お客様各位

カタログ等資料中の旧社名の扱いについて

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

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

2010年4月1日
ルネサスエレクトロニクス株式会社

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

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

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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

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

ユーザーズ・マニュアル

RENESAS

保守／廃止

IE-17K-ET

CLICE-ET
バージョン1.6

資料番号 EEU-931（第1版）
発行年月 August 1993 P

保守／廃止

保守／廃止

保守／廃止

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

標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

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

特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート／データ・ブック等の資料で，特に品質水準の表示がない場合は標準水準製品であることを表します。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。
○この製品は耐放射線設計をしておりません。

M4 94.11

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

保守／廃止

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

保守／廃止

はじめに

対象者　このマニュアルは，4ビット・シングルチップ・マイクロコントローラ　17Kシリーズを採用し，IE-17K-ETを使ってその応用システムを設計，開発するエンジニアを対象としています。

目　的　このマニュアルは17Kシリーズの応用システムを設計，開発する際に使用するインサーキット・エミュレータ　IE-17K-ETについてと，そのインタプリタであるCLICE(Command Language for In-Circuit Emulator)-ETについて説明し，このIEの持つ各種機能を理解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，大きく分けて以下の内容で構成しています。

概説
仕様
設置
起動
コマンド説明
エディタ
プログラムの実行
SEボード用PROMの作成
エラー・メッセージ

読み方　このマニュアルの読者には，電気，論理回路，およびマイクロコンピュータに関する一般的知識が必要です。

一通り，IE-17K-ETの機能を理解しようとするとき
→　目次に従って読んでください。

特定の命令の機能を詳細に調べたいとき
→　**第５章　コマンド説明**を読んでください。

凡　例　このマニュアルでは，以下のような記号を使用しています。

↵	：改行入力です。
{ }	：{ } の中に記載されている文字列のいずれかを選択することを意味しています。
[ ]	：入力省略可能を意味します。
_____（アンダバー）	：コンソール入力を意味します。

↑　　：コントロール・キーの入力を意味します。

SPC　　：スペース・キーの入力を意味します。

用語について

このマニュアルの中で使用する用語について，その意味を下表に示します。

用　　語	意　　味
ターゲット・デバイス	エミュレーションの対象となっているデバイスです（本チップ）。
ユーザ・プログラム	ディバグの対象となるプログラムです（ユーザが作ったプログラム）。
ターゲット・システム	ディバグの対象となるシステムです（ユーザが作ったシステム）。 ターゲット・プログラムおよびユーザの作成したハードウエアを含みます。 狭義にはハードウエアのみを指します。

保守／廃止

目　　次

保守／廃止

図　の　目　次

表　の　目　次

保守／廃止

第1章　概　　説

この章では，IE-17K-ETの特徴およびシステム構成について説明します。

1.1　概　　要

IE-17K-ETは，4ビット・シングルチップ・マイクロコントローラ　17Kシリーズ用のソフトウエア開発用サポート・ツールです。

IE-17K-ETは，17Kシリーズの全品種をサポートしています。各品種ごとに専用のSEボードが用意されており，IE-17K-ETはそれらと組み合わせて使用します。SEボードは，各製品特有のハードウエアのエミュレーション機能を持っており，プログラム評価用に単体でも使用できます。

また，IE-17K-ETは，ターミナルを接続してスタンド・アロンで動作することが可能です。ホスト・マシンに接続し，IE-17K-ETとオペレータとのマン・マシン・インタフェース・ソフトであるSIMPLEHOST™を動作させれば，より高度なディバグ環境を実現することもできます。

IE-17K-ETのインタプリタであるCLICE (Command Language for In-Circuit Emulator) -ETとSIMPLEHOSTの使用できるバージョンの組み合わせは次のとおりです。

