンをクリックします。ファイル名は、[Browse...]ボタンで表示される標準ファイル選択ダイアログボックスによっても選択可能です。コマンドの実行間隔を指定する場合は[With Wait]を選択し、[Executing Wait Time]に秒数を入力しておきます。

## cmw (execute command file with wait)

---

### 機能

コマンドファイルを読み込み、その中に記述されたデバッグコマンドを一定時間ごとに実行します。個々のコマンドの実行間隔は、[Option]メニューの[Setting...]を選択することにより表示されるダイアログボックスで1～256秒の範囲(1秒単位)で設定できます。デバッグの初期設定は1秒です。

### 書式

- (1) >cmw <file name>␣ (直接入力モード)
- (2) >cmw␣ (ガイダンスモード)
- File name ? <file name>␣
- >(実行コマンドを表示)
- <file name>: コマンドファイル名(パスも指定可能)

### 例

書式1) >cmw batch1.cmd␣

>.....

書式2) >cmw␣

File name ? test.cmd␣

>.....

### 注

- コマンドファイルにコマンド以外の内容は記述しないでください。
- 指定のファイルが見つからない場合はエラーとなります。
- コマンドファイル内から別のコマンドファイルを読み込むことも可能です。ただし、最大5階層までに制限されます。6階層目のcmw( com )コマンドが現れるとエラーとなり、そのcmw( com )コマンドで指定されるコマンドファイルは実行されません。そのcmw( com )コマンドをスキップし、それ以後の実行を継続します。
- comコマンドにより読み込むコマンドファイル内にcmwコマンドを記述すると、それ以降のファイル内のコマンドはすべて( comコマンドがあった場合でも )指定の時間ごとに実行されます。
- [Ctrl]+[Q]キーの入力により、コマンドファイルの実行を停止することができます。

### GUI

なし

ただし、[Run]メニューの[Command File...]で同様の機能が実行可能です( comコマンド参照 )。

## rec (record commands to file)

### 機能

実行したデバグコマンドを指定のコマンドファイルに保存します。

### 書式

- (1) >rec <file name>␣ (直接入力モード)  
 (2) >rec␣ (ガイダンスモード) ...ガイダンスについては例を参照  
 <file name>: コマンドファイル名(パスも指定可能)

### 例

- (1) デバグ起動後、最初のrecコマンド実行時

```
>rec␣
File name    ? sample.cmd␣
1. append    2. clear and open    ...? 2␣    ...既存のファイルを指定した場合に表示
>
```

- (2) 2回目以降のrecコマンド実行時

```
>rec␣
Set to record off mode.    ...recコマンド実行ごとにレコードON/OFFを切り換え
.....
>rec␣
Set to record on mode.
```

### 注

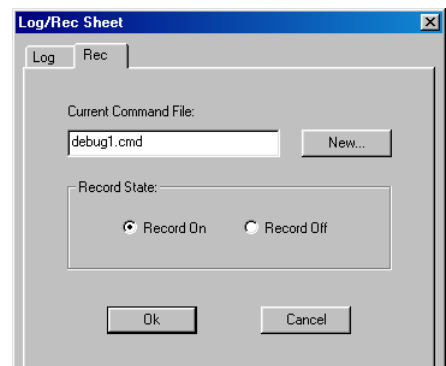
- レコード機能がONの場合、[Command]ウィンドウに直接入力したコマンド以外にも、メニューやツールバーボタンで選択したコマンド([Help]メニューコマンド/ボタンを除く)が[Command]ウィンドウに表示され、指定のファイルに出力されます。  
[Register]ウィンドウ、[Dump]ウィンドウで直接レジスタ値やデータメモリの内容を修正した場合、あるいは[Source]ウィンドウ内でのダブルクリックによってブレークポイントを設定した場合も、対応するコマンドが[Command]ウィンドウに表示され、ファイルに出力されます。
- 最初にrecコマンドを実行する場合は、それに続く実行コマンドを記録するためのファイル名を指定する必要があります。
- 一度コマンドファイルが開かれると、それ以降はrecコマンドの実行ごとに記録を中断、再開(トグル)します。この切り換えは、デバグを終了するまで有効です。コマンドの記録を別のファイルに変更するには、書式1を使用してファイルを指定し直してください。その時点でそれまでのコマンドファイルは閉じられ、以降の記録は新しく指定したファイルに対して行われます。
- 頻繁に使用する一連のコマンドをrecコマンドでコマンドファイルに記録することにより、2回目の実行からはcom、cmwコマンドが使用できます。

### GUI

[Option | Record...]メニューコマンド

このメニューコマンドを選択すると、コマンドファイルを指定するダイアログボックスが表示されます。新しいコマンドファイルを指定するには、[Current Command File]にコマンドファイル名を入力するか、[New...]ボタンをクリックして選択してください。

すでにコマンドの記録を開始している場合は、[Record State]のラジオボタンでレコードON/OFFを切り換えることができます。



## 13.9.14 ログコマンド

## log (log)

## 機能

入力したコマンドと実行結果をファイルに保存します。

## 書式

(1) >log <file name>↓ (直接入力モード)

(2) >log↓ (ガイダンスモード) ...ガイダンスについては例を参照  
<file name>: ログファイル名(パスも指定可能)

## 例

(1) デバッガ起動後、最初のlogコマンド実行時

```
>log↓
File name    ? debug1.log↓
1. append    2. clear and open    ...? 2↓    ...既存のファイルを指定した場合に表示
>
```

(2) 2回目以降のlogコマンド実行時

```
>log↓
Set to log off mode.    ...logコマンド実行ごとにログ出力ON/OFFを切り換え
.....
>log↓
Set to log on mode.
```

## 注

- ログファイルには[Command]ウィンドウに表示された内容がそのままファイルに書き込まれます。ログ出力中にはメニューやツールバーボタンで選択したコマンド([Help]メニューコマンド/ボタンを除く)も[Command]ウィンドウに表示され、指定のファイルに出力されます。  
[Register]ウィンドウ、[Dump]ウィンドウで直接レジスタ値やデータメモリの内容を修正した場合、あるいは[Source]ウィンドウ内でのダブルクリックによってブレークポイントを設定した場合も、対応するコマンドが[Command]ウィンドウに表示され、ファイルに出力されます。

コマンドの実行による[Source]、[Dump]、[Trace]、[Register]ウィンドウへの表示内容が、[Command]ウィンドウにも表示されます。オンザフライ情報も表示されます。  
ただし、ウィンドウ操作コマンドによる表示の更新や、各ウィンドウのスクロールバーまたは矢印キーによるスクロール結果は表示されません。

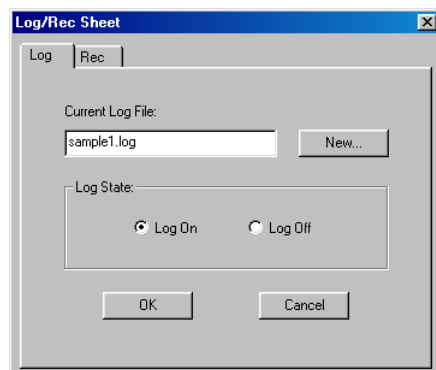
- 最初にlogコマンドを実行する場合は、それに続く実行コマンドと実行結果を記録するためのファイル名を指定する必要があります。
- 一度ログファイルが開かれると、それ以降はlogコマンドの実行ごとに記録を中断、再開(トグル)します。この切り換えは、デバッガを終了するまで有効です。ログ出力を別のファイルに変更するには、書式1を使用してファイルを指定し直してください。その時点でそれまでのログファイルは閉じられ、以降の記録は新しく指定したファイルに対して行われます。

## GUI

[Option ! Log...]メニューコマンド

このメニューコマンドを選択すると、ログファイルを指定するダイアログボックスが表示されます。新しいログファイルを指定するには、[Current Log File]にログファイル名を入力するか、[New...]ボタンをクリックして選択してください。

すでにログの記録を開始している場合は、[Log State]のラジオボタンで保存のON/OFFを切り換えることができます。



### 13.9.15 マップ情報表示コマンド

## ma (map information)

---

#### 機 能

パラメータファイルによって設定されたマップ情報を表示します。

#### 書 式

>ma␣ (直接入力モード)

#### 例

コマンド入力後、内部メモリ領域と外部メモリ領域の構成、I/O領域のマップ情報を表示します。

```
>ma␣
[Internal memory]
RAM 00F000 - 00F7FF
STK 00F500 - 00F7FF
LCD 00F800 - 00F842
LCD 00F900 - 00F942
LCD 00FA00 - 00FA42
LCD 00FB00 - 00FB42
LCD 00FC00 - 00FC42
LCD 00FD00 - 00FD42
[External memory]
ROM 000000 - 00BFFF
RAM 080000 - 080001
RAM 100000 - 107FFF
RAM 180000 - 1801FF
[I/O memory]
      0 1 2 3 4 5 6 7 8 9 A B C D E F
FF00  * * *
FF10  * * * *
FF20  * * * * * *
FF30  * * * * * * *
FF40  * * * * * *      * * *
FF50  * * * * * *
FF60  * * * *
FF70  * * * * * * * *
FF80
FF90
FFA0
FFB0
FFC0
FFD0
FFE0
FFF0
```

I/O領域の表示は、マップされているアドレスを"\*"によって示します。

#### GUI

なし

## 13.9.16 FPGA操作コマンド

**xfer** (xilinx fpga data erase)**機 能**

ICEに挿入されている標準ペリフェラルボード上のFPGAの内容を消去します。

**書 式**

>xfer↵ ( 直接入力モード )

**例**

```
>xfer↵  
>
```

コマンド入力後、消去実行前に確認のダイアログボックスが表示され、実行と中止が選択できます。

**注**

- 消去中はプログレスバーにより進捗状況がわかります。そのダイアログボックス上の[Cancel]ボタンか[ESC]キーで消去を中断できます。中断した場合、再度消去およびデータ書き込みを行うまで標準ペリフェラルボードは使用できません。
- 消去にはTBD分TBD秒程度( 最大 )の時間がかかります。

**G U I**

なし

## xfwr (xilinx fpga data write)

### 機 能

ICEに挿入されている標準ペリフェラルボード上のFPGAに周辺回路データを書き込みます。

### 書 式

```
>xfwr <file name> ;[H | S] [:N]␣      ( 直接入力モード )
      <file name>: FPGAデータファイル( .mot: モトローラS、.mcs: インテルHEX )
      H:          インテルHEXファイルを指定
      S:          モトローラSファイルを指定
      N:          書き込み前の消去を省略
```

### 例

```
>xfwr ..¥ice¥fpga¥c88xxx.mot ;S␣
>
```

この例では、FPGAを消去後、c88xxx.mot( モトローラSファイル )のデータをFPGAに書き込みます。

```
>xfwr ..¥ice¥fpga¥c88xxx.mot ;S ;N␣
>
```

この例では、データ書き込み前の消去を省略します。ただし、FPGAは事前に消去しておく必要があります。

### 注

- 書き込むデータファイルはセイコーエプソンが用意したものをそのまま使用してください。ファイル名の拡張子も.mo( モトローラS )、.mcs( インテルHEX )に固定で変更することはできません。不正なファイルを指定すると、エラーとなり書き込みは行えません。
- Nオプションは、事前にxferコマンドでFPGAを完全に消去している場合にのみ指定できます。消去していないFPGAにデータを書き込む場合、Nオプションを指定しないでください。
- 実行中はプログレスバーにより進捗状況がわかります。そのダイアログボックス上の[Cancel]ボタンか[ESC]キーで実行を中断できます。中断した場合、再度消去およびデータ書き込みを行うまで標準ペリフェラルボードは使用できません。
- データ書き込みには消去も含めTBD分程度( 最大 )の時間がかかります。

### GUI

なし

## xfcp (xilinx fpga data compare)

### 機能

標準ペリフェラルボード上のFPGAの内容と指定ファイルの内容を比較します。

### 書式

```
>xfcp <file name> ;{H | S}␣ ( 直接入力モード )
  <file name>: FPGAデータファイル( .mot: モトローラS、.mcs: インテルHEX )
  H:          インテルHEXファイルを指定
  S:          モトローラSファイルを指定
```

### 例

```
>xfcp ..¥ice¥fpga¥c88xxx.mot ;S␣
>
...エラーがなかった場合

>xfcp ..¥ice¥fpga¥c88yyy.mot ;S␣
Warning : Verify error
0X00000  0XFF
0X00001  0X84
0X00002  0XAB
      :      :
>
...データの不一致が見つかった場合
...FPGA内のエラーアドレスとデータを表示
```

### 注

- 実際に比較されるのは、指定ファイル内でデータが存在するアドレス範囲のみです。その範囲外にあるFPGA内のデータは比較されません。
- 比較するデータファイルはセイコーエプソンが用意したものをそのまま使用してください。ファイル名の拡張子も.mo( モトローラS )、.mcs( インテルHEX )に固定で変更することはできません。不正なファイルを指定すると、エラーとなります。
- 実行中はプログレスバーにより進捗状況がわかります。そのダイアログボックス上の[Cancel]ボタンか[ESC]キーで実行を中断できます。

### GUI

なし



## xdp (xilinx fpga data dump)

### 機 能

標準ペリフェラルボード上のFPGAの内容を、16バイト/行の16進ダンプ形式で[Command]ウィンドウに表示します。

### 書 式

```
>xdp <address1> [<address2>]↵      (直接入力モード)
<address1>: 表示開始アドレス 16進数
<address2>: 表示終了アドレス 16進数
条件:      0 address1 address2 FPGAエンドアドレス
```

### 例

<address1>のみを指定した場合、<address1>から256バイトのデータを表示します。

```
>xdp 0↵
Addr   +0 +1 +2 +3 +4 +5 +6 +7   +8 +9 +A +B +C +D +E +F
00000: FF 84 AB EF F9 D8 FF BB   FB BB BF FB BF BF FB BF
00010: BB FB BB BF BB BF FB BB   BF BF FB BB FF EE FF EE
00020: EF FE D7 FB FE EE EF EF   EE EE FE EE FB FE EF EF
      :           :
000E0: FF FF FF FF FB FF FF FF   BD DF FB FD DF FF FF FF
000F0: FF FF BF FF FF FF FF F9   FF FF FF FF FF FF FF FF
>
```

<address1>と<address2>を指定した場合、その範囲のデータを表示します。

```
>xdp 100 100↵
Addr   +0 +1 +2 +3 +4 +5 +6 +7   +8 +9 +A +B +C +D +E +F
00100: FF
>
```

### 注

- アドレスが16進数以外の場合、エラーとなります。
- 先頭アドレスが終了アドレスより大きい場合、エラーとなります。

### GUI

なし

## 13.9.17 終了コマンド

### q (quit)

---

#### 機 能

デバッガを終了します

#### 書 式

>q↵ ( 直接入力モード )

#### G U I

[File ! Exit]メニューコマンド

このメニューコマンドの選択によっても、デバッガを終了できます。

## 13.9.18 ヘルプコマンド

**? (help)****機 能**

各コマンドの入力書式を表示します。

**書 式**

- (1)? (直接入力モード)  
 (2)? <n> (直接入力モード)  
 (3)? <command> (直接入力モード)
- <n>: コマンドグループ番号 10進数  
 <command>: コマンド名  
 条件: 1 n 6

**例**

書式1、書式2のコマンド入力によって機能別のコマンド一覧を表示します。  
 個別のコマンドの入力書式は書式3のコマンド入力によって表示させます。

```
>?↓
group 1: data & register ..... dd,de,df,dm,ds/rd,rs
group 2: execution & break ..... g,gr,s,n,se,rst/bp,bpa,bpr,bc(bpc),bas,ba,bar,bd,bdr,bl,bac
group 3: source & symbol ..... u,sc,m/sy/w
group 4: file & flash rom ..... lf, par/xfer,xfwr,xfcp,xcp
group 5: trace & coverage ..... td,ts,tf/cv,cvc
group 6: others ..... par/com,cmw,rec/log/ma/q/?
Type "? <group #>" to show group or "? <command>" to get usage of the command.
>? 1↓
group 1: data & register
dd (data dump), de (data enter), df (data fill), dm (data move), ds (data search)
rd (register display), rs (register set)
Type "? <command>" to get usage of the command.
>? dd↓
dd (data dump): dump memory content with hexadecimal format
usage: dd [addr1] [addr2] [unit] ... dump from 0x0 in byte unit if without parameter
        dd [addr1] [@size] [unit] ... dump from 0x0 in byte unit if without parameter
unit: display unit (-B (default) / -W / -L / -F / -D)
```

**GUI**

なし

## 13.10 エラーメッセージ

### デバッグエラー

エラーメッセージ	メッセージ内容
Error : Address out of range : use 0x000000 - 0xfffff	指定されたアドレスは有効範囲外です
Error : Address out of range, use 0 - 0x7FFFFF	プログラムメモリ領域外のアドレスが指定されました
Error : Address out of range, use 0 - 0xFFFFF	データメモリ領域外のアドレスが指定されました
Error : Cannot open device(ICE88UR)	ICEとの接続に失敗しました
Error : Cannot open file	ファイルがオープンできません
Error : Checksum error	チェックサムでエラーになりました
Error : Coverage mode is off or the coverage mode is not supported	カバレッジのモードがOFFまたは、当該ICEはカバレッジをサポートしていません
Error : Data out of range, use 0 - 0xFF	指定の数値はデータの有効範囲外です
Error : DLL Initialization error	DLLの初期化に失敗しました
Error : End address < start address	開始アドレスより小さい終了アドレスが指定されました
Error : End index < start index	開始サイクルより小さい終了サイクルが指定されました
Error : Error file type (extension should be CMD)	指定のファイル拡張子は、コマンドファイルとして無効です
Error : Error file type (extension should be PAR)	指定のファイル拡張子は、パラメータファイルとして無効です
Error : Failed ICE88UR initialization	ICEの初期化に失敗しました
Error : Failed to initialize DLL : %s	DLLの初期化に失敗しました
Error : Failed to Load DLL	DB88起動に必要なDLLのロードに失敗しました
Error : Failed to open : %s	ファイルを開けませんでした
Error : Failed to read BA	BAレジスタ読み込みエラー
Error : Failed to read BR	BRレジスタ読み込みエラー
Error : Failed to read CB	CBレジスタ読み込みエラー
Error : Failed to read CC	CCレジスタ読み込みエラー
Error : Failed to read EP	EPレジスタ読み込みエラー
Error : Failed to read file : %s	ファイルの読み込みに失敗しました
Error : Failed to read HL	HLレジスタ読み込みエラー
Error : Failed to read NB	NBレジスタ読み込みエラー
Error : Failed to read PC	PCレジスタ読み込みエラー
Error : Failed to read SC	SCレジスタ読み込みエラー
Error : Failed to read SP	SPレジスタ読み込みエラー
Error : Failed to read X	Xレジスタ読み込みエラー
Error : Failed to read Y	Yレジスタ読み込みエラー
Error : Failed to read DLL : %s	DLLのロードに失敗しました
Error : Failed to write BA	BAレジスタ書き込みエラー
Error : Failed to write BR	BRレジスタ書き込みエラー
Error : Failed to write CB	CBレジスタ書き込みエラー
Error : Failed to write CC	CCレジスタ書き込みエラー
Error : Failed to write EP	EPレジスタ書き込みエラー
Error : Failed to write HL	HLレジスタ書き込みエラー
Error : Failed to write NB	NBレジスタ書き込みエラー
Error : Failed to write PC	PCレジスタ書き込みエラー
Error : Failed to write SC	SCレジスタ書き込みエラー
Error : Failed to write SP	SPレジスタ書き込みエラー
Error : Failed to write X	Xレジスタ書き込みエラー
Error : Failed to write Y	Yレジスタ書き込みエラー
Error : ICE88UR Diagnostic error	ICE自己診断処理でエラーが検出されました
Error : Ice88ur Initialization failed	ICEの初期化に失敗しました
Error : Ice88ur is already running	ICE88UR.EXEが起動されています (DB88とICE88URは同時に起動できません)
Error : ICE88UR is turned off	ICEの電源がOFFになっています
Error : Illegal initialization packet data	初期化パケットエラー
Error : Incorrect number of parameters	コマンドのパラメータ数が不正です
Error : Incorrect r/w option, use r/w/*	無効なR/Wオプションが指定されました
Error : Incorrect register name, use PC/SP/IX/IY/A/B/HL/BR/CB/EP/XP/YP/SC	無効なレジスタ名が指定されました
Error : Index out of range, use 0 - 8191	指定のトレースサイクル番号は、有効範囲外です
Error : Initialization failed! Please quit and restart!	DB88の初期化に失敗しました。DB88を再起動してください
Error : Input address does not exist	未設定のブレークポイントアドレスが指定されました
Error : Invalid command	無効なコマンドが入力されました

エラーメッセージ	メッセージ内容
Error : Invalid data pattern	入力データパターンが不正です
Error : Invalid display unit, use -B/-W/-L/-F/-D	表示単位の指定が無効です
Error : Invalid DLL ModuleID	DLL識別エラー
Error : Invalid file name	指定のファイル拡張子は、プログラムファイルまたはファンクションオプションファイルとして無効です
Error : Invalid fsa file	FSAファイルが不正です
Error : Invalid hexadecimal string	不正な16進数文字列です
Error : Invalid value	入力した値が不正です
Error : Maximum nesting level(5) is exceeded, cannot open file	コマンドファイルのネストレベルが、制限を越えました
Error : Memory ranges in %s are invalid or the file is not exist	CPU INIファイルのメモリ範囲が不正です
Error : No symbol information	シンボル情報がありません (シンボルファイルがロードされていません)
Error : Number of steps out of range, use 0 - 65535	指定ステップ数は制限を越えています
Error : The Memory Area cannot include the boundary between 0x00FFFF and 0x010000	指定されたエリアは、0x00FFFF-0x010000の境界を含んでいます
Error : The Memory Area must be above 0x10000, and longer than 256 bytes	0x010000以上で領域を指定する場合、256バイトより小さいサイズは指定できません
Error : This command is not supported in current mode	トレース/カバレッジコマンドは、トレースOFF/カバレッジOFF時は無効です
Error : Unable to get the coverage area number	カバレッジエリア番号取得に失敗しました
Error : Unable to get the coverage mode	カバレッジ情報の取得に失敗しました
Error : Unable to set SelfFlash check function	自己書き換えチェック機能が設定できませんでした
Error : Unable to set the coverage area number	カバレッジエリア番号設定に失敗しました
Error : Unable to set the coverage mode	カバレッジモード設定に失敗しました
Error : Wrong Command line parameter	起動パラメータに誤りがあります
Please load the selfflash library program	自己書き換えライブラリプログラムをロードしてください (自己書き換え機能を有効にした場合、ライブラリプログラムをロードする必要があります)
Warning : 64 break addresses are already set	64ヶ所を越えるブレークポイントが指定されました
Warning : Break address already exists	指定のアドレスは既にブレークポイントに設定されています
Warning : Identical break address input	コマンドラインに同じアドレスが2回以上指定されています
Warning : Memory may be modified by SelfFlash	自己書き換えプログラムにより、メモリが書き換えられている可能性があります
Warning : SelfFlash program area is out of the current software pc break area. Please clear the break point(Address)	自己書き換えプログラムエリアが、現在設定されているソフトウェアブレイクエリアと一致していません。(Address)に設定されているブレークポイントを解除してください (解除しない場合、予期せぬ場所でプログラムが停止する場合があります)

## ICEハードウェアエラー

エラーメッセージ	メッセージ内容
Error : Cannot be run in Free-Run mode	ICEはフリーラン動作中です
Error : Cannot find specified data	指定したデータは見つかりません ( 検索の結果、該当するデータは見つかりませんでした )
Error : ICE88UR is still keep a conservative mode	ICEは保守モード動作中です
Error : ICE88UR power off execution abort	ICE本体の電源が入っていません。実行を中断しました ( プログラム実行中に本体の電源供給が断たれました )
Error : Insufficient memory for loading program	メモリの確保に失敗しました ( Windowsのシステムリソースが不足していると思われます。リソース残量を確認の上、いくつかのアプリケーションを終了してください )
Error : Vdd down or no clock	ターゲットシステムの電源電圧が低下しているか、電源が入っていない、もしくはクロックが供給されていません。 ( パラメータファイルの"Vdddown="を"1"に設定しているときのみ有効です )
Error : Verify error	ベリファイエラーが発生しました
ICE88UR system error : ?? illegal packet	不正なパケットを検出しました
ICE88UR system error : Command timeout	コマンドタイムアウトを検出しました
ICE88UR system error : Firmware packet error	EB:Firmwareパケットでエラーを検出しました
ICE88UR system error : Master reset	MR:マスタリセットを検出しました
ICE88UR system error : Not connected	ICEが未接続または電源が入っていません
ICE88UR system error : Not ready	ICEの準備ができていません
Internal error : ICE88UR does not support this command version	本バージョンでは対応していません ( 速やかにDB88デバッグを終了してください )
Internal error : Illegal error code fetched. System crash possible	存在しないエラーコードが見つかりました (速やかにICE88URデバッグを終了してください)
Processing terminated by hitting ESC-key	ESCキーにより処理を中断しました

# Appendix A アセンブラ( サブツールチェーン )

## A.1 パッケージの概要

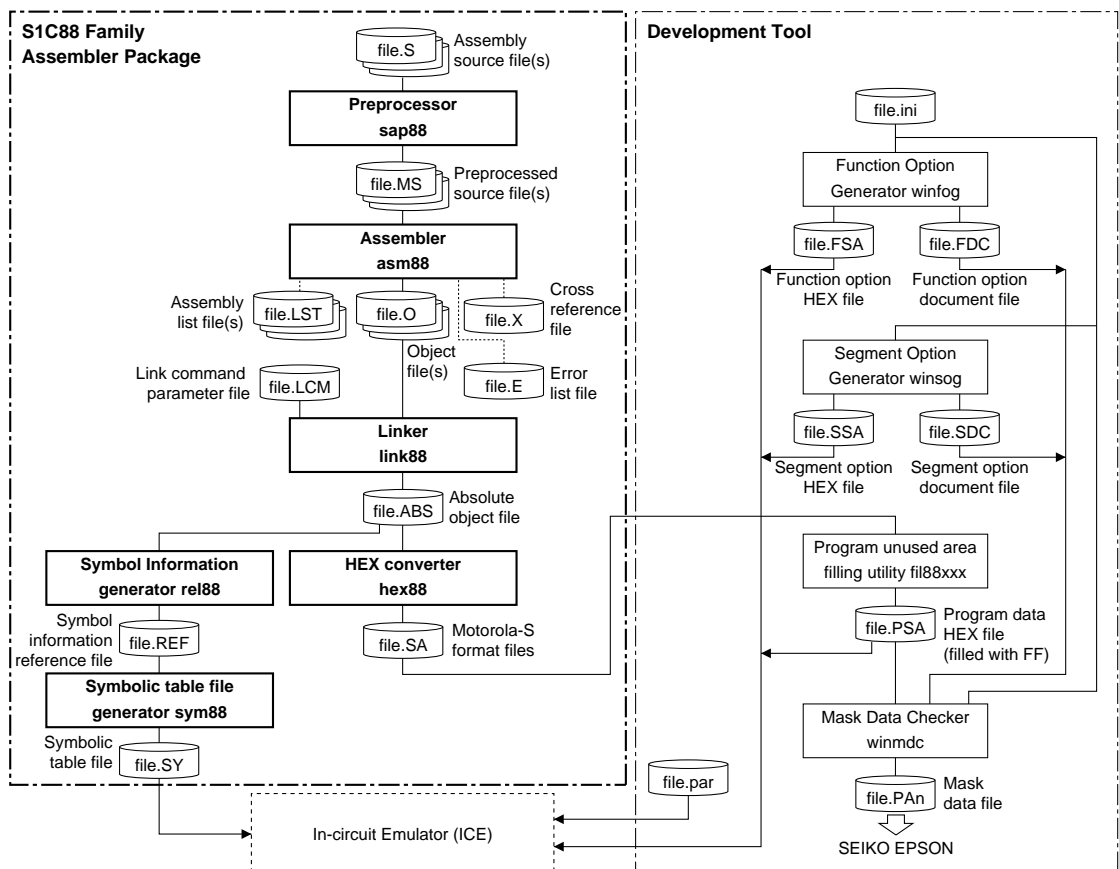
### A.1.1 はじめに

"S1C88 Familyアセンブラ"はCMOS 8ビットシングルチップマイクロコンピュータS1C88 Familyのソフトウェア開発ツールの1つで、プログラム作成用のクロスアセンブラを中心にリンカ、ユーティリティ等で構成されています。

このパッケージはS1C88 Family全ての機種に共通で、マクロ機能によるプログラムの開発が行えます。

### A.1.2 ソフトウェアツールの概要

図A.1.2.1に構造化アセンブラのソフトウェア開発フローを示します。



図A.1.2.1 構造化アセンブラソフトウェア開発フロー

各プログラムの基本的な機能の概要は以下のとおりです。

### 構造化プリプロセッサ<sap88>

構造化プリプロセッサsap88は、クロスアセンブラasm88にマクロ機能を付加するためのプリプロセッサです。

マクロ機能を記述したアセンブリソースファイルを作成後、最初にsap88を通してasm88でアセンブル可能なソースファイル(マクロがS1C88の命令セットに展開されたファイル)に変換してからasm88を実行します。

### クロスアセンブラ<asm88>

クロスアセンブラasm88は、S1C88の命令セットや擬似命令が記述されたプログラムのソースファイルをアセンブルし、機械語に変換します。

asm88はモジュール別開発のためのリロケータブルアセンブルに対応しています。リロケータブルアセンブルでは、リンカによって他のモジュールと連結するためのリロケータブルオブジェクトファイルが生成されます。

### リンカ<link88>

asm88によって生成されたリロケータブルオブジェクトファイルを、複数の場合は連結して、1つのアブソリュート(バイナリ形式)オブジェクトファイルに変換します。

### その他のユーティリティ

本パッケージには前述の主要なプログラム以外に、以下に挙げるユーティリティプログラムが含まれています。

#### (1) シンボル情報生成ユーティリティ<rel88>

リロケータブルオブジェクトファイルのシンボリックテーブル等の情報を取得するプログラムです。シンボリックテーブルの作成の前処理に使用します。

#### (2) バイナリ/HEXコンバータ<hex88>

バイナリファイルをもとローラS2フォーマットのHEXファイル(ASCIIファイル)に変換します。

基本的には、リンカlink88が出力したアブソリュートオブジェクトファイルをプログラムデータHEXファイルに変換するために使用します。変換後のプログラムデータHEXファイルはハードウェアツールによるデバッグおよびマスクデータ作成のもとになるものです。

#### (3) シンボリックテーブルファイル生成ユーティリティ<sym88>

sym88は、シンボル情報生成ユーティリティrel88からファイルリダイレクトによって生成されたシンボル情報ファイルを、ICEにてシンボリックデバッグが可能となるシンボリックテーブルファイルに変換します。

### 一括処理(バッチファイル)

効率よくプログラム開発が行えるよう、基本的な一連のツールと操作を自動的に一括処理するバッチファイルを準備しています。必要に応じてカスタマイズを行い、利用してください。

- ra88.bat: リロケータブルアセンブル用バッチファイル
- lk88.bat: リンク用バッチファイル

バッチファイルの処理内容とカスタマイズ方法については、A.2項のそれぞれの処理に該当する箇所で説明します。



## A.2 プログラム開発手順

ここでは、始めにプログラム開発の流れを説明し、その流れに添って本パッケージの各ソフトウェアツールの使用法を説明します。なお、各ソフトウェアツールについては一括処理用のバッチファイルの使用コマンドを例として基本的な処理手順と、それに必要なフラグ設定(起動コマンドフラグ)を説明しますので、それ以外のフラグ等についてはAppendix Cの各ツールのリファレンスを参照してください。

### A.2.1 開発フロー

クロスアセンブラasm88を用いたプログラム開発は、基本的に次のようになります。

#### <リロケータブルアセンブルとリンク>

プログラム全体を複数のモジュールとして作成する (モジュール別開発)

リロケータブルアセンブルはプログラムを処理内容別等の複数に分割し(分割したそれぞれの部分をモジュールと呼びます)、モジュール別に開発する場合のアセンブル方法です。

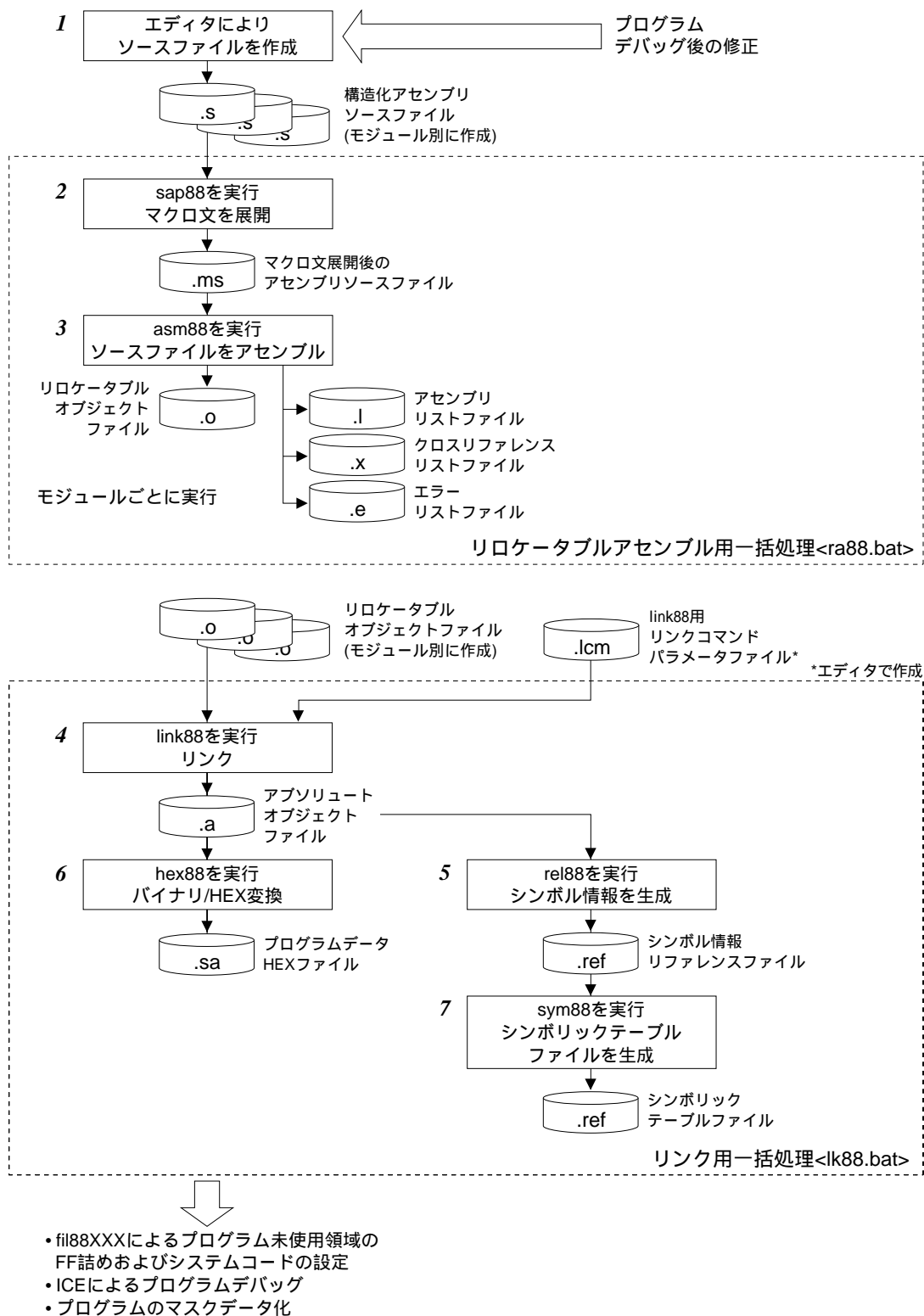
モジュール別のアセンブリソースファイルはエディタ等で作成したものの他にsap88によるマクロ展開出力も使用できます。

アセンブル後の各モジュール(リロケータブルオブジェクトファイル)はリンクによって連結し、1つのプログラムにまとめます。各モジュールが配置されるプログラムメモリのアドレスはリンクによって確定します。したがって、ソースプログラム作成時点ではアドレスを意識せずに開発を進めることができます。この方法では、小さく分割したモジュールごとにデバッグが行えるため、デバッグ効率が上がります。

リロケータブルアセンブルによるプログラム開発フローを図A.2.1.1に示します。なお、一連のツールによる処理をバッチファイルの形で1つにまとめた"ra88.bat"および"lk88.bat"が本パッケージに収められていますので、必要に応じカスタマイズを行い利用してください。("ra88.bat"および"lk88.bat"の詳細については"A.2.3.4 リロケータブルアセンブルの一括処理"および"A.2.4.5 リンクの一括処理"を参照してください。)

注: リロケータブルモジュールを作成する場合、それぞれのモジュールのプログラムサイズを1バンクに納まる32Kバイト以内としてください。それを越えるモジュールはリンクの際にエラーとなりますので、32Kバイト以下に分割する必要があります。同様に、データ部のサイズは1ページに納まる64Kバイト以内としてください。

また、リンクの際、モジュールがバンク境界にまたがった形で再配置を行うことはできません。その場合、モジュールが次のバンクの先頭から始まるように配置されます。このため、全てのモジュールを大きなサイズで作成するとプログラムメモリの無駄な領域(未使用領域)が増加します。これを防ぐため、各モジュールのサイズに対する配慮も行ってください。



図A.2.1.1 リロケートブルアセンブルの開発フロー

## A.2.2 ソースファイルの作成

使用ソフト ..... エディタ

ソースファイルはお手持ちのエディタを使用して作成してください。

小規模なアプリケーションでは、プログラム全体を単一モジュールとしてアセンブラ言語のみによって作成することができます。

また、単一モジュールでも構造化プリプロセッサsap88のINCLUDE擬似命令を使用することにより、ソースファイルの分割も可能です。

通常はデバッグの点も考慮して適切なモジュール分けを行い、それぞれのモジュール個々にソースファイルを作成します。

アセンブラ言語のモジュールはS1C88のCPUの命令セットやアセンブラに備わる擬似命令等を用いてソースファイルを作成します。

アセンブリソースファイルのファイル名は拡張子を".s"としてください。

ソースプログラムの各ステートメント(行)は基本的に次の書式で記述します。

シンボルフィールド	ニーモニックフィールド	オペランドフィールド	コメントフィールド
-----------	-------------	------------	-----------

- ・ **シンボルフィールド:**  
このフィールドにはシンボルを記述します。シンボルの直後には、EQU命令またはSET命令のステートメントを除いてコロン(:)を必ず付けます。
- ・ **ニーモニックフィールド:**  
このフィールドにはオペコード、擬似命令を記述します。
- ・ **オペランドフィールド:**  
このフィールドには、各命令のオペランドや定数、変数、定義したシンボル、メモリアドレスを示すシンボル、演算式を記述します。
- ・ **コメントフィールド:**  
このフィールドの先頭には、セミコロン(;)を置き、以降にコメント文を記述します。

ソースファイルの作成についての詳細はAppendix Bに説明されていますので、そちらを参照してください。本アセンブラでは、構造化プリプロセッサsap88で提供されるマクロ文、クロスアセンブラasm88に備わる各種擬似命令を使用することができます。ここでは、これらの概要のみに触れておきます。

### <命令セット>

S1C88 Familyの全機種はコアCPUにS1C88を使用しており、命令セットはCPUのMODELやモードの制限を除き共通となっています。命令セットの詳細については"S1C88コアCPUマニュアル"を、各機種に内蔵した周辺回路の制御プログラム例等については"S1C88xxxテクニカルマニュアル"を参照してください。クロスアセンブラasm88は、S1C88の持つ命令セットの全てのニーモニックを機械語に変換することができます。

### <マクロ文>

マクロはプログラム中でよく使用する処理(一連の命令)を任意の名前であらかじめ定義しておき、プログラム内からその名前呼び出すようにするもので、同様のルーチンをその都度記述する手を省くことができます。(詳細はAppendix Bを参照してください。)

マクロ文はsap88の擬似命令として提供されており、sap88を通すことによってアセンブル可能なニーモニックとしてマクロ呼び出し箇所に展開されます。

## マクロ文定義例

## 展開前

```

        subtitle          "example"
        public            main,work
        external          src_address,dst_address,counter
;
abc     equ               0ffh
;
        data
work: db                  [1]
;
        code
;*****
;**      * macro define *          **
;*****
nop3 macro
    nop
    nop
    nop
    endm
;*****
;**      * example *              **
;*****
main:
    ld      a,#abc
    lb      b,[work]
    nop3                                ; macro call ***
    ld      ix,#src_address
    ld      iy,#dst_address
    ld      hl,[counter]
;***
    end

```

マクロ文定義

マクロ呼び出し

## 展開後

```

        subtitle          "example"
        public            main,work
        external          src_address,dst_address,counter
;
abc     equ               0ffh
;
        data
work: db                  [1]
;
        code
;*****
;**      * macro define *          **
;*****
;*****
;*****
;**      * example *              **
;*****
main:
    ld      a,#abc
    lb      b,[work]
    nop
    nop
    nop
    ld      ix,#src_address
    ld      iy,#dst_address
    ld      hl,[counter]
;***
    end

```

ニーモニックに展開  
されたマクロ文

## &lt;擬似命令&gt;

機能別擬似命令	説 明
領域設定擬似命令 (CODE, DATA)	セクション指定*を行います。 *プログラム領域とデータ領域を指定します。 (詳細は"A.2.3.2 クロスアセンブラ(asm88)"を参照してください。)
データ定義擬似命令 (DB, DW, DL, ASCII, PARITY)	プログラムメモリ領域内に各種のデータを設定します。
シンボル定義擬似命令 (EQU, SET)	ソースプログラム中で使用するシンボル(任意の名前)に定数を割り当てます。
ロケーションカウンタ制御擬似命令 (ORG)	プログラムカウンタの設定を行います。
外部定義・外部参照擬似命令 (EXTERNAL, PUBLIC)	モジュール間でシンボルやラベルを参照し合えるようにします。
ソースファイル挿入擬似命令 (INCLUDE) sap88 only	他のソースファイルの内容を任意の箇所に挿入します。
アセンブル終了擬似命令 (END)	アセンブル終了箇所を指定します。
マクロ関係擬似命令 (MACRO ~ ENDM, DEFINE, LOCAL, PURGE, UNDEF, IRP ~ ENDR, IRPC ~ ENDR, REPT ~ ENDR) sap88 only	マクロ定義等を行います。
条件アセンブル擬似命令 (IFC ~ ENDIF, IFDEF ~ ENDIF, IFNDEF ~ ENDIF) sap88 only	シンボルの定義状態にしたがって、アセンブルを行うかスキップさせるかを設定できます。
出力リスト制御擬似命令 (LINENO, SUBTITLE, SKIP, NOSKIP, LIST, NOLIST, EJECT)	アセンブリリストファイルへの出力を制御します。

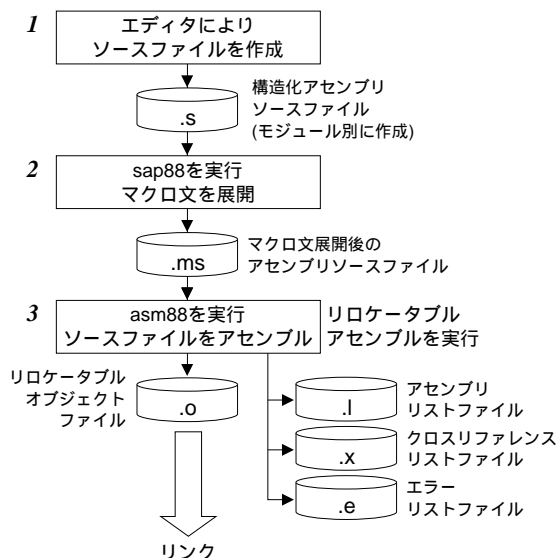
擬似命令はsap88、asm88に対する実行制御命令で、CPUの命令セットのように直接的にアプリケーションプログラムを構成するものではありません。

本アセンブラで利用できる擬似命令を機能別に分類すると上記のようになります。(詳細はAppendix Bを参照してください。)

## A.2.3 アセンブル

ここでは、アセンブリソースファイルのアセンブルの方法、それによって生成されるリロケータブルオブジェクトファイル等について説明します。

使用ソフト ..... **sap88, asm88**



図A.2.3.1 リロケータブルアセンブルのフローチャート

### A.2.3.1 構造化プリプロセッサ(sap88)

本アセンブラシステムは、構造化プリプロセッサsap88とクロスアセンブラasm88で構成されます。

A.2.2項に述べたように、sap88はマクロ文をニーモニックに展開する役割を持ちます。

asm88ではマクロ文を解釈できませんので、これらを含むアセンブリソースファイルを直接asm88の入力ファイルとすることはできません。

また、ニーモニックを機械語に変換するアセンブラ本体はasm88で、sap88ではアセンブルは行えません。したがって、構造化アセンブルはsap88とasm88の両ツールを使用する必要があります。仮に構造化アセンブルが必要ない場合においても、ソースファイルには特に影響を与えませんので、sap88を通すことを推奨します。

sap88は".s"の拡張子を持つアセンブリソースファイルを入力し、マクロ文を展開した出力ファイルを生成します。出力ファイル名の拡張子は".ms"として設定します。

