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い申し上げます。

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ものですが、誤りがな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て、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が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。

ユーザーズ・マニュアル

RENESAS

保守／廃止

CC78Kシリーズ

Cコンパイラ

操作編

資料番号　EEU-656C（第4版）
発行年月　May 1997　N

ユーザーズ・マニュアル

NEC

保守／廃止

CC78Kシリーズ

Cコンパイラ

操作編

保守／廃止

○文書による当社の承諾なしに本資料の転載複製を禁じます。
○本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的所有権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。
○当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。
○当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「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の上ご使用願います。

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

特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

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，生命維持のための装置またはシステム等

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。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。
○この製品は耐放射線設計をしておりません。

M4 94.11

本製品は外国為替および外国貿易管理法の規定により戦略物資等（または役務）に該当しますので、日本国外に輸出する場合には、同法に基づき日本国政府の輸出許可が必要です。

○文書により当社の承諾なしに本資料の転載複製を禁じます。
○この製品を使用したことにより、第三者の工業所有権等にかかわる問題が発生した場合，当社製品の構造製法に直接かかわるもの以外につきましては、当社はその責を負いませんのでご了承ください。

保守／廃止

はじめに

本マニュアルは，ＣＣ７８Ｋシリーズ　Ｃコンパイラについての機能および操作方法を正しく理解していただくことを目的として書かれています。

本マニュアルでは，ＣＣ７８Ｋシリーズのソース・プログラムの記述方法に関する説明はいたしません。したがって，本マニュアルをお読みになる前に”ＣＣ７８Ｋシリーズ　Ｃコンパイラ　ユーザーズ・マニュアル　言語編”（以降”言語編”とします）をお読みください。

【ターゲット・デバイス】

本Ｃコンパイラでは，次のマイクロコンピュータのソフトウエア開発が可能です。

シリーズ名	ターゲット・デバイス
７８Ｋ／０	μPD78001，78002 μPD78011，78012，78013，78014，78P014 μPD78022，78023，78024，78P024 μPD78042，78043，78044，78P044
７８Ｋ／Ⅱ	μPD78210注 μPD78212，78213，78214，78P214 μPD78217A，78218A，78P218A μPD78220，78224，78P224 μPD78233，78234，78237，78238，78P238 μPD78243，78244
７８Ｋ／Ⅲ	μPD78310注，78312注，78P312注 μPD78310A，78312A，78P312A μPD78320，78322，78P322，78323，78324，78P324 μPD78327，78328，78P328 μPD78330，78334，78P334 μPD78350，78P352

注　保守品です。

注意　上記の製品のうち，一部，開発中のものがあります。

【対　象　者】

本マニュアルは，デバイスのユーザーズ・マニュアル一読程度の知識がありソフトウエア・プログラミングの経験がある方を対象として書かれていますが，Ｃコンパイラ やＣ言語の知識は特に必要ありませんので，Ｃコンパイラをはじめて使われる方でもお読みいただけます。

ただし，本パッケージはコンパイラの単体であるため，実際に本コンパイラをご使用になる場合には，“ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ”が必要になります。

【構　　成】

本マニュアルの構成を次に示します。

第１章　概　　説

マイクロコンピュータの開発における本Ｃコンパイラの役割り，位置付けなど，本コンパイラ全体の機能概要を説明します。

また，本Ｃコンパイラの特徴についても紹介します。

第２章　製品概要

本Ｃコンパイラが提供するプログラムのファイル名，プログラムの動作環境などについて説明します。

第３章　Ｃコンパイラの実行

サンプル・プログラムを使用して，実際に本Ｃコンパイラを実行する手順を説明します。

第４章　Ｃコンパイラ

本Ｃコンパイラの入出力ファイルや起動方法について詳細に説明します。

第５章　コンパイラ・オプション

コンパイラ・オプションの指定方法，優先順位などについて説明します。また，おのおののコンパイラ・オプションごとに使用例を使って詳細に説明します。

第６章　Ｃコンパイラの出力ファイル

本Ｃコンパイラが出力する各種リスト・ファイルの出力項目をフォーマットを用いて説明します。

第７章　Ｃコンパイラの活用法

本コンパイラをうまく使うための手段を紹介します。

第８章　スタートアップ・ルーチン，エラー処理ルーチン

サンプル・プログラムを例として，スタートアップ・ルーチンやエラー処理ルーチンの内容，使い方およびサンプル・プログラム改良のポイントなどについて説明します。

第９章　エラー・メッセージ

本コンパイラが出力するエラー・メッセージについて説明します。

付　録

付録として，サンプル・プログラムの実行例，使用上の注意点一覧，オプション一覧があります。

【読　み　方】

まず，実際に本Ｃコンパイラを使ってみたい方は“第３章　Ｃコンパイラの実行”をお読みください。

Ｃコンパイラの一般的な知識のある方や“言語編”を読まれた方であれば“第１章　概説”を読み飛ばされても結構です。

Ｃコンパイラの操作になれた方は，付録の一覧表をご活用ください。また，“第５章　コンパイラ・オプション”，“第９章　エラー・メッセージ”なども容易に索引できるようになっています。

【凡　　例】

本マニュアル中で共通に使用される記号などの意味を示します。

…　：同一の形式を繰り返す

［　］：［　］内は省略可能

「　」：「　」で囲まれた文字そのもの

“　”：“　”で囲まれた文字そのもの

‘　’：‘　’で囲まれた文字そのもの

太文字：文字そのもの

＿　：重要箇所，使用例での下線は入力文字列

△　：１文字以上の空白

⋮　：プログラム記述の省略形

（　）：（　）で囲まれた文字そのもの

／　：区切り記号

Ⓕ　：改行キーの入力

＼　：バックスラッシュ

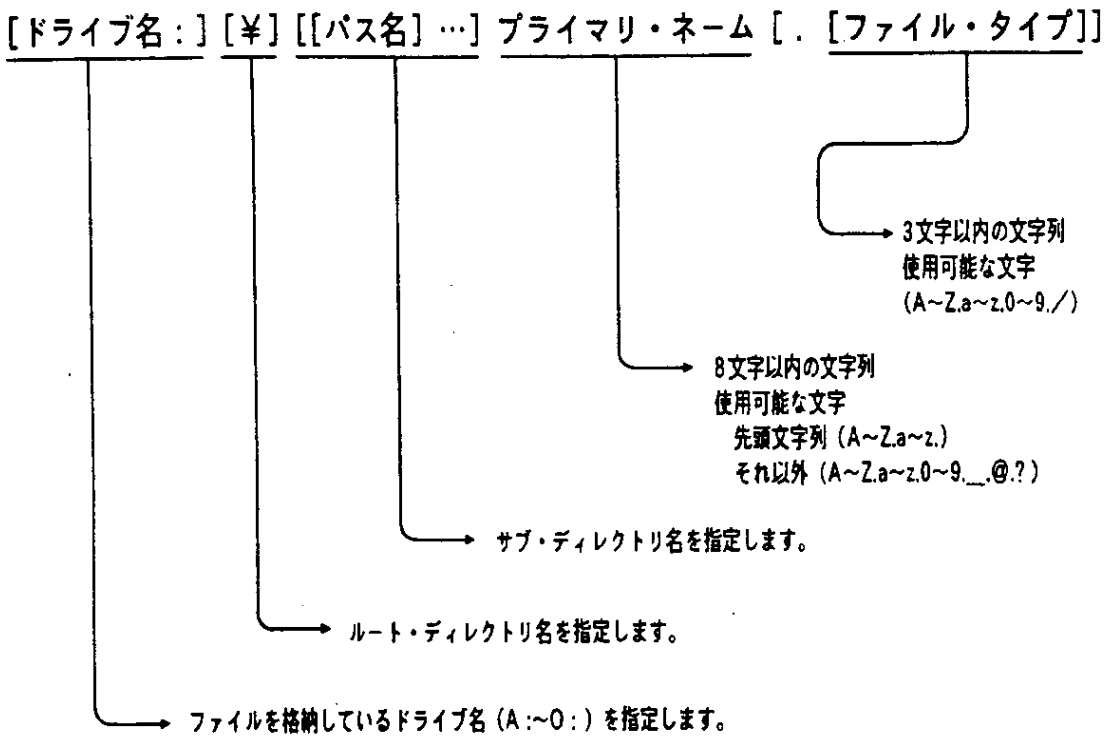
ＰＣ－ＤＯＳ[TM]の場合，‘¥’は‘＼（バックスラッシュ）’となります。

保守／廃止

【ファイル名の規則】

コマンド行で指定する入力ファイルの指定規則を次に示します。

(1) ディスク型ファイル名指定



例　A:¥sample¥prime.c

備考1.　':', '.', '¥' の前後に空白は指定できません。

2.　英大文字と小文字の区別はされません。

3.　ＰＣ－ＤＯＳの場合，'¥' は '＼（バックスラッシュ）' となります。

(2) デバイス型ファイル名指定

論理デバイスとして次のものがあります。

論理デバイス	説　明
ＣＯＮ	コンソールへ出力します。
ＰＲＮ	プリンタへ出力します。
ＡＵＸ	補助出力装置へ出力します。
ＮＵＬ	ダミー出力（何も出力しません。）

保守／廃止

【関連資料】

本マニュアルに関連する資料（ユーザーズ・マニュアルなど）を示します。

（1／3）

			資料名	資料番号
開発ツール	言語処理	共通	ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ言語編	EEU-654
			ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ操作編	EEU-662
			７８Ｋシリーズ　構造化アセンブラ・プリプロセッサ	EEU-643
			ＣＣ７８Ｋシリーズ　Ｃコンパイラ操作編	EEU-656
	ディバッガ	78K/0	ＩＥ－７８０００－Ｒ　ハードウエア編	EEU-750
			ＳＤ７８Ｋ／０　スクリーン・ディバッガ　ﾚﾌｧﾚﾝｽ編	作成中
			ＳＤ７８Ｋ／０　スクリーン・ディバッガ　入門編	作成中
		78K/Ⅱ	ＩＥ－７８２１０－Ｒ　ハードウエア取扱説明書	EEP-640
			ＩＥ－７８２１０－Ｒ　ソフトウエア取扱説明書	EEM-685
			ＩＥ－７８２１０－Ｒ　ｿﾌﾄｳｴｱ取扱説明書(MS-DOS)	EEM-677
			ＩＥ－７８２２０－Ｒ　ハードウエア取扱説明書	EEP-642
			ＩＥ－７８２２０－Ｒ　ソフトウエア取扱説明書	EEM-682
			ＩＥ－７８２２０－Ｒ　ｿﾌﾄｳｴｱ取扱説明書(MS-DOS)	EEM-678
			ＩＥ－７８２３０－Ｒ　ハードウエア編	EEM-682
			ＩＥ－７８２３０－Ｒ　ソフトウエア編	EEU-685
			ＩＥ－７８２４０－Ｒ　ハードウエア編	EEU-705
			ＩＥ－７８２４０－Ｒ　ソフトウエア編	EEU-706
			ＩＥ－７８２３０－Ｒ－Ａ　ハードウエア編	EEU-789
			ＩＥ－７８２４０－Ｒ－Ａ　ハードウエア編	EEU-796
			ＳＤ７８Ｋ／Ⅱ　スクリーン・ディバッガ　ﾚﾌｧﾚﾝｽ編	作成中
			ＳＤ７８Ｋ／Ⅱ　スクリーン・ディバッガ　入門編	作成中
		78K/Ⅲ	ＩＥ－７８３１０Ａ－Ｒ　ハードウエア編	EEU-645
			ＩＥ－７８３１０Ａ－Ｒ　ソフトウエア編	EEU-637
			ＩＥ－７８３１０Ａ－Ｒ　ソフトウエア編(MS-DOS)	EEU-646
			ＩＥ－７８３２０－Ｒ　ハードウエア編	EEU-709
			ＩＥ－７８３２０－Ｒ　ソフトウエア編	EEU-712
			ＩＥ－７８３３０－Ｒ　ハードウエア編	EEU-713
			ＩＥ－７８３３０－Ｒ　ソフトウエア編	EEU-714
			ＩＥ－７８３５０－Ｒ　ハードウエア編	EEU-754
			ＩＥ－７８３５０－Ｒ　ソフトウエア編	EEU-753

保守／廃止

			資　料　名	資料番号
デバイス	ユーザーズ・マニュアル	78K/0	μＰＤ７８０１Ｘシリーズ	IEU-780
		78K/Ⅱ	μＰＤ７８２１４シリーズ	IEM-5119
			μＰＤ７８２２４シリーズ	IEM-5019
			μＰＤ７８２１８Ａシリーズ	IEU-755
			μＰＤ７８２３４シリーズ	IEU-718
			μＰＤ７８２４４シリーズ	IEU-747
		78K/Ⅲ	μＰＤ７８３１２Ａ	IEM-5086
			μＰＤ７８３２２	IEU-619
			μＰＤ７８３２８	IEU-693
			μＰＤ７８３３４	IEU-729
			μＰＤ７８３５０	IEU-781
	レファレンス	78K/0	μＰＤ７８００Ｘシリーズ　ｲﾝｽﾄﾗｸｼｮﾝ･ｾｯﾄ	IEM-5546
			μＰＤ７８００Ｘシリーズ　インストラクション活用表	IEM-5545
			μＰＤ７８００Ｘシリーズ　モード・レジスタ活用表	IEM-5547
			μＰＤ７８０１Ｘシリーズ　ｲﾝｽﾄﾗｸｼｮﾝ･ｾｯﾄ	IEM-5521
			μＰＤ７８０１Ｘシリーズ　インストラクション活用表	IEM-5522
			μＰＤ７８０１Ｘシリーズ　モード・レジスタ活用表	IEM-5527
		78K/Ⅱ	７８Ｋ／Ⅱシリーズ　インストラクション活用表	IEM-5101
			７８Ｋ／Ⅱシリーズ　インストラクション・セット	IEM-5102
			μＰＤ７８２１４シリーズ　モード・レジスタ活用表	IEM-5100
			μＰＤ７８２１８Ａシリーズ　モード・レジスタ活用表	IEM-5532
			μＰＤ７８２２４シリーズ　モード・レジスタ活用表	IEM-999
			μＰＤ７８２３４シリーズ　モード・レジスタ活用表	IEM-5515
			μＰＤ７８２４４シリーズ　モード・レジスタ活用表	IEM-5528

			資　料　名	資料番号
デバイスレファレンス		78K/Ⅲ	μＰＤ７８３１２Ａ　インストラクション・セット	IEM-5116
			μＰＤ７８３１２Ａ　インストラクション活用表	IEM-5115
			μＰＤ７８３１２Ａ　モード・レジスタ活用表	IEM-5118
			μＰＤ７８３２２　インストラクション・セット	IEM-601
			μＰＤ７８３２２　インストラクション活用表	IEM-602
			μＰＤ７８３２２　モード・レジスタ活用表	IEM-5501
			μＰＤ７８３３４　モード・レジスタ活用表	IEM-5518
			μＰＤ７８３２８　モード・レジスタ活用表	IEM-5514
			μＰＤ７８３５０　インストラクション・セット	IEM-5543
			μＰＤ７８３５０　モード・レジスタ活用表	IEM-5540

注意　関連資料の最新情報については，販売員にご連絡ください。

目次要約

保守／廃止

目　次

図の目次

保守／廃止

表の目次

保守／廃止

第１章　概　　説

ＣＣ７８Ｋシリーズ　Ｃコンパイラは，７８ＫシリーズのＣ言語で記述された，Ｃソース・プログラムを機械語に変換するプログラムです。

1.1 Ｃコンパイラとは

1.1.1 Ｃ言語とアセンブリ言語

マイクロプロセッサに仕事をさせるには，プログラムやデータが必要です。これを人間がプログラミングして，マイクロコンピュータのメモリ部に記憶させます。マイクロコンピュータが扱うことのできるプログラムやデータは２進数の集まりで，これを機械語（コンピュータの理解できる言葉）といいます。

この機械語に英語の略記号を１対１で対応させたものがアセンブリ言語です。アセンブリ言語は，機械語と１対１で対応しているためコンピュータに対して詳細な指示を与えることができます（たとえば，入出力時の処理速度の向上など）。しかし，このことはコンピュータのあらゆる動作を１つ１つ指示しなければならないことを意味しています。そのためにプログラムの論理構造が，一目見ただけでは理解しにくく，またエラーなども発生しやすいものです。

このようなアセンブリ言語に代わるものとして高級言語が開発されました。その中の１つにＣ言語があります。これによりプログラマは，コンピュータのアーキテクチャを気にせずにプログラミングすることができ，プログラム自体もアセンブリ言語に比べて論理構造などが理解しやすくなったといえます。

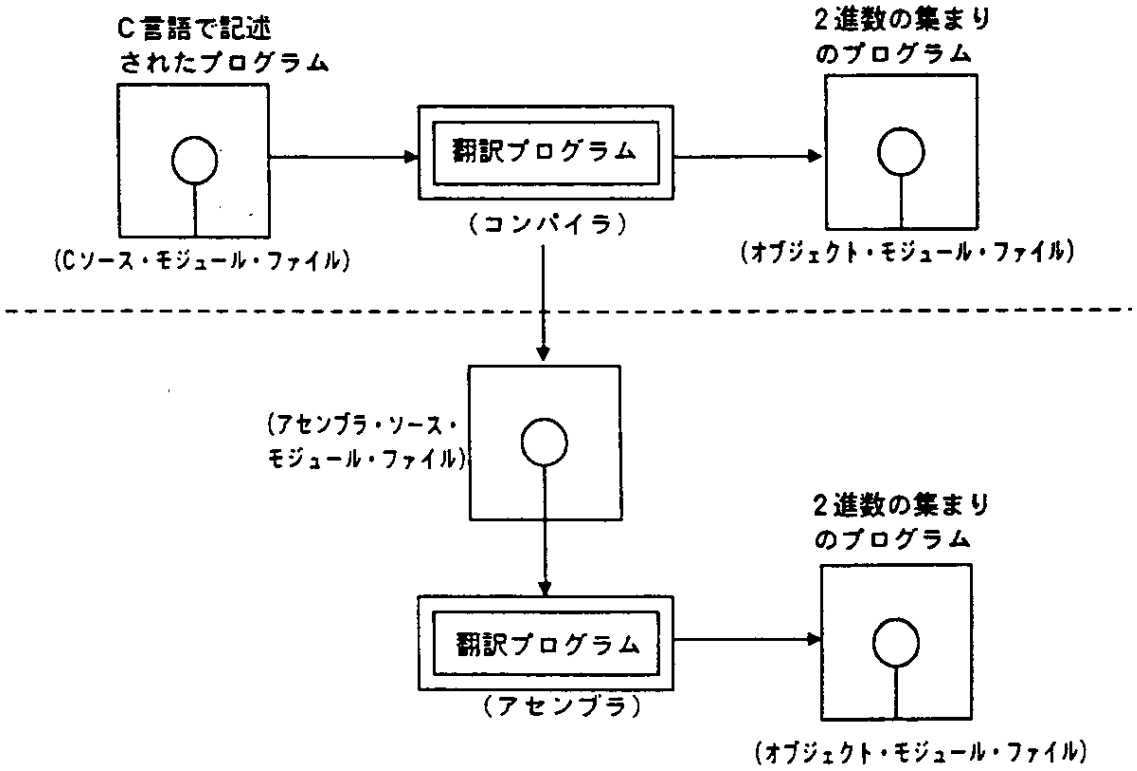
また，Ｃ言語ではプログラムを作成するための多くの部品（関数）が用意されているので，プログラマはこれらを組み合わせてプログラムを作成することができます。

Ｃ言語は，人間にとって理解しやすいという特徴を持っています。しかし，Ｃ言語で書かれたプログラムのままでは，マイクロコンピュータは理解することができません。Ｃ言語を理解させるには，それに相当する機械語に翻訳するプログラムが必要となります。このようにＣ言語を機械語に翻訳する翻訳プログラムをＣコンパイラと呼びます。

本Ｃコンパイラは，Ｃソース・モジュールを入力しオブジェクト・モジュールとアセンブラ・ソース・モジュールを出力します。したがって，プログラマはＣ言語を用いてプロ

グラムを作成し，プログラムの実行の細部まで指示したい場合にはアセンブリ言語でプログラムを修正することができます。本Ｃコンパイラの流れを“図１－１　コンパイルの流れ”に示します。

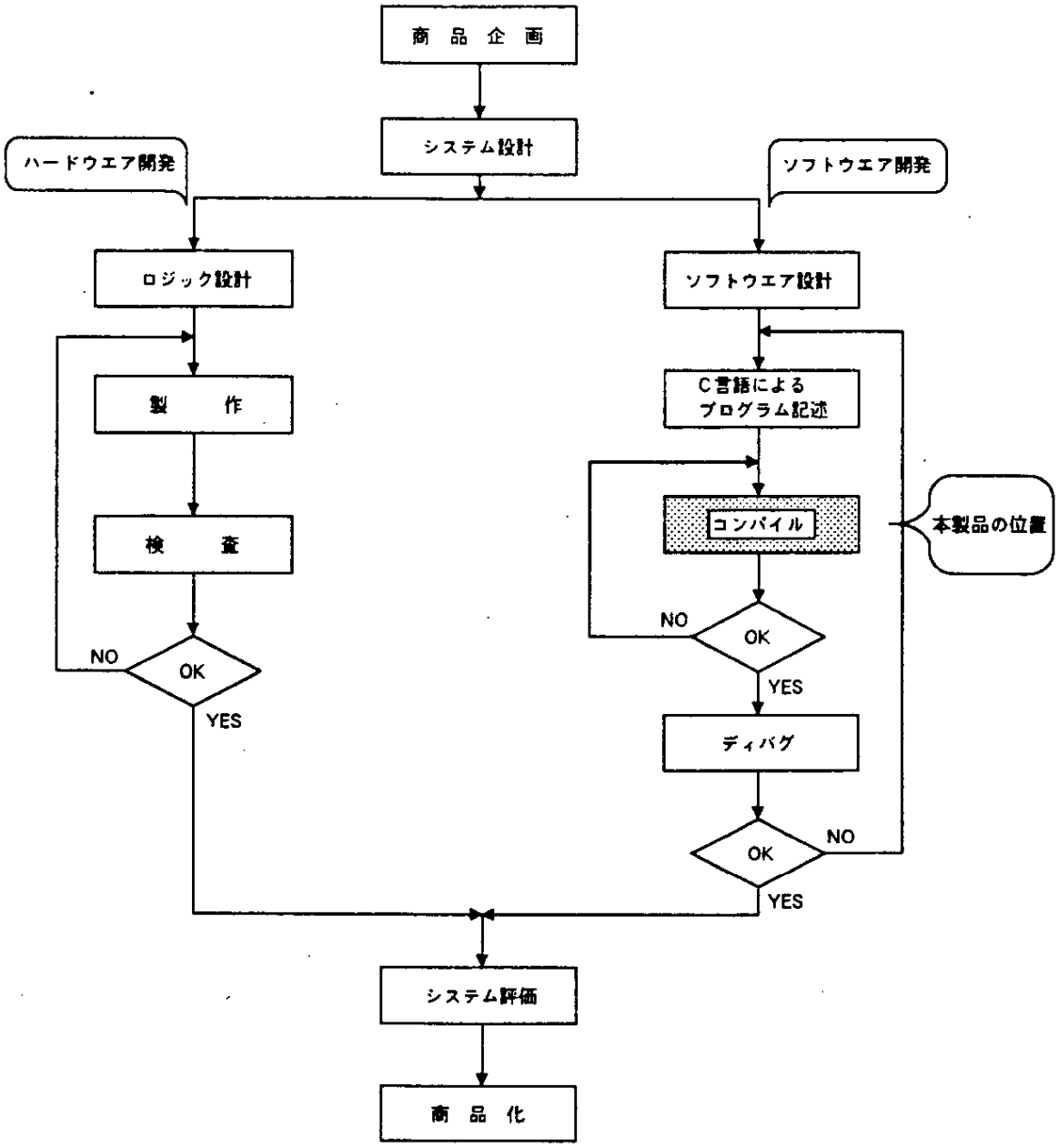
図１－１　コンパイルの流れ



1.1.2 マイクロコンピュータ応用製品の開発と本製品の役割

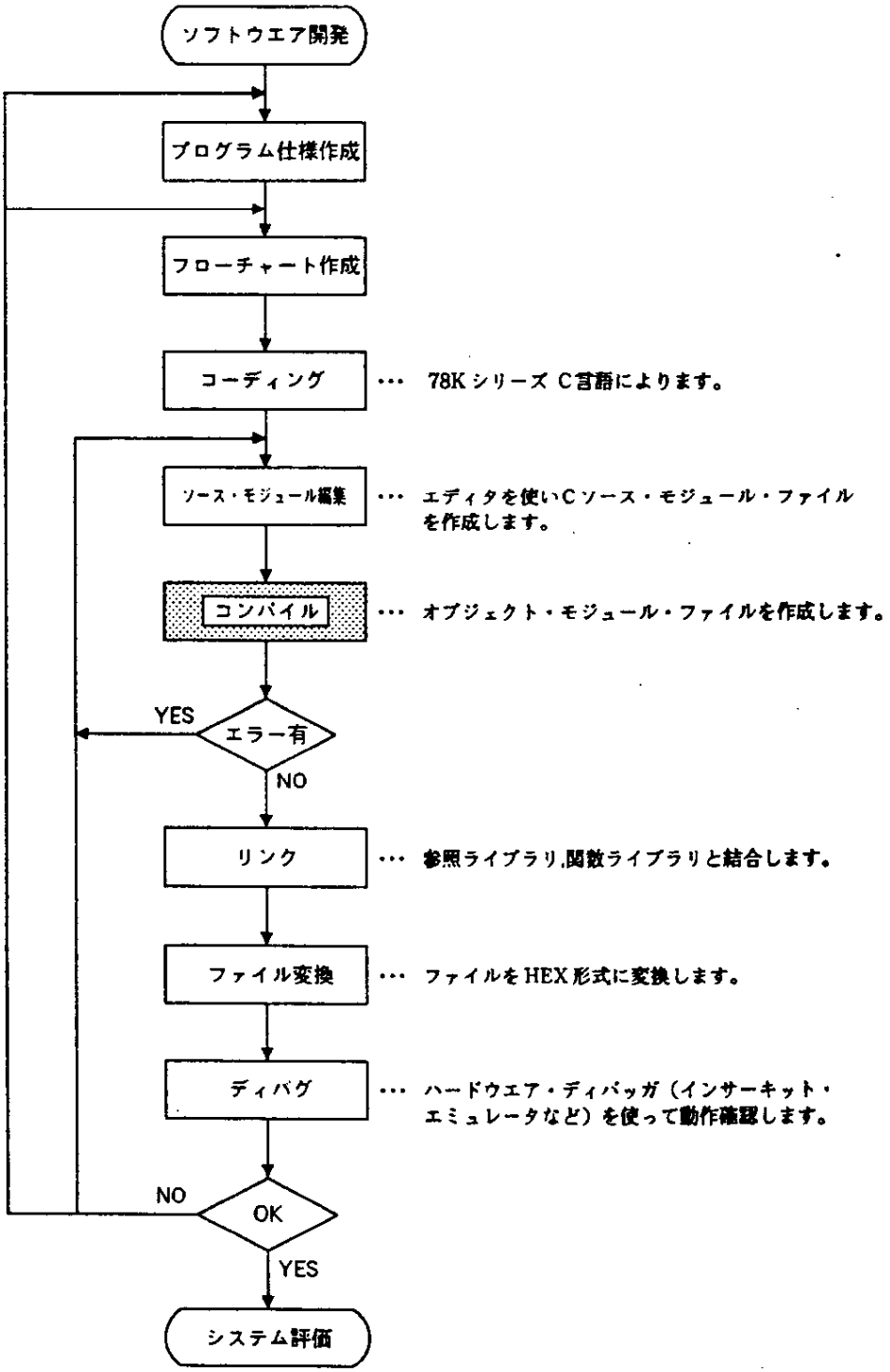
C言語でのプログラミングは，製品開発のどこに位置するかの概略を“図１－２　マイクロコンピュータ応用製品の開発行程”に示します。

図１－２　マイクロコンピュータ応用製品の開発行程



ソフトウエア開発の行程を“図１－３　ソフトウエア開発行程”に示します。

図１－３　ソフトウエア開発行程

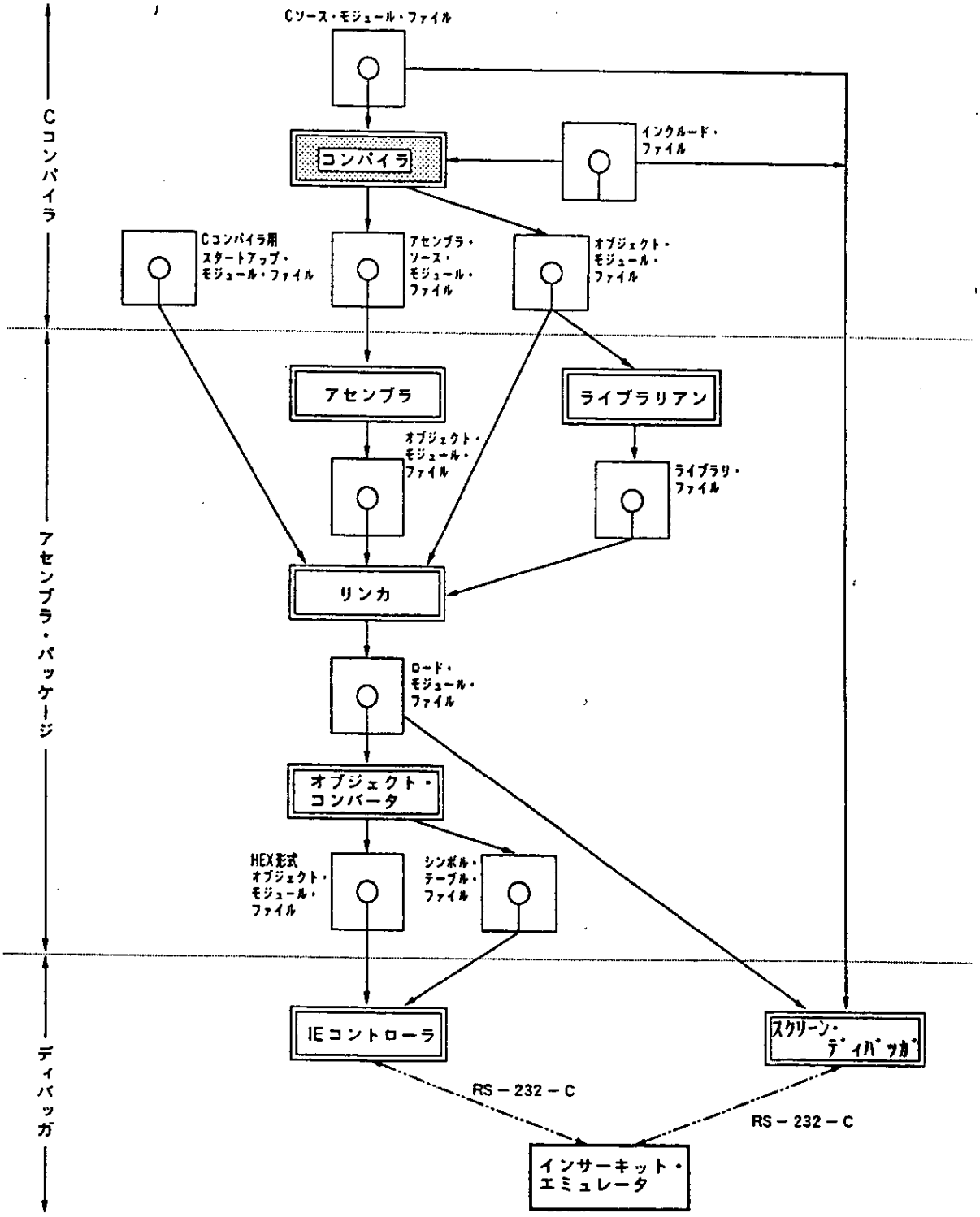


保守／廃止

1.2 本Cコンパイラによる開発手順

本Cコンパイラにおける開発手順を“図１－４　本Cコンパイラによるプログラム開発手順”に示します。

図１－４　本Cコンパイラによるプログラム開発手順



1.2.1 エディタによるソース・モジュール・ファイルの作成

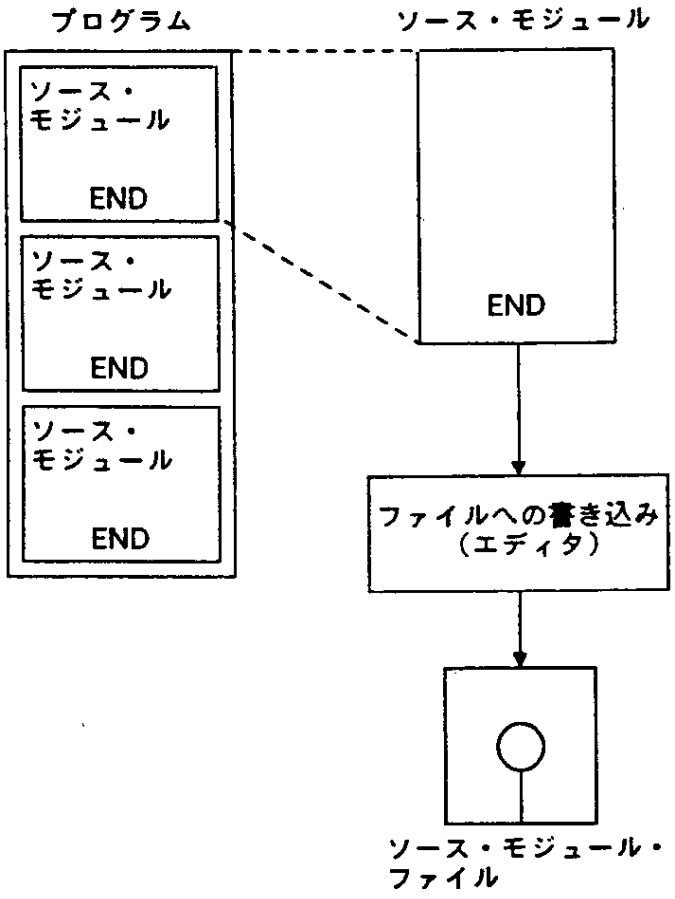
１つのプログラムを機能的にいくつかに分割します。

１つのモジュールは，コーディングの単位になるもので，またコンパイラの入力単位ともなります。Ｃコンパイラの入力単位となるモジュールを，Ｃソース・モジュールと呼びます。

