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。

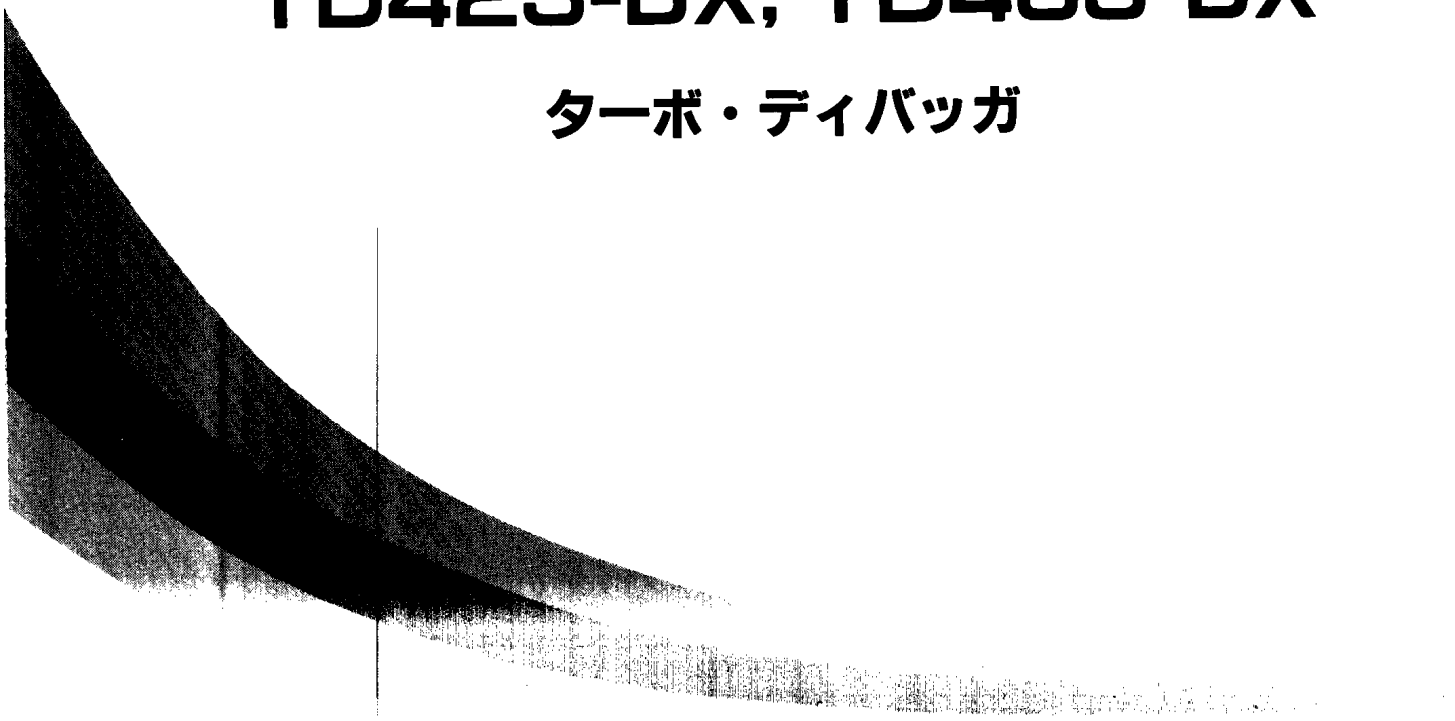
注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

ユーザーズ・マニュアル

RENESAS

TD423-BX, TD433-BX

ターボ・ディバッガ



対象エミュレータ

TD423-BX : IE-70423-BX

TD433-BX : IE-70433-BX

資料番号 EEU-980（第1版）
発行年月 August 1994 P

はじめに

対象者　このマニュアルは，μPD70423，70433（別名称V55SC™，V55PI™）の応用システムを設計，開発するユーザを対象としています。

目　的　TD423-BX，TD433-BXは，V55SC，V55PIの応用システムを設計，開発する際に，プログラムを検証，またはテストするためのサポート・ソフトウエアです。
このマニュアルは，TD423-BX，TD433-BXを用い，効率よくシステムをディバグしていただくことを目的としています。

構　成　このマニュアルは，大きく分けて次の内容で構成しています。

- 概　説
- ディバッガの使用環境
- 主な機能，操作方法
- TD423，TD433の起動
- 機能の詳細
- 周辺リソース管理
- エミュレータ機能
- コマンド機能の詳細
- メッセージ一覧
- 用語の説明

表記方法について　このマニュアルでは，次のような表記方法をとっています。

(1) 製品名の省略

このマニュアル中の説明では，次のように製品名を省略しています。

- TD423-BX，TD433-BX　→　TD423，TD433またはTDシリーズ[注]
- IE-70423，70433-BX　→　IE

注　TDシリーズ全般に該当する説明中では，各製品名をTDシリーズと置き換えています。

(2) レジスタ名，インストラクションの表示

この製品ではVシリーズ™表記のほか，インテル表記での表示ができます。このマニュアルでは，画面表示例の中ではインテル表記，本文中ではVシリーズ表記を用い

ています。

用語に関する注意　このマニュアルでは，次の用語を通常使用されるよりも広い意味で使用しています。

モジュール　Cやアセンブラでモジュールと呼ばれるものを指すだけでなく，Pascalでユニットと呼ばれるものも含みます。

関　数　特定の言語のための説明を除いて，"関数"をCの関数とPascalの関数および手続きを含む広い意味で使用しています。

引き数　パラメータと同じ意味で使用します。関数の引き数やコマンド・ライン引き数もこの意味で使用しています。

凡　例　注　：本文中につけた注の説明

注　意：気をつけて読んでいただきたい内容

備　考：本文の補足説明

数の表記：10進数…××××

16進数…××××h（実際の値の設定ではhは付けないでください）

２進数…××××b

このマニュアルでは，次のような記号を使用して説明しています。

< >	入力する項目を表します。
[ ]	[ ] 内に記述されている文字列の入力が省略できることを表します。
…	直前の入力項目について繰り返し入力可能なことを表します。
,	カンマの入力を表します。
△	スペース・キーを表します。
CTRL-	それぞれのキーを押しながらの入力を表します。
GRPH-	
SHIFT-	

画面について　このマニュアルでは，PC-9800シリーズを使用した場合の画面表示例を用いています。なお，表示内容はTD423とTD433でほぼ同一ですので，TD433の画面表示例をおもに用い，大きく異なる場合だけ両製品の画面表示例を掲載しています。

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

V55SC　ユーザーズ・マニュアル　ハードウエア編 (IEU-772)

V55PI　ユーザーズ・マニュアル　ハードウエア編 (IEU-776)

V55SC, V55PI　ユーザーズ・マニュアル命令編　(IEU-812)

IE-70423-BX　ユーザーズ・マニュアル　(EEU-820)

IE-70433-BX　ユーザーズ・マニュアル　(EEU-818)

目　　次

図の目次（1/4）

図の目次（2/4）

図の目次（3/4）

図の目次（4/4）

第1章　概　　説

この章では，TDシリーズのディバッガとしての基本仕様について説明します。

1.1　機能概要

TDシリーズは，16ビットVシリーズ用IEに対応した，組み込みシステム・サポート用の最先端ソース・レベル・ディバッガです。

ホスト・マシンとしてはPC-9800シリーズとPC/AT™を対象としています。オーバラップ可能な複数のウインドウやプルダウン・メニュー，ポップアップ・メニューの組み合わせ，およびマウスのサポートにより，操作性に優れ応答の速い対話型のディバグ環境を提供します。またディバグ・セッションのすべての面で，今操作している内容に応じた説明を表示するヘルプ・システムを備えています。

1.2　特　　徴

TDシリーズの特徴を次に示します。

- 米国Borland International Inc.，Microsoft Corp.製のコンパイラをサポート
- 大規模なプログラムをディバグするために，EMSを使用可能
- すべてのC，Pascal，アセンブラの式を評価可能
- 画面表示のレイアウトを自由に設定可能
- 必要なときにいつでもアセンブラ，CPUにアクセス可能
- 強力なブレークポイントとログ機能
- キー入力の記録が可能（マクロ）
- バック・トレース機能
- C++をフルサポート
- IEのハードウエア・ブレークポイント，トレース・バッファ，エミュレーション・メモリをサポート

1.3 用意していただくもの

項　目		機　種		条　件
		PC-9800シリーズ	PC/AT	
ハードウエア	パーソナル・コンピュータ	PC-9800シリーズ[注1]	PC/AT[注2]	RAM：640Kバイト実装
	ハード・ディスク	特に限定しません(必要に応じて用意してください)		約2Mバイト必要
	マウス	特に限定しません(必要に応じて用意してください)		—
	インサーキット・エミュレータ	IE-70423-BX（V55SC用）または IE-70433-BX（V55PI用）		—
	パソコン・インタフェース・ボード	IE-70000-98-IF-A IE-7000-98N-IF （ノート型パソコン用）	IE-70000-PC-IF-A	—
	増設エミュレーション・メモリ・ボード	IE-70000-BX-MM1(必要に応じて用意してください)		—
ソフトウエア	OS	MS-DOS™	—	Ver3.3以上
		—	PC DOS™	Ver3.0以上
	Cコンパイラ[注3]	TurboC		Ver2.×
		Turbo C++		Ver1.×
		Borland C++,		Ver2.×
		MS-C		Ver.5.1またはVer.6.0
	アセンブラ[注3]	Turbo Assembler		—
		MASM		Ver.5.1またはVer.6.0
	ロケータ	LC70116		—

注1. 拡張スロットのある機種またはノート型パソコン。快適な操作性を得るために，CPUに80386(動作周波数16MHz以上）を使用した機種を推奨します。

2. 拡張スロットのある機能だけ。快適な操作性を得るために，CPUに80386(動作周波数16MHz以上）を使用した機種を推奨します。

3. 開発元および販売元は，次のとおりです。

ソフトウエア名	開　発　元	販　売　元
Turbo C, Turbo C++	米国Borland International Inc.	株式会社マイクロソフトウエア　アソシエイツ
Borland C++		
Turbo Assembler		
MS-C, MASM	米国Microsoft Corp.	日本電気株式会社，マイクロソフト株式会社

1.4 ホスト・マシンによる操作方法の違い

このマニュアルでは機能面での違いがないかぎり，PC-9800シリーズ・ベースを代表品種として説明してあります。PC-9800シリーズとPC/ATでは，一部ファンクション・キーの名称が異なります。PC/ATベースのマニュアルとして使用する場合は次の表に従って読み替えてください。

PC-9800シリーズ	PC/AT
GRPH	ALT
HOMECLR	HOME
↵	ENTER
SHIFT-HOMECLR	END
ROLLDOWN	PgDn
ROLLUP	PgUp

なお，このマニュアルでは画面表示例についてもPC-9800シリーズのものを使用してあります。

〔メ　モ〕

第2章　ディバグを始める前に

TDシリーズのパッケージには一組のマスタ・ディスクとユーザーズ・マニュアル（このマニュアル）が入っています。マスタ・ディスクには，C++，C，アセンブラ，Pascal，PL/Mなどの言語で書かれたプログラムをディバグするために必要なすべてのプログラム，ファイル，ユーティリティが入っています。

ディバグを始める前にマスタ・ディスクから完全なコピーを作り，マスタ・ディスクはバックアップとして安全な場所に保管してください。通常は，コピーしたディスクを使用し，このディスクに何か問題が発生した場合には，マスタ・ディスクからコピーし直してください。

2.1　マスタ・ディスク

マスタ・ディスクのファイル・リストはディスク中のPACKING.LSTを見てください。

TDシリーズのマスタ・ディスクは2HD（3.5インチまたは５インチ）です。

2.2　READMEファイル

まず，最初にREADMEファイルを読んでください。このマニュアルに記述できなかった最新情報があります。簡単なコメントの付いたファイル・リストも入っています。

2.3　ユーティリティ

TDシリーズのパッケージには，いくつかのユーティリティ・プログラムが含まれています。各ユーティリティの機能は次のとおりです。

- TDCONVRT.EXE（CodeViewからTDシリーズへの変換ユーティリティ）
  このユーティリティは，Microsoft Corp.製のコンパイラおよびアセンブラで作成したプログラムを，TDシリーズでディバグできる形式に変換します。
- TDINSTBX
  TDシリーズのディフォールト設定に使用します。詳細は，**付録B　TDシリーズのカスタマイズ**を参照してください。

2.4　インストール

マスタ・ディスクにはTDシリーズをシステムに組み込むためのINSTALL.COMというプログラムが入っています。インストールを開始するには，まずマスタ・ディスクをディスク・ドライブのどれかに挿入し

ます。次にカレント・ドライブをそのドライブに変更します。それからINSTALLと入力して↵を押します。たとえばドライブAからインストールしている場合には，次のように入力します。

```
C>a:
A>install
```

INSTALL.COMが次の項目を聞いてきます。

- ●TD423，TD433をインストールするディレクトリ
- ●例題プログラムをインストールするディレクトリ

TD423,TD433および付帯するユーティリティのコピー先として選択したディレクトリをPATH環境変数に設定してください。

インストレーションには，次のプログラムとユーティリティと例題が含まれます。

TDシリーズ	TD423-BX	TD433-BX
ファイル名	TD423BX.EXE	TD433BX.EXE
	TD423BX.PIF	TD433BX.PIF
	TD423BX.ICO	TD433BX.ICO
	TD423BX.PDH	TD433BX.PDH
	README	
	TDINSTBX.EXE	
	INSTALL.BAT	

インストール時に選択したディレクトリに，実行ファイルとヘルプ・ファイルがコピーされます。

その他のファイルはオプションです。セットアップやディバグ中に問題が生じた場合に必要になることがあります。

2.5 Windows™セットアップ

TDシリーズは，Windowsの操作環境からも実行可能です。インストール・ユーティリティが設定に必要な.PIFとアイコン・ファイルを自動的にコピーすると，Windowsがこれを認識します。

次に示す順序に従ってWindowsを設定すると,アイコンをクリックするだけでTDシリーズが実行可能になります。

① Windowsを開始する前に，TD423，TD433のあるディレクトリをPATH環境変数に設定する。

② Windowsのプログラム・マネージャを実行し，メイン・メニューからFiles | Newを選択する。

③ Program Itemを選択し，OKする。

④ Descriptionフィールドに"TD423, TD433/IE-70423, 70433-BX"と入力し，Command Lineフィールドに TD423，TD433の実行ファイル名を入力する。

⑤ プログラム・マネージャのメイン・メニューからFile | Propertiesを選択する。

⑥ Change | Iconを選択しTD423，TD433のアイコン・ファイルのパスとファイル名を入力する。
アイコン・ファイルは実行ファイルと同じ名前とディレクトリを持ち，拡張子が.ICOとなります。

Windows環境内では，TDシリーズはフルスクリーン・アプリケーションとして実行されます。付属の.PIFファイルは，このオプションを選択するようにセットアップされています。

2.6 設　　定

TD423, TD433を起動する前に，ディフォールトの設定を変更します。付属のTDINSTBXファイルを実行することにより，オプションをカスタマイズできます。オプションの詳細は，**付録B　TDシリーズのカスタマイズ**を参照してください。TDINSTBXは次のように起動します。

TD423の場合：C>tdinstbx△td423bx.exe　（①）

TD433の場合：C>tdinstbx△td433bx.exe　（①）

注意　TDINSTBXはBorland International Inc.のTDINSTカスタマイゼーション・ユーティリティと同じものです。①の部分は，必ず指定してください。指定しないと，ターボ・ディバッガの実行ファイルをカスタマイズしてしまいます。

2.7 ファイル名規則

通常の.EXEファイルとディバグされるリモート・ターゲット実行ファイルを区別するため，.AXE(Absolute eXEcutable format) をディバグされるロード・モジュールの第1の拡張子として使用します。

TDシリーズはLC70116の生成したAXEフォーマットとインテル拡張HEXファイルをロードできます。

.AXE拡張子のファイルが見つからない場合は，ロード・モジュールと同じ名前で.EXE拡張子のあるファイルを探します。また，ロード・ファイルにターボ・ディバッガのディバグ情報がない場合は，ロード・モジュールと同じ名前で.TDS拡張子のあるファイルを探します。

注意　通常の.EXEファイルをロードしようとすると，プログラム・ロード・エラーが発生します。

TDシリーズが使用するディフォールトのファイル拡張子を次に示します。

.AXE	ディフォールトのロード・モジュール
.EXE	２番目のロード・モジュール
.TDS	リロケータブル・ディバグ情報ファイル
.PD	ディフォールトのTD423, TD433コンフィギュレーション・ファイル
.EMU	インサーキット・エミュレータ・コンフィギュレーション・ファイル
.HEX	HEXファイル

第３章　主な機能，操作方法

TDシリーズは，任意の場所でプログラムの実行を停止させたり，実行速度を遅くしてプログラムの状態を調べる多数の機能があります。また，変数に新しい値を設定して，それらの値がプログラムに与える影響を試してみることもできます。

3.1　基本的な機能

TDシリーズの基本的な機能を次に示します。

(1) トレース (Trace)

プログラムを１行ずつ実行することができます。関数呼び出しが行われた場合，その関数の１行目を実行します。

(2) バック・トレース (Back trace)

プログラムを逆向きに１行ずつ実行することができます。

(3) ステップ (Step)

関数の呼び出しを１行とみなして，プログラムを１行ずつ実行することができます。関数呼び出しが行われた場合，その関数の処理をすべて実行します。

関数にエラーがないことが確実なときは，それらをスキップすることによりディバグ作業をスピードアップできます。

(4) プログラム参照 (View)

プログラムの状態を表示する特別のウインドウをオープンすることができます。プログラムの変数，変数の値，ブレークポイント，スタックの内容，ログ，データ・ファイル，ソース・ファイル，CPUレジスタ，メモリ，レジスタ，周辺レジスタ，数値演算コプロセッサの情報，オブジェクトまたはクラスの階層構造，実行ヒストリなどを見ることができます。

(5) プログラムの調査 (Inspect)

プログラムの動作を詳しく調べることができます。また，配列や構造体のような複雑なデータ構造の内容を表示させることができます。

(6) プログラムの変更 (Change)

変数の現在の値を，指定した値に変更することができます。

(7) プログラムの監視 (Watch)

プログラムの変数を選んで，プログラムの実行によるそれらの変数の値の変化を監視することができます。

3.2 機能上の注意点

- TDシリーズには，ソース・コードを変更するためのエディタは組み込まれていません。ただし，Fileウインドウからローカル・コマンドEditを選択することにより，簡単にエディタに制御を渡すことができます。TDINSTBXカスタマイズ・プログラムで指定されたエディタが使用されますので，好みのエディタを使用してください。
- プログラムをコンパイルすることはできません。ソース・コードを変更したときそれを再コンパイルするには，元のコンパイラが必要です。

3.3 特徴的な機能

TDシリーズは，使いやすさと便利さを徹底的に追究してあり，次のような強力な機能を備えています。

- 便利で合理的なグローバル・メニュー。
- 状況に応じた適切なコマンドを表示する，状況感知型のローカル・メニュー。
  コマンドを覚えることもキーボードから入力することも不要になります。
- オプションの選択，セット，トグル，情報の入力が可能なダイアログ・ボックス。
- 同じような状況のもとで入力されたテキストを記憶するヒストリ・リスト。
  入力する必要があるときは，このリストから選択してそのまま入力することも，編集して変更してから入力することもできます。また新しく入力することもできます。
- 一連のコマンドやキーの入力をスピード・アップするフルマクロ機能。
- 便利で，完全に自由なウインドウ管理。
- マウスのサポート。
- ほとんどすべての地点からアクセスできるオンライン・ヘルプ。
- セッションの記録とバック・トレース。

3.4 プルダウン・メニュー・システム

TDシリーズは，画面最上段のメニュー・バーからアクセスできるメニュー・システムを備えています。メニュー・バーの項目を選ぶと，プルダウン・メニューがオープンします。プルダウン・メニューでは次

の３つのことが行えます。

- コマンドを実行する。
- ポップアップ・メニューを開く。

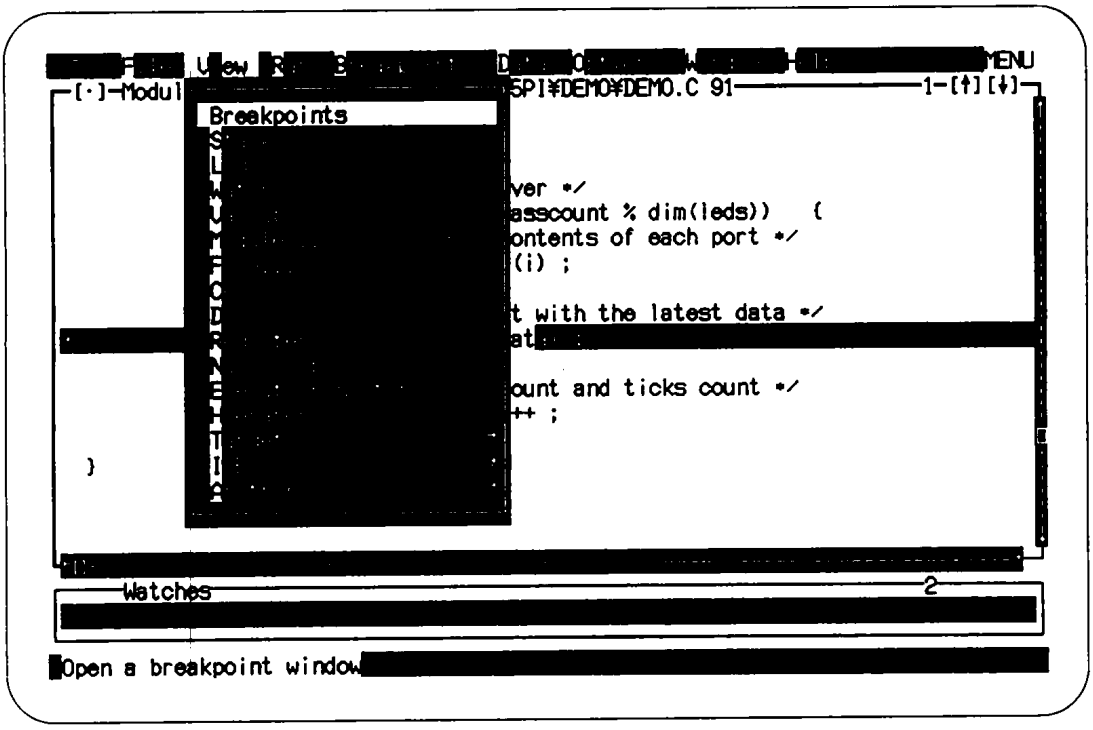
  メニュー項目に続いてメニュー・アイコン（→）が表示されているメニュー項目を選択すると，ポップアップ・メニューが開きます。
- ダイアログ・ボックスを開く。

  メニュー項目に続いてダイアログ・アイコン（...）が表示されているメニュー項目を選択すると，ダイアログ・ボックスが開きます。

メニュー・バーの各メニューの項目は，どのウインドウがアクティブであるか（どのウインドウにカーソルがあるか）にかかわらず，常に表示されています。また，各メニューの内容は変化しません。

なお，ローカル・メニュー（**3.8　ローカル・メニュー**参照）と対応させて，メニュー・バーとプルダウン・メニューを総称してグローバル・メニューと呼びます。

図３－１　グローバル・メニュー（Viewメニューをプルダウンしたところ）



3.5　グローバル・メニューの使い方

グローバル・メニューを開く（メニュー・バーのメニューをプルダウンする）には，次の４つの方法があります。

- F10を押し，→または←キーを使って希望するメニュー項目に移動し，↵を押す。
- F10を押し，希望するメニュー項目の名前の先頭の文字（スペース・バー，F，V，R，B，D，W，O，H）を押す。
- GRPHキーを押しながら希望するメニュー項目の先頭の文字（スペース・バー，F，V，R，B，D，W，O，H）を押す。たとえば，システム上どこにいても，GRPH-Fを押すとFileメニューに入ります。GRPH-スペース・バーを押すと，≡（システム）メニューが開きます。
- メニュー項目をマウスでクリックする。

グローバル・メニュー・システムの実行中は，次の方法でメニュー・システム上を移動します。

- →または←キーを押すと，隣りのプルダウン・メニューに移ることができます（たとえばFileメニューにいるときに→キーを押すとViewメニューに移ります）。
- ↑↓キーを押すと，そのプルダウン・メニューの中を上下に移動することができます。
- HOMECLRキーおよびSHIFT-HOMECLRを押すと，それぞれ，そのプルダウン・メニューの先頭および最後のメニュー項目に移動することができます。
- メニュー・アイコン（→）またはダイアログ・アイコン（...）が表示されているメニュー項目をハイライト表示させて↵を押すと，それぞれ下位レベルのメニュー（ポップアップ・メニュー）またはダイアログ・ボックスがオープンします。
- メニュー・アイコン（→）またはダイアログ・アイコン（...）が表示されているメニュー項目をマウスでクリックすると，それぞれ，下位レベルのメニューまたはダイアログ・ボックスが開きます。

各メニューまたはメニュー・システムから抜け出すには，次の方法があります。

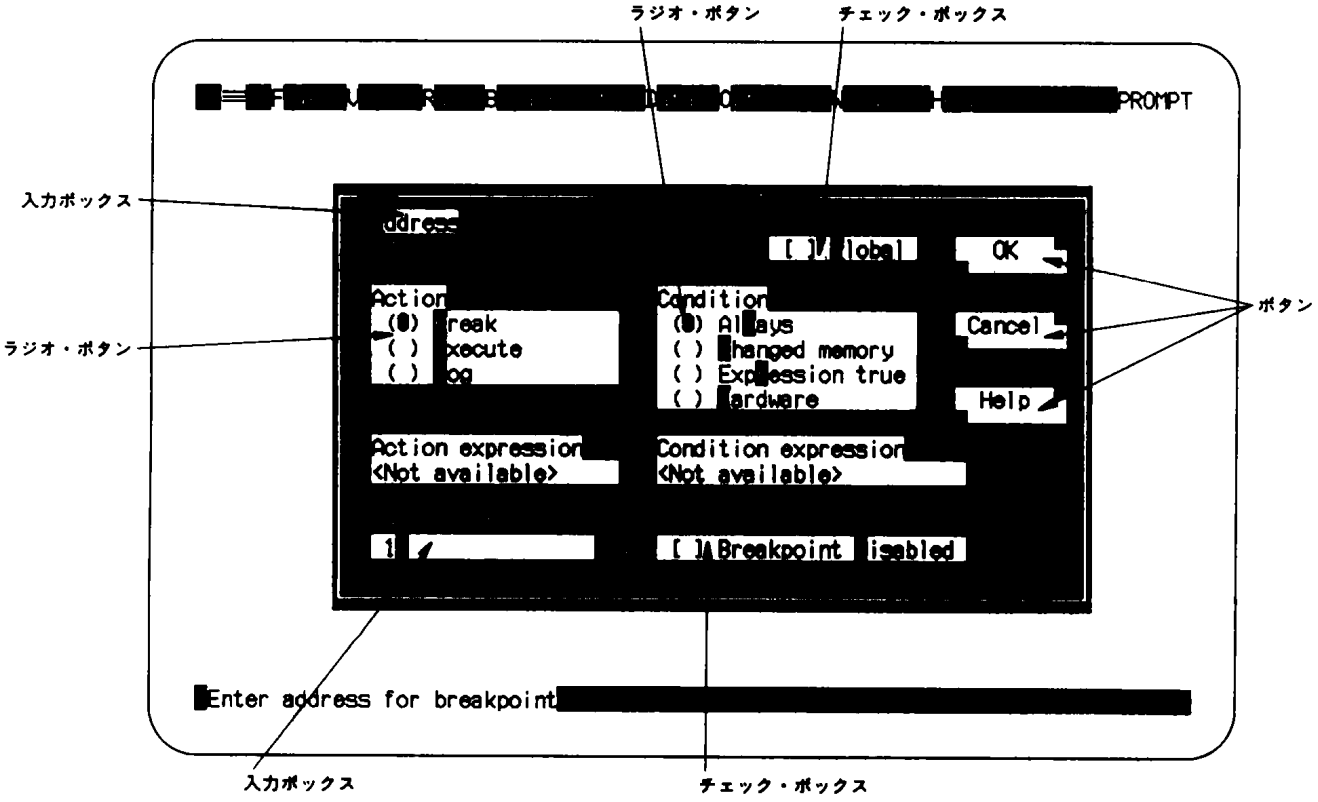
- 下位レベルのメニューを開いているときにESCキーを押すと，その前に開いたプルダウン・メニューに戻ります。
- プルダウン・メニューを開いているときにESCキーを押すと，メニュー・システムを抜け出してアクティブ・ウインドウに戻ります。
- ダイアログ・ボックス以外のどのメニュー・レベルにいるときでもF10を押すと，メニュー・システムを抜け出してアクティブ・ウインドウに戻ります。
- アクティブ・ウインドウをマウスでクリックすると，メニュー・システムを抜け出してアクティブ・ウインドウに戻ります。

いくつかのメニュー・コマンドには，そのコマンドをすばやく実行するためのホット・キーが割り当てられています。ホット・キーのあるコマンドに対しては，そのコマンドの右にホット・キーが表示されます（**第14章　コマンド・レファレンス参照**）。

3.6　ダイアログ・ボックス

コマンドにおけるオプションの多くは，ダイアログ・ボックスの中から選択することができます。ダイアログ・ボックスには，次の項目のいくつかが入っています。

図3-2　ダイアログ・ボックスの構成



(1) ボタン

ボタンは影付きのテキストです。ボタンを選択すると，そのボタンに表示されている動作がただちに実行されます。たとえばOKと表示されているボタンを選ぶと，設定内容が受け入れられ，ダイアログ・ボックスがクローズされます。Cancelを選ぶと設定内容がキャンセルされ，ダイアログ・ボックスがクローズされます。ダイアログ・ボックスにはHelpボタンもあり，このボタンを押すとオンライン・ヘルプが表示されます。

(2) チェック・ボックス

チェック・ボックスは，オン／オフを切り替えるトグルです。チェック・ボックスを選択すると，

そのオプションがオンまたはオフになります。チェック・ボックス・オプションがオンになっているときは，カッコの中にXが表示されます（[X]）。

(3) ラジオ・ボタン

ラジオ・ボタンは，複数オプションのトグルです。1組のラジオ・ボタンの中で選択できるのは1つだけです。選択されているラジオ・ボタンは，そのカッコの中に黒丸（●）が表示されます。

(4) 入力ボックス

入力ボックスは，文字列（ファイル名など）を入力するよう要求するものです。入力ボックスには，多くの場合，ヒストリ・リストが付いています（**3.9　ヒストリ・リスト**を参照）。

(5) リスト・ボックス

リスト・ボックスには，選択できる項目のリスト（たとえばオープンできるファイルのリスト）が入っています。

TABキーまたはSHIFT-TABを押すと，ダイアログ・ボックスの中を移動することができます。1組のラジオ・ボタンの中で設定を変更するにはカーソル・キーを使用します。ボタンは，TABキーで選択して↵を押します。

マウスを使うと，選択したい項目をクリックするだけで，簡単にダイアログ・ボックス内で選択できます。ダイアログ・ボックスを閉じるには，左上隅のクローズ・ボックスをクリックします。

また，ホット・キーすなわち各コマンド（項目）のハイライト表示された文字を押すことによっても，ダイアログ・ボックスの中の項目を選択できます。

3.7　状況感知機能

TDシリーズには，便利なプルダウン・メニュー・システムに加えて，メニューの数を減らしてユーザがまごつかないようにする状況感知機能があります。

ユーザがコマンドを選択するとき，TDシリーズは，ユーザが見ているものとカーソルの置かれている場所を正確に認識しています。コマンドに応答するときにこの情報が利用されます。

この状況感知機能により，メニュー・コマンドの名前や覚えにくいコマンド・ライン・スイッチを記憶したり入力したりするわずらわしさがなくなります。調べたいと思う変数や関数にカーソルを移動して(またはINSキーまたはマウスで選択して)，コマンドを呼び出すだけでプログラムを調査できます。

たとえば，ディバグしているCのプログラムに，次のような行があるとします。

```
MyArray [theIndex] Δ+=  2;
```

CTRL-I（Inspectorウインドウをオープンするホット・キー）を押して調査するだけで，簡単にデータ構

造に関する情報を得ることができます。カーソルがMyArrayの上にある場合は，この配列変数の内容がすべて表示されます。配列名だけでなく配列のインデクスまで含めて選択(ハイライト表示される)してCTRL-Iを押した場合は，ユーザが１つのメンバだけを調査したいと思っていることを読み取り，配列のその要素だけが表示されます。

また，すでに配列を調査しているときにCTRL-Iを押すと，１つのメンバを調査することができます。このようにして，次々にプログラムの細部を調べることができます。

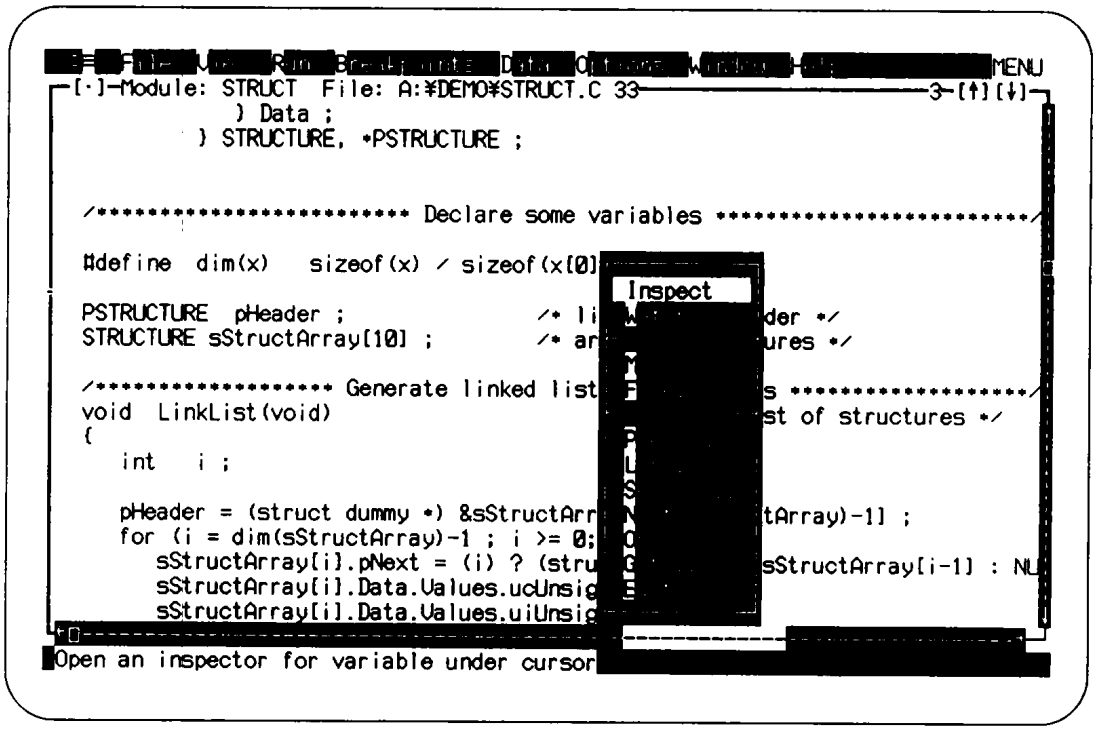
3.8　ローカル・メニュー

TDシリーズは，ウインドウごとにまたウインドウのペイン（ウインドウ内で分割された表示領域）ごとに別々のローカル・メニューを使用しています。

ローカル・メニューには，各ウインドウまたはペインで使用できるコマンドが表示されます。このローカル・メニューをグローバル・メニュー（プルダウン・メニュー）と混同しないようにしてください（実際には，この２種類のメニューが同時に表示されることはありません)。

ローカル・メニューは，その時点で選択可能なコマンドが一目で分かるように表示されるため，大変便利です。ローカル・メニューにより，不適切なコマンドの選択が防止され，またメニューが小さく見やすくなります。

図３-３　ローカル・メニュー



次にローカル・メニューの特徴を示します。

- ローカル・メニューは，GRPH-F10またはCTRL-F10を押すか，マウスの右ボタンを押すことにより，呼び出すことができます。
- メニューの表示位置や内容が，どのウインドウまたはどのペインにいるか，およびカーソルがどこにあるかにより変化します。
- ローカル・メニューの内容は，メニューにより変化します（ただし，ほとんどすべてのローカル・メニューに現われる核となるコマンドがあります）。同じ名前のコマンドでも，状況によって結果が異なる場合があります。
- ローカル・メニューのコマンドには，ホット・キー（CTRLキーとコマンドの先頭文字との組み合わせ）によるショート・カットがあります。

ローカル・メニューでは，その構成上，ホット・キー（たとえばCTRL-S）の機能が，状況によってまったく異なる場合があります。ただ，核となるコマンドには，メニューによらず機能に一貫性がありますので，たとえばGotoコマンドやSearchコマンドは，異なるペインから呼び出されたときにも同じことを実行します。

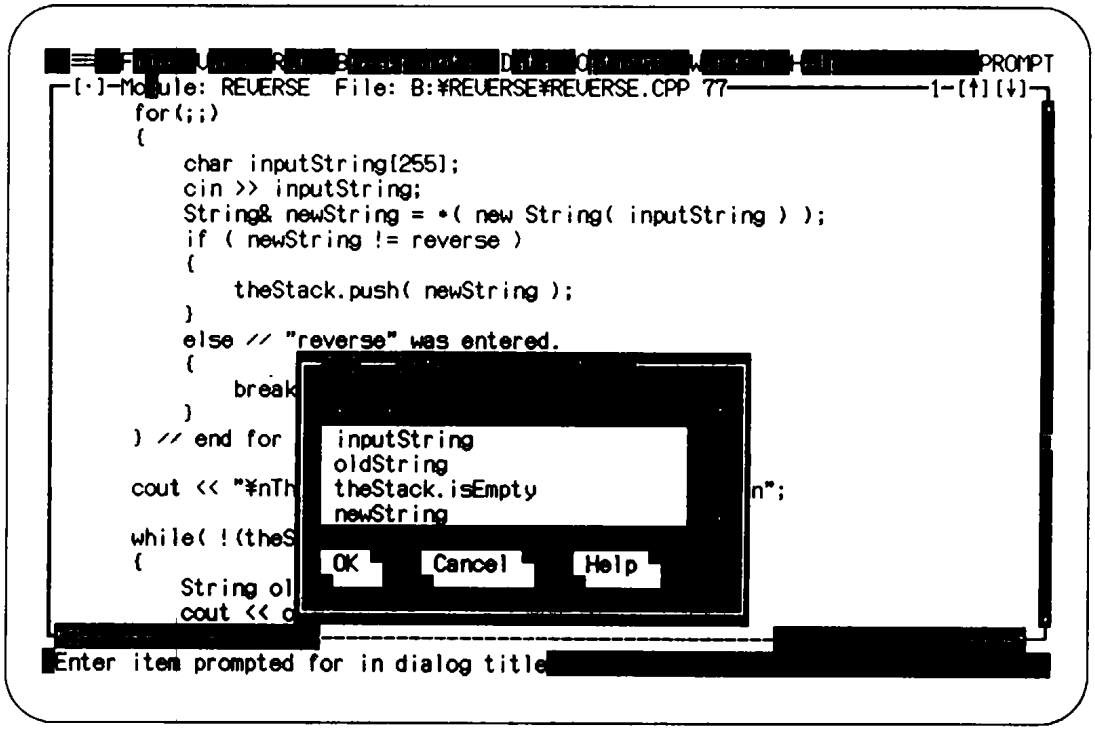
3.9　ヒストリ・リスト

ヒストリ・リスト機能とは，プロンプト・ボックスに入力されたものを記憶し，そのボックスが再び呼び出されるたびに，記憶したテキストを表示する機能です。これにより，ユーザのキー・ボードからの入力が必要最低限に抑えられます。

たとえば，inputStringという名前の関数とoldStringという名前の関数をSearchコマンドで検索した場合，ヒストリ・リストにはこの２つの名前が記憶されています。ヒストリ・リストには，Searchコマンドで検索した文字列が自動的に追加されていきます。この文字列は次回検索するときには，ヒストリ・リストから選択するだけで使用できます。

検索入力ボックスを，次に示します。

図3－4　Searchコマンドのヒストリ・リスト例



カーソル・キーを使って，以前に入力された文字列に移動して↵を押すと，その文字列の検索を開始することができます。マウスでは，スクロール・バーを使ってリストをスクロールさせることができます。ヒストリ・リストの項目を変更したときは，その項目がリストの一番上に移動します。

また，文字列を編集することもできます（カーソル・キーを使ってハイライト表示されている文字列内にカーソルを挿入し，通常どおりDELキーまたはBSキーを使って文字列を編集する）。たとえばoldStringを入力する場合，テキスト全部をタイプする必要はありません。inputStringを選択して，それをoldStringへと変更することができます。文字列をハイライト表示させてから，すぐにテキストを入力し始めると，元の文字列をキャンセルして新たに入力することになります。最初にカーソル・キーやBSキーなどの編集キーを押すと編集モードに入ります。

ヒストリ・リストには，設定が変更されていなければ，最新の5（または10）個の入力が保持されています（ヒストリ・リストのサイズは，インストール・プログラムTDINSTBXで変更できます）。

ほとんどの入力ボックスに対して，別々のヒストリ・リストが保存されます。したがって，検索のために入力した文字列が，たとえばあるラベルまたはあるプログラム行に移動するためのボックスを乱すことはありません。

3.10　名前の自動作成

入力ボックスの中で入力が必要なときは，プログラムの中のシンボル名の一部だけを入力してCTRL-Nを押します。

画面の右上隅に“READY...”が表示されているときは，シンボル・テーブルのソート中であることを示します。READYの後ろの３個のピリオドが消えるまでCTRL-Nは働きません。３個のピリオドが消えると，シンボル・テーブルが再びアクセス可能になります。

- シンボル名を判断できるだけの文字数が入力されると，そのシンボル名の残りの部分が付け加えられます。
- 入力した文字列が，登録されているどのシンボル名の先頭部分とも一致しない場合は，何も起こりません。
- 入力した文字列が，複数のシンボル名と一致する場合には，一致する名前のリストが表示されます。そのリストの中から希望するシンボル名を選択してください。

3.11　インクリメンタル・マッチ

インクリメンタル・マッチ機能は，ダイアログ・ボックス内のファイル名およびディレクトリ名のリストの中から希望する名前を見つけ出すのに便利な機能です。ファイル名の一覧などのリスト・ボックスの中で文字列の入力を始めると，１文字入力されるごとに，入力に一致するエントリにハイライト・バーが移動します。その後はOKボタンを選択するだけです。

3.12　マクロの作成

マクロは，簡単に言えば，ユーザが定義するホット・キーです。

どのようなコマンドやキー・ストロークでも１つのキーに定義することができ，いつでもそのキーを押すだけで再入力ができます。

マクロを作成するには，Options | Macrosを選択します。すると次の４つのコマンドが表示されます。

Create　Remove　Delete all　Stop recording

ここでCreateを選択すると，マクロを定義したいキーの入力が求められます。あまり使用しないキーまたは覚えやすいキー（たとえば，プログラムを再実行させるためのコマンドに対してSHIFT-F1）を押してから，そのキーに定義したい操作をすべて行います。

マクロの記録セッションを終了するには，次のどれかを行います。

- Options | Macros | Stop recordingを選択する。
- 新しく定義したマクロ・キー（この例ではSHIFT-F1）を押す。
- GRPH--を押す（GRPHキーを押しながらマイナス記号を押す）。

3.13　ウインドウ

TDシリーズは，グローバル／ローカル・メニュー，ダイアログ・ボックス（オプション選択や情報入力に使用），およびウインドウにすべての情報やデータを表示します。

ウインドウは多種多様であり，内部に表示される情報の種類で分けられます。また，ウインドウ内で分割された表示領域をペインと呼び，異なる種類のデータが各ペインに表示されます。

ウインドウのオープン／クローズは，メニュー・コマンド（またはコマンドのホット・キーによるショート・カット）により行います。ほとんどのウインドウはViewメニューからオープンしますが，別のクラスのウインドウであるInspectウインドウは，Data | InspectまたはローカルメニューのInspectを選択してオープンします。

3.13.1　Viewメニューからオープンできるウインドウ

Viewメニューからウインドウを１つまたは複数開くと，Windowメニューおよび≡(システム)メニューのコマンドにより，ウインドウの移動や大きさの変更，終了などの操作が行えます。Windowメニューおよび≡（システム）メニューのコマンドについては，**3.18　ウインドウの操作**を参照してください。

3.13.2　Moduleウインドウ

ディバグしているプログラムのコードを表示します。表示されているモジュールの中を移動したり，プログラムの変数にカーソルを置いて，適当なローカル・メニュー・コマンドを呼び出すことにより，データやコードを調べることができます。

Moduleウインドウは，もっともよく使われるウインドウです。このため，十分な時間をかけて，このウインドウのすべてのローカル・メニュー・コマンドを完全に習得してください。

Moduleウインドウは，GRPH-F2を押してオープンすることもできます。

詳細は，**8.2　Moduleウインドウ**を参照してください。

3.13.3　Watchesウインドウ

変数とそれらの現在の値を表示します。変数の上にカーソルを置いてCTRL-Wを押すことにより，その変数をこのウインドウに入れることができます。

詳細は，**6.3　Watchesウインドウ**を参照してください。

3.13.4　Breakpointsウインドウ

ブレークポイントを変更したり，削除したり，追加したりするときに，このウインドウを使います。

このウインドウは，現在設定されているブレークポイントを表示します。ブレークポイントとは，プログラムのステータスを調べるときにプログラムの実行を停止させるプログラム中の場所のことです。

このウインドウの左のペインは，すべてのブレークポイントの位置を表示します(グローバル・ブレークポイントに対しては，グローバルであることを示します)。右のペインは，左のペインで現在ハイライト表示されているブレークポイントの条件を表示します。

詳細は，第７章　ブレークポイントを参照してください。

3.13.5 Stackウインドウ

スタックの現在の状態を表示します。最初に呼び出された関数が最下位に（Cではこの関数はmain），その上に，続いて連鎖的に呼び出された関数が呼び出された順番に表示されます。

スタック内の関数のどれかをハイライト表示させると，次のことが可能です。

① CTRL-Iを押すと，その関数のソース・コードを表示させ調べることができます。

② CTRL-Lを押すと，プログラムのグローバル変数，その関数にローカルな変数，その関数の呼び出しの引き数を表示するVariableウインドウがオープンします。

詳細は，5.1.2 Stackウインドウを参照してください。

3.13.6 Logウインドウ

メッセージ・ログの内容を表示します。ログには，TDシリーズの動作中に生成されたメッセージやその他の情報のリストが入っています。ログを見ると，プログラムが停止した原因，ブレークポイントの結果，ユーザがログに記録した構造体の値などが分かります。

このウインドウにより，過去にプログラムが行った動作を確認することができます。

詳細は，7.8 Logウインドウを参照してください。

3.13.7 Variablesウインドウ

プログラムの中のある場所からアクセス可能なすべての変数を表示します。上のペインにはグローバル変数が，下のペインには現在の関数またはモジュールにローカルな変数が表示されます（もし変数があれば）。

このウインドウは，先頭の何文字かは分かっているが正確な名前が思い出せない関数や変数を見つけ出したいときに便利です。

詳細は，5.1.1 Variablesウインドウを参照してください。

3.13.8 Fileウインドウ

ディスク・ファイルの内容を表示します。ファイルの内容は16進数バイト形式またはASCIIテキスト形式のどちらの形式でも見ることができます。テキストまたはバイト列を検索したり，ディスク上のファイルの一部を直接パッチすることもできます。

このウインドウは，ディスク・ファイルを扱うプログラムのディバグ中に，ファイルの内容を変更してプログラムの動作を変えたいときに便利です。またファイルの内容の間違いを訂正したり，プログラムにより作成されたファイルの内容が正しいかどうかを調べたりすることもできます。

詳細は，8.4 Fileウインドウを参照してください。

3.13.9　CPUウインドウ

CPUの現在の状態を表示します。このウインドウには，次の５つのペインがあります。

- ●逆アセンブルされた機械語が表示されるペイン
- ●データが16進数バイト形式で表示されるペイン
- ●スタックの内容が16進数バイト形式で表示されるペイン
- ●CPUのレジスタの内容が表示されるペイン
- ●CPUのフラグの状態が表示されるペイン

CPUウインドウは，ソース・コードの１行に対応する一群の命令や，構造体を構成するバイトを監視したいときに便利です。またアセンブリ・コードが分かる方には，発見しにくいタイプのバグを見つけるときに使えます。

ディバグしているプログラムが，ソース・コードの行の途中の命令で停止した場合に，自動的にCPUウインドウがオープンすることがあります。

詳細は，11.2　CPUウインドウを参照してください。

3.13.10　Dumpウインドウ

メモリの，ある領域の内容だけを見たいときに，このウインドウを使います。このウインドウは，CPUウインドウのDataペインと同じものです。

Dumpウインドウは，メモリの指定された領域の内容を表示します。データは，ASCII形式，16進数バイト形式，ワード形式，ダブル・ワード形式，または任意の浮動小数点形式で表示させます。このウインドウのローカル・メニューには，表示されているデータを変更したり，データの表示形式を切り替えたり，データのブロックを操作したりするためのコマンドがあります。

詳細は，11.5　Dumpウインドウを参照してください。

3.13.11　Registersウインドウ

CPUのレジスタやフラグの内容だけを見たいときに，このウインドウを使います。このウインドウには，２つのペインがあり，それぞれCPUウインドウのレジスタ・ペインおよびフラグ・ペインと同じものです。

Registersウインドウは，CPUのレジスタとフラグの内容を表示します。このウインドウのローカル・メニューには，レジスタやフラグの内容を変更するためのコマンドがあります。

詳細は，11.6　Registersウインドウを参照してください。

3.13.12　Numeric processorウインドウ

V55SC，V55PIには浮動小数点演算用コプロセッサが用意されていないため，TD423，TD433ではこのウインドウを開くことができません。

3.13.13　Execution historyウインドウ

実行された最後の行まで，プログラムのアセンブリ・コードとソース・コードを表示します。上のペインには，実行されたコードが表示され，プログラムの実行をある場所まで戻すために使用できます。下のペインはTDシリーズでは使用できません。

詳細は，5.3　実行ヒストリ機能を参照してください。

3.13.14　Hierarchyウインドウ

現在のモジュールで使われているすべてのオブジェクトやクラスの型の階層ツリーを表示します。

このウインドウには次の２つのペインがあります(ただし，多重継承を使うC++プログラムをディバグしている場合は，ハイライト表示されているクラスの型の親を表示する３番目のペインが開きます)。

- オブジェクト／クラスの型のリストを表示するペイン
- オブジェクト／クラスの階層ツリーを表示するペイン

このウインドウは，現在のモジュールで使用されているオブジェクトまたはクラスの関係を示します。またこのウインドウのローカル・メニューにより，オブジェクトまたはクラス，その要素のデータ・フィールドやメンバ，およびメソッドまたはメンバ関数などを調べることができます。

詳細は，10.1　オブジェクトやクラスの階層を調べるを参照してください。

3.13.15　Targetウインドウ

V55SC，V55PIの，選択された周辺レジスタ・リソースの現在の内容を表示します。Targetウインドウでは，ローカル・メニューのコマンドを使用し，周辺レジスタやその他のターゲット依存のリソースの内容を変更可能です。

詳細は，12.4　周辺レジスタ・ビューを参照してください。

3.13.16　ICEウインドウ

IE特有のリソースであるエミュレーション・メモリやハードウエア・ブレークポイント，リアルタイム・トレース・バッファなどの現在の状態を表示します。View | ICEコマンドやBreakpoint | Hardware breakpointコマンドにより，上記のリソースをユーザのアプリケーションの目的に合わせて変更できます。

詳細は，第13章　IEレファレンスを参照してください。

3.14　同じ種類のウインドウ

Dump, File, Moduleの３種類のウインドウは，View | Anotherを選択することにより，同時に複数個オープンできます。これにより，アセンブリ・コードの異なる部分，プログラムが使用または作成する複数のファイル，複数のプログラム・モジュールなどを同時に見ることができます。

TDシリーズは，コマンドに応答してこれらのウインドウを自動的にオープンすることがあります。たと

えばディバグしているプログラムがソース・コードの行の途中の命令で停止した場合，自動的にCPUウインドウがオープンします。

3.15　Inspectウインドウ

Inspectウインドウは，選択されている変数の現在の値を表示します。このウインドウはData | Inspectを選択するかローカル・メニューからInspectを選択するとオープンします。クローズするには，通常はESCキーを押すか，マウスでクローズ・ボックスをクリックします。複雑なデータ構造を調べるために複数のInspectウインドウを連続してオープンしているときは，GRPH-F3を押すかWindow | Closeコマンドを選択することにより，すべてのInspectウインドウを一度にクローズすることができます。

変数や式の値，配列の要素などを見たいときも，このウインドウを使います。ペインの数は，調べようとしているデータの性質によって異なります。Inspectウインドウは，表示されるデータの型に適した形式でオープンします。単純なスカラ（int，floatなど）だけでなく，ポインタ，配列，レコード（Pascalのデータ型の1つ），構造体，共用体などの複雑なデータ構造も表示することができます。各データ型の項目は，プログラムのソース・コードで見なれている書式に近い書式で表示されます。

Inspectウインドウは，Inspectコマンドを選択することにより，いくつでもオープンできます。これに対し，Module，File，Dumpの各ウインドウを複数個オープンできるのはView | Anotherコマンドだけです。

3.16　アクティブ・ウインドウ

TDシリーズでは，複数のウインドウを同時にオープンできますが，アクティブなウインドウは一度に1つだけです。アクティブなウインドウは，次の基準により見つけ出すことができます。

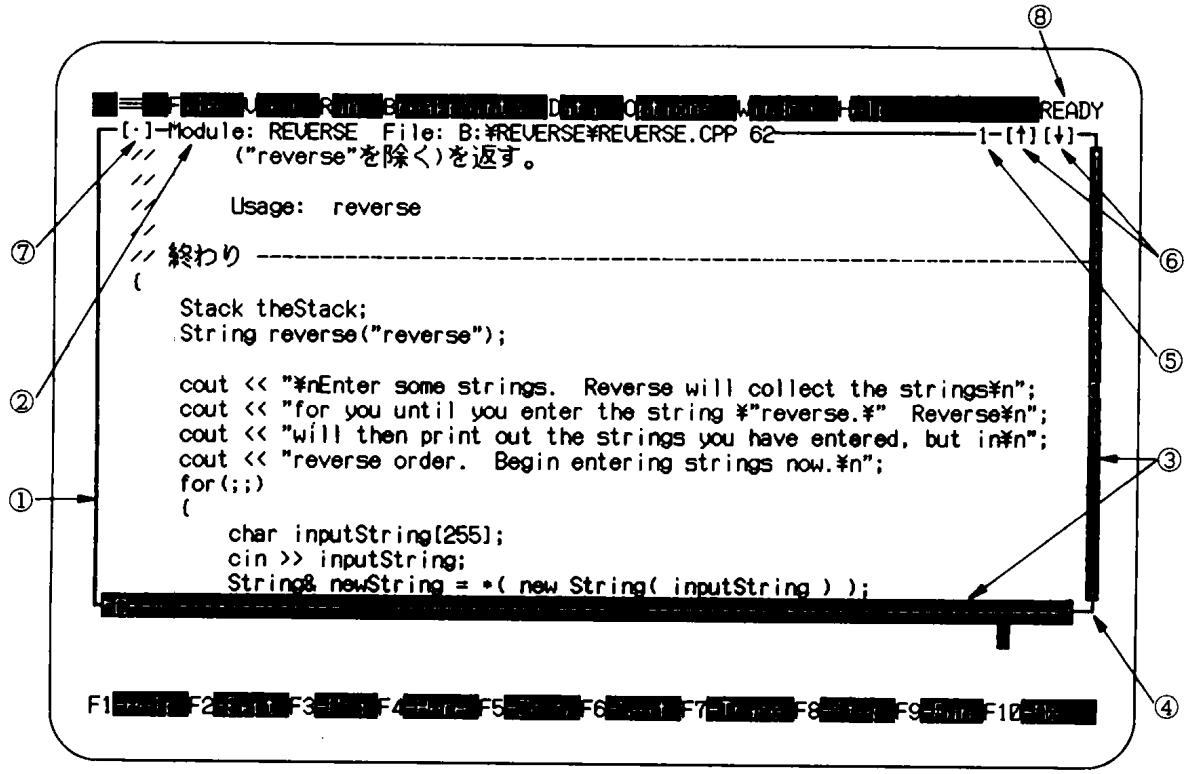
- 太い線で囲まれているウインドウ。
- タイトルがハイライト表示されているウインドウ。
- その中にカーソルやハイライト・バーがあるウインドウ。
- ウインドウが重なりあっているときは，一番上のウインドウ。

コマンドを呼び出したり，テキストを入力したり，表示をスクロールしたりしたとき，影響を受けるのはアクティブ・ウインドウだけです。ほかのウインドウには，影響しません。

3.17　ウインドウの構成

ウインドウの構成とその機能を次に示します。

図３－５　ウインドウの構成



①　外枠

ウインドウがアクティブのときは太線，アクティブでないときは細線。

②　タイトル

左上に表示されます。

③　スクロール・バー（１つまたは２つ）

ウインドウに一度に表示できる量を越える情報があるときに表示されます。スクロール・バーはマウスで操作します。

- バーの両端の矢印をクリックすると，その矢印の方向に１行または１文字スクロールします。
- バーのスクロール・ボックスの両側の領域をクリックすると，矢印の方向に１ウインドウ・サイズ分移動します。
- スクロール・ボックスをドラッグすると，その方向にその距離だけ移動します。

備考　特定のアイコンの上でマウスのボタンを押し，押したまま移動することをドラッグするといいます。

④　リサイズ・ボックス

右下隅に表示されます。リサイズ・ボックスをドラッグすると，ウインドウが拡大または縮小します。

⑤　ウインドウ番号

右上に表示されます。そのウインドウが何番目にオープンされたかを示します。

⑥　ズーム・ボックス，アイコン化ボックス

右上隅に表示されます。左のボックスにはズーム・アイコンが，右のボックスにはアイコン化アイコンが入っています。ズーム・アイコンをマウスでクリックすると，ウインドウが画面いっぱいに拡大します。アイコン化アイコンをクリックすると，そのウインドウがアイコン化（アイコンと呼ばれる小さいボックスに縮小）されます。ウインドウが画面いっぱいにズームされているときは，アイコン化ボックスだけが表示されます。ウインドウがアイコン化されているときは，ズーム・ボックスだけが表示されます。

⑦　クローズ・ボックス

左上隅に表示されます。このボックスをマウスでクリックすると，ウインドウがクローズします。

⑧　動作状態の表示

画面の右上隅には，いつでも現在の動作状態（カーソルの位置）が表示されます。

カーソルの位置	画面右上隅の表示
ウインドウ	READY
メニュー	MENU
ダイアログ・ボックス	PROMPT

備考　ウインドウが重なり合っているとき，一番上のウインドウが常にアクティブ・ウインドウです。

3.18　ウインドウの操作

TDシリーズには，このように多種類のウインドウがあるため，同時に複数のウインドウをオープンすることになります。したがって，あるウインドウから別のウインドウに移動したり，ウインドウを動かしたり，邪魔にならないように重ね合わせたり，縮小したりといった操作が簡単に行えるようになっています。

ほとんどのウインドウ管理コマンドはWindowメニューの中にあります。それ以外に≡(システム)メニューにもコマンドがいくつかあります。

3.18.1　ウインドウ（ペイン）間の移動

(1) ウインドウ間の移動

各ウインドウには，オープンされた順に右上隅に番号が付けられます。通常は，Moduleウインドウが1に，Watchesウインドウが2になります。その後にオープンするウインドウは3以上の番号になります。

この番号システムにより，ウインドウからウインドウへ容易に移動することができます。ウインドウ間の移動は，次のいずれかの方法で行います。

- GRPHキーと移動先のウインドウ番号を押す

  ウインドウ番号が1-9のものはどれでもアクティブになります。たとえばGRPH-2を押すと，Wachesウインドウがアクティブになります。

- Window | Nextを選択するか，F6を押す

  ウインドウ番号順にアクティブになります。この機能は，アクティブにしたいウインドウがほかのウインドウの下にあって番号が見えないときに便利です。

- Windowメニュー（メニュー・バーからWindowを選択）の下半分から目的のウインドウを選択する

  Windowメニューの下半分には現在オープンしている1-9番のウインドウ名が，表示されます。この中から目的のウインドウ番号を選択すると，そのウインドウがアクティブになります。ウインドウを10個以上オープンしているときは，WindowメニューからWindow pickコマンドを選択してください。すべてのウインドウ名を表示したポップアップ・メニューが現れます。

- マウスで任意のウインドウをクリックする

(2) ペイン間の移動

ウインドウに複数のペイン（異なる種類のデータが表示されているウインドウ内の区画）があるときは，次のような方法でペイン間を移動します。

- TABキーまたはSHIFT-TABを押す
- Window | Next Paneを選択する
- マウスで任意のペインをクリックする

ペイン上には，点滅するカーソルかまたはハイライト・バーが現れます。

●点滅するカーソルが現れたとき

標準のキー・パッド・コマンド（カーソル・キー）を使ってテキスト上を移動できます。（ROLLDOWN：1画面上へ移動，CTRL-HOMECLR：ペインの最上部へ移動，CTRL-ROLL-DOWN：リストの先頭の行へ移動など）

●ハイライト・バーが現れたとき

上の場合と同様のキー・パッド・コマンドのほかに特殊な操作ができます。たとえば各項目が英字の文字列のリスト上では，インクリメンタル・マッチ機能（3.11　インクリメンタル・マッチ　参照）によって容易にリストから項目を選択できます。

3.18.2　ウインドウの移動と大きさの変更

新しいウインドウをオープンすると，そのウインドウは現在カーソルのある場所の近くに現われます。ウインドウの大きさは，そのウインドウの種類に適したディフォールトの，あらかじめ設定された大きさです。ウインドウの大きさまたは場所が不便なときは，Window | Size/moveコマンドにより調節できます。

Window | Size/moveコマンドを選択すると，アクティブ・ウインドウの枠が細線に変わります。この状態になったら，カーソル・キーを使ってウインドウを移動させることができます。また，SHIFT-カーソル・キーを使うとウインドウの大きさを変えることができます。目的の場所と大きさに設定したら↵を押します。

マウスを使うと，ウインドウの移動と大きさの変更がより簡単に行えます。

●右下隅のリサイズ・ボックスをドラッグすると，ウインドウの大きさが変わります。

●タイトル・バーまたはウインドウの端（スクロール・バー以外ならどこでもよい）をドラッグすると，ウインドウが動きます。

ウインドウをすばやく拡大または縮小したいときは，Window | Zoomを選択します。マウスでは，ウインドウの右上隅のズーム・ボックス（[↑]，[↓]）をクリックします。

あるウインドウが邪魔だがクローズはしたくないというときには，そのウインドウをアクティブにしてWindow | Iconize/restoreを選択します（または下向きのズーム・ボックスをクリックします）。すると，そのウインドウは縮小して小さなボックス（アイコン）になります。このアイコンには，ウインドウの名前，クローズ・ボックス，およびズーム・ボックスが表示されます。

このウインドウを元の大きさに戻すには，アイコンをアクティブにして再びWindow | Iconize/restoreを選択するか，アイコンのズーム・ボックスをクリックします。

3.18.3　ウインドウのクローズと復活

Window | Closeを選択するか，ホット・キーGRPH-F3を押すと，アクティブなウインドウをクローズすることができます。マウスでは，ウインドウの左上隅のクローズ・ボックスをクリックします。

誤って，クローズしたくないウインドウをクローズしてしまったときは，Window | Undo closeを選択するかGRPH-F6を押せば，元に戻すことができます。

≡（システム）| Restore standardを選択すると，画面を元のレイアウトに戻すことができます。

3.18.4　ウインドウのレイアウトの保存

Options | Save optionsコマンドを使って，その画面の構成を保存することができます。画面の構成をTDCONFIG.TDという名前のファイルに保存したときは，MS-DOSからTDシリーズを起動すると常にその画面が現われます。このファイルは，起動時に自動的にロードされるコンフィギュレーション・ファイルです。ほかの名前のコンフィギュレーション・ファイルに保存したときは，Options | Restore optionsコマンドを使ってロードすることができます。

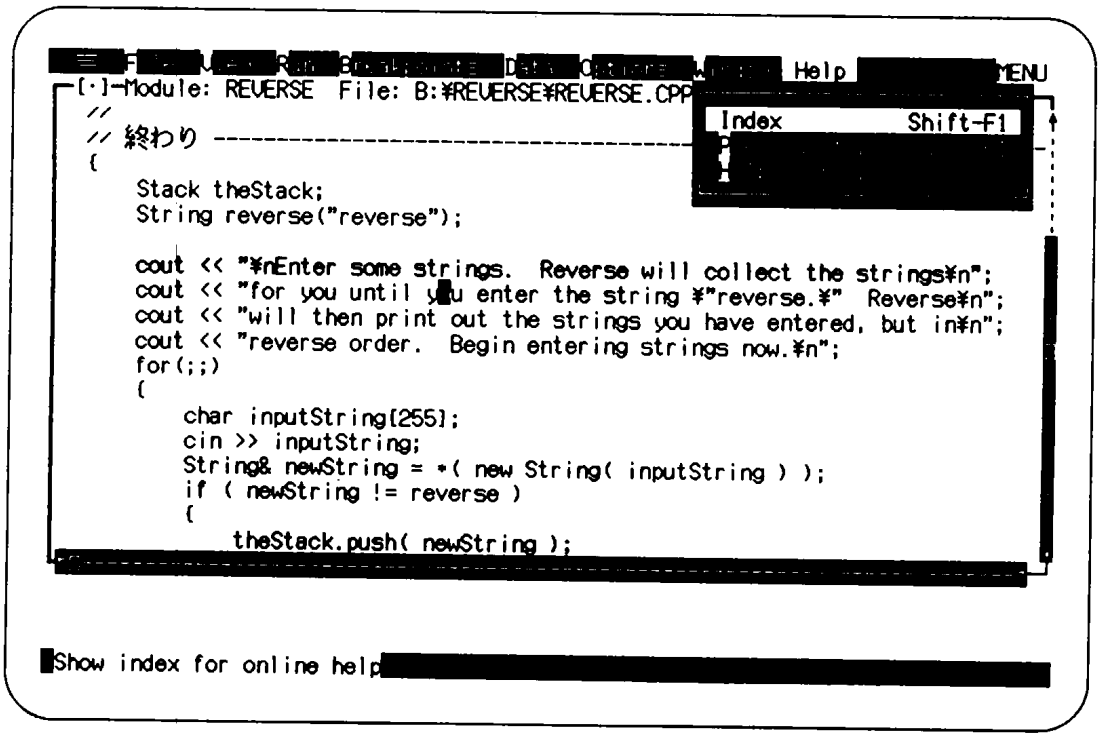
3.19　オンライン・ヘルプ

F1を押すと，TDシリーズはヘルプ・ウインドウをオープンして，現在の状況（どのウインドウまたはペインにいるか，またはどのメニュー・コマンドを選択しているかなど）に応じて適切な情報を表示します。マウスでは，F1をクリックしてヘルプ画面を表示します。

いくつかのヘルプ画面には，ハイライト表示されたキー・ワードがあります。キー・ワードに対しては，さらに詳しいヘルプがあります。TABキーまたはSHIFT-TABでキー・ワードのどれかに移動して ↵ を押すと，そのキー・ワードのヘルプ画面が表示されます。HOMECLRキーまたはSHIFT-HOMECLRを押すと，それぞれ現在の画面の最初のキー・ワードまたは最後のキー・ワードに移動することができます。

メニュー・バーからHelpを選択して，オンライン・ヘルプにアクセスすることもできます。

図3-6　Helpメニュー



直前に見ていたヘルプ画面に戻るにはGRPH-F1を押します。ヘルプ・システムの中にいるときにROLLDOWNを押すと前のページが表示されます。ROLLUPは次のページを表示します（関連する一群の画面の中にいるときにのみ有効です）。

ヘルプの目次（Help Index）にアクセスするには，ヘルプ・システムの中でF1を押します。ヘルプ・システムから抜け出すにはESCキーを押します。

3.20　ステータス行

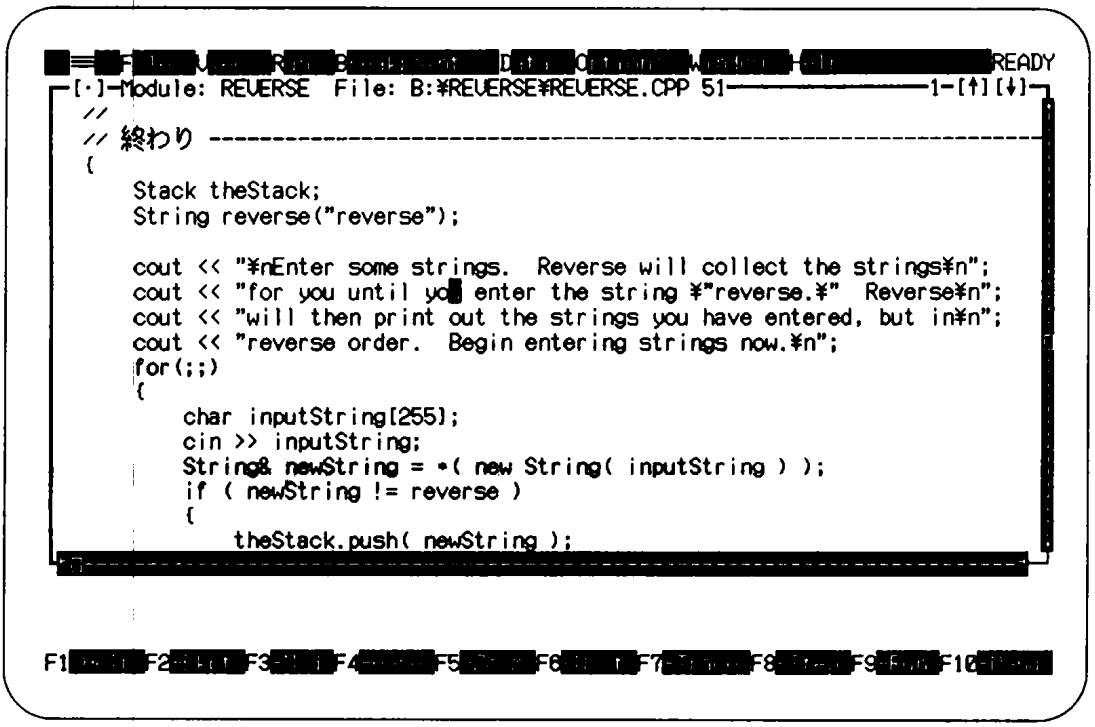
TDシリーズでは，常に画面の一番下にクイック・レファレンス行が表示されます。この行の表示は，カーソルの位置によって異なります。

カーソルの位置	クイック・レファレンス行
ウインドウ	キー・ストローク・コマンド
メニュー	選択されているメニュー・コマンドの要約説明
ダイアログ・ボックス	入力の指示

3.20.1　ウインドウの中にいるとき

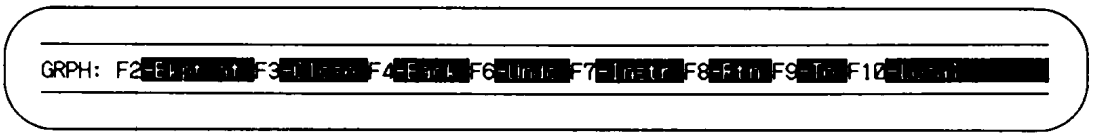
最下行（クイック・レファレンス行）には，通常，ファンクション・キーにより入力できるコマンドが表示されます。

図３－７　通常画面



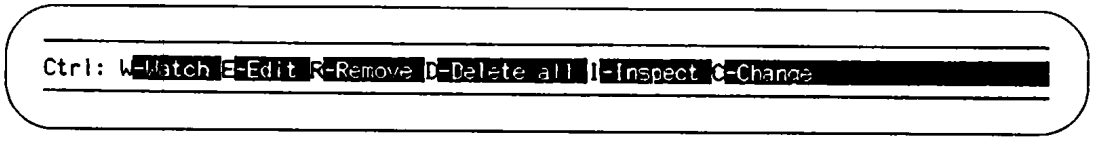
GRPHキーを少し押したままにすると，GRPHキーとファンクション・キーの組み合わせで入力できるコマンドが表示されます。

図３－８　クイック・レファレンス行（GRPHキー）



CTRLキーを少し押したままにすると，CTRLキーと文字キーの組み合わせで入力できるコマンドが表示されます。これらのコマンドは，現在選択されているウインドウまたはペインのローカル・メニュー・コマンドです。ローカル・メニュー・コマンドの数が多くてクイック・レファレンス行に表示しきれないときは，始めの９つのコマンドだけが表示されます。GRPH-F10またはCTRL-F10を押してローカル・メニューを表示すれば，すべてのローカル・メニュー・コマンドを見ることができます。

図３－９　クイック・レファレンス行（CTRLキー）



マウスでは，レファレンス行に表示されているコマンドをクリックして，そのコマンドを実行します（GRPH-およびCTRL-キー・コマンドは，GRPHキーまたはCTRLキーでそのコマンドを表示させてからクリックします）。

3.20.2　メニューまたはダイアログ・ボックスにいるとき

メニューまたはダイアログ・ボックスにいるときは，現在選択されているコマンドの要約説明やダイアログ・ボックスへの入力の指示が表示されます。たとえば，View | Registersがハイライト表示されているときは，最下行に“Open a CPU register window”と表示されます。

グローバル・メニューおよびローカル・メニューのどちらのメニューにいるときでも，メニュー項目についてのヘルプを表示します。

〔メ　モ〕

第4章　起　　動

この章では次のことを説明します。

- ●準備
- ●ハードウエアの設定
- ●MS-DOSのコマンド・ラインからTDシリーズを起動する方法
- ●TDシリーズのコマンド・ライン・オプションの使い方[注]
- ●指定したオプションをコンフィギュレーション・ファイルに保存する方法
- ●TDシリーズの中からMS-DOSのコマンドを実行する方法
- ●ディバグを終了してMS-DOSに戻る方法

注　コマンド・ライン・オプションは，ディバグ環境を設定するためのものです。

4.1　準　　備

TDシリーズを使用するには，プログラムをコンパイル・リンクするときに，コンパイラに必要なすべてのディバグ情報を生成するように指示してください。ディバグ情報を生成せずにオブジェクト・モジュールをひとつでもコンパイルしたときは，プログラム全体をソース・コードでディバグできるようにするために，プログラムのすべてのモジュールを再コンパイルする必要があります。ただし，プログラムの一部のモジュールに対してだけディバグ情報を生成することもできます（非常に大きなプログラムのディバグではその必要があります）。

4.1.1　Borland International Inc.製コンパイラの準備

Borland International Inc.製のCコンパイラであるBorland C++，Turbo C++，Turbo Cを使用するには，コンパイラのコマンド行に−vオプションを指定して，完全ディバグ情報をオブジェクト・ファイルに入れてください。TLINKでは/vオプションを使って，リロケータブル・ロード・モジュール.EXEファイルの最後にディバグ情報が付加されるようにしてください。

LC70116を使い，AXE出力を生成するとTDシリーズでソース・ディバグが可能になります。

注意　最適化（オプティマイズ）を必ずオフにしてください。
最適化をオフにすると，ソース・コードを1行ずつ実行しているときにごくまれに起こる，複数行のソース・コードをスキップするように表示される現象を防ぐことができます。

4.1.2 Microsoft Corp.製コンパイラの準備

Microsoft Corp.製のCコンパイラであるMS-Cを使用するには，コマンド行に/Ziオプションを指定して，完全ディバグ情報をオブジェクト・ファイルへ入れてください。また，LINKでは/COオプションを使って，リロケータブル・ロード・モジュール.EXEファイルの最後にディバグ情報が付加されるようにしてください。

ディバグ情報は，TDCONVRTユーティリティを使用して，Code View情報をターボ・ディバッガ形式へ変換してください。最後にLC70116を使用して，AXE形式を生成します。

注意　最適化（オプティマイズ）を必ずオフにしてください。
最適化をオフにすると，ソース・コードを1行ずつ実行しているときにごくまれに起こる，複数行のソース・コードをスキップするように表示される現象を防ぐことができます。

