

お客様各位

---

## カタログ等資料中の旧社名の扱いについて

---

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

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

2010年4月1日

ルネサスエレクトロニクス株式会社

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

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

## ご注意書き

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

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

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



お客様各位

---

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

---

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

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

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

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

77XXシリーズ用  
Cコンパイラ/アセンブラ  
NC77WA / RASM77 ガイドブック

Microsoft、MS-DOS、WindowsおよびWindows NTは、米国Microsoft Corporationの米国およびその他の国における登録商標です。  
HP-UXは、米国Hewlett-Packard Companyのオペレーティングシステムの名称です。  
SolarisおよびSunは、米国およびその他の国における米国Sun Microsystems, Inc.の商標または登録商標です。  
UNIXは、X/Open Company Limitedが独占的にライセンスしている米国ならびに他の国における登録商標です。  
IBMおよびATは、米国International Business Machines Corporationの登録商標です。  
HP 9000は、米国Hewlett-Packard Companyの商品名称です。  
SPARCおよびSPARCstationは、米国SPARC International, Inc.の登録商標です。  
Intel、Pentiumは、米国Intel Corporationの登録商標です。  
AdobeおよびAcrobatは、Adobe Systems Incorporated(アドビシステムズ社)の登録商標です。  
NetscapeおよびNetscape Navigatorは、米国およびその他の諸国のNetscape Communications Corporation社の登録商標です。  
その他すべてのブランド名および製品名は個々の所有者の登録商標もしくは商標です。

#### 《安全設計に関するお願い》

三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

#### 《本資料ご利用に際しての留意事項》

本資料は、お客様が用途に応じた適切な三菱半導体製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について三菱電機株式会社・三菱電機セミコンダクタシステム株式会社が所有する知的財産権その他の権利の実施、使用を許諾するものではありません。

本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は責任を負いません。

本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は、予告なしに、本資料に記載した製品または仕様を変更することがあります。三菱半導体製品のご購入に当たりますと、事前に三菱電機株式会社・三菱電機セミコンダクタシステム株式会社または特約店へ最新の情報をご確認頂きますとともに、三菱電機半導体情報ホームページ( <http://www.semicon.melco.co.jp/> )および三菱開発ツールホームページ( <http://www.tool-spt.mesc.co.jp/> )などを通じて公開される情報に常にご注意ください。

本資料に記載した情報は、正確を期すため、慎重に制作したものです。万一本資料の記述誤りに起因する損害がお客様に生じた場合には、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社はその責任を負いません。

本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。三菱電機株式会社・三菱電機セミコンダクタシステム株式会社は、適用可否に対する責任を負いません。

本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、三菱電機株式会社・三菱電機セミコンダクタシステム株式会社または特約店へご照会ください。

本資料の転載、複製については、文書による三菱電機株式会社・三菱電機セミコンダクタシステム株式会社の事前の承諾が必要です。

本資料に関し詳細についてのお問い合わせ、その他お気づきの点がございましたら三菱電機株式会社・三菱電機セミコンダクタシステム株式会社または特約店までご照会ください。

#### 製品の内容及び本書についてのお問い合わせ先

電子メールの場合： インストーラが生成する以下のテキストファイルに必要事項を記入の上、開発ツールサポート窓口 support@tool.mesc.co.jpまで送信ください。

Windows 98/95/Windows NT 4.0版：¥SUPPORT¥製品名¥SUPPORT.TXT

EWS版：/support/製品名/toolinfo.txt

FAXの場合： 各ユーザーズマニュアル及びガイドブックの最後に添付されている「技術サポート連絡書」に必要事項を記入の上、開発ツールサポート窓口まで送信ください。FAX送信先は「技術サポート連絡書」に記載してあります。

## NC77WA / RASM77 ガイドブック

ガイドブックの構成 .....	5
ガイドブックの構成 .....	5
NC77WA へようこそ .....	7
製品内容 .....	7
製品の最新情報 .....	8
NC77WA の特徴 .....	8
ANSI 規格に準拠した C コンパイラ : NC77 .....	10
リロケータブルアセンブラ : RASM77 .....	11
その他の機能 .....	12
製品に関する注意事項 .....	13
NC77WA をインストールしよう .....	15
インストールを始める前に .....	15
NC77WA のインストーラ .....	16
RASM77 のインストーラ .....	16
PC 版のインストール .....	17
EWS 版のインストール .....	17
インストール後に生成されるディレクトリとソフトウェア .....	20
インストール後の環境設定を行う .....	22
ユーザー登録をしましょう .....	24
Acrobat Reader のインストール .....	24
freeware に格納されているプログラムについて .....	25

## 目次

NC77WA を使ってみよう .....	27
NC77WA をご使用になる前に .....	27
NC77 を使ってみよう .....	28
RASM77 を使ってみよう .....	30
電子マニュアルを使ってみよう .....	32
オンラインヘルプを使ってみよう .....	33
NC77WA のスタートアッププログラム .....	35
スタートアップサンプルプログラム ncrt0.a77 の内容 .....	35
スタートアップサンプルプログラム section.inc の内容 .....	42
NC77WA ヒント集 .....	47
stk77 を使ってスタック使用量を計算しよう .....	47
プログラムを ROM に書き込む準備をする .....	48
Q&A 集 .....	49
付録 A NC77WA コマンドオプション一覧 .....	55
付録 B NC77 拡張機能一覧 .....	63
付録 C NC77 標準関数一覧 .....	65
付録 D RASM77 アセンブラ疑似命令一覧 .....	73

## ガイドブックの構成

### ガイドブックの構成

#### NC77WA へようこそ

製品の機能概要を紹介しています。  
「製品に関する注意事項」は必ずお読みください。

#### NC77WA をインストールしよう

製品をインストールする際に必要となる情報が記載されています。  
製品をインストールされる前にお読みください。

#### NC77WA を使ってみよう

NC77WA の基本的な使用方法を説明しています。  
製品をご使用になる前にお読みください。

#### NC77WA のスタートアッププログラム

マイクロコンピュータ組み込みプログラムをC言語で開発する場合の  
様々な設定を行うファイルについて説明しています。  
NC77WA を初めてご使用になる際は必ずお読みください。

#### NC77WA ヒント集

NC77WA のプログラミングテクニックなどを紹介しています。  
「Q&A集」には、よくあるご質問について掲載しておりますので、ご一  
読ください。

#### 付録

コマンドオプションや拡張機能などの一覧を掲載しています。

#### 注意！

ガイドブックには、本製品をご使用いただくための基本的な情報を掲載して  
います。より詳細な情報は製品 CD-ROM に含まれている電子マニュアルを  
参照してください。

RASM77 をご購入の場合、NC77 と TM はご使用になれません。





## NC77WA へようこそ

### 製品内容

NC77WA をご購入いただきありがとうございます。

NC77WA には以下のものが含まれております。これらのものが含まれていない場合には、ご購入いただいた三菱電機営業および特約店にご連絡ください。

### 梱包内容

NC77WA / RASM77 ガイドブック(本書)

NC77WA またはRASM77 リリースノート

使用権許諾契約書

ライセンス ID 証書 (新規購入のみ)

CD-ROM

TM リリースノート

### 注意！

---

ライセンス ID 証書は、今後のバージョンアップ時に必要となりますので大切に保管してください。

---

## CD-ROM の内容

CD-ROM には、以下に示す以外のコンパイラ製品も含まれておりますが、NC77WA をご購入の場合に、ご使用できる製品は以下のとおりです。

注意！

---

RASM77 製品をご購入の場合、NC77 と TM はご使用になれません。

---

NC77WA

NC77 電子マニュアル (PDF 形式ファイル)

RASM77 電子マニュアル (PDF 形式ファイル)

TM

TM 電子マニュアル (PDF 形式ファイル)

7700 ファミリ関連開発支援ツールデータシート (HTML 形式ファイル)

Adobe Acrobat Reader

MM (マスクファイル生成ツール)

GNU ユーティリティ

注意！

---

NC77、RASM77 および TM の各マニュアルは、CD-ROM 内の電子マニュアルのみです。

---

## 製品の最新情報

製品をご使用になる際には、製品の最新情報を以下のホームページなどでご確認ください。

<http://www.tool-spt.mesc.co.jp>

こちらでは、他のツール製品に関する最新情報についても掲載されていますのでご活用ください。

## NC77WA の特徴

### 製品の構成

NC77WA は次のようなコンポーネントから構成されています。

- ANSI 規格に準拠した C コンパイラ
- リロケータブルアセンブラ

## 製品の概要

NC77WA は、デバッガで読み込み可能な機械語ファイル（IEEE695 アブソリュートファイル）や、ROM に書き込み可能な機械語ファイル（モトローラ S フォーマット、インテル HEX フォーマット）を生成します。

## 関連開発支援ツール

言語系開発支援ツール（ここでは NC77WA）に関連した開発支援ツールに次のようなものがあります。詳細は「7700 関連開発支援ツールデータシート」を参照してください。

### エミュレータデバッガ

エミュレータを PC または EWS 上で制御するソフトウェアです。

### シミュレータデバッガ

エミュレータを使用せずアプリケーションプログラムのデバッグを可能にするソフトウェアです。

### エミュレータ/エミュレーションポッド

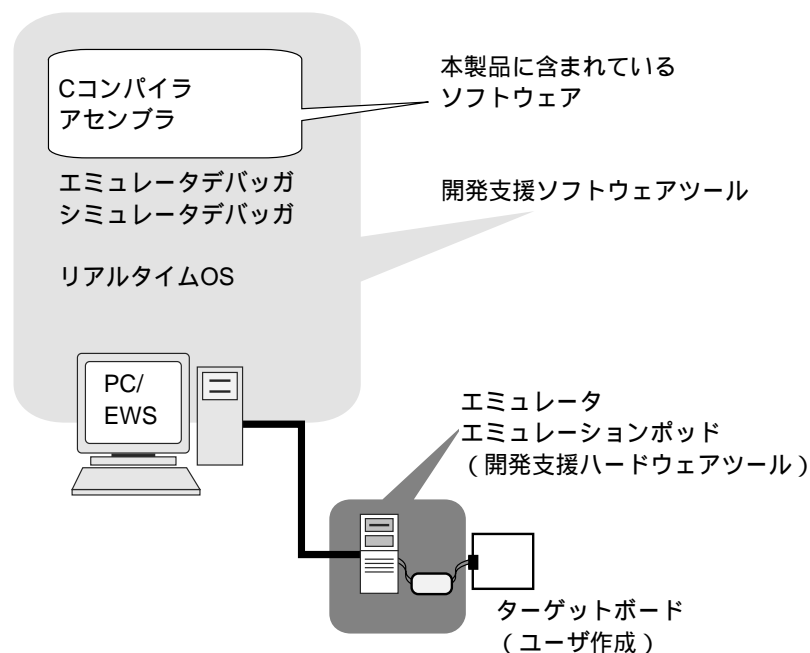
エミュレータおよびエミュレーションポッドは、マイコンの機能を実現したハードウェアをいいます。

### リアルタイムOS

アプリケーションプログラムのリアルタイム処理機能を向上できる組み込み型のリアルタイムOS です。

## NC77WA を使ったプログラム開発環境例

NC77WA を使ってアプリケーションプログラムを開発するときの開発環境例を次に示します。



## ANSI 規格に準拠した C コンパイラ : NC77

NC77 の特徴とコンパイル処理の流れについて説明します。

### NC77 の機能概要

- ANSI 規格に準拠した C 言語で記述されたプログラムをアセンブリ言語に変換します。
- マイコンへの組み込み型プログラムを開発するための拡張機能を持っています。
- リアルタイムOS(MR7700) のための拡張機能を持っています。
- C ソースレベルでデバッグするためのデバッグ情報を出力します。

### コンパイル処理の流れ

- 1 C 言語のプリプロセス処理、コンパイル処理を順次行い、C 言語で記述されたプログラムをアセンブリ言語ファイルに変換します。

C 言語ファイル ( 拡張子 .c )

↓

cpp77

↓

ccom77

↓

アセンブリ言語ファイル ( 拡張子 .a77 )

- 2 アセンブリ言語ファイルをアセンブル処理し、リロケータブルオブジェクトファイルに変換します。

アセンブリ言語ファイル ( 拡張子 .a77 )

↓

rasm77

↓

リロケータブルオブジェクトファイル ( 拡張子 .r77 )

- 3 リロケータブルオブジェクトファイルをリンク処理し、16 進機械語ファイルに変換します。

リロケータブルオブジェクトファイル (拡張子 .r77)

↓

link77

↓

16 進機械語ファイル (拡張子 .HEX)