各Ｃソース・モジュールのコーディング終了後，エディタを使用してソース・モジュールをファイルに書き込みます。こうしてできたファイルをＣソース・モジュール・ファイルと呼びます。

Ｃソース・モジュール・ファイルは，本Ｃコンパイラの入力ファイルとなります。

図１－５　ソース・モジュール・ファイルの作成

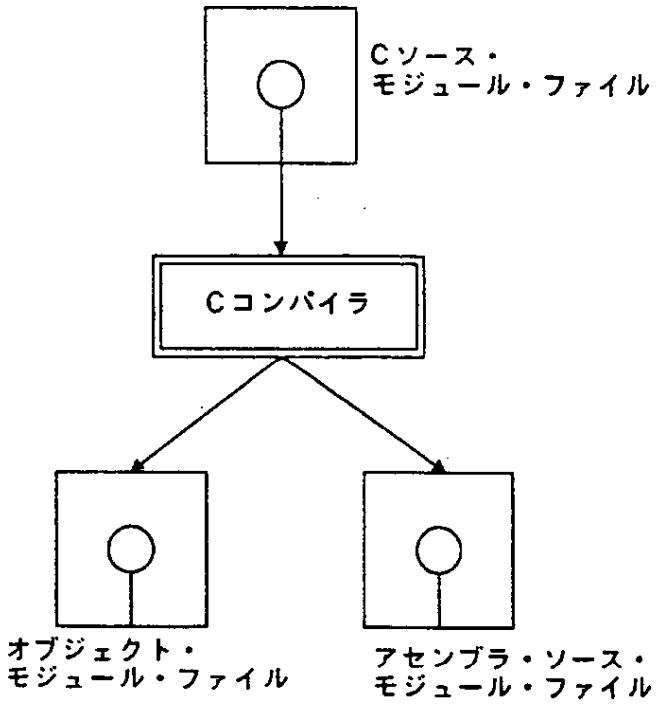


1.2.2 Cコンパイラ

本Cコンパイラは，Cソース・モジュールを入力し，C言語を機械語（2進数の集まり）に変換します。Cソース・モジュール中に記述ミスを発見した場合は，コンパイル・エラーを出力します。

コンパイル・エラーがない場合には，オブジェクト・モジュール・ファイルを出力します。また，アセンブリ言語レベルでプログラムの修正，確認を行えるようにアセンブラ・ソース・モジュール・ファイルを出力します。出力したい場合には，コンパイルの際にアセンブラ・モジュール・ファイル作成指定の－Aオプションを指定してください。

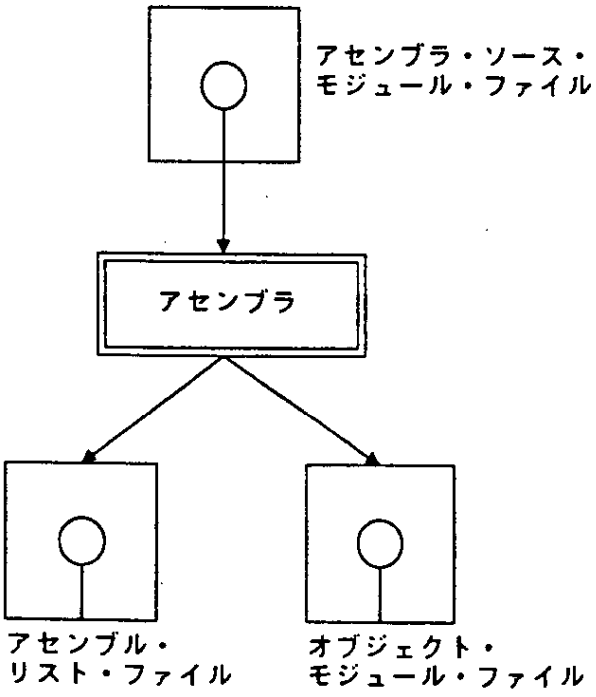
図1－6　Cコンパイラの機能



1.2.3 アセンブラ

アセンブラは，アセンブラ・ソース・モジュール・ファイルを入力し，アセンブリ言語を機械語に翻訳するプログラムです。ソース・モジュール中に記述ミスを発見した場合は，アセンブル・エラーを出力します。アセンブル・エラーがない場合には，機械語情報と各機械語がメモリ中のどのアドレスに配置されるべきかなどの配置情報とを含むオブジェクト・モジュール・ファイルを出力します。また，アセンブル時の情報をアセンブル・リスト・ファイルとして出力します。

図１－７　アセンブラの機能



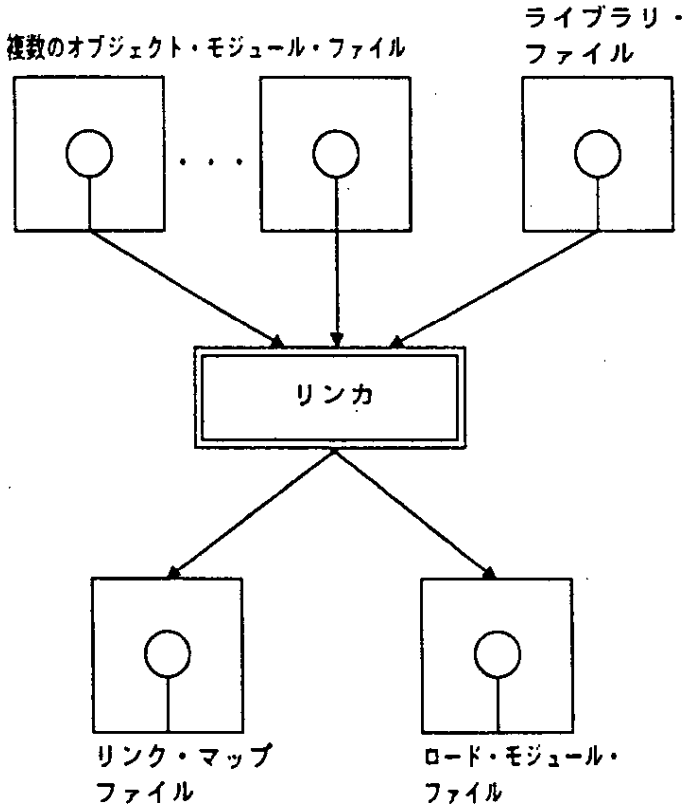
1.2.4 リンカ

リンカは，コンパイラの出力したオブジェクト・モジュール・ファイル，またはアセンブラの出力したオブジェクト・モジュール・ファイルを複数個入力し，それらをライブラリ・ファイルと結合します（オブジェクト・モジュールが１つの場合でも，リンクしなければなりません）。そして，１つのロード・モジュール・ファイルを出力します。

この際リンカは，入力されたモジュール中のリロケータブル・セグメントの配置アドレスを決定します。これにより，リロケータブル・シンボルや外部参照シンボルの値を決定し，ロード・モジュール・ファイルに正しい値を埋め込みます。

また，リンカはリンク時の情報をリンク・マップ・リストとして出力します。

図１－８　リンカの機能

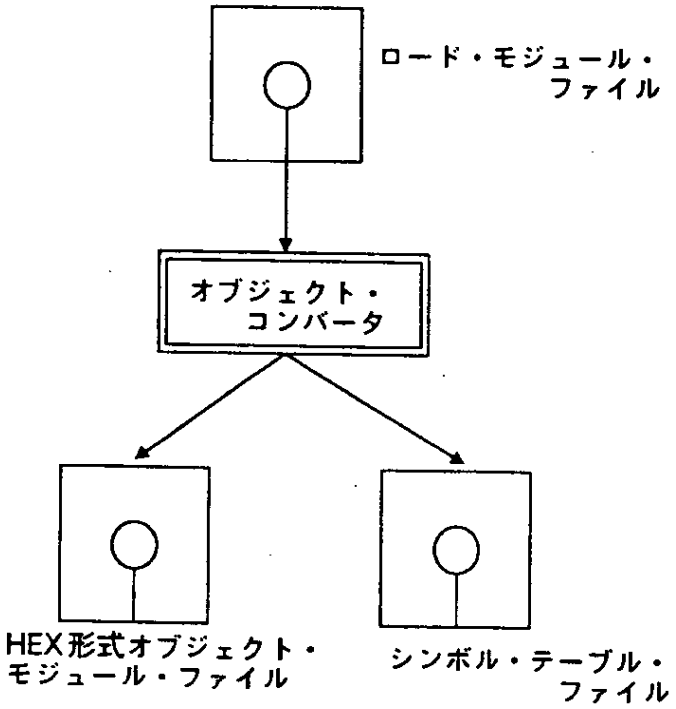


1.2.5 オブジェクト・コンバータ

オブジェクト・コンバータは，リンカの出力したロード・モジュール・ファイルを入力し，ファイル形式を変換します。そして，その結果をＨＥＸ形式オブジェクト・モジュール・ファイルとして出力します。

また，シンボリック・ディバグ時に必要となるシンボル情報をシンボル・テーブル・ファイルとして出力します。

図１－９　オブジェクト・コンバータの機能



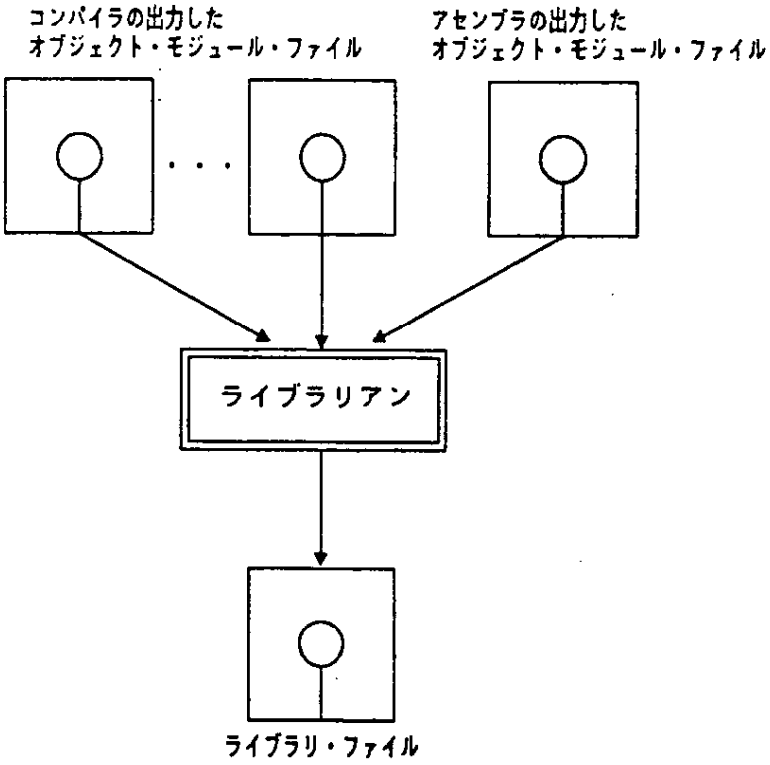
1.2.6 ライブラリアン

汎用性のある，インタフェースの明確なモジュールは，ライブラリ化しておくと便利です。ライブラリ化を行うことにより，多くのオブジェクト・モジュールも１つのファイルになり，扱いやすくなります。

リンカには，ライブラリ・ファイルの中から必要なモジュールだけを取り出してリンクする機能があります。したがって，複数のモジュールを１つのライブラリ・ファイルに登録しておけば，リンク時に必要なモジュール・ファイル名をいちいち指定する必要はなくなります。

ライブラリ・ファイルの作成と更新にライブラリアンを使用します。

図１－１０　ライブラリアンの機能

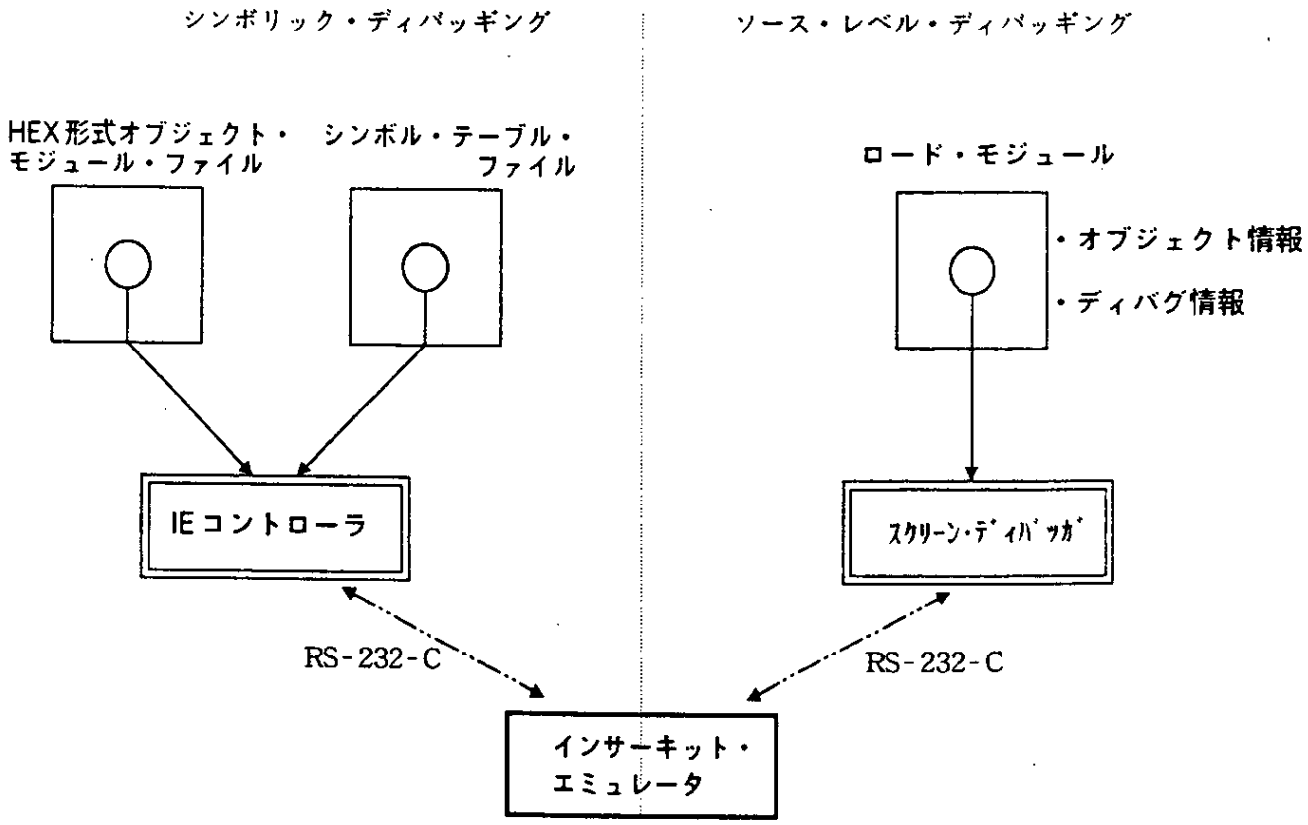


1.2.7 スクリーン・ディバッガ

オブジェクト・コンバータが出力したＨＥＸ形式オブジェクト・ファイルを，ターゲット・システムのＩＥ（インサーキット・エミュレータ）またはＥＢ（評価用ボード）にダウン・ロードし，さらにシンボル・テーブル・ファイルを読み込むことによってシンボリック・レベルのディバグが可能になります。

もう１つの方法として，対象のソース・プログラムをコンパイルする際にディバグ情報出力指定オプションを指定しておきます。そうすることにより，ディバグに必要なシンボルと行番号の情報がオブジェクト・モジュール内に付加され，ソース・レベルのディバグが可能となります。

図１－１１　スクリーン・ディバッガの機能



1.3 プログラム開発をはじめる前に

表１－１　Ｃコンパイラの最大性能

項番	項　　目	制　限　値	ページ
1	複文，繰り返し制御文，選択制御文のネスト	４５	言 102
2	条件コンパイルのネスト	２５５	言 134
3	修飾宣言子のネスト	１２	言 33
4	式中のかっこのネスト	３２	言 61
5	マクロ名で意味を持つ文字数	３１	言 145
6	内部，外部シンボル名で意味を持つ文字数	７ 注1	－
7	１ソース・モジュール・ファイル中のシンボル数	１０２４ 注2	－
8	１ブロックでブロック・スコープを持つシンボル数	２５５ 注2	言 16
9	１ソース・モジュール・ファイル中のマクロ数	１０２４ 注3	言 145
１０	関数定義，関数呼び出しのパラメータ	３９	言 129
１１	１つのマクロ定義，マクロ呼び出しのパラメータ	３１	言 145
１２	１つの論理ソース行の文字数	５０９	－
１３	結合後の文字列リテラル内の文字数	５０９	言 29
１４	１つのオブジェクト・サイズ（データを示します）	６５５３５byte	－
１５	#includeのネスト	８	言 141
１６	switch文のcaseラベル数	２５７	言 105
１７	１コンパイル単位のソース行数	約３０００	－
１８	テンポラリ・ファイルを作成せずにコンパイルできるソース行数	約３００	－
１９	関数コールのネスティング	４０	言 129
２０	１ステートメントの前のラベル数	３３	言 104
２１	１オブジェクト・モジュールあたりのコード，データ，スタック・セグメントのトータルサイズ	６５５３５byte	－
２２	１つの構造体または，共用体のメンバ数	１２７	言 121
２３	１つの列挙の列挙定数の数	１２７	言 39
２４	１つの構造体または，共用体における構造体または共用体のネスト	１５	言 121
２５	初期化子要素のネスト	１５	－

・ 項番５，６，１２，１３，１４，１５，２１以外は制限ではなく，保証されている最小の値です。

・ ページ番号は，“ユーザーズ・マニュアル　言語編”のページ番号です。

注1. コンパイラ・オプション（－Ｓ）により、シンボル名の長さを３０文字に拡張することができます。

2. テンポラリ・ファイルを使用せずに，メモリ・スペースのみで処理できる制限値を示します。メモリ・スペースで処理しきれない場合は，テンポラリ・ファイルを使用し，そのときの制限値はファイル・サイズにより変わります。

3. コンパイラの予約マクロ定義を含みます。

1.4 本Cコンパイラの特徴

本Cコンパイラは，ＡＮＳＩ（American lational Standards Institute）にないＣＰＵのコードを生成する拡張機能を備えています。本Cコンパイラの拡張機能には，７８Ｋシリーズの特殊機能用レジスタをＣ言語レベルで記述可能にするものや，オブジェクト・コードを短縮し実行速度の向上を図るものがあります。拡張機能の詳細については，“言語編　第１１章　拡張機能”をご覧ください

オブジェクト・コードを短縮し，実行速度を向上させる方法としては，次のものがあります。

- callt領域を利用して関数を呼び出す……callt関数
- 変数をレジスタに割り当てる……レジスタ変数
- saddr領域に変数を割り当てる……saddr領域
- sfr名を使用できる……sfr領域
- 前後処理（スタック・フレーム）のない関数を生成する……noauto関数，norec関数
- Ｃソース・プログラム中にアセンブリ言語を記述する……ＡＳＭ文
- saddr,sfr領域へのビット・アクセスを行う……bit型変数
- callf領域に関数本体を格納する……callf関数

①callt関数

呼び出される関数のアドレスをcallt領域に置き関数が呼び出されます。通常の呼び出しに比べ関数の切り換えが速くなります。また，オブジェクト・コードを短縮できます。

②レジスタ変数

レジスタ，またはsaddr領域に変数を取ることができ，通常の変数を使用した場合と比べ実行速度が向上します。また，オブジェクト・コードを短縮できます。

③saddr領域

変数をsaddr領域に取ることができ，通常の変数を使用した場合と比べ実行速度が向上します。また，オブジェクト・コードを短縮できます。

④sfr領域

特殊機能レジスタ（ＳＦＲ）を，sfrの略号（sfr名）によってＣソース・ファイル中で使用することができます。

⑤noauto関数

前後処理（スタック・フレーム）のない関数を生成します。noauto関数の呼び出しで引数は，可能な限りレジスタ渡しになります。これにより，実行速度が向上しオブジェクト・コードが短縮ができます。

⑥norec関数

前後処理(スタック・フレーム)のない関数を生成します。norec関数の呼び出しで引数は，可能な限りレジスタ渡しになります。また，norec関数内で使用するオートマティック変数は，saddr領域に割り当てられます。これにより，実行速度の向上およびオブジェクト・コードの短縮ができます。

⑦bit型変数

bit型の変数を生成します。bit型変数によりsaddr領域へのビット・アクセスができます。

⑧ＡＳＭ文

Ｃコンパイラが出力したアセンブラ・ソース・ファイルにユーザが記述したアセンブラ・ソースが埋め込まれます。

⑨漢　字

Ｃソース・ファイルのコメント文中に漢字（シフトＪＩＳコード）を記述することができます。

⑩割り込み関数

ベクタ・テーブルを生成し，割り込みに対応したオブジェクト・コードを出力します。これにより，Ｃソース・レベルで割り込み関数の記述が可能となります。

⑪割り込み機能

オブジェクトに割り込み禁止命令，割り込み許可命令を埋め込みます。

⑫callf関数

callf命令は，callf領域に関数本体を格納し，call命令に比べて短いコードで関数を呼ぶことを可能にします。これにより，オブジェクト・コードを短縮できます。

⑬１Mbyte拡張空間利用法（７８Ｋ／Ⅱのみサポート）

オブジェクトに１Mbyte拡張空間をアクセスするコードを関数呼び出しではなく，直接インライン展開して出力し，オブジェクト・ファイルを作成します。

⑭テーブル切り換え機能（７８Ｋ／Ⅲのみサポート）

Ｃコンパイラの出力するベクタ・テーブル，calltテーブルのアドレスを変更することができます。

第２章　製品概要

2.1 フロッピィ・ディスクの内容

システム・ファイル
（フロッピィ・ディスク(1)）
- ＢＩＮ（実行用ファイル）
- ＩＮＣＬＵＤＥ（ヘッダ・ファイル）
- ＳＡＭＰＬＥ（サンプル・プログラム）
- ＬＩＢ（リンク用標準ディレクティブ・ファイル）
- ＢＡＴ（スタートアップ・ルーチン
  エラー処理ルーチン）
- ＳＲＣ（関連ファイル）
- ＩＮＣ（デバイス情報インクルード・ファイル）

ライブラリ・ファイル　…　各デバイス別ライブラリ・ファイル
（フロッピィ・ディスク(2)以降）

2.1.1 システム・ファイル

システム・ファイルは，フロッピィ・ディスク(1)で提供されます。フロッピィ・ディスク(1)にある各ディレクトリの内容を次の表に示します。

表２－１　システム・ファイル　（１／２）

ディレクトリ名	ファイル名　注1	ファイルの役割
ＢＩＮ	ＣＣ７８Ｋｎ．ＥＸＥ	・コントローラ
	ＳＡ７８Ｋｎ．ＥＸＥ	・構文解析部
	ＸＯ７８Ｋｎ．ＥＸＥ	・ｸﾛｽ･ﾚﾌｧﾚﾝｽ･ﾘｽﾄ出力部
	ＣＧ７８Ｋｎ．ＥＸＥ	・コード生成部
	ＬＯ７８Ｋｎ．ＥＸＥ	・オブジェクト・リスト出力部
	ＯＰ７８Ｋｎ．ＥＸＥ（V2.00以上のみ）	・最適化処理部
	ＣＣ７８Ｋｎ．ＯＭｘ 注2 ～ ＣＣ７８Ｋｎ．ＯＭｘ	・オーバレイ・ファイル（デバイス情報）
	ＣＣ７８Ｋｎ．ＨＬＰ	・ヘルプ・ファイル
	ＣＣ７８Ｋｎ．ＭＳＧ	・メッセージ・ファイル
ＩＮＣＬＵＤＥ	○○○．Ｈ	・CC78Kｼﾘｰｽﾞ Cｺﾝﾊﾟｲﾗ ﾕｰｻﾞｰｽﾞ･ﾏﾆｭｱﾙ 言語編 10.2 ﾍｯﾀﾞ･ﾌｧｲﾙ 参照
ＩＮＣＬＵＤＥ¥７８ｘｘｘ	ＳＦＲＢＩＴ．Ｈ	・ｓｆｒのビット機能名称を定義したヘッダ・ファイル
ＳＡＭＰＬＥ	ＰＲＩＭＥ．Ｃ	・サンプルＣソース・ファイル
	ＳＡＭＰＬＥ．ＢＡＴ	・バッチ・ファイル
	ＲＥＡＤＭＥ．ＤＯＣ	・ドキュメント・ファイル
ＬＩＢ	ＬＩＮＫｘｘｘ．ＤＩＲ ＬＩＮＫｘｘｘ．ＲＯＭ	・リンク用標準ディレクティブ・ファイル .DIR … ＲＯＭ化しない場合 .ROM … ＲＯＭ化する場合 XXX : デバイス種別 （表5-3～表5-5 ﾃﾞﾊﾞｲｽ種別参照）

表２－１　システム・ファイル　　（２／２）

ディレクトリ名	ファイル名	ファイルの役割
ＬＩＢ￥ＢＡＴ	ＭＫＳＴＵＰ．ＢＡＴ	・スタートアップ・ルーチン作成用バッチ・ファイル
	ＭＫＥＲＲＬＩＢ．ＢＡＴ	・エラー処理ルーチン更新用バッチ・ファイル
	○○○．ＥＲＲ	・MKERRLIB.BATで使用されるファイル（ﾗｲﾌﾞﾗﾘｱﾝ起動時に使用するｻﾌﾞ・ｺﾏﾝﾄﾞ・ﾌｧｲﾙ）
ＬＩＢ￥ＳＲＣ	○○○．ＡＳＭ	・スタートアップ・ルーチン，エラー処理ルーチンのソース・ファイル
	○○○．ＩＮＣ	・○○○．ＡＳＭ用インクルード・ファイル
ＬＩＢ￥ＩＮＣ	○○○．ＩＮＣ	・デバイス情報インクルード・ファイル

注1.　ｎ　＝　０，２，３：ｎは各シリーズの番号となります。

2.　オーバレイ・ファイルの数は各シリーズにより異なります。

備考1.　コマンド・ファイル（ファイル・タイプがＥＸＥ）は，プログラムが起動されたときに最初にメモリ内に読み込まれるファイルです。

2.　オーバレイ・ファイルは，プログラムの実行中に必要なファイルだけメモリに読み込まれます。

2.1.2 ライブラリ・ファイル

ライブラリ・ファイルは，２枚目以降のフロッピィ・ディスクで提供されます。

・各デバイス用の標準ライブラリおよびスタートアップ・ルーチンで構成されます。

・フロッピィ・ディスクは，以下で示されるような２つのディレクトリで構成されます。

ＬＩＢＮＣＡ … コンパイル・オプション－ＮＣＡを指定して作成されたオブジェクト・ファイルをリンクするときに必要となる標準ライブラリおよびスタートアップ・ルーチンを提供します。

ＬＩＢＣＡ … コンパイル・オプション－ＣＡを指定して作成されたオブジェクト・ファイルをリンクするときに必要となる標準ライブラリおよびスタートアップ・ルーチンを提供します。

以下の表に，ディレクトリの内容を示します。

表２－２　ライブラリ・ファイル

ディレクトリ名	ファイル名 注１	ファイルの役割
ＬＩＢＮＣＡ	ＣＬＫｎＣＯＭ．ＬＩＢ	・78K/n用の共通ﾗｲﾌﾞﾗﾘ･ﾌｧｲﾙ 注２
ＬＩＢＮＣＡ¥ ＬＩＢｘｘｘ	ＣＬｘｘｘ．ＬＩＢ	・sfrﾁｪｯｸ用ﾗｲﾌﾞﾗﾘ･ﾌｧｲﾙ 注３
	ＣＳｘｘｘ．ＲＥＬ ＣＳｘｘｘＲ．ＲＥＬ ＥＳｘｘｘＲ．ＲＥＬ	・スタートアップ・ルーチン
	ＲＯＭｘｘｘ．ＲＥＬ	・ＲＯＭ化用モジュール・ファイル

注１．ｘｘｘ　：デバイス種別（表５－３～表５－５参照）

ｎ＝０，２，３　：ｎは各シリーズの番号となります。

２．以下のデバイス種別のライブラリ・ファイルについては，共通ライブラリ・ファイルの代わりに矢印で示すライブラリ・ファイルをお使いください。

３１０　→　ＣＬ３１０．ＬＩＢ

３１２　→　ＣＬ３１２．ＬＩＢ

３１０Ａ　→　ＣＬ３１０Ａ．ＬＩＢ

３１２Ａ　→　ＣＬ３１２Ａ．ＬＩＢ

注３．ＣＬｘｘｘ．ＬＩＢは，ｓｆｒに対して不正な読み出し，書き込みが行われていないかをチェックするためのものです。このチェック用ライブラリは，Ｃコンパイラの実行時エラー・チェック指定（－Ｌ）オプションを指定することにより，オブジェクト・ファイルに組み込まれます。したがって，実行時エラー・チェック指定（－Ｌ）オプションを指定の際は，リンク時に共通ライブラリ・フィアルとともに，ＣＬｘｘｘ．ＬＩＢも，ライブラリ・ファイルとして指定してください。

ディレクトリＬＩＢＣＡについても，ファイル名の最後に’Ｕ’が付加される点を除いて同様です。

2.2 供給形態

本製品は，次の媒体で提供されます。

・５インチの２ＨＤフロッピィ・ディスク
・３．５インチの２ＨＤフロッピィ・ディスク

2.3 システム構成

本Ｃコンパイラは，“表２－３　システム構成”で示す環境で動作します。

表２－３　システム構成

ホスト・コンピュータ	ＣＰＵ	ＯＳ	CONFIG.SYS	メモリ・サイズ
ＰＣ－９８００シリーズ	V30™ 80386™	ＭＳ－ＤＯＳ™ (V2.11,V3.10, V3.30A)	FILES=25 以上に設定	使用可能メモリが３６０Ｋバイト必要です。
IBM PC,IBM PC/XT™ IBM PC/AT™	80286™ 8086™	ＰＣ－ＤＯＳ (V3.10)		

注1. ＣＣ７８Ｋシリーズ　Ｃコンパイラは，弊社提供のＰＣ－９８００シリーズ用ＭＳ－ＤＯＳにて動作いたします。
このほかの市販ＭＳ－ＤＯＳ上における本プログラムの動作については，その責を負いかねますのでご了承ください。

2. メモリ・サイズは本Ｃコンパイラが動作時に必要とするメモリの最大値です。システム領域は含んでいません。

3. ＰＣ－ＤＯＳの場合には，‘¥’は‘＼（バックスラッシュ）’となります。

第３章　Ｃコンパイラの実行

本章では，ＣＣ７８Ｋ３を例に用いてＣコンパイラの実行手順を説明します。本章の実行手順に従って実際にサンプル・プログラムを実行することにより，Ｃコンパイラの操作に慣れることができます。

3.1　Ｃコンパイラ実行の前に

3.1.1　ディスク内容の確認

本Ｃコンパイラのディスク（または，そのバックアップ・ディスク）をドライブにセットし，ディスクの内容を確認します。
ディスクの中に“2.1　フロッピィ・ディスク内容”で紹介された，すべてのファイルがあることを確認してください。

3.1.2 サンプル・プログラム

本Cコンパイラの説明にあたり，以下のサンプル・プログラム‘PRIME．C’を使用します。

```
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;

        count = 0;

        for ( i = 0 ; i <= SIZE ; i++)
                mark[i] = TRUE;
        for ( i = 0 ; i <= SIZE ; i++) {
                if (mark[i]) {
                        prime = i + i + 3;
                        printf("%6d",prime);
                        count++;
                                if((count%8) == 0) putchar('¥n');
                        for ( k = i + prime ; k <= SIZE ; k += prime)
                                mark[k] = FALSE;
                }
        }
        printf("¥n%d primes found.",count);
}

printf(s,i)
char *s;
int i;
{
        int j;
        char *ss;

        j = i;
        ss = s;
}

putchar(c)
char c;
{
        char d;
        d = c;
}
```

3.2 Cコンパイラの実行

(1) サンプル・プログラム‘ＰＲＩＭＥ．Ｃ’をコンパイルします。

・起動行に次のように入力します。起動行の入力は，英大文字，英小文字のどちらでも可能です。

```
A>cc78k3 -c310 sample¥prime.c
          └──────────  ターゲット・デバイスの種別を指定します。
```

・次のメッセージがコンソールに出力されます。

```
78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx
```

(2) ドライブＡの内容をチェックします。

Ｃコンパイラは，ＰＲＩＭＥ．ＲＥＬ（オブジェクト・モジュール・ファイル）を出力しました。

なお，コンパイル時に－Ａ，－Ｐ，－Ｘ，－Ｅなどのコンパイラ・オプションを指定することにより，本Ｃコンパイラはアセンブラ・ソース・モジュール・ファイル，プリプロセス・リスト・ファイル，クロス・レファレンス・リスト・ファイルおよびエラー・リスト・ファイルの４つのリスト・ファイルを出力することができます。

保守／廃止

第４章　Ｃコンパイラ

本Ｃコンパイラは，７８Ｋシリーズの Ｃ言語で記述されたＣソース・モジュール・ファイルとインクルード・ファイルを入力します。そして，それらを機械語に変換してオブジェクト・モジュール・ファイルとして出力します。また，コンパイル後にユーザがアセンブリ言語レベルで内容を確認，修正できるようにアセンブラ・ソース・モジュール・ファイルを出力します。さらに，プリプロセス・リスト，クロス・レファレンス・リスト，エラー・リストなどのリスト・ファイルを出力します。

コンパイル・エラーがある場合，エラー・メッセージをコンソール，エラー・リスト・ファイルなどに出力します。エラーがある場合は，エラー・リスト・ファイル以外の各種ファイルは出力されません。