### A.2.3.2 クロスアセンブラ(asm88)

クロスアセンブラasm88は、S1C88 FamilyのCPUの命令セットおよびasm88に備わる擬似命令をアセンブルし機械語に変換します。

asm88は、リロケータブルアセンブルに対応しています。

リロケータブルアセンブルでは、リンカによって他のモジュールと連結するためのリロケータブルオブジェクトファイル(".o")が生成されます。また、asm88は複数のアセンブリソースファイルの入力が可能で、同時に複数のリロケータブルモジュールのアセンブルを行うことができます。

asm88はまた、プログラマーのためにアセンブリリスト(".i")、エラーリスト(".e")そしてクロスリファレンスリスト(".x")の3種類のリストを出力します。

アセンブリリストは、行番号、ターゲットアドレス、ソースに対応するコード、ソースステートメントで構成され、行番号は10進数、アドレスとコードは16進数で出力されます。

アセンブル時にエラーが発生した場合は、ソースファイル名、エラーの発生した行番号、エラーのレベル、そしてエラーメッセージで構成されるエラーリストファイルが作成されます。さらにアセンブリリストファイルにもエラーの発生した行番号にマーク"\*\*\*"が付きます。また、エラーが発生してもそれが致命的なエラーでない限り処理は続行されます。

また、クロスリファレンスリストによって、ファイル内のシンボル定義と参照の関係をたやすく把握できるように考慮されています。

これらは別々のファイルとして生成されますので、ファイルの管理がしやすくなっています。

### <プログラムとデータのメモリ管理について>

ここで、プログラムとデータのメモリ管理について触れておきます。

S1C88XXXのメモリマップはプログラムコード用のプログラムメモリ(ROM)とRAMやI/Oメモリ等のデータメモリに分類されます。

たとえば、あるシンボルをアセンブリソースファイルの任意の場所に記述しても、asm88はそれがプログラムメモリ内か、あるいはデータメモリ内かを判断することができません。

そこで、あらかじめ領域設定擬似命令を記述して、全ての行がどちらのメモリに属するかを明確にしておく必要があります。

以下にリロケータブルアセンブルの領域指定方法と、それに対応したasm88による処理について説明します。

#### 領域設定

リロケータブルアセンブルでは、各モジュールの配置される絶対アドレスはリンク時に指定あるいは決定します。したがって、アセンブリソースファイル内で絶対アドレスは指定できません。ORG擬似命令によって相対的なアドレス指定は行えますが、その場合、相対アドレスの基準となるものがが必要です。また、asm88に対してプログラム領域とデータ領域の区分点を指定する必要もあります。

本アセンブラでは、プログラムの全体はCODEとDATAの2つに分類されます。これらは基本的に次の領域を表します。

CODE領域 ... ROMに書き込まれるプログラムデータ領域

DATA領域 ... ROM以外のデータメモリ領域

asm88にはセクションを指定するCODE擬似命令とDATA擬似命令が設定されており、アセンブリソースファイル内に記述することによって領域の設定が行えます。

#### • CODEセクションの指定

アセンブリソースファイル中にCODE擬似命令が記述されていると、asm88はそれ以降DATA擬似命令が現われるまでをCODEセクションに配置するものとしてアセンブルを行います。1つのモジュールの中でも、CODE擬似命令は複数箇所で使用することができます。asm88はモジュール内のCODEセクションの先頭を相対アドレス0000HとしてCODE擬似命令が現われた順序で連続的に並べ換え、1つのブロックとしてまとめます。つまり、1つのモジュールのCODE指定領域が1つのCODEセクションとして扱われます。(図A.2.3.2.1参照)

各モジュールのCODEセクションはリンクによってさらに全体がまとめられます。リンクはプログラムメモリ領域のバンク管理に合わせ、セクション単位でリンクの処理を行います。

CODEセクションは1つ、あるいは複数のモジュールのCODEセクションで構成され、その最大サイズは1バンクと同じ32Kバイトに制限されます。(セクション管理の詳細は"A.2.4.2 セクション管理"で説明します。)したがって、各モジュールを作成する場合にはコード部のサイズが32Kバイトを越えないように注意してください。asm88の起動時フラグ-ROM#を使用することによってCODEセクションの容量チェックが行えますので、これを利用してください。たとえば、"-ROM 32768"のフラグ指定を行うと、1つのモジュールのCODEセクションが32Kバイトを越えた場合にエラーが表示されます。

#### • DATAセクションの指定

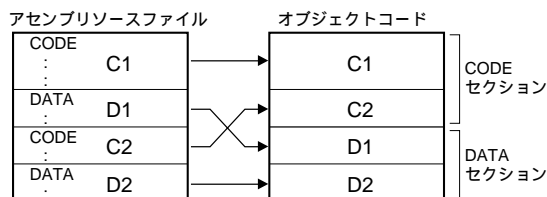
アセンブリソースファイル中にDATA擬似命令が記述されていると、asm88はそれ以降CODE擬似命令が現われるまでをDATAセクションに配置するものとしてアセンブルを行います。1つのモジュールの中でも、DATA擬似命令は複数箇所で使用することができます。asm88はモジュール内のDATAセクションの先頭を相対アドレス0000HとしてDATA擬似命令が現われた順序で連続的に並べ換え、1つのブロックとしてまとめます。つまり、1つのモジュールのDATA指定領域が1つのDATAセクションとして扱われます。(図A.2.3.2.1参照)

各モジュールのDATAセクションはリンクによってさらに全体がまとめられます。リンクはデータメモリ領域のページ管理に合わせ、セクション単位でリンクの処理を行います。DATAセクションは1つ、あるいは複数のモジュールのDATAセクションで構成され、その最大サイズは1ページと同じ64Kバイトに制限されます。(セクション管理の詳細は"A.2.4.2 セクション管理"で説明します。)したがって、各モジュールを作成する場合にはデータ部のサイズが64Kバイトを越えないように注意してください。



asm88の起動時フラグ-RAM#を使用することによってDATAセクションの容量チェックが行えますので、これを利用してください。

たとえば、"-RAM 65535"のフラグ指定を行うと、1つのモジュールのDATAセクションが64Kバイトを越えた場合にエラーが表示されます。



図A.2.3.2.1 CODEセクションとDATAセクション

#### ・注意

リロケータブルアセンブルにおいて、CODE擬似命令およびDATA擬似命令のどちらか一方が記述されていない場合、エラーとなります。そのため、使用する領域設定擬似命令、プログラムメモリにはCODEおよびデータメモリにはDATA擬似命令を必ず使用してください。

### A.2.3.3 sap88、asm88の起動方法

#### <sap88の操作方法>

- (1) 構造化アセンブリソースファイル(.s)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでsap88を起動します。

**sap88\_ [フラグ]\_入力ファイル名**

\_はスペースの入力を表します。

はリターンキーの入力を表します。

リロケータブルアセンブル(ra88.bat)の一括処理で使用しているフラグを以下に示します。

フラグ	説 明
-o <ファイル名>	出力ファイル名を指定します。(出力ファイル名の拡張子は".ms"としてください。) このフラグを省略した場合、標準出力に出力されます。

他のフラグについてはAppendix Cを参照してください。

例 A:¥USER>a:¥EPSON¥sap88 -o sample.ms sample.s

AドライブのサブディレクトリUSER内に作成されているアセンブリソースファイル"sample.s"を入力し、asm88に入力するアセンブリソースファイル"sample.ms"を入力ファイルと同じディレクトリに生成します。

sap88へのPATHが設定されている場合は、sap88の前のパス指定は不要です。

入出力ファイルや表示されるメッセージについては、"A.2.3.9 アセンブルの実行例"を参照してください。

#### <asm88の操作方法>

- (1) sap88で生成したアセンブリソースファイル(.ms)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでasm88を起動します。

**asm88\_ [フラグ]\_入力ファイル名**

\_はスペースの入力を表します。

はリターンキーの入力を表します。

フラグは省略可能です。



リロケータブルアセンブル(ra88.bat)の一括処理で使用しているフラグを以下に示します。

フラグ	説 明
-ROM#	ROM容量をバイト単位で指定します。リロケータブルアセンブル時に有効で、CODE領域のサイズのチェックに使用されます。
-RAM#	RAM容量をバイト単位で指定します。リロケータブルアセンブル時に有効で、DATA領域のサイズのチェックに使用されます。

他のフラグについてはAppendix Cを参照してください。

例1 リロケータブルアセンブルで複数のアセンブリソースファイルを続けてアセンブルする場合

```
A:¥USER>a:¥EPSON¥asm88 sample1.ms sample2.ms
```

AドライブのサブディレクトリUSER内に作成されているアセンブリソースファイル"sample1.ms"および"sample2.ms"を入力し、リロケータブルアセンブル実行後、リロケータブルオブジェクトファイル"sample1.o"および"sample2.o"を入力ファイルと同じディレクトリに生成します。同時に、アセンブリリストファイル"sample1.l"と"sample2.l"、クロスリファレンスリストファイル"sample1.x"と"sample2.x"、エラーリストファイル"sample1.e"と"sample2.e"も同じディレクトリに生成されます。

asm88へのPATHが設定されている場合は、asm88の前のパス指定は不要です。

例2 リロケータブルアセンブルでROM、RAMの容量チェックを含めてアセンブルする場合

```
A:¥USER>a:¥EPSON¥asm88 -ROM 32768 -RAM 65536 sample.ms
```

AドライブのサブディレクトリUSER内に作成されているアセンブリソースファイル"sample.ms"を入力し、リロケータブルアセンブル実行後、リロケータブルオブジェクトファイル"sample.o"を入力ファイルと同じディレクトリに生成します。

同時に、アセンブリリストファイル"sample.l"、クロスリファレンスリストファイル"sample.x"、エラーリストファイル"sample.e"も同じディレクトリに生成されます。

アセンブル時、-ROMフラグと-RAMフラグによりCODEセクションとDATAセクションの容量チェックが行われます。この例では、CODEセクションが32Kバイトを、DATAセクションが64Kバイトを越えるとエラーとなります。

asm88へのPATHが設定されている場合は、asm88の前のパス指定は不要です。

入出力ファイルや表示されるメッセージについては、"A.2.3.9 アセンブルの実行例"を参照してください。

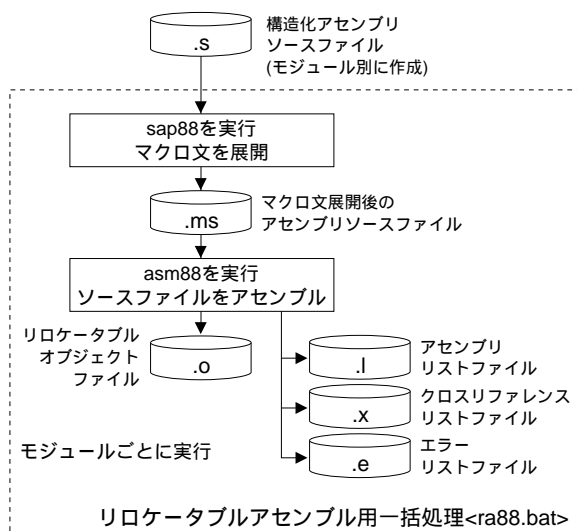
#### A.2.3.4 リロケータブルアセンブルの一括処理(ra88.bat)

前項でsap88、asm88個々の起動方法を述べましたが、それらをバッチファイルとしてまとめると1つの操作で一括処理が行えます。

バッチファイルはお客さまが任意に作成することもできますが、本パッケージにはリロケータブルアセンブル用のra88.batというバッチファイルが収められていますので、ここではこのバッチファイルの内容と使用方法を説明します。

このバッチファイルは汎用的に使用できるような処理内容となっています。必要に応じてフラグの設定等のカスタマイズを行い、有効に活用してください。

ra88.batの処理フローを図A.2.3.4.1に示します。



図A.2.3.4.1 ra88.batの処理フロー

## &lt;処理概要&gt;

ra88.batは指定のアセンブリソースファイルを入力し、sap88、asm88を順次実行してリロケートブルアセンブルを行い、リロケートブルオブジェクトファイルを生成します。sap88は複数のアセンブリソースファイルの入力を許可していませんので、sap88のINCLUDE擬似命令で複数の構造化アセンブリソースファイルを読み込む場合を除き、1モジュールごとのアセンブルに限定してあります。

## &lt;入出力ファイル&gt;

ra88.batの入出力ファイルを以下に示します。

## 入力ファイル

構造化アセンブリソースファイル(リロケートブル): file\_name.s

エディタで作成した構造化アセンブリソースファイル(リロケートブル)です。

## 出力ファイル

## 1. アセンブリソースファイル: file\_name.ms

sap88にてマクロ展開されたアセンブリソースファイルが出力されます。

## 2. リロケートブルオブジェクトファイル: file\_name.o

リロケートブルアセンブルによって再配置可能な機械語に変換されたバイナリファイルです。(またこのファイルは、リンク用一括処理lk88.batの入力ファイルでもあります。)

## 3. アセンブリリストファイル: file\_name.l

アセンブルによって変換された機械語とそのアドレス(CODEセクションまたはDATAセクションの先頭を000000Hとした相対アドレス)が各ステートメントに対応したリストとして出力されるファイルです。

## 4. クロスリファレンスリストファイル: file\_name.x

シンボル定義と参照が行われているアドレスのリストです。

## 5. エラーリストファイル: file\_name.e

アセンブル時に発生したエラーのリストです。

## &lt;操作方法&gt;

- (1) 構造化アセンブリソースファイル(.s)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでra88.batを起動します。

ra88\_ファイル名□

\_はスペースの入力を表します。

□はリターンキーの入力を表します。

ファイル名の拡張子は入力しないでください。".s"の拡張子に固定されています。

例 A:¥USER>a:¥EPSON¥ra88 sample□

AドライブのサブディレクトリUSER内に作成されている構造化アセンブリソースファイル"sample.s"を入力し、リロケータブルアセンブルを実行して以下のファイルを入力ファイルと同じディレクトリに生成します。

sample.ms, sample.o, sample.l, sample.x, sample.e

ra88へのPATHが設定されている場合は、ra88の前のパス指定は不要です。

入出力ファイルや表示されるメッセージについては、"A.2.3.9 アセンブルの実行例"を参照してください。

## ra88.batのカスタマイズ

## &lt;ra88.bat実行パラメータのカスタマイズ&gt;

ra88.batはプログラム実行を制御するため、実行パラメータのカスタマイズ領域を持っています。デフォルトでは一般的なパラメータが仮に記述されていますが、お客様の開発方法に合わせカスタマイズを行った上でご使用ください。

## 1. ROM容量の設定(CODEセクションのサイズチェック)

set rom = 32768: エラーチェックするCODEセクションのROM容量をバイト単位で指定します。  
(デフォルト 32768=32Kバイト)

## 2. RAM容量の設定(DATAセクションのサイズチェック)

set ram = 65536: エラーチェックするDATAセクションのRAM容量をバイト単位で指定します。  
(デフォルト 65536=64Kバイト)

注: 基本的にこれらの設定パラメータに関してはエラーチェックを行っていませんので、指定以外のパラメータは記述しないでください。

## &lt;ra88.bat実行コマンドのカスタマイズ&gt;

ra88.batは実行にあたり、以下のコマンドラインを用いています。もし、デフォルトで設定している以外のフラグを用いたい場合、これらのコマンドラインをカスタマイズしてください。

sap88

%drv%sap88 -o %1.ms %1.s

asm88

%drv%asm88 -ROM %rom% -RAM %ram% %1.ms

%drv%はra88.batの実行コマンド検索パスです。そのため、定義しているSET文とも合わせ変更不可です。%1はコマンドラインから入力したファイル名です。

以下にra88.batのプログラムソースリスト、およびra88.batで使用しているメッセージ一覧を示しますので、カスタマイズの際参考にしてください。

## ra88.batのプログラムソースリスト

```

echo off
rem *****
rem *      E0C88 Family Auto Relocatable Assemble Execution Utility
rem *
rem *                               (Ver. X.XX)
rem *
rem *                               Copyright(C) SEIKO EPSON CORP. 1993-1996
rem *****
rem * customized parameter information
rem *   rom=*       * : rom capacity(32768 max.)
rem *   ram=*       * : ram capacity(65536 max.)
rem *****
rem ***** customized parameter area (default) *****
rem *   caution : customized parameters value do not check, therefore
rem *               please be carefully when you set
rem *****
set rom=32768          バイト単位によるROM容量制限の設定
set ram=65536          バイト単位によるRAM容量制限の設定

rem ***** command searching path *****
rem set drv=a:¥

rem *****
rem *   main program
rem *       if you want to use another option(s), please append
rem *       option flag(s) at command line.
rem *****
:start
    echo E0C88 Family Auto Relocatable Assemble Execution Utility Ver. X.XX
    echo Copyright (C) SEIKO EPSON CORP. 1993-1996

    if "%1"==" " goto usage

:error_chk
    if not exist %drv%nul goto exit04
    if not exist %1.s goto exit05
    if not exist %drv%sap88.exe goto exit06
    if not exist %drv%asm88.exe goto exit07

rem (sap88)
:sap88
%drv%sap88 -o %1.ms %1.s          sap88の起動コマンド
    if errorlevel 1 goto exit01

rem (asm88)
:asm88
%drv%asm88 -ROM %rom% -RAM %ram% %1.ms          asm88の起動コマンド
    if errorlevel 1 goto exit02

:usage
echo usage : ra88 needs [input file_name]
    goto skip

:exit01
echo Error stop at %drv%sap88.exe
    goto skip

:exit02
echo Error stop at %drv%asm88.exe
    goto skip

:exit03
echo Cannot find %drv% installed E0C88 dev. tools directory
    goto skip

:exit04

```

お客様のカスタマイズ領域  
注: 基本的に設定パラメータに関してはエラーチェックを行っていないので、指定以外のパラメータは記述しないでください。

ra88.bat実行コマンド検索パスです。デフォルトはルートディレクトリに設定されていますので、必要に応じてカスタマイズしてください。

```

echo Cannot find input file
        goto skip

:exit05
echo Cannot find %drv%sap88.exe
        goto skip

:exit06
echo Cannot find %drv%asm88.exe
        goto skip

:end
echo ra88.bat utility has been successfully executed.
:skip
set rom=
set ram=
set drv=

```

## メッセージ一覧

### 1. 起動メッセージ

```

E0C88 Family Auto Relocatable Assemble Execution Utility Ver. X.XX
Copyright (C) SEIKO EPSON CORP. 1993-1996

```

### 2. 正常終了メッセージ

```

ra88.bat utility has been successfully executed.

```

### 3. エラーメッセージ

エラーメッセージ	意 味
usage : ra88 needs [input file_name]	usage出力
Error stop at [drive and path name] sap88.exe	sap88でエラーが発生しました。
Error stop at [drive and path name] asm88.exe	asm88でエラーが発生しました。
Cannot find [drive and path name] installed E0C88 dev. tools directory	S1C88 Familyのソフトウェアツールをインストールしている [ドライブおよびパス]が見つかりません。
Cannot find input file	ra88.batの入力ファイル(s)が見つかりません。
Cannot find [drive and path name] sap88.exe	sap88が見つかりません。
Cannot find [drive and path name] asm88.exe	asm88が見つかりません。

注: エラーが発生した場合、以降の処理は中止されます。

#### <バッチファイル使用上の注意事項>

- (1) バッチ処理中に表示されるメッセージの一部は、MS-DOS/PC-DOSのバッチ処理機能およびコマンドを用いて自動的に発生しています。そのため、エラー発生時はMS-DOS/PC-DOSの管理下におかれる場合があり、それによってバッチ処理が強制的に中断されることがあります。
- (2) エラー発生の際、以降の処理が自動的に継続しないよう管理をしておりますが、前記(1)の理由により管理できない場合があります。
- (3) ra88.batおよび後述のlk88.batではS1C88 Familyのツール以外にMS-DOS/PC-DOSのCOPYコマンドを使用しております。  
このため、バッチファイルを実行する際にはPATHの設定等によりCOPYコマンドが実行可能な状態にしておいてください。
- (4) バッチファイルの実行パラメータ(お客さまのカスタマイズ領域)は、基本的に設定パラメータのエラーチェックを行っていませんので、指定以外のパラメータは記述しないでください。
- (5) バッチファイル実行のためにMS-DOS/PC-DOSの環境変数を使用しますので、環境変数のサイズはCONFIG.SYSで、できるかぎり大きくしておいてください。

### A.2.3.5 リロケータブルオブジェクトファイル

リロケータブルオブジェクトファイルは、asm88のリロケータブルアセンブルにより生成されるバイナリファイルです。

生成されるファイル名は特に指定した(-oフラグ)場合を除き、asm88に入力したファイル名と同じで拡張子が".o"となります。

このファイルは、オブジェクト(機械語)コード以外に、リンカによって再配置を行うために必要なヘッダ情報やシンボルテーブル等で構成されています。

### A.2.3.6 アセンブリリストファイル

アセンブリリストファイルは、asm88に入力されたアセンブリソースファイルにオブジェクトコード(16進数)やコードのアドレス(16進数)等を付加したASCIIファイルで、asm88のアセンブルにより生成されます。

また、各ページの先頭にはファイル名や作成年月日を示すヘッダが出力されます。

生成されるファイル名は特に指定した(-oフラグ)場合を除き、asm88に入力したファイル名と同じで拡張子が".l"となります。

アセンブリリストファイルには、次の内容が出力されます。

LINE ..... 先頭から連続する行番号です。  
 ADDRESS ..... オブジェクトコードのターゲットアドレスのことです。  
 CODE ..... 同じ行のソースステートメントに対応したオブジェクト(機械語)コードです。  
 SOURCE STATEMENT ..... asm88に入力したアセンブリソースファイルです。

リロケータブルアセンブルを行った場合、コードのアドレスはCODEセクションの先頭からの相対アドレスとなっています。同様に、データ領域のアドレスはDATAセクションの先頭からの相対アドレスとなっています。

また、エラーが発生した場合、発生した行の頭に"\*"が付いて出力されます。

アセンブリリストファイルに関しては、以下のasm88の擬似命令と起動時のフラグ指定により出力の制御が行えます。

#### 出力リスト制御擬似命令

擬似命令	説 明
LINENO	行番号(LINE)を任意の値に変更します。
SUBTITLE	カラム説明の次の行に任意に設定したサブタイトル文字列を挿入します。
SKIP	ASCIIやDB、DW等のデータ設定によりコード(CODE)が1行(5バイト)を越える場合、越えた部分の出力を禁止します。(デフォルト設定)
NOSKIP	SKIPの設定を解除し、全てのコードを出力させます。
LIST	NOLISTの設定を解除し、以降の行をリスト出力させます。(デフォルト設定)
NOLIST	この擬似命令以降の行をリスト出力しないように設定します。
EIECT	強制的に改ページを行います。

擬似命令の詳細についてはAppendix Bを参照してください。

#### 起動フラグ

フラグの詳細についてはAppendix Cを参照してください。

フラグ	説 明
-1	アセンブリリストファイルの生成を禁止します。

### A.2.3.7 クロスリファレンスリスト

クロスリファレンスリストファイルは、モジュール内で定義、あるいは参照しているシンボル情報の一覧を内容とするASCIIファイルで、asm88のアセンブルにより生成されます。

生成されるファイル名は特に指定した(-oフラグ)場合を除き、asm88に入力したファイル名と同じで拡張子が".x"となります。

クロスリファレンスリストファイルの出力フォーマットは以下のようになっています。

R	SYMBOL	A	VALUE	LINE No.	INFORMATION
---	--------	---	-------	----------	-------------

R            参照定義  
              G: グローバル  
              L: ローカル

SYMBOL   シンボル名 (最大15文字)

A            属性  
              L: ラベル  
              C: 定数  
              V: 変数  
              U: モジュール内で未定義

VALUE     シンボル値 (6桁16進数表記)

LINE No. INFORMATION

シンボルが定義、あるいは参照されている行番号の一覧で、次のように出力されます。

lineno\* lineno lineno . . . lineno

lineno\*: 当該シンボルが定義されている行番号

lineno: 当該シンボルが参照されている行番号

LINE No. INFORMATIONは最大12個の行番号で構成されます。

各ページの先頭には、例のようなページヘッダが出力されます。

なお、ラベルのうち、ニューメリックラベルは一時的なラベルで、通常のラベルで囲まれた範囲外であれば同一の名前が何度でも使用できるため、クロスリファレンスリスト上には出力されません。(ニューメリックラベルについてはAppendix Bを参照してください。)

クロスリファレンスリストファイルはasm88の-xフラグによって出力を禁止することもできます。

—— クロスリファレンスリスト例 ——

CROSS REFERENCE TABLE OF asm88   error.x   1993-06-07   17:28   PAGE   1

L   delay	L   000100H	5*	14	15
L   delay_00	L   000103H	7*	9	
L   delay_3times	L   000107H	13*		

### A.2.3.8 エラーリスト

asm88のアセンブル時に発生したエラーは、エラーリストファイルとして出力されます。生成されるファイル名は特に指定した(-oフラグ)場合を除き、asm88に入力したファイル名と同じで拡張子が".e"となります。

エラーリストの出力フォーマットは以下のようになっています。

SOURCE FILE	LINE No.	ERROR LEVEL	ERROR MESSAGE
-------------	----------	-------------	---------------

SOURCE FILE	ソースファイル名
LINE No.	エラーが発生した行番号
ERROR LEVEL	エラーのレベル
Warning	警告レベルで出力オブジェクトには影響を与えません。
Severe	一般のエラーレベルです。出力オブジェクトは無効となります。
Fatal	致命的なレベルのエラーで、アセンブル処理が中断されます。致命的エラーはCRT上に表示されるだけで、エラーリストファイルには出力されません。

ERROR MESSAGE エラーの内容

asm88のエラーメッセージについてはAppendix Cを参照してください。

#### エラーリスト例

```
error.s 16: Severe: delay not defined
```

エラーが発生しなかった場合、エラーリストファイルには何も出力されません。

### A.2.3.9 アセンブルの実行例

ここではアセンブルの実行例を示します。

リロケータブルアセンブラ一括処理ra88.bat実行時のメッセージ

```
A:¥USER>a:¥EPSON¥ra88 sample□
A:¥USER>echo off
E0C88 Family Auto Relocatable Assemble Execution Utility Ver. X.XX
Copyright (C) SEIKO EPSON CORP. 1993-1996
sap88 Structured Assembler Preprocessor Version X.XX

Copyright (c) 1993 by Advanced Data Controls, Corp.
Licenced to SEIKO EPSON CORP.
asm88 Cross Assembler Version X.XX

Copyright (c) 1993 by Advanced Data Controls, Corp.
Licenced to SEIKO EPSON CORP.

9 Symbol(s) Used

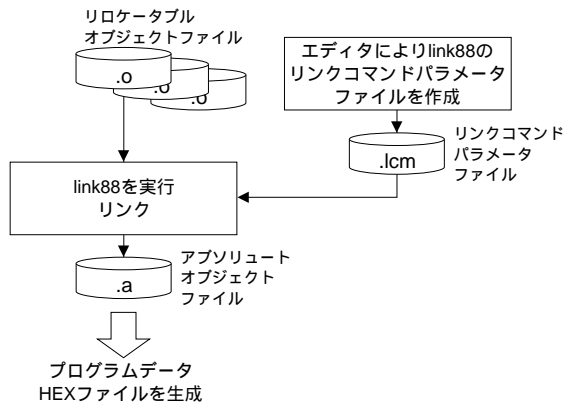
0 Warning Error(s)
0 Severe Error(s)
ra88.bat utility has been successfully executed.
A:¥USER>
```



## A.2.4 リンク

ここではリロケートブルモジュールのリンクに関する説明を行います。

使用ソフト ..... link88



図A.2.4.1 リンク処理のフローチャート

### A.2.4.1 モジュールのリンク

asm88のリロケートブルアセンブルにより生成された各モジュールのオブジェクトコードは、実際のROMのどの部分に配置されるか確定していません。配置されるアドレスは、各モジュールをどういった並び方で連結するかによって決定します。この処理を行うツールがリンカlink88です。

リンクが成功すると、それまで未解決だった外部参照ラベルに対する相対アドレスなども決定され、全てのモジュールが1つにまとめられたアブソリュートオブジェクトファイル(.a)が生成されます。

このアブソリュートオブジェクトファイルをA.2.5項で説明するバイナリ/HEXコンバータhex88で処理することにより、プログラムのマスクデータ生成やハードウェアデバッグに使用するためのプログラムデータHEXファイルが生成されます。

### A.2.4.2 セクション管理

S1C88 Familyは24ビット幅のアドレス空間(最大16Mバイト)を持ち、最上位の8ビットをコードバンクレジスタ(CB)やエクスパンドページレジスタ(EP、XP、YP)などのレジスタで管理することによって、アドレス空間を32Kバイトのバンク(コード部)または64Kバイトのページ(データ部)単位に分割し、その範囲内でのアクセスパフォーマンスの向上が図られています。これらのレジスタの内容を書き換えることによって、任意のバンクから任意のバンクまたはページをアクセスすることも可能で、大きなプログラムやデータベースなども容易に管理できるようになっています。

ただし、バンクやページの変更はプログラムによって行う必要があり、プログラムの実行に伴って自動的に変更されるものではありません。したがって、16Mバイトのアドレス空間をリニアに記述するようなプログラムを作成することはできません。

このことは、複数のモジュールを単純にリンクすることができないことを意味します。

link88では、これを解決するため、および任意のモジュールを任意のアドレスに配置するためにマルチセクション方式を採用しています。

これはセクションというブロック単位にアドレス指定を可能として配置を行う方式です。

セクションは配置先をROMとするCODEセクションとデータメモリのDATAセクションに分類されます。そして上記のバンク/ページの問題を解決するため、1つのCODEセクションの大きさは最大32Kバイトに、1つのDATAセクションの大きさは最大64Kバイトに制限されます。ただし、このサイズはバンクやページの境界を越えて配置しないという前提のもので、バンクやページの途中から配置する場合は、そこからの残りのサイズに制限されます。

セクションの手法によって希望するマルチセクションのオブジェクトコードを作成するためには、セクションを定義し、さらにそのセクションの配置に関するアドレス情報を与えることによってアドレスの割り付けを行う必要があります。

セクションの定義にはリンカの二次フラグ(セクションに関する定義を行うフラグ)+codeと+dataを使用し、配置アドレスの割り付けには-pフラグを使用します。

なお、1回のリンクで最大255個のセクション定義が行えます。

### <セクション定義の例>

セクション定義を簡単な例で見ていくことにします。

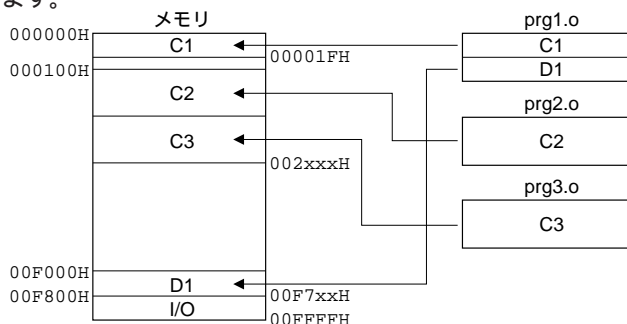
まず、図A.2.4.2.1に示すようなメモリマッピングを実現する方法について説明します。

C1とD1を記述した"prg1.s"、C2を記述した"prg2.s"、C3を記述した"prg3.s"のアセンブルを行ってそれぞれのリロケータブルオブジェクトファイル"prg1.o"、"prg2.o"、"prg3.o"が生成されているものとします。

この場合、CはCODEセクション、DはDATAセクションを意味します。

link88へのフラグ指定は入力リダイレクトによって行うことができます。

そして下記のようなフラグ指定を行って、セクションの定義とアドレスの割り付けを行うリンクコマンドパラメータファイル(filename.lcm)を作成し、link88 < filename.lcmを実行すると図A.2.4.2.1に示すメモリマッピングが実現できます。



図A.2.4.2.1 メモリマップ例

link88に渡すファイルの内容 (link88< filename.lcm)

```
-o prg.a          ... (1)
+code -p0x000000 ... (2)
+data -p0x00f000 ... (3)
prg1.o           ... (4)
+code -p0x000100 ... (5)
prg2.o prg3.o    ... (6)
```

(1) -oフラグで出力するアブソリュートオブジェクトファイル名を指定します。

(2) 物理アドレス000000Hから始まるCODEセクションを定義します。

(3) 物理アドレス00F000Hから始まるDATAセクションを定義します。

(4) 上記(2)と(3)で定義したセクションに"prg1.o"の内容を割り付けます。

この場合、"prg1.o"の中のCODEセクションの内容C1が(2)で定義したCODEセクションの先頭から配置され、DATAセクションの内容D1が(3)で定義したDATAセクションの先頭から配置されます。

(5) 物理アドレス000100Hから始まるCODEセクションを定義します。このCODEセクションは(2)で定義したCODEセクションとは別のもので、新たなセクション定義が行われたこの時点で(2)のCODEセクションは完結します。

(6) (5)で新たに定義したCODEセクションの先頭から"prg2.o"のCODEセクションの内容C2、"prg3.o"のCODEセクションの内容C3の順に連続して配置します。

この例では"prg2.o"と"prg3.o"はDATAセクションを持ちませんが、もしDATAセクションがあれば(3)で定義したDATAセクションのD1に続くアドレスから配置されます。

以上のようにこの例では3つのセクションが定義され、リンクが行われます。リンクが成功すると"prg.a"という名称のアブソリュートオブジェクトファイルが生成されます。

このように定義したセクションには容量の制限以内であれば、複数のモジュールを割り付けることができます。また、1つのバンク内にも複数のセクションを配置することができます。

### <セクションの配置アドレスとりロケーション>

前記の例に示したように-pフラグは直前に定義したセクションの物理的な開始アドレスを決定します。たとえば、あるセクションに対して次のような設定が行われているとします。

```
-p 0x10000
```

このセクションの開始アドレスは物理的に10000H番地となり、CODEセクションの場合はバンク2の先頭、DATAセクションの場合はページ1の先頭が指定されます。

このセクションの先頭から1234H番地のオフセットに位置するようなシンボルが定義され、そのシンボルを用いてそのアドレスが参照されるような場合、そのシンボルに対しては次のようなりロケーション(アドレス情報の再配置)が行われます。

(1) データメモリとして扱う場合 (シンボル名を仮に"SYMBOL"として説明します。)

オペランド		リロケート値
#SYMBOL	→	#1234H
[SYMBOL]	→	[1234H]
#POD SYMBOL	→	01H
#LOD SYMBOL	→	1234H
#HIGH SYMBOL	→	12H
#LOW SYMBOL	→	34H
[BR:LOW SYMBOL]	→	[BR:34H]

(2) プログラムメモリとして扱う場合 (シンボル名を仮に"LABEL"として説明します。)

オペランド		リロケート値
#BOC LABEL	→	02H
#LOC LABEL	→	9234H

"JRL LABEL"のようなPC相対分岐命令の場合は、分岐命令が配置されるアドレスにしたがった相対値が計算されて設定されます。

上記の例ではセクションの開始アドレスをバンクあるいはページの先頭に指定した訳ですが、次のようにバンクあるいはページの途中から始まるような指定も行えます。

```
-p 0x15000
```

この場合、開始アドレスは物理的に15000H番地となり、バンクあるいはページの先頭から5000H番地のオフセットを持つこととなります。link88は物理アドレスから各シンボルのリロケーションを行いますので、このようなオフセットも正しく処理されます。

再配置後のシンボル情報は全てアプソリュートオブジェクトファイルに記録され、シンボル情報生成ユーティリティrel88によってその内容のリストを生成することができます。rel88については"A.2.6.1 シンボル情報の生成(rel88)"を参照してください。

### A.2.4.3 モジュールのアロケーション情報

前項のセクション定義例に示したように、セクション定義やファイルを指定するコマンドラインは入力リダイレクトによってlink88に渡すことができます。

例のようにモジュール数も少なく単純なリンクであれば、例と同様のファイルを作成しlink88に直接入力しても問題になりません。

モジュールが増えるような場合は、メモリ効率なども考慮する必要が出てきます。

1つのCODEセクションは最大32Kバイト、DATAセクションは最大64Kバイトの容量的な制限があり、それに納まるように各モジュールを配置することになります。このときに、各セクションを構成するモジュールの組み合わせを考慮しないと、メモリの未使用領域が増えて無駄なメモリの拡張が必要になることが考えられます。

### A.2.4.4 link88の起動方法

#### <link88の操作方法>

- (1) リンクするリロケートブルオブジェクトファイル(.o)とエディタで作成したlink88のコマンドラインを記述したリンクコマンドパラメータファイル(.lcm)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでlink88を起動します。

link88 \_<\_リンクコマンドパラメータファイル名□

\_はスペースの入力を表します。

□はリターンキーの入力を表します。

リンクコマンドパラメータファイルは、入力リダイレクトによらずコマンドラインに直接入力することも可能ですが、実用的ではありませんのでここでの説明は省略します。フォーマットについてはAppendix Bに掲載されていますので、そちらを参照してください。

また、コマンドラインを構成するフラグについてもここでは説明を省略します。フラグの詳細についてはAppendix Bを参照してください。

例 リンクコマンドパラメータファイル(.lcm)によりリンクを実行

```
A:¥USER>a:¥EPSON¥link88 < sample.lcm□
```

AドライブのサブディレクトリUSER内に作成されているリンクコマンドパラメータファイル"sample.lcm"を入力リダイレクトとしてlink88を起動し、リンクの処理を実行させます。

リンクコマンドパラメータファイル内で指定した名称のアブソリュートオブジェクトファイルが、入力ファイルと同じディレクトリに生成されます。

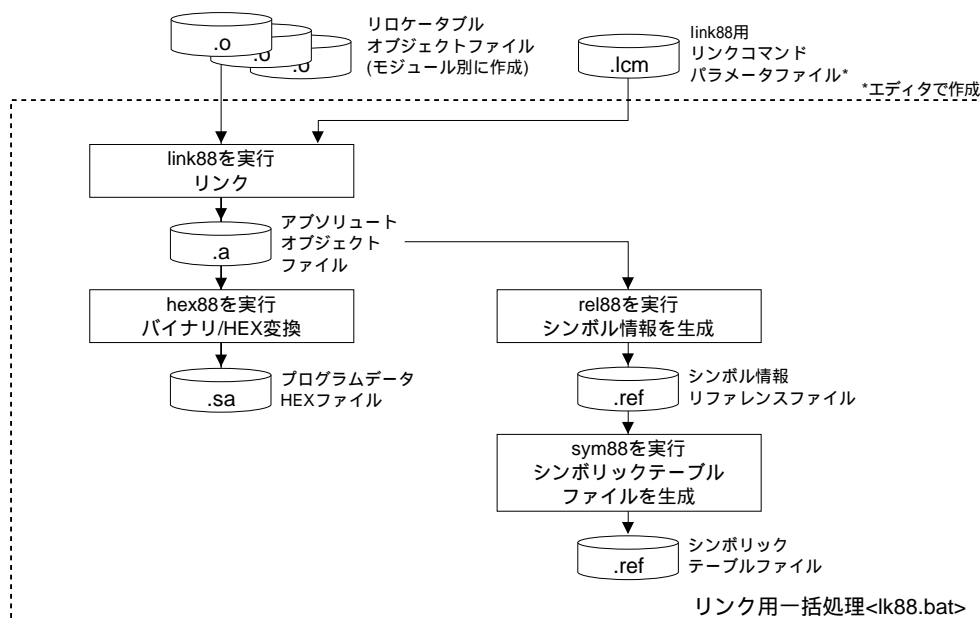
link88へのPATHが設定されている場合は、link88の前のパス指定は不要です。

リンクコマンドパラメータファイルの内容については、A.2.4.2項を参照してください。

### A.2.4.5 リンクの一括処理(lk88.bat)

アセンブルと同様に、本パッケージにはリンク用の一括処理バッチファイルlk88.batが収められています。このバッチファイルは、リンクからプログラムデータHEXファイル生成までを一括処理できるように作成されています。(なお、リンク以降の処理の詳細については後述します。)

lk88.batの処理フローを図A.2.4.5.1に示します。



図A.2.4.5.1 lk88.batの処理フロー

## &lt;処理概要&gt;

link88に直接入力するリンクコマンドパラメータファイルを入力し、リンク処理を実行します。  
link88によってアブソリュートオブジェクトファイルが生成されると、次にシンボル情報生成ユーティリティrel88を使用して再配置後のシンボリックテーブル情報ファイルを生成します。その後、ICEにてシンボリックデバッグをするために必要なシンボリックテーブルファイルをsym88を実行し、生成します。そして、アブソリュートオブジェクトファイルからバイナリ/HEXコンバータhex88によってプログラムデータHEXファイルを生成します。

## &lt;入出力ファイル&gt;

## 入力ファイル

1. リンクコマンドパラメータファイル:

file\_name.lcm

link88のコマンドパラメータファイルです。S1C88のメモリ空間上にリロケータブルオブジェクトを再配置するための情報を記述します。

2. リロケータブルオブジェクトファイル:

file\_name.o

クロスアセンブラでリロケータブルアセンブルすることにより出力される、再配置可能な機械語ファイルです。

## 出力ファイル

1. アブソリュートオブジェクトファイル:

file\_name.a

リンカにより生成されたマルチセクションオブジェクトファイルです。

2. プログラムデータHEXファイル:

file\_name.sa

アブソリュートオブジェクトファイルをバイナリ/HEXコンバータにより変換したモトローラS2フォーマットのASCIIレコードファイルです。

3. シンボル情報リファレンスファイル:

file\_name.ref

物理アドレスに再配置されたアブソリュートオブジェクトファイルのシンボル情報リファレンスファイルです。

4. シンボリックテーブルファイル:

file\_name.sy

シンボリックデバッグに必要な情報としてシンボル名、アドレスの一覧を示したファイルです。

## &lt;操作方法&gt;

- (1) リンクするリロケータブルオブジェクトファイル(.o)が存在するディレクトリをカレントドライブにします。link88に渡すコマンドラインパラメータファイルも同じディレクトリに用意してください。
- (2) 次のフォーマットでlk88.batを起動します。

**lk88** □

□はリターンキーの入力を表します。

例 A: ¥USER>a: ¥EPSON¥lk88 □

AドライブのサブディレクトリUSER内に作成されているリンクコマンドパラメータファイル"sample.lcm"からバッチ処理を実行させます。

バッチ処理によって、アブソリュートオブジェクトファイル(.a)、シンボル情報リファレンスファイル(.ref)、プログラムデータHEXファイル(.sa)、シンボリックテーブルファイル(.sy)が入力ファイルと同じディレクトリに生成されます。

lk88へのPATHが設定されている場合は、lk88の前のパス指定は不要です。

## lk88.batのカスタマイズ

## &lt;lk88.bat実行パラメータのカスタマイズ&gt;

lk88.batはプログラム実行を制御するため、実行パラメータのカスタマイズ領域を持っています。デフォルトでは一般的なパラメータが記述されています。しかし、ファイル名などはお客様のアプリケーションに依存しますので、お客様の開発方法に合わせ必ずカスタマイズを行った上でご使用ください。

## 1. 入力するパラメータファイル名

set parfn = file\_name :link88に入力するリンクコマンドパラメータ(.lcm)名

## 2. 出力するファイル名

set outfn = file\_name :オブジェクトファイルおよびプログラムデータHEXファイルの  
ファイル名

## 3. シンボル情報生成ユーティリティ(rel88)の使用有無

set rel88 = y :rel88を使用する(デフォルト)  
シンボル情報リファレンスファイル(.ref)が生成されます。  
= n :rel88を使用しない

4. シンボル情報生成ユーティリティ(rel88)の  
+secフラグ(個別セクション情報)の使用有無

set secf = y :rel88に+secフラグを付ける(デフォルト)  
= n :rel88に+secフラグを付けない

注: rel88を使用しないとした場合、このパラメータは無視されます。

注: 基本的にこれらの設定パラメータに関してはエラーチェックを行っていませんので、指定以外のパラメータは記述しないでください。

## &lt;lk88.bat実行コマンドのカスタマイズ&gt;

lk88.batは実行にあたり、以下のコマンドラインを用いています。もし、デフォルトで設定している以外のフラグを用いたい場合、これらのコマンドラインをカスタマイズしてください。

```
link88
%drv%link88<%parfn%.lcm

rel88(+secフラグ使用時)
%drv%rel88 -v +sec %outfn%.a>%outfn%.ref

rel88(+secフラグ未使用時)
%drv%rel88 -v %outfn%.a>%outfn%.ref

hex88
%drv%hex88 -o %outfn%.sa %outfn%.a

sym88
%drv%sym88 %outfn%.ref
```