## リロケータブルアセンブラ : RASM77

RASM77 の特徴とアセンブル処理の流れについて示します。

### RASM77 の機能概要

- ・ アセンブリ言語で記述されたプログラムを機械語に変換します。
- ・ マクロ機能をサポートしています。
- ・ 構造化記述文をサポートしています。

### アセンブルの処理の流れ

- 1 アセンブリ言語ファイルをリロケータブルオブジェクトファイルに変換する

アセンブリ言語ファイル (拡張子 .a77)

↓

rasm77

↓

リロケータブルオブジェクトファイル (拡張子 .r77)

注意！

---

構造化記述命令を使用しているときは、アセンブラオプション -P を指定してください。

---

- 2 リロケータブルオブジェクトファイルを 16 進機械語ファイルに変換する

リロケータブルオブジェクトファイル (拡張子 .r77)

↓

link77

↓

16 進機械語ファイル (拡張子 .HEX)

## その他の機能

NC77WA には、プログラム開発をサポートするユーティリティプログラムが添付されています。

### stk77

C 言語でプログラム開発を行う場合に、関数を使用するスタックサイズと関数の呼び出し関係を元にスタック使用量を計算します。

### s2ie

シンボルファイルの形式を IEEE-695 に変換します。

### crf77

アセンブリ言語ファイルからラベルやシンボルの定義および参照情報をリストにしたクロスリファレンスファイルを生成します。

### lib77

新しくライブラリファイルを生成できます。C 言語やアセンブリ言語で作成したプログラムのリロケータブルファイルをライブラリファイルに登録できます。

## 製品に関する注意事項

製品をご使用になる際には、以下の内容に従ってくださるようお願いいたします。

### PC 版に関する注意事項

#### 動作環境についての注意事項

- PC 版は、Windows 95、Windows 98、Windows NT 4.0 以降の環境で動作します。Windows 3.1 および Windows NT 3.5x 以前のバージョンでは動作しません。
- 日本語 Windows NT 環境で DOS 窓のサイズが 80 × 25 以外に設定されている場合、NC77、RASM77 等を起動すると窓のサイズが頻繁に切り替わります。窓のサイズを 80 × 25 に設定してください。

#### ファイル名についての注意事項

ソースプログラムファイルの名前や作業を行うディレクトリ名は、つぎの注意事項に従ってください。

- 漢字を含むディレクトリ名、ファイル名は使用できません。
- ファイル名に使用するピリオド (.) は一つのみ使用可能です。
- ネットワークパス名は使用できません。ドライブ名に割り当ててご使用ください。
- 「ショートカット」は使用できません。
- 空白文字を含むディレクトリ名、ファイル名は使用できません。
- "... " 表記を用いて 2 つ以上のディレクトリを指定することはできません。
- パス指定を含めたファイル名の長さが 128 文字以上になるものは使用できません。

#### ウイルスチェックプログラムに関する注意事項

ウイルスチェックプログラムが常駐した状態で NC77WA を起動すると正常に起動しない場合があります。その場合は、ウイルスチェックプログラムの常駐を解除してから NC77WA を起動しなおしてください。



## PC 版 NC77WA をバージョンアップするときの注意事項

NC77WA をバージョンアップする場合は、あらかじめ、インストールされている NC77WA をアンインストールしてから、新しいバージョンをインストールしてください。

### NC77WA のアンインストール手順

NC77WA をアンインストールするには、「コントロールパネル」-「アプリケーションの追加と削除」を選択しアンインストールを実行してください。

## 体験版のインストールに関する注意事項

RASM77 製品がインストールされた PC に NC77WA 体験版をインストールすると RASM77 体験版が上書きされてインストールされます。その場合は、再度 RASM77 製品版をインストールしなおしてください。

## 電子マニュアルに関する注意事項

現在、EWS 環境上では日本語の電子マニュアルを表示できません。

日本語環境に対応している Acrobat Reader が動作するパソコンでマニュアルを参照してください。Acrobat Reader が動作する環境については、Acrobat Reader のマニュアルやアドビシステムズ社のホームページなどをご確認ください ( <http://www.adobe.co.jp> )。

## NC77WA をインストールしよう

### インストールを始める前に

インストールを始める前に次の内容をご確認ください。

本製品の「使用権許諾契約書」、「リリースノート」などをよくお読みください。製品をインストールした場合は、契約書の記載内容に同意されたものとみなします。

NC77WA を快適に使用するには、32M バイト以上のメモリと 20M バイト以上の空きハードディスク領域が必要です。

製品のインストールは専用のインストーラを使用してください。

インストールの途中でライセンス ID を入力する必要があります。インストールを始める前にライセンス ID を確認してください。

RASM77 のライセンス ID では NC77WA はインストールできません。

## NC77WA をインストールしよう

### NC77WA のインストーラ

インストーラは、次に示す環境（対応ホスト、対応 OS、言語）毎に用意されています。ご購入になった製品を確認の上、該当するインストーラを使用してください。

#### 日本語環境

対応ホスト	対応 OS	インストーラ名	CD-ROM 上のディレクトリ
PC	Windows 95	SETUP.EXE	¥NC77WA¥W95J
	Windows 98		
	Windows NT		
SPARC Station	SunOS 4.x	setup	/nc77wa/sparc
	Solaris2.x	setup	/nc77wa/solaris
HP9000/700	HP/UX 10.x	setup	/nc77wa/hp700

#### 英語環境

対応ホスト	対応 OS	インストーラ名	CD-ROM 上のディレクトリ
	Windows 95	SETUP.EXE	\NC77WA\W95E
	Windows 98		
	Windows NT		

### RASM77 のインストーラ

インストーラは、次に示す環境（対応ホスト、対応 OS、言語）毎に用意されています。ご購入になった製品を確認の上、該当するインストーラを使用してください。

#### 日本語環境

対応ホスト	対応 OS	インストーラ名	CD-ROM 上のディレクトリ
PC	Windows 95	SETUP.EXE	¥RASM77¥W95J
	Windows 98		
	Windows NT		
SPARC Station	SunOS 4.x	setup	/rasm77/sparc
	Solaris2.x	setup	/rasm77/solaris
HP9000/700	HP/UX 10.x	setup	/rasm77/hp700

#### 英語環境

対応ホスト	対応 OS	インストーラ名	CD-ROM 上のディレクトリ
	Windows 95	SETUP.EXE	\RASM77\W95E
	Windows 98		
	Windows NT		

## PC 版のインストール

PC 版の製品は次の手順でインストールしてください。

- (1) CD-ROM 上の対象製品のインストーラが配置されているディレクトリに移動します。
- (2) インストーラを起動して表示されるメッセージにしたがってインストールを完了してください。

### 注意！

インストールの途中で入力するデータは、ユーザー登録のためのファイルを作成するのに使用されます（ファイルを作成するのは PC 版のインストーラのみです）。

## EWS 版のインストール

EWS 版のインストールは次の手順にしたがってインストールしてください。

以下の説明では、SOLARIS 版をもとに記述しています。他のホストにインストールする場合はディレクトリ名を読みかえてください。

- (1) インストーラのあるディレクトリに移動  
CD-ROM 上の、対象製品のインストーラが配置されているディレクトリに移動します。  
ここでは、CD-ROM をマウント<sup>注</sup>したディレクトリ名を /cdrom と記述しています。システムに設定されているディレクトリ名を読みかえてください。

### 注) HP9000/700 版の CD-ROM マウント方法

HP-UX 上では、CD-ROM のファイル名が大文字で見えたり、「;1」と言った表示が付加されることがあります。この場合は製品を取り出すことができませんので、下記の例にしたがってマウントをやり直してください。

```
% pfs_mountd &
% pfsd 4 &
% pfs_mount -t iso9660 -x unix /dev/dsk/c0t2d0 /cdrom
```

### 注意！

- ・スーパーユーザが実行してください。
- ・CD-ROM のドライブ名 (/dev/dsk/c0t2d0 )はホストマシンにより異なりますので別途ご確認ください。
- ・マウントポイント ( /cdrom ) は例です。
- ・例で使用している各コマンドについての詳細は、HP-UX のマニュアルを参照ください。
- ・% はプロンプトを表します。

## NC77WA をインストールしよう

- (2) インストーラを実行  
setup コマンドを実行してください。
- (3) ライセンス ID を入力  
-(ハイフン) も含めて正確に入力してください。  
製品に同梱されているライセンス ID 証書に書かれている ID を入力してください。  
ライセンス ID に誤りがあると、下記のメッセージが表示されます。この場合は、もう一度ライセンス ID を入力してください。

```
Illegal license ID. '1234-5678-9012-3456-7890'
```

- (4) 圧縮ファイルを指定  
data.0 があらかじめ指定されています。そのままリターンキーを押してください。
- (5) インストール先ディレクトリを指定  
指定したディレクトリが存在する場合は、このディレクトリが書き込み可能であり、かつ十分な空き容量があることが必要です。
- (6) インストール先のディレクトリを確認  
指定したディレクトリが存在しない場合、ディレクトリを作成するかを尋ねられます。y(はい) または n(いいえ) を入力してを押してください。

### 注意！

---

ここで y (はい) を選んだ場合でもディレクトリが作成できないことがあります。この場合は、インストールを中断して、OS の mkdir コマンドでディレクトリを作成してから、インストールをやり直してください。

---

- (7) 作業用ディレクトリを指定  
インストーラの作業のために一時的に使用するディレクトリです。  
ここで指定するディレクトリは、書き込み可能であり、かつ十分な空き容量を持っていることが必要です。  
/tmp があらかじめ指定されています。/tmp に十分な空き容量がある場合は、何も入力せずにそのままリターンキーを押してください。

## (8) ファイル転送の開始

転送中のファイルの名前が逐次表示されます。表示が出るまで時間がかかる場合があります。

入力したライセンス ID が対象製品と一致しない場合は、ファイル転送は行われず次のメッセージを表示して処理を中止します。この場合は、インストーラの実行からやり直してください。

```
License ID error 'data.0'
```

## (9) ファイル転送の終了

メッセージに表示されるファイル（名前、ディレクトリはメッセージ参照）に、製品情報を記録したファイルが作られます。

ディレクトリ名は、インストール先ディレクトリにより変わります。

次に手順(1)~(9)にしたがってインストールを実行したときの画面表示例を示します。

- ```
(1) >cd /cdrom/nc77wa/solaris
(2) >./setup
Tool Installation Utility V.1.00.00
Copyright(c) 1998 MITSUBISHI ELECTRIC CORPORATION
AND MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION
All Rights Reserved.

(3) Please type license ID.[xxxx-xxxx-xxxx-xxxx-xxxx]

(4) Program package file name[data.0]

(5) Install Directory[/usr/local/mtool] /usr/common/tool

(6) The directory '/usr/common/tool' does not exist.
Do you want the directory to be created (y/n) y

(7) Installer Temporary Directory[/tmp]

(8) Setup made a file, named 'toolinfo.txt', at the directory
"/usr/common/tool/support/nc77wa".
And Setup recorded product information in this file.
Please use this file if you need calling us.

(9) Installation completed.
```

## インストール後に生成されるディレクトリとソフトウェア

インストールが完了するとインストーラのデフォルトのインストール先ディレクトリの下に、表に示めずディレクトリが生成され、各ディレクトリに表のファイルがコピーされます。

PC 版のインストール先ディレクトリ (デフォルト)

¥mtool

EWS 版のインストール先ディレクトリ (デフォルト)

/usr/local/mtool

### 注意!

EWS 対応版では実行形式のファイルに拡張子.exe はつきません。

インストール先ディレクトリとインストールファイル一覧 -1-

| ディレクトリ | ファイル                                           |
|--------|------------------------------------------------|
| bin    | nc77.exe (コンパイルドライバ)                           |
|        | stk77.exe (スタックサイズ算出ユーティリティ)                   |
|        | rasm77.exe (アセンブラ)                             |
|        | pre77.exe (構造化プリプロセッサ)                         |
|        | link77.exe (リンケージエディタ)                         |
|        | lib77.exe (ライブラリアン)                            |
|        | s2ie.exe (IEEE-695 ファイルコンバータ)                  |
|        | hextos2.exe (モトローラ S フォーマットファイルコンバータ)          |
|        | crf77.exe (クロスリファレンサ)                          |
|        | br77.exe (分岐最適化ツール)                            |
|        | make.exe                                       |
|        | NC77WA ヘルプファイル                                 |
| lib77  | cpp77.exe (C 言語プリプロセッサ)                        |
|        | ccom77.exe (C コンパイラ本体)                         |
|        | loop77.exe (ブランチオブティマイザ)                       |
|        | nc77lib.lib (ランタイムおよび標準ライブラリ)                  |
|        | nc77lib.stk (ライブラリ関数のスタック使用量表示ファイル)            |
|        | nc50lib.lib (7750 シリーズ専用ランタイムおよび標準ライブラリ)       |
|        | nc50lib.stk (7750 シリーズ専用ライブラリ関数のスタック使用量表示ファイル) |