4.1 Ｃコンパイラの入出力ファイル

本Ｃコンパイラの入出力ファイルを“表４－１　Ｃコンパイラの入出力ファイル”に示します。

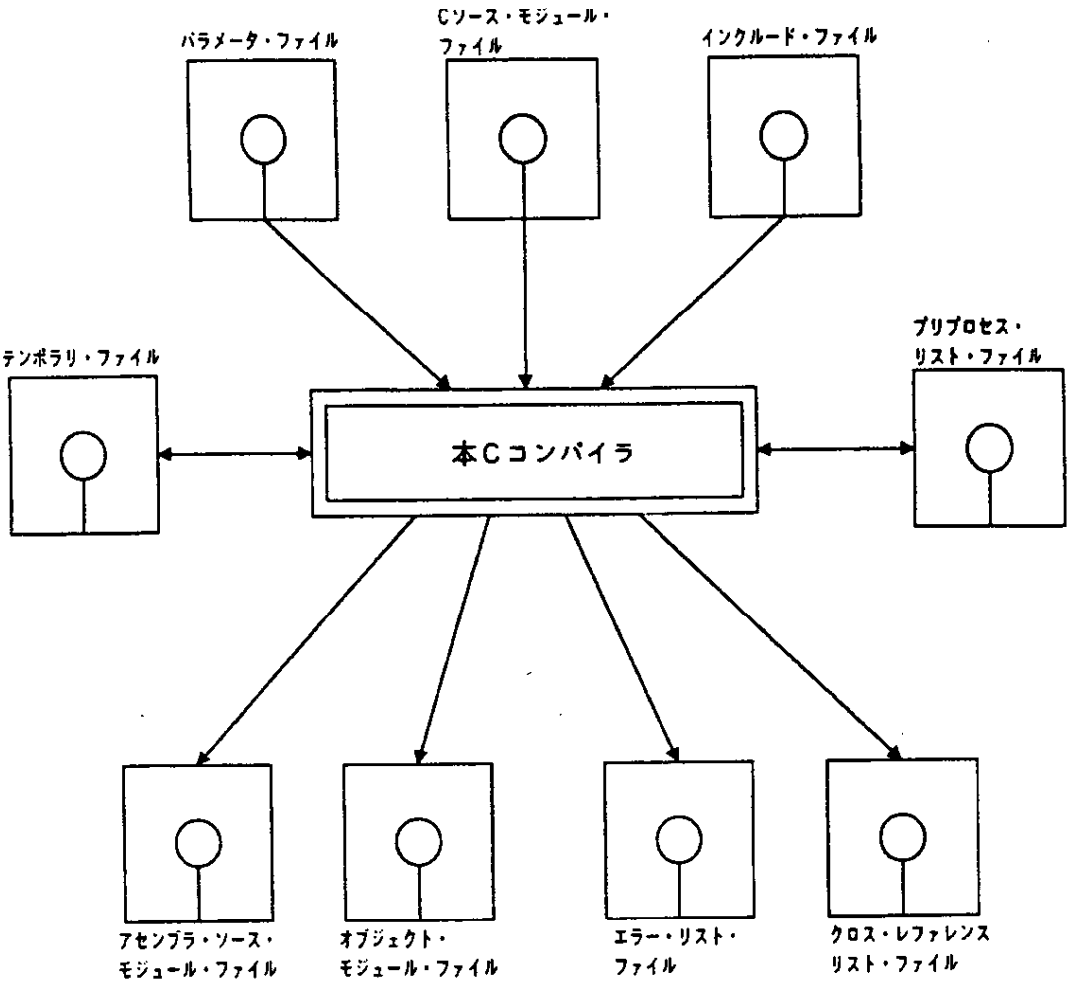
表４－１　Ｃコンパイラの入出力ファイル（１／２）

種　類	ファイル名	説　明	ﾃﾞｨﾌｫｰﾙﾄ・ﾌｧｲﾙ・ﾀｲﾌﾟ
入力ファイル	Ｃソース・モジュール・ファイル	・７８Ｋシリーズの Ｃ言語で記述されたソース・モジュール・ファイルです。 ・ユーザ作成ファイルです。	Ｃ
	インクルード・ファイル	・Ｃソース・モジュール・ファイルで参照するファイルです。 ・７８Ｋシリーズの Ｃ言語で記述されたファイルです。 ・ユーザ作成ファイルです。	Ｈ
	パラメータ・ファイル	・Ｃコンパイラ実行の際にコマンド行で指定不可能な多数のコマンドを指定したいときにユーザがエディタで作成するファイルです。	ＰＣＣ

表４－１　Ｃコンパイラの入出力ファイル（２／２）

種　類	ファイル名	説　　明	ﾃﾞｨﾌｫｰﾙﾄ・ﾌｧｲﾙ・ﾀｲﾌﾟ
出力ファイル	オブジェクト・モジュール・ファイル	・機械語情報と機械語の配置アドレスに関する再配置情報およびシンボル情報を含んだバイナリ・イメージ・ファイルです。	ＲＥＬ
	アセンブラ・ソース・モジュール・ファイル	・コンパイル結果のオブジェクト・コードのＡＳＣＩＩイメージ・ファイルです。	ＡＳＭ
	プリプロセス・リスト・ファイル	・＃include等のプリプロセス命令を処理した結果のリスト・ファイルです。 ・ＡＳＣＩＩイメージ・ファイルです。	ＰＰＬ
	クロス・レファレンス・リスト・ファイル	・Ｃソース・モジュール・ファイル中で使用されている関数名，変数名の情報が入ったリスト・ファイルです。	ＸＲＦ
	エラー・リスト・ファイル	・ソース・ファイルとコンパイル・エラー・メッセージを内容とするリスト・ファイルです。	ＥＣＣ ＣＥＲ ＨＥＲ ＥＲ
入出力ファイル	テンポラリ・ファイル	・コンパイルのための，中間ファイルです。 ・コンパイル正常終了時には正式な名前にリネームされ，異常終了時には削除されます。	＄ｎｎ （ファイル名固定）

図４－１　Ｃコンパイラの入出力ファイル



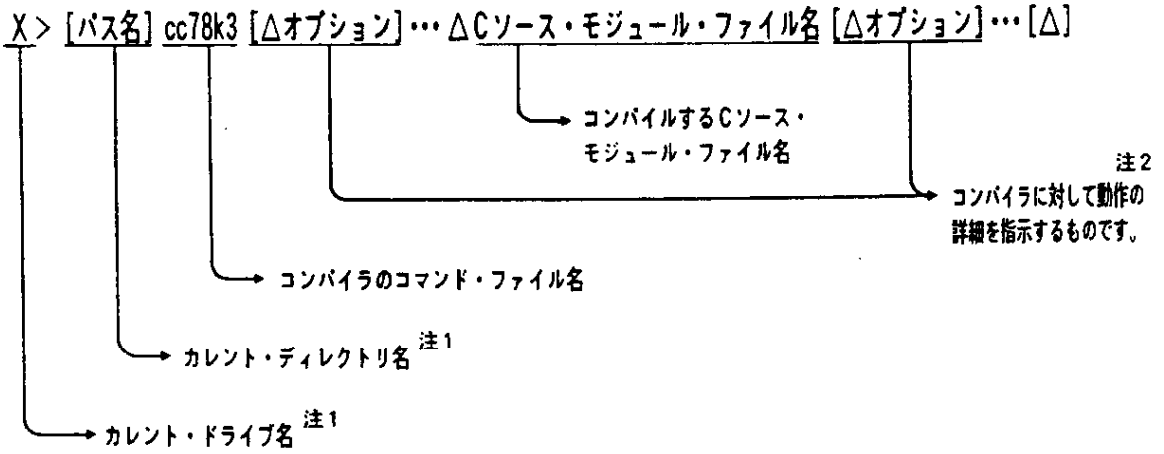
備考　コンパイル・エラーがある場合，エラー・リスト・ファイル以外の各種ファイルは出力されません。テンポラリ・ファイルは，コンパイル正常終了時に正式な名前にリネームされます。また，異常終了時には削除されます。

4.2 Cコンパイラの起動方法

4.2.1 Cコンパイラの起動

本Cコンパイラの起動には，次の２つの方法があります。

(1) コマンド行での起動



例　A>cc78k3 -c310 prime.c -a -p -x -e

注1. ＭＳ－ＤＯＳのバージョン２．１１では，コンパイラのコマンド・ファイル（実行形式ファイル）は，カレント・ドライブ／カレント・ディレクトリに格納されていなければなりません。

2. 複数のコンパイラ・オプションを指定する場合には，おのおののコンパイラ・オプション間を空白で区切ります。オプションの記述は英大文字，英小文字のいずれでも構いません。

詳細については，“第５章　コンパイラ・オプション”をご覧ください。

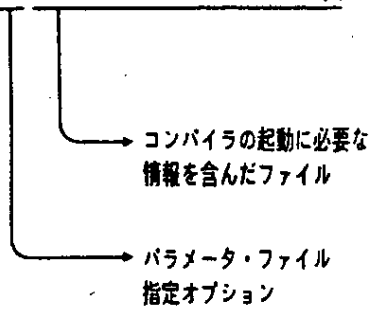
(2) パラメータ・ファイルによる起動

コンパイル時に複数のオプションを入力する際に，コマンド行で起動に必要な情報を指定しきれない場合，また同じ指定を何回も繰り返すことがあります。このようなときにパラメータ・ファイルを使用してコンパイルを行います。

パラメータ・ファイルを使用する場合は，パラメータ・ファイル指定オプションを起動行の中で指定してください。

パラメータ・ファイルによる起動方法は，次のようになります。

X＞cc78k3 [△ [Cソース・モジュール・ファイル]] △－fパラメータ・ファイル名



・パラメータ・ファイルは，エディタで作成します。

・パラメータ・ファイル内でのオプション記述規則

[[[△] オプション [△オプション] ･･･ [△] △]] ･･･

・ソース・ファイル名は，１つのみ指定できます。

・ソース・ファイル名は，オプションの後に記述することも可能です。

・パラメータ・ファイルには，コマンド行で指定すべきすべてのコンパイラ・オプション，出力ファイル名を記述します。

例　パラメータ・ファイルＰＲＩＭＥ．ＰＣＣをエディタで作成します。

・ＰＲＩＭＥ．ＰＣＣの内容

```
sample¥prime.c -c310 -aprime.asm
-e -x
```

・パラメータ・ファイルＰＲＩＭＥ．ＰＣＣを使用してＣコンパイラを起動します。

```
A>cc78k3 -fprime.pcc
```

4.2.2 実行開始・終了メッセージ

(1) 実行開始メッセージ

コンパイラが起動すると，コンソールに実行開始メッセージが表示されます。

```
78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx
```

(2) 実行終了メッセージ

コンパイルの結果，コンパイル・エラーが検出されなかった場合，コンパイラは，次のメッセージをコンソールに出力して制御をＯＳに戻します。

```
Compilation complete,     0 error(s) and     0 warning(s) found.
```

コンパイルの結果，コンパイル・エラーが検出された場合，コンパイラは，エラー・メッセージとエラーの数をコンソールに出力して制御をＯＳに戻します。

```
SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

コンパイル中にコンパイラ処理継続が不可能な致命的エラーが検出された場合，コンパイラはメッセージをコンソールに出力してコンパイルを中止し，制御をＯＳに戻します。

例１

```
A>cc78k3 -c310 -e sample.c

uCOM-78K/III Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

A006 File not found 'SAMPLE.C'
Program aborted
```

この例では，ドライブＡに存在しないソース・ファイルを指定したためにエラーとなり，コンパイルが中止されました。

例２

```
A>cc78k3 -c310 -e sample¥prime.c -m

uCOM-78K/III Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

A018 Option in not recognized '-m'
Program aborted
```

この例では，存在しないコンパイラ・オプションを入力したためにエラーとなりコンパイルが中止されました。

コンパイラがエラー・メッセージを出力してコンパイルを中止した場合には，そのエラー・メッセージの原因を“第９章　エラー・メッセージ”で調べて対処してください。

4.3 Cコンパイラのオプション

Cコンパイラを起動した際にコンパイラ・オプションを指定することができます。コンパイラ・オプションは，コンパイラの動作に対して細かい指示を与えるものです。

コンパイラ・オプションは，単一ではなく複数を同時に指定することができます。ユーザは，目的に合わせてコンパイラ・オプションを選択し，効率の良い作業を行うことができます。

コンパイラ・オプションの種類，指定方法および優先順位については“第５章　コンパイラ・オプション”をご覧ください。

4.4 最適化

本Cコンパイラでは，効率の良いオブジェクト・モジュール・ファイルを生成するために，最適化を行います。サポートする最適化手法を“表４－２　最適化手法”で示します。

表４－２　最適化手法（１／２）

フェーズ		内　　容	例
構文解析部	①	定数計算のコンパイル時実行	a=3*5; → a=15;
	②	論理式の部分評価による真／偽の判定	0&& (a||b) → 0 1|| (a&&b) → 1
	③	ポインタ，配列などのオフセット計算	オフセットをコンパイル時に計算します。
コード生成部	④	レジスタ管理	使用していないレジスタを，有効に利用します。
	⑤	ターゲットＣＰＵの持つ特殊な命令の利用	a=a+1; → inc命令を使用します。 配列要素の代入に転送命令を使用します。
	⑥	短い命令の利用	同じ動作をする命令があった場合，バイト数の短い方を利用します。 mov a,#0 または xor a,a （デバイスによって異なります）
	⑦	ロング・ジャンプ命令のショート・ジャンプ命令への変更	出力された中間コードを再操作して行います。

備考　・①～⑦は，最適化オプションの指定によらず行われます。

表４－２　最適化手法（２／２）

フェーズ		内　　容	例
オプティマイザ	⑧	共通部分式の削除	a=b+c;　→　a=b+c; d=b+c+e;　　d=a+e;

	⑨	命令のループの外への移動	for(i=0;i<10;i++)
			{
			...
			a=b+c;
			...
			}
			↓
			a=b+c;
			for(i=0;i<10;i++)
			{
			...
			...
			}

	⑩	無用命令の削除	a=a;　→　削除 a=b; の後，a が　→　削除 参照されません。 （a はオートマチック変数）
	⑪	複写の削除	a=b; c=a+d;　→　c=b+d これ以降，a が 参照されません。 （a はオートマチック変数）
	⑫	式の演算順序の変更	レジスタに演算結果を残しておいて有効となる演算を先に実行します。
	⑬	記憶装置の割付け （テンポラリ変数）	局所的に使われる変数をレジスタに割付けます。
	⑭	のぞき穴式最適化	特殊パターンの置き換え 例　a*1　→　a，a+0　→　a
	⑮	演算の強さの軽減	例　a*2　→　a+a，a<<1
	⑯	記憶装置の割付け （レジスタ変数）	アクセスの速いメモリへデータを割付けます。 例　レジスタ，saddr

備考　・⑧～⑬の最適化は，最適化オプションが指定された場合に行われます。

・⑭～⑮は，最適化オプションの指定によらず行われます。

・⑯は，Ｃソース・プログラム中にregister宣言がある場合，または最適化

オプションの指定がある場合に行います。

・⑧～⑬，⑯は，将来サポート予定のものです。

・ＣＣ７８Ｋ２では，最適化オプションの指定により，⑭，⑮のより高度な最適化を行います。

4.5 プログラムのＲＯＭ化

プログラムのＲＯＭ化を行う方法について説明します。これは，初期値あり外部変数などの初期値をＲＯＭに配置しておき，システム実行時にＲＡＭにコピーする機能です。

4.5.1 コンパイル時

コンパイル時に－Ｒオプションを指定することにより，コンパイラの出力したオブジェクト・プログラムのＲＯＭ化が可能となります。ディフォールトでＲＯＭ化指定がされているので，－Ｒオプションは省略可能です。そのため，ＲＯＭ化指定を無効にする場合は，－ＮＲオプションを指定します。

4.5.2 リンク時

リンク時には，スタートアップ・モジュール・ファイルとオブジェクト・モジュール・ファイルをリンクしなければなりません。スタートアップ・モジュール・ファイルは，オブジェクト・プログラムの初期化を行います。また，ＲＯＭ化を指定したときは，ＲＯＭ化対応用のスタートアップ・モジュール・ファイルに替える必要があります（次に示すのは，デバイスがμＰＤ７８３１０の場合です）。

(1) ＲＯＭ化を行う場合

ｃｓ３１０ｒ．ｒｅｌ……デバイス種別３１０用スタートアップ・モジュール・ファイル（ＲＯＭ化対応）。初期化データのコピー・ルーチンを含み，初期化データの開始を示します。

ｒｏｍ３１０．ｒｅｌ……初期化データの終端を示します。

・リンク順序はｃｓ３１０ｒ．ｒｅｌが先頭で，ｒｏｍ．ｒｅｌを最後とします。

(2) ＲＯＭ化を行わない場合

ｃｓ３１０．ｒｅｌ……デバイス種別３１０用スタートアップ・モジュール・ファイル。

・リンク順序は任意とします。

ユーザ・プログラムを実行すると，スタートアップ・モジュールから実行されます。スタート・アドレスには，_@cstartというラベル（シンボル）が付けられます。

4.6 実行時エラー・チェック

ソース・ディバッガを起動をしたときに，各種のエラーが生じてユーザが意図していなかった動作をすることがあります。そのため，コンパイラの起動の際に－Ｌオプションを指定して，通常のオブジェクト・コードに加え，エラーのチェックを行うオブジェクト・コードを出力させます。

4.6.1 エラー処理ルーチン

エラーが発見されると，エラー処理ルーチンが呼ばれます。このルーチンは，前処理と後処理を持った永久ループです。これによって，ブレークした時のアドレスを調べれば，どの種類のエラーが起きていたのか知ることが可能となります。エラー処理ルーチンの種類を“表４－３　エラー処理ルーチン”で示します。

エラー処理ルーチンの詳細については，“8.4　エラー処理ルーチン”を参照してください。

表４－３　エラー処理ルーチンの種類

チェック項目	ルーチン名	機　　能
初期化データ	errini.asm	ＲＯＭ領域に割り当てられた初期化データ・セグメントとＲＡＭ領域のセグメント・サイズの不一致を検出した場合に呼ばれます。
スタック・チェック	errstk.asm	スタック・オーバフローが生じる場合に呼ばれます。
０除算	errdiv.asm	０除算が行われる場合に呼ばれます。
ポインタ・アクセス	errptr.asm	ポインタを使って不正な領域をアクセスしようとした場合に呼ばれます。
オーバフロー	errovf.asm	各種の演算によってオーバフローが生じる場合か，生じた場合に呼ばれます。
ｓｆｒアクセス	errsfr.asm	sfrに対して不正な呼び出し，書き込みが行われる場合に呼ばれます。

4.6.2 エラー・チェック・ライブラリ名

エラー・チェックのオプションを指定した場合，ライブラリ名は次のようになります。

(1) ランタイム・ライブラリ

エラー・チェックなしのランタイム・ライブラリ名はシンボルの先頭に@@を付加します。エラー・チェックありの場合はエラー・チェックのパターンによりシンボルの先頭を以下の文字に変更したものとします。

エラー・チェックなし	@@関数名
0除算	?@関数名
0除算＋オーバフロー	?_関数名
オーバフロー	@_関数名
スタック・チェック	_@関数名
sfrアクセス	??関数名

（sfrアクセスでは，エラー・チェックなしのライブラリはありません）

sfr.bitへの代入	例	??asto8
sfr　への代入		??asto8
sfrp　への代入		??asto16

これ以外のsfrに対する演算のエラー・チェックは行いません。

sfrのアドレスと書き込もうとするデータをレジスタやsaddr領域に格納し，各エラー・チェック・ルーチンをコールします。

(2) 標準ライブラリ

エラー・チェックなしの標準ライブラリ名はシンボルの先頭に_を付加します。エラー・チェックありの場合には，エラー・チェックのパターンによりシンボルの先頭に以下の文字を付加し，文字長を８文字にカットします（シンボル名長指定オプションにより，認識するシンボル名長が３０文字になった場合でも同じです）。

エラー・チェックなし	_関数名
スタック・チェック	_@関数名
スタック・チェック＋ポインタ・アクセス	_?関数名
０除算	?@関数名
０除算＋オーバフロー	?_関数名
ポインタ・アクセス	@?関数名
オーバフロー	@_関数名

(3) ユーザ作成の関数

エラー・チェックの有無に関わらずシンボルの先頭に_を付加します。

保守／廃止

第５章　コンパイラ・オプション

5.1 コンパイラ・オプションの種類

コンパイラ・オプションは，コンパイラの動作に細かい指示を与えるものです。コンパイラ・オプションは，１９種類のオプションに分類できます。

表５－１　コンパイラ・オプション（１／３）

項番	分　類	ｵﾌﾟｼｮﾝ	説　明
1	デバイス種別指定	－Ｃ	ターゲット・デバイスの種別を指定します。
2	オブジェクト・モジュール・ファイル作成指定	－Ｏ	オブジェクト・モジュール・ファイルの出力を指定します。
3	シンボル名長指定	－Ｓ	シンボル名の長さを拡張します。
4	シンボル名ケース指定	－ＣＡ	シンボル名のケースを区別しません。
5	ＲＯＭ化指定	－Ｒ	ＲＯＭ化可能なオブジェクト・モジュールを作成します。
6	最適化指定	－Ｑ	効率よいオブジェクトを生成します。
7	ディバグ情報出力指定	－Ｇ	ディバグ情報をオブジェクト・モジュー・ファイルへ出力します。
8	実行時エラー・チェック指定	－Ｌ	オブジェクト・モジュールにエラー・チェック・ライブラリを付加します。
9	プリプロセス・リスト・ファイル作成指定	－Ｐ	プリプロセス・リストの出力を指定します。
		－Ｋ	プリプロセス・リストに対する処理を指定します。

表５－１　コンパイラ・オプション（２／３）

項番	分　類	ｵﾌﾟｼｮﾝ	説　明
１０	プリプロセス指定	－Ｄ	#define文と同様のマクロ定義を行います。
		－Ｕ	#undef文と同様にマクロ定義を無効にします。
		－Ｉ	インクルード・ファイルを指定されたディレクトリから入力します。
１１	アセンブラ・ソース・モジュール・ファイル作成指定	－Ａ	アセンブラ・ソース・モジュール・ファイルを出力します。
		－ＳＡ	Ｃソースを付加したアセンブラ・ソース・モジュール・ファイルを出力します。
１２	エラー・リスト・ファイル作成指定	－Ｅ	エラー・リスト・ファイルを出力します。
		－ＳＥ	Ｃソースを付加したエラー・リスト・ファイルを出力します。
１３	クロス・レファレンス・リスト作成指定	－Ｘ	クロス・レファレンス・リストを出力します。
１４	リスト形式指定	－ＬＷ	リストの１行に印字する文字数を変更します。
		－ＬＬ	リストの１ページに印字する行数を変更します。
		－ＬＴ	タブの展開文字数を変更します。
		－ＬＦ	リスト・ファイルの最後に改ページ・コードを付加します。
１５	ワーニング出力指定	－Ｗ	コンソールへのワーニング・メッセージの出力を指定します。
１６	実行状態表示指定	－Ｖ	コンソールへ実行状態を出力します。
１７	パラメータ・ファイル指定	－Ｆ	入力ファイル名，オプションを指定したファイルより入力します。

表５－１　コンパイラ・オプション（３／３）

項番	分　類	ｵﾌﾟｼｮﾝ	説　明
１８	テンポラリ・ファイル作成パス指定	－Ｔ	テンポラリ・ファイルを指定したパスに作成します。
１９	ヘルプ指定	－－	コンソールへヘルプ・メッセージを出力します。

以上の表は，コンパイラ・オプションを紹介するための表です。実際にコンパイラ・オプションを使用する場合には，“付録Ｃ　コンパイラ・オプション一覧”をご覧ください。“付録Ｃ　コンパイラ・オプション一覧”には，記述形式や他のオプションとの関係などが記述されています。

5.2 コンパイラ・オプションの指定方法

コンパイラ・オプションは，次の方法で指定できます。

(1) Ｃコンパイラを起動するときにコマンド行で指定します。

(2) パラメータ・ファイル内で指定します。

上記コンパイラ・オプションの指定方法については，“4.2　Ｃコンパイラの起動方法”をご覧ください。

5.3 コンパイラ・オプションの優先度

次の表に示すアセンブラ／オプションのうち，縦軸のものと横軸のものを同時に２つ以上指定した場合の優先度について説明します。

表５－２　コンパイラ・オプションの優先度

	-NO	-G	-P	-NP	-D	-U	-A	-E	-X	--
-R	×									×
-Q	×									×
-QS	×	×								×
-QZ	×	×								×
-QE	×	×								×
-QJ	×	×								×
-G	×									×
-L	×									×
-K			△	×						×
-D						○				×
-U					○					×
-SA							×			×
-LW			△				△	△	△	×
-LL			△				△	△	△	×
-LT			△				△	△	△	×
-LF			△				△	△	△	×

←横軸

↑

縦軸

【×で記した箇所】

横軸に示したオプションを指定した場合，縦軸に示したオプションは無効となります。

例　A>cc78k3 -c310 -e sample.c -no -r -g

－Ｒ，－Ｇオプションは，無効となります。

【△で記した箇所】

横軸に示したオプションを指定しない場合，縦軸のオプションは無効となります。

例　A>cc78k3 -c310 -e sample.c -p -k

－Ｐオプションが指定されているので－Ｋオプションは有効です。

【○で記した箇所】

横軸のオプションと縦軸のオプションで後に指定したものが優先です。

例　A>cc78k3 -c310 -e sample.c -utest -dtest=1

－Ｄオプションが後に指定されているので－Ｕオプションは無効となり，
－Ｄオプションが優先されます。

また，－Ｏ／－ＮＯオプションのようにオプション名の前にＮを付加することのできるオプションでも後に指定したものが優先となります。

例　A>cc78k3 -c310 -e sample.c -o -no

－ＮＯオプションが後に指定されているので－Ｏオプションは無効となり，－ＮＯオプションが優先されます。

“表５－２コンパイラ・オプションの優先度”に記述されていないオプションは，他のオプションの影響を特に受けません。しかし，ヘルプ指定オプション“－－”が指定された場合には，すべてのオプション指定が無効となります。

5.4 コンパイラ・オプションの説明

以降に，各コンパイラ・オプションの詳細を説明します。

使用例はＣＣ７８Ｋ３のものを使用します。

(1)　デバイス種別指定（－Ｃ）

－Ｃ	デバイス種別指定

記述形式	－Ｃデバイス種別
省略時解釈	省略不可

【機　能】

・－Ｃオプションは，コンパイルの対象となるターゲット・デバイスを指示します。

【用　途】

・必ず指定してください。Ｃコンパイラは，指定されたターゲット・デバイスに対してコンパイルを行い，それに対応したオブジェクト・コードを生成します。

【説　明】

・－Ｃオプションで指定できるターゲット・デバイスと，それに対応するデバイス種別を次に示します。

＜７８Ｋ／０の場合＞

表５－３　デバイス種別（７８Ｋ／０）（１／２）

ターゲット・デバイス名	デバイス種別
μＰＤ７８００１	－Ｃ００１
μＰＤ７８００２	－Ｃ００２
μＰＤ７８０１１	－Ｃ０１１
μＰＤ７８０１２	－Ｃ０１２
μＰＤ７８０１３	－Ｃ０１３
μＰＤ７８０１４，７８Ｐ０１４	－Ｃ０１４
μＰＤ７８０２２	－Ｃ０２２
μＰＤ７８０２３	－Ｃ０２３
μＰＤ７８０２４，７８Ｐ０２４	－Ｃ０２４

表５－３　デバイス種別（７８Ｋ／０）（２／２）

ターゲット・デバイス名	デバイス種別
μＰＤ７８０４２	－Ｃ０４２
μＰＤ７８０４３	－Ｃ０４３
μＰＤ７８０４４，７８Ｐ０４４	－Ｃ０４４

＜７８Ｋ／Ⅱの場合＞

表５－４　デバイス種別（７８Ｋ／Ⅱ）

ターゲット・デバイス名	デバイス種別
μＰＤ７８２１０	－Ｃ２１０
μＰＤ７８２１２	－Ｃ２１２
μＰＤ７８２１３	－Ｃ２１３
μＰＤ７８２１４，７８Ｐ２１４	－Ｃ２１４
μＰＤ７８２１７Ａ	－Ｃ２１７Ａ
μＰＤ７８２１８Ａ，７８Ｐ２１８Ａ	－Ｃ２１８Ａ
μＰＤ７８２２０	－Ｃ２２０
μＰＤ７８２２４，７８Ｐ２２４	－Ｃ２２４
μＰＤ７８２３３	－Ｃ２３３
μＰＤ７８２３４	－Ｃ２３４
μＰＤ７８２３７	－Ｃ２３７
μＰＤ７８２３８，７８Ｐ２３８	－Ｃ２３８
μＰＤ７８２４３	－Ｃ２４３
μＰＤ７８２４４	－Ｃ２４４

＜７８Ｋ／Ⅲの場合＞

表５－５　デバイス種別（７８Ｋ／Ⅲ）

ターゲット・デバイス名	デバイス種別
μＰＤ７８３１０	－Ｃ３１０
μＰＤ７８３１２，７８Ｐ３１２	－Ｃ３１２
μＰＤ７８３１０Ａ	－Ｃ３１０Ａ
μＰＤ７８３１２Ａ，７８Ｐ３１２Ａ	－Ｃ３１２Ａ
μＰＤ７８３２０	－Ｃ３２０
μＰＤ７８３２２，７８Ｐ３２２	－Ｃ３２２
μＰＤ７８３２３	－Ｃ３２３
μＰＤ７８３２４，７８Ｐ３２４	－Ｃ３２４
μＰＤ７８３２７	－Ｃ３２７
μＰＤ７８３２８，７８Ｐ３２８	－Ｃ３２８
μＰＤ７８３３０	－Ｃ３３０
μＰＤ７８３３４，７８Ｐ３３４	－Ｃ３３４
μＰＤ７８３５０	－Ｃ３５０
μＰＤ７８Ｐ３５２	－Ｃ３５２

【注　意】

・－Ｃオプションは，省略不可能なオプションです。ただし，Ｃソース中に次の記述がある場合には，コマンド行での指定を省略できます。

```
#pragma    pc（デバイス種別）
```

・Ｃソース中とコマンド行で異なるデバイス指定をした場合，コマンド行のデバイスを優先します。

【使用例】

・コマンド行で指定します。ターゲット・デバイスはμＰＤ７８３１０です。

```
A>cc78k3 -c310 sample¥prime.c

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・Ｃソース中で指定して，起動させます。

```
#pragma         pc(310)
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;
```

```
A>cc78k3 sample¥prime.c

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

ターゲット・デバイス指定は，コマンド行で省略可能です。

・Cソース中とコマンド行で異なるデバイス指定をし起動させます。

```
#pragma         pc(320)
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;
```

```
A>cc78k3 -c310 sample¥prime.c

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(1) : W832 Duplicated chip specifier
SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and    6 warning(s) found.
```

コマンド行で指定したターゲット・デバイス指定が優先されます。

(2)　オブジェクト・モジュール・ファイル作成指定（－O／－NO）

－O／－NO	オブジェクト・モジュール・ファイル作成指定

記述形式	－O［出力ファイル名］
	－NO
省略時解釈	－O［入力ファイル名．ＲＥＬ］

【機　能】

・－Oオプションは，オブジェクト・モジュール・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

・－NOオプションは，－Oオプションを無効にします。

【用　途】

・オブジェクト・モジュール・ファイルの出力先や出力ファイル名を変更したいときに，－Oオプションを指定します。

・アセンブラ・ソース・モジュール・ファイルの出力のみが目的でコンパイルする場合などに－NOオプションを指定します。これによりコンパイル時間が短縮されます。

【説　明】

・－Oオプションを指定しても，コンパイル・エラーがある場合は，オブジェクト・モジュール・ファイルは出力されません。

・－Oオプションを指定する際にドライブ名を省略すると，カレント・ドライブにオブジェクト・モジュール・ファイルが出力されます。

・－Oと－NOの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－NOと－Oの両オプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -no -o

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

－NOオプションは無効となり，－Oオプションが有効となります。

(3)　シンボル名長指定（－Ｓ／－ＮＳ）

－Ｓ／－ＮＳ　　シンボル名長指定

記述形式	－Ｓ
	－ＮＳ
省略時解釈	－ＮＳ

【機　能】

・－Ｓオプションは，シンボル名の長さを最大３０文字に拡張するよう指示します。

・－ＮＳオプションは，－Ｓオプションを無効にします。

【用　途】

・シンボル名を８文字以上にしたい場合に，－Ｓオプションを使用して認識できる文字数を拡張することができます。

【説　明】

・－Ｓオプションを指定した場合，コンパイラはシンボルを３０文字まで認識します。また，オブジェクトに出力するシンボル情報は先頭に＿を付加して３１文字です。

・－Ｓオプションを省略した場合，コンパイラはシンボルを７文字まで認識します。また，オブジェクトに出力するシンボル情報は先頭に＿を付加して８文字です。

・－ＮＯオプションが指定された場合，コンパイラはシンボルを３０文字まで認識しますが，オブジェクトに出力されるシンボル情報は，－Ｓオプションの省略時と同様に先頭に＿を付加して８文字です。

・－Ｓと－ＮＳの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－Ｓオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c  -a -s


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

・ＰＲＩＭＥ．ＡＳＭを参照します。

```
; 78K/III Series C Compiler Vx.xx Assembler Source
;                                                    Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 sample¥prime.c -a -s
; In-file   : SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem
        PUBLIC  _mark
        PUBLIC  _main
        PUBLIC  _printf
        PUBLIC  _putchar

@@CODE  CSEG
; line     5
; line     8
_main:
        push    hl
        movw    ax,sp
        subw    ax,#08H
        movw    hl,ax
        movw    sp,ax
; line    11
        movw    ax,#00H ; 0
        mov     [hl+1],a        ; count
        xch     a,x
                :
```

(4)　シンボル名ケース指定（－ＣＡ／－ＮＣＡ）

－ＣＡ／－ＮＣＡ　　シンボル名ケース指定

記述形式	－ＣＡ
	－ＮＣＡ
省略時解釈	－ＮＣＡ

【機　能】

・－ＣＡオプションは，シンボル名のケース（大文字，小文字）を区別しないよう指示します。

・－ＮＣＡオプションは，－ＣＡオプションを無効にします。

【用　途】

・－ＣＡオプションは，大文字，小文字の区別をなくす必要がある場合に指定します。

【説　明】

・－ＣＡオプションを指定するとコンパイラは，シンボル名中の小文字を大文字に変換してシンボル情報をオブジェクトに出力します。

・－ＣＡオプションを省略するとコンパイラは小文字を大文字に変換せずにシンボル情報をオブジェクトに出力します。

・－ＣＡ，または－ＮＣＡオプションを指定する場合に，ライブラリを替えます。これについては，“2.1.2　ライブラリ・ファイル”を参照してください。

・－ＣＡと－ＮＣＡの両オプションが同時に指定された場合は，後に指定したものが優先となります。

・－ＮＯオプションを指定すると，－ＣＡオプションは無視されます。

【使用例】

・－ＣＡオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -a -ca


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

・ＰＲＩＭＥ．ＡＳＭを参照します。