4.2 ハードウエアの設定

TDシリーズを起動する前に，操作対象であるIEの設定を行ってください。

4.2.1 IE本体の設定

それぞれのユーザーズ・マニュアルの説明に従って，内部クロックの変更と付属品の接続を行ってください。

IE-70423-BX　ユーザーズ・マニュアル（EEU-820）
IE-70433-BX　ユーザーズ・マニュアル（EEU-818）

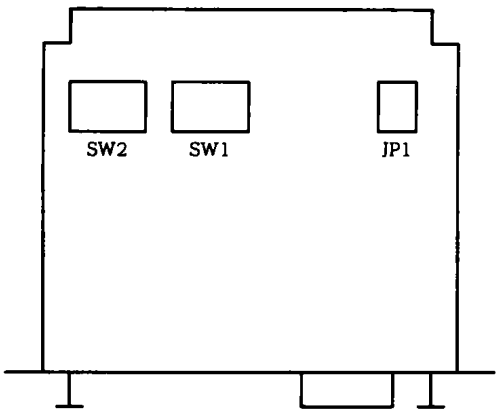
4.2.2 パソコン・インタフェース・ボードの設定

IE-70423，70433-BXとホスト・マシン（PC-9800シリーズまたはPC/AT）をハンドシェークするために，パソコン・インタフェース・ボード上で次の2つのことを行ってください。

・DIPスイッチ（SW1，SW2）でI/Oアドレスを設定する。
・ジャンパ・スイッチ（JP1）で割り込みレベルを設定する。

(1) PC-9800シリーズの場合

ボード上の設定箇所は，次の3箇所（SW1，SW2，JP1）です。



(a) I/Oアドレスの設定

IEの制御に使用するパソコンのI/Oポートのアドレスを設定します。

I/Oアドレスは，インタフェース・ボード上のDIPスイッチ1（SW1），DIPスイッチ2（SW2）で設定します。SW1，SW2は，ハンドシェーク・バス・アドレスの設定スイッチです。SW1，SW2を使用してI/O空間の先頭アドレスを×××0h番地に設定し，×××0h番地から×××Eh番地を使用します。

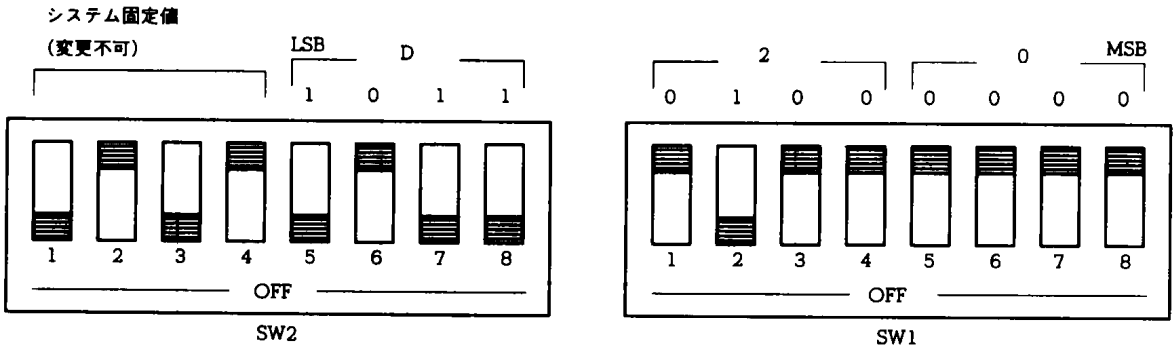
先頭アドレスの値とDIPスイッチの対応は次のとおりです。

先頭アドレス＝×××0h
- 0h：自動的に0が設定されます(SW2の4-1はこのアドレス設定には無関係です)
- ×0h：SW2の8-5
- ××0h：SW1の4-1
- ×××0h：SW1の8-5

DIPスイッチのアドレス値指定は1がOFF，0がONです。

TD423，TD433を使用する場合は，先頭アドレスを02D0hにしてください。この場合次のように設定します。

図4-1 DIPスイッチ1，2の設定（PC-9800シリーズ）

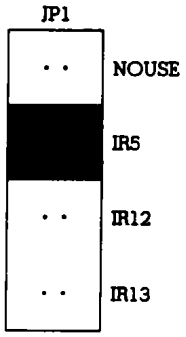


(b) 割り込みレベルの設定

IEの制御に使用する割り込みレベルをJP1で設定します。

TD423，TD433を使用する場合は，次の図のようにIR5と書かれたピンにショート・ピンを差し込んでください。

図4-2 JP1の設定（PC-9800シリーズ）



JP1の名称と，パソコンの拡張用スロットでの割り込み名称の対応は次のとおりです。

JP1	割り込み
IR5	INT1
IR12	INT5
IR13	INT6

INT1（IR5）は，通常は空きの割り込みとなっていますが，シリアル・インタフェースなどですでに使用している場合があります。この場合，そのボードの設定を変更するか，このインタフェース・ボードの設定をIR12に変更してください。

IR13はマウスの割り込みで使用しますので，設定しないでください。

(c) 環境変数の設定

インタフェース・ボードの設定を変更した場合，次の環境変数の設定が必要になります。

●PC_BASE　インタフェース・ボードのポート・アドレスの環境変数を設定します。
何も設定しない場合，先頭アドレスは02D0h番地とみなされます。

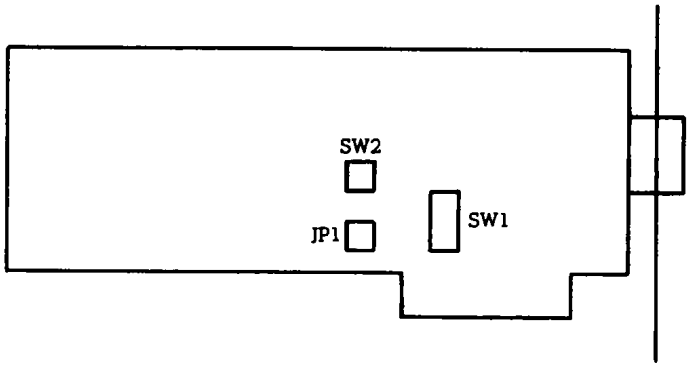
例　先頭アドレスを1230h番地にする場合
SET PC_BASE＝1230

●IE_INT　割り込みレベルの環境変数を設定します。
IR12の場合は14hを指定します。
何も設定しない場合はIR5とみなされます。

例　SET IE_INT＝14

(2) PC/ATの場合

ボード上の設定箇所は，次の3箇所（SW1，SW2，JP1）です。



(a) I/Oアドレスの設定

IEの制御に使用するパソコンのI/Oポートのアドレスを設定します。
SW1，SW2を使用してI/O空間の先頭アドレスを×××0h番地に設定し，×××0h番地から×××Eh番地を使用します。
先頭アドレスの値とDIPスイッチの対応は次のとおりです。

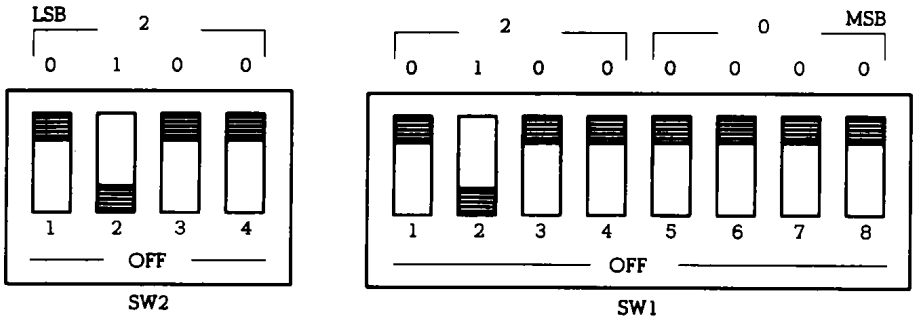
先頭アドレス＝×××0h

- 自動的に0が設定されます（対応するSWはありません）
- SW2の4-1
- SW1の4-1
- SW1の8-5

DIPスイッチのアドレス値指定は1がOFF，0がONです。

TD423，TD433を使用する場合は，先頭アドレスを0220hにしてください。この場合次のように設定します。

図4-3　DIPスイッチ1，2の設定（PC/AT）

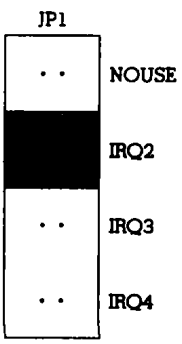


(b) 割り込みレベルの設定

IEの制御に使用する割り込みレベルをJP1で設定します。

TD423，TD433を使用する場合は，次の図のようにIRQ2と書かれたピンにショート・ピンを差し込んでください。

図4-4　JP1の設定（PC/AT）



IRQ2は，通常は空きの割り込みとなっていますが，ほかの目的に使用されている場合があります。この場合システム・ソフトが正常に動作しなくなるため，このJP1の設定をIRQ3に変更してください。

また，IRQ4は使用できません。

(c) 環境変数の設定

インタフェース・ボードの設定を変更した場合，次の環境変数の設定が必要になります。

●PC_BASE　インタフェース・ボードのポート・アドレスの環境変数を設定します。
何も設定しない場合，先頭アドレスは0220h番地とみなされます。

例　先頭アドレスを0120h番地にする
SET PC_BASE＝0120

●IE_INT　割り込みレベルの環境変数を設定します。
IRQ3の場合は0Bhを指定します。
何も設定しない場合は，IRQ2とみなされます。

例　SET IE_INT＝0B

(d) CONFIG.SYSの設定

FILES，BUFFERSはそれぞれ10程度あれば充分です。

ANSI.SYSは必ず組み込んでください。

その他の不要なデバイス・ドライバはなるべく組み込まないでください。

4.3 起　　動

TD423, TD433の起動方法の手順を次に示します。

① MS-DOSのプロンプトから，td423bxまたはtd433bxと入力する。

② オプションのコマンド・ライン引き数，ディバグするプログラム名およびプログラムの引き数を入力し，↵を押します。プログラムがロードされ，文ごとにステップできるようにソース・コードが表示されます。

コマンド・ラインの一般的な書式は，次のとおりです。

①　　②
td423bx△ [options] △ [progname]

または

td433bx△ [options] △ [progname]

カッコ（[ ]）は入力する必要はありません。

②の入力は省略することができます。①だけ入力するとディフォールトのオプションが使用されます。

TD423, TD433でプログラムをソース・レベル・ディバグするには，.AXEファイルと元のソース・ファイルの両方が必要です。

4.4 コマンド・ライン・オプション

TDシリーズのコマンド・ライン・オプションは，すべて頭にマイナス記号（−）を付け，シリーズ名，およびプログラム名と，少なくとも１つの空白で区切らなければなりません。コマンド・ライン・オプションのあとにマイナス記号を付けると，そのオプションをオフにすることができます。たとえば−l−とすると，アセンブラ・スタートアップ・オプションがオフになります。

次におもなコマンド・オプションについて説明します（付録A　コマンド・ライン・オプション参照）。

4.4.1 コンフィギュレーション・ファイルのロード（−c）

このオプションは，指定されたコンフィギュレーション・ファイルをロードします。cとファイル名の間にスペースを入れないでください。

例　−cmyconfig.td

- −cオプションを指定しない場合は，TDCONFIG.TDがロードされます。
- −cオプションもTDCONFIG.TDもない場合は，ユーザのコンフィギュレーション指定がないものとして，初期状態で起動します。

4.4.2 ヘルプの表示（−h, −？）

コマンド・ライン・オプションの意味と書式を説明するヘルプ画面を表示します。

4.4.3 アセンブラ・モードの起動（−l）

このオプションを指定したときは，起動時にCPUウインドウが表示されます。コンパイラのスタートアップ・コードは実行されません。

通常は，このスタートアップ・コードはプログラムのロード時に自動的に実行されます。このオプションによってスタートアップ・コードをステップして検査することが可能になります。

4.4.4 マウスのサポート（−p）

このオプションは，マウスのサポートをオンにします。マウスのサポートのディフォールトはオンなので，TDINSTBXを使ってディフォールトをオフにしていないかぎり，このオプションを使う必要はありません。マウスの使用を不可にするには−p−と指定してください。

4.4.5 キャッシュ・サポート（−rc）

TDシリーズによるメモリ・リードのキャッシングをオン／オフします。最高のバンド幅（スピード）を提供するために，アクセスされた最新のメモリ・ロケーションのキャッシュを維持します。障害の危険があった場合，キャッシングをオフするには−rc−をコマンド行で使用します。

4.4.6 ソース・コードの扱い（−s）

−sオプションは，すべてプログラムのシンボルやソース・コードを扱う方法に関するものです。

(1) −sc

このオプションを指定すると，プログラムが文字の大文字／小文字を区別してリンクされていても，シンボル名を入力するときは無視されます。

−scオプションを指定しないときは，プログラムが大文字／小文字を無視するオプションをオンにしてリンクされている場合にのみ無視されます。

(2) −sddirname

ソース・ファイルを探すディレクトリを指定します。1つの−sdオプションでは1つのディレクトリしか指定できません。複数のディレクトリを指定したいときは複数の−sdオプションを使用してください。

TDシリーズは，指定された順に指定されたディレクトリを探します。dirnameは相対パス，絶対パスのどちらでもかまいません。またディスク・ドライブ名を含めることもできます。コンフィギュレーション・ファイルでディレクトリが設定されている場合には，そのリストのあとに，この−sdオプションで指定したディレクトリが追加されます。

(3) −smN

このオプションは，シンボル・テーブルのために確保するメモリ・サイズを設定します。Nには確保したいメモリ・サイズ（キロバイト数）を指定してください。File | Table loadコマンドを使って

マニュアルでシンボル・テーブルをロードしたいときは，このオプションを指定してください。

4.4.7 オーバレイ・プール・サイズ（−y）

−yオプションは，メイン・メモリまたはEMSメモリ内に確保したいオーバレイ領域のサイズを設定するために使用します。

(1) −yN　このオプションは，メイン・メモリ内に確保したいオーバレイのプール・サイズを設定します。Nは確保したいキロバイト数です。TDシリーズでは，通常80 Kの領域サイズを使用します。
設定できる最小のプール・サイズは20 K，最大のプール・サイズは200 Kです。
プログラムをロードするのに十分なメモリがない場合，および小さなプログラムをディバグしていてTDシリーズの性能を向上させたい場合には，このオプションを使用してください。コード領域サイズが小さければ小さいほど，TDシリーズがオーバレイ・プログラムをディスクからロードする頻度が高くなります。コード領域を大きくすると，ディバグするプログラムのために使用できるメモリは少なくなりますが，TDシリーズの実行速度は速くなります。

(2) −yeN　このオプションは，EMSメモリ内に確保したいオーバレイ領域のサイズを設定します。Nは確保したいEMSのページ数（16 Kバイト単位）です。たとえば−ye4は，オーバレイ領域をEMS上に４ページ分確保します。ディフォールトは12ページです。

4.4.8 Vシリーズのレジスタ名，インストラクション表示（−n）

−nオプションを指定すると，レジスタ名，インストラクションがVシリーズ表記になります。

4.5 コンフィギュレーション・ファイル

コンフィギュレーション・ファイルを読み込んだ場合，そこに書き込まれているコマンド・ライン・オプションの設定値が組み込みのディフォールト値に優先します。TDINSTBXを使うと，コンフィギュレーション・ファイルがない場合のディフォールトとなるオプションを設定することができます。またTDINSTBXでは，新たにプロジェクトごとにコンフィギュレーション・ファイルを作ることもできます。

TDシリーズは，コンフィギュレーション・ファイル（ディフォールトはTDCONFIG.TD）を次に示す順序で探します。

① カレント・ディレクトリ
② TDINSTBXで設定されたターボ・ディレクトリ
③ TDシリーズの入っているディレクトリ

TDシリーズがコンフィギュレーション・ファイルを読み込むと，コンフィギュレーション・ファイルの

設定が組み込みのディフォールト値に優先します。MS-DOSからTDシリーズを起動するときに指定されたコマンド・ライン・オプションは，ディフォールトの設定およびコンフィギュレーション・ファイルに優先します。

4.6 エミュレータ・コンフィギュレーション・ファイル

コンフィギュレーション・ファイルに加えてIEのステートをエミュレータ・コンフィギュレーション・ファイルに保存できます。

TD423，TD433の場合，まずディフォールトであるTD423BX (TD433BX).EMUエミュレータ・コンフィギュレーション・ファイルをターゲットのプログラムと同じディレクトリ中で見つけます。この検索が失敗すると，PATH環境変数に指定されたディレクトリを使ってもう一度試みます。

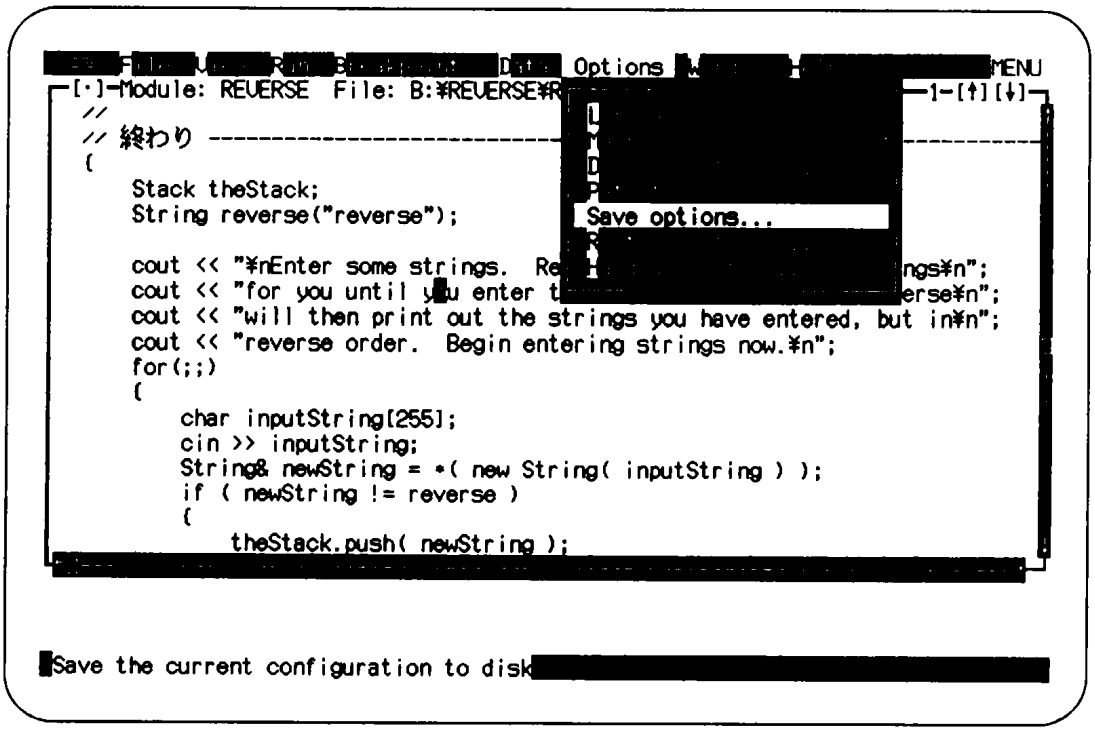
TD423BX(TD433BX).EMUの検索後，プログラムがロードされていればそれぞれのプロジェクト専用のエミュレータ・コンフィギュレーションを探します。また，ディバグされるプログラムと同じ名前で拡張子が.EMUのものを追加します。ロード・ファイルのあるこのディレクトリは，次にそれぞれのプロジェクト専用のエミュレータが存在するかどうかをチェックします。

どちらのコンフィギュレーション・ファイルも見つからない場合，IEコンフィギュレーションは，第13章 IEレファレンスで定義されたディフォールトのステートにセットされます。

4.7 Optionsメニュー

Optionsメニューには，表示画面や全体的な動作を制御する多数のパラメータを設定するメニュー・コマンドがあります。次に各メニュー・コマンドについて説明します。

図4-5 Optionsメニュー



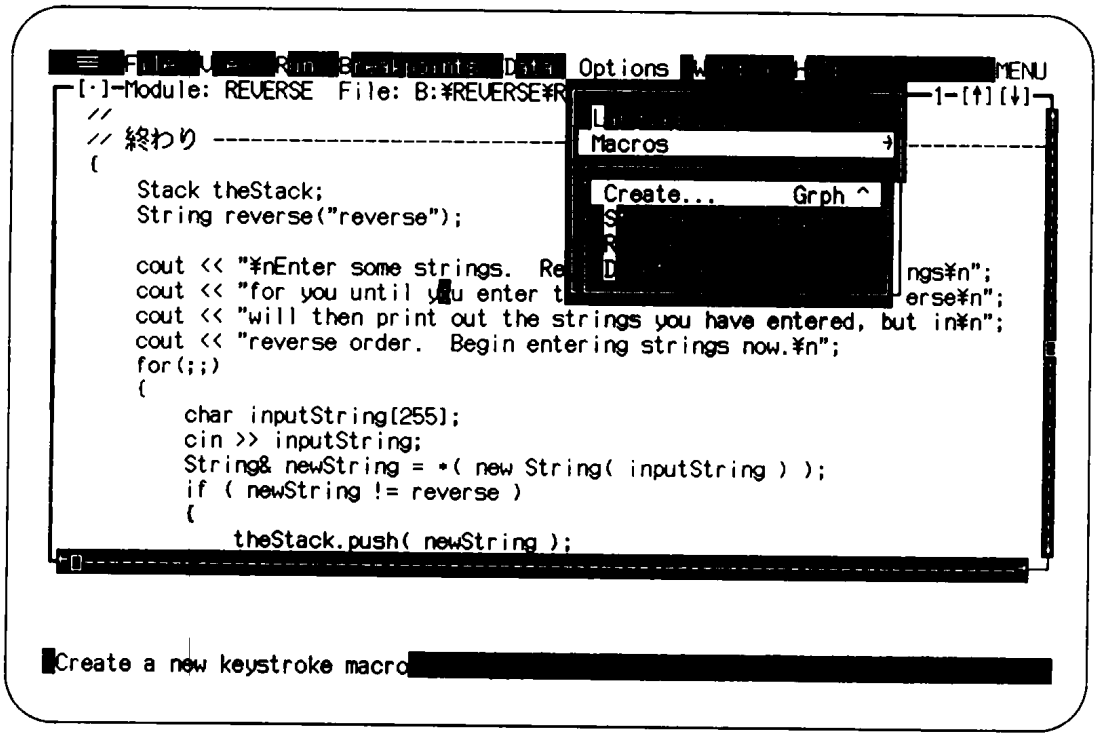
4.7.1 Languageコマンド

式の記述に使用する言語を選択します。各言語で使用できる演算子や式の評価規則については，第9章 式を参照してください。

4.7.2 Macrosコマンド

Macrosコマンドは，新しいキー・ストローク・マクロを設定したり，すでにキーに割り当てられているマクロを削除することができるMacrosメニューを表示します。このメニューには，次の4つのコマンドがあります。

図4-6 Macrosコマンドのポップアップ・メニュー（Options | Macros）



(1) Createコマンド

希望するキー（たとえばGRPH-M）に設定したいキー・ストロークの記録を開始します。記録を開始するには，Options | Macros | Createを選択します。マクロを設定したいキーを入力するよう要求されますので，希望するキーを押してください。記録が開始されます。

記録中は，右上隅に"RECORDING"というメッセージが表示されます。記録したいキー・ストロークを入力すると，入力されたキーはマクロの記録中でないときとまったく同じように機能します。

キー・ストロークの入力が終わったら，Options | Macros | Stop recordingコマンドを選択するか，そのホット・キーGRPH--を押してください。マクロを設定しようとするキー(ここではGRPH-M）をもう一度押すこともできます。

GRPH-^は，マクロの記録を開始するためのホット・キーです。

(2) Stop recordingコマンド

キーに割り当てられているキー・ストロークの記録を終了します。Options | Macros | Createコマンドでキー・ストローク・マクロの設定を開始したときは，このコマンドで記録を終了してください。

GRPH--は，マクロの記録を終了するためのホット・キーです。

(3) Removeコマンド

キーに割り当てられているマクロを削除します。このコマンドを選択すると，マクロを削除したいキーを入力するよう求められます。

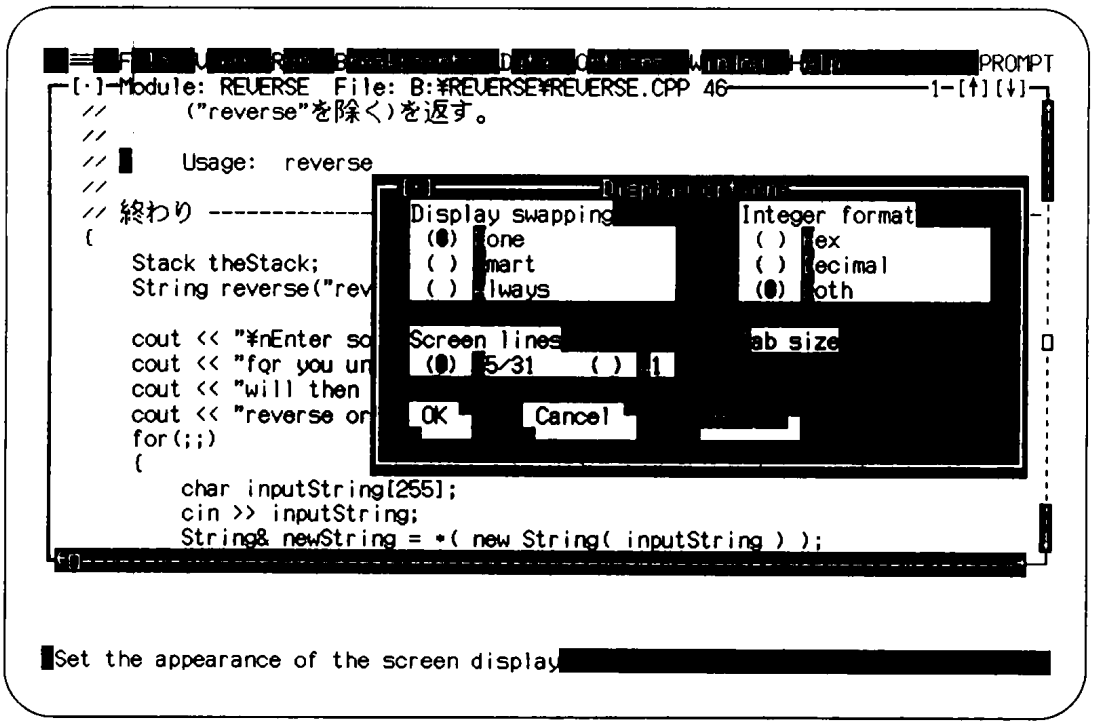
(4) Delete allコマンド

すべてのキー・ストローク・マクロの定義を削除し，マクロが割り当てられていたキーを元の機能に戻します。

4.7.3 Display optionsコマンド

このコマンドは，ダイアログ・ボックスを表示します。ダイアログ・ボックスは，画面表示を制御するいくつかのオプションを設定することができます。

図4－7 Display optionsダイアログ・ボックス（Options | Display options）



(1) Display swappingラジオ・ボタン

このオプションは，TDシリーズでは使用しません。Noneのままにしておいてください。

(2) Integer formatラジオ・ボタン

このラジオ・ボタンにより，整数の表示形式を選択することができます。

Decimal　整数を通常の10進数で表示します。

Hex　整数を現在選択されている言語の16進数表記で表示します。

Both　整数を10進数と16進数の両方で表示します。16進数は10進数のあとにカッコで囲んで表示されます。

(3) Screen linesラジオ・ボタン

このラジオ・ボタンでは，テキスト画面による25行表示とグラフィック画面による46行表示のどちらにするかを設定します。

(4) Tab size入力ボックス

この入力ボックスは，タブ位置の間隔を設定します(タブ間隔は1から32に設定可能)。タブによりインデントされて画面からはみ出しているコードのあるソース・ファイルは，タブ間隔を狭くすることにより，より多くのテキストを見ることができます。

4.7.4 Path for sourceコマンド

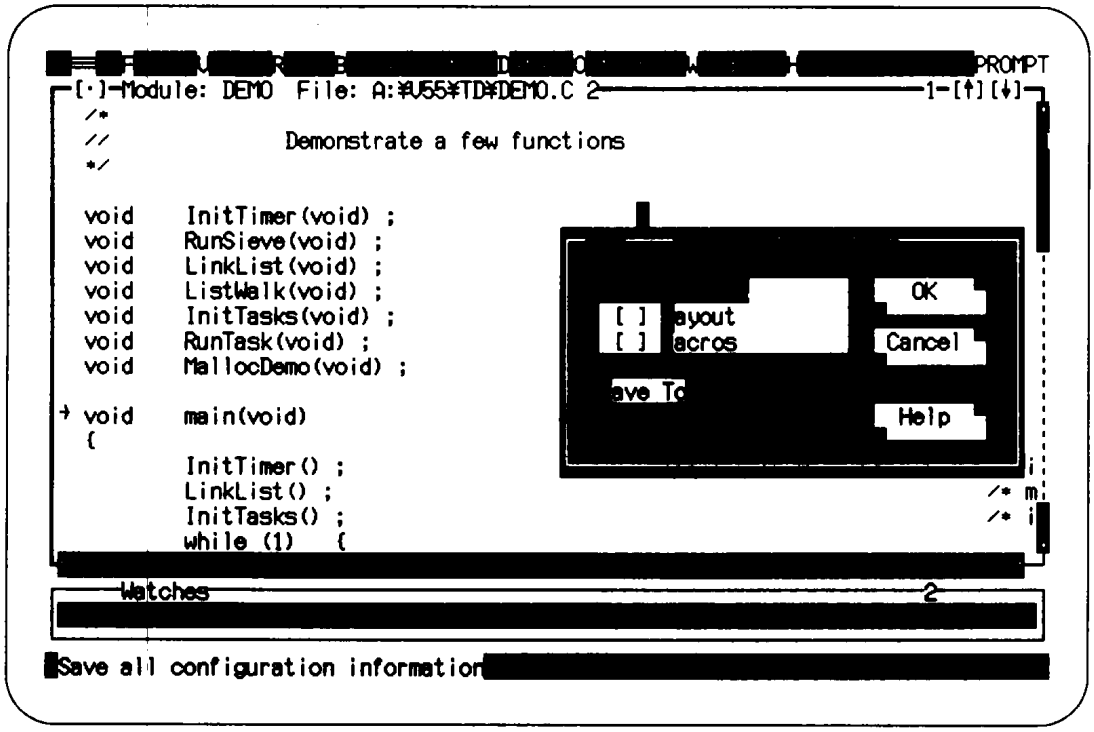
ソース・ファイルを探すディレクトリを設定します。このオプションについては，8.2 Moduleウインドウを参照してください。

4.7.5 Save optionsコマンド

このコマンドは，図4-8のようなSave Configurationダイアログ・ボックスを開きます。現在のオプションをディスク上のコンフィギュレーション・ファイルへ保存可能です。オプションには次のようなものがあります。

- ユーザのマクロ
- 現在のウインドウ・レイアウトとペインの形式
- Optionsメニューでのすべての設定

図4-8 Save Configurationダイアログ・ボックス（Options | Save options）



このSave Configurationダイアログ・ボックスのオプションをオンにすることにより，次の設定のどれか，またはすべてを保存することができます。

- ●Options　Optionsメニューで行ったすべての設定を保存します。
- ●Layout　ウインドウ（およびペイン）の配置だけを保存します。
- ●Macros　現在設定されているマクロだけを保存します。

Save To入力ボックスを使うと，設定を保存するコンフィギュレーション・ファイルの名前を変更することができます。複数のコンフィギュレーション・ファイルに，異なるマクロやウインドウ配置などを保存しておくことができます。

4.7.6 Restore optionsコマンド

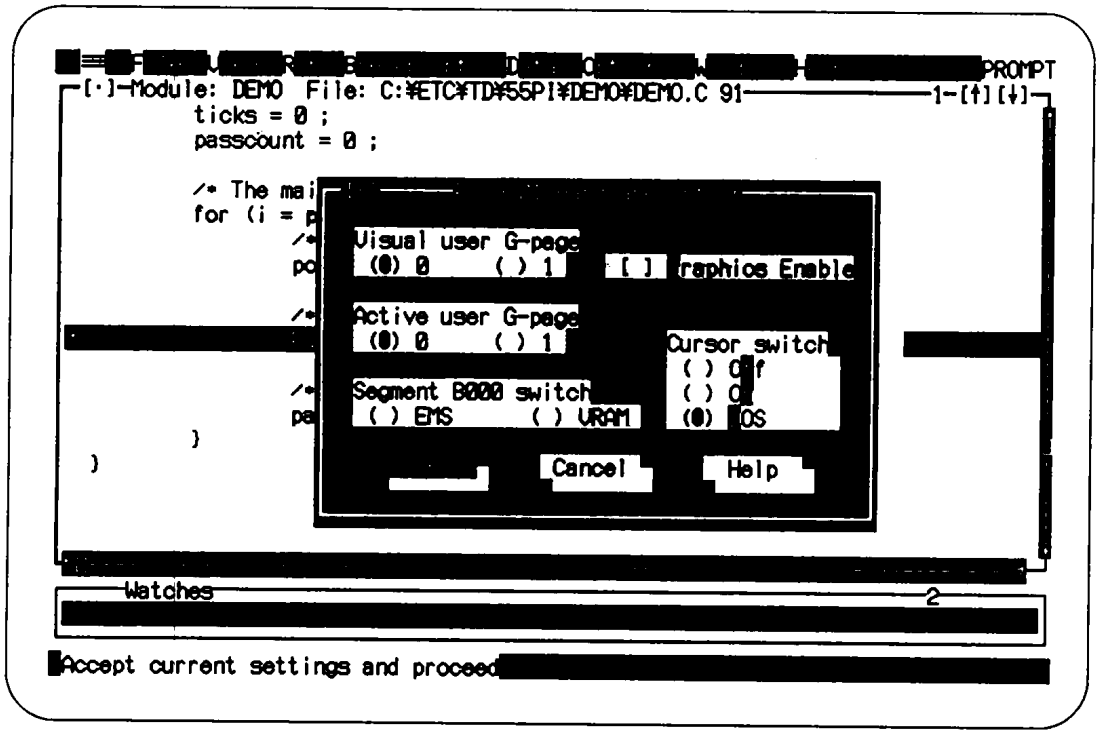
Save optionsコマンドで保存した設定をコンフィギュレーション・ファイルから読み込みます。複数のコンフィギュレーション・ファイルに，異なるマクロやウインドウ配置などを保存しておくことができます。

Save optionsコマンドかTDINSTBXで作成されたファイルを指定してください。

4.7.7 Hardware optionコマンド(PC-9800シリーズ用TDシリーズだけサポート)

このコマンドは，グラフィクス画面やメモリ，カーソルなどのハードウエア資源に関する設定を行います。

図4-9 PC-9801 Hardware optionダイアログ・ボックス（Options | Hardware option）



(1) Visual user G-pageラジオ・ボタン

ユーザ・グラフィクス画面の表示がオンになっているときに，グラフィクスVRAMのバンク0とバンク1のどちらを表示するかを設定します。グラフィック画面のオン／オフは，Graphics Enableチェック・ボックスで行います。

(2) Active user G-pageラジオ・ボタン

TDシリーズからグラフィックVRAMにアクセスするときに，バンク0とバンク1のどちらを対象とするかを指定します。

(3) Graphics Enableチェック・ボックス

このチェック・ボックスでは，ユーザ・グラフィクス画面のオン／オフを指定します。オンにすると，25行表示のディバッガ画面にグラフィクス画面が常に表示されます。オフにすると，グラフィクス画面がオフのままになります。

(4) Cursor switchラジオ・ボタン

ラジオ・ボタンで，カーソルの表示について指定します。

- DOS：DOSが保持しているカーソル状態に従って，表示するかしないかを決めます。
- On　：カーソルを常に表示します。
- Off　：カーソルを常に表示しません。

(5) Segment B000 switchラジオ・ボタン

EMSメモリのセグメントがB000h番地から始まるPC-9801RA/RLなどのホスト・マシンをご使用のときに必要な機能です。

PC-9801RA/RL（および同じ形式のメモリ・ボードを装着しているマシン）では，グラフィックVRAMのセグメントの先頭番地も同じくB000hです。したがって，ディバッガからそのメモリ領域にアクセスするときにEMSとグラフィックVRAMのどちらを有効にするか，このラジオ・ボタンで指定します。

対象とならないハードウエアをご使用の場合，この機能は使用できません。

4.8 TDシリーズの内部からMS-DOSのコマンドを実行する

プログラムをディバグしているときに，別のプログラムやユーティリティが必要になることがあります。このような場合には，File | DOS shellを選択してください。

MS-DOSのコマンド・プロセッサを起動するとき，ディバグしているプログラムは必要に応じてディスクに退避されます。これにより，使用可能なメモリのすべてを使うような大きなプログラムをディバグしている途中でも，MS-DOSのコマンドを実行することができます。だだし，プログラムをディスクに退避したり，ディスクから再ロードするまでに数秒ほど時間がかかります。

MS-DOSのコマンドを実行したあとで再びディバッガに戻るには，EXITと入力して↵を押してください。

注意　MS-DOSのシェルにいる間，TDシリーズより上位のメモリにはTSR(常駐プログラム)をロードしないでください。

4.9 MS-DOSに戻る

GRPH-Xを押すと，ダイアログ・ボックスがアクティブでなければ，ディバグを終了してMS-DOSに戻ります。ダイアログ・ボックスがアクティブのときは，まずESCキーを押してそのダイアログ・ボックスを閉じてください。またFile | Quitを選択しても終了できます。

MS-DOSに戻ると，ディバグしていたプログラムに最初に割り当てられていたメモリは，すべて解放されます。またディバグしていたプログラムが，MS-DOSのメモリ・ブロック・アロケーション・ルーチンを使ってメモリを割り当てていた場合には，そのメモリも解放されます。

〔メ　モ〕

第５章　プログラムの実行

プログラムをディバグするときは，通常，プログラムの調べたい部分を実行して停止させ，その部分が正しく動作するかどうかを調べます。TDシリーズには，次のようにプログラムの実行に関する数多くの方法が用意されています。

- 機械語命令を１命令，またはソース行を１行実行する。
- 関数や手続きの呼び出しをスキップする。
- 連続的にトレース（アニメート）する。
- 現在の手続きから呼び出し側（caller）に戻るまで実行する。
- 指定した位置まで実行する。
- ブレークポイントに到達するまで実行する。
- プログラムを逆向きに実行する。

ディバグ・セッションには，ディバグしているプログラムが実行されている期間と停止している期間があります。プログラムが停止しているとき（ディバッガが動作しているとき）は，Runメニュー・コマンドのどれかを選択するか，そのホット・キーを押すことにより，プログラムを実行することができます。

プログラムの実行は，次の場合に終了し，制御は再びディバッガに戻ります。

- 指定した部分の実行が終了したとき。
- 特別のキー操作で実行を中断させたとき。
- ブレークポイントに到達したとき。

この章では，プログラムの実行から終了まで，次の項目に分けて説明します。

- ディバッガの制御のもとでプログラムの状態を調べる方法
- プログラムの目的の部分を実行するための方法
- プログラムの実行中に実行を中断する方法
- 同じプログラムの再ロード，または別のプログラムのロードにより，ディバグ・セッションを再開する方法

5.1　プログラムの状態を調べる方法

プログラムの状態（ステート）は次の要素で構成されます。

- ●関数のコール・スタック
- ●ソース・コードまたは機械語コードの中のカレント・ロケーションPS：PC
- ●レジスタの値
- ●メモリの内容
- ●プログラムのデータ変数の値
- ●ディバッガがプログラムの実行を停止させた理由

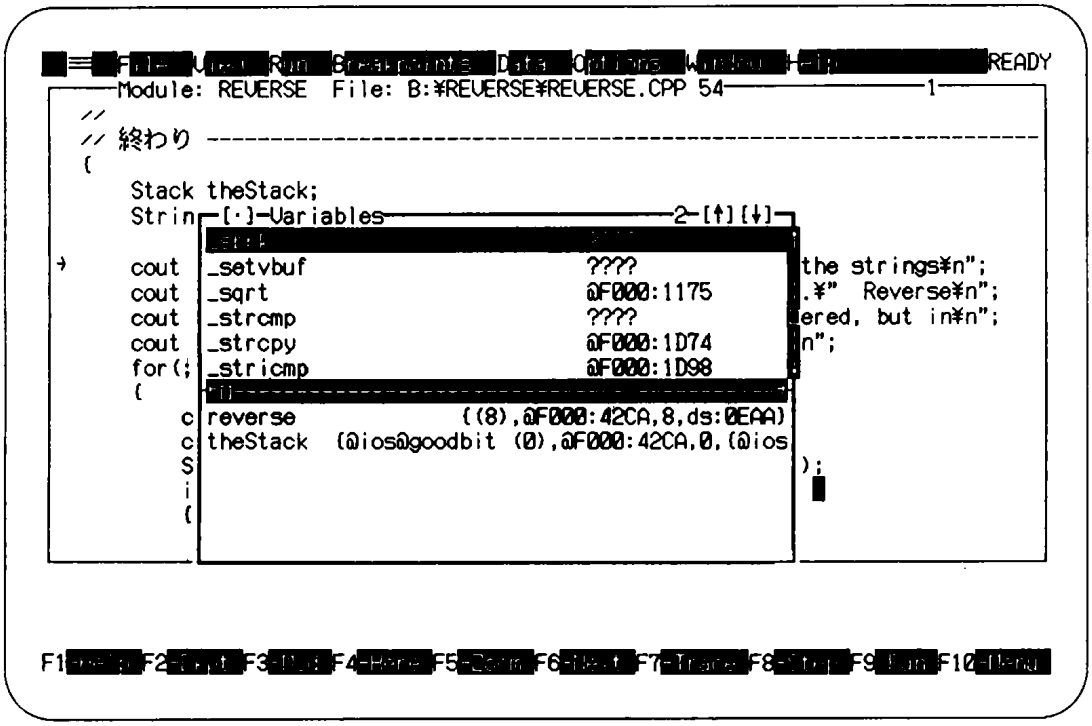
次に，Variablesウインドウ，StackウインドウおよびOriginとGet Infoコマンドの使用方法について説明します。

5.1.1　Variablesウインドウ

Variablesウインドウは，View | Variablesを選択するとオープンします。このウインドウは，プログラムのカレント・ロケーション（次に実行される命令行またはソース行）からアクセスできるすべての変数（名前と値）を表示します。

名前が思い出せない変数を探すときは，このウインドウを使うと便利です。変数が見つかったら，ローカル・メニュー・コマンドを使って，値を調べたり変更したりすることができます。また，現在呼び出されている関数のローカル変数を調べることもできます。

図5－1　Variablesウインドウ（View | Variables）



Variablesウインドウには次の２つのペインがあります。

●グローバル・ペイン（上のペイン）

プログラムのグローバル・シンボルが表示されます。

●スタティック・ペイン（下のペイン）

現在のモジュール（カレント・プログラム・ロケーションPS：PCを含むモジュール）のスタティック・シンボル，および現在の関数のすべてのローカル変数が表示されます。

どちらのペインでも，左側に変数の名前，右側にその値が表示されます。シンボルのデータ型情報が見つからないときは，その値が表示される場所に４個の疑問符（？？？？）が表示されます。

グローバル・ペインの中でGRPH-F10を押すと，グローバル・ペインのローカル・メニューが現われます。CTRLキーによるショート・カットを使用可能に設定しているときは，CTRLキーとローカル・メニュー・コマンドの先頭の文字を一緒に押して，そのコマンドにアクセスすることができます。

Stackウインドウを使うと，現在呼び出されている関数のローカル変数の呼び出しにおける値を調べることができます。その手順を次に示します。

① View | Stackを選択してStackウインドウをオープンする。

② ローカル変数の値を調べたい関数呼び出しにハイライト・バーを移動する。

③ GRPH-F10を押してローカル・メニューをオープンしLocalsコマンドを選択する。

Variablesウインドウのスタティック・ペインにその関数の呼び出しにおけるローカル変数の値が表示されます。

(1) グローバル・ペインのローカル・メニュー

このローカル・メニューには，次の２つのコマンドがあります。

(a) Inspectコマンド

Inspectウインドウをオープンします。

このInspectウインドウは，現在ハイライト表示されている（選択されている）グローバル・シンボルの内容を表示します。調べたい変数が関数名のときは，その関数のソース・コードを表示します。ソース・ファイルが見つからない場合には，CPUウインドウに逆アセンブルされたコードを表示します。

Inspectウインドウは，通常，プログラムのカレント・ロケーション（PS：PC）において有効な変数の値を表示します。ただし，グローバル・ペインのInspectウインドウは，通常の動作とは少し異なります。ハイライト表示されている変数と同じ名前のローカル変数があっても，ローカル変数の値ではなく，グローバル変数の値を表示します。同じ名前がローカル変数としても使用されるグローバル変数の値を調べる場合に便利です。

(b) Changeコマンド

現在ハイライト表示されているグローバル・シンボルの値を，Changeダイアログ・ボックスに入力された値に変更します。必要に応じて，現在選択している言語の代入演算子を使って代入したのと同様なデータの型変換が行われます。

また，Inspectウインドウをオープンして，Changeコマンドを選択せずに新しい値を入力すると，現在ハイライト表示されているシンボルの値を変更できます。

新しい値の入力を始めると，Changeコマンドを選択したときと同じダイアログ・ボックスが現れます。

(2) スタテイック・ペインのローカル・メニュー

GRPH-F10を押すと，スタティック・ペインのローカル・メニューが現われます。CTRLキーによるショート・カットを使用可能に設定しているときは，CTRLキーとコマンドの先頭の文字を一緒に押すことにより，そのコマンドにアクセスすることもできます。

スタティック・ペインには，次の２つのローカル・メニュー・コマンドがあります。

(a) Inspectコマンド

Inspectウインドウをオープンします。

このInspectウインドウは，現在ハイライト表示されているモジュールのローカル・シンボルの内容を表示します。

(b) Changeコマンド

現在ハイライト表示されているスタティック・シンボルおよびローカル・シンボルの値を，Changeダイアログ・ボックスに入力された値に変更します。必要に応じて，現在選択している言語の代入演算子を使って代入したのと同様なデータの型変換が行われます。

またInspectウインドウをオープンし，Changeコマンドを選択せずに新しい値を入力すると，現在ハイライト表示されているシンボルの値を変更できます。

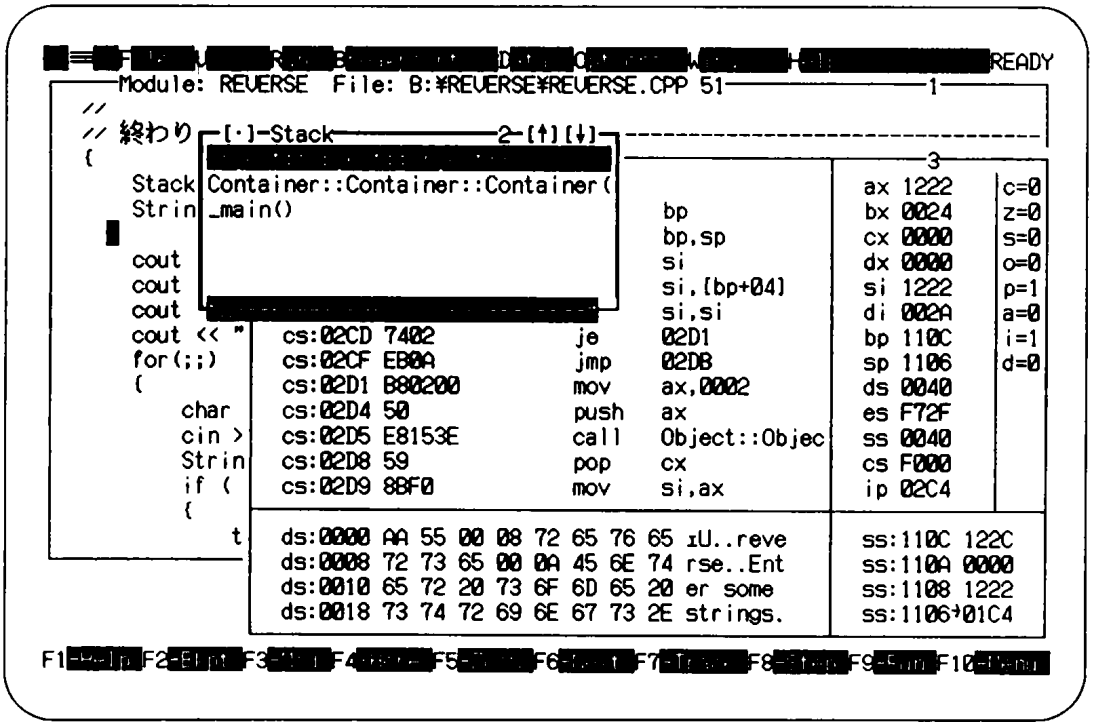
新しい値の入力を始めると，Changeコマンドを選択したときと同じダイアログ・ボックスが現れます。

5.1.2 Stackウインドウ

View | Stackを選択するとStackウインドウがオープンします。このウインドウは，すべてのアクティブな関数および手続きのリストを表示します。

リストには，先頭に最後に呼び出されたルーチン，次にそのルーチンを呼び出したルーチン，その次に２番目のルーチンを呼び出したルーチンという順で，そのプログラムで最初に実行される関数または手続き（Pascalのプログラムではメイン・ボディ，Cのプログラムでは通常main () と呼ばれる関数）までさかのぼって表示されます。各関数または手続きに対して，呼び出し引き数の値が表示されます。

図５－２　Stackウインドウ（View | Stack）



(1) Stackウインドウのローカル・メニュー

Stackウインドウのローカル・メニューには，次の２つのコマンドがあります。

(a) Inspectコマンド

現在ハイライト表示されている関数のアクティブな行を表示するModuleウインドウをオープンします。

ハイライト表示されている関数がリストの最上位の（最後に呼び出された）関数の場合には，プログラムのカレント・ロケーション（PS：PC）を表示します。ハイライト表示されている関数が最後に呼び出された関数でない場合には，その関数が呼び出した関数から戻ったあとに実行される行を表示します。

ハイライト・バーで関数を選択して↵を押すことにより，このコマンドを呼び出すこともできます。

(b) Localsコマンド

現在ハイライト表示されている関数のローカル・シンボルと，現在のモジュールのローカル・シンボルを表示するVariablesウインドウをオープンします。

関数が自分自身を再帰的に呼び出している場合には，その関数が複数個連続して表示されます。ハイライト・バーでそのうちの1つを指定してこのコマンドを選択すると，その関数のローカル変数のその呼び出しにおける値を見ることができます。

5.1.3 Originローカル・メニュー・コマンド

Moduleウインドウのローカル・メニューと CPUウインドウのCodeペインのローカル・メニューには，Originコマンドが入っています。

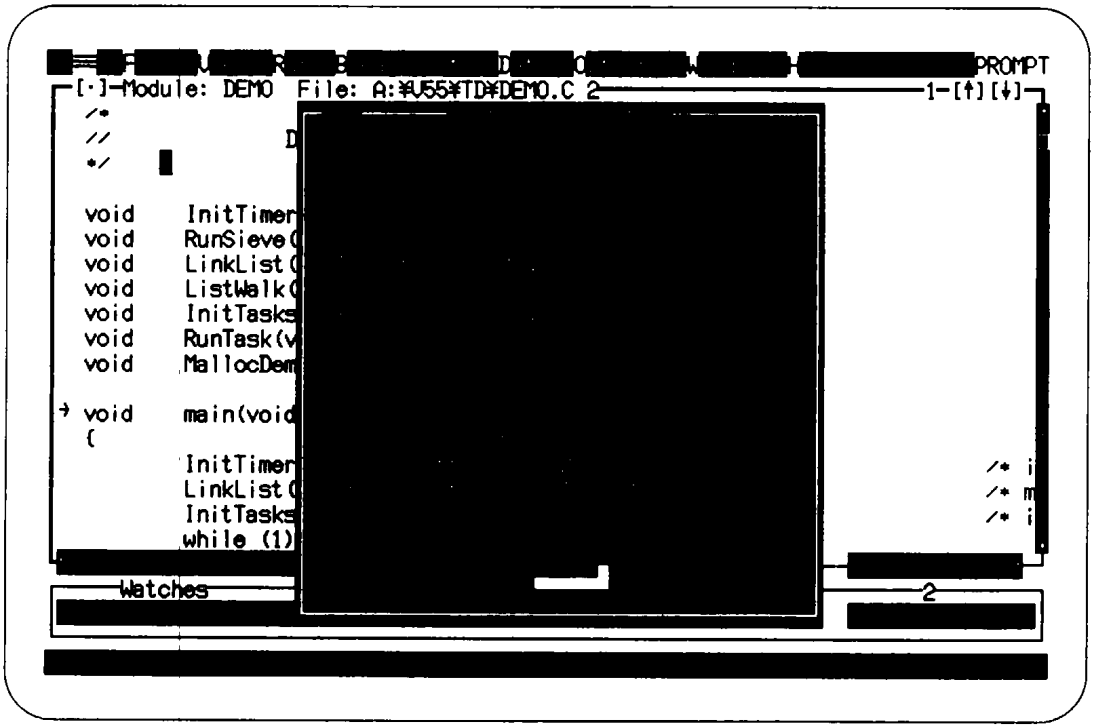
Originコマンドは，カーソルをカレント・コード・セグメント（PS：PC）に移動させます。このコマンドは，プログラムのコードをあちこち見たあと，プログラムが停止した位置に戻りたいときに非常に便利です。

5.1.4 Get infoコマンド

File | Get infoコマンドを選択すると，メモリの使用状況を調べたり，プログラム実行が途中で中断した理由を調べたりすることができます。このコマンドは，テキスト・ボックスに次の情報を表示します（図5-3を参照）。

キーをどれか押すと表示は消えます。

図5-3　Get infoコマンドのテキスト・ボックス（File | Get info）



プログラムが停止した理由を示すメッセージが表示されます。表示されるメッセージは次のどれかです。

(1) Stopped at __

プログラムが，Run | Execute to，Run | Go to cursor，またはRun | Until returnコマンドの完了の結果停止したことを示します。またプログラムがロードされ，コンパイラ・スタートアップ・コードが実行されてソース・コードの先頭に到達したときにも，このメッセージが表示されます。

(2) No program loaded

プログラム名を指定しないでTDシリーズを起動したことを示します。プログラムをロードするか，CPUウインドウのコード・ペインの中のAssembleローカル・メニュー・コマンドを使用して何か命令をアセンブルするまで，何も実行することはできません。

(3) Trace

F7（Run | Trace）で，ソース・コードを1行または機械語命令を1命令実行したことを示します。

(4) Breakpoint at __

ブレークポイントで実行が停止したことを示します。"at" のあとには実行が停止したアドレスが

表示されます。

(5) Terminated, exit code __

プログラムの実行が終了したことを示します。"code"の後ろにはプログラムがMS-DOSに返した数値の終了コードが表示されます。プログラムが値を返さないときは，無意味な値が表示されることもあります。

Run | Program resetでプログラムを再ロードするまで，プログラムを実行できません。

(6) Loaded

コンパイラ・スタートアップ・コードの実行を抑制するオプションを指定して，プログラムをロードしたことを示します。この時点では，プログラムのスタックやセグメント・レジスタを設定する命令も含めて，命令は1つも実行されていません。この段階でプログラムの中のデータを調べても，正しい値を見ていることになりません。

(7) Step

F8（Run | Step over）により，関数呼び出しをスキップしながら，ソース・コードを1行または機械語命令を1命令実行したことを示します。

(8) Interrupt

実行中断キー（通常はCTRL-STOP）を押して実行を停止したことを示します。実行中断キーを押すと，プログラムの実行がただちに停止し，ディバッガに制御が戻ります。

(9) Divide by zero

プログラムがゼロによる除算を実行したことを示します。

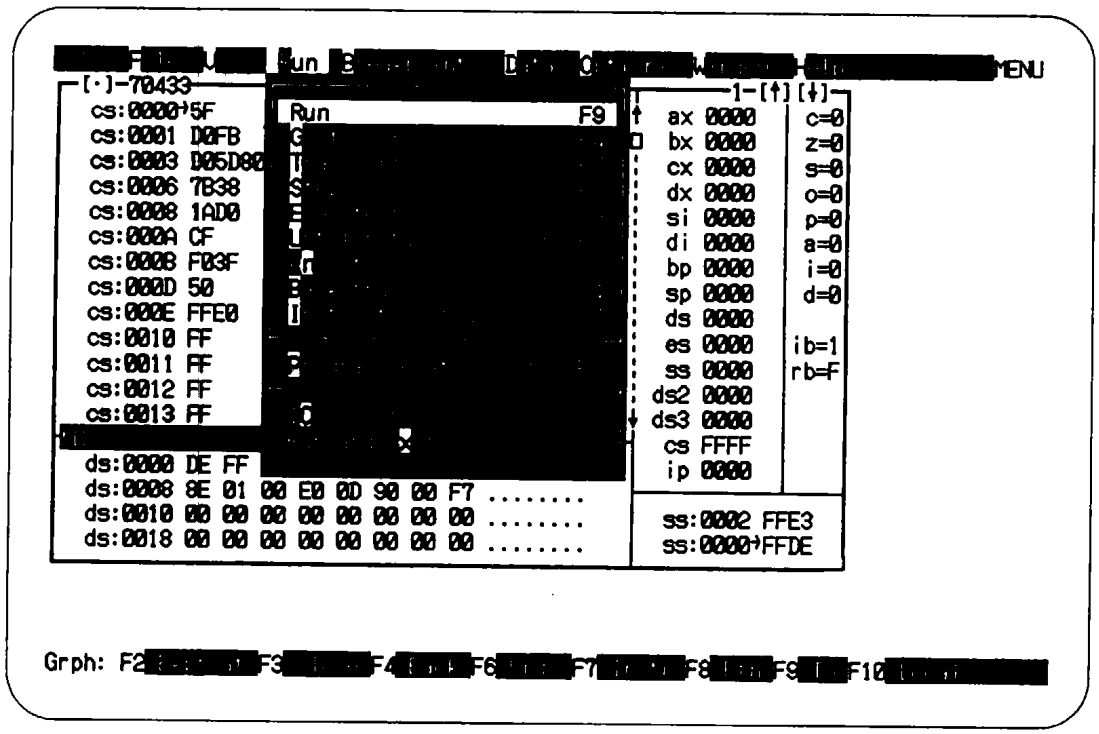
(10) Global breakpoint __ at __

グローバル・ブレークポイントがトリガされたことを示します。ブレークポイント番号とブレークポイントが発生したロケーションが表示されます。

5.2　プログラムの目的の部分を実行するための方法…Runメニュー

Runメニューには，プログラムのいろいろな部分を実行するための多数のオプションがあります。これらのコマンドはよく使用されるので，すべてのコマンドがファンクション・キーにより選択できるようになっています。

図5-4　Runメニュー



5.2.1　Runコマンド

プログラムを自由に実行させます。次のイベントのどれかが発生したときは，ディバッガに制御が戻ります。

- プログラムが終了したとき
- プログラムを停止させるブレークポイントに到達したとき
- CTRL-STOPを押したとき

F9がこのコマンドのホット・キーです。

5.2.2　Go to cursorコマンド

ModuleウインドウまたはCPUウインドウのコード・ペインの中のカーソルが置かれている行まで，プログラムを実行します。現在のウインドウがModuleウインドウの場合には，カーソルを必ずソース・コード行に置いてください。

F4がこのコマンドのホット・キーです。

5.2.3　Trace intoコマンド

ソース・コードを1行または機械語命令を1命令実行します。現在のウインドウがModuleウインドウの場合には，ソース・コードを1行実行します。

現在のウインドウがCPUウインドウの場合には，機械語命令を1命令実行します。現在の行が関数または手続きの呼び出しを含むときは，そのルーチンの中に入っていきます。ただし，現在のウインドウがCPUウインドウの場合には，機械語命令を1命令だけ実行します。

TDシリーズは，オブジェクトのメソッドおよびクラスのメンバを，ほかの関数や手続きと同様に扱います。

F7がこのコマンドのホット・キーです。インストラクションごとのステッピングに関しては，5.2.9 Instruction traceコマンドを参照してください。

5.2.4　Step overコマンド

関数や手続きの呼び出しをスキップしながら，ソース・コードを1行または機械語命令を1命令実行します。現在のウインドウがModuleウインドウの場合には，ソース・コードを1行実行します。現在のウインドウがCPUウインドウの場合には，機械語命令を1命令実行します。

このコマンドでソース・コード行を実行するときは，その行の中で呼び出される関数および手続きはその行の一部として扱われます。そのため，呼び出された関数や手続きの先頭ではなく，現在のルーチンの次の行，または現在のルーチンを呼び出したルーチンで停止します。

CPUウインドウの中では，複数の機械語命令を実行させるいくつかのアセンブリ命令を1命令として扱います。1命令として扱う命令は，次のとおりです。

CALL	サブルーチン・コール（near，far両方）
BRK	インタラプト・コール
DBNZ	CWカウンタによるループ制御
DBNZE	CWカウンタによるループ制御
DBNZNE	CWカウンタによるループ制御

また，REP，REPZ，REPNZのあとにCMPBK，CMPBKB，CMPBKW，LDMB，LDMW，MOVBK，MOVBKB，MOVBKW，CMPM，CMPMB，CMPMW，STM，STMB，STMWが続くときは，これらの命令をステップ実行します。

Run | Step overコマンドは，オブジェクトのメソッドおよびクラスのメンバ関数の呼び出しを単一の文のように扱い，ほかの関数および手続きの呼び出しと同じようにステップ実行します。

F8がこのコマンドのホット・キーです。

5.2.5　Execute to...コマンド

ダイアログ・ボックスに指定されたアドレスに到達するまでプログラムを実行します。ブレーク動作をするブレークポイントに先に到達した場合，および途中で実行を停止させた場合には，指定したアドレスまで実行されません。

F9がこのコマンドのホット・キーです。

5.2.6　Until returnコマンド

現在の関数から呼び出し側のルーチンに戻るまで，プログラムを実行します。このコマンドは，次の場合に便利です。

- 間違えてRun | Step overの代わりにRun | Trace intoを選択し，目的以外の関数や手続きの中に入り込んだ場合。
- 現在の関数が正しく動作することが分かり，残りの部分を1行ずつ実行する必要がないと判断した場合。

GRPH-F8がこのコマンドのホット・キーです。

5.2.7　Animate...コマンド

Trace intoコマンドを繰り返し実行し，1回ごとに画面を更新します。このコマンドでは，プログラムをスローモーションで実行することができます。

プログラムの実行中に，ソース・コードのカレント・ロケーションや変数の値を見ることができます。どれかキーを押すと，コマンドは停止します。

このコマンドを選択すると，1つのTrace intoコマンドの実行が終了してから次のTrace intoコマンドの実行が開始されるまでの遅延時間を入力するよう求められます。遅延時間は1/10秒単位で指定することができ，ディフォールトは3です。

5.2.8　Back traceコマンド

Traceコマンド（Trace intoコマンド，Instruction traceコマンド）でプログラムをトレースしているときは，実行の順番を逆にすることができます。このBack traceコマンドは，バグがあるかもしれないと考える場所を通りすぎてしまい，その点まで戻りたいときに便利です。

Back traceコマンドを用いる前に，次の表に従って，必ずどちらかのコマンドで，バック・トレースが可能となるように設定してください。

コマンド名	場　　所	設定モード
Full historyコマンド	Execution historyウインドウ （View	Execution history） 命令ペインのローカル・メニューの中	Yes
fullhistコマンド	Emulator controlsウインドウの中 （View	ICE	Emulator controls）	Enabled

Back traceコマンドを使うと，1ステップずつ，またはExecution historyウインドウの命令ペイン内のハイライト表示されている場合まで，一度にコードを逆方向にステップします。これにより，プログラムの実行を取り消す（undo）ことができます。

GRPH-F4がこのコマンドのホット・キーです。

注意　バック・トレース（プログラムの実行の逆戻り）には，次の制限があります。

- 割り込み呼び出しは，バック・トレースできません。
- 関数または手続きをステップ実行したときは，呼び出し命令の次の命令までしかバック・トレースできません。
- ポートに関係する命令のバック・トレースは無効です（ポートの読み込み／書き込みは取り消すことができないため）。

5.2.9　Instruction traceコマンド

機械語命令を１命令実行します。このコマンドは，割り込みをトレースしたいときや，Moduleウインドウの中で，そのモジュール内のディバグ情報のない関数や手続き（たとえばライブラリ・ルーチン）をトレースしたいときに使用します。

通常は，該当するソース行がないためCPUウインドウがオープンされます。

Instruction Traceコマンドでは，コマンド実行中に割り込みを受け付けた場合，割り込みルーチンを実行し，最初の命令で止まります。Trace IntoコマンドやStep Overコマンドの場合は，割り込みルーチンを実行し，次の１行で止まります。

このコマンドを用いる前に，**5.2.8　Back traceコマンド**と同じ方法でfullhistコマンドをEnableに設定してください。

5.2.10　Program resetコマンド

プログラムを再起動します。プログラムを実行しすぎたとき（バグがあると考えられる場所を通り越してしまったとき）は，このコマンドを使って最初からやり直してください。CTRL-F2がこのコマンドのショート・カットです。

このコマンドはまったく新しいアプリケーションのリロードはしません。プログラムやデータへ変更を加え，初期状態から再スタートしたい場合は，メニュー・バーからFile | Openを選択しアプリケーションをリロードするか，ICE | resetを選択してください。

5.2.11　Run and exitコマンド

ターゲット・システムで実行しているアプリケーションを開始し，TDシリーズを終了します。

機能的には＜F9＞または＜GRPH-X＞と同等ですが，GRPH-Xはターゲット・システムが停止中のときのみ使用可能です。

5.2.12　ICE resetコマンド

このコマンドは，ターゲット・システムへのアプリケーションのダウン・ロードを含めたディフォールトのIEのステートをリストアするために使用します。このコマンドを選択するとIEのコンフィギュレー

ションへの変更がすべて失われます。

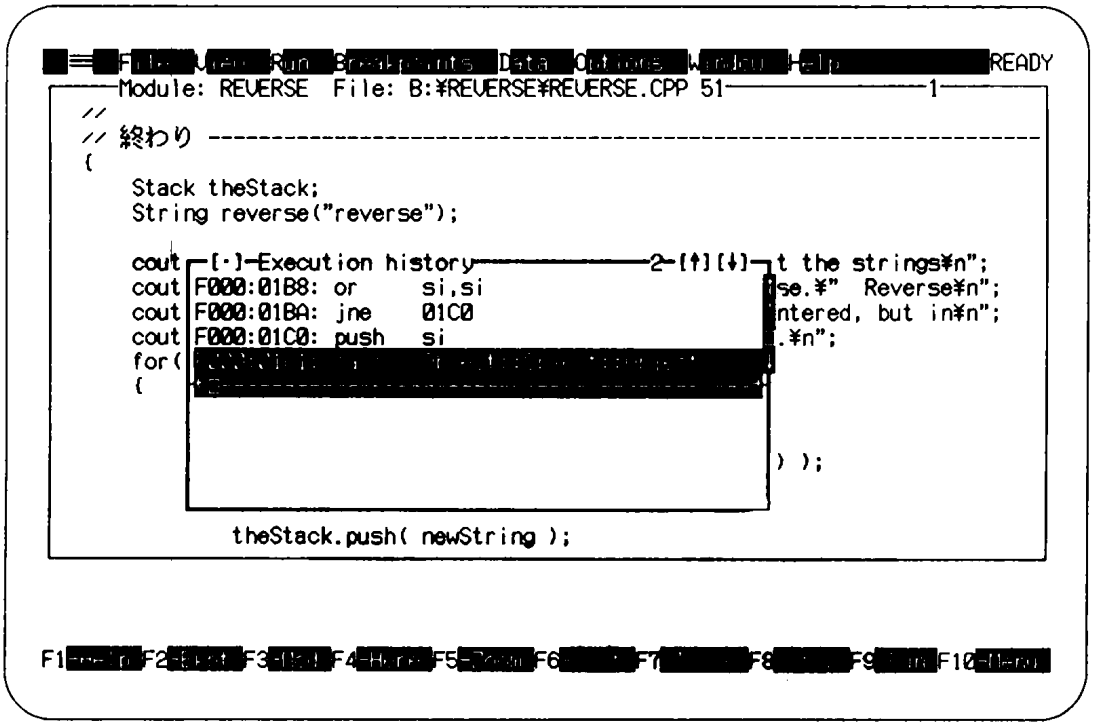
5.3　実行ヒストリ機能…Execution historyウインドウ

TDシリーズには，実行ヒストリと呼ぶ特別の機能があります。この機能は，コードをトレースしているときに，実行された命令を記録します。この機能はトレース中だけに有効です。

この機能を使うと，実行された命令を調べたり，実行された命令を取り消してバグがあると考えられる場所に戻ったりすることができます。EMSメモリがないシステムでは約400の命令，EMSメモリを使用している場合には約3,000の命令を記録することができます。

実行ヒストリを調べるには，View | Execution historyを選択し，Execution historyウインドウをオープンします。

図5-5　Execution historyウインドウ（View | Execution history）



このウインドウには，命令ペイン（上）とキー・ストローク記録ペイン（下）の2つのペインがあります。TDシリーズでは下のキー・ストローク記録ペインは使用していません。

5.3.1　命令ペイン

命令ペインには，すでに実行された命令が表示されます。表示されている命令をハイライト・バーで選択し，調べたり実行を取り消したりすることができます。

実行ヒストリは，Trace intoコマンド（F7）またはInstruction traceコマンド（GRPH-F7）で実行され

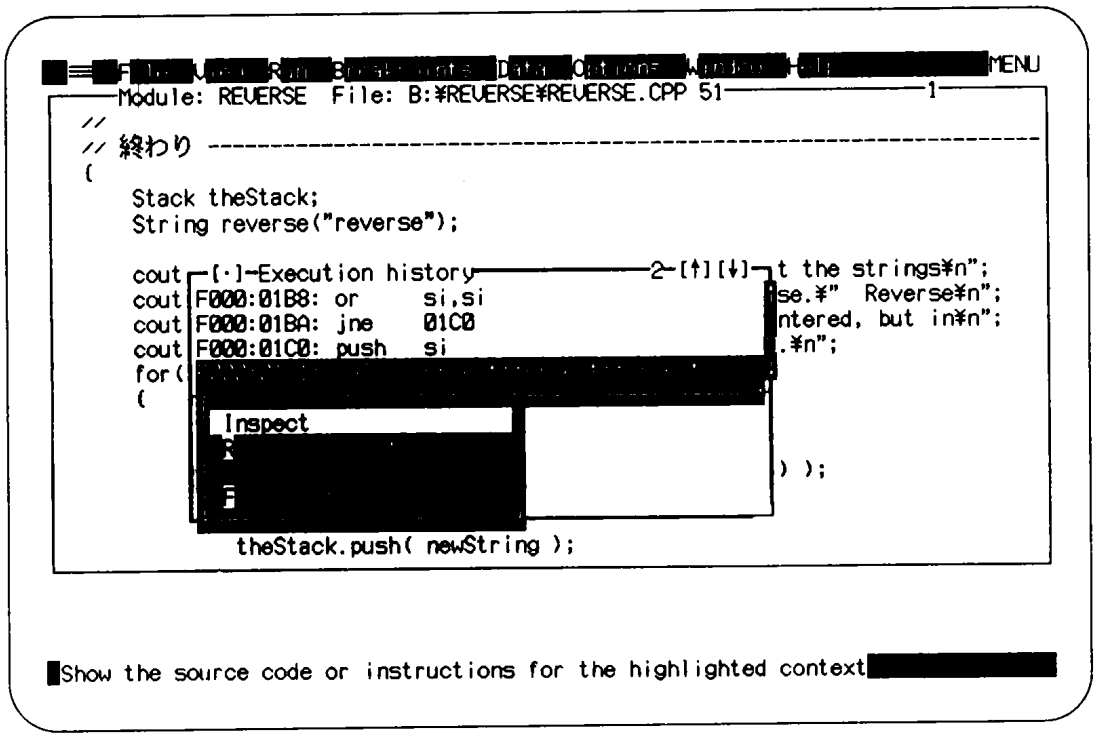
た命令だけを記録します。またStep overコマンドで1命令とみなされる命令に到達するまで，Step overコマンドで実行される命令も記録します。Runコマンドを使用したとき，または割り込みが実行されたときは，実行ヒストリはただちに削除されます。トレースを再開すると，ただちに記録が再開されます。

Step overコマンドで1命令とみなされる命令を次に示します。

CALL　リピート・プリフィクスのあるストリング命令
BRK　リピート・プリフィクスのあるストリング命令
DBNZ　リピート・プリフィクスのあるストリング命令
DBNZE　サブルーチン・コール（near，far両方）
DBNZNE　インタラプト・コール
REP　CXカウンタによるループ制御
REPZ　CXカウンタによるループ制御
REPNZ　CXカウンタによるループ制御

(1) 命令ペインのローカル・メニュー

図5-6　命令ペインのローカル・メニュー



命令ペインのローカル・メニューには，次の3つのコマンドがあります。

(a) Inspectコマンド

このコマンドは，命令ペインでハイライト表示されている命令に対応するプログラムの場所を表示します。

対応するソース行があるときは，Moduleウインドウにその行が表示されます。対応するソース行がないときはCPUウインドウがオープンし，コード・ペイン内にその命令がハイライト表示されます。

(b) Reverse executeコマンド

このコマンドは，命令ペイン内のハイライト表示されているロケーションまで，プログラムの実行を逆戻りさせます。ソース・コード行を選択している場合は，Moduleウインドウに戻ります。そうでない場合にはCPUウインドウがオープンし，コード・ペイン内でその命令がハイライト表示されます。

プログラムのトレース（1行ずつ実行）しなかった部分は，バック・トレースさせることはできません。たとえば，ある場所にブレークポイントを設定してF9を押し，そのブレークポイントのところまで実行したときは，実行ヒストリはすべて放棄されます。このような場合にF9を押す前の場所まで戻りたいときは，Execution historyウインドウのキー・ストローク再生機能を使ってプログラムを再ロードし，その地点まで実行してください。

INT命令を実行すると，それまでの実行ヒストリが放棄されます。GRPH-F7を押して割り込みをトレースした場合を除き，INT命令より前にバック・トレースすることはできません。

注意　次に示す命令は，実行された結果を取り消して元に戻すことはできません（実行ヒストリは放棄しません）。またこれらの命令をバック・トレースしたときは，その副作用に注意してください。

- **IN**
- **INSW**
- **OUT**
- **OUTSB**
- **INSB**
- **OUTSW**

(c) Full historyコマンド

このコマンドはトグルになっています。Yesに設定されていればバック・トレース機能を使用できますが，Noに設定されている場合，バック・トレース機能は使用できません。

5.4　プログラム実行の中断

対話型のプログラムでは，プログラムの実行後，コードの目的とする部分に到達するまでプログラムに応答し，その部分に到達した時点で実行を停止することがあります。このため，プログラム中の目的とする場所に非常に早く到達できます。この方法は，調べようとするコードが，着目している呼び出しの前に

何回か呼び出される場合に特に有効です。

また，コードのどこかに無限ループが存在するなどの理由で，ディバッガに制御が戻らなくなり，プログラムの実行を停止させたいことがあります。このような場合は，CTRL-STOPキーを押します。ほとんどの場合，このCTRL-STOPキーでプログラムの実行を停止し，ディバッガに制御を戻すことができます。

備考　このキーは，押すとすぐに機能します。

5.5　ディバグ・セッションの再開

TDシリーズには，ディバグ・セッションをできるだけ簡単に再開するために，いくつかの機能が用意されています。

5.5.1　プログラムの再開始

Run | Program resetを選択するとプログラムを再開始します。

Program Resetはブレークポイントとウォッチポイントはそのままにし，プログラムを再ロードしません。

5.5.2　プログラムの再ロード

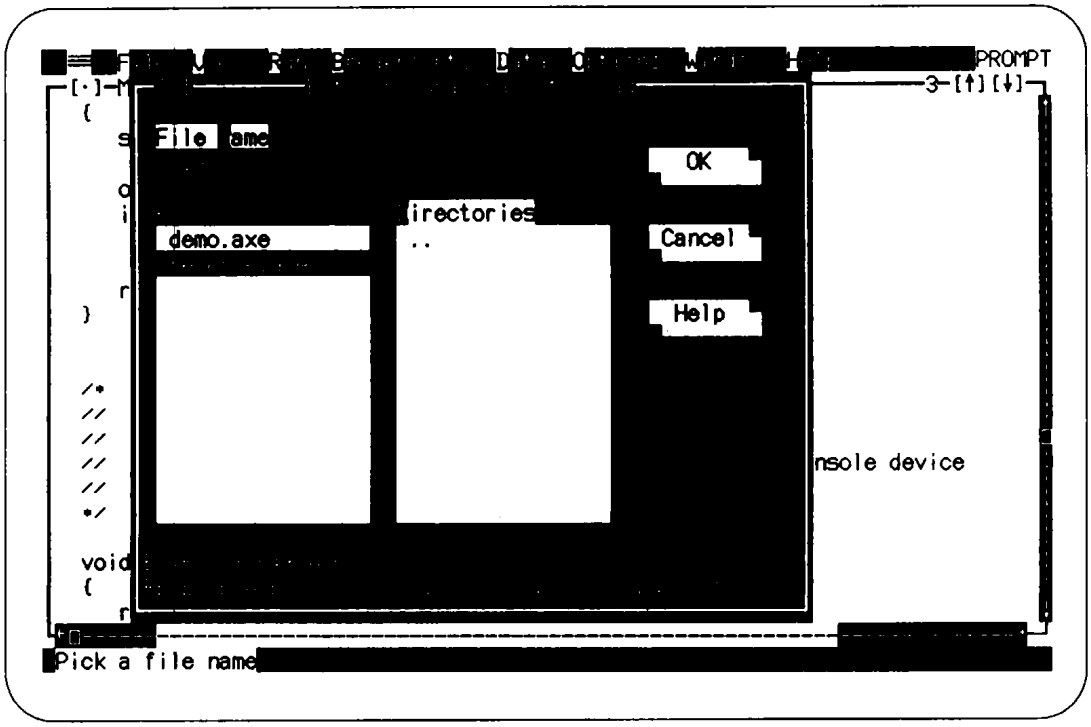
ディバグしているプログラムを再ロードするには，File | Openを選択します。