## インストール先ディレクトリとインストールファイル一覧-2-

| ディレクトリ         | ファイル                                                                                                                      |
|----------------|---------------------------------------------------------------------------------------------------------------------------|
| inc77          | 標準ヘッダファイル                                                                                                                 |
| src77¥startup  | ncrt0.a77 (スタートアッププログラム)<br>section.inc (メモリ配置設定インクルードファイル)                                                               |
| smp77          | makefile (NC77 用サンプル make ファイル)<br>ncrt0.a77 (スタートアッププログラム)<br>smp77.c (C 言語サンプルプログラム)<br>section.inc (メモリ配置設定インクルードファイル) |
| support        | mtoolspt.htm (ツールサポートホームページアドレス)                                                                                          |
| support¥nc77wa | regist.txt (ユーザー登録用紙)<br>Support.txt (技術サポート連絡書テキストファイル)                                                                  |
| manual         | nc77uj.pdf (NC77 電子マニュアル)<br>rasm77uj.pdf (RASM77 電子マニュアル)<br>br77j.pdf (br77 電子マニュアル)                                    |

## 電子マニュアルのインストールについて

EWS 版のインストーラでは電子マニュアルはインストールされません。

EWS 版をご購入の場合、日本語環境に対応している Acrobat Reader が動作するパソコンでマニュアルを参照してください。Acrobat Reader が動作する環境については、Acrobat Reader のマニュアルやアドビシステムズ社のホームページなどをご確認ください。

<http://www.adobe.co.jp>



## NC77WA をインストールしよう

### インストール後の環境設定を行う

インストールが完了した後、次の環境変数を設定してください。

#### PC の環境設定

表の中の「自動」は、PC 版のインストーラが AUTOEXEC.BAT を書きかえます。したがって、デフォルトでインストールを実行した場合は、AUTOEXEC.BAT を書きかえる必要はありません。

| 環境変数   | 設定例                             |
|--------|---------------------------------|
| BIN77  | 自動 ( SET BIN77=C:%MTOOL%BIN )   |
| INC77  | 自動 ( SET INC77=C:%MTOOL%INC77 ) |
| LIB77  | 自動 ( SET LIB77=C:%MTOOL%LIB77 ) |
| TMP77  | 自動 ( SET TMP77=C:%MTOOL%TMP )   |
| NCKIN  | SET NCKIN=SJIS                  |
| NCKOUT | SET NCKOUT=SJIS                 |
| コマンドパス | 自動 ( C:%MTOOL%BIN を追加 )         |

#### EWS の環境設定

製品のインストール完了後、プログラムを実行する前に以下の環境変数を設定してください。

注意！

環境変数名は大文字で設定してください。

| 環境変数   | 設定例                                   |
|--------|---------------------------------------|
| BIN77  | >setenv BIN77 /usr/local/mttool/bin   |
| INC77  | >setenv INC77 /usr/local/mttool/inc77 |
| LIB77  | >setenv LIB77 /usr/local/mttool/lib77 |
| TMP77  | >setenv TMP77 /usr/local/mttool/tmp   |
| NCKIN  | >setenv NCKIN EUC                     |
| NCKOUT | >setenv NCKOUT EUC                    |
| コマンドパス | /usr/local/mttool/bin を追加             |

## NCKIN と NCKOUT 環境変数について

NCKIN と NCKOUT に、NC77 を実行した場合の入出力コードの設定をします。

### NCKIN

入力コード系を指定する環境変数です。次の二つの環境から設定できます。

#### EUC

日本語文字の記述が可能です。ただし、1文字が3バイトで構成される外字コードは使用できません。

#### SJIS

日本語文字の記述が可能です。

### NCKOUT

出力コード系を指定する環境変数です。次の二つの環境から設定できます。内容は NCKIN と同様です。

- EUC
- SJIS

## NCKIN,NCKOUT のデフォルト（環境変数を設定しなかった場合）の設定

| ホスト   | 環境変数   | デフォルト値 |
|-------|--------|--------|
| PC 版  | NCKIN  | SJIS   |
|       | NCKOUT | SJIS   |
| EWS 版 | NCKIN  | EUC    |
|       | NCKOUT | EUC    |

### 注意！

日本語及び英語以外（たとえばヨーロッパ語圏）でコンパイラを使用される際は、必ず EUC と設定してください。

## NC77WA をインストールしよう

### ユーザー登録をしましょう

バージョンアップ情報や技術サポートなどのサービスを受けるためにユーザー登録を行ってください。ユーザー登録をされていない場合は、これらのサービスを受けることができません。

また、ご購入後 30 日以内 に登録してくださるようお願い申し上げます。

#### PC 版のユーザー登録方法

- 1 PC 版をインストールすると以下のファイルが生成されます。

```
¥mtool¥support¥nc77wa¥regist.txt
```

¥mtool はデフォルトでインストールした場合のディレクトリです。

- 2 regist.txt のファイル内容をすべてカット & ペーストして以下の電子メールアドレス宛に送付してください。

```
regist@tool.mesc.co.jp
```

電子メールをご使用になれない場合は、regist.txt のファイル内容をプリントアウトしファクシミリで送付してください。送付先は、製品に添付されているリリースノートで確認してください。

#### EWS 版のユーザー登録方法

- 1 ライセンス ID 証書に添付されているフォームに必要事項を記入しファクシミリで送付してください。  
送付先は、製品に添付されているリリースノートで確認してください。

### Acrobat Reader のインストール

電子マニュアルを参照するためには Acrobat Reader が必要になります。CD-ROM に Acrobat Reader を添付していますので必要に応じてインストールしてください。

- ・ インストーラは製品 CD-ROM 上で起動してください。
- ・ 添付されている Readme ファイルの内容にしたがってインストールしてください。

また、Acrobat Reader は、アドビシステムズ社のホームページからダウンロードすることができます。Acrobat Reader の最新情報などはつぎの URL を参照してください。

```
http://www.adobe.co.jp
```

```
http://www.adobe.com
```

## freeware に格納されているプログラムについて

製品 CD-ROM の freeware ディレクトリには、製品の機能を補足する機能をもつプログラムが格納されています。これらのプログラムはインストーラではインストールしません。各プログラム毎に CD-ROM から直接インストールしてください。インストール方法および機能の詳細については、各プログラム毎のドキュメントファイルを参照してください。

### chk77

複数ファイル間での関数の引数、戻り値および外部変数の型検査を行います。

### dbsym77

NC77 のコマンドオプション -g を追加したときに出力されるシンボルファイルから NC77 が生成するローカルシンボルを削除します。

### dump695

IEEE-695 フォーマットのオブジェクトモジュールおよびロードモジュールをダンプします。

### sfr77

特殊機能レジスタ(SFR) にたいして不正な命令でアクセスしている場合に警告メッセージを出力します。

### lst77

アセンブラが生成するリストファイルのアドレスデータをリンク後の絶対番地書き換えます。

### xref77

ローカル、グローバルシンボルの相互参照リストを生成します。

### 注意

---

freeware は動作保証、保守は原則として行いません。本ソフトウェアを使用することにより、直接的、間接的に発生した損害については三菱電機株式会社および三菱電機セミコンダクタシステム株式会社は一切の責任を負いません。

---

NC77WA をインストールしよう



## NC77WA を使ってみよう

### NC77WA をご使用になる前に

NC77WA をご使用になる前に次の内容をご確認ください。

「使用権許諾契約書」、「リリースノート」をよくお読みください。

NC77WA を快適に使用するには、32M バイト以上のメモリと 20M バイト以上の空きハードディスク領域が必要です。

以下の URL に開設していますホームページなどで、製品に関する最新情報をご確認ください。

<http://www/tool-spt.mesc.co.jp>

製品に添付されているスタートアップファイル( ncr0.a77, section.inc )をご使用になる場合は、ご使用のマイコン機種、お客様のシステムにあうように内容を変更する必要があります。

マイコン機種による変更点についてはマイクロコンピュータデータブックなどをご参照ください。

ここでは、製品に添付されているインストーラの標準で、パソコン上に製品をインストールしたものとして説明を行っています。インストールの際にディレクトリ名などを変更している場合は、ディレクトリ名を読み替えてください。

RASM77 製品をご購入の場合、C コンパイラと TM はご使用になれません。

## NC77WA を使ってみよう

## NC77 を使ってみよう

製品に添付されているサンプルファイルを使用してコンパイル手順を説明します。  
インストーラのデフォルトで製品をインストールした場合、次の C 言語のサンプルファイルが  
¥mtool¥smp77 ディレクトリにインストールされます。

ncrt0.a77

スタートアッププログラムのサンプルファイルです。

section.inc

スタートアッププログラムのサンプルファイルです。

samp77.c

C 言語プログラムのサンプルファイルです。

### (1) サンプルファイルをワークディレクトリにコピーします

```
>mkdir C:¥local¥work  
>copy C:¥mtool¥smp77¥*. * C:¥local¥work  
>cd C:¥local¥work
```

### (2) ncrt0.a77 / section.inc を修正します

ここでは、サンプルファイルncrt0.a77 / section.inc をそのまま使用します。

注意！

---

プログラム開発の際は、開発するシステムの構成にあうようにこれらのプログラムを書きかえてください。サンプルファイルの詳細については「NC77WA のスタートアッププログラム」を参照してください。

---

### (3) C 言語でプログラムを作成する

ここでは、サンプルファイルsamp77.c をそのまま使用します。

注意！

---

C 言語でプログラムを記述する際は次の内容に注意してください。  
NC77 は最適化オプションの有無に関わらず、メモリへの転送や演算を実行する場合、一時的にそのメモリの内容を書きかえた後で、転送や演算を行なう場合があります。すなわち、割り込みなどで非同期にそのメモリを参照すると期待した結果が参照できない場合があります。  
したがって、割り込み処理プログラムやリアルタイム OS のタスク間で共有する変数（メモリ）に対して、その変数を volatile 修飾子で宣言する、割り込みの一時禁止を行なうなどの排他制御を必ず行なってください。

---

#### (4) コンパイラを実行します

##### 1 スタートアッププログラム (ncrt0.a77) をアセンブルします

はじめに、スタートアッププログラムをアセンブルします。nc77 コマンドを実行してリロケータブルオブジェクトファイル ncrt0.r77 を生成します。

```
>nc77 -c ncrt0.a77
```

##### 2 C 言語サンプルプログラム (samp77.c) をコンパイルします

nc77 コマンドを実行してリロケータブルオブジェクトファイル samp77.r77 を生成します。

```
>nc77 -c samp77.c
```

##### 3 ncrt0.r77 と samp77.r77 をリンクします

link77 コマンドを実行して 16 進機械語ファイル samp77.HEX を生成します。

```
>link77 ncrt0 samp77,,, -fsamp77
```

#### 注意！

---

エラーが発生した場合は、エラーメッセージにしたがってファイルを修正してください。

---



## NC77WA を使ってみよう

## RASM77 を使ってみよう

RASM77 では、アセンブリソースプログラムを処理する場合、構造化記述を使用している場合と、使用していない場合とでコマンド入力が異なります。

任意のワークディレクトリでアセンブラを実行してください。

```
>mkdir C:¥local¥work  
>cd C:¥local¥work
```

### アセンブリ言語だけで記述した場合のコマンド入力

アセンブリソースファイルを rasm77 コマンドでアセンブルし、アセンブルエラーおよびワーニングがなければ link77 コマンドで機械語ファイルを生成します。

- 1 sample.a77 をエディタなどで作成

ファイル記述例)

```
.SECTION    sample  
NOP  
NOP  
.END
```

- 2 rasm77 コマンドを実行

```
>rasm77 sample
```

- 3 link77 コマンドを実行

```
>link77 sample,,,,
```

## 構造化記述を使用している場合のコマンド入力

構造化記述を含むソースファイルを pre77 コマンドで構造化記述をアセンブリ記述に変換します。pre77 の実行結果にエラーおよびワーニングがなければ、pre77 が生成したアセンブリソースファイルをアセンブル (rasm77 コマンド)、リンク (link77 コマンド) します。

- 1 strsm7.p77 ファイルをエディタなどで作成

ファイル記述例)

```
.SECTION    sample
A = 0
B = 0
.END
```

- 2 pre77 コマンドを実行

```
>pre77 strsm7
```

- 3 rasm77 コマンドを実行

```
>rasm77 strsm7
```

- 4 link77 コマンドを実行

```
>link77 strsm7,,,,
```

## 電子マニュアルを使ってみよう

本製品の電子マニュアルは PDF ( Portable Document Format ) ファイルで提供しています。マニュアルを参照される場合は、製品に添付の Acrobat Reader などの PDF ファイル表示プログラムを使用してください。