%drv%はlk88.batの実行コマンド検索パスです。そのため、定義しているSET文とも合わせ変更不可です。

カスタマイズパラメータoutfnは、リンクコマンドパラメータ(.lcm)に記述した名称と同じものにしてください。

以下にlk88.batのプログラムソースリスト、およびlk88.batで使用しているメッセージ一覧を示しますので、カスタマイズの際参考にしてください。



## lk88.batのプログラムソースリスト

```

echo off
rem *****
rem *      E0C88 Family Auto Link Execution Utility
rem *
rem *                               (Ver. X.XX)
rem *                               Copyright(C) SEIKO EPSON CORP. 1993-1996
rem *****
rem * customized parameter information
rem *   parfn=           : input parameter file_name
rem *                     (file_name_lcm) for link88.exe       i.e. c8316xxx.lcm
rem *   outfn=           : output file_name which is written
rem *                     in the input parameter file_name   i.e. c8316xxx
rem *   rel=y   y : use rel88 for absolute symbol map generation
rem *           =n  n : do not use rel88
rem *
rem *   secf=y   y : show physical address and module size with absolute
rem *             symbolic table after link procedure
rem *           =n  n : do not show physical address and module size just
rem *             symbolic table after link procedure
rem *****
rem ***** customized parameter area (default) *****
rem * caution : customized parameters value do not check, therefore
rem *           please be carefully when you set
rem *****
set parfn=sample      入力するリンクコマンドパラメータファイル名
set outfn=sample      出力するファイル名
set rel=y             rel88の使用有無
set secf=y            rel88の+secフラグの使用有無

rem ***** command searching path *****
rem set drv=a:¥

rem *****
rem *      main program
rem *
rem *      if you want to use another option(s), please append
rem *      option flag(s) at command line
rem *****
:start
    echo E088 Family Auto Link Execution Utility Ver. X.XX
    echo Copyright (C) SEIKO EPSON CORP. 1993-1996

:error_chk
    if not exist %drv%nul goto exit05
    if not exist %parfn%.lcm goto exit06
    :chk00
    if not exist %drv%link88.exe goto exit07
    if not exist %drv%rel88.exe goto exit08
    if not exist %drv%hex88.exe goto exit09
    if not exist %drv%sym88.exe goto exit10

:link88
%drv%link88<%parfn%.lcm                                link88の起動コマンド
    if errorlevel 1 goto exit01

rem (rel88 no sec option)
:rel88_01
    if "%rel%"=="n" goto hex88
    if "%secf%"=="y" goto rel88_02
%drv%rel88 -v %outfn%.a>%outfn%.ref                    rel88の起動コマンド(+secフラグなし)
    if errorlevel 1 goto exit02

```

お客様のカスタマイズ領域  
注: 基本的に設定パラメータに関してはエラーチェックを行っていないので、指定以外のパラメータは記述しないでください。

lk88.bat実行コマンド検索パスです。デフォルトはルートディレクトリに設定されていますので、必要に応じてカスタマイズしてください。

```

                                goto hex88
rem (rel88 with sec option)
:rel88_02
%drv%rel88 -v +sec %outfn%.a>%outfn%.ref           rel88の起動コマンド(+secフラグ付き)
                                if errorlevel 1 goto exit02

:hex88
%drv%hex88 -o %outfn%.sa %outfn%.a                 hex88の起動コマンド
                                if errorlevel 1 goto exit03

:sym88
%drv%sym88 %outfn%.ref                             sym88の起動コマンド
                                if errorlevel 1 goto exit04
                                goto end

:exit01
echo Error stop at %drv%link88.exe
                                goto skip
:exit02
echo Error stop at %drv%rel88.exe
                                goto skip
:exit03
echo Error stop at %drv%hex88.exe
                                goto skip
:exit04
echo Error stop at %drv%sym88.exe
                                goto skip
:exit05
echo Cannot find %drv% installed E0C88 dev. tools directory
                                goto skip
:exit06
echo Cannot find %parfn% input parameter file
                                goto skip
:exit07
echo Cannot find %drv%link88.exe
                                goto skip
:exit08
echo Cannot find %drv%rel88.exe
                                goto skip
:exit09
echo Cannot find %drv%hex88.exe
                                goto skip
:exit10
echo Cannot find %drv%sym88.exe

:end
                                echo lk88.bat utility has been successfully executed.
:skip
set parfn=
set outfn=
set rel=
set secf=
set drv=

```

## メッセージ一覧

### 1. 起動メッセージ

E0C88 Family Auto Link Execution Utility Ver. X.XX  
 Copyright (C) SEIKO EPSON CORP. 1993-1996



## 2. 正常終了メッセージ

```
lk88.bat utility has been successfully executed.
```

## 3. エラーメッセージ

エラーメッセージ	意 味
Error stop at [drive and path name] link88.exe	link88でエラーが発生しました。
Error stop at [drive and path name] rel88.exe	rel88でエラーが発生しました。
Error stop at [drive and path name] hex88.exe	hex88でエラーが発生しました。
Error stop at [drive and path name] sym88.exe	sym88でエラーが発生しました。
Cannot find [drive and path name] installed E0C88 dev. tools directory	S1C88 Familyのソフトウェアツールをインストールしている [ドライブおよびパス]が見つかりません。
Cannot find [file_name] input parameter file	lk88.batで使用する入力パラメータファイル(.lcm)が見つかりません。
Cannot find [drive and path name] link88.exe	link88が見つかりません。
Cannot find [drive and path name] rel88.exe	rel88が見つかりません。
Cannot find [drive and path name] hex88.exe	hex88が見つかりません。
Cannot find [drive and path name] sym88.exe	sym88が見つかりません。

注: エラーが発生した場合、以降の処理は中止されます。

## &lt;バッチファイル使用上の注意事項&gt;

- (1) バッチ処理中に表示されるメッセージの一部は、MS-DOS/PC-DOSのバッチ処理機能およびコマンドを用いて自動的に発生しています。そのため、エラー発生時はMS-DOS/PC-DOSの管理下におかれる場合があり、それによってバッチ処理が強制的に中断されることがあります。
- (2) エラー発生の際、以降の処理が自動的に継続しないよう管理をしておりますが、前記(1)の理由により管理できない場合があります。
- (3) バッチファイルの実行パラメータ(お客さまのカスタマイズ領域)は、基本的に設定パラメータのエラーチェックを行っていませんので、指定以外のパラメータは記述しないでください。
- (4) バッチファイル実行のためにMS-DOS/PC-DOSの環境変数を使用しますので、環境変数のサイズはCONFIG.SYSで、できるかぎり大きくとっておいてください。

## A.2.4.6 アブソリュートオブジェクトファイル

アブソリュートオブジェクトファイルは、link88により生成されるバイナリファイルです。生成されるファイル名は-oフラグで指定した内容となります。ファイルのフォーマットはマルチセクションオブジェクトフォーマットです。このファイルは、オブジェクト(機械語)コード以外に各種再配置情報等で構成されています。

## A.2.4.7 リンクの実行例

ここでは、lk88のリンクの実行例を示します。

```
A:¥USER>a:¥EPSON¥lk88
```

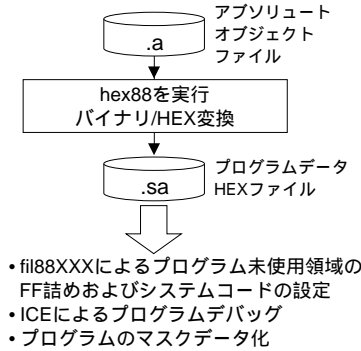
```
A:¥USER>echo off
E0C88 Family Auto Link Execution Utility Ver. X.XX
Copyright (C) SEIKO EPSON CORP. 1993-1996
link88 Linker Version X.XX
```

```
Copyright (c) 1993 by Advanced Data Controls, Corp.
Licenced to SEIKO EPSON CORP.
lk88.bat utility has been successfully executed.
A:¥USER>
```

A.2.5 プログラムデータHEXファイルの生成

ここでは、プログラムデータHEXファイルとバイナリ/HEXコンバータhex88によるその生成方法について説明します。

使用ソフト ..... **hex88**



図A.2.5.1 プログラムデータHEXファイルの生成フロー

A.2.5.1 プログラムデータHEXファイル

プログラムデータHEXファイルは、バイナリオブジェクトコードがHEXデータに変換されたASCIIファイルです。

HEXファイルの形式としては、S1C88 Familyが16Mバイトのアドレス空間を持つため、通常モトローラS2フォーマットが用いられています。(A.2.5.3項参照)

このファイルはプログラムデータのマスク化、ICEによるプログラムデバッグに必要となります。リロケータブルアセンブルによるモジュール別開発の場合は、リンカによって生成したアブソリュートオブジェクトファイルを、バイナリ/HEXコンバータhex88によってHEXデータに変換し、プログラムデータHEXファイルを生成します。

こうして生成されたプログラムデータHEXファイルは、機種別のソフトウェアツールfil88XXXで、内蔵ROM未使用領域のFF詰めおよび機種ごとのシステムコードを設定します。

A.2.5.2 hex88によるプログラムデータHEXファイルの生成方法

以下にhex88によるプログラムデータHEXファイルの生成方法を示します。

- (1) アブソリュートオブジェクトファイル(.a)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでhex88を起動します。

```
hex88_ [フラグ]_ファイル名 [ ]
_ はスペースの入力を表します。
[ ] はリターンキーの入力を表します。
```

リンカー一括処理(lk88.bat)で使用しているフラグを次に示します。

フラグ	説 明
-o <ファイル名>	出力ファイル名を指定します。出力ファイル名の拡張子は".sa"にしてください。 このフラグを省略した場合、標準出力に出力されます。

例 sample.aのプログラムデータHEXファイルを作成

```
A:¥USER>a:¥EPSON¥hex88 -o sample.sa sample.a [ ]

AドライブのサブディレクトリUSER内に作成されているアブソリュートオブジェクトファイル"sample.a"
を入力し、HEXデータに変換後、"sample.sa"を入力ファイルと同じディレクトリに生成します。
hex88へのPATHが設定されている場合は、hex88の前のパス指定は不要です。
```

なお、バッチファイルにより、リンク処理から続けてhex88を実行させることができます。このバッチ処理については、"A.2.4.5 リンクの一括処理(lk88.bat)"を参照してください。

### A.2.5.3 モトローラS2フォーマットについて

モトローラS2フォーマットのHEXファイルは、次のようなフィールドで構成されたレコードの集合です。

&lt;S FIELD&gt;&lt;COUNT&gt;&lt;ADDR&gt;&lt;DATA BYTES&gt;&lt;CHECKSUM&gt;

全ての情報は16進数字のペアとして表示され、それぞれのペアは1バイトの数値を表します。

<S FIELD>      その行のフォーマットを示します。このフィールドには"S2" が入ります。

<COUNT>      <ADDR>、<DATA BYTES>、  
<CHECKSUM>のデータの総数(バイト数)を16進数で示します。

<ADDR> その行の最初のデータバイトのアドレスを示します。S2フォーマットの<ADDR>フィールドは3バイトの長さを持ちます。

<DATA BYTES> 1バイト単位のデータが、アドレスが増加する順に割り付けられます。通常、このフィールドは32バイト(最大)のデータを含みます。

<CHECKSUM> その行に割り付けられた(Sフィールドを除く)全てのバイトの数値合計の1の補数となる値です。

## モトローラS2フォーマット

S224000380788812CF7C8812CFC0CFC1CFC2CFC3CFC4CFC5CFC6CFC7CFD0CFD1CFD2CFD3CF7C  
S2240003A0D4CFD5CFD6CFD7CFD8CFD9CFDACFDBCFDCCFDDECDFDCFE0CFE1CFE2CFE3CF90  
S2240003C0E4CFE5CFE6CFE7CFE8CFE9CFEACFEBCFECFFEDCFEECFEFCEFF0CFF1CFF2CFF3CE71  
S2240003E0F4CEF5CEF8CFE9CFFACFFEDD8812C8C8C9C9CACACCCCCCCCCDCDA8A9AAABACAD28  
S224000400AEAFCFB4CFB5CFB6CFB7CFBCFFBDA01A2A3A4A5A6A7CFB0CFB1CFB2CFB3CFB8AC  
S224000420CFB9CF67CE94CE95CE9688CE97CE90CE91CE9288CE93CE94CE95CE9688CE97CE98CE99CE22  
S22400044098CE99CE9A88CE9BCE80CE81CE8288CE83CE84CE85CE8688CE87CE88CE89CE8A9E  
S22400046000CE8BCE8CCE8DCE8E88CE8FE438E536E634E732CEE02FCEE12CEE229CEE326CE

## A.2.6 シンボル情報

### A.2.6.1 シンボル情報の生成(rel88)

シンボル情報生成ユーティリティrel88は、指定したオブジェクトファイルからシンボル情報を取得してそのリストを生成します。対象となるオブジェクトファイルは、asm88で生成されたりロケータブルオブジェクトファイルとlink88で生成されたアブソリュートオブジェクトファイルです。

通常、このツールを使用するのはリンク後のシンボルリストを見るためと、ICEによるシンボリックデバッグ用にシンボリックテーブルファイルを生成するsym88の入力ファイルを生成する場合です。

rel88は標準出力に対してリストを出力します。

以下にアブソリュートオブジェクトファイルのシンボルリストを得るための操作方法について説明します。

#### <rel88の操作方法>

アブソリュートオブジェクトファイルのシンボルリスト作成時

- (1) アブソリュートオブジェクトファイル(.a)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでrel88を起動します。

```
rel88[_フラグ]_入力ファイル名_>_出力ファイル名□
```

\_はスペースの入力を表します。

□はリターンキーの入力を表します。

一般的なフラグ

フラグ	説 明
+sec	各セクションの開始アドレスとサイズを出力します。
-v	セクション内をシンボルの値でソートします。

フラグの効果については下記の例を参照してください。また、フラグの詳細についてはAppendix Cを参照してください。

rel88の出力は標準出力に対して行われるため、出力リダイレクトによってファイルを生成します。

例 A:¥USER>a:¥EPSON¥rel88 -v +sec sample.a > sample.ref□

AドライブのサブディレクトリUSER内に作成されているアブソリュートオブジェクトファイル"sample.a"を入力し、シンボルリストファイル"sample.ref"を入力ファイルと同じディレクトリに生成します。

rel88へのPATHが設定されている場合は、rel88の前のパス指定は不要です。

生成されるシンボルリストは以下のようになります。

#### フラグとの対比説明

\*\*\* rel88(デフォルト)のフォーマット \*\*\*

```
0x8000c      acia.o
0x80b8d      acia.o
0x8000C  n_getch
0x80bcD  _buffer
0x8059C  n_recept
0x8045C  n_outch
0x80baD  _ptlec
0x80b8D  _ptecr
0x8082C  n_main
```

\*\*\* rel88 -v のフォーマット \*\*\*

```
SECTION 1
0x008000 c      acia.o
0x008000 C  n_getch
```

```

0x008045 C  n_outch
0x008059 C  n_recept
0x008082 C  n_main

```

```

SECTION 2
0x0080b8 d      acia.o
0x0080b8 D  _ptecr
0x0080ba D  _ptlec
0x0080bc D  _buffer

```

\*\*\* rel88 +sec のフォーマット \*\*\*

```

SECTION 1:  code
      address = 0x008000  size = 0x000b8

```

```

SECTION 2:  data
      address = 0x0080b8  size = 0x00000

```

(以下参考)

\*\*\* -a のフォーマット \*\*\*

```

0x000000 c  sec: 1      acia.o
0x0000b8 d  sec: 2      acia.o
0x0000bc D  sec: 2  _buffer
0x0000b8 D  sec: 2  _ptecr
0x0000ba D  sec: 2  _ptlec
0x000000 C  sec: 1  n_getch
0x000082 C  sec: 1  n_main
0x000045 C  sec: 1  n_outch
0x000059 C  sec: 1  n_recept

```

\*\*\* -d のフォーマット \*\*\*

```

0x000000 c      acia.o
0x0000b8 d      acia.o
0x000000 C  n_getch
0x0000bc D  _buffer
0x000059 C  n_recept
0x000045 C  n_outch
0x0000ba D  _ptlec
0x0000b8 D  _ptecr
0x000082 C  n_main

```

\*\*\* -g のフォーマット \*\*\*

```

0x000000 C  n_getch
0x0000bc D  _buffer
0x000059 C  n_recept
0x000045 C  n_outch
0x0000ba D  _ptlec
0x0000b8 D  _ptecr
0x000082 C  n_main

```

\*\*\* +dec のフォーマット \*\*\*

```

      0 c      acia.o
    184 d      acia.o
      0 C  n_getch
    188 D  _buffer
      89 C  n_recept
      69 C  n_outch
    186 D  _ptlec
    184 D  _ptecr
    130 C  n_main

```

### A.2.6.2 シンボリックテーブルファイルの生成(sym88)

シンボリックテーブルファイル生成ユーティリティsym88は、シンボル情報生成ユーティリティrel88から出力されたシンボル情報リファレンスファイル(.ref)を、ICEでシンボリックデバッグを行う際に必要なシンボリックテーブルの情報ファイルに変換します。

#### <sym88の操作方法>

- (1) シンボル情報リファレンスファイル(.ref)が存在するディレクトリをカレントドライブにします。
- (2) 次のフォーマットでsym88を起動します。

**sym88\_入力ファイル名** ☐

☐はスペースの入力を表します。

☐はリターンキーの入力を表します。

例 A:¥USER>a:¥EPSON¥sym88 sample.ref ☐

AドライブのサブディレクトリUSER内に作成されているシンボル情報リファレンスファイル"sample.ref"を入力し、シンボリックテーブルファイル"sample.sy"を入力ファイルと同じディレクトリに生成します。

sym88へのPATHが設定されている場合は、sym88の前のパス指定は不要です。

# Appendix B アセンブリソースファイルの作成方法(サブツールチェーン)

## B.1 概要

アセンブラ言語によってプログラムを開発される場合は、はじめにCPUの命令セットおよびクロスアセンブラの持つ擬似命令を使用し、アセンブリソースファイルを作成します。アセンブリソースファイルは以降説明する内容および規則にしたがって、お手持ちのエディタを使用して作成してください。

### B.1.1 ファイル名

本アセンブラはA.2.3項で説明したとおり、マクロ命令をasm88でアセンブル可能な形式に展開する構造化プリプロセッサsap88と、実際にアセンブルを行うクロスアセンブラasm88にわかれています。この一連の流れの中で扱われるファイルは全てアセンブリソースファイルですが、それぞれの内容が少しずつ異なるため、ファイル名の拡張子を以下のように規定しています。

#### 構造化アセンブリソースファイル

file\_name.s

構造化プリプロセッサsap88に入力する、マクロ命令等を含んだアセンブリソースファイルです。アセンブラ言語によりプログラムを作成する場合は、この".s"の拡張子を持つファイル名にしてアセンブリソースファイルを作成してください。

#### アセンブリソースファイル

file\_name.ms

構造化プリプロセッサsap88によって生成される、マクロ命令が展開されたアセンブリソースファイルです。

構造化プリプロセッサsap88およびクロスアセンブラasm88では他の拡張子のファイルも入力可能ですが、基本的には上記の拡張子を使用してください。

### B.1.2 sap88、asm88におけるソースファイルの相違点

前項でも説明したとおり、構造化プリプロセッサsap88で入力するソースファイルとクロスアセンブラasm88で入力するソースファイル形式は内容的に異なります。

マクロ命令やsap88擬似命令など、構造化プリプロセッサsap88で使用できるステートメント(行)はクロスアセンブラasm88では判別できずエラーとなります。そのため、マクロ命令を使用するときは、必ず構造化プリプロセッサsap88を用いてクロスアセンブラasm88に入力可能な形式に展開してください。

したがって、デバッグ時に直接asm88に入力するソースファイル".ms"を修正する場合などは注意が必要です。なお、クロスアセンブラasm88が機能的に持つ擬似命令などは、構造化プリプロセッサsap88でエラーとなることはありません。

以降、擬似命令等の説明の中で構造化プリプロセッサsap88に使用可能なものについては、[sap88 only]の記述、あるいは注釈を入れておきますので注意してください。

### B.1.3 マクロ命令

マクロ命令は一連の命令ブロックをユーザが定義した仮想命令に置き換える機能を持つもので、構造化プリプロセッサsap88はクロスアセンブラasm88でアセンブル可能なソース形式への展開を行います。以下に、概要を述べておきます。

プログラム上で同じステートメントブロックを複数箇所で使用する場合、そのステートメントブロックをあらかじめ任意の名前で定義しておくことにより、以後はその定義された名前でステートメントブロックを呼び出すことができます。

この定義されたステートメントブロックをマクロと呼びます。

プログラム中では定義済みのマクロ名と必要なパラメータを記述しマクロを呼び出します。その部分は構造化プリプロセッサsap88によりマクロ定義したステートメントブロックの内容に展開され、その際与えられたパラメータの置き換えも行われます。

なお、構造化プリプロセッサsap88ではマクロ定義、およびその呼び出し以外にもマクロにかかわる擬似命令がいくつか用意されています。その詳細についてはB.3.8項を参照してください。

## B.2 ソースファイルの一般形式

アセンブリソースファイルは、CPUの命令セットや、sap88、asm88の持つ擬似命令、コメント等のステートメント(行)で構成され、END擬似命令(アセンブルを終了させる擬似命令)で終了します。(END擬似命令以降にステートメントを記述することは可能ですが、その部分はアセンブルされません。)

以下に述べる内容は基本的にasm88を前提として説明を行います。(なお、asm88で許されている機能については、sap88でエラーとなることはありません。)

### ソースファイル例

```

        subtitle      "assembly source file example (sample.s)"
        public        main
        external      src_address,dst_address,counter

;
        code
main:
        ld            ix,[src_address]
        ld            iy,[dst_address]
        ld            hl,[counter]
        ret
;***
        end

```

以降、ステートメントの構成、使用できる文字や数値表記などの一般事項について説明します。

ソースプログラムの各ステートメントは次の書式で記述します。

シンボルフィールド	ニーモニックフィールド	オペランドフィールド	コメントフィールド
例 on	equ	1000h	
start:	jrl	init	;to initialize
flag:	db	[1]	
value:	db	080h	

上記のようなフォーマットの行は、普通、行末が終端となりますが、オペランドを複数の行に渡って記述することもできます。

#### ・シンボルフィールド

このフィールドにはシンボルを記述します。

シンボルの直後には、EQUまたはSET命令のステートメントを除いてコロン(:)を必ず付けます。

シンボルは以下の定義に基づいて、使い分けてください。

シンボル

- (1) ラベル (コロン(:)を後ろに必ず付ける)
- (2) 名前 (EQUまたはSET命令による定数定義)

#### ・ニーモニックフィールド

このフィールドにはオペコード、擬似命令を記述します。

#### ・オペランドフィールド

このフィールドには、各命令のオペランドや定数、変数、定義したシンボル、メモリアドレスを示すシンボル、演算式を記述します。

#### ・コメントフィールド

このフィールドの先頭には、セミコロン(;)を置き、以降にコメント文を記述します。



## B.2.1 シンボル

シンボルは、特定の値に対して定義する名前です。シンボルの定義方法には、次の2通りがあります。

### (1) ラベル

CPUの命令セットやデータ定義をするステートメントにはラベルとして定義します。このときシンボルの持つ値は、CPUの命令セットやデータ領域のアドレスです。

### (2) 名前

EQUやSET擬似命令を使って定義します。このときシンボルの持つ値は、EQUやSET擬似命令で指定した<式>の値です。

シンボルは以下の規則にしたがいます。

- シンボルの長さは自由ですが、シンボルとして区別できるのは、最大、先頭から15文字までです。
- ラベルの場合はどのカラムからでも記述できますが、必ず後尾にコロン(:)を付けなければなりません。
- 名前の場合は、1カラム目から始めなければなりません。
- シンボルに使える文字は次のとおりです。  
英文字 (A～Z, a～z)、アラビア数字 (0～9)、\_
- シンボルは大文字、小文字のいずれで入力しても構いません。デフォルトでは大文字、小文字の区別はしませんので、ABCとabcのシンボルは同一として扱われます。ただし、-cフラグを使用すると、大文字、小文字を区別します。
- シンボルは数字で開始することはできません。必ず英文字または "\_" で開始するようにしてください。

## B.2.2 ニーモニック

ニーモニックフィールドにはCPUの命令セット、擬似命令が置かれます。これらについては後述しますが、通常は空白で終わる文字列で構成されます。

asm88およびsap88ではデフォルト設定において大文字、小文字の区別を行いません。その場合、次のように入力しても全て正しく、同じものと見なされます。

例

```
byte
BYTE
bYtE
```

CPUの命令セットも、デフォルト設定では大文字、小文字どちらで書いても許されます。しかし、プログラムを書くときはお客さまが通常使われている文字で書くほうがよいでしょう。

ただし、-cフラグでシンボル名の大文字、小文字を区別した場合、CPUの命令セットおよびレジスタ名は必ず小文字で記述してください。

例

```
jrl  ABC      ;jumps to label ABC
ld   a,b      ;A register <- B register
```

## B.2.3 オペランド

ニーモニックフィールドの内容に対して、0個以上のオペランドを置くことができます。これらはパラメータの列で与えられ、ニーモニックフィールドの終端を示す空白文字から始まり、カンマで区切られ、空白文字またはセミコロンで終わります。

## B.2.4 コメント

コメントはアセンブル過程では無視されます。コメントは";"(セミコロン)で始まり行末の終端(改行コード)で終わります。

## B.2.5 数値表現

機能組み込み用マイコンでは、ビット操作が頻繁に行われます。そのため、asm88およびsap88では数値表現の基数としてbinary(2進数)、octal(8進数)、hexadecimal(16進数)、decimal(10進数)の各表現を取り扱うことができます。

基数は、次の文字を数に後置させることで認識されます。

B (binary) : 2進数  
 O,Q (octal) : 8進数  
 H (hexadecimal): 16進数  
 なし : 10進数(D(decimal)を使うこともできます)  
 (これらは小文字で書くことも可能です)

数は必ずアラビア数字(0～9)で始まらなくてはなりません。例えば、"10"という数字は次のように表すことができます。

10 : 10進数  
 1010B : 2進数  
 12Q : 8進数  
 0AH : 16進数  
 (名前のAHと区別するため、16進数のA～Fで始まる数については数字の先頭に0が必要です。)

## B.2.6 文字

sap88およびasm88は、文字、文字列の表現としてAmerican variant of the ISO 5 alphabet(通常ASCII)と呼ばれている表現法を採用しています。

## B.2.7 ASCIIキャラクタセット

ASCIIキャラクタセットコードは、文字に対応した7bitデータ、転送の際にエラーがあるかどうかチェックするための1bitパリティの2つの部分で構成されています。ASCIIキャラクタセットは次のように4つに分類できます。

表B.2.7.1 ASCIIキャラクタコード表

L\H	00	01	02	03	04	05	06	07
00	NUL	DEL	SP	0	@	P	`	p
01	SOH	DC1	!	1	A	Q	a	q
02	STX	DC2	"	2	B	R	b	r
03	ETX	DC3	#	3	C	S	c	s
04	EOT	DC4	\$	4	D	T	d	t
05	ENQ	NAK	%	5	E	U	e	u
06	ACK	SYN	&	6	F	V	f	v
07	BEL	ETB	'	7	G	W	g	w
08	BS	CAN	(	8	H	X	h	x
09	HT	EM	)	9	I	Y	i	y
0a	LF	SUB	*	:	J	Z	j	z
0b	VT	ESC	+	;	K	[	k	{
0c	FF	FS	'	<	L	\(¥)	l	
0d	CR	GS	-	=	M	]	m	}
0e	SO	RS	.	>	N	^	n	~
0f	SI	US	/	?	O	_	o	DEL
	00	01	10	11				

領 域

asm88において表記文字は、'A'、'Z'、'X'のように引用符で囲むことで文字定数として扱われます。なお、引用符自身は特別に'¥'とします。

制御コードなど表示されない文字を表す際、asm88では特に使用頻度が高いと思われる以下の制御文字に対し、次のようなエスケープシーケンスによる表記法を許しています。

'¥a'	ベル	(07H)
'¥n'	改行	(0AH)
'¥r'	復帰	(0DH)
'¥t'	タブ	(09H)
'¥b'	バックスペース	(08H)
'¥e'	エスケープ	(1BH)
'¥i'	シフトイン	(0FH)
'¥o'	シフトアウト	(0EH)

また、¥nnnという表記(nnnは8進数)も使えます。この表記を使った場合、例えばベルは'¥007'と書くことができます。

これらエスケープシーケンスによる記述は文字列だけに許されます。文字列は、ASCII命令で取り扱うことができ、やはり引用符で囲まれたキャラクタの集合で表されます。

## B.2.8 式

定数は、例えばCPUの命令セットのオペランド、擬似命令のパラメータなどプログラム中の多くの場所で設定されます。また、定数は式を用いて表すこともできます。クロスアセンブラasm88は、アセンブル時に式を評価し、その値を定数とすることができます。この式の計算は、アセンブル中CPUが扱う数と同じか、またはより大きいサイズの変数を使い評価します。

注意(1) リロケータブルコードを作るとき、アドレスは式の結果がリロケータブルとなる量、もしくは定数になるような式の中でしか使えません。

したがって、次の式は使用可能です。

```
label1 - label2 ;2つのラベルが同じプログラムセクションにある場合
label1 + <定数>
label1 - <定数>
```

次の式はリロケータブルな量、または定数とならないので使用できません。

```
label1 + label2
label1 & label2
label1 * <定数>
label1 / <定数>
label1 % <定数>
label1 * label2
label1 / label2
label1 % label2
<定数> + label2
label1 - label2 ;2つのラベルが別のプログラムセクションにある場合
```

注意(2) リロケータブルなアドレスを使った論理演算は、結果がリロケータブルな量にならないのでアセンブル時エラーとなります。

式は、二項演算子(例えば、+)で結ばれたいくつかの項で構成されます。これらの式の評価は16bit精度で計算されます。

式の中の項には、次のものが使えます。

- 1 数
- 2 EQU、SET命令によってユーザが定義した変数、宣言済みのラベル
- 3 ロケーションカウンタ\$

\$をCPUの命令のオペランドとして使った場合、アセンブル時その命令の直前のアドレスが与えられます。asm88は2パスアセンブラで、パス1の段階ではプログラム中で使用されているいくつかの変数は値が確定しません。パス1実行では値が不確定の変数が式中に現われた場合、0が与えられます。そしてパス2実行で、もし値がまだ不確定な変数があるとエラーとなります。またパス2では、パス1で式に使われたときに値が不確定だった変数を使うとフェーズエラーの原因になります。したがって、変数は式に使われる前に値を確定しておいてください。

## B.2.9 演算子

asm88は、以下の演算子を受け付けます。

表B.2.9.1a 単項演算子

演算子	機 能
+a	正号 例) <code>ld a,#+25h</code>
-a	負号 例) <code>add b,#-13h</code>
~a	各bitを反転させた値を与えます。 例) <code>and a,#~10h</code>
LOW a	式の下位8ビットの値を与えます。 例) <code>or b,#low 1234h</code>
HIGH a	8回右へシフトした後の下位8ビットの値を与えます。これは、16ビットの式の上位8ビットを返すことと同じです。 例) <code>ld h,#high 1020h</code>
BOC	物理アドレスのみに作用することができ、物理アドレスからバンク値を算出します。 (Bank Of Code) 例) <code>ld a,#boc label</code> <code>ld nb,a</code>
LOC	物理アドレスのみに作用することができ、物理アドレスから論理空間内の論理アドレスを算出します。 (Logical address Of Code) 例) <code>ld hl,#loc label</code> <code>jp hl</code> <code>label: ^</code>
POD	物理アドレスのみに作用することができ、物理アドレスからページ値を算出します。 (Page Of Data) 例) <code>ld a,#pod label</code> <code>ld ep,a</code>
LOD	物理アドレスのみに作用することができ、物理アドレスからページ内の論理アドレスを算出します。 (Logical address Of Data) 例) <code>ld ix,#lod label</code> <code>ld a,[ix]</code> <code>label: ^</code>

表B.2.9.1b 二項演算子

演算子	機 能
a+b	加算(32ビットの整数-符号付き)を行います。 例) <code>sbc [hl],#25h+10h</code>
a-b	減算(32ビットの整数-符号付き)を行います。 例) <code>sub a,#63h-03h</code>
a*b	乗算(32ビットの整数-符号付き)を行います。 例) <code>xor l,#48h*5h</code>
a/b	整数除算(32ビットの整数-符号付き)を行います。 例) <code>cp ba,#1256h/31h</code>
a%b	剰余の計算を行います。左オペランドを右オペランドで割り、その余り(剰余)の値を返します。 例) <code>add a,#0d7h%4fh</code>
a&b	論理積。両オペランドが真の場合に真を返します。いずれかのオペランドが偽であるか、両オペランドが偽の場合に偽を返します。 例) <code>ld sp,#04ah&amp;2030h</code>
a b	論理和。いずれかのオペランドが真であるか、両オペランドが真の場合に真を返します。 例) <code>ld ix,#3026h 1000h</code>
a^b	排他的論理和。いずれかのオペランドが真であり、他方が偽である場合に真を返します。両オペランドが共に偽である場合に偽を返します。 例) <code>ld [iy],#44h^10h</code>
a<<b	左シフトを行います。b(整数)の数だけ左へシフトします。 例) <code>adc hl,#5000h&lt;&lt;3</code>
a>>b	右シフトを行います。b(整数)の数だけ右へシフトします。 例) <code>cp ba,#8130h&gt;&gt;10h</code>

### 演算子の優先順位

表現式は左から右へ評価されていきますが、優先順位の高い演算子は、その直前、直後の他の演算子より先に評価されます。2つの演算子の優先順位が等しいとき、左側の演算子から評価されます。左カッコ("("と右カッコ")"は、同数でなければいけません。

次に、演算子の優先順位の表を示します。

表B.2.9.2 演算子の優先順位

演算子	優先順位
!, ^, & +(加算), -(減算) *, / , %, <<, >> BOC, LOC, POD, LOD HIGH, LOW, ~, -, +	低い       高い

### BOC, LOC, POD, LODの演算規則

単項演算子のうち、BOC、LOC、POD、LODの4つはS1C88に固有の演算子で、それぞれが以下のように独自の演算規則を持ちます。

```
BOC (Physical address & 0x7f8000) >> 15
LOC If (Physical address & 0x7f8000)
    (Physical address & 0x7fff) ! 0x8000
    else
        (Physical address & 0x7fff) ! 0x0000
POD (Physical address & 0xff0000) >> 16
LOD (Physical address & 0xffff)
```

ここで、Physical addressはオペランドが持つ物理的な値を示し、アSEMBル時には、asm88は各演算子に対応した特殊なリロケーション情報を生成するだけで、実際のアドレス計算はリンク時にlink88によって行われます。

### B.2.10 命令セット

asm88はCPUの命令セットとして次の各命令を受け付けます。

— S1C88 Family命令一覧 —						
adc	cp	inc	neg	rete	sep	swap
add	cpl	int	nop	rets	sla	upck
and	dec	jp	or	rl	sll	xor
bit	div	jrl	pack	rlc	slp	
call	djr	jrs	pop	rr	sra	
carl	ex	ld	push	rrc	srl	
cars	halt	mlt	ret	sbc	sub	

### B.2.11 レジスタ名

asm88では次に示すCPUのレジスタ名が予約語として予約されています。各レジスタの機能等に関しては"S1C88コアCPUマニュアル"を参照してください。

a	データレジスタA
b	データレジスタB
ba	AとBのレジスタペア
h	データレジスタH
l	データレジスタL
hl	インデックスレジスタHL
ix	インデックスレジスタIX
iy	インデックスレジスタIY
sp	スタックポインタSP
br	ベースレジスタBR
sc	システムコンディションフラグSC
cc	カスタマイズコンディションフラグCC
pc	プログラムカウンタPC
nb	ニューコードバンクレジスタNB
cb	コードバンクレジスタCB
ep	エキスパンドページレジスタEP
xp	IX用エキスパンドページレジスタXP
yp	IY用エキスパンドページレジスタYP
ip	XPおよびYPレジスタ

## B.2.12 アドレッシングモード

S1C88は次の12種類のアドレッシングモードによって実行アドレスを決定します。

表B.2.12.1 S1C88アドレッシングモード一覧

No.	アドレッシングモード
1	即値データアドレッシング
2	レジスタ直接アドレッシング
3	レジスタ間接アドレッシング
4	ディスプレースメント付きレジスタ間接アドレッシング
5	インデックスレジスタ付きレジスタ間接アドレッシング
6	8ビット絶対アドレッシング
7	16ビット絶対アドレッシング
8	8ビット間接アドレッシング
9	16ビット間接アドレッシング
10	符号付き8ビットPC相対アドレッシング
11	符号付き16ビットPC相対アドレッシング
12	インプライドレジスタアドレッシング

各アドレッシングモードの詳細に関しては"S1C88コアCPUマニュアル"を参照してください。

これらのアドレッシングモードに対応したオペランドの表記規則は次のとおりです。

表B.2.12.2 オペランドの表記規則

アドレッシングモードNo.	表 記 規 則
1	数値式やシンボルに#を前置させる
2	レジスタ名を直接書く
3	インデックスレジスタをブラケット([ ])で囲む
4	インデックスレジスタとディスプレースメントをブラケット([ ])で囲む
5	インデックスレジスタ+I(+L)をブラケット([ ])で囲む
6	数値式やシンボルにbr:(BR:)を前置させブラケット([ ])で囲む
7	数値式やシンボルをブラケット([ ])で囲む
8	数値式やシンボルをブラケット([ ])で囲む
9	数値式やシンボルをブラケット([ ])で囲む
10	数値式やシンボルを直接書く
11	数値式やシンボルを直接書く
12	なし

### B.2.13 ニーモニックの表記例

以下に各アドレッシングモードのニーモニック表記例を示します。

アドレッシング	定 数	名 前 name equ 50h	ラベル (デフォルト) label: アドレス00ffh	デフォルトの定義
#nn 0 ~ 255	例) ld a, #0ffh	例) ld a, #name	例) ld a, #label	_____
#mmnn 0 ~ 65535	例) ld ba, #1000h	例) ld ba, #name	例) ld ba, #label	_____
[br:l] 0 ~ 255	例) ld b, [br:0ffh]	例) ld b, [br:name]	例) ld b, [br:label]	[br:low lod label]
[hhll] 0 ~ 65535	例) ld l, [1000h]	例) ld l, [name]	例) ld l, [label]	[lod label]
[ix+dd] [iy+dd] [sp+dd] -128 ~ 127	例) ld [ix+10h], a	例) ld [ix+name], a	_____	_____
#hh 0 ~ 255	例) ld br, #0ffh	例) ld br, #name	例) ld br, #label	high lod label
#pp 0 ~ 255	例) ld ep, #05h	例) ld ep, #name	例) ld ep, #label	pod label
#bb 0 ~ 255	例) ld nb, #05h	例) ld nb, #name	例) ld nb, #label	boc label
rr -128 ~ 127	例) jrs 10h	例) jrs name	例) jrs label	loc label
[kk] 0 ~ 255	例) jp [10h]	例) jp [name]	例) jp [label]	[low lod label]
qrr -32768 ~ 32767	例) jrl 1000h	例) jrl name	例) jrl label	loc label

- 上記、デフォルトの定義の意味は以下のとおりです。  
例として、

```
jrl label
```

のように記述した場合は、

```
jrl loc label
```

とクロスアセンブラasm88が判断し、ラベルの物理アドレスを論理アドレスに変換したアドレスにロングジャンプします。

- 上記のアドレッシングの範囲を越えて指定した場合、または越えたと判断した場合はエラーとなります。
- ショート分岐およびロング分岐命令は、以下の点に注意してプログラミングしてください。

```
jrs(1) 10h ..... カレントアドレスから相対的に(10+1)H離れたアドレスにジャンプします。
```

```
jrs(1) $+10h .... カレントアドレスから相対的に10H離れたアドレスにジャンプします。
```

上記以外の表記に関しては、"S1C88コアCPUマニュアル"の表記をそのまま使用できます。

### B.3 擬似命令

この項にはasm88、sap88でサポートされている各種擬似命令の使用法が機能別に分類された形で説明されています。各説明は、いつでも参照できるように、以下のような形式を採用しています。

#### 説明の見方

各擬似命令の説明内容は次のようなフォーマットで構成されています。

##### 1) 名前

擬似命令の名称 .... 命令の機能

##### 2) 形式

ここでは、命令の書式が記述されています。

なお、書式については次の規則にしたがう表記法を用いて説明します。

オペランドの表記に使用されている各語の凡例は次のとおりです。

<式> ..... 演算子を含むシンボルや定数によって構成される一般式

<数値式> ..... 数値表現による定数式(EQU命令で定数定義された名前も含む)

<ラベル> ..... リロケート可能な性質を持つ自己モジュール内に定義のあるシンボル

<名前> ..... EQUとSET命令で定義されたシンボル

<シンボル> .. 特定の値に対して定義する名前

<文字列> ..... 二重引用符で囲まれた文字列

次の表記には特別な意味を持たせています。

{ } ..... ここで囲まれた部分は任意選択を意味します。

{ } \* ..... このオプションは何回も繰り返し置くことができます。

{ } { } { } ..... 何種類かの違うパラメータを取ることができるとき、この記号で区切られている中の1つをパラメータとして必ず使わなければなりません。

他の記号

カンマ','、括弧ブラケット'[、]'および'('、')'はアセンブラのソースとして入力できるものです。

##### 3) 機能

ここでは、命令の働きが詳しく述べられています。

##### 4) 例

ここでは、使用例が書かれています。

命令によっては、数種類書かれているものもあります。

##### 5) 関連項目

ここでは似たような働きをする命令や、理解の助けとなる命令が書かれています。

##### 6) 制限

ここでは、命令の使用上の制限を述べています。また、命令を使う際に多く発生するエラーの原因(例えば、セパレータを忘れる等)についても述べられています。



### B.3.1 領域設定擬似命令

領域設定擬似命令は、各セクション(コードセクション/データセクション)を設定し、プログラムの領域を確定します。領域設定擬似命令は以下のとおりです。

#### CODE DATA

クロスアセンブラasm88の領域設定擬似命令は、コードセクションをROMに、データセクションをRAMに配置することを前提として定義されています。これは、機器組み込み用マイコンが電源投入時、初期値が不定となるRAM領域を持っているため、RAMにプログラムコードおよび定数データ等の不揮発性データを配置しないようにすることを目的としています。このため、プログラマはプログラムコードおよび定数データ等の不揮発性データを記述する場合、CODE擬似命令によってコードセクションを設定し、コードセクション内に記述する必要があります。また、ワークエリア、スタックエリア等の揮発性データは、DATA擬似命令によってデータセクションを設定し、データセクション内に記述する必要があります。

各擬似命令、設定領域、使用領域および記述する内容の対応を下表に示します。

セクション名	使用領域	記述する内容
コードセクション (CODE)領域	ROM	プログラムコード、定数データおよびテーブル等 電源投入時より確定しておく必要性のあるデータ配置
データセクション (DATA)領域	RAM	ワークエリア、スタックエリア、フラグおよびバッファ等 電源投入時の初期値は不定でも構わないデータの領域確保

名前:

**CODE .....** コードセクションの定義

形式:

**CODE**

機能:

この命令は、アセンブル時にプログラムおよび定数をコードセクション(ROM領域)に配置させるために用いられます。1つのモジュール内に任意の数のコードセクションが定義でき、アセンブル時にレジュームされます。

この命令は、DATA擬似命令と同様にセクション指定をするため、コードセクションをアセンブルするときには、必ず指定するようにしてください。指定していない場合は、エラーメッセージが出力されます。

例:

プログラムおよび定数をコードセクションに定義します。

```
code
trans: ld    [iy],[ix]
      inc    ix
      inc    iy
      djr    nz,trans
      ret
      db     01h,02h,03h,04h,05h
```

関連項目:

DATA、ORG

名前:

**DATA .....** データセクションの定義

形式:

DATA

機能:

この命令は、データ領域をデータセクション(RAM領域)に確保し配置させるために用いられます。1つのモジュール内に任意の数のデータセクションが定義でき、アセンブル時にレジュームされます。通常データセクションは領域確保のみ行い、アセンブル結果のオブジェクトには出力されません。しかし、このセクションはRAM領域であり、機器組み込みでは電源投入時RAM領域は不定で、初期値は意味を持ちませんので充分注意してください。この命令は、CODE擬似命令と同様にセクション指定をするため、データセクションをアセンブルするときには、必ず指定するようにしてください。指定していない場合は、エラーメッセージが出力されます。

例:

フラグおよびバッファテーブルをデータセクションに領域確保します。

```
        data
flag:   db    [1]
buffer: db    [256*8]
```

関連項目:

CODE、ORG

### B.3.2 データ定義擬似命令

データ定義擬似命令は、メモリに格納するデータを定義する擬似命令です。データ定義擬似命令は以下のとおりです。

DB  
DW  
DL  
ASCII  
PARITY

名前:

## DB .... バイト単位のデータ領域の確保/定数の設定

形式:

DB	<式>{,<式>}*	形式1
DB	<式>(<数值式>){,<式>(<数值式>)}*	形式2
DB	[<数值式>]{, [<数值式>]}*	形式3

機能説明:

この命令は、1バイト単位のリデータ領域の確保および定数を設定するために用いられます。定数の設定はカンマで区切られた数値の列、もしくは繰り返し数の指定によって行われます。この命令のパラメータは複数の行に渡って記述することができますが、リンクの際のリロケーションに関する情報は一切含まれていないように注意してください。なお、この命令を使用するとき、データ領域の確保を行う場合はDATA(RAM)領域に、定数の設定を行う場合はCODE(ROM)領域に記述してください。各形式のコード生成規則は次のとおりです。

### 形式1

任意の定数を1バイト単位で任意個数オブジェクトコードとして定義する形式で、オペランドフィールドには複数個の式が指定できます。式は1バイトの定数値として扱われ、複数個指定した場合は指定順にオブジェクトコードが生成されます。

形式2

任意の定数を1バイト単位で繰り返し定義する形式で、繰り返し数を括弧で囲んだ<数値式>で設定します。

### 形式3

ブラケットで囲んだ<数値式>によって与えられたバイト数分の領域を確保します。このとき、オブジェクト中に生成されるコードは0です。

形式1および2の<式>としては、整数の数値定数、文字定数、シンボルが使用できますが、必ず絶対数値属性を持たなくてはなりません。また、式の値は-128～255の範囲でなければなりません。演算結果が、上記の範囲外となった場合はエラーとし、下位1バイトの値を評価値とします。