このコマンドは，ターゲット・システムへアプリケーションのまったく新しいコピーをダウン・ロードし，設定されているブレークポイントやウォッチポイントをクリアします。

5.5.3　ディバグのために新しいファイルをオープンする

File | Openを選択し，Load Programダイアログ・ボックスをオープンすると，ディバグするための新しいプログラムをロードすることができます。

図5-7　Load Programダイアログ・ボックス（File | Open）



ここで，File name入力ボックスにファイル名（拡張子は.AXEまたは.HEX）を入力することもできます。↵を押して，カレント・ディレクトリのすべての.AXEファイルを表示するFilesリスト・ボックスをアクティブにすることもできます。

Filesリスト・ボックスをアクティブにしたときは，ロードしたいファイルにハイライト・バーを移動して↵を押します。ハイライト・バーで選択する代わりにロードしたいファイルの名前を入力すると，入力した文字（1文字以上）で始まるファイル名にハイライト・バーが移動します（インクリメンタル・マッチ）。ハイラト・バーが目的のファイルに移動したら↵を押します。

注意　.HEXファイルはソース・ディバグ，シンボリック・ディバグをすることができません。

〔メ　モ〕

第６章　データの調査，変更

TDシリーズは，プログラム・データを調べる場合，次のような独自の方法を備えています。したがって，修正する箇所が一目で分かります。

- Inspectウインドウを使うと，プログラム・データをソース・ファイル内に記述されているとおりに見ることができます。ポインタが指し示す先をたどったり，配列をスクロールさせて内容を調べることができます。また，構造，レコード，共用体を，プログラマが書き込んだとおりの形式で調べることができます。
- 変数や式をWatchesウインドウ内に入れてプログラムを実行するときに，その変数や式を見ることができます。
- Evaluate/modifyダイアログ・ボックスは，いろいろな変数の値を表示しますが，新しい値を代入することもできます。

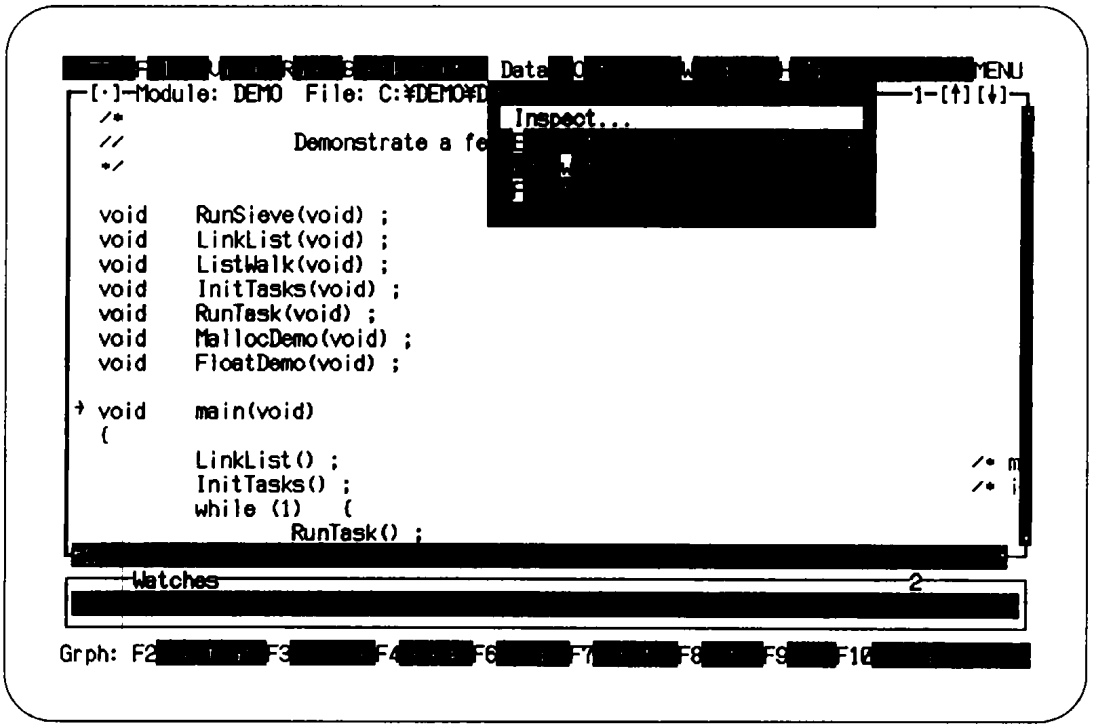
この章では，プログラム内の変数を調査，変更する方法について，次のことを説明します。

- Dataメニューとそのオプション
- ソース・モジュール内でデータ項目を直接ポイントする方法
- Watchesウインドウ
- 各データ型のInspectウインドウ内での表示形式

6.1　データを調査，変更する方法の選択…Dataメニュー

Dataメニューにより，プログラム・データの調査，変更する方法を選択することができます。ユーザは，式を評価し，変数の値を変更し，Inspectウインドウをオープンすることで，変数の内容を表示できます。

図６－１　Dataメニュー



6.1.1　Inspect...コマンド

プロンプトが表示され，点検したいデータを参照するための変数を入力するよう求められます。次に，プログラム内の変数または式の値を表示するためのInspectウインドウが開きます。変数名を単独で入力することも，複雑な式を入力することもできます。

このコマンドを選択したときに，カーソルがテキスト・ペイン内の変数上にあると，カーソル位置にある変数(もしあれば)がダイアログ・ボックス内に自動的に表示されます。INSキーを使ってテキスト・ペイン内の式を選択すると，その式がダイアログ・ボックス内に表示されます。

構造体の配列やリンクされた項目リストなど，複雑なデータ構造を調べたい場合は，それに適したInspectウインドウが表示されます。Inspectウインドウ内では，項目を点検することができます。このため，Moduleウインドウ内でソース・コードをスクロールするのと同じように，簡単にプログラムのデータ項目をたどっていくことができます。

Inspectウインドウの動作については，**6.4　Inspectウインドウ**を参照してください。

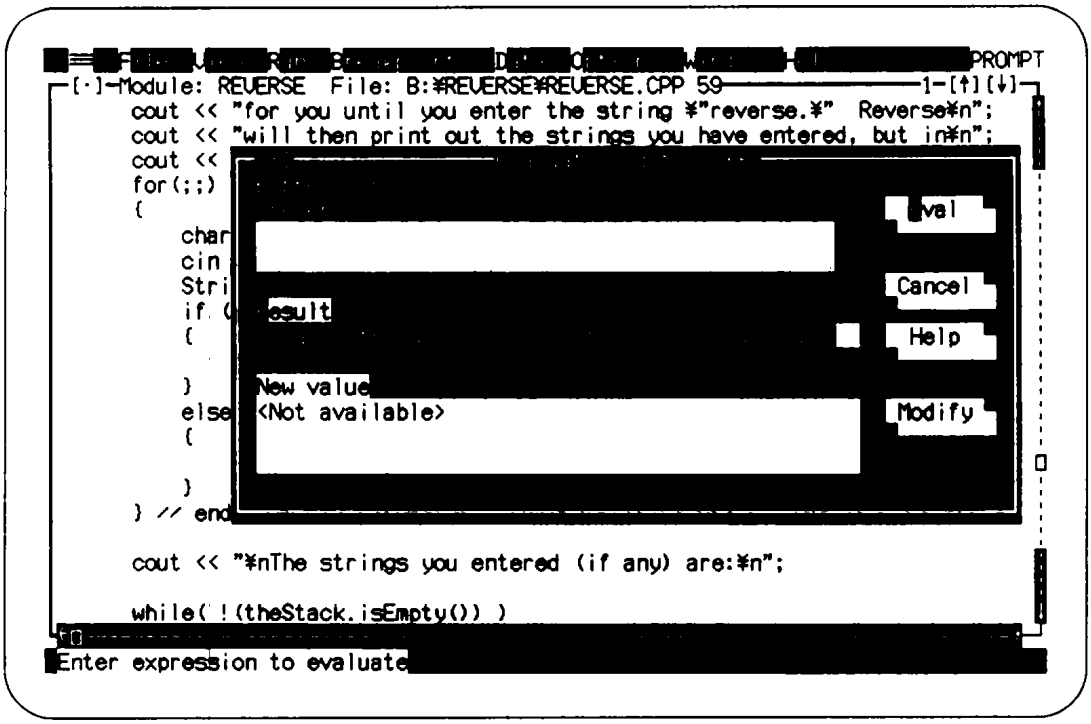
6.1.2　Evaluate/modify...コマンド

Evaluate/modifyダイアログ・ボックス（図６－２を参照）が現われ，評価する式を入力するよう求めるプロンプトが表示されます。式を入力してEvalボタンを選択すると，コンパイラがコンパイル中に行うのと同じように式が評価されます。

このコマンドを選択したとき，カーソルがテキスト・ペイン内にあると，カーソル位置にある変数(もしあれば）がダイアログ・ボックス内に自動的に表示されます。INSキーを使って式を選択すると，マー

クされたその式がダイアログ・ボックスに表示されます。

図6-2　Evaluate/modifyダイアログ・ボックス（Data | Evaluate/modify...）



監視したい式のあとには，書式制御文字列が付けられます。

TDシリーズは，結果をそのデータ型に適した書式で表示します。結果を別の書式で表示するには，式のあとに分離文字カンマ（,）と書式制御文字列を入力します。これは，監視したいデータを，そのデータ型に対して設定されたディフォールトの表示書式と異なる書式で表示させたい場合に便利です。

（1）ダイアログ・ボックスの構成

ダイアログ・ボックスには，次の3つのフィールドがあります。

- ●一番上のフィールド…評価したい式は，ここに入力します（Evaluate入力ボックス）。これには，ほかの入力ボックスとまったく同じヒストリ・リストが付いています。
- ●中間のフィールド　…式の評価結果が表示されます（Result入力ボックス）。
- ●一番下のフィールド…式に新しい値を入力する入力ボックスです（New value入力ボックス）。式が修正できない場合は，このボックスに〈not available〉というメッセージが表示され，カーソルをこのボックスに移動することはできません。

New value入力ボックスに値を入力し，Modifyボタンを選択すると，Evaluate入力ボックス中の変数の値が変更されます。あるボックスから別のボックスに移動するには，ほかのダイアログ・ボックス内の場合と同様に，TABキーまたはSHIFT-TABキーを使います。ダイアログ・ボックスを削除するには，カーソルが入力ボックス内にあるときにESCキーを押すか，またはマウスを右クリックします。

データ文字列が長すぎて，Result入力ボックス内で表示しきれない場合は，そのデータ文字列の末尾に矢印（→）が表示されます。その場合は，右にスクロールすると残りの文字列が表示されます。

(2) C++およびオブジェクト指向のPascalの場合

C++またはオブジェクト指向のPascalプログラムをディバグする場合は，Evaluate/modifyダイアログ・ボックスを使うと，あるオブジェクトのインスタンス・フィールドまたはあるクラスのインスタンス・メンバを表示することもできます。レコードの評価に使えるインスタンスと一緒に書式指定子を使えます。

あるメソッドまたはメンバ関数内でトレースしていくときに，TDシリーズはSelf/thisパラメータのスコープとそれらのパラメータが存在するかどうかを知っています。ユーザは，Self/thisを評価し，書式指定子と修飾子を使ってSelf/thisを追跡できます。

また，Evaluate/modifyダイアログ・ボックス内からあるメソッドまたはメンバ関数を呼び出せます。インスタンス名の次にピリオドを入力し，次にメソッドまたはメンバ関数名を入力して，さらに実際のパラメータ（パラメータがない場合は空の丸カッコ）を入力します。

(3) C言語を使用する場合の注意

C言語には，"副作用を伴う式"と呼ばれる機能があります。この場合，副作用とは，計算式の途中で変数の値が変化することを意味します。

副作用は，変数の値またはメモリ領域を変更する，手早く簡単な方法です。たとえば，countという名の変数の値に1を加えるには，Cの式count++を評価します。

副作用を伴う式は，1つまたは複数の変数やメモリ領域が評価されるときに，その値を変化させます。たとえば，Cのインクリメント演算子（++），デクリメント演算子（--）および代入演算子（=，+=など）にはこの効果があります。Cの式内にあるプログラムの関数（たとえばmyfunc(2)）を実行する場合は，関数が予想外の副作用を引き起こす可能性があることに注意してください。

どの変数の値を修正するつもりもないが，自分のプログラムの変数が入っている式を評価したい場合は，副作用のある演算子を使わないでください。

また，プログラム変数の代わりにオペランドとして数を入力することで，Evaluate/modifyダイアログ・ボックスを簡単な計算機として使うこともできます。

6.1.3 Add watch...コマンド

監視したい式を入力するように求めるプロンプトが表示されます。次にユーザが↵を押すかOKボタ

ンをクリックすると，Watchesウインドウ内に表示される変数リストに，式またはプログラム変数が置かれます。

このコマンドを選択したときにカーソルがテキスト・ペイン内にあると，カーソル位置にある変数(もしあれば）がダイアログ・ボックス内に自動的に表示されます。

INSキーを使ってある式を選択すると，選択したその式がダイアログ・ボックス内に表示されます。

6.1.4　Function returnコマンド

現在の関数が返そうとしている値を表示します。このコマンドは，現在の関数が呼び出し側に戻ろうとしている場合にのみ使えます。

戻り値はInspectウインドウに表示されるため，戻り値が複合データ項目のポインタであっても簡単に調べることができます。

このコマンドを使うと，CPUレジスタに入っている戻り値を調べるときCPUウインドウに切り替える必要がなくなります。また，このコマンドは返されるデータ型を知っており，データを適切な書式に整えるので，16進ダンプよりもはるかに使いやすくなります。

6.2　ソース・ファイル内のデータ項目をポイントする

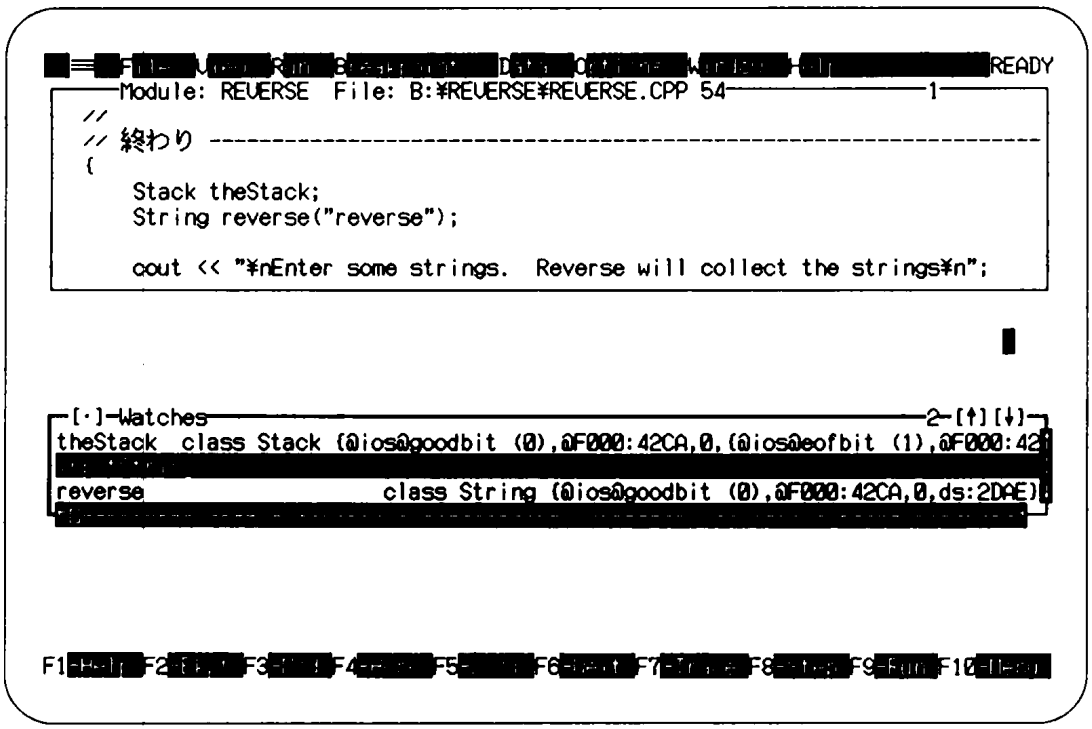
TDシリーズは，調べたいプログラム変数名をそのつど入力しなくてもすむような強力な機能を備えています。どのModuleウインドウにいるときでも，変数名にカーソルを置きローカル・メニューのInspectコマンドを呼び出せば，その変数の値を示すInspectウインドウをオープンすることができます。また，Inspectコマンドを選択する前に，INSキーを押しカーソル・キーを使って，調べたい式または変数をハイライト表示させて選択できます。

6.3　Watchesウインドウ

View | Watchesを選択して，Watchesウインドウをオープンします。

Watchesウインドウには，値を監視したいプログラム内の変数と式が表示されます。簡単な変数(整数など）や複雑なデータ項目（配列など）の値を監視できます。また，x＊y＋4のように，メモリ位置を直接示さない計算式の値も監視できます。

図6-3　Watchesウインドウ（View | Watches）



各項目の左側には変数名または式が表示され，右側にはそのデータ型と値が表示されます。配列や構造体などの複雑な値は，Cプログラムの場合は中カッコ {} の中に，Pascalプログラムの場合は丸カッコ () の中に表示されます。名前または式全体を表示する余白がない場合は，表示できない部分が省略されます。

C言語では，その関数が呼び出されたときに初めて値の確定する変数があります。これは，スタック上に生成されるローカル変数です。TDシリーズでは，このような変数に対して，変数が有効になる前から監視したい変数または式として指定することができます。ただし，**変数名を誤って指定した場合，エラーは検出されませんので注意してください。**

スコープ・オーバライド（**第9章　式参照**）を使わないかぎりは，プログラムが停止する現在位置のスコープ内で，Watchesウインドウ内の式が評価されます。したがって，Watchesウインドウ内の式は，プログラムが停止している位置にその式が現われるのと同様に評価されます。監視しようとする式にカレント・スコープからアクセスできない変数名が含まれている場合，たとえば別のモジュールからアクセスできない場合には，式の値は定義されず，4つの疑問符（？？？？）が表示されます。

オブジェクト・メソッドの内部でトレースする場合は，WatchesウインドウにSelf/thisパラメータを追加できます。

6.3.1　Watchesウインドウのローカル・メニュー

すべてのローカル・メニューと同様に，GRPH-F10キーを押すとWatchesウインドウのローカル・メニューが表示されます。CTRLキーによるショート・カットが使える場合は，CTRLキーと希望するコマンドの先頭文字を一緒に押すと，そのコマンドにアクセスできます。

(1) Watch...コマンド

Watchesウインドウに追加する変数名または式を入力するよう求めるプロンプトが表示されます。リストの先頭に加えられます。

(2) Edit...コマンド

Watchesウインドウ内の式を編集できるダイアログ・ボックスをオープンします。

ボックス内に表示された監視する式を変更することも，新しい式を入力することもできます。

変更したい監視する式の上にハイライト・バーを置いてから ↵ を押して，このコマンドを起動することもできます。編集した式をWatchesウインドウに入力するには，↵ を押すかOKボタンを選択します。

(3) Removeコマンド

現在選択されている項目をWatchesウインドウから削除します。

(4) Delete allコマンド

すべての項目をWatchesウインドウから削除します。

このコマンドは，プログラムのある領域から別の領域に移動するときに，それまで監視していた変数が不要になったときに便利です（さらに変数を入力するにはWatchコマンドを使います)。

(5) Inspectコマンド

現在，Watchesウインドウ内でハイライト表示されている項目の値を表示するには，Inspectウインドウをオープンします。

その項目が複雑なもの(配列，レコード，または構造体)であれば，このコマンドを使って，Watchesウインドウ内に表示されている要素だけでなく，すべての要素を見ることができます。

(6) Changeコマンド

Watchesウインドウ内で現在ハイライト表示されている項目の値を，ダイアログ・ボックスに入力された値に変更します。

現在使っている言語で可能であれば，適切な代入演算子（＝または：＝）を使って変数を変更したかのように，必要な型変換が実行されます。

6.4 Inspectウインドウ

Inspectウインドウは，ユーザが点検しているデータの型に応じて，プログラム・データを適切に表示します。スカラ（charまたはintなど)，ポインタ（Cではchar ＊，Pascalでは^)，配列（long x [4]，array [1..10]of word)，関数，構造体，レコード，共用体，および集合に対して，それぞれ異なる動作をします。

Inspectウインドウは，点検されているデータ・オブジェクトを構成する項目をリストします。ウインドウのタイトルは，点検されるデータの型と名前（もしあれば）を示します。

ウインドウ内の第1の項目は，調査しているデータのアドレスをセグメント：オフセットの形式で表します。ただし，このデータがコンパイラの最適化によりレジスタ上に割り当てられていたり，定数の場合には表示されません。

Inspectウインドウの内容をバイト・データの並びとして見るには，Inspectウインドウ内でView | Dumpコマンドを選択してください。Dumpウインドウは，Inspectウインドウ内に表示されたデータにカーソルが置かれた状態で表示されます。Window | Closeコマンド(GRPH-F3)を使うか，マウスでクローズ・ボックスをクリックして，このウインドウをクローズすればInspectウインドウに戻ることができます。

6.4.1　Inspectウインドウのタイプ

この項では，TDシリーズがサポートする各言語，すなわち，C, Pascal, およびアセンブラに対して表示されるさまざまなInspectウインドウについて説明します。どのプログラム言語を使用するかによって，Inspectウインドウに表示される情報の書式が決まります。データ項目とその値は，常に，ソース・ファイル内で宣言されたのと同じ書式で表示されます。

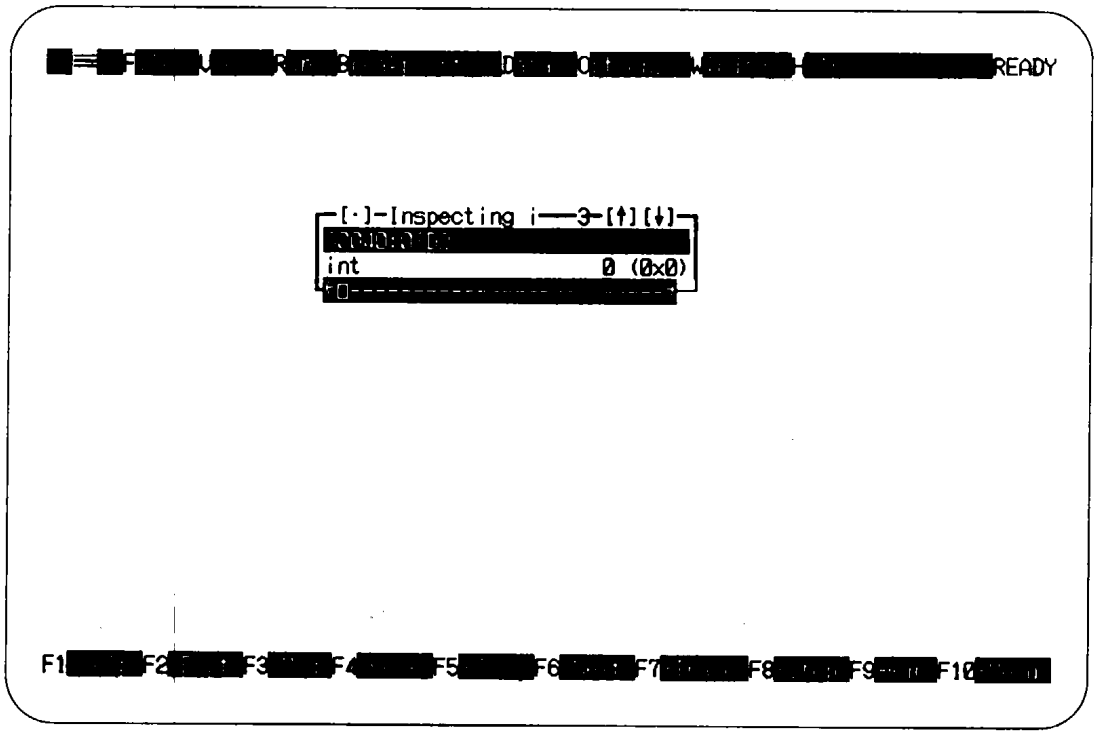
いろいろなInspectウインドウがありますが，特別な操作を行う必要はありません。点検しているデータに応じて，自動的に正しいウインドウが表示されます。

(1) スカラ

スカラのInspectウインドウは，単純なデータ項目の値を表示します。これらのInspectウインドウでは，1行目のあとに，変数のアドレスを示す1行の情報しか表示されません。次の行の左側には，スカラ変数の型（char, unsined longなど）が表示され，右側にはその現在の値が表示されます。この値は，10進数，16進数，またはその両方で表示できます。通常は，最初に10進数で表示され，次に（Cの16進プリフィクス0xを使って）中カッコで囲まれた16進数が表示されます。変数の表示方法はTDINSTBXを使って変更することができます。

表示される変数の型がcharであれば，その文字も表示されます。現在の値が印字可能文字に相当しない場合は，円記号のあとに16進値をつけて，文字の値を表示します。この文字の値は，10進値または16進値の前に表示されます。

図6－4　スカラのInspectウインドウ（Data | Inspect）



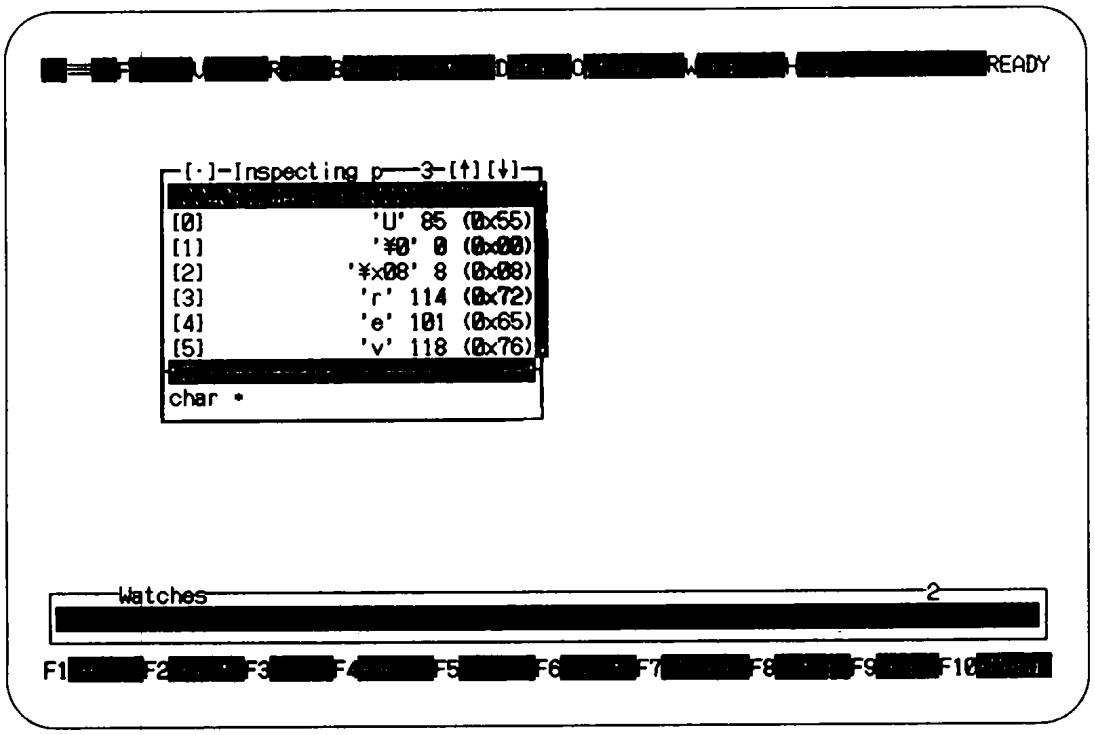
(2) ポインタ

ポインタのInspectウインドウは，ほかのデータ項目を指し示すデータ項目の値を表示します。これらのInspectウインドウでは，通常，1行目に変数のアドレスが表示され，次にそのアドレスで示されるデータに関する1行の情報が表示されます。その左側には，配列の最初のメンバを示す [0] が表示され，右側には，それが指し示す項目の値が表示されます。値が構造体や配列のような複雑なデータ項目の場合には，できるだけ多くの値が中カッコ {} で囲まれて表示されます。

ポインタの型がcharで，ヌルで終了する文字列を指し示すときは，文字配列内の各項目の値を示す多数の情報が表示されます。各行の左側には，配列のインデクス（[1]，[2] など）が表示され，右側には，スカラのInspectウインドウ内に表示されるとおりに値が表示されます。この場合，文字列全体が，ポインタ変数のアドレスと，それが指し示す文字列のアドレスとともに1行目に表示されます。

また，Inspectウインドウをオープンして Rangeローカル・メニュー・コマンドを使うと，複数の行を表示できます。

図6-5　ポインタのInspectウインドウ（Data | Inspect）



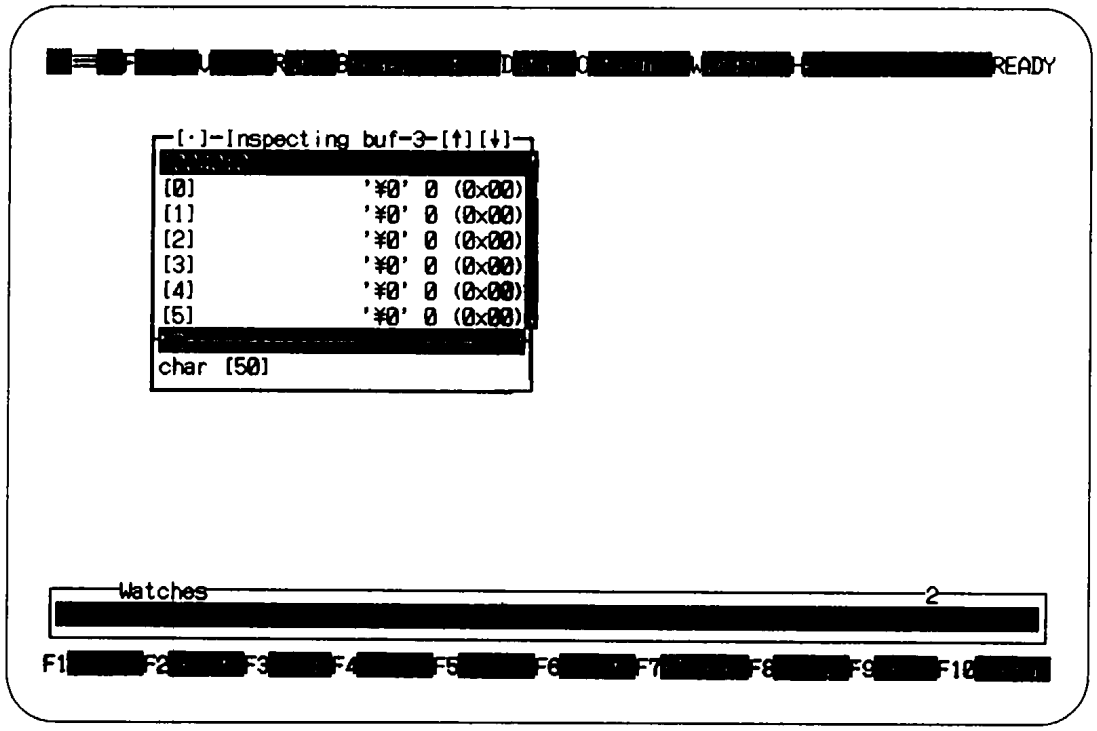
ポインタのInspectウインドウの下部には，ポインタが指し示すデータ型を示すペインもあります。

(3) 配　列

配列のInspectウインドウは，配列の要素ごとに1行ずつ表示されます。各行の左側には，項目の配列インデクスが表示されます。右側には，項目の値が表示されます。値が構造体や配列などの複雑なデータ項目であれば，できるだけ多くの値が表示されます。

Rangeローカル・メニュー・コマンドを使うと，配列のどの部分でも調べることができます。このコマンドは，配列に多数の要素があって，配列の中間の値を見たいときに便利です。

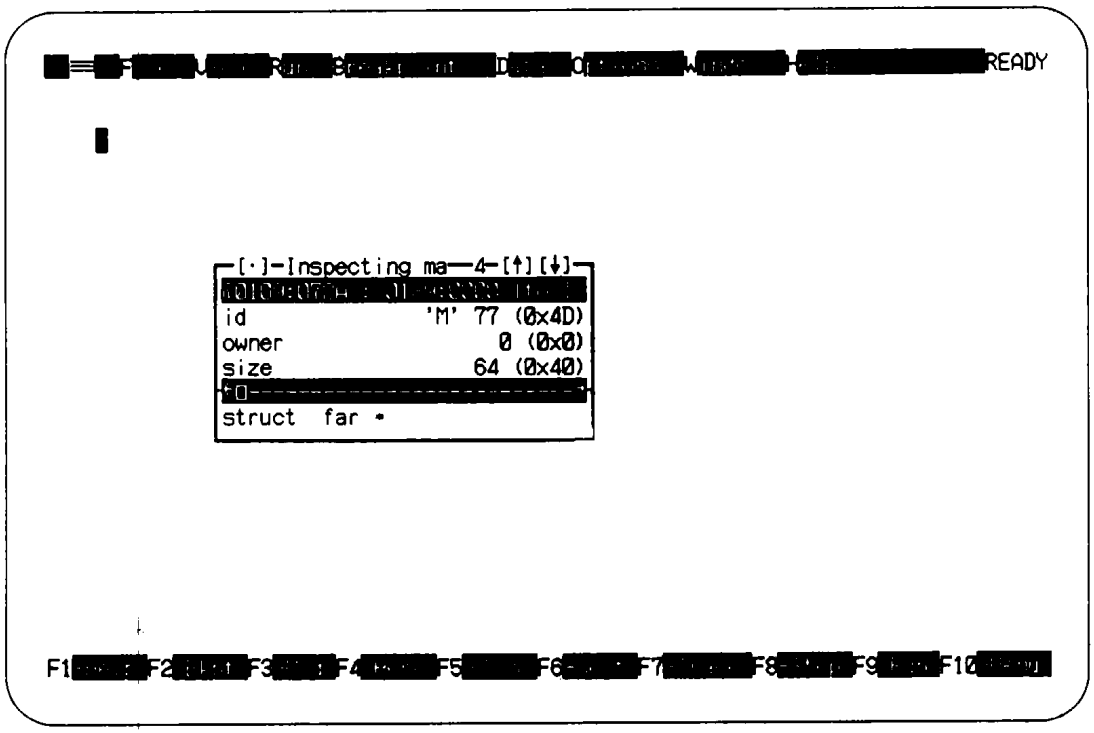
図６－６　配列のInspectウインドウ（Data | Inspect）



(4) 構造体と共用体

構造体と共用体のInspectウインドウには，メンバの値を示すペインの下にもう１つペインがあります。下のペインには，上のペイン内でハイライト表示されたメンバのデータ型が表示されます。

図6－7　構造体と共用体のInspectウインドウ（Data | Inspect）

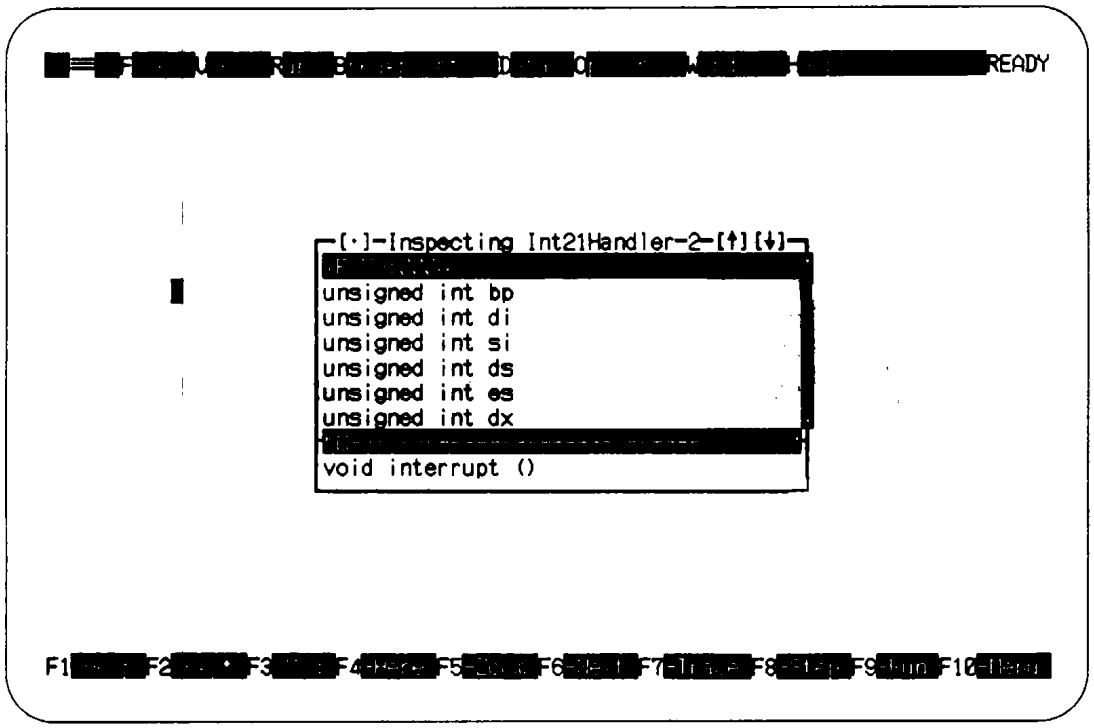


構造体と共用体は，Inspectウインドウの中では同じように表示されます。Inspectウインドウの下のペインは，構造体と共用体のどちらが表示されているのかを示します。これらのInspectウインドウでは，アドレスの後ろに，構造体または共用体内のメンバと同じ数のデータ項目が表示されます。各項目の左側にはメンバ名が，右側にはその値が，それぞれCのデータ型に適した書式で表示されます。

(5) 関　数

関数のInspectウインドウでは，ウインドウ上部のメモリ・アドレスの下に，関数が呼び出されたときの各パラメータが表示されます。

図6-8　関数のInspectウインドウ（Data | Inspect）



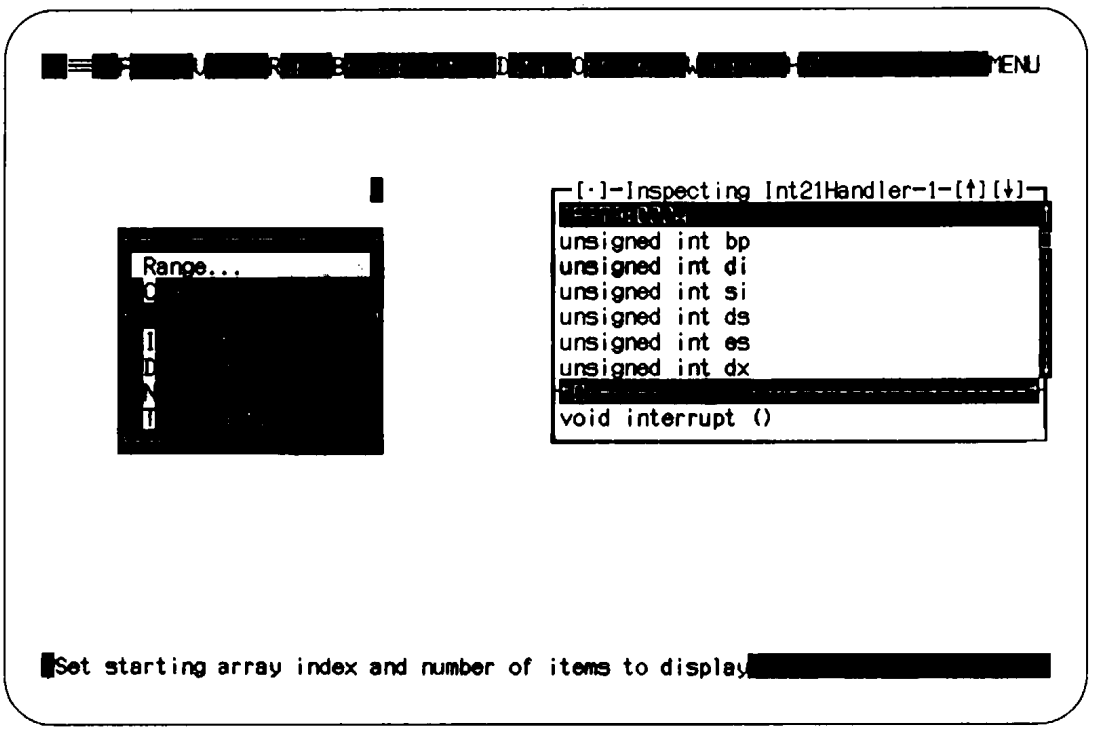
また，呼び出し時のパラメータ，戻り値のデータ型，および関数の呼び出し規則に関する情報も表示されます。下のペインには関数が返すデータ型が表示されます。

6.4.2　Inspectウインドウのローカル・メニュー

Inspectウインドウに最も影響を与えているのが，ローカル・メニューのコマンドです。たとえば，Inspectローカル・メニュー・コマンドを選択し，Inspectウインドウをもう1つオープンして，データ項目をたどることができます。ローカル・メニューのほかのコマンドを使うと，ある範囲の値を調べたり，新しい変数を調べたりすることができます。

Inspectウインドウのローカル・メニューを表示するにはGRPH-F10を押します。CTRLキー・ショート・カットが使用可能であれば，CTRLキーとコマンドの先頭文字を一緒に押してそのコマンドにアクセスします。

図６－９　Inspectウインドウのローカル・メニュー



(1) Range...コマンド

表示させたい最初の要素と要素数を設定します。このコマンドは，ある配列を調べるときに，その配列中の一部の要素だけを見たい場合に使います。

長い配列があって，中央付近にある少数の要素を見たい場合は，このコマンドを使います。調べたい配列インデクスのところでInspectウインドウをオープンします。

(2) Change...コマンド

現在ハイライト表示されている項目の値を，ダイアログ・ボックスで入力された値に変更します。現在の言語で可能であれば，対応する代入演算子を使って変数を変更したように，必要な型変換が行われます。

(3) Inspectコマンド

現在ハイライト表示されている項目の内容を示す，新しいInspectウインドウをオープンします。このコマンドは，Inspectウインドウ内の項目に構造体や配列のように複数の項目が含まれており，それらの各項目を見たいときに便利です。

現在のInspectウインドウで関数を調べている場合は，Inspectコマンドを選択すると，その関数のソース・コードが表示されます。

また，調べたい項目をハイライト表示してから↵を押して，このコマンドを起動することもできます。

ESCキーを押し新しいInspectウインドウをクローズすれば，以前のInspectウインドウに戻ることができます。データ構造を調べてから，すべてのInspectウインドウを消去したい場合は，Window | Closeコマンドまたはそのショート・カットであるGRPH-F3を使うか，マウスでクローズ・ボックスをクリックします。

(4) Descendコマンド

このコマンドは，Inspectローカル・メニュー・コマンドと同様に機能します。ただし，新しいInspectウインドウをオープンしてハイライト表示された項目の内容を表示するのではなく，現在のInspectウインドウに新しい項目を表示します。これは，New expressionとInspectの２つのコマンドを合成したようなものです。

このコマンドでデータ構造を下の方にたどっていく場合，展開する以前のデータ構造に戻ることはできません。複雑なデータ構造や連結された長いリストをたどるときに，以前のデータ・レベルに戻る必要がなければDescendコマンドを使います。

このコマンドを使うと，画面上のInspectウインドウの数を減らすことができます。

(5) New expression...コマンド

調べたい変数名または式を入力するよう求めるプロンプトが表示されます。このコマンドを使うと，画面にそれ以上Inspectウインドウをオープンしなくても，ほかのデータを調べることができます。このコマンドは，それまでのデータを現在のInspectウインドウに表示しておく必要がないときに使ってください。

オブジェクトは通常のInspectウインドウとは多少異なっています。オブジェクト型／クラスのInspectウインドウについては，**第10章　C++とオブジェクト指向のディバグ**を参照してください。

(6) Type cast...コマンド

調べている項目のデータ型（Byte，Word，Int，Charポインタ）を指定できます。このコマンドは，Inspectウインドウに型情報がないシンボルが含まれていたり，型情報がないポインタの型を明らかにするシンボルを設定したりする場合に便利です。

〔メ　モ〕

第7章　ブレークポイント

TDシリーズでは，通常，ブレークポイント，ウォッチポイント，およびトレースポイントと呼ばれるディバッガの機能を，まとめてブレークポイントという言葉で表わします。

この章では，TDシリーズのブレークポイントが，従来のブレークポイント，ウォッチポイント，およびトレースポイントに比べて，どれくらい高性能で柔軟性を備えているかについて説明します。

具体的に，次の項目について説明します。

- TDシリーズでのブレークポイントの定義
- BreakpointsウインドウとLogウインドウ
- 単純ブレークポイントと条件ブレークポイント
- プログラム変数の値をログに書き込むブレークポイントの設定方法
- プログラム変数，式，またはデータ項目の値が，いつ変化するかを監視するブレークポイントの設定方法

7.1　ブレークポイントの定義

TDシリーズでは，ブレークポイントを次のような項目で定義します。

(1) ブレークの発生場所

プログラム内の単一の場所かグローバルな場所のどちらでも発生します（グローバルであれば，ブレークは，プログラム内のどのソース行またはどの命令にも発生する可能性があります）。

(2) トリガ条件

- 設定されたブレークポイントを実行するとき
- 式が真のとき
- データ項目の値が変化したとき

(3) トリガ時の動作

次の３つのうちのいずれかを行ってください。

- プログラムの実行を停止する（ブレークポイント）
- 式の値をログに書き込む
- 式を実行する（スプライス・コード）

なお，ソフトウエア・ブレーク条件の設定数に制限はありません。ハードウエア・ブレーク条件の設定数は最大４つまでです。

7.2 Breakpointsメニュー

ホット・キーGRPH-Bを押すと，いつでもBreakpointsメニューにアクセスできます。

7.2.1 Toggleコマンド

ModuleウインドウまたはCPUウインドウのコード・ペインの中で，現在ハイライト表示されているアドレスにブレークポイントを設定したり，すでに設定されているブレークポイントを解除します。ホット・キーはF2です。

7.2.2 At...コマンド

プログラム内の特定の位置にブレークポイントを設定できます。また，すべてのブレークポイント・オプションを設定できるダイアログ・ボックスをオープンします。ホット・キーはGRPH-F2です。

7.2.3 Changed memory global...コマンド

メモリ領域の値が変化するときにトリガされるブレークポイントを設定します。監視するメモリ領域の入力を求めるプロンプトが表示されます。詳細は，7.4.1 （1）Set options...コマンドの(b) Conditionラジオ・ボタンのChanged memoryを参照してください。

7.2.4 Expression true global...コマンド

ユーザが与える式の値が真になるときにトリガされるブレークポイントを設定します。式の入力を求めるプロンプトが表示されます。詳細は，7.4.1 （1）Set options...コマンドの(b) Conditionラジオ・ボタンのExpression trueを参照してください。

7.2.5 Hardware breakpoint...コマンド

このコマンドは，ハードウエアのディバグ時にハードウエア・ブレークポイントをセットするとき使用します。

IEを使ったハードウエア・ブレークポイントの設定に関する詳細は，第13章　IEレファレンスを参照してください。

7.2.6 Delete allコマンド

ユーザが設定したすべてのブレークポイントを削除します。

7.3　ブレークポイント式に適用されるスコープ

ブレークポイントが実行するアクションと，それがトリガされる条件は，ユーザが与える式によって制御できます。

式の評価には，プログラムが停止する現在の位置のスコープではなく，ブレークポイントが設定されているアドレスのスコープが使われます。したがって，スコープのオーバライドを使わないかぎり，ブレークポイント式には，ブレークポイントを設定するプログラム内のアドレスで有効な変数名しか使えません。スコープについては，**9.4　スコープ・オーバライドの構文**を参照してください。

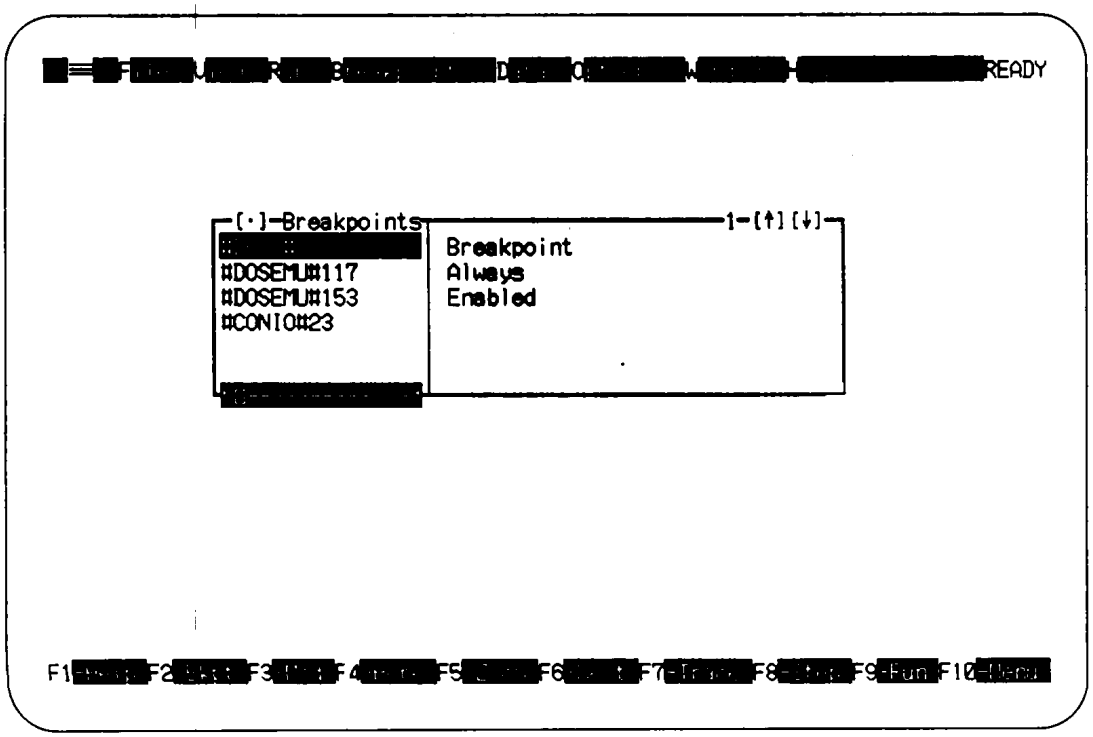
あるルーチンに対してローカルな変数を式の一部として使用すると，そのブレークポイントは，グローバル変数またはモジュールのローカル変数のみを使うブレークポイントよりも，実行速度が大幅に低下します。

7.4　Breakpointsウインドウ

View | Breakpointsコマンドを選択すると，Breakpointsウインドウがオープンします。

このウインドウでは，ブレークポイントをトリガする条件を調べたり，調整したりできます。また，新しいブレークポイントを追加したり，ブレークポイントを削除したり，設定済みのブレークポイントを調整したりできます。

図7－1　Breakpointsウインドウ（View | Breakpoints）



Breakpointsウインドウには，次の２つのペインがあります。

●左側のペイン（ブレークポイント・リスト・ペイン）

ブレークポイントが設定されているすべてのアドレスのリストが表示されます。

●右側のペイン（ブレークポイントの詳細）

左側のペインの中で現在ハイライト表示されているブレークポイントの詳細が表示されます。

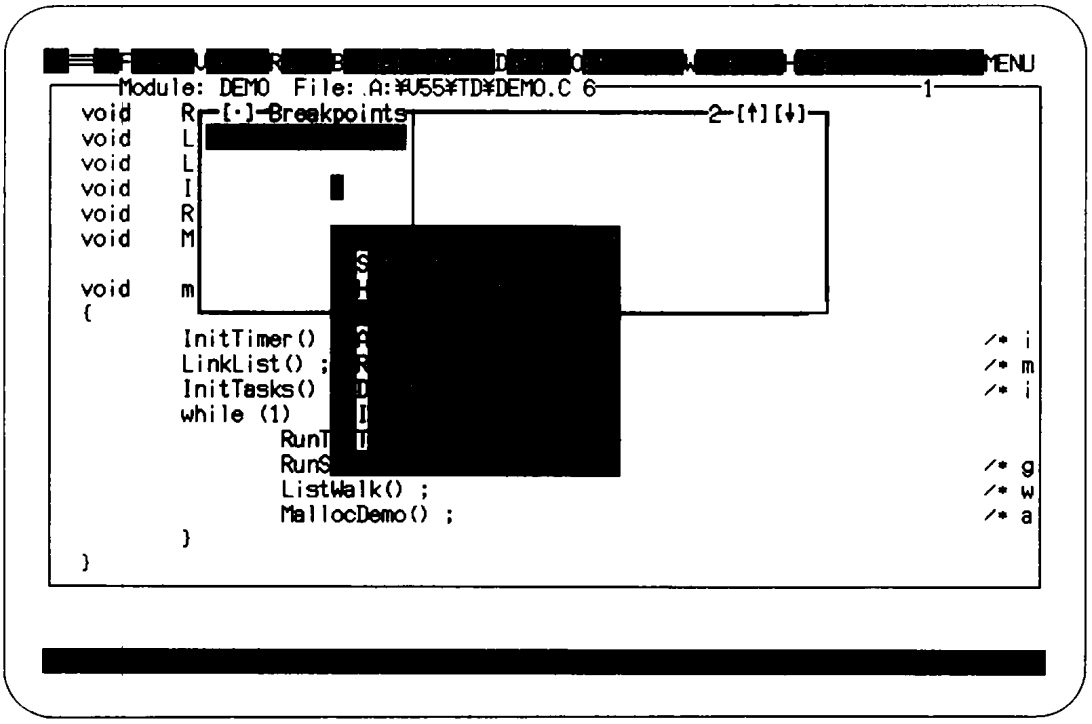
ローカル・メニューがあるのは左側のブレークポイント・リスト・ペインだけです。GRPH-F10を押すとローカル・メニューが表示されます。ローカル・メニューのオプションは，ブレークポイント・リスト・ペイン内でハイライト表示されているブレークポイントに影響を与えます。

7.4.1　Breakpointsウインドウのローカル・メニュー

このメニューのコマンドを使うと，新しいブレークポイントを追加したり，設定済みのブレークポイントを削除したり，ブレークポイントの動作を変更したりできます。

GRPH-F10を押すと，Breakpointsウインドウのローカル・メニューが表示されます。CTRLキーショート・カットが使用可能になっていれば，CTRLキーとともに目的のコマンドの先頭文字を押すことでそのコマンドに直接アクセスできます。

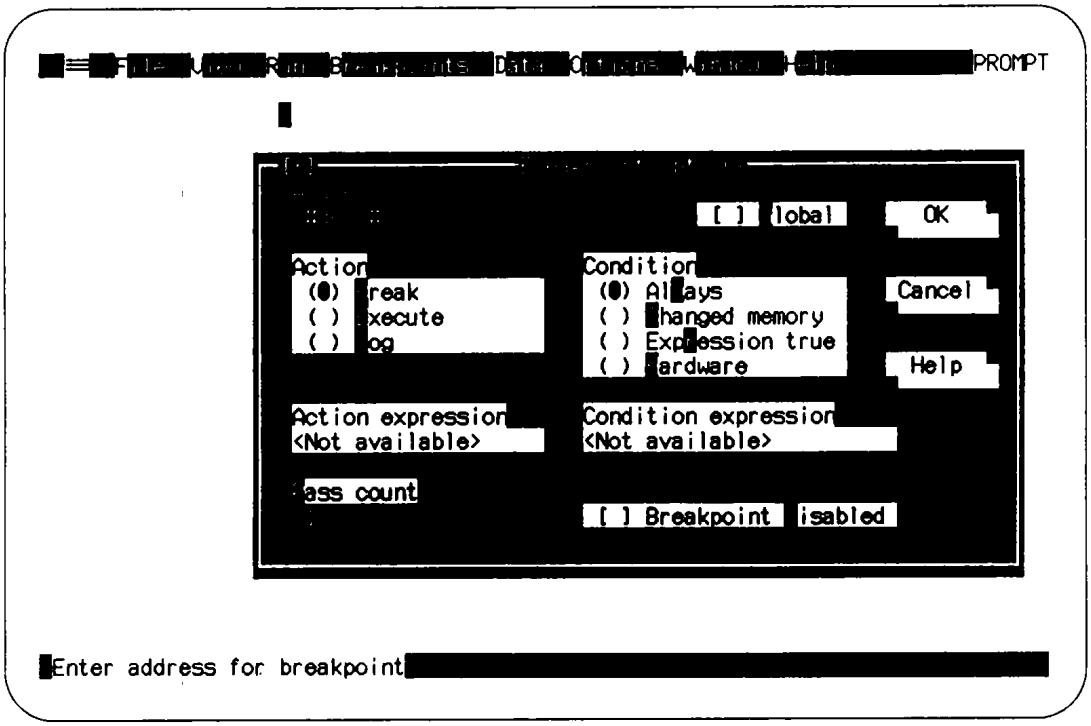
図７－２　Breakpointsウインドウのローカル・メニュー



(1) Set options...コマンド

Breakpoint optionsダイアログ・ボックスをオープンします。このダイアログ・ボックスで，ハードウエア・ブレークポイント以外のブレークポイントの設定を行います。

図7-3　Breakpoint optionsダイアログ・ボックス（ローカル・メニューから選択）



このダイアログ・ボックスには，2つのラジオ・ボタン（Actionラジオ・ボタン，Conditionラジオ・ボタン），3つの入力ボックス（Action expression，Pass count，Condition expression），2つのチェック・ボックス（Breakpoint disabled，Global）があります。これらの機能を次の表に示します。

項　目		機　能
ラジオ・ボタン	Action ラジオ・ボタン	ブレークポイント・リスト・ペイン内でハイライト表示されているブレークが発生したときどうするかを選択します。
	Condition ラジオ・ボタン	ブレークが成立する条件を制御します。
入力ボックス	Action expression	式を設定します。この入力ボックスは，Actionラジオ・ボタンでLogかExecuteを選択したときのみ入力できます。
	Pass count	ブレーク条件が成立した場合，何度目の条件成立でブレークを発生するかを設定します。
	Condition expression	アドレスを設定します。 この入力ボックスは，Conditionラジオ・ボタンでchange memoryかExpression trueを選択したときのみ入力できます。
チェック・ボックス	Breakpoint disabled	ブレーク条件を使用可能または使用不可にします。
	Global	ブレークポイントをグローバルにします。

(a) Actionラジオ・ボタン

Actionラジオ・ボタンには，次の３つの設定があります。

● Break

ブレーク条件が成立したときに，プログラムを停止させます。画面が再び表示されるので，もう一度コマンドを入力して，プログラムのデータ構造を見ることができます。

● Execute

ブレーク条件が成立したときに，式を実行させます。Action expression入力ボックスに式を入力してください。この式には変数をある値に設定するなど，いくつかの副作用がなければなりません。このオプションは，コード・スプライスとして機能します。ユーザは，自分のプログラム内であるコードの前に実行させる式を，現在の行番号のところに挿入できます。

● Log

ブレーク条件が成立したときに，式の値をLogウインドウ内に記録させます。値をログする式を入力するように求めるプロンプトが表示されます。

注意　式中で変数の値が変化するような記述（a＝＊b＋＋など）を行うと，プログラム中の変数も変化してしまいますので注意してください。

(b) Conditionラジオ・ボタン

Conditionラジオ・ボタンには，次の４つの設定があります。

●Always

ブレークポイントがトリガされる前に，ほかの条件が真になる必要がないことを示します。

●Changed memory

メモリの変数またはオブジェクトを監視して，オブジェクトが変化するとブレークポイントをトリガさせます。Condition expression入力ボックスを使って，監視したいオブジェクトを再生する式を入力し，次に，監視するオブジェクト数を入力してください。

グローバル・ブレークポイントに以上の２つの条件を付けると，すべてのソース行が実行されるたびにメモリ領域が変化していないかチェックしなければならなくなります。このため，プログラムの実行速度は大幅に低下します。特定のアドレスにあるブレークポイントにこの条件を設定すると，グローバル・ブレークポイントを設定しても実行速度が低下せず，特定のコード行を実行するたびに変数をチェックできます。

●Expression true

式が真(ゼロ以外)になると，ブレークポイントをトリガさせます。Condition expression入力ボックスを使って，評価される式をそのつど入力してください。

●Hardware

IEのハードウエア・ブレークポイント・リソースを使用してハードウエア・ブレークポイントの設定ができます。

(c) Pass count入力ボックス

Pass count入力ボックスを使うと，ブレークポイントがトリガされる前に，ブレークポイントのアクションが発生しなければならない回数（パス・カウント）を設定できます。ディフォールトの回数は１です。パス・カウントnと条件を設定すると，n回目に条件が真になったときにブレークポイントがトリガされます。

(d) Breakpoint disabledチェック・ボックス

Breakpoint disabledチェック・ボックスを使うと，現在ハイライト表示されているブレークポイントをオンまたはオフにできます。オフにされたブレークポイントは，ユーザが再びオンにするまでは表示されず，削除されたのと同様に扱われます。このチェック・ボックスは，すぐには使わないがあとでもう一度使いたいような複雑なブレークポイントを定義した場合に便

利です。このチェック・ボックスを使うと，ブレークポイントを削除しても，再びその条件とアクションを入力し直す必要がありません。

(2) Hardware options...コマンド

このオプションはTDシリーズでは使用しません。

(3) Add...コマンド

Breakpoint optionsダイアログ・ボックスに似たダイアログ・ボックスをオープンします。Address入力ボックスにアドレスを入力してください。

Addコマンドを呼び出さずにBreakpointsウインドウの中でアドレスの入力を始めると，Addコマンドを起動した場合と同様にダイアログ・ボックスが表示され，ブレークポイントを追加することができます。

ブレークポイントを追加すると，ほかのローカル・メニュー・コマンドを使って，その動作を修正できます。最初にブレークポイントを追加するときは，そのパス・カウントを1に，条件は必ず発生するように設定し，プログラムを停止させます。

(4) Removeコマンド

現在ハイライト表示されているブレークポイントを削除します。

(5) Delete allコマンド

グローバル・ブレークポイントも，特定のアドレスに設定されたブレークポイントも，すべて削除します。プログラムをブレークポイントで停止させたい場合は，さらにブレークポイントを設定しなければなりません。

(6) Inspectコマンド

現在ハイライト表示されているブレークポイントの項目に対応するソース・コード行またはアセンブラ命令を表示します。

そのブレークポイントがプログラム内のソース行に対応するアドレスに設定されている場合は，Moduleウインドウをオープンしてその行に設定します。そうでなければCPUウインドウをオープンし，コード・ペインにブレークポイントが設定された命令を表示します。

また，ハイライト・バーの位置をブレークポイントに合わせて↵を押すと，このコマンドを起動できます。

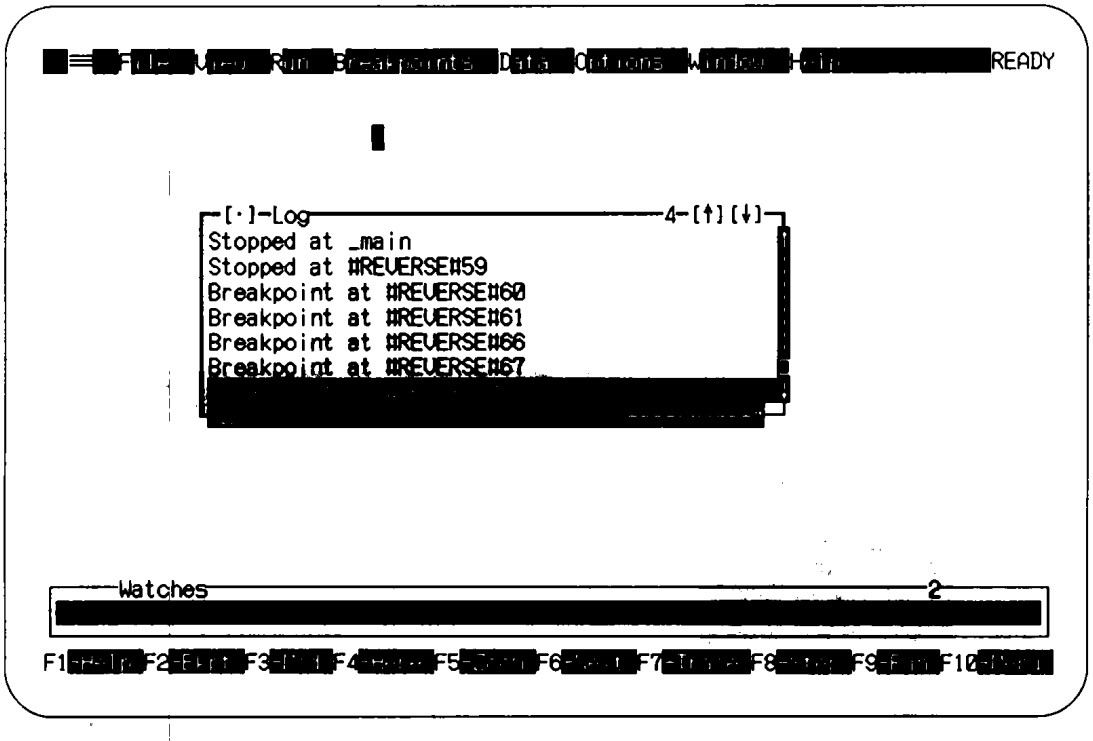
(7) Toggle enableコマンド

CPUウインドウのコード・ペインまたはModuleウインドウ内の希望アドレス行の左端でマウスをクリックすると，ブレークポイントの設定／解除が行えます。なお，この機能は**7.2.1 Toggle**コマンドと同じ内容です。

7.5　Logウインドウ

View | Logコマンドを選択すると，Logウインドウをオープンします。このウインドウを使うと，ディバグ・セッションで発生した重要なイベントのリストを再検討できます。

図7-4　Logウインドウ（View | Log）



Logウインドウは，ウインドウに出力される行のスクロール・リストを表示します。ログに50行以上書き込まれている場合は，スクロールされるリストの先頭から順番に古い行が失われます。行数を調整するには，起動時にコマンド・ライン・オプションを使うか，またはカスタマイズ・プログラムTDINSTBXを使って，行数を半永久的に変更します。Open log fileローカル・メニュー・コマンドを使って，絶えずディスク・ファイルに書き込むようにすれば，ログ全体を保存することができます。

次に，ログに行を書き込むイベントを示します。

- プログラムが指定された位置で停止する。

  停止位置は，ログに記録されます。
- Add commentローカル・メニュー・コマンドが選択される。

  ログに書き込むコメントを入力するよう求めるプロンプトが表示されます。
- 式の値をログに書き込むブレークポイントがトリガされる。

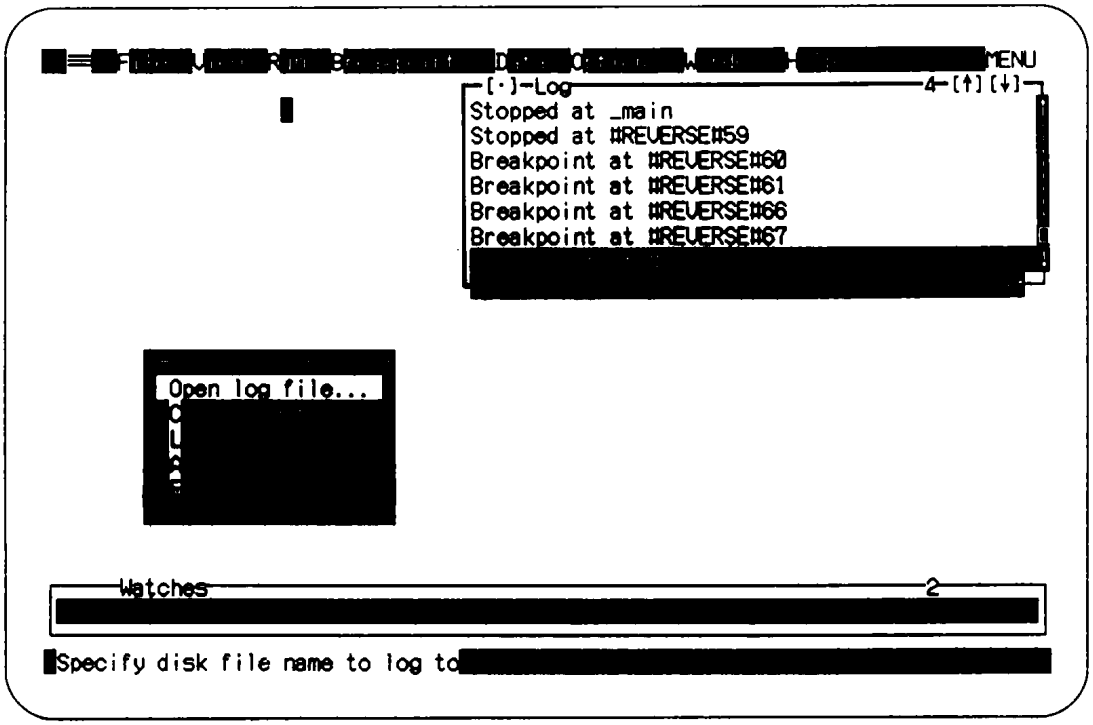
  この値がログに書き込まれます。
- メニュー・バーからWindow | Dump pane to logコマンドを使って，ペインの現在の内容をウインドウに記録する。

7.5.1　Logウインドウのローカル・メニュー

このメニューのコマンドを使うと，ディスク・ファイルへのログの書き込み，ログへの記録の停止と開始，ログへのコメントの追加，およびログの削除ができます。

GRPH-F10を押すと，Logウインドウのローカル・メニューが表示されます。CTRLキー・ショート・カットが使用可能になっていれば，CTRLキーと一緒に目的のコマンドの先頭文字を押すと，そのコマンドに直接アクセスできます。

図7-5　Logウインドウのローカル・メニュー



(1) Open log file...コマンド

ログに書き込まれるすべての行を，ディスク・ファイルにも書き込みます。ダイアログ・ボックスが表示され，ログを書き込むファイル名を入力するようプロンプトが表示されます（または，リスト・ボックスからディレクトリとファイルを選択できます)。

ログ・ファイルをオープンすると，Logウインドウのスクロール・リスト内にすでに表示されているすべての行が，ディスク・ファイルに書き込まれます。したがって，ディスクに記録したいログの内容を見てから，ディスク・ファイルをオープンすることができます。

まったく新しいログから開始したい場合は，Open log fileを選択する前に，まずErase logを選択してください。

(2) Close log fileコマンド

Open log fileローカル・メニュー・コマンドで指定したログ・ファイルへの行の書き込みを停止して，ファイルをクローズします。

(3) Loggingコマンド

ログを有効（Yes）または無効（No）にし，実際にlogウインドウに書き込みを行うかどうかを制御します。

(4) Add comment...コマンド

ログにコマンドを挿入できます。希望する文字を含むテキスト行を入力するようにプロンプトが表示されます。

(5) Erase logコマンド

ログ・リストをクリアします。Logウインドウは空になります。このコマンドは，ディスク・ファイルへのログの書き込みには影響を与えません。

7.6　単純ブレークポイント

ディバグ中に最もよく行うことは，コードの特定の部分が実行されそうになった場合にプログラムを停止させることです。

ブレークポイントを設定する方法はいろいろあるので，次のように状況に応じて使い分けてください。

- Moduleウインドウ内で目的のソース行に移動し，Breakpoints | Toggleコマンドを選択します（または，F2を押すか，その行をマウスでクリックします）。すでにブレークポイントが設定されている行でこの操作を行うと，そのブレークポイントが削除されます。
- CPUウインドウのコード・ペイン内である命令に移動し，Breakpoints | Toggleコマンドを選択します（またはF2を押すか，その行をマウスでクリックします）。すでにブレークポイントが設定されている行でこの操作を行うと，そのブレークポイントが削除されます。
- Breakpoints | Atコマンドを選択して，ブレークポイントを設定したいコード・アドレスを入力します（コード・アドレスの書式は，現在の言語で書かれたポインタの書式と同じです）。
- BreakpointsウインドウのブレークポイントリストペインからAddローカル・メニュー・コマンドを選択して，ブレークポイントを設定したいコード・アドレスを入力します。

7.7　条件ブレークポイントとパス・カウント

あるソース文が実行されるごとに毎回ブレークポイントがトリガされるのを避けたい場合があります。たとえば，目的の状況が発生するまでに，その行のコードが何度も実行される場合などです。TDシリーズ

は，ブレークポイントが実際にはいつトリガされるかを限定する２つの方法，パス・カウントと条件を備えています。

ある関数の10回目の呼び出しでプログラムを停止させたい場合は，その関数の先頭にブレークポイントを設定し，図７－３ **Breakpoint options**ダイアログ・ボックス内のPass count入力ボックスを使って，そのブレークポイントが実際にトリガされるまでにスキップしたい回数を設定します。

一定の条件が真のときにだけプログラムを特定の位置で停止させたい場合は，Breakpoint optionsダイアログ・ボックスのExpression trueラジオ・ボタンをオンにして，Condition expression入力ボックスに条件式を入力します。この式は，そのブレークポイントに出会うたびに評価され，真（ゼロ以外）であればブレークポイントがトリガされます。これをパス・カウントと組み合わせて使うと，式が一定回数だけ真になったあとにブレークポイントをトリガすることができます。

Changed memoryラジオ・ボタンをオンにすると，データ項目の値が変化したあとでトリガされるブレークポイントを指定できます。

この方法は，グローバル・ブレークポイントを指定するよりはるかに効率的です。変化するものを監視するのが，特定のソース文に到達したときだけなので，変化の発生を検出するためにTDシリーズの処理量は減少します。

7.8　グローバル・ブレークポイント

ソース行または命令に出会うたびにブレークポイントを発生させたい場合は，グローバル・ブレークポイントを使います。グローバル・ブレークポイントを設定する方法はいろいろあるので，次のように状況に応じて使い分けてください。

- Breakpoint optionsダイアログ・ボックスで，Globalチェック・ボックスをオンにします。この方法は，限定条件かパス・カウントを設定したい場合，または，ブレークポイントがトリガされたときに停止させる以外のことをしたい場合に使ってください。
- 特定のメモリ領域が変化したときにプログラムを停止させたい場合は，Breakpoints | Changed memory globalコマンドを選択します。
- 式が真になったときに実行を停止させたい場合は，Breakpoints | Expression true globalコマンドを選択します。

グローバル・ブレークポイントを設定する場合は，通常Breakpointsウインドウ内のローカル・メニューを使って，条件またはアクションを修正します。修正しない場合，各ソース行ごとにブレークポイントのアクションが発生してしまいますので注意してください（Run | Trace intoメイン・メニュー・コマンドを使っているようなものです）。

各ソース行が実行される直前に毎回グローバル・ブレークポイントをテストしたい場合は，現在のウインドウがCPUウインドウではないことを確認してから，メニュー・バーからRunコマンドの１つを選択して（または対応するファンクション・キーを押して）プログラムを再開してください。

1命令が実行されるたびにグローバル・アクションをテストするには，プログラムを再開するときに，現在のウインドウがCPUウインドウであることを確認してください。

グローバル・アクションは，すべてのソース行または命令ごとに発生します。変数がいつ変化するか，または，条件がいつ真になるかを正確に知りたい場合には，グローバル・ブレークポイントを使ってください。

インサーキット・エミュレータなどのハードウエア・ディバグ・ツールを使用している場合は，グローバル・ブレークポイントの代わりにハードウエア・ブレークポイントを使用してください。

グローバル・ブレークポイントは，プログラムの実行速度を大幅に低下させます。しかし，プログラムがデータを破壊してしまう場所を見つけるときなどには大変便利です。ただし，グローバル・ブレークポイントを追加する場合は，それをトリガする条件を設定してください。