電子マニュアルは、インストーラによって mtool ( デフォルトでインストールした場合 ) の下の次のディレクトリにインストールされます。

### 電子マニュアルのインストールディレクトリ

| ディレクトリ | PDF ファイル   | 内容               |
|--------|------------|------------------|
| manual | nc77uj.pdf | NC77 電子マニュアル日本語版 |
|        | as77uj.pdf | AS77 電子マニュアル日本語版 |
|        | br77j.pdf  | br77 電子マニュアル日本語版 |
| manual | nc77ue.pdf | NC77 電子マニュアル英語版  |
|        | as77ue.pdf | AS77 電子マニュアル英語版  |
|        | br77ue.pdf | br77 電子マニュアル英語版  |

### 注意！

EWS 対応版のインストーラは電子マニュアルをインストールしません。製品 CD-ROM から各電子マニュアルの PDF ファイルを Acrobat Reader がインストールされている PC 上にコピーして参照してください。電子マニュアルは CD-ROM の ¥NC77WA¥MANUAL に格納されています。

各マニュアルは、目次または「しおり」にリンクがはられています。ここから目的のページを開くことができます。

電子マニュアルを表示するには次の方法があります。

スタートメニューからオープンする

スタートメニューに登録されるので、[スタート] [プログラム (P)] [MITSUBISHI-TOOL]の目的のツール名からマニュアルを起動することもできます。

PDF ファイルをダブルクリックする

参照したい電子マニュアルの PDF ファイルをダブルクリックすると、Acrobat Reader がマニュアルのデータを読み込んで起動します。

Acrobat Reader を起動して開く

または、Acrobat Reader を起動してから「ファイル-開く ...」メニューで電子マニュアルを指定してください。

## オンラインヘルプを使ってみよう

本製品には、NC77WA のオンラインヘルプが用意されています。オンラインヘルプは、Windows のヘルプ環境に対応しています。

オンラインヘルプは、インストーラによって mtool ( デフォルトでインストールした場合 ) の下の次のディレクトリにインストールされます。

| ディレクトリ | 日本語ヘルプファイル                 | 英語ヘルプファイル                  |
|--------|----------------------------|----------------------------|
| bin    | nc77waj.cnt<br>nc77waj.hlp | nc77wae.cnt<br>nc77wae.hlp |

オンラインヘルプを利用するには次の方法があります。

スタートメニュー から起動する

スタートメニューに登録されるので、[スタート] [プログラム (P)] [MITSUBISHI-TOOL]から起動することもできます。

オンラインヘルプを直接起動する

オンラインヘルプを起動するには、ヘルプファイル ( 拡張子 .hlp ) をダブルクリックしてください。

NC77WA を使ってみよう

➤ **memo** ➤

## NC77WA のスタートアッププログラム

### スタートアップサンプルプログラム ncrt0.a77 の内容

ncrt0.a77 は C 言語とマイコンの初期化などを行うスタートアッププログラムのサンプルです。アセンブリ言語で記述されています。

開発するシステムの構成にあうように ncrt0.a77 を書きかえます。

#### section.inc のインクルード部

メモリ配置を設定するファイル section.inc をインクルードします。この部分は書きかえないでください。

```

;*****
    .include    section.inc
;-----

```

#### セクションサイズ格納部

セクションサイズを格納します。

```

; section-size
;-----
    .section    rom_NE
    .pub       bss_NEsZ
    .pub       bss_NOsZ
    .pub       bss_FEsZ
    .pub       bss_FOsZ
    .pub       data_NEsZ
    .pub       data_NOsZ
    .pub       data_FEsZ
    .pub       data_FOsZ
bss_NEsZ:    .dword    sizeof bss_NE
bss_NOsZ:    .dword    sizeof bss_NO
bss_FEsZ:    .dword    sizeof bss_FE
bss_FOsZ:    .dword    sizeof bss_FO
data_NEsZ:   .dword    sizeof data_NE
data_NOsZ:   .dword    sizeof data_NO
data_FOsZ:   .dword    sizeof data_FO
data_FEsZ:   .dword    sizeof data_FE

```

### RAM 初期化マクロ定義部

RAM 領域を初期化するマクロ命令の定義です。この部分は書きかえないでください。

- BZERO  
RAM をゼロクリアします。
- BCOPY  
ROM にある初期値を RAM にコピーします。

```
;/=====
; Initialize Macro declaration
;-----
BZERO: .macro  SIZE, TOP
    lda A, DT:SIZE + 2
    pha
    lda A, DT:SIZE
    pha
    pea #bank TOP
    pea #offset TOP
    .ext    _bzero
    jsrl   _bzero
    .endm

BCOPY: .macro  SIZE, TO, FROM
    lda A, DT:SIZE + 2
    pha
    lda A, DT:SIZE
    pha
    pea #bank TO
    pea #offset TO
    pea #bank FROM
    pea #offset FROM
    .ext    _bcopy
    jsrl   _bcopy
    .endm
```

## ライブラリファイルの指定

コンパイラが読み込むライブラリファイルを指定します。

```

;=====
; Libraly file name definition
;-----
        .lib          nc77lib
  
```

### ライブラリファイルの注意事項

NC77WA のライブラリファイルは、nc77lib.lib および nc50lib.lib の 2 種類があります。指定するライブラリファイルをコンパイルオプション "-m7750" の有無にあわせて変更してください。

| -m7750 オプション | 指定するライブラリファイル |
|--------------|---------------|
| 指定なし         | nc77lib.lib   |
| 指定あり         | nc50lib.lib   |

## プログラムのスタートアドレス定義部

リセット直後のプログラム開始ラベルの定義です。この部分は書きかえないでください。

- ・ プログラム開始ラベル  
'start:' の次の行に書かれたラベルからプログラムが始まります。

### 注意！

section.inc の「固定ベクタテーブルの設定」のリセットベクトルにはこのラベルを設定します。

```

;=====
; Interrupt section start
;-----
        .pub      start
        .section  interrupt
start:
  
```

## データバンクレジスタの設定

near 型データを配置する領域のデータバンクレジスタ値を設定します。

- ・ データバンクレジスタ  
CPU アドレスの上位 8 ビットを設定します。データバンクレジスタで設定された範囲を near 領域として扱います。

### 注意！

コマンドオプション -bank=n で設定する値と同じ値を設定してください。

```

DT .equ      00h
   ldt #__DT      ; Initialize data bank register
  
```



### プロセッサ動作モードを設定します

開発するシステムの構成にあうようにプロセッサモードレジスタを設定してください。

- ・ プロセッサモードレジスタ  
スペシャルファンクションレジスタ領域にある、マイコンの動作モード（シングルチップモードなど）を設定するレジスタです。

```
sem
ldm.B      #24H,DT:5eH ; set processor mode register
clp      m,x,d
```

#### 注意！

nc77 の起動オプション `-bank=` で `near` 領域のバンクを 0 以外に設定したときは、次に示す方法で設定してください。

```
sem
lda.W      #24H          ; set processor mode register
sta.W      LG:5eH
clp      m,x,d
```

### スタックポインタ設定部

スタックポインタ (S) の値を設定します。

#### 注意！

この部分は書きかえないでください。

```
lda.w      a, #offset stack_top - 1 ; Initialize stack
tas
```

## near 領域の bss セクションをゼロクリアします

near 領域の bss セクションをゼロクリアします。

- ANSI-C 規格では初期値を持たない RAM 上の変数はゼロクリアすることになっています。bss セクションのゼロクリアはこの規格に従って行っていきます。開発するシステムの構成にあわせてゼロクリアをしないようにコメントにしてください。
- また、一括して RAM 領域を一度にゼロクリアするように変更することも可能です。

## bss セクション

bss セクションは C 言語で初期値を持たない RAM 上の変数の領域 (セクション) です。

## セクション名

bss\_NE、bss\_NO などはセクション名です。アンダーバー ( \_ ) の後の文字は以下の内容を示します。

- N: near 領域にある変数
- F: far 領域にある変数
- E: 偶数サイズの変数
- O: 奇数サイズの変数

```
; NEAR area initialize.
;-----
; bss_NE & bss_NO zero clear
;-----
    BZERO    bss_NEsZ,bss_NE_top
    BZERO    bss_NOsz,bss_NO_top
```

## near 領域の data セクションを初期化します

ROM 上に格納されている変数の初期値を RAM に転送する処理を行います。

初期値を持つ RAM 上の変数がない場合はこの部分をコメントにしてください。また、各セクションが連続している場合は初期値を一度に転送するように変更することも可能です。

- data セクション  
C 言語で初期値をもつ RAM 上の変数の領域 (セクション) です。

```
; Copy data_NE(NO) section from data_INE(INO) section
;-----
    BCOPY    data_NEsZ,data_NE_top,data_INE_top
    BCOPY    data_NOsz,data_NO_top,data_INO_top
```

## NC77WA のスタートアッププログラム

far 領域の bss セクションをゼロクリアします

far 領域の bss セクションをゼロクリアします。

注意！

far 領域の RAM データを使用しない場合やシングルチップモードで使用する場合はコメントにしてください。

```
; FAR area initialize.
;-----
; bss_FE & bss_FO zero clear
;-----
    BZERO    bss_FEsz,bss_FE_top
    BZERO    bss_FOsiz,bss_FO_top
```

far 領域の data セクションを初期化します

far 領域の data セクションの初期値を RAM 領域に転送します。

注意！

far 領域の RAM データを使用しない場合やシングルチップモードで使用する場合はコメントにしてください。

```
; Copy data_FE(FO) section from data_IFE(IFO) section
;-----
    BCOPY    data_FEsiz,data_FE_top,data_IFE_top
    BCOPY    data_FOsiz,data_FO_top,data_IFO_top
```

スタックポインタの再設定部

BZERO マクロ等で使用したスタックを元に補正します。この部分は書きかえないでください。

```
    lda.w    a,#offset stack_top - 1
    tas
```

## heap 領域を初期化します

メモリ管理関数で使用する heap 領域の初期化を行います。

## 注意！

メモリ管理関数を使用しない場合はコメントにしてください。

- heap  
メモリ管理関数 ( malloc など ) によって、プログラム実行中に動的に確保されるメモリ領域です。

```

; heap initialize
;-----
        .ext      __mbase, __mnext, __msize
        lda.w    a, #offset heap_top
        lda.w    b, #bank heap_top
        sta      a, __mbase
        sta      b, __mbase + 0002h
        sta      a, __mnext
        sta      b, __mnext + 0002h
        lda.w    a, #offset HEAPSIZE
        lda.w    b, #bank HEAPSIZE
        sta      a, __msize
        sta      b, __msize + 0002h

```

## 標準入出力の初期化を行います

標準入出力の初期化を行う init 関数を呼び出します。

## 注意！

入出力関数を使用しない場合は必ずコメントにしてください。

```

;=====
; Initialize standard I/O
;-----
        .ext      _init
        jsrl     _init

```

## main 関数の呼び出し部

main 関数の呼び出しを行います。この部分は書きかえないでください。

## 注意！

main 関数呼び出し時には M、X フラグともにクリアされた状態で呼び出して  
ください。

```

;=====
; Call main() function
;-----
        .ext      _main
        jsrl     _main

```

## exit 関数部

プログラムを終了させる exit 関数部です。

```
=====
; exit() function
;-----
    .func      _exit
    .pub      _exit
_exit:          ; End program
    bra      _exit
    .endfunc  _exit
    .func      ?exit
    .pub      ?exit
?exit:          ; End program
    bra      ?exit
    .endfunc  ?exit
```

## ダミー割り込み処理関数部

ダミーの割り込み処理関数部です。

```
=====
; dummy interrupt function
;-----
dummy_int:
    rti
    .end
```

## スタートアップサンプルプログラム section.inc の内容

プログラムやデータ等をメモリのどの領域に配置するかを設定します。配置はセクション単位で行います。ncrt0.a77 ファイルからインクルードされます。

### near 領域の RAM 配置

サンプルでは、RAM データ領域の配置を行っています。配置を変更する場合はアセンブリ言語の .ORG 指示命令で番地を指定してください。番地が指定されていないセクションはその前のセクションの直後に配置されます。

```
    .section    data_NE
    .org      80H
data_NE_top:
;
    .section    data_NO
data_NO_top:
;
    .section    bss_NE
bss_NE_top:
;
    .section    bss_NO
bss_NO_top:
```

## スタックサイズを設定します

使用するスタックサイズを定義してください。

スタック領域は以下のデータを格納する領域です。

記憶クラス auto の変数

複雑な演算を実行する場合のテンポラリ

関数呼び出し元への戻りアドレスと旧フレームポインタアドレス

関数への引数

64 ビットの浮動小数点を格納する内部レジスタを配置

スタックサイズは、スタックサイズ算出ユーティリティ stk77 を用いて求めることができます。

注意！

stk77 の使用方法は「NC77WA ヒント集」の「stk77 を使ってスタック使用量を計算しよう」を参照してください。

```
.section    stack
.blkb     300H          ; stack size
stack_top:
```