表1－1　CLICE-ETとSIMPLEHOSTのバージョンの組み合わせ

CLICE-ET / SIMPLEHOST	Ver.1.5	Ver.1.6
Ver.1.10	○	×
Ver.1.11	×	○

1.2 IE-17K-ETの特徴

1.2.1 ターゲット・システムとのインタフェース

ターゲット・システムとのインタフェースに実際の製品を使用していますので，対象製品と同等の電気的特性が得られます。

1.2.2 プログラム・メモリ

プログラム・メモリには，SEボード上に装着されているCMOSスタティックRAMが使用されます。

1.2.3 エミュレーションの方法

ユーザ・プログラムのエミュレーションの方法には，リアルタイム・エミュレーション，1ステップ・エミュレーションがあります。

1.2.4 ブレーク機能

(1) **プログラマブル・ブレーク機能**

プログラマブル・ブレーク機能は次の4段階の設定が可能です。

① 単一条件が成立したらブレークする。

② 上記①の条件を複数個設定し，それらの条件がすべて成立するか，いずれか1つの条件が成立したらブレークする。

③ 上記②の条件を複数個（最大4条件）設定し，それらのいずれか1つの条件が成立したらブレークする。

④ 上記③の条件が設定した順序で成立したらブレークする。

設定可能なブレーク条件としては，以下のものがあります。

- プログラム・メモリ・アドレス
- データ・メモリ・アドレス
- データ・メモリの内容
- レジスタ・ファイルのアドレス
- レジスタ・ファイルの内容
- スタック・レベル
- 割り込み
- DMA
- 命令コード
- 命令実行数
- 条件成立回数

(2) 誤り検出機能

ソフトウエア開発の対象となる製品に許されていない資源をプログラムがアクセスした場合などに，強制的にブレークまたはワーニングを発する機能です。

検出する誤りは以下のものがあります。

・許されないデータ・メモリへのアクセス
・許されないシステム・レジスタへのアクセス
・スタック・レベルのオーバフロー／アンダフロー
・一度もデータを書き込んだことのないメモリの読み出しまたはテスト

1.2.5 リアルタイム・トレース機能

プログラムの実行結果をリアルタイムに記憶する機能で，トレース・メモリの大きさは32Kステップです。

(1) トレース・データは以下のとおりです。
・プログラム・メモリ・アドレス
・実行した命令コード
・スキップされた命令
・書き込まれたデータ・メモリ・アドレス
・書き込まれたデータ・メモリの内容
・各実行命令の相対時間

(2) トレースのオン／オフの条件が指定できます。

1.2.6 データ・メモリ・カバレージ機能

データ・メモリのどのアドレスに対して書き込みが行われたかを記憶する機能です。

この機能により以下の情報を得ることができます。

・未書き込みビット
・“1”が書き込まれたビット
・“0”が書き込まれたビット
・“0”も“1”も書き込まれたビット

1.2.7 プログラム・メモリ・カバレージ機能

プログラム・メモリの各番地を何回実行したかを記憶する機能です。

各番地のカウンタの最大値は255で，255回以上実行した場合は255となります。このカウンタがカウント・アップされるのは，その番地の命令がスキップされずに実行された場合か，またはテーブル参照命令（MOVT等）でその番地が参照された場合であり，命令がスキップされた場合はカウント・アップされません。

1.2.8 その他の特徴

(1) RS-232-Cシリアル・チャネルを2チャネル持ち，チャネル0はコンソール用，チャネル1はPROMプログラマなどに使用します。また，チャネル0はホスト・マシンPC-9800シリーズと接続し，マン・マシン・インタフェース・ソフトであるSIMPLEHOSTを動作させれば，より高度なディバグ環境を設定することができます。

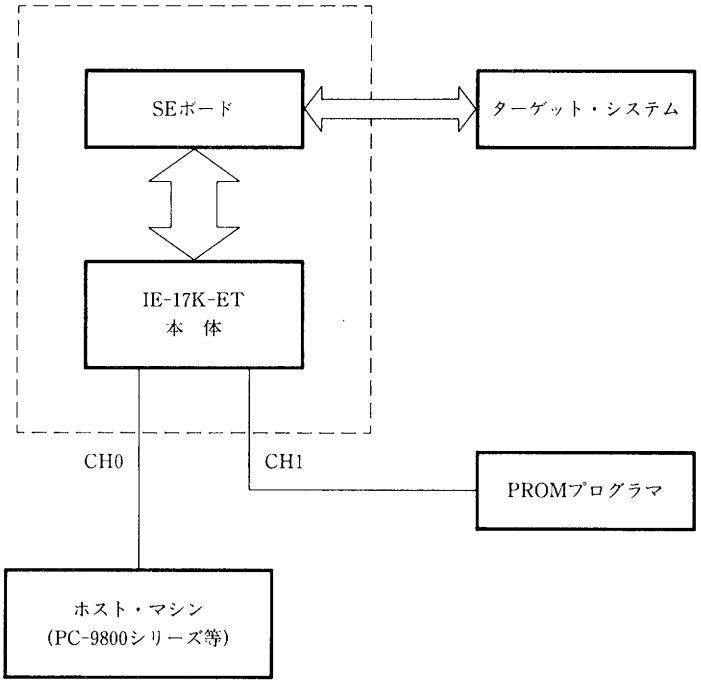
(2) 外形はB5サイズ（25.7 cm×18.2 cm×6 cm）と非常にコンパクトで持ち運びにとても便利です。