7.9　データ項目の変化に伴う停止

プログラム内で，あるデータ項目が変更される場所を見つけたい場合は，まず前述した方法の1つを使って，グローバル・ブレークポイントを設定します。次に，Breakpoint optionsダイアログ・ボックスのChanged memoryラジオ・ボタンをオンにします。入力ボックスが表示されたら，調べたいメモリ領域を指す式と，必要であれば，調べる項目の数のカウントを入力してください。

注意　このコマンドを使うと，データ項目が変化する正確な位置を特定することができます。ただし，プログラムの実行速度は低下します。まず，問題箇所が存在し得る範囲をある程度狭めておいてください。

7.10　変数の値をログに書き込む

プログラム内の特定の場所に到達したとき，そのつど，ある変数の値をログに記録しておくと便利な場合があります。どのような式の値でもログに書き込むことができます。たとえば，関数の呼び出しパラメータの値を書き込む場合，その関数が呼び出されるたびにログを調べると，いつ誤りのあるパラメータで呼び出されたかを知ることができます。

Breakpoint optionsダイアログ・ボックスからLogラジオ・ボタンを選択してください。ブレークポイントがトリガされるたびに，その値をログに書き込む式を入力するようにプロンプトが表示されます。複数の変数の値をログに書き込みたい場合は，複数のブレークポイントを設定してください。

7.11　式の実行

ブレークポイントがトリガされるたびに副作用をもつ式を実行すると，与えられたソース行の前に新しいコード部分を効率よく接続することができます。この方法は，検査やバグの修正をテストするため，ルーチンの動作を変更したい場合に便利です。これにより，あるルーチンに対して行ったわずかな変更をテス

トするために，コンパイル→リンク→ロケートのサイクルを繰り返さなくてもすみます。

注意　この方法では，既存のコード行が実行される前に式を挿入することはできますが，既存のソース行を直接修正することはできません。

第8章　ファイルの調査，変更

TDシリーズは，ディスク・ファイルを，ディバグ中のプログラムが延長された部分として扱います。ディスク上のファイルも，ASCIIテキストまたは16進数データとして表示させて，調べたり修正したりできます。また，好みのテキスト・エディタを使って，TDシリーズの中からテキスト・ファイルを変更できます。

この章では，2種類のディスク・ファイル（プログラムのソース・コードとディスク上のほかのファイル）を調べたり修正したりする方法について説明します。

8.1　プログラムのソース・ファイルの調査…Moduleウインドウ

プログラムのソース・ファイルとは，コンパイルされて最終的にTDシリーズが使用するロード・モジュールを生成するソース・ファイルのことです。通常，コードの一部の動作や設計を調べたいときには，このソース・ファイルを調べます。ディバグ中には，関数の引き数が有効であるか，関数が正しい値を返しているかなどを調べるために，その関数のソース・コードを頻繁に見る必要があります。

プログラムをステップごとに実行しているときに，自動的にプログラム内の現在位置に対応するソース・コードが表示されます。

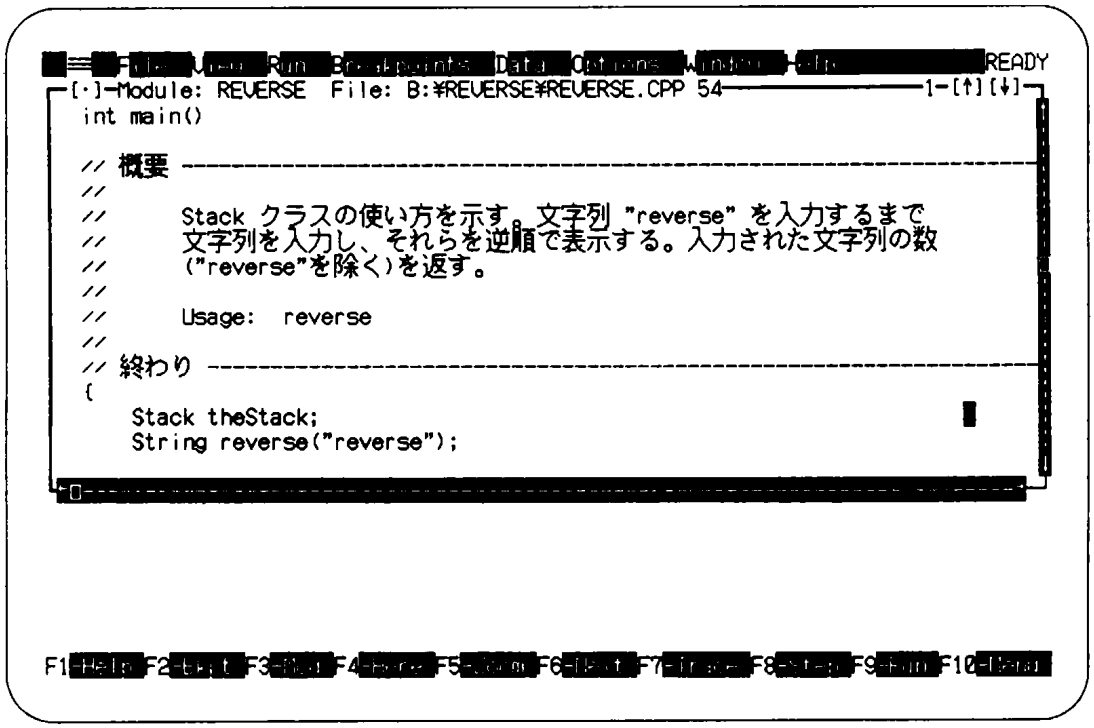
コンパイラ指令(Cの#includeやアセンブラのINCLUDE)によってソース・ファイルに読み込まれるファイルも，プログラムのソース・ファイルとみなされます（View | Moduleを選択すると，Pick a moduleダイアログ・ボックスにプログラムのソース・ファイル名として表示されます)。

Moduleウインドウは，そのファイルがソース・モジュールであることをTDシリーズに知らせる機能を持っています。このため，プログラムのソース・ファイルを調べるときには，必ずModuleウインドウを使わなければなりません。このウインドウでは，ファイル内の適切な位置に移動するだけで，ブレークポイントを設定したり，プログラム変数を調べることができます。この方法については次の節で説明します。

8.2　Moduleウインドウ

メニュー・バーからView | Moduleコマンドを選択して（またはホット・キーF3を押して）Moduleウインドウをオープンします。

図8－1　Moduleウインドウ（View | Module）



モジュール名を入力するダイアログ・ボックスが表示されます。

TDシリーズは，ユーザが選択したモジュールのソース・ファイルをロードし，ソース・ファイルを次に示す順序で検索します。

- コンパイラがそのソース・ファイルを見つけたディレクトリ
- Options | Path for sourceコマンドまたは－sdコマンド・ライン・オプションで指定されたディレクトリ
- カレント・ディレクトリ
- ディバグしているプログラムが入っているディレクトリ

Moduleのウインドウには，選択されたモジュールのソース・ファイルの内容が表示されます。Moduleウインドウのタイトルには，現在表示されているモジュール名，ソース・ファイル名，およびカーソルが置かれている行番号が示されます。ウインドウの1桁目の矢印(→)は，プログラムの現在位置(PS：PC)を示します。

注意　.AXEファイルとオリジナルのソース・ファイルの両方が必要です。TDシリーズはソース・ファイルを，前述した場所と順序で探します。

ウインドウ・タイトルのファイル名のあとにmodifiedと表示された場合，現在ディバグしているプログ

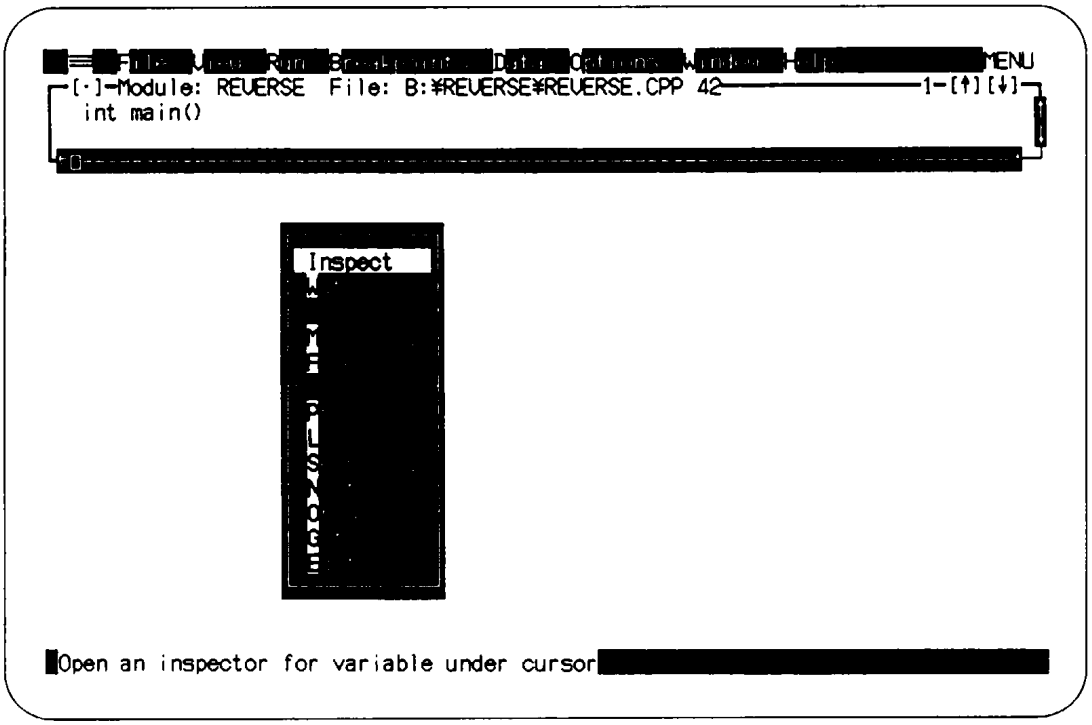
ラムを作成するために最後にコンパイルまたはリンクされ，そのあとでそのファイルが変更されたことを示しています。これは，更新されたソース・ファイル内のルーチンの行番号が，ディバグ中のプログラムの構築に使われたバージョンの行番号と異なっている可能性があることを意味します。行番号が異なると，プログラムの現在位置（PS：PC）を示す矢印が間違った行に表示される可能性があります。

8.2.1　Moduleウインドウのローカル・メニュー

Moduleウインドウのローカル・メニューには，表示されているモジュールの中を移動したり，データ項目を指定してそれを調べたり，ウインドウが新しいファイルやモジュールを表示するように設定するための，多数のコマンドがあります。TDシリーズでは，ほかのメニューに比べて，このモジュールを使うことが多いので，いろいろなオプションを十分に知っておく必要があります。

Moduleウインドウのローカル・メニューをポップアップするには，GRPH-F10を使うか，またはCTRLキーのショート・カットが使用可能であれば，CTRLキーと一緒に目的のコマンドの先頭文字を押して，そのコマンドにアクセスします（たとえばSearchの場合はCTRL-S)。

図8-2　Moduleウインドウのローカル・メニュー



(1) Inspectコマンド

Inspectorウインドウをオープンして，カーソル位置にあるプログラム変数の内容を表示します。このコマンドを選択する前に，プログラム変数を入力することもできます。調べたいソース・ファイル内のプログラム変数にカーソルを合わせるか，または表示されるダイアログ・ボックスの入力ボックスに入力してください。

INSキーを使って，調べたい式を選択することもできます。この方法を使うと，ソース・モジュール中に書かれている式であれば入力の手間を省くことができます。目的の名前をそのつど入力しなくても済むので，プログラム変数の内容を調べるときには大変便利です。

(2) Watchコマンド

カーソル位置の変数をWatchesウインドウに追加します。このコマンドは，プログラムの実行中に変数の値を絶えず監視したいときに便利です。このコマンドを選択する前に，プログラム変数を入力することもできます。調べたいソース・ファイル内のプログラム変数にカーソルを合わせるか，または表示されるダイアログ・ボックスの入力ボックスに入力してください。

INSキーを使って，監視したい式をマークすることもできます。この方法を使うと，ソース・モジュール中に書かれている式であれば入力の手間を省くことができます。

(3) Module...コマンド

表示されているモジュール・リストから目的のモジュールを1つ選択して，そのモジュールを見ることができます。現在のモジュールをもう見る必要がなく，画面上にこれ以上Moduleウインドウをオープンしたくない場合に便利です。

(4) File...コマンド

現在表示されているモジュールを構成するファイル以外のソース・ファイルを表示するように切り替えます。表示されるファイルのリストから，見たいファイルを選択してください。たいていのモジュールでは，コードを含むソース・ファイルは1つだけです。モジュールを構成するほかのファイルは，通常は定数とデータ構造を定義したものです。モジュールのソース・コードが複数のファイルに分かれている場合は，このコマンドを使ってください。

第1のファイルは，View | Fileでオープンします。複数のファイルを見たい場合は，View | Another | Fileで，次々にFileウインドウを開いてください。

(5) Previousコマンド

直前のソース・モジュールの位置に戻ります。

現在のモジュール内の位置を変更するコマンドを実行したあとで，このコマンドを使って元の位置に戻ることもできます。

(6) Line...コマンド

ファイル内の新しい行番号に移動します。移動先の行番号を入力してください。ファイル内の最終行よりも大きな行番号を入力すると，ファイル内の最終行に移動します。

(7) Search...コマンド

現在のカーソル位置から文字列の検索を開始します。検索したい文字列を入力してください。カー

ソルが変数名などに置かれていると，検索ダイアログ・ボックスには最初にその名前が表示されます。また，INSキーを使ってファイル内のブロックをマークしておくと，検索ダイアログ・ボックスには最初にそのブロックが表示されます。この方法を使うと，検索したい文字列が，表示されているファイル内にあれば，入力の手間を省くことができます。

ワイルド・カード“？”と“＊”を検索文字列に使用することができます。“？”は任意の１文字に一致し，“＊”は０文字以上の任意の文字列に一致します。検索は，ファイルの最後に達すると終了します(先頭には戻りません)。ファイル全体を検索するには，先にCTRL-ROLLDOWNを押してファイルの１行目に移動してください。

(8) Nextコマンド

Searchコマンドで指定した文字列で，次に一致するものを検索します。

このコマンドを使うと，希望する文字列が見つかるまで検索を繰り返せます。ただし，このコマンドはSearchコマンドを実行したあとでのみ使えます。Searchコマンドの実行では文字列を希望する場所で見つけられなかったときに便利です。

(9) Originコマンド

プログラムの現在位置(PS：PC)であるモジュールと行番号に移動します。現在表示されているモジュールがプログラムの現在位置を含むモジュールでない場合は，そのモジュールを表示するためにModuleウインドウが切り替えられます。このコマンドは，コードのあちらこちらを調べたあとで，プログラムが現在停止している位置に戻りたいときに便利です。

(10) Goto...コマンド

プログラム内の任意の位置に移動できます。調べたいアドレスを入力してください。行番号，関数名，または16進アドレスを入力できます。アドレスの入力方法については，第９章　式を参照してください。

また，ローカル・メニューを表示しなくても，移動先を示すラベルを入力するだけでこのコマンドを起動できます。すると，Run | Execute toコマンドを指定したときと同じダイアログ・ボックスが表示されます。Gotoコマンドは頻繁に使われるため，このホット・キーを使うと便利です。

(11) Editコマンド

現在表示されているモジュールのソース・ファイルが変更できるように，選択したエディタを起動します。また，カスタマイズ・プログラムTDINSTBXを使って，エディタを起動するコマンドを設定することができます。

8.3　ほかのディスク・ファイルを調べる…Fileウインドウ

Fileウインドウを使うと，システム上のどのファイルでも，調べたり修正したりできます。後述するDisplay

asコマンドを使って，ファイルをASCIIテキストまたは16進データ・バイトとして表示することができます。

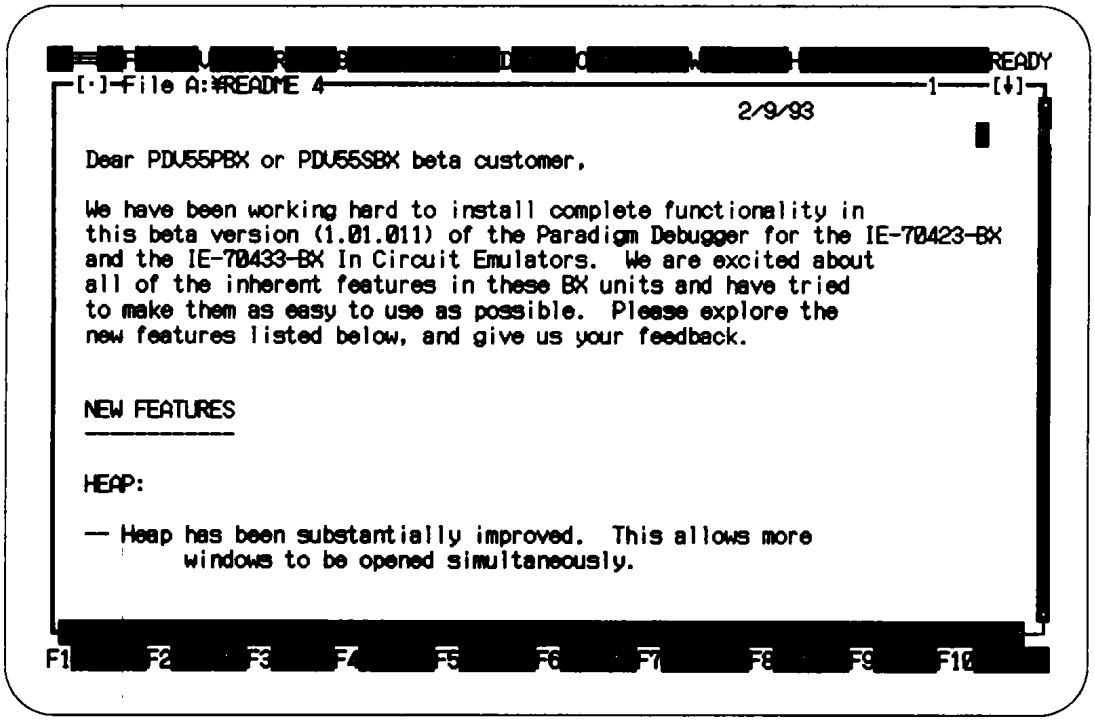
8.4　Fileウインドウ

メニュー・バーからView | Fileコマンドを選択して，Fileウインドウをオープンします。MS-DOS形式のワイルド・カードを使ってファイル・リストから選択するか，特定のファイル名を入力してロードできます。

Fileウインドウには，選択したファイルの内容が表示されます。ウインドウのタイトルには，現在表示されているファイルの名前と，カーソル位置の行番号（ファイルがASCIIテキストとして表示されていれば）が表示されます。

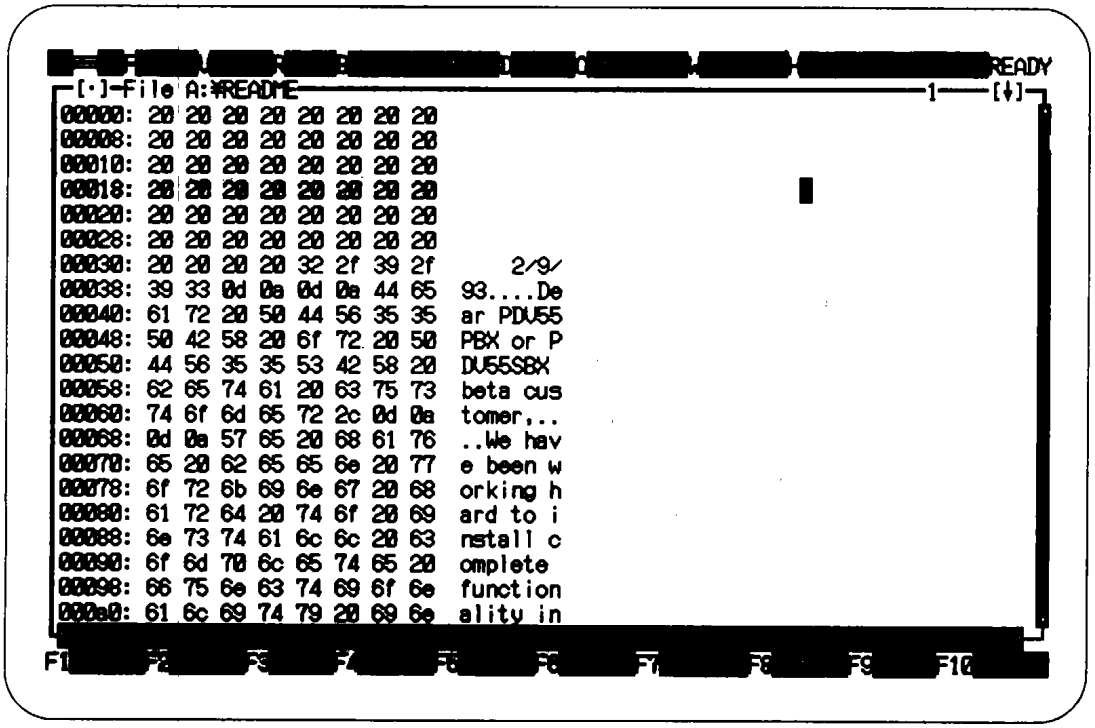
最初にFileウインドウを生成すると，TDシリーズは，そのファイルの内容がASCIIテキストかバイナリ・データかを判断して，ASCIIテキストまたは16進バイトのどちらかでファイルを表示します。

図8－3　ASCII表示のFileウインドウ（View | File→File名選択）



ASCII表示と16進表示は，後述するDisplay asローカル・メニュー・コマンドを使って，いつでも切り替えることができます。

図８－４　16進表示のFileウインドウ

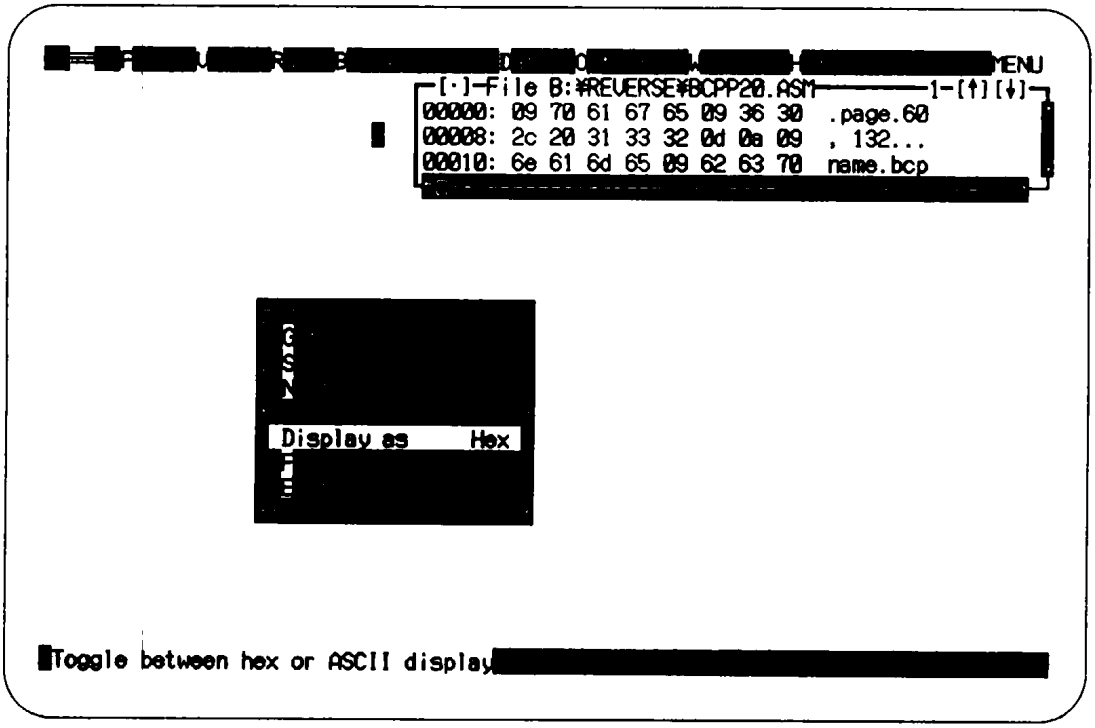


8.4.1　Fileウインドウのローカル・メニュー

Fileウインドウのローカル・メニューには，ディスク・ファイルの中を移動したり，ファイルの内容を表示する方法を変更したり，ファイルに変更を加えたりするための多数のコマンドがあります。

GRPH-F10でFileウインドウのローカル・メニューを表示するか，CTRLキー・ショート・カットが使用可能であれば，CTRLキーと一緒に目的のコマンドの先頭文字を押して，そのコマンドにアクセスします。

図8-5　Fileウインドウのローカル・メニュー



(1) Gotoコマンド

ファイル内の新しい行番号またはオフセットに移動します。ファイルがASCIIテキストで表示されている場合は，移動先の行番号を入力してください。ファイルが16進バイトで表示されている場合は表示を開始するファイルの先頭からのオフセットを入力してください。オフセットの入力には完全な式が使えます。ファイル内の最終行よりも大きい行番号，または，ファイルの終わりよりあとのオフセットを入力すると，ファイルの最後に移動します。

(2) Searchコマンド

現在のカーソル位置から文字列の検索を開始します。検索する文字列を入力するようプロンプトが表示されます。カーソルがシンボル名などに置かれていると，Searchダイアログ・ボックスには最初にその名前が表示されます。また，INSキーを使ってファイル内のブロックをマークしておくと，Searchダイアログ・ボックスには最初にそのブロックが表示されます。この機能により，検索したい文字列が現在表示されているファイル内にあれば，その文字列を入力しなくてもすみます。

検索文字列の書式は，ファイルがASCIIと16進のどちらで表示されているかによって異なります。ファイルがASCIIで表示されている場合は，ワイルド・カード“?”と“*”を使うことができます。“?”は任意の1文字に一致し，“*”は0文字以上の任意の文字列に一致します。

16進バイトの場合は，式に用いられている言語の構文を使って，一連のバイト値または引用符で囲んだ文字列からなるバイト・リストを入力してください。

検索は，ファイルの最後に達すると終了します(先頭には戻りません)。ファイル全体を検索する

には，先にCTRL-ROLLDOWNを押してファイルの１行目に移動してください。

単に検索したい文字列を入力して，このコマンドを起動することもできます。このコマンドを使うと，Searchコマンドを指定したのと同様にダイアログ・ボックスが表示されます。

(3) Nextコマンド

Searchコマンドを使って指定した文字列で，次に一致するものを検索します。このコマンドを使うと，希望する文字列が見つかるまで検索を繰り返せます。ただし，Searchコマンドを実行したあとでのみ使えます。

このコマンドは，Searchコマンドでは文字列を希望する場所で見つけられなかったときに便利です。

(4) Display asコマンド

表示形式をASCIIテキストと16進バイトの間で切り替えます。ASCII表示を選択すると，ファイルはエディタやワード・プロセッサの画面表示のように表示されます。Hex表示を選択すると，各行の先頭には，その行のバイトのファイルの先頭からの16進オフセットが表示されます。1行につき８バイトのデータが表示されます。その８バイトの16進表示の右側には，各バイトに対応する表示文字が示されます。制御コード（00h-1Fh，7Fh）はピリオドで示されます。

また，２バイトの漢字コードは漢字として表示されますが，あるバイトが漢字第１バイト目に該当し，その次のバイトが漢字第２バイトに該当しないときには，最初のバイトはピリオドで表示されます。

(5) File...コマンド

Fileウインドウの表示を別のファイルに切り替えます。MS-DOS形式のワイルド・カードを使ってファイルのリストから選択するか，特定のファイル名を入力してロードすることができます。これにより，新しいFileウインドウを画面上にオープンしなくても，別のファイルを見ることができます。2つの異なるファイル，または同じファイルの２つの部分を同時に見たいときは，View | Another | Fileコマンドを選択してFileウインドウをもう１つオープンしてください。

(6) Editコマンド

ファイルがASCIIテキストとして表示されている場合は，このコマンドを使ってカスタマイズ・プログラムTDINSTBXで指定したエディタを起動します。これにより，現在表示されているファイルを変更できます。

ファイルが16進データ・バイトとして表示されている場合は，エディタが起動されません。その代わりに，カーソル位置のバイトを置き換えるバイトを入力するようにプロンプトが表示されます。バイト・リストを検索するときの入力と同じように，バイト・リストを入力してください。

〔メ モ〕

第９章　式

式は，プログラム中のシンボル(変数やルーチン名)と，TDシリーズがサポートするC，Pascal，またはアセンブラの３種類の言語の１つで使用できる定数および演算子を組み合わせたものです。

TDシリーズでは，式を評価してその値を知ることができます。また，式を使って，データ項目のメモリ上の値を示すこともできます。さらに，メモリ内の値やアドレスの入力を求めるプロンプトに対して，式をダイアログ・ボックスに入力することもできます。

Data | Evaluate/modifyを選択すると，式の値が表示されるEvaluate/modifyダイアログ・ボックスがオープンします（このダイアログ・ボックスは簡易電卓としても使えます)。

この章では，次の項目について説明します。

- 式の評価に使う言語の選択方法
- C，Pascal，アセンブラの式の構成要素（定数，プログラム変数，文字列，演算子など）

  また，各言語ごとに，TDシリーズがサポートする演算子と式の構文を表にして説明します。

9.1　式を評価する言語の選択

通常，現在のモジュールの言語から，使用する式の評価機能と言語が決定します。現在のモジュールとは，プログラムが停止したモジュールのことです。Options | Languageコマンドでオープンする Expression Languageダイアログ・ボックスを使うと，この言語をオーバライド（再設定）することができます。このダイアログ・ボックス内では，Source，C，Pascal，Assemblerのいずれかのラジオ・ボタンを設定します。Sourceを選択すると，式はそのモジュールの言語で評価されます(TDシリーズがモジュールの言語を決定できなければ，アセンブラ式の規則を使います)。

通常は，TDシリーズに，使用する言語を選択させて問題はありません。ただし，別の言語から呼び出されるアセンブラ・モジュールをディバグするときなどは，言語を明示的に指定した方が便利なことがあります。式の評価に特定の言語を使うように明示的に指定すると，現在のモジュールに別の言語が使われていても，ユーザが指定した言語を使ってデータを参照するのと同じようにデータをアクセスすることができます。

式や変数は，別の言語で書かれているものとして扱った方が操作しやすいことがあります。たとえば，Pascalプログラムをディバグしている場合，アセンブリ言語やCの規則を使った方が文字列に格納されているバイト値は簡単に変更できます。

最初に適切な言語を選択しておけば，ほかの言語の規則を使うのにわずらわされることはありません。TDシリーズは，元のソース言語に関する情報を常に保持しているので，規則やデータ記憶域を適切に処理することができます。言語があいまいであるとみなすと，ディフォールトの状態でアセンブリ言語が使われます。

注意 TDシリーズを起動するときに間違った言語を選択すると誤動作の原因になります。

9.2 コード・アドレス，データ・アドレス，および行番号

通常，プログラム内で変数またはルーチン名にアクセスしたいときは，その名前を入力するだけです。しかし，メモリ・ポインタを評価する式を入力したり，＃123のように行番号の前にシャープ記号＃を付けて，コード・アドレスをソース行番号として指定することもできます。次の節では，カレント・スコープの外にあるシンボルにアクセスする方法を説明します。

プログラムのソース・コード言語用の16進数構文を使って，正規のセグメント：オフセット・アドレスを指定することもできます。

言 語	書 式	例
C	0xnnnn	0x1234：0x0010
Pascal	$nnnn	$1234：$0010
アセンブラ	nnnnh	1234h：0010h 1234h：0B234h

アセンブラでは，AからFまでの文字で始まる16進数の前にプリフィクス０（ゼロ）をつけてください。

9.3 カレント・スコープ外のシンボルへのアクセス

ディバッガがシンボルを探す場所を，そのシンボルのスコープと呼びます。

通常，TDシリーズは，コンパイラと同じ方法で式の中のシンボルを探します。たとえばCは，最初に現在の関数の中でローカルなシンボルを探し，最後にグローバル・シンボルを探します。それでもシンボルが見つからなければ，ほかのすべてのモジュールを検索して，そのシンボルと一致するローカル・シンボルを探します。これにより，モジュール名を明示的に指定しなくても，ほかのモジュール内の識別子を参照することができます。

通常のスコープ外でシンボルを検索したい場合は，モジュール，モジュール内のファイル，またはルーチンの内側を探すように指定します。プログラム内の定義された値を持つどんなシンボルにもアクセス可能です。

9.4 スコープ・オーバライドの構文

シンボル名のスコープをオーバライドするには，使用している言語に関わらず同じ方法を使います。

通常，スコープの構成要素の区切りにはシャープ記号＃を使います。現在使用中の言語内でまぎらわしくなければ，＃の代わりにピリオド（.）を使って最初のシャープ記号を省略することもできます。

次の構文は，スコープをオーバライドするためのものです。カッコ [] は，指定しなくてもよい項目であることを示します。

[＃module [＃filename]] ＃linenumber [＃variablename]

または次のようにも表せます。

[＃module [＃filename]] [＃functionname] ＃variablename

モジュールを指定しない場合，現在のモジュールを指定したものとみなされます。次に，スコープ・オーバライドを指定した有効なシンボル式の例を示します。スコープのオーバライドに使える要素を正しく組み合わせて，1つずつ例を示します。

＃123	カレント・モジュール内の123行目
＃123＃myvar	カレント・モジュールの123行目からアクセスできるシンボルmyvar1
＃mymodule＃123	モジュールmymodule内の123行目
＃mod＃file＃func＃var	modに含まれるfile内のfuncからアクセス可能な変数var

また，次のようにスコープのオーバライドにクラスとメンバ関数名を使用しても可能です。

AnInstance	現在のスコープ中でアクセス可能なインスタンスAnInstance
AnInstance. AField	現在のスコープ中でアクセス可能なインスタンスAnInstanceのフィールドAField

これらの修飾付き識別式は，式が有効であれば，Evaluate/modifyダイアログ・ボックスやWatchesウインドウなどどこにでも入力できます。また，ソース・コード内のメソッドやメンバ関数や手続きのアドレスにGotoコマンドで移動する際，Inspectorウインドウ内の式を変更したり，Moduleウインドウ内のローカル・メニューを使うときにも，これらの修飾付き識別式を入力できます。

C++プログラムをディバグしていて，オーバロードされた名前を持つ関数を調べたい場合は，その関数名を適切な入力ボックスに入力してください。Pick a symbol nameダイアログ・ボックスがオープンし，引き数付きでその名前のすべての関数のリスト・ボックスが表示されます。その中から目的の関数を選択してください。

9.5 式の評価に使用される暗黙のスコープ

一般に，C言語では同名シンボルの使用を許していますので，シンボル名だけ示したのではどの範囲で有効なシンボルか判別できません。これを解決するために，TDシリーズではスコープ・オーバライドとカレント・スコープの考え方を用いています。

そのシンボルがどこで有効なシンボルかを明示的に示すには，シンボルにスコープ・オーバライドを付けてください。

シンボルだけが指定された場合には，現在のカーソルの位置を基準に有効範囲が決定されます。これがカレント・スコープです。したがって，Moduleウインドウ内の特定の行にカーソルを移動すると，式が評価されるスコープを設定することになります。このため，カーソルをプログラムが停止した現在行から移動すると，式の評価結果は予想外のものになることがあります。プログラムのカレント・スコープ内で確実に式が評価されるようにしたい場合は，ModuleウインドウのOriginローカル・メニュー・コマンドを使って，ソース・コード内の現在位置に戻ってください。また，CPUウインドウのコード・ペイン内で移動したり，Stackウインドウ内でルーチンをカーソルで選択したり，Variablesウインドウ内のルーチン名をカーソルで選択すると，式のスコープを設定することになります。

9.6 バイト・リスト

コマンドには，CPUウインドウのデータ・ペインのSearchおよびChangeローカル・メニュー・コマンドや，ファイルを16進形式で表示するときのFileウインドウのSearchおよびChangeローカル・メニュー・コマンドなど，バイト・リストを入力するように求めるものがあります。

バイト・リストは，Options | Languageコマンドによって決定された現在の言語の構文によるスカラ数（浮動小数点数以外の数）と文字列の任意の組み合わせです。文字列とスカラは，式と同じ構文を使います。スカラは，対応するバイト列に変換されます。たとえば，PascalのLongint値123456は，4バイトの16進数の列40 E2 01 00になります。

言　語	バイト・リスト	16進データ
C	"ab"0X04"c"	61 62 04 63
Pascal	'ab'#4'c'	61 62 04 63
アセンブラ	1234"AB"	34 12 41 42

9.7 Cの式

TDシリーズは，Cの式の構文を完全にサポートします。Cの式は，シンボル，演算子，文字列，変数，および定数の組み合わせで構成されます。次の各項では，これらの構成要素について説明します。

9.7.1 Cのシンボル

シンボルとは，プログラム内のデータ項目またはルーチンの名前です。シンボル名は文字(a-z, A-Z)またはアンダ・スコア (_) で始まります。シンボル内の２番目以降は，これらの文字でも０から９までの数字でもかまいません。

また，最初のアンダ・スコアを省略することもできます。アンダ・スコアを付けずにシンボル名を入

力してその名前が見つからなかった場合には，先頭にアンダ・スコアが付き，再度検索されます。コンパイラによっては，シンボル名の先頭に自動的にアンダ・スコアが付け加えられます。

9.7.2 Cのレジスタ疑似変数

TDシリーズでは，疑似変数を使ってプロセッサ・レジスタにアクセスできます。疑似変数とは，所定のプロセッサ・レジスタに対応する変数名です。

疑似変数		型	レジスタ
Vシリーズ表記版	インテル表記版		
_AW	_AX	unsigned int	AW
_AL	_AL	unsigned char	AL
_AH	_AH	unsigned char	AH
_BX	_BX	unsigned int	BW
_BL	_BL	unsigned char	BL
_BH	_BH	unsigned char	BH
_CW	_CX	unsigned int	CW
_CL	_CL	unsigned char	CL
_CH	_CH	unsigned char	CH
_DW	_DX	unsigned int	DW
_DL	_DL	unsigned char	DL
_DH	_DH	unsigned char	DH
_PS	_CS	unsigned int	PS
_DS0	_DS	unsigned int	DS0
_SS	_SS	unsigned int	SS
_DS1	_ES	unsigned int	DS1
_SP	_SP	unsigned int	SP
_BP	_BP	unsigned int	BP
_IY	_DI	unsigned int	IY
_IX	_SI	unsigned int	IX
_PC	_IP	unsigned int	PC

9.7.3 Cの定数と数の表記

定数には，浮動小数点または整数の両方が使えます。

整数定数は，これをオーバライドするCの規則を使わないかぎり，10進数で指定されます。

表 記	例	基 数
数字	100	10進数
0数字	074	8進数
0X数字	0XFFE0	16進数
0x数字	0xA100	16進数

定数は，通常，int型（16ビット）です。long型（32ビット）の定数を定義したい場合は，123456Lのように，定数のあとにLまたはlを付け加えなければなりません。

浮動小数点定数は，小数点を含む10進数で，次のように10進数表記または指数表記のどちらも使えます。

1.234　　4.5e+11

9.7.4 エスケープ・シーケンス

文字列とは，二重引用符（"）で囲まれた一連の文字のことです。エスケープ文字として，Cで使われる円記号（¥）を使うことができます。

シーケンス	値	文 字
¥¥	0X5C	円記号（¥）
¥a	0X07	ベル
¥b	0X08	バック・スペース
¥f	0X0C	フォーム・フィード
¥n	0X0A	改行
¥r	0X0D	復帰
¥t	0X09	水平タブ
¥v	0X0B	垂直タブ
¥xnn	nn	16進バイト値
¥nnn	nnn	8進バイト値

円記号のあとに上記の文字以外の文字を書くと，その文字はそのまま文字列に挿入されます。

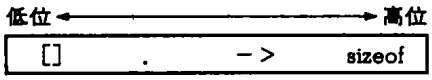
9.7.5 Cの演算子の優先順位

TDシリーズは，Cと同じ演算子を同じ優先順位で使います。ディバッガには，C++の演算子セットの一部である演算子ダブル・コロン（::）があります。この演算子は，通常のC演算子のどれよりも高い優先順位を持ち，次のように，この演算子の前後にある式から定数のfarアドレスを生成するために使われます。

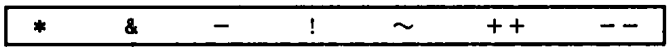
0x1234 : : 0x5678

_DS1 : : _BW

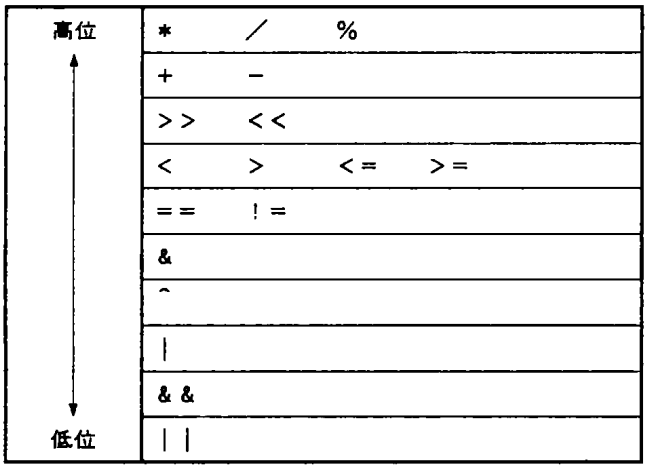
次の一次式演算子は，左から右に優先順位が高くなります。



次の単項演算子の優先順位は，一次演算子よりも低く二項演算子よりも高く，右から左に結合されます。

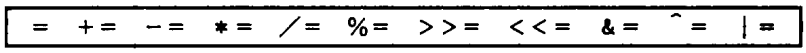


二項演算子の優先順位は降順で，次のようになります（同じ行に複数の演算子がある場合，優先順位は同じです）。



三項演算子？：の優先順位は，二項演算子よりも下位になります。

代入演算子の優先順位は三項演算子よりも下位になり，相互の優先順位は同等で右から左に結合されます。



9.7.6 プログラム内でCの関数を実行する

ソース・コード内で呼び出すのとまったく同様に，Cの式から関数を呼び出すことができます。TDシリーズは，実際には，与えられた関数の引き数を使ってプログラムのコードを実行します。

この方法は，ユーザが書き込んだ関数の動作を手早くテストする際に大変便利です。別の引き数で関数を繰り返し呼び出し，そのつど戻り値が正しいかどうかチェックすることなどが可能です。

9.7.7 副作用のあるCの式

評価の途中でデータ項目の値を変えるCの式を評価すると，副作用が発生します。副作用とは，式の途中に変数の値が変化することをいいます。副作用は，次の３つの場合に発生します。

- 代入演算子（＝，＋＝など）を使用した場合。
- インクリメント／デクリメント演算子（＋＋，－－）を使用した場合。
- 式中で呼びだした関数中で，グローバル変数，静的変数の値を変更した場合。

このような副作用を伴う式を実行すると，変数の値が変更してしまいますので注意してください。

9.7.8 Cの予約語と型変換

TDシリーズでは，Cプログラム内で行うのとまったく同様に，ポインタの型変換を行うことができます。型変換は，丸カッコで囲まれたCデータ型の宣言で構成されます。これは，メモリ・ポインタへと評価される式の前に置いてください。

型変換は，メモリ位置の内容を調べたいときに便利です。メモリ位置は，ダブル・コロン演算子(：：)を使って生成したfarアドレスによって次のように示されます。

```
(long far *)_DS1 :: _BW
```

型変換を使うと，ディバグ情報を生成せずにコンパイルされたモジュール中の，型情報がないプログラム変数にアクセスすることができます。変数の型情報が分かっている場合は，再コンパイルや再リンクを行わなくても，単に変数名の前に型変換を指定することで正しく表示できます。また，Inspectウインドウの Type castローカル・コマンドを使って調べることもできます。

TDシリーズ用の型変換を実行するには，次のようなCの予約語を使います。

● char	● float	● near	● unsigned
● double	● huge	● short	
● enum	● int	● struct	
● far	● long	● union	

9.8 Pascalの式

TDシリーズは，Pascalの式の構文をサポートします。ただし，文字列の連結演算子と集合演算子はサポートしません。

Pascalの式は，シンボル，演算子，文字列，変数，および定数を組み合わせたもので構成されます。次の各項では，式を構成する各構成要素について説明します。

9.8.1 Pascalのシンボル

Pascalのシンボルは，プログラム内のデータ項目やルーチン名です。Pascalのシンボル名は，英字（a-z，A-Z）またはアンダ・スコア（_）で始めます。シンボル名の２文字目以降には，英字のほかに数字（0-9）とアンダ・スコアを使うことができます。

シンボルは，通常Pascalのスコープ規則に従っています。ネストされたローカル・シンボルは，同じ名前をもつほかのシンボルをオーバライドします。ほかのスコープ内のシンボルにアクセスしたい場合は，このスコープをオーバライドできます。詳細は，**9.3 カレント・スコープ外のシンボルへのアクセス**を参照してください。

9.8.2 Pascalの定数と数の表記

定数は，実数（浮動小数点）でも整数定数でもかまいません。負の定数の前には，マイナス符号（－）が付きます。数の定数に，次のように小数点またはe（そのあとの数が指数であることを示す）が付く場合は実数（浮動小数点数）です。

```
123.4          456e34          123.45e-5
```

整数型定数は，通常は10進数です。先頭にドル記号（$）が付く場合は，16進数であることを示します。10進整数定数は，－2，137，483，648から2，147，483，647までの範囲で，16進定数は，$00000000から$FFFFFFFFまでの範囲でなければなりません。

9.8.3 Pascalの文字列

文字列は，次のように単引用符で囲まれた一連の文字です。

```
'abc'
```

10進の制御文字の値の前にシャープ（#）を付けて，次のように文字列に制御文字を記述することができます。

```
'def'#7'xyz'
```

9.8.4 Pascalの演算子の優先順位

TDシリーズは，Pascalの式と同じ演算子を同じ優先順位で使います。

単項演算子は，最も高い優先順位を持ち，相互の優先順位は同等です。

演算子	機　能
@	識別子のアドレスを取り出す
^	ポインタの内容
not	ビットの補数
typeid	型キャスト
+	単項プラス，正
−	単項マイナス，負

二項演算子の優先順位は，単項演算子よりも下位になります。優先順位の高いものから降順で示します（同じ行に複数の演算子がある場合，優先順位は同じです）。

高位	*	/	div	mod	and	shl	shr
↕	in	+	−	or	xor		
低位	<	<=	>	>−	=	<>	

代入演算子（：＝）の優先順位は最も低く，Cの場合と同じように値を返します。

9.8.5 Pascalの関数と手続きの呼び出し

式の中で，Pascalの関数と手続きを参照することができます。たとえば，HalfFuncという関数を宣言しているとすると，Data | Evaluate/Modifyコマンドを選択しHalfFuncを次のようにコールします。

```
HaflFunc(3)
```

手続きを呼び出すこともできます(式の中で呼び出すことはできません)。手続きまたは関数の名前自体を入力すると，そのアドレスと宣言が表示されます。パラメータを持たない関数または手続きを呼び出すには，シンボル名のあとに空のカッコ（）を置きます。

9.9 アセンブラの式

TDシリーズは，アセンブラの式の構文を完全にサポートします。アセンブラの式は，シンボル，演算子，文字列，変数，および定数を組み合わせたもので構成されます。次の各項では，これらの構成要素について説明します。

9.9.1 アセンブラのシンボル

シンボルとは，プログラム内のデータ項目とルーチン名です。アセンブラのシンボル名は，英字（a-z, A-Z）または記号@，?，_，$のどれかで始まります。シンボルの２文字目以降は，これらの文字のほかに0-9の数字を使うことができます。また，シンボル名の中ではなくシンボル名の１字目には，ピリオド（.）を使うこともできます。

特殊記号$は，PS：PCレジスタ・ペアが示すプログラムの現在位置を指します。

9.9.2 アセンブラの定数

定数は，浮動小数点でも整数でもかまいません。浮動小数点定数には10進数の小数点が含まれ，次のように10進表記または指数表記を使うことができます。

1.234　　4.5e+11

整数定数は，その基数をオーバライドするアセンブラ規則を使わないかぎり，16進数とみなされます。

表　記	例	基　数
数字h	0FF00h	16進数
数字o	7100o	8進数
数字q	3777q	8進数
数字d	300d	10進数
数字b	11010111b	2進数

16進数は，必ず０から９までの数字で始まらなければなりません。英字AからFで始まる数を入力するときは，その前に０（ゼロ）を付けてください。

9.9.3 アセンブラの演算子の優先順位

TDシリーズは，アセンブラのほとんどの演算子をサポートします。次に，サポートされる演算子を優先順位の高い順に示します。

高	xxx PTR (BYTE PTRなど)
↑	. (構造体のメンバ・セレクタ)
	: (セグメント・オーバライド)
	OR XOR
	AND
	NOT
	EQ NE LT LE GT GE
	+ −
	* / MOD SHR SHL
	単項+ 単項−
↓	OFFSET SEG
低	() []

変数は，次のように代入演算子＝を使って変更できます。

```
a = [BYTE PTR DS0 : 4]
```

9.10 書式制御

表示させる式を与えると，そのデータ型に基づく書式で表示されます。特定のデータ型に不適切な書式制御は無視されます。

ある式のディフォールトの表示書式を変更したい場合は，式の最後にカンマを置き，繰り返し回数（オプション）と書式文字（オプション）を与えてください。ポインタまたは配列に対してのみ，繰り返し回数を与えることもできます。

文 字	書 式
c	文字または文字列の式をそのまま表示します。制御文字などの表示できない文字は，スペースで表示されます。
d	整数を10進数として表示します。
f [#]	指定した桁数を浮動小数点形式で表示します。桁数を指定しないと，必要な桁数が使用されます。
m	メモリを参照する式を16進数バイトとして表示します。
md	メモリを参照する式を10進数バイトとして表示します。
P	ポインタの値をそのまま表示します。可能であれば，セグメントをレジスタ名として示します。また，ポインタが指し示すオブジェクトも表示します。書式制御を指定しなければ，この書式がディフォールトです。
s	配列または文字配列のポインタを，引用符で囲んだ文字列として表示します。 この文字列の最後にはヌル文字がつけられます。
xまたはh	整数を16進数として表示します。

第10章　C++とオブジェクト指向のディバグ

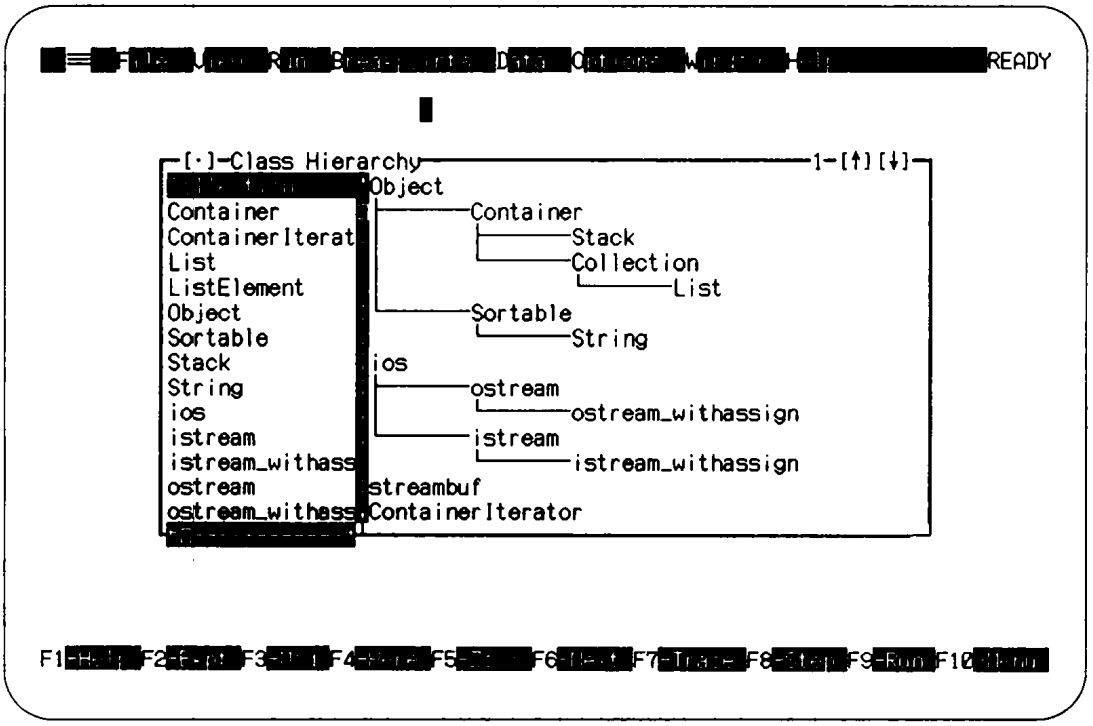
TDシリーズには，Turbo C++やBorland C++を使用して設計された組み込みシステムをディバグする機能があります。

オブジェクト・メソッドやクラス・メンバ関数をトレースしたり，Evaluate/modifyダイアログ・ボックスやWatchesウインドウ内でオブジェクトやクラスを調べることができる拡張機能のほかに，特殊なウインドウ・セットと，オブジェクトとクラス用に特別に設計されたローカル・メニューを備えています。

10.1　オブジェクトやクラスの階層を調べる…Hierarchyウインドウ

TDシリーズは，オブジェクトやクラスの階層を調べるための特殊なウインドウを備えています。View | Hierarchyを選択すると，Hierarchyウインドウがオープンします。

図10－1　Hierarchyウインドウ（View | Hierarchy）



Hierarchyウインドウには，値ではなく，オブジェクトまたはクラスの型に関する情報が表示されます。左側のペインには，ディバグしているモジュールに使われる型がアルファベット順に表示されます。右側のペインには，その階層内のすべてのオブジェクトやクラスが示されます。基本型がペインの左側に，下位型が基本型の右下に表示されます。また，上位と下位の関係を示す線も表示されます。

備考　多重継承が使われているC++プログラムを実行している場合は，右側のペインは２つあります。この場合は，上位型も基本型の右下に表示されます。

10.1.1　オブジェクト型／クラス・リスト・ペイン

左側のペインは，カレント・モジュールが使用するすべてのオブジェクトまたはクラスのアルファベット順リストを表示します。このペインでは，インクリメンタル・マッチ機能を使えるため，項目の多い型リストでもカーソルを移動する必要がありません。ハイライト・バーが左側のペインにあるときは，探しているオブジェクトまたはクラスの型の名前を入力してください。キーを押すたびに，そのときまでに押されたすべてのキーに最初に一致する型をハイライト表示します。

ハイライト表示された型に関するウインドウ（オブジェクト型／クラスInspectウインドウ）をオープンするには，↵を押してください。

(1) オブジェクト型／クラス・リスト・ペインのローカル・メニュー

GRPH-F10を押すと，ペインのローカル・メニューが表示されます。TDINSTBXでCTRLキー・ショート・カットを使用可能にしている場合は，CTRLキーとコマンドの先頭文字のショート・カットを使うことができます。このローカル・メニューには，次の２つのコマンドがあります。

(a) Inspectコマンド

ハイライト表示された型に関するオブジェクト型／クラスInspectウインドウが表示されます。

(b) Treeコマンド

ウインドウの右側の，階層ツリーが表示されているペインに移動して，左側のペインでハイライト表示されていた型にハイライト・バーを置きます。

10.1.2　階層ツリー・ペイン

右側のペインには，現在のモジュールで使われているすべてのオブジェクトまたはクラスの階層ツリーが表示されます。上位型と下位型の関係は，下位型が上位型の右下方向に線で示されます。

単一のオブジェクトまたはクラスの型を複雑な階層ツリーで見つけるには，左側のペインに戻って，インクリメンタル検索機能を使ってください。ローカル・メニューからTreeコマンドを選択して階層ツリーに移動すると，一致した型がハイライト・バーで表示されます。↵を押すと，ハイライト表示された型に関するオブジェクト型／クラスInspectウインドウがオープンします。

(1) 階層ツリー・ペインのローカル・メニュー

階層ツリー・ペインのローカル・メニュー（右ペインの中でGRPH-F10を押す）には，Inspectコマンド１つしかありません。Inspectを選択すると，ハイライト表示された型について，オブジェクト型／クラスInspectウインドウが表示されます。ローカル・メニューからInspectコマンドを選択し

なくても，調べたい型にハイライト・バーを移動して↵を押せば，オブジェクト型／クラスInspectウインドウがオープンします。

多重継承を持つクラスを使ったC++プログラムをロードすると，階層ツリー・ペインのローカル・メニューにもう1つParentsコマンドが含まれています。これにより，親ツリー・ペイン内でクラスの上位型を表示するかどうかを選択します。調べているクラスに多重継承があるときに便利です。

Parentsのディフォールト値はYesです。

10.1.3　親ツリー・ペイン

多重継承を持つクラスを使ったC++プログラムをロードすると，Hierarchyウインドウの階層ツリー・ペインの下に，第3のペインである親ツリー・ペインが表示されます。調べているクラスに複数の上位型があり，階層ツリー・ペインのローカル・メニューのParentsコマンドをYesに設定すると，親ツリー・ペインに逆方向のツリーが表示されます。ペインの左上にメッセージParents of Classが表示され，上位型／下位型を示す線で右下方向に上位型が表示されます。

階層ツリー・ペインの場合と同様に，親ツリー・ペインに表示されるすべてのクラスについて，オブジェクト型／クラスInspectウインドウをオープンすることができます。

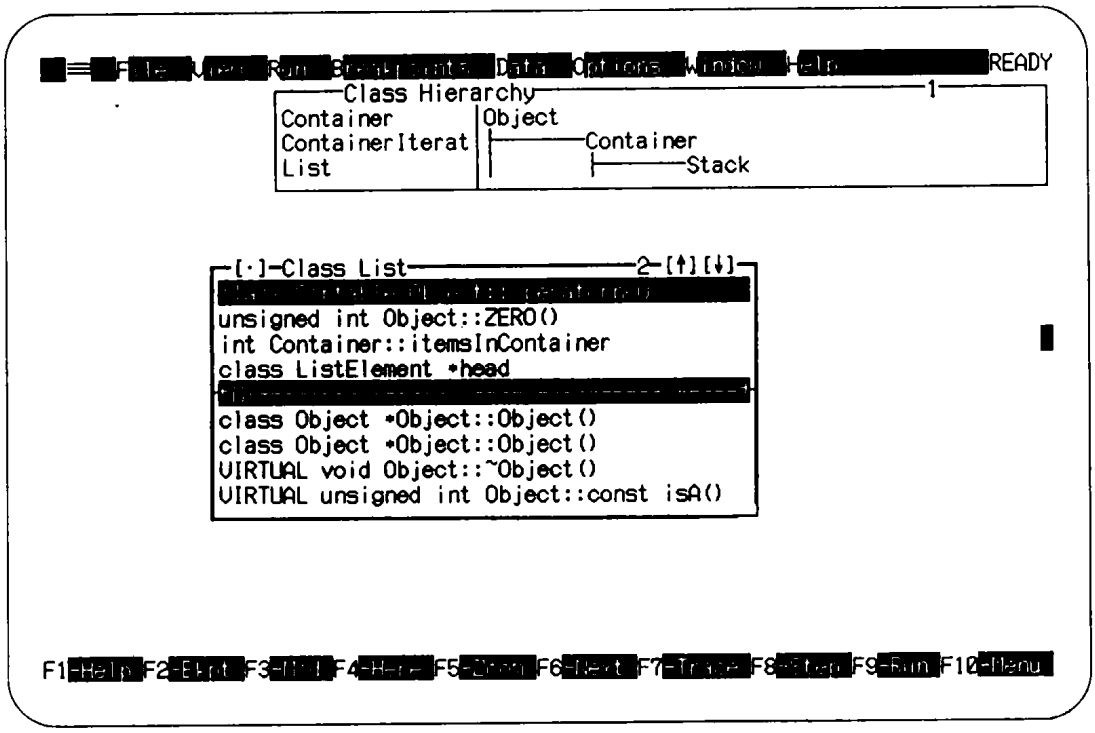
(1) 親ツリー・ペインのローカル・メニュー

親ツリー・ペインのローカル・メニューには，Inspectコマンド1つだけがあります。このコマンドは，階層ツリー・ペインのローカル・メニュー内のInspectコマンドと同じように機能します。またこのコマンドは，ハイライト表示されたオブジェクトの型またはクラスの型に関するInspectウインドウをオープンします。

10.2　オブジェクト型／クラスInspectウインドウ

オブジェクト型／クラスInspectウインドウは，オブジェクトの型の詳細を調べる特別なInspectウインドウです。このウインドウは，型情報の要約を表示しますが，特定のインスタンスを参照しません。

図10－2　オブジェクト型／クラスInspectウインドウ



このウインドウは上下２つのペインに分かれています。

- ●オブジェクト・データ・フィールド・ペイン（上のペイン）…型のデータ・フィールドやメンバを表示します。
- ●オブジェクト・メソッド・ペイン（下のペイン）…メソッドまたはメンバ関数名を表示します。選択された項目が手続きではない関数のときは，関数の戻り値の型も表示されます。

これら２つのペイン間の移動にはTABキーを用います。

(1) ライブラリまたはクラスのネスト構造体を調べる

ハイライト表示されたデータ・フィールドがオブジェクトかクラスの型，またはそのポインタであるとき，↵を押すとハイライト表示された型に対してさらにもう１つのオブジェクト型／クラスInspectウインドウを開くことができます（このペインについてのローカル・メニューからInspectコマンドを選択しても同様のことが行えます）。これにより容易にライブラリまたはクラスのネスト構造体を調べることができます。

(2) パラメータを調べる

メソッドかメンバ関数をハイライト表示して↵を押すと，メソッド／メンバ関数inspectウインドウを開くことができます。このウインドウの最上段のペインには，選択したメソッドかメンバ関数の，オブジェクトかクラスの型の実現部のコード・アドレス，およびそのすべてのパラメータの名前と型を表示します。

メソッド／メンバ関数Inspectウインドウ内のどこかで↵を押すと，調べているメソッドまたはメンバ関数をインプリメントしているコードにカーソルが置かれた状態で，Moduleウインドウが最前面に表示されます。

標準のInspectウインドウについては，ESCキーを押すと現在のInspectウインドウがクローズし，GRPH-F3を押すとすべてのInspectウインドウがクローズします。

10.2.1　オブジェクト型／クラスInspectウインドウのローカル・メニュー

GRPH-F10を押すと，どちらか一方のペインのローカル・メニューが表示されます。CTRLキー・ショート・カットがTDINSTBXで使用可能に設定されていれば，CTRLキーとメニュー・コマンドの先頭文字を押して，ローカル・メニュー・コマンドを選択できます。

(1) オブジェクト・データ・フィールド・ペインのローカル・メニュー

(a) Inspectコマンド

ハイライト表示されたフィールドが，オブジェクトかクラスの型またはそのポインタであれば，ハイライト表示されたフィールドに関してさらにオブジェクト型／クラスInspectウインドウがオープンします。

(b) Hierarchyコマンド

調べているオブジェクトまたはクラスの型に関するHierarchyウインドウをオープンします。

(c) Show inheritedコマンド

このコマンドのディフォールト値はYesです。この場合，調べているオブジェクトまたはクラスの型の中で定義されているか，上位型から継承されたものであるかに関わらず，すべてのデータ・フィールドまたはメンバが表示されます。Noに設定すると，調べている型の中で定義されたフィールドまたはメンバのみが表示されます。

(2) オブジェクト・メソッド・ペインのローカル・メニュー

(a) Inspectコマンド

ハイライト表示された項目に関するメソッド／メンバ関数Inspectウインドウをオープンしま

す。このウインドウに表示されているアドレスにカーソルを合わせてCTRL-Iを押すと，調べている項目をインプリメントしているコードにカーソルが置かれた状態で，Moduleウインドウが前面に表示されます。

(b) Hierarchyコマンド

調べているオブジェクトまたはクラスの型に関するHierarchyウインドウをオープンします。

(c) Show inheritedコマンド

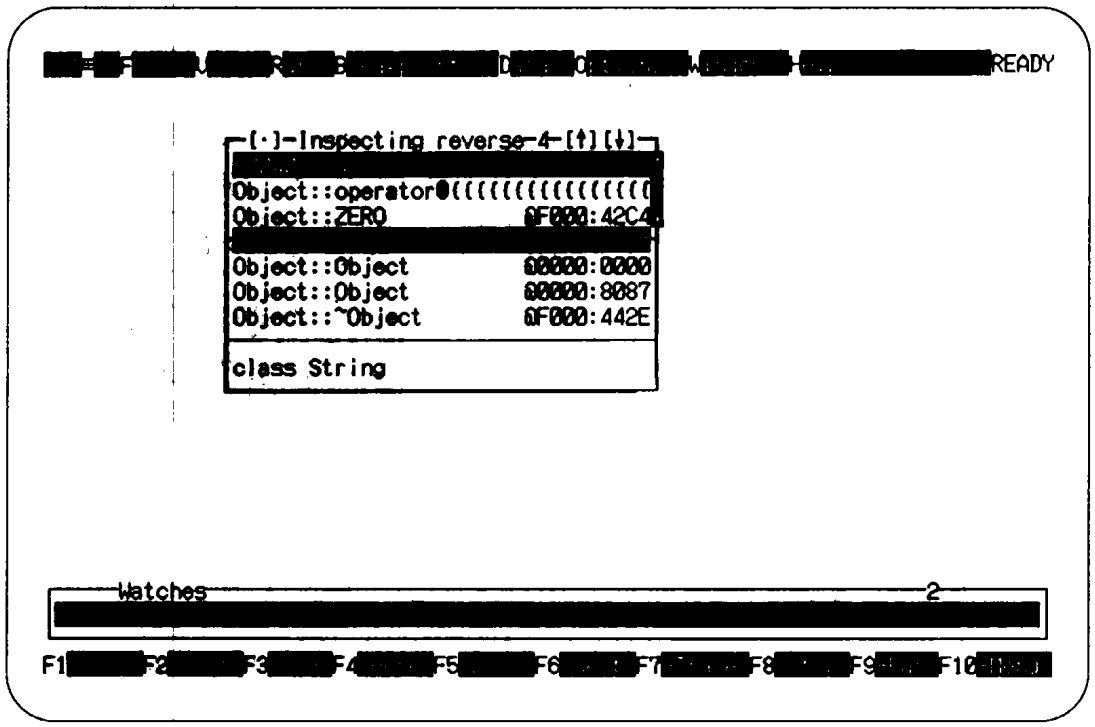
このコマンドのディフォールト値はYesです。この場合，調べている型の中で定義されたものであるか，上位型から継承されたものであるかに関わらず，すべてのメソッドまたはメンバ関数が表示されます。Noに設定すると，調べているオブジェクト型の中で定義されたメソッドまたはメンバ関数のみが表示されます。

10.3　オブジェクト／クラス・インスタンスInspectウインドウ

Moduleウインドウ内のオブジェクトまたはクラスの値(インスタンス)にカーソルを合わせてCTRL-Iを押すと，このウインドウがオープンします。

オブジェクト型／クラスInspectウインドウは，プログラム実行中の特定の時刻における特定のオブジェクトまたはクラスのインスタンスに含まれるデータについては何も表示しません。このウインドウは，オブジェクトとクラスのインスタンスを調べるために，特に拡張された形式の分かりやすいデータ・レコードInspectウインドウです。

図10-3　オブジェクト／クラス・インスタンスInspectウインドウ



オブジェクト／クラス・インスタンスInspectウインドウには，次の３つのペインがあります。

- 上のペイン…レコードのフィールドまたはメンバの型と，その現在値が表示されます。
- 中間のペイン…インスタンスのメソッドまたはメンバ関数，およびそれぞれのコード・アドレスが表示されます（コード・アドレスは，多態性オブジェクトとVMTが考慮されています）。
- 下のペイン…上のペインでハイライト表示されたフィールドまたはメンバの型が表示されます。

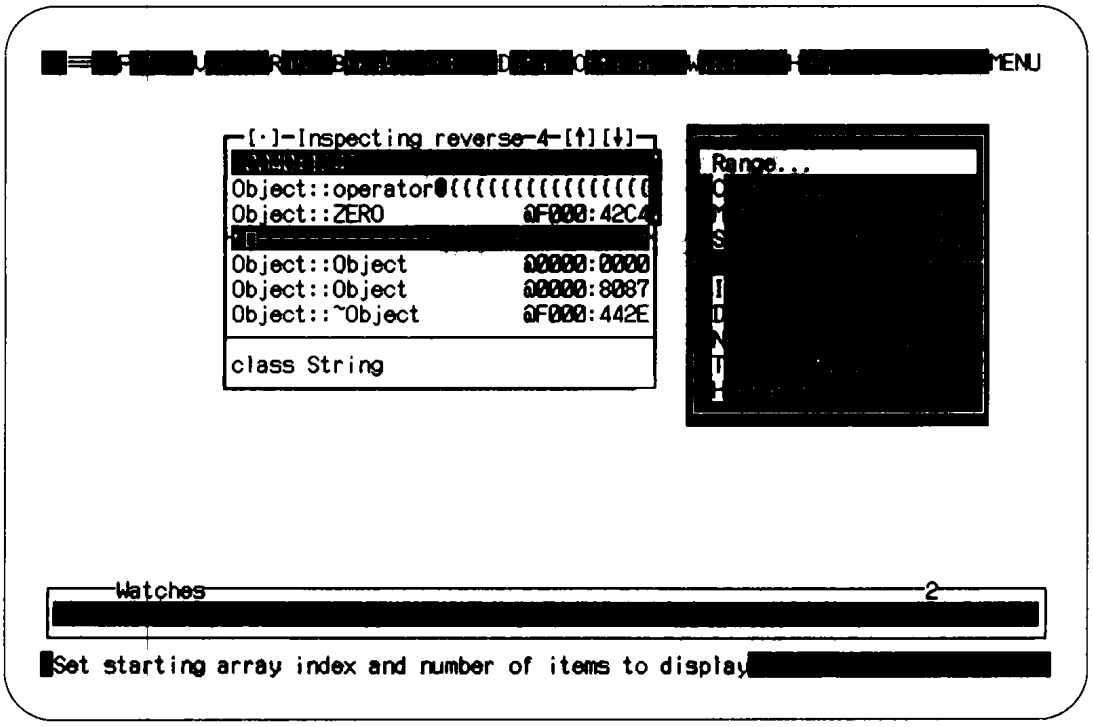
TDシリーズのほとんどのデータ・レコードInspectウインドウは，この３つのペインのうち上下２つのペインを持っています。

10.3.1　オブジェクト／クラス・インスタンスInspectウインドウのローカル・メニュー

オブジェクト／クラス・インスタンスInspectウインドウの，上と中間の２つのペインには，それぞれローカル・メニューがあり，そのペイン内でGRPH-F10を押すと表示されます。TDINSTBXでCTRLキー・ショート・カットを使用可能にしてあれば，CTRLキーとメニュー・コマンドの先頭文字を押して，個々のコマンドを選択できます。

データ・レコードInspectウインドウでの場合と同様に，下のペインはハイライト表示されたフィールドの型を表示するだけで，ローカル・メニューはありません。

図10－4　オブジェクト／クラス・インスタンスInspectウインドウのローカル・メニュー



(1) 上のペイン

上のペインには，選択した項目のデータ・フィールドまたはメンバがまとめられています。上のペインのローカル・メニューを次に示します。

(a) Range...コマンド

配列項目の範囲を表示します。調べたい項目が配列またはポインタでなければ，その項目をアクセスできません。

(b) Change...コマンド

このコマンドを選択すると，ハイライト表示されたデータ・フィールドまたはメンバに新しい値をロードすることができます。

(c) Methodsコマンド

このコマンドはYes/Noのトグルです。Yesがディフォールトです。その場合は，中間のペインにメソッドまたはメンバ関数が表示されます。Noに設定すると，中間のペインは表示されません。このトグルは，次のInspectウインドウがオープンされるまで記憶されています。

(d) Show inheritedコマンド

このコマンドはYes/Noのトグルです。Yesに設定すると，調べている型の中で定義されてい

るか上位型から継承されたものであるかに関わらず，すべてのデータ・フィールドとメンバまたはすべてのメソッドとメンバ関数が表示されます。Noに設定すると，調べている型の中で定義されたフィールドとメソッドのみが表示されます。

(e) Inspectコマンド

このコマンドを選択すると，ハイライト表示されたフィールドまたはメンバに関するInspectウインドウがオープンします。ハイライト表示されたフィールドまたはメンバにカーソルを合わせて ↵ を押しても，同じ結果になります。

(f) Descendコマンド

ハイライト表示された項目は，現在のInspectウインドウ内の項目の位置を占めます。新しいInspectウインドウはオープンされません。ただし，Inspectオプションを使った場合のように，以前に調べたフィールドに戻ることはできません。

(g) New expression...コマンド

このコマンドを選択すると，調べたい新しいデータ項目または式を入力するよう求めるプロンプトが表示されます。新しい項目は，ウインドウ内で現在の項目に置き換わります。別のウインドウは開きません。

(h) Type cast...コマンド

調べている項目についてのデータ型（Byte, Word, Int, Charポインタ）を指定できます。

このコマンドは，Inspectウインドウに型情報のないシンボルが含まれているときや，型のないポインタに対して型を明示的に設定するシンボルが入っているときに便利です。

(i) Hierarchyコマンド

このコマンドを選択するとHierarchyウインドウがオープンします。

(2) 中間のペイン

中間のペインには，あるオブジェクトのメソッドまたはあるクラスのメンバ関数がまとめられています。オブジェクト・メソッド・ペインのローカル・メニューや上のペインのローカル・メニューと異なるのは，Changeコマンドがないという点だけです。ただし，データ・フィールドやクラス・メンバと違って，メソッドとメンバ関数は実行中に変更できないので，Changeコマンドは必要ありません。

〔メ　モ〕

第11章　アセンブラ・レベルのディバグ

この章では，次の項目について説明します。

- ソース・ディバグでは不十分なときの，アセンブラ・レベルでのディバグの方法
- 16進バイト・データの表示，変更方法
- スタック内容の表示方法
- CPUレジスタ，CPUフラグの表示，変更方法

11.1　ソース・ディバグでは不十分なとき

プログラムのディバグは，その大部分をソース・レベルのデータとコードの参照に費やします。この場合，シンボル名をソース・コード内に入力したとおりに参照し，ソース・コードの一部を実行してプログラムを進行させます。

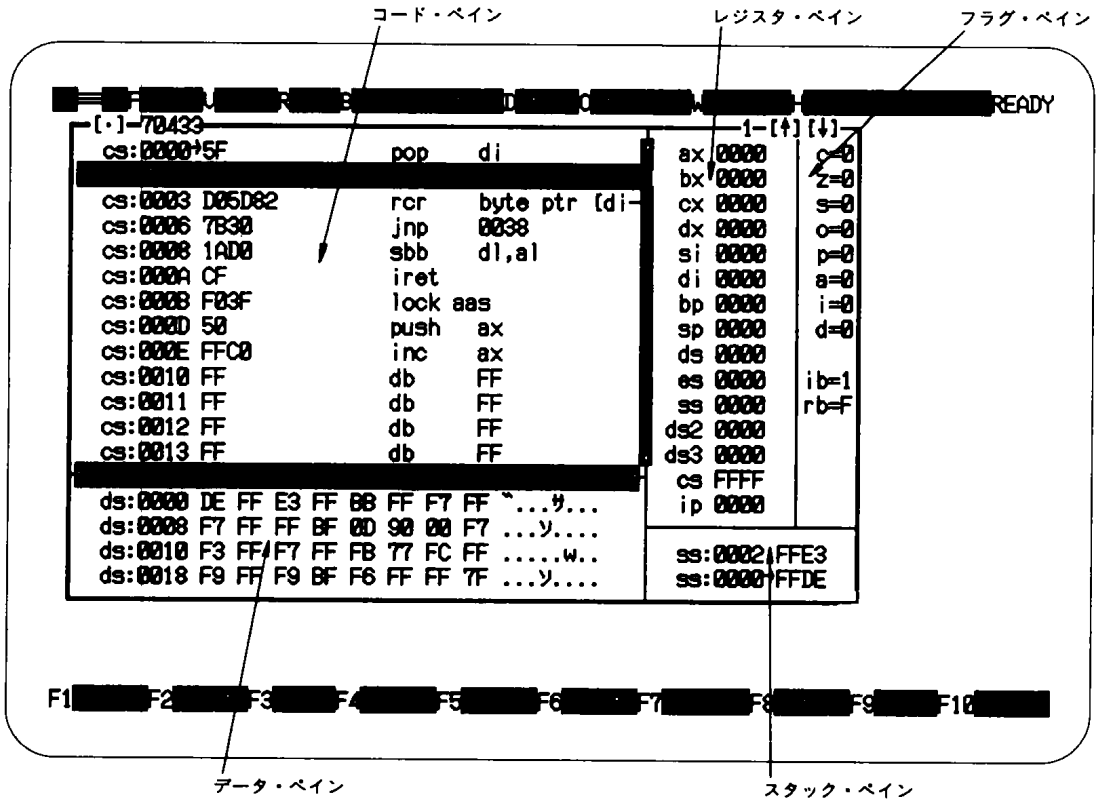
ただし，コンパイラが生成する正確な命令，CPUレジスタの内容，およびスタックの内容を調べると，問題が明らかになる場合があります。V55SC，V55PIとコンパイラが，ソース・コードをどのようにしてマシン・コードに翻訳するかを理解したうえでこの方法を使用してください。