## heap サイズを設定します

使用する heap サイズを定義してください。

```
HEAPSIZE    .equ     300h
```

## heap セクション配置

メモリ操作関数を使用しない場合はコメントにしてください。

サンプルでは、セクションの番地を指定していないため、その前のセクションの直後、すなわち near 領域に配置されますが、heap セクションは far 領域の RAM に配置することもできます。その場合は、次の部分を far 領域に配置されるように移動してください。

```
.section    heap
heap_top:
.blkb     HEAPSIZE
```

## near 領域の ROM 配置

near 領域に ROM データを配置しない場合は、以下の部分をコメントにしてください。

```
;-----
; Near ROM data area
;-----
.section    rom_NE
rom_NE_top:

.section    rom_NO
rom_NO_top:
```

## 割り込みベクタテーブル開始アドレスを設定します

割り込みベクタテーブルの開始アドレスを定義してください。

```
.section    vector           ; Interrupt vector table
.org      0ffd6H
```

## 割り込みベクタテーブルの設定

vector セクションでは割り込みベクタアドレスを設定します。

### 注意！

割り込みベクタの内容は、7700 ファミリの機種により異なります。使用するマイコンにあわせて修正してください。

```
ADCOMP:
.word    dummy_int
TRN1:
.word    dummy_int
REC1:
.word    dummy_int
TRN0:
.word    dummy_int
REC0:
.word    dummy_int
BS2I:
.word    dummy_int
BS1I:
.word    dummy_int
BS0I:
.word    dummy_int
TA4I:
.word    dummy_int
TA3I:
.word    dummy_int
TA2I:
.word    dummy_int
TA1I:
.word    dummy_int
TA0I:
.word    dummy_int
INT2:
.word    dummy_int
INT1:
.word    dummy_int
INT0:
.word    dummy_int
WDT:
.word    dummy_int
```

```

RESERVED:
        .word    dummy_int
BRK:
        .word    dummy_int
DIV0:
        .word    dummy_int
;
RESET:
        .word    offset start

```

**注意！**

割り込み処理関数を設定するには "dummy\_int" の部分を割り込み処理関数の先頭ラベル名（関数名に \_ を付加した名前）に書きかえてください。

**例）UART1 送信割り込みを使用する場合**

```

        .section    vector ;interrupt vector table
        .ext        _uarttrn
        .org        0ffd6H
ADCMP:
        .word    dummy_int
TRN1:
        .word    _uarttrn
REC1:
        :

```

**far 領域の RAM 配置**

シングルチップモードで使用する場合等、この領域を使用しない場合はコメントにしてください。

```

        .section    data_FE
        .org        12000H
data_FE_top:
;
        .section    data_FO
data_FO_top:
;
        .section    bss_FE
bss_FE_top:
;
        .section    bss_FO
bss_FO_top:

```



## NC77WA のスタートアッププログラム

### far 領域の ROM 配置

サンプルでは、ROM データ、RAM データの初期値、プログラムを配置しています。

```
.section      program_F
.ORG         18000H
;
.section      rom_FE
rom_FE_top:
;
.section      rom_FO
rom_FO_top:
;
.section      data_IFE
data_IFE_top:
;
.section      data_IFO
data_IFO_top:
```

## NC77WA ヒント集

## stk77 を使ってスタック使用量を計算しよう

stk77 は C 言語で記述されたプログラムが使用するスタック容量を算出するプログラムです。

stk77 を使用するには nc77 実行時にコマンドオプション `-fshow_stack_usage` (`-fSSU`) を指定してスタック使用情報ファイル(拡張子 `.stk`) を生成しておく必要があります。

サンプルファイル `samp77.c` のスタック使用量を算出する例を示します。スタック使用量の算出結果はファイル `samp77.siz` に出力されます。

- 1 スタック使用情報ファイル(拡張子 `.stk`) をコンパイル時に生成する  
nc77 コマンドにオプションを付加してスタック使用情報ファイルを生成します。

```
>nc77 -c -fSSU samp77.c
```

- 2 stk77 を使用しスタック使用量を計算する  
stk77 コマンドで算出結果ファイル(拡張子 `.siz`) を生成します。

```
>stk77 -o samp77.stk
```

samp77.siz の内容

```
*** Stack Size ***  
  
                2 bytes  
  
*** C Flow ***  
  
main(samp77.stk)
```

### 割り込み関数からのスタック使用量を算出

stk77 は、main()関数からのスタック使用量を算出します。割り込み処理関数などの特定の関数から算出したい場合は、-e オプションを指定してください。

### 複数ファイルからスタック使用量を算出

アプリケーションプログラムが複数のソースファイルから構成される場合は、スタック使用情報ファイル（拡張子 .stk）を複数指定するか -s オプションでシンボルファイル（link77 が生成）を指定してください。

注意！

オプション -e および -s の詳細はマニュアルを参照してください。

## プログラムを ROM に書き込む準備をする

デバッグが終了したアプリケーションプログラムは、実際の応用製品により近い環境で動作評価をする必要があります。このとき使用する EPROM 内蔵マイコンやワンタイム PROM 内蔵マイコンにプログラムを書き込むためには次の準備が必要です。

- 1 ROM ライタ対応フォーマットファイルを生成
- 2 書き込みアダプタなどのツールを用意

### ROM ライタ対応フォーマットファイルを生成

hextos2 コマンドで機械語ファイル（拡張子 .hex）をモトローラ S フォーマットファイルに変換します。

### モトローラ S フォーマットファイルを生成

hextos2 コマンドで次のように入力してください。sample は機械語ファイル名です。生成されるファイルの拡張子は s2 です。

```
>hextos2 sample.hex<RET>
```

### 書き込みアダプタなどのツールを用意

製品に添付されているデータシートやホームページに掲載されている最新の情報を参考にして必要なハードウェアを用意してください。

```
http://www.tool-spt.mesc.co.jp
```

## Q&amp;A 集

## コマンドオプションが指定できない場合がありますがなぜですか？

NC77では、コマンドオプションの大文字と小文字を区別しています。コマンドオプションを指定する場合には、大文字小文字が一致していることを確認してください。

## リンクするファイル数が多くてコマンド行の制限を越えてしまいます。リンクする方法はありますか？

コマンド行の内容をファイルに書き込んで link77 を実行させることができます。このファイルをコマンドファイルと呼びます。コマンドファイルについての詳しい情報は次の項を参照してください。

## 「コマンドファイル」はどのように使うのですか？

ソフトウェアの実行時にコマンドファイルを指定するときは、通常ファイルとの区別のために、ファイル名の先頭に '@' を付加して指定します。

link77 のコマンドファイル指定例)

```
>link77 @cmdfile
```

コマンドファイル記述例)

```
MAIN SUB  
,  
,WORKRAM=100 PROM=C000 DROM  
,,
```

このコマンドファイル例は、次のように link77 コマンドを入力した場合と同じ結果になります。

```
>link77 MAIN SUB,,WORKRAM=100 PROM=C000 DROM,,
```

## #include 指定でファイル名にパスを指定できますか？

相対パス名を指定できます。コンパイラは、#include で指定されているファイルを #include が記述されているファイルからの相対ディレクトリで検索します。絶対パス指定は、プログラムの移植性をそこの可能性があるので禁止しています。

src\_1.c ファイル内で src\_1.h を #include で指定する場合の記述は次のようになります。

src\_1.c の記述

```
#include    "..\header\src_1.h"
:
```

ファイル構成

```
C:\work\src_1.c
C:\work\src_2.c
:
C:\header\src_1.h
C:\header\src_2.h
:
```

**注意！**

このとき、パスを含めたファイル名の指定を "%header%src\_1.h" と記述するとエラーとなります。

## レジスタ (register) 記憶クラスを指定して、それをレジスタに割り当てるには？

ローカル変数にレジスタ記憶クラスを指定し、その変数を確実にレジスタに割り当てるには、コンパイル時にコマンドオプション "-fenable\_register(-fER)" を指定してください。ただし、一部の型を持つ変数はレジスタに割り当てることができません。

レジスタ変数の宣言例)

```
register int    i
```

また、レジスタ変数は関数の中で、関数呼び出しをまたいで使用しないように記述してください。

## register修飾子を付けていない変数がレジスタ変数として扱われていますがなぜですか？

最適化コマンドオプション '-O'、'-OR' および '-OS' を指定してコンパイルした場合、最適化が行われた結果、レジスタ変数として扱われる場合があります。

## C言語で記述した関数名はアセンブリソース上ではどのような名前になるのですか？

関数名はアセンブリソース上ではラベル名として現れます。たとえば、関数名が "func\_proto" のとき、アセンブリソース上では "?func\_proto:" または "\_func\_proto:" として表されます。

"?func\_proto:" と "\_func\_proto:" で、どのような場合にどちらのラベルが選択されるかの規則は次のように決まっています。

?func\_proto

関数の引数のうち、一つ以上の引数がレジスタで渡されている関数

\_func\_proto

上記の条件にあてはまらないすべての関数

## 関数名に相当するラベルにたいしてリンク時に未定義シンボルであるというエラーが出力されますが原因は？

関数のプロトタイプが宣言されているものとされていないものが混在している可能性があります。

プロトタイプ宣言がされていない関数を見つけるには、コマンドオプション "-Wnon\_prototype(-WNP)" を指定してコンパイルを実行すると、該当する関数に対してワーニングが出力されます。見つけた関数をプロトタイプ宣言してください。

## 関数のプロトタイプ宣言は必要ですか？

関数をプロトタイプ宣言することで、関数の引数の型拡張を抑止したり、引数をレジスタに割り当てるなど、効率の良い関数呼び出しが可能になります。

## 最適化オプションをつけるとコードが生成されない行がありますが問題ありませんか？

コンパイラは、最適化オプションが指定されると文法上無意味であると思われるコードを生成しません。

最適化を抑止したい場合は、"volatile" 修飾子を指定して最適化されないようにしてください。

最適化される例)

- port を読み出すだけの関数

```
extern int port;
func()
{
    port;
}
```

最適化されない例)

```
volatile extern int port;
func()
{
    port;
}
```

## near/far 属性はどのように区別されますか？

NC77 は near/far が指定されていない場合は、関数を far、変数を near として扱います。

変数は通常 near として扱われますが、コマンドオプションを指定するとデフォルトの属性を変更できます。

- -ffar\_ROM\_data(-fFROM)  
ROM データのデフォルト属性を far にします。
- -ffar\_RAM\_data(-fFRAM)  
RAM データのデフォルト属性を far にします。

## ROM 容量を圧縮するには？

far 領域に配列データがあって、そのサイズが 64K バイトを越えないとき有効な方法として、要素数を指定していない配列が far 領域にある場合に、-fsmall\_array (-fSA) オプションを付加してコンパイルを実行してください。ROM 容量を圧縮し、処理速度を高速にできる可能性があります。

## スタートアッププログラムの処理時間を短くするには？

スタートアッププログラム ncrt0.a77 に記述されているセクションの初期化およびゼロクリアのうち使用しない部分を削除することにより処理時間を短縮できます。また、ゼロクリアを一括して行うことにより処理時間を短縮できる場合があります。

## 高級言語ソースレベルでデバッグを行うには？

コマンド実行時に、Cのソース行や構造化記述文のソース行でデバッグを行うための情報を機械語ファイルに出力するようにコマンドオプションを指定してください。

```
>nc77 -gie -c samp77.c  
>rasm77 -s ncrt0.a77  
>link77 ncrt0 samp77,,, -ms -s -fsamp77  
>s2ie samp77.sym
```