```
;78K/III Series C Compiler Vx.xx Assembler Source
;                                                    Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 sample¥prime.c -a -ca
; In-file   : SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem
        PUBLIC  _MARK
        PUBLIC  _MAIN
        PUBLIC  _PRINTF
        PUBLIC  _PUTCHAR

@@CODE  CSEG
; line     5
; line     8
_MAIN:
        push    hl
        movw    ax,sp
        subw    ax,#08H
        movw    hl,ax
        movw    sp,ax
; line    11
        movw    ax,#00H ; 0
        mov     [hl+1],a        ; COUNT
        xch     a,x
                :
```

(5)　ＲＯＭ化指定（－Ｒ／－ＮＲ）

－Ｒ／－ＮＲ　　ＲＯＭ化指定

記述形式	－Ｒ
	－ＮＲ
省略時解釈	－Ｒ

【機　能】

・－Ｒオプションは，ＲＯＭ化可能なオブジェクト・モジュールの作成を指示します。

・－ＮＲオプションは，－Ｒオプションを無効にします。

【用　途】

・オブジェクト・ファイルをＲＯＭ化可能にしたいときに－Ｒオプションを指定します。

【説　明】

・ディフォールトでＲＯＭ化指定がされているので，－Ｒオプションは省略可能です。そのため，ＲＯＭ化指定を無効にする場合は，－ＮＲオプションを指定します。

・オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルのいずれも出力されない場合，－Ｒオプションは無効となります。

・－Ｒと－ＮＲの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－Rオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -a -r


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.    0 error(s) and    5 warning(s) found.
```

・ＰＲＩＭＥ．ＡＳＭを参照します。

```
; 78K/III Series C Compiler Vx.xx Assembler Source
;                                                    Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 sample¥prime.c -a -r
; In-file   : SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem
                :


        movw    ax,hl
        addw    ax,#02H
        movw    sp,ax
        pop     hl
        ret

@@CNST  CSEG
L0012:  DB      '%6d'
        DB      00H
L0018:  DB      0AH
        DB      '%d primes found.'
        DB      00H

@@R_DATA        CSEG            /*  初期化データ用セグメント  */
        DB      (201)

@@DATA  DSEG                    /*  仮データ領域用セグメント  */
_mark:  DS      (201)
        END
```

・－ＮＲオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -a -nr


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・ＰＲＩＭＥ．ＡＳＭを参照します。

```
; 78K/III Series C Compiler Vx.xx Assembler Source
;                                                  Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 sample¥prime.c -a -nr
; In-file   : SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem

                :


        movw    ax,hl
        addw    ax,#02H
        movw    sp,ax
        pop     hl
        ret

@@CNST  CSEG
L0012:  DB      '%6d'
        DB      00H
L0018:  DB      0AH
        DB      '%d primes found.
        DB      00H

@@DATA  DSEG                    /*  初期化データ用セグメント  */
_mark:  DB      (201)
        END
```

(6)　最適化指定（－Ｑ／－ＮＱ）

－Ｑ／－ＮＱ　　最適化指定

記述形式	－Ｑ［最適化種別］	（複数指定可能）
	－ＮＱ	
省略時解釈	－ＮＱ	

【機　能】

・－Ｑオプションは，最適化フェーズを呼び出し効率の良いオブジェクトを生成するよう指示します。

・－ＮＱオプションは，－Ｑオプションを無効にします。

【用　途】

・オブジェクトの実行速度を速めたり，サイズを小さくしたいときに－Ｑオプションを指定します。詳細は“表５－６　最適化種別”を参照してください。

【説　明】

・－Ｑオプションで指定できる最適化種別は次のとおりです。指定できる最適化種別は各シリーズにより異なります。表５－６に最適化種別を示します。

・最適化種別の中には、－Ｇオプションにより無効と見なされるものがあります。詳細は”表５－２　コンパイラ・オプションの優先度”を参照してください。

表５－６　最適化種別

最適化種別	処　理　内　容	シリーズ名		
		0	Ⅱ	Ⅲ
省略	Rのみが指定されたと見なします。(78K/ⅡはZ,R)	○	○	○
S	オブジェクトの実行速度を重視した最適化を行います。	○	○	○
Z	オブジェクトのｻｲｽﾞを重視した最適化を行います。	○	○	○
U	修飾子なしのcharをunsignedと見なします。	○	○	○
C	charに関する演算を符号拡張なしに行います。	○	○	○
R	レジスタ変数をレジスタに加えてsaddrにも割り当てます。	○	○	○
L	関数の前後処理をライブラリ・コールにします。	○	○	○
E	共通部分式の削除と複写の伝播を行います。			○
J	ジャンプ最適化を行います。			○
X	最大の最適化を行います。 -QZCREJと同じ。			○

・最適化種別は複数指定可能です。
・－Ｑオプションを省略した場合，最適化を行いません。
・オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルのいずれも出力されない場合，－Ｑオプションは無効となります。
・－Ｑと－ＮＱの両オプションが同時に指定された場合は，後に指定したものが優先となります。
・－Ｑオプションが同時にいくつか指定された場合，後に指定したものが優先となります。

【使用例】

・修飾子なしのcharをunsignedと見なすよう最適化を行います。

```
A>cc78k3 -c310 sample¥prime.c -qu

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

・次のように－ＱＣと－ＱＲの両オプションを指定すると，－ＱＣオプションは無効になり，－ＱＲオプションが有効となります。

```
A>cc78k3 -c310 sample¥prime.c -qc -qr

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

・－ＱＣ／－ＱＲオプション両方を有効にしたい場合は，次のようにコマンド入力します。

```
A>cc78k3 -c310 sample¥prime.c -qcr
```

(7)　ディバグ情報出力指定（－G／－NG）

－G／－NG　　ディバグ情報出力指定

記述形式	－G
	－NG
省略時解釈	－NG

【機　能】

・－Gオプションは，オブジェクト・モジュール・ファイル中にディバグ情報を付加するよう指示します。

・－NGオプションは，－Gオプションを無効にします。

【用　途】

・－Gオプションを指定しないと，スクリーン・ディバッガへの入力となるオブジェクト・モジュール・ファイルに必要な行番号，シンボル情報が出力されません。したがって，ソース・レベル・ディバッギングを行うときにはリンクするすべてのモジュールを－Gオプション指定でコンパイルします。

【説　明】

・－Gと－NGが同時に指定された場合は，後に指定したものが優先となります。

・オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルのいずれも出力されない場合，－Gオプションは無効となります。

【使用例】

・－Gオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -g

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

(8)　実行時エラー・チェック指定（－Ｌ／－ＮＬ）

－Ｌ／－ＮＬ　　実行時エラー・チェック指定

記述形式	－Ｌ［エラー・チェック種別］	（複数指定可能）
	－ＮＬ	
省略時解釈	－ＮＬ	

【機　能】

・－Ｌオプションは，作成されるオブジェクト・モジュールに実行時エラー・チェック・ライブラリを付加するよう指示します。

・－ＮＬオプションは，－Ｌオプションを無効にします。

【用　途】

・－Ｌオプションは，エラー・チェックの目的に応じて項目を指定します。

【説　明】

・－Ｌオプションで指定できるエラー・チェック種別を次に示します。

表５－７　エラー・チェック種別

ｴﾗｰ･ﾁｪｯｸ種別	内　容
省略	2，P，S，Z，Dのすべてが指定されたものと見なします。
1	乗算，除算のオーバフローのチェックを行います。
2	1に加えてその他の演算のオーバフローのチェックを行います。
P	ポインタを使用して，不正な番地をアクセスしないかをチェックします。
S	使用するスタックが確保可能かをチェックします。
Z	0除算のチェックを行います。
D	sfrに対して，不正な読み出し，書き込みが行われていないかをチェックします。

備考　エラー・チェック種別は，複数指定可能です。

・ソース・ディバッガを起動したときに，各種のエラーが生じてユーザが意図していなかった動作をすることがあります。そのため，コンパイラの起動の際に－Ｌオプションを指定して通常のオブジェクト・コードに加え，エラーのチェックを行うオブジェクト・コードを出力させます。

・オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルのいずれも出力されない場合，－Ｌオプションは無効となります。

・－Ｌと－ＮＬの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・スタック・オーバフローのチェックおよび０除算のチェックを行います。

```
A>cc78k3 -c310 sample¥prime.c -lsz

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 E.pected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

(9) プリプロセス・リスト・ファイル作成指定（－P／－NP，－K／－NK）

－P／－NP	プリプロセス・リスト・ファイル作成指定

記述形式	－P［出力ファイル名］
	－NP
省略時解釈	－NP

【機　能】

・－Pオプションは，プリプロセス・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

・－NPオプションは，－Pオプションを無効にします。

【用　途】

・プリプロセス・リスト・ファイルの出力先や出力ファイル名を変更したいときに－Pオプションを指定します。

・アセンブラ・ソース・モジュール・ファイルの出力のみが目的でコンパイルする場合などに－NPオプションを指定します。これによりコンパイル時間が短縮されます。

【説　明】

・－Pオプションを指定する際に出力ファイル名を省略するとプリプロセス・リスト・ファイル名は，‘入力ファイル名．ＰＰＬ’となります。

・－Pオプションを指定する際にドライブ名を省略するとカレント・ドライブにプリプロセス・リスト・ファイルが出力されます。

・－Pと－NPの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・プリプロセス・リスト・ファイルＳＡＭＰＬＥ．ＰＰＬを出力します。

```
A>cc78k3 -c310 sample¥prime.c -psample.ppl


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

記述形式	－K［処理種別］	（複数指定可能）
	－NK	
省略時解釈	－KFLN	

【機　能】

・－Kオプションは，プリプロセス・リストに対する処理を指示します。

・－NKオプションは，－Kオプションを無効にします。

【用　途】

・プリプロセス・リスト・ファイルを出力する際にコメントを削除したり，定義の展開を参照するときに指定します。

【説　明】

・－Kオプションで指定できる処理種別を次に示します。

表5－8　－Kオプションの処理種別

処理種別	内　容
省　略	F，L，Nが指定されたものと見なします。
C	コメントの削除
D	#defineの展開
F	#if，#ifdef，#ifndefの条件コンパイル
I	#includeの展開
L	#lineの処理
N	行番号とページング処理

備考　処理種別は，複数指定可能です。

・－Pオプションが指定されない場合は，－Kオプションは無効となります。

・－Kと－NKの両オプションが同時に指定された場合は，後に指定したものが優先です。

・－Kオプションが同時にいくつか指定された場合は，後に指定したものが優先となります。

【使用例】

・プリプロセス・リストPRIME．PPLのコメントの削除，行番号およびページング処理を行います。

```
A>cc78k3 -c310 sample¥prime.c -p -kcn

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・PRIME．PPLを参照します。

```
/*
78K/III Series C Compiler Vx.xx Preprocess List
                              Date:xx xxx xxxx Page:   1

Command    : -c310 sample¥prime.c -p -kcn
In-file    : SAMPLE¥PRIME.C
PPL-file   : PRIME.PPL
Para-file :
*/

    1 : #define TRUE    1
    2 : #define FALSE   0
    3 : #define SIZE    200
    4 :
    5 : char    mark[SIZE+1];
    6 :
    7 : main()
    8 : {
    9 :         int i, prime, k, count;
   10 :
   11 :         count = 0;
   12 :
                ⋮
```

・－KNと－KCの両オプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -p -kn -kc


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

・PRIME．PPLを参照します。

```
/*
78K/III Series C Compiler Vx.xx Preprocess List       Date:xx xxx xxxx Page:   1

Command   : -c310 sample¥prime.c -p -kn -kc
In-file   : SAMPLE¥PRIME.C
PPL-file  : PRIME.PPL
Para-file :
*/

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;

        count = 0;

        for ( i = 0 ; i <= SIZE ; i++)
                mark[i] = TRUE;
        for ( i = 0 ; i <= SIZE ; i++) {
                if (mark[i]) {
                        prime = i + i + 3;
                        printf("%6d",prime);
                        count++;
                                if((count%8) == 0) putchar('¥n');
                        for ( k = i + prime ; k <= SIZE ; k += prime)
                                mark[k] = FALSE;
                }
```

－KNオプションは無効になり，－KCオプションが有効となります。

(10)　プリプロセス指定（－D／－ND，－U／－NU，－I）

－D／－ND　　プリプロセス指定

記述形式	－Dマクロ名＝［定義名］［，マクロ名［＝定義名］］･･･ （複数指定可能） －ND
省略時解釈	Cソース・モジュール・ファイル中のマクロ定義のみを有効とします。

【機　能】

・－Dオプションは，Cソース中の＃define文と同様のマクロ定義を行うよう指示します。

・－NDオプションは，－Dオプションを無効にします。

【用　途】

・特定の定数をあるマクロ名にすべて置き換えたいときに指定します。

【説　明】

・各定義を‘，’で区切ることにより1度に複数のマクロ定義を行うことができます。

・‘＝’と‘，’の前後には空白を許しません。

・定義名が省略されると名前は‘1’として定義されます。

・－Dと－Uの両オプションで同じマクロ名が指定された場合，後者優先となります。

・－Dと－NDの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

```
A>cc78k3 -c310 sample¥prime.c -dtest.time=10

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

記述形式	－Uマクロ名［，マクロ名］･･･　　（複数指定可能）
	－NU
省略時解釈	－Dで指定したマクロ定義を有効とします。

【機　能】

・－Uオプションは，Cソース中の＃undef文と同様にマクロ定義を無効にするよう指示します。

・－NUオプションは，－Uオプションを無効にします。

【用　途】

・－Dオプションで定義されたマクロ名を無効にするときに指定します。

【説　明】

・各マクロ名を‘，’で区切ることにより1度に複数のマクロ定義を無効にすることができます。また，‘，’の前後には空白を許しません。

・－Uオプションで無効にできるマクロ定義は，－Dオプションで定義されたものです。Cソース・モジュール・ファイル中に＃defineによって定義されたマクロ名やコンパイラの持つシステム・マクロ名は，－Uオプションで無効にできません。

・－Dと－Uの両オプションで同じマクロ名が指定された場合は，後者が優先となります。

・－Uと－NUの両オプションが同時に指定された場合は，後に指定したものが優先です。

【使用例】

・－D，－Uオプションで同名のマクロを指定します。

```
A>cc78k3 -c310 sample¥prime.c -dtest -utest

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

記述形式	－Ｉディレクトリ［，ディレクトリ］･･･　（複数指定可能）
省略時解釈	・環境変数ＩＮＣ７８Ｋ３により指定されたディレクトリ ・ソース・ファイルのあるディレクトリ

【機　能】

・Ｃソース中の＃include文で指定されたインクルード・ファイルを指定されたディレクトリから入力するよう指示します。

【用　途】

・インクルード・ファイルをあるディレクトリから検索したいときに指定します。

【説　明】

・‘，’で区切ることにより，１度に複数のディレクトリを指定することができます。

・‘，’の前後には空白を入れることはできません。

・－Ｉに続いてディレクトリが複数指定されるか，あるいは－Ｉオプションが複数指定された場合，指定された順番に＃includeで指定したファイルを検索します。その後，省略時解釈と同じ順序で検索します。

【使用例】

・－Ｉオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -ib:,b:¥sample

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

(11)　アセンブラ・ソース・モジュール・ファイル作成指定

（－A／－NA，－SA／－NSA）

－A／－NA	アセンブラ・ソース・モジュール・ファイル作成指定

記述形式	－A［出力ファイル名］
	－NA
省略時解釈	－NA

【機　能】

・－Aオプションは，アセンブラ・ソース・モジュール・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

・－NAオプションは，－Aオプションを無効にします。

【用　途】

・アセンブラ・ソース・モジュール・ファイルの出力先や出力ファイル名を変更したいときに－Aオプションを指定します。

【説　明】

・ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定することができます。

・－Aオプションを指定する際に出力ファイル名を省略するとアセンブラ・ソース・モジュール・ファイル名は，‘入力ファイル名．ＡＳＭ’となります。

・－Aオプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブラ・ソース・モジュール・ファイルが出力されます。

・－Aと－ＳＡの両オプションが同時に指定された場合は，－ＳＡオプションは無視されます。

・－Aと－ＮＡの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・アセンブラ・ソース・モジュール・ファイルＳＡＭＰＬＥ．ＡＳＭを作成します。

```
A>cc78k3 -c310 sample¥prime.c -asample.asm

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・アセンブラ・ソース・モジュール・ファイルをプリンタに出力します。

```
A>cc78k3 -c310 sample¥prime.c -aprn

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

－ＳＡ／－ＮＳＡ　アセンブラ・ソース・モジュール・ファイル作成指定

記述形式	－ＳＡ［出力ファイル名］
	－ＮＳＡ
省略時解釈	－ＮＳＡ

【機　能】

・－ＳＡオプションは，アセンブラ・ソース・モジュール・ファイルにコメントとしてＣソースを付加します。また，その出力先や出力ファイル名を指示します。

・－ＮＳＡオプションは，－ＳＡオプションを無効にします。

【用　途】

・アセンブラ・ソース・モジュール・ファイルとＣソース・モジュール・ファイルを一緒に出力したいときに－ＳＡオプションを指定します。

【説　明】

・ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定することができます。

・－ＳＡオプションを指定する際に出力ファイル名を省略するとアセンブラ・ソース・モジュール・ファイル名は，‘入力ファイル名．ＡＳＭ’となります。

・－ＳＡオプションを指定する際にドライブ名を省略すると，カレント・ドライブにアセンブラ・ソース・モジュール・ファイルが出力されます。

・－ＳＡと－Ａの両オプションが同時に指定された場合は，－ＳＡオプションは無視されます。

・－ＳＡと－ＮＳＡの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－ＳＡオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -sa


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

・ＰＲＩＭＥ．ＡＳＭを参照します。

```
; 78K/III Series C Compiler Vx.xx Assembler Source
;                                                    Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 sample¥prime.c -sa
; In file   : SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem
        PUBLIC  _mark
        PUBLIC  _main
        PUBLIC  _printf
        PUBLIC  _putchar

@@CODE  CSEG
; line     1 : #define TRUE    1
; line     2 : #define FALSE   0
; line     3 : #define SIZE    200
; line     4 :
; line     5 : char    mark[SIZE+1];
; line     6 :
; line     7 : main()
; line     8 : {
_main:
        push    hl
        movw    ax,sp
        subw    ax,#08H
                :
```

コメントとしてＣソースが付加されています。

(12)　エラー・リスト・ファイル作成指定（－E／－NE，－SE／－NSE）

－E／－NE　　エラー・リスト・ファイル作成指定

記述形式	－E［出力ファイル名］
	－NE
省略時解釈	－NE

【機　能】

・－Eオプションは，エラー・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

・－NEオプションは，－Eオプションを無効にします。

【用　途】

・エラー・リスト・ファイルの出力先や出力ファイル名を変更したいときに－Eオプションを指定します。

【説　明】

・ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定することができます。

・－Eオプションを指定する際に出力ファイル名を省略するとエラー・リスト・ファイル名は，‘入力ファイル名．ＥＣＣ’となります。

・－Eオプションを指定する際にドライブ名を省略するとカレント・ドライブにエラー・リスト・ファイルが出力されます。

・－W0オプションが指定された場合には，ワーニング・メッセージは出力されません。

・－Eと－NEの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－Ｅオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -e

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

・エラー・リスト・ファイルを参照します。

```
SAMPLE¥PRIME.C(   18) : W745 Expected function prototype
SAMPLE¥PRIME.C(   20) : W745 Expected function prototype
SAMPLE¥PRIME.C(   26) : W622 No return value
SAMPLE¥PRIME.C(   37) : W622 No return value
SAMPLE¥PRIME.C(   44) : W622 No return value

Compilation complete,    0 error(s) and    5 warning(s) found.
```

－ＳＥ／－ＮＳＥ　　エラー・リスト・ファイル作成指定

記述形式	－ＳＥ［出力ファイル名］
	－ＮＳＥ
省略時解釈	－ＮＳＥ

【機　能】

・－ＳＥオプションは，エラー・リスト・ファイルにＣソース・モジュール・ファイルを付加します。また，その出力先や出力ファイル名を指示します。

・－ＮＳＥオプションは，－ＳＥオプションを無効にします。

【用　途】

・エラー・リスト・ファイルとＣソース・モジュール・ファイルを一緒に出力したいときに－ＳＥオプションを指定します。

【説　明】

・ファイル名としてディスク型ファイル名とデバイス型ファイル名を指定することができます。

・－ＳＥオプションを指定する際に出力ファイル名を省略するとエラー・リスト・ファイル名は，‘入力ファイル名．ＣＥＲ’となります。

・－ＳＥオプションを指定する際にドライブ名を省略すると，カレント・ドライブにエラー・リスト・ファイルが出力されます。

・インクルード・ファイルに対しては，ファイル名の指定はできません。インクルード・ファイルのファイル・タイプが‘Ｈ’の場合は‘ＨＥＲ’，‘Ｃ’の場合は‘ＣＥＲ’，それ以外の場合は‘ＥＲ’のファイル・タイプを持つエラー・リスト・ファイルを出力します。

・エラーがなかった場合はＣソースは付加されません。また，インクルード・ファイルに対しては，エラー・リスト・ファイルは作成されません。

・－Ｗ０オプションが指定された場合には，ワーニング・メッセージは出力されません。

・－ＳＥと－ＮＳＥの両オプションが同時に指定された場合は，後に指定したものが優先となります。

【使用例】

・－ＳＥオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -se


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・ＰＲＩＭＥ・ＣＥＲを参照します。

```
/*
78K/III Series C Compiler Vx.xx Error List        Date:xx xxx xxxx Time:xx:xx:xx

Command   : -c310 sample¥prime.c -se
In-file   : SAMPLE¥PRIME.C
Err-file  : PRIME.CER
Para-file :
*/

#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;

        count = 0;

        for ( i = 0 ; i <= SIZE ; i++)
                mark[i] = TRUE;
        for ( i = 0 ; i <= SIZE ; i++) {
                if (mark[i]) {
                        prime = i + i + 3;
                        printf("%6d",prime);
*** WARNING W745 Expected function prototype
                        count++;
                }

/*
Compilation complete.     0 error(s) and     5 warning(s) found.
*/
```

Ｃソースが付加されています。

(13)　クロス・レファレンス・リスト・ファイル作成指定（－X／－NX）

－X／－NX	クロス・レファレンス・リスト・ファイル作成指定

記述形式	－X［出力ファイル名］
	－NX
省略時解釈	－NX

【機　能】

・－Xオプションは，クロス・レファレンス・リスト・ファイルを出力することを指示します。また，その出力先や出力ファイル名を指示します。

・－NXオプションは，－Xオプションを無効にします。

【用　途】

・クロス・レファレンス・リスト・ファイルの出力先や出力ファイル名を変更したいときに－Xオプションを指定します。

【説　明】

・ファイル名としてディスク型とデバイス型ファイル名を指定することができます。

・－Xオプションを指定する際に出力ファイル名を省略するとクロス・レファレンス・リスト・ファイル名は，‘入力ファイル名．ＸＲＦ’となります。

・－Xと－NXの両オプションが同時に指定された場合は，後に指定したものが優先です。

【使用例】

・－Xオプションを指定します。

```
A>cc78k3 -c310 sample¥prime.c -x

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

(14)　リスト形式指定（－ＬＷ，－ＬＬ，－ＬＴ，－ＬＦ）

－ＬＷ　　リスト形式指定

記述形式	－ＬＷ［文字数］
省略時解釈	－ＬＷ１３２　（コンソール出力の場合は８０文字とします。）

【機　能】

・－ＬＷオプションは，各種リスト・ファイルの１行の文字数を指示します。

【用　途】

・各種リスト・ファイルの１行の文字数を変更したいときに－ＬＷオプションを指定します。

【説　明】

・－ＬＷオプションで指定できる文字数の範囲はターミネータ（ＣＲ，ＬＦ）は含めないで次のとおりです（コンソール出力の場合には，８０文字までとなります）。

　　７２　≦　１行に印字する文字数　≦　１３２

・文字数を省略した場合は，１行の文字数は１３２文字となります。

・リスト・ファイル指定が何も指定されない場合，－ＬＷオプションは無効となります。

【使用例】

・－ＬＷオプションを省略した場合のクロス・レファレンス・リスト・ファイルをプリンターに出力します。

```
A>cc78k3 -c310 sample¥prime.c -xprn

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.    0 error(s) and    5 warning(s) found.
```

・ＰＲＩＭＥ．ＸＲＦを参照します。

－ＬＷ　　リスト形式指定

・クロス・レファレンス・リスト・ファイルの１行の文字数を８０文字と指定します。

```
A>cc78k3 -c310 sample¥prime.c -x -lw80


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,    0 error(s) and    5 warning(s) found.
```

・ＰＲＩＭＥ．ＸＲＦを参照します。

```
78K/III Series C Compiler Vx.xx Cross reference List   Date:xx xxx xxxx Page:  1

Command   : -c310 sample¥prime.c -x -lw80
In-file   : SAMPLE¥PRIME.C
Xref-file : PRIME.XRF
Para-file :

ATTRIB MODIFY TYPE    SYMBOL          DEFINE  REFERENCE

EXTERN        array   mark               5      14      16      22
EXTERN        func    main               7
AUTO1         int     i                  9      13      13      13      14
                                                15      15      15      16
                                                17      17      21
AUTO1         int     prime              9      17      18      21      21
AUTO1         int     k                  9      21      21      21      22
AUTO1         int     count              9      11      19      20      25
EXTERN        func    printf            28      18      25
EXTERN        func    putchar           39      20
PARAM         pointer s                 29      36
PARAM         int     i                 30      35
AUTO1         int     j                 32      35
AUTO1         pointer ss                33      36
PARAM         char    c                 40      43
AUTO1         char    d                 42      43
              #define TRUE               1      14
              #define FALSE              2      22
              #define SIZE               3       5      13      15      21
```

記述形式	－ＬＬ［行数］
省略時解釈	－ＬＬ６６（コンソール出力の場合は改ページしません）

【機　能】

・－ＬＬオプションは，各種リスト・ファイルの１ページの行数を指示します。

【用　途】

・各種リスト・ファイルの１ページの行数を変更したいときに－ＬＬオプションを指定します。

【説　明】

・－ＬＬオプションで指定できる行数の範囲は次のとおりです。

２０　≦　１ページに印字する行数　≦　６５５３５

・－ＬＬ０を指定すると，コンソール出力の場合は改ページしません。

・行数を省略した場合は，１ページの行数は６６行となります。

・リスト・ファイル指定が何も指定されない場合，－ＬＬオプションは無効となります。

【使用例】

・クロス・レファレンス・リスト・ファイルの１ページの行数を２０行と指定します。

```
A>cc78k3 -c310 sample¥prime.c -x -ll20

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・ＰＲＩＭＥ．ＸＲＦを参照します。

```
78K/III Series C Compiler Vx.xx Cross reference List
                          Date:xx xxx xxxx Page:   1

Command    : -c310 sample¥prime.c -x -l120
In-file    : SAMPLE¥PRIME.C
Xref-file  : PRIME.XRF
Para-file  :

ATTRIB MODIFY TYPE     SYMBOL          DEFINE  REFERENCE

EXTERN        array    mark               5       14       16       22
EXTERN        func     main               7
AUTO1         int      i                  9       13       13       13       14
 15       15       15       16       17       17
                                                  21
AUTO1         int      prime              9       17       18       21       21


78K/III Series C Compiler Vx.xx Cross reference List
                          Date:xx xxx xxxx Page:   2

AUTO1         int      k                  9       21       21       21       22
AUTO1         int      count              9       11       19       20       25
EXTERN        func     printf            28       18       25
EXTERN        func     putchar           39       20
PARAM         pointer  s                 29       36
PARAM         int      i                 30       35
AUTO1         int      j                 32       35
AUTO1         pointer  ss                33       36
PARAM         char     c                 40       43
AUTO1         char     d                 42       43
              #define  TRUE               1       14
              #define  FALSE              2       22


78K/III Series C Compiler Vx.xx Cross reference List
                          Date:xx xxx xxxx Page:   3

              #define  SIZE               3        5       13       15       21
```

記述形式	－ＬＴ［文字数］
省略時解釈	－ＬＴ８

【機　能】

・－ＬＴオプションは，ソース・モジュール中のＨＴ（Horizontal Tabulation）コードを，各種リスト上でいくつかのブランク（空白）に置き換えて出力する（タビュレーション処理）ための基本となる文字数を指示します。

【用　途】

・－ＬＷオプションで各種リストの１行の文字数を少なく指定した場合などにＨＴコードによるブランクを少なくし，文字数を節約するために－ＬＴオプションを指定します。

【説　明】

・－ＬＴオプションで指定できる文字数の範囲は次のとおりです。

　　０　≦　指定できる文字数　≦　８

・－ＬＴ０を指定した場合，タビュレーション処理は行わず，タブ・コードを出力します。

・文字数を省略した場合は，タブの展開文字数は８文字となります。

・リスト・ファイル指定が何も指定されない場合，－ＬＴオプションは無効となります。

【使用例】

・－ＬＴオプションを省略します。

```
A>cc78K3 -c310 sample¥prime.c -p


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・プリプロセス・リストを参照します。

```
/*
78K/III Series C Compiler Vx.xx Preprocess List
                        Date:xx xxx xxxx Page:   1

Command    : -C310 SAMPLE¥PRIME.C -P
In-file    : SAMPLE¥PRIME.C
PPL-file   : PRIME.PPL
Para-file :
*/

    1 : #define TRUE    1
    2 : #define FALSE   0
    3 : #define SIZE    200
    4 :
    5 : char    mark[SIZE+1];
    6 :
    7 : main()
    8 : {
    9 :         int i, prime, k, count;
   10 :
   11 :         count = 0;
   12 :
   13 :         for ( i = 0 ; i <= SIZE ; i++)
   14 :                 mark[i] = TRUE;
   15 :         for ( i = 0 ; i <= SIZE ; i++) {
   16 :                 if (mark[i]) {
   17 :                         prime = i + i + 3;
   18 :                         printf("%6d",prime);
   19 :                         count++;
   20 :                                 if((count%8) == 0) putchar('¥n');
   21 :                         for ( k = i + prime ; k <= SIZE ; k += prime)
   22 :                                 mark[k] = FALSE;
   23 :                 }
   24 :         }
   25 :         printf("¥n%d primes found.",count);
   26 : }
   27 :
```

・ＨＴコードによるブランクを１に指定します。

```
A>cc78K3 -c310 sample¥prime.c -p -lt1


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.     0 error(s) and     5 warning(s) found.
```

・プリプロセス・リストを参照します。

```
/*
78K/III Series C Compiler Vx.xx Preprocess List
                              Date:xx xxx xxxx Page:   1

Command   : -C310 SAMPLE¥PRIME.C -P -LT1
In-file   : SAMPLE¥PRIME.C
PPL-file  : PRIME.PPL
Para-file :
*/

    1 : #define TRUE 1
    2 : #define FALSE 0
    3 : #define SIZE 200
    4 :
    5 : char mark[SIZE+1];
    6 :
    7 : main()
    8 : {
    9 :  int i, prime, k, count;
   10 :
   11 :  count = 0;
   12 :
   13 :  for ( i = 0 ; i <= SIZE ; i++)
   14 :   mark[i] = TRUE;
   15 :  for ( i = 0 ; i <= SIZE ; i++) {
   16 :   if (mark[i]) {
   17 :    prime = i + i + 3;
   18 :    printf("%6d",prime);
   19 :    count++;
   20 :     if((count%8) == 0) putchar('¥n');
   21 :    for ( k = i + prime ; k <= SIZE ; k += prime)
   22 :     mark[k] = FALSE;
   23 :   }
   24 :  }
   25 :  printf("¥n%d primes found.",count);
   26 : }
   27 :
```

ＨＴコードによるブランクは１つです。

記述形式	－ＬＦ
省略時解釈	なし

【機　能】

・－ＬＦオプションは，各種リスト・ファイルの最後に改ページ・コードを付加することを指示します。

【説　明】

・リスト・ファイル指定が何も指定されない場合，－ＬＦオプションは無効となります。

【使用例】

・－ＬＦオプションを指定します。

```
A>cc78K3 -c310 sample¥prime.c -a -lf

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

(15)　ワーニング出力指定（－W）

－W　　ワーニング出力指定

記述形式　　－W［レベル］ 省略時解釈　　－W 1

【機　能】

・－Wオプションは，ワーニング・メッセージをコンソールに出力することを指示します。

【用　途】

・ワーニング・メッセージをコンソールに出力するかしないかを指定します。または詳細なメッセージを出力させることもできます。

【説　明】

・ワーニング・メッセージのレベルには，次のものがあります。

表５－９　ワーニング・メッセージのレベル

レベル	説　明
0	ワーニング・メッセージを出力しません。
1	通常のワーニング・メッセージを出力します。
2	詳細なワーニング・メッセージを出力します。

・－E，－ＳＥオプションが指定された場合には，エラー・リスト・ファイルにもワーニング・メッセージが出力されます。

・レベル０は，ワーニング・メッセージをコンソール，エラー・リスト・ファイル（－E，－ＳＥ指定時）に出力しないことを指示します。

【使用例】

・－Wオプションを省略した場合のワーニング・メッセージを参照します。

```
A>cc78K3 -c310 sample¥prime.c


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

・ワーニング・メッセージを出力しません。

```
A>cc78K3 -c310 sample¥prime.c -w0


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


Compilation complete,     0 error(s) and     5 warning(s) found.
```

(16)　実行状態表示指定（－Ｖ／－ＮＶ）

－Ｖ／－ＮＶ　　実行状態表示指定

記述形式	－Ｖ
	－ＮＶ
省略時解釈	－ＮＶ

【機　能】

・－Ｖオプションは，現在のコンパイルの実行状態をコンソールに出力します。

・－ＮＶオプションは，－Ｖオプションを無効にします。

【用　途】

・コンパイルの実行状態をコンソールに出力しながら実行を行うときに指定します。

【説　明】

・フェーズ名および処理中の関数名を出力します。

・－Ｖと－ＮＶの両オプションが同時に指定された場合は，後に指定したものが優先です。

【使用例】

・－Ｖオプションを指定します。

```
A>cc78K3 -c310 sample¥prime.c -v

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


Phase:parser
main():
SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
printf():
SAMPLE¥PRIME.C(37) : W622 No return value
putchar():
SAMPLE¥PRIME.C(44) : W622 No return value
Phase:code generator
main():
printf():
putchar():
Phase:list and object output
main():
printf():
putchar():
Compilation complete.     0 error(s) and     5 warning(s) found.
```

(17)　パラメータ・ファイル指定（－Ｆ）

－Ｆ	パラメータ・ファイル指定

記述形式	－Ｆファイル名
省略時解釈	コマンド行上からのみオプション，入力ファイル名の入力が可能

【機　能】

・－Ｆオプションは，オプションあるいは入力ファイル名を指定のファイルから入力することを指示します。

【用　途】

・コンパイル時に複数のオプションを入力するため，コマンド行ではコンパイラの起動に必要な情報を指定しきれないときに－Ｆオプションを指定します。
・繰り返し同じようにオプションを指定しコンパイルする際には，それらをパラメータ・ファイルに記述しておき，－Ｆオプションを指定します。

【説　明】

・パラメータ・ファイルのネストは許されません。
・パラメータ・ファイル中に記述できる文字数の制限はありません。
・空白とタブおよび‘　’をオプションあるいは入力ファイル名の区切りとします。
・パラメータ・ファイル中に記述したオプションあるいは入力ファイル名はコマンド行上のパラメータ・ファイル指定のあった位置に展開されます。
・展開されたオプションの優先順位は後者優先です。
・‘;’および‘#’以降に記述された文字は‘　’の前まですべてコメントと解釈します。

【使用例】

・パラメータ・ファイルＰＲＩＭＥ．ＰＣＣの内容

```
;parameter file
sample¥prime.c -c310 -aprime.asm
-e -x
```

・ＰＲＩＭＥ．ＰＣＣを使用してコンパイルします。

```
A>cc78K3 -fprime.pcc

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx

SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete.    0 error(s) and    5 warning(s) found.
```

(18)　テンポラリ・ファイル作成ディレクトリ指定（－T）

－T	テンポラリ・ファイル 作成ディレクトリ指定

記述形式	－Tディレクトリ
省略時解釈	環境変数ＴＭＰにより指定されたドライブ，ディレクトリに作成します。指定されていない場合は，カレント・ドライブ，カレント・ディレクトリに作成されます。

【機　能】

・－Tオプションは，テンポラリ・ファイルを作成するドライブ，ディレクトリを指示します。

【用　途】

・テンポラリ・ファイルの作成場所を指定できます。

【説　明】

・以前に作成されたテンポラリ・ファイルが存在している場合でも，ファイル保護がされていなければ，次の作成時には上書きします。
・テンポラリ・ファイルは，必要とするメモリ・サイズがある場合は，メモリに展開します。必要とするメモリ・サイズがなくなった場合は，メモリの内容を指定されたディレクトリ下にテンポラリ・ファイルを作成してファイルに書き出し，それ以降のテンポラリ・ファイルに対するアクセスは，メモリ上でなくファイルに対して行います。
・テンポラリ・ファイルは，コンパイル終了時に削除されます。また，ＣＴＲＬ－Ｃを押すことによってコンパイルが中止されたときも，削除されます。

	テンポラリ・ファイル
－T	作成ディレクトリ指定

【使用例】

・テンポラリ・ファイルをディレクトリＴＭＰに出力するよう指定します。

```
A>cc78K3 -c310 sample¥prime.c -ttmp

78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

(19)　ヘルプ指定（－－）

－－　　ヘルプ指定

記述形式	－－
省略時解釈	表示しない

【機　能】

・－－オプションは，ヘルプ・メッセージをコンソールに表示します。

【用　途】

・オプションとその説明が表示されます。Cコンパイラ実行時に参照してください。

【説　明】

・－－オプションを指定すると他のコンパイラ・オプションは，すべて無効となります。

・ヘルプ・メッセージの続き表示をご覧になる場合には，改行キーを入力し，表示を途中で終了する場合には，改行キー以外の文字を入力した後に改行キーを入力してください。

【使用例】

・－－オプションを指定します。

```
A>cc78K3 --


 78K/III Series C Compiler Vx.xx  [xx xxx xx]
    Copyright (C) NEC Corporation xxxx


Usage : CC78K3 [option[...]] input-file [option[...]]