1命令に対し、各形式を混在させて置くこともできます。

例:

```
buffer: db [50] ;50バイト分の領域を確保
tratbl: db '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
;16バイト分のデータを定数として確保
xhdbuf: db '(64) ;64バイトを確保しスペースのキャラクタコードで初期化
;*(64) ;64バイト'*'を定数として確保
```

関連項目:

DW、DL

名前:

**DW....** ワード単位のデータ領域の確保/定数の設定

形式:

DW <式>{,<式>}*	形式1
DW <式>(<数値式>){,<式>(<数値式>)}*	形式2
DW [<数値式>]{,<数値式>}*	形式3

機能説明:

この命令は、ワード(2バイト)単位の領域の確保および定数を設定するために用いられます。定数の設定はカンマで区切られた数値の列、もしくは繰り返し数によって行われます。

この命令のパラメータは複数の行に渡って記述することができます。なお、この命令を使用するとき、データ領域の確保を行う場合はDATA(RAM)領域に、定数の設定を行う場合はCODE(ROM)領域に記述してください。各形式のコード生成規則は次のとおりです。

形式1

任意の定数をワード(2バイト)単位で任意個数オブジェクトコードとして定義する形式で、オペランドフィールドには複数個の式が指定できます。式はロングワードの定数値またはシンボル値として扱われ、複数個指定した場合は指定順にオブジェクトコードが生成されます。

形式2

任意の定数をワード単位で繰り返し定義する形式で、繰り返し数を括弧で囲んだ<数値式>で設定します。

形式3

ブラケットで囲んだ<数値式>によって与えられた個数分ワードの領域を確保します。このとき、オブジェクト中に生成されるコードは0です。

形式1および2の<式>としては、整数の数値定数、文字定数、シンボルが使用できます。<式>がリロケータブルな性質を持つ場合、リンク時に当該シンボルが割り付けられている場所の論理アドレスが再配置されます。また、定数の場合式の値は-32768～65535の範囲でなければなりません。演算結果が、上記の範囲外となった場合はエラーとし、下位2バイトの値を評価値とします。

1命令に対し、各形式を混在させて置くこともできます。

例:

```
array:  dw [10]      ;10個のワードサイズの領域を確保
        external func1,func2,func3,func4,func5
jmpTbl: dw func1,func2,func3,func4,func5
        ;関数のジャンプテーブル
```

関連項目:

DB、DL

名前:

**DL ..... ロングワード単位のデータ領域の確保/定数の設定**

形式:

DL <式>{,<式>}*	形式1
DL <式>(<数値式>){,<式>(<数値式>)}*	形式2
DL [<数値式>]{,<数値式>}*	形式3

機能説明:

この命令は、ロングワード(4バイト)単位の領域の確保および定数を設定するために用いられます。定数の設定はカンマで区切られた数値の列、もしくは繰り返し数によって行われます。この命令のパラメータは複数の行に渡って記述することができます。なお、この命令を使用するとき、データ領域の確保を行う場合はDATA(RAM)領域に、定数の設定を行う場合はCODE(ROM)領域に記述してください。各形式のコード生成規則は次のとおりです。

形式1

任意の定数をロングワード単位で任意個数オブジェクトコードとして定義する形式で、オペランドフィールドには複数個の式が指定できます。式はロングワードの定数値またはシンボル値として扱われ、複数個指定した場合は指定順にオブジェクトコードが生成されます。

形式2

任意の定数をロングワード単位で繰り返し定義する形式で、繰り返し数を括弧で囲んだ<数値式>で設定します。

形式3

ブラケットで囲んだ<数値式>によって与えられた個数分ロングワードの領域を確保します。このとき、オブジェクト中に生成されるコードは0です。

形式1および2の<式>としては、数値定数、文字定数、シンボルが使用できます。<式>がリロケータブルな性質を持つ場合、リンク時に下位16ビットの値が有効な値として再配置されます。

1命令に対し、各形式を混在させて置くこともできます。

例:

```
lubarr: dl [10] ;10個の4バイトサイズの領域を確保
lonum: dl 13768 ;lonumをロングワードサイズの整数で定数設定
```

関連項目:

DB、DW

## 名前:

**ASCII .....** ASCIIテキストのメモリ格納

## 形式:

ASCII 文字式 {,文字式}\*  
文字式 = 文字列 | 文字定数 | バイト定数

## 機能:

この命令は、ASCII キャラクタコードをメモリに格納するために用います。この命令によって確保された領域は、必ず与えられたパラメータのASCII テキストでメモリに格納されます。パラメータの文字列はデコードされ、メモリ下位アドレスから順に格納されます。領域の大きさは、デコードされたパラメータのバイト数となります。オペランドは二重引用符で囲まれた1文字以上の文字列です。

ASCII命令は文字列の各文字のキャラクタコードでメモリに格納しますが、長さや文字列の終端を示す情報は出力しないので自由に文字列を設定することができます。

## 例:

```
ascii "S1C88 Family"
ascii "bell", '\a'           ;bellとBELLコード
ascii "bell\07"             ;他のフォーマット例
ascii "bell", '\07'         ;他のフォーマット例
ascii 62h,65h,6ch,6ch,07h ;他のフォーマット例
```

## 関連項目:

ASCIIキャラクタセット表

## 名前:

**PARITY ....** パリティビットのセット/リセット

## 形式:

PARITY &lt;オペランド&gt;

## 機能:

クロスアセンブラasm88で採用されているアルファベットは、ASCIIキャラクタセットです。ASCIIキャラクタセットは7bitで表され、最上位bitはパリティを表しています。このビットをPARITY命令によって常時0、もしくは1に自由にセット/リセットが可能です。また、1のビットの総数を奇数や偶数にすることもできます。<オペランド>には次のものが指定できます。

```
PARITY 7      パリティビットを0にします (デフォルト)
PARITY 8      パリティビットを1にします
PARITY ODD    8bit中に'1'が奇数個になるようにセットされます
PARITY EVEN   8bit中に'1'が偶数個になるようにセットします
```

## 関連項目:

ASCIIキャラクタセット表

### B.3.3 シンボル定義擬似命令

シンボル定義擬似命令は、式を名前で定義する擬似命令です。シンボル擬似命令は、以下のとおりです。

**EQU**  
**SET**

---

名前:

**EQU** ..... 名前の値設定

形式:

<名前> EQU <式>

機能:

この命令は、<式>を<名前>で定義するために用いられます。この命令で定義された名前の値を以後変更することは許されません。また、等号の右辺にEXTERNAL宣言されたシンボルを置くことはできません。式の長さは自由ですが、アセンブリリストには16進6桁までしか表示されません。16進7桁以上の値を定義した場合は、ワーニングが出力されます。

また、sap88ではEQUによって定義された名前が、これ以降に現われるIFC文の条件式の中やIFDEF/IFNDEF文のパラメータとしても使用できます。[sap88 only]

例:

```
false equ 0           ;初期化
true  equ -1
tablen equ TABFIN-TABSTA ;テーブル長の計算
nul   equ 00h          ;ASCIIキャラクタを示す文字列を定義
soh   equ 01h
stx   equ 02h
etx   equ 03h
eot   equ 04h
enq   equ 05h
```

関連項目:

SET、IFC、IFDEF、IFNDEF、REPT

制限:

<名前>の記述は1カラム目から始めなければなりません。

名前:

**SET** ..... 名前の値設定

形式:

&lt;名前&gt; SET &lt;式&gt;

機能:

この命令は、EQUと同様、アセンブラのソースコードのメンテナンス性向上などを目的に、<名前>と<数値式>を結び付ける働きをします。SET命令で定義された名前はEQU命令の場合と異なり何回でも別の値に再定義することができ、アセンブラの変数として扱うことができます。このとき、アセンブラの出力リストの1つであるクロスリファレンスリストのシンボル属性が変数として定義されているものがこのシンボルに相当し、等号の右辺はこの命令の前に定義されていなくてはなりません。この命令は、条件アセンブラやマクロの変数として名前を使用することが主な目的であり、構造化プリプロセッサsap88では有用な機能として使われますが、クロスアセンブラasm88単体では名前の再定義ができるという機能以外あまり用途がありません。また、式の長さは自由ですが、アセンブリリストには16進6桁までしか表示されません。16進7桁以上の値を定義した場合は、ワーニングが出力されます。

また、sap88ではSETによって定義された名前が、これ以降に現われるIFC文の条件式の中やIFDEF/IFNDEF文のパラメータとしても使用できます。[sap88 only]

例:

```
abc  set  1
      ld   a, #abc
abc  set  2
      ld   a, #abc
```

関連項目:

EQU、IFC、IFDEF、IFNDEF、REPT

制限:

&lt;名前&gt;の記述は1カラム目から始めなければなりません。



### B.3.4 ロケーションカウンタ制御擬似命令

ロケーションカウンタ制御擬似命令は以下のとおりです。

#### ORG

---

名前:

**ORG** ..... ロケーションカウンタの値を変更

形式:

**ORG** <式>

機能:

この命令は、プログラムが置かれるアドレスを指定するために用いられます。<式>はカレントプログラムセクション中のラベルとの相対的な値しか取れません。このとき、プログラムカウンタにアブソリュートアドレスを入れようとするエラーになります。なお、式の長さは16進6桁まで定義でき、これ以上の値はエラーとなります。

例:

```
sizstk equ 200h ;スタックのサイズは512バイト
topstk:         ;スタックのためのスペースを確保

    org topstk + sizstk
```

関連項目:

CODE、DATA

### B.3.5 外部定義・外部参照擬似命令

モジュール間で共通に使用するシンボルを定義、参照する擬似命令です。

- 外部参照擬似命令

#### EXTERNAL

- 外部定義擬似命令

#### PUBLIC

名前:

**EXTERNAL** ..... シンボルの外部定義宣言

形式:

**EXTERNAL** <シンボル> {,<シンボル>}\*

機能:

複数のモジュール間で同じシンボルが使えるようにするために、EXTERNAL命令とPUBLIC命令が用いられます。自己モジュール内に定義がなく、他のモジュール中で定義されたシンボルを参照するには、EXTERNAL命令で宣言しなければなりません。

また、EXTERNALで宣言すればPUBLICも兼ねて宣言もされるようになっています。

例:

```
external    sqrt
carl        sqrt
```

関連項目:

PUBLIC

名前:

**PUBLIC** ..... シンボルのグローバル宣言

形式:

**PUBLIC** <シンボル> {,<シンボル>}\*

機能:

任意のシンボルを複数のモジュールで使う場合には、PUBLIC命令とEXTERNAL命令で宣言します。PUBLICは自己モジュール内に定義があり、別のモジュールからの参照を許可するようなシンボルの宣言に用います。

例:

```
public sqrt ;SQRTは他のモジュールから参照
sqrt:      ;整数の平方根を計算するルーチン
.....
etc.
```

関連項目:

EXTERNAL

### B.3.6 ソースファイル挿入擬似命令 [sap88 only]

ソースファイルの任意の場所に別のファイルを読み込み、挿入する擬似命令です。

#### INCLUDE

- \* この命令は、構造化プリプロセッサsap88でのみ使用可能です。sap88により、指定のファイルが挿入されたソースファイルに展開されます。なお、この命令はクロスアセンブラasm88では使用不可能であり、エラーとなります。

名前:

**INCLUDE ....** 別ファイルのインクルード

形式:

**INCLUDE** <ファイル名>

機能:

指定されたファイルをINCLUDE文の直後へ読み込みます。

インクルードは任意の深さまで入れ子にでき、インクルードされたファイル中からさらに別のファイルをインクルードできます。

この擬似命令はsap88で解析され、指定ファイルが挿入された出力ファイルが生成されます。この擬似命令がそのままasm88に渡されることはありません。

例:

```
include chargen.s ;キャラクタジェネレータ
include utilsub   ;汎用サブルーチン群
```

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

### B.3.7 アセンブル終了擬似命令

アセンブル終了擬似命令は、各ソースプログラムの終了を指示します。

**END**

---

名前:

**END** ..... アセンブルの停止

形式:

**END** {<ラベル>}

機能:

この命令は、アセンブルを停止させるために用います。この領域より後の部分はリストは出ますが、アセンブルされません。

### B.3.8 マクロ関係擬似命令 [sap88 only]

以下の擬似命令はマクロに関係するもので、マクロの定義や抹消、リピート定義等を行います。

**MACRO ~ ENDM**  
**DEFINE**  
**LOCAL**  
**PURGE**

**UNDEF**  
**IRP ~ ENDR**  
**IRPC ~ ENDR**  
**REPT ~ ENDR**

- \* 本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。sap88により、これらの擬似命令の設定内容がクロスアセンブラasm88によってアセンブル可能な形に展開されたソースファイルが出力されます。また、これらのマクロ関連擬似命令はasm88では受け付けられず、エラーとなります。

名前:

**MACRO .....** マクロ定義

形式:

```
<マクロ名>  MACRO [<パラメータ>[,<パラメータ>]*]
               <ステートメント列>
               [EXITM]
               <ステートメント列>
<マクロ名>] ENDM
```

機能:

マクロ定義を行います。指定されたマクロ名がすでに使用されているとき、以前の定義は失われ、再定義となります。マクロ名には空白文字と括弧"(", ")", "{", "}", "[", "]"とコロン":"を含まない任意の名前が使用できます。ENDM行のマクロ名指定は、マクロ定義がネストしたとき以外は、指定する必要はありません。また、パラメータの個数には制限はありません。

マクロ呼び出し時に、カンマ","で区切られた任意個数のアーギュメントを指定できます。この個数は必ずしも、マクロ定義時のパラメータの個数と等しい必要はありません。マクロ本体中にパラメータと同じ文字列が存在すれば、呼び出し時の対応するアーギュメント文字列と置き換えられます。対応するアーギュメントが存在しないときには空文字列と置き換えられます。また、アーギュメントに空文字列を指定することもできます。その場合には、アーギュメントに空白文字以外の文字を含まないものを指定します。例えば、あるマクロ"xmac"の呼び出し時に、

```
xmac 1,,2
```

のように指定すると、2番目のアーギュメントは空文字列となります。また、呼び出し時のアーギュメントの個数はsap88のシステムパラメータNARGおよびnargと置き換えられます。そのときには、空文字列アーギュメントも数えられます。

パラメータは、トークンとして独立したものばかりでなく、文字列中に現われてもアーギュメントと置き換えられます。

置き換えが起きすぎないようにするためには、パラメータに特殊記号を使うなどして適宜に回避してください。パラメータやアーギュメントに用いる特殊記号としてはカンマ","と括弧"(", ")", "{", "}", "[", "]"以外は全て使用できます。例えば、

```
sum  macro    c,d
      ld      a,[c]
      add     a,d
      ld      [c],a
      endm

      sum     total,#20
```

は、

```
1#20    a,[total]
a#20#20 a,#20
1#20    [total],a
```

になってしまいますので、そのようなときには、

```
sum macro    c,&d
    ld      a,[c]
    add     a,&d
    ld      [c],a
endm
```

のようにマクロ定義を行えば正しく置き換えられます。

なお、パラメータやアргументの前後の空白文字は捨てられますが、中の空白文字は有効となりますので、注意してください。

また、マクロ定義のマクロ本体中からもマクロ呼び出しが行えます。その場合には、マクロ呼び出しが起こったときに、その中のマクロの呼び出しを行います。すなわち、

```
maca macro    x,y
    add      x,y
endm

macb macro    x,y
    maca     x,y
endm

        macb    a,#2        add    a,#2

maca macro    x,y
    sub      x,y
endm

        macb    a,#2        sub    a,#2
```

となります。マクロ本体からのマクロ呼び出しは任意の深さまで行えます。ただし、呼び出しがループしてしまうような場合には、マクロ呼び出しは行われません。単純な例では、

```
add macro    x,y
    ld      a,x
    add     a,y
    ld      x,a
endm
```

このように、定義されているマクロを以下のようにマクロ呼び出しを行うと、

```
                                ld      a,b
add    b,#2                    add     a,#2
                                ld      b,a
```

と展開され、3行目の、

```
add      a,y
```

は、自分自身の呼び出しになりますので、マクロ呼び出しは起きず、"add"命令となります。もう少し複雑な例では、

```
maca macro    x,y
    macb     x,y
    macc     x,y
endm

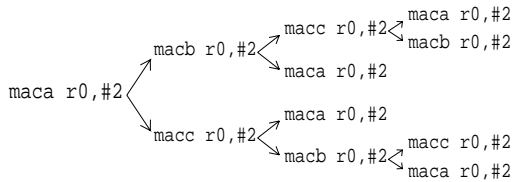
macb macro    x,y
    macc     x,y
    maca     x,y
endm
```

```

macc macro    x,y
        maca    x,y
        macb    x,y
    endm

```

となります。



マクロ本体中でIFC文を用い、条件アセンブルを行うときには、その判定はマクロの呼び出し時に行われます。そのとき、EXITM行が現われると、マクロ展開は中止され、呼び出しはそこで終了します。例えば、

```

xmac macro    x,y
    ...
    ifc      MODE == 2
        exitm
    endif
    ...
endm

```

では、

```
MODE set 2
      xmac #3, #4
```

というように呼び出されると、EXITM行でマクロの展開は終了しますが、

```
MODE set      1
          xmac  #3, #4
```

では、最後まで展開されます。

マクロ本体にはマクロ定義を含むこともできます。ただしその場合には、

```
x      macro
      ...
y      macro
      ...
z      macro
      ...
z      endm
      ...
y      endm
      ...
      endm
```

のように、ENDM行には対応するMACROのマクロ名が必要になります。上の場合、マクロ"x"が呼び出されたときにマクロ"y"の定義が行われます。なお、この場合でも一番外側(上の例では"x")のマクロ定義のENDM行にはマクロ名の指定は必要ありません。ネストの深さは任意まで可能です。

関連項目:

EQU, IFC, IFDEF, IFNDEF, IRP, IRPC, PURGE, SET

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**DEFINE** ..... 文字列マクロ定義

形式:

**DEFINE** <文字列マクロ名> [<置換文字列>]

機能:

文字列マクロ定義を行います。DEFINE文以降のソース中で文字列マクロ名と同じトークンは、IFDEF文とIFDEF文を除く全ての文の評価の先だって、指定された置換文字列にマクロ置換されます。置換文字列が指定されていない場合は空文字列に置換されます。また、文字列マクロ名はIFDEF文やIFDEF文での評価対象となります。

例:

```
define   XMAX      #128

        cp         a,XMAX

        cp         a,#128
```

関連項目:

IFDEF、IFDEF、UNDEF

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。



名前:

**LOCAL .....** ローカルラベルの定義

形式:

LOCAL [<ローカルラベル名>,<ローカルラベル名>]\*

機能:

ローカルラベルを宣言します。ローカルラベル名と同じトークンがマクロ定義内に現われると、展開するごとに自動生成される異なったラベル名にマクロ置換されます。ローカルラベル名の生成規則は、前置文字列"L"の後に0001から始まる4桁の数字が続くというものです。また、前置文字列はsap88起動時に指定することによって変更できます。

例:

```
mac1 macro
    local    x
    cp       a,#3
    jr       c,x
    ld       d,r0
x:
    endm
mac1
mac1

    cp       a,#3
    jr       c,L001
    ld       d,a
L001:
    cp       a,#3
    jr       c,L002
    ld       d,a
L002:
```

関連項目:

MACRO

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**PURGE** ..... マクロの抹消

形式:

**PURGE** [<マクロ名>]

機能:

それ以降、指定された名前のマクロ定義を抹消します。名前が指定されなかったときは、全てのマクロが抹消されます。定義されていないマクロ名を指定しても構いません。

例:

```
purge    add      ; マクロaddを抹消し、
add      ba, #10  ; add命令を用いる。
```

関連項目:

MACRO

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**UNDEF** ..... 文字列マクロの抹消

形式:

**UNDEF** <文字列マクロ名>

機能:

それ以降、指定された名前の文字列マクロ定義を抹消します。定義されていない文字列マクロ名を指定しても構いません。

例:

```
undef XMAX ; 文字列マクロXMAXを消去する
```

関連項目:

DEFINE、IFDEF、IFNDEF

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**IRP** ..... 文字列による繰り返し

形式:

```
IRP<パラメータ>,<アーギュメント>[,<アーギュメント>]*
<ステートメント列>
ENDR
```

機能:

パラメータにアーギュメントを左から順に代入しながら、アーギュメントの回数回、ENDR行までを繰り返し展開します。そのとき、IRP行からENDR行までの間にパラメータと同じ文字列が存在すれば、それは、そのときのアーギュメントの文字列に置き換えられます。

パラメータは、トークンとして独立したものばかりでなく、文字列中に現われてもアーギュメントと置き換えられます。置き換えが起きすぎないようにするためには、パラメータに特殊記号を使うなどして適宜に回避してください。パラメータやアーギュメントに用いる特殊記号としてはカンマ","と括弧"(",")","{","}"、"[","]"以外は全て使用できます。例えば、

```
irp    w,10,20,30
      dw    w
endr
```

は、

```
      d10    10
      d20    20
      d30    30
```

になってしまいますので、そのようなときには、

```
irp    &w,10,20,30
      dw    &w
endr
```

のようにすれば正しく置き換えられます。

なお、パラメータやアーギュメントの前後の空白文字は捨てられますが、中の空白文字は有効となりますので、注意してください。

IRP文、IRPC文、REPT文は任意の深さまで入れ子にできます。そのとき、ENDR行は内側のIRP/IRPC/REPT行と対応します。

例:

```
irp char,30,31,32,33,34,35,36,37,38,39
c_char: dw    charh
endr

c_30:   dw    30h
c_31:   dw    31h
c_32:   dw    32h
c_33:   dw    33h
c_34:   dw    34h
c_35:   dw    35h
c_36:   dw    36h
c_37:   dw    37h
c_38:   dw    38h
c_39:   dw    39h
```

関連項目:

IRPC、MACRO、REPT

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**IRPC** ..... 文字による繰り返し

形式:

```
IRPC <パラメータ>,<アークギュメント文字列>
      <ステートメント列>
ENDR
```

機能:

パラメータにアークギュメントの文字列を1文字ずつ左から順に代入しながら、アークギュメントの文字数回、ENDR行までを繰り返し展開します。そのとき、IRPC行からENDR行までの間にパラメータと同じ文字列が存在すれば、それは、そのときのアークギュメントの文字に置き換えられます。

パラメータは、トークンとして独立したものばかりでなく、文字列中に現われてもアークギュメントと置き換えられます。置き換えが起きすぎないようにするためには、パラメータに特殊記号を使うなどして適宜に回避してください。パラメータやアークギュメントに用いる特殊記号としてはカンマ","と括弧"(",")","{","}" ,"[" "]"以外は全て使用できます。例えば、

```
irpc w,abc
      dw      'w'
endr
```

は、

```
      da 'a'
      db 'b'
      dc 'c'
```

になってしまいますので、そのようなときには、

```
irpc &w,abc
      dw      '&w'
endr
```

のようにすれば正しく置き換えられます。

なお、パラメータやアークギュメントの前後の空白文字は捨てられますが、中の空白文字は有効となりますので、注意してください。

IRP文、IRPC文、REPT文は任意の深さまで入れ子にできます。そのとき、ENDR行は内側のIRP/IRPC/REPT行と対応します。

例:

```
irp  char,Hello, world!
      dw      'char'
endr
```

```
      dw      'H'
      dw      'e'
      dw      'l'
      dw      'l'
      dw      'o'
      dw      ', '
      dw      ' '
      dw      'w'
      dw      'o'
      dw      'r'
      dw      'l'
      dw      'd'
      dw      '!'
```

**関連項目**

IRPC、MACRO、REPT

**制限:**

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

**名前:****REPT** ..... 回数指定による繰り返し**形式:**

```
REPT <演算式>
    <ステートメント列>
ENDR
```

**機能:**

REPT行からENDR行までの間を、演算式の値の回数、繰り返し展開します。演算式中に未定義の名前があった場合は、その名前の値を0として評価します。

IRP文、IRPC文、REPT文は任意の深さまで入れ子にできます。そのとき、ENDR行は内側のIRP/IRPC/REPT行と対応します。

**例:**

```
rept    4                ;4ビットシフト
    sll    a
endr
```

**関連項目:**

EQU、IRP、IRPC、SET

**制限:**

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

### B.3.9 条件アセンブル擬似命令 [sap88 only]

条件アセンブル擬似命令は、条件式の評価結果、または名前の定義の有無によって指定範囲のアセンブルを行うかどうかを決定します。条件アセンブル擬似命令は以下のとおりです。

**IFC ~ ENDIF**  
**IFDEF ~ ENDIF**  
**IFNDEF ~ ENDIF**

- \* 本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。sap88により、アセンブルの対象となるステートメントのみが挿入されたソースファイルが出力されます。また、これらの条件アセンブル擬似命令もクロスアセンブラasm88では受け付けられず、エラーとなります。

名前:

**IFC** ..... 条件式による条件アセンブル

形式:

```
IFC   <条件式>
      <ステートメント列>[
ELSEC
      <ステートメント列>]
ENDIF
```

機能:

条件式を評価し、真ならば、IFC行に続く文をELSEC行またはENDIF行が現われるまでアセンブルの対象とします。偽であれば、IFC行に続く文はアセンブルの対象とはなりません。また、ELSEC行がある場合には、IFC行の条件式が偽であれば、IFC行に対応するELSEC行からENDIF行までの間がアセンブルの対象となります。真であれば、ELSEC行からENDIF行まではアセンブルの対象とはなりません。

IFC文、IFDEF文、IFNDEF文は任意の深さまで入れ子にできます。そのとき、ELSEC行とENDIF行は内側のIFC/IFDEF/IFNDEF行と対応します。

条件式には、次の3つの場合があります。

#### 1) <演算式>

演算式のみの場合、演算式の値が0でないかどうかを判定し、0でなければ真、0であれば偽とします。演算式中で未定義の名前があった場合は、その名前の値を0として評価します。例えば、

```
IFC ee
```

と、

```
IFC ee != 0
```

は等価になります。

#### 2) <演算式> <関係演算子> <演算式>

演算式同士の値を比較します。このとき、演算式中で未定義の名前があった場合は、その名前の値を0として評価します。

関係演算子には以下のものがあります。

```
==   左辺値と右辺値が等しければ真
!=   左辺値と右辺値が等しくなければ真
<    左辺値が右辺値よりも小さければ真
>    左辺値が右辺値よりも大きければ真
<=   左辺値が右辺値よりも小さいか、または等しければ真
>=   左辺値が右辺値よりも大きい、または等しければ真
```

## 3) [&lt;条件式&gt;] &lt;論理演算子&gt; &lt;条件式&gt;

複雑な条件式を論理演算子を使って表現できます。  
論理演算式には以下のものがあります。

単項演算子として

! 条件式が偽であれば真

二項演算子として

&& 左辺が真でかつ右辺も真ならば真

|| 左辺が真かまたは右辺が真ならば真

演算子の間の優先順位は、高い方から順に

丸括弧で囲まれた演算式または条件式>単項演算子>通常の演算式の演算子>関係演算子>&&>||

となります。丸括弧の内部では同様の優先順位になります。また、単項演算子とは、通常の演算式の単項演算子と論理演算子の!です。

また、演算式として

"文字列"

も使用できます。関係演算子の両辺にこの形が現われたときには文字列同士の比較となり、それ以外の場合には文字列の長さの値になります。

例:

```
table macro &1,&2
    ifc    narg == 1
        ifc !USE_DEFAULT || DEFAULT_SIZE < 64
            &1: db    0(64)
        elsec
            &1: db    0(DEFAULT_SIZE)
        endif
    elsec
        &1: db    0(&2)
    endif
endm
```

関連項目:

EQU、IFDEF、IFDEF、SET

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

名前:

**IFDEF .....** 名前が定義されているか否かによる条件アセンブル

形式:

```
IFDEF <名前>
    <ステートメント列>[
ELSEC
    <ステートメント列>]
ENDIF
```

機能:

名前がEQU文またはSET文で定義されているか、またはDEFINE文で定義されている文字列マクロであるならば、IFDEF行に続く文をELSEC行またはENDIF行が現われるまでアセンブルの対象とします。定義されていない場合は、IFDEF行に続く文はアセンブルの対象となりません。また、ELSEC行がある場合には、IFDEF行の名前が定義されていない場合は、IFDEF行に対応するELSEC行からENDIF行までの間がアセンブルの対象となります。定義されているときには、ELSEC行からENDIF行まではアセンブルの対象となりません。

IFC文、IFDEF文、IFNDEF文は任意の深さまで入れ子にできます。そのとき、ELSEC行とENDIF行は内側のIFC/IFDEF/IFNDEF行と対応します。

例:

```
ifdef EXTRA_MEMORY
stack_start equ 4000h
stack_size equ 1000h
elsec
stack_start equ 3800h
stack_size equ 800h
endif
```

関連項目:

DEFINE、EQU、IF、IFNDEF、SET

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。



名前:

**IFNDEF ....** 名前が定義されていないか否かによる条件アセンブル

形式:

```
IFNDEF  <名前>
        <ステートメント列>[
ELSEC
        <ステートメント列>]
ENDIF
```

機能:

名前がEQU文またはSET文で定義されていないか、DEFINE文で文字列マクロ名としても定義されていないならば、IFNDEF行に続く文をELSEC行またはENDIF行が現われるまでアセンブルの対象とします。定義されていれば、IFNDEF行に続く文はアセンブルの対象となりません。また、ELSEC行がある場合には、IFNDEF行の名前が定義されていれば、IFNDEF行に対応するELSEC行からENDIF行までの間アセンブルの対象となります。定義されていないときには、ELSEC行からENDIF行まではアセンブルの対象となりません。

IFC文、IFDEF文、IFNDEF文は任意の深さまで入れ子にできます。そのとき、ELSEC行とENDIF行は内側のIFC/IFDEF/IFNDEF行と対応します。

例:

```
ifndef SMALL_MEMORY
stack_start equ 3800h
stack_size  equ 800h
elsec
stack_start equ 4000h
stack_size  equ 1000h
endif
```

関連項目:

DEFINE、EQU、IF、IFNDEF、SET

制限:

本擬似命令は構造化プリプロセッサsap88でのみ使用可能です。クロスアセンブラasm88では受け付けられずエラーとなります。

### B.3.10 出力リスト制御擬似命令

出力リスト制御擬似命令はアセンブリリストを見やすくするために用いられ、以下の7種類があります。

**LINENO**  
**SUBTITLE**  
**SKIP**  
**NOSKIP**  
**LIST**  
**NOLIST**  
**EJECT**

---

名前:

**LINENO** ..... アセンブリリストファイルの行番号を変更

形式:

**LINENO** <数値式>

機能:

この命令は、アセンブリリストファイルの行番号を、強制的に<数値式>で設定されている値の次の行番号に変更します。行番号は65535まで変更可能で、その上限を越えると0から始まります。

例:

`lineno 99 ;行番号を100行から始める`

---

名前:

**SUBTITLE** ... アセンブリリストファイルへのサブタイトルの設定

形式:

**SUBTITLE** <文字列>

機能:

SUBTITLE命令は、リスト出力の4行目に任意の文字列をサブタイトルとして出力させるのに使用します。1ページ目以降では、現ページ内に現われたSUBTITLEは次のページのサブタイトルとして使われ、新たにSUBTITLEが現われるまで使われ続けます。

文字列は二重引用符で囲みます。

例:

`subtitle "asm88 Special function library"`

名前:

**SKIP ....** アセンブリリストファイルへの4バイトを越える初期化コード出力サプレス

形式:

SKIP

機能:

この命令が現われると、以降、ASCII、DB、DL、DWの各命令で、アセンブリリストファイルの1行を越えるような、すなわち、5バイトサイズ以上の初期化リストがあっても、アセンブリリストファイルには1行分のコードだけを出力し、アセンブリリストファイルに収まりきらない分のコード出力がサプレスされます。また、この機能を反転させるための命令NOSKIPがあります。デフォルトではSKIPが設定されています。

例:

```
noskip
  db      1,2,3,4,5,6,7,8,9,0
          ;16進コードは全てアセンブリリストファイルに出力

skip
  ascii "1234567890"
          ;ASCIIコードは1行分だけアセンブリリストファイルに出力
```

関連項目:

NOSKIP

名前:

**NOSKIP .....** アセンブリリストファイルへ初期化コードを全て出力

形式:

NOSKIP

機能:

この命令はアセンブリリストファイルへの4バイトを越えるコード出力をサプレスさせる命令SKIP(デフォルト)の機能を反転させるために使用します。この命令が現われると、以降、ASCII、DB、DL、DWの各命令に初期化コードが設定されていれば、そのコードが全てリストに出力されます。

例:

```
noskip
  db      1,2,3,4,5,6,7,8,9,0
          ;16進コードは全てアセンブリリストファイルに出力

skip
  ascii "1234567890"
          ;ASCIIコードは1行分だけアセンブリリストファイルに出力
```

関連項目:

SKIP

名前:

**LIST .....** アセンブリリストファイルの出力

形式:

LIST

機能:

この命令が現われると、以降、アセンブリリストファイルが出力されます。デフォルトではLISTが設定されています。

関連項目:

NOLIST

---

名前:

**NOLIST .....** アセンブリリストファイルの出力を禁止

形式:

NOLIST

機能:

この命令が現われると、以降、アセンブリリストファイルの出力が禁止されます。アセンブリリストファイルの出力を再開させたい場合、LIST命令を使用してください。なお、NOLISTでアセンブリリストファイルの出力が禁止されている場合にも、行番号は更新されています。

関連項目:

LIST

---

名前:

**EJECT .....** アセンブリリストファイルの改ページ

形式:

EJECT

機能:

この命令が現われると、自動改ページと同様にページヘッダなどを伴ってアセンブリリストファイルに改ページが挿入されます。この命令自身は、改ページされたページの1行目に表されます。

# Appendix C アセンブルツールリファレンス (サブツールチェーン)

## 説明の見方

各ソフトウェアツールの説明は以下に示す項目ごとにまとめられています。

- 1) プログラム名  
プログラムファイル名を示します。
- 2) 概要  
ソフトウェアツールの機能が説明されています。
- 3) 入出力ファイル  
実行フローおよび入出力ファイルが示されています。
- 4) 起動フォーマット

ソフトウェアツールの起動コマンドの形式が示されています。この形式には、コマンドラインの主な構成要素...ツール自身の名前、ツールが受け入れる全てのフラグが含まれています。ツールに無効なフラグや引数を渡したり、必要な引数を渡し忘れたりすると、コマンドは起動されません。

フラグは、区切り文字-と[ ]の中に名前を入れてリストされます。原則的にフラグはアルファベット順に現われます。値だけからなるフラグは、他の全てのフラグの後にリストされます。何らかの値を伴うフラグの場合、その値の種類も以下のどれかのコード(フラグ名の直後に与えられる)により示されます。

コード	値の種類
*	文字列
#	整数(ワードサイズ)
##	整数(ロングワードサイズ)
?	単一文字

#はワードサイズ(2バイト)の整数を表します。  
##はロングワードサイズ(4バイト)の整数を表します。整数は、0xまたは0Xで始まる場合は16進数として、0で始まる場合は8進数として、それ以外の場合は10進数として解釈されます。+または-符号を任意に先行させることができます。

同一フラグによって2回以上与えられ、値がスタックされるようなタイプのものは、値コードの後にキャレット^が続きます。たとえば、asm88は次の形式となっています。

```
asm88 -[all c l o* q RAM# ROM#  
sig# suf* x] [ドライブ名:] <files>□
```

asm88は、次のような10の異なるフラグを受け入れるということがわかります。

つまり、-RAM、-ROM、-sigにはワードサイズの整数値が与えられ、-all、-c、-l、-q、-xは値を持たず、-oと-sufには文字列が与えられます。ハイフン-(とりたてて指定がないかぎりハイフンが仮定される)が前置されるタイプのフラグは、ハイフンなしで表されていることに注意してください。個別に指定する場合、上で示したリスト中のRAM#は-RAM#として与えることになります。また、値を持たないフラグ等は、-clqのように先頭に指定するフラグのみにハイフン-を付け、残りのフラグを続けて指定することも可能です。非フラグ引数の位置と意味は、<と>で囲まれた語(上の例では<files>)で示されます。各メタ概念は、コマンドラインで与えられるゼロ個または1個以上の引数を表します。コマンドラインを入力するときは、メタ概念が表しているものを全てその行のその位置にタイプしてください。asm88の場合では、<files>で示される位置には1つ以上のファイル名を入力します。角括弧で囲まれたメタ概念は任意指定であり、省略または複数指定も可能です。

- 5) フラグ  
各フラグの機能が一覧できるようにまとめられています。そして必要に応じて、フラグに関する補足説明が続きます。
- 6) エラーメッセージ  
実行中に表示されるエラーメッセージの一覧です。
- 7) 戻り値  
実行を終了すると、各ツールは『成功』または『失敗』と呼ばれる2つの値のどちらかを返します。この項では、ツールがどちらか一方を返す条件が述べられています。一般に、成功した場合の戻り値というのは、そのツールが必要な全てのファイル処理を実行することができたことを表します。この戻り値は、バッチ処理を行う場合にツールの実行結果を評価する場合に用いられます。
- 8) 例  
ここでは、ソフトウェアツールの使用例を述べます。
- 9) 注意  
使用上の注意事項等が述べられています。

## C.1 構造化プリプロセッサ <sap88>

### プログラム名

**sap88.exe**

### 概要

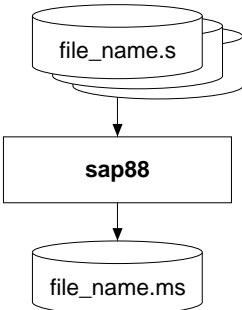
構造化プリプロセッサsap88は、クロスアセンブラasm88にマクロ機能を付加するためのプリプロセッサです。

sap88は、指定されたファイル名のS1C88アセンブリソースファイル中に含まれるマクロをasm88でアセンブル可能な形式へ展開し出力します。またこのとき、モジュール化されたS1C88アセンブリソースファイルのインクルードや条件アセンブル等の処理も行います。なお、ファイル名が指定されないときには、sap88は標準入力(コンソール)から読み込みます。

### 入出力ファイル

#### 実行フロー

構造化アセンブリ  
ソースファイル



アセンブリ  
ソースファイル

sap88の実行フロー

#### 入力ファイル

構造化アセンブリソースファイル: file\_name.s

EDLINなどのエディタで作成した構造化アセンブリソースファイルです。

#### 出力ファイル

アセンブリソースファイル: file\_name.ms

構造化アセンブリソースファイル内のマクロ等がasm88でアセンブル可能なS1C88の命令セットに展開され、出力されるファイルです。このファイルはasm88への入力ファイルでもあります。出力ファイルの拡張子は".ms"としてください。

### 起動フォーマット

**sap88 -[d\*^ l\* o\* q] [ドライブ名:] <file> □**

#### フラグ:

[ ]内の文字列はフラグを意味し、各フラグの説明は後述します。

#### ドライブ名:

入力ファイルが、カレントドライブ以外に存在する場合は、入力ファイル名の前にそのドライブ名を入力します。カレントドライブにある場合は省略できます。

#### file:

sap88に入力するファイル名を指定します。このファイルは大文字、小文字どちらでも入力できます。<file>が指定されないときには標準入力から読み込みます。

注: 構造化アセンブリソースファイルの拡張子は".s"としてください。

## フラグ

sap88は以下のフラグを受け付けます。  
フラグは小文字で入力してください。

機 能	フラグ	説 明
文字列マクロの定義	-d*^	入力ファイルの読み込みに先だって、文字列マクロを定義します。 *は <文字列マクロ>=<置換文字列> の書式を持ち、また置換文字列を指定せず、 <文字列マクロ名> のみ指定した場合、文字列マクロの定義のみが行われ、置換文字列は空文字列になります。-dフラグによる文字列マクロ定義は最大20個まで指定できます。
前置文字列指定	-l*	構造化制御文を展開するときに生成されるラベル名の前置文字列を指定します。デフォルトでは"l"が設定されています。
出力ファイルの作成	-o*	出力ファイル名を*にします。デフォルトでは標準出力が設定されています。
起動メッセージの削除	-q	構造化プリプロセッサの処理に関するメッセージを一切出力しません。

## エラーメッセージ

エラーメッセージ	意 味
unexpected EOF in ~	~の途中でファイルが終了した
can't include ~	~がインクルードできない
illegal ~	~が不正である
illegal define	define文が不正である
illegal expression at ~	式中~が不正である
illegal undef	undef文が不正である

## 戻り値

sap88は、入力ファイルに構文上の誤りがなければ、『成功』を返します。構文上の誤りがあれば、出力ファイルの内容が正しくても、『失敗』を返します。

## 例

構造化アセンブリソースファイルsample.sをアセンブリソースファイルsample.msに展開します。

```
C>sap88 -o sample.ms sample.s
```

## 注意

sap88は、マクロ文中のレジスタ名などが間違っているような場合でも、構文上正しければ、許されないオペランドが含まれていても展開します。これらの誤りはアセンブラasm88によって検出されることになります。

## C.2 クロスアセンブラ <asm88>

### プログラム名

**asm88.exe**

### 概要

クロスアセンブラasm88は、構造化プリプロセッサsap88にてマクロ等が展開されたアセンブリソースファイルをアセンブルし、機械語に変換します。asm88は、高速化等を考慮して機能が簡略化された高速アセンブラであり、マクロや条件アセンブル等の付加機能は全て他のユーティリティ(sap88)でカバーしています。asm88は、モジュール別開発のためのリロケータブルアセンブルに対応しています。

リロケータブルアセンブルでは、リンカlink88によって他のモジュールと連結するためのリロケータブルオブジェクトファイルが生成されます。

またasm88は、直接アセンブリソースファイルを入力することも可能であり、ソースプログラムはその場合、次の形式のフリーフォーマットで記述ができます。

ラベル:    ニーモニック    オペランド    ;コメント

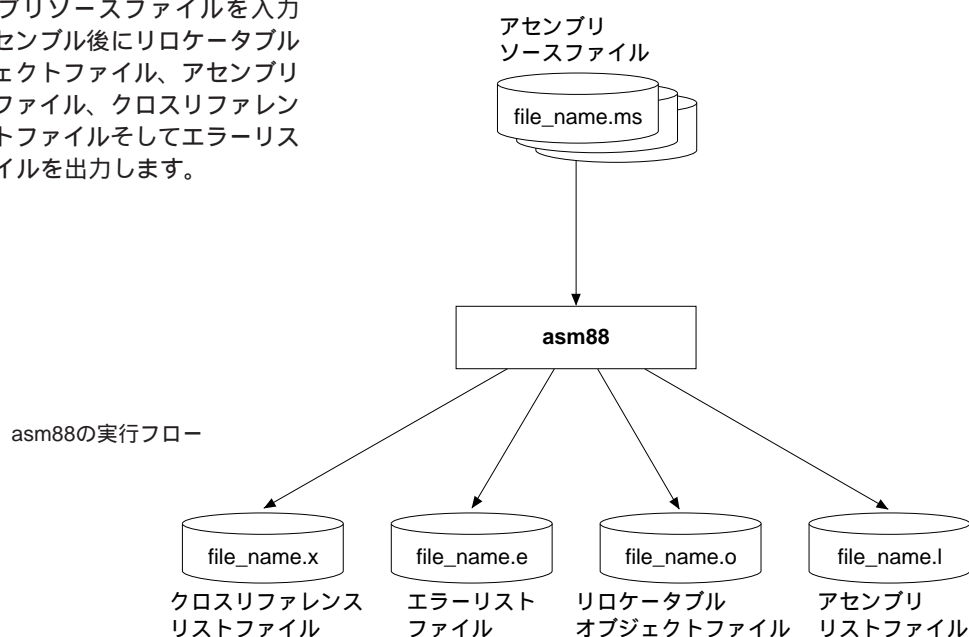
ここで":"はラベルの終わり、";"はコメントの始まりを示しています。これらのセパレータにより、自由にフォーマット可能です。

asm88はまた、プログラマーのためにアセンブリリスト、エラーリスト、クロスリファレンスリストの3種類のリストを出力します。アセンブリリストは行番号、アドレス、各ソースステートメントに対応するマシンコードで構成され、行番号は10進数、アドレスとマシンコードは16進数で出力されます。アセンブル時にエラーが発生した場合は、ソースファイル名、エラーの発生した行番号、エラーのレベル、そして英語のエラーメッセージで構成されるエラーリストファイルが作成されます。アセンブリリストファイルにもエラーの発生した行番号にマーク"\*"が付きます。また、クロスリファレンスによって、ファイル内シンボルの定義と参照の関係をたやすく把握できるよう考慮されています。これらは別々のファイルとして生成されますので、ファイルの管理がしやすくなっています。エラーが発生してもそれが致命的なエラーでない限り処理は続行されます。

### 入出力ファイル

#### 実行フロー

アセンブリソースファイルを入力し、アセンブル後にリロケータブルオブジェクトファイル、アセンブリリストファイル、クロスリファレンスリストファイルそしてエラーリストファイルを出力します。