## 付録 A NC77WA コマンドオプション一覧

## nc77 コマンドオプション

## nc77 の起動オプション

**-C**

**-c**  
リロケータブルファイル生成

**-D**

**-Ddata1=0x2F**  
識別子を定義(#define と同様)

**-I**

**-Idirname**  
#include ファイルの検索ディレクトリを指定

**-E**

**-E**  
プリプロセスコマンドのみ起動(結果は標準出力)

**-P**

**-P**  
プリプロセスコマンドのみ起動(中間ファイル生成)

**-S**

**-S**  
アセンブリ言語ソースファイル生成

**-U**

**-Umacroname**  
指定したプリデファインドマクロ名を無効化

**-silent**

**-silent**  
起動時のコピーライトメッセージ非表示

**-time**

**-time**  
プロセスの処理時間を表示 (UNIX 版のみ)

## 出力ファイル指定オプション

**-O**

**-ofilename**  
リンカの生成ファイル名称を指定

**-dir**

**-dir/usr/disk**  
出力ファイルの出力先ディレクトリを指定します。

## バージョン情報表示オプション

**-V**

**-v**  
実行中のコマンドプログラム名とコマンドラインを表示

**-V**

**-V**  
コンパイラの各プログラムの起動時メッセージを表示し処理終了

## デバッグ用オプション

**-g**

**-g**  
シンボルファイル(属性 .sym)を生成

**-genter**

**-genter**  
関数呼び出し時にスタックフレームを生成

## 付録 A NC77WA コマンドオプション一覧

### -gie

-gie  
IEEE-695 形式ファイルを生成

### -gie\_no\_local\_symbol/-gINLS

-gie\_no\_local\_symbol/-gINLS  
ローカルシンボル情報を持たない IEEE-695 形式ファイルを生成

## 最適化オプション

### -O[1-5]

-O  
-O1  
-O4  
速度、ROM 効率ともに良くなる (影響を与えない) 最適化を行います。  
レベルが高いほど最適化が強くなります。  
レベルを省略した場合はレベル 3 で処理します。  
レベル 5 を指定する場合は必ずマニュアルを参照してください。

### -OR

-OR  
ROM 容量を重視した最適化を実行

### -OS

-OS  
速度を重視した最適化を実行

### -Oconst/-OC

-Oconst/-OC  
const 修飾子にたいする最適化を実行

### -Ono\_bit/-ONB

-Ono\_bit/-ONB  
ビット操作をまとめる最適化を抑制

### -Ono\_break\_source\_debug/-ONBSD

-Ono\_break\_source\_debug/-ONBSD  
ソース行情報に影響のある最適化を抑制

### -Ono\_float\_const\_fold/-ONFCF

-Ono\_float\_const\_fold/-ONFCF  
浮動小数点の定数量み込み処理を抑制

### -Osp\_adjust/-OSA

-Osp\_adjust/-OSA  
スタック補正コードを取り除く最適化を実行

### -Ostack\_frame\_align/-OSFA

-Ostack\_frame\_align/-OSFA  
スタックフレームの偶数アライメントを実行

### -Ocompare\_byte\_to\_word/-OCBTW

-Ocompare\_byte\_to\_word/-OCBTW  
連続した領域のバイト単位の比較をワードで行います。

### -Ono\_logical\_or\_combine/-ONLOC

-Ono\_logical\_or\_combine/-ONLOC  
論理 OR をまとめる最適化を抑制します。

## 分岐命令の選択オプション

### -OB1

-OB1  
コードサイズのみを考慮し分岐命令を生成

### -OB2

-OB2  
速度を考慮し同一バンク内で分岐命令を生成

### -OB3

-OB3  
速度を考慮しバンク外の分岐命令を生成

## 生成コード変更オプション

### -fansi

-fansi  
-fnot\_reserve\_far\_and\_near、-fnot\_reserve\_asm、  
-fnot\_reserve\_inline、及び -fextend\_to\_int を有効化

**-fnot\_reserve\_asm/-fNRA**

-fnot\_reserve\_asm/-fNRA  
asm を予約語にしない (\_asm のみ有効)

**-fnot\_reserve\_far\_and\_near/-fNRFAN**

-fnot\_reserve\_far\_and\_near/-fNRFAN  
far、near を予約語にしない (\_far、\_near のみ有効)

**-fnot\_reserve\_inline/-fNRI**

-fnot\_reserve\_inline/-fNRI  
inline を予約語にしない (\_inline のみ有効)

**-fextend\_to\_int/-fETI**

-fextend\_to\_int/-fETI  
char 型データを int 型に拡張

**-fchar\_enumerator/-fCE**

-fchar\_enumerator/-fCE  
enumerator(列挙子)の型を unsigned char 型で処理

**-fno\_even/-fNE**

-fno\_even/-fNE  
データをすべて odd 属性のセクションに配置

**-fall\_far/-fAF**

-fall\_far/-fAF  
デフォルトをすべて far 型に設定

**-fnear\_function/fNF**

-fnear\_function/fNF  
関数のデフォルトを near に設定

**-ffar\_program\_section/-fFPS**

-ffar\_program\_section/-fFAA  
near/far 関数を program\_F セクションに配置

**-fnot\_use\_MVN/-fNUM**

-fnot\_use\_MVN/-fNUM  
MVN 命令でのブロック転送を抑制

**-fshow\_stack\_usage/-fSSU**

-fshow\_stack\_usage/-fSSU  
スタック使用状況をファイル(拡張子 .stk)に出力

**-ffar\_RAM\_data/-fFRAM**

-ffar\_RAM\_data/-fFRAM  
RAM データのデフォルト属性を far にする

**-ffar\_ROM\_data/-fFROM**

-ffar\_ROM\_data/-fNROM  
ROM データのデフォルト属性を far にする

**-fconst\_not\_rom/-fCNR**

-fconst\_not\_ROM/-fCNR  
const で指定した型を ROM データとして扱わない

**-fnot\_address\_volatile/-fNAV**

-fnot\_address\_volatile/-fNAV  
#pragma ADDRESS(#pragma EQU)で指定した変数をvolatileとしない

**-fsmall\_array/-fSA**

-fsmall\_array/-fSA  
far 型の配列の添字を 16 ビットとする

**-fenable\_register/-fER**

-fenable\_register/-fER  
レジスタ記憶クラスを有効にする

**-fswitch\_table/-fST**

-fswitch\_table/-fST  
switch 文に対してテーブルジャンプ方式コードを生成

**-fuse\_DIV/-fUD**

-fuse\_DIV/-fUD  
オーバーフローを考慮せずに除算を行います。

**-bank**

-bank=1  
データバンクレジスタ値を指定

**警告オプション**

**-Wnon\_prototype/-WNP**

-Wnon\_prototype/-WNP  
プロトタイプ宣言されていない関数にたいして警告出力

## 付録 A NC77WA コマンドオプション一覧

### -Wunknown\_pragma/-WUP

-Wunknown\_pragma/-WUP  
サポートしていない #pragma にたいして警告出力

### -Wno\_stop/-WNS

-Wno\_stop/-WNS  
エラーが発生してもコンパイル作業を継続

### -Wstdout

-Wstdout  
エラーメッセージを標準出力 ( stdout ) に出力

### -Werror\_file/-WEF

-Werror\_file/-WEF  
タグファイルを出力

### -Wstop\_at\_warning/-WSAW

-Wstop\_at\_warning/-WSAW  
ワーニング発生時にコンパイル処理を停止

### -Wnesting\_comment/-WNC

-Wnesting\_comment/-WNC  
コメント中の /\* にたいして警告を出力

### -Wccom\_max\_warnings/-WCMW

-Wccom\_max\_warnings/WCMW  
ccom77 のワーニング出力数の上限を指定

### -Wall

-Wall  
検出可能な警告を全て表示

### -Wmake\_tagfile/-WMT

-Wmake\_tagfile/-WMT  
ファイル単位でエラーメッセージを出力します。

### -Wlarge\_to\_small/-WLTS

-Wlarge\_to\_small/-WLTS  
大きいサイズから小さいサイズへの暗黙の転送に対して警告を出します。

### -Wuninitialize\_variable/-WUV

-Wuninitialize\_variable/-WUV  
初期化されていない auto 変数に対して警告を出します。

## その他のオプション

### -dsource/-dS

-dsource/-dS  
アセンブリ言語ソースリストにC言語ソースをコメントとして出力

## 7750/7751 対応コード生成オプション

### -m7750

-m7750  
7750/7751 シリーズ対応のコード出力

## stk77 コマンドオプション

### -e

-efunctionname  
スタック使用量の計算を開始する関数名を指定

### -S

-sSYMfile  
シンボルファイルを指定

### -O

-o  
スタックサイズと関数の呼び出し関数の表示をファイル( 拡張子 .siz ) に出力

### -C

-c  
関数の呼び出し関係を標準出力に表示

### -I

-Ifilename  
ライブラリ関数用のスタック使用量表示ファイルを指定

## s2ie コマンドオプション

- .
- .
- 画面へのメッセージ出力を停止
- NLS**
- NLS
- アセンブラローカルシンボルの出力を抑制
- V**
- V
- バージョン番号を表示し終了
- O**
- osample
- 出力ファイル名を指定
- Wstdout**
- Wstdout
- エラー、ワーニングメッセージを標準出力に表示

## RASM77 コマンドオプション

- .
- .
- 画面へのメッセージ出力を停止
- A**
- A
- 条件アセンブル偽部分のエラー検出停止
- B**
- B
- ビット長の整合性を評価
- C**
- C
- デバッグ情報をオブジェクトファイルに出力

- D**
- Dsymbol=0
- シンボルを定義
- E**
- Evi
- タグファイルを生成しエディタを起動
- F**
- F
- M\_FLAG、X\_FLAG を有効にする
- L[C:D;D0;D1]**
- L
- PRN ファイルを生成
- LC
- 条件アセンブルの条件偽の部分、@および.DEFINEの変換文字列をPRNファイルに出力
- LD/-LD0
- .DEFINE および@の置換結果のみをPRNファイルに出力
- LD1
- .DEFINEおよび@の記述行と置換結果をPRNファイルに出力
- M**
- M
- マクロ展開部をリストファイルに出力
- N**
- N
- リストファイルへのシンボルリスト出力を抑制
- O**
- OC:\work
- ファイルの生成ディレクトリを指定
- P**
- P
- プリントファイルのマクロ展開部分の行番号を非更新
- Q**
- Q
- .EQUの再設定時にワーニング出力

## 付録 A NC77WA コマンドオプション一覧

### -S

-S  
ローカルシンボル情報をオブジェクトファイルに出力

### -T

-T  
マクロ呼び出し行情報をもつエラータグファイルを生成

### -U

-U  
ラベル直後のコロン(:)の省略を許可

### -V

-V  
RASM77 のバージョン番号を表示

### -X

-X  
アセンブル終了後 CRF77 を起動

## PRE77 コマンドオプション

### -.

-.  
画面へのメッセージ出力を停止

### -C

-C  
構造化記述命令のソースでバグ情報を出力

### -E

-E  
タグファイルを生成

### -L

-L  
構造化記述命令にたいする分岐先番地をラベルで出力

### -O

-O:\work  
生成ファイルの出力ディレクトリを指定

### -RASM77

-RASM77  
PRE77 終了後 RASM77 を起動

### -S

-S  
構造化記述部分をコメントとしてリストファイルに出力

### -V

-V  
PRE77 のバージョン番号を画面出力して終了

## LINK77 コマンドオプション

### -A

-A  
同名の絶対属性セクションのオーバーラップを許可

### -C

-C  
特定の分岐命令がバンク境界にある場合ワーニングを出力

### -F

-Ftest  
出力ファイル名を指定

### -M

-M  
マップファイルを生成

### -MS

-MS  
シンボル情報を含むマップファイルを生成

### -N

-N  
.OBJ、.LIB の指定を無視

### -O

-O:\work  
生成ファイルの出力ディレクトリを指定

**-S**

-S  
シンボルファイルを生成

**-V**

-V  
リロケータブルファイルのバージョン情報を確認

**-W**

-W  
セクションの配置をワードアライメントで実行

### LIB77 コマンドオプション

**-O**

-Oapl\_lib.lib  
ライブラリファイル名を指定

**-A**

-A  
ライブラリファイルにリロケータブルファイルを追加

**-D**

-D  
ライブラリファイル内のリロケータブルファイルを削除

**-F**

-Fmain.r77  
リロケータブルファイル名を指定

**-L**

-L  
ライブラリリストファイルを生成

**-R**

-R  
ライブラリファイルのリロケータブルファイルを更新

**-U**

-U  
ライブラリファイルのリロケータブルファイルより新しいファイルのみ更新

**-V**

-V  
処理中のファイル名を表示

**-X**

-X  
ライブラリファイル内のリロケータブルモジュールファイルを抽出

### CRF77 コマンドオプション

**-O**

-OC:\work  
ファイルの生成ディレクトリを指定

**-I**

-I  
.INCLUDE を無視

### BR77 コマンドオプション

**-.**

-.  
画面へのメッセージ出力を抑止

**-OB1**

-OB1  
分岐コードを bra -> bral -> jmp の順で変換

**-OB2**

-OB2  
分岐コードを bra -> jmp の順で変換

**-OB3**

-OB3  
分岐コードを bra -> jmp の順で変換

**-V**

-V  
バージョン番号を表示し終了





## 付録 B NC77 拡張機能一覧

### near・far 修飾子

```
int far i;          ...1
int far func();    ...2
```

1. データをアクセスするアドレッシングモードを指定  
near... 同一バンク (64K バイト以内) のアクセス  
far... バンク外 (64K バイトを越える領域の) のアクセス
2. 関数の呼び出しに使用する命令を指定  
near...JSR 命令で呼び出し  
far...JSRL 命令で呼び出し