保守／廃止

1.3 構　成

1.3.1 システム構成図

図1－1　IE-17K-ETシステム構成図



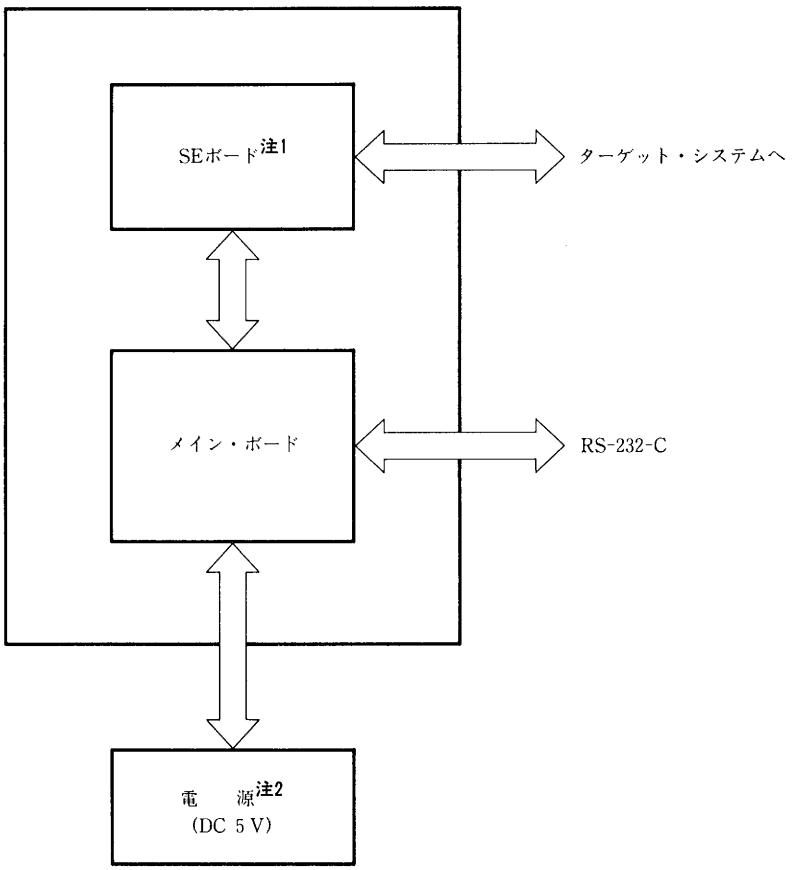
1.3.2 ブロック・ダイアグラム

IE-17K-ETは本体およびその付属品で構成されており，本体は次のような構成要素から成り立っています。

・筐体（接続用コネクタ，スイッチ等含む）

・メイン・ボード

図1－2　IE-17K-ETブロック・ダイアグラム



注1．SEボード（別売）は各品種ごとに用意されています。

2．IE-17K-ETは，電源を内蔵していませんので，市販のDC電源が別途必要です。

保守／廃止

第2章　仕　　様

2.1　コンソール・インタフェース

RS-232-C×2ch（CH0，CH1）

ボー・レート（bps）：600，1200，2400，4800，9600，19200，38400，76800

キャラクタ長　　：7，8ビット