The option is as follows ([] means omissible, ... means repeat).
  -cx        : Select target chip. ( x = chip name ) *Must be specified.
  -o[file]/no    : Create object file / Not.
  -s/ns      : Expand symbol length up to 30 / or symbol length is 7.
  -ca/nca    : Convert alphabet to capital for symbol / Not.
  -r/nr      : Generate ROMable object module / Not.
  -q[y]/nq   : Optimize output code / Not. (default x = r)
               y = s : optimize object speed
               y = z : optimize object size
               y = c : assign char variable without sign expand
               y = l : call function prepare/dispose library.
               y = r : use SADDR area for register variable
               y = u : change plain char to unsigned char
               y = e : common subexpression elimination and copy propagation
               y = j : jump optimization
               y = x : max. optimization(y = zcrej)
  -g/ng      : Output debug information for object file / Not.
:
Usage : CC78K3 [option[...]] input-file [option[...]]
  -l[x]      : Add error check library / Not. (default : x = 2dpsz)
               x = 1 : overflow check (level 1)
               x = 2 : overflow check (level 2)
               x = d : sfr access check
               x = p : illegal pointer access check
               x = s : stack overflow check
               x = z : 0 divide check
  -p[file]/np    : Create the preprocess list file / Not.
  -k[x]/nk   : Specified preprocess list mode / None. (default x = fln)
               x = c : delete comment
               x = d : execute #define
               x = f : execute #if, #ifdef, #ifndef
               x = i : execute #include
               x = l : execute #line
               x = n : add line number and paging
  -dname[=data][,name[=data]]...
        : Define name with data.
  -uname[,name]...
        : Undefine name.
  -idirectory[,directory]...
        : Set include search path.
```

保守／廃止

```
Usage : CC78k3 [oputionl...]] input-file [oputionl...]]
  -a[file]/na    : Create the assembler source file / Not.
  -sa[file]/nsa : Create the assembler source file with the C source / Not.
  -e[file]/ne    : Create the error list file / Not.
  -se[file]/nse : Create the error list file with the C source / Not.
  -x[file]/nx    : Create the cross reference list file / Not.
  -lw[width]     : Specify list file columns per line. (width=72 to 132)
  -ll[length]    : Specify list file lines per pate. (length=0, 20 to 65535)
          0 means no paging.
  -lt[n]      : Expand TAB character for list file. (n=1 to 8)
          Not expand. (n=0)
  -lf/nlf    : Add Form Feed at end of list file / Not.
  -w[n]       : Change warning level. (n=0 to 2)
  -v/nv           : Verbose compile messages / Not.
  -ffile     : Input option or source file name from specified file.
  -tdirectory    : Set temporary directory
  --             : show this message

  DEFAULT ASSIGNMENT
        : -o -r -lw32 -ll66 -nlf -w1
;;
```

第6章　Cコンパイラの出力ファイル

Cコンパイラは，次のファイルを出力します。

・オブジェクト・モジュール・ファイル
・アセンブラ・ソース・モジュール・ファイル
・プリプロセス・リスト・ファイル
・クロス・レファレンス・リスト・ファイル
・エラー・リスト・ファイル

6.1 オブジェクト・モジュール・ファイル

オブジェクト・モジュール・ファイルは，Cソース・プログラムのコンパイル結果のバイナリ・イメージ・ファイルです。

また，ディバグ情報出力指定オプションによりディバグ情報を含みます。

6.2 アセンブラ・ソース・モジュール・ファイル

アセンブラ・ソース・モジュール・ファイルは，Ｃソース・プログラムのコンパイル結果のＡＳＣＩＩイメージのリストで，Ｃソース・プログラムに対応したアセンブリ言語のソース・モジュール・ファイルです。

また，オプションによりコメントとしてＣソース・プログラムを含みます。

【出力形式】

```
  ; 78K/III Series C Compiler V①x.xx Assembler Source
  :                                                    Date:②xxxxx Time:③xxxxx

  ; Command    : ④-c310 sample¥prime.c -sa
  ; C-file     : ⑤SAMPLE¥PRIME.C
  ; Asm-file   : ⑥PRIME.ASM
  ; Para-file : ⑦

  $PROCESSOR(⑧310)
⑨$NODEBUG

⑫         NAME    PRIME
⑫         EXTRN   @@isrem
            :

  ; line   ⑩1  :⑪#define TRUE    1
  ; line   ⑩2  :⑪#define FALSE   0
  ; line   ⑩3  :⑪#define SIZE    200
            :

⑫_main:
⑫         push    hl
⑫         movw    ax,sp
            :
```

【出力項目の説明】（１／２）

項番	内　　容	桁　　数	形　　式
①	バージョン番号	4	‘x.yz’ の形式で表します。
②	日付	１１ （固定）	システム日付。 ‘DD Mmm YYYY’ の形式で表します。
③	時間	8（固定）	システム時間。 ‘HH:MM:SS’ の形式で表します。
④	コマンド行		コマンド行の ‘CC78K3’ 以降を出力します。１行に入らない場合は，超過分を次行の15カラム目から出力します。ただし，１カラム目に ‘;’ を出力します。１個以上の空白タブは１個の空白で置き換えます。

【出力項目の説明】（２／２）

項番	内　　容	桁　　数	形　　式
⑤	Ｃソース・モジュール・ファイル名	最大７８（可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.c’を付加します。１行に入らない場合は，超過分を次行の15カラム目から出力します。ただし，１カラム目に‘;’を出力します。１個以上の空白タブは１個の空白で置き換えます。
⑥	アセンブラ・ソース・モジュール・ファイル名	最大７８（可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.asm’を付加します。１行に入らない場合は，超過分を次行の15カラム目から出力します。ただし，１カラム目に‘;’を出力します。１個以上の空白タブは１個の空白で置き換えます。
⑦	パラメータ・ファイルの内容		パラメータ・ファイルの内容を出力します。１行に入らない場合は，超過分を次行の15カラム目から出力します。ただし，１カラム目に‘;’を出力します。１個以上の空白タブは１個の空白で置き換えます。
⑧	デバイス種別	最大４（可変）	－Ｃオプションで指定された文字列です。‘第５章　コンパイラ・オプション’を参照してください。
⑨	ディバグ情報	最大８（可変）	ＤＥＢＵＧコントロールを出力します。＄ＤＥＢＵＧあるいは＄ＮＯＤＥＢＵＧのいずれかです。
⑩	行番号	５（固定）	Ｃソース・モジュール・ファイルの行番号右詰めゼロ・サプレスの10進数です。
⑪	Ｃソース		入力Ｃソース・イメージ。１行に入らない場合は，超過分を次行の1カラム目から出力します。
⑫	アセンブラ・ソース本体		コンパイルの結果のアセンブラ・ソースを出力します。80カラム目以降も折り返しを行わず出力します。

6.3 エラー・リスト・ファイル

エラー・リスト・ファイルは，コンパイル中に発生したワーニングやエラー・メッセージの集まりでできています。
コンパイラ・オプションを指定することによりエラー・リスト中にＣソース・プログラムを付加することができます。Ｃソース・プログラムを含むエラー・リスト・ファイルはＣソース・プログラムを修正し，リスト・ヘッダ等のコメントを削除することによりＣソース・モジュール・ファイルとして使用することができます。

6.3.1 Ｃソース付きのエラー・リスト・ファイル

【出力形式】

```
  /*
  78K/III Series C Compiler V①x.xx Error List          Date:②xxxxx Time:③xxxxx

  Command   : ④-c310 sample¥prime.c -se
  C-file    : ⑤SAMPLE¥PRIME.C
  Err-file  : ⑥PRIME.CER
  Para-file : ⑦
  */

⑧#define TRUE    1
⑧#define FALSE   0
⑧#define SIZE    200

⑧char    mark[SIZE+1];

⑧main()
⑧{
⑧       int i, prime, k, count;

⑧       cont = 0;
  *** ERROR ⑨F711 ⑩Undeclared 'cont' ; function 'main'

⑧       for ( i = 0 ; i <= SIZE ; i++)
⑧               mark[i] = TRUE;
⑧       for ( i = 0 ; i <= SIZE ; i++) {
⑧               if (mark[i]) {
  *** ERROR ⑨F306 ⑩Illegal index , indirection not allowed



  /*
  Compilation complete.  ⑪1 error(s) and  ⑫5 warning(s) found.
  */
```

【出力項目の説明】

項番	内　　容	桁　　数	形　　式
①	バージョン番号	4	‘x.yz’の形式で表します。
②	日付	11 （固定）	システム日付。 ‘DD Mmm YYYY’の形式で表します。
③	時間	8（固定）	システム時間。 ‘HH:MM:SS’の形式で表します。
④	コマンド行		コマンド行の‘CC78K3’以降を出力します。 1行に入らない場合は，超過分を次行の13カラム目から出力します。1個以上の空白タブは1個の空白で置き換えます。
⑤	Cソース・ モジュール・ ファイル名	最大78 （可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.c’を付加します。1行に入らない場合は，超過分を次行の13カラム目から出力します。
⑥	プリプロセス・ リスト・ファイル名	最大78 （可変）	指定されたファイル名を出力します。 1行に入らない場合は，超過分を次行の13カラム目から出力します。
⑦	パラメータ・ ファイルの内容		パラメータ・ファイルの内容を出力します。 1行に入らない場合は，超過分を次行の13カラム目から出力します。1個以上の空白タブは1個の空白で置き換えます。
⑧	Cソース		入力Cソース・イメージ。80カラム以降も折り返しを行わず出力します。
⑨	エラー・メッセージ番号	4（固定）	エラー番号を‘#nnn’の形式で出力します。 #はエラーの場合はF，ワーニングの場合はWとなります。nnnはエラー番号です。
⑩	エラー・メッセージ		“第8章　エラー・メッセージ”参照。80カラム以降も折り返しを行わず出力します。
⑪	エラー個数	4（固定）	右詰めゼロ・サプレスの10進数。
⑫	ワーニング個数	4（固定）	右詰めゼロ・サプレスの10進数。

6.3.2 エラー・メッセージのみのエラー・リスト・ファイル

【出力形式】

```
①SAMPLE¥PRIME.C(② 18) : ③W745 ④Expected function prototype
①SAMPLE¥PRIME.C(② 20) : ③W745 ④Expected function prototype
①SAMPLE¥PRIME.C(② 26) : ③W622 ④No return value
①SAMPLE¥PRIME.C(② 37) : ③W622 ④No return value
①SAMPLE¥PRIME.C(② 44) : ③W622 ④No return value

Compilation complete.  ⑤0 error(s) and  ⑥5 warning(s) found.
```

【出力項目の説明】

項番	内　　容	桁　　数	形　　式
①	Cソース・モジュール・ファイル名	最大78（可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.c’を付加します。
②	行番号	5（固定）	右詰めゼロ・サプレスの10進数。
③	エラー・メッセージ番号	4（固定）	エラー番号を‘#nnn’の形式で出力します。#はエラーの場合はF，ワーニングの場合はWとなります。nnnはエラー番号です。（ゼロ・サプレスしません。）
④	エラー・メッセージ		“第8章　エラー・メッセージ”参照。
⑤	エラー個数	4（固定）	右詰めゼロ・サプレスの10進数。
⑥	ワーニング個数	4（固定）	右詰めゼロ・サプレスの10進数。

6.4 プリプロセス・リスト・ファイル

プリプロセス・リスト・ファイルは，Ｃソース・プログラムのプリプロセス処理のみを行った結果のＡＳＣＩＩイメージ・ファイルです。

－Ｋオプションを指定する際，処理種別として‘Ｎ’を指定しなければ，Ｃソース・モジュール・ファイルとして使用できます。

【出力形式】

ＰＡＧＥＷＩＤＴＨ＝８０の場合

```
/*
78K/III Series C Compiler V①x.xx Preprocess List      Date:②xxxxx Page:③xxx

Command   : ④-c310 sample¥prime.c -p -lw80
C-file    : ⑤SAMPLE¥PRIME.C
PPL-file  : ⑥PRIME.XRF
Para-file : ⑦
*/

  ⑧1 : ⑨#define TRUE    1
  ⑧2 : ⑨#define FALSE   0
  ⑧3 : ⑨#define SIZE    200
  ⑧4 : ⑨
  ⑧5 : ⑨char    mark[SIZE+1];
  ⑧6 : ⑨
```

【出力項目の説明】

項番	内　　容	桁　　数	形　　式
①	バージョン番号	4	‘x.yz’ の形式で表します。
②	日付	１１ （固定）	システム日付。 ‘DD Mmm YYYY’ の形式で表します。 す。
③	ページ数	4（固定）	右詰めゼロ・サプレスの10進数。
④	コマンド行		コマンド行の ‘CC78K3’ 以降を出力します。 1行に入らない場合は，超過分を次行の13カラム目から出力します。1個以上の空白タブは1個の空白で置き換えます。
⑤	Cソース・ モジュール・ ファイル名	最大７８ （可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.c’ を付加します。1行に入らない場合は，超過分を次行の13カラム目から出力します。1個以上の空白タブは1個の空白で置き換えます。
⑥	プリプロセス・ リスト・ファイル名	最大７８ （可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.pp1’ を付加します。1行に入らない場合は，超過分を次行の13カラム目から出力します。1個以上の空白タブは1個の空白で置き換えます。
⑦	パラメータ・ ファイルの内容		パラメータ・ファイルの内容を出力します。 1行に入らない場合は，超過分を次行の13カラム目から出力します。ただし，1カラム目に ‘;’ を出力します。1個以上の空白タブは1個の空白で置き換えます。
⑧	行番号	5（固定）	右詰めゼロ・サプレスの10進数。
⑨	Cソース		入力Cソース。1行に入らないときは超過分を次行の9カラム目から出力します。

6.5 クロス・レファレンス・リスト・ファイル

クロス・レファレンス・リスト・ファイルは，Ｃソース・プログラム中で宣言，定義，参照されている関数名，変数名などの識別子の一覧表です。属性やその行番号などの情報も含んでいます。これらを出現順に出力します。

【出力形式】

ＰＡＧＥＷＩＤＴＨ＝８０の場合

```
78K/III Series C Compiler V①x.xx Cross reference List Date:②xxxxx Page:③xxx

Command   : ④-c310 sample¥prime.c -x -lw80
C-file    : ⑤SAMPLE¥PRIME.C
Xref-file : ⑥PRIME.PPL
Para-file : ⑦
Inc-file  : [n] ⑧

ATTRIB MODIFY TYPE       SYMBOL      DEFINE  REFERENCE

⑨EXTERN ⑩     ⑪array ⑫mark          ⑬5      ⑭14  ⑭16  ⑭22
⑨EXTERN ⑩     ⑪func  ⑫main          ⑬7
⑨AUTO1  ⑩     ⑪int   ⑫i             ⑬9      ⑭13  ⑭13  ⑭13  ⑭14
                                             ⑭15  ⑭15  ⑭15  ⑭16
                                             ⑭17  ⑭17  ⑭21
⑨AUTO1  ⑩     ⑪int   ⑫prime         ⑬9      ⑭17  ⑭18  ⑭21  ⑭21
⑨AUTO1  ⑩     ⑪int   ⑫k             ⑬9      ⑭21  ⑭21  ⑭21  ⑭22
⑨AUTO1  ⑩     ⑪int   ⑫count         ⑬9      ⑭11  ⑭19  ⑭20  ⑭25
```

【出力項目の説明】（１／３）

項番	内　　容	桁　　数	形　　式
①	バージョン番号	4	‘x.yz’の形式で表します。
②	日付	１１（固定）	システム日付。 ‘DD Mmm YYYY’の形式で表します。
③	ページ数	４（固定）	右詰めゼロ・サプレスの10進数。
④	コマンド行		コマンド行の‘CC78K3’以降を出力します。 １行に入らない場合は，超過分を次行の13カラム目から出力します。１個以上の空白タブは１個の空白で置き換えます。

【出力項目の説明】（２／３）

項番	内　　容	桁　　数	形　　式
⑤	Ｃソース・モジュール・ファイル名	最大７８（可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.c’を付加します。１行に入らない場合は，超過分を次行の13カラム目から出力します。
⑥	クロス・レファレンス・リスト・ファイル名	最大７８（可変）	指定されたファイル名を出力します。ファイルタイプが省略されている場合は，‘.xrf’を付加します。１行に入らない場合は，超過分を次行の13カラム目から出力します。
⑦	パラメータ・ファイルの内容		パラメータ・ファイルの内容を出力します。 １行に入らない場合は，超過分を次行の13カラム目から出力します。１個以上の空白タブは１個の空白で置き換えます。
⑧	インクルード・ファイル	最大７８（可変）	Ｃソース中で指定されたファイル名を出力します。nは１で始まる数字でインクルード・ファイル番号を示します。 １行に入らない場合は，超過分を次行の13カラム目から出力します。 インクルード・ファイルがない場合は，この行は出力されません。
⑨	シンボル属性	４（固定）	シンボル属性を表します。左詰め。 外部変数はEXTERN，static変数は外部がEXSTCで内部がINSTC，auto変数はAUTOnn，レジスタ変数はREGnn（nnは１で始まる数字でスコープを示します），typedef宣言は外部がEXTYPで内部がINTYP，ラベルはLABEL，構造帯・共用体のタグはTAG，メンバはMEMBER，関数のパラメータはPARAMです。

【出力項目の説明】（３／３）

項番	内　　容	桁　　数	形　　式
⑩	シンボル修飾属性	4（固定）	シンボル修飾属性を表します。左詰め。 const変数はCONST，volatile変数はVLT，callt関数はCALLT，callf関数はCALLF，noauto関数はNOAUTO，norec関数はNOREC，sreg･bit変数はSREG，sfr変数はRWSFR，リードオンリーのsfr変数はROSFR，ライトオンリーのsfr変数はWOSFR，割り込み関数はVECTです。
⑪	シンボル・タイプ	7（固定）	シンボルのタイプを表します。左詰め。 char，int，short，long，fieldとなります。 unsignedタイプはそれぞれ先頭にuが付加されます。他にvoid，func，array，pointer，struct，union，enum，bit，#defineがあります。
⑫	シンボル名	１６（固定）	15カラムを越えた場合は，シンボル名をそのまま出力し⑬，⑭の項目を次行の39カラム目から出力します。
⑬	シンボル定義行番号	5（固定）	シンボルの定義された行番号とファイル名で，行番号（５桁）:インクルード・ファイル番号の形式で表します。
⑭	シンボル参照行番号	5（固定）	シンボルを参照している行番号とファイル名で行番号（５桁）:インクルード・ファイル番号の形式で表します。 １行に入らない時は，超過分を次行の47カラム目から出力します。

保守／廃止

第７章　Ｃコンパイラの活用法

7.1 効率良く作業する（ＥＸＩＴステータス機能）

本Ｃコンパイラは，コンパイル終了時にコンパイル中に発生した最大のエラー・レベルをＥＸＩＴステータスとして，ＯＳに返します。

ＥＸＩＴステータスを次に示します。

- 正常終了時　：０
- ＷＡＲＮＩＮＧあり　：０
- ＦＡＴＡＬ　ＥＲＲＯＲあり　：１
- ＡＢＯＲＴ時　：２

これらをバッチ・ファイルで利用することにより効率良く作業を進めることができます。

【使用例】

```
cc78k3 -c310 %1
IF ERRORLEVEL 1 GOTO ERR
cc78k3 -c310 %2
IF ERRORLEVEL 1 GOTO ERR
GOTO EXIT
:ERR
echo Some error found.
:EXIT
```

【説　明】

％１に渡されたＣソースをコンパイルした時点でＦＡＴＡＬ　ＥＲＲＯＲが生じたとします。本来ならばエラー・メッセージを出力したあと処理を続行しますが，ＥＸＩＴステータスに１が返されることを利用して，次の％２のＣソースの処理を行わずに実行を停止させることができます。

7.2 開発環境を整える（環境変数）

プログラム開発を行う場合，関連ファイル別にディレクトリを作成し，関連ファイルを１まとめにしておくと作業が円滑に行えます。

このことは，環境変数を指定することにより実現できます。

本Ｃコンパイラでサポートする環境変数を次に示します。

- ＰＡＴＨ　　：実行形式のサーチパス
- ＩＮＣ７８Ｋｎ：インクルード・ファイルのサーチパス
  （ｎ＝０，２，３　：　ｎは各シリーズの番号となります。）
- ＴＭＰ　　：テンポラリ・ファイルのサーチパス

【使用例】

```
;AUTOEXEC.BAT
PATH A:¥BIN;A:¥BAT;A:¥CC78K3;A:¥TOOL;
verify on
break on
SET INC78K3=A:¥CC78K3¥INCLUDE
SET LIB78K3=A:¥CC78K3¥LIB
SET TMP=A:¥
```

【説　明】

・パス指定により，Ａ：￥ＢＩＮ，Ａ：￥ＢＡＴ，Ａ：￥ＣＣ７８Ｋ３，Ａ：￥ＴＯＯＬという順で実行形式ファイルを検索します。

・インクルード・ファイルは，Ａ：￥ＣＣ７８Ｋ３￥ＩＮＣＬＵＤＥから検索されます。

・ライブラリ・ファイルは，Ａ：￥ＣＣ７８Ｋ３￥ＬＩＢから検索されます。また，ライブラリ・ファイルはリンカにその場所を示します。

・テンポラリ・ファイルは，Ａ：￥に作成されます。

7.3 コンパイルを中断する

キー入力（ＣＴＲＬ－Ｃ）によりコンパイルを中断できます。break onを指定した場合にはキー入力のタイミングに関係なしに，また，break offの場合には画面表示中のみに制御をＯＳに戻します。そして，オープン中のすべてのテンポラリ・ファイル，出力ファイルを削除します。

第８章　スタートアップ・ルーチン，エラー処理ルーチン

■スタートアップ・ルーチン

最近，シングルチップ・マイコンの世界にも，Ｃ言語によるプログラム開発の必要性が強まってきました。Ｃ言語によるプログラムを実行させるには，システムへ組み込むためのＲＯＭ化処理，ユーザ・プログラム（main関数）の起動などを行うプログラムが必要となります。このプログラムのことをスタートアップ・ルーチンといいます。

ユーザが作成したプログラムを実行させるためには，そのプログラムに応じたスタートアップ・ルーチンを作成しなければなりません。しかし，Ｃ言語でのマイコン開発経験が少ないユーザにとって，スタートアップ・ルーチンを作成することは容易ではありません。

本章では，そのような人たちを対象にサンプル・プログラムを例として，スタートアップ・ルーチンの内容，使い方，サンプル・プログラム改良のポイントなどについて説明します。

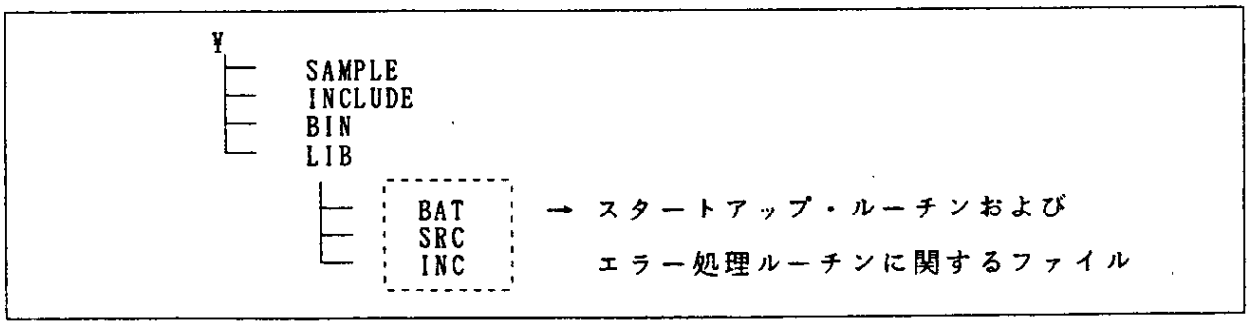
■エラー処理ルーチン

ＣＣ７８Ｋシリーズ　Ｃコンパイラは，プログラム実行時にエラー・チェックを行うことができるライブラリをサポートしています。エラー・チェックの内容は，演算によるオーバーフロー，０除算，ポインタ・アクセスなどがあります。これらのエラーが発生したときには，そのエラー内容，あるいはターゲット・システムに応じて，適切な処理を行うためのプログラムが呼び出されます。このプログラムのことをエラー処理ルーチンといいます。このエラー処理ルーチンを使用することにより，ターゲット・システムに応じたディバグを行うことができます。

本章は，スタートアップ・ルーチンと同様，サンプル・プログラムを例にエラー処理ルーチンの内容，使い方，サンプル・プログラム改良のポイントなどについて説明します。

8.1 ファイルの構成

スタートアップ・ルーチン，エラー処理ルーチンに関するファイルは，フロッピィ・ディスク(1)のディレクトリＬＩＢの下にある３つのディレクトリに格納されています。



次にＢＡＴ，ＳＲＣ，ＩＮＣの３つのディレクトリの内容について示します。

表８－１　ディレクトリ“ＢＡＴ”の内容

ファイル名	機　　能
《バッチ・ファイル》	
ＭＫＳＴＵＰ．ＢＡＴ	・スタートアップ・ルーチン作成用バッチ・ファイル
ＭＫＥＲＲＬＩＢ．ＢＡＴ	・エラー処理ルーチンのライブラリ更新用バッチ・ファイル
ＣＬＸＸＸ．ＥＲＲ	・ライブラリ・ファイルＣＬＸＸＸ．ＬＩＢ作成用のサブコマンド・ファイル （ＭＫＥＲＲＬＩＢ．ＢＡＴで使用）
ＣＬＸＸＸＵ．ＥＲＲ	・ライブラリ・ファイルＣＬＸＸＸＵ．ＬＩＢ作成用のサブコマンド・ファイル （ＭＫＥＲＲＬＩＢ．ＢＡＴで使用） ＸＸＸ：デバイス種別 （”表5-3～5-5　デバイス種別”参照）

表８－２　ディレクトリ“ＳＲＣ”の内容

ファイル名	機　　能
《ｽﾀｰﾄｱｯﾌﾟ・ﾙｰﾁﾝ・ｿｰｽ・ﾌｧｲﾙ》	
ＣＳＴＡＲＴＲ．ＡＳＭ	・ＲＯＭ化用スタートアップ・ルーチン
ＥＳＴＡＲＴＲ．ＡＳＭ	・ＲＯＭ化用スタートアップ・ルーチン （ＲＯＭ化処理のエラー・チェックあり）
ＣＳＴＡＲＴ．ＡＳＭ	・ＲＯＭ化を行わない場合のスタートアップ・ルーチン
ＲＯＭ．ＡＳＭ	・ＲＯＭ化用ファイル
《ｴﾗｰ処理・ﾙｰﾁﾝ・ｿｰｽ・ﾌｧｲﾙ》	
ＥＲＲＳＴＫ．ＡＳＭ	・スタック・オーバーフローのエラー処理
ＥＲＲＤＩＶ．ＡＳＭ	・０除算のエラー処理
ＥＲＲＰＴＲ．ＡＳＭ	・ポインタ・アクセスのエラー処理
ＥＲＲＯＶＦ．ＡＳＭ	・オーバーフローのエラー処理
ＥＲＲＳＦＲ．ＡＳＭ	・sfrアクセスのエラー処理
ＥＲＲＩＮＩ．ＡＳＭ	・ＲＯＭ化領域のエラー処理
《ｲﾝｸﾙｰﾄﾞ・ﾌｧｲﾙ》	
ＤＥＦＩＮＥ．ＩＮＣ	・saddr領域のラベル定義
ＭＡＣＲＯ．ＩＮＣ	・各デバイスにおける転送命令についてのマクロ定義

表８－３　ディレクトリ“ＩＮＣ”の内容

ファイル名	機　　能
《ｲﾝｸﾙｰﾄﾞ・ﾌｧｲﾙ》	
ＣＨＩＰＸＸＸ．ＩＮＣ	・デバイス情報 ＸＸＸ：デバイス種別 （”表5-3～5-5　デバイス種別”参照）

8.2 バッチ・ファイルの説明

8.2.1 スタートアップ・ルーチン作成用バッチ・ファイル

スタートアップ・ルーチンのオブジェクト・ファイルを作成するには，フロッピィ・ディスク(1)のディレクトリＢＡＴにあるＭＫＳＴＵＰ．ＢＡＴを使用します。

また，ＭＫＳＴＵＰ．ＢＡＴでは，“ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ”の中のアセンブラが必要となります。したがって，バッチ・ファイルがあるディレクトリに設定するか，ＰＡＴＨを設定するかしてください。必要なファイルを次に示します。

＜７８Ｋ／Ⅲの場合＞

ＲＡ７８Ｋ３．ＥＸＥ ＲＡ７８Ｋ３．ＯＭ１～ＲＡ７８Ｋ３．ＯＭ６

以下に使用方法を示します。

【使用方法】

ＭＫＳＴＵＰ．ＢＡＴのあるディレクトリＢＡＴで，以下のように実行してください。

A>MKSTUP　デバイス種別(注1)　シンボル名ケース指定(注2)

注1.　“表５－３～５－５　デバイス種別”を参照してください。

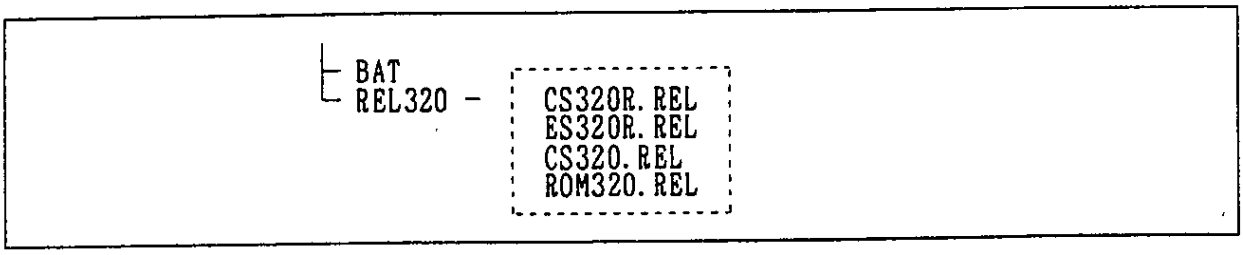
2.　“５．４（４）　シンボル名ケース指定”参照してください。

【使用例】

対象品種がμＰＤ７８３２０で，コンパイル時に，シンボル名ケース指定のオプションを指定しないときに，使用するスタートアップ・ルーチンを作成します。