### 入力ファイル

アセンブリソースファイル: file\_name.ms

sap88で生成されたアセンブリソースファイルです。

asm88は、デフォルトでは".ms"が拡張子として設定されています。拡張子はフラグで自由に変わりますが、必要でない限りは設定を変えないようにしてください。

### 出力ファイル

1. リロケートブルオブジェクトファイル: file\_name.o

asm88のリロケートブルアセンブルによってアセンブリソースファイルを再配置可能なS1C88の機械語に変換し、出力されるファイルです。また、このファイルはリンカlink88への入力ファイルでもあります。

2. アセンブリリストファイル: file\_name.l

アセンブルによって変換された機械語とそのアドレスが、各ソースステートメントに対応したリストとして出力されるファイルです。アドレスは、そのファイルのCODEセクションまたはDATAセクションの先頭を000000Hとした相対アドレスとして出力されます。起動時フラグによって生成を禁止することも可能です。

3. クロスリファレンスリストファイル: file\_name.x

シンボル定義と参照が行われているアドレスのリストです。起動時フラグによって生成を禁止することも可能です。

4. エラーリストファイル: file\_name.e

アセンブル時に発生したエラーのリストです。

## 起動フォーマット

```
asm88 -[all c l o* q RAM# ROM# sig# suf* x] [ドライブ名:] <files> 
```

### フラグ:

[ ]内の文字列はフラグを意味し、各フラグの説明は後述します。

### ドライブ名:

ソースファイルが、カレントドライブ以外に存在する場合は、ソースファイル名の前にそのドライブ名を入力します。カレントドライブにある場合は省略できます。

### files:





asm88に入力するファイル名を指定します。このソースファイルは大文字、小文字どちらでも入力でき、複数のソースファイル指定も可能です。<files>が指定されないとエラーとなります。

注: ソースファイル名は8文字以内です。また、拡張子".ms"は必ず入力してください。

## フラグ

asm88は以下のフラグを受け付けます。

-ROM#、-RAM#は大文字で、その他のフラグは小文字で入力してください。

機 能	フラグ	説 明
全てのシンボルを出力	-all	ローカルシンボルを含む全てのシンボルをシンボリックテーブルに出力します。デフォルトでは、グローバルシンボルと未定義シンボルのみが出力されます。
ソースプログラム内の 大(小)文字の区別	-c	入力ソースでの大文字、小文字を区別します。デフォルトでは大文字と小文字は区別されないで、ABCとabcは同一シンボルとして扱われます。このフラグを指定したときはCPU命令セットやレジスタ名は小文字で記述してください。
アセンブリリスト ファイルの作成禁止	-l	アセンブリリストファイルの作成を禁止します。デフォルトでは".l"の拡張子を持つアセンブリリストファイルが作成されます。
出力ファイルの作成	-o*	*という名前で出力ファイルを作成します。デフォルトでは、入力ファイル名のファイル名拡張子が".ms"のとき、このフラグに入力ファイル名の拡張子を".o"に変更したファイル名を設定するのと同じ働きをします。入力ファイル名のファイル名拡張子が".ms"以外の場合、デフォルトの出力ファイル名はxeqとなります。 例) sample.msの出力をout.oとしたい場合、以下のように指定します。 asm88 -o out.o sample.ms 
起動メッセージの削除	-q	アセンブル処理に関するメッセージを一切出力しません。
RAM容量の設定	-RAM#	バイト単位によるRAMエリアの容量を#で設定します。DATAセクションの合計サイズがこのフラグによる設定値を越えた場合、エラーが出力されます。 例) 内部RAM容量を2K(2048byte)としたい場合、以下のように設定します。 asm88 -RAM 2048 sample.ms 
ROM容量の設定	-ROM#	バイト単位によるROMエリアの容量を#で設定します。CODEセクションの合計サイズがこのフラグによる設定値を越えた場合、エラーが出力されます。 例) 内部ROM容量を16K(16384byte)としたい場合、以下のように設定します。 asm88 -ROM 16384 sample.ms 
シンボルの文字数設定	-sig#	#の値によりシンボルの有効文字数を設定することができます。 デフォルトでは15文字となっています。
入力ファイル拡張子の 変更	-suf*	入力ファイルの拡張子を*(セパレータは含まれない)に変更します。 デフォルトは".ms"。 例) 入力ソースファイル(sample.ms)の拡張子を".bs"に変えて入力したい場合は、以下のように指定します。 asm88 -suf bs sample.bs 
クロスリファレンスリス トファイルの作成禁止	-x	クロスリファレンスリストファイルの作成を禁止します。デフォルトでは".x"の拡張子を持つクロスリファレンスリストファイルが作成されます。

-oフラグ指定がなく1つ以上の<files>が指定され、かつ入力ファイル名のファイル名拡張子がデフォルトファイル名サフィックスのとき、asm88は-oに入力ファイル名の拡張子を".o"に変更したファイル名が指定されたのと同じ働きをします。そのため、

```
asm88 file1.ms file2.ms files3.ms
```

というコマンド入力で3つのオブジェクトファイルfile1.o、file2.o、file3.oが自動的に生成されます。<files>に複数ファイルを指定したときには、-oフラグは機能しませんので注意してください。

## エラーメッセージ

## 1. Fatalエラー

エラーメッセージ	意 味
can't create <file>	<file>が作成できない
can't open <file>	<file>がオープンできない
can't read tmp file	中間ファイルが読めない
can't write tmp file	中間ファイルが書けない
namelist full	名前のリストテーブルが満杯になった
no i/p file	入力ファイル指定がない
insufficient memory	メモリが不十分である
can't seek on vmem file	バーチャルメモリファイルのシークに失敗した
can't seek to end of vmem file	バーチャルメモリファイル終端に到達できない
no swappable page	スワップスペースがとれない
read error on vmem file	バーチャルメモリファイルの読み込みに失敗した
write error on vmem file	バーチャルメモリファイルの書き込みに失敗した

## 2. Severeエラー

エラーメッセージ	意 味
<numeric label> already defined	ニューメリックラベルが衝突している
<identifier> wrong type	不正な識別子が現われた
<token> expected	トークンが必要である
' missing	引用符のアンバランス
attempted division by zero	ゼロによる除算が行われようとしていた
attempt to redefine <identifier>	識別子が再定義されようとしている
constant expected	定数を期待していた
end expected	end命令がない
encountered too early end of line	行が途中で終了した
field overflow	確保する領域がオーバーフローしている
invalid branch address	ショートジャンプ命令のオペランドに外部定義シンボルが置かれた
invalid byte relocation	バイトリロケーションが無効である
invalid character	不正文字を使用している
invalid flag	フラグが無効である
invalid operand	オペランドが無効である
invalid relocation item	リロケーション項目が無効である
invalid register	レジスタが無効である
invalid register pair	レジスタの組み合わせが無効である
invalid symbol define	シンボル定義が無効である
invalid word relocation	ワードリロケーションが無効である
new origin incompatible with current psect	相対セクション内(リロケータブルモード)で絶対originが存在した
non terminated string	文字列の終端が見つからない
<identifier> not defined	未定義の識別子が現われた
missing numeric expression	数値式が欠如している
cars or jrs out of range	carsまたはjrsの飛び先が遠すぎる
carl or jrl out of range	carlまたはjrlの飛び先が遠すぎる
operand expected	オペランドがない
psect name required	セクション名の指定が必要である
phase error <identifier>	パス1とパス2でラベルのアドレスが異なった
CODE or DATA missing	領域確保擬似命令がない
ROM capacity overflow	ROM容量をオーバーした
RAM capacity overflow	RAM容量をオーバーした
relocation error in expression	式中にリロケーションエラーが現われた
<identifier> reserved word	<identifier>は予約語である

エラーメッセージ	意 味
syntax error <token> expected	トークン不足による文法エラー
syntax error <token> unexpected	過剰トークンによる文法エラー
syntax error - invalid identifier <identifier>	不正識別子による文法エラー
syntax error <token> invalid in expression	不正トークンによる文法エラー
system error <> <token>	不正トークンによるシステムエラー
unsupported instruction	未サポート命令が現われた
unsupported operand	未サポートオペランドが現われた

### 3. Warningエラー

エラーメッセージ	意 味
directive is ignored in relocatable mode	リロケートブルモードのため擬似命令がスキップされた
possibly missing relocatability	リロケートブル性が失われる恐れがある
constant overflow	名前を7桁以上定義した
expected operator	演算子がない (BOC, LOC, POD, LOD)

### 戻り値

asm88は、入力ファイルに構文上の誤りがなく、またパス2のエラーがなく全ての処理が正常に終了すると、『成功』を返します。

### 例

ファイルsample.msをリロケートブルアセンブルして同時にリストファイルsample.lを得ます。

```
A>asm88 sample.ms
```

### C.3 リンカ <link88>

#### プログラム名

**link88.exe**

#### 概要

link88は、S1C88のマルチセクションリロケータブルオブジェクトファイルを結合してアプソリュートオブジェクトファイルを生成します。このファイルは、ICEでデバッグを行うため、プログラムデータHEXファイルを得る、バイナリHEXコンバータhex88への入力ファイルとなります。また、リロケータブルアセンブルされたファイルのリンク後のアプソリュートシンボル情報生成rel88を実行するための入力ファイルにもなります。

link88プロセスの基本形式は、次のとおりです。

- 1) グローバルフラグが、link88全体のプロセスを制御します。
- 2) フラグとファイルの追加により、新しいICODEセクションおよびDATAセクションを定義します。
- 3) セクションをリロケートして、物理的メモリの任意の場所に再配置したり、適当な記憶域境界で交互に『積み重ね(連続化)』することができます。
- 4) 各オブジェクトファイル入力は、現在のCODEセクションおよびDATAセクションに影響します。
- 5) 最終出力は、ヘッダ、その後に(名前を付けられた順に)全CODEセクション、全DATAセクション、シンボリックテーブル、全CODEセクションおよび全DATAセクションのリロケーションストリームと続きます。このような出力の各構成要素は、前述のグローバルフラグを使うことにより制御されています。
- 6) 全てのセクションはリンカ出力では連続しているのですが、メモリ内の特定の場所で実行するためには、断片を適切な物理的位置に書き出すためバイナリHEXコンバータhex88を使う必要があります。

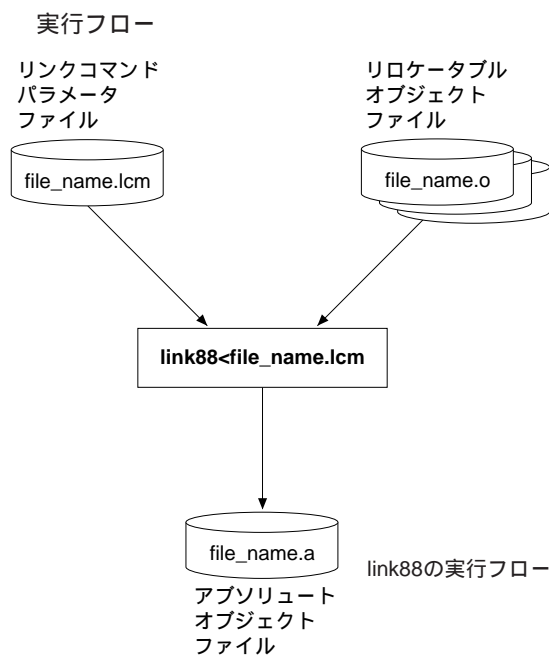
S1C88では24ビット幅のアドレス空間(最大16Mバイト)を持ち、最上位の8ビットをコードバンクレジスタ(CB)やエクスパンドページレジスタ(EP、XP、YP)などのレジスタで管理することによって、同一アドレス空間を32Kバイトのバンク(コード部)または64Kバイトのページ(データ部)単位に分割し、その範囲内でのアクセスパフォーマンスの向上が図られています。これらのレジスタの内容を書き換えることによって、任意のバンクまたはページから任意のバンクまたはページをアクセスすることも可能で、大きなプログラムやデータベースなども容易に管理できるようになっています。ただし、バンクやページがクロスしても、レジスタの自動更新は行われませんので、16Mバイトのアドレス空間をリニアに記述するようなロードモジュールのイメージを作成することはできません。

S1C88が管理できるアドレス空間の任意の物理アドレスに配置させるロードイメージを作成するために、リロケータブルオブジェクトのリンクにはマルチセクション方式を採用しています。

これは、『全アドレス空間を64Kバイト(ページ)または32Kバイト(バンク)単位の任意のセクションに分割し、メモリ配置に必要なアドレス情報を各セクション単位に与えることによって全てのアドレス情報を解決させる』手法です。

この手法では、大きさが64Kバイト(ページ)または32Kバイト(バンク)を越えるような連続したデータオブジェクトを1個のセクションとして作成することは許されませんので、アセンブル単位のモジュールに含まれるCODEセクションの場合はその合計サイズが32Kバイトを、DATAセクションの場合はその合計サイズが64Kバイトを、それぞれ越えることはできないという制約が生じます。この制約はCPUの持つアドレス的な制約をそのまま反映しており、仮にアセンブル時に生成データのオーバーフローに関する診断を見落としても、リンク時に再び診断が行われることとなります。ただし、デフォルトでは、64Kバイトを越えた場合にエラーを出力しますが、32Kバイトを越えた場合にはエラーを出力しません。よって、32Kバイトを越える場合の判定にはフラグを必ず付ける必要があります。

## 入出力ファイル



### 入力ファイル

1. リロケートブルオブジェクトファイル:  
file\_name.o  
クロスアセンブラasm88をリロケートブルアセンブルすることにより、出力される再配置可能なS1C88の機械語ファイルです。
2. リンクコマンドパラメータファイル:  
file\_name.lcm  
お客さまが直接記述されたリンクコマンドパラメータファイルです。

### 出力ファイル

アブソリュートオブジェクトファイル:  
file\_name.a  
link88によって生成されたマルチセクションオブジェクトファイルです。

注: マルチセクションオブジェクトファイルのフォーマットはグローバルヘッダ、セクションディスクリプタ、全てのCODEセクション部のオブジェクト、全てのDATAセクション部のオブジェクト、全てのDEBUGセクション部のオブジェクト、全てのZPAG部のオブジェクト、シンボリックテーブル、デバッグシンボリックテーブル、全てのリロケーション情報で構成されるアブソリュートオブジェクトのイメージです。

## 起動フォーマット

```
link88 -[c cd +dead max## o* q] <sections>
```

ここで、<sections>は下記のものの1個以上の出現です。

```
-[+code +data m## p##] [ドライブ名:]
```

### フラグ:

[ ]内の文字列はフラグを意味し、初めの[ ]内のフラグはグローバルフラグ、次の[ ]内のフラグはローカルフラグを表します。

### ドライブ名:

リロケートブルオブジェクトファイルまたはライブラリが、カレントドライブ以外に存在する場合は、これらのファイル名の前にそのドライブ名を入力します。カレントドライブにある場合は省略できます。

注: リロケートブルオブジェクトファイルの拡張子は".o"としてください。

## フラグ

link88は以下のフラグを受け付けます。  
フラグは小文字で入力してください。

### 1. グローバルフラグ

機 能	フラグ	説 明
シンボルの大(小)文字の 区別	-c	リロケータブルオブジェクトファイルのシンボルの大文字と小文字を区別します。 デフォルトでは大文字と小文字は区別されないで、ABCとabcは同一シンボル として扱われます。
DATA コード部の削除	-cd	DATAセクション向けのコード部を出力しません。-cdは、共用ライブラリ用に アドレスを指定するなどのため、シンボル値だけを定義するモジュールを作る ために使用します。
未定義シンボルのリスト	+dead	枯木シンボル、すなわち、定義されてはいるが絶対に参照されないシンボルの リストをCRT上に出力します。
セクション最大サイズの 設定	-max##	セクションの最大サイズを##バイトにします。 デフォルト値はFFFFFFFH(16Mバイト)です。この値は、セクションが結合され るときに使われ、この値を越えた場合、エラーとなります。
出力ファイル名の設定	-o*	出力モジュールをファイル*に書き込みます。デフォルトでは、出力ファイル名 がxeqとなります。
起動メッセージの削除	-q	リンク処理に関するメッセージを一切出力しません。

link88の引数となるフラグとファイルのリストは、link88にコマンドラインで引数が渡されない場合には、標準入力から渡されます。また、コマンドラインの引数のリストに-(ハイフン)が初めて出現すると、標準入力が引数リスト内に組み込まれ、"-と入れ換わります。それ以降の"-の出現は無視されます。指定された<files>は、その順にリンクされます。

### 2. ローカルフラグ

- ・ セクションごとのフラグ

機 能	フラグ	説 明
CODEセクションの開始	+code	新しいCODEセクションを開始して、そのセクションのためのローカルフラグ を処理します。
DATAセクションの開始	+data	新しいDATAセクションを開始して、そのセクションのためのローカルフラグ を処理します。

指定された形式の新しいセクションは、その形式の最後のセクションがゼロサイズの場合は、実際には作成されません。しかし、新しいローカルフラグは処理され、前の値をオーバーライト(上書き)します。この2つのフラグは、以降のローカルフラグを適切に処理し、かつどのフラグを適用するかを決めるために、ローカルフラグの集合のすぐ前に先行させなければなりません。

- ・ +codeおよび+dataと共にだけ使われるフラグ

機 能	フラグ	説 明
個別セクションの サイズ設定	-m##	個別セクションの最大サイズを##バイトにします。 デフォルトサイズは、CODEセクション: 8000H、DATAセクション: 10000H となり、これらの設定値を越えるとエラーとなります。
物理アドレスのセット	-p##	セクションの物理アドレスの先頭を##にセットします。



## エラーメッセージ

エラーメッセージ	意 味
bad file format: 'FILE NAME'	入力ファイル'FILE NAME'のフォーマットが不正である
bad relocation item	long integerタイプのリロケーション情報があった
bad symbol number: 'NUMBER'	'NUMBER'が不正なシンボルコードとして検出された
can't create 'FILE NAME'	ファイル'FILE NAME'が作成できない
can't create tmp file	中間ファイルが作成できない
can't open: 'FILE NAME'	入力ファイル'FILE NAME'がオープンできない
can't read binary header: 'FILE NAME'	ファイル'FILE NAME'のヘッダ部が読み込めない
can't read file header: 'FILE NAME'	ファイル'FILE NAME'の最初の2バイトが読み込めない
can't read symbol table: 'FILE NAME'	ファイル'FILE NAME'からシンボルテーブルが読み込めない
can't read tmp file	中間ファイルが読めない
can't write output file	出力ファイルに書き込みができない
can't write tmp file	中間ファイルに書き込みができない
field overflow	carsまたはjrsの飛び先が遠すぎる
inquiry phase error: 'SYMBOL NAME'	'SYMBOL NAME'のシンボル値がパス1とパス2で異なった
link: early EOF in pass2	パス2の最中に予想外のEOFが検出された
multiply defined 'SYMBOL NAME'	'SYMBOL NAME'が多重定義されている
no object files	入力オブジェクトファイルが存在しない
no relocation bits: 'FILE NAME'	ファイル'FILE NAME'に対応するリロケーション情報がサプレスされている
'SECTION NAME' overflow	セクション名'SECTION NAME'に属するセクションとサイズが上限値を上回った
phase error: 'SYMBOL NAME'	'SYMBOL NAME'のシンボル値がパス1とパス2で異なった
previous reference blocked: 'SYMBOL NAME' range error	リロケーションのビット幅に関する情報がミスマッチを起こしている
read error in pass2	パス2の最中に読み込みエラーが発生した
undefined 'SYMBOL NAME'	シンボル'SYMBOL NAME'がどこにも定義されていない

## 戻り値

link88は、エラーメッセージが標準出力に出力されない場合、すなわち、未定義シンボルが残らず、かつ全ての読み込みおよび書き込みが成功した場合、『成功』を返します。そうでない場合、『失敗』を返します。

## 例

sample.oをlink88の標準入力でリンクします。

```
A>link88
-o c88xxx.a +code -p0x100 +data -p0x8000
sample.o
^Z
A>
A>link88 < sample.lcm
```



## C.4 シンボル情報生成ユーティリティ <rel88>

### プログラム名

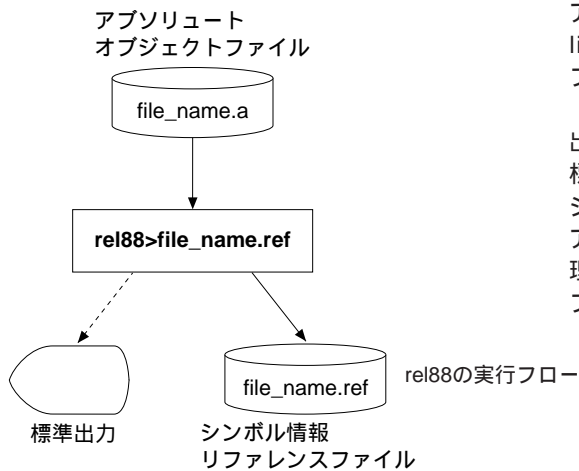
**rel88.exe**

### 概要

rel88は、S1C88のマルチセクション形式のリロケータブルオブジェクトの検査をします。この対象となるファイルは、クロスアセンブラasm88からのリロケータブルオブジェクトファイル、link88から出力されたアブソリュートオブジェクトファイルです。rel88は、オブジェクトファイルの大きさと構成を検査したり、link88から出力されたアブソリュートオブジェクトファイルのシンボル情報を出力するために使うことができます。

### 入出力ファイル

#### 実行フロー



#### 入力ファイル

アブソリュートオブジェクトファイル: file\_name.a  
link88にて生成されたアブソリュートオブジェクト  
ファイルを入力します。

#### 出力ファイル

標準出力または  
シンボル情報リファレンスファイル: file\_name.ref  
アブソリュートオブジェクトファイルからrel88は物  
理アドレスに配置されたシンボル情報リファレンス  
ファイルを出力します。

### 起動フォーマット

**rel88 -[a +dec d g +in +sec v] [ドライブ名:] <files> **

#### フラグ:

[ ]内の文字列はフラグを意味し、各フラグの説明は後述します。

#### ドライブ名:

入力ファイルが、カレントドライブ以外に存在する場合は、入力ファイル名の前にそのドライブ名を入力します。カレントドライブにある場合は省略できます。

#### files:

rel88にを入力するファイル名を指定します。このファイルは大文字、小文字どちらでも入力でき、複数のファイル指定も可能です。<files>が指定されないとエラーメッセージが出力されます。

## フラグ

rel88は以下のフラグを受け付けます。  
フラグは小文字で入力してください。

機 能	フラグ	説 明
シンボル名のソート	-a	出力をシンボル名のアルファベット順にソートします。
10進数出力	+dec	シンボル値とセクションサイズを10進数で出力します。デフォルトは、16進数。
定義済みシンボル出力	-d	各ファイル内の全ての定義済みシンボルを1行に1個ずつ出力します。 各行には、シンボルの値、値が何に関係するかを表す『リロケーションコード』、およびシンボル名が入っています。値は、S1C88で整数を表すために必要な桁数で出力されます。出力におけるリロケーションコードの意味は、次のとおりです。 <ul style="list-style-type: none"> <li>・ CODE相対を示すC</li> <li>・ DATA相対を示すD</li> <li>・ 絶対(リロケータブルでない)を示すA</li> <li>・ rel88が認識できないものを示す?</li> </ul> 小文字の英字は、ローカルシンボルを表すために使われます。大文字は、グローバルシンボル用に使われます。
グローバルシンボルのみ出力	-g	グローバルシンボルだけを出力します。
標準入力	+in	<files>を標準入力から取り、コマンド行に追加します。 入力リダイレクトも可能で、多くのファイルを指定するときに用います。
マルチセクションの物理アドレス、サイズ	+sec	マルチセクションオブジェクトファイルの各セクションの物理アドレス、サイズを出力します。
シンボルの値によるソート	-v	セクション内をシンボルの値でソートします。前述の-dフラグが暗黙的に指定されます。同じ値を持つシンボルは、アルファベット順にソートされます。絶対(リロケータブルでない)シンボルがまず表示され、CODE相対のシンボル、DATA相対のシンボルと続きます。

<files>はゼロ個または1個以上のファイルで、その形式はマルチセクション形式でなければなりません。2個以上のファイルが指定されると、各々のファイルまたはモジュールの名前が、それに関して出力される情報に先行します。各々の名前の後には、コロンと改行が続きます。<files>の指定がない場合、または "-" がコマンドラインにある場合、xeqが入力ファイルとして使われます。

## エラーメッセージ

エラーメッセージ	意 味
can't read binary header	マジック番号とコンフィグレーションバイトを除くオブジェクトヘッダ部の読み出しに失敗した
can't read header	オブジェクトヘッダ部の最初の2バイト(マジック番号とコンフィグレーションバイト)の読み出しに失敗した
can't read symbol table	オブジェクト内のシンボリックテーブルの読み出しに失敗した

## 戻り値

rel88は、診断メッセージが作成されない場合(すなわち、全ての読み込みが成功かつ全てのファイル形式が有効である場合)、『成功』を返します。

## 例

---

モジュール内の全てのシンボルのアルファベット順のリストを16進数で得ます。

```
C>rel88 -a alloc.o
0x0074C _alloc
0x0000D _exit
0x01feC _free
0x00beC _nalloc
0x0000D _sbreak
0x0000D _write
```

## 注意

---

オブジェクト内にシンボルが存在しないとき、またはローカルシンボルだけの場合、rel88は"no memory"のメッセージを出力します。

しかしローカルシンボルは、asm88の-allフラグ(全てのシンボルを出力)を設定することによりシンボリックテーブルに登録されます。

このように全てのシンボルを参照したい場合は、asm88の-allフラグを設定してください。

### C.5 シンボリックテーブルファイル生成ユーティリティ <sym88>

#### プログラム名

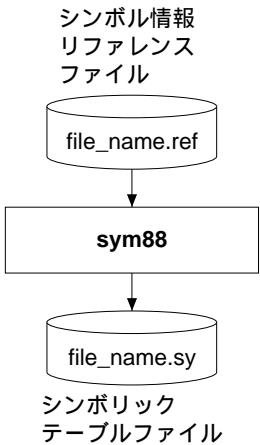
sym88.exe

#### 概要

sym88は、シンボル情報生成ユーティリティrel88からファイルリダイレクトによって生成された、アブソリュートオブジェクトファイルに対応したシンボル情報ファイル(file\_name.ref)をICE上で参照可能なシンボリックテーブルファイル(file\_name.sy)にフォーマット変換するユーティリティです。これによって、リロケータブルアセンブルされたプログラムおよび上記のツールによって作成されたシンボリックテーブルファイルをICE上にロードし、シンボリックデバッグを実行することが可能となります。

#### 入出力ファイル

##### 実行フロー



sym88の実行フロー


##### 入力ファイル

シンボル情報リファレンスファイル: file\_name.ref  
rel88にて生成されたシンボル情報リファレンスファイルを入力します。

##### 出力ファイル

シンボリックテーブルファイル: file\_name.sy  
sym88は、シンボル情報リファレンスファイルをICE上にロード可能なフォーマットに変換し、シンボリックテーブルファイルを出力します。

#### 起動フォーマット

sym88 <file> 

file:  
sym88に入力するシンボル情報ファイル(.ref)を指定します。  
このファイルは大文字、小文字どちらでも入力できます。  
<file>が指定されないとエラーメッセージが出力されます。

#### エラーメッセージ

エラーメッセージ	意 味
No Input File	入力ファイル".ref"が指定されていない

---

## 戻り値

---

sym88は、入力ファイルに誤りがなく出力ファイルが生成された場合、『成功』を返します。入力ファイルおよび内部生成ファイルに誤りがあれば、『失敗』を返します。

---

## 例

---

シンボル情報リファレンスファイルsample.refをシンボリックテーブルファイルsample.syに変換します。

```
A:¥>sym88 sample.ref
```

---

## 注意

---

- 1 sym88を起動させる場合、入力ファイルのドライブ、ディレクトリの設定は不可能です。このため、必ず入力ファイルの存在するディレクトリ上でsym88を実行してください。
- 2 sym88は入力ファイルに対し、入力フォーマットチェックを行っていません。このため、sym88に入力するシンボル情報リファレンスファイルは、必ずシンボル情報生成ユーティリティrel88で、以下のフラグのみを使用し、作成してください。

```
A:¥>rel88 -v +sec sample.a>sample.ref
```

## C.6 バイナリ/HEXコンバータ <hex88>

### プログラム名

**hex88.exe**

### 概要

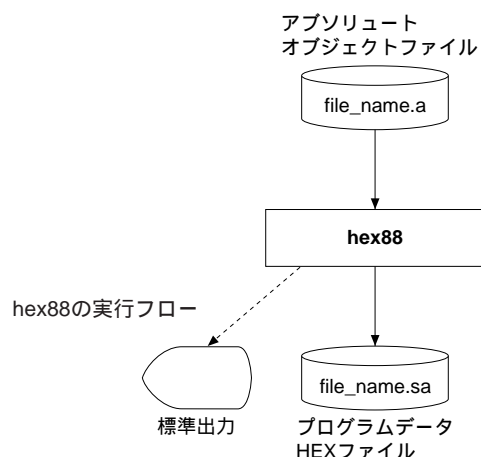
hex88はlink88により生成されたアブソリュートオブジェクトファイルを16進のデータ変換形式(プログラムデータHEXファイル)に変換します。形式としては、モトローラSレコード形式を採用しています。アブソリュートオブジェクトファイルは<ifile>から読み込まれます。<ifile>が与えられない場合、あるいは与えられたファイル名が-(ハイフン)である場合、ファイルxeqが読み込まれます。

また、S1C88は最大16Mバイトのアドレス空間(000000～FFFFFFH)を持っていますので、モトローラSレコード形式の中でもアドレスが3バイトまでデータ変換可能なS2フォーマットを使用しています。

### 入出力ファイル

#### 実行フロー

hex88は、リンカ(link88)から出力されたアブソリュートオブジェクトファイルを入力し、16進形式のプログラムデータHEXファイルに変換するツールです。以下に実行フローを示します。



#### 入力ファイル

アブソリュートオブジェクトファイル: file\_name.a  
hex88に入力するファイルは、リンカから出力されるアブソリュートオブジェクトファイルです。

#### 出力ファイル

標準出力または  
プログラムデータHEXファイル: file\_name.sa  
hex88によって、アブソリュートオブジェクトを未使用領域FF詰めユーティリティfil88XXXに入力可能なファイル(ASCIIファイル)にフォーマット変換します。

### 起動フォーマット

**hex88 -[o\*] [ドライブ名:] <ifile>**

#### フラグ:

[ ]内の文字列はフラグを意味し、各フラグの説明は後述します。

#### ドライブ名:

アブソリュートオブジェクトファイルが、カレントドライブ以外に存在する場合は、このファイル名の前にそのドライブ名を入力します。カレントドライブにある場合は省略できます。

#### ifile:

hex88に入力するアブソリュートオブジェクトファイル名を指定します。このファイル名は大文字、小文字どちらでも入力できます。<ifile>が与えられない場合、あるいは与えられたファイル名が-(ハイフン)である場合、ファイルxeqが読み込まれます。

注: アブソリュートオブジェクトファイルの拡張子は".a"としてください。

## フラグ

hex88は以下のフラグを受け付けます。  
フラグは小文字で入力してください。

機 能	フラグ	説 明
出力ファイル指定	-o*	出力モジュールをファイル*に書き込みます。 デフォルトは標準出力に出力します。(hex88固定設定フラグ)

## エラーメッセージ

エラーメッセージ	意 味
bad file format	入力ファイルのフォーマットが不正である
can't read <input file>	入力ファイル<input file>の読み出しに失敗した
can't write <output file>	出力ファイル<output file>への書き込みに失敗した

## 戻り値

hex88はエラーメッセージがプリントされなければ、すなわち全部のレコードに意味があり、全ての読み込みおよび書き込みが成功した場合『成功』を返します。そうでない場合、『失敗』を返します。

## 例

アブソリュートオブジェクトファイルsample.aをモトローラS2フォーマット形式のプログラムデータHEXファイルに変換します。

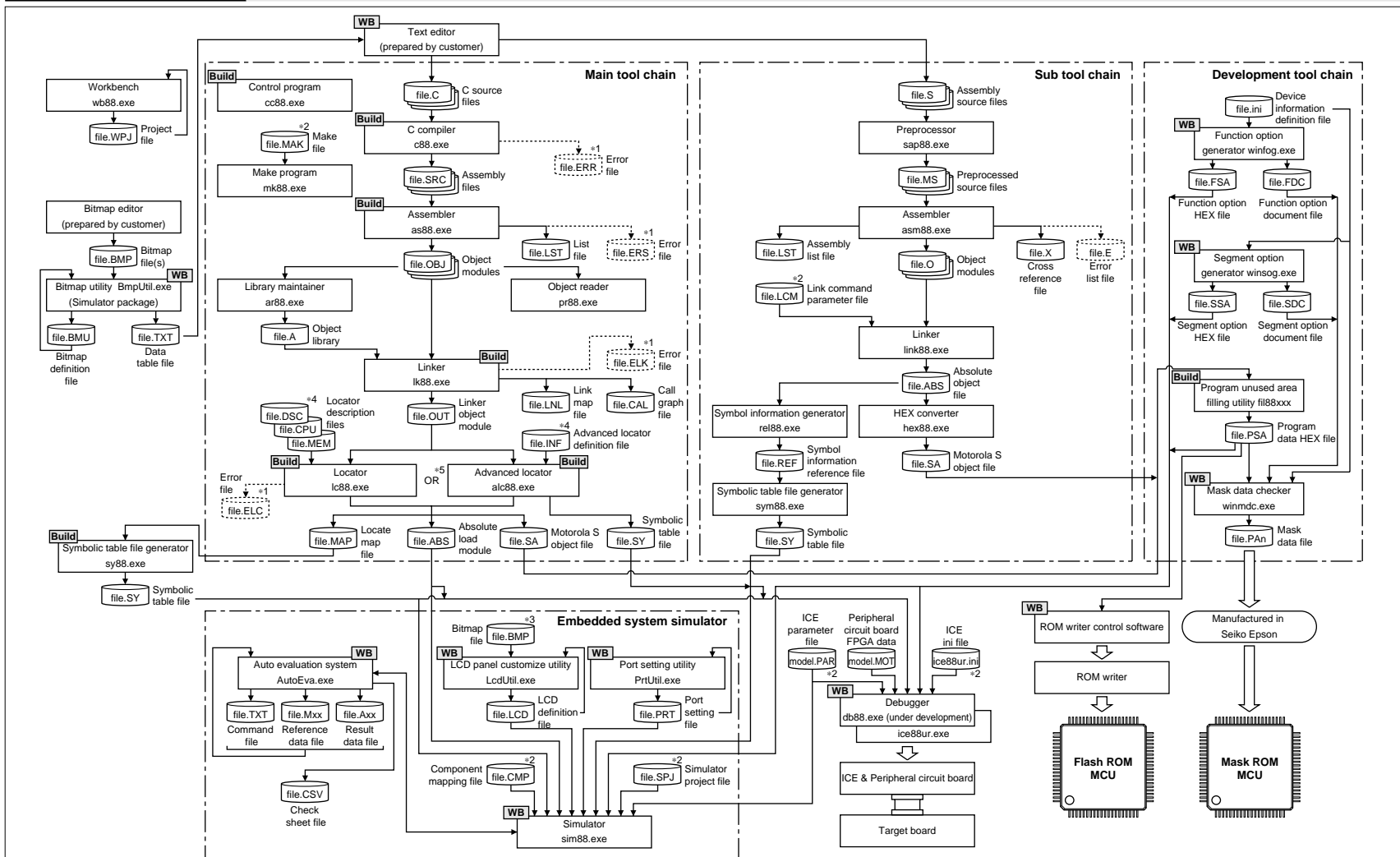
```
A>hex88 -o sample.sa sample.a
```





S1C88 Family Development Tools

# Quick Reference



**WB:** ワークベンチwb88から起動可能。 **Build:** ビルド時にワークベンチwb88が自動実行。

\*1: エラーファイルが生成された場合、wb88はその内容をメッセージビュー内にタグジャンプ機能付きで表示します。 \*2: テキストエディタで作成。 \*3: ビットマップエディタで作成。 \*4: wb88のセクションエディタ(またはテキストエディタ)で作成。 \*5: wb88で選択。

## 概要

Windows GUIベースのアプリケーションで、統合開発環境を提供します。指定エディタによるソースの作成や編集、ファイルの選択、ツールのオプション選択と起動などが、Windowsの基本操作のみで可能となります。

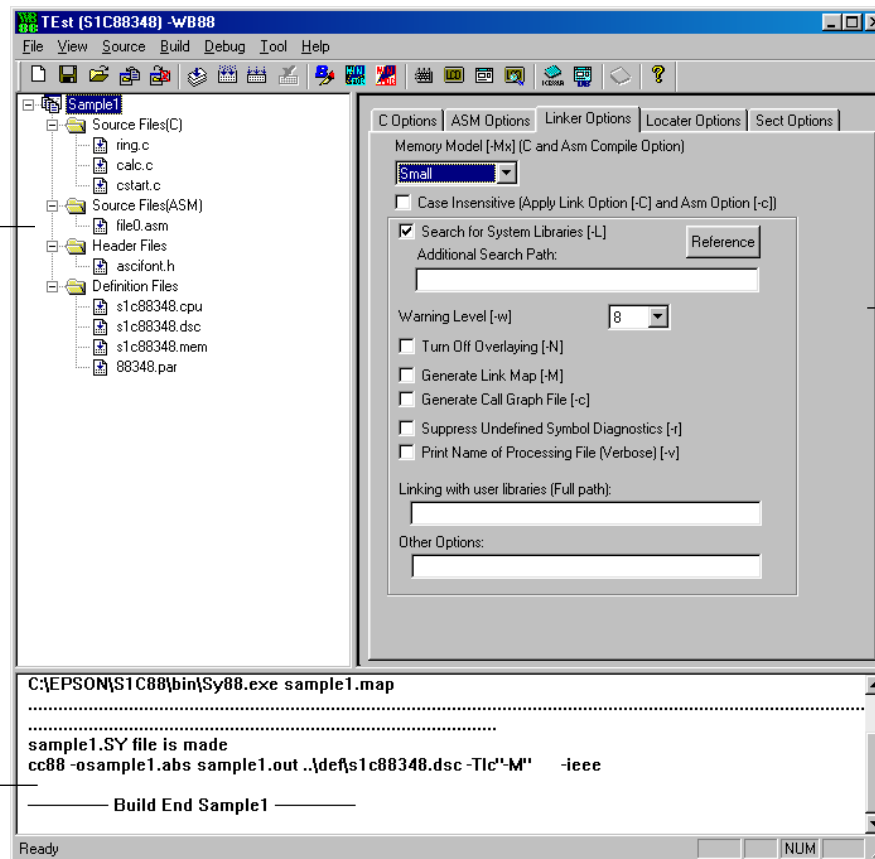
## ウィンドウ

### プロジェクトビュー

現在開いているワークスペースフォルダとプロジェクト中のユーザが編集可能なファイルを、Windowsエクスプローラと同様に表示します。リストされているソースファイルのアイコンをダブルクリックすると、指定のエディタがそのファイルを開き、編集可能となります。

### メッセージビュー

ビルドやコンパイル等の処理で実行されるツールが出力するメッセージを表示します。ここに表示されるソース行番号を含む文法エラーなどのエラーメッセージをダブルクリックすると、エラーが発生したソースファイルが指定のエディタにより開かれ、該当する行を表示します（wb88で実行可能なタグジャンプ機能を持つエディタを使用している場合のみ）。



### オプションビュー









Cコンパイラ、アセンブラ、リンカ、ロケータのオプション、およびセグメントエディタを表示します。ここで、オプションの選択も行います。タグをクリックしてツールを選択する以外に、プロジェクトビュー内のノードあるいはファイルの選択に従って表示が切り替わります。

## ボタン

## ツールバー

-  **[New Project]ボタン**  
プロジェクトを新規作成します。
-  **[Save Project]ボタン**  
編集中のプロジェクトをファイルにセーブします。ファイルは上書きされます。  
このボタンは、プロジェクトを開いていない場合は無効になります。
-  **[Insert a file]ボタン**  
プロジェクトにソース/ヘッダファイルを追加します。  
このボタンは、プロジェクトを開いていない場合は無効になります。
-  **[Remove a file]ボタン**  
選択されているファイルをプロジェクトから削除します。
-  **[Open]ボタン**  
ファイルを選択するダイアログボックスが表示されます。  
ソースファイルやヘッダファイルを選択した場合は、指定のエディタが起動してファイルを開きます。
-  **[Compile/Assemble]ボタン**  
プロジェクトビュー内で選択されているファイルを、ソースの種類に応じてコンパイルまたはアセンブルします。
-  **[Build]ボタン**  
現在開いているプロジェクトのmake処理を行い、実行形式オブジェクトをビルドします。
-  **[Rebuild]ボタン**  
現在開いているプロジェクトのビルドを行います。ファイルの更新状況にかかわらず、すべてのソースのコンパイル/アセンブルから処理されます。
-  **[Stop Build]ボタン**  
実行中のビルド処理を中止します。
-  **[BMPUtil]ボタン**  
ビットマップユーティリティBmpUtilを起動します。
-  **[WinFOG]ボタン**  
ファンクションオプションジェネレータWinfogを起動します。
-  **[WinMDC]ボタン**  
マスクデータチェッカWinmdcを起動します。

## ツールバー

-  **[PrtUtil]ボタン**  
ポート設定ユーティリティPrtUtilを起動します。
-  **[LCDUtil]ボタン**  
LCDパネルカスタマイズユーティリティLCDUtilを起動します。
-  **[Sim88]ボタン**  
シミュレータSim88を起動します。
-  **[AutoEva]ボタン**  
自動評価システムAutoEvaを起動します。
-  **[ICE88UR]ボタン**  
ice88urデバッガを起動します。
-  **[DB88]ボタン**  
db88デバッガを起動します。
-  **[ROM Writer]ボタン**  
オンボードROMライタコントロールソフトウェアを起動します。
-  **[About]ボタン**  
wb88のバージョンを表示します。

## メニュー

## [File]メニュー

## File(F)

- New
- Open Ctrl+O
- Open Workspace
- Save Workspace
- Close Workspace
- 1 C:\EPSON\...\src\file0.asm
- 2 c:\epson\s1c88\test\src\ring.c
- 3 c:\epson\...\test\src\cstart.c
- 4 c:\epson\s1c88\test\src\calc.c
- 5 Test.wpj
- Exit(X)

このメニューにリストされているファイル名は最近開いたソースとプロジェクトファイルです。ここからの選択でも、そのファイルを開くことができます。

## New - C Source File

Cソースファイルを新規作成します。  
(エディタを起動)

## New - Asm Source File

アセンブリソースファイルを新規作成します。  
(エディタを起動)

## New - Header File

ヘッダファイルを新規作成します。  
(エディタを起動)

## New - Project

プロジェクトを新規作成します。

## Open ([Ctrl]+[O])

ソースファイル、ヘッダファイルを開きます。  
(エディタを起動)  
プロジェクトを開くこともできます。

## Open Workspace

プロジェクトを開きます。

## Save Workspace

編集中のプロジェクトをセーブします。

## Close Workspace

現在開いているプロジェクトを閉じます。

## Exit

wb88を終了します。

## [Build]メニュー

## Build

- Compile/Assemble
- Build
- ReBuild All
- Stop Build

## Compile/Assemble

プロジェクトビュー内で選択されているファイルを、コンパイルまたはアセンブルします。

## Build

現在開いているプロジェクトをビルドします。

## ReBuild All

現在開いているプロジェクトを新規にビルドします。

## Stop Build

実行中のビルド処理を中止します。

## [Debug]メニュー

## Debug

- Sim88 Simulator
- DB88 Debugger
- ICE88UR Debugger

## Sim88 Simulator

シミュレータSim88を起動します。

## DB88 Debugger

db88デバッガを起動します。

## ICE88UR Debugger

ice88urデバッガを起動します。

## [Tools]メニュー

## Tool

- Simulator Tools
- Dev Tools
- On-Board ROM Writer
- Sim88 Configuration
- Editor Configuration

Auto Evaluation System

Bitmap Utility

LCD Panel Customize Utility

Port Setting Utility

## Simulator Tools - Auto Evaluation System

自動評価システムAutoEvaを起動します。

## Simulator Tools - Bitmap Utility

ビットマップユーティリティBmpUtilを起動します。

## Simulator Tools - LCD Panel Customize Utility

LCDパネルカスタマイズユーティリティLCDUtilを起動します。

## Simulator Tools - Port Setting Utility

ポート設定ユーティリティPrUtilを起動します。

## [View]メニュー

## View

- ✓ Tool Bar
- ✓ Status Bar

## Tool Bar

ツールバーの表示/非表示を切り替えます。

## Status Bar

ステータスバーの表示/非表示を切り替えます。

## [Source]メニュー

## Source

- Insert file into Project
- Remove file from Project

## Insert file into Project

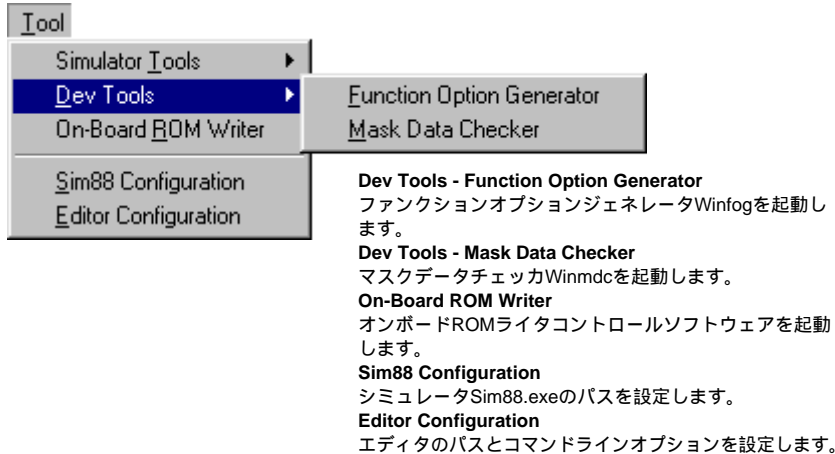
指定のソースファイルを現在開いているプロジェクトに追加します。

## Remove file from Project

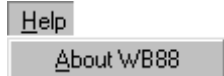
プロジェクトビュー内で選択されているファイルをプロジェクトから削除します。

## メニュー

## [Tools]メニュー



## [Help]メニュー



**About WB88**  
ワークベンチのバージョン情報を表示します。

エラーメッセージ  
システムエラー

not enough memory	wb88を実行するために十分なメモリがありません。
-------------------	---------------------------

## プロジェクト生成時のエラー

The file is not a WB88 project file. (<filename>)	<filename>は、wb88のプロジェクトファイルではありません。
The version of the project file is not supported. (<filename>)	そのプロジェクトファイル<filename>のバージョンは、サポートしていません。
Unable to create a project : cannot access. <filename>	<filename>に正しくアクセスできなかったため、プロジェクトを生成できません。

## エラーメッセージ

## プロジェクト生成時のエラー

Unable to create a project : Unable to copy DEF file.(<filename>)	定義ファイル<filename>のコピーに失敗したため、プロジェクトを生成できません。
The project is already existed.(<filename>)	<filename>は既に存在するため、プロジェクトを作成できません。
Unable to create a project : Dev Directory of S1C88 family package does not exist.	DEV Directory が存在しないため、プロジェクトを作成できません。

## ファイル追加時のエラー

The file cannot be added to the project. It is not a C file.(<filename>)	Cソースファイルでないため、<filename>をプロジェクトに追加できません。
The file cannot be added to the project. It is not an ASM file.(<filename>)	アセンブリソースファイルでないため、<filename>をプロジェクトに追加できません。
The file cannot be added to the project. It is not a header file.(<filename>)	ヘッダファイルでないため、<filename>をプロジェクトに追加できません。
The file is already existed in the project. It cannot be added in the project.(<filename>)	<filename>は既にプロジェクトに存在するため、追加できません。
WB88 does not support such source file type.(<filename>)	wb88 がサポートしていないソースファイルです。

## ファイルエラー

Failed to access the file.(<filename>)	<filename>の操作に失敗しました。
Unable to open the file.(<filename>)	<filename>のオープンに失敗しました。

## ツール起動時のエラー

Unable to execute ICE88UR.exe : Unable to access <filename>.	<filename>のアクセスに失敗したため、S5U1C88000H5を起動できません。
Unable to execute Sim88 : Unable to access the DEF file.(<filename>)	定義ファイルのアクセスに失敗したため、Sim88を起動できません。
Unable to execute <toolname>.	<toolname> の起動に失敗しました。

## ビルド時のエラー

Select a C or an ASM file.	Cソースもしくはアセンブリソースファイルを選択してください。
Build Command needs an active project.	ビルドするには、プロジェクトが必要です。
No target file is found in the project.	ビルドターゲットファイルがプロジェクト内にありません。

## その他のエラー

The command needs an active project.	そのコマンドには、プロジェクトが必要です。
--------------------------------------	-----------------------

## 起動コマンド

c88 [[*option*...]...[*file*]...]...