ストップ・ビット長：1，2ビット

パリティ　　：なし，偶数，奇数

2.2　環境条件

動作温度範囲　　0～＋40℃

保存温度範囲　　－10～＋50℃（ただし結露しないこと）

2.3　電　　源

DC 5V±5％　1.0A（MAX.）

　　0.5A（TYP.）

注意　IE-17K-ETは，電源を内蔵していませんので，市販のDC電源が別途必要です。

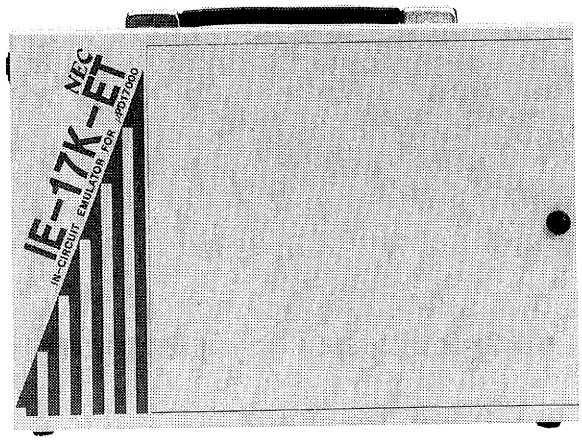
2.4　外形寸法

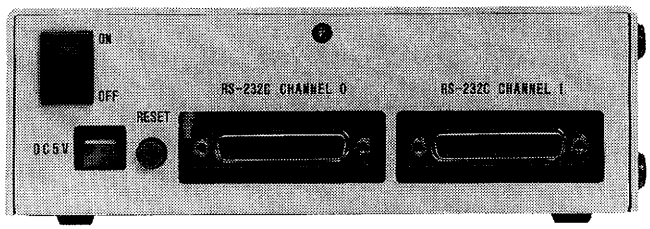
筐体寸法　257 mm×182 mm×60 mm（突起部含まず）

保守／廃止

２.５　外　　観

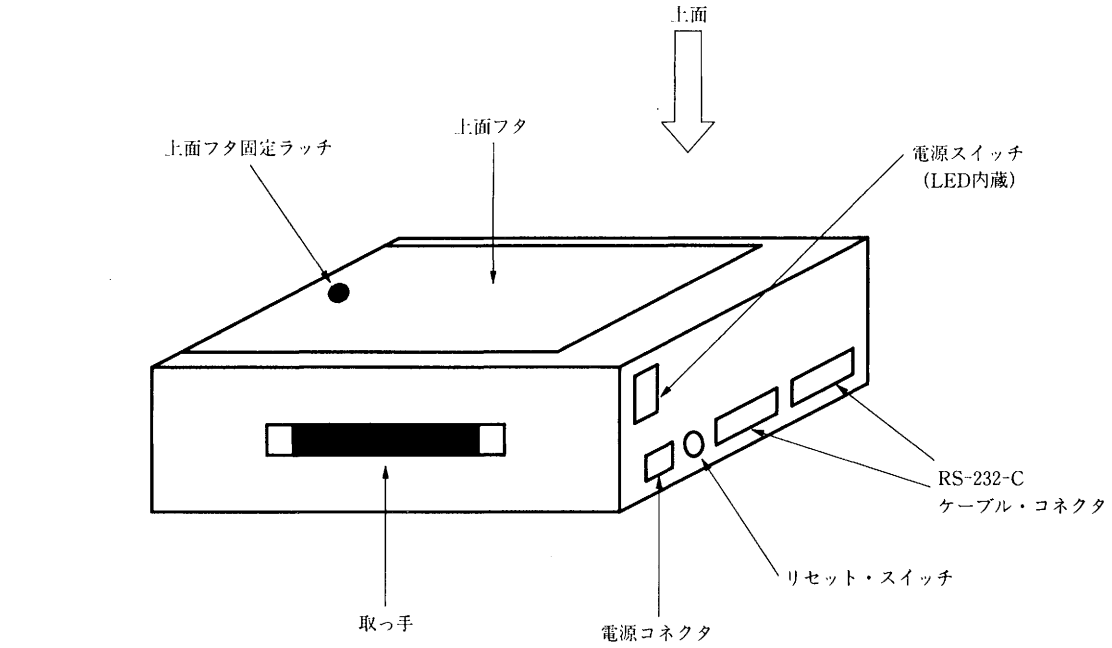
図２－１　IE-17K-ET外観図

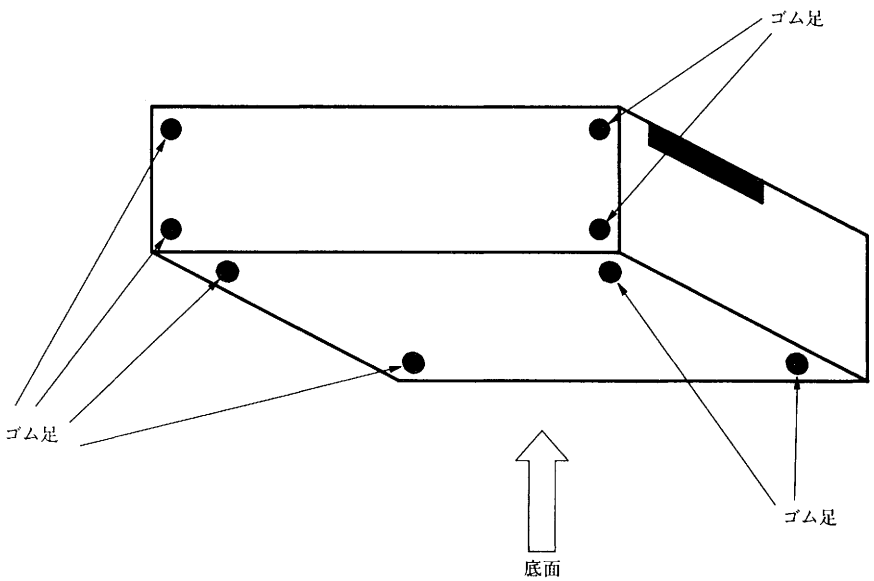