### asm 関数

```
asm(" LDA A,DP:1");    ...1
asm(0,0); /* CLP m,x */ ...2
asm(" LDA A,DP:$$,i"); ...3
asm();                 ...4
asm(" LDA A,$@,i");    ...5
```

1. C 言語プログラム中にアセンブリ言語を直接記述 (関数外でも記述可能)
2. プロセッサステータスレジスタ中の m、x フラグの切り替えを指示
3. 変数名を指定可能
4. 最適化を部分的に抑止するダミーの asm 関数が記述可能
5. auto 変数名および外部変数名を指定可能。DT:,DP:はコンパイルが生成します。

### 関数のデフォルト引数宣言

```
extern int func(int I=1, int j=2);
```

関数の引数にデフォルト値を定義

関数を宣言するよりも前にデフォルト値として使用する変数の宣言を行ってください。

デフォルト値は引数の後ろから順に埋めてください。

### inline 記憶クラスのサポート

```
inline int func(int);
```

inline記憶クラス指定子により関数をインライン展開することが可能

インライン関数を使用する前に必ずインライン関数の実体定義を行ってください。

### 日本語文字のサポート

```
L" 文字列 "; ...1
L' 文 '; ...2
/* 漢字 */ ...3
```

1. 文字列中に日本語文字を使用可能
2. 日本語文字の文字定数を使用可能
3. コメント中に日本語文字を記述可能  
シフト JIS コード及び EUC コードをサポートしています。

### コメント記述の拡張

```
// comment
```

C++ 言語ライクなコメント "//" を記述可能

### #pragma SECTION

```
#pragma SECTION bss nonval_data
NC77 が生成するセクション名を変更
```

### #pragma ROM

```
#pragma ROM val
指定した変数を rom セクションに配置
```

### #pragma STRUCT

```
#pragma STRUCT TAG1 unpack ...1
#pragma STRUCT TAG1 arrange ...2
```

1. 指定したタグを持つ構造体のパックを禁止
2. 指定したタグを持つ構造体のメンバを並べ替え、偶数サイズのメンバを先に配置

### #pragma INTERRUPT [#pragma INTF]

```
#pragma INTERRUPT int_func
C 言語で記述した割り込み処理関数を宣言
```

### #pragma ADDRESS[#pragma EQU]

```
#pragma ADDRESS port0 2H
```

変数の絶対アドレスを指定

near 指定された変数はバンク内アドレスを示します。

## 付録 B NC77 拡張機能一覧

### #pragma PARAMETER

```
#pragma PARAMETER asm_func(R0,R1)
```

アセンブラ関数を呼び出す際に、その引数をレジスタを介して渡すことを宣言  
本宣言を行う前に、必ず関数のプロトタイプ宣言を行ってください。

### #pragma INTHANDLER[#pragma HANDLER]

```
#pragma INTHANDLER int_func  
#pragma HANDLER int_func
```

MR7700 の割り込みハンドラ関数名を宣言

### #pragma ALMHANDLER

```
#pragma ALHANDLER alm_func
```

MR7700 のアラームハンドラ関数名を宣言

### #pragma CYCHANDLER

```
#pragma CYCHANDLER cyc_func
```

MR7700 の周期起動ハンドラ関数名を宣言

### #pragma TASK

```
#pragma TASK task1
```

MR7700 のタスクの開始関数名を宣言

### #pragma LOADDT

```
#pragma LOADDT func
```

コンパイル時のデータバンクレジスタ(DT)の値を関数の先頭でロードする関数名を指定

### #pragma M1FUNCTION

```
#pragma M1FUNCTION
```

M フラグが1の状態のまま呼び出す関数を指定

### #pragma ASM - #pragma ENDASM

```
#pragma ASM  
    mov.w R0, R1  
    add.w R1, 02H  
#pragma ENDASM
```

アセンブリ言語で記述を行う領域を指定

### #pragma PAGE

```
#pragma PAGE
```

アセンブラリスティングファイルの改ページの指定を行います。

## 付録 C NC77 標準関数一覧

### 文字列操作関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <string.h>
```

#### strcpy

```
char_far * _far strcpy(s1,s2);
```

文字列の複写

#### strncpy

```
char_far * _far strncpy(s1,s2,n);
```

文字列の複写(n文字の複写)

#### strcat

```
char_far * _far strcat(s1,s2);
```

文字列の連結

#### strncat

```
char_far * _far strncat(s1,s2,n);
```

文字列の連結(n文字の連結)

#### strcmp

```
int_far strcmp(s1,a2);
```

文字列の比較

#### strcoll

```
int_far strcoll(s1,s2);
```

文字列の比較(ロケール情報を使用)

#### stricmp

```
int_far stricmp(s1,s2);
```

文字列の比較(すべての英字は英大文字として扱う)

#### strncmp

```
int_far strncmp(s1,s2,n);
```

文字列の比較(n文字の比較)

#### strnicmp

```
int_far strnicmp(s1,s2,n);
```

文字列(n文字)の比較(英字は英大文字として扱う)

#### strchr

```
char_far * _far strchr(s,c);
```

文字列の先頭より指定文字を検索

#### strcspn

```
size_t_far strcspn(s1,s2);
```

文字列より指定以外の文字列の長さを計算

#### strpbrk

```
char_far * _far strpbrk(s1,s2);
```

文字列より指定文字の検索

#### strrchr

```
char_far * _far strrchr(s,c);
```

文字列の末尾より指定文字を検索

#### strspn

```
size_t_far strspn(s1,s2);
```

文字列より指定文字列の長さを計算

#### strstr

```
char_far * _far strstr(s1,s2);
```

文字列より指定文字列の検索

#### strtok

```
char_far * _far strtok(s1,s2);
```

文字列より文字列を切り出す

#### strlen

```
size_t_far strlen(s);
```

文字列中の文字数を計算

## 付録 C NC77 標準関数一覧

### strerror

```
char *_far strerror(errcode);
```

エラー番号を文字列に変換

### strxfrm

```
size_t _far strxfrm(s1,s2,n);
```

文字列を変換(ロケール情報を使用)

### 文字判定関数

これらの関数を使用するには次の記述が必要です。

```
#include <ctype.h>
```

### isalnum

```
int isalnum(c);
```

英数字の判定

### isalpha

```
int isalph(c);
```

英字の判定

### isctrl

```
int isctrl(c);
```

コントロール文字の判定

### isdigit

```
int isdigit(c);
```

数字の判定

### isgraph

```
int isgraph(c);
```

英数字、空白以外の文字判定

### islower

```
int islower(c);
```

英小文字の判定

### isprint

```
int isprint(c);
```

空白文字を含む印字可能文字の判定

### ispunct

```
int ispunct(c);
```

区切り文字の判定

### isspace

```
int isspace(c);
```

空白、タブ、改行の判定

### isupper

```
int isupper(c);
```

英大文字の判定

### isxdigit

```
int isxdigit(c);
```

16進数字の判定

### tolower

```
int tolower(c);
```

大文字から小文字への変換

### toupper

```
int toupper(c);
```

小文字から大文字への変換

### 入出力関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdio.h>
```

### init

```
void _far init(void);
```

7700 ファミリの入出力の初期化

### clearerror

```
void _far clearerr(stream);
```

エラー状態指示子を初期化(クリア)

### fgetc

```
int _far fgetc(stream);
```

一文字入力

### getc

```
int _far getc(stream);
```

一文字入力

### getchar

```
int _far getchar(void);
```

stdin からの一文字入力

**fgets**

```
char *_far fgets(buffer, n, stream);
```

一行入力

**gets**

```
char *_far gets(buffer);
```

stdin からの一行入力

**fread**

```
size_t_far fread(buffer, size, count, stream);
```

指定データ数入力

**scanf**

```
int_far scanf(format, argument...);
```

stdin からの書式付き入力

**fscanf**

```
int_far fscanf(stream, format, argument...);
```

書式付き入力

**sscanf**

```
int_far sscanf(string, format, argument...);
```

文字からの書式付きデータ入力

**fputc**

```
int_far fputc(c, stream);
```

一文字出力

**putc**

```
int_far putc(c, stream);
```

一文字出力

**putchar**

```
int_far putchar(c);
```

stdout への一文字出力

**fputs**

```
int_far fputs(str, stream);
```

一行出力

**puts**

```
int_far puts(str);
```

stdout への一行出力

**fwrite**

```
size_t_far fwrite(buffer, size, count, stream);
```

指定データ数出力

**perror**

```
void_far perror(s);
```

stdout へのエラーメッセージ出力

**printf**

```
int_far printf(format, argument);
```

stdout への書式付き出力

**fflush**

```
int_far fflush(stream);
```

出力バッファのストリームをフラッシュ

**fprintf**

```
int_far fprintf(stream, format, argument);
```

書式付き出力

**sprintf**

```
int_far sprintf(pointer, format, argument);
```

書式付き文字列設定

**vfprintf**

```
int_far vfprintf(stream, format, ap);
```

ストリームへの書式付き出力

**vprintf**

```
int_far vprintf(format, ap);
```

stdout への書式付き出力

**vsprintf**

```
int_far vsprintf(s, format, ap);
```

バッファへの書式付き出力

**ungetc**

```
int_far ungetc(c, stream);
```

一文字入力の返還

**ferror**

```
int_far ferror(stream);
```

入出力エラーの判定

## feof

```
int_far feof(strem);
```

EOF(End Of File)の判定

### メモリ管理関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdlib.h>
```

## calloc

```
void_far * _far calloc(n,size);
```

メモリの確保と0(ゼロ)による初期化

## free

```
void_far free(cp);
```

メモリの解放

## malloc

```
void_far * _far malloc(nbytes);
```

メモリの確保

## realloc

```
void_far * _far realloc(cp,nbytes);
```

確保済み領域の大きさを変更

### メモリ操作関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <string.h>
```

## bzero

```
void_far bzero(top,size);
```

メモリ領域の初期化(ゼロクリア)

## bcopy

```
void_far bcopy(src,dtop,size);
```

メモリ領域の複写

## memcpy

```
void_far * _far memcpy(s1,s2,n);
```

メモリ領域の複写(n文字の複写)

## memset

```
char_far* _far memset(s,c,n);
```

メモリ領域の設定

## memcmp

```
int_far memcmp(s1,s2,n);
```

メモリ量域の比較(nバイトの比較)

## memicmp

```
int_far memicmp(st,s2,n);
```

メモリ領域の比較(英文字は大文字として扱う)

## memmove

```
void_far * _far memmove(s1,s2,n);
```

文字列の領域を移動

## memchr

```
void_far * _far memchr(s,c,n);
```

メモリ領域より文字を検索

### 実行制御関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdlib.h>
```

```
#include <setjmp.h>
```

## abort

```
void_far abort(void);
```

プログラムの実行を終了

## longjmp

```
void_far longjmp(env,val);
```

大域ジャンプ

## setjmp

```
int_far setjmp(env);
```

大域ジャンプのためのスタック環境の設定

### 数学関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <math.h>
```

## acos

```
double_far acos(x);
```

逆コサインを計算

**asin**

```
double_far asin(x);
```

逆サインを計算

**atan**

```
double_far atan(x);
```

逆タンジェントを計算

**atan2**

```
doubel_far atan2(x,y);
```

逆タンジェントを計算

**ceil**

```
double_far ceil(x);
```

整数繰り上げ値を計算

**cos**

```
double_far cos(x);
```

コサインを計算

**cosh**

```
double_far cosh(x);
```

双曲線コサインを計算

**exp**

```
double_far exp(x);
```

指数関数を計算

**fabs**

```
double_far fabs(x);
```

倍精度浮動小数の絶対値を計算

**floor**

```
double_far floor(x);
```

整数繰り下げ値を計算

**fmod**

```
double_far fmod(x,y);
```

剰余計算

**frexp**

```
double_far frexp(x,exp);
```

浮動小数を仮数部と指数部に分割

**labs**

```
long_far labs(n);
```

long 型整数の絶対値を計算

**ldexp**

```
double_far ldexp(x,exp);
```

浮動小数の巾を計算

**log**

```
double_far log(x);
```

自然対数を計算

**log10**

```
double_far log10(x);
```

常用対数を計算

**modf**

```
double_far modf(val,pd);
```

実数を仮数部と指数部に分割

**pow**

```
double_far pow(x,y);
```

巾乗計算

**sin**

```
double_far sin(x);
```

サインを計算

**sinh**

```
double_far sinh(x);
```

双曲線サインを計算

**sqrt**

```
double_far sqrt(x);
```

数値の平方根を計算

**tan**

```
double_far tan(x);
```

タンジェントを計算

**tanh**

```
double_far tanh(x);
```

双曲線タンジェントを計算



## 付録 C NC77 標準関数一覧

### 整数算術関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdlib.h>
```

#### abs

```
int_far abs(n);
```

整数の絶対値を計算

#### bsearch

```
void_far bsearch(key,base,nlem,cmp);
```

配列内のバイナリサーチを行う

#### div

```
dif_t_far div(number,denom);
```

int 型整数の除算と剰余

#### labs

```
long_far labs(n);
```

long 型整数の絶対値を計算

#### ldiv

```
ldiv_t_far ldiv(number,denom);
```

long 型整数の除算と剰余

#### qsort

```
void_far qsort(base,nelen,size,cmp(e1,e2));
```

配列をソート

#### rand

```
int_far rand(void);
```

疑似乱数を発生

#### srand

```
void_far srand(seed);
```

疑似乱数にシードを与える

### 文字列数値変換関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdlib.h>
```