```
A>MKSTUP 320 NCA
```

バッチ・ファイルＭＫＳＴＵＰは，以下のようにディレクトリＢＡＴと同じ階層にディレクトリＲＥＬ３２０を作成します。次にこのＲＥＬ３２０の下にスタートアップ・ルーチンのオブジェクト・ファイルを格納します。



8.2.2 エラー処理ルーチンのライブラリ更新用バッチ・ファイル

改良したエラー処理ルーチンをライブラリ・ファイル中にあるものと置き換えるには，フロッピィ・ディスク(1)のディレクトリＢＡＴにあるＭＫＥＲＲＬＩＢ．ＢＡＴを使用します。

また，ＭＫＥＲＲＬＩＢ．ＢＡＴでは，以下の２つの準備を行う必要があります。

(1) アセンブラ，ライブラリアンの設定

ＭＫＥＲＲＬＩＢ．ＢＡＴでは，“ＲＡ７８Ｋシリーズ　アセンブラ・パッケージ”の中のアセンブラとライブラリアンが必要となります。したがって，バッチ・ファイルがあるディレクトリに設定するか，ＰＡＴＨを設定するかしてください。必要なファイルを，次に示します。

＜７８Ｋ／Ⅲの場合＞

```
ＲＡ７８Ｋ３．ＥＸＥ
ＲＡ７８Ｋ３．ＯＭ１～ＲＡ７８Ｋ３．ＯＭ６
ＬＢ７８Ｋ３．ＥＸＥ
```

(2) ライブラリ・ファイルの設定

更新するライブラリ・ファイルを格納するためのディレクトリＬＩＢをＢＡＴと同じ階層に作り，そこにライブラリ・ファイルを格納してください。

```
├ BAT
└ LIB  －  CLXXX.LIB  … 更新するライブラリ・ファイル
```

次に使用方法を示します。

【使用方法】

ＭＫＥＲＲＬＩＢ．ＢＡＴのあるディレクトリＢＡＴで，次のように実行してください。

```
A>MKERRLIB　デバイス種別　シンボル名ケース指定
```

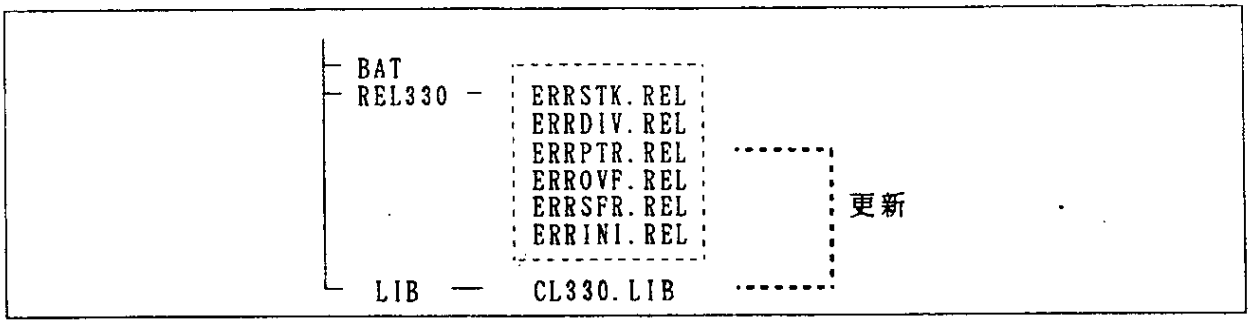
デバイス種別，シンボル名ケース指定については，「8.2.1 スタートアップ・ルーチン作成用バッチ・ファイル」と同様です。

【使用例】

対象品種がμＰＤ７８３３０で，コンパイル時に，シンボル名ケース指定のオプションを指定しないときに，使用するライブラリ・ファイルＣＬ３３０．ＬＩＢを更新します（ディフォールトで－ＮＣＡ指定されています）。

```
A>MKERRLIB 330 NCA
```

バッチ・ファイルＭＫＥＲＲＬＩＢは，以下のようにディレクトリＢＡＴと同じ階層にディレクトリＲＥＬ３３０を作成します。次にこのＲＥＬ３３０の下にエラー処理ルーチンのオブジェクト・ファイルを作成し，格納します。最後に，ライブラリアンにより，ライブラリ・ファイルＣＬ３３０．ＬＩＢのエラー処理ルーチンをＲＥＬ３３０にあるエラー処理ルーチンで更新します。



8.3 スタートアップ・ルーチン

8.3.1 スタートアップ・ルーチンの概要

スタートアップ・ルーチンは，ユーザが作成したＣソース・プログラムを実行させるために必要な準備を行うことです。ユーザのプログラムとリンクさせることにより，目的を果たすロード・モジュール・ファイルを作成できます。

(1) 機　能

メモリの初期化，システムへ組み込むためのＲＯＭ化処理，Ｃソース・プログラムの起動，終了処理などを行います。

ＲＯＭ化処理…Ｃソース・プログラム中で定義された外部変数，スタティック変数，sreg変数の初期値は，ＲＯＭに配置されます。しかし，ＲＯＭに配置されたままでは，変数の値を書き換えたりすることができません。よって，ＲＯＭに配置された初期値をＲＡＭにコピーする必要があります。この処理をＲＯＭ化処理といい，プログラムをＲＯＭに書き込んだとき，マイコン上で動作できるようにします。

(2) 構　成

スタートアップ・ルーチンに関連するサンプル・プログラムとその構成を“表８－４　使用するスタートアップ・ルーチン”に示します。

表８－４　使用するスタートアップ・ルーチン

ＲＯＭ化しない場合	ＲＯＭ化を行う場合	
	ディバグ用	システム組み込み用
cstart.asm 前処理 初期設定 main関数起動 後処理	estartr.asm 前処理 初期設定 ROM化処理 main関数起動 後処理	cstartr.asm 前処理 初期設定 ROM化処理 main関数起動 後処理
	rom.asm ＲＯＭ化処理で使用するレーベルの定義	rom.asm ＲＯＭ化処理で使用するレーベルの定義

cstartr.asm，estartr.asmは，cstart.asmとほぼ同じ内容です。estartr.asmは，ＲＯＭ化処理のエラー・チェックも行います。

rom.asmは，ＲＯＭ化処理でコピーされるデータの最終アドレスを示すレーベルを定義しています。rom.asmは，プログラムのＲＯＭ化を行うときに必要となります。cstartr.asm，estartr.asmを使用する場合は，rom.asmもリンクしてください。リンク方法については，“4.5　プログラムのＲＯＭ化”を参照してください。

(3) スタートアップ・ルーチンの使い分け

ユーザは，各デバイスに対応したスタートアップ・ルーチンのオブジェクト・ファイルを作成し，プログラムとリンクさせてください。オブジェクト・ファイルの作成については“8.2.1　スタートアップ・ルーチン作成用バッチ・ファイル”を参照してください。

各ソース・ファイルに対応したオブジェクト・ファイル名を“表８－５　ソース・ファイルとオブジェクト・ファイルの対応”に示します。

表８－５　ソース・ファイルとオブジェクト・ファイルの対応

ファイルの種類	ソース・ファイル	オブジェクト・ファイル
スタートアップ・ルーチン	ｃｓｔａｒｔ．ａｓｍ	ｃｓｘｘｘ．ｒｅｌ　注1 （ｃｓｘｘｘｕ．ｒｅｌ）注2
	ｅｓｔａｒｔｒ．ａｓｍ	ｅｓｘｘｘ．ｒｅｌ （ｅｓｘｘｘｕ．ｒｅｌ）
	ｃｓｔａｒｔｒ．ａｓｍ	ｃｓｘｘｘｒ．ｒｅｌ （ｃｓｘｘｘｒｕ．ｒｅｌ）
ＲＯＭ化ファイル	ｒｏｍ．ａｓｍ	ｒｏｍｘｘｘ．ｒｅｌ （ｒｏｍｘｘｘｕ．ｒｅｌ）

注1．ｘｘｘ：デバイス種別（”表5-3～5-5　デバイス種別”を参照してください。）

2．（　）内は，コンパイル時に－ＣＡオプションを指定したときに使用するスタートアップ・ルーチンです。－ＣＡオプションは，シンボル名のケース（大文字，小文字）を区別しないように指示します。

これらのルーチンは，使用する開発フェーズによって使い分けてください。“表８－６　スタートアップ・ルーチンの使い分け”に各ルーチンを使用する推奨フェーズなどを示します。

表８－６　スタートアップ・ルーチンの使い分け

ｽﾀｰﾄｱｯﾌﾟ・ﾙｰﾁﾝの種類	相違点		使用時のメリット	使用推奨フェーズ
	ROM化	ROM化ﾁｪｯｸ		
cstart.asm	×	×	ｺﾝﾊﾟｲﾙ,ﾘﾝｸが速い	机上ディバグまで
estartr.asm	○	○	ROM化処理ﾁｪｯｸ可	ｼｽﾃﾑ組み込み (ROM化処理ﾁｪｯｸ時)
cstartr.asm	○	×	－	ｼｽﾃﾑ組み込み (ROMを起こす場合)

8.3.2 サンプル・プログラムの説明

ここでは，スタートアップ・ルーチンの内容について，estartr.asm，rom.asmを例に説明します。

サンプル・プログラムは，７８Ｋ／Ⅲのものを使用します。

●ｅｓｔａｒｔｒ．ａｓｍ

スタートアップ・ルーチンのサンプル・プログラムestartr.asmを”表８－４　使用するスタートアップ・ルーチン”で示した前処理，初期設定，ＲＯＭ化処理，main関数起動と後処理の順に説明します。

estartr.asmとcstart.asm，cstartr.asmとの相違点，プログラム・リストの内容とそれを説明しているページについて“表８－７　スタートアップ・ルーチンの内容の比較”に示します。

表８－７　スタートアップ・ルーチンの内容の比較

estartr.asm	estartr.asmとの比較	
	cstart.asm	cstartr.asm
前処理	同じ	同じ
初期設定	同じ	同じ
ＲＯＭ化処理	なし	一部異なる
main関数起動，後処理	同じ	同じ

前処理 | 説明：133ページ

```
        NAME    @cstart

$INCLUDE 'CHIP.INC'
$INCLUDE 'DEFINE.INC'
$INCLUDE 'MACRO.INC'

        PUBLIC  _@cstart

        PUBLIC  _errno,_@FNCTBL,_@FNCENT,_@BRKADR,_@MEMTOP,_@MEMBTM,_@SEED
        PUBLIC  _@DIVR,_@LDIVR,_@TOKPTR

        PUBLIC  _@D_FUNC
        PUBLIC  _@D_LINE
        PUBLIC  _@KREG00
                .
                .
                .
                .
        PUBLIC  _@FARG1H
        PUBLIC  _@FARG0L
        PUBLIC  _@FARG0H

        EXTRN   _main,_exit,_@STBEG
        EXTRN   _?R_INIT,_?R_DATA,_?R_INIS,_?R_DATS
;error check
        EXTRN   _@errini
        EXTRN   _?INIT,_?DATA,_?INIS,_?DATS
```

①インクルード・ファイルの取り込み

②シンボルの外部定義宣言

③saddr領域用のラベルの外部定義宣言

④スタック解決用のシンボルの外部参照宣言

⑤ＲＯＭ化処理用レーベルの外部参照宣言

⑥⑤と同じ

初期設定 | 説明：134ページ

```
@@DATA  DSEG

_errno:         DS      2
_@FNCTBL:       DS      2*32
_@FNCENT:       DS      2
_@BRKADR:       DS      2
_@SEED:         DS      4
_@DIVR:         DS      4
_@LDIVR:        DS      8
_@TOKPTR:       DS      2
_@MEMTOP:       DS      32
_@MEMBTM:

@@CODE  CSEG

_@cstart:
        SEL     RB7     ; ②レジスタ・バンクの設定
        MOVW    AX,#_@STBEG
        MOVW    SP,AX           ;SP <- stack begin address
$_IF(P312A OR P322 OR P328 OR P334)
        MOVW    AX,#0
        MOVW    !_errno,AX      ;errno <- 0
        MOVW    !_@FNCENT,AX    ;FNCENT <- 0
        MOVW    !_@SEED+2,AX
        MOVW    AX,#1
        MOVW    !_@SEED,AX      ;SEED <- 1
        MOVW    AX,#_@MEMTOP
        MOVW    !_@BRKADR,AX    ;BRKADR <- #MEMTOP
$ELSE
        MOV     A,#0
        MOV     !_errno,A
        MOV     !_errno+1,A     ;errno <- 0
        MOV     !_@FNCENT,A
        MOV     !_@FNCENT+1,A   ;FNCENT <- 0
        MOV     !_@SEED+1,A
        MOV     !_@SEED+2,A
        MOV     !_@SEED+3,A
        MOV     A,#1
        MOV     !_@SEED,A       ;SEED <- 1
        MOVW    AX,#_@MEMTOP
        MOV     !_@BRKADR+1,A
        XCH     A,X
        MOV     !_@BRKADR,A     ;BRKADR <- #MEMTOP
$ENDIF
```

①ライブラリで使用するシンボルの領域確保

③スタック・ポインタの設定

④シンボルの初期値設定

ﾃﾞﾊﾞｲｽ種別:310,312以外

⑤④と同じ

ﾃﾞﾊﾞｲｽ種別:310,312

ＲＯＭ化処理	説明：136ページ

```
;ROM DATA COPY
        MOVW    DE,#_@INIT
        MOVW    HL,#_@R_INIT
        MOVW    UP,#_?R_INIT
                                ;error check
        MOVW    AX,#_?INIT
        SUBW    AX,DE
        ADDW    AX,HL
        SUBW    AX,UP
        BZ      $LINIT1
        CALL    !_@errini
                                ;
LINIT1:
        CMPW    HL,UP
        BE      $LINIT2
        MOV     A,[HL+]
        MOV     [DE+],A
        BR      $LINIT1
LINIT2:
              .
              .
              .
              .
LINIS2:
        MOVW    DE,#_@DATS
        MOVW    HL,#_@R_DATS
        MOVW    UP,#_?R_DATS
                                ;error check
        MOVW    AX,#_?DATS
        SUBW    AX,DE
        ADDW    AX,HL
        SUBW    AX,UP
        BZ      $LDATS1
        CALL    !_@errini
                                ;
LDATS1:
        CMPW    HL,UP
        BE      $LDATS2
        MOV     A,[HL+]
        MOV     [DE+],A
        BR      $LDATS1
LDATS2:
```

②ＲＯＭ化処理のエラー・チェック

①ＲＯＭ化処理

②

①

main関数の起動と後処理 | 説明：138ページ

```
        CALL    !_main          ;main();  ― ①main関数の起動
        MOVW    AX,#0                     ┐
        PUSH    AX                        ├ ②exit関数の起動
        CALL    !_exit          ;exit(0);  ┘
        POP     AX
        BR      $$
;
@@R_INIT        CSEG                      ┐
_@R_INIT:                                 │
@@R_DATA        CSEG                      │
_@R_DATA:                                 │
@@R_INIS        CSEG                      │
_@R_INIS:                                 │
@@R_DATS        CSEG                      │
_@R_DATS:                                 │
@@INIT  DSEG                              ├ ③ＲＯＭ化処理で利用する
_@INIT:                                   │   セグメント，レーベルの定義
@@DATA  DSEG                              │
_@DATA:                                   │
@@INIS  DSEG    SADDRP                    │
_@INIS:                                   │
@@DATS  DSEG    SADDRP                    │
_@DATS:                                   │
@@CALT  CSEG    CALLTO                    │
@@CNST  CSEG                              │
@@BITS  BSEG                              ┘

        END
```

(1) 前処理

estartr.asmの前処理（129ページ）にある①～⑥について説明します。

①インクルード・ファイルの取り込み

CHIP.INC → デバイス情報（スタートアップ・ルーチンの初期設定で，各デバイスに応じた設定を行うためのファイル）

DEFINE.INC → saddr領域のラベル定義用のファイル

MACRO.INC → 各デバイスの転送命令についてのマクロ定義

②シンボルの外部定義宣言

・各シンボルの詳細は，(2)初期設定を参照してください。

③saddr領域用のラベルの外部定義宣言

・各レーベルの詳細は，“言語編　付録D　saddr領域のラベル一覧”を参照してください。

④スタック解決用のシンボルの外部参照宣言（_@STBEG）

・スタック解決用のパブリック・シンボル（_@STBEG）を自動生成します。_@STBEGは，スタック領域の最終アドレス＋１を値として持ちます。

・_@STBEGは，リンカのスタック解決用シンボル生成オプション（－Ｓ）を指定することにより，自動生成されます。よって，リンク時には－Ｓオプションを指定してください。

⑤⑥ＲＯＭ化処理用レーベルの外部参照宣言

・⑤のレーベルは，後処理の部分で定義されます（132ページ）。

・⑥のレーベルは，rom.asm（ＲＯＭ化用ファイル）で定義されます（133ページ）。

(2) 初期設定

estartr.asmの初期設定（130ページ）にある①～⑤について説明します。

①ライブラリで使用するシンボルの領域確保

ライブラリに使用されるシンボルの領域を確保します。各ライブラリ関数の詳細については，“言語編　10.3　標準ライブラリ関数”を参照して下さい。

①－1
```
_errno:          DS      2
```

・エラー・コードを設定する領域が確保されます。使用するのは，以下の４つのライブラリ関数です。

```
strtol,strtoul（文字・文字列関数）
brk,sbrk       （メモリ関数）
```

①－2
```
_@FNCTBL:        DS      2*32
_@FNCENT:        DS      2
```

・exit関数で使用される以下の領域が確保されます。

_@FNCTBL　：　atexit関数で使用する領域の先頭アドレスを示します。
　　atexit関数は，この領域に関数のアドレスを登録します。

_@FNCENT　：　atexit関数で登録された関数の総数を格納します（最大32個まで）。

①－3
```
_@BRKADR:        DS      2
```

・ブレーク値を設定する領域が確保されます。

ブレーク値とは，メモリ関数が使用する領域の先頭アドレスを示します。

①－4
```
_@SEED:          DS      4
```

・疑似乱数列の基となる値を格納する領域が確保されます。

・rand，srand関数（数学関数）で使用されます。

①－5
```
_@DIVR:          DS      4
```

・div関数（数学関数）において，計算処理の結果を格納するための領域が確保されます。

①－6
```
_@LDIVR:         DS      8
```

・ldiv関数（数学関数）において，計算処理の結果を格納するための領域が確保されます。

①－7
```
_@TOKPTR:        DS      2
```

・strtok関数（文字・文字列関数）において，ポインタを格納するための領域が確保されます。

①－8
```
_@MEMTOP:        DS      32
_@MEMBTM:
```

・メモリ関数で使用する領域が確保されます。サンプル・プログラムでは，３２バイトの領域が確保されています。

・_@MEMTOP，_@MEMBTMは，この領域の先頭および最終アドレスを示すレーベルです。

②レジスタ・バンクの設定

・レジスタ・バンクＲＢ７をワーク・レジスタとして設定します。

③スタック・ポインタの設定

・スタック・ポインタに，_@STBEGを設定します。

_@STBEGは，リンカのスタック解決用シンボル生成オプション（－Ｓ）を指定することにより，自動生成されます。

④⑤シンボルの初期値設定

・_errno，_@FNCENTに初期値０を設定します。各シンボルに対応するライブラリを実行中に，正の整数値が与えられます。

・_@SEEDに，初期値１を設定します。疑似乱数列は，_@SEEDの値により決まります。この値は，srand関数により設定できます。rand関数がsrand関数が呼び出される前に呼ばれた場合は，_@SEEDを１としてsrand関数が呼ばれた状態と同じです。

・ブレーク値として，メモリ関数用に確保した領域の先頭アドレス（@_MEMTOP）を設定します。

(3) ＲＯＭ化処理

estartr.asmのＲＯＭ化処理（131ページ）にある①～②について説明します。

①ＲＯＭ化処理

ＲＯＭ化処理では，ＲＯＭに配置された外部変数，sreg変数の初期値をＲＡＭにコピーします。処理される変数は，次の例に示すように(ａ)～(ｄ)の４種類あります。

```
例）
        char     c = 1;          -------  (a) 初期値あり外部変数
        int      i;              -------  (b) 初期値なし外部変数   注1

        sreg  int     si = 0;    -------  (c) 初期値ありsreg変数
        sreg  char    sc;        -------  (d) 初期値なしsreg変数   注1

        main()
        {
              .
              .
              .
        }

        注1. 初期値なし外部変数、sreg変数は，０で初期化されます。
             よって，初期値０を持つことと同じになります。
```

・“図８－１　ＲＯＭ化処理”に，(a)初期値あり外部変数のＲＯＭ化処理を示します。

変数(a)の初期値は，コンパイラにより，ＲＯＭ上の@@R_INITというセグメントに配置されます。配置された領域の先頭アドレスと最終アドレスを示したレーベルは，それぞれ_@R_INITと_?R_INITです。ＲＯＭ化処理は，これらの値をＲＡＭ上の@@INITというセグメントにコピーします。そして，この領域の先頭アドレスと最終アドレスのレーベルを，それぞれ_@INITと_?INITで定義します。

・変数(b)，(c)，(d)についても，同様な処理を行います。

なお，(a)～(d)の変数が配置されるＲＯＭ、ＲＡＭ領域のセグメント名，および各セグメントでの初期値の先頭，最終レーベルを“表８－８ 初期値のＲＯＭ領域”と“表８－９　初期値のＲＡＭ領域（コピー先）”に示します。

図８－１　ＲＯＭ化処理

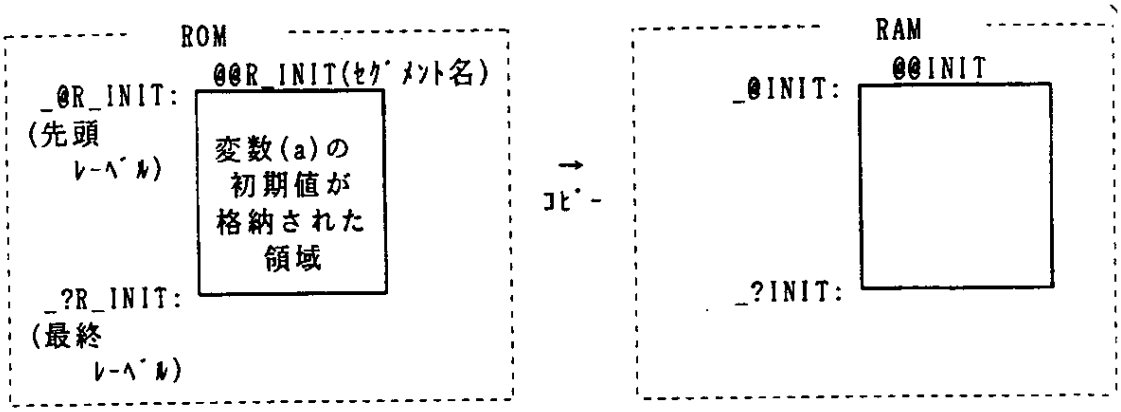


表８－８　初期値のＲＯＭ領域

変数の種類	セグメント	先頭レーベル	最終レーベル
初期値あり外部変数(a)	@@R_INIT	_@R_INIT	_?R_INIT
初期値なし外部変数(b)	@@R_DATA	_@R_DATA	_?R_DATA
初期値ありsreg変数(c)	@@R_INIS	_@R_INIS	_?R_INIS
初期値なしsreg変数(d)	@@R_DATS	_@R_DATS	_?R_DATS

表８－９　初期値のＲＡＭ領域（コピー先）

変数の種類	セグメント	先頭レーベル	最終レーベル
初期値あり外部変数(a)	@@INIT	_@INIT	_?INIT
初期値なし外部変数(b)	@@DATA	_@DATA	_?DATA
初期値ありsreg変数(c)	@@INIS	_@INIS	_?INIS
初期値なしsreg変数(d)	@@DATS	_@DATS	_?DATS

②ＲＯＭ化処理のエラー・チェック（estartr.asmのみ）

・estartr.asmでは，初期値をＲＡＭへコピーすると同時に，その処理が，正常に行われているかどうかをチェックします。正常に行われていないときには，エラー処理ルーチンerrini.asmを呼びます。チェック内容は，ＲＯＭ上のデータ・サイズとコピー先であるＲＡＭ上のデータ・サイズの比較です。

(4) main関数の起動と後処理

estartr.asmのmain関数の起動と後処理（132ページ）にある①～③について説明します。

①main関数の起動

・main関数を呼び出します。

②exit関数の起動

・exit関数を呼び出します。

・exit関数では，atexit関数により登録された全ての関数を，最後に登録されたものから順に実行します。なお，atexit関数により登録された関数のデータは，初期設定で定義された_@FNCTBL、_@FNCENTに格納されます。

③ＲＯＭ化処理で使用するセグメント，レーベルの定義

・ＲＯＭ化処理で，(a)～(d)の変数ごとに，使用するセグメント，レーベルを定義します。

セグメントは，各変数の初期値を格納する領域を示します。

レーベルは，各セグメントの先頭アドレスを示します。

● r o m

ＲＯＭ化用ファイルrom.asmについて説明します。

```
        NAME    @rom
;
        PUBLIC  _?R_INIT,_?R_DATA,_?R_INIS,_?R_DATS
        PUBLIC  _?INIT,_?DATA,_?INIS,_?DATS
;
@@R_INIT        CSEG
_?R_INIT:
@@R_DATA        CSEG
_?R_DATA:
@@R_INIS        CSEG
_?R_INIS:
@@R_DATS        CSEG
_?R_DATS:
@@INIT  DSEG
_?INIT:
@@DATA  DSEG
_?DATA:
@@INIS  DSEG    SADDRP
_?INIS:
@@DATS  DSEG    SADDRP
_?DATS:
;
        END
```

①ＲＯＭ化処理で使用するレーベルの定義

①ＲＯＭ化処理で使用するレーベルの定義

・ＲＯＭ化処理で，（ａ）～（ｄ）の変数毎に，使用するレーベルを定義します。
これらのレーベルは，各変数の初期値を格納するセグメントの最終アドレスを示します。

8.3.3 改良のポイント

(1) ライブラリ関数で使われるシンボル

“表８－１０　ライブラリ関数で使われるシンボル”で示されるライブラリ関数を使わないのであれば，スタートアップ・ルーチン中の各関数に対応するシンボルは削除できます。

ただし，exit関数は，スタートアップ・ルーチンで使用されるので，_@FNCTBL,_@FNCENTは削除できません。

表８－１０　ライブラリ関数で使われるシンボル

ライブラリ関数		使われるシンボル
関数の種類	関数名	
文字・文字列関数	strtol strtoul	_errno
メモリ関数	brk sbrk	
プログラム制御関数	exit	_@FNCTBL _@FNCENT
メモリ関数	malloc calloc realloc free brk sbrk	_@MEMTOP _@MEMBTM _@BRKADR
数学関数	rand srand	_@SEED
数学関数	div	_@DIVR
数学関数	ldiv	_@LDIVR
文字・文字列関数	strtok	_@TOKPTR

(2) メモリ関数で使われる領域

メモリ関数で使われる領域サイズをユーザが定義する場合は，以下の例のように設定します。

例） メモリ関数用として，７２バイト確保したい場合，スタートアップ・ルーチンの初期設定で，次のように指定してください。

```
                                                    ＲＡＭ領域
                                          _@MEMTOP  ┌──────────┐
_@MEMTOP:        DS      72      →                  │メモリ関数用│
_@MEMBTM:                                           │領域として  │
                                                    │７２バイト  │
                                                    │確保        │
                                          _@MEMBTM  └──────────┘
```

指定したサイズが大きすぎる場合，ＲＡＭ領域に入りきらず，リンク時にエラーとなることがあります。

このような場合には，以下のような２つの方法で回避してください。

(a) 指定するサイズを小さくする。

例）

```
_@MEMTOP:        DS      72      →   40に変更
```

(b) リンク・ディレクティブ・ファイルのＲＡＭサイズを大きくする。

例）標準リンク・ディレクティブ・ファイル

ＬＩＮＫ３２０．ＲＯＭの場合

```
                    先頭
                    ｱﾄﾞﾚｽ        ｻｲｽﾞ
  ------------------------------------
  memory CLT  : ( 00040h , 00040h )
  memory ROM  : ( 02000h , 0B000h )
  memory RAM  : ( 0D000h , 01000h )  →  このサイズを大きくする。
  memory SDR  : ( 0FE2Ch , 00050h )     サイズを変更する場合は，
  merge CALT  : = CLT                   他の領域と重ならないように
  merge CNST  : = ROM                   注意してください。
  merge CODE  : = ROM
  merge R_DATA : = ROM
  merge DATA  : = RAM
  merge R_INIT : = ROM
  merge INIT  : = RAM
  merge R_DATS : = ROM
  merge DATS  : = SDR
  merge R_INIS : = ROM
  merge INIS  : = SDR
  merge BITS  : = SDR
```

標準リンク・ディレクティブ・ファイルは，フロッピィ・ディスク(1)のディレクトリＬＩＢにあります。

8.4 エラー処理ルーチン

8.4.1 エラー処理ルーチンの概要

本パッケージでは，プログラム実行時に，エラー・チェックを行うことができるライブラリをサポートしています。チェック可能なエラーは，“(1) 機　能”に示されています。これらのエラーが発生したときに呼び出されるのが，エラー処理ルーチンです。エラー処理ルーチンの目的は，エラーの内容，あるいはターゲット・システムに応じたディバグを行えるようにすることです。

(1) 機　能

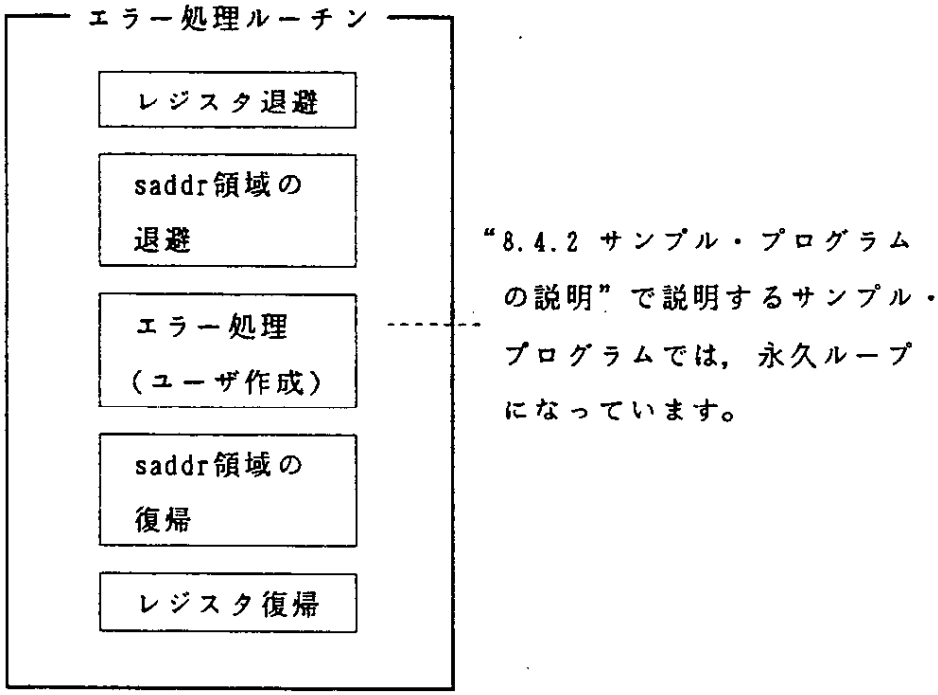
ライブラリ実行時に，エラーが発生したとき，エラーの内容，あるいはターゲット・システムに応じて，適切な処理を行います。チェックできるエラーの内容には，以下の６種類があります。

- スタック・オーバーフロー
- ０除算のエラー
- ポインタ・アクセスのエラー
- 演算によるオーバーフロー
- sfrアクセスのエラー
- ＲＯＭ化領域のエラー

(2) 構　成

エラー処理ルーチンのサンプル・プログラムの構成を，“図８－２　エラー処理ルーチンの構成”に示します。

図８－２　エラー処理ルーチンの構成



(3) ライブラリとの関連

“表８－１１　エラー処理ルーチン”は，各エラー処理ルーチンが呼び出された原因，呼び出した関数などを示します。

表８－１１　エラー処理ルーチン

ｴﾗｰ処理ﾙｰﾁﾝ	呼び出された原因	呼び出した関数	条　　件
errstk.asm	ｽﾀｯｸ・ｵｰﾊﾞｰﾌﾛｰ	ライブラリ・ファイルの中の関数	コンパイル時に，実行時ｴﾗｰ・ﾁｪｯｸ・ｵﾌﾟｼｮﾝを指定
errdiv.asm	０除算		
errptr.asm	ﾎﾟｲﾝﾀ・ｱｸｾｽ・ｴﾗｰ		
errovf.asm	演算によるｵｰﾊﾞｰﾌﾛｰ		
errsfr.asm	sfrｱｸｾｽ・ｴﾗｰ		
errini.asm	ROM化処理ｴﾗｰ	ｽﾀｰﾄｱｯﾌﾟ・ﾙｰﾁﾝ estartr	リンク時にestartrを使用

ライブラリ・ファイル中の各関数が，どのエラー・チェックを行うことができるかについては，“4.6.2　エラー・チェック・ライブラリ名”を参照してください。

例えば，数学関数divは，コンパイル時に，実行時エラー・チェック指定オプション（－Ｌ）を指定することにより，演算によるオーバーフローをチェックします。div関数実行中に，オーバーフローが発生すると，エラー処理ルーチンerrovf.asmが呼び出されます。

8.4.2 サンプル・プログラムの説明

ここでは，エラー処理ルーチンのサンプル・プログラムの中から，errstk.asmについて説明します。

他のエラー処理ルーチンは，外部定義宣言のシンボル名を除いて，errstk.asmと同じです。

サンプル・プログラムは，７８Ｋ／Ⅲのものを使用します。

●ｅｒｒｓｔｋ．ａｓｍ