ソース・コードの各行ごとに生成される命令を調べると，コンパイラがソース・コードをどのように機械語命令に翻訳するかが分かります。たとえば，CとPascalでは，多数のアクションを一度に実行するソース・コードの行を複数書くことができます。TDシリーズでは，一度に1式ずつではなく，一度に1ソース行ずつ進むことができます。1ソース行の一部の実行結果だけを知りたいこともありますが，その場合はプログラムを一度に1機械語命令ずつ進めて，中間結果を調べます。ただし，それにはコンパイラがソース文を機械語命令にどのように翻訳したかを理解する必要があります。

11.2　CPUウインドウ

CPUウインドウには，CPU全体の状態が表示されます。プログラムのコードやデータを構成するビットとバイトを調べたり変更したりすることができます。また，コード・ペインに組み込まれたアセンブラを使って，アセンブラのソース文を入力するのと同じように，命令を入力してプログラムを一時的にパッチすることができます。なお，どんなデータ構造でも，それを構成するバイトにアクセスして，それぞれの書式で表示，変更ができます。

図11－1　CPUウインドウ（View | CPU）



メニュー・バーでView | CPUを選択すると，CPUウインドウがオープンします。現在のウインドウに表示されている内容に応じて，適切なコード，データ，またはスタック位置にあった新しいCPUウインドウが表示されます。これは，現在カーソルが置かれているコード，データ，またはスタック位置を，低レベル（インストラクション・レベル）で調べる便利な方法です。

次に，CPUコマンドを選択したときカーソルが置かれる位置を示します。

現在のウインドウ	CPUウインドウ・ペイン	カーソル位置
Stack	Stack	カレントSS：SP
Module	Code	カレントPS：PC
Variables	Data[注]	項目のアドレス
Inspector	Data	項目のアドレス
Breakpoints （グローバルでないとき）	Code	ブレークポイントのアドレス

注　ウインドウ内の項目がルーチンであればコード・ペイン

11.3　CPUウインドウのペイン

CPUウインドウには次の５つのペインがあります。

●コード・ペイン（左上のペイン）…ソース行と逆アセンブルされたプログラム・コードが一緒に表示されます。

●レジスタ・ペイン（コード・ペインの右のペイン）…CPUレジスタの内容が表示されます。

●フラグ・ペイン（右側のペイン）…８つのCPUフラグの状態が表示されます。

●データ・ペイン（左下のペイン）…選択したメモリ領域の16進ダンプが表示されます。

●スタック・ペイン（右下のペイン）…スタックの内容が表示されます。

あるペインから次のペインに移るには，TABキーまたはSHIFT-TABを押すか，マウスでペインをクリックしてください。CPUウインドウのタイトル行にはプロセッサの種類が表示されます。

コード・ペインの中では，矢印（→）はプログラムの現在位置（PS：PC）を示します。スタック・ペイン内では，矢印（→）は現在のスタック・ポインタ（SS：SP）を示します。

コード・ペイン内でハイライト表示された命令がメモリ位置を参照するときは，メモリ・アドレスとその現在の内容がCPUウインドウの最上段の行に表示されます。これにより，命令オペランドが指し示すメモリ内の場所と，次に読み込まれるか書き込まれる値を見ることができます。

すべてのウインドウやペインと同様に，GRPH-F10を押すと，コード・ペインのローカル・メニューが表示されます。また，CTRLキー・ショート・カットが使用可能であれば，CTRLキーとメニュー・コマンドの先頭文字を押してそのコマンドにアクセスできます。

コード・ペイン，データ・ペイン，およびスタック・ペインの中では，CTRLキーと→または←キーを押してそのペインの表示開始アドレスを１バイトずつ上または下にシフトできます。表示を少しだけ調節したい場合，Gotoコマンドよりも簡単に行えます。

11.3.1　コード・ペイン

このペインには，ユーザが選択したアドレスで逆アセンブルされた命令が表示されます。

逆アセンブルされた各行の左側の部分には，命令のアドレスが表示されます。このアドレスは，16進数のセグメントとオフセットとして表示されます。また，セグメント値が現在のPSレジスタの値と等しければ，セグメント値はPSレジスタ名に置き換えられます。ウインドウの幅が十分に広ければ（ズームされているか，サイズが再設定されている場合），命令を構成するバイトが表示されます。逆アセンブルされた命令は，その右側に表示されます。

(1) 逆アセンブラ

コード・ペインは，プログラムの命令を自動的に逆アセンブルして表示します。アドレスがグローバル・シンボル，静的シンボル，または行番号に対応する場合は，Mixed displayモードがYesに設定されていれば，そのシンボルが逆アセンブルされた命令の前の行に表示されます。また，シンボルのアドレスに対応するソース・コード行があれば，それがシンボルのあとに表示されます。

グローバル・シンボルは，単にシンボル名として表示されます。静的シンボルは，最初にモジュール名，次にシャープ記号（#）またはピリオド（.），最後に静的シンボル名という形式で表示されます。行番号は，最初にモジュール名，次にシャープ記号（#）またはピリオド（.），最後に10進

表記の行番号という形式で表示されます。

イミーディエト・オペランドが表示されている場合は，そのサイズを桁数から推測することができます。1バイトの即値は2桁で，1ワードの即値は4桁です。

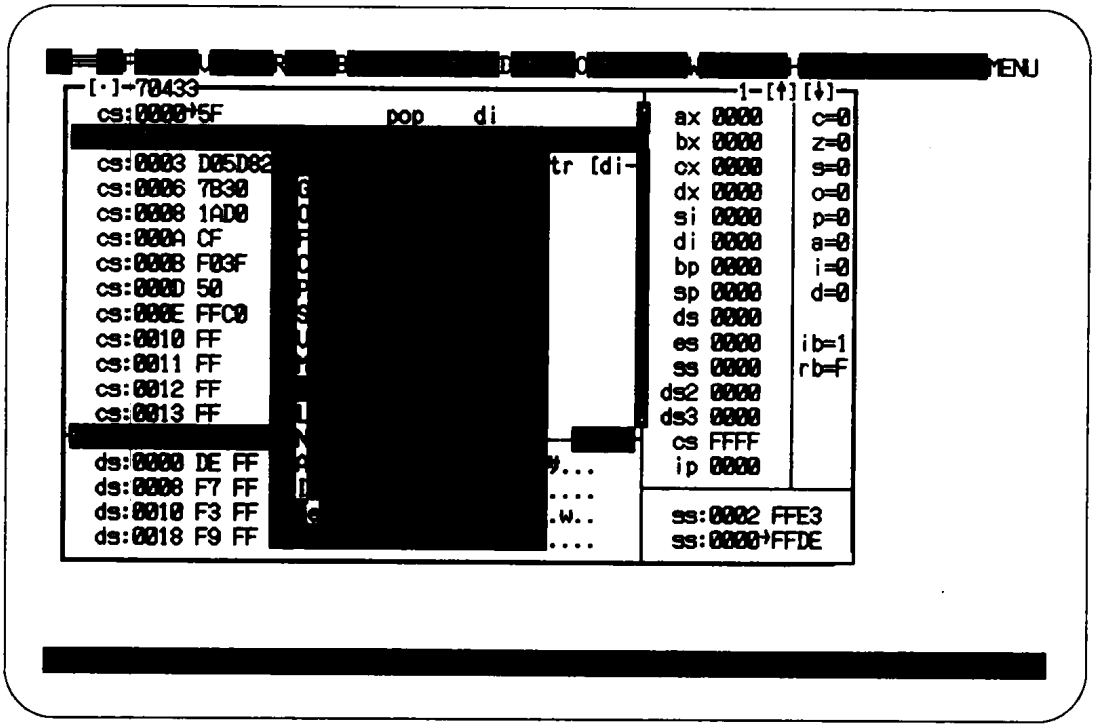
命令ニモニックRETFは，この命令がfarリターン命令であることを示します。通常のRETニモニックは，nearリターン命令を示します。

可能であれば，BR命令とCALL命令の分岐先がシンボルで表示されます。PS：PCがBR命令または条件付きブランチ命令であれば，実行命令によってジャンプが発生する場合にのみ，ジャンプの方向を示す矢印(↑または↓)が表示されます。また，MOV，ADD，その他の命令に使われるメモリ・アドレスは，シンボル・アドレスで表示されます。

(2) コード・ペインのローカル・メニュー

TABキーまたはSHIFT-TABを押してコード・ペインに移動してください。次にGRPH-F10を押すと，ローカル・メニューが表示されます。

図11－2　コード・ペインのローカル・メニュー



(a) Gotoコマンド

このコマンドを選択すると，移動先の新しいアドレスを入力するよう求めるプロンプトが表示されます。アドレスの入力については第9章　式を参照してください。

このコマンドを使用する前の位置にコード・ペインを戻すには，Previousコマンドを使います((e) Previousコマンド参照)。

(b) Originコマンド

PS：PCレジスタ・ペアが示すプログラムの現在位置に戻します。このコマンドは，最初の位置に戻りたいときに便利です。

このコマンドを使用する前の位置にコード・ペインを戻すには，Previousコマンドを使います（**(e) Previousコマンド参照**）。

(c) Followコマンド

現在ハイライト表示されている命令のデスティネーション・アドレス（分岐先）に移動します。命令の分岐先のコードを表示するように，コード・ペインが移動します。条件ジャンプの場合は，ジャンプが発生したかのようにアドレスが表示されます。

このコマンドは，CALL，BR，条件付きブランチ（BZ，BNE，DBNZ，BCWZなど），およびBRK命令とともに使うことができます。

このコマンドを使用する前の位置にコード・ペインを戻すには，Previousコマンドを使います（**(e) Previousコマンド参照**）。

(d) Callerコマンド

現在の割り込みルーチンまたはサブルーチンを呼び出した命令に移動します。

このコマンドは，スタック操作命令によりスタック・ポインタが変更されたあとでは機能しません。たとえば，割り込みルーチンやサブルーチンがデータをスタックに退避してしまうと，TDシリーズはルーチンがどこから呼び出されたのか判断できません。

このコマンドを選択する前の位置にコード・ペインを戻すには，Previousコマンドを使います（**(e) Previousコマンド参照**）。

(e) Previousコマンド

コード・ペインを，表示アドレスを明示的に変更した最後のコマンドの前のアドレスに戻します。カーソル・キーまたはROLLDOWN，ROLLUPを使った場合には，位置は記憶されません。

Previousコマンドを選択すると，コード・ペインの位置が記憶されます。このため，このコマンドを繰り返し使うことにより，コード・ペインの表示を２つのアドレス間で交互に切り替えることができます。

(f) Searchコマンド

検索したい命令またはバイト・リストを入力できます。Assembleコマンドを使うときと同じように命令を入力してください。バイト・リストの入力については，**第9章　式**を参照してください。

目的のものが見つかるか，セグメントの最後に到達するか，またはCTRL-STOPを押すまで，検索は続行します。

(g) Mixedコマンド

逆アセンブルされた命令とソース・コードの表示方法には次の３つがあり，クリックするごとに選択する項目が変わります。

- ●No　ソース・コードは表示されません。逆アセンブルされた命令だけが表示されます。
- ●Yes　そのソース・コードについて最初に逆アセンブルされた命令の前に，ソース・コード行が表示されます。現在のモジュールが高水準言語のソース・モジュールであれば，ペインがこの表示モードに設定されます。
- ●Both　ソース・コード行が，対応するソース・コードを持つ行の逆アセンブルされた行に置き換えられます。対応するソース・コードがなければ，逆アセンブルされた命令が表示されます。

  このモードは，アセンブラ・モジュールのディバグ中に，対応する逆アセンブルされた命令ではなく元のソース・コードを見たいときに使ってください。現在のモジュールがアセンブラ・ソース・モジュールであれば，ペインはこの表示モードに設定されます。

(h) New PS：PCコマンド

プログラムのロケーション・カウンタ（PS：PCレジスタ）を，現在ハイライト表示されているアドレスに設定します。プログラムを再実行すると，このアドレスから実行が始まります。このコマンドは，コードの一部を実行せずにスキップしたいときに便利です。

注意　このコマンドの使用時に，スタックが現在のPS：PCと異なる状態にある位置に，PS：PCを合わせて調整すると，必ずプログラムが暴走します。したがって，PS：PCを現在のルーチン外のアドレスに設定するときは，このコマンドを使わないでください。

(i) Assemble...コマンド

命令をアセンブルして，現在ハイライト表示されている位置にある命令と置き換えます。アセンブルしたい命令を入力するよう求めるプロンプトが表示されます。

アセンブルしたい文を入力するだけでも，このコマンドを起動できます。この方法を使うと，Assembleを指定したときと同じように，ダイアログ・ボックスが表示されます。

(j) I/Oコマンド

CPUのI/O空間からの読み込みまたは書き込みを行います。カード上のI/Oレジスタの内容を調べたり，そこに値を書き込むことができます。このコマンドのプルダウン・メニューを次に示します。

●**In byteコマンド**

I/Oポートから1バイトを読み込みます。プロンプトの後ろに，値を調べたいI/Oポート名を入力します。

●**Out byteコマンド**

I/Oポートに1バイトを書き込みます。プロンプトの後ろに，書き込みたいI/Oポート名と書き込みたい値を入力します。

●**Read wordコマンド**

I/Oポートから1ワードを読み込みます。プロンプトの後ろに，値を調べたいI/Oポート名を入力します。

●**Write wordコマンド**

I/Oポートに1ワードを書き込みます。プロンプトの後ろに，書き込みたいI/Oポート名と書き込みたい値を入力します。

注意　**I/Oコマンドの使用時に，ディバグしているプログラムやデバイスが正常に動作しなくなることがあります。これは，IN命令とOUT命令が，周辺装置のコントローラ（シリアル・ポート，ディスク・コントローラ，CRTコントローラなど）があるI/O空間にアクセスするためです。I/Oデバイスの中には，そのポートから読み込みが行われると，ステータス・ビットをリセットしたり，ポートに新しいデータ・バイトをロードするなど，何らかのアクションを実行するものがあるので注意が必要です。**
ディバグ中にこのような状態になった場合は，ターゲット・システムをリセットし，RUN | ICE resetコマンドを実行してください。

備考　V55SC，V55PIの内部周辺レジスタに関係したI/Oポートを操作するには，View | Targetコマンドの方が便利です。TDシリーズが，ユーザに代わってこのレジスタ・アドレスを管理します。

(k) Memory mapped I/Oコマンド

メモリ・マップトI/O空間からの読み込みまたは書き込みを行います。このコマンドのプルダウン・メニューを次に示します。

●**In byteコマンド**

メモリ・マップトI/Oポートから1バイトを読み込みます。プロンプトの後ろに，値を調べたいMemory Mapped I/Oポート名を入力します。

●**Out byteコマンド**

メモリ・マップトI/Oポートに1バイトを書き込みます。プロンプトの後ろに，書き込みたいMemory Mapped I/Oポート名と書き込みたい値を入力します。

●**Read wordコマンド**

メモリ・マップトI/Oポートから1ワードを読み込みます。プロンプトの後ろに，値を調べたいMemory Mapped I/Oポート名を入力します。

●**Write wordコマンド**

メモリ・マップトI/Oポートに1ワードを書き込みます。プロンプトの後ろに，書き込みたいMemory Mapped I/Oポート名と書き込みたい値を入力します。

11.3.2　レジスタ・ペインとフラグ・ペイン

レジスタ・ペインは，コード・ペインの右側に表示されます。CPUレジスタの内容を表示します。

フラグ・ペインは，CPUウインドウの右上に表示されます。12のCPUフラグの状態と現在選択されているバンク番号を表示します。

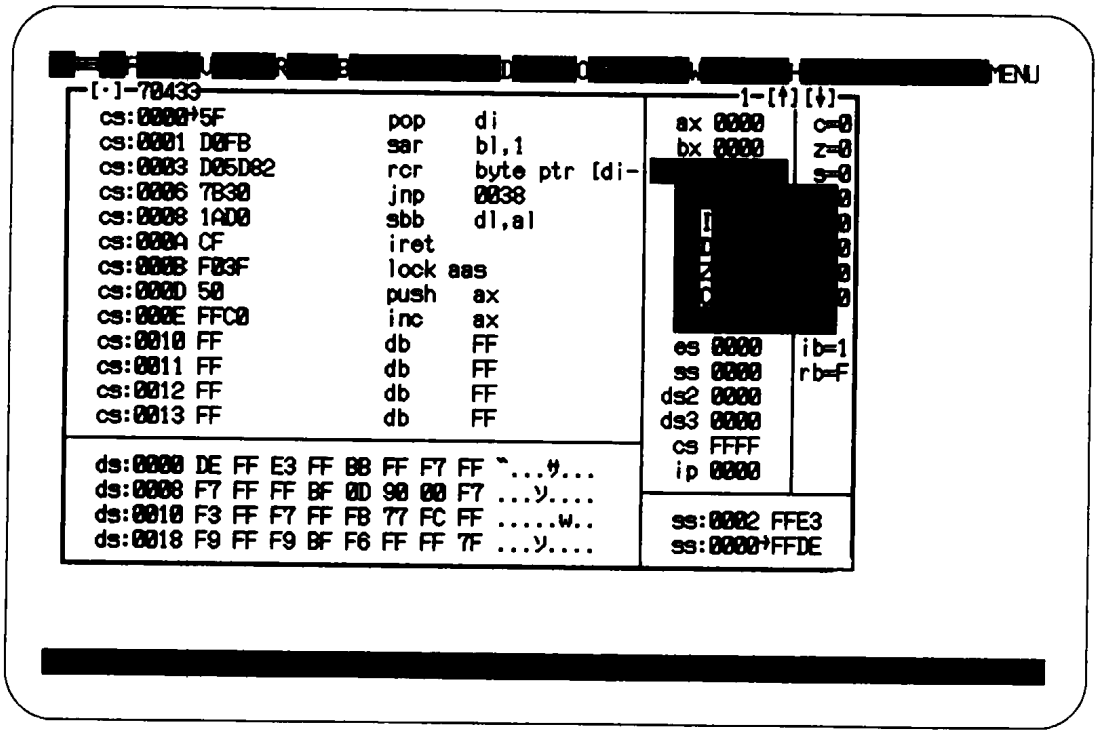
次に，各フラグがフラグ・ペイン内にどのように表示されるかを示します。

ペインの中の文字	フラグ名
c	キャリー
z	ゼロ
s	サイン（符号）
o	オーバフロー
p	パリティ
a	補助キャリー
i	割り込み許可
d	ディレクション
ib	I/Oブレーク
f0	ユーザ・フラグ0
f1	ユーザ・フラグ1
rb	レジスタ・バンク

(1) レジスタ・ペインのローカル・メニュー

GRPH-F10を押すと，レジスタ・ペインのローカル・メニューがポップアップします。また，CTRLキー・ショート・カットが使用可能であれば，CTRLキーと目的のコマンドの先頭文字を押して，そのコマンドにアクセスできます。

図11－3　レジスタ・ペインのローカル・メニュー



(a) Incrementコマンド

現在ハイライト表示されているレジスタの値に1を加えます。これは，1だけずれているバグを修正するなど，レジスタ値の調整が少ししかないときに便利です。

(b) Decrementコマンド

現在ハイライト表示されているレジスタの値から1を引きます。

(c) Zeroコマンド

現在ハイライト表示されているレジスタの値を0に設定します。

(d) Change...コマンド

現在ハイライト表示されているレジスタの値を変更します。新しい値の入力を求めるプロンプトが表示されます。新しい値の入力には，式評価機能を十分に活用できます。

また，レジスタの新しい値を入力するだけでこのコマンドを起動できます。Changeコマンドを指定したときと同じように，ダイアログ・ボックスが表示されます。

(2) フラグ・ペインのローカル・メニュー

GRPH-F10を押すと，フラグ・ペインのローカル・メニューがポップアップします。また，CTRLキー・ショート・カットが使用可能であれば，CTRLキーと目的のコマンドの先頭文字を押して，そ

のコマンドにアクセスできます。

(a) Toggleコマンド

フラグの値が1であれば0に，0であれば1に設定します。0はクリア，1はセットを示します。また，↵を押して，現在ハイライト表示されているフラグの値を切り替えられます。

11.3.3 データ・ペイン

このペインには，選択したメモリ領域がそのまま表示されます。各行の一番左側の部分は，その行に表示されているデータのアドレスを示します。アドレスは，16進数のセグメントとオフセットとして表示されます。また，セグメント値が現在のDS0レジスタと同じであれば，セグメント値がDS0レジスタ名に置き換えられます。

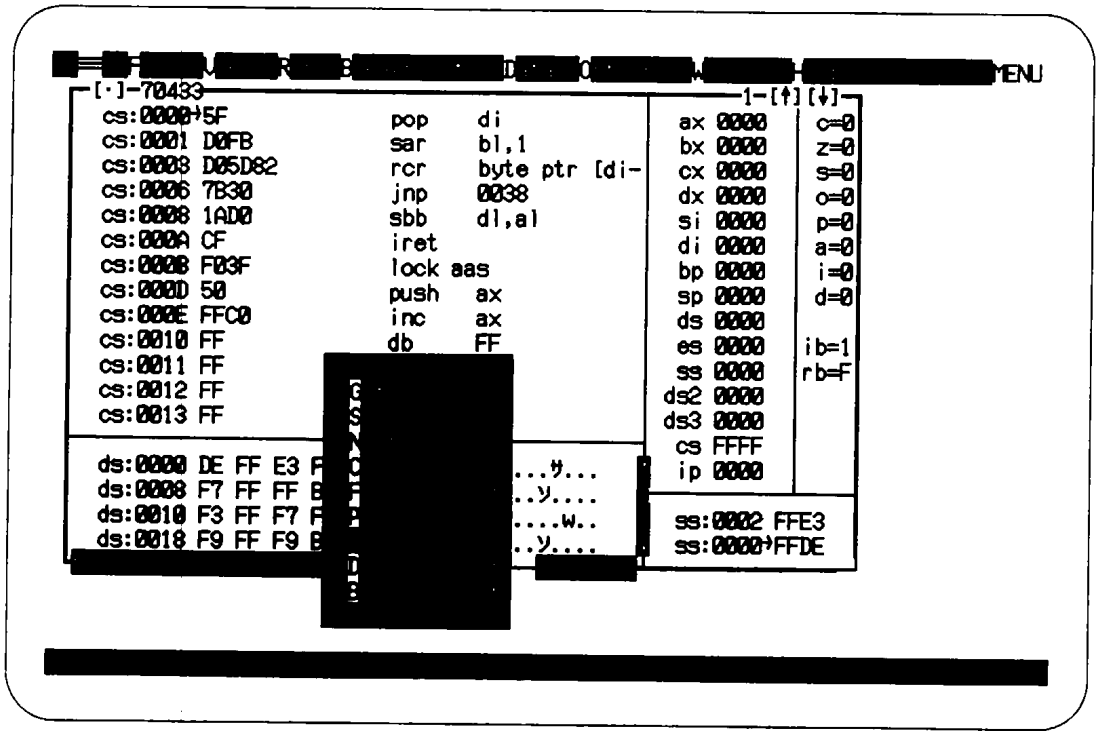
ペインの右の部分には，1つまたは複数のデータ項目がそのまま表示されます。この領域の書式は，Display asローカル・メニュー・コマンドを使って選択した表示形式によって異なります。浮動小数点表示形式（Comp, Float, Real, Double, Extended）のうちの1つを選択すると，浮動小数点数が各行に1つずつ表示されます。バイト形式では各行に8バイトずつ表示され，ワード形式では各行に4ワードずつ表示され，ロング形式では各行に2ロング・ワードずつ表示されます。

データがバイトとして表示される場合，各行の右の部分には，表示されているデータ・バイトに対応する表示文字が表示されます。制御コードなどの表示できない文字は，すべてピリオド(.)で示されます。

(1) データ・ペインのローカル・メニュー

データ・ペイン内にカーソルがあるときは，GRPH-F10を押すとローカル・メニューが現われます。また，CTRLキー・ショート・カットが使用可能であれば，CTRLキーと目的のコマンドの先頭文字を押して，そのコマンドにアクセスできます。

図11-4　データ・ペインのローカル・メニュー



(a) Gotoコマンド

データ内のアドレスに移動します。移動したい新しいアドレスを入力してください。アドレスの入力方法については**第9章　式**を参照してください。

(b) Searchコマンド

カーソル位置で示されるカレント・メモリ・アドレスから，文字列の検索を開始します。検索したいバイト・リストを入力してください。

検索は，セグメントの終わりかCTRL-STOPで終了します。ただし，先頭には戻りません。バイト・リストについては**第9章　式**を参照してください。

(c) Nextコマンド

以前にSearchコマンドで指定したバイト・リストで，次に一致するものを検索します。

(d) Change...コマンド

現在のカーソル位置にあるバイトを変更することができます。表示がASCII形式またはByte形式であれば，バイト・リストを入力するようプロンプトが表示されます。そうでなければ，現在の表示形式で項目を入力するようプロンプトが表示されます。

新しい値を入力するだけでも，このコマンドを起動できます。この方法でも，Changeコマンドを選択したときと同じダイアログ・ボックスが表示されます。

(e) Followコマンド

このコマンドを選択するとメニューが表示され，nearまたはfarポインタの連鎖をたどることができます。

- Near code　このコマンドは，データ・ペイン内のカーソル位置にあるワードが，PSレジスタが指定する現在のコード・セグメントへのオフセットであると解釈します。コード・ペインがアクティブになり，このアドレスが表示されます。
- Far code　このコマンドは，データ・ペイン内のカーソル位置にあるダブル・ワードが，farアドレス（セグメントとオフセット）であると解釈します。コード・ペインがアクティブになり，このアドレスが表示されます。
- Offset to data

  このコマンドを使うと，ワード・サイズ（near，オフセットのみ）・ポインタの連鎖をたどることができます。データ・ペインは，現在のカーソル位置にあるメモリ内のワードで指定されるオフセットに設定されます。
- Segment：offset to data

  このコマンドを使うと，long（far，セグメントとオフセット）ポインタの連鎖をたどることができます。データ・ペインは，現在のカーソル位置にあるメモリ内の２つのワードで指定されるオフセットに設定されます。
- Base segment：0 to data

  このコマンドは，カーソル位置にあるワードをセグメント・アドレスであると解釈して，データ・ペインをそのセグメントの先頭に移動します。

(f) Previousコマンド

データ・ペインのアドレスを，最後にコマンドにより表示アドレスを明示的に変更する前のアドレスに戻します。カーソル・キー，ROLLDOWNまたはROLLUPを使って移動した場合には，位置は記憶されません。

TDシリーズは，最後の５つのアドレスのスタックを保持するため，Followメニューまたは Gotoコマンドを何回か使ったあと，それを逆に追跡することができます。

(g) Display asコマンド

データ・ペインの中でのデータの表示形式を選択することができます。C，Pascal，およびアセンブラで使われるどんな書式でも選択することができます。次に，メニュー・オプションについて説明します。

- Byte　データ・ペインを，１バイト長の16進数として表示するように設定します。これは，Cのchar型に対応します。
- Word　データ・ペインを，ワード長の16進数として表示するように設定します。2

バイトの16進数値が表示されます。これは，Cのintデータ型に対応します。

- ●Long　データ・ペインを，倍長の16進整数として表示するように設定します。4バイトの16進数値が表示されます。これは，Cのlongデータ型に対応します。
- ●Comp　データ・ペインを，8バイト長の整数を表示するように設定します。整数の10進数値が表示されます。
  PascalのComp（IEEE）データ型です。
- ●Float　データ・ペインを，short浮動小数点数として表示するように設定します。科学的記数法による浮動小数点値が表示されます。これは，Cのfloat型と同じです。
- ●Real　データ・ペインを，Pascalの6バイト長の浮動小数点数を表示するように設定します。
- ●Double　データ・ペインを，8バイト長の浮動小数点数を表示するように設定します。これは，Cのdouble型と同じです。
- ●Extended　データ・ペインを，10バイト長の浮動小数点数を表示するように設定します。科学的記数法による浮動小数点値が表示されます。これは，コプロセッサが使う内部書式です。またこれは，Cのlong double型に対応します。
- ●S-JIS　表示形式はByteオプションと同じですが，シフトJISコードとみなせるコードに対しては，データ・ペインの右側の部分に対応する全角文字を表示します。

(h) Block

メモリ・ブロックを操作することができます。メモリ・ブロックの移動，クリア，設定またはメモリ・ブロックのディスク・ファイルへの書き込み，ディスク・ファイルからメモリ・ブロックへの読み込みができます。

Blockを選択すると，ポップアップ・メニューが表示されます。

- ●Clear　一連のメモリ・ブロックをゼロ(0)に設定します。クリアするアドレスとバイト数を入力するよう求めるプロンプトが表示されます。
- ●Move　メモリ・ブロックをあるアドレスから別のアドレスにコピーします。ソース・アドレス，デスティネーション・アドレス，およびコピーするバイト数を入力するよう求めるプロンプトが表示されます。
- ●Set　隣接するメモリ・ブロックを特定のバイト値に設定します。ブロックのアドレス，設定するバイト数，および設定する値を入力するよう求めるプロンプトが表示されます。
- ●Read　ファイルの全体または一部をメモリ・ブロックに読み込みます。最初に読み込むファイルの名前を入力するダイアログ・ボックスが表示され，次に読み込み先のアドレス，最後に読み込むバイト数を入力するよう求めるプロンプトが表示されます。

● Write　メモリ・ブロックをファイルに書き込みます。最初に書き込むファイルの名前を入力するダイアログ・ボックスが表示され，次に書き込むブロックのアドレス，および書き込むバイト数を入力するよう求めるプロンプトが表示されます。

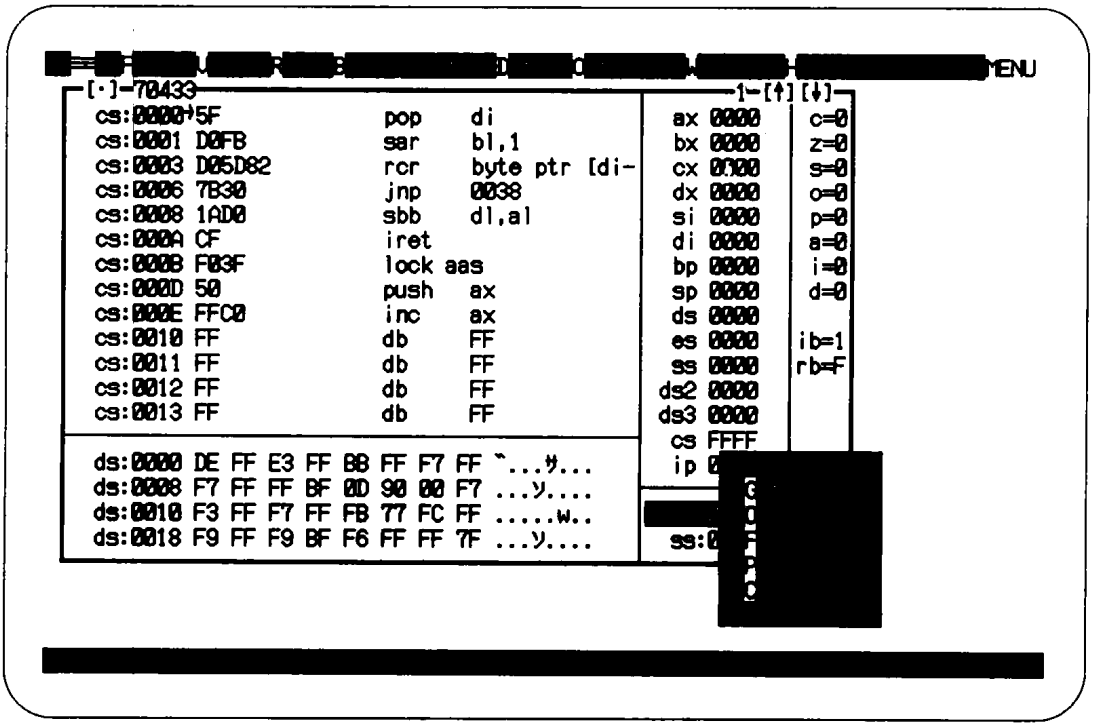
11.3.4 スタック・ペイン

スタック・ペインはCPUウインドウの右下隅にあり，スタックの内容が表示されます。

(1) スタック・ペインのローカル・メニュー

スタック・ペイン内でGRPH-F10を押すとローカル・メニューが表示されます。また，CTRLキー・ショート・カットが使用可能であれば，CTRLキーと目的のコマンドの先頭文字を押して，そのコマンドにアクセスすることができます。

図11-5　スタック・ペインのローカル・メニュー



(a) Gotoコマンド

スタック内のアドレスに移動します。新しいスタック・アドレスを入力してください。プログラム・スタックの外側のアドレスを入力することもできますが，通常，プログラムの外側の任意のデータを調べるにはデータ・ペインを使います。アドレスの入力方法については第9章式を参照してください。

このコマンドを使う前の位置にスタック・ペインを戻すには，Previousコマンドを使います((d)

Previousコマンド参照）。

(b) Originコマンド

PS：PCレジスタ・ペアが示す現在のスタック位置に移動します。このコマンドは，最初の位置に戻りたいときに便利です。

このコマンドを使う前の位置にスタック・ペインを戻すには，Previousコマンドを使います（(d) **Previous**コマンド参照）。

(c) Followコマンド

現在ハイライト表示されているワードが指し示すスタック内のワードに移動します。このコマンドは，呼び出し関数までスタック・フレームを逆にたどるときに便利です。

このコマンドを使う前の位置にスタック・ペインを戻すには，Previousコマンドを使います（(d) **Previous**コマンド参照）。

(d) Previousコマンド

スタック・ペインの位置を，最後にコマンドにより表示アドレスを明示的に変更する前のアドレスに戻します。カーソル・キーとROLLDOWNとROLLUPを使って移動した場合には，位置は記憶されません。

Previousコマンドを繰り返し使うと，スタック・ペインの表示が２つのアドレスの間で交互に切り替わります。

(e) Changeコマンド

現在ハイライト表示されているスタック・ワードについて，新しいワード値を入力できます。

また，ハイライト表示されているスタック項目の新しい値を入力するだけで，このコマンドを起動することもできます。Changeコマンドを指定したときと同じダイアログ・ボックスが表示されます。

11.4　アセンブラ

TDシリーズでは，コード・ペイン内のローカル・メニュー中のAssembleコマンドによりターゲット・マイクロプロセッサ命令セットを使用すると，命令のアセンブルが可能になります。

TDシリーズの組み込みアセンブラを使ってプログラムを修正しても，その変更内容は永続的なものではありません。Run | Program resetコマンドを使ってプログラムを再ロードしたり，File | Openコマンドを使って別のプログラムをロードすると，変更内容は失われます。

通常は，プログラムの問題を解決するためのアイデアをテストするには，アセンブラを使います。変更内容が機能することを確認したら，実際にソース・コードを変更して，プログラムを再コンパイル-リンクしてください。

Run | IE resetによってもプログラムが再ロードされ，すべてのアセンブラのパッチが消えます。

次の節では，組み込みアセンブラと，Turbo AssemblerおよびMicrosoft Assemblerで使える構文との違いについて説明します。

11.4.1　オペランドのアドレス・サイズのオーバライド

サブルーチン制御命令(CALL)，ブランチ命令(BR)，および条件付きブランチ命令(BNE，BLTなど)の場合，アセンブラはデスティネーション・アドレスに到達できる最小の命令を生成します。次のように，デスティネーション・アドレスの前にNEARとFARのオーバライドを使って，特定のサイズで命令をアセンブルすることができます。

```
CALL  FAR   XYZ
BR    NEAR  A1
```

11.4.2　メモリとイミーディエト・オペランド

プログラムからのシンボルを命令オペランドとして使うときは，組み込みアセンブラに対して，シンボルの内容とシンボル・アドレスのどちらを意味するのかを指示してください。シンボル名だけを使うと，アセンブラは，シンボルの前にアセンブラのOFFSET演算子を付けた場合と同じように，それをアドレスとして扱います。シンボルをカッコ [] で囲むとメモリ参照になります。

11.4.3　オペランドのデータ・サイズのオーバライド

命令の中には，オペランドの前に次の式のうちの１つを使って，オペランド・サイズを指定しなければならないものがあります。

```
BYTE PTR
WORD PTR
```

これらのオーバライドを使う命令の例を次に示します。

```
add byte ptr  [IX], 10
mov word ptr  [bp+10], 99
```

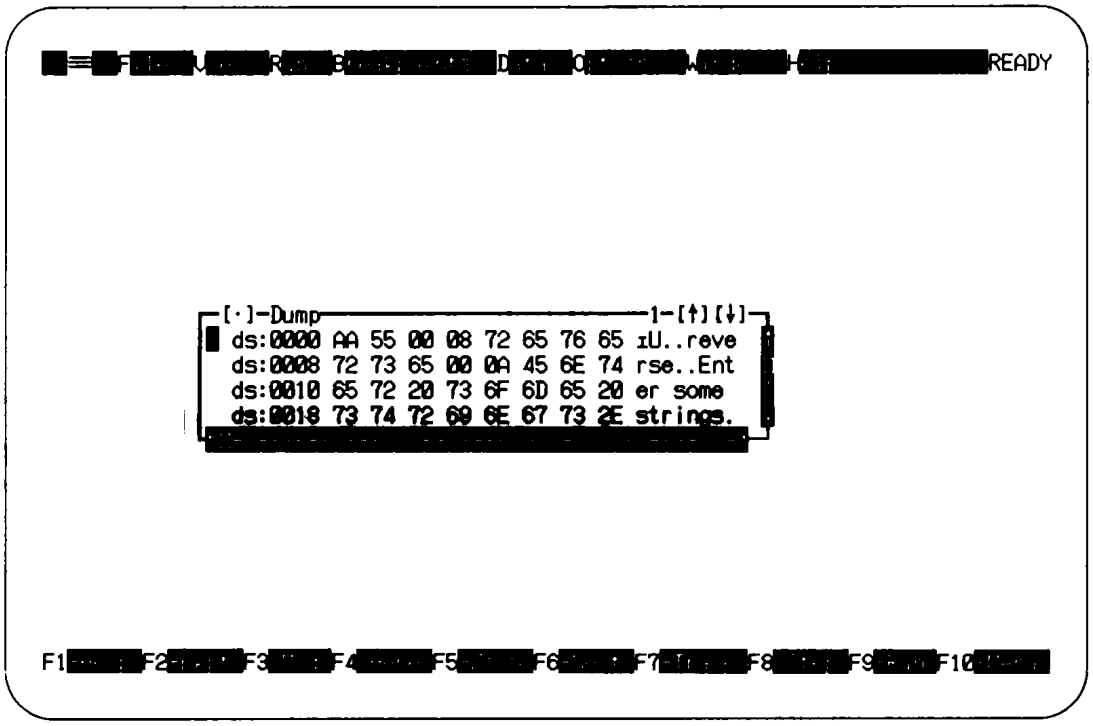
11.4.4　プリミティブ・ブロック転送命令

プリミティブ・ブロック転送命令をアセンブルするときには，命令ニモニックの一部としてサイズ(バイトまたはワード）を含めてください。アセンブラはサイズを指定するオペランドを使うため，サイズ指定のないニモニックを使うプリミティブ・ブロック転送命令形式は受け付けません。たとえば，STM WORD PTR [IY] は使用できません。STMWを使用してください。

11.5　Dumpウインドウ

Dumpウインドウは，メモリ領域のデータ・ダンプをそのまま表示します。このウインドウは，CPUウインドウのデータ・ペインと同じように機能します（**11.3.3　データ・ペイン参照**）。

図11－6　Dumpウインドウ（View | Dump）



一般に，アセンブラ・プログラムをソース・レベルでディバグしていて，一部のデータ領域を下位レベルで調べたいときにはDumpウインドウを使います。Dumpウインドウをオープンするには View | Dumpを使います。

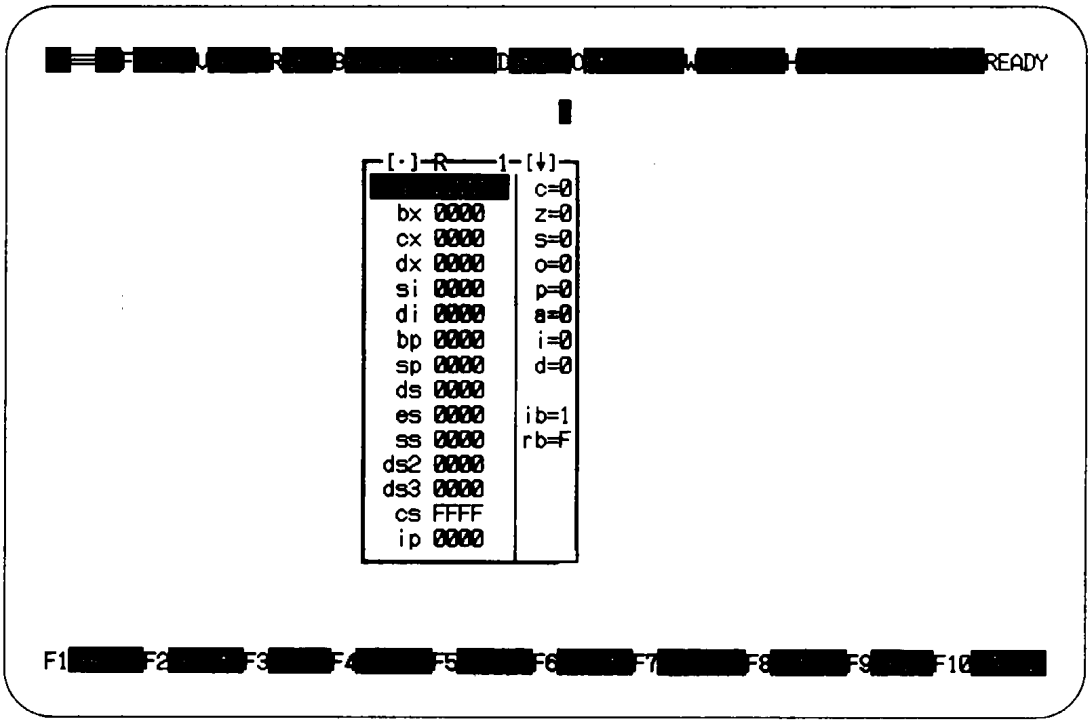
また，Inspectウインドウ内にいて，調べている項目を構成するバイトそのものを見たいときにも，このウインドウを使うことができます。Inspectウインドウ内のデータを指し示すDumpウインドウをオープンするにはView | Dumpを使います。

Dumpウインドウのローカル・メニュー（CPUウインドウのデータ・ペインと同じ）のBlock | writeコマンドでは，.HEXファイルだけをセーブすることができます。

11.6　Registersウインドウ

Registersウインドウは，CPUレジスタとフラグの内容を表示します。このウインドウは，CPUウインドウのレジスタ・ペインとフラグ・ペインを組み合わせた機能を持っています。

図11－7　Registersウインドウ（View | Registers）



ソース・レベルでアセンブラ・プログラムをディバグしていて，レジスタ値を調べたいときにはこのウインドウを使ってください。View | RegistersコマンドでRegistersウインドウをオープンできます。

第12章　周辺リソース管理

この章では，V55SC，V55PIのSFR，レジスタ・バンク，マクロ・サービス・チャネル，内蔵RAM，拡張メモリの内容を表示，変更する方法を説明します。

12.1　Targetメニュー

V55SC，V55PIの内部データ領域を操作するときは，View | Targetコマンドを使用します。

周辺レジスタ・ビューは，レジスタ・ウインドウやCPUウインドウのレジスタ・ペインに似ています（第11章　アセンブラ・レベルのディバグ参照）。一部を除いて，これらのローカル・メニューと同じものを使用します。

12.2　周辺レジスタ表示規則

ライト・オンリーのレジスタは内容フィールドに“－－”が表示されます。これは，このレジスタのリードが不可能でも，Changeコマンドにより新たな出力値を供給できることを示します。

12.3　周辺レジスタのローカル・メニュー

各周辺レジスタの表示には，View | Registerウインドウに似たローカル・メニューがあります。周辺レジスタ・ローカル・メニューとの違いは，ローカル・メニューにReadコマンドとUpdateコマンドが追加されたことです。

周辺レジスタのビューを出すにはGRPH-F10を使います。周辺レジスタ・ビュー中で，変更したいレジスタにハイライト・バーを移動させます。CTRLキーと選択したいメニューの先頭文字（I，D，Z，C，R）を押すことにより，次の変更が可能です。

(1) Incrementコマンド

現在選択されている周辺レジスタの値をインクリメントします。このコマンドはリード・オンリー，ライト・オンリーのレジスタでは使用できません。

(2) Decrementコマンド

現在選択されている周辺レジスタの値をデクリメントします。このコマンドはリード・オンリー，ライト・オンリーのレジスタでは使用できません。

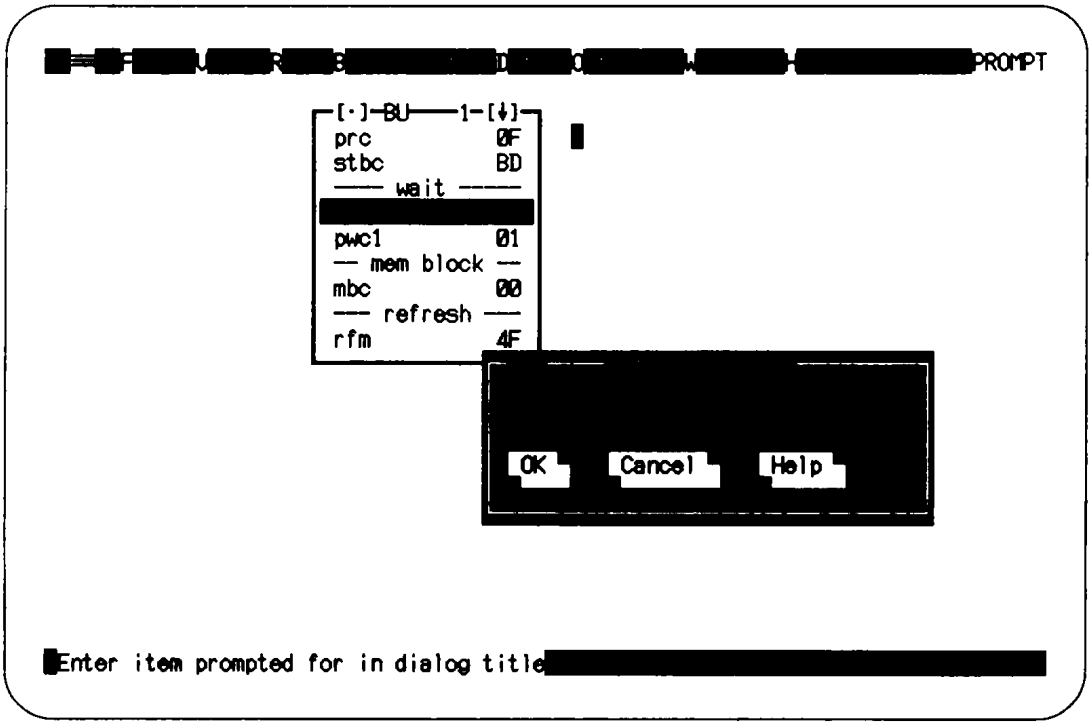
(3) Zeroコマンド

現在選択されている周辺レジスタの値をクリアします。

(4) Changeコマンド

このコマンドにより，次のようなダイアログ・ボックスがオープンし，現在選択されている周辺レジスタの値を変更できます。

図12－1　Changeダイアログ・ボックス



(5) Readコマンド

選択されている周辺レジスタのリードを強制します。この場合，周辺レジスタのリードがターゲット・システムの状態を変化させることがあります。このため，表示する前にTDシリーズへあらかじめレジスタ内容のリードを指示しなければいけません。

(6) Updateコマンド

V55SC，V55PIのすべての周辺レジスタ・ビューを更新します。ビューが最後に更新されてから，あるイベントによって周辺レジスタの状態が変化した可能性がある場合，レジスタの現在の内容を見たいときにこのコマンドを使用します。

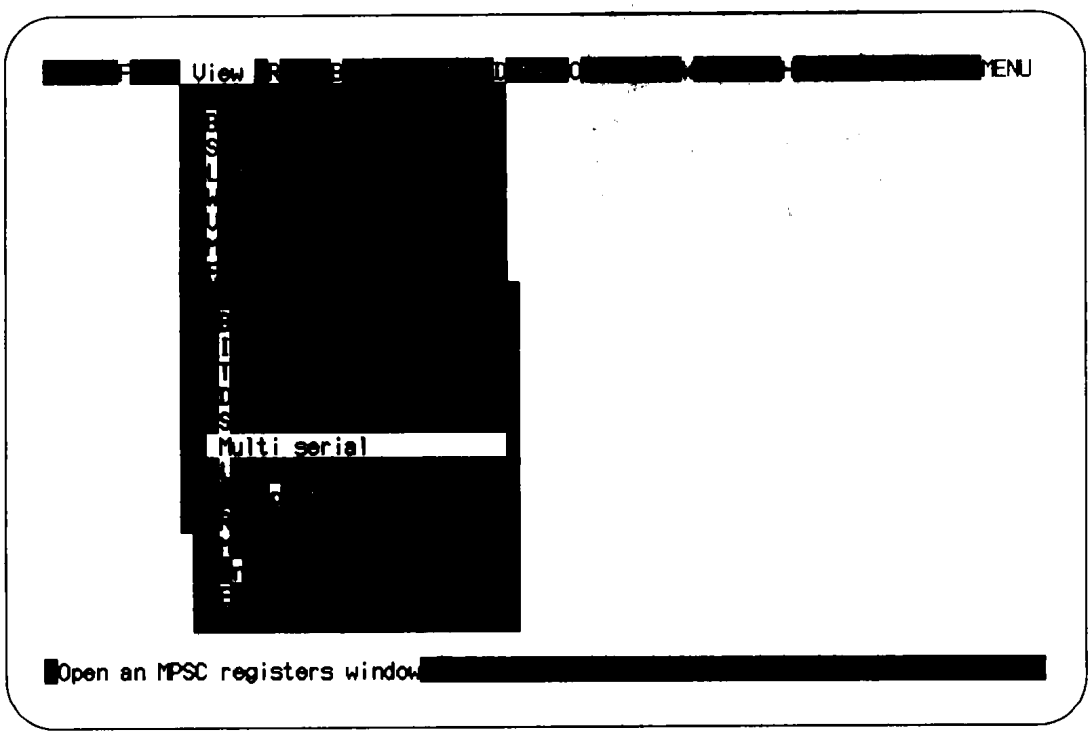
12.4　周辺レジスタ・ビュー

ここでは，V55SC，V55PI用の周辺レジスタ・ビューについて説明します。

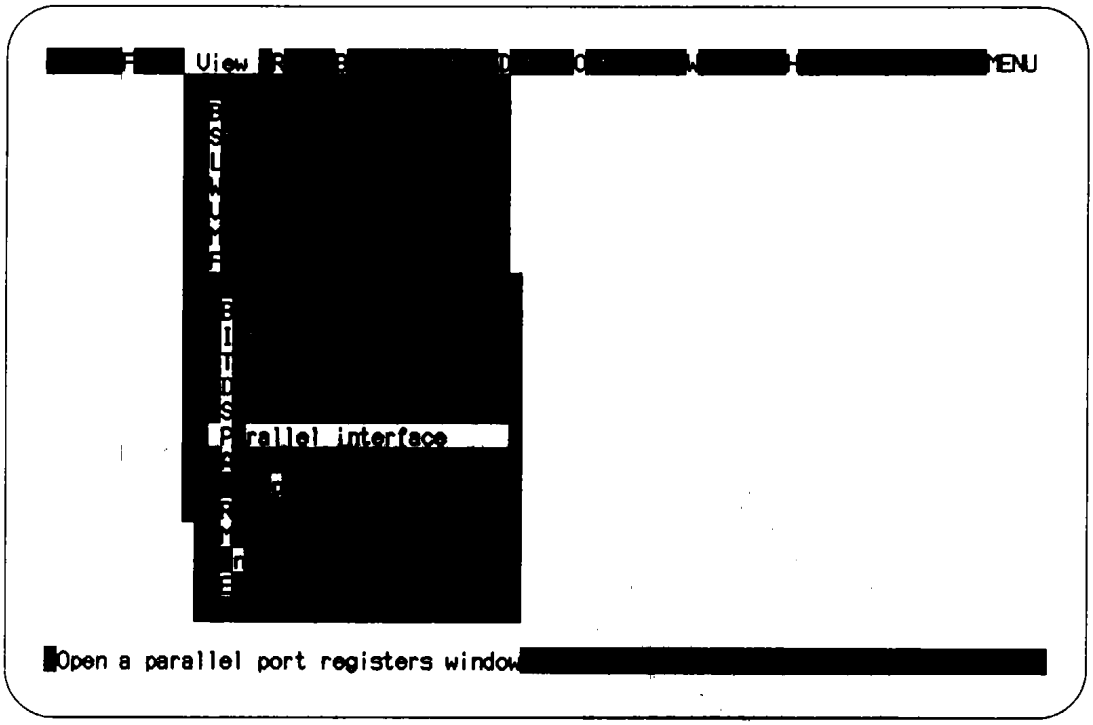
View | Targetコマンドを選択し，希望の周辺レジスタのグループ名を指定します。これにより，ターゲット・マイクロプロセッサにある各種の周辺レジスタ・グループへアクセスできます。

図12－2　Targetコマンドのポップアップ・メニュー（View | Target）

（a）TD423の場合



(b) TD433の場合



(1) SFRの表示

TD423, TD433では, SFR（特殊機能レジスタ）を機能ごとに分類して表示, 変更できます。

それぞれのウインドウは, Targetコマンドのポップアップ・メニューのコマンド[注]で開きます。ウインドウ内の現在選択されている（ハイライト表示されている）レジスタの内容を変更できます。

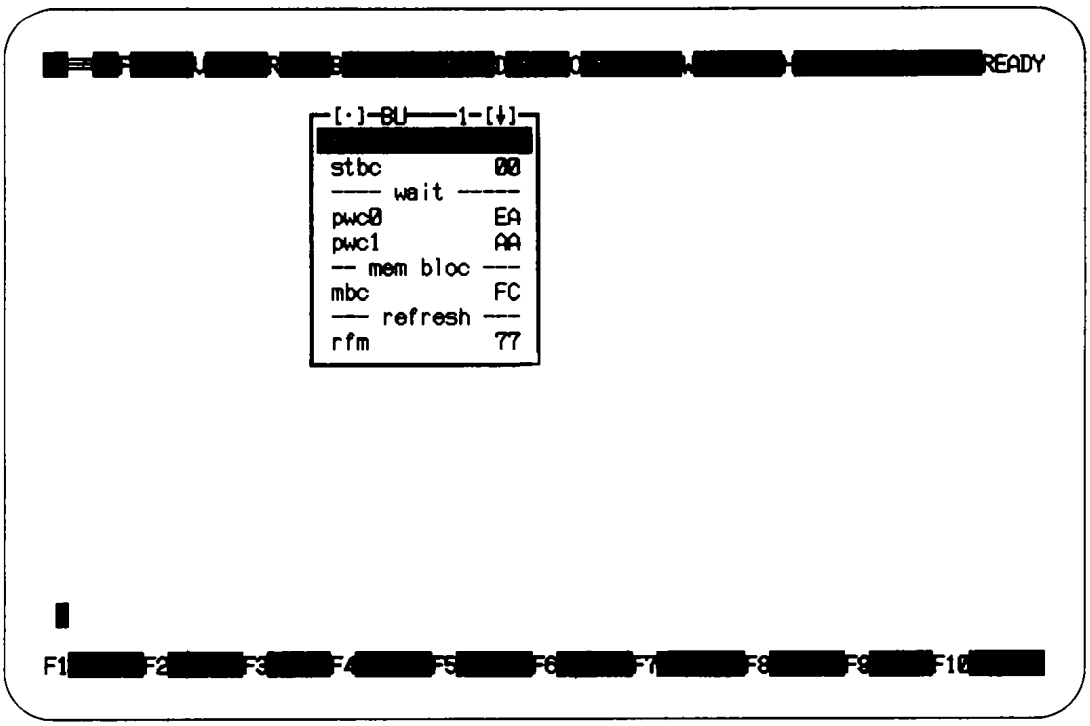
注　メニューの項目はTD423とTD433で異なります。

SFRについてのコマンドには次のものがあります。

① Bus control
② Interrupt controller
③ Timers
④ DMA controller
⑤ Serial
⑥ I/O ports
⑦ Multi serial（TD423だけ）
⑧ Local DMA（　〃　）
⑨ Parallel interface（TD433だけ）
⑩ A/D converter（　〃　）

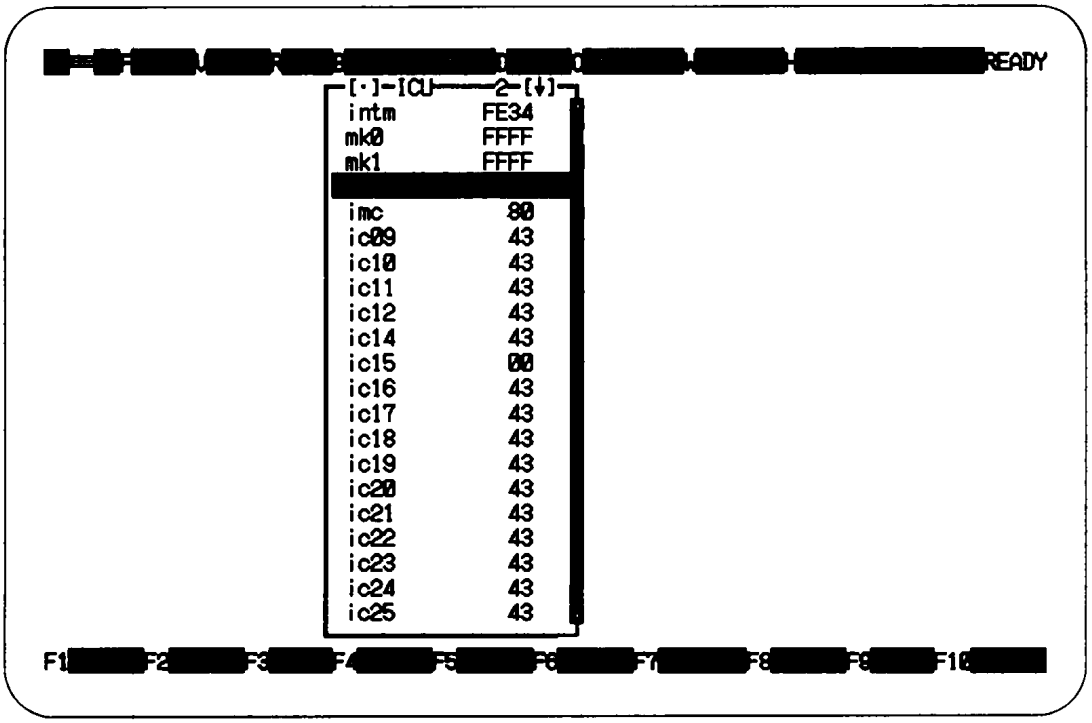
① Bus controlコマンド

図12－3　Bus controlウインドウ（View | Target | Bus control）



② Interrupt controllerコマンド

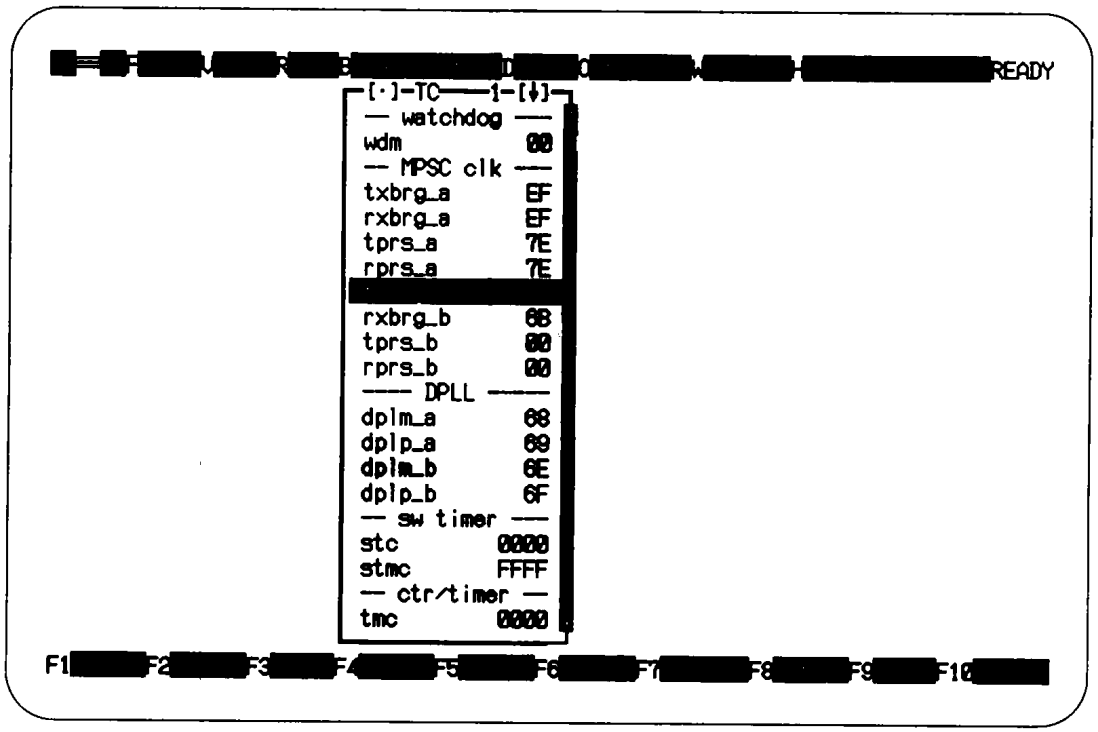
図12－4　Interrupt controllerウインドウ（View | Target | Interrupt controller）



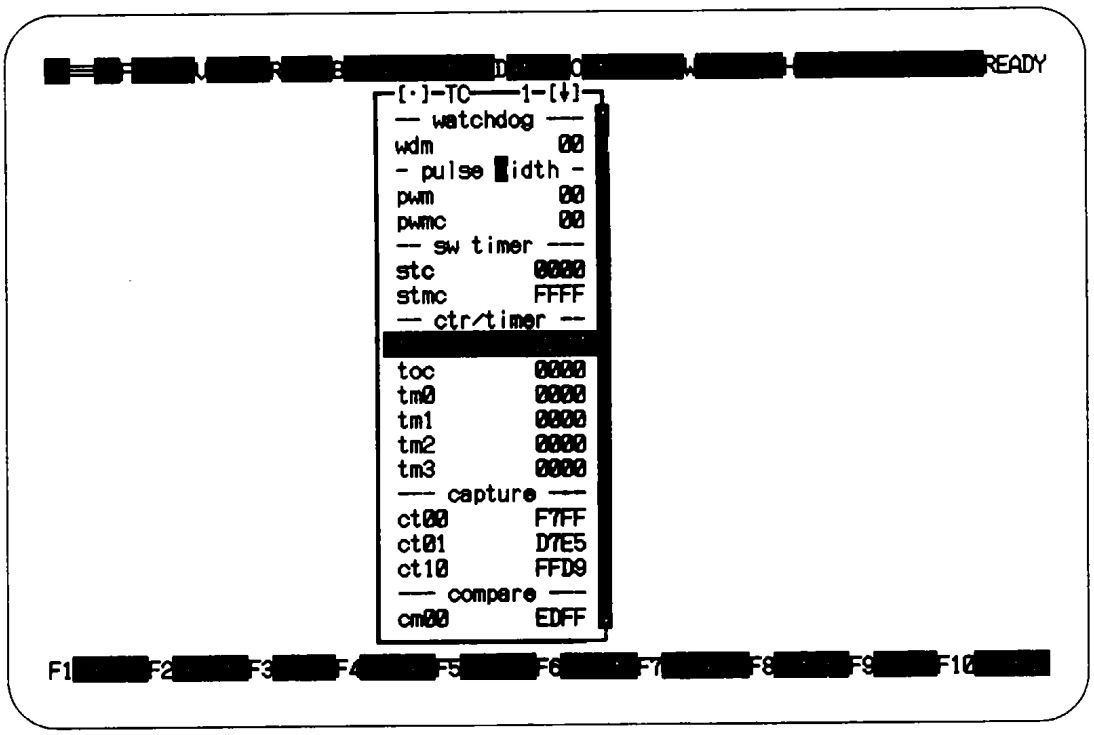
③ Timersコマンド

図12－5　Timersウインドウ（View | Target | Timers）

（a）TD423の場合

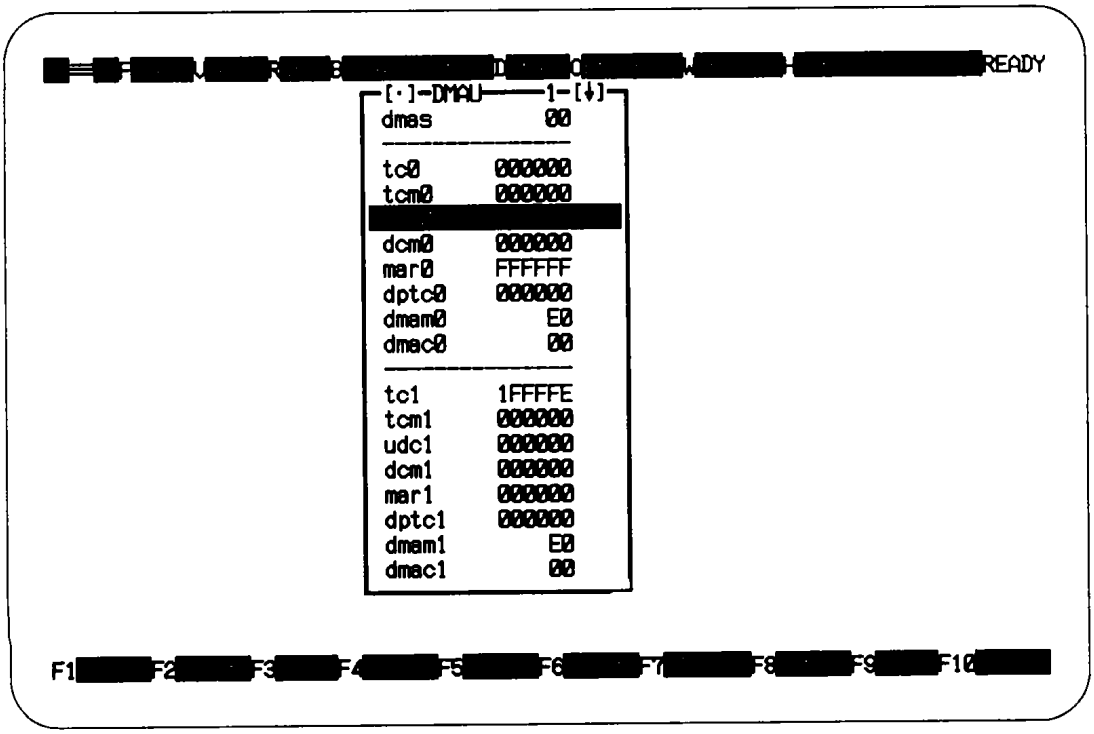


(b) TD433の場合



④ DMA controllerコマンド

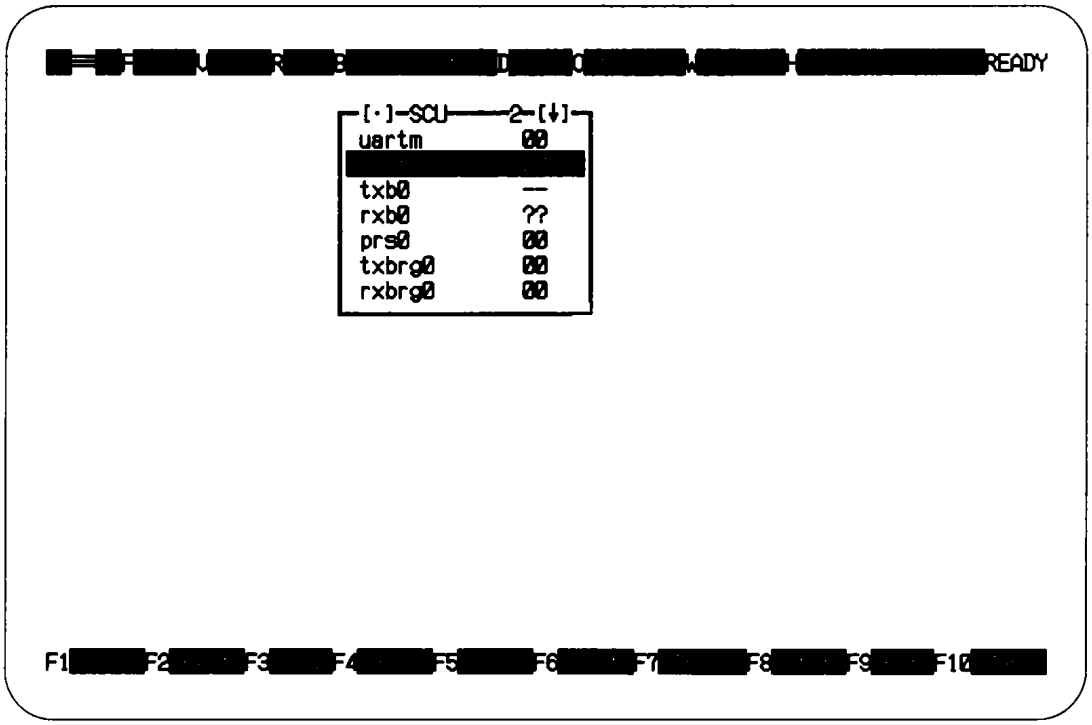
図12-6　DMA controllerウインドウ（View | Target | DMA controller）



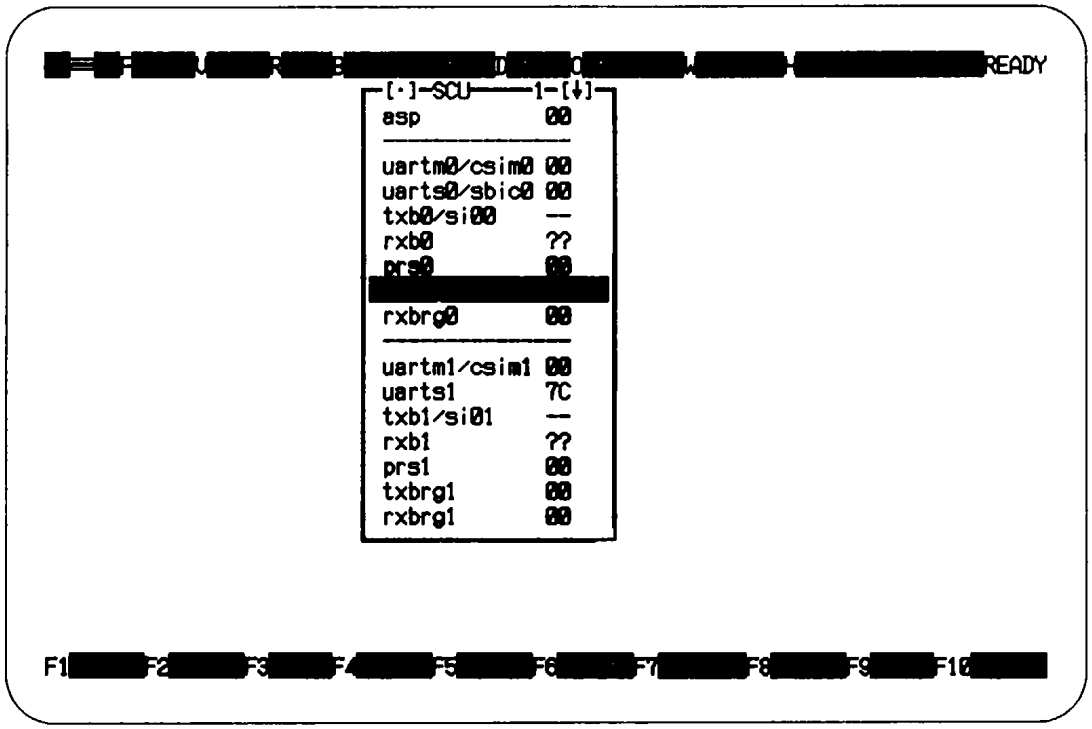
⑤ Serialコマンド

図12－7　Serialウインドウ（View | Target | Serial）

（a）TD423の場合



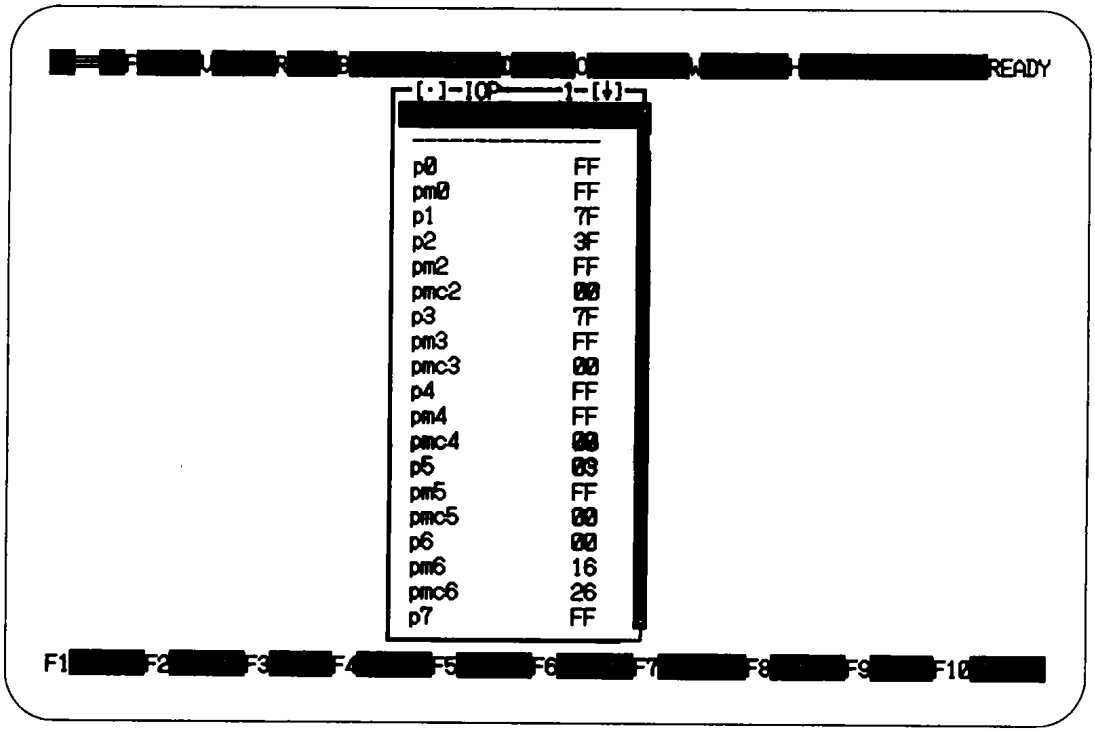
(b) TD433の場合


READY
[·]-SCU-1-[↓]
asp 00
uartm0/csim0 00
uarts0/sbic0 00
txb0/si00 —
rxb0 ??
prs0 00
rxbrg0 00
uartm1/csim1 00
uarts1 7C
txb1/si01 —
rxb1 ??
prs1 00
txbrg1 00
rxbrg1 00
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10