保守／廃止

図2-2　各部の名称



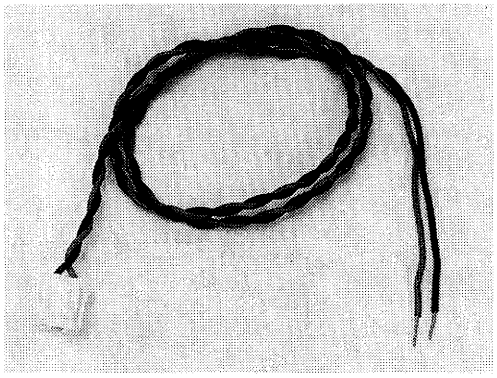


保守／廃止

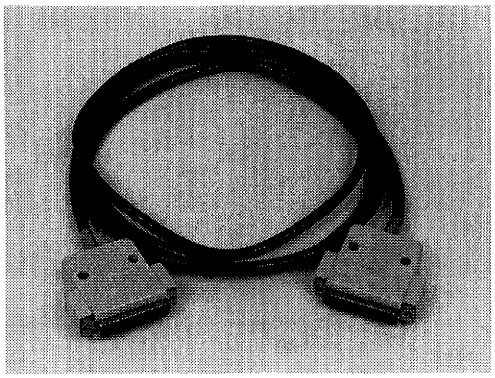
2.6　付 属 品

下記掲載の品物がIE-17K-ET本体の付属品として出荷時，梱包箱中に納められています。

(1)　DC 5 V用電源ケーブル…… 1 本



(2)　RS-232-Cケーブル（クロス・ケーブル）…… 1 本



(3)　そ の 他

・ユーザーズ・マニュアル（このマニュアルおよび英文版）……各 1 部

・保証書　…… 1 部

・梱包一覧表　…… 1 部

第３章　設　　置

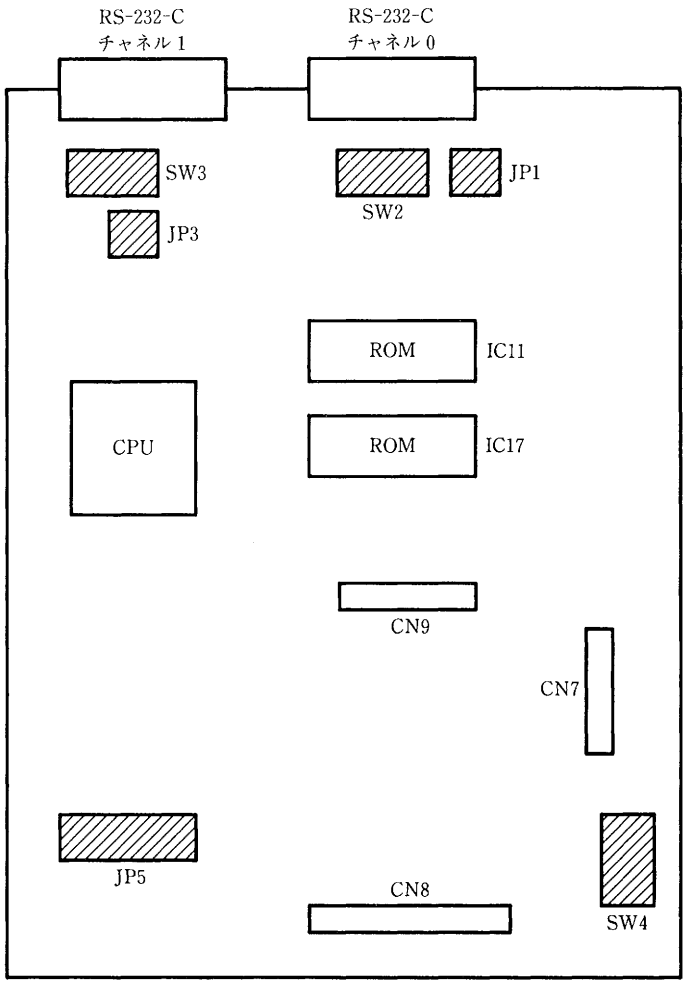
この章では，IE-17K-ETを起動する前に必要な，メイン・ボード上の各スイッチの設定，ホスト・マシンやターミナルとの接続の仕方を説明します。