## オプション

## インクルードオプション

-f <i>file</i>	オプションを <i>file</i> から読み込みます。
-H <i>file</i>	コンパイルの前に <i>file</i> をインクルードします。
-Idirectory	directoryでインクルードファイルを探します。

## 前処理オプション

-Dmacro[=def]	プリプロセッサmacroを定義します。
---------------	---------------------

## コード生成オプション

-M{s c d l}	それぞれスモール、コンパクトコード、コンパクトデータ、ラージに対応します。
-O{0 1}	最適化を制御します。

## 出力ファイルオプション

-e	コンパイラエラーが発生した場合、出力ファイルを削除します。
-o <i>file</i>	出力ファイルの名前を <i>file</i> で指定します。
-s	Cソースコードをアセンブラ出力とマージします。

## 診断オプション

-V	バージョンヘッダのみを表示します。
-err	診断をエラーリストファイル(.err)に送信します。
-g	シンボリックデバッグ情報を有効にします。
-w[num]	1つまたはすべての警告メッセージを抑制します。

## エラー/ワーニングメッセージ

I: 情報 E: エラー F: 致命的なエラー S: システムエラー W: ワーニング

## フロントエンド

F 1: evaluation expired	評価版の有効期限が切れています。
W 2: unrecognized option: ' <i>option</i> '	このオプションが存在しません。
E 4: expected <i>number</i> more ' <i>endif</i> '	コンパイラのプリプロセッサ部分に、"#if"、"#ifdef"、"#ifndef"指示文がありますが、同じソースファイルに、対応する"#endif"がありません。
E 5: no source modules	コンパイルするソースファイルを少なくとも1つ指定する必要があります。
F 6: cannot create " <i>file</i> "	出力ファイルまたは一時ファイルが作成できませんでした。
F 7: cannot open " <i>file</i> "	このファイルが実際に存在するかどうかチェックしてください。
F 8: attempt to overwrite input file " <i>file</i> "	出力ファイルの名前は、入力ファイルと異なる名前にする必要があります。
E 9: unterminated constant character or string	このエラーは文字列を二重引用符(")で閉じなかった場合や文字定数を一重引用符(')で閉じなかった場合に発生します。
F 11: file stack overflow	このエラーは、ファイルインクルードのネストの深さが最大数(50)を越えた場合に発生します。
F 12: memory allocation error	すべての空きメモリ空間が使用されています。
W 13: prototype after forward call or old style declaration - ignored	それぞれの関数のプロトタイプが実際の呼び出しの前にあることをチェックしてください。
E 14: ';' inserted	式の文にセミコロンが必要です。
E 15: missing filename after -o option	-oオプションの後に、出力ファイル名を付ける必要があります。
E 16: bad numeric constant	定数は、その構文に準拠する必要があります。また定数は、割り当てられた型を表す上で大きくなりすぎないようにします。
E 17: string too long	このエラーは、最大文字列サイズ(1500)を越えた場合に発生します。
E 18: illegal character (0x <i>hexnumber</i> )	16進ASCII値0x <i>hexnumber</i> の文字は、ここでは使用できません。
E 19: newline character in constant	文字定数または文字列定数で改行を使用できるのは、円記号(¥)またはバックスラッシュ(\)が前に付いている場合のみです。
E 20: empty character constant	文字定数に入れる文字は1文字だけです。空の文字定数("")は使用できません。
E 21: character constant overflow	文字定数に入れる文字は1文字だけです。エスケープシーケンスは1文字に変換されます。
E 22: '#define' without valid identifier	"#define"の後には識別子を指定する必要があります。

## エラー/ワーニングメッセージ

## フロントエンド

E 23: 'else' without 'if'	"#else"は、対応する"#if"、"#ifdef"、"#ifndef"構文の中でのみ使用することができます。
E 24: '#endif' without matching '#if'	"#endif"がありますが、それに対応するプリプロセッサ指示文"#if"、"#ifdef"、"#ifndef"がありません。
E 25: missing or zero line number	"#line"では、0以外の数値を指定する必要があります。
E 26: undefined control	コントロール行("#identifier"を含む行)には、既知のプリプロセッサ指示文が入っていない必要があります。
W 27: unexpected text after control	"#ifdef"および"#ifndef"には、識別子が1つだけ必要です。また、"#else"および"#endif"には、改行以外の文字は必要ありません。"#undef"では、識別子が1つだけ必要です。
W 28: empty program	ソースファイルには、外部定義が最低でも1つ必要になります。コメント以外何もないソースファイルは、空のプログラムと見なされます。
E 29: bad '#include' syntax	"#include"の後には、有効なヘッダ名構文を続ける必要があります。
E 30: include file "file" not found	"#include"指示文の後に、必ず既存のインクルードファイルを指定する必要があります。また、インクルードファイルには、正しいパスを指定していなければなりません。
E 31: end-of-file encountered inside comment	コンパイラがコメントを読み込んでいるときに、エンドオブファイルが見つかりました。コメントが正しく終了していない可能性があります。
E 32: argument mismatch for macro "name"	関数形式のマクロを起動する場合、引数の数は、定義のパラメータの数と一致させる必要があります。また、関数形式のマクロを起動するときは、")"を最後に付ける必要があります。
E 33: "name" redefined	この識別子が、複数回定義されているか、後の宣言が前の宣言と異なっているかのいずれかです。
W 34: illegal redefinition of macro "name"	マクロは、再定義されるマクロの本体が、最初に定義されたマクロの本体とまったく同じ場合に限り、再定義することができます。
E 35: bad filename in '#line'	#lineの文字列リテラルは、(ある場合)"wide-char"文字列でなければなりません。
W 36: 'debug' facility not installed	"#pragma debug"は、コンパイラのデバッグバージョンでのみ使用できます。
W 37: attempt to divide by zero	0による除算または剰余算が見つかりました。
E 38: non integral switch expression	switch条件式は、整数値を評価する必要があります。
F 39: unknown error number: number	このエラーは、発生してはならないものです。
W 40: non-standard escape sequence	不正なエスケープ文字が含まれています。

## フロントエンド

E 41: '#elif' without 'if'	"#elif"指示文が、"#if"、"#ifdef"、"#ifndef"構文内にありませんでした。
E 42: syntax error, expecting parameter type/declaration/ statement	パラメータリスト、宣言、文のいずれかに、構文エラーがあります。
E 43: unrecoverable syntax error, skipping to end of file	コンパイラが、回復不可能なエラーを検出しました。
I 44: in initializer "name"	定数イニシャライザが正しいかどうかチェックするときの情報メッセージです。
E 46: cannot hold that many operands	値スタックには、20を超えるオペランドを入れることができません。
E 47: missing operator	式の中に演算子が必要です。
E 48: missing right parenthesis	")"が必要です。
W 49: attempt to divide by zero - potential run-time error	0による除算または剰余算を含む式が見つかりました。
E 50: missing left parenthesis	"("が必要です。
E 51: cannot hold that many operators	状態スタックには、20を超えるオペレーターを入れることができません。
E 52: missing operand	オペランドが必要です。
E 53: missing identifier after 'defined' operator	#if defined(identifier)には識別子が必要です。
E 54: non scalar controlling expression	反復条件および"if"条件は、スカラ型になっている必要があります(struct, union, ポインタは不可)。
E 55: operand has not integer type	"#if"指示文のオペランドは、int型定数を評価する必要があります。
W 56: '<debugoption><level>' no associated action	このデバッグオプションおよびレベルには、対応するデバッグアクションがありません。
W 58: invalid warning number: number	-wオプションで指定されたワーニング番号が存在しません。
F 59: sorry, more than number errors	40以上のエラーがあるため、コンパイルが停止しました。
E 60: label "label" multiple defined	同じ関数内では、1つのラベルを複数回定義することができません。
E 61: type clash	型の競合が見つかりました。
E 62: bad storage class for "name"	記憶クラス指定子autoおよびregisterは、外部定義の宣言指定子で使用することはできません。また、パラメータ宣言で利用できる記憶クラス指定子は、registerのみです。
E 63: "name" redeclared	この識別子は、すでに宣言されています。コンパイラは2番目の宣言を使用します。



## エラー/ワーニングメッセージ

## フロントエンド

E 64: incompatible redeclaration of "name"	この識別子は、すでに宣言されています。
W 66: function "name": variable "name" not used	使用されていない変数が宣言されています。
W 67: illegal suboption: <i>option</i>	サブオプションがこのオプションで有効ではありません。
W 68: function "name": parameter "name" not used	使用されていない関数パラメータが宣言されています。
E 69: declaration contains more than one basic type specifier	型指定子を繰り返すことはできません。
E 70: 'break' outside loop or switch	break文は、switchまたはループ(do, for, while)内でのみ使用できます。
E 71: illegal type specified	指定した型は、このコンテキストでは使用できません。
W 72: duplicate type modifier	指定子リストまたは修飾子リストで型修飾子を繰り返すことはできません。
E 73: object cannot be bound to multiple memories	1つのオブジェクトではメモリ属性を1つだけ使用してください。
E 74: declaration contains more than one class specifier	1つの宣言に入れることができる記憶クラス指定子は1つだけです。
E 75: 'continue' outside a loop	continueは、ループ本体(do, for, while)の中でのみ使用できます。
E 76: duplicate macro parameter "name"	この識別子が、マクロ定義のformat1パラメータリストで複数回使用されています。
E 77: parameter list should be empty	関数定義の一部でない識別子リストは、空でなければなりません。
E 78: 'void' should be the only parameter	引数をとらない関数の関数プロトタイプでは、voidが唯一のパラメータになります。
E 79: constant expression expected	定数式には、カンマを入れることができません。また、ビットフィールド幅、enumを定義する式、配列にバインドされた定数、switch case式は、すべてint型定数式でなければなりません。
E 80: '#' operator shall be followed by macro parameter	"#"演算子の後にはマクロ引数を続ける必要があります。
E 81: '###' operator shall not occur at beginning or end of a macro	"###"(トークン連結)演算子は、隣接するプリプロセッサトークンを一緒にペーストするときに使用されます。そのためマクロ本体の最初や最後で使用することはできません。
W 86: escape character truncated to 8 bit value	16進エスケープシーケンス("¥"または"\")の後に"x"と数値)の値は、8ビット記憶域に収まる必要があります。
E 87: concatenated string too long	生成された文字列が、1500文字の制限を越えています。
W 88: "name" redeclared with different linkage	この識別子は、すでに宣言されています。

## フロントエンド

E 89: illegal bitfield declarator	ビットフィールドは、int型としてのみ宣言することができます。ポインタや関数として宣言することはできません。
E 90: #error message	messageは、"#error"プリプロセッサ指示文で指定する説明テキストです。
W 91: no prototype for function "name"	それぞれの関数には、正しい関数プロトタイプが必要です。
W 92: no prototype for indirect function call	それぞれの関数には、正しい関数プロトタイプが必要です。
I 94: hiding earlier one	エラーE 63の後に追加されるメッセージです。2番目の宣言が使用されます。
F 95: protection error: message	プロテクションキーの初期化時に問題が発生しました。
E 96: syntax error in #define	#define id(には、右のかっこ")"が必要です。
E 97: "..." incompatible with old-style prototype	2つの関数が同じ名前のパラメータ型リストを持っている場合、これは古いスタイルの宣言であるため、パラメータリストに省略記号を入れることはできません。
E 98: function type cannot be inherited from a typedef	typedefは関数定義で使用することはできません。
F 99: conditional directives nested too deep	"#if"、"#ifdef"、"#ifndef"指示文は、50レベルより深くネストすることができません。
E 100: case or default label not inside switch	case:ラベルまたはdefault:ラベルは、switch内でのみ使用することができます。
E 101: vacuous declaration	宣言の中に足りないものがあります。
E 102: duplicate case or default label	switch caseでは、評価の後値がそれぞれ固有でなければならず、switch内にdefault:ラベルが少なくとも1つ必要になります。
E 103: may not subtract pointer from scalar	ポインタの減算で使用するオペランドは、ポインタ - ポインタ、またはポインタ - スカラのみです。
E 104: left operand of operator has not struct/union type	","または"->"の最初のオペランドは、struct型またはunion型にならなければなりません。
E 105: zero or negative array size - ignored	配列にバインドされる定数は、0より大きくなければなりません。
E 106: different construction	パラメータ型リストを持つ互換関数型は、パラメータ数と省略記号の使用の点で共通でなければなりません。また対応するパラメータは互換性のある型でなければなりません。
E 107: different array sizes	互換関数型のそれぞれの配列パラメータは同じサイズでなければなりません。
E 108: different types	互換関数型のそれぞれのパラメータ、およびそれぞれのプロトタイプパラメータの型は、公開されている定義パラメータを持つ互換性のある型でなければなりません。

## エラー/ワーニングメッセージ

## フロントエンド

E 109: floating point constant out of valid range	浮動小数点定数には、割り当てられている型に収まる値がなければなりません。
E 110: function cannot return arrays or functions	関数では、配列型または関数型の戻り型を使用できません。関数に対するポインタは使用できます。
I 111: parameter list does not match earlier prototype	パラメータリストをチェックするか、プロトタイプを調整してください。パラメータの数と型は一致する必要があります。
E 112: parameter declaration must include identifier	宣言がプロトタイプの場合、それぞれのパラメータの宣言に識別子がなければなりません。また、typedef名として宣言された識別子はパラメータ名として使用できません。
E 114: incomplete struct/union type	structまたはunionを使用する前に、その型を知らせる必要があります。
E 115: label " <i>name</i> " undefined	goto文が見つかりましたが、同じ関数またはモジュール内にこのラベルがありませんでした。
W 116: label " <i>name</i> " not referenced	このラベルが定義されていますが、参照されませんでした。ラベルの参照は、同じ関数内またはモジュール内になければなりません。
E 117: " <i>name</i> " undefined	この識別子は定義されていませんでした。変数の型は、使用する前に宣言で指定する必要があります。
W 118: constant expression out of valid range	caseラベルで使用されている定数式が長すぎる可能性があります。また、浮動小数点値をint型に変換するとき、浮動小数点定数が長すぎることがあります。
E 119: cannot take 'sizeof' bitfield or void type	ビットフィールドまたはvoid型のサイズが知らされていません。そのためそのサイズを使用することができません。
E 120: cannot take 'sizeof' function	関数のサイズが知らされていません。そのためそのサイズを使用することができません。
E 121: not a function declarator	これは有効な関数ではありません。
E 122: unnamed formal parameter	パラメータには、正しい名前を指定する必要があります。
W 123: function should return something	非void関数の戻り値には式がなければなりません。
E 124: array cannot hold functions	関数の配列は使用できません。
E 125: function cannot return anything	式を持つreturnが、void関数にありません。
W 126: missing return (function " <i>name</i> ")	空でない関数を持つ非void関数には、return文が必要になります。
E 129: cannot initialize " <i>name</i> "	宣言子リスト内の宣言子では、初期化文を入れないください。またextern宣言では、イニシャライザを使用できません。
W 130: operands of operator are pointers to different types	演算子または割り当て(=)のポインタオペランドは同じ型でなければなりません。

## フロントエンド

E 131: bad operand type(s) of <i>operator</i>	この演算子には、他の型のオペランドが必要です。
W 132: value of variable " <i>name</i> " is undefined	変数が定義される前に使用されている場合、このワーニングが発生します。
E 133: illegal struct/union member type	関数は、structまたはunionのメンバにすることができません。また、ビットフィールドはint型またはunsigned型のみをとることができます。
E 134: bitfield size out of range - set to 1	ビットフィールドの幅は、この型のビット数より多くしたり、負の値にしたりすることはできません。
W 135: statement not reached	指定された文が、実行されません。
E 138: illegal function call	関数以外のオブジェクトで関数呼び出しを実行することはできません。
E 139: <i>operator</i> cannot have aggregate type	(キャスト)の型名と(キャスト)のオペランドは、スカラでなければなりません(struct、union、ポインタは不可)。
E 140: type cannot be applied to a register/bit/bitfield object or builtin/inline function	たとえば"&"演算子(アドレス)は、レジスタやビットフィールドでは使用できません。
E 141: <i>operator</i> requires modifiable lvalue	"++"演算子や"--"演算子のオペランド、および割り当てや複合割り当てでの左の演算子(lvalue)は、変更可能でなければなりません。
E 143: too many initializers	イニシャライザの数をオブジェクトの数より多くすることはできません。
W 144: enumerator " <i>name</i> " value out of range	enum定数がintの制限を越えています。
E 145: requires enclosing curly braces	複合イニシャライザは中かっこで閉じる必要があります。
E 146: argument # <i>number</i> : memory spaces do not match	プロトタイプでは、引数のメモリ空間が一致する必要があります。
W 147: argument # <i>number</i> : different levels of indirection	プロトタイプでは、引数と割り当ての型に互換性がなければなりません。
W 148: argument # <i>number</i> : struct/union type does not match	プロトタイプでは、プロトタイプ化した関数の引数と実際の引数の両方がstructまたはunionでしたが、異なるタグが付いています。タグの型は一致しなければなりません。
E 149: object " <i>name</i> " has zero size	structまたはunionに、不完全な型のメンバがあります。
W 150: argument # <i>number</i> : pointers to different types	プロトタイプでは、引数のポインタ型に互換性がなければなりません。
W 151: ignoring memory specifier	struct、union、enumのメモリ指定子が無視されます。
E 152: operands of <i>operator</i> are not pointing to the same memory space	オペランドが同じメモリ空間をポインタしていることを確認してください。

## エラー/ワーニングメッセージ

## フロントエンド

E 153: 'sizeof' zero sized object	暗黙的または明示的なsizeof演算で、未知のサイズのオブジェクトが参照されています。
E 154: argument #number: struct/union mismatch	プロトタイプでは、プロトタイプ化した関数の引数と実際の引数のうち、いずれかだけがstructまたはunionでした。型は一致しなければなりません。
E 155: casting lvalue 'type' to 'type' is not allowed	"++"演算子や"--"演算子のオペランド、または割り当てや複合割り当ての左の演算子(lvalue)は、別の型にキャストすることができません。
E 157: "name" is not a formal parameter	宣言子に識別子リストがある場合、宣言子リストにはその識別子のみを入れることができます。
E 158: right side of operator is not a member of the designated struct/union	".または"->"の2番目のオペランドは、指定されたstructまたはunionのメンバでなければなりません。
E 160: pointer mismatch at operator	operatorの両方のオペランドが、有効なポインタでなければなりません。
E 161: aggregates around operator do not match	operatorの両側にある構造体、共用体、配列の内容は同じでなければなりません。
E 162: operator requires an lvalue or function designator	"&"演算子(アドレス)には、lvalueや関数指名子が必要になります。
W 163: operands of operator have different level of indirection	演算子のポインタまたはアドレスの型は、割り当てと互換性がなければなりません。
E 164: operands of operator may not have type 'pointer to void'	operatorのオペランドにオペランド(void *)がありません。
W 165: operands of operator are incompatible: pointer vs. pointer to array	ポインタの型または演算子のアドレスは、割り当てと互換性がなければなりません。ポインタは、配列へのポインタに割り当てることができません。
E 166: operator cannot make something out of nothing	型voidを他の型にキャストすることはできません。
E 170: recursive expansion of inline function "name"	_inline関数は再帰的に使用することができません。
E 171: too much tail-recursion in inline function "name"	関数レベルが40以上の場合、このエラーが現れます。
W 172: adjacent string have different types	2つの文字列を連結するとき、両方の文字列が同じ型でなければなりません。
E 173: 'void' function argument	関数は、void型の引数をとることができません。
E 174: not address constant	定数アドレスが使用される予定になっていました。スタティク変数と異なり、auto変数には、固定されたメモリロケーションがないため、auto変数のアドレスは定数になりません。
E 175: not an arithmetic constant	定数式では、割り当て演算子、++演算子、--演算子、関数を使用できません。

## フロントエンド

E 176: address of automatic is not a constant	スタティク変数と異なり、auto変数には、固定されたメモリロケーションがないため、auto変数のアドレスは定数になりません。
W 177: static variable "name" not used	使用されていないスタティク変数が宣言されています。
W 178: static function "name" not used	呼び出されていない静的関数が宣言されています。
E 179: inline function "name" is not defined	インライン関数のプロトタイプのみがあり、実際のインライン関数が存在しないことが原因です。
E 180: illegal target memory (memory) for pointer	ポインタがmemoryをポイントしていません。
W 182: argument #number: different types	プロトタイプで、引数のタイプに互換性がなければなりません。
I 185: (prototype synthesized at line number in "name")	古いスタイルのプロトタイプが含まれているソースファイルの位置を通知する情報メッセージです。
E 186: array of type bit is not allowed	配列には、ビット型変数を入れることができません。
E 187: illegal structure definition	構造体は、メンバが知らされている場合に限り、定義(初期化)することができます。
E 188: structure containing bit-type field is forced into bitaddressable area	このエラーは、ビット型メンバを含む構造体でビットアドレス可能な記憶タイプを使用するときに発生します。
E 189: pointer is forced to bitaddressable, pointer to bitaddressable is illegal	ビットアドレス可能なメモリを示すポインタは使用できません。
W 190: "long float" changed to "float"	ANSI Cの浮動小数点定数は、定数に接尾辞"f"が付いている場合を除き、double型を持つものとして扱われます。
E 191: recursive struct/union definition	structまたはunionには、それ自体を入れることができません。
E 192: missing filename after -f option	-fオプションにはファイル名引数を付ける必要があります。
E 194: cannot initialize typedef	typedef変数に値を割り当てることはできません。
F 199: demonstration package limits exceeded	デモパッケージには製品版にはない一定の制限があります。
W 200: unknown pragma - ignored	コンパイラは、未知のプリAGMAについては無視します。
W 201: "name" cannot have storage type - ignored	register変数またはオートマチック/パラメータには、記憶タイプを使用することができません。
E 202: "name" is declared with 'void' parameter list	関数が何も受け付けない場合(voidパラメータリスト)、関数に引数を付けて呼び出すことはできません。
E 203: too many/few actual parameters	プロトタイプでは、関数の引数の数が、その関数のプロトタイプと一致している必要があります。

## エラー/ワーニングメッセージ

## フロントエンド

W 204: U suffix not allowed on floating constant - ignored	浮動小数点定数には、"U"接尾辞または"u"接尾辞を付けることができません。
W 205: F suffix not allowed on integer constant - ignored	int型定数には、"F"接尾辞または"f"接尾辞を付けることができません。
E 206: 'name' named bit-field cannot have 0 width	ビットフィールドは、0より大きい値を持つint型定数式でなければなりません。
E 212: "name": missing static function definition	staticプロトタイプを持つ関数にその定義がありません。
W 303: variable 'name' possibly uninitialized	関数が何かを返すことになっているにもかかわらず、初期化文が、到達しない箇所にあります。
E 327: too many arguments to pass in registers for _asmfunc 'name'	_asmfunc関数は、Cとアセンブラの間で固定レジスタベースのインタフェースを使用しますが、このときに受け渡しする引数の数は、使用可能なレジスタの数によって制限を受けます。関数nameでこの制限を越えてしまいました。

## バックエンド

W 501: function qualifier used on non-function	関数修飾子は、関数でのみ使用できます。
E 502: Intrinsic function '_int()' needs an immediate value as parameter	_int()組み込み関数の引数は、任意のタイプの整数式ではなく、整数定数式でなければなりません。
E 503: Intrinsic function '_jrsf()' needs an immediate value 0..3	指定する値は0から3までの定数値でなければなりません。
W 508: function qualifier duplicated	使用できる関数修飾子は1つだけです。
E 511: interrupt function must have void result and void parameter list	_interrupt(n)で宣言された関数は、引数を受け付けないため、何も返すことができません。
W 512: 'number' illegal interrupt number (0, or 3 to 251) - ignored	割り込みベクタ番号は、0、または3から251の範囲内になければなりません。他の数値の場合は不正になります。
E 513: calling an interrupt routine, use '_swi()'	割り込み関数は、直接呼び出すことができません。組み込み関数_swi()を使用しなければなりません。
E 514: conflict in '_interrupt/' '_asmfunc' attribute	現在の関数修飾子宣言および前回の関数修飾子宣言の属性が同じになっていません。
E 515: different '_interrupt' number	現在の関数修飾子宣言および前回の関数修飾子宣言の番号が同じになっていません。
E 516: 'memory_type' is illegal memory for function	記憶タイプがこの関数で有効ではありません。
E 517: conversion of long address to short address	このエラーは、ポインタ変換が必要な場合に出されます。

## バックエンド

F 524: illegal memory model	コンパイラの使用法を参照して、-Mオプションの正しい引数を調べてください。
E 526: function qualifier '_asmfunc' not allowed in function definition	_asmfuncは、関数プロトタイプでのみ使用できます。
E 528: _at( ) requires a numerical address	数値アドレスに評価される式のみを使用することができます。
E 529: _at( ) address out of range for this type of object	指定されたメモリ空間に、絶対アドレスがありません。
E 530: _at( ) only valid for global variables	絶対アドレスには、グローバル変数のみを置くことができます。
E 531: _at( ) only allowed for uninitialized variables	絶対変数は初期化することができません。
E 532: _at( ) has no effect on external declaration	externと宣言した場合、その変数はコンパイラによって割り当てられません。
W 533: c88 language extension keyword used as identifier	言語拡張機能キーワードは予約語になっており、予約語は識別子として使用することができません。
E 536: illegal syntax used for default section name 'name' in -R option	-Rオプションの説明を参照し、正しい構文を調べてください。
E 537: default section name 'name' not allowed	-Rオプションの説明を参照し、正しい構文を調べてください。
W 538: default section name 'name' already renamed to 'new_name'	-Rオプションのいずれか、renamesectプラグマ、他の名前のみを使用してください。
W 542: optimization stack underflow, no optimization options are saved with #pragma optimize	このワーニングは、前の#pragma endoptimizeでオプションが保存されていないときに#pragma endoptimizeを使用すると発生します。
W 555: current optimization level could reduce debugging comfort (-g);	これらの最適化設定ではHLLデバッグ競合が発生します。
E 560: Float/Double: not yet implemented	浮動小数点は以下のバージョンでサポートされています。

## ライブラリ

<ctype.h>	isalnum, isalpha, isascii, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, toascii, _tolower, tolower, _toupper, toupper
<errno.h>	エラー番号 C関数はありません
<float.h>	浮動小数点演算に関連する定数
<limits.h>	int型の制限とサイズを定義しています C関数はありません
<locale.h>	localeconv, setlocale 骨組みとして提供されています
<math.h>	acos, asin, atan, atan2, ceil, cos, cosh, exp, fabs, floor, fmod, frexp, ldexp, log, log10, modf, pow, sin, sinh, sqrt, tan, tanh
<setjmp.h>	longjmp, setjmp
<signal.h>	raise, signal これらの関数は骨組みとして提供されています
<simio.h>	_simi, _simo
<stdarg.h>	va_arg, va_end, va_start
<stddef.h>	offsetof, 特殊型の定義
<stdio.h>	clearerr, fclose, _fclose, feof, ferror, fflush, fgetc, fgetpos, fgets, fopen, _fopen, fprintf, fputc, fputs, fread, freopen, fscanf, fseek, fsetpos, ftell, fwrite, getc, getchar, gets, _joread, _iowrite, _lseek, perror, printf, putc, putchar, puts, _read, remove, rename, rewind, scanf, setbuf, setvbuf, sprintf, sscanf, tmpfile, tmpnam, ungetc, vfprintf, vprintf, vsprintf, _write
<stdlib.h>	abort, abs, atexit, atof, atoi, atol, bsearch, calloc, div, exit, free, getenv, labs, ldiv, malloc, mblen, mbstowcs, mbtowc, qsort, rand, realloc, srand, strtod, strtol, stroul, system, wcstombs, wctomb
<string.h>	memchr, memcmp, memcpy, memmove, memset, strcat, strchr, strcmp, strcol, strcpy, strcspn, strerror, strlen, strncat, strncmp, strncpy, strpbrk, strchr, strspn, strstr, strtok, strxfrm
<time.h>	asctime, clock, ctime, difftime, gmtime, localtime, mktime, strftime, time これらの関数はすべて骨組みとして提供されています

## 起動コマンド

```
as88 [option]...source-file [map-file]
```

## オプション

<b>-C file</b>	fileをソースの前にインクルードします。
<b>-Dmacro[=def]</b>	プリプロセッサmacroを定義します。
<b>-L[flag...]</b>	指定されたソース行をリストファイルから削除します。
<b>-M[sic dl]</b>	メモリモデルを指定します。
<b>-V</b>	バージョンヘッダのみを表示します。
<b>-c</b>	大文字と小文字を区別しないモードに切り換えます(デフォルトでは区別する)。
<b>-e</b>	アセンブリエラーの場合、オブジェクトファイルを削除します。
<b>-err</b>	エラーメッセージをエラーファイルにリダイレクトします。
<b>-f file</b>	オプションをfileから読み込みます。
<b>-i[lg]</b>	デフォルトのラベルスタイルをローカルまたはグローバルとして指定します。
<b>-l</b>	リストファイルを生成します。
<b>-o filename</b>	出力ファイルの名前を指定します。
<b>-t</b>	セクションの要約を表示します。
<b>-v</b>	冗長モード。進行中にファイル名とパスの回数をプリントします。
<b>-w[num]</b>	1つまたはすべての警告メッセージを抑制します。

## 関数

```
@function_name(argument[,argument]...)
```

## 数学関数

<b>ABS</b>	絶対値
<b>MAX</b>	最大値
<b>MIN</b>	最小値
<b>SGN</b>	記号を返します

## 文字列関数

<b>CAT</b>	文字列を連結します
<b>LEN</b>	文字列の長さ
<b>POS</b>	文字列内の副文字列の位置
<b>SCP</b>	文字列を比較します
<b>SUB</b>	文字列から副文字列を抽出します

## マクロ関数

<b>ARG</b>	マクロ引数関数
<b>CNT</b>	マクロ引数の数
<b>MAC</b>	マクロ定義関数
<b>MXP</b>	マクロ展開関数

## アセンブラモード関数

<b>AS88</b>	アセンブラの実行可能ファイル名
<b>DEF</b>	シンボル定義関数
<b>LST</b>	LIST擬似命令のフラグ値
<b>MODEL</b>	選択したアセンブラのモデル

## アドレス操作関数

<b>CADDR</b>	コードアドレス
<b>COFF</b>	コードページオフセット
<b>CPAG</b>	コードページ番号
<b>DADDR</b>	データアドレス
<b>DOFF</b>	データページオフセット
<b>DPAG</b>	データページ番号
<b>HIGH</b>	256バイトページ番号
<b>LOW</b>	256バイトページオフセット

## アセンブラ擬似命令

## デバッグ

<b>CALLS</b>	呼び出し情報をオブジェクトファイルに渡します。オーバーレイセクションを重ね書きするためにリンク時に呼び出しツリーを構築するときに使用されます。
<b>SYMB</b>	シンボリックデバッグ情報を渡します。

## アセンブラコントロール

<b>ALIGN</b>	調整を指定します。
<b>COMMENT</b>	コメント行を開始させます。この擬似命令は、IF/ELIF/ELSE/ENDIF構成体およびMACRO/DUP定義では使用できません。
<b>DEFINE</b>	置換文字列を定義します。
<b>DEFSECT</b>	セクション名と属性を定義します。
<b>END</b>	ソースプログラムの終了を示します。
<b>FAIL</b>	プログラマが生成するエラーメッセージです。
<b>INCLUDE</b>	二次ファイルをインクルードします。
<b>MSG</b>	プログラマが生成するメッセージです。
<b>RADIX</b>	定数に対する入力基数を変更します。
<b>SECT</b>	セクションを起動します。
<b>UNDEF</b>	DEFINEシンボルの定義を解除します。
<b>WARN</b>	プログラマが生成する警告です。

## シンボル定義

<b>EQU</b>	シンボルと値を等しいものとして定義します。順方向の参照を受け付けます。
<b>EXTERN</b>	外部シンボル宣言です。モジュールの本体でも使用できます。
<b>GLOBAL</b>	グローバルシンボル宣言です。モジュールの本体でも使用できます。
<b>LOCAL</b>	ローカルシンボル宣言です。
<b>NAME</b>	オブジェクトファイルを識別します。
<b>SET</b>	シンボルに値を設定します。順方向の参照を受け付けます。

## データ定義/記憶域の割り当て

<b>ASCII</b>	ASCII文字列を定義します。
<b>ASCIZ</b>	NULLを埋め込んだASCII文字列を定義します。
<b>DB</b>	定数バイトを定義します。
<b>DS</b>	記憶域を定義します。
<b>DW</b>	定数ワードを定義します。

## マクロおよび条件アセンブラ

<b>DUP</b>	ソース行のシーケンスを複製します。
<b>DUPA</b>	引数を付けてシーケンスを複製します。
<b>DUPC</b>	文字を付けてシーケンスを複製します。
<b>DUPF</b>	ループでシーケンスを複製します。
<b>ENDIF</b>	条件アセンブラの終了。
<b>ENDM</b>	マクロ定義の終了。
<b>EXITM</b>	マクロを終了させます。
<b>IF</b>	条件アセンブラ擬似命令。
<b>MACRO</b>	マクロ定義。
<b>PMACRO</b>	マクロ定義を消去します。

## エラーメッセージ

## 警告(W)

W 101: use <i>option</i> at the start of the source; ignored	プライマリオプションは、ソースの最初に使用しなければなりません。
W 102: duplicate attribute " <i>attribute</i> " found	EXTERN擬似命令の属性が複数回使用されています。重複している属性を削除してください。
W 104: expected an attribute but got <i>attribute</i> ; ignored	
W 105: section activation expected, use <i>name</i> directive	SECT擬似命令を使用してセクションをアクティブにしてください。
W 106: conflicting attributes specified " <i>attributes</i> "	EXTERN擬似命令で競合する属性が2回使用されています。
W 107: memory conflict on object " <i>name</i> "	ラベルまたは他のオブジェクトが明示的または暗黙的に定義されていますが、使用されているメモリタイプ同士に互換性がありません。
W 108: object attributes redefinition " <i>attributes</i> "	ラベルまたは他のオブジェクトが明示的または暗黙的に定義されていますが、使用されている属性同士に互換性がありません。
W 109: label " <i>label</i> " not used	ラベル <i>label</i> が、GLOBAL擬似命令で定義されていますが、定義も参照もされていません。またはラベルが、LOCAL擬似命令で定義されていますが、参照されていません (LOCALラベルの場合)。
W 110: extern label " <i>label</i> " defined in module, made global	ラベル <i>label</i> が、EXTERN擬似命令で定義されていますが、ソース内のラベルとして定義されています。このラベルはグローバルラベルとして扱われます。
W 111: unknown \$LIST flag " <i>flag</i> "	\$LISTコントロールに未知の <i>flag</i> を指定しています。
W 112: text found after END; ignored	END擬似命令がソースファイルの終わりを指定しています。END擬似命令の後にあるすべてのテキストは無視されます。
W 113: unknown \$MODEL specifier; ignored	未知のモデルを指定しています。
W 114: \$MODEL may only be specified once, it remains " <i>model</i> "; ignored	複数のモデルを指定しています。
W 115: use ON or OFF after control name	指定したコントロールには、コントロール名の後にONかOFFを指定する必要があります。
W 116: unknown parameter " <i>parameter</i> " for <i>control-name</i> control	使用できるパラメータについては、コントロールの説明を参照してください。
W 118: inserted "extern <i>name</i> "	シンボル <i>name</i> が式の中で使用されていますが、EXTERN擬似命令で定義されていません。
W 119: " <i>name</i> " section has not the MAX attribute; ignoring RESET	

## 警告(W)

W 120: assembler debug information: cannot emit non-tiof expression for <i>label</i>	IEEE-695オブジェクトフォーマットによってサポートされない式が、SYMBレコードにあります。
W 121: changed alignment size to <i>size</i>	
W 123: expression: <i>type-error</i>	式が、アドレスで不正な動作を実行しているか、互換性のないメモリ空間を結合しています。
W 124: cannot purge macro during its own definition	
W 125: " <i>symbol</i> " is not a DEFINE symbol	DEFINEされていないシンボル、またはすでに定義解除されたシンボルをUNDEFしようとしてしました。
W 126: redefinition of " <i>define-symbol</i> "	現在の有効範囲ではシンボルがすでにDEFINEされています。シンボルはこのDEFINEに従って再定義されます。
W 127: redefinition of macro " <i>macro</i> "	マクロがすでに定義されています。マクロはこのマクロ定義に従って再定義されます。
W 128: number of macro arguments is less than definition	マクロを定義するとき、引数が少なすぎました。
W 129: number of macro arguments is greater than definition	マクロを定義するとき、引数が多すぎました。
W 130: DUPA needs at least one value argument	DUPA擬似命令には、ダミーパラメータと値パラメータの2つ以上の引数が必要です。
W 131: DUPF increment value gives empty macro	DUPFマクロに指定したステップ値がDUPFマクロ本体をスキップしています。
W 132: IF started in previous file " <i>file</i> ", line <i>line</i>	ENDIFまたはELSEプリプロセッサ擬似命令が、他のファイルのIF擬似命令と対応しています。
W 133: currently no macro expansion active	@CNT()@関数と@ARG()関数は、マクロ展開の中でのみ使用できます。
W 134: " <i>directive</i> " is not supported, skipped	指定された擬似命令は、本アセンブラではサポートされていません。
W 135: define symbol of " <i>define-symbol</i> " is not an identifier; skipped definition	コマンド行で-Dオプションに不正な識別子を指定しています。
W 137: label " <i>label</i> " defined <i>attribute</i> and <i>attribute</i>	ラベルはEXTERN擬似命令およびGLOBAL擬似命令で定義されています。
W 138: warning: <i>WARN-directive-arguments</i>	WARN擬似命令からの出力です。
W 139: expression must be between <i>hex-value</i> and <i>hex-value</i>	
W 140: expression must be between <i>value</i> and <i>value</i>	



## エラーメッセージ

## 警告(W)

W 141: <i>global/local</i> label " <i>name</i> " not defined in this module; made extern	ラベルが宣言されて使用されていますが、このソースファイルでは定義されていません。
W 170: code address maps to zero page	@CPAG関数で指定したコードオフセットが0ページにあります。
W 171: address offset must be between 0 and FFFF	@CADDR関数または@DADDR関数で指定したオフセットが大きすぎました。
W 172: page number must be between 0 and FF	@CADDR関数または@DADDR関数で指定したページ番号が大きすぎました。

## エラー(E)

E 200: <i>message</i> ; halting assembly	アセンブラが、ソースファイルの処理を停止します。
E 201: unexpected newline or line delimiter	構文チェッカーが、アセンブラの文法に準拠していない改行文字または行区切り文字を見つけました。
E 202: unexpected character: ' <i>character</i> '	構文チェッカーが、アセンブラの文法に準拠していない文字を見つけました。
E 203: illegal escape character in string constant	構文チェッカーが、文字列定数の中に、アセンブラの文法に準拠していない不正なエスケープ文字を見つけました。
E 204: I/O error: open intermediate file failed ( <i>file</i> )	アセンブラは、中間ファイルをオープンして、語彙スキャンフェーズを最適化しますが、アセンブラがこのファイルをオープンできません。
E 205: syntax error: expected <i>token</i> at <i>token</i>	構文チェッカーが、あるトークンを見つけようとしたしましたが、別のトークンが見つかりました。
E 206: syntax error: <i>token</i> unexpected	構文チェッカーが、予定外のトークンを見つけました。
E 207: syntax error: missing ':'	構文チェッカーが、ラベル定義またはメモリ空間変更子を見つけましたが、セミコロンが追加されていませんでした。
E 208: syntax error: missing ')'	構文チェッカーが、閉じかっこを見つけれませんでした。
E 209: invalid radix value, should be 2, 8, 10 or 16	RADIX擬似命令は、2、8、10、16のみを受け付けます。
E 210: syntax error	構文チェッカーがエラーを見つけました。
E 211: unknown model	正しいモデルを置き換えてください。
E 212: syntax error: expected <i>token</i>	構文チェッカーがトークンを見つけようとしたしますが、見つかりませんでした。
E 213: label " <i>label</i> " defined <i>attribute</i> and <i>attribute</i>	ラベルが、LOCAL擬似命令、GLOBAL擬似命令、EXTERN擬似命令のいずれかで定義されています。
E 214: illegal addressing mode	ニーモニックが不正なアドレッシングモードを使用しています。
E 215: not enough operands	ニーモニックに指定されているオペランドが少なすぎます。
E 216: too many operands	ニーモニックに指定されているオペランドが多すぎます。

## エラー(E)

E 217: <i>description</i>	ニーモニックのアセンブル中にエラーが見つかりました。
E 218: unknown mnemonic: " <i>name</i> "	アセンブラが、未知のニーモニックを見つけました。
E 219: this is not a hardware instruction (use \$OPTIMIZE OFF "H")	アセンブラが汎用命令を見つけましたが、-Oh(ハードウェアのみ)オプションまたは\$OPTIMIZE ON "H"コントロールが指定されていました。
E 223: unknown section " <i>name</i> "	SECT指示文で指定したセクション名が、DEFSECT指示文で定義されていません。
E 224: unknown label " <i>name</i> "	定義されていないラベルが使用されています。
E 225: invalid memory type	不正なメモリ変更子が指定されました。
E 226: unknown symbol attribute: <i>attribute</i>	
E 227: invalid memory attribute	アセンブラが、未知のロケーションカウンタまたはメモリマッピング属性を見つけました。
E 228: <i>attr</i> attribute needs a number	属性 <i>attr</i> には、1つのパラメータが必要です。
E 229: only one of the <i>name</i> attributes may be specified	
E 230: invalid section attribute: <i>name</i>	アセンブラが、未知のセクション属性を見つけました。
E 231: absolute section, expected "AT" expression	絶対セクションは、"AT <i>address</i> "式を使用して指定する必要があります。
E 232: MAX/OVERLAY sections need to be named sections	MAX属性またはOVERLAY属性を持つセクションには、名前を付ける必要があります。
E 233: <i>type</i> section cannot have <i>attribute</i> attribute	コードセクションには、CLEARまたはOVERLAY属性を指定することができません。
E 234: section attributes do not match earlier declaration	同じセクションの前の定義で、他の属性が使用されています。
E 235: redefinition of section	同じ名前の絶対セクションは一度しかロケートできません。
E 236: cannot evaluate expression of <i>descriptor</i>	一部の関数および擬似命令は、アセンブル中にその引数を評価する必要があります。
E 237: <i>descriptor</i> directive must have positive value	一部の擬似命令は、正の引数をとる必要があります。
E 238: Floating point numbers not allowed with DB directive	DB擬似命令は浮動小数点数をとることができません。
E 239: byte constant out of range	DB擬似命令は式をバイト単位で格納します。
E 240: word constant out of range	DW擬似命令は式をワード単位で格納します。
E 241: Cannot emit non ti of functions, replaced with integral value '0'	浮動小数点式および一部の関数は、IEEE-695オブジェクトフォーマットで表現することができません。
E 242: the <i>name</i> attribute must be specified	セクションには、CODE属性またはDATA属性がなければなりません。