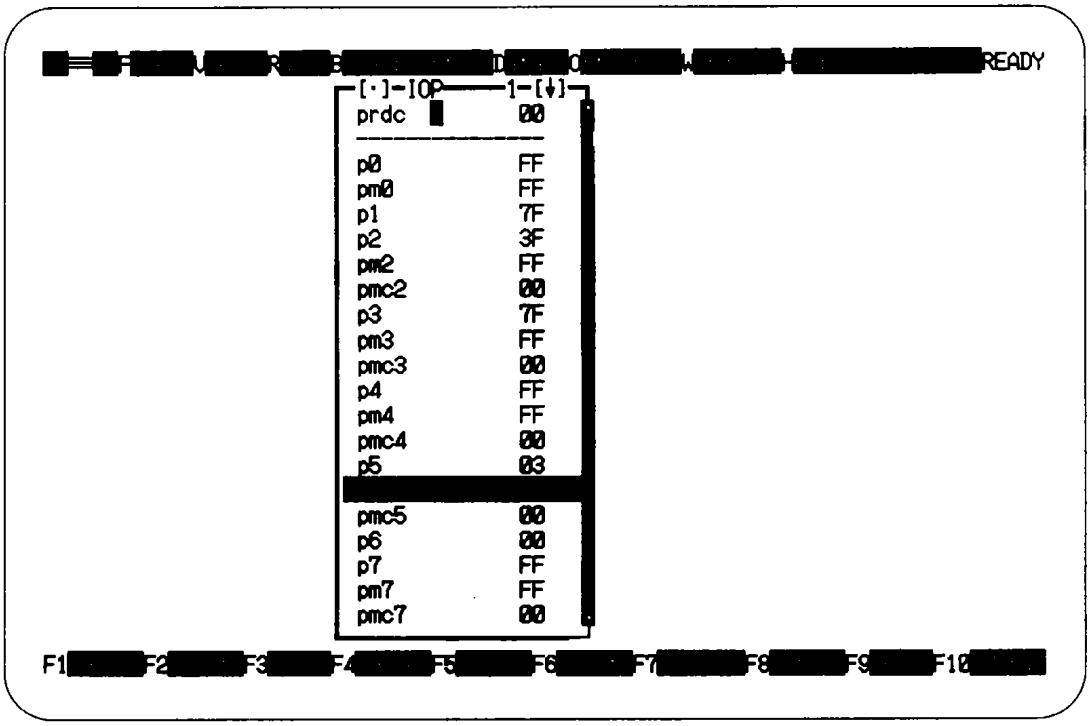
⑥I/O portsコマンド

図12－8　I/O portsウインドウ（View | Target | I/O ports）

（a）TD423の場合



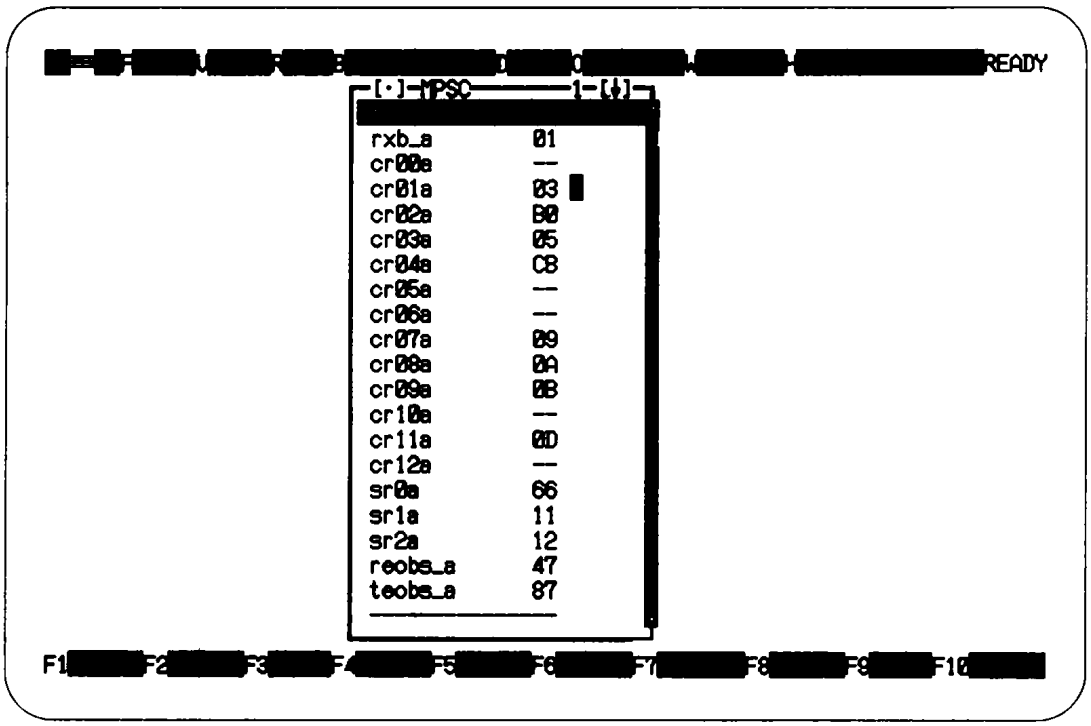
(b) TD433の場合


READY
[·]-IOP
1-[↓]
prdc
00
p0
FF
pm0
FF
p1
7F
p2
3F
pm2
FF
pmc2
00
p3
7F
pm3
FF
pmc3
00
p4
FF
pm4
FF
pmc4
00
p5
03
pmc5
00
p6
00
p7
FF
pm7
FF
pmc7
00
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10

⑦ Multi serialコマンド

このコマンドは，TD423だけにあります（TD433にはありません）。

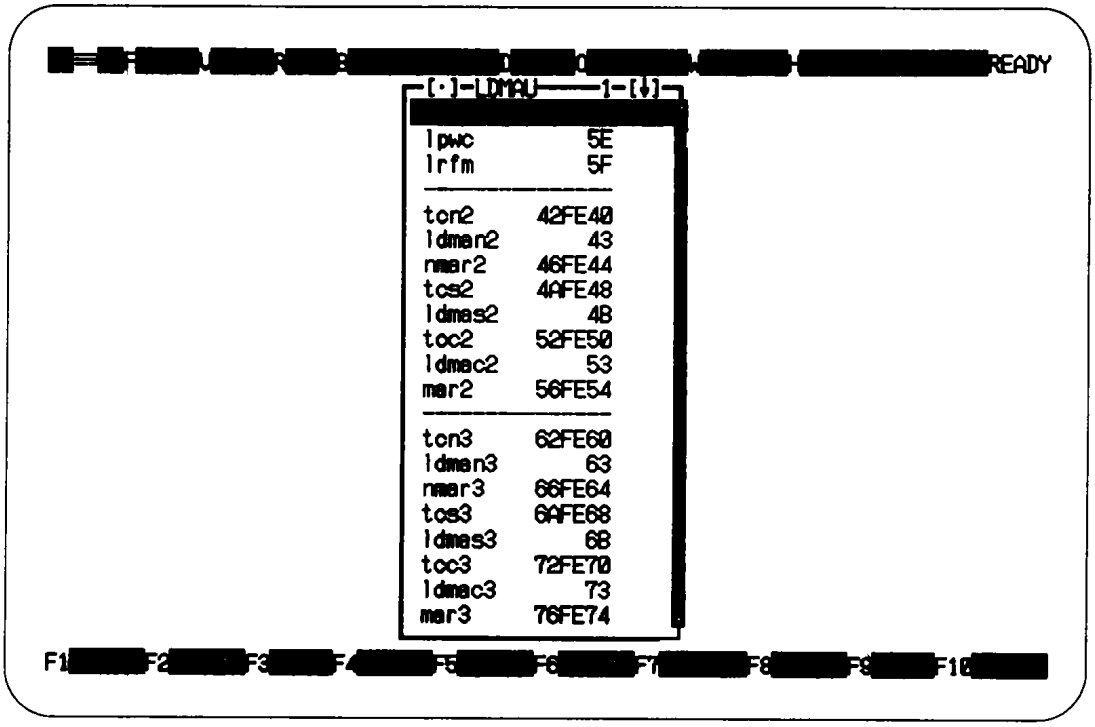
図12－9　Multi serialウインドウ（View | Target | Multi Serial）



⑧ Local DMAコマンド

このコマンドは，TD423だけにあります（TD433にはありません）。

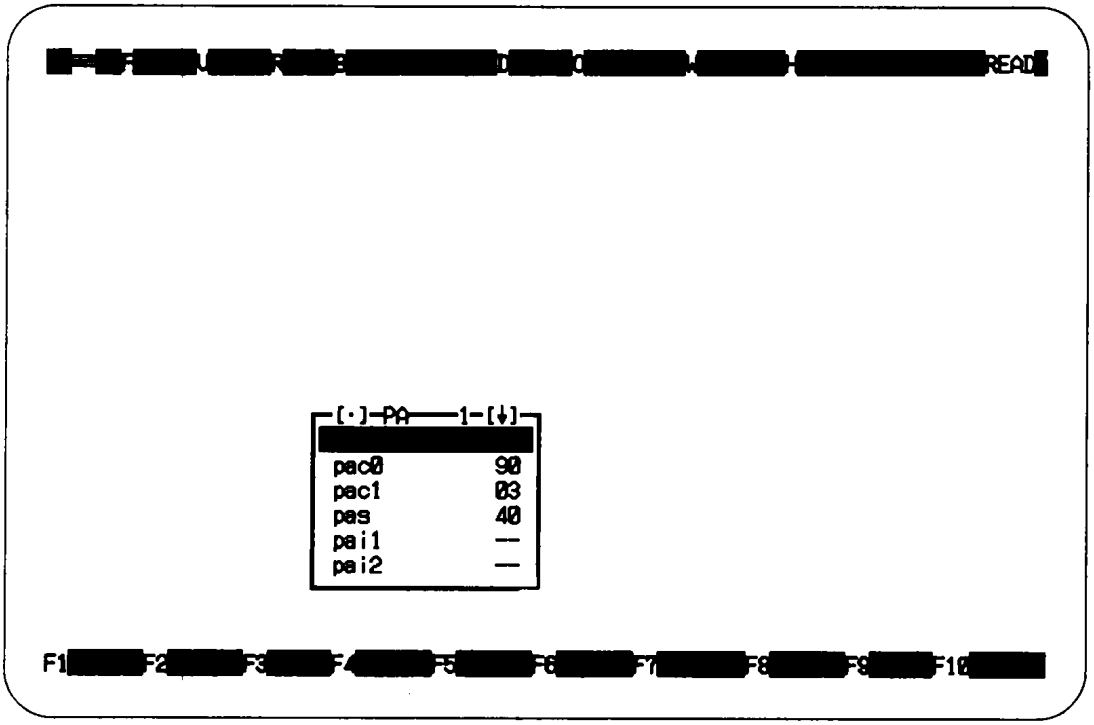
図12－10　Local DMAウインドウ（View | Target | Local DMA）



⑨ Parallel interfaceコマンド

このコマンドは，TD433だけにあります（TD423にはありません）。

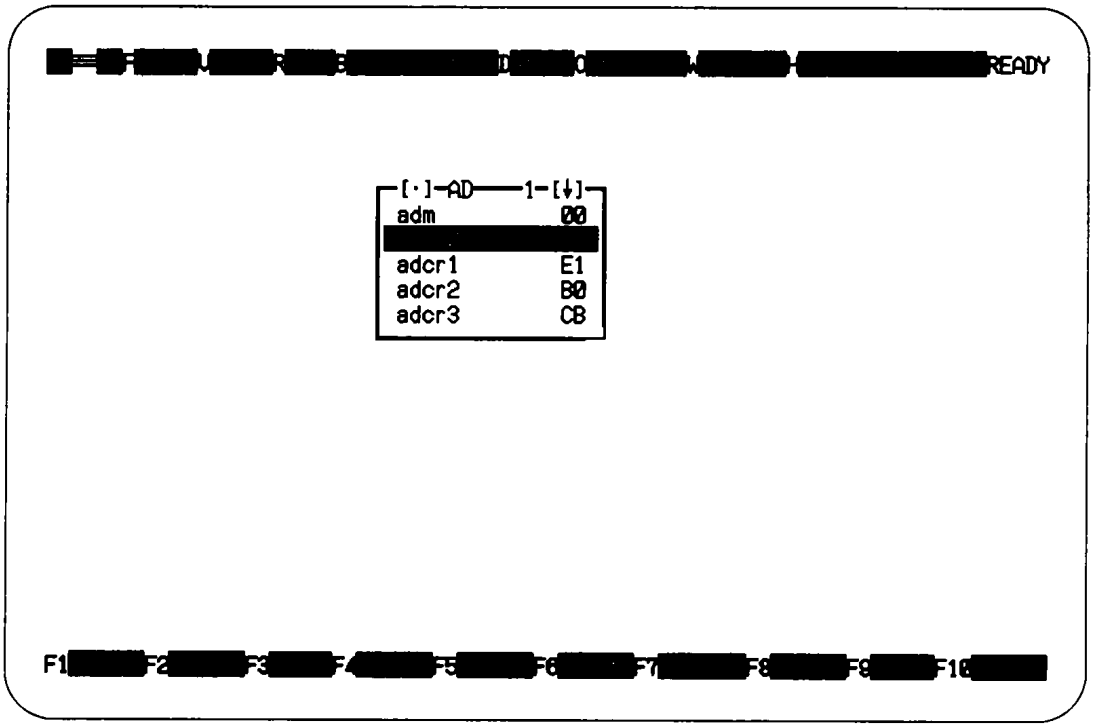
図12－11　Parallel interfaceウインドウ（View | Target | Parallel interface）



⑩ **A/D converterコマンド**

このコマンドは，TD433だけにあります（TD423にはありません）。

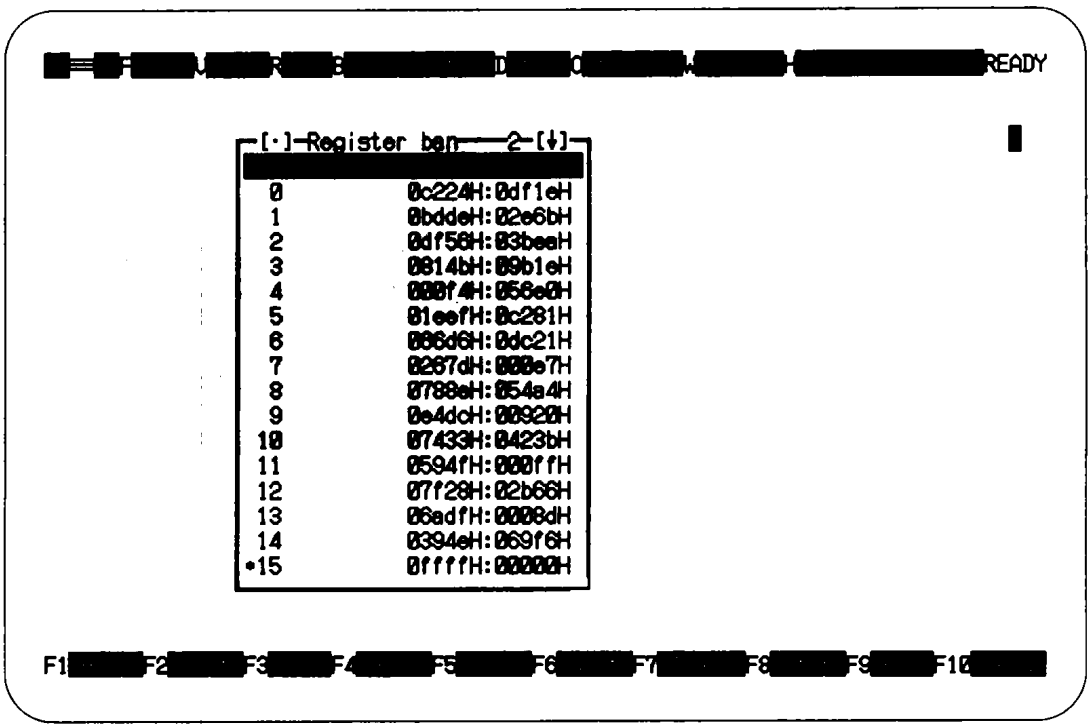
図12－12　A/D converterウインドウ（View | Target | A/D converter）



(2) Register banks (RB) コマンド

このコマンドは，V55SC，V55PIのそれぞれのレジスタ・バンクのPS：ベクタPCの内容をポップアップし，表示します。

図12－13　Register banksウインドウ（View | Target | Register banks）



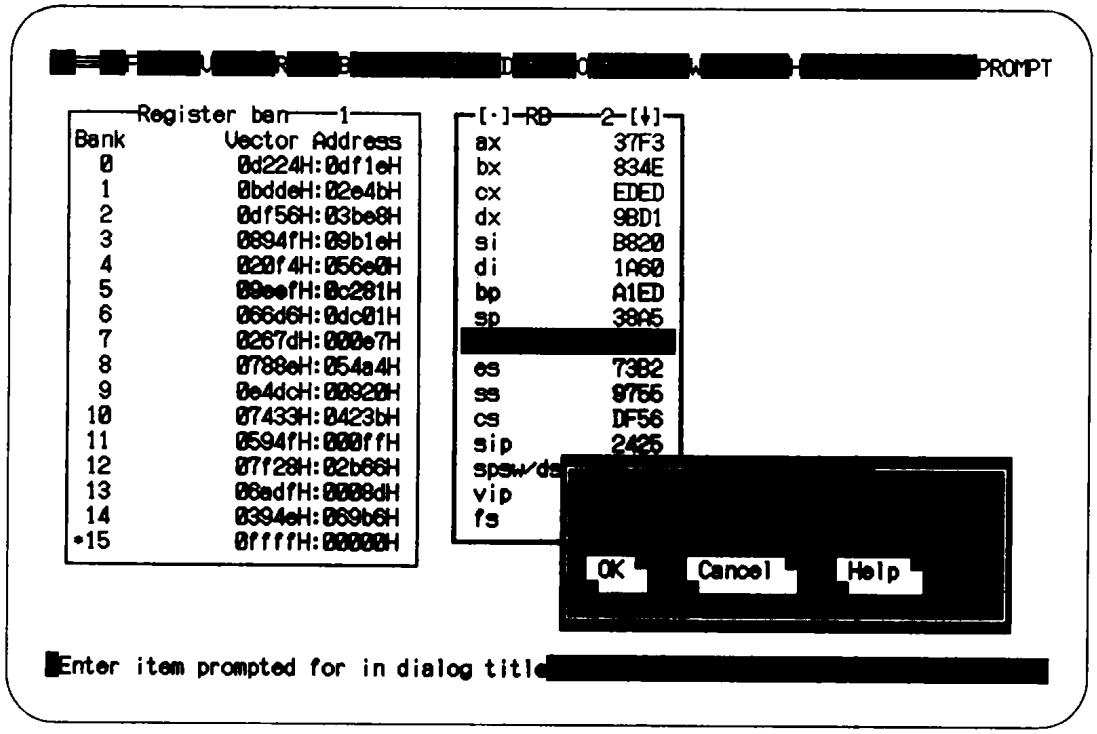
現在選択されているレジスタ・バンクの番号の前に＊（アスタリスク）をつけて表示します。

Register banksウインドウはローカル・メニューを持っており，新たなPS：ベクタPCやレジスタ・バンクの値を入力できます。

次の図12－14のように選ばれたレジスタ・バンクの内容を表示することができます。

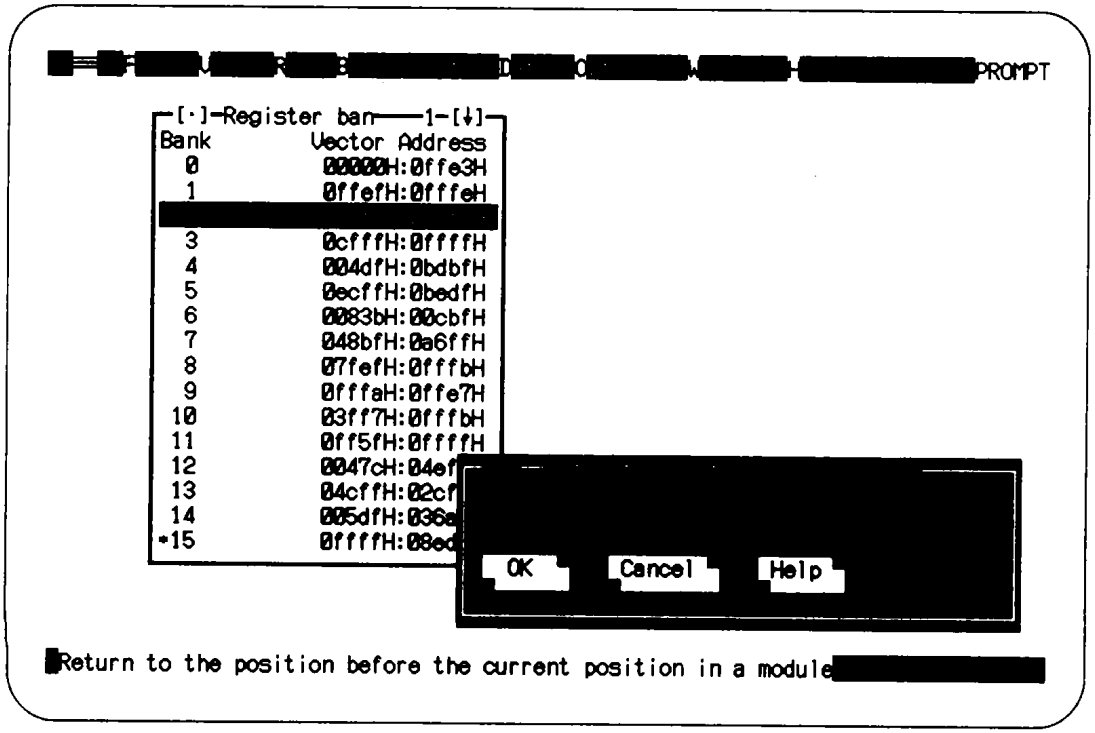
このウインドウは，いくつかを同時に重ねて開くことができます。

図12－14　レジスタ・バンクの内容の表示（ローカル・メニューから選択）



Register banksウインドウのローカル・メニュー中のChangeコマンドは，現在選択されているレジスタ・バンクのPS：ベクタPCレジスタ・ペアの変更に使用します。

図12－15　PS：ベクタPC変更ボックス（ローカル・メニューから選択）

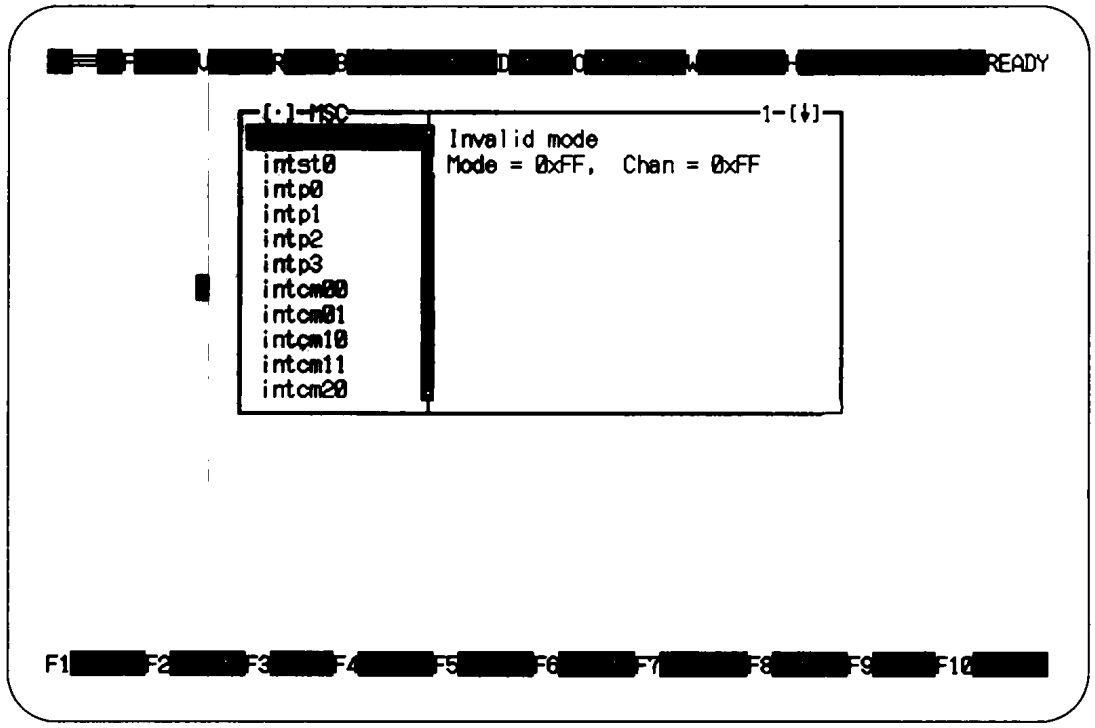


(3) Macro service channelsコマンド

Macro service channelsウインドウをオープンします。

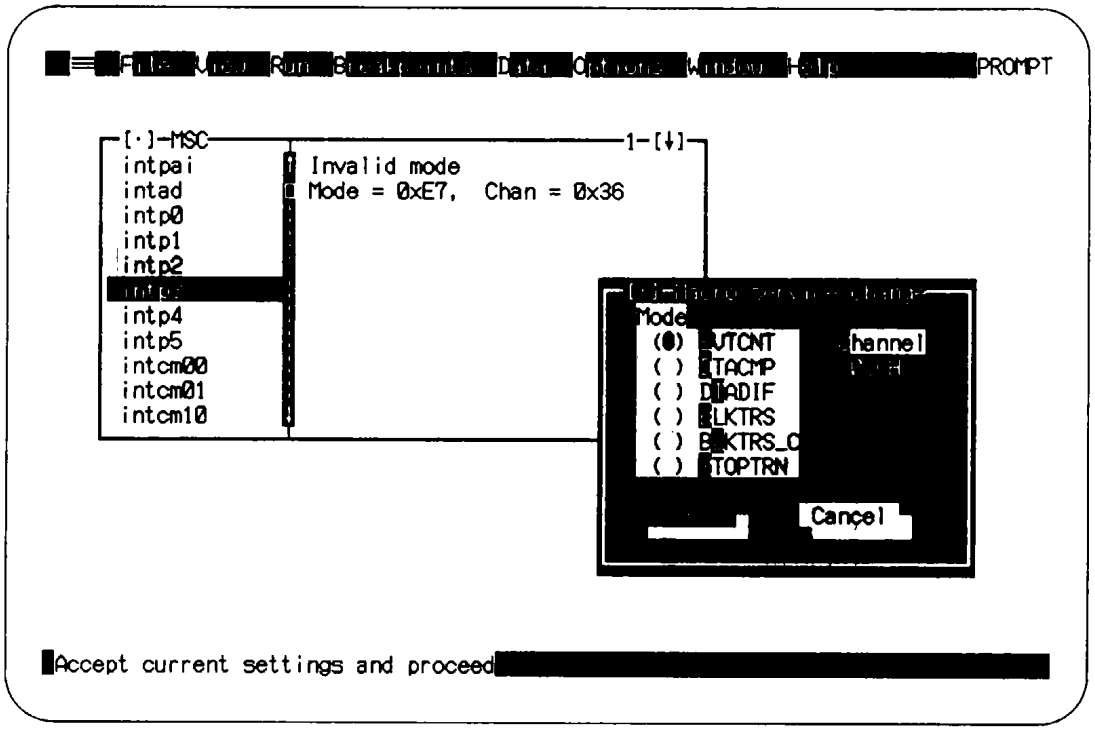
８つのマクロ・サービス・チャネルの内容を表示します。

図12－16　Macro service channelsウインドウ（View | Target | Macro service channels）



このウインドウのローカル・メニューには，InspectコマンドとChangeコマンドがあります。Inspectコマンドは，サポートされているマクロ・サービスを変更します。次に例として，このダイアログ・ボックスを使ってBlock transferマクロ・サービス機能の設定を示します。

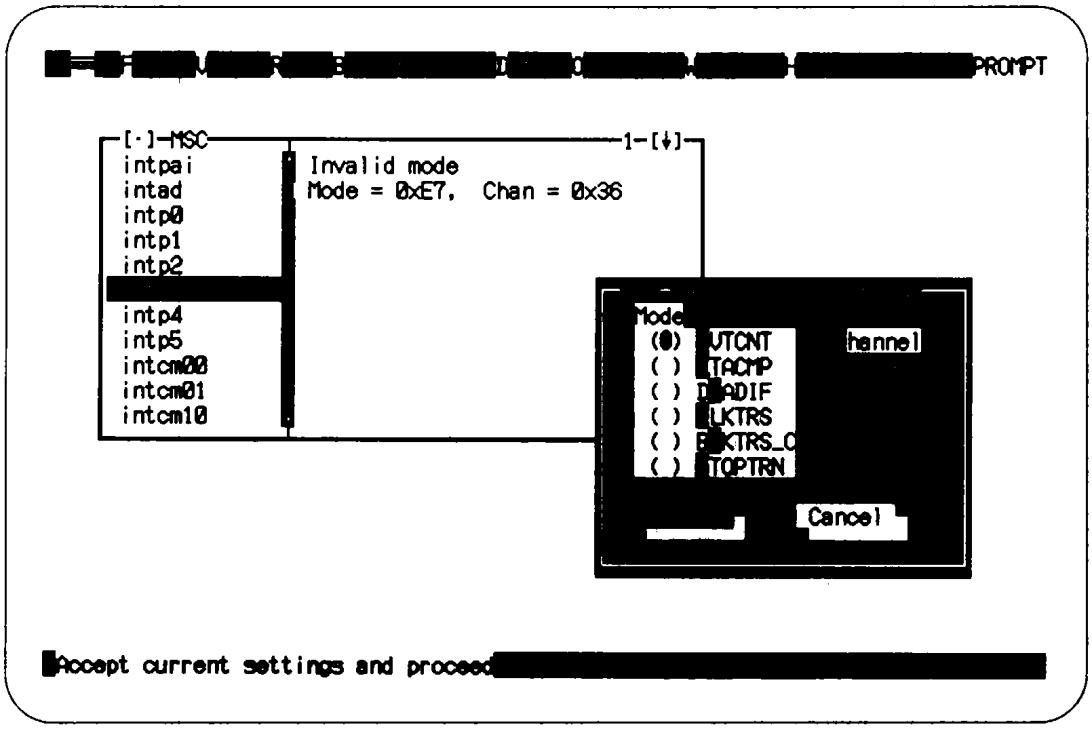
図12－17　Inspectマクロ・サービス・チャネル・ダイアログ・ボックス（ローカル・メニューから選択）



ダイアログ・ボックス内の移動は，TABキー（またはSHIFT-TAB）とカーソルを使います。マクロ・サービス・チャネルの設定は，↵キーで行います。ESCキーを押すとこのダイアログ・ボックスが閉じます。

Changeコマンドは，現在選択されているマクロ・サービス・チャネルの内容を変更します。

図12－18　Changeマクロ・サービス・チャネル・ダイアログ・ボックス（ローカル・メニューから選択）



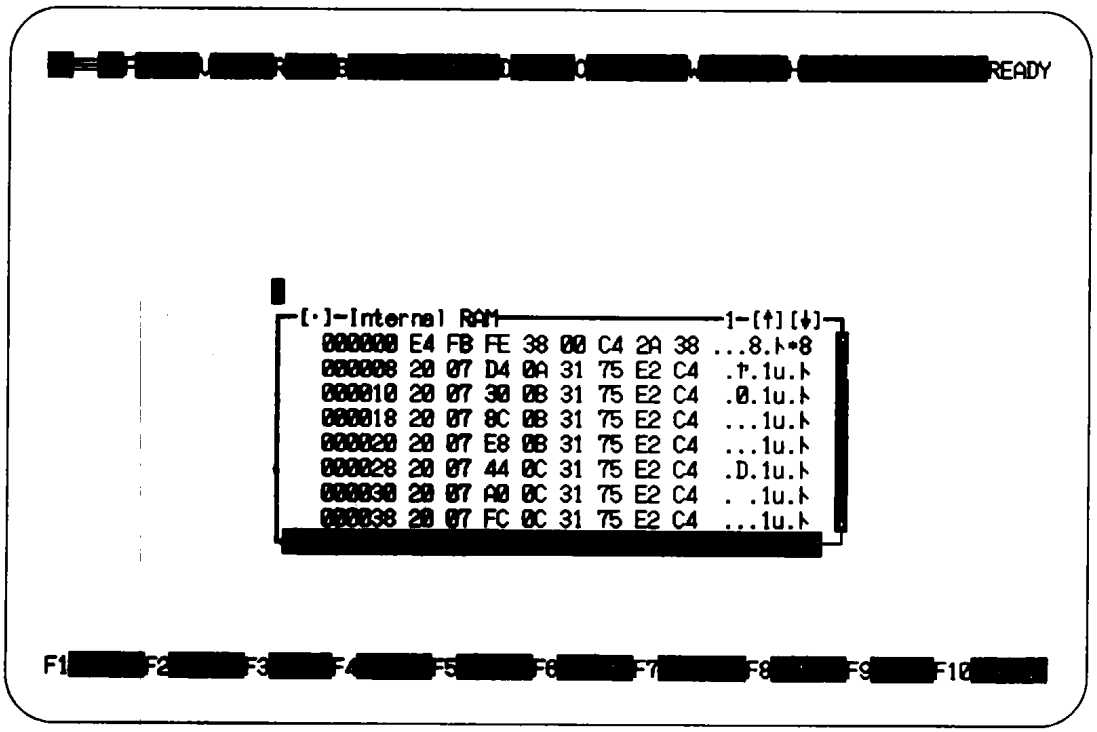
ダイアログ・ボックス内の移動は，TABキー（またはSHIFT-TAB）とカーソルを使います。マクロ・サービス・チャネルの設定は，↵で行います。ESCキーでこのダイアログ・ボックスが閉じます。

(4) Internal RAMコマンド

V55SC, V55PIは，通常のメモリ空間とは分離された512バイトの内蔵RAM空間を持ち，その中にはマクロ・サービス・レジスタと16のレジスタ・バンクが含まれています。

Internal RAMコマンドは，V55SC, V55PIの内蔵RAM空間を表示する内蔵RAMウインドウを開きます。このコマンドは，ブロック単位，バイト単位で実行できます。

図12－19　内蔵RAMウインドウ（View | Target | Internal RAM）

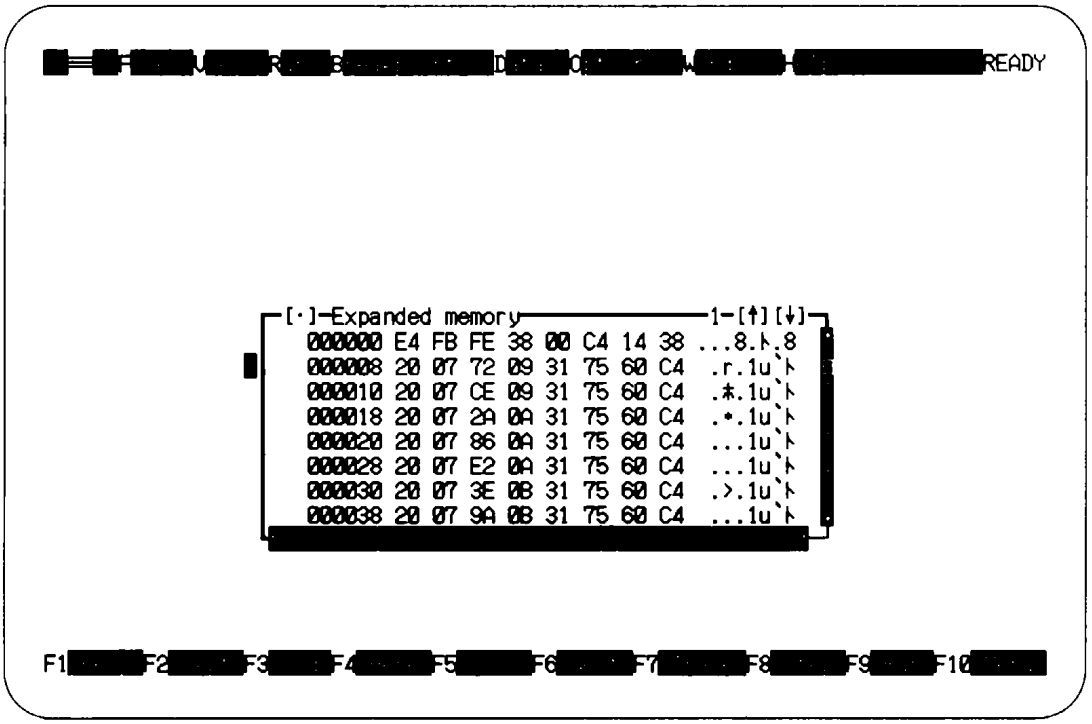


内蔵RAMのローカル・メニュー・コマンドは，CPUウインドウのデータ・ペインのローカル・メニューと同じです（第11章　アセンブラ・レベルのディバグ参照）。

12.5　Expanded memoryコマンド

このコマンドは，拡張メモリ・ウインドウをオープンします。このウインドウではV55SC，V55PIの16Mバイトのアドレス空間の任意の箇所の表示，変更ができます。

図12－20　拡張メモリ・ウインドウ（View | Target | Expanded memory）



〔メ　モ〕

第13章　IEレファレンス

この章では，Borland International Inc. 製のターボ・ディバッガに対してIE用に拡張された機能について説明します。

TDシリーズでは，メニューを選択するだけで，ハードウエア・ブレークポイントのセットやメモリのマッピング，IEのトレース・バッファ・ハードウエアの使用などができます。

13.1　インサーキット・エミュレータ・リソース

IEは，プログラム中の問題点を発見するハードウエア機能を数多く備えています。TDシリーズでは，Viewメニューに**ICE**コマンドを追加することにより，必要に応じてIEのハードウエア・リソースへすばやくアクセスできます。

メニュー・バーからView | ICEコマンドを選択すると，IEのインサーキット・エミュレータ・リソースへアクセスできます。ただし，ハードウエア・ブレーク・ポイントは除きます。

ICEメニューの項目は，次の機能を制御します。

- エミュレータ・トレース・バッファ
- メモリ・マッピング
- エミュレータ制御
- エミュレータ・コンフィギュレーション・ファイルの保存とリストア

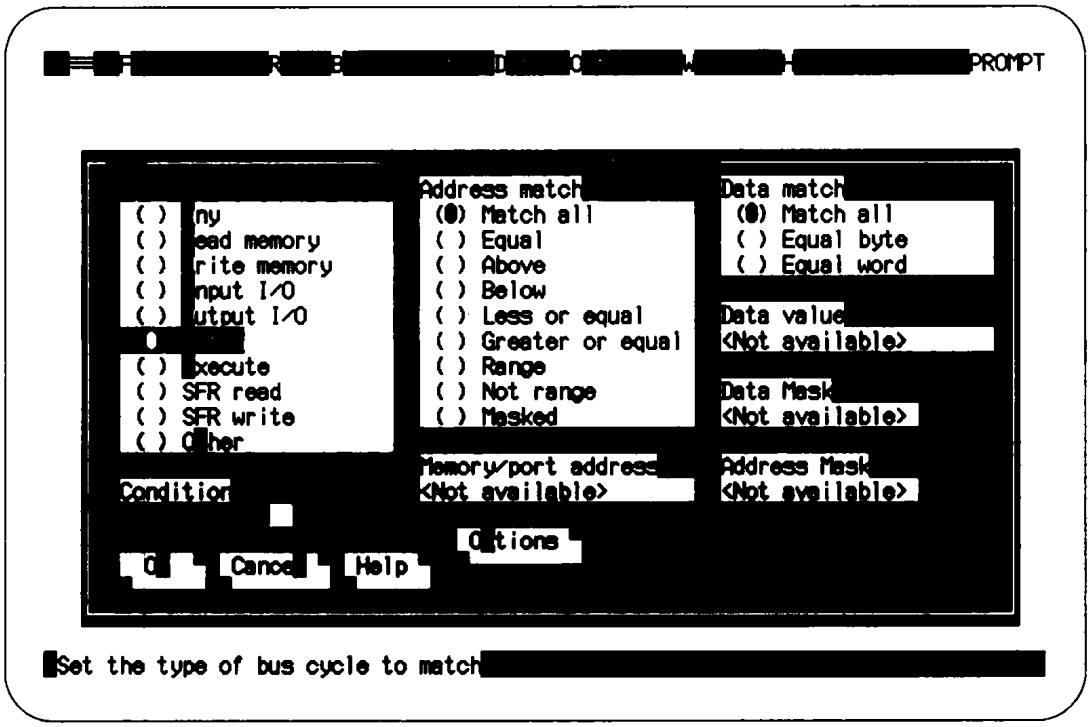
13.2　ハードウエア・ブレークポイント

TDシリーズでは，ハードウエア／ソフトウエア・ブレークポイントを混在して設定できます。

ソフトウエア・ブレークポイントはディフォールトでセットされています。ハードウエア・ブレークポイントはBreakpoints | Hardware breakpointコマンドで明示的に要求した場合だけ使用されます。

Hardware breakpointコマンドをBreakpointsメニューから選択すると，ブレークポイントをトリガし，Hardware breakpointダイアログ・ボックスを表示します。

図13－1　Hardware breakpointダイアログ・ボックス（Bleakpoints | Hardware breakpoint）



このダイアログ・ボックスは，プログラムの実行を停止するために使用した条件を収集します。TD423, TD433では，違うパターン設定を４回行うと最高４つのハードウエア・ブレークポイントを設定できます。ただし，イベント・レジスタのどれかをトレース・トリガが使用している場合には，４つのうちの３つだけを受け付けます。

ハードウエア・ブレーク条件は次のラジオ・ボタン，ボタン，入力ボックスで指定します。

(1) Cycleラジオ・ボタン

バス・サイクル・タイプを指定します。

(2) Address matchラジオ・ボタン

アドレスの成立条件を指定します。

- Match all　：常に成立
- Equal　：Memory/port address入力ボックスの値と一致したとき成立
- Above　：Memory/port address入力ボックスの値より大きいとき成立
- Below　：Memory/port address入力ボックスの値より小さいとき成立
- Less or equal　：Memory/port address入力ボックスの値以下になったとき成立
- Greater or equal：Memory/port address入力ボックスの値以上になったとき成立
- Range　：Memory/port address入力ボックスの値の範囲内になったとき成立
- Not range　：Memory/port address入力ボックスの値の範囲外になったとき成立

● Masked　　：Memory/port address入力ボックスの値がAddress Mask 入力ボックスでマスクした値と一致した場合に成立

(3) Data matchラジオ・ボタン

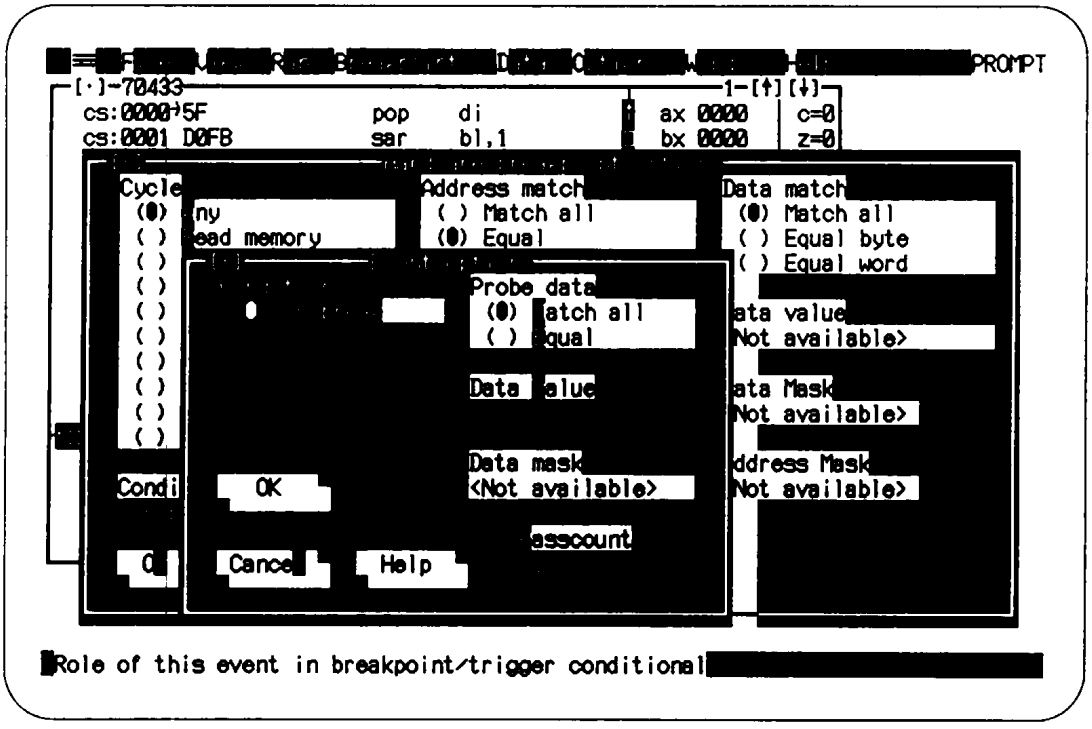
データの成立条件を指定します。

- Match all　：常に成立
- Equal byte ：Data value入力ボックスに指定した値とバイト単位で一致した場合に成立
- Equal word：Data value入力ボックスに指定した値とワード単位で一致した場合に成立

(4) Optionボタン

Optionボタンを押すと，Event optionダイアログ・ボックスが開きます。パス・カウントや外部信号入力（外部ロジック・プローブ使用）によるブレークを指定できます。

図13－2　Event optionsダイアログ・ボックス（Optionsボタンを選択）



(a) Pass count入力ボックス

パス・カウント（2-65535）を指定します。

(b) Probe dataラジオ・ボタン

外部ロジック・プローブが特定の外部入力信号を検出したときのハードウエア・ブレークポイントを設定できます。ここで設定した条件が成立するかまたはハードウエア・ブレーク条件が成立したときブレークが発生します。

Probe data設定例

ブレーク条件：外部ロジック・プローブが11010011b (D3h) を検出したときにブレークを発生する

- Probe data (●) Equal　　：Equalを選択
- Data value 11010011b (またはD3h)　　：値を入力

ハードウエア・ブレークポイントの条件をすべて指定したら，OKを選択します。これで，TDシリーズがこのブレークポイントを受け入れます。

注意　ブレークポイント位置

ハードウエア・ブレークポイントが発生するのは，常にブレークポイントが認識されたあとです。したがって，目的のイベントは現在のPS：PCレジスタの直前に設定してください。

13.3　ソフトウエア・ブレークポイント

Breakpoints | Hardware breakpointコマンドを使用しないでブレークポイントを設定する場合は，ソフトウエア・ブレークポイントによってセットします。ソフトウエア・ブレークポイントのセットには，IEからのハードウエア・リソースがまったく必要ありません。無制限で自由にセットできます。

ソフトウエア・ブレークポイントは，プログラムの実行停止以外に，ログや式の評価を更新できます。ただしこれは，Breakpoint optionsダイアログ・ボックスから選択されている場合にかぎります。ソフトウエア・ブレークポイントは，ブレークポイントに到達するたびにログや式の評価を更新できます。ソフトウエア・ブレークポイントの設定の詳細は第９章　式を参照してください。

注意　ソフトウエア・ブレークポイント自体には制限はありませんが，パス・カウントや条件，式の評価を行うイベントが発生するとエミュレーションが短時間停止します。プログラムをリアルタイムで実行するためには，複雑なソフトウエア・ブレークポイントの使用を避けてください。

13.4　ハードウエア／ソフトウエア・ブレークポイントの相違点

ハードウエア・ブレークポイントとソフトウエア・ブレークポイントの違いは，ブレークポイントが認識され受け入れられたときのPS：PCの内容にあります。

- ソフトウエア・ブレークポイント…プログラム実行と同期しているため，ブレークポイント・アドレスにある命令が実行される前に発生します。
- ハードウエア・ブレークポイント…プログラムの実行には非同期であり，ブレークポイント条件を発生させる命令を実行後にブレークします。

このような違いにより，ソース行にセットされたハードウエア・ブレークポイントがブレークポイントの発生時にCPUウインドウをオープンしてしまうことがあります。ただしPS：PCの現在の内容が，偶然に新しいソース行の開始点にあればそうなりません。

IEを使用してセットした場合，コードのブレークポイントができるアクションは，エミュレーションの停止だけです。したがって，ソフトウエア・ブレークポイントを使用してセットしたコード・ブレークポイントとほぼ同じです（ただし，パス・カウントや式の評価が含まれた場合は別です）。

13.5 Trace triggerダイアログ・ボックス

IEのトレース・トリガにより，指定のイベントの発生後にプログラムの実行トレースを見ることができます。

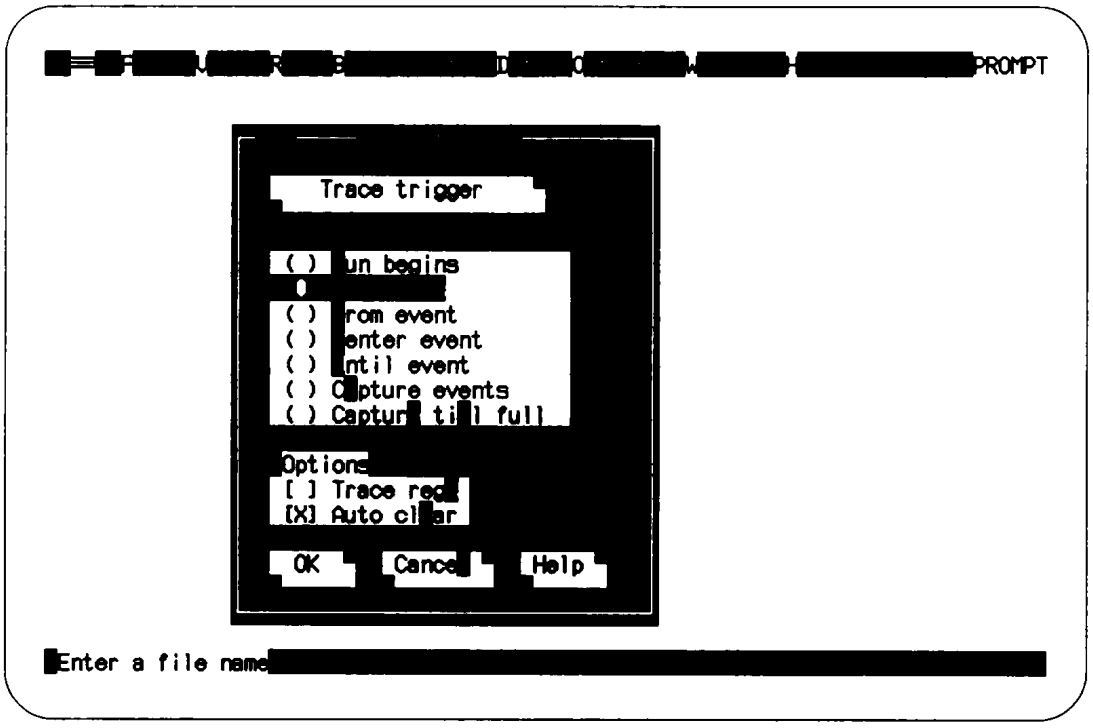
TDシリーズでは，トレースポイントの設定，トレース・バッファに記録されたバス・サイクルのソース，ミックス，バス・サイクルのビューを完全にサポートしています。ユーザは，無用なサイクルをトレース・バッファ・ウインドウから削除することができるなど，トレース・バッファを完全に制御できます。

IEのトレース・バッファは，TDシリーズによって保持される実行ヒストリとは関係ありません。実行ヒストリは，命令のステッピングを使用し，CPU状態全体を追跡してバック・トレースを可能にするためにあります。IEのトレース・バッファは，実行バス・サイクル・トレースのリアルタイムでの記録です。これはエミュレータ内にあり，アドレス，データ，ステータス・バスを含んでいます。

13.5.1 トレースポイントの設定

トレースポイントは，IEのトレース・バッファ内で実行トレースの記録に関する基準を選択する方法です。ICEメニューから選択されるとTrace triggerコマンドがダイアログ・ボックスをポップアップします。このダイアログ・ボックスによってトリガ条件とトレース・バッファ内でのトリガポイントを動かすためのトレース・モードを選択できます。

図13-3 Trace triggerダイアログ・ボックス（View | ICE | Trace trigger）

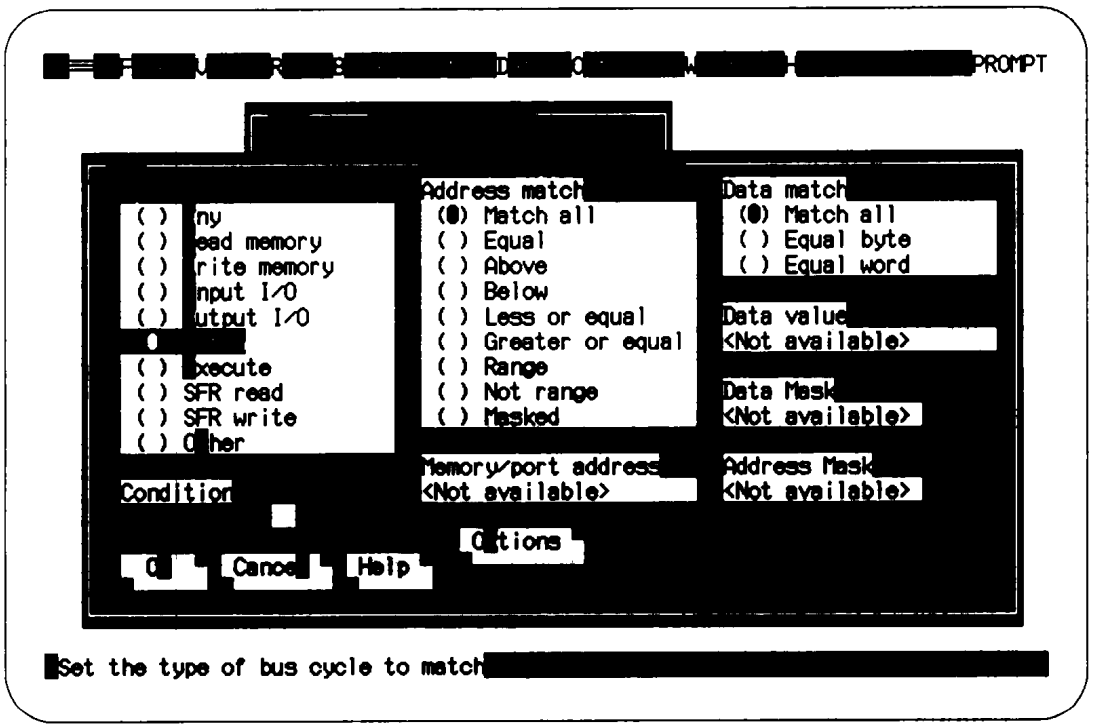


13.5.2　triggerオプション

実行トレースの記録を可能にするために使用するトリガ条件は，Trace triggerダイアログ・ボックス内のTrace triggerボタンからアクセスするダイアログ・ボックスで選択します。バス・サイクル・タイプ，アドレスとデータ・バスの内容を選択するフィールドを含んだダイアログ・ボックスが表示されるので，指定したいトレース条件を選択します。

IEのトレース・トリガ・ロジックのハードウエアのために，トリガ条件のアドレス範囲指定には制限があります。

図13－4　トリガ条件を選択するダイアログ・ボックス（View | ICE | Trace trigger→Trace trigger）



13.5.3　Trace modeオプション

Trace Modeオプションは，トリガ・リングとデータ収集のためのトレース・バッファ・モードの選択に使用します。

IEは，トレース・バッファがいっぱいになったとき，エミュレーションの停止もサポートしています。このオペレーション・モードを使いたい場合は，Trace triggerダイアログ・ボックスでBreak when Fullを選択してください。

(1) Run begins

エミュレーションが始まると，トレース・バッファはプログラム実行の現在の点からトレース・バッファがいっぱいになるまでデータを記録します。検査されたとき，トレース・バッファは実行の開始から最大8191のバス・サイクルを保持します。

このトレース・モードではイベント・レジスタが動作する必要はありません。

(2) Halt ends

トレース・バッファの動作が許可され，エミュレーションのブレークポイント条件が発生するまでデータは継続的に収集されます。このモードではトレース・バッファはブレーク条件までに最大8191のバス・サイクルを保持しています。

このトレース・モードではイベント・レジスタが動作する必要はありません。

(3) From event

トレースの記録はトリガ条件が満たされたときに始まり，トレース・バッファがフルになるまで継続します。このモードではトリガ条件は実行トレースの最初に置かれます。

(4) Center event

トレースの記録が許可され，トレース条件が満たされるまで継続します。このモードではトリガ条件は実行トレースの真ん中に置かれます。

(5) Until event

トレースの記録が許可され，トリガ条件が満たされるまで継続します。このモードではトリガは記録されたバス・サイクルに続く実行トレースの最後に置かれます。

(6) Capture events

トレースの記録は外部ロジック・プローブのブレーク条件が満たされたときに始まり，トレース・バッファがフルになるまで継続します。

(7) Capture till full

トレースの記録が許可され，外部ロジック・プローブのブレーク条件が満たされるまで継続します。

13.6　Trace bufferウインドウ

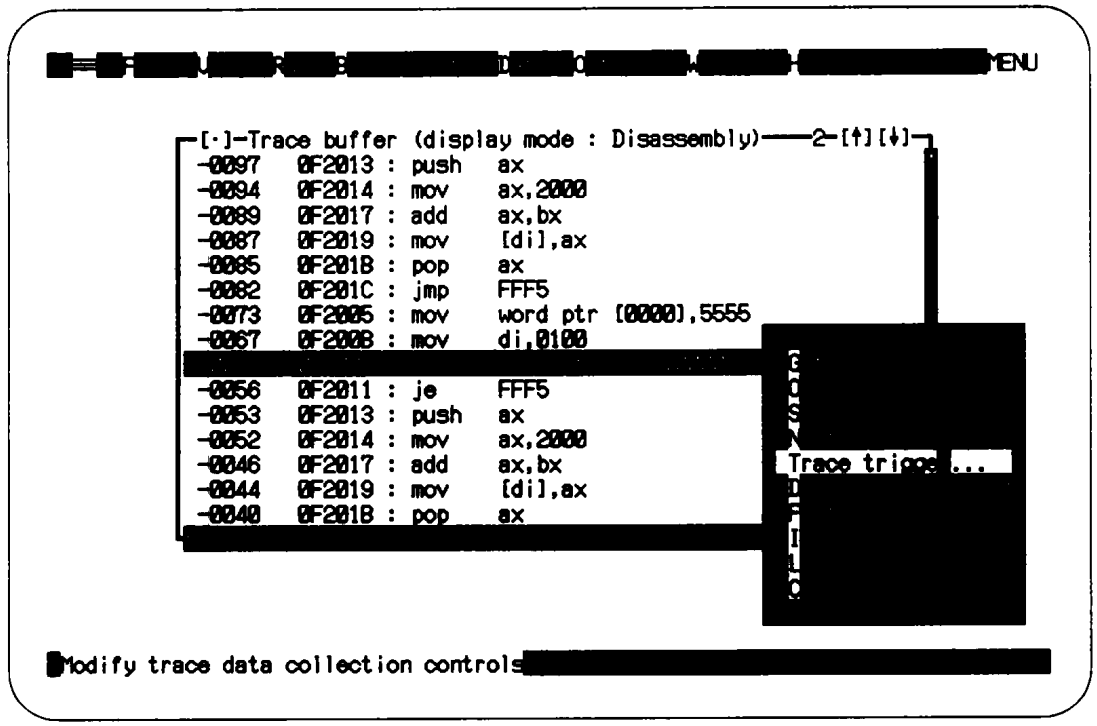
トレース結果がトレース・バッファに記録されたら，View | ICE | Trace bufferコマンドによりトレース・バッファのウインドウを開くと，トレースをいろいろな形式で見ることができます。このため，すぐに目的のイベントを見つけて引き出すことができます。

13.6.1　Trace bufferウインドウのローカル・メニュー

Trace bufferコマンド・ローカル・メニューにより，トレース・バッファのセットアップとビューを制御できます。GRPH-F10を押すと，Trace bufferウインドウのローカル・メニューがポップアップしま

す。コントロール・キー・ショート・カットを許可している場合は，CTRLキーを押したまま，希望のコマンドの最初の文字のキーを押せばコマンドに直接アクセスできます。

図13－5　Trace bufferウインドウとそのローカル・メニュー（View | ICE | Trace buffer）



(1) Gotoコマンド

トレース・バッファ内での表示開始点を選択します。このコマンドを使うとトレース・バッファの指定点にすぐに移動できます。

(2) Originコマンド

トレース・バッファ内での表示を最初のトリガのオフセットに移動します。これはトレース・バッファ内のトリガ・ポイントを見つける一番早い方法です。

(3) Searchコマンド

トレース・バッファ内でイベントを探すときに，そのイベントを指定するためのダイアログ・ボックスをオープンします。イベントを検索するオプションの詳細は，**13.9　トレース・バッファの検索**を参照してください。

(4) Nextコマンド

トレース・バッファを検索条件が合う次の場所へ移動します。これはトレース・バッファ中で検索条件の複数の発生を見つけるときに使用します。

(5) Display modeコマンド

トレース・バッファの表示形式を選択します。実行トレース表示はソース・オンリーから各バス・サイクルまで各種の形式で表示できます。トレース表示モードの選択オプションは，13.7　実行トレースを見るを参照してください。

(6) Trace triggerコマンド

トレース・バッファを見ながら新しいトレースポイントをセットするTrace triggerコマンドへのショート・カットです。

(7) Filtersコマンド

バス・サイクル・フィルタ・ダイアログ・ボックスをオープンします。13.8　バス・サイクル・フィルタでは，トレースから目的以外のバス・サイクルを削除する効果的な方法を説明しています。

(8) Inspectコマンド

現在トレース・ウインドウ中のハイライト・バーに対応するソース・プログラムを表示します。

(9) Logコマンド

現在のトレース・バッファの内容をファイルにセーブします。

(10) Clearコマンド

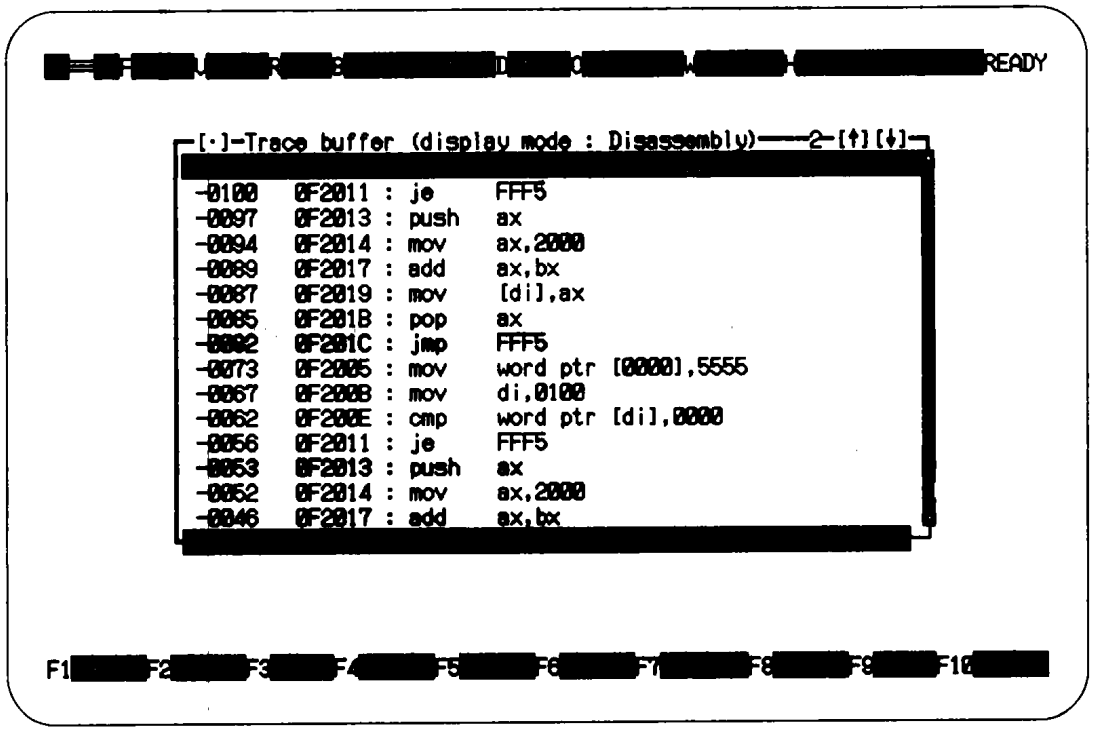
トレース・バッファの内容をクリアします。

13.7　実行トレースを見る

いったん実行トレースが記録されると，トレース・データを見るのにいろいろな方法があります。たとえば，ソース・レベルのビューを使い高級言語の文をトレース・バッファ中に見ることができます。また，同じデータをプロセッサの一連のバス・サイクルとして見ることもできます。この場合には，ターゲット・システムのバス動作のきわめて詳細な動作を見ることができます。

TDシリーズでは，ソース・コード，命令ニモニックおよびバス・サイクルをIntermediateビューで表示することができます。

図13－6　Trace bufferウインドウ（View | ICE | Trace buffer）



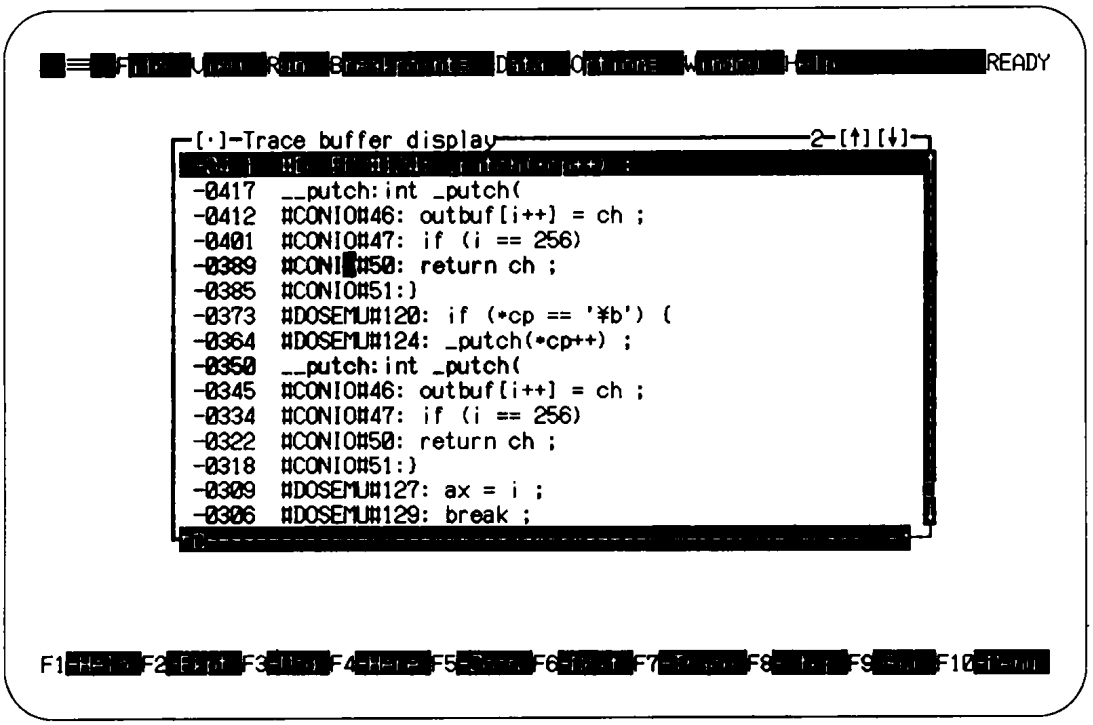
上図のように各トレース・バッファ・エントリにはトレース・トリガとの関係を表す番号が付いています。マイナスの行番号はトリガ以前のイベントであり，トリガ点から離れるほどマイナス値が大きくなります。正の整数はトリガのあとのイベントを反映していて，トレース・トリガから離れるほど数値が大きくなります。トリガ条件を満足した各トレース・バッファ・エントリには“＊”マークも付いているため，トリガ・イベントとの関係が分かります。

トレース・バッファの選択されたビューは，Display modeローカル・コマンドによって制御します。ローカル・メニューをオープンしDisplay modeコマンドを選択するか，CTRL-Dのショート・カットを使うと次の４種類の表示が順に回転します。

(1) Sourceコマンド

Sourceビューは最高レベルのビューです。トレース・バッファ内の実行されたコードのモジュールとソース行を表示します。Sourceビューの各行には，TDシリーズの形式でトレース・バッファの行番号，ソース・モジュールおよび行番号があり，続けてその行のソース・コードがあります。

図13－7　Trace bufferウインドウのSourceビュー

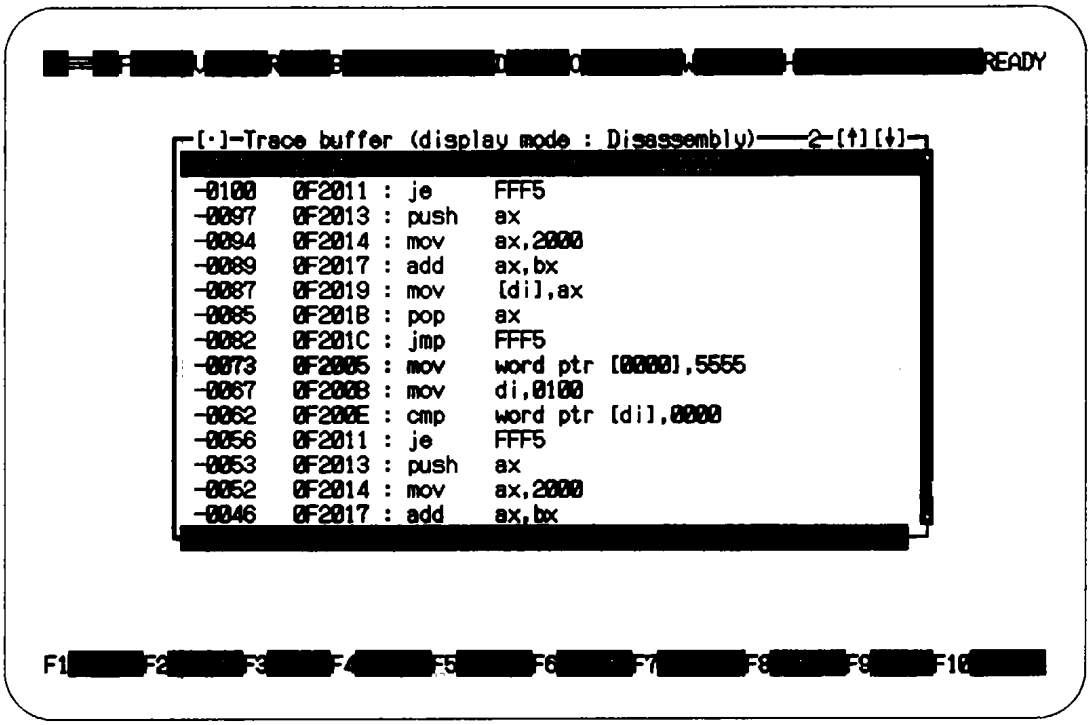


Sourceビューは処理に時間がかかります。特にトレース・バッファにソース行がなかった場合には顕著です。トレース・バッファの内容がフォーマットされている間，トレース・バッファ表示の状態を知らせるためにメニュー・バーの右側のREADYインジケータが点滅します。

(2) Disassemblyコマンド

Disassemblyビューは，ソース・ビューと各行に相当する命令のアセンブリ・ビューの組み合わせです。表示される各アセンブリ命令にはトレース・バッファ行番号とアドレス・バスの内容に続き，逆アセンブルされた命令があります。

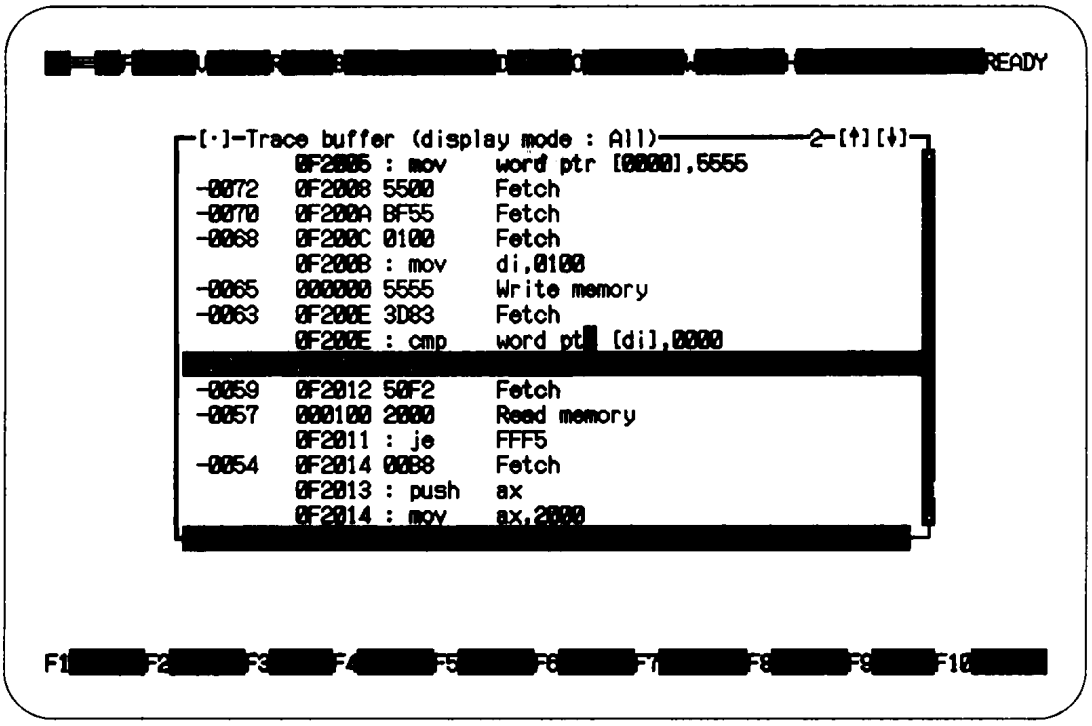
図13－8　Trace bufferウインドウのDisassemblyビュー



(3) Allコマンド

Allビューには Disassemblyビューに加えてトレース・バッファからのバス・サイクル情報があります。表示されるそれぞれのバス・サイクルはトレース・バッファ行番号，アドレスおよびデータ・バスの内容さらにバス・サイクルのタイプと続きます。

図13－9　Trace bufferウインドウのAllビュー

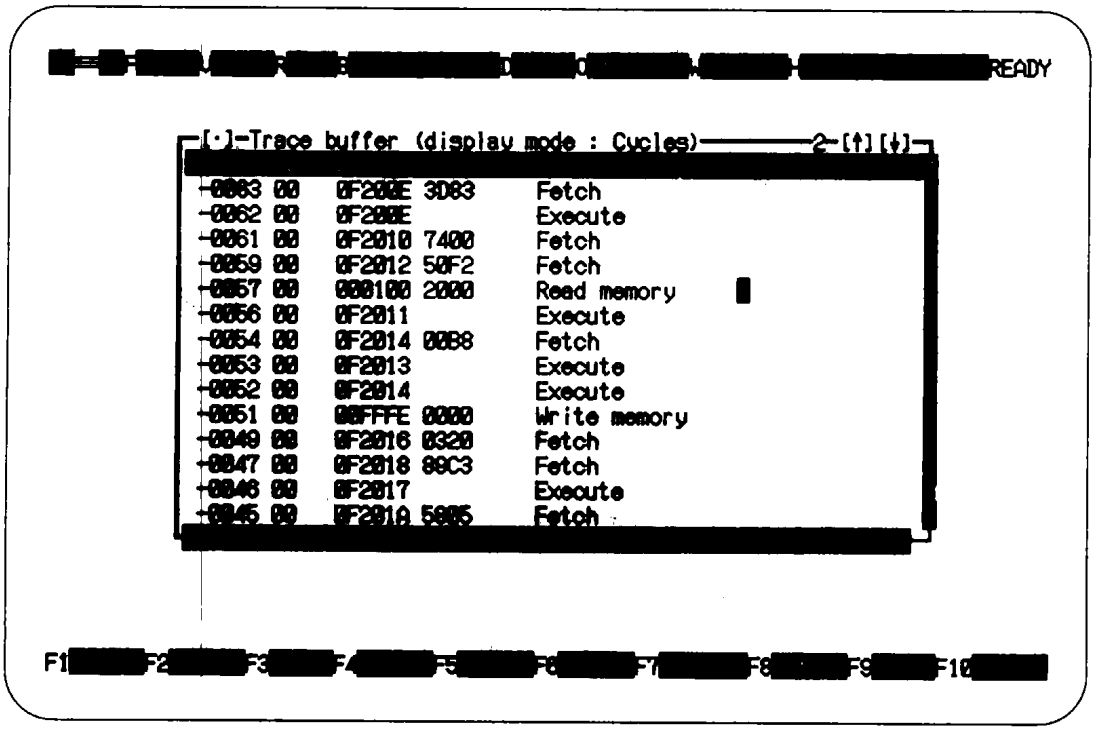


16ビット・システム上では，データ・バスは上位／下位の２バイトです。ハイかロウのどちらか一方だけが表示されるため，８ビット・アクセスと16ビット・アクセスが区別できます。

(4) Cyclesコマンド

もっともシンプルなトレース・バッファ・ビューです。このビューでは記録された各バス・サイクルごとにトレース・バッファ行番号，アドレス・バス，データ・バスおよびバス・サイクル・タイプが表示されます。