#### atof

```
double_far atof(s);
```

文字列を double 型に変換

#### atoi

```
int_far atoi(s);
```

文字列を int 型に変換

#### atol

```
long_far atol(s);
```

文字列を long 型に変換

#### strtod

```
double_far strtod(s,endprt);
```

文字列を double 型に変換

#### strtol

```
long_far strtol(s,endptr,base);
```

文字列を long 型に変換

#### strtoul

```
unsignedlong_far strtoul(s,endptr,base);
```

文字列を unsigned long 型に変換

### 多バイト文字・多バイト文字列操作関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <stdlib.h>
```

#### mblen

```
int_far mblen(s,n);
```

マルチバイト文字列の長さを計算

#### mbstowcs

```
size_t_far mbstowcs(wcs,s,n);
```

マルチバイト文字列をワイド文字列に変換

#### mbtowc

```
int_far mbtowc(wcs,s,n);
```

マルチバイト文字をワイド文字に変換

#### wcstombs

```
size_t_far wcstombs(s,wcs,n);
```

ワイド文字列をマルチバイト文字列に変換

#### wctomb

```
int_far wctomb(s,wchar);
```

ワイド文字をマルチバイト文字に変換

## 地域化関数

これらの関数を使用する場合は次の記述が必要です。

```
#include <local.h>
```

### localeconv

```
struct lconv_far *localeconv(void);
```

構造体 lconv を初期化

### setlocale

```
char_far *setlocale(category, locale);
```

プログラムのロケール情報の設定と検索



## 付録 D RASM77 アセンブラ疑似命令一覧

### RASM77 が処理する疑似命令

#### .ADDR

```
[str:].ADDR    exp[,exp,...]
```

3バイト長のデータ領域をROMに確保し値を格納

#### .ASSERT

```
.ASSERT    "str"
```

オペランドの文字列を標準エラー出力に表示

#### .BLKA

```
[str:].BLKA    exp
```

3バイト(24ビット)単位でRAMに領域を確保

#### .BLKB

```
[str:].BLKB    exp
```

1バイト(8ビット)単位でRAMに領域を確保

#### .BLKD

```
[str:].BLKD    exp
```

8バイト(64ビット)単位でRAMに領域を確保

#### .BLKW

```
[str:].BLKW    exp
```

2バイト(16ビット)単位でRAMに領域を確保

#### .BYTE

```
str: .BYTE exp[,exp,...]
```

1バイト長のデータ領域をROMに確保し値を格納

#### .COL

```
.COL 120
```

リストファイルの桁数を指定

#### .DATA [8:16]

```
.DATA 8
```

SEM  
データ長を宣言(本命令以降のデータを宣言したデータ長で処理する)  
本命令でデータ長を宣言する場合は、必ずCLMまたはSEMでCPU内のフラグも操作してください。

#### .DEFINE

```
FLAG1 .DEFINE    "#01,DATA1"
```

シンボルに文字列を定義

#### .DP

```
.DP 1000H
```

```
LDA 1000H
```

STD  
ダイレクトページレジスタ値(0000H ~ 0FFFFH)を宣言

#### .DPEXT

```
.DPEXT    WORK1,WORK2
```

ラベルをダイレクトページアドレッシングで外部参照することを指定

#### .DT

```
.DT 01H
```

データバンクレジスタ値宣言

#### .DTEXT

```
.DTEXT    WORK1,WORK2
```

ラベルをアブソリュートアドレッシングで外部参照することを指定

#### .DWORD

```
DWMMEM:    .DWORD    0E1000H
```

4バイト長のデータ領域をROMに確保し値を格納

## 付録 D RASM77 アセンブラ疑似命令一覧

### .END

.END  
アセンブルソースの終了を宣言

### .EQU

str .EQU exp  
数値シンボルに値を定義

### .ERROR

.ERROR 'Error occurs'  
オペランドの文字列を標準出力に表示

### .EVEN

.EVEN  
アドレスを偶数番地に補正

### .EXT

.EXT WORK1,WORK2  
ラベルを外部参照指定

### .IF (- .ELSE -) .ENDIF

```
.IF CDATA <= 0
    LDA WORK1
.ELSE
    LDA WORK2
.ENDIF
```

条件付きアセンブル (.ELSE とそれに続く命令は省略可能)

### .INCLUDE

.INCLUDE filename  
指定したファイルの記述内容を読み込む

### .INDEX

.INDEX 8  
SEP X  
インデックスレジスタ長を宣言

### .LIB

.LIB LIB1,LIB2,LIB3  
リンク時に参照するライブラリファイル名を指定

### .LINE

.LINE 60  
リストファイルの1ページの行数 (5 ~ 255) を指定

### .LIST

.LIST  
リストファイルへの行の出力を開始

### .LISTM

.LISTM  
リストファイルへのマクロ展開部分の出力開始

### .MCU [M37750|M37751]

.MCU M37750  
出力する命令コードのCPUタイプを指定

### .NLIST

.NLIST  
リストファイルへの出力停止

### .NLISTM

.NLISTM  
マクロ展開部のリストファイルへの出力停止

### .OBJ

.OBJ OBJ1,OBJ2,OBJ3  
リンク対象となるリロケータブルファイル名を指定

### .ORG

.ORG 0C000H  
この行以降のアドレスを宣言

### .PAGE

.PAGE 'Start main routine'  
リストファイルを改ページし、指定した文字列をリストファイルに出力

### .PUB

.PUB WORK1,WORK2,WORK3  
ラベルのパブリック指定 (外部ファイルからの参照を可能にする)

### .SECTION

.SECTION DATA  
セクション名を宣言

## .VER

```
.VER 'str'
```

指定した文字列をマップファイルに出力します。

## .WORD

```
str: .WORD 0E00H
```

2バイト長のデータ領域をROMに確保し値を格納

## マクロ命令

### .LOCAL

```
.LOCAL LOOP1, LOOP2
```

マクロローカルラベルを宣言(マクロ定義内でのみ使用可能)

### .MACRO - (.EXITM -) .ENDM

```
DATA1: .MACRO VAL
        .IF LEVEL
        .BYTE VAL
        .EXITM
        .ENDIF
        .WORD VAL
        .ENDM
```

マクロ定義命令(.EXITMはマクロ展開を中止し.ENDMに分岐する)

### .REPEAT - .ENDM

```
.REPEAT 5
        NOP
.ENDM
```

.REPEATと.ENDMに囲まれた部分を指定回数だけ展開

### .REPEATC - .ENDM

```
DATA: .REPEATC VAL, ABC
        .BYTE 'VAL'
```

.ENDM

実引数の文字数回だけ展開

実引数は文字列ABCを示します。VALは仮引数と表しません。

### .REPEATI - .ENDM

```
.REPEATI VAL, 0, 1, 2
        .BYTE 'VAL'
```

.ENDM

実引数の回数回だけ展開

実引数は文字列ABCを示します。VALは仮引数と表しません。

## デバッグ用疑似命令

これらの疑似命令はソースラインでバグのためにPRE77が生成するものです。ユーザーは使用しないでください。

### .CLINE

```
.CLINE 10
```

オペランドにはソース行番号が出力される

### .FUNC - .ENDFUNC

```
.FUNC SUB
SUB: :
        .ENDFUNC
```

関数(C言語)またはサブルーチンの開始と終了を示す

### .LANGUAGE

```
.LANGUAGE C
```

ソースプログラムの記述に使用された言語を示す

### .POINTER

```
.POINTER 2
```

Cコンパイラで使用されたポインタ変数のバイト長を示す

### .SOURCE

```
.SOURCE PROG1.C
```

ソースファイル名を示す

## 予約疑似命令

次の疑似命令は将来の拡張のためにアセンブラが予約しているものです。これらの疑似命令を使用する場合は将来特定の機能が割り付けられる可能性がありますのでご注意ください。

### .IO - .ENDIO

I/O領域を示す

### .PROCINT - .ENDPROC

割り込み処理プログラムの範囲を示す

### .PROCMAIN - .ENDPROC

メインプログラムの範囲を示す

### .PROCSUB - .ENDPROC

サブプログラムの範囲を示す

## 付録 D RASM77 アセンブラ疑似命令一覧

### .PROGNAME

プログラム名を宣言

### .RAM - .ENDRAM

RAM 領域を宣言

### PRE77 が処理する疑似命令

### .DATA

.DATA 8

データ長を宣言(本命令以降のデータを宣言したデータ長で処理する)

本命令でデータ長を宣言する場合は、必ずCLMまたはSEMでCPU内のフラグも操作してください。

### .END

.END

ソースプログラムの終了を宣言

### .EQU

SYM1: .EQU 1

FLAG1: .EQU 0,100H

シンボル(ビットシンボル)に値を定義

### .INCLUDE

.INCLUDE MEMDEF.P77

指定したファイルの記述内容を読み込む

### .INDEX

.INDEX 8

インデックスレジスタ長を宣言

### .SECTION

.SECTION STRPRG

セクション名を宣言

### 構造化記述命令

### = (代入文)

[work] = 0

C = 0

右辺を左辺へ代入

```
if - else - endif [ lif-lelse-endif | llif - lelse - endif ]
```

```
if [flag]
    [work] = 0
```

```
else
    [work] = 1
```

```
endif
```

条件を判断し分岐処理を制御

```
for - next [ lfor - next | lfor - next ]
```

```
for [flag]
```

```
    JSR output
```

```
next
```

条件を判断し繰り返し処理を制御

```
do - while [ ldo - while | ldo - while ]
```

```
do
```

```
    JSR output
```

```
while [flag]
```

条件を判断し繰り返し処理を制御

```
switch - case - ends [ lswitch - case - ends |
llswitch - case - ends ]
```

```
switch [work]
```

```
    case 1
```

```
        JSR output1
```

```
        break
```

```
    case 2
```

```
        JSR output2
```

```
ends
```

条件式の値によっていずれかの文(CASE)に分岐

```
break [ lbreak | llbreak ]
```

```
break
```

for、do、switchの実行を中止し各ブロックの次に分岐

```
continue [ lcontinue | llcontinue]
```

```
continue
```

for、doブロック内から繰り返しの条件を判断するところへ分岐

```
goto [ lgoto | llgoto ]
```

```
goto LABEL
```

任意のアドレスに無条件で分岐

# 技術サポート連絡書

年 月 日 (合計 枚)

三菱電機セミコンダクタシステム株式会社  
マイコンソフトツール部

## 開発ツールサポート窓口行

[ 電子メール ] support@tool.mesc.co.jp

[ 大阪地区 ] FAX : 06-6398-6191

[ 東京地区 ] FAX : 03-5783-7339

[ 中部地区 ] FAX : 052-221-7318

[ 九州地区 ] FAX : 092-452-1427

インストーラが生成する以下のテキストファイルもサポート連絡書としてご利用できます。  
Windows 98/95/Windows NT 4.0版の場合 : ¥SUPPORT¥製品名¥SUPPORT.TXT  
EWS版の場合 : /support/製品名/toolinfo.txt

| ご連絡先    | 製品情報         |
|---------|--------------|
| 会社名 :   | ソフトウェア :     |
| 部署名 :   | バージョン番号 : V. |
| 担当者名 :  | ライセンスID :    |
| 電話番号 :  | - - - -      |
| FAX番号 : | ホストマシン :     |
| 電子メール : | OS : V.      |
| 通信欄 :   |              |



# MEMO

## **NC77WA/RASM77ガイドブック**

---

第1版：2000年7月1日発行

資料番号：MSD-NC77WA-UG-000701

**Copyright ©2000 Mitsubishi Electric Corporation**

**Copyright ©2000 Mitsubishi Electric Semiconductor Systems Corporation**

**All rights reserved.**

三菱電機株式会社

三菱電機セミコンダクタシステム株式会社

77XX シリーズ用 C コンパイラ / アセンブラ  
NC77WA / RASM77 ガイドブック



ルネサスエレクトロニクス株式会社  
神奈川県川崎市中原区下沼部1753 〒211-8668