## エラーメッセージ

## エラー(E)

E 243: use \$OBJECT OFF or \$OBJECT "object-file"	
E 244: unknown control "name"	指定されたコントロールが存在しません。
E 246: ENDM within IF/ENDIF	アセンブラがIF/ENDIFの間にENDM擬似命令を見つけました。
E 247: illegal condition code	アセンブラが命令の中に不正な条件コードを見つけました。
E 248: cannot evaluate origin expression of org "name: address"	絶対セクションのすべての起点は、オブジェクトファイルの作成前に評価されなければなりません。
E 249: incorrect argument types for function "function"	指定された引数が、予定外の型に評価されました。
E 250: tlof function not yet implemented: "function"	指定されたtlof関数は、まだ実装されていません。
E 251: @POS(,start) start argument past end of string	start引数が、最初のパラメータの文字列の長さより長くなっています。
E 252: second definition of label "label"	ラベルが、同じ有効範囲内で二度定義されています。
E 253: recursive definition of symbol "symbol"	シンボルの評価が、それ自身の値に依存しています。
E 254: missing closing '>' in include directive	構文チェッカーが、INCLUDE擬似命令で閉じかっ">"を見つけられませんでした。
E 255: could not open include file include-file	アセンブラが、このinclude-fileをオープンできませんでした。
E 256: integral divide by zero	式に、0による除算が含まれています。
E 257: unterminated string	すべての文字列は、開始した行と同じ行で終了しなければなりません。
E 258: unexpected characters after macro parameters, possible illegal white space	マクロパラメータの間にはスペースを入れることはできません。
E 259: COMMENT directive not permitted within a macro definition and conditional assembly	本アセンブラでは、MACRO/DUP定義またはIF/ELSE/ENDIF構成体でCOMMENT擬似命令を使用できません。
E 260: definition of "macro" unterminated, missing "endm"	マクロ定義がENDM擬似命令で終わっていません。
E 261: macro argument name may not start with an '_'	MACRO引数とDUP引数の最初にアンダースコアを付けることはできません。
E 262: cannot find "symbol"	"%"演算子または"?演算子の引数の定義をマクロ展開内で見つけることができませんでした。
E 263: cannot evaluate: "symbol", value is unknown at this point	マクロ展開内で%"演算子または"?演算子を使用されているシンボルは、まだ定義されていません。

## エラー(E)

E 264: cannot evaluate: "symbol", value depends on an unknown symbol	"%"演算子または"?演算子の引数を、マクロ展開内で評価することができません。
E 265: cannot evaluate argument of dup (unknown or location dependant symbols)	DUP擬似命令の引数が評価できませんでした。
E 266: dup argument must be integral	DUP擬似命令の引数が整数ではありません。
E 267: dup needs a parameter	DUP擬似命令の構文をチェックしてください。
E 268: ENDM without a corresponding MACRO or DUP definition	アセンブラが、対応するMACRO定義またはDUP定義がないENDM擬似命令を見つけました。
E 269: ELSE without a corresponding IF	アセンブラが、対応するIF擬似命令のないELSE擬似命令を見つけました。
E 270: ENDIF without a corresponding IF	アセンブラが、対応するIF擬似命令のないENDIF擬似命令を見つけました。
E 271: missing corresponding ENDIF	アセンブラが、対応するENDIF擬似命令のないIF擬似命令またはELSE擬似命令を見つけました。
E 272: label not permitted with this directive	一部の擬似命令はラベルを受け付けません。
E 273: wrong number of arguments for function	この関数には、もっと多い引数またはもっと少ない引数を指定する必要があります。
E 274: illegal argument for function	引数に不正な型があります。
E 275: expression not properly aligned	
E 276: immediate value must be between value and value	命令の即値オペランドは、この範囲の値のみを受け付けます。
E 277: address must be between \$address and \$address	アドレスオペランドが、この範囲内にありません。
E 278: operand must be an address	オペランドはアドレスでなければなりません、アドレス属性がありません。
E 279: address must be short	
E 280: address must be short	オペランドは短い範囲のアドレスでなければなりません。
E 281: illegal option "option"	アセンブラが、未知またはスペルミスのあるコマンド行オプションを見つけました。
E 282: "Symbols:" part not found in map file "name"	マップファイルが不完全な可能性があります。
E 283: "Sections:" part not found in map file "name"	マップファイルが不完全な可能性があります。
E 284: module "name" not found in map file "name"	マップファイルが不完全な可能性があります。

## エラーメッセージ

## エラー(E)

E 285: <i>file-kind</i> file will overwrite <i>file-kind</i> file	出力ファイルの1つが、コマンド行で指定したソースファイルや他の出力ファイルを上書きするとき、アセンブラは警告を出します。
E 286: \$CASE options must be given before any symbol definition	\$CASEオプションは、シンボルが定義される前にのみ指定することができます。
E 287: symbolic debug error: <i>message</i>	アセンブラが、シンボリックデバッグ(SYMB)命令でエラーを見つけました。
E 288: error in PAGE directive: <i>message</i>	PAGE擬似命令に指定された引数は制限に従っていません。
E 290: fail: <i>message</i>	FAIL擬似命令の出力。これはユーザ生成エラーです。
E 291: generated check: <i>message</i>	Cコンパイラとアセンブラとの間の整合性チェックです。
E 293: expression out of range	命令オペランドは、指定されたアドレス範囲内になければなりません。
E 294: expression must be between <i>hexvalue</i> and <i>hexvalue</i>	
E 295: expression must be between <i>value</i> and <i>value</i>	
E 296: optimizer error: <i>message</i>	オブティマイザがエラーを見つけました。
E 297: jump address must be a code address	ジャンプおよびジャンプサブルーチンには、コードメモリのターゲットアドレスを指定しなければなりません。
E 298: size depends on location, cannot evaluate	一部の構成体(特にalign擬似命令)のサイズは、メモリアドレスに依存します。

## 致命的エラー(F)

F 401: memory allocation error	空きメモリに対する要求がシステムによって拒絶されました。すべてのメモリが使用されています。
F 402: duplicate input filename " <i>file</i> " and " <i>file</i> "	アセンブラでは、コマンド行で入力ファイル名を1つだけ指定する必要があります。
F 403: error opening <i>file-kind</i> file: " <i>file-name</i> "	アセンブラがこのファイルをオープンできませんでした。
F 404: protection error: <i>message</i>	プロテクションキーがないか、またはIBM互換PCではありません。
F 405: I/O error	アセンブラが出力をファイルに書き込むことができません。
F 406: parser stack overflow	
F 407: symbolic debug output error	シンボリックデバッグ情報が、オブジェクトファイルに不正に書き込まれています。
F 408: illegal operator precedence	演算子の優先順テーブルが壊れています。
F 409: Assembler internal error	アセンブラが、内部の不整合を見つけました。

## 致命的エラー(F)

F 410: Assembler internal error: duplicate mufom " <i>symbol</i> " during rename	アセンブラは、有効範囲でローカルなシンボルの名前をすべて固有のシンボル名に変更します。ここでは、アセンブラが、固有の名前を生成できませんでした。
F 411: symbolic debug error: " <i>message</i> "	SYMB擬似命令の解析時にエラーが発生しました。
F 412: macro calls nested too deep (possible endless recursive call)	ネストするマクロ展開の数には制限があります。現在の制限は1000になっています。
F 413: cannot evaluate " <i>function</i> "	すでに処理されているはずの関数呼び出しが見つかりました。
F 414: cannot recover from previous errors, stopped	前に発生したエラーにより、アセンブラの内部状態が破壊され、プログラムのアセンブルが停止しました。
F 415: error opening temporary file	アセンブラは、デバッグ情報およびリストファイルを生成するとき、一時ファイルを使用します。これらの一時ファイルがオープンまたは作成できませんでした。
F 416: internal error in optimizer	オブティマイザがデッドロックの状態を検出しました。最適化オプションを付けずにアセンブルしてください。

## 起動コマンド

```
lk88 [option]...file...
```

## オプション

<b>-C</b>	大文字と小文字を区別しないでリンクします(デフォルトでは区別)。
<b>-L directory</b>	システムライブラリを探すための検索パスを追加します。
<b>-L</b>	システムライブラリの検索をスキップします。
<b>-M</b>	リンクマップ(.lnl)を生成します。
<b>-N</b>	重ね書きをオフにします。
<b>-O name</b>	生成されるマップファイルのベース名を指定します。
<b>-V</b>	バージョンヘッダのみを表示します。
<b>-c</b>	独立した呼び出しグラフファイル(.cal)を生成します。
<b>-e</b>	結果にエラーがある場合、その結果を消去します。
<b>-err</b>	エラーメッセージをエラーファイル(.elk)にリダイレクトします。
<b>-f file</b>	コマンド行の情報をfileから読み込みます。"-の場合stdinを示します。
<b>-I x</b>	システムライブラリlibx.a内も検索します。
<b>-o filename</b>	出力ファイルの名前を指定します。
<b>-r</b>	定義されていないシンボルの診断を抑制します。
<b>-u symbol</b>	symbolを、シンボルテーブルで定義されていないものとして入力します。
<b>-v or -t</b>	冗長オプション。進行中にそれぞれのファイル名をプリントします。
<b>-w n</b>	警告レベルがnより上のメッセージを抑制します。

## エラーメッセージ

## 警告(W)

W 100: Cannot create map file filename, turned off -M option	このファイルは作成できませんでした。
W 101: Illegal filename (filename) detected	不正な拡張子の付いたファイル名が検出されました。
W 102: Incomplete type specification, type index = Thexnumber	未知の型が参照されました。
W 103: Object name (name) differs from filename	オブジェクトファイルの内部名がファイル名と同じではありません。
W 104: '-o filename' option overwrites previous '-o filename'	2番目の-oオプションが見つかったため、最初の名前が破棄されます。
W 105: No object files found	起動文でファイルが指定されていません。
W 106: No search path for system libraries. Use -L or env "variable"	システムライブラリファイル(-lオプションで指定されたもの)に対して、環境変数またはオプション-Lで定義された検索パスが指定されていなければなりません。
W 108: Illegal option: option (-H or -¥? for help)	不正なオプションが検出されました。
W 109: Type not completely specified for symbol <symbol> in file	現在のファイルまたはこのファイルに、不完全な型指定があります。
W 110: Compatible types, different definitions for symbol <symbol> in file	互換性のある型の間に名前競合があります。
W 111: Signed/unsigned conflict for symbol <symbol> in file	両方の型のサイズは正しいですが、一方の型が符号なしで、もう一方の型が符号付きになっています。
W 112: Type conflict for symbol <symbol> in file	実数型の競合があります。
W 113: Table of contents of file out of date, not searched. (Use ar ts <name>)	arライブラリに、最新でないシンボルテーブルがあります。
W 114: No table of contents in file, not searched. (Use ar ts <name>)	arライブラリにシンボルテーブルがありません。
W 115: Library library contains ucode which is not supported	ucodeは、このリンクではサポートされていません。
W 116: Not all modules are translated with the same threshold (-G value)	ライブラリファイルに未知のフォーマットがあるか、ファイルが壊れています。
W 117: No type found for <symbol>. No type check performed	このシンボルに対する型が生成されていません。

## エラーメッセージ

## 警告(W)

W 118: Variable <i>&lt;name&gt;</i> , has incompatible external addressing modes with file <i>&lt;filename&gt;</i>	変数はまだ割り当てられていませんが、2つの外部参照が、オーバーラップしないアドレッシングモードで作成されています。
W 119: error from the Embedded Environment: <i>message</i> , switched off relaxed addressing mode check	リンクで組み込み環境が読み込み可能な場合、アドレッシングモードのチェックが解放されます。そのため、たとえば、データとして定義された変数が、HUGEとしてアクセスされる可能性があります。

## エラー(E)

E 200: Illegal object, assignment of non existing var <i>var</i>	MUFOM変数が存在しません。オブジェクトファイルが壊れています。
E 201: Bad magic number	指定されたライブラリファイルのマジックナンバーに問題があります。
E 202: Section <i>name</i> does not have the same attributes as already linked files	指定されたセクションに、異なる属性が指定されています。
E 203: Cannot open <i>filename</i>	このファイルが見つかりません。
E 204: Illegal reference in address of <i>name</i>	変数の式で不正なMUFOM変数が使用されています。オブジェクトファイルが壊れています。
E 205: Symbol ' <i>name</i> ' already defined in <i>&lt;name&gt;</i>	シンボルが二度定義されています。
E 206: Illegal object, multi assignment on <i>var</i>	前に発生したE 205の"already defined"エラーにより、MUFOM変数が、複数回割り当てられています。
E 207: Object for different processor characteristics	MAU単位のビット、アドレス単位のMAU、このオブジェクトのエンディアンなどが、最初にリンクしたオブジェクトと異なります。
E 208: Found unresolved external(s):	見つからないシンボルがあります。
E 209: Object format in <i>file</i> not supported	オブジェクトファイルに未知のフォーマットがあります。またはオブジェクトファイルが壊れています。
E 210: Library format in <i>file</i> not supported	ライブラリファイルに未知のフォーマットがあります。またはライブラリファイルが壊れています。
E 211: Function <i>&lt;function&gt;</i> cannot be added to the already built overlay pool <i>&lt;name&gt;</i>	オーバーレイプールが前のリンクアクションで構築されています。
E 212: Duplicate absolute section name <i>&lt;name&gt;</i>	絶対セクションが固定アドレスで始まっています。そのためリンクできません。
E 213: Section <i>&lt;name&gt;</i> does not have the same size as the already linked one	EQUAL属性を持つセクションが、他のリンク済みのセクションと同じサイズになっていません。

## エラー(E)

E 214: Missing section address for absolute section <i>&lt;name&gt;</i>	それぞれの絶対セクションには、オブジェクト内にセクションアドレスコマンドがなければなりません。オブジェクトファイルが壊れています。
E 215: Section <i>&lt;name&gt;</i> has a different address from the already linked one	2つの絶対セクションを同じ条件でリンク(重ね書き)することはできません。それぞれに同じアドレスがなければなりません。
E 216: Variable <i>&lt;name&gt;</i> , name <i>&lt;name&gt;</i> has incompatible external addressing modes	変数が、参照アドレッシング空間の外側に割り当てられています。
E 217: Variable <i>&lt;name&gt;</i> , has incompatible external addressing modes with file <i>&lt;filename&gt;</i>	変数はまだ割り当てられていませんが、2つの外部参照がオーバーラップしないアドレッシングモードで行われています。
E 218: Variable <i>&lt;name&gt;</i> , also referenced in <i>&lt;name&gt;</i> has an incompatible address format	現在のファイルとこのファイルとの間で、異なるアドレス形式のリンクが試みられました。
E 219: Not supported/illegal <i>feature</i> in object format <i>format</i>	このオブジェクトフォーマットでオプション/機能がサポートされていないか、または不正です。
E 220: page size (0 <i>hexvalue</i> ) overflow for section <i>&lt;name&gt;</i> with size 0 <i>hexvalue</i>	セクションが大きすぎてページに収まりません。
E 221: <i>message</i>	オブジェクトが生成したエラーです。
E 222: Address of <i>&lt;name&gt;</i> not defined	変数にアドレスが割り当てられていません。オブジェクトファイルが壊れています。

## 致命的エラー(F)

F 400: Cannot create file <i>filename</i>	このファイルを作成できませんでした。
F 401: Illegal object: Unknown command at offset <i>offset</i>	オブジェクトファイルで未知のコマンドが検出されました。オブジェクトファイルが壊れています。
F 402: Illegal object: Corrupted hex number at offset <i>offset</i>	16進数のバイトカウントが不正です。オブジェクトファイルが壊れています。
F 403: Illegal section index	範囲外のセクションインデックスが検出されました。オブジェクトファイルが壊れています。
F 404: Illegal object: Unknown hex value at offset <i>offset</i>	オブジェクトファイルで未知の変数が検出されました。オブジェクトファイルが壊れています。
F 405: Internal error <i>number</i>	内部の致命的エラーです。
F 406: <i>message</i>	キーがないか、またはIBM互換PCではありません。
F 407: Missing section size for section <i>&lt;name&gt;</i>	それぞれのセクションでは、オブジェクトにセクションサイズコマンドがなければなりません。オブジェクトファイルが壊れています。

## エラーメッセージ

## 致命的エラー(F)

F 408: Out of memory	より多くのメモリを割り当てようとして失敗しました。
F 409: Illegal object, offset <i>offset</i>	オブジェクトモジュールに不整合が見つかりました。
F 410: Illegal object	オブジェクトモジュールの未知のオフセットに不整合が見つかりました。
F 413: Only <i>name</i> object can be linked	他のプロセッサ用のオブジェクトは、リンクすることができません。
F 414: Input file <i>file</i> same as output file	入力ファイルと出力ファイルは同じにすることができません。
F 415: Demonstration package limits exceeded	このデモバージョンの制限を越えました。

## 冗長(V)

V 000: Abort!	プログラムがユーザによってアボートされました。
V 001: Extracting files	ライブラリからファイルを抽出することを示す冗長メッセージ。
V 002: File currently in progress:	ファイルを現在処理していることを示す冗長メッセージ。
V 003: Starting pass <i>number</i>	このパスの開始を示す冗長メッセージ。
V 004: Rescanning...	ライブラリの再スキャンを示す冗長メッセージ。
V 005: Removing file <i>file</i>	削除を示す冗長メッセージ。
V 006: Object file <i>file</i> format <i>format</i>	指定されたオブジェクトファイルに標準ツールチェーンオブジェクトフォーマットTIOF-695がありません。
V 007: Library <i>file</i> format <i>format</i>	指定されたライブラリに、標準ツールチェーンar88フォーマットがありません。
V 008: Embedded environment <i>name</i> read, relaxed addressing mode check enabled	組み込み環境が読み込まれました。

## 起動コマンド

```
alc88 project_path file.out file.inf
```

## エラーメッセージ

Illegal Inf File	アドバンスドロケータ定義ファイル(.inf)が不正です。
Duplicate Memory -- 0xn timer ~ 0xn timer & 0xn timer ~ 0xn timer	0xn timer ~ 0xn timerと0xn timer ~ 0xn timerでメモリ割り当てが重複しています。
No physical memory available for xxxx	シンボルxxxxを割り当てておくべき、指定されたアドレスが存在しません。
Duplicate Symbol Name -- xxxx	シンボル名xxxxが重複しています。
Cannot find 0xn timer bytes for xxxx section	セクションxxxxの割り当てに必要な0xn timerバイトの領域がありません。
Found unresolved external -- xxxx	外部シンボル(Extern)xxxxの情報がありません。
There is no stack area	内蔵RAMエリアが足りないため、スタックエリアが確保できませんでした。
Absolute address 0xn timer occupied	0xn timerで始まる絶対アドレスセクションのエリアが既に他のエリアに占有されています。

## 起動コマンド

```
lc88 [option]...[file]...
```

## オプション

<b>-M</b>	ロケータマップファイル(.map)を生成します。
<b>-S space</b>	特定のspaceを生成します。
<b>-V</b>	バージョンヘッダのみを表示します。
<b>-d file</b>	記述ファイルの情報をfileから読み込みます。 "-"の場合stdinを示します。
<b>-e</b>	結果にエラーがある場合、その結果を消去します。
<b>-err</b>	エラーメッセージをリダイレクトします。(.elc)
<b>-f file</b>	コマンド行の情報をfileから読み込みます。 "-"の場合stdinを示します。
<b>-f format</b>	出力フォーマットを指定します。
<b>-o filename</b>	出力ファイルの名前を指定します。
<b>-p</b>	stdoutにソフトウェア部分のプロポーザルを作成します。
<b>-v</b>	冗長オプション。進行中にそれぞれのファイル名をプリントします。
<b>-w n</b>	警告レベルがnより上のメッセージを抑制します。

## エラーメッセージ

## 警告(W)

W 100: Maximum buffer size for <i>name</i> is <i>size</i> (Adjusted)	このフォーマットの場合、最大バッファサイズが定義されています。
W 101: Cannot create map file <i>filename</i> , turned off -M option	このファイルが作成できませんでした。
W 102: Only one -g switch allowed, ignored -g before <i>name</i>	デバッグできる.outファイルは1つだけです。
W 104: Found a negative length for section <i>name</i> , made it positive	負の値の長さをとることができるのは、スタックセクションのみです。
W 107: Inserted ' <i>name</i> ' keyword at line <i>line</i>	記述ファイルにないキーワードが挿入されました。
W 108: Object name ( <i>name</i> ) differs from filename	オブジェクトファイルの内部名がファイル名と同じではありません。
W 110: Redefinition of system start point	通常、システムテーブル(__lc_pm)にアクセスできるロードモジュールは1つだけです。
W 111: Two -o options, output name will be <i>name</i>	2番目の-oオプションが見つかりました。メッセージには、有効な名前が示されます。
W 112: Copy table not referenced, initial data is not copied	layout部分でcopy文を使用すると、初期データがROMにロケートされます。
W 113: No .out files found to locate	起動構文で指定されたファイルがありません。
W 114: Cannot find start label <i>label</i>	開始点が見つかりませんでした。
W 116: Redefinition of name at line <i>line</i>	識別子が二度定義されています。
W 119: File <i>filename</i> not found in the argument list	ロケートする必要があるすべてのファイルは引数として指定しなければなりません。
W 120: unrecognized name option <i>&lt;name&gt;</i> at line <i>line</i> (inserted ' <i>name</i> ')	オプションの割り当てが不正です。
W 121: Ignored illegal sub-option ' <i>name</i> ' for <i>name</i>	不正なフォーマットサブオプションが検出されました。
W 122: Illegal option: <i>option</i> (-H or -¥? for help)	不正なオプションが検出されました。
W 123: Inserted <i>character</i> at line <i>line</i>	この文字が、記述ファイルにありませんでした。
W 124: Attribute <i>attribute</i> at line <i>line</i> unknown	記述ファイルで、未知の属性が指定されています。
W 125: Copy table not referenced, blank sections are not cleared	属性がブランクになっているセクションが検出されましたが、コピーテーブルは参照されていません。ロケータは、コピーテーブルにスタートアップモジュールについての情報を生成して、スタートアップ時にブランクセクションを消去します。



## エラーメッセージ

## 警告(W)

W 127: Layout <i>name</i> not found	指定されているファイルで使用するレイアウトが、layout部分で定義されていなければなりません。
W 130: Physical block <i>name</i> assigned for the second time to a layout	ブロックをlayoutブロックに複数回割り当てることはできません。
W 136: Removed <i>character</i> at line <i>line</i>	ここでは、この文字は必要ありません。
W 137: Cluster <i>name</i> declared twice (layout part)	指定されたクラスタが二度宣言されています。
W 138: Absolute section <i>name</i> at non-existing memory address <i>0xhexnumber</i>	絶対セクションのアドレスが物理メモリの外に指定されています。
W 139: <i>message</i>	組み込み環境からの警告メッセージです。
W 140: File <i>filename</i> not found as a parameter	ロケータ記述ファイル(software部分)で定義されたすべてのプロセスは、起動行で指定しなければなりません。
W 141: Unknown space <i>&lt;name&gt;</i> in -S option	-Sオプションで未知の空間名が指定されました。
W 142: No room for section <i>name</i> in read-only memory, trying writable memory ...	読み取り専用属性を持つセクションは、読み取り専用メモリに配置することができません。このセクションは、書き込み可能メモリに配置されます。

## エラー(E)

E 200: Absolute address <i>0xhexnumber</i> occupied	絶対アドレスが要求されましたが、そのアドレスはすでに他のセクションによって占有されています。
E 201: No physical memory available for section <i>name</i>	絶対アドレスが要求されましたが、そのアドレスには物理メモリがありません。
E 202: Section <i>name</i> with mau size <i>size</i> cannot be located in an addressing mode with mau size <i>size</i>	ビットセクションは、バイト専用アドレッシングモードでロケートすることができません。
E 203: Illegal object, assignment of non existing var <i>var</i>	MUFOM変数が存在しません。
E 204: Cannot duplicate section ' <i>name</i> ' due to hardware limitations	プロセスを複数回ロケートする必要がありますが、セクションが、メモリ管理機能のない仮想空間にマッピングされています。
E 205: Cannot find section for <i>name</i>	セクションのない変数が見つかりました。
E 206: Size limit for the section group containing section <i>name</i> exceeded by <i>0xhexnumber</i> bytes	小さなセクションが、これ以上ページに収まりません。
E 207: Cannot open <i>filename</i>	このファイルが見つかりません。

## エラー(E)

E 208: Cannot find a cluster for section <i>name</i>	書き込み可能なメモリがないか、アドレッシングモードが未知です。
E 210: Unrecognized keyword <i>&lt;name&gt;</i> at line <i>line</i>	記述ファイルで未知のキーワードが使用されています。
E 211: Cannot find <i>0xhexnumber</i> bytes for section <i>name</i> (fixed mapping)	仮想メモリまたは物理メモリのいずれかが占有されています。また物理メモリがまったくない可能性もあります。
E 213: The physical memory of <i>name</i> cannot be addressing in space <i>name</i>	マッピングが失敗しました。仮想アドレス空間が残っていません。
E 214: Cannot map section <i>name</i> , virtual memory address occupied	絶対マッピングが失敗しました。
E 215: Available space within <i>name</i> exceeded by <i>number</i> bytes for section <i>name</i>	アドレッシングモードで利用可能なアドレッシング空間がなくなりました。
E 217: No room for section <i>name</i> in cluster <i>name</i>	.dscファイルで定義されているクラスタのサイズが小さすぎます。
E 218: Missing <i>identifier</i> at line <i>line</i>	識別子を指定しなければなりません。
E 219: Missing ' <i>'</i> ' at line <i>line</i>	閉じかっこが足りません。
E 220: Symbol ' <i>symbol</i> ' already defined in <i>&lt;name&gt;</i>	シンボルが二度定義されています。
E 221: Illegal object, multi assignment on <i>var</i>	MUFOM変数が複数回割り当てられています。
E 223: No software description found	それぞれの入力ファイルは、.dscファイルのソフトウェア記述に記述しなければなりません。
E 224: Missing <i>&lt;length&gt;</i> keyword in block ' <i>name</i> ' at line <i>line</i>	ハードウェア記述に長さの定義がありません。
E 225: Missing <i>&lt;keyword&gt;</i> keyword in space ' <i>name</i> ' at line <i>line</i>	このマッピングに対して、キーワードを指定しなければなりません。
E 227: Missing <i>&lt;start&gt;</i> keyword in block ' <i>name</i> ' at line <i>line</i>	ハードウェア記述に開始の定義がありません。
E 230: Cannot locate section <i>name</i> , requested address occupied	絶対アドレスが要求されましたが、そのアドレスはすでに他のプロセスまたはセクションによって占有されています。
E 232: Found file <i>filename</i> not defined in the description file	ロケートするすべてのファイルには、それに対する定義レコードが記述ファイルに必要になります。
E 233: Environment variable too long in line <i>line</i>	記述ファイルにある環境変数の文字数が多すぎます。
E 235: Unknown section size for section <i>name</i>	この.outファイルにセクションサイズが見つかりませんでした。実際には.outファイルが壊れています。

## エラーメッセージ

## エラー(E)

E 236: Unrecoverable specification at line <i>line</i>	記述ファイルに回復不能なエラーがあります。
E 238: Found unresolved external(s):	ロケート時に、すべての外部参照が解決されなければなりません。
E 239: Absolute address <i>addr.addr</i> not found	この空間に、絶対アドレスが見つかりませんでした。
E 240: Virtual memory space <i>name</i> not found	記述ファイルにある、このファイルのsoftware部分に、存在しないメモリ空間が記述されています。
E 241: Object for different processor characteristics	MAU単位のビット、アドレス単位のMAU、このオブジェクトのエンディアンなどが、最初にリンクしたオブジェクトと異なります。
E 242: <i>message</i>	オブジェクトが生成したエラーです。
E 244: Missing <i>name</i> part	記述ファイルに、この部分が見つかりませんでした。
E 245: Illegal <i>name</i> value at line <i>line</i>	記述ファイルに、有効でない値が見つかりました。
E 246: Identifier cannot be a number at line <i>line</i>	記述ファイルに、有効でない識別子が見つかりました。
E 247: Incomplete type specification, type index = <i>Thexnumber</i>	このファイルにより、未知の型が参照されました。オブジェクトファイルが壊れています。
E 250: Address conflict between block <i>block1</i> and <i>block2</i> (memory part)	記述ファイルのmemory部分にオーバーラップするアドレスがあります。
E 251: Cannot find 0 <i>hexnumber</i> bytes for section <i>section</i> in block <i>block</i>	セクションをロケートしなければならない物理ブロックに空きがありません。
E 255: Section ' <i>name</i> ' defined more than once at line <i>line</i>	1つのlayout/loadmod部分でセクションを複数回宣言することはできません。
E 258: Cannot allocate reserved space for process <i>number</i>	空間の予約部分のメモリが占有されています。
E 261: User assert: <i>message</i>	ユーザがプログラムした表明が失敗しました。
E 262: Label ' <i>name</i> ' defined more than once in the software part	記述ファイルで定義されているラベルは固有のものでなければなりません。
E 264: <i>message</i>	組み込み環境からのエラーメッセージです。
E 265: Unknown section address for absolute section <i>name</i>	この.outファイルにセクションアドレスが見つかりませんでした。実際には.outファイルが壊れています。
E 266: %s %s not (yet) supported	要求された機能は、このリリースでは(まだ)サポートされていません。

## 致命的エラー(F)

F 400: Cannot create file <i>filename</i>	このファイルを作成できませんでした。
F 401: Cannot open <i>filename</i>	このファイルが見つかりません。
F 402: Illegal object: Unknown command at offset <i>offset</i>	オブジェクトファイルで未知のコマンドが検出されました。オブジェクトファイルが壊れています。
F 403: Illegal filename ( <i>name</i> ) detected	不正な拡張子の付いたファイル名がコマンド行で検出されました。
F 404: Illegal object: Corrupted hex number at offset <i>offset</i>	16進数のバイトカウントが不正です。オブジェクトファイルが壊れています。
F 405: Illegal section index	範囲外のセクションインデックスが検出されました。
F 406: Illegal object: Unknown hex value at offset <i>offset</i>	オブジェクトファイルで未知の変数が検出されました。オブジェクトファイルが壊れています。
F 407: No description file found	ロケータには、システムのハードウェアとソフトウェアを記述している記述ファイルが必要になります。
F 408: <i>message</i>	プロテクションキーがないか、またはIBM互換PCではありません。
F 410: Only one description file allowed	ロケータは、記述ファイルを1つだけ受け付けます。
F 411: Out of memory	より多くのメモリを割り当てようとして失敗しました。
F 412: Illegal object, offset <i>offset</i>	オブジェクトモジュールに不整合が見つかりました。
F 413: Illegal object	オブジェクトモジュールの未知のオフセットに不整合が見つかりました。
F 415: Only <i>name</i> .out files can be located	他のプロセス用のオブジェクトは、ロケートすることができません。
F 416: Unrecoverable error at line <i>line</i> , <i>name</i>	記述ファイルのこの部分に回復不能なエラーがあります。
F 417: Overlaying not yet done	この.outファイルでは重ね書きがまだ実行されていません。
F 418: No layout found, or layout not consistent	レイアウトに構文エラーがある場合、そのレイアウトがロケータで使用できない可能性があります。
F 419: <i>message</i>	組み込み環境からの致命的エラーメッセージです。
F 420: Demonstration package limits exceeded	このデモバージョンの制限を越えました。

## エラーメッセージ

## 冗長(V)

V 000: File currently in progress:	冗長メッセージ。次の行に、単一のファイル名が、処理される順にプリントされます。
V 001: Output format: <i>name</i>	出力フォーマットの生成を示す冗長メッセージ。
V 002: Starting pass <i>number</i>	このパスの開始を示す冗長メッセージ。
V 003: Abort!	プログラムがユーザによってアボートされました。
V 004: Warning level <i>number</i>	使用されている警告レベルを報告する冗長メッセージ。
V 005: Removing file <i>file</i>	削除を示す冗長メッセージ。
V 006: Found file < <i>filename</i> > via <i>path pathname</i>	記述(インクルード)ファイルが標準ディレクトリにありませんでした。
V 007: <i>message</i>	組み込み環境からの冗長メッセージです。

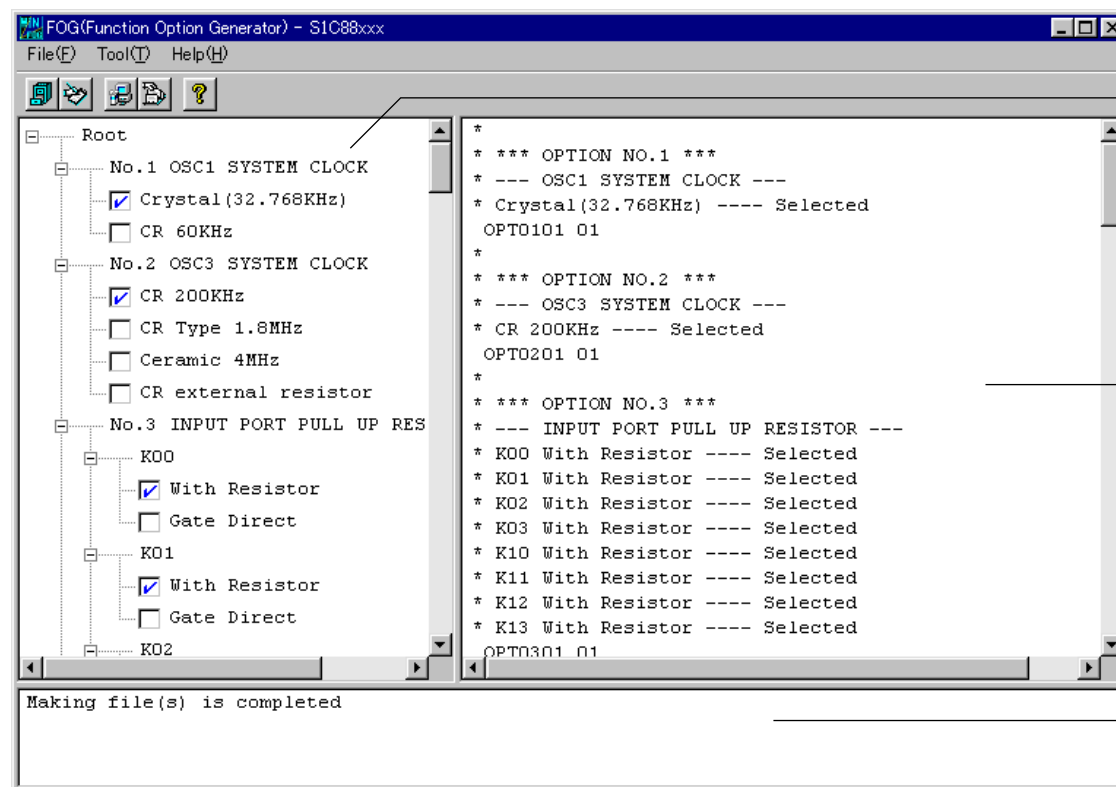
## キーワード

<b>address</b>	絶対アドレスを指定します。
<b>amode</b>	アドレッシングモードを指定します。
<b>assert</b>	表明が失敗した場合エラーを出します。
<b>attribute</b>	属性をクラスタ、セクション、スタック、ヒープに割り当てます。
<b>block</b>	物理メモリ領域を定義します。
<b>bus</b>	アドレスバスを定義します。
<b>chips</b>	CPUチップを定義します。
<b>cluster</b>	クラスタの順序と配置を指定します。
<b>copy</b>	データセクションのROMコピーの配置を定義します。
<b>cpu</b>	cpu部分を定義します。
<b>dst</b>	デスティネーションアドレスを指定します。
<b>fixed</b>	メモリマップの固定ポイントを定義します。
<b>gap</b>	動的なメモリギャップを予約します。
<b>heap</b>	ヒープを定義します。
<b>label</b>	仮想アドレスラベルを定義します。
<b>layout</b>	layout記述の最初です。
<b>length</b>	スタック、ヒープ、物理ブロック、予約空間の長さを指定します。
<b>load_mod</b>	ロードモジュールを定義します(プロセス)。
<b>map</b>	ソースアドレスをデスティネーションアドレスにマッピングします。
<b>mau</b>	最小アドレス可能単位を(ビット単位で)定義します。
<b>mem</b>	チップの物理開始アドレスを定義します。
<b>memory</b>	memory部分を定義します。
<b>regsfr</b>	デバッガが使用するレジスタファイルを指定します。
<b>reserved</b>	メモリを予約します。
<b>section</b>	セクションがロケートされる方法を定義します。
<b>selection</b>	セクションをクラスタにグループ化するときの属性を指定します。
<b>size</b>	アドレス空間またはメモリのサイズを指定します。
<b>software</b>	software部分を定義します。
<b>space</b>	アドレス空間を定義するが、メモリブロックを指定します。
<b>src</b>	ソースアドレスを指定します。
<b>stack</b>	stackセクションを定義します。
<b>start</b>	他の開始ラベルを定義します。
<b>table</b>	tableセクションを定義します。

## 概要

ファンクションオプションジェネレータwinfogは、I/Oポート機能など、いくつかのハードウェア仕様のマスクパターン生成のためのファイルを作成するソフトウェアツールです。  
また、ICEを用いてデバッグを行う際に必要なマスクオプション設定用ファイルも同時に作成できます。

## ウィンドウ



### オプションリスト領域

機種情報定義ファイル(s1c88xxx.ini)で設定される、マスクオプションの一覧です。チェックボックスを使用して、各オプションを選択します。  
チェックマーク(✓)は現在選択されているオプションを示します。

### ファンクションオプションドキュメント領域

オプションの選択内容がファンクションオプションドキュメントの形式で表示されます。ファンクションオプションドキュメントファイルには、この領域の表示内容が出力されます。オプションリスト領域で選択項目を変更すると、表示が即時更新されます。

### メッセージ領域

[Tool]メニューから[Generate]を選択、あるいは[Generate]ボタンをクリックしてファイルを作成した際に、その結果を示すメッセージを表示します。

## ボタン

### ツールバー

**[Open]**ボタン

ファンクションオプションドキュメントファイルを開きます。

**[Generate]**ボタン

オプションリストの選択内容でファイルを作成します。

**[Setup]**ボタン

作成日や出力ファイル名、ファンクションオプションドキュメントファイルに含めるコメントなどを設定します。

**[Device INI Select]**ボタン

機種情報定義ファイル(s1c88xxx.ini)をロードします。

**[Help]**ボタン

winfogのバージョンを表示します。

## メニュー

### [File]メニュー

File(F)

Open(O)

End(X)

**Open**

ファンクションオプションドキュメントファイルを開きます。

**End**

winfogを終了します。

### [Tool]メニュー

Tool(T)

Generate(G)

Setup(S)

Device INI Select

**Generate**

オプションリストの選択内容でファイルを作成します。

**Setup**

作成日や出力ファイル名、ファンクションオプションドキュメントファイルに含めるコメントなどを設定します。

**Device INI Select**

機種情報定義ファイル(s1c88xxx.ini)をロードします。

### [Help]メニュー

Help(H)

Version(A)

**Version**

winfogのバージョンを表示します。

## エラーメッセージ

File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。
Illegal character	入力禁止文字が入力された。
Please input file name	ファイル名が未入力。
Can't open File : xxxx	ファイル(xxxx)がオープンできない。
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。
INI file does not include FOG information	指定した機種情報定義ファイル(.ini)にファンクションオプション情報が含まれていない。
Function Option document file is not found	指定したファンクションオプションドキュメントファイルが存在しない。
Function Option document file does not match INI file	指定したファンクションオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。
A lot of parameter	コマンドラインの引数が多すぎる。
Making file(s) is completed [xxxx is no data exist]	ファイル作成完了。ただし、作成したファイル(xxxx)にはデータが含まれていない。
Can't open File: xxxx	Generate実行時、ファイル(xxxx)がオープンできない。
Making file(s) is not completed	
Can't write File: xxxx	Generate実行時、ファイル(xxxx)に書き込みができない。
Making file(s) is not completed	

## ワーニングメッセージ

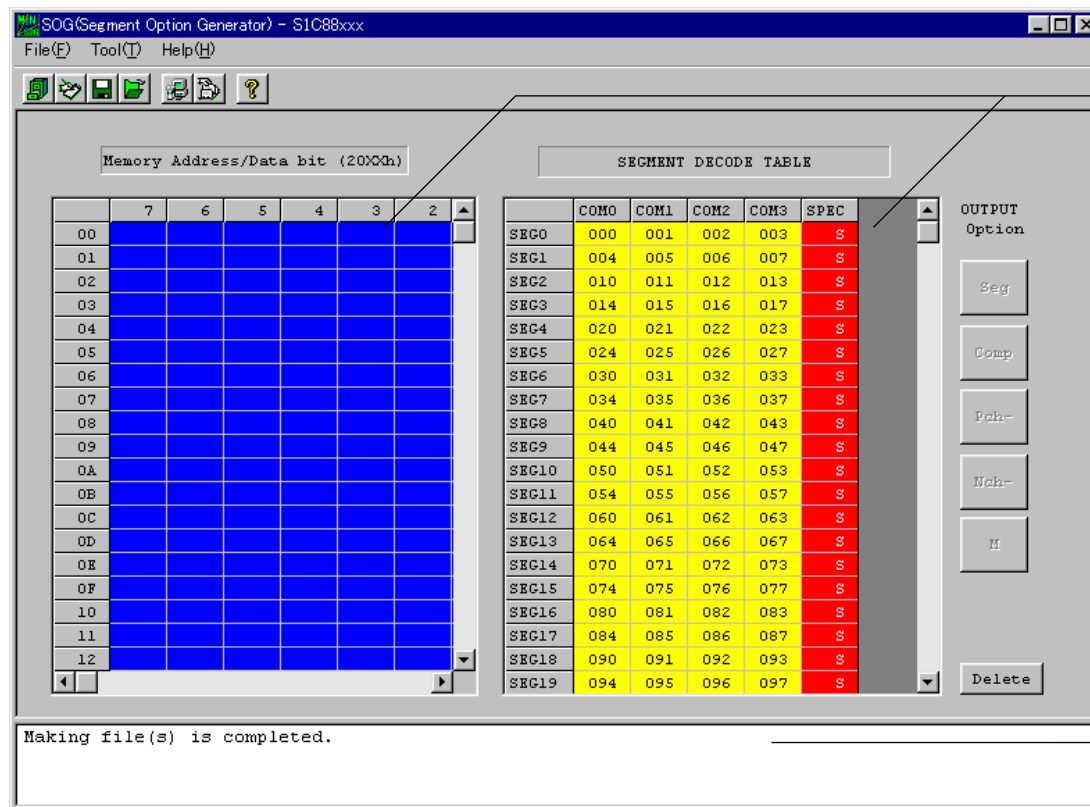
Are you file update? xxxx is already exist	上書き確認メッセージ (指定したファイルは既に存在する。)
---	----------------------------------

## 概要

セグメントオプションジェネレータwinsogは、LCD出力端子の出力仕様、表示メモリとLCD出力端子割り付けのマスクパターン生成のためのファイルを作成するソフトウェアツールです。

また、ICEを用いてデバッグを行う際に必要なマスクオプション設定用ファイルも同時に作成できます。

## ウィンドウ



## オプション設定領域

表示メモリマップとセグメントデコードテーブル、端子の仕様を選択するボタンで構成されています。表示メモリマップとセグメントデコードテーブルのセルをクリックすることで、表示メモリアドレス/ビットの割り付けが行えます。

- Seg** LCDセグメント出力を選択します。
- Comp** DC-コンプリメンタリ出力を選択します。
- Pch-** DC-Pchオープンドレイン出力を選択します。
- Nch-** DC-Nchオープンドレイン出力を選択します。
- M** セグメント/コモン共有出力を選択します。
- Delete** 選択したセグメント割り付けをクリアします。

## メッセージ領域

[Tool]メニューから[Generate]を選択、あるいは[Generate]ボタンをクリックしてファイルを作成した際に、その結果を示すメッセージを表示します。