```
$INCLUDE 'CHIP.INC'
$INCLUDE 'DEFINE.INC'

        PUBLIC  _@errstk        …他のエラー処理ルーチンは，このシンボル名を除いて
                                  同じ内容です。
@@CODE  CSEG

_@errstk:
        PUSH    RP0,RP1,RP2,RP3,RP4,RP5,RP6,RP7    — ①レジスタの退避
;
;       MOVW    AX,_@D_FUNC     ┐
;       PUSH    AX              │
;       MOVW    AX,_@D_LINE     │
;       PUSH    AX              │
                ⋮               │  ②Ｃコンパイラが使用するsaddr領域の退避
;       MOVW    AX,_@FARG1H     │
;       PUSH    AX              │
;       MOVW    AX,_@FARG0L     │
;       PUSH    AX              ┘
;
;_@errstk + 9AH
L_A:                            ┐  ③エラー処理ルーチン
        BR      $L_A            ┘
;
;       POP     AX              ┐
;       MOVW    _@FARG0H,AX     │
;       POP     AX              │
;       MOVW    _@FARG0L,AX     │
                ⋮               │  ④saddr領域の復帰
;       POP     AX              │
;       MOVW    _@D_LINE,AX     │
;       POP     AX              │
;       MOVW    _@D_FUNC,AX     ┘
;
;       POP     RP0,RP1,RP2,RP3,RP4,RP5,RP6,RP7    — ⑤レジスタの復帰
;       RET
;
        END
```

注　ＣＣ７８Ｋ３ V2.0，ＣＣ７８Ｋ２ V1.0 以降では，①，②，④，⑤はコメントにしていますが，実際のユーザ・プログラムでは，コメントを取り除いてご使用ください。

errstk.asmにある①～⑤について，説明します。

①レジスタの退避

・RP0～RP7のレジスタの内容を退避させます。

②Cコンパイラが使用するsaddr領域の退避

・Cコンパイラが使用するsaddr領域の内容を退避させます。詳細は“言語編　付録D　saddr領域のラベル一覧”を参照してください。

③エラー処理ルーチン

・ここには，エラー処理ルーチンを作成して埋め込んでください。
サンプル・プログラムでは，永久ループとなっています。

④saddr領域の復帰

・Cコンパイラが使用するsaddr領域の内容を復帰させます。

⑤レジスタの復帰

・RP0～RP7のレジスタの内容を退避させます。

他のエラー処理ルーチン（errdiv.asm, errptr.asm, errovf.asm, errsfr.asm, errini.asm）についても内容は同じです。

エラーが発生したときには，saddr領域のディバグ用予約領域（３０ページ）に，以下で示すようなエラー情報が格納されます。

_@D_FUNC　…　エラーが発生した関数名の文字列を格納します。

_@D_LINE　…　エラーが発生したCソース・ファイル中の行番号を格納します。

8.4.3 改良のポイント

本パッケージで提供するエラー処理ルーチンのサンプル・プログラムは，エラー処理の内容が永久ループとなっています。この部分に関しては，エラーの内容，あるいはユーザの開発するターゲット・システムに応じて，アセンブラのプログラムの埋め込み等の処理を行ってください。

例）
- エラー発生を知らせるブザーを鳴らす。
- エラー発生を知らせるＬＥＤを点滅させる。
- エラー内容をプリンタに出力させる。
- 他の関数を呼び出す。

第９章　エラー・メッセージ

9.1 エラー・メッセージの種類

コンパイラが出力するエラー・メッセージには，次の１０種類があります。

(0) コマンド行に対するエラー・メッセージ

(1) 内部エラー，メモリに対するエラー・メッセージ

(2) 文字に対するエラー・メッセージ

(3) 構成要素に対するエラー・メッセージ

(4) 変換に対するエラー・メッセージ

(5) 式に対するエラー・メッセージ

(6) 文に対するエラー・メッセージ

(7) 宣言，関数定義に対するエラー・メッセージ

(8) 前処理指令に対するエラー・メッセージ

(9) 致命的なファイルＩ／Ｏ，許されないＯＳ上での起動に対するエラー・メッセージ

9.2 エラー・メッセージ一覧

エラー・メッセージ一覧を活用する前に，エラー番号の形式を理解しておく必要があります。

エラー番号は，エラー・メッセージの種類とエラーに対するコンパイラの処理を示しています。したがって，エラー・メッセージ一覧にコンパイラの処理が記述されていない場合には，エラー番号を参照してください。

エラー番号の形式は，次のようになります。

Ａ／Ｆ／Ｗnnn

Ａ／Ｆ／Ｗ（アルファベット）については，

Ａ：ＡＢＯＲＴ　・エラー・メッセージ出力後，直ちにコンパイル処理を終了します。オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルは，出力されません。

F：FATAL　・エラー・メッセージ出力後，エラー部分を無視し処理を続行します。オブジェクト・モジュール・ファイル，アセンブラ・ソース・モジュール・ファイルは，出力されません。

W：WARNING　・ワーニング・メッセージを出力後，処理を続行します。オプションで指定したファイルが出力されます。

nnn（３桁の数字）については，

００１～　コマンド行に対するエラー・メッセージ
１０１～　内部エラー，メモリに対するエラー・メッセージ
２０１～　文字に対するエラー・メッセージ
３０１～　構成要素に対するエラー・メッセージ
４０１～　変換に対するエラー・メッセージ
５０１～　式に対するエラー・メッセージ
６０１～　文に対するエラー・メッセージ
７０１～　宣言，関数定義に対するエラー・メッセージ
８０１～　前処理指令に対するエラー・メッセージ
９０１～　致命的なファイルＩ／Ｏ，許されないＯＳ上での起動に対するエラー・メッセージ

とします。

注　ファイル名に文法的誤りがあった場合には，メッセージにファイル名が付加されます。またデバイス型ファイルを指定禁止箇所で指定した場合は，指定文字列がそのまま出力されます。それ以外のときは，ドライブ名とパス名および拡張子が必ず付加されます。エラー・メッセージは，開発するＣコンパイラの言語仕様により追加，変更，削除もありえます。

(0) コマンド行（１／３）

A001	メッセージ	Missing input file
	原　　因	－F，－－以外のオプションのみの指定で入力ファイルが１つも指定されていないか，実行プログラム名のみの指定で起動されたがヘルプ・ファイルが存在しません。
A002	メッセージ	Too many input files
	原　　因	入力ファイルの総数が制限を越えて指定されました。
A003	メッセージ	Unrecognized string '指定されたもの'
	原　　因	会話形式のコマンド行にオプション以外のものが指定されました。
A004	メッセージ	Illegal file name 'ファイル名'
	原　　因	ファイル名として形式、文字、文字数のいずれかに誤りがあります。
A005	メッセージ	Illegal file specification 'ファイル名'
	原　　因	ファイル名に不当なものが指定されました。
A006	メッセージ	File not found 'ファイル名'
	原　　因	指定された入力ファイルが存在しません。
A007	メッセージ	Input file specification overlapped 'ファイル名'
	原　　因	入力ファイル名が重複して指定されました。
A008	メッセージ	File specification conflicted 'ファイル名'
	原　　因	入出力ファイル名が重複して指定されました。
A009	メッセージ	Unable to make file 'ファイル名'
	原　　因	指定された出力ファイルがリード・オンリ・ファイルとしてすでに存在しているため，作成することができません。
A010	メッセージ	Directory not found 'ファイル名'
	原　　因	出力ファイル名中に存在しないドライブ，またはディレクトリが含まれています。
A011	メッセージ	Illegal path 'ファイル名'
	原　　因	パラメータにパス名を指定するオプションで，パス名以外が指定されました。

(0) コマンド行（２／３）

A012	メッセージ	Missing parameter 'オプション'
	原　　因	必要なパラメータが指定されていません。
A013	メッセージ	Parameter not needed 'オプション'
	原　　因	不要なオプション・パラメータが指定されました
A014	メッセージ	Out of range 'オプション'
	原　　因	オプション・パラメータの指定数値が範囲外です。
A015	メッセージ	Parameter is too long 'オプション'
	原　　因	オプション・パラメータの文字数が制限を越えて指定されました。
A016	メッセージ	Illegal parameter 'オプション'
	原　　因	オプション・パラメータの文法に誤りがあります。
A017	メッセージ	Too many parameters 'オプション'
	原　　因	オプション・パラメータの総数が制限を越えました。
A018	メッセージ	Option is not recognized 'オプション'
	原　　因	誤ったオプションが指定されました。
A019	メッセージ	Parameter file nested
	原　　因	パラメータ・ファイル中に－Ｆオプションが指定されました。
A020	メッセージ	Parameter file read error 'ファイル名'
	原　　因	パラメータ・ファイルの読み込みに，失敗しました。
A021	メッセージ	Memory allocation failed
	原　　因	メモリ・アロケーションに失敗しました。
W022	メッセージ	Same category option specified - ignored 'オプション'
	原　　因	相反するオプションが重複して指定されました。
	コンパイラの処理	あとに指定されたオプションを優先します。
W023	メッセージ	Incompatible chip name
	原　　因	コマンド行上のデバイス種別とソース中のデバイス種別が異なります。
	コンパイラの処理	コマンド行上のデバイス種別を優先します。
A024	メッセージ	Illegal chip specifier on command line
	原　　因	コマンド行上のデバイス種別が異なります。

(0) コマンド行（３／３）

Ｗ０２５	メッセージ	'-G' option specifide - ignored '-QSZ'
	原　　因	・ＣＣ７８Ｋ３の場合 ディバッグする指定オプション（－Ｇ）が指定されたため，最適化指定オプション（－Ｑ）は無視されます。 ・ＣＣ７８Ｋ３以外の場合 ディバッグ指定オプション（－Ｇ）が指定されたため，最適化指定オプション（－ＱＳＺ）は無視されます。

(1) 内部エラー，メモリに関するもの（１／２）

Ｆ１０１	メッセージ	Internal error
	原　　因	内部エラーが起こりました。
Ｆ１０２	メッセージ	Too many errors
	原　　因	Ｆａｔａｌエラーの合計が３０を越えました。
	コンパイラの処理	処理を継続しますが，これ以降のエラー・メッセージは出力しません。
Ｆ１０３	メッセージ	Intermediate file error
	原　　因	中間ファイルの内容に誤りがあります。
Ｆ１０４	メッセージ	Illegal use of register
	原　　因	レジスタの使い方に誤りがあります。
Ｆ１０５	メッセージ	Register overflow : simplify expression
	原　　因	式が複雑すぎるので使用できるレジスタがなくなりました。
Ａ１０６	メッセージ	Stack overflow
	原　　因	スタックのオーバフローが起こりました。
Ｆ１０７	メッセージ	Symbol table overflow
	原　　因	シンボル・テーブルのオーバフローが起こりました。
Ｆ１０８	メッセージ	Compiler limit : too much automatic data in function
	原　　因	関数のオートマティック変数に割り当てられた領域が６４Ｋバイトの制限を越えました。
Ｆ１０９	メッセージ	Compiler limit : too much parameter of function
	原　　因	関数のパラメータに割り当てられた領域が，６４Ｋバイトの制限を越えました。

(1) 内部エラー，メモリに関するもの（2／2）

F 1 1 0	メッセージ	Compiler limit : too much code defined in file
	原　　因	ファイル内のコードに割り当てられた領域が，６４Ｋバイトの制限を越えました。
F 1 1 1	メッセージ	Compiler limit : too much global data defined in file
	原　　因	ファイル内のグローバル変数に割り当てられた領域が６４Ｋバイトの制限を越えました。

(2) 文字

F 2 0 1	メッセージ	Unknown character '１６進数'
	原　　因	指定された内部コードを持つ文字は認識できません。
F 2 0 2	メッセージ	Unexpected EOF
	原　　因	関数の途中でファイルが終了しました。

(3) 構成要素（1／3）

F 3 0 1	メッセージ	Syntax error
	原　　因	構文エラーが起こりました。
F 3 0 3	メッセージ	Expected identifier
	原　　因	識別子が必要です。
F 3 0 4	メッセージ	Compiler limit : too long identifier '識別子'
	原　　因	指定された識別子が長すぎます。
F 3 0 5	メッセージ	Compiler limit : too many identifier with block scope
	原　　因	１つのブロック内でブロック・スコープを持つシンボルの数が多すぎます。
F 3 0 6	メッセージ	Illegal index , indirection not allowed
	原　　因	ポインタの値をとらない式に添字が使われています。
F 3 0 7	メッセージ	Call of non-function '変数名'
	原　　因	変数名が関数名として使われています。

(3) 構成要素（２／３）

Ｆ３０８	メッセージ	Improper use of a typedef name
	原　　因	typedef名が正しく使われていません。
Ｗ３０９	メッセージ	Unused '変数名'
	原　　因	指定された変数はソース中で宣言されていますが，まったく使われていません。
Ｗ３１０	メッセージ	'変数名' is assigned a value which is never used
	原　　因	指定された変数は代入文には使われていますが，その他ではまったく使われていません。
Ｆ３１１	メッセージ	Number syntax
	原　　因	定数の表現が誤っています。
Ｆ３１２	メッセージ	Illegal octal digit
	原　　因	８進数字としてふさわしくないものがあります。
Ｆ３１３	メッセージ	Illegal hexadecimal digit
	原　　因	１６進数字としてふさわしくないものがあります。
Ｆ３１４	メッセージ	Too big constant
	原　　因	定数が大きすぎて表現できません。
Ｆ３１５	メッセージ	Too small constant
	原　　因	文字が小さすぎて表現できません。
Ｆ３１６	メッセージ	Too many character constants
	原　　因	文字定数が２文字を越えています。
Ｆ３１７	メッセージ	Empty character constant
	原　　因	文字定数’’の中が空になっています。
Ｆ３１８	メッセージ	No terminated string literal
	原　　因	文字列の終わりに‘”’がありません。
Ｆ３２０	メッセージ	No null terminator in string literal
	原　　因	文字列リテラル中にヌル文字を付加していません。
Ｆ３２１	メッセージ	Compiler limit : too many characters in string literal
	原　　因	文字列リテラルの文字数が５０９を越えました。
Ｆ３２２	メッセージ	Ellipsis requires three periods
	原　　因	コンパイラは，“．．”を検出したが“．．．”がくる必要があります。

(3) 構成要素（３／３）

Ｆ３２３	メッセージ	Missing '区切り子'
	原　　因	区切り子に誤りがあります。
Ｆ３２４	メッセージ	Too many }' s
	原　　因	'{' と '}' が正しく対応していません。
Ｆ３２５	メッセージ	No terminated comment
	原　　因	コメントの終わりに“＊／”がありません。

(4) 変換（１／２）

Ｗ４０１	メッセージ	Conversion may lose significant digits
	原　　因	longからintへの変換などが行われています。
Ｆ４０２	メッセージ	Incompatible type conversion
	原　　因	代入文で許されない型変換が行われています。
Ｆ４０３	メッセージ	Illegal indirection
	原　　因	整数型の式に＊演算子が使われています。
Ｆ４０４	メッセージ	Incompatible structure type conversion
	原　　因	構造体どうしまたは構造体への代入文で両辺の型が異なっています。
Ｆ４０５	メッセージ	Illegal lvalue
	原　　因	左辺値として正しくないものがあります。
Ｆ４０６	メッセージ	Cannot modify a const object '変数名'
	原　　因	const属性の変数の書き替えを行っています。
Ｆ４０７	メッセージ	Cannot write for read/only sfr 'sfr名'
	原　　因	read onlyのsfrに対し書き込みを行っています。
Ｆ４０８	メッセージ	Cannot read for write/only sfr 'sfr名'
	原　　因	write onlyのsfrに対し読み出しを行っています。
Ｆ４０９	メッセージ	Write invalid data for sfr 'sfr名'
	原　　因	sftに対して不正なデータ書き込みを行っています。
Ｗ４１０	メッセージ	Illegal pointer conversion
	原　　因	ポインタとポインタ以外のものの変換が行われています。

(4) 変換（２／２）

W４１１	メッセージ	Illegal pointer combination
	原　　因	ポインタ同士で異なる型のものを混合して使用しています。
W４１２	メッセージ	Illegal pointer combination in conbitional expression
	原　　因	ポインタ同士で異なる型のものを条件式に使用しています。
W４１３	メッセージ	Illegal structure pointer combination
	原　　因	型の異なる構造体へポインタを混合して使用しています。
F４１４	メッセージ	Expected pointer
	原　　因	ポインタが必要です。

(5) 式（１／４）

Ｆ５０１	メッセージ	Expression syntax
	原　　因	式の構文エラーが起こりました。
Ｆ５０２	メッセージ	Compiler limit : too many parentheses
	原　　因	式中のかっこのネストが３２を越えました。
Ｗ５０３	メッセージ	Possible use of '変数名' before definition
	原　　因	変数に値が代入される前に，その変数を使用しています。
Ｗ５０４	メッセージ	Possibly incorrect assignment
	原　　因	if,while,do文などで条件式のおもな演算子が代入演算子です。
Ｆ５０７	メッセージ	Expected integral index
	原　　因	配列の添字に許されるのは整数型の式だけです。
Ｗ５０８	メッセージ	Too many actual arguments
	原　　因	関数呼び出しで指定された引数の数が，引数の型のリストまたは関数定義で指定されたパラメータの数より多い状態です。
Ｗ５０９	メッセージ	Too few actual arguments
	原　　因	関数呼び出しで指定された引数の数が，引数の型のリストまたは関数定義で指定されたパラメータの数より少ない状態です。
Ｗ５１０	メッセージ	Pointer mismatch in function '関数名'
	原　　因	与えられた引数が，引数の型のリストや関数定義で指定されたものとは異なるポインタの型を持ちます。
Ｗ５１１	メッセージ	Different argument types in function '関数名'
	原　　因	関数の呼び出しで与えられた引数の型が，引数の型のリストや関数定義と一致していません。
Ｆ５１２	メッセージ	Function call in norec function
	原　　因	norec関数中で関数呼び出しを行っています。
Ｆ５１３	メッセージ	Illegal structure/union member 'メンバ名'
	原　　因	構造体の参照で，定義されていないメンバを指しています。

(5) 式（2／4）

F514	メッセージ	Expected structure/union pointer
	原　　因	'－＞' 演算子の前の式が，構造体または共用体へのポインタではなく構造体または共用体の名前です。
F515	メッセージ	Expected structure/union name
	原　　因	'．' 演算子の前の式が，構造体または共用体の名前ではなく構造体または共用体へのポインタです。
F516	メッセージ	Zero sized structure `構造体名'
	原　　因	構造体の大きさが0です。
F517	メッセージ	Illegal structure operation
	原　　因	構造体に使用できない演算子を使用しています。
F518	メッセージ	Illegal structure/union comparison
	原　　因	2個の構造体または共用体を比較することはできません。
F519	メッセージ	Illegal bit field operation
	原　　因	ビット・フィールドに対して許されない記述があります。
F520	メッセージ	Illegal use of pointer
	原　　因	ポインタに対して使用できる演算子は，加減，代入，関係，間接（＊），メンバ参照（－＞）だけです。
W522	メッセージ	Ambiguous operators need parentheses
	原　　因	2つのシフト，関係，ビット論理演算子が，かっこなしに連続して現れています。
F523	メッセージ	Illegal bit type operation
	原　　因	bit型変数に対して許されない演算を行っています。
F524	メッセージ	'&' on constant
	原　　因	定数のアドレスは得られません。
F525	メッセージ	'&' requires lvalue
	原　　因	'&' 演算子は左辺値に代入する式にのみ使用可能です。
F526	メッセージ	'&' on register variable
	原　　因	レジスタ変数のアドレスは得られません。

(5) 式（３／４）

Ｆ５２７	メッセージ	'&' on ignored
	原　　因	ビット・フィールド，bit型変数のアドレスは得られません。
Ｗ５２８	メッセージ	'&' is not allowed array/function , ignored
	原　　因	配列名や関数名に＆演算子を付ける必要はありません。
Ｆ５２９	メッセージ	Sizeof returus zero
	原　　因	sizeof式の値が０になっています。
Ｆ５３０	メッセージ	Illegal sizeof operand
	原　　因	sizeof式のオペランドは，識別子または型名でなければなりません。
Ｆ５３１	メッセージ	Disallowed conversion
	原　　因	不正なキャストを行っています。
Ｆ５３２	メッセージ	Pointer on left , needs integral rigth : 'operator'
	原　　因	左辺オペランドがポインタであるので，右辺オペランドは整数値でなければなりません。
Ｆ５３３	メッセージ	Invalid left-or-right operand : 'operator'
	原　　因	左辺または右辺オペランドが，演算子に対して不正です。
Ｆ５３４	メッセージ	Divide check
	原　　因	／演算，％演算の除数が０です。
Ｆ５３５	メッセージ	Invalid pointer addition
	原　　因	２つポインタを加算してはなりません。
Ｆ５３６	メッセージ	Must be integral value addition
	原　　因	ポインタに加算できるものは整数値のみです。
Ｆ５３７	メッセージ	Illegal pointer subtraction
	原　　因	ポインタ同士の減算は同じ型でなければなりません。
Ｆ５３８	メッセージ	Illegal conditional operator
	原　　因	条件演算子が正しく記述されていません。
Ｆ５３９	メッセージ	Expected constant expression
	原　　因	定数式が必要です。

(5) 式（４／５）

W５４０	メッセージ	Constant out of range in comparison
	原　　因	定数部分式が，もう一方の部分式の型によって許される範囲外の値と比較されています。
F５４１	メッセージ	Function argument has void type
	原　　因	関数の引数がvoid型です。
F５４２	メッセージ	Arguments mismatch : 1M byte function '%s'
	原　　因	１Ｍ byte関数の引数に誤りがあります。（７８Ｋ／Ⅱの場合のみ）

(6) 文（１／２）

F６０２	メッセージ	Compiler limit : too many characters in logical source line
	原　　因	論理ソース行の文字数が５０９を越えました。
F６０３	メッセージ	Compiler limit : too many labels
	原　　因	ラベル数が３３を越えました。
F６０４	メッセージ	Case not in switch
	原　　因	case文が正しい位置に記述されていません。
F６０５	メッセージ	Duplicate case 'ラベル名'
	原　　因	switch文の中で同じcaseラベルが２度以上記述されています。
F６０６	メッセージ	Non constant case expression
	原　　因	case文で整数定数以外のものを指定しています。
F６０７	メッセージ	Compiler limit : too many case labels
	原　　因	switch文のcaseラベルが２５７を越えました。
F６０８	メッセージ	Default not in switch
	原　　因	default文が正しい位置に記述されていません。
F６０９	メッセージ	More than one 'default'
	原　　因	switch文の中でdefault文が複数記述されています。
F６１０	メッセージ	Compiler limit : block nest level too depth
	原　　因	ブロックのネストが４５を越えました。

(6) 文（2／2）

F611	メッセージ	Inappropriate 'else'
	原　因	ifとelseの対応がとれていません。
W613	メッセージ	Loop entered at top of switch
	原　因	switch文の最後にwhile,do,for等を指定しています。
W615	メッセージ	Statement not reached
	原　因	絶対に到達しない文があります。
F617	メッセージ	Do statement must have 'while'
	原　因	doの終わりにはwhileが必要です。
F620	メッセージ	Back/continue error
	原　因	break,continue文の位置が誤っています。
F621	メッセージ	Void function '関数名' cannot return value
	原　因	void宣言した関数が値を返しています。
W622	メッセージ	No return Value
	原　因	値を返すべき関数が値を返していません。
F623	メッセージ	No effective code and data,_ cannot create output file
	原　因	横文解析時に中間文が作成されなかった。

(7) 宣言，関数定義（１／５）

F７０１	メッセージ	External definition syntax
	原　　因	関数が正しく定義されていません。
F７０２	メッセージ	Too many callt functions
	原　　因	callt関数の宣言が多すぎます。
F７０３	メッセージ	Function has illegal storage class
	原　　因	関数が不正な記憶クラスで指定されています。
F７０４	メッセージ	Function returns illegal type
	原　　因	関数の返り値が不正な型です。
F７０５	メッセージ	Too many parameters in noauto or norec function
	原　　因	noauto,norec関数のパラメータが多すぎます。
F７０６	メッセージ	Parameter list error
	原　　因	関数パラメータ・リスト中に誤りがあります。
F７０７	メッセージ	Not parameter '文字列'
	原　　因	関数定義でパラメータがないものを宣言しています。
F７１０	メッセージ	Illegal storage class
	原　　因	関数の外部でauto,register宣言が行われています。
F７１１	メッセージ	Undeclared '変数名' in function '関数名'
	原　　因	宣言されていない変数が使用されています。
F７１２	メッセージ	Declaration syntax
	原　　因	宣言文が文法にあっていません。
F７１３	メッセージ	Redefined '変数名'
	原　　因	同じ変数が２回以上定義されています。
F７１４	メッセージ	Too many register variables
	原　　因	レジスタ変数の宣言が多すぎます。
F７１５	メッセージ	Too many sreg variables
	原　　因	sreg変数の宣言が多すぎます。
F７１６	メッセージ	Not allowed automatic date in noauto function
	原　　因	noauto関数中ではオートマティック変数は使用できません。
F７１７	メッセージ	Too many automatic data in norec function
	原　　因	norec関数のオートマティック変数が，　多すぎます。

(7) 宣言，関数定義（２／５）

F７１８	メッセージ	Too many bit type variables
	原　　因	bit型変数が多すぎます。
F７１９	メッセージ	Illegal use of type
	原　　因	型名が不正に使用されています。
F７２０	メッセージ	Illegal void type for '識別子'
	原　　因	識別子をvoidで宣言しています。
F７２１	メッセージ	Illegal type for regiaster declaration
	原　　因	register宣言が，許されない型に指定されています。
	コンパイラの処理	register宣言を無視して処理を継続します。
F７２２	メッセージ	Illegal type for sreg declaration
	原　　因	sreg宣言が許されない型に指定されています。
F７２３	メッセージ	Illegal type for parameter in noauto or norec function
	原　　因	noauto,norec関数のパラメータの型が大きすぎます。
F７２４	メッセージ	Structure redefinition
	原　　因	同じ構造体が再定義されています。
F７２５	メッセージ	Illegal zero sized structure member
	原　　因	構造体のメンバとして取られている領域が１以上確保されていません。
F７２６	メッセージ	Function cannot be structure/union member
	原　　因	関数は，構造体または共用体のメンバであってはなりません。
F７２７	メッセージ	Unknown size structure/union '名前'
	原　　因	サイズが未定義の構造体，または共用体があります。
F７２８	メッセージ	Compiler limit : too many structure/union members
	原　　因	構造体または共用体のメンバが127を越えています。
F７２９	メッセージ	Compiler limit : structure/union nesting
	原　　因	構造体または共用体のネストが１５を越えています。

(7) 宣言，関数定義（３／５）

Ｆ７３０	メッセージ	Bit field outside of structure
	原　　因	構造体の外でビット・フィールドの宣言が行われています。
Ｆ７３１	メッセージ	Illegal bit field type
	原　　因	ビット・フィールドの型に整数型以外の型を指定しています。
Ｆ７３２	メッセージ	Too long bit field size
	原　　因	ビット・フィールド宣言のビット指定数がその型のビット数を越えています。
Ｆ７３３	メッセージ	Negative bit field size
	原　　因	ビット・フィールド宣言のビット指定数が負です。
Ｆ７３４	メッセージ	Illegal enumeration
	原　　因	列挙型宣言が文法にあっていません。
Ｆ７３５	メッセージ	Illegal enumeration constant
	原　　因	列挙定数が不正です。
Ｆ７３６	メッセージ	Compiler limit : too many enumeration constants
	原　　因	列挙定数の数が１２７を越えました。
Ｆ７３７	メッセージ	Undeclared structure/union/enum tag
	原　　因	タグが宣言されていません。
Ｆ７３８	メッセージ	Compiler limit : too many pointer modifying
	原　　因	ポインタの定義で間接演算子（＊）の数が１２を越えました。
Ｆ７３９	メッセージ	Expected constant
	原　　因	配列の宣言で添字に変数を使用しています。
Ｆ７４０	メッセージ	Negative subscript
	原　　因	配列の大きさの指定が負です。
Ｆ７４１	メッセージ	Unknown size array '配列名'
	原　　因	配列の大きさが不定です。
Ｆ７４２	メッセージ	Compiler limit : too many array modifying
	原　　因	配列の宣言が１２次元を越えています。

(7) 宣言，関数定義（４／５）

F７４３	メッセージ	Array element type cannot be function
	原　因	関数の配列は許されません。
F７４４	メッセージ	Zero sized array '配列名'
	原　因	定義した配列の要素数が０です。
F７４５	メッセージ	Expected function prototype
	原　因	関数プロトタイプ宣言がありません。
F７４７	メッセージ	Function prototype mismatch
	原　因	関数プロトタイプ宣言に誤りがあります。
F７４８	メッセージ	A function is declared as a parameter
	原　因	関数が引数として宣言されています。
F７４９	メッセージ	Unused parameter 'パラメータ名'
	原　因	パラメータが使用されていません。
F７５０	メッセージ	Initializer syntax
	原　因	初期化が文法にあっていません。
F７５１	メッセージ	Illegal initialization
	原　因	初期値設定の定数がその変数の型にあっていません。
F７５２	メッセージ	Undeclared initializer name '名前'
	原　因	初期化子名が宣言されていません。
F７５３	メッセージ	Cannot initialize static with automatic
	原　因	オートマチック変数を使って，スタティック変数を初期化できません。
F７５４	メッセージ	Cannot initialize array in function '関数名'
	原　因	関数内の配列は初期化できません。
F７５５	メッセージ	Cannot initialize structure/union in function '関数名'
	原　因	関数内の構造体，共用体は初期化できません。
F７５６	メッセージ	Too many initializers '配列名'
	原　因	宣言された配列の要素数より初期値の方が多い。
F７５７	メッセージ	Too many structure initializers
	原　因	宣言された構造体のメンバ数より初期値の方が多い。

(7) 宣言，関数定義（５／５）

F７５８	メッセージ	Cannot initialize a function '関数名'
	原　　因	関数は初期化できません。
F７５９	メッセージ	Compiler limit : initializers too deeply nested
	原　　因	初期化要素のネストの深さが制限を越えました。
F７６０	メッセージ	Not including floating function
	原　　因	浮動小数点演算をサポートしていません。
F７７０	メッセージ	Parameters are not allowed for interrupt function
	原　　因	interrupt関数には引数は許されません。
W７０８	メッセージ	Already declared symbol '変数名'
	原　　因	同じ変数がすでに宣言されています。

(8) 前処理指令（１／４）

Ｆ８０１	メッセージ	Undefined control
	原　　因	#で始まるものでキーワードとして認識できないものがあります。
Ｆ８０２	メッセージ	Illegal preprocess directive
	原　　因	プリプロセス指令に誤りがあります。
Ｆ８０３	メッセージ	Unexpected non-whitespace before preprocess directive
	原　　因	プリプロセス指令の前に空白文字以外の文字があります。
Ｗ８０４	メッセージ	Unexpected characters following 'プリプロセス指令' directive newline expected
	原　　因	プリプロセス指令の後に余分な文字があります。
Ｆ８０５	メッセージ	Misplaced else or elif
	原　　因	#if,#ifdef,#ifndefと#else,#elifの対応がとれていません。
Ｆ８０６	メッセージ	Misplaced endif
	原　　因	#if,#ifdef,#ifndefと#endifの対応がとれていません。
Ｆ８０７	メッセージ	Compiler limit : too many conditional inclusion nesting
	原　　因	条件コンパイルのネストが２５５を越えました。
Ｆ８１０	メッセージ	Cannot find include file 'ファイル名'
	原　　因	インクルード・ファイルが見つかりません。
Ｆ８１１	メッセージ	Too long file name 'ファイル名'
	原　　因	ファイル名が長すぎます。
Ｆ８１２	メッセージ	Include directive syntax
	原　　因	#include文の定義でファイル名が" "または< >で正正しく囲まれていません。
Ｆ８１３	メッセージ	Compiler limit : too many include nesting
	原　　因	インクルード・ファイルのネストが８を越えました。

(8) 前処理指令（２／４）

Ｆ８１４	メッセージ	Illegal macro name
	原　　因	マクロ名が正しくありません。
Ｗ８１６	メッセージ	Redefined macro name 'マクロ名'
	原　　因	マクロ名が再定義されています。
Ｗ８１７	メッセージ	Redefined system macro name 'マクロ名'
	原　　因	システム・マクロ名が再定義されています。
Ｆ８１８	メッセージ	Redeclared parameter in macro 'マクロ名'
	原　　因	マクロ定義内のパラメータ・リストに同じ識別子が現れています。
Ｗ８１９	メッセージ	Mismatch number of parameter 'マクロ名'
	原　　因	#defineで定義したパラメータの数と参照するときのパラメータの数が異なっています。
Ｆ８２１	メッセージ	Illegal macro parameter 'マクロ名'
	原　　因	関数形式マクロで（）内の記述が正しくありません。
Ｆ８２２	メッセージ	Missing) 'マクロ名'
	原　　因	関数形式マクロで#define定義の同じ行内に ‘)’ が見つかりません。
Ｆ８２３	メッセージ	Too long macro expansion 'マクロ名'
	原　　因	マクロ展開時の実引数が長すぎます。
Ｆ８２４	メッセージ	Compiler limit : too long macro name 'マクロ名'
	原　　因	マクロ名が長すぎます。
Ｗ８２５	メッセージ	Macro recursion 'マクロ名'
	原　　因	#define定義がリカーシブになっています。
Ｆ８２６	メッセージ	Compiler limit : too many macro defines
	原　　因	マクロ定義数が１０２４を越えました。
Ｆ８２７	メッセージ	Compiler limit : too many macro parameters
	原　　因	１つのマクロ定義，呼び出しのパラメータが３１を越えました。

(8) 前処理指令（３／４）

Ｆ８２８	メッセージ	Not allowed #undef for system macro name 'マクロ名'
	原　　因	システム・マクロ名が#undefにより指定されています。
Ｗ８２９	メッセージ	Unrecognized pragma '文字列'
	原　　因	この文字列はサポートしていません。
Ｆ８３０	メッセージ	No chip specifier : #pragma pc()
	原　　因	デバイス種別指定がありません。
Ｆ８３１	メッセージ	Illegal chip specifier : '#ptagma pc(デバイス種別)'
	原　　因	デバイス種別指定に誤りがあります。
Ｆ８３２	メッセージ	Duplicated chip specifier
	原　　因	デバイス種別指定が重複しています。
Ｆ８３３	メッセージ	Expected #asm
	原　　因	#asm文がありません。
Ｆ８３４	メッセージ	Expected #endasm
	原　　因	#endasm文がありません。
Ｆ８３５	メッセージ	Too many characters in assembler source line
	原　　因	アセンブラ・ソースの１行が長すぎます。
Ｗ８３６	メッセージ	Expected assembler source
	原　　因	#asmと#endasmの間にアセンブラ・ソースがありません。
Ｗ８３７	メッセージ	Output assmbler source file , not object file
	原　　因	オブジェクト・ファイルの代わりにアセンブラ・ソースを出力します。

(8) 前処理指令（４／４）

Ｆ８３８	メッセージ	Duplicated pragma VECT '%s'
	原　　因	#pragma VECT '文字列'が重複しています。
Ｆ８３９	メッセージ	Unrecognized pragma VECT '%s'
	原　　因	認識されない#pragma VECT '文字列'があります。
Ｆ８４０	メッセージ	Undefined interrupt function '%s'
	原　　因	interrupt関数'%s'が宣言されていません。
Ｆ８４１	メッセージ	Unrecgnized pragma TABLE '%s'
	原　　因	認識されない#pragma TABLE '文字列'があります。

(9) Ｉ／Ｏ，最適化（１／３）

Ａ９０１	メッセージ	File I/O error
	原　　因	ファイルの入出力の際に物理的なＩ／Ｏエラーが発生しました。
Ａ９０２	メッセージ	Cannot open 'ファイル名'
	原　　因	ファイルのopenエラーが発生しました。
Ａ９０３	メッセージ	Cannot open overlay file 'ファイル名'
	原　　因	オーバレイ・ファイルのopenエラーが発生しました。
Ａ９０４	メッセージ	Cannot open temp
	原　　因	入力用のテンポラリ・ファイルのopenエラーが発生しました。
Ａ９０５	メッセージ	Cannot create 'ファイル名'
	原　　因	ファイルのcreateエラーが発生しました。
Ａ９０６	メッセージ	Cannot create temp
	原　　因	出力用のテンポラリ・ファイルのcreateエラーが発生しました。
Ａ９０７	メッセージ	No available data block
	原　　因	ドライブのファイル容量の不足によりテンポラリ・ファイルが作成できません。
Ａ９０８	メッセージ	No available directory space
	原　　因	ドライブのディレクトリ・エリアの不足によりテンポラリ・ファイルが作成できません。

(9)　I／O，最適化（2／3）

A909	メッセージ	R/O : read/only disk
	原　　因	ドライブがread only属性のためテンポラリ・ファイルが作成できません。
A910	メッセージ	R/O file : read/only , file opened read/only mode
	原　　因	次の理由によりテンポラリ・ファイルのwriteエラーが発生しました。 1.テンポラリ・ファイルと同一名のファイルがドライブ上にすでに存在しread only属性が与えられています。 2.内部矛盾により出力テンポラリ・ファイルをread only属性でopenしています。
A911	メッセージ	Reading unwritten data , no available directory space
	原　　因	次の理由により入出力エラーが発生しました。 1.ＥＯＦを越えて入力を行おうとしました。 2.ドライブのディレクトリ・エリアの不足によりテンポラリ・ファイルが作成できません。
A912	メッセージ	Write error on temp
	原　　因	出力用のテンポラリ・ファイルへのwriteエラーが発生しました。
A913	メッセージ	Requires MS-DOS V2.11 or greater
	原　　因	ＯＳがＭＳ－ＤＯＳ（Ｖ２．１１以上）ではありません。
A914	メッセージ	In sufficient memory in host machine
	原　　因	メモリ不足のためコンパイラを起動できません。

(9)　Ｉ／Ｏ，最適化（ＣＣ７８Ｋ３のみ）（３／３）

Ｗ９１５	メッセージ	Asm statement found. optimize skip this function
	原　　因	ＡＳＭ文が検出されました。 この関数の最適化を中止します。
Ｗ９１６	メッセージ	Too many basic blocks for optimizing
	原　　因	基本ブロックが多すぎて最適化できません。
Ｗ９１７	メッセージ	Too many nesting for optimizing
	原　　因	関数のネストが深すぎて最適化できません。
Ｗ９１８	メッセージ	Too many function nesting for optimizing
	原　　因	関数引数が多すぎて最適化できません。
Ｗ９１９	メッセージ	Too many function nesting for optimizing
	原　　因	関数のネストが深すぎて最適化できません。
Ｗ９２０	メッセージ	Too many symbol for optimizing
	原　　因	シンボルが多すぎて最適化できません。
Ｗ９２１	メッセージ	Too many value_ no for optimizing
	原　　因	値番号が多すぎて最適化できません。

保守／廃止

付録A　サンプル・プログラム

A.1　Cソース・モジュール・ファイル