３.１　スイッチの設定

IE-17K-ETのなかには，メイン・ボードが１枚入っています。

メイン・ボード上のスイッチの配置図を図３－１に示します（図中の斜線の部分がスイッチです）。

図３－１　メイン・ボード上のスイッチ配置図



注意　JP5は設定済みです。設定を変更しないでください。

メイン・ボード上のスイッチは次のように設定します。

○RS-232-C制御用スイッチ

JP1，JP2，SW2，SW3は，RS-232-Cの制御用のスイッチです。

JP1，SW2はチャネル0用，JP2，SW3はチャネル1用です。

JP1，JP2はRTS信号の切り替えです。ご使用のホスト・マシンに合わせて設定してください。

SW2，SW3は，ターミナル・モードとモデム・モードの切り替えスイッチです。

出荷時は以下のように設定されています。

JP1，JP2 ……オープン
SW2，SW3……ターミナル・モード

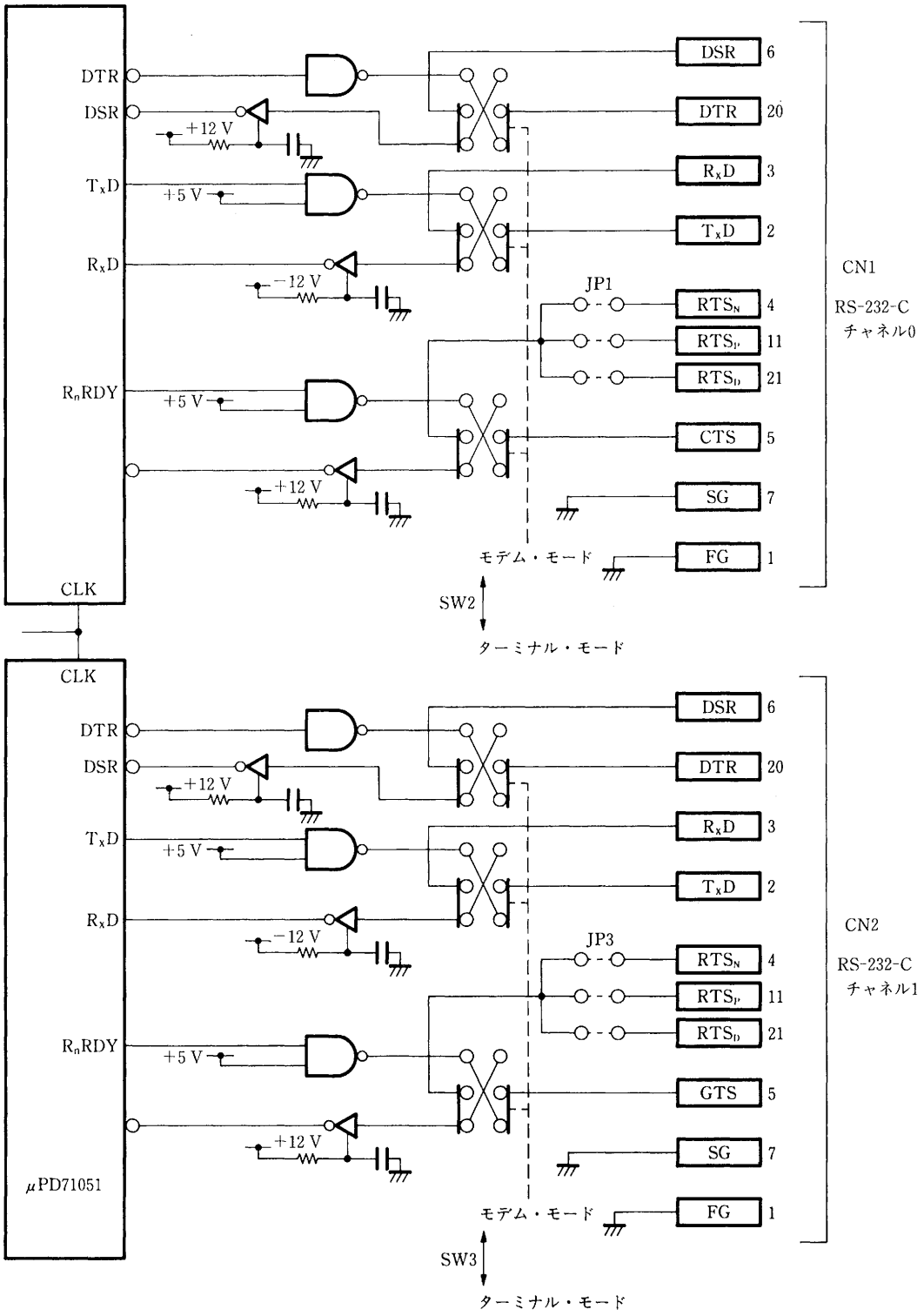
付属のRS-232-Cケーブルを使い，PC-9800シリーズと接続する場合は，上記の状態のままで使用できます。