## ボタン

## ツールバー

**[Open]**ボタン

セグメントオプションドキュメントファイルを開きます。

**[Save]**ボタン

現在のオプション設定内容をファイル(セグメント割り付けデータファイル)に保存します。

**[Load]**ボタン

セグメント割り付けデータファイルを読み込みます。

**[Generate]**ボタン

オプションリストの選択内容でファイルを作成します。

**[Setup]**ボタン

作成日や出力ファイル名、セグメントオプションドキュメントファイルに含めるコメントなどを設定します。

**[Device INI Select]**ボタン

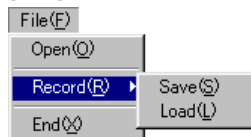
機種情報定義ファイル(s1c88xxx.ini)をロードします。

**[Help]**ボタン

winsogのバージョンを表示します。

## メニュー

## [File]メニュー

**Open**

セグメントオプションドキュメントファイルを開きます。

**Record - Save**

現在のオプション設定内容をファイル(セグメント割り付けデータファイル)に保存します。

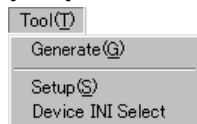
**Record - Load**

セグメント割り付けデータファイルを読み込みます。

**End**

winsogを終了します。

## [Tool]メニュー

**Generate**

オプションリストの選択内容でファイルを作成します。

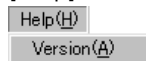
**Setup**

作成日や出力ファイル名、ファンクションオプションドキュメントファイルに含めるコメントなどを設定します。

**Device INI Select**

機種情報定義ファイル(s1c88xxx.ini)をロードします。

## [Help]メニュー

**Version**

winsogのバージョンを表示します。

## エラーメッセージ

File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。
Illegal character	入力禁止文字が入力された。
Please input file name	ファイル名が未入力。
Can't open File : xxxx	ファイル(yyyy)がオープンできない。
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。
INI file does not include SOG information	指定した機種情報定義ファイル(.ini)にセグメントオプション情報が含まれていない。
Function Option document file is not found	指定したファンクションオプションドキュメントファイルが存在しない。
Function Option document file does not match INI file	指定したファンクションオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。
Segment Option document file is not found	指定したセグメントオプションドキュメントファイルが存在しない。
Segment Option document file does not match INI file	指定したセグメントオプションドキュメントファイルの内容が機種情報定義ファイル(.ini)と異なる。
Segment assignment data file is not found	指定したセグメント割り付けデータファイルが存在しない。
Segment assignment data file does not match INI file	指定したセグメント割り付けデータファイルの内容が機種情報定義ファイル(.ini)と異なる。
Can't open File: xxxx	Generate実行時、ファイル(yyyy)がオープンできない。
Making file(s) is not completed	Generate実行時、ファイル(yyyy)に書き込みができない。
Can't write File: xxxx	
Making file(s) is not completed	
ERROR: SPEC is not set	空白のSPECセルがある状態でGenerateを実行した。
Making file(s) is not completed	

## ワーニングメッセージ

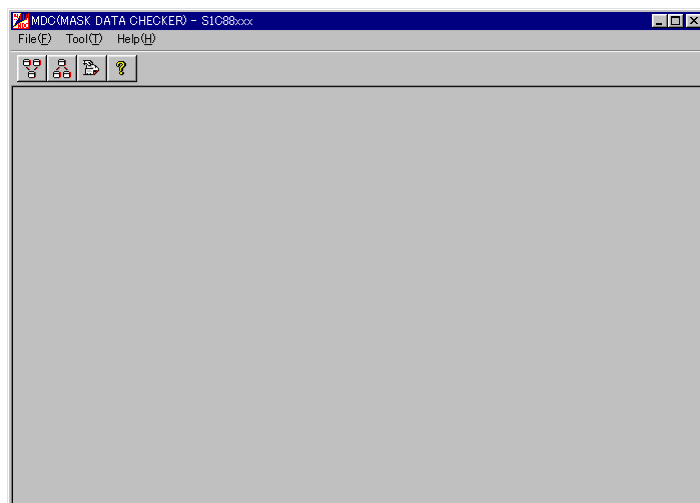
Are you file update?	上書き確認メッセージ
xxxx is already exist	(指定したファイルは既に存在する。)



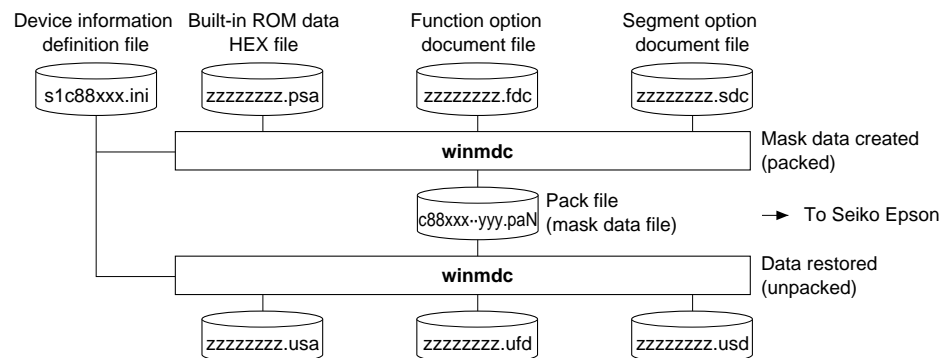
## 概要

マスクデータチェッカ<winmdc>は、内蔵ROM未使用領域FF詰めユーティリティ<fil88xxx>によって生成された内蔵ROMデータHEXファイル、ファンクションオプションジェネレータ<winfog>によって生成されたファンクションオプションドキュメントファイル、セグメントオプションジェネレータ<winsog>によって生成されたセグメントオプションドキュメントファイルの各フォーマットをチェックし、マスクパターン生成のためのデータファイルを作成するソフトウェアツールです。

また、作成されたマスクデータファイルを元のファイル形式に復元する機能も持っています。



## フローチャート



## ボタン

## ツールバー

**[Pack]ボタン**

ROMデータファイルとオプションドキュメントファイルをバックして、提出用のマスクデータファイルを作成します。

**[Unpack]ボタン**

バック後のファイルから元の形式のファイルを復元します。

**[Device INI Select]ボタン**

機種情報定義ファイル(s1c88xxx.ini)をロードします。

**[Help]ボタン**

winmdcのバージョンを表示します。

## メニュー

**[File]メニュー****End**

winmdcを終了します。

File(F)

End(X)

**[Tool]メニュー****Pack**

ROMデータファイルとオプションドキュメントファイルをバックして、提出用のマスクデータファイルを作成します。

**Unpack**

バック後のファイルから元の形式のファイルを復元します。

**Device INI Select**

機種情報定義ファイル(s1c88xxx.ini)をロードします。

Tool(T)

Pack(P)

Unpack(U)

Device INI Select

**[Help]メニュー****Version**

winmdcのバージョンを表示します。

Help(H)

Version(A)

## I/Oエラーメッセージ

File name error	ファイル名または拡張子名の文字数が使用可能範囲を超えている。
Illegal character	入力禁止文字が入力された。
Please input file name	ファイル名が未入力。
INI file is not found	指定した機種情報定義ファイル(.ini)が存在しない。
INI file does not include MDC information	指定した機種情報定義ファイル(.ini)にMDC情報が含まれていない。
Can't open file : xxxx	ファイル(yyyy)がオープンできない。
Can't write file: xxxx	ファイル(yyyy)に書き込みができない。

## ROMデータエラーメッセージ

Hex data error: Not S record.	データが"S"で始まっていない。
Hex data error: Data is not sequential.	データが昇順に並んでいない。
Hex data error: Illegal data.	不当なキャラクタがある。
Hex data error: Too many data in one line.	1行中のデータ数が多すぎる。
Hex data error: Check sum error.	チェックサムが合わない。
Hex data error: ROM capacity over.	データ容量が大きいの。(データサイズ>ROMサイズ)
Hex data error: Not enough the ROM data.	データ容量が少ない。(データサイズ<ROMサイズ)
Hex data error: Illegal start mark.	スタートマークが不当である。
Hex data error: Illegal end mark.	エンドマークが不当である。
Hex data error: Illegal comment.	データの最初の機種名表示が不当である。

## ファンクションオプションデータエラーメッセージ

Option data error : Illegal model name.	機種名が不当である。
Option data error : Illegal version.	バージョンが不当である。
Option data error : Illegal option number.	オプションNo.が不当である。
Option data error : Illegal select number.	選択肢No.が不当である。
Option data error : Mask data is not enough.	マスクデータが充分でない。
Option data error : Illegal start mark.	スタートマークが不当である。
Option data error : Illegal end mark.	エンドマークが不当である。

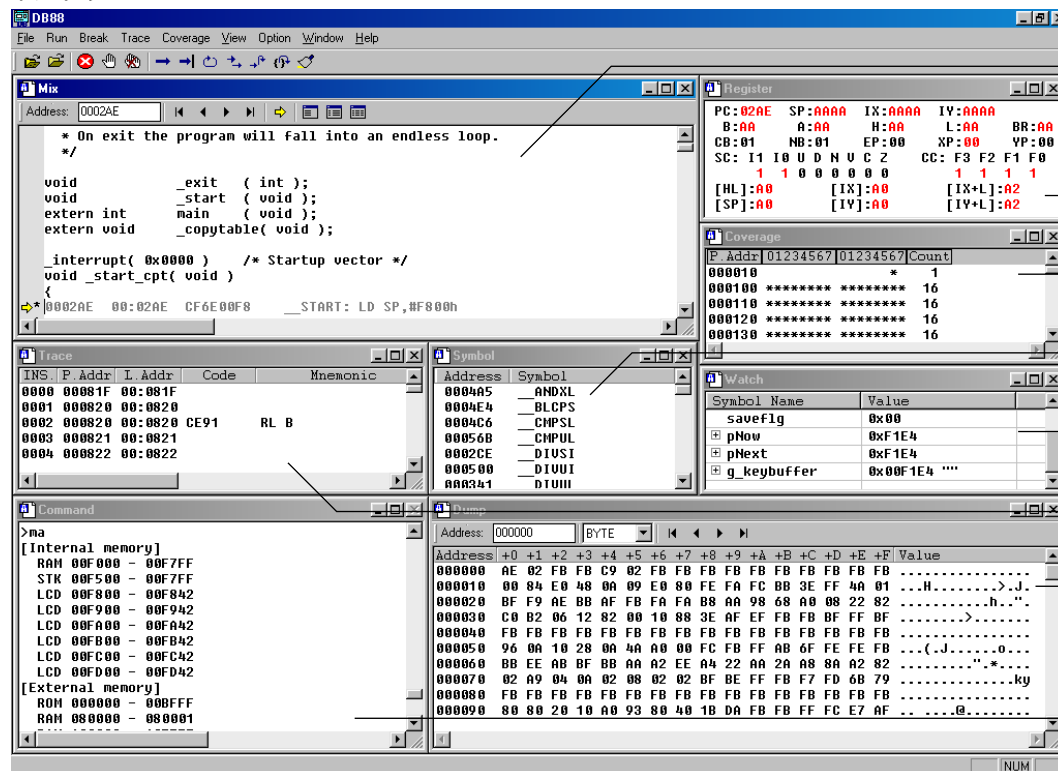
## セグメントオプションデータエラーメッセージ

LCD segment data error : Illegal model name.	機種名が不当である。
LCD segment data error : Illegal version.	バージョンが不当である。
LCD segment data error : Illegal segment No.	セグメントNo.が不当である。
LCD segment data error : Illegal segment area.	表示メモリのアドレスが範囲外である。
LCD segment data error : Illegal segment output specification.	出力仕様が不当である。
LCD segment data error : Illegal data in this line.	16進数と出力仕様以外の記述がある。
LCD segment data error : Data is not enough.	セグメントデータが充分でない。
LCD segment data error : Illegal start mark.	スタートマークが不当である。
LCD segment data error : Illegal end mark.	エンドマークが不当である。

## 概要

ハードウェアツールとして用意されているICEを制御してデバッグを行うソフトウェアです。ブレークやステップ実行など、頻繁に使用するコマンドはツールバーに登録されており、キーボード操作の量を抑えています。また、ソースやレジスタ内容、コマンド実行結果がマルチウィンドウ上に表示できるため、デバッグ作業が効率良く行えます。

## ウィンドウ



## [Source]ウィンドウ

プログラムを逆アセンブルコード、ソースコードまたは両方をミックスした形で表示します。

## [Register]ウィンドウ

レジスタ値およびレジスタで指定されるメモリデータを表示します。

## [Coverage]ウィンドウ

カバレッジデータを表示します。

## [Symbol]ウィンドウ

シンボル情報を表示します。

## [Watch]ウィンドウ

監視しているシンボルの値を表示します。

## [Trace]ウィンドウ

トレースデータを表示します。

## [Dump]ウィンドウ

メモリの内容を表示します。

## [Command]ウィンドウ

デバッグコマンドの入力、コマンド実行結果の表示に使用します。

## ボタン

## ツールバーボタン

**[Load File]**ボタン

プログラムファイルまたはファンクションオプションファイルを読み込みます。

**[Load Parameter]**ボタン

パラメータファイルを読み込みます。

**[Key Break]**ボタン

ターゲットプログラムの実行を強制的にブレークします。

**[Break]**ボタン

[Source]ウィンドウ上のカーソル位置をブレークポイントに設定(解除)します。

**[Break All Clear]**ボタン

すべてのブレーク条件を解除します。

**[Go]**ボタン

現在のPCからターゲットプログラムを実行します。

**[Go to Cursor]**ボタン

現在のPCから[Source]ウィンドウのカーソル位置までターゲットプログラムを実行します。

**[Go after Reset]**ボタン

CPUをリセット後、リセットベクタをフェッチしてターゲットプログラムを実行します。

**[Step]**ボタン

現在のPCからターゲットプログラムを1ステップ実行します。

**[Next]**ボタン

現在のPCから1ステップ実行します。サブルーチンコールはリターンするまで実行します。

**[Step Exit]**ボタン

現在のPCからステップ実行し、現在のサブルーチンを抜けたところで停止します。

**[Reset CPU]**ボタン

CPUをリセットします。

## [Source]ウィンドウ上のボタン

**[Disassemble]**ボタン

[Source]ウィンドウを逆アセンブル表示モードにします。

**[Source]**ボタン

[Source]ウィンドウをソース表示モードにします。

**[Mix]**ボタン

[Source]ウィンドウをミックス表示モードにします。

**[Find]**ボタン

[Source]ウィンドウ内で指定文字列を検索します。

**[Find Next]**ボタン

プログラムの後方に次の候補を検索します。

**[Find Previous]**ボタン

プログラムの前方に次の候補を検索します。

**[Watch]**ボタン

[Source]ウィンドウで選択したシンボルを[Watch]ウィンドウに登録します。

## メニュー

## [File]メニュー

File

Load File...

Load Parameter File...

1 clkdemo.abs

2 Sample1.psa

Exit

## Load File...

プログラムファイルまたはファンクションオプションファイルを読み込みます。

## Load Parameter File...

パラメータファイルを読み込みます。

## Exit

デバッガを終了します。

## [Run]メニュー

Run

Go (F5)

Go to Cursor

Go after Reset

Step (F11)

Next (F10)

Step Exit

Stop (E8C)

Reset CPU

Setting...

Command File...

## Go

現在のPCからターゲットプログラムを実行します。

## Go to Cursor

現在のPCから[Source]ウィンドウのカーソル位置までターゲットプログラムを実行します。

## Go after Reset

CPUをリセット後、リセットベクタをフェッチしてターゲットプログラムを実行します。

## Step

現在のPCからターゲットプログラムを1ステップ実行します。

## Next

現在のPCから1ステップ実行します。サブルーチンコールはリターンするまで実行します。

## Step Exit

現在のPCからステップ実行し、現在のサブルーチンを抜けたところで停止します。

## Stop

実行中のプログラムを強制的に中断します。

## Reset CPU

CPUをリセットします。

## Setting...

実行オプションを設定します。

## Command File...

コマンドファイルを読み込み、実行します。

## [Break]メニュー

Break

Breakpoint Setting

Break List

Break All Clear

Setting...

## Breakpoint Setting

ブレークポイントやブレーク条件を設定/解除します。

## Break List

設定されているすべてのブレーク条件を表示します。

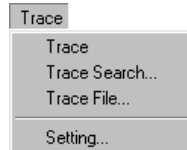
## Break All Clear

すべてのブレーク条件を解除します。

## Setting...

ブレークオプションを設定します。

## [Trace]メニュー

**Trace**

トレース情報を表示します。

**Trace Search...**

トレースデータバッファ内のトレース情報を検索します。

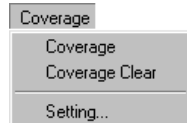
**Trace File...**

トレース情報の指定範囲をファイルに保存します。

**Setting...**

トレースモードを選択します。

## [Coverage]メニュー

**Coverage**

ICEに取得されているカバレッジ情報を表示します。

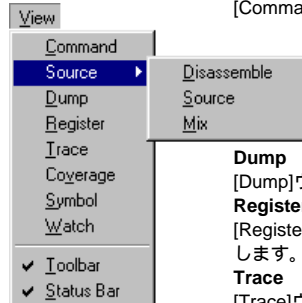
**Coverage Clear**

カバレッジ情報をクリアします。

**Setting...**

カバレッジオプションを選択します。

## [View]メニュー

**Command**

[Command]ウィンドウをアクティブにします。

**Source (Disassemble, Source, Mix)**

[Source]ウィンドウを開いてアクティブにし、プログラムを現在のPCアドレスから、サブメニューで選択した表示モードで表示します。

**Dump**

[Dump]ウィンドウを開いてアクティブにし、メモリの内容を表示します。

**Register**

[Register]ウィンドウを開いてアクティブにし、各レジスタの内容を表示します。

**Trace**

[Trace]ウィンドウを開いてアクティブにし、トレースデータバッファの内容を表示します。

**Coverage**

[Coverage]ウィンドウを開いてアクティブにし、カバレッジ情報を表示します。

**Symbol**

[Symbol]ウィンドウを開いてアクティブにし、シンボル情報を表示します。

**Watch**

[Watch]ウィンドウを開いてアクティブにし、シンボル値を表示します。

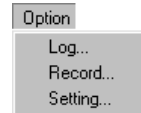
**Toolbar**

ツールバーの表示/非表示を切り換えます。

**Status Bar**

ステータスバーの表示/非表示を切り換えます。

## [Option]メニュー

**Log...**

ログ出力のON/OFFを切り換えます。

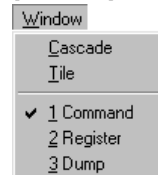
**Record...**

実行コマンドのファイルへの記録を制御します。

**Setting...**

システムオプションを設定します。

## [Window]メニュー

**Cascade**

開いているウィンドウを斜めに整列させます。

**Tile**

開いているウィンドウを縦に整列させます。

このメニューには、現在開いているウィンドウ名が表示されます。いずれかを選択すると、そのウィンドウがアクティブになります。

## [Help]メニュー

**About DB88...**

デバッグのアバウトダイアログボックスを表示します。

## デバッグコマンド

## メモリ操作

dd [ <code>&lt;addr1&gt; [&lt;addr2&gt; [{-B -W -L -F -D}]]</code> [ <code>&lt;addr1&gt; &lt;size&gt; [{-B -W -L -F -D}]]</code> ]	メモリダンプ
de [ <code>&lt;addr&gt; &lt;data1&gt; [...&lt;data16&gt;]</code> ]	データの入力
df [ <code>&lt;addr1&gt; &lt;addr2&gt; &lt;data&gt;</code> ]	領域のフィル
dm [ <code>&lt;addr1&gt; &lt;addr2&gt; &lt;addr3&gt;</code> [ <code>&lt;addr1&gt; &lt;size&gt; &lt;addr2&gt;</code> ]	領域のコピー
ds [ <code>&lt;addr1&gt; {&lt;addr2&gt;[@&lt;byte&gt;]}...</code> ...{"<str>"<data>[:{B W L}]} [S=<step>]]	データの検索

## レジスタ操作

rd	レジスタの表示
rs [ <code>&lt;reg&gt; &lt;value&gt;</code> ]	レジスタ値の変更 reg={PC SP IX IY A B HL BR CB EP XP YP  SC I1 I0 U D N V Z C}

## プログラム実行

g [ <code>&lt;addr&gt;</code> ]	カレントPCから連続実行
gr [ <code>&lt;addr&gt;</code> ]	CPUリセット後に連続実行
s [ <code>&lt;step&gt;</code> ]	カレントPCからシングルステップ実行
n [ <code>&lt;step&gt;</code> ]	関数/サブルーチン以外をステップ実行
se	関数/サブルーチン終了

## CPUリセット

rst	CPUをリセット
-----	----------

## ブレーク

bp [ <code>{- + _} &lt;addr&gt;</code> ]	ソフトウェアブレークポイントの設定
bpa [ <code>&lt;addr1&gt; &lt;addr2&gt;</code> ]	ソフトウェアブレーク領域の設定
bpr	ソフトウェアブレークポイントの解除
bc [ <code>&lt;addr&gt;</code> ]	
bpc [ <code>&lt;addr&gt;</code> ]	
bas {0 1 2 3}	シーケンシャルブレークモードの設定
ba [ <code>&lt;ch&gt; &lt;addr&gt; [&lt;count&gt;]</code> <code>&lt;ch&gt; {- + _}</code> ]	ハードウェアブレークポイントの設定
bar	ハードウェアブレークポイントの解除
bd [ <code>&lt;ch&gt; [A=&lt;addr&gt;][D=&lt;data&gt;][{R W}]</code> <code>&lt;ch&gt; {- + _}</code> ]	ハードウェアデータブレーク条件の設定
bdr	ハードウェアデータブレーク条件の解除
bl	全ブレーク条件の表示
bac	全ブレーク条件の解除

## プログラム表示

u [ <code>&lt;addr&gt;</code> ]	逆アセンブル表示
sc [ <code>&lt;addr&gt;</code> ]	ソース表示
m [ <code>&lt;addr&gt;</code> ]	ミックス表示

## シンボル情報

sy [ <code>/a</code> ]	シンボル一覧の表示
w [ <code>&lt;symbol&gt; [{:H D Q B}] [/A]</code> ]	シンボル情報の表示

## ファイルロード

lf [ <code>&lt;file&gt;</code> ]	プログラム/オプションファイルのロード
par [ <code>&lt;file&gt;</code> ]	パラメータファイルのロード

## トレース

td [ <code>&lt;cycle&gt;</code> ]	トレース情報の表示
ts [{ <code>&lt;pc dr dw&gt; &lt;addr&gt;</code> }]	トレース情報の検索
tf [ <code>&lt;file&gt; [&lt;cycle1&gt; [&lt;cycle2&gt;]]</code> ]	トレース情報の保存

## カバレッジ

cv [ <code>&lt;addr1&gt; [&lt;addr2&gt;]</code> ]	カバレッジ情報の表示
cvc	カバレッジ情報のクリア

## コマンドファイル, ログ

com [ <code>&lt;file&gt; [&lt;interval&gt;]</code> ]	コマンドファイル実行
cmw [ <code>&lt;file&gt;</code> ]	ウェイト付きコマンドファイル実行
rec [ <code>&lt;file&gt;</code> ]	実行コマンドの記録
log [ <code>&lt;file&gt;</code> ]	ログ出力

## マップ情報

ma	マップ情報の表示
----	----------

## FPGA操作

xfer	FPGAの消去
xfwr [ <code>&lt;file&gt; ;{H S} [:N]</code> ]	FPGAデータ書き込み
xfcp [ <code>&lt;file&gt; ;{H S}</code> ]	FPGAデータコンペア
xdp [ <code>&lt;addr1&gt; [&lt;addr2&gt;]</code> ]	FPGAデータダンプ

## 終了

q	デバッガ終了
---	--------

## ヘルプ

?	コマンドusageの表示
---	--------------

## デバッグメッセージ

## デバッグエラー

Error : Address out of range : use 0x000000 - 0xfffff	指定されたアドレスは有効範囲外です
Error : Address out of range, use 0 - 0x7FFFFFFF	プログラムメモリ領域外のアドレスが指定されました
Error : Address out of range, use 0 - 0xFFFFFFFF	データメモリ領域外のアドレスが指定されました
Error : Cannot open device (ICE88UR)	ICEとの接続に失敗しました
Error : Cannot open file	ファイルがオープンできません
Error : Checksum error	チェックサムでエラーになりました
Error : Coverage mode is off or the coverage mode is not supported	カバレッジのモードがOFFまたは、当該ICEはカバレッジをサポートしていません
Error : Data out of range, use 0 - 0xFF	指定の数値はデータの有効範囲外です
Error : DLL Initialization error	DLLの初期化に失敗しました
Error : End address < start address	開始アドレスより小さい終了アドレスが指定されました
Error : End index < start index	開始サイクルより小さい終了サイクルが指定されました
Error : Error file type (extension should be CMD)	指定のファイル拡張子は、コマンドファイルとして無効です
Error : Error file type (extension should be PAR)	指定のファイル拡張子は、パラメータファイルとして無効です
Error : Failed ICE88UR initialization	ICEの初期化に失敗しました
Error : Failed to initialize DLL : %s	DLLの初期化に失敗しました
Error : Failed to Load DLL	DB88起動に必要なDLLのロードに失敗しました
Error : Failed to open : %s	ファイルを開けませんでした
Error : Failed to read BA	BAレジスタ読み込みエラー
Error : Failed to read BR	BRレジスタ読み込みエラー
Error : Failed to read CB	CBレジスタ読み込みエラー
Error : Failed to read CC	CCレジスタ読み込みエラー
Error : Failed to read EP	EPレジスタ読み込みエラー
Error : Failed to read file : %s	ファイルの読み込みに失敗しました
Error : Failed to read HL	HLレジスタ読み込みエラー
Error : Failed to read NB	NBレジスタ読み込みエラー
Error : Failed to read PC	PCレジスタ読み込みエラー
Error : Failed to read SC	SCレジスタ読み込みエラー
Error : Failed to read SP	SPレジスタ読み込みエラー
Error : Failed to read X	Xレジスタ読み込みエラー
Error : Failed to read Y	Yレジスタ読み込みエラー
Error : Failed to read DLL : %s	DLLのロードに失敗しました
Error : Failed to write BA	BAレジスタ書き込みエラー
Error : Failed to write BR	BRレジスタ書き込みエラー
Error : Failed to write CB	CBレジスタ書き込みエラー
Error : Failed to write CC	CCレジスタ書き込みエラー
Error : Failed to write EP	EPレジスタ書き込みエラー

## デバッグエラー

Error : Failed to write HL	HLレジスタ書き込みエラー
Error : Failed to write NB	NBレジスタ書き込みエラー
Error : Failed to write PC	PCレジスタ書き込みエラー
Error : Failed to write SC	SCレジスタ書き込みエラー
Error : Failed to write SP	SPレジスタ書き込みエラー
Error : Failed to write X	Xレジスタ書き込みエラー
Error : Failed to write Y	Yレジスタ書き込みエラー
Error : ICE88UR Diagnostic error	ICE自己診断処理でエラーが検出されました
Error : Ice88ur Initialization failed	ICEの初期化に失敗しました
Error : Ice88ur is already running	ICE88UR.EXEが起動されています
Error : ICE88UR is turned off	ICEの電源がOFFになっています
Error : Illegal initialization packet data	初期化パケットエラー
Error : Incorrect number of parameters	コマンドのパラメータ数が不正です
Error : Incorrect r/w option, use r/w/*	無効なR/Wオプションが指定されました
Error : Incorrect register name, use PC/SP/IX/IY/A/B/HL/BR/CB/EP/XP/YP/SC	無効なレジスタ名が指定されました
Error : Index out of range, use 0 - 8191	指定のトレースサイクル番号は、有効範囲外です
Error : Initialization failed!	DB88の初期化に失敗しました
Please quit and restart!	DB88を再起動してください
Error : Input address does not exist	未設定のブレークポイントアドレスが指定されました
Error : Invalid command	無効なコマンドが入力されました
Error : Invalid data pattern	入力データパターンが不正です
Error : Invalid display unit, use -B/-W/-L/-F/-D	表示単位の指定が無効です
Error : Invalid DLL ModuleID	DLL識別エラー
Error : Invalid file name	指定のファイル拡張子は無効です
Error : Invalid fsa file	FSAファイルが不正です
Error : Invalid hexadecimal string	不正な16進数文字列です
Error : Invalid value	入力した値が不正です
Error : Maximum nesting level(5) is exceeded, cannot open file	コマンドファイルのネストレベルが制限を越えました
Error : Memory ranges in %s are invalid or the file is not exist	CPU INIファイルのメモリ範囲が不正です
Error : No symbol information	シンボル情報がありません
Error : Number of steps out of range, use 0 - 65535	指定ステップ数は制限を越えています
Error : The Memory Area cannot include the boundary between 0x00FFFF and 0x010000	指定されたエリアは、0x00FFFF ~ 0x010000の境界を含んでいます
Error : The Memory Area must be above 0x010000, and longer than 256 bytes	0x010000以上で領域を指定する場合、256バイトより小さいサイズは指定できません

## デバッグメッセージ

## デバッグエラー

Error : This command is not supported in current mode	トレース/カバレッジコマンドは、トレースOFF/カバレッジOFF時は無効です
Error : Unable to get the coverage area number	カバレッジエリア番号取得に失敗しました
Error : Unable to get the coverage mode	カバレッジ情報の取得に失敗しました
Error : Unable to set SelfFlash check function	自己書き換えチェック機能が設定できませんでした
Error : Unable to set the coverage area number	カバレッジエリア番号設定に失敗しました
Error : Unable to set the coverage mode	カバレッジモード設定に失敗しました
Error : Wrong Command line parameter	起動パラメータに誤りがあります
Please load the selfflash library program	自己書き換えライブラリプログラムをロードしてください
Warning : 64 break addresses are already set	64ヶ所を越えるブレークポイントが指定されました
Warning : Break address already exists	指定のアドレスは既にブレークポイントに設定されています
Warning : Identical break address input	コマンドラインに同じアドレスが2回以上指定されています
Warning : Memory may be modified by SelfFlash	自己書き換えプログラムにより、メモリが書き換えられている可能性があります
Warning : SelfFlash program area is out of the current software pc break area.	自己書き換えプログラムエリアが、現在設定されているソフトウェアブレイクエリアと一致していません
Please clear the break point(Address)	(Address)に設定されているブレイクポイントを解除してください

## ICEエラー

Error : Cannot be run in Free-Run mode	ICEはフリーラン動作中です
Error : Cannot find specified data	指定したデータは見つかりません
Error : ICE88UR is still keep a conservative mode	ICEは保守モード動作中です
Error : ICE88UR power off execution abort	ICE本体の供給が断たれました
Error : Insufficient memory for loading program	メモリの確保に失敗しました
Error : Vdd down or no clock	ターゲットシステムの電源電圧が低下しているか、電源が入っていない、もしくはクロックが供給されていません。
Error : Verify error	ベリファイエラーが発生しました
ICE88UR system error : ?? illegal packet	不正なパケットを検出しました
ICE88UR system error : Command timeout	コマンドタイムアウトを検出しました
ICE88UR system error : Firmware packet error	EB:Firmwareパケットでエラーを検出しました
ICE88UR system error : Master reset	MR:マスタリセットを検出しました
ICE88UR system error : Not connected	ICEが未接続または電源が入っていません
ICE88UR system error : Not ready	ICEの準備ができていません
Internal error : ICE88UR does not support this command version	本バージョンでは対応していません
Internal error : Illegal error code fetched. System crash possible	存在しないエラーコードが見つかりました
Processing terminated by hitting ESC-key	ESCキーにより処理を中断しました



## 概要

構造化プリプロセッサsap88は、クロスアセンブラasm88にマクロ機能を付加するためのプリプロセッサです。

sap88は、指定されたファイル名のS1C88アセンブリソースファイル中に含まれるマクロをasm88でアセンブル可能な形式へ展開し出力します。またこのとき、モジュール化されたS1C88アセンブリソースファイルのインクルードや条件アセンブル等の処理も行います。

## 起動コマンド

sap88 [flags] <file name>

## フラグ

-d<macro>	入力ファイルの読み込みに先だって、文字列マクロを定義します。 <macro>: <文字列マクロ>=<置換文字列> または <文字列マクロ名>
-l<label>	構造化制御文を展開するときに生成されるラベル名の前置文字列を指定します。デフォルトでは"l"が設定されています。
-o<file name>	出力ファイル名を<file name>にします。デフォルトでは標準出力が設定されています。
-q	構造化プリプロセッサの処理に関するメッセージを一切出力しません。

## エラーメッセージ

unexpected EOF in ~	~の途中でファイルが終了した
can't include ~	~がインクルードできない
illegal ~	~が不正である
illegal define	define文が不正である
illegal expression at ~	式中~が不正である
illegal undef	undef文が不正である

## 擬似命令

INCLUDE	<file>	指定されたファイルをINCLUDE文の直後へ読み込む
<macro>	MACRO [<param>,...] <statements> [EXITM] <statements> [<macro>] ENDM	マクロ定義
DEFINE	<macro> [<character string>]	文字列マクロ定義
LOCAL	[<label>,...]	ローカルラベルの定義
PURGE	[<macro>]	マクロの抹消
UNDEF	<macro>	文字列マクロの抹消
IRP	<param>,<arg>[,<arg>...] <statements>	文字列による繰り返し
ENDR		
IRPC	<param>,<arg> <statements>	文字による繰り返し
ENDR		
REPT	<expression> <statements>	回数指定による繰り返し
ENDR		
IFC	<condition> <statements>[ <statements>]	条件式による条件アセンブル
ELSEC		
ENDIF		
IFDEF	<name> <statements>[ <statements>]	名前が定義されているか否かによる条件アセンブル
ELSEC		
ENDIF		
IFNDEF	<name> <statements>[ <statements>]	名前が定義されていないか否かによる条件アセンブル
ELSEC		
ENDIF		

概要

クロスアセンブラasm88は、構造化プリプロセッサsap88にてマクロ等が展開されたアセンブリソースファイルをアセンブルし、機械語に変換します。asm88は、モジュール別開発のためのリロケータブルアセンブルに対応しています。リロケータブルアセンブルでは、リンカlink88によって他のモジュールと連結するためのリロケータブルオブジェクトファイルが生成されます。

起動コマンド

```
asm88 [flags] <file names>
```

フラグ

-all	ローカルシンボルを含む全てのシンボルをシンボリックテーブルに出力します。
-c	入力ソースでの大文字、小文字を区別します。
-i	アセンブリリストファイルの作成を禁止します。
-o<file name>	<file name>という名前で出力ファイルを作成します。
-q	アセンブル処理に関するメッセージを一切出力しません。
-RAM<size>	RAMエリアの容量をバイト単位の<size>に設定します。
-ROM<size>	ROMエリアの容量をバイト単位の<size>に設定します。
-sig<number>	<number>の値によりシンボルの有効文字数を設定することができます。
-suf<ext>	入力ファイルの拡張子を<ext>(セパレータは含まない)に変更します。
-x	クロスリファレンスリストファイルの作成を禁止します。

擬似命令

CODE	コードセクションの定義
DATA	データセクションの定義
DB <exp>[,<exp>...]	1バイト単位の領域を確保、定数の設定
DW <exp>[,<exp>...]	ワード(2バイト)単位のデータ領域を確保、定数の設定
DL <exp>[,<exp>...]	4バイト整数の領域を確保、定数の設定
ASCII <exp>[,<exp>...]	ASCIIテキストのメモリ格納
PARITY	パリティビットのセット/リセット
<name> EQU <exp>	名前の値設定
<name> SET <exp>	名前の値設定
ORG <exp>	ロケーションカウンタの値を変更
EXTERNAL <symbol>[,<symbol>]	シンボルの外部参照宣言
PUBLIC <symbol>[,<symbol>]	シンボルのグローバル宣言
LINENO <exp>	アセンブリリストファイルの行番号を変更
SUBTITLE <title>	アセンブリリストファイルへのサブタイトルの設定
SKIP	アセンブリリストファイルへの4バイトを越える初期化コード出力サプレス
NOSKIP	アセンブリリストファイルへ初期化コードを全て出力
LIST	アセンブリリストファイルの出力
NOLIST	アセンブリリストファイルの出力を禁止
EJECT	アセンブリリストファイルの改ページ
END [<label>]	アセンブルの停止

## エラーメッセージ

## Fatalエラー

can't create <file>	<file>が作成できない
can't open <file>	<file>がオープンできない
can't read tmp file	中間ファイルが読めない
can't write tmp file	中間ファイルが書けない
namelist full	名前のリストテーブルが満杯になった
no i/p file	入力ファイル指定がない
insufficient memory	メモリが不十分である
can't seek on vmem file	バーチャルメモリファイルのシークに失敗した
can't seek to end of vmem file	バーチャルメモリファイル終端に到達できない
no swappable page	スワップスペースがとれない
read error on vmem file	バーチャルメモリファイルの読み込みに失敗した
write error on vmem file	バーチャルメモリファイルの書き込みに失敗した

## Severeエラー

<numeric label> already defined	ニューメリックラベルが衝突している
<identifier> wrong type	不正な識別子が現われた
<token> expected	トークンが必要である
' missing	引用符のアンバランス
attempted division by zero	ゼロによる除算が行われようとしていた
attempt to redefine <identifier>	識別子が再定義されようとしている
constant expected	定数を期待していた
end expected	end命令がない
encountered too early end of line	行が途中で終了した
field overflow	確保する領域がオーバーフローしている
invalid branch address	ショートジャンプ命令のオペランドに外部定義シンボルが置かれた
invalid byte relocation	バイトリロケーションが無効である
invalid character	不正文字を使用している
invalid flag	フラグが無効である
invalid operand	オペランドが無効である
invalid relocation item	リロケーション項目が無効である
invalid register	レジスタが無効である
invalid register pair	レジスタの組み合わせが無効である
invalid symbol define	シンボル定義が無効である
invalid word relocation	ワードリロケーションが無効である
new origin incompatible with current psect	相対セクション内(リロケータブルモード)で絶対originが存在した
non terminated string	文字列の終端が見つからない
<identifier> not defined	未定義の識別子が現われた
missing numeric expression	数値式が欠如している
cars or jrs out of range	carsまたはjrsの飛び先が遠すぎる
carl or jrl out of range	carlまたはjrlの飛び先が遠すぎる

## Severeエラー

operand expected	オペランドがない
psect name required	セクション名の指定が必要である
phase error <identifier>	バス1とバス2でラベルのアドレスが異なった
CODE or DATA missing	領域確保擬似命令がない
ROM capacity overflow	ROM容量をオーバーした
RAM capacity overflow	RAM容量をオーバーした
relocation error in expression	式中にリロケーションエラーが現われた
<identifier> reserved word	<identifier>は予約語である
syntax error <token> expected	トークン不足による文法エラー
syntax error <token> unexpected	過剰トークンによる文法エラー
syntax error - invalid identifier	不正識別子による文法エラー
<identifier>	
syntax error <token> invalid in expression	不正トークンによる文法エラー
system error <> <token>	不正トークンによるシステムエラー
unsupported instruction	未サポート命令が現われた
unsupported operand	未サポートオペランドが現われた

## Warningエラー

directive is ignored in relocatable mode	リロケータブルモードのため擬似命令がスキップされた
possibly missing relocatability	リロケータブル性が失われる恐れがある
constant overflow	名前を7桁以上定義した
expected operator	演算子がない (BOC, LOC, POD, LOD)

## 概要

link88は、S1C88のマルチセクションリロケータブルオブジェクトファイルを結合してアプソリュートオブジェクトファイルを生成します。このファイルは、ICEでデバッグを行うためにプログラムデータHEXファイルを得る、バイナリHEXコンバータhex88への入力ファイルとなります。また、リロケータブルアセンブルされたファイルのリンク後のアプソリュートシンボル情報生成rel88を実行するための入力ファイルにもなります。

## 起動コマンド

```
link88 [global flags] [local flags] [<drive name>:]
```

## フラグ

### グローバルフラグ

<b>-c</b>	リロケータブルオブジェクトファイルのシンボルの大文字と小文字を区別します。
<b>-cd</b>	DATAセクション向けのコード部を出力しません。
<b>+dead</b>	定義されているが絶対に参照されないシンボルのリストをCRT上に出します。
<b>-max&lt;size&gt;</b>	セクションの最大サイズを<size>バイトにします。
<b>-o&lt;file name&gt;</b>	出力モジュールをファイル<file name>に書き込みます。
<b>-q</b>	リンク処理に関するメッセージを一切出力しません。

### ローカルフラグ

<b>+code</b>	新しいICODEセクションを開始して、そのセクションのためのローカルフラグを処理します。
<b>+data</b>	新しいIDATAセクションを開始して、そのセクションのためのローカルフラグを処理します。
<b>-m&lt;size&gt;</b>	個別セクションの最大サイズを<size>バイトにします。
<b>-p&lt;addr&gt;</b>	セクションの物理アドレスの先頭を<addr>にセットします。

## エラーメッセージ

bad file format: 'FILE NAME'	入力ファイル'FILE NAME'のフォーマットが不正である
bad relocation item	long integerタイプのリロケーション情報があった
bad symbol number: 'NUMBER'	'NUMBER'が不正なシンボルコードとして検出された
can't create 'FILE NAME'	ファイル'FILE NAME'が作成できない
can't create tmp file	中間ファイルが作成できない
can't open: 'FILE NAME'	入力ファイル'FILE NAME'がオープンできない
can't read binary header: 'FILE NAME'	ファイル'FILE NAME'のヘッダ部が読み込めない
can't read file header: 'FILE NAME'	ファイル'FILE NAME'の最初の2バイトが読み込めない
can't read symbol table: 'FILE NAME'	ファイル'FILE NAME'からシンボルテーブルが読み込めない
can't read tmp file	中間ファイルが読めない
can't write output file	出力ファイルに書き込みができない
can't write tmp file	中間ファイルに書き込みができない
field overflow	carsまたはjrsの飛び先が遠すぎる
inquiry phase error: 'SYMBOL NAME'	'SYMBOL NAME'のシンボル値がバス1とバス2で異なった
link: early EOF in pass2	バス2の最中に予想外のEOFが検出された
multiply defined 'SYMBOL NAME'	'SYMBOL NAME'が多重定義されている
no object files	入力オブジェクトファイルが存在しない
no relocation bits: 'FILE NAME'	ファイル'FILE NAME'に対応するリロケーション情報がサブレスされている
'SECTION NAME' overflow	セクション名'SECTION NAME'に属するセクションとサイズが上限値を上回った
phase error: 'SYMBOL NAME'	'SYMBOL NAME'のシンボル値がバス1とバス2で異なった
previous reference blocked: 'SYMBOL NAME' range error	リロケーションのビット幅に関する情報がミスマッチを起こしている
read error in pass2	バス2の最中に読み込みエラーが発生した
undefined 'SYMBOL NAME'	シンボル'SYMBOL NAME'がどこにも定義されていない

概要

rel88は、S1C88のマルチセクション形式のリロケータブルオブジェクトの検査をします。この対象となるファイルは、クロスアセンブラasm88からのリロケータブルオブジェクトファイル、link88から出力されたアプソリュートオブジェクトファイルです。rel88は、オブジェクトファイルの大きさと構成を検査したり、link88から出力されたアプソリュートオブジェクトファイルのシンボル情報を出力するために使うことができます。

起動コマンド

```
rel88 [flags] <file names>
```

フラグ

-a	出力をシンボル名のアルファベット順にソートします。
+dec	シンボル値とセクションサイズを10進数で出力します。
-d	各ファイル内の全ての定義済みシンボルを1行に1個ずつ出力します。
-g	グローバルシンボルだけを出力します。
+in	<file names>を標準入力から取り、コマンド行に追加します。
+sec	マルチセクションオブジェクトファイルの各セクションの物理アドレス、サイズを出力します。
-v	セクション内をシンボルの値でソートします。前述の-dフラグが暗黙的に指定されます。

エラーメッセージ

can't read binary header	マジック番号とコンフィグレーションバイトを除くオブジェクトヘッダ部の読み出しに失敗した
can't read header	オブジェクトヘッダ部の最初の2バイト(マジック番号とコンフィグレーションバイト)の読み出しに失敗した
can't read symbol table	オブジェクト内のシンボリックテーブルの読み出しに失敗した

## 概要

sym88は、シンボル情報生成ユーティリティrel88からファイルリダイレクトによって生成された、アブソリュートオブジェクトファイルに対応したシンボル情報ファイル(file\_name.ref)をICE上で参照可能なシンボリックテーブルファイル(file\_name.sy)にフォーマット変換するユーティリティです。これによって、リロケータブルアセンブルされたプログラムおよび上記のツールによって作成されたシンボリックテーブルファイルをICE上にロードし、シンボリックデバッグを実行することが可能となります。

## 起動コマンド

```
sym88 <file name>
```

## エラーメッセージ

No Input File

入力ファイル".ref"が指定されていない

## 概要

hex88はlink88により生成されたアブソリュートオブジェクトファイルを16進のデータ変換形式(プログラムデータHEXファイル)に変換します。形式としては、モトローラSレコード形式を採用しています。

## 起動コマンド

```
hex88 [-o<file name>] <file name>
```

## フラグ

**-o<file name>** 出力モジュールをファイル<file name>に書き込みます。

## エラーメッセージ

bad file format	入力ファイルのフォーマットが不正である
can't read <input file>	入力ファイル<input file>の読み出しに失敗した
can't write <output file>	出力ファイル<output file>への書き込みに失敗した

## S5U1C88000C Manual II

(S1C88 Family統合ツールパッケージ)

ワークベンチ/Development Tools/旧アセンブラパッケージ

## セイコーエプソン株式会社

### 半導体事業部 IC営業部

〈IC国内営業グループ〉

東京 〒191-8501 東京都日野市日野421-8  
TEL (042) 587-5313 (直通) FAX (042) 587-5116

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

インターネットによる電子デバイスのご紹介

<http://www.epson.jp/device/semicon/>