図13－10　Trace bufferウインドウのCyclesビュー

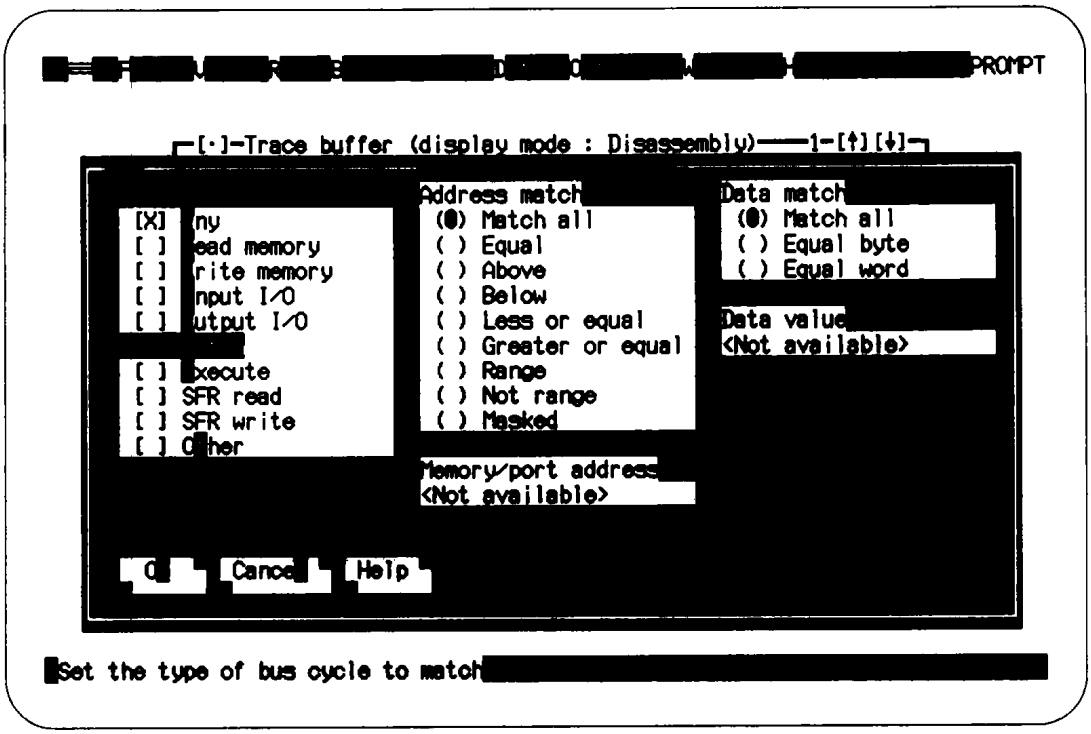


13.8　バス・サイクル・フィルタ

バス・サイクルの検索は１つ以上のバス・サイクル・タイプを対象とし，トレース・バッファ全部を検索するため時間がかかります。また，間違いも起こりがちです。このため，TDシリーズでは目的のバス動作を発見するのに表示フィルタを使用します。

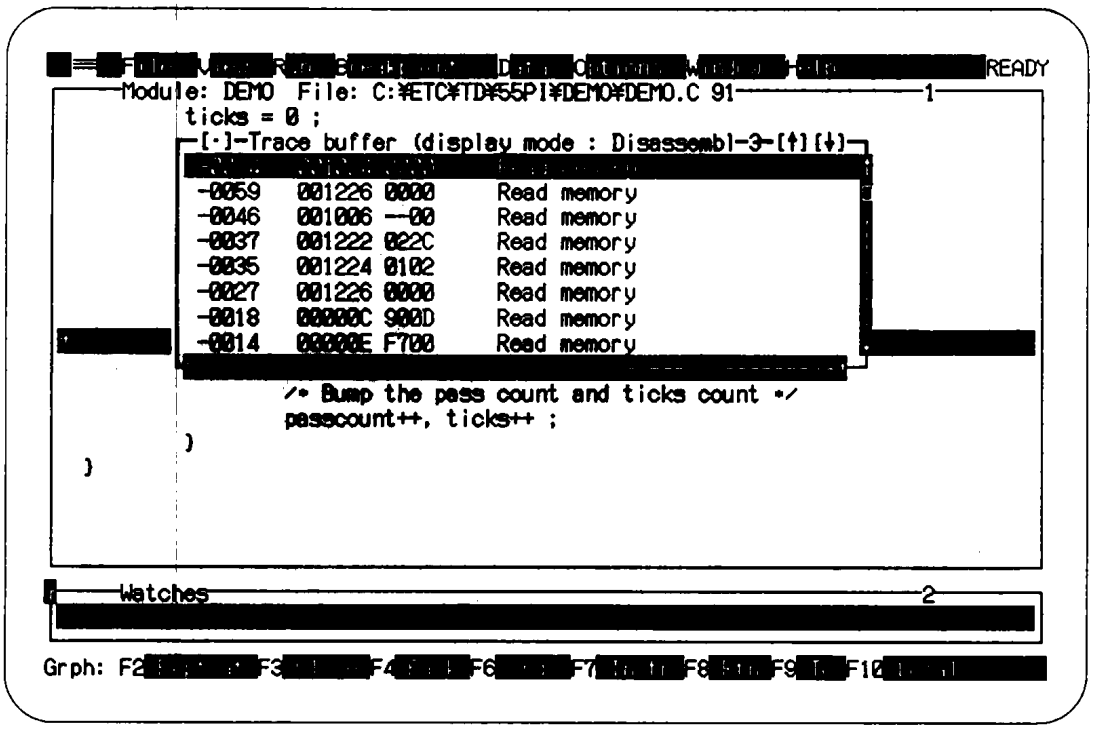
Trace buffer | Filters選択により，ダイアログ・ボックスを開き，関心のあるバス・サイクルを選択します。適切なラジオ・ボタンを選択し，表示基準に見合うバス・サイクルを選択してください。さらにトレース表示の対象をしぼるにはアドレスおよびデータ・フィルタ・オプションかOptionsボタンを使います。

図13－11　Filtersコマンドのダイアログ・ボックス



たとえば同じバス・サイクル表示に，上図のように指定したバス・サイクル・フィルタを使うと，次のようにメモリ・リード・サイクルをトレース表示します。

図13－12　Trace bufferウインドウのメモリ・リード・サイクル・ビュー（バス・サイクル・フィルタを使用のとき）



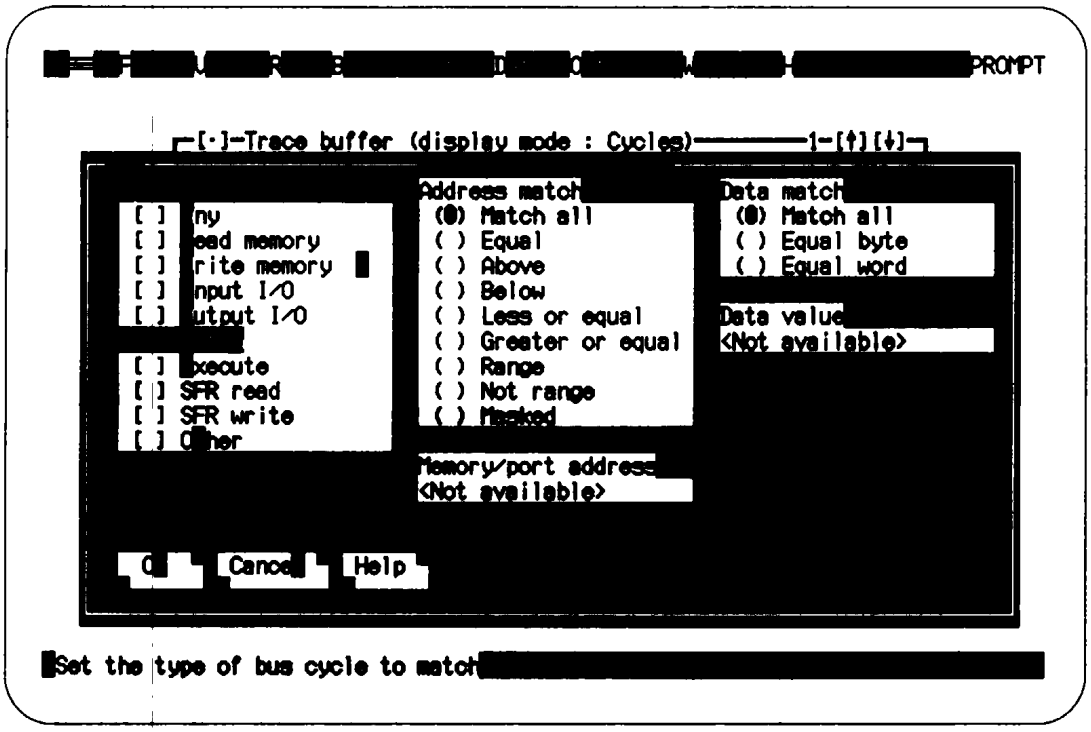
フィルタ・オプションによっては表示モードを強制的に変更する場合があります。たとえば，実行バス・サイクルがTDシリーズによってフィルタ・アウトされると実行命令ストリームを表示するのは不可能なため，バス・サイクル・ビューに強制変更されます。

13.9　トレース・バッファの検索

実行トレースが記録されると，トレース・バッファの中で求めているイベントの検索ができます。トレース・バッファに入れることができるエントリは数千個にも及びますが，この検索機能を使うと，トレース・バッファを特定の位置にスクロールする必要がなくなり，大変便利です。

検索基準の定義はトレースポイントの設定と似ています。Searchコマンドを選択するとダイアログ・ボックスがオープンし，目的のイベントを指定できます。

図13－13　Searchコマンドのダイアログ・ボックス



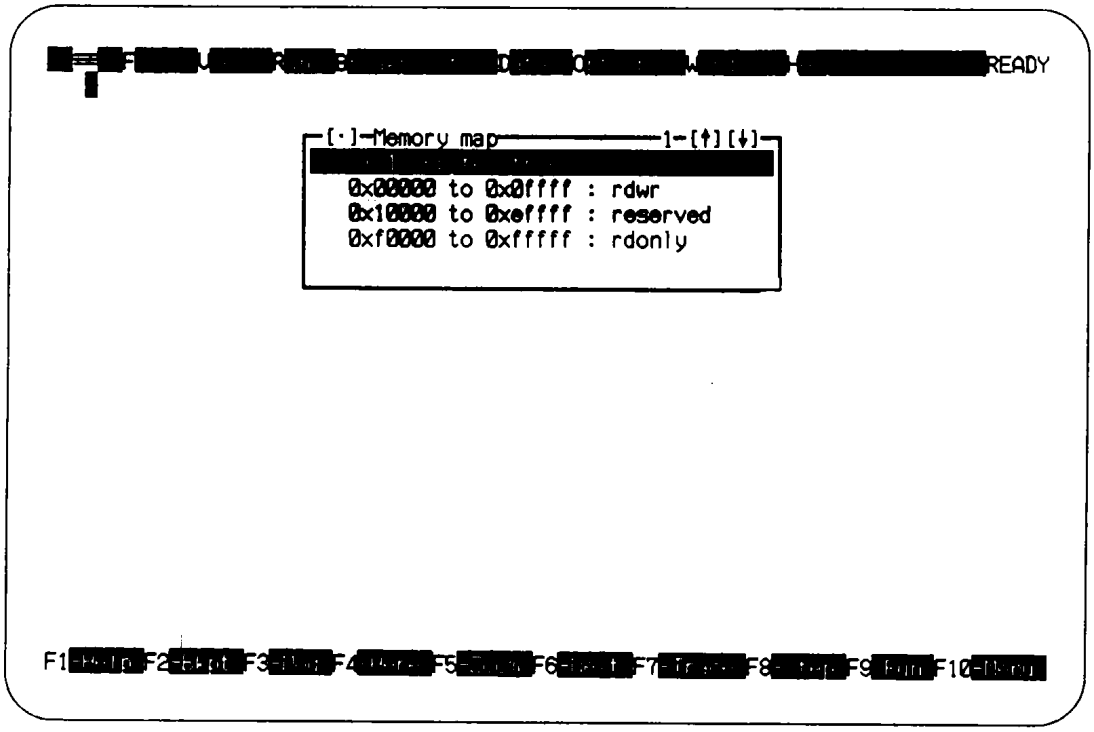
トレース・バッファ中で検索しているイベントの次の回の発生を見つけるには，ローカル・メニューからNextを選択するかCTRL-Nのショート・カットを使います。

13.10　エミュレーション・メモリ管理

IEにはターゲット・システムのアドレス空間を，アドレス範囲とアクセス・タイプによって定義されるあるアドレスにマップする機能があります。アドレス・スペースの分割機能に加えて，このエミュレータにはエミュレーション機能があり，ターゲット・システムのメモリ空間に存在するかのように使用できます。

View | ICE | Memory mapコマンドでは，ターゲット・システム・メモリ・マップの現在の状態へアクセスできます。ローカル・メニュー・コマンドを使うとアプリケーションとターゲット・システムの要求に応じてメモリ・マップを変更したり，1アドレス・スペースの内容をほかのアドレスへコピーできます。

図13－14　Memory mapウインドウ（View | ICE | Memory map）



IEのメモリ・マップ表示には，現在の割り付けのソートされたリストとともに，割り付けに使用可能なエミュレータ・メモリ量が表示されます。rdonlyまたはrdwrとマークされた領域では，IEのエミュレーション・メモリ機能を使用します。targetまたはreservedとマークされた領域は，ターゲット・システム中に存在するメモリにマッピングされるかno accessとなります。

13.10.1　Memory mapローカル・メニュー

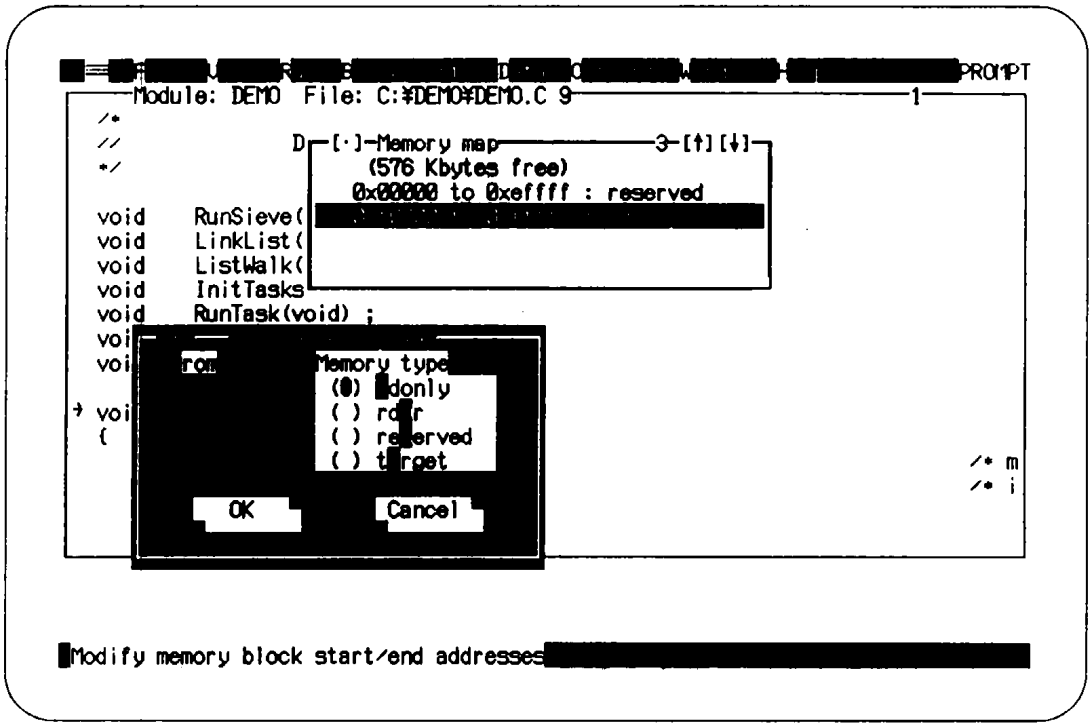
ウインドウのローカル・メニューにはChange，Move，Release，Targetの４つのコマンドがあります。CTRLキー・ショート・カットが可能ならば，CTRLキーと希望のコマンドの最初の文字を使ってコマンドへアクセスできます。

(1) Changeコマンド

現在のアドレス空間マッピングは，メモリ・マップ・ビューにハイライト・バーがあればいつでもChangeコマンドを使って変更できます。Changeコマンドはディフォールトの動作なので，変更したい領域を選択し↵を押すと，現在のメモリ・マップに対して変更要求が出ます。

次にダイアログ・ボックスがポップアップします。このダイアログ・ボックスには，メモリ・アドレス空間の割り付けを変更するのに必要なフィールドがあります。

図13－15　Change memory mapダイアログ・ボックス（ローカル・メニューから選択）



(a) From

マッピング属性を割り付ける領域の開始アドレスを設定します。

(b) To

マッピング属性を割り付ける領域の終了アドレスを設定します。

(c) Memory type

IEには，ターゲットとエミュレータのどちらのメモリかを決定するアクセス・タイプがあります。rdonlyまたはrdwrとマッピングされたアドレス空間領域はエミュレータ・メモリとして，使用可能なエミュレーション・メモリの最後までが割り付けられます。

ターゲット属性のある領域へのメモリ・アクセスはすべてターゲット・システム中にあるメモリにマッピングされます。reserved属性のある領域へのメモリ・アクセスはすべて禁止され，エミュレーションが例外で停止します。

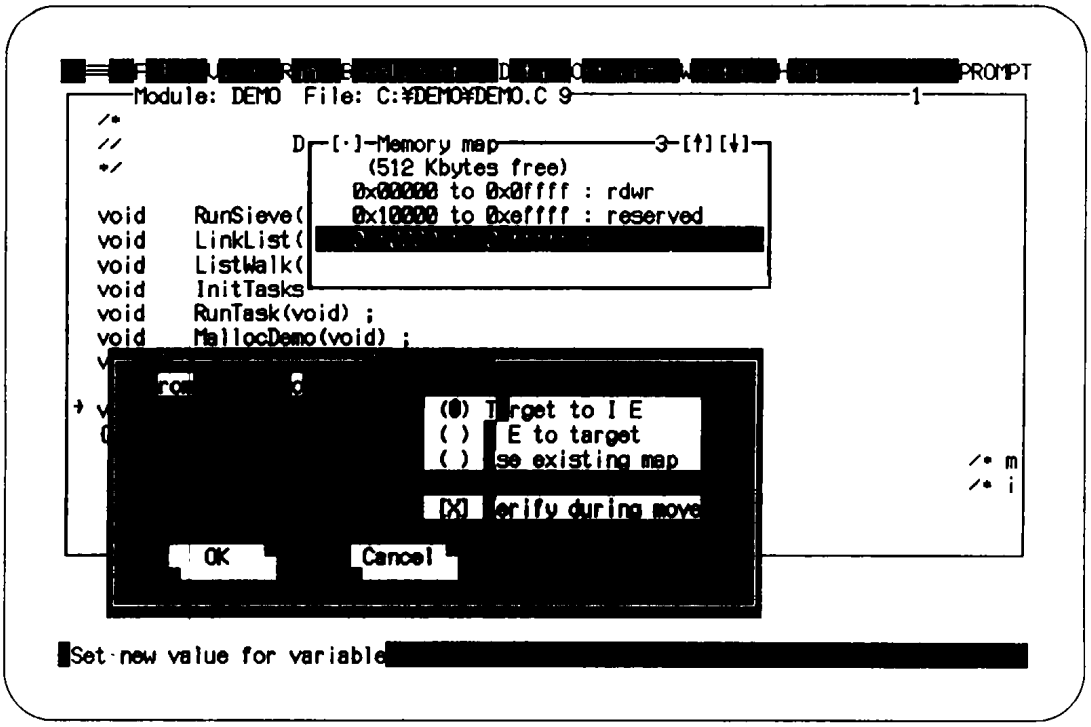
(2) Moveコマンド

1アドレス空間の内容を別のアドレス空間へ移動することがあります。たとえば，ターゲット・システムのEPROMメモリをエミュレータ・メモリへコピーして，EPROM中でソフトウエア・ブレークポイントが設定できるようにする場合などです。

Moveコマンドで次のダイアログ・ボックスが開き，2つのメモリ領域と操作内容の入力が求めら

れます。

図13－16　Move memoryダイアログ・ボックス（ローカル・メニューから選択）



(a) From

コピー元領域の開始アドレスです。

(b) To

コピー元領域の終了アドレスです。

(c) Destination

コピー先の開始アドレスです。領域の大きさは，FromとToのフィールドにより決定されます。

(d) Action

コピー・オペレーションを選択するラジオ・ボタンです。

(e) Verify

コピー後ベリファイし，内容が違っていたらエラーを報告します。

(3) Releaseコマンド

現在選択されているメモリ・マップ・エントリをTargetタイプとしてマークします。これは，以前にマップされたアドレス空間領域を解放するショート・カットです。

(4) Targetコマンド

ターゲットの状態が表示されます。

13.10.2　ディフォールト・メモリ・マッピング

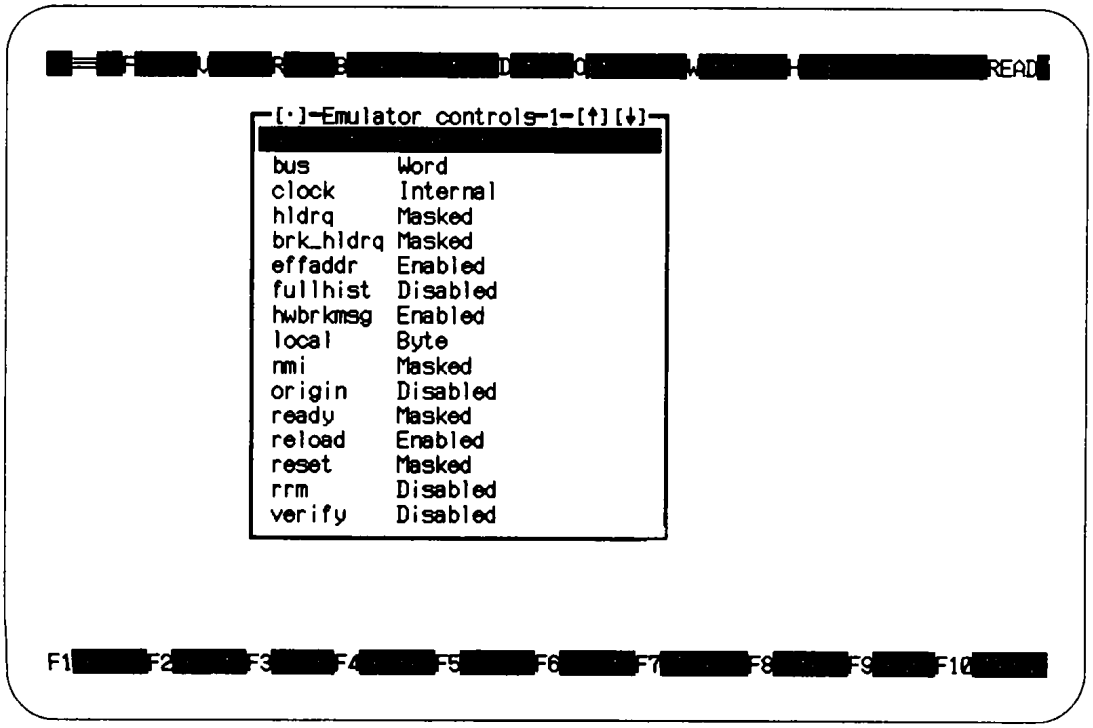
エミュレータ・セットアップ・ファイルやロード・プログラム中のマッピング命令に何も指定がなければ，IEはディフォールトですべてのメモリをターゲット・システム・アドレス空間にマッピングします。

13.11　エミュレータ制御

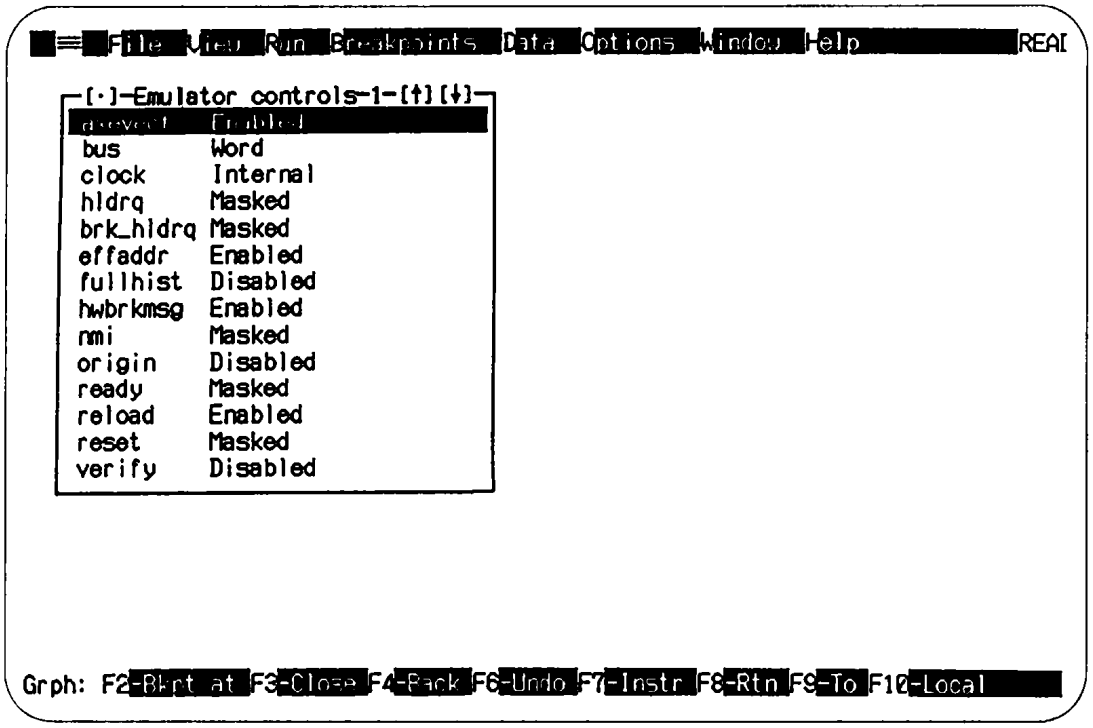
IEには，CPUのクロック・ソース，入出力端子の状態およびその他のエミュレータに関係したパラメータを選択する多くのエミュレータ制御機能があります。TDシリーズは，ICEメニューのEmulator controlsウインドウの現在のエミュレータ制御を見て，新しい値を設定するための便利な機能を備えています。

図13－17　Emulator controlsウインドウ（View | ICE | Emulator controls）

(a) TD423の場合



(b) TD433の場合



設定内容は，ローカル・メニューのChangeコマンドを実行するごとに反転します(rrmコマンドを除く)。

一度設定すると，エミュレータ制御はコンフィギュレーション・ファイルに保存でき，TDシリーズが実行するたびに自動的に，またはRestore setupコマンドを使用してロードされます。

(1) clockコマンド

クロック・ソースの選択をします。

・Internal ：エミュレータ内部のクロックを使用します。

・External：ターゲット・システムの実装クロックを使用します。

(2) hldrq, brk_holdrq, nmi, ready, resetコマンド

CPUへの端子入力は，IEでマスクすることができます。これは，不完全なシステム上でこれらの端子を使用する場合，不安定動作を防止するための機能です。

・Masked　：次の入力をマスクします。

・Unmasked：次の入力のマスクを解除します。

コマンド名	対象入力
hldrp	HLDRQ
brk_holdrp	IEがブレーク中のHLDRQ
nmi	NMI
ready	READY
reset	RESET

(3) axevectコマンド

リセット解除後のプログラムの開始アドレスを選択します。

・Enabled ：ロード・モジュールで定義されたスタート・アドレスを開始アドレスとします。
・Disabled：FFFF0h番地を開始アドレスとします。

(4) effaddrコマンド

データ・ペイン内でのメモリの先読みを指定します。

(5) fullhistコマンド

インストラクション・トレース時のフル・ヒストリ・モードを指定します。[注]

注　フル・ヒストリ・モードをEnabledとした場合，実行スピードは遅くなりますがバック・トレースができます(Disabledとした場合，バック・トレースはできません)（**5.2.8 Back traceコマンド参照**)。

(6) hwbrkmsgコマンド

ハードウエア・ブレーク発生時にブレークポイントを示すダイアログ・ボックスを指定します。

(7) originコマンド

ICE reset, Program resetコマンドを実行したとき，またはプログラム・ブレークが発生したときのカレント・プログラム・ポジション（ソース・モジュール・ウインドウのハイライトされている行）を指定します。

●Enabled ：リセット／ブレーク後，自動的にカレント・プログラム・ポジションを現在のカレント・ポジションとする。
●Disabled：カレント・プログラム・ポジションは動かさない。

(8) reloadコマンド

IE resetコマンド，Program resetコマンドの実行時に，ロード・モジュールをリロードする機能を指定します。

(9) verifyコマンド

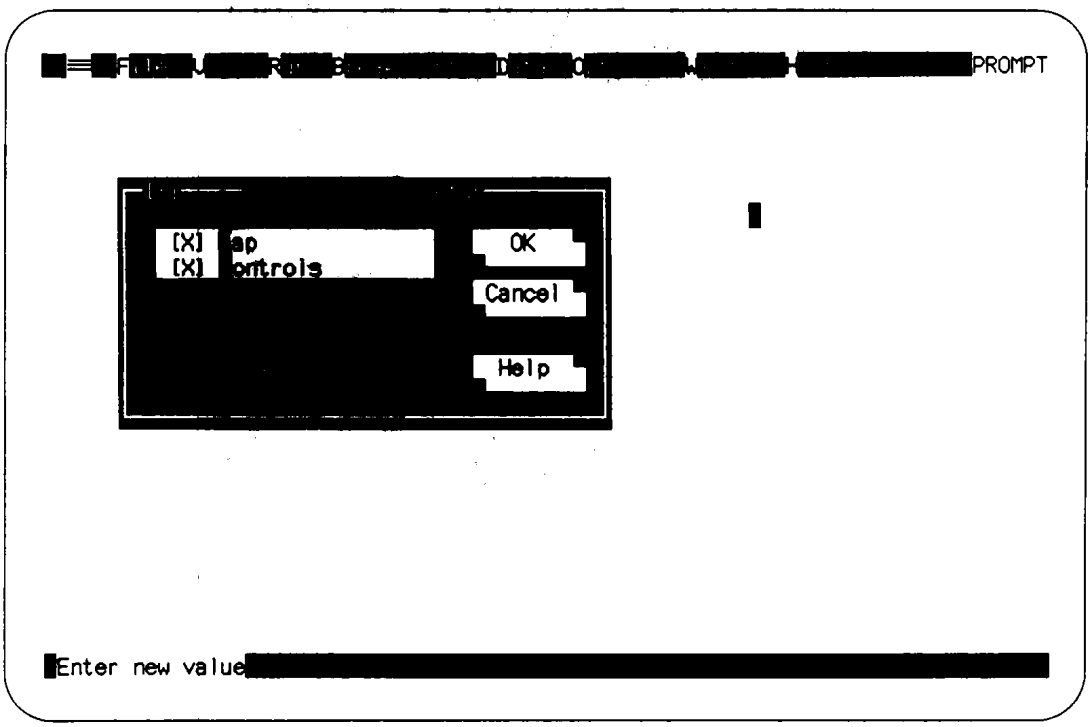
メモリ・ライト時のベリファイ・チェック機能の許可／禁止を設定します。

13.12　セットアップの保存

Save setupダイアログ・ボックスは，IEの現在の状態をコンフィギュレーション・ファイルに保存し，TDシリーズがIE用に自動的にディフォールト状態を設定するためにあります。全体的にも，またはプロジェクトごとにも可能です。Save setupダイアログ・ボックスを開くにはViewメニューからICE | Save setupを選択します。

プログラムをロードしていなければ，自動的にディフォールトのファイル名TD423BX(TD433BX).EMUがコンフィギュレーション・ファイルになりますが，ほかのファイル名を選択してもかまいません。

図13－18　Save setupダイアログ・ボックス（View | ICE | Save setup）



エミュレーション・メモリ・マップ，エミュレータ制御，または両方をコンフィギュレーション・ファイルに保存可能です。

- Map　　エミュレータ・コンフィギュレーション・ファイルに現在のメモリ・マップを保存します。
- Controls　エミュレータ・コンフィギュレーション・ファイルに現在のエミュレータ制御を保存します。

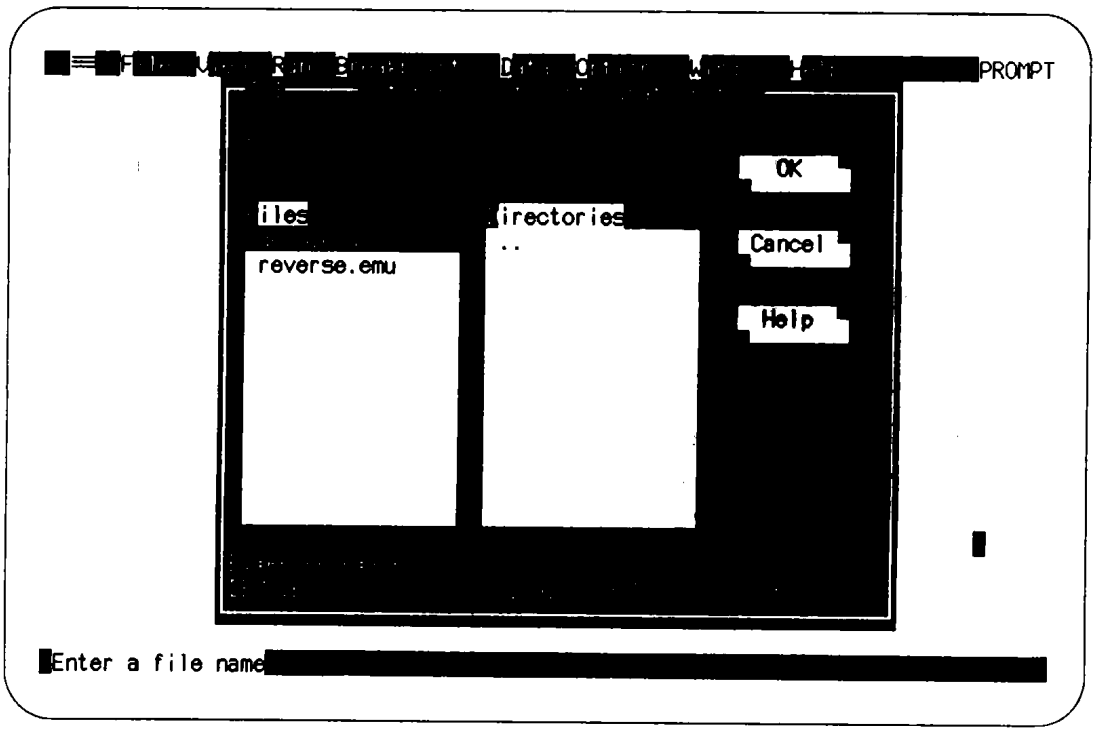
注意　エミュレータ・コンフィギュレーション・ファイルは，TDシリーズのコンフィギュレーション・ファイル（マクロやスクリーン・レイアウトの保存に使用）とはまったく別のものになっていますので注意してください。

13.13　セットアップのリストア

このコマンドにより，以前に保存したエミュレータ・コンフィギュレーションをSave setupコマンドで生成したファイルから元に戻すこと（リストア）ができます。

このコマンドが選択されると，Restore setupコマンドがIEのセットアップ・ファイルのファイル・リストのあるダイアログ・ボックスをポップアップします。

図13－19　Restore setupダイアログ・ボックス（View | ICE | Restore setup）



TDシリーズが開始すると，同じディレクトリ中にロードされたプログラムと同じ名前で.EMU拡張子のあるエミュレータ・コンフィギュレーション・ファイルを探します。この検索が失敗するとTDシリーズはもう一度PATH環境変数を使い，プログラムがロードされていないときのディフォールト・コンフィギュレーション・ファイルとして使用するファイルTD423BX（TD433BX).EMUを探します。

コンフィギュレーション・ファイルが見つからない場合は，IEはディフォールトに設定されます。

13.14　エミュレータのリセット

ディバグ・セッションを最初からやり直したい場合，RunメニューのICE resetコマンドを使うと初期状態からスタートできます。

ICE resetコマンドを選択すると次のイベントが発生します。

- TDシリーズはエミュレータ状態をリセットし，すべてのイベントとメモリ・マップをクリアするコマンドを送ります。
- ターゲット・システムがリセットされます。
- プログラムがロードされていればリロードします。

注意　エミュレータのリセットはシステム全体にかかる負担が大きいため，頻繁には行わないでください。このコマンドを選択すると，Yes/Noの入力が要求され，本当にエミュレータをリセットするかどうかが確認されます。

〔メ　モ〕

第14章　コマンド・レファレンス

この章では，コマンドに関する次の項目について説明します。

- ファンクション・キーやその他のキーで入力できるすべてのシングルキー・ストローク・コマンド
- すべてのメニュー・バー・コマンドと各ウインドウのローカル・メニュー・コマンド
- ２種類のペイン（テキストを入力するペインと項目を選択するペイン）で使うキー・ストローク
- ウインドウ間で移動したり，ウインドウ・サイズを設定し直すキー・ストローク

14.1　ホット・キー

ホット・キーとは，コマンドをすばやく実行するためのショート・カット・キーのことです。TDシリーズの環境内であればどこでもその動作を実行します。次の表にホット・キーの要約を示します。

キー	メニュー・コマンド	機　　能
F1		状況感知型のヘルプ画面を表示する
F2	Breakpoints | Toggle	カーソル位置にブレークポイントを設定する
F3	View | Module	モジュール・ピック・リストを表示する
F4	Run | Go to cursor	カーソル位置まで実行する
F5	Window | Zoom	カレント・ウインドウをズーム／アンズームする
F6	Window | Next	次のウインドウに移動する
F7	Run | Trace into	１ソース行または１命令実行する
F8	Run | Step over	１ソース行または１命令実行し，呼び出しをスキップする
F9	Run | Run	プログラムを実行する
F10		メニュー・バーを起動してメニューから抜け出す
GRPH-F1	Help | Previous topic	最後のヘルプ画面を表示する
GRPH-F2	Breakpoints | At	特定のアドレスにブレークポイントを設定する
GRPH-F3	Window | Close	カレント・ウインドウをクローズする
GRPH-F4	Run | Back trace	プログラムの実行を反転させる
GRPH-F6	Window | Undo close	最後に閉じたウインドウをもう一度開く
GRPH-F7	Run | Instruction trace	１命令実行する
GRPH-F8	Run | Until return	関数から戻るまで実行する
GRPH-F9	Run | Execute to	指定のアドレスまで実行する
GRPH-F10		ウインドウのローカル・メニューを起動する

キー	メニュー・コマンド	機　　能
GRPH-1～ GRPH-9		1-9の番号がついたウインドウの間で切り替える
GRPH-スペース		システム・メニューに移動する
GRPH-B		Breakpointsメニューに移動する
GRPH-D		Dataメニューに移動する
GRPH-F		Fileメニューに移動する
GRPH-H		Helpメニューに移動する
GRPH-O		Optionsメニューに移動する
GRPH-R		Runメニューに移動する
GRPH-V		Viewメニューに移動する
GRPH-W		Windowメニューに移動する
GRPH-X	File | Quit	TDシリーズを終了してMS-DOSに戻る
GRPH-^	Options | Macros | Create	キー・ストローク・マクロを設定する
GRPH- －	Options | Macros | Stop recording	マクロの記録を終了する
CTRL-F2	Run | Program reset	ディバグ・セッションを停止して再開のためにプログラムをリセットする
CTRL-F4	Data | Evaluate	式を評価する
CTRL-F5	Window | Size/move	ウインドウの移動またはサイズ変更を行う
CTRL-F7	Data | Add watch	Watchesウインドウに変数を追加する
CTRL-F8	Breakpoints | Toggle	カーソル位置でブレークポイントをトグルする
CTRL-F9	Run | Run	プログラムを実行する
CTRL-F10		ウインドウのローカル・メニューを起動する
CTRL-→		CPUウインドウ内のコード, データ, またはスタック・ペイン内にある開始アドレスを1バイト上にシフトする
CTRL-←		CPUウインドウ内のコード, データ, またはスタック・ペイン内にある開始アドレスを1バイト下にシフトする
CTRL-A		前のワードに移動する
CTRL-C		1画面をスクロール・ダウンする
CTRL-D		右に1桁移動する
CTRL-E		1行上に移動する
CTRL-F		次のワードに移動する
CTRL-R		1画面をスクロール・アップする
CTRL-S		左に1桁移動する
CTRL-X		1行下に移動する
SHIFT-F1	Help | Index	オンライン・ヘルプのインデクスに移動する

キー	メニュー・コマンド	機　　能
SHIFT-TAB		カーソルを前のウインドウ・ペインまたはダイアログ・ボックスの項目に移動する
SHIFT-→ SHIFT-← SHIFT-↑ SHIFT-↓		ウインドウ内のペイン間でカーソルを移動する（矢印の方向にあるペインがアクティブ・ペインになる）
ESC		Inspectorウインドウをクローズし，メニューから抜け出す
INS		テキスト・ブロックの選択を開始する。ハイライト表示するには←と→を使う
TAB	Window/Next pane	次のウインドウ・ペインまたはダイアログ・ボックスの項目にカーソルを移動する

14.2　メニュー・バーから選択できるコマンド

F10キーを押すとメニュー・バーをアクティブにすることができます。次の動作を行うと，個々のメニューに直接移動することができます。

- メニュー・タイトルにカーソルを移動して，↵を押す
- メニュー・タイトルでハイライト表示されている文字のキーを押す

また，GRPHキーと目的のメニュー名の先頭文字を押すと，あらかじめメニュー・バーに移動しなくても直接メニューを開くことができます。

(1) ≡（システム・）メニュー

Restore standard	標準ウインドウ・レイアウトをリストアする
Repaint desktop	画面全体を再表示する
About	TDシリーズに関する情報を表示する

(2) Fileメニュー

Open	ディバグする新しいプログラムをオープンする
Change dir	新しいディスクまたはディレクトリに変更する
Get info	プログラム情報を表示する
DOS shell	MS-DOSコマンド・プロセッサを起動する
Symbol load	シンボル・テーブルをロードする
Table relocate	シンボル・テーブルのベース・セグメントを設定する
Quit	MS-DOSに戻る

(3) Viewメニュー

Breakpoints		ブレークポイントを表示する
Stack		関数呼び出しスタックを表示する
Log		イベントとデータのログを表示する
Watches		監視している変数を表示する
Variable		グローバル変数とローカル変数を表示する
Module		プログラムのソース・モジュールを表示する
File		ディスク・ファイルをASCII形式または16進数形式で表示する
CPU		CPUの命令，データ，スタックを表示する
Dump		未処理のデータ・ダンプを表示する
Registers		CPUのレジスタとフラグを表示する
Numeric processor		V55SC, V55PIには浮動小数点演算用コプロセッサが用意されていないため, TD423, TD433ではこのウインドウを開くことができません。
Execution history		バック・トレースまたはキー・ストロークのプレイバックのために，セーブされたアセンブル・コードを表示する
Hierarchy		オブジェクトまたはクラスの型リストと階層ツリーを表示する
Target		ターゲット・システムの周辺リソースを表示する
ICE		インサーキット・エミュレータ・ハードウエアを表示する
Another		
	Module	もう1つのModuleウインドウを開く
	Dump	もう1つのDumpウインドウを開く
	File	もう1つのFileウインドウを開く

(4) Runメニュー

Run	プログラムを停止せずに実行する
Go to cursor	カレントのカーソル位置に移動する
Trace into	1ソース行または1命令を実行する
Step over	トレースして呼び出しをスキップする
Execute to	指定されたアドレスまで実行する
Until return	関数が戻るまで実行する
Animate	プログラムを連続的にステップする
Back trace	プログラムの実行を1ソース行，または1命令だけ反転させる
Instruction trace	1命令を実行する
Program reset	カレント・プログラムを再起動するプログラムは再ロードしない
ICE reset	インサーキット・エミュレータをリセットし，プログラムを再ロードする
Run and Exit	プログラムを実行モードに設定しMS-DOSへ抜ける

(5) Breakpointsメニュー

Toggle	カーソル位置でブレークポイントをトグルする
At	指定されたアドレスにブレークポイントを設定する
Changed memory global	メモリ領域にグローバル・ブレークポイントを設定する
Expression true global	式にグローバル・ブレークポイントを設定する
Hardware breakpoint	ハードウエア・ブレークポイントを設定する
Delete all	すべてのブレークポイントを削除する

(6) Dataメニュー

Inspect	データ項目を調べる
Evaluate/modify	式を評価する
Add watch	Watchesウインドウに変数を追加する
Function return	カレント・ルーチンの戻り値を調べる

(7) Optionsメニュー

Language		ソース・モジュールから式の言語を設定する
Macros		
	Create	キー・ストローク・マクロを定義する
	Stop recording	記録セッションを終了する
	Remove	1キー・ストローク・マクロを削除する
	Delete all	すべてのキー・ストローク・マクロを削除する
Display options		画面の表示オプション（画面の切り替え，サイズ，タブ）を設定できる
Path for source		ソース・ファイルのディレクトリ・リストを表示する
Save options		オプション，画面のレイアウト，およびマクロをディスクにセーブする
Restore options		ディスクからオプションを復元する
Hardware options		ホスト・マシンの環境を設定する

(8) Windowメニュー

Zoom	ウインドウを画面いっぱいのサイズまでズームし，元に戻す
Next	連続して画面に開いているウインドウをアクティブにする
Next pane	ウインドウ内の次のペインに移動する
Size/move	ウインドウを移動したり，そのサイズを変更する
Iconize/restore	ウインドウを小さいシンボルに縮小し，それをリストアする
Close	ウインドウをクローズする
Undo close	最後にクローズしたウインドウを再びオープンする
Dump pane to log	カレント・ペインをLogウインドウに書き込む

(9) Helpメニュー

Index	オンライン・ヘルプ用のインデクスに移動する
Previous topic	最後のヘルプ画面を表示する
Help on help	ヘルプ・システムに関するオンライン・ヘルプ画面にアクセスする

14.3　ローカル・メニュー・コマンド

GRPH-F10を押すと，カレント・ウインドウのローカル・メニューが起動します。CTRLキー・ショート・カットが使用可能であれば，CTRLキーと目的の項目の先頭文字を押して，個々のメニュー項目へ直接移動できます（CTRLキー・ショート・カットは，カスタマイズ・プログラムTDINSTBXを使って使用可能または使用不可に設定できます）。

14.4　各ウインドウのローカル・メニュー

次に，各ウインドウとペインのローカル・メニューについて説明します。

一部のペインには，そのローカル・メニュー上で共通して使えるショート・カットがあります。これらの特殊キーを示し，それを適用するペインのメニュー・コマンドを説明します。

ほとんどのペインでは，リターン・キーが，現在ハイライト表示されている項目を調べたり，変更するためのショート・カットになっています。DELキーは，ほとんどの場合，ハイライト表示されている項目を削除するローカル・メニュー・コマンドを起動します。

また，ローカル・メニューを表示させなくても，文字または数字を入力するだけでコマンドが起動するペインもあります。このような場合，入力を受け入れるために，ローカル・メニュー項目のうちの１つに対するダイアログ・ボックスが現れます。

14.4.1　Breakpointsウインドウ

Breakpointsウインドウには２つのペインがあります。ウインドウの左側にはリスト・ペインが，右側には詳細ペインが表示されます。ローカル・メニューがあるのはリスト・ペインのみです。

Set options	ブレークポイントのアクション，条件，パス・カウント，および使用可能／使用不可を設定する
Hardware options	このオプションはTD423，TD433では使用しません
Add	新しいブレークポイントを追加する
Remove	ハイライト表示されているブレークポイントを削除する
Delete all	すべてのブレークポイントを削除する
Inspect	ブレークポイントが設定されているコードを調べる
Toggle enable	ブレークポイントの設定／解除を行う

DELキーは，このウインドウではRemoveのショート・カットです。

14.4.2　CPUウインドウ

CPUウインドウには，それぞれローカル・メニューを持つ次の５つのペインがあります。

- コード・ペイン
- データ・ペイン
- スタック・ペイン
- レジスタ・ペイン
- フラグ・ペイン

(1) コード・ペイン

Goto		新しいアドレスにあるコードを表示する
Origin		PS：PCにあるコードを表示する
Follow		JMPターゲットまたはCALLターゲットにあるコードを表示する
Caller		呼び出し関数のコードを表示する
Previous		最後のアドレスにあるコードを表示する
Search		命令またはバイトを検索する
View source		Moduleウインドウに切り替える
Mixed		ソース・コードを逆アセンブラと組み合わせる（No/Yes/Both）
Log		指定アドレスから，指定バイト数分の逆アセンブル表示をLogする
New PS：PC		PS：PCを新しいアドレスで実行するように設定する
Assemble		カーソルI/Oにある命令をアセンブルする
I/O		
	In byte	I/Oポートから１バイト読み込む
	Out byte	I/Oポートに１バイト書き込む
	Read word	I/Oポートから１ワード読み込む
	Write word	I/Oポートに１ワード書き込む
Memory Mapped I/O		
	In byte	メモリ・マップトI/Oポートから１バイト読み込む
	Out byte	メモリ・マップトI/Oポートに１バイト書き込む
	Read word	メモリ・マップトI/Oポートから１ワード読み込む
	Write word	メモリ・マップトI/Oポートに１ワード書き込む

このペインでは，どの文字を入力しても，Assembleローカル・メニュー・コマンドのショート・カットになります。

(2) データ・ペイン

Goto		新しいアドレスにあるデータを表示する
Search		文字列またはデータ・バイトを検索する
Next		次に発生するまで検索する
Change		カーソル・アドレスにあるデータ・バイトを変更する
Follow		
	Near code	コード・ペインをカーソル位置にあるnearアドレスに設定する
	Far code	コード・ペインをカーソル位置にあるfarアドレスに設定する
	Offset to data	データ・ペインをカーソル位置にあるnearアドレスに設定する
	Segment : offset to data	データ・ペインをカーソル位置にあるfarアドレスに設定する
	Base segment : 0 to data	データ・ペインを，カーソル位置にあるアドレスを含むセグメントの先頭に設定する
Previous		最後のアドレスにあるデータを表示する
Display As		
	Byte	16進数バイトを表示する
	Word	16進数ワードを表示する
	Long	16進数の32ビットlongワードを表示する
	Comp	８バイトのPascalのComp整数を表示する
	Float	ショート（４バイト）の浮動小数点数（PascalのSingle，Cのfloat）を表示する
	Real	６バイトの浮動小数点数（PascalのReal）を表示する
	Double	８バイトの浮動小数点数（PascalとCのdouble）を表示する
	Extended	10バイトの浮動小数点数（Cのlong double，PascalのExtended）を表示する
	S-JIS	バイト単位で16進数表示と対応するASCII/シフトJIS漢字コードを右側に表示する
Block		
	Clear	メモリ・ブロックをゼロに設定する
	Move	メモリ・ブロックを移動する
	Set	メモリ・ブロックに値を設定する
	Read	ファイルからメモリに読み込む
	Write	メモリからファイルに書き込む
	Log	メモリ・データをLogファイルに書き込む

このペインでは，どの文字を入力しても，Changeローカル・メニュー・コマンドのショート・カットになります。

(3) フラグ・ペイン

Toggle	ハイライト表示されているフラグを設定またはクリアする

このペインでは，↵またはスペースを押すと，Toggleローカル・メニュー・コマンドのショート・カットになります。

(4) レジスタ・ペイン

Increment	ハイライト表示されているレジスタに1を追加する
Decrement	ハイライト表示されているレジスタから1を引く
Zero	ハイライト表示されているレジスタをクリアする
Change	ハイライト表示されているレジスタを新しい値に設定する

このペインでは，どの文字を入力しても，Changeローカル・メニュー・コマンドのショート・カットになります。

(5) スタック・ペイン

Goto	新しいアドレスにあるスタックを表示する
Origin	SS：SPにあるデータを表示する
Follow	カレント項目が指し示すコードを表示する
Previous	最後のアドレスまで表示をリストアする
Change	情報を編集することができる

このペインでは，どの文字を入力しても，Changeローカル・メニュー・コマンドのショート・カットになります。

14.4.3 Dumpウインドウ

Dumpウインドウは，CPUウインドウのデータ・ペインと同じです。また，このウインドウのローカル・メニューは，データ・ペインのローカル・メニューと同じです。

14.4.4 Fileウインドウ

Fileウインドウには，ディスク・ファイルの内容が16進バイトまたはディスク・ファイルとして表示されます。

Goto	行番号または16進数オフセットを表示する
Search	文字列またはデータ・バイトを検索する
Next	次に発生するまで検索する
Display as	ファイルの表示モードASCII/16進を設定する
File	新しいファイルが表示されるように切り替える
Edit	ファイルを編集するか，カーソル位置にあるバイトを変更する

どの文字を入力しても，Searchローカル・メニュー・コマンドのショート・カットになります。

14.4.5　Logウインドウ

Logウインドウには，ログに書き込まれるメッセージが表示されます。

Open log file	ファイルへのログを開始する
Close log file	ファイルへのログを停止する
Logging	ログのYes/Noをトグルする
Add comment	ログにユーザ・コマンドを書き込む
Erase log	すべてのログ・メッセージをクリアする

どの文字を入力しても，Add commentローカル・メニュー・コマンドのショート・カットになります。

14.4.6　Moduleウインドウ

Moduleウインドウには，プログラム・モジュールのソース・ファイルが表示されます。

Inspect	カーソル位置にある変数の内容を表示する
Watch	カーソル位置にある変数をウォッチ・リストに追加する
Module	別のモジュールを表示するように変更する
File	別のファイルを表示するように変更する
Previous	最後のモジュールと位置を表示する
Line	モジュール内の行にあるソースを表示する
Search	テキスト文字列を検索する
Next	文字列の次の発生を検索する
Origin	プログラムの現在位置を表示する
Goto	アドレスにあるソースまたは命令を表示する
Edit	エディタを起動してソース・ファイルを編集する

どの文字を入力しても，Gotoローカル・メニュー・コマンドのショート・カットになります。

14.4.7　Numeric processorウインドウ

V55SC, V55PIには浮動小数点演算用コプロセッサが用意されていないため, TD423, TD433ではこのウインドウを開くことができません。

14.4.8　Hierarchyウインドウ

Hierarchyウインドウには，オブジェクト型／クラス・リスト・ペインと階層ツリー・ペインの２つのペインがあります。また，多重継承が使われているC＋＋プログラムを実行している場合は，第３のペインである親ツリー・ペインがあります。

(1) オブジェクト型／クラス・リスト・ペイン

Inspect	ハイライト表示されているオブジェクトまたはクラスの型の内容を表示する
Tree	階層ツリー・ペインに移動する

(2) 階層ツリー・ペイン

Inspect	ハイライト表示されているオブジェクトまたはクラスの型の内容を表示する
Parents	多重継承が使われているC＋＋プログラムを実行している場合は，親ツリー・ペインが表示されているかどうかをトグルする。

(3) 親ツリー・ペイン

Inspect	ハイライト表示されているオブジェクトまたはクラスの型の内容を表示する

14.4.9　Registersウインドウ

Registersウインドウは，CPUウインドウのレジスタ・ペインおよびフラグ・ペインと同じです。また，このウインドウのローカル・メニューは，レジスタ・ペインのローカル・メニューおよびフラグ・ペインのローカル・メニューと同じです。

14.4.10　Stackウインドウ

Stackウインドウには，現在アクティブな関数が表示されます。

Inspect	ハイライト表示されている関数のソース・コードを表示する
Locals	ハイライト表示されている関数のローカル変数を表示する

↵を押すと，Inspectローカル・メニュー・コマンドのショート・カットになります。

14.4.11　Variablesウインドウ

Variablesウインドウには，それぞれローカル・メニューを持つ次の２つのペインがあります。

(1) グローバル・ペイン

Inspect	ハイライト表示されているシンボルの内容を表示する
Change	ハイライト表示されているシンボルの値を変更する

このペインでは，↵を押すとInspectローカル・メニュー・コマンドのショート・カットになります。

(2) スタティック・ペイン

Inspect	ハイライト表示されているシンボルの内容を表示する
Change	ハイライト表示されているシンボルの値を変更する

このペインでは，↵を押すとInspectローカル・メニュー・コマンドのショート・カットになります。

14.4.12　Watchesウインドウ

Watchesウインドウには，監視している変数の名前と値が表示されるペインが１つあります。

Watch	監視する変数または式を追加する
Edit	監視する変数または式を編集可能にする
Remove	ハイライト表示されている変数または式を削除する
Delete all	すべてのwatch変数または式を削除する
Inspect	ハイライト表示されている変数または式の内容を表示する
Change	ハイライト表示されている変数の内容を変更する 式には影響を与えない

このウインドウでは，次のキーがローカル・メニュー・コマンドのショート・カットです。

- すべての文字　Watch
- ↵　Edit
- DEL　Remove

14.4.13　Inspectウインドウ

Inspectウインドウには，データ項目の内容が表示されます。

Range	調べたい配列メンバを選択する
Change	ハイライト表示されている項目の値を変更する
Inspect	ハイライト表示されている項目について新しいInspectウインドウをオープンする
Descend	ハイライト表示されている項目をこのInspectウインドウまで拡大する
New expression	このInspectウインドウ内で新しい式を調べる
Type cast	ハイライト表示されている項目を新しい型に型キャストする

14.4.14　オブジェクト型／クラスInspectウインドウ

オブジェクト型／クラスInspectウインドウには，オブジェクトまたはクラスの内容（データ・フィールドまたはメンバと，メソッドまたはメンバ関数）を表示する２つのペインがあります。このウインドウのローカル・メニューは（この２つのペインに共通）通常のInspectウインドウのローカル・メニューとは多少異なっています。

Inspect	ハイライト表示されている型の内容を表示する
Hierarchy	Hierarchyウインドウに戻る
Show inherited	オブジェクトまたはクラスのすべての内容と，カレントのオブジェクトまたはクラス内で宣言された内容の間で，表示を切り替える

14.4.15　オブジェクト／クラス・インスタンスInspectウインドウ

オブジェクト／クラス・インスタンスInspectウインドウには３つのペインがあり，最初の２つのペインにローカル・メニューがあります（第３のペインは，インスタンスが属しているオブジェクト型またはクラスが表示されるだけです）。この２つのローカル・メニューは同じで，次のコマンドがあります。

Range	調べたい配列メンバを選択する
Change	ハイライト表示されている項目の値を変更する
Methods	中央のペインでメソッドとメンバ関数のどちらが要約されるかをトグルする
Show inherited	オブジェクトまたはクラスのすべての内容と，カレントのオブジェクトまたはクラス内で宣言された内容の間で表示をトグルする
Inspect	ハイライト表示された項目について新しいInspectウインドウを開く
Descend	ハイライト表示されている項目をこのInspectウインドウ内で展開する
New expression	このInspectウインドウ内で新しい式を調べる
Type cast	ハイライト表示されているデータ項目を新しい型に型キャストする
Hierarchy	Hierarchyウインドウに戻る

(1) テキスト・ペイン

これは，テキスト・ファイルの内容を表示するペインに共通の名前です。カーソルが点滅している場合は，ファイル内の現在位置を示します。次に，コマンドのリストを示します。

キー	機　　能
INS	テキスト・ブロックをマークする
↑	1行上に移動する
↓	1行下に移動する
→	1桁右に移動する
←	1桁左に移動する
CTRL-→	次のワードに移動する
CTRL-←	前のワードに移動する
HOMECLR	行の先頭に移動する
SHIFT-HOMECLR	行の最後の文字に移動する
ROLLDOWN	1画面を上にスクロールする
ROLLUP	1画面を下にスクロールする
CTRL-HOMECLR	ペインの最上行に移動する
CTRL-SHIFT-HOMECLR	ペインの最下行に移動する
CTRL-ROLLDOWN	ファイルの1行目に移動する
CTRL-ROLLUP	ファイルの最終行に移動する

(2) リスト・ペイン

これは，スクロールできる情報がリストされるペインの本来の名前です。ハイライト・バーは，リスト内の現在位置を示します。次に，使用可能なすべてのコマンドを示します。

キー	機　　能
↑	1項目上に移動する
↓	1項目下に移動する
→	右にスクロールする
←	左にスクロールする
HOMECLR	行の先頭に移動する
SHIFT-HOMECLR	行の最後の文字に移動する
ROLLDOWN	1画面上にスクロールする
ROLLUP	1画面下にスクロールする
CTRL-HOMECLR	リスト・ペインの先頭行に移動する
CTRL-SHIFT-HOMECLR	リスト・ペインの最終行に移動する
CTRL-ROLLDOWN	リスト内の最初の項目に移動する
CTRL-ROLLUP	リスト内の最後の項目に移動する
BS	インクリメンタル検索で1文字戻る
文字	インクリメンタル検索を行う（入力して選択する）

14.5　入力ボックスとヒストリ・リスト・ボックス内のコマンド

次に，入力ボックスまたはヒストリ・リスト・ボックス内で使用可能なコマンドを示します。

キー	機　　能
↑	1リスト項目上に移動する
↓	1リスト項目下に移動する
→	1文字右に移動する
←	1文字左に移動する
CTRL-→	次のワードに移動する
CTRL-←	前のワードに移動する
HOME	行の先頭に移動する
SHIFT-HOMECLR	行の最終文字に移動する
ROLLDOWN	1画面上にスクロールする
ROLLUP	1画面下にスクロールする
CTRL-HOMECLR	リスト・ペインの最上行に移動する
CTRL-SHIFT-HOMECLR	リスト・ペインの最終行に移動する
CTRL-ROLLDOWN	リスト内の最初の項目に移動する
CTRL-ROLLUP	リスト内の最後の項目に移動する
BS	カーソルの前の文字を削除する
⏎	入力を受け入れて進行する
DEL	カーソル位置にある文字を削除する
ESC	ダイアログ・ボックスをキャンセルしてメニューに戻る
CTRL-N	入力ボックス内に部分的に入力された名前を完成する

14.6　ウインドウ移動コマンド

次に，ウインドウの再配置と拡大／縮小のコマンドを示します。

キー	機　　能
CTRL-F5	ウインドウ位置を決めモードをトグルする
↑	ウインドウを１行上に移動する
↓	ウインドウを１行下に移動する
→	ウインドウを１桁右に移動する
←	ウインドウを１桁左に移動する
SHIFT-↑	ウインドウのサイズを変更して，下辺を上に移動する
SHIFT-↓	ウインドウのサイズを変更して，下辺を下に移動する
SHIFT-→	ウインドウのサイズを変更して，右辺を右に移動する
SHIFT-←	ウインドウのサイズを変更して，右辺を左に移動する
HOMECLR	画面の左端に移動する
SHIFT-HOMECLR	画面の右端に移動する
ROLLDOWN	画面の最上行に移動する
ROLLUP	画面の最下行に移動する
↵	現在の位置を受け入れる
ESC	ウインドウの位置を決めるコマンドをキャンセルする

14.7　ワイルド・カード検索テンプレート

次の２つの状況では，ワイルド・カード検索テンプレートを使うことができます。

- ファイル名を入力してロードまたは調べるとき
- テキスト・ペイン内にテキスト検索式を入力するとき

？（疑問符）は，検索文字列の中の任意の１文字と一致します。＊（アスタリスク）は，検索文字列の中の０個以上の任意の文字に一致します。

〔メ　モ〕

付録A　コマンド・ライン・オプション

MS-DOSのコマンド・ラインからTDシリーズを起動するとき，同時にコマンド・ライン・オプションを使って動作を設定することができます。コマンド・ライン・オプションを指定する書式は次のとおりです。

TD423の場合

td423bx [options] △ [load_module]

TD433の場合

td433bx [options] △ [load_module]

[] で囲まれている項は省略することができます。オプションのあとにマイナス（－）を付けると，コンフィギュレーション・ファイルの中でそのオプションがオンに設定されていても，明示的にそれをオフにすることができます。なお，△はスペース・キーの入力を示します。

オプション	機　　能
－N	レジスタ名，インストラクション表示がVシリーズ表記になります。このオプションを付けない場合はインテル表記の表示です。
－cfilename	コンフィギュレーション・ファイルを指定する。
－g，－g＋ －g－ －gN	ディバッガの46行画面を使用可能にする。 46行画面を使用不可にする。 46行画面に使うG-VRAMのバンク（０または１）を指定する。
－h，－？	すべてのコマンド・ライン・オプションに関するヘルプ画面を表示する。
－l	アセンブラ・モードで起動する。
－mN	ヒープ・サイズをN（Kバイト単位）に設定する。
－p	マウスを使用可能にする。
－rc	ディバグ・メモリ・キャッシングを禁止する。
－sc	文字の大小を無視する。
－sddir	ソース・ファイルを探すディレクトリを指定する。
－smN	シンボル・テーブルのメモリ・サイズをN（Kバイト単位）に設定する。
－yN	オーバレイ領域のサイズをN（Kバイト単位）に設定する。
－yeN	EMSオーバレイ領域のサイズをN（16 Kバイト単位）に設定する。

〔メ　モ〕

付録B　TDシリーズのカスタマイズ

TDシリーズは，マスタ・ディスクのファイルの作業用コピーを作成すればそのまま実行できるようになっています。ただし，TDINSTBXというカスタマイズ・プログラムを実行すると，いろいろなディフォールト設定を変更できます。また，MS-DOSからTDシリーズを起動するときに，コマンド・ライン・オプションを使ってオプションの一部を変更できます。同じコマンド・ライン・オプションを頻繁に指定する場合は，カスタマイズ・プログラムを実行するとそれらのオプションを永続的なものにできます。

カスタマイズ・プログラムにより，次の項目を設定できます。

- ウインドウ，ダイアログ・ボックス，およびメニューの色
- 表示パラメータ

  整数の表示形式，起動時の表示（ソースまたはアセンブラ），画面の行数，タブの桁数，Watchesウインドウの最大行数，46モードの使用，ログ・リストの長さ，起動時のグラフィクス・モード，浮動小数点数の精度，調査の範囲
- ソース・ファイル，ヘルプ・ファイル，およびコンフィギュレーション・ファイルを探すディレクトリ，エディタの起動コマンド
- ユーザ入力およびプロンプト・パラメータ

  実行中断キー，ヒストリ・リストの長さ，エラー発生時のビープ音，マウス，キー・ストロークの記録，CTRLキー・ショート・カット
- ソース・ディバグ

  言語オプションと大文字，小文字の区別

備考　TDINSTBXがサポートしているのは，上記の項目のみです。TDINSTBXのカスタマイズ・オプションの中にはTDシリーズで使用しないものがありますので無視してかまいません。

B.1　TDINSTBXの起動

TDINSTBXを起動するコマンド・ラインの構文は次のとおりです。

TD423の場合　tdinstbx△td423bx（①）

TD433の場合　tdinstbx△td433bx（①）

TDINSTBXの初期化が終了するとメイン・メニューが表示されます。メニュー・オプションのハイライト表示されている先頭文字を押すか，↑↓キーを使って目的の項目に移動し，↵を押してください。たとえば，表示に関する設定を変更するには，D(Display)を押してください。TDINSTBXのほかのメニューから選択する場合も，これと同じ方法を使います。前のメニューに戻るにはESCキーを押します。メイン・メニューに戻るには，ESCキーを数回押さなければならないこともあります。

TDINSTBXを終了するには，メイン・メニューからQuit（GRPH-X）を選択します。

注意　TDINSTBXは，Borland International Inc.のTDINSTカスタマイゼーション・ユーティリティと同じものです。①の部分は，必ず指定してください。指定しないと，ターボ・ディバッガの実行ファイルをカスタマイズしてしまいます。

B.2　画面の色の設定（Colors）

メイン・メニューからColorsを選択すると，Colorsメニューが表示されます。Colorsメニューでは，CustomizeとDefault color setのどちらかを選択します。

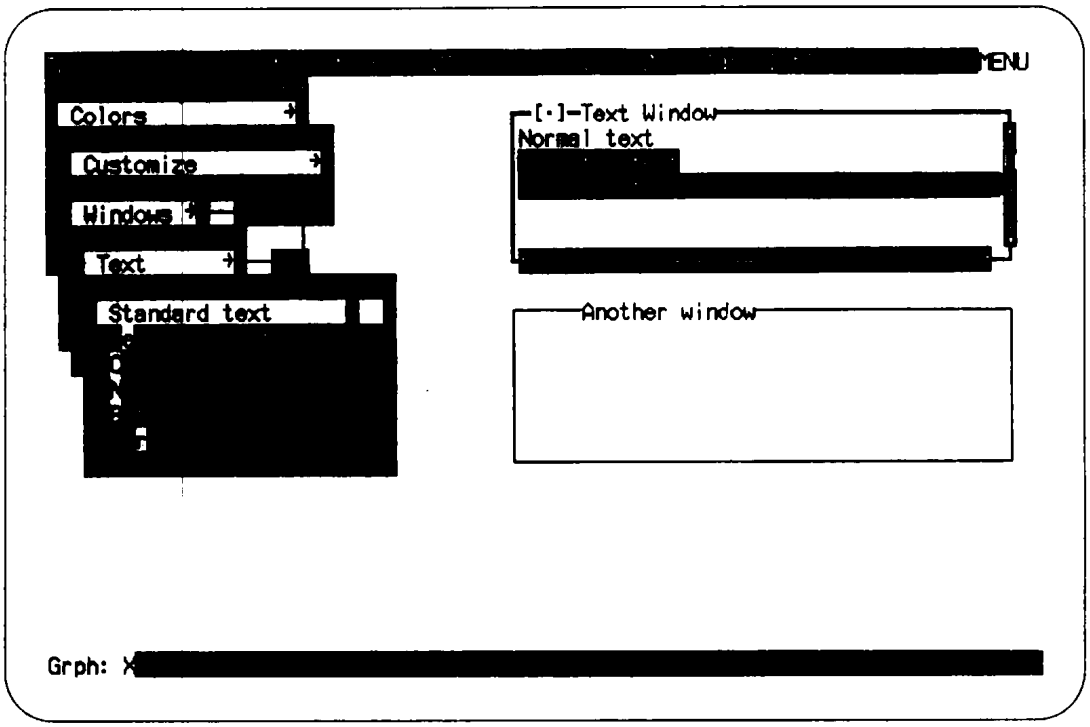
B.2.1　画面の色のカスタマイズ（Customize）

Customizeを選択すると，第３のメニューが表示されます。このメニューには，ウインドウ，ダイアログ・ボックス，メニュー，画面のほかの部分，およびアナログCRT用パレットのカスタマイズといった５つのオプションがあります。

（1）Windowsコマンド

ウインドウをカスタマイズするにはWindowsコマンドを選択します。このコマンドを選択すると第４のメニューが表示され，カスタマイズしたいウインドウの種類を，Text，Data，Low level(CPUウインドウなど)，およびOther (Breakpointsウインドウなど) の中から選択できます。オプションの１つを選択すると，ウインドウの要素が示されたもう１つのメニューが表示され，いろいろな色の組み合わせを確認するための２つのサンプル・ウインドウ（一方はアクティブな場合，もう一方はアクティブでない場合）が表示されます。

図B－1　画面の色のカスタマイズの際のサンプル・ウインドウ



変更したい項目を選択すると，メニューの位置にパレット・ボックスが現われます。パレット・ボックス内では，カーソル・キーを使って移動してください。パレットの中で選択ボックスを移動するにつれて，確認用ウインドウの構成要素の色も更新され，現在選択されている色の組み合わせが表示されます。目的の色が見つかったら，↵を押して確定してください。

TDシリーズは，次の３つのカラー・テーブルを持っています。

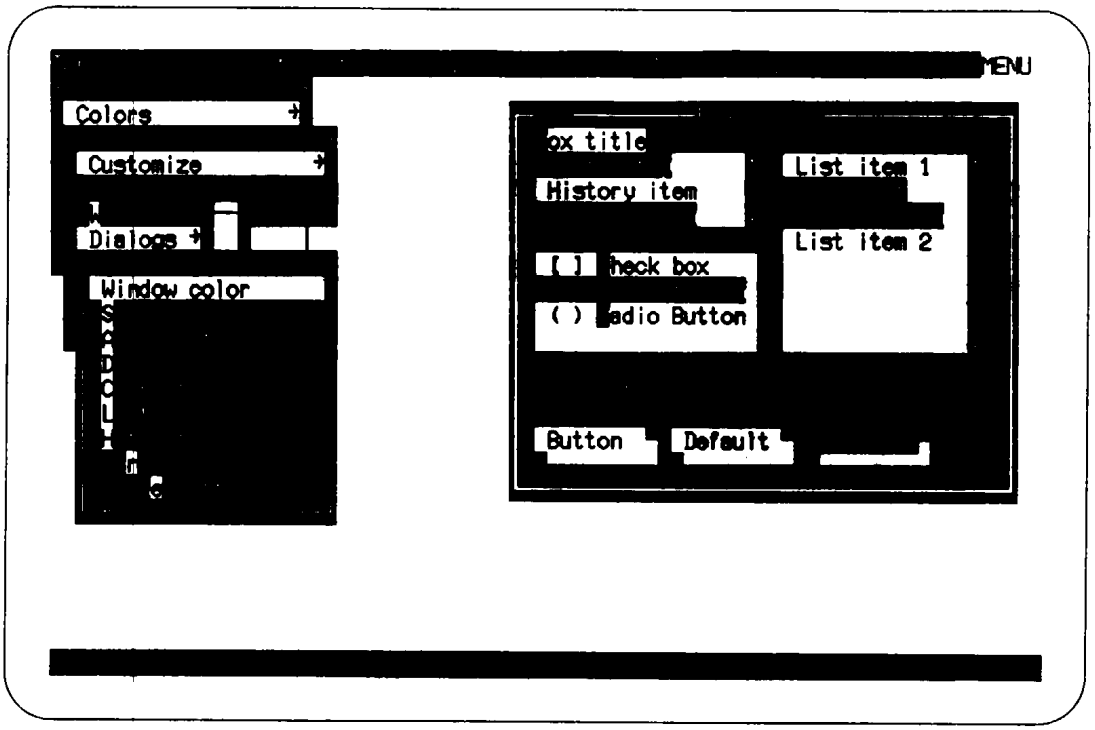
- テキスト（25行）表示用
- グラフィック（46行）表示用…アナログCRT用
- グラフィック（46行）表示用…デジタルCRT用

現在設定されている画面表示モードとディスプレイ・ハードウエアに基づき一度に変更できる色は１セットです。たとえば，ディジタルCRTを使っていてグラフィック（46行）表示の色をカスタマイズしたい場合には，Mode for displayオプションをColor Digitalに設定します。Displayオプションの Screen linesラジオ・ボタンを46に設定してから，Colorsメニューを選択してください。

(2)　Dialogsコマンド

Customizeメニューから Dialogsを選択すると，ダイアログ・ボックスの構成要素を示したメニューと，確認用のサンプル・ダイアログ・ボックスが表示されます。

図B-2　画面の色のカスタマイズの際のサンプル・ダイアログ・ボックス



Windowsメニューの場合と同じように，メニューから項目を選択すると，その項目の色を選択するためのパレット・ボックスが表示されます。

(3)　Menusコマンド

CustomizeメニューからMenusを選択すると，メニューの構成要素を示したメニューと確認用のサンプル・メニューが表示されます。メニューから項目を選択するとパレット・ボックスが表示されます。

(4)　Screenコマンド

Screenコマンドでは，上述したコマンド(Windows，Dialogs，Menus)に分類されない次のような画面要素について設定します。

- 画面全体の背景の色とパターン
- 移動／リサイズ中のウインドウ枠
- ウインドウやダイアログ・ボックスの影

これらの画面要素に対する画面パターンとパレット，それらをテストするサンプルの画面背景を使って選択します。

(5) Paletteコマンド

Paletteを選択すると，アナログCRTでグラフィック（46行）表示を行うときのカラー・パレットをカスタマイズすることができます。パレットのカスタマイズは，Display optionsダイアログ・ボックスのScreen linesラジオ・ボタンを46に設定し，Mode for displayオプションをColor Analogに設定している場合にのみ行えます。

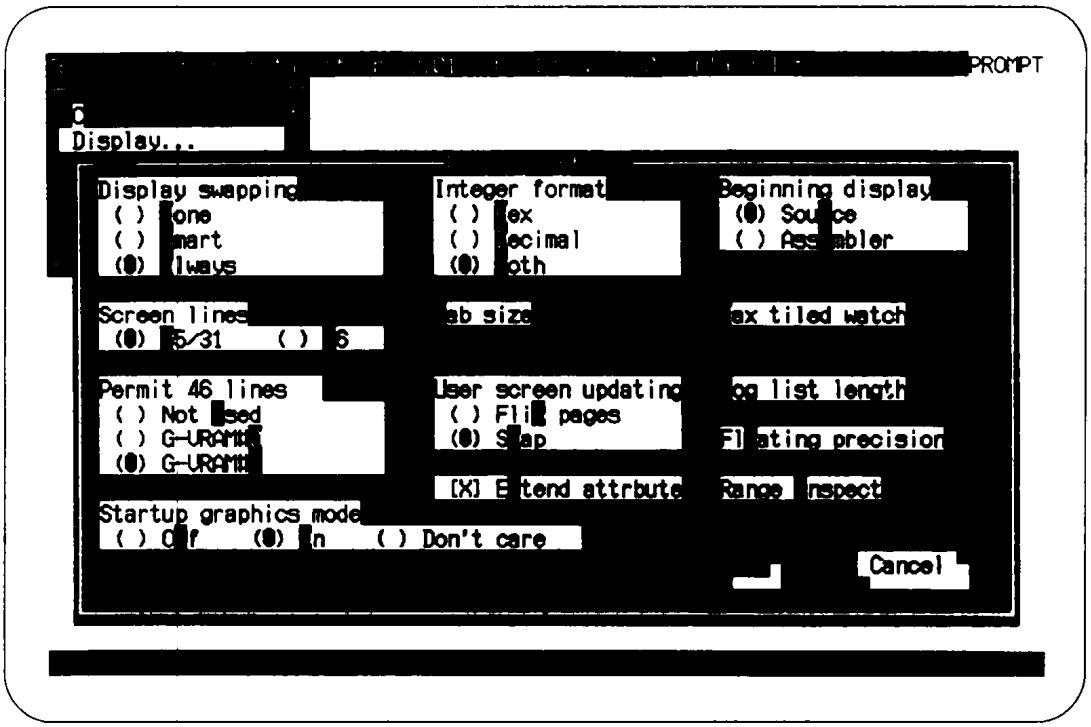
B.2.2 ディフォールトのカラー・セット（Default color set）

ColorsメニューからDefault color setを選択すると，テキスト・ウインドウがアクティブな状態とアクティブでない状態で表示され，構成要素のディフォールトの色を調べることができます。Customizeメニューであれこれ設定しているうちに色の構成が混乱してしまった場合など，すべての色をディフォールトに戻したいときにこのオプションを使ってください。

B.3 ディスプレイ・パラメータの設定

メイン・メニューからDisplayを選択すると，Display optionsダイアログ・ボックスが表示されます。

図B-3 Display optionsダイアログ・ボックス



ディスプレイ・パラメータには，TDシリーズの起動時にコマンド・ライン・オプションで設定できるものと，TDINSTBXを使わなければ設定できないものがあります。Display swapping, Full Graphics Saveなどのその他のオプションはTDシリーズでは使用しませんので無視してください。

Displayオプションでは以下のディフォールトを選択可能です。

(1) Display swappingラジオ・ボタン

TDシリーズには，ユーザ・スクリーンのサポートはありません。Display swappingはNoneに設定してください。

(2) Integer formatラジオ・ボタン

Integer formatラジオ・ボタンを使うと，整数の表示形式を設定できます。次のオプションの間で設定が切り替わります。

- Hex　　16進数表示を選択します。
- Decimal　　10進数表示を選択します。
- Both　　16進数と10進数の両方を表示します。

(3) Beginning displayラジオ・ボタン

Beginning displayラジオ・ボタンは，TDシリーズが起動されるときにプログラムが表示される言語を決定します。

- Assembler…アセンブラ起動

  プログラムは何も実行されません。CPUウインドウにプログラム内の最初の命令が表示された状態で起動します。
- Source…ソース起動

  コンパイラが生成したプログラムの開始（スタートアップ）コードが実行され，Moduleウインドウにソース・コードの始まりの部分を表示した状態で起動します。

(4) Screen linesラジオ・ボタン

このラジオ・ボタンを使うと，TDシリーズ起動時の画面を25行または46行のどちらで表示するかを指定できます。

(5) Tab size入力ボックス

この入力ボックスでは，テキスト・ファイルまたはソース・ファイルの表示でタブ・ストップの桁数を設定できます。タブの桁数（1-32）を入力するプロンプトが表示されます。

ディフォールトは３桁です。

(6) Maximum tiled watch入力ボックス

この入力ボックスは，ウインドウが並置された状態で，監視する式の追加に伴ってWatchesウインドウが自動的に最大何行まで拡大されるかを設定します。最大行数（1-20）を入力するプロンプトが表示されます。

(7) Permit 46 linesラジオ・ボタン

このラジオ・ボタンでは，グラフィック画面を用いた46行表示を使用可能にするかどうか，また使用する場合にグラフィックVRAM（G-VRAM）のどちらのバンクを使うかを選択します。

(a) Not used

46行表示を使用しません。46行表示には，より大きなウインドウ・バッファが必要なため，これを選択すると約８Kバイトのメモリを節約できます。これは，実行時にできるだけ大きなメモリが必要となる非常に大きなプログラムをディバグするときなどに便利です。

(b) G-VRAM＃0

G-VRAMのバンク０を46行表示用に使用します。46行表示を使いたいときに，次のような制約または要求がある場合にはこのオプションを選択してください。

- PC-9801やPC-9801Uなど，G-VRAMが１バンクしかないマシンを使用している場合。
- G-VRAMのバンク１にシンボル・テーブルを置きたい場合。
- G-VRAMのバンク１をRAMディスクとして使いたい場合。

(c) G-VRAM＃1

G-VRAMのバンク１を46行表示用に使用します。これがディフォールトです。

(8) User screen updatingラジオ・ボタン

TDシリーズはリモート・モードだけで動作するため，ユーザ・スクリーンのサポートはありません。User screen updatingはFlip PagesかSwapに設定してください。

(9) Log list length入力ボックス

この入力ボックスでは，以前のエントリをログ・ファイル内にいくつセーブするかを設定します。最大数は200，最小数は４です。

(10) Startup graphics modeラジオ・ボタン

このラジオ・ボタンでは，TDシリーズの起動時にグラフィック画面を表示するかどうかを指定します。OffとOnは，グラフィック画面を強制的に非表示または表示します。Don't Careを選択すると，起動前の状態が維持されます。

(11) Floating precision入力ボックス

浮動小数点数を表示するときの精度（桁数）を1-32の範囲で指定します。ディフォールトは6桁です。

(12) Range inspect入力ボックス

InspectウインドウのRangeローカル・コマンドで指定する範囲のディフォールト値を1-4,096の範囲で指定します。ディフォールトは5です。

B.4　Optionsメニュー

メイン・メニューでOptionsコマンドを選択すると，さまざまなオプションを分類したもう1つのメニューが表示されます。このメニューからオプションのどれかを選択すると，次のようなダイアログ・ボックスが表示されます。

- Directories...
- Input & prompting...
- Source debugging...
- Miscellaneous...