```
#define TRUE    1
#define FALSE   0
#define SIZE    200

char    mark[SIZE+1];

main()
{
        int i, prime, k, count;

        count = 0;

        for ( i = 0 ; i <= SIZE ; i++)
                mark[i] = TRUE;
        for ( i = 0 ; i <= SIZE ; i++) {
                if (mark[i]) {
                        prime = i + i + 3;
                        printf("%6d",prime);
                        count++;
                                if((count%8) == 0) putchar('¥n');
                        for ( k = i + prime ; k <= SIZE ; k += prime)
                                mark[k] = FALSE;
                }
        }
        printf("¥n%d primes found.",count);
}

printf(s,i)
char *s;
int i;
{
        int j;
        char *ss;

        j = i;
        ss = s;
}

putchar(c)
char c;
{
        char d;
        d = c;
}
```

A.2 実行例

```
A>cc78K3 -c310 sample¥prime.c -a -p -x -e


78K/III Series C Compiler Vx.xx  [xx xxx xx]
   Copyright (C) NEC Corporation xxxx


SAMPLE¥PRIME.C(18) : W745 Expected function prototype
SAMPLE¥PRIME.C(20) : W745 Expected function prototype
SAMPLE¥PRIME.C(26) : W622 No return value
SAMPLE¥PRIME.C(37) : W622 No return value
SAMPLE¥PRIME.C(44) : W622 No return value
Compilation complete,     0 error(s) and     5 warning(s) found.
```

A.3 出力リスト

(1) アセンブラ・ソース・モジュール・ファイル

```
; 78K/III Series C Compiler V1.10 Assembler Source
;                                                    Date:xx xxx xxxx Time:xx:xx:xx

; Command   : -c310 ¥sample¥prime.c -a -p -x -e
; In-file   : ¥SAMPLE¥PRIME.C
; Asm-file  : PRIME.ASM
; Para-file :

$PROCESSOR(310)
$NODEBUG

        NAME    PRIME
        EXTRN   @@isrem
        PUBLIC  _mark
        PUBLIC  _main
        PUBLIC  _printf
        PUBLIC  _putchar

@@CODE  CSEG
; line    5
; line    8
_main:
        push    hl
        movw    ax,sp
        subw    ax,#08H
        movw    hl,ax
        movw    sp,ax
; line   11
        movw    ax,#00H ; 0
        mov     [hl+1],a        ; count
        xch     a,x
        mov     [hl],a  ; count
; line   13
        movw    ax,#00H ; 0
        mov     [hl+7],a        ; i
        xch     a,x
        mov     [hl+6],a        ; i
L0003:
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        cmpw    ax,#0C8H        ; 200
        bgt     $L0004
; line   14
        movw    up,#_mark
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    up,ax
        mov     a,#01H  ; 1
        mov     [up],a
L0005:
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    ax,#01H ; 1
        mov     [hl+7],a        ; i
        xch     a,x
        mov     [hl+6],a        ; i
```

```
        br      $L0003
L0004:
; line    15
        movw    ax,#00H ; 0
        mov     [hl+7],a        ; i
        xch     a,x
        mov     [hl+6],a        ; i
L0006:
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        cmpw    ax,#0C8H        ; 200
        ble     $$+6
        br      !L0007
; line    16
        movw    up,#_mark
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    up,ax
        mov     a,[up]
        xch     a,x
        mov     a,#0FFH ; 255
        bt      x.7,$L0011
        inc     a
L0011:
        cmpw    ax,#00H ; 0
        bne     $$+5
        br      !L0009
; line    17
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        movw    bc,ax
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    ax,bc
        addw    ax,#03H ; 3
        mov     [hl+5],a        ; prime
        xch     a,x
        mov     [hl+4],a        ; prime
; line    18
        mov     a,[hl+4]        ; prime
        xch     a,x
        mov     a,[hl+5]        ; prime
        push    ax
        movw    ax,#L0012
        push    ax
        call    !_printf
        movw    ax,sp
        addw    ax,#04H ; 4
        movw    sp,ax
; line    19
        mov     a,[hl]  ; count
        xch     a,x
        mov     a,[hl+1]        ; count
        addw    ax,#01H ; 1
        mov     [hl+1],a        ; count
        xch     a,x
        mov     [hl],a  ; count
; line    20
        movw    rp2,#08H        ; 8
        mov     a,[hl]  ; count
        xch     a,x
        mov     a,[hl+1]        ; count
        movw    bc,ax
```

```
        call    !@@isrem
        movw    ax,bc
        cmpw    ax,#00H ; 0
        bne     $L0013
        movw    ax,#0AH ; 10
        push    ax
        call    !_putchar
        pop     ax
L0013:
L0014:
; line   21
        mov     a,[hl+4]        ; prime
        xch     a,x
        mov     a,[hl+5]        ; prime
        movw    bc,ax
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    ax,bc
        mov     [hl+3],a        ; k
        xch     a,x
        mov     [hl+2],a        ; k
L0015:
        mov     a,[hl+2]        ; k
        xch     a,x
        mov     a,[hl+3]        ; k
        cmpw    ax,#0C8H        ; 200
        bgt     $L0016
; line   22
        movw    up,#_mark
        mov     a,[hl+2]        ; k
        xch     a,x
        mov     a,[hl+3]        ; k
        addw    up,ax
        mov     a,#00H  ; 0
        mov     [up],a
L0017:
        mov     a,[hl+4]        ; prime
        xch     a,x
        mov     a,[hl+5]        ; prime
        movw    bc,ax
        mov     a,[hl+2]        ; k
        xch     a,x
        mov     a,[hl+3]        ; k
        addw    ax,bc
        mov     [hl+3],a        ; k
        xch     a,x
        mov     [hl+2],a        ; k
        br      $L0015
L0016:
L0009:
L0010:
L0008:
        mov     a,[hl+6]        ; i
        xch     a,x
        mov     a,[hl+7]        ; i
        addw    ax,#01H ; 1
        mov     [hl+7],a        ; i
        xch     a,x
        mov     [hl+6],a        ; i
        br      !L0006
L0007:
; line   25
        mov     a,[hl]  ; count
        xch     a,x
        mov     a,[hl+1]        ; count
        push    ax
```

```
        movw    ax,#L0018
        push    ax
        call    !_printf
        movw    ax,sp
        addw    ax,#04H ; 4
        movw    sp,ax
        movw    ax,hl
        addw    ax,#08H
        movw    sp,ax
        pop     hl
        ret
; line    31
_printf:
        push    hl
        movw    ax,sp
        subw    ax,#04H
        movw    hl,ax
        movw    sp,ax
; line    35
        mov     a,[hl+10]       ; i
        xch     a,x
        mov     a,[hl+11]       ; i
        mov     [hl+3],a        ; j
        xch     a,x
        mov     [hl+2],a        ; j
; line    36
        mov     a,[hl+8]        ; s
        xch     a,x
        mov     a,[hl+9]        ; s
        mov     [hl+1],a        ; ss
        xch     a,x
        mov     [hl],a  ; ss
        movw    ax,hl
        addw    ax,#04H
        movw    sp,ax
        pop     hl
        ret
; line    41
_putchar:
        push    hl
        movw    ax,sp
        subw    ax,#02H
        movw    hl,ax
        movw    sp,ax
; line    43
        mov     a,[hl+6]        ; c
        mov     [hl+1],a        ; d
        movw    ax,hl
        addw    ax,#02H
        movw    sp,ax
        pop     hl
        ret

@@CNST  CSEG
L0012:  DB      '%6d'
        DB      00H
L0018:  DB      0AH
        DB      '%d primes found.'
        DB      00H

@@R_DATA        CSEG
        DB      (201)

@@DATA  DSEG
_mark:  DS      (201)
        END
```

(2) プリプロセス・リスト・ファイル

```
/*
78K/III Series C Compiler V1.10 Preprocess List      Date:xx xxx xxxx Page:   1

Command   : -c310 ¥sample¥prime.c -a -p -x -e
In-file   : ¥SAMPLE¥PRIME.C
PPL-file  : PRIME.PPL
Para-file :
*/

    1 : #define TRUE    1
    2 : #define FALSE   0
    3 : #define SIZE    200
    4 :
    5 : char    mark[SIZE+1];
    6 :
    7 : main()
    8 : {
    9 :         int i, prime, k, count;
   10 :
   11 :         count = 0;
   12 :
   13 :         for ( i = 0 ; i <= SIZE ; i++)
   14 :                 mark[i] = TRUE;
   15 :         for ( i = 0 ; i <= SIZE ; i++) {
   16 :                 if (mark[i]) {
   17 :                         prime = i + i + 3;
   18 :                         printf("%6d",prime);
   19 :                         count++;
   20 :                                 if((count%8) == 0) putchar('¥n');
   21 :                         for ( k = i + prime ; k <= SIZE ; k += prime)
   22 :                                 mark[k] = FALSE;
   23 :                 }
   24 :         }
   25 :         printf("¥n%d primes found.",count);
   26 : }
   27 :
   28 : printf(s,i)
   29 : char *s;
   30 : int i;
   31 : {
   32 :         int j;
   33 :         char *ss;
   34 :
   35 :         j = i;
   36 :         ss = s;
   37 : }
   38 :
   39 : putchar(c)
   40 : char c;
   41 : {
   42 :         char d;
   43 :         d = c;
   44 : }
```

(3) クロス・レファレンス・リスト・ファイル

```
78K/III Series C Compiler V1.10 Cross reference List Date:xx xxx xxxx Page:   1

Command   : -c310 ¥sample¥prime.c -a -p -x -e
In-file   : ¥SAMPLE¥PRIME.C
Xref-file : PRIME.XRF
Para-file :

ATTRIB MODIFY TYPE     SYMBOL          DEFINE  REFERENCE

EXTERN        array    mark               5      14      16      22
EXTERN        func     main               7
AUTO1         int      i                  9      13      13      13      14
 15      15      15      16      17      17
                                                 21
AUTO1         int      prime              9      17      18      21      21
AUTO1         int      k                  9      21      21      21      22
AUTO1         int      count              9      11      19      20      25
EXTERN        func     printf            28      18      25
EXTERN        func     putchar           39      20
PARAM         pointer  s                 29      36
PARAM         int      i                 30      35
AUTO1         int      j                 32      35
AUTO1         pointer  ss                33      36
PARAM         char     c                 40      43
AUTO1         char     d                 42      43
              #define  TRUE               1      14
              #define  FALSE              2      22
              #define  SIZE               3       5      13      15      21
```

(4) エラー・リスト・ファイル

```
SAMPLE¥PRIME.C(   18) : W745 Expected function prototype
SAMPLE¥PRIME.C(   20) : W745 Expected function prototype
SAMPLE¥PRIME.C(   26) : W622 No return value
SAMPLE¥PRIME.C(   37) : W622 No return value
SAMPLE¥PRIME.C(   44) : W622 No return value

Compilation complete,     0 error(s) and     5 warning(s) found.
```

保守／廃止

付録B　使用上の注意点一覧

■コンパイル時

項番	注意事項
1	MS－DOS（PC－DOS）のシステムに関する注意事項： ・本Cコンパイラを起動する際にはMS－DOS（PC－DOS）の環境設定ファイルCONFIG．SYS内で「FILES＝25」以上に設定する必要があります。
2	オプション指定に関する注意事項： ・複数指定が可能でないオプションを複数指定した場合には，後に指定した方を優先します。 ・－Cに続けて指定する種別は省略できません。省略した場合は，アボート・エラーとなります。－Cオプションで指定しない場合はCソース・モジュール・ファイル中に#pragma pc（種別）を記述してください。また，実行の際のオプションとCソース中の種別が異なった場合にはオプションの指定を優先します。このときワーニング・メッセージが出力されます。 ・ヘルプ指定があった場合には，他のすべてのオプションは無効になります。
3	ファイルの出力先に関しての注意事項： ・オブジェクト・モジュール・ファイルの出力先は，ディスク型ファイルのみです。
4	エラー・メッセージに関しての注意事項： ・ファイルに文法的誤りがあった場合には，エラー・メッセージにファイル名が付加されます。またデバイス型ファイルを指定禁止箇所で指定した場合は，指定文字列がそのまま出力されます。それ以外のときは，ドライブ名とパス名と拡張子が必ず付加されます。

■アセンブル時

項番	注意事項
1	Ｃソース・プログラム中にアセンブリ言語による記述がある場合： ・Ｃソース・プログラム中にアセンブリ言語による記述がある場合，ロード・モジュール・ファイル作成手順は，コンパイル，アセンブル，リンクの順に行います。この場合，Ｃコンパイラで出力されたオブジェクト・ファイルをアセンブルするときには，オブジェクト・ファイル中のシンボル名の大文字，小文字を区別させるため－ＮＣＡオプションを指定しなければなりません。

■リンク時（１／３）

項番	注意事項
1	ＲＯＭ化を行う場合： ・ＣＣ７８Ｋｎでは，コンパイル時にディフォールトでＲＯＭ化指定がされています。したがって，ディフォールトでコンパイルしたときはリンク時にＲＯＭ化対応用のスタート・アップルーチンとリンクしなければならない。 ｽﾀｰﾄ･ｱｯﾌﾟ･ﾙｰﾁﾝ　：　csxxxr.rel romxxx.rel 【使用例】　μPD78320の場合 A>lk78k3　cs320r.rel　test.rel　rom320.rel ＲＯＭ化する場合 必要となる test.rel　：　ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑの ｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ
2	スタック解決用シンボル生成指定（－ｓ）オプション： ・ＣＣ７８Ｋｎではユーザがスタック領域を確保することができません。したがって，これを確保するためにリンク時に－ｓオプションを指定してください。

■リンク時（２／３）

項番	注意事項
3	ディレクティブ・ファイル指定（－ｄ）オプション： ・ディフォールトで定義しているメモリ領域でリンクすると以下のようなエラーが出ることがあります。 *** ERROR F206 Segment 'xxx' can't allocate to memory－ignored. 【原　因】 リンカがディフォールトで定義しているメモリ領域(内蔵ＲＡＭ空間)では，領域不足のために，指摘されたセグメントを配置することができないためです。 【対処方法】 対処方法は，大きく分けて次の３つの手順によります。 1.配置できないセグメントのサイズを調べる。 2.手順１で調べたセグメントのサイズをもとに，ディレクティブ・ファイル中のセグメントが配置されている領域のサイズを拡大する。 3.ディレクティブ・ファイル指定オプション（－ｄ）を指定してリンクする。 ただし，手順１ではエラーで指摘されたセグメントの種類により，以下のようにセグメントのサイズを調べる方法が異なります。 (1)コンパイル時に自動生成されるセグメントのとき セグメント名 @@CALT,@@CNST,@@CODE,@@R_DATA,@@DATA,@@R_INIT,@@INIT, @@R_DATS,@@DATS,@@R_INIS,@@INIS,@@BITS 手順1 ディレクティブ・ファイルにおいて，指摘されたセグメントが配置されているメモリ領域を十分大きくとる。 手順2 リンクし作成されたマップ・ファイルによりセグメントのサイズを調べる。 (2)ユーザが作成したセグメントのとき アセンブル・リスト・ファイルにより指摘されたセグメントのサイズを調べる。 【備　考】 ディレクティブ・ファイルについては，フロッピィ・ディスク(1)のディレクトリＬＩＢに各デバイスの標準ディレクトリ・ファイルがサンプル・ファイルとしてありますのでご使用ください。

■リンク時（３／３）

項番	注意事項
4	ライブラリ・ファイル指定（－Ｂ）オプション： ・ＣＣ７８Ｋｎにより，コンパイルしたオブジェクト・ファイルをリンクする際は，使用例１のように，Ｃコンパイラ提供のライブラリ・ファイルを指定してください。ただし，７８Ｋ／Ⅲシリーズのデバイス種別３１０，３１２，３１０Ａ，３１２Ａについては，使用例２のように，ライブラリ・ファイルを指定してください。 【使用例１】μPD78320の場合 A>lk78k3 cs320r.rel test.rel rom320.rel -s -bclk3com.lib test.rel:ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑのｵﾌﾞｼﾞｪｸﾄ･ﾌｧｲﾙ 【使用例２】μPD78312Aの場合 A>lk78k3 cs312ar.rel test.rel rom312a.rel -s -bcl312a.lib test.rel:ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑのｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ
	・コンパイル時に，実行時エラー・チェック指定オプション（－Ｌ）により，ｓｆｒのアクセス・チェック指定を行った場合は，リンク時に使用例３のように ～～～ 部のライブラリ・ファイル　ＣＬｘｘｘ．ＬＩＢも指定してください。ただし，７８Ｋ／Ⅲシリーズのデバイス種別３１０，３１２，３１０Ａ，３１２Ａについては，この必要はありません（使用例２と同様です）。 【使用例３】μPD78320の場合 A>lk78k3 cs320r.rel test.rel rom320.rel -s -bclk3com.lib -bcl320.lib test.rel:ﾕｰｻﾞ･ﾌﾟﾛｸﾞﾗﾑのｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ

付録C　コンパイラ・オプション一覧

コンパイラ・オプション一覧（１／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
1	デバイス種別指定	－ＣＸ （Ｘ：種別）	ターゲット・デバイスの種別を指定します。	独立	省略不可	49
2	オブジェクト・モジュール・ファイル作成指定	－Ｏ［ファイル名］	オブジェクト・モジュール・ファイル名を指定のファイル名に変更します。	－Ｏと－ＮＯが同時に指定された場合には後者優先となります。	－Ｏ［入力ファイル名．ＲＥＬ］	54
		－ＮＯ	オブジェクト・モジュール・ファイルを出力しません。			
3	シンボル名長指定	－Ｓ	シンボル名の長さを最大３０文字に拡張します。	独立	－ＮＳ	56
		－ＮＳ	シンボル名の長さを８文字にします。			
4	シンボル名ケース指定	－ＣＡ	シンボル名のケース（大，小文字）を区別しません。	独立	－ＮＣＡ	58
		－ＮＣＡ	シンボル名のケースを区別します。			

コンパイラ・オプション一覧（２／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
5	ＲＯＭ化指定	－Ｒ	ＲＯＭ化可能なオブジェクト・モジュールを作成します。	ｵﾌﾞｼﾞｪｸﾄ・ﾓｼﾞｭｰﾙ・ﾌｧｲﾙ，ｱｾﾝﾌﾞﾗ・ｿｰｽのいずれも出力されない場合は無効です。	－Ｒ	60
		－ＮＲ	ＲＯＭ化不可のオブジェクト・モジュールを作成します。			
6	最適化指定	－Ｑ［Ｘ］（Ｘ：処理の種別 複数指定可能）	効率の良いオブジェクトを作成します。	ｵﾌﾞｼﾞｪｸﾄ・ﾓｼﾞｭｰﾙ・ﾌｧｲﾙ，ｱｾﾝﾌﾞﾗ・ｿｰｽのいずれも出力されない場合は無効です。	－ＮＱ	63
		－ＮＱ	最適化を行いません。			
7	ディバグ情報出力指定	－Ｇ	オブジェクト・モジュール・ファイル中にディバグ情報を付加します。	ｵﾌﾞｼﾞｪｸﾄ・ﾓｼﾞｭｰﾙ・ﾌｧｲﾙ，ｱｾﾝﾌﾞﾗ・ｿｰｽのいずれも出力されない場合は無効です。	－ＮＧ	66
		－ＮＧ	オブジェクト・モジュール・ファイル中にディバグ情報を付加しません。			
8	実行時エラー・チェック指定	－Ｌ［Ｘ］（Ｘ：処理の種別 複数指定可能）	作成されるオブジェクト・モジュールに実行時エラーチェック・ライブラリを付加します。	ｵﾌﾞｼﾞｪｸﾄ・ﾓｼﾞｭｰﾙ・ﾌｧｲﾙ，ｱｾﾝﾌﾞﾗ・ｿｰｽのいずれも出力されない場合は無効です。	－ＮＬ	67

コンパイラ・オプション一覧（３／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
8	実行時エラー・チェック指定（続き）	－ＮＬ	作成されるオブジェクト・モジュールに実行時エラーチェック・ライブラリを付加しません。	ｵﾌﾞｼﾞｪｸﾄ･ﾓｼﾞｭｰﾙ･ﾌｧｲﾙ，ｱｾﾝﾌﾞﾗ･ｿｰｽのいずれも出力されない場合は無効です。		67
9	プリプロセス・リスト・ファイル作成指定	－Ｐ［ファイル名］	プロプロセス・リストをファイルに出力します。	独立	－ＮＰ	69
		－ＮＰ	プリプロセス・リストを出力しません。			
		－Ｋ［Ｘ］（Ｘ：処理の種別 複数指定可能）	プリプロセス・リストに対する処理を指定します。	－Ｐが指定されない場合，－ＮＰが指定された場合は無効です。	－ＫＦＬＮ	71
		－ＮＫ	プリプロセス・リストに対する処理を行いません。			
１０	プリプロセス指定	－Ｄ名前＝定義名［，名前［＝定義名］］…（複数指定可能）	#define文同様のマクロ定義を行います。	同じ名前（マクロ名）に対する－Ｄと－Ｕが同時に指定された場合，後者優先とします。	Ｃソース・モジュール・ファイル中のマクロ定義のみを有効とします。	74
		－ＮＤ	－Ｄオプションを無効にします。			

コンパイラ・オプション一覧（４／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
１０	プリプロセス指定	－Ｕ名前［，名前］… （複数指定可能）	#undef文と同様にマクロ定義を無効にします。	同じ名前（マクロ名）に対する－Ｄと－Ｕが同時に指定された場合，後者優先とします。	－Ｄで指定したマクロ定義を有効とします。	75
		－ＮＵ	－Ｕオプションを無効にします。			
		－Ｉディレクトリ［，ディレクトリ］… （複数指定可能）	#include文で指定されたインクルード・ファイルを指定されたディレクトリから入力します。	独立	ファイルを捜す順序は次に示します。 ・環境変数ＩＮＣ７８Ｋ３のディレクトリ ・ソース・ファイルのあるディレクトリ	76
１１	アセンブラ・ソース・モジュール・ファイル作成指定	－Ａ［ファイル名］	アセンブラ・ソース・モジュールを，ファイルに出力します。	独立	－ＮＡ	77
		－ＮＡ	アセンブラ・ソース・モジュールを，ファイルに出力しません。			

コンパイラ・オプション一覧（５／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
１１	アセンブラ・ソース・モジュール・ファイル作成指定	－ＳＡ［ファイル名］	アセンブラ・ソース・モジュールを，ファイルに出力します。コメントとしてＣ・ソースが付加されます。	独立	－ＮＳＡ	79
		－ＮＳＡ	－ＳＡオプションを無効にします。			
１２	エラー・リスト・ファイル作成指定	－Ｅ［ファイル名］	エラー・リストをファイルに出力します。	－Ｗ０が指定された場合はワーニング・メッセージは出力されません。	－ＮＥ	81
		－ＮＥ	エラー・リストをファイルに出力しません。			
		－ＳＥ［ファイル名］	エラー・リストをファイルに出力します。Ｃ・ソースを付加します。	－Ｗ０が指定された場合はワーニング・メッセージは出力されません。	－ＮＳＥ	83
		－ＮＳＥ	－ＳＡオプションを無効にします。			
１３	クロス・レファレンス・リスト作成指定	－Ｘ［ファイル名］	クロス・レファレンス・リストをファイルに出力します。	独立	－ＮＸ	85
		－ＮＸ	クロス・レファレンス・リストをファイル出力しません。			

コンパイラ・オプション一覧（６／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
１４	リスト形式指定	－ＬＷ［文字数］	各種リスト・ファイルの１行の文字数（ＰＡＧＥＷＩＤＴＨ）を指定します。	リスト・ファイル指定が何も指定されない場合，本オプションは無効です。	－ＬＷ１２３（コンソール出力の場合は，８０文字とします。）	86
		－ＬＬ［行数］	各種リスト・ファイルの１ページの行数（ＰＡＧＥＬＥＮＧＴＨ）を指定します。	リスト・ファイル指定が何も指定されない場合，本オプションは無効です。	－ＬＬ６６（コンソール出力の場合は，改ページなしとします。	89
		－ＬＴ［文字数］	タブの展開文字数を指定します。	リスト・ファイル指定が何も指定されない場合，本オプションは無効です。	－ＬＴ８	91
		－ＬＦ	リスト・ファイルの最後に改ページコードを付加します。	リスト・ファイル指定が何も指定されない場合，本オプションは無効です。	なし	94

コンパイラ・オプション一覧（７／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
１５	ワーニング出力指定	－Ｗ［レベル］	ワーニング・メッセージをコンソールに出力します。	－Ｅ，－ＳＥが指定された場合には，エラー・リスト・ファイルにもワーニング・メッセージが出力されます。	－Ｗ１	95
１６	実行状態表示指定	－Ｖ	コンパイルの実行状態をコンソールに出力します。	独立	－ＮＶ	97
		－ＮＶ	コンパイルの実行状態をコンソールに出力しません。			
１７	パラメータ・ファイル指定	－Ｆファイル名	オプションあるいは入力ファイル名を指定のファイルから入力します。	独立	コマンド行上からのみオプションと入力ファイル名を入力します。	98

コンパイラ・オプション一覧（８／８）

項番	分　類	記述形式	機　能	他のオプションとの関係	省略時解釈	参照ページ
１８	テンポラリ・ファイル作成ディレクトリ指定	－Tディレクトリ	テンポラリ・ファイルを指定されたパスに作成します。	独立	環境変数ＴＭＰにより指定されたドライブ，ディレクトリに作成されます。ＴＭＰが指定されていない場合はカレント・ドライブ，カレント・ディレクトリに作成されます。	100
１９	ヘルプ指定	－－	ヘルプ・メッセージをコンソールに表示します。	他のオプションをすべて無効とします。	なし	102

索　引

【英 文】

【ア】

【カ】

保守／廃止

保守／廃止

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］CC78Kシリーズ　Cコンパイラ　操作編　ユーザーズ・マニュアル
（EEU-656C（第4版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）（　　）
ご住所（　　）
お電話番号（　　）
お仕事の内容（　　）
お名前（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）
理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC販売員，特約店販売員，NEC半導体ソリューション技術本部員，
その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC半導体インフォメーションセンター
FAX：（044）548-7900

キリトリ

保守／廃止

保守／廃止

お問い合わせは，最寄りのNECへ

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部	〒108-01 東京都港区芝五丁目７番１号（NEC本社ビル）	東京	(03)3454-1111 (大代表)
中部支社 半導体第一販売部 半導体第二販売部	〒460 名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社 半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540 大阪市中央区城見一丁目４番24号（NEC関西ビル）	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社 札幌	(011)231-0161	太田支店 太田	(0276)46-4011	富山支店 富山	(0764)31-8461
東北支社 仙台	(022)267-8740	宇都宮支店 宇都宮	(028)621-2281	三重支店 津	(0592)25-7341
岩手支店 盛岡	(019)651-4344	小山支店 小山	(0285)24-5011	京都支社 京都	(075)344-7824
山形支店 山形	(0236)23-5511	長野支社 松本	(0263)35-1662	神戸支社 神戸	(078)333-3854
郡山支店 郡山	(0249)23-5511	甲府支店 甲府	(0552)24-4141	中国支社 広島	(082)242-5504
いわき支店 いわき	(0246)21-5511	埼玉支社 大宮	(048)641-1411	鳥取支店 鳥取	(0857)27-5311
長岡支店 長岡	(0258)36-2155	立川支社 立川	(0425)26-5981	岡山支店 岡山	(086)225-4455
土浦支店 土浦	(0298)23-6161	千葉支社 千葉	(043)238-8116	四国支社 高松	(0878)36-1200
水戸支店 水戸	(029)226-1717	静岡支社 静岡	(054)255-2211	新居浜支店 新居浜	(0897)32-5001
神奈川支社 横浜	(045)324-5524	北陸支社 金沢	(0762)23-1621	松山支店 松山	(089)945-4149
群馬支店 高崎	(0273)26-1255	福井支店 福井	(0776)22-1866	九州支社 福岡	(092)271-7700

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 マイクロコンピュータ技術部	〒210 川崎市幸区塚越三丁目484番地	川崎	(044)548-7950	半導体 インフォメーションセンター FAX(044)548-7900 (FAXにてお願い致します)
半導体販売技術本部 東日本販売技術部	〒108-01 東京都港区芝五丁目7番1号（NEC本社ビル）	東京	(03)3798-9619	
半導体販売技術本部 中部販売技術部	〒460 名古屋市中区錦一丁目17番１号（NEC中部ビル）	名古屋	(052)222-2125	
半導体販売技術本部 西日本販売技術部	〒540 大阪市中央区城見一丁目４番24号（NEC関西ビル）	大阪	(06) 945-3383	