(1) Directories...ダイアログ・ボックス

このダイアログ・ボックスには，次の3つの入力ボックスがあります。

(a) Editor program name入力ボックス

エディタを起動するMS-DOSコマンドを指定します。このオプションでエディタを指定しておくと，ディバグ中にファイルのどこかを変更したいときに，TDシリーズの中から使い慣れたエディタを起動することができます。TDシリーズは，ここで指定されたコマンドの後ろにスペースを入れて，編集するファイル名を追加します。

(b) Source directories入力ボックス

TDシリーズがソース・ファイルを探すディレクトリのリストを設定します。

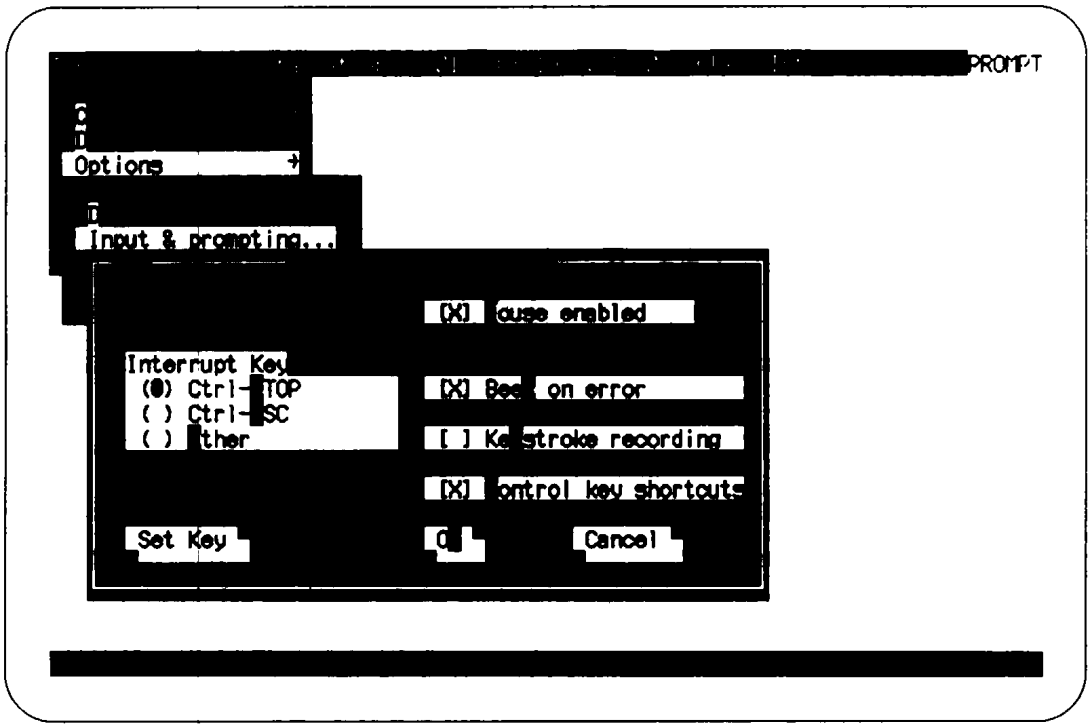
(c) Turbo directory入力ボックス

TDシリーズがヘルプ・ファイルとコンフィギュレーション・ファイルを探すディレクトリを設定します。

(2) Input & prompting...ダイアログ・ボックス

このダイアログ・ボックスでは，TDシリーズでの入力とプロンプトに関する設定を行います。

図B-4　Input&promptingダイアログ・ボックス



(a) History list length入力ボックス

この入力ボックスでは，入力ボックスのヒストリ・リストに以前のエントリをいくつセーブするかを指定します。

(b) Interrupt keyラジオ・ボタン

このラジオ・ボタンでは，CTRL-STOP，CTRL-ESC，およびOtherの中から，ディフォールトの実行中断キーを指定することができます。Otherを選択した場合には，その下にあるSet Keyボタンを使って，実行中断キーに割り当てるキー・ストロークを指定できます。

(c) Set Keyボタン

Interrupt keyでOtherを選択した場合は，Set Keyボタンを押して，実際の実行中断キーを選択してください。キーを入力するようプロンプトが表示されます。

(d) Mouse enabledチェック・ボックス

このチェック・ボックスは，TDシリーズがディフォールトの状態でマウスをサポートするかどうかを制御します。

(e) Beep on errorチェック・ボックス

ディフォールトの状態では，無効なキーを押したりエラー・メッセージを生成するような操

作を行うと，警告音が鳴ります。Beep on errorチェック・ボックスを使うと，このビープ音を出さないようにすることができます。

(f) Keystroke recordingチェック・ボックス

このチェック・ボックスは，Execution historyウインドウがディフォールトの状態でキー・ストロークを自動的に記録するかどうかを決定します。

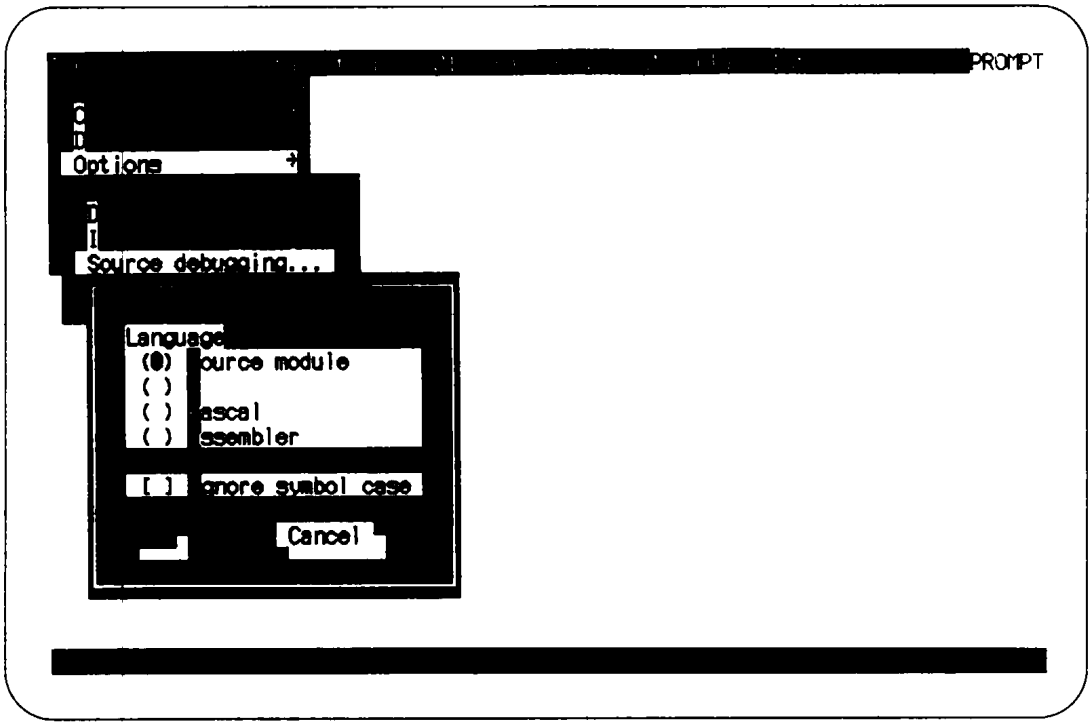
(g) Control key shortcutsチェック・ボックス

このチェック・ボックスは，CTRLキー・ショート・カットを使用可能または使用不可にします。CTRLキー・ショート・カットを使用可能にすると，CTRLキーとメニュー項目の先頭文字を一緒に押して，ローカル・メニュー・コマンドを直接起動することができます。

(3) Source debugging...ダイアログ・ボックス

Source debuggingダイアログ・ボックスでは，TDシリーズがどの言語を使って式を評価するか，およびシンボルの大文字，小文字を区別するかどうかを指定します。

図B-5　Source debuggingダイアログ・ボックス



(a) Languageラジオ・ボタン

Languageラジオ・ボタンは，TDシリーズが式の評価に使う言語をトグルします。

● **Source Module**

現在のソース・モジュールの言語に基づいて，どの言語を使うかをTDシリーズが判断します。

● **C**

現在のモジュールがどの言語で書かれているかに関係なく，常にCの式評価を使います。

● **Pascal**

現在のモジュールがどの言語で書かれているかに関係なく，常にPascalの式評価を使います。

● **Assembler**

現在のモジュールがどの言語で書かれているかに関係なく，常にアセンブラの式評価を使います。

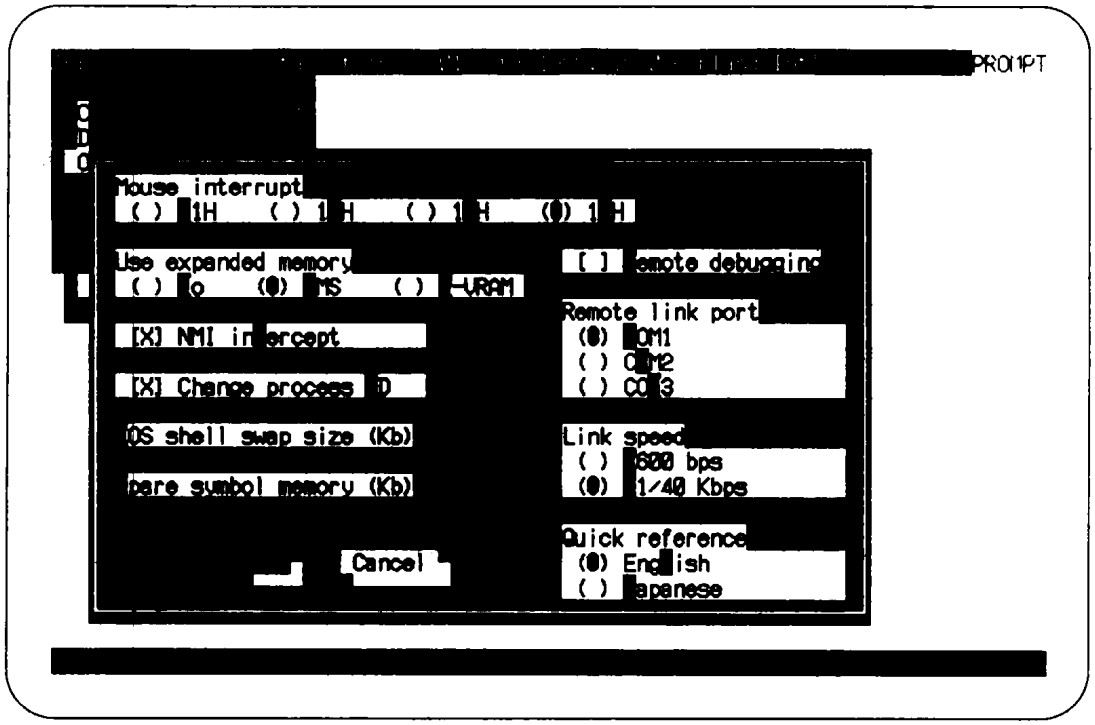
(b) Ignore symbol caseチェック・ボックス

このチェック・ボックスをオンにすると，TDシリーズは大文字と小文字を同じものとして扱います。オフにすると，大文字と小文字は区別されます。

(4) Miscellaneous...ダイアログ・ボックス

Miscellaneous optionsダイアログ・ボックスでは，マウス制御に使用する割り込み番号，拡張メモリの使用形式，NMIの割り込み，プロセスIDの切り替え，MS-DOSシェル起動時のスワップ・サイズ，シンボル・テーブルのサイズに関するオプションを設定することができます。

図B-6　Miscellaneous optionsダイアログ・ボックス



(a) NMI interruptチェック・ボックス　　未使用

(b) Change Process IDチェック・ボックス　　未使用

(c) DOS shell swap size入力ボックス　　未使用

(d) Spare symbol memory入力ボックス

この入力ボックスでは，マニュアルで（File | Symbol loadコマンドで）ロードされるシンボル・テーブル用に確保するメモリ容量をKバイト単位で指定します。コマンド行の－smオプションからも設定可能です。

(e) Remote debuggingチェック・ボックス

TDシリーズは，常にリモート・モードで動作するのでこのオプションは無視されます。

(f) Remote link portラジオ・ボタン　　未使用

(g) Link speedラジオ・ボタン　　未使用

(h) **Quick reference**

Helpの表示を日本語にするか英語にするかを指定します。ただし，Borland International Inc.のターボ・ディバッガに対して追加した機能（たとえば，View | Targetコマンドなど）は，ここで日本語表示を指定した場合でも，その設定が無効となり，英語表示となります。

B.5　Mode for displayオプション

メイン・メニューからMode for displayを選択すると，グラフィック（46行）表示のモードを選択するメニューが現われます。

(1) **Color Digital**

ディジタルCRTを使用している場合に選択します。また，アナログCRTでもディジタル・モード（８色）で表示したいときにはこれを選択してください。

(2) **Color Analog**

アナログCRTを使用している場合に選択します。ディジタルCRTを使っている場合にこのオプションを選択すると，46行モードの画面は正しく表示されないので注意してください。

B.6　設定のセーブとTDINSTBXの終了

B.6.1　設定内容のセーブ

TDシリーズのオプションの設定がすべて終了したら，メイン・メニューからSaveを選択し，セーブする方法を指定してください。

(1) **Save configuration file...コマンド**

Save configuration fileを選択するとダイアログ・ボックスがオープンされ，ディフォールトのコンフィギュレーション・ファイル名TDCONFIG.TDが表示されます。↵を押してこのファイルにセーブするか，新しいコンフィギュレーション・ファイル名を入力してください。

別のファイル名を指定した場合，TDシリーズを起動するときに，次のように－cコマンド・ライン・オプションを使って，そのコンフィギュレーションをロードします。

TD423の場合	td423bx　－cmyconfig myprog
TD433の場合	td433bx　－cmyconfig myprog

また，TDシリーズのOptions | Restore configurationコマンドにより，TDシリーズを起動したあとでコンフィギュレーションをロードすることもできます。

(2) Modify td423bx (td433bx). exeコマンド

Modify... を選択すると，TDINSTBXの中で行った変更が，TD423，TD433の実行可能ファイルTD423BX(TD433BX). EXEに直接書き込まれます。次にTD423，TD433を起動すると，その設定がディフォールトの状態となります。

備考　カスタマイズしても，ファイルの日付けや時間はカスタマイズする前と同じです。元のファイルと区別するために，カスタマイズする際は別のファイル名に変えておくことをお奨めします。出荷時のディフォールト設定に戻したいときは，マスタ・ディスクから作業用システム・ディスクにもう一度インストールしてください。

B.6.2　TDINSTBXの終了

メイン・メニューからQuitを選択するかまたはGRPH-Xを押すと，TDINSTBXが終了します。

付録C　ダイアログ・ボックスとエラー・メッセージ

TDシリーズは，エラー・メッセージまたはダイアログ・ボックスを現在のカーソル位置に表示します。ここでは，TDシリーズが表示するすべてのダイアログ・ボックスとエラー・メッセージについてどのように応答すればよいかを説明します。

ダイアログ・ボックスとエラー・メッセージ（起動時の致命的エラー・メッセージも含まれます）は，それぞれアルファベット順に説明します。

C.1　ダイアログ・ボックス

コマンドの実行時さらに情報が必要な場合には，ダイアログ・ボックスが表示されます。ダイアログ・ボックスのタイトルには必要とされる情報が，内部にはそれまでに与えられた応答のヒストリ・リストが表示されます。

ダイアログ・ボックスには，次の２つの対応方法があります。

- 応答を入力し，↵を押して確定する。
- ESCキーを押してダイアログ・ボックスをキャンセルし，そのダイアログ・ボックスが表示される前のメニュー・コマンドに戻る。

ダイアログ・ボックスには２つの項目（Yes/Noなど）の選択を求めるだけのものもあります。このようなダイアログ・ボックスに対しては次のように選択してください。

- カーソル・キーにより選択して↵を押す。
- YまたはNを押して選択する。
- ESCキーを押してコマンドをキャンセルする。

以下に，すべてのダイアログ・ボックスをアルファベット順に示します。

Already recording, do you want to abort ?

すでに記録中です。中止しますか？

キー・ストローク・マクロの記録中です。現在のマクロの記録を終了するかまたは中止するまでは，別のキー・ストローク・マクロの記録を開始することはできません。

マクロの記録を中止するときはYを，記録を続行するときはNを押してください。

Device error - Retry ?

デバイス・エラー…再試行しますか？

プリンタのようなキャラクタ・デバイスに書き込んでいるときに，エラーが発生しました。このエラーは，プリンタのケーブルが接続されていないとき，プリンタがオフラインになっているとき，用紙がなくなったときなどに発生します。

エラーの原因を取り除いてから，再試行するときはYを，キャンセルするときはNを押してください。

Disk error on drive ddd - Retry ?

ドライブdddでディスク・エラー…再試行しますか？

ドライブdddにアクセスしているときに，ハードウエア・エラーが発生したことを示します。このエラーの原因は，多くの場合，ドライブにディスクが挿入されていないか，またハード・ディスクではアクセスした部分が読み出しまたは書き込みできないことです。

再試行するときはYを，キャンセルしたいときはNを押してください。

Edit watch expression

監視する式を編集してください。

式の一部を変更したり，式全体を新しい式に書き換えることができます。ダイアログ・ボックスは，現在ハイライト表示されている監視する式に初期化されます。

Enter address, count, byte value

アドレス，バイト数，バイト値を入力してください。

希望するバイト値にセットしたいメモリ・ブロックの先頭アドレス，ブロックのバイト数，セットしたいバイト値を入力することができます。

Enter address to position to

プログラム中で見たいアドレスを入力してください。

関数名，行番号，絶対アドレスまたはポインタ式を入力できます。アドレスの入力については第9章　式を参照してください。

Enter animate delay (10th of sec)

Animateの遅延時間を入力してください（1/10秒単位)。

Animateコマンドによるトレースの進行速度を指定することができます。指定する数が大きいほど，ステップ（1ソース行または1命令の実行）間の遅延時間が長くなります。

Enter code address to execute to

実行を停止するコード・アドレスを入力してください。

実行を停止したいプログラム内のアドレスを入力してください。

Enter comment to add to end of log

ログの最後に付け加えるコメントを入力してください。

Logウインドウによって表示されるメッセージに付け加える，任意のテキスト行を入力してください。どんなテキストでも入力することができ，タイプされたとおりにログに書き込まれます。

Enter expression for conditional breakpoint

条件ブレークポイントのための式を入力してください。

条件ブレークポイントにセットする条件式を入力してください。ブレークポイントがトリガされるためには，ここで入力した式が真（ゼロ以外）にならなければなりません。

この式は，プログラムの実行中に該当のブレークポイントに遭するたびに評価されます。起こり得る副作用には十分注意してください。

Enter expression to evaluate

評価する式を入力してください。

その値を知りたい式を入力してください。その値と型は，エラーと同じ形式のウインドウに表示され，次に何かキーを押すとそのウインドウが消えます。

Enter expression to watch

監視する式を入力してください。

Watchウインドウの中に入れて値を監視したい変数の名前，または式を入力してください。アドレスを指す式だけでなく，次のようなアドレスを参照しない式も入力できます。

```
x * y + 4
```

テキスト・ペイン内で指定されている変数，または式がダイアログ・ボックス内に表示されているときは，↵を押してそれを受け入れます。部分的に変更したり，まったく別の変数または式を入力することもできます。

Enter inspect start index, range

調べたい先頭の要素の添字と範囲を入力してください。

配列の中の表示させたい先頭の要素の添字と，表示させたい要素の数を入力してください。要素の添字と個数は，カンマ（,）で区切ってください。

Enter instruction to assemble

アセンブルする命令を入力してください。

コード・ペインの中のカレント・アドレスのアセンブラ命令と置き換えるアセンブラ命令を入力してください。

Enter log file name

ログ・ファイル名を入力してください。

ログを書き込みたいファイルの名前を入力してください。Close log fileコマンドを入力するまでは，ログに送られるすべての行がディスプレイに表示され，このファイルにも書き込まれます。

ファイル名を入力しないで↵を押すと，ディフォールトのファイル名が使用されます。ディフォールトのファイル名は，ディバグしているプログラムと同じファイル名で，拡張子が.LOGになります。

Enter memory address

メモリ・アドレスを入力してください。シンボル名や式を使うこともできます。

Enter memory address, count

メモリ・アドレスとカウントを入力してください。

メモリ・アドレスに続けて，カンマ（省略可能）と項目の個数を入力してください。シンボル名または完全な式を入力することができます。

Enter name of configuration file

コンフィギュレーション・ファイルの名前を入力してください。

読み込みまたは書き込みを行うコンフィギュレーション・ファイルの名前を入力してください。コンフィギュレーション・ファイルから読み込むときは，MS-DOS形式のワイルド・カード・マスクを使って一致するファイルのリストを表示させ，そのリストから選択することもできます。

Enter name of file to view

参照するファイルの名前を入力してください。

MS-DOS形式のワイルド・カードを使ってファイルのリストを表示させ，そのリストから選択することもできます。また直接ファイル名を入力することもできます。

Enter new bytes

新しいバイトを入力してください。

ファイル中のカーソルにより指定した位置のバイトと置き換えるバイト・リストを入力してください。バイト・リストの説明は第９章　式を参照してください。

Enter new data bytes

新しいデータ・バイトを入力してください。

セグメント内の，カーソルにより指定した位置のバイトと置き換えるバイト・リストを入力してください。バイト・リストの説明は第９章　式を参照してください。

Enter new directory

新しいディレクトリを入力してください。

カレント・ディレクトリにしたい新しいディレクトリ名（ドライブ名やパスを含む）を入力してください。

Enter new file offset

新しいファイル・オフセットを入力してください。

ディスク・ファイルを16進データ・バイト形式で表示しているときにGotoローカル・コマンドを選択すると，このダイアログ・ボックスが表示されます。表示させたいデータ・バイトの位置を，ファイルの先頭からのオフセットで入力してください。ファイルの内容が，指定されたオフセットから表示されます。

Enter new line number

新しい行番号を入力してください。

カレント・モジュール中で表示させたい行番号を入力してください。ファイルの最後よりあとの行番号を入力すると，ファイル中の最後の行が表示されます。行番号は，ファイルの先頭行を１として始まります。カーソルが位置している現在の行番号は，Moduleウインドウの最上行に表示されます。

Enter new relocation segment value

新しい再配置セグメント値を入力してください。

現在の言語で有効な式を入力してください。この値は，File | Symbol loadコマンドでロードされたシンボル・テーブルのベース・セグメント・アドレスをセットするために使われます。入力する式は，シンボル・テーブルに適用されるコードの始まりのセグメント値として評価されます。

Enter new value

新しい値を入力してください。

現在ハイライト表示されているCPUレジスタにセットする新しい値を入力してください。式を入力して新しい値を生成することもできます。

Enter port number

ポート番号を入力してください。

読み込みたいI/Oポートの番号を入力してください。有効なポート番号は0-65,535です。

Enter port number, value to output

ポート番号と出力する値を入力してください。

書き込みを行いたいポート番号と書き込む値を入力してください。ポート番号と値の間はカンマで区切ってください。有効なポート番号は0-65,535です。

Enter program name to load

ロードするプログラムの名前を入力してください。

ディバグするプログラムの名前を入力してください。MS-DOS形式のワイルド・カードを使ってファイルのリストを表示させ，そのリストから選択することもできます。またロードするファイルの名前を入力することもできます。ファイル名に拡張子を付けないときは，.AXEを指定したものとみなされます。

Enter read file name

リードするファイル名を入力してください。

メモリに読み込みたいファイル名，またはワイルド・カードを含む指定を入力してください。ワイルド・カード指定を入力するか，またはディフォールトの＊.＊を受け入れたときは，一致するファイルのリストが表示され，その中から選択することができます。

Enter search bytes

検索するバイトを入力してください。

検索したいバイト・リストを入力してください。検索は，カーソルにより指定されるメモリ位置から開始されます。バイト・リストの説明は第９章　式を参照してください。

Enter search instruction or bytes

検索する命令またはバイトを入力してください。

命令は，Assembleローカル・メニュー・コマンドに対して入力するのと同じように入力してください。バイト・リストは，データ・ペインのSearchコマンドに対して入力するのと同じように入力してください。

Enter search string

検索する文字列を入力してください。

正確に記憶していない文字列を指定するときは，MS-DOS形式のワイルド・カード機能を使うことができます。“*”は0個以上の任意の文字に相当し，“?”は任意の1文字に相当します。

Enter source address, destination, count

ソース・アドレス，デスティネーション・アドレス，バイト数を入力してください。

移動させたいメモリ・ブロックのアドレス，移動先のアドレス，移動させるバイト数を入力してください。各式はカンマ（,）で区切ってください。

Enter source directory path

ソース・ディレクトリ・パスを入力してください。

ディレクトリのリストを入力してください。各ディレクトリは，空白またはセミコロン（;）で区切ってください。TDシリーズは，ソース・ファイルを見つけるため，リストに現われる順にこれらのディレクトリを探します。

Enter symbol table name

シンボル・テーブルの名前を入力してください。

ディスクからロードするシンボル・テーブル・ファイルの名前を入力してください。通常シンボル・テーブル・ファイルには.TDSという拡張子が付いています。入力の際には拡張子も指定しなければなりません。

Enter tab column spacing

タブ桁の間隔を入力してください。

TDシリーズがFileウインドウやModuleウインドウ内にファイルを表示するときの，タブの間隔を入力してください。指定できる数は1-32です。

Enter variable to inspect

調べたい変数を入力してください。

内容を調べたい変数の名前または式を入力してください。ただし，評価結果がアドレスを指すものでなければなりません。テキスト・ペイン内で指定されている変数または式がダイアログ・ボックス内に表示されているときは，↵を押してそれを受け入れるか，部分的に変更するか，まったく別の変数または式を入力することができます。

Enter write file name

書き込むファイルの名前を入力してください。

メモリ・ブロックの内容を書き込むファイルの名前を入力してください。

Overwrite ddd ?

dddを上書きしますか？

書き込みを行うファイルとして，すでに存在するファイルの名前が指定されています。上書きしてファイルの古い内容を更新するか，コマンドをキャンセルしてそのファイルをそのまま残すかを選択できます。

Overwrite existing macro on selected key ?

選択されたキーにすでに設定されているマクロに上書きしますか？

マクロを設定するキーとして指定した（押した）キーに，すでにマクロが設定されていることを示します。すでに設定されているマクロに上書きしたいときはYを，コマンドをキャンセルするときはNを押してください。

Pick a method name

メソッド名を選択してください。

指定されたルーチン名で参照されるメソッドが，項目中に複数存在します。表示されたリストの中から正しいものを選択してください。

Pick a module

モジュールを選択してください。

プログラムを構成するすべてのモジュールのリストが提示されます。そのリストからModuleウインドウに表示したいモジュールの名前を選択してください。

プログラムのモジュールでないファイルを表示したいときは，View/Fileコマンドを使ってください。

Pick a source file

ソース・ファイルを選択してください。

表示されたリストからソース・ファイルを選択してください。リストには，カレント・モジュールが生成されたソース・ファイルしか表示されません。

Pick a symbol

シンボルを選択してください。

表示されたリストの中からシンボルを選択してください。シンボル名をタイプしていくと，タイプされた文字で始まるシンボルにハイライト表示が移動します。

Pick a window

ウインドウを選択してください。

アクティブ・ウインドウのタイトルのリストからウインドウを選択してください。

Press key to assign macro to

マクロを設定するキーを押してください。

次にそのマクロ・キーに設定したいコマンド・シーケンスを入力してください。コマンドは，マクロの設定のために入力するときも実際に実行されます。マクロの記録シーケンスを終了するには，そのマクロを設定するキーをもう一度押します。このマクロは，ほかのすべてのキー・ストローク・マクロとともにディスクに記録されます。

Press key to delete macro from

マクロを削除するキーを押してください。このキーで，マクロを設定する前の機能に戻ります。

C.2　エラー・メッセージ

TDシリーズは，何か異常なことが発生したとき，エラー・メッセージによってそれを知らせます。たとえば，選択されたコマンドが実行できなかったり，何かがユーザの期待どおりにいかなかったときにメッセージが表示されます。

通常，エラー・メッセージは表示と同時にブザーが鳴るようになっています。ただし，インストール・プログラムTDINSTBXの設定を変更することにより，ブザーを鳴らさないようにすることもできます。

C.2.1　致命的エラー

致命的エラーが発生すると，TDシリーズは必ず動作を中止してMS-DOSに戻ります。致命的エラーには，MS-DOSからTDシリーズを起動しようとしたときに発生するものもあります。また，TDシリーズの実行中何か回復不可能なことが発生したときに起こるものもあります。どちらの場合にも，問題を解決したあとで，MS-DOSプロンプトからディバッガを再起動してください。

Bad configuration file

コンフィギュレーション・ファイルが異常。

コンフィギュレーション・ファイルの内容が破壊されているか，TDシリーズのコンフィギュレーション・ファイルでないかのどちらかです。

Emulator error - check setup

TDシリーズがIEを初期化しようとして失敗しました。

エミュレータ制御ボードがパソコンにしっかりと入っていることと，IEとのインタフェース・ケーブルがしっかり接続されていることを確認してください。確認後もこのエラーが消えなければ当社または特約店へ連絡してください。

Fatal EMS error

致命的EMSエラー

EMSメモリ・ドライバが，回復不可能なエラーのコードを返しました。EMSメモリのハードウエアが故障しているか，ソフトウエア・ドライバが破壊されているかどちらかです。システムをリブートしてもう一度試してください。それでも同じ状況の場合は，EMSハードウエアが破壊されている可能性があります。

Invalid AXE file header

ロード・モジュール中のローディング指令が存在しないか破壊されています。

当社または特約店へ連絡してください。

Invalid switch : sss

スイッチsssが無効

MS-DOSのコマンド・ラインに無効なオプション・スイッチが指定されました。

付録A　コマンド・ライン・オプションを参照してください。

各コマンド・ライン・スイッチの詳細な説明は，第4章　起動を参照してください。

Not enough memory

メモリが足りません。

TDシリーズがローディング中に，作業メモリが足りなくなりました。

Old configuration file

古いバージョン用のコンフィギュレーション・ファイルでTDシリーズを起動しようとしました。TDシリーズの現在のバージョン用に，新しくコンフィギュレーション・ファイルを作成してください。

Target fault - check target power

TDシリーズが，ターゲット・システムを制御しようとして失敗しました。このエラーは，通常，エミュレータに外部クロックを使用するよう設定するときに，ボードに電源が入っていないのが原因です。エミュレータ・クロック制御が正常であること，エミュレーション・ボードとの接続，エミュレーション・ボードの電源がオンになっていることを確認してください。

Unsupported video adapter

TDシリーズが，使用中のディスプレイ・アダプタを確認できません。

C.2.2　その他のエラー・メッセージ

')' expected

')'が必要。

式を評価しているときに，右カッコが足りませんでした。このエラーは，左カッコから始まった式（または式の一部）がそれに対応する右カッコで終わらないときに発生します。

':' expected

':'が必要。

Cの式を評価しているときに，三項演算子（？：）の最初の２つの式を区切る疑問符（?）はありましたが，２番目と３番目の式を区切るコロン（：）が見つかりませんでした。

']' expected

']'が必要。

式を評価しているときに，配列の添字の左カッコ（[）に対応する右カッコ（]）が見つかりませんでした。

このエラーは，組み込みアセンブラを使ってアセンブラ命令をアセンブルするときにも，発生することがあります。ベース・レジスタまたはインデクス・レジスタを使ったメモリ・アクセスで，レジスタの名前を囲む右カッコが脱落している場合です。

Already logging to a file

すでにログをファイルに書き込んでいる。

Open log fileコマンドを使ってファイルをオープンしたあと，Close log fileコマンドを使ってそのファイルをクローズせず，再度Open log fileコマンドを実行しました。

別のファイルにログを書き込みたい場合は，先に現在のログ・ファイルをClose log fileコマンドでクローズしてください。

Ambiguous symbol name

シンボル名があいまいです。

式中に入力されたシンボル名が，C++またはPascalプログラム中のメソッドを特定しておらず，またリストの中から正しいシンボルを選択していません。

式を評価するには，表示されたリストの中から適切なシンボルを選択してください。

Assignment out of range

代入しようとした値が範囲外です。

Pascalで代入を行うとき，変数にその変数の取り得る値の範囲外の値を代入しようとしました。

Bad configuration file name

コンフィギュレーション・ファイル名が正しくありません。

-cコマンド・ライン・オプションで，存在しないコンフィギュレーション・ファイル名が指定されました。

Cannot be changed

変更できないシンボルを変更しようとしました。

直接変更できるシンボルは，スカラ(Cではint，longなど，PascalではByte，Integer，Longintなど)およびPascalのポインタと文字列だけです。構造体または配列を変更したいときは，各メンバまたは要素を一度に1つずつ変更してください。

Can't execute DOS command processor

MS-DOSのコマンド・プロセッサを実行できません。

MS-DOSのコマンド・プロセッサを実行するのに十分なメモリがないか，コマンド・プロセッサが見つからないかのどちらかです。COMSPEC環境変数により，MS-DOSのコマンド・プロセッサを探すディレクトリが正しく指定されているかどうか調べてください。

Can't have more than one segment override

セグメント・オーバライドは1つしか指定できません。

両方のオペランドにセグメント・オーバライドを指定した命令をアセンブルしようとしました。どちらか1つのオペランドだけにしか，セグメント・オーバライドを指定できません。

Can't load segment at 'segment : offset'

ダウン・ロードしたセグメントの最初と最後のバイトをそれぞれチェックし，セグメントがテストに通らないことを認めました。

指定の場所にリード／ライト用のメモリが不足している可能性があります。

Can't set a breakpoint at this address

このアドレスにブレークポイントを設定できません。

ROM，存在しないメモリ，またはセグメント0内にブレークポイントを設定しようとしました。ROM内のプログラムの実行を調べるには，Run | Trace intoコマンドを使って1命令ずつ実行する方法だけです。

Can't set any more hardware breakpoints

これ以上ハードウエア・ブレークポイントを設定できません。

IEのイベント・レジスタの不足が原因で，新しく設定することはできません。すでに設定されているハードウエア・ブレークポイントをどれか解除するか（View | Breakpointsを使用）トレース・トリガをクリアするか（トレース・トリガ制御ダイアログでRun beginsかHalt ends modeを選択）して，まずイベント・レジスタを解放してください。

Can't set any more hardware tracepoints

これ以上ハードウエア・トレース・ポイントを設定できません。

IEのイベント・レジスタの不足が原因で，新しく設定することはできません。すでに設定されているハードウエア・トレースポイントをどれか解除するか（View | Breakpointsを使用）トレース・トリガをクリアするか（トレース・トリガ制御ダイアログでRun beginsかHalt ends modeを選択）して，まずイベント・レジスタを解放してください。

Can't set hardware condition on this breakpoint

このブレークポイントにはハードウエア条件は設定できません。

グローバルでないブレークポイントにハードウエア条件を設定しようとしました。ハードウエア条件はグローバル・ブレークポイントにしか設定できません。

Can't set that sort of hardware breakpoint

指定したハードウエア・ブレークポイントが設定できません。

CONFIG.SYSファイルにインストールされているハードウエア・デバイス・ドライバは，指定されたハードウエア・ブレークポイントをサポートしません。

Can't use same register twice

同じレジスタを二度使用できません。

同一のメモリ・オペランドで，ベース・レジスタまたはインデクス・レジスタを二度使用している命令をアセンブルしようとしました。どのオペランドでも，1つのレジスタは一度しか使用できません。

Cannot access an inactive scope

アクティブでないスコープにはアクセスできません。

Moduleウインドウ内の変数で，現在アクティブな関数の外部にある変数を含む式を入力したか，それらの変数をポイントします。

アクティブでない関数の中の変数の値は無意味であり，そのような変数を式中に使用したり，それらの値を調べたりすることはできません。

Constructors and destructors cannot be called

コンストラクタまたはデストラクタは呼び出せません。

このエラー・メッセージは，オブジェクトを使用したプログラムをディバグしているときにのみ表示されます。

コンストラクタまたはデストラクタであるオブジェクト・メソッドを評価しようとしてもできません。

Control is not writable

制御は書き込み不可です。

リード・オンリーのエミュレータ制御を変更しようとしました。

Destination too far away

デスティネーションが遠すぎます。

デスティネーション・アドレスが，遠すぎる条件付きブランチ命令をアセンブルしようとしました。条件付きブランチ命令のデスティネーション・アドレスは，命令自身のアドレスから－128～＋127バイトの範囲内でなければなりません。

Divide by zero

ゼロによる除算。

入力した式が除算演算子（/，div）または剰余演算子（%，mod）を含んでいるため，演算子の右側の式（除数）の評価値がゼロになります。除算演算子および剰余演算子では，この場合の値は定義されていません。

Edit program not specified

エディタ・プログラム名が指定されていません。

ModuleウインドウまたはDiskウインドウからEditローカル・メニュー・コマンドを使おうとしましたが，インストール・プログラムを使ってエディタ起動コマンドを指定していません。

Emulation memory exceeded at address aaa

アドレスaaaでエミュレーション・メモリを越えています。

使用可能以上のIEのエミュレーション・メモリを割り付けしようとしました。この要求により，使用可能なメモリ全部を使用するようになります。

Error loading program

プログラムのロード中のエラー。

MS-DOSが指定されたプログラムをロードできませんでした。指定されたファイルが適正な.AXEファイルでないか，内容が破壊されている可能性があります。

Error opening file fname

ファイルfnameをオープン中のエラー。

Fileウインドウに表示するように指定されたファイルをオープンできませんでした。

Error opening log file fname

ログ・ファイルfnameをオープン中のエラー。

Log to fileローカル・メニュー・コマンドに指定されたファイル名で，ファイルをオープンできませんでした。ファイルを作成するために必要な空き領域がないか，指定したドライブ，ディレクトリ・パス，ファイル名が正しくないかのどちらかです。

ディスクからファイルのどれかを削除してそのファイルのための空き領域を作るか，正しいディスク，パス，ファイル名を指定してください。

Error reading block into memory

ブロックをメモリに読み込み中のエラー。

指定されたブロックを，ファイルからメモリに読み込むことができませんでした。ファイルのバイト数よりも大きなバイト数を指定した可能性があります。

Error recording key stroke macro

キー・ストローク・マクロの記録中のエラー。

記録されたキー・ストローク・マクロをコンフィギュレーション・ファイルに書き込んでいるときに，エラーが発生しました。このマクロは，ディスクに保存されていない可能性があります。

Error saving configuration

コンフィギュレーションを保存中のエラー。

TDシリーズは，コンフィギュレーション・ファイルをディスクに書き込むことができませんでした。ディスクに空き領域を用意してください。

Error writing block to disk

ブロックのディスク書き込み中のエラー。

指定されたブロックを，指定されたファイルに書き込むことができませんでした。ディスクの空き領域の大きさを越えるバイト数を指定した可能性があります。

Error writing log file

ログ・ファイル書き込み中のエラー。

Logウインドウからの出力をログ・ファイルに書き込んでいるときにエラーが起こりました。ディスクがいっぱいの可能性があります。

Error writing to file

ファイル書き込み中のエラー。

変更をファイルに書き戻すことができませんでした。ファイルが読み出し専用ファイルであるか，ディスクへの書き込み中にハードウエア・エラーが発生した可能性があります。

Expression accesses more than one scope

式が複数のスコープにアクセスしています。

ブレークポイントとの組み合わせで入力した式に，アクセスできないスコープ内の変数の参照があります。Pascalではローカルな変数およびパラメータ，グローバル変数，外部のサブプログラムのローカル変数（そのブレークポイントが，ネストされた手続きまたは関数の内部にあるとき）を参照できます。

Cでは関数の自動変数，モジュールの静的変数，プログラムのグローバル変数を参照できますが，自動変数は１つの関数からしか参照できません。

Expression too complex

式が複雑すぎます。

入力された式が複雑すぎます。演算子とオペランドの数を少なくして式を入力してください。1つの式に入れることができる演算子とオペランドの数は64までです。オペランドは定数や変数など，演算子は，たとえば加算演算子（+），代入演算子（=，:=），構造体メンバ・アクセス演算子（->），集合要素演算子（in）などです。

Expression with side effects not permitted

副作用のある式は使用できません。

入力した式は，評価されるときにメモリ・アドレスの内容を変更します。TDシリーズが繰り返し式を評価しなければならない場合（InspectウインドウまたはWatchesウインドウ内に表示されるときなど）は，このような式は入力できません。

Extra input after expression

式のあとに余分なものが入力されました。

正しい式が入力されましたが，式のあとに余分なテキストがあります。式中の演算子が脱落しているときなどに発生します。

また，使用している言語では不適切な書式で数値を入力した可能性もあります。たとえば，アセンブラ・モードでは，0xF000ではなく0F000hと入力します。

File exists and will be overwritten. Continue ?

IEのコンフィギュレーションの保存ファイルとして既存のファイルを指定しました。ファイルをオーバライトしたくなければESCキーを，新しいコンフィギュレーションで既存のファイルをオーバライトするのならば↵を押してください。

Hardware breakpoint ___ at ___

IE上の指定のハードウエア・ブレークポイントの条件が満足されました。アドレス・フィールドの内容はブレークポイント発生後のPS：PCレジスタの値です。

Help file fname not found

ヘルプ・ファイルfnameが見つかりません。

ヘルプ画面を含むディスク・ファイルが見つかりませんでした。ヘルプ・ファイルがディバッガと同じディレトクリに存在しているかどうか調べてください。

I/O timeout at address___

I/Oバス・サイクルの終了を待つ間にIEがタイムアウトしました。これは，多くはターゲット・システムからのREADY信号がインアクティブのままであるのが原因です。

Illegal procedure or function call

不正な手続きまたは関数呼び出し。

次のような状況では，このエラー・メッセージが表示されます。この場合，関数を正しく評価することができません。

- Pascalのオーバレイ中の手続きまたは関数を呼びだそうとした。
- 現在のプログラム位置がPascalのオーバレイ中にあるときに関数を呼び出そうとした。
- Pascalのスマート・リンカによって削除されたPascalのオブジェクト・メソッドを呼び出そうとした。

Immediate operand out of range

イミーディエト・オペランドが範囲外です。

バイト・サイズのオペランドと１バイトでは表現できないイミーディエト・オペランドが混在する命令が入力されました。

Initialization not complete

初期化が終了していません。

コンパイラの初期化コードによってデータ・セグメントが正しくセットアップされる前に，プログラムが変数をアクセスしようとしました。

ほとんどのプログラム変数は，ソース・コードの先頭までコンパイラの初期化コードを実行したあとでなければ，プログラム変数をアクセスできません。

Initializing emulator-please be patient

エミュレータの初期化中…お待ちください。

IEおよびメモリ全体の初期化には数秒かかります。このメッセージが30秒たっても消えない場合は，当社または特約店へ連絡してください。

Internal error___

内部エラー発生。

重大な内部エラーが発生しました。このエラーが返したエラー・コードを当社または特約店へ連絡してください。

Invalid argument list

引き数リストに誤りがあります。

入力した式の中に，引き数リストの書式に誤りがある手続き呼び出しまたは関数呼び出しが含まれています。

引き数リストは，左カッコで始まり，カンマで区切られた０個以上の引き数(式)が並び，右カッコで終わります。TDシリーズでは，Pascalの引き数のない関数や手続きを呼び出すときでも，空のカッコが必要なので注意してください。

Invalid character constant

文字定数に誤りがあります。

入力された式中に正しくない書式の文字定数があります。文字定数は，１文字を単引用符（'）で囲んだものです。

Invalid far address

farアドレスの書式に誤りがあります。

アセンブルするために入力したBR命令またはCALL命令の，ジャンプ先アドレスとして指定したfarアドレスの書式が間違っています。farアドレスは，４桁の16進数２つをコロン（：）で区切ったものです。

Invalid format string

書式文字列に誤りがあります。

式のあとに書式文字列を入力しましたが，書式文字列が間違っています。書式文字列の詳しい説明は，第９章　式を参照してください。

Invalid function parameters

関数のパラメータに誤りがあります。

式中で関数を呼び出そうとしましたが，その関数呼び出しのパラメータの指定が間違っています。

Invalid instruction

命令に誤りがあります。

入力された命令は正しい命令ニモニックですが，指定されたオペランドが間違っています。通常このエラーは，POP PSをアセンブルしようとしたときに発生します。

Invalid instruction mnemonic

命令ニモニックに誤りがあります。

アセンブルする命令を入力するときに，命令のニモニックを入力しませんでした。命令は，命令ニモニックとそれに続くオプションのオペランドから構成されます。

Invalid operand separator

オペランドの区切り文字に誤りがあります。

Invalid operand(s)

オペランドに誤りがあります。

アセンブルしようとした命令に，間違ったオペランドが指定されています。たとえば，MOV命令にメモリ参照オペランド２つを指定することはできません。また命令の中には，ワード・サイズのオペランドしか操作できないものもあります。

Invalid operator/data combination

演算子とデータの組み合わせに誤りがあります。

式の中に，演算を実行できない演算子とオペランドの組み合わせがあります。たとえば，定数にプログラム中の関数のアドレスを掛けようとした場合などです。

Invalid pass count entered

無効なパス・カウントが入力されました。

1-65,535の範囲にないブレークポイントのパス・カウントを入力しました。

パス・カウントを0にセットすることはできません。パス・カウント1は，ブレークポイントに最初に遭したときにトリガされることを意味します。

Invalid register

レジスタの指定に誤りがあります。

アセンブルする命令の一部として間違った浮動小数点レジスタを指定しました。浮動小数点レジスタは，文字STと，カッコで囲んだ0-7の数字(指定してもしなくてもよい)により指定します。たとえば，STまたはST(4)のように指定します。

Invalid register combination in address expression

アドレス式中に不正なレジスタの組み合わせがあります。

入力した命令に，ベース・レジスタとインデクス・レジスタの間違った組み合わせを含むオペランドを指定しました。アドレス式は，ベース・レジスタのみ，インデクス・レジスタのみ，またはベース・レジスタとインデクス・レジスタを1つずつ含むことができます。ベース・レジスタはBPとBW，インデクス・レジスタはIXとIYです。

Invalid register in address expression

アドレス式中に不正なレジスタがあります。

入力した命令のカッコ［］で囲んだメモリ・アドレス式の中に，間違ったレジスタを使おうとしました。アドレス式には，BW，BP，IX，IYレジスタしか使えません。

Invalid symbol in operand

オペランド中のシンボルに誤りがあります。

アセンブルする命令を入力するときに，オペランドの先頭に，使用してはいけない文字，たとえばコロン（：）などを使用しました。

Invalid typecast

型キャストに誤りがあります。

入力した式中に，構文に誤りのある型キャストが含まれています。

Cの型キャストの正しい構文は，次に示すように左カッコで始まり，データ型の宣言が続き(変数名は除く)，右カッコで終わります。

Invalid value entered

無効な値が入力されました。

メモリ・アドレスの入力を要求するダイアログ・ボックスに対して，整数値でなく浮動小数点数を入力しました。

Key word not a symbol

シンボルでなくキー・ワードです。

入力した式の変数名が入るべき場所（Cとアセンブラのみ）にキー・ワードが入っています。特殊な演算子sizeof以外のキー・ワードは，型キャスト演算の一部として使用するとき以外，式の中には使用できません。

Left side not a record, structure, or union

左側がレコード，構造体，または共用体ではありません。

Cの構造体のメンバ・アクセス演算子（.または->），またはPascalのレコード・フィールド修飾子（.）を使用する式を入力しましたが，これらのシンボルの前（左側）に構造体やレコードの名前，または構造体やレコードを指すポインタがありません。

Memory or I/O timeout breakpoint

メモリまたはI/Oバス・サイクルの終了を待つ間に，IEがタイムアウトしたのが原因でブレークポイントが発生しました。

このエラーの多くは，ターゲット・システムからのREADY信号がインアクティブのままであるのが原因です。失敗したバス・サイクルの元になる命令をちょうど過ぎたところをPS：PCレジスタが指します。

Memory timeout at address___

メモリまたはI/Oバス・サイクルの終了を待つ間に，IEがタイムアウトしました。

このエラーの多くは，ターゲット・システムからのREADY信号がインアクティブのままであるのが原因です。

No coprocessor or emulator installed

コプロセッサまたはエミュレータが組み込まれていません。

メイン・メニュー・バーのView | Numeric processorコマンドを使ってNumeric Processorウインドウを開こうとしましたが，システムに浮動小数点演算用コプロセッサが接続されていません。また，ディバグしているプログラムがソフトウエア・エミュレータを使用していません。

No help for this context

現在の状況のためのヘルプ画面がありません。

ヘルプ画面を表示させるためにF1キーを押しましたが，現在の状況でどうしたらよいかを示す必要なヘルプ画面を見つけることができませんでした。当社または特約店へ連絡してください。

No modules with line number information

行番号情報の付いたモジュールがありません。

View | Moduleコマンドを発行しましたが，TDシリーズは必要なディバグ情報の付いたモジュールを見つけることができません。ディバグ情報の付いたモジュールがないと，Moduleウインドウでソース・モジュールを見ることができません。

このエラーは，通常シンボル・テーブルのないプログラムのディバグ中に発生します。シンボル・テーブルについては，エラー・メッセージ“Program has no symbol table”の説明を参照してください。

No previous search expression

検索対象が指定されていません。

テキスト・ペインのローカル・メニューからNextコマンドを実行しようとしましたが，その前にSearchコマンドで検索対象を指定していませんでした。

Nextコマンドが使えるのは，同じペインでSearchコマンドを発行したあとのみです。

No program loaded

プログラムがロードされていません。

プログラムがロードされていることを必要とするコマンドを発行しました。

このようなコマンドはたくさんあります。たとえばRunメニュー内のコマンドは，どれもプログラムがロードされていなければ実行できません。このようなコマンドを使用するときは，その前に，File | Openコマンドを使ってプログラムをロードしてください。

No source file for module mmm

モジュールmmmのソース・ファイルがありません。

指定されたモジュールのソース・ファイルが見つかりません。ソース・ファイルがカレント・ディレクトリにないときは，Options | Path for sourceコマンドを使って，ソース・ファイルが存在するディレクトリを指定することができます。

No type information for this symbol

このシンボルの型情報がありません。

ディバグ情報の付いていないプログラム変数を含む式を入力しました。このエラーは，ディバグ情報を生成せずにモジュールをコンパイルした場合に発生します。変数名の前に型キャスト式を付けて変数のデータ型を指定すれば，型情報を補うことができます。

Not a function name

関数名ではありません。

関数呼び出しを含む式を入力しましたが，左カッコの前の名前が関数名ではありません。名前の直後にカッコが付いているときは，式はそれを関数呼び出しとみなします。その名前の関数が見つからない場合，このエラー・メッセージを表示します。

Not a memory referencing expression

メモリを参照する式ではありません。

メモリ・アドレスを参照しない式を入力しました。単に値を返す式ではなく，メモリ・アドレスを参照する式を入力しなければならない場合が多くあります。たとえばDataメニューのInspectコマンドは，メモリ内のデータ項目を参照する式の入力を求めます。

Not an Object Pascal or C++ programs

オブジェクトPascalまたはC++のプログラムではありません。

このプログラムは，オブジェクトPascalプログラムでもC++プログラムでもないのでオブジェクトを含んでいません。したがって，選択されたコマンドは実行されません。

Not a record, structure, or union member

レコード，構造体，または共用体のメンバではありません。

Cの構造体のメンバ・アクセス演算子（.または->），またはPascalのレコード・フィールド修飾子（.）のどれかを使う式を入力しましたが，これらのシンボルの前に構造体，レコードの名前，または構造体，レコードを指すポインタがありません。

Not enough memory for selected operation

メモリ不足のため選択されたコマンドが実行できません。

選択されたコマンドを実行するには新しいウインドウを開かなければなりませんが，メモリ不足のため新しいウインドウを開くことができません。ウインドウのどれかを閉じるか，または大きさを縮小してから，このコマンドをもう一度選択してください。

Not enough memory to load symbol table

シンボル・テーブルをロードするメモリが足りません。

ディバグするプログラムのシンボル・テーブルをメモリ・ロードするための空き領域がありません。シンボル・テーブルには，TDシリーズがソース・コードやプログラム変数を表示するときに使う情報が入っています。メモリ上に常駐ユーティリティがある場合は，それらの領域を解放してからディバッガを再起動してください。また，コンパイル時に，ディバグするモジュールとそれに関係するモジュールに対してだけディバグ情報を生成することにより，シンボル・テーブルを小さくして試してみることもできます。

Only one operand size allowed

オペランド・サイズは１つしか指定できません。

２つ以上のサイズ指定子を含む命令を入力しました。オペランドのサイズは，いったんセットしたらそれを変更することはできません。

Operand must be memory location

オペランドはメモリ・アドレスを参照していなければなりません。

式中のメモリを参照しなければならないオペランドに，メモリを参照しないオペランドが指定されています。メモリを参照するオペランドを指定しなければならない演算子は，たとえば代入演算子（＝，＋＝など），インクリメント演算子（＋＋），デクリメント演算子（－－）などです。

Operand size unknown

オペランドのサイズが不明です。

アセンブルする命令を入力しましたが，オペランドのサイズを指定しませんでした。

バイト・サイズのオペランドおよびワード・サイズのオペランドのどちらでも取ることのできる命令の中には，オペランドからサイズを判断することができない場合に，どちらのサイズを使用するか指定しなければならないものがあります。

Path not found

パスが見つかりません。

入力されたドライブ名とディレクトリ名の組み合わせは存在しません。正しいドライブを指定したかどうか，ディレクトリ・パスのスペルに誤りがないかどうか調べてください。

Path or file not found

パスまたはファイルが見つかりません。

ロードするファイル名の入力を求めるプロンプトに対して，存在しないか不正なファイル名またはパスを指定しました。

ロードしたいファイルの正しい名前が分からないときは，ダイアログ・ボックスが現われたときに↵を押してファイルのリストを表示させ，その中から選択してください。

末尾に円記号¥が付いている名前はディレクトリを示します。リストからディレクトリを選択することにより，ディレクトリ・ツリー構造内を上または下に移動することができます。

Program has invalid symbol table

プログラムのシンボル・テーブルが無効です。

プログラムの末尾に連結されているシンボル・テーブルが破壊されています。ファイルをもう一度作成して再ロードしてください。

Program has no symbol table

プログラムにシンボル・テーブルが付いていません。

ディバグするプログラムは正常にロードされましたが，ディバグ・シンボル情報がありません。

CPUウインドウを使って，機械語レベルでプログラムをディバグすることはできますが，名前でコードやデータを参照することはできません。

Program linked with wrong linker version

プログラムが最新バージョンでないリンカでリンクされています。

旧バージョンのディバグ情報を使ってプログラムをディバグしようとしました。最新バージョンのリンカを使ってプログラムを再リンクしてください。

このエラーが発生した場合は，当社または特約店へ連絡してください。

Program not found

指定されたプログラムが見つかりません。正しいプログラム名を入力するか，ファイル・リストからプログラムを選択してください。

Register cannot be used with this operator

レジスタにこの演算子は使えません。

ページ・レジスタまたはインデクス・レジスタを負のオフセットとして使うアセンブラ命令を入力しました。これらのレジスタは正のオフセットとしてのみ使用できます。

Register or displacement expected

レジスタまたは変位でなければなりません。

入力した命令のオペランドの，カッコ [] 内の式に誤りがあります。カッコ内の式にはレジスタ名と定数の変位だけを使うことができ，セグメント・ベースからのインデクスを示すオペランドになります。

Repeat count not allowed

繰り返し回数は指定できません。

繰り返し回数を含む書式文字列を入力しましたが，その書式文字列を適用する式には繰り返し回数を指定できません。

Reserved memory access breakpoint

ターゲット・ソフトウエアがreservedとマッピングされたメモリ・エリアをアクセスしようとしたため，ブレークポイントが発生しました。障害を起こした命令の次をPS：PCが指しています。

Reserved memory access error at address___

reservedとマッピングされたメモリ・エリアへ書き込もうとしました。

Reset IE and CPU ?

このダイアログ・ボックスは，IEをリセットし，リセット動作を避けるエスケープ・パスを避けて，以降のディフォールト・コンフィギュレーションのリロードを本当に行いたいかどうかの確認です。ダイアログ・ボックスのOKボタンか ↵ でエミュレータをリセットするか，ESCキーを押してオペレーションを継続します。

Run out of space for key stroke macros

キー・ストローク・マクロの登録スペースがいっぱいです。

現在のマクロの記録中に，マクロの登録スペースがいっぱいになりました。マクロの登録スペースは，全体で256キー・ストロークです。

Search expression not found

検索式が見つかりません。

指定された文字列またはバイト列が見つかりません。検索は，カーソルで示されるファイルのカレント・ロケーションから始まり，ファイルの最後で終了します。ファイル全体を検索したい場合は，CTRL-ROLLDOWNでカーソルをファイルの先頭に移動してから検索してください。

Source file fname not found

ソース・ファイルfnameが見つかりません。

指定されたモジュールのソース・ファイルが見つかりません。このメッセージを表示する前に，TDシリーズは次のディレクトリを調べます。

- コンパイラがソース・ファイルを見つけたディレクトリ
- −sdコマンド・ライン・オプションおよびOptions | Path for sourceコマンドで指定されたディレクトリ
- カレント・ディレクトリ
- ディバッガがディバグ中のプログラムを見つけたディレクトリ

Options | Path for sourceコマンドを使って，ソース・ファイルが入っているディレクトリを，ディレクトリ検索リストに加えてください。

Symbol not found

シンボルが見つかりません。

無効な変数名を含む式を入力しました。変数名のスペルを間違えたか，アクティブでない手続きや関数，または別のモジュールにあるスコープ外の変数を指定した可能性があります。

Symbol table file not found

指定されたシンボル・テーブル・ファイルが見つかりません。

Syntax error

構文エラーです。

入力した式の構文に誤りがあります。このメッセージよりもっと具体的なメッセージがない場合に表示される一般的なエラー・メッセージです。

Target memory access error at address___

target systemとマッピングされたメモリ・エリアへの書き込みに失敗しました。READY信号を受け取れませんでした。

Too many files match wild card mask

ワイルド・カード・マスクに一致するファイルが多すぎます。

該当するファイルの数が100を越えるワイルド・カード・マスクを入力しました。先頭の100個のファイルが表示されます。

Trace buffer full breakpoint

トレース・セットアップ・ダイアログ・ボックスが選択されたときに，トレース・バッファがいっぱいでかつBreakなのでブレークポイントが発生しました。

Unexpected end of line

行が終わるべきでないところで終わっています。

式を評価している途中で式の終わりに違してしまいました。

Unknown character

識別できない文字です。

式中に使用できない文字を含む式を入力しました。たとえば，Cではバック・クォート(“`”, 0x60)などです。

Unknown record or structure name

識別できないレコード名または構造体名です。

式の中に識別できないレコード名，構造体名，共用体名，menu名があります。

Cの構造体とアセンブラの構造体には，変数名とは別にそれ自身の名前があることに注意してください。

Unknown symbol

識別できないシンボルです。

無効なローカル変数名を含む式を入力しました。モジュール名が間違っているか，そのローカル・シンボル名または行番号が間違っているかどちらかです。

Unterminated string

文字列が終了していません。

入力した文字列を終了する引用符（Cでは”，Pascalでは’）がありません。Pascalでは，引用符を含む文字列を入力するときは引用符（’）を１つ余分に入力しなければなりません。Cでは，引用符を含む文字列を入力するときは，引用符の前に円記号¥を付けなければなりません。

Use Breakpoints/hardware menu

TDシリーズは，Breakpoints | Hardware breakpointコマンドによりハードウエア・ブレークポイントをサポートしています。

Value must be between 1 and 32

値は1-32でなければなりません。

タブの桁数に無効な値を指定しました。タブの桁数として指定できる値は1-32の範囲です。

Value out of range

値が範囲外です。

有効範囲外の値を持つPascal変数を入力しました。

Video mode not available

指定したビデオ・モードは使えません。

46行モードに切り替えようとしましたが，46行モードは使用できない設定になっています。TDシリーズの起動時に－g－コマンド・ライン・オプションが指定されている場合，またはTDINSTBXのMode for haradware | Screen modeオプションでText onlyが指定されている場合は，46行モードは使用できません。

Write read-only memory breakpoint

ディバグ中のソフトウエアがread-onlyとマッピングされたメモリ・エリアへ書き込もうとしたのでブレークポイントが発生しました。PS：PCが，障害を発生した命令の次を指しています。

〔メ　モ〕

付録D　用 語 集

ここでは，本文で頻繁に使用される用語について説明します。コンピュータおよびそのソフトウエアの分野で一般的に使用されている用語と，TDシリーズで特別の意味で使用されている用語があります。

ASCII

American Standard Code for Information Interchange（情報交換のための米国標準コード）の略語。通常，この標準規格で規定された文字および制御命令のセットのコード（符号）を示します。

PS：PC

プログラム・セグメント（PS）とプログラム・カウンタ（PC）のレジスタ・ペアにより指し示されるアドレス。このアドレスを，カレント・プログラム・ロケーション（プログラムの現在位置）と呼びます。

Cの式（C expression）

C言語の文法に従って書かれた式。

CPU

中央処理装置。このマニュアルではV55SC，V55PIを意味します。CPU内部には多数のレジスタやフラグがあります。TDシリーズのCPUウインドウは，CPUの現在の状態を表示します。

CPUのフラグ（CPU flag）

CPUがある状態に設定されていること，または特別の状態が発生したことを示すために使用されるビット。フラグ・レジスタの各ビットを指します。

CPUレジスタ（CPU register）

リード／ライトの時間を短縮するためにCPUチップの内部に設けられたデータの格納場所。V55SC，V55PIには，AW，BW，CW，DW，IX，IY，BP，SP，PS，DS0，DS1，SSその他のレジスタがあります。

EMS（Expanded Memory Specification）

拡張メモリ仕様。TDシリーズは，ディバグするプログラムのためにできるだけ広いメイン・メモリの領域を残すことができるよう，プログラムのシンボル・テーブルをEMSメモリに格納します(メイン・メモリに格納するように設定を変更することもできます)。

Inspectウインドウ

データの値を調べたり，変更したりするために使用されるウインドウ。

アクション（action）

ブレークポイントがトリガされたときに実行される処理や動作。アクションには，プログラムの実行の停止，式の値のログへの書き込み，式の実行などがあります。

アクティブ・ウインドウ（active window）

画面上で現在ユーザが使用しているウインドウ。ウインドウが重なり合っているときは，一番上のウインドウだけがアクティブ・ウインドウです。アクティブ・ウインドウは，タイトルがハイライト表示され，枠線が二重になります。

アクティブ・ペイン（active pane）

アクティブ・ウインドウの中の，現在ユーザの入力を受け付けているペイン。すべてのカーソル移動コマンドおよびローカル・メニュー・コマンドは，このペインに作用します。

アセンブラ（assembler）

機械語命令を人間に読みやすくしたアセンブラ命令を，機械語に翻訳するプログラム。

ウインドウ（window）

画面上で互いに独立に情報を表示することができる四角形の領域（アクティブ・ウインドウ参照）。

ウォッチポイント（watchpoint）

式が真になるのを監視するブレークポイント。

演算子（operator）

オペランドに対する演算を示す記号。たとえば，加算演算子（＋）や乗算演算子（＊）などです。

オペランド（operand）

演算子や命令により演算，操作されるデータ。たとえば，３＊４では３と４がオペランドです。

キャスト（casting）

オペランドのデータ型を一時的に指定したデータ型に変換すること。型キャストまたは型変換ともいいます。

逆アセンブラ（disassembler）

機械語の命令をアセンブラ命令に変換するプログラム。CPUウインドウのCodeペインには，自動的に逆アセンブルした命令が表示されます。

グローバル・ブレークポイント（global breakpoint）

1ソース行または1命令実行されるごとに設定した条件がテストされ，条件が成立するときは設定したアクションが実行されるブレークポイント。

構造体（structure）

Cのデータ型の1つで，異なる型のデータをひとまとめにしたもの。Pascalのレコード（record）に相当します。

コンフィギュレーション・ファイル（configuration file）

TDシリーズの画面やマクロの設定が入っているファイル。

式（expression）

使用する言語の文法に従って書かれた演算子とオペランドの組み合わせ。TDシリーズがサポートする（式を評価できる）言語は，C，Pascal，アセンブラです。

シンボル（symbol）

変数，定数，手続き，関数などの名前。

自動変数（auto-variable）

C言語で，関数が呼び出されて実行するときにスタック上に生成され，実行が終了したときに消滅する変数。自動変数のスコープは，そのブロック内(Cでは1対の中カッコ{}によって囲まれたソース行）だけです。

スカラ（scalar）

Cのchar，integer，floatや，Pascalのbyte，integer，char，booleanのような基本データ型。スカラは，配列や構造体のような，より複雑なデータ型の構成要素になることができます。

スコープ（scope）

通用範囲。ある変数が見えるプログラムの範囲。グローバルなスコープを持つ（プログラムのどこからでもアクセスできる）変数や，モジュールや手続きの内部でしかアクセスできないローカルな変数があります。

スタック（stack）

関数や手続きの呼び出しに関係するデータ（リターン・アドレス，呼び出しパラメータ，その他）が格納されるメモリの領域。

ステップ実行（step）

関数や手続きの呼び出しを単一の要素として扱いながら，プログラムを1ソース行ずつ実行すること。1行ずつ実行したくないサブルーチンはスキップすることができます。

ダイアログ・ボックス（dialog box）

各種設定の確認と指定，情報の入力を行うための画面上に表示されるボックス。

タイプ（type）

データの型。各言語で使用できるデータ型については，その言語の説明書を参照してください。

チェック・ボックス（check box）

オン／オフを切り替えるダイアログ・ボックス中の項目です。オプションがオンにセットされると，チェック・ボックスのカッコの中にXが表示されます（[X]）。

ディフォールト（default）

ユーザが指定しなかったときに自動的に使用される値または設定。

トリガされる（triggered）

設定されたすべての条件（パス・カウント，マッチング条件など）が真になって，ブレークポイントのアクションが実行されること。

トレース（trace）

プログラムを1ソース行または1命令ずつ実行すること。実行する行に関数や手続きの呼び出しがあるときは，その関数や手続きに入って行きます。

トレースポイント（tracepoint）

変数またはメモリの領域が変化するのを監視するグローバル・ブレークポイント。

配列（array）

CやPascalのデータ型のひとつで，型の同じデータをまとめたもの。配列の要素は，添字で指定することができます。

バック・トレース（back trace）

プログラムを1命令ずつ逆にたどりながら，実行の結果をアンドゥしていくプロセス。

パス（path）

階層構造のディレクトリにおけるルート・ディレクトリから目的のディレクトリまでの道筋。TDシリーズが実行可能なプログラムを探すディレクトリは，MS-DOSの環境変数PATHで指定することができます。

ヒストリ・リスト（history list）

各プロンプトごとに保存された，そのプロンプトに対するユーザの応答のリスト。入力を要求するプロンプトが表示されたときは，そのプロンプトのヒストリ・リストの中から選択することができます。

副作用（side effect）

計算式の途中で変数の値が変化すること。たとえば，代入演算子を含む式や，データの値を変更する関数を呼び出す式には副作用があります。

プリフィクス（prefix）

オペランドの前に置かれる演算子。たとえばCでは＋＋x。

プルダウン・メニュー（pull-down menu）

メイン・メニューの項目を選択したときに開かれるメニュー。プルダウン・メニューのコマンドは，どのウインドウからでも有効です。

ブレークポイント（breakpoint）

プログラムの実行がその場所に到達したときに，設定されたアクションが実行されるプログラム内のアドレス（アクション参照）。

ペイン（pane）

論理的に関連した情報が表示されるウインドウ内の領域。各ペインの表示は，ほかのペインとは独立にスクロールすることができます。ウインドウのサイズを変更したときは，そのウインドウ内のペインは，新しいウインドウ・サイズを最も有効に利用するように調節されます。各ペインの内部では，そのペインのローカル・メニュー（ポップアップ・メニュー）を開いてコマンドを選択することができます（アクティブ・ペイン参照）。

ポストフィクス（postfix）

オペランドのあとに置かれた演算子。たとえばCではx＋＋。

ボタン（button）

影付きで表示されるダイアログ・ボックス中の項目。ダイアログ・ボックス内で設定を行ったり，コマンドの実行を指示するために使います。

ポップアップ・メニュー（pop-up menu）

メニュー・バーからプルダウンするのではなく，画面の中央に表示されるメニュー（ローカル・メニュー参照）。

メニュー・バー（menu bar）

画面最上部のメイン・メニューが表示されている行。このメニューの項目を選択すると，プルダウン・メニューが開きます。GRPHキーと各項目の先頭文字を組み合わせて押すことにより，TDシリーズの内部のどこからでも，プルダウン・メニューを開くことができます。

ラジオ・ボタン（radio button）

２個以上のオプションが組になったもので，そのうちの１つだけがアクティブ（オン）になっています。オンになっているラジオ・ボタンは，カッコの中に丸印（●）が表示されます。

レコード（record）

構造体参照。

ローカル・メニュー（local menu）

特定のウインドウあるいはペインでのみ利用できるコマンドのメニュー。GRPH-F10を押すと，現在のペインのローカル・メニューがポップアップします。

ワイルドカード（wildcard）

ファイル名や文字列の一部をマスクするために使用される“＊”および“？”。

“？”は任意の１文字に一致し，“＊”は０個以上の文字に一致します。たとえば，abc＊.1は，abc99.1やabcdef.1には一致しますが，xyz99.1には一致しません。

付録E　索　　引

E.1　50音順

【あ】

【か】

【さ】

【た】

E.2 アルファベット順

【M】

【N】

【O】

【P】

【R】

【S】

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 半導体販売部	〒460 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2755
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	小山支店 小山	(0285)24-5011	富山支店 富山	(0764)31-8461
東北支社 仙台	(022)261-5511	長野支社 長野	(0262)35-1444	三重支店 津	(0592)25-7341
岩手支店 盛岡	(0196)51-4344	松本支店 松本	(0263)35-1666	京都支社 京都	(075)344-7824
山形支店 山形	(0236)23-5511	上諏訪支店 諏訪	(0266)53-5350	神戸支社 神戸	(078)332-3311
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	(0292)26-1717	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)324-5511	沼津支店 沼津	(0559)63-4455	松山支店 松山	(0899)45-4111
群馬支店 高崎	(0273)26-1255	浜松支店 浜松	(053)452-2711	九州支社 福岡	(092)271-7700
太田支店 太田	(0276)46-4011	北陸支社 金沢	(0762)23-1621	北九州支店 北九州	(093)541-2887
宇都宮支店 宇都宮	(0286)21-2281	福井支店 福井	(0776)22-1866		

【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 川崎市幸区堀越三丁目484番地	川崎	(044)548-7950
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 名古屋市中区栄四丁目14番5号 (松下中日ビル)	名古屋	(052)242-2762
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

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)