また，SW4は，RS-232-Cの設定用のDIPスイッチです。

DIPスイッチの設定を図3-4に示します。

保守／廃止

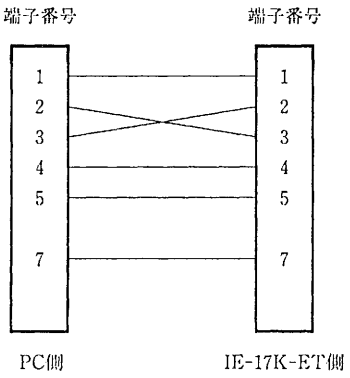
図３－２　RS-232-C　インタフェース回路



保守／廃止

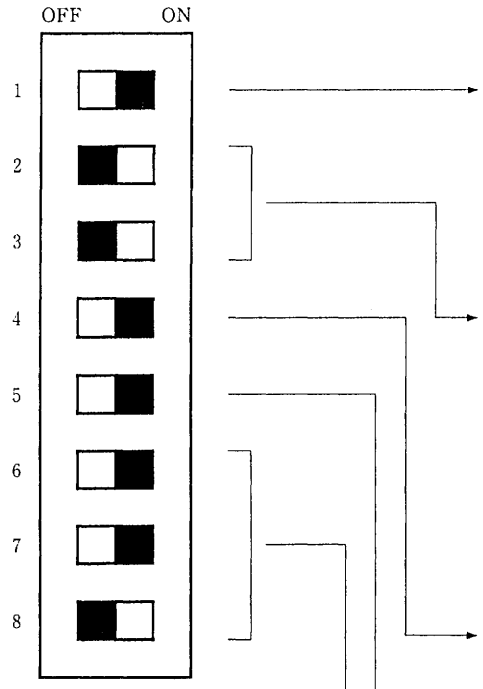
図３－３　付属ケーブルの接続図

〔RS-232-Cケーブル（クロス・ケーブル）〕



保守／廃止

図3－4　DIPスイッチの設定



コントロール方式

1	設　定　内　容
ON	フロー・コントロール
OFF	ライン・コントロール

パリティ・ビット

2	3	設　定　内　容
ON	ON	奇数
ON	OFF	使用不可
OFF	ON	偶数
OFF	OFF	なし

ストップ・ビット

4	設　定　内　容
ON	2ビット
OFF	1ビット

キャラクタ長

5	設　定　内　容
ON	8ビット
OFF	7ビット

ボー・レート

6	7	8	設　定　内　容
OFF	OFF	ON	76800　bps
OFF	OFF	OFF	38400　bps
ON	ON	ON	19200　bps
ON	ON	OFF	9600　bps
ON	OFF	ON	4800　bps
ON	OFF	OFF	2400　bps
OFF	ON	ON	1200　bps
OFF	ON	OFF	600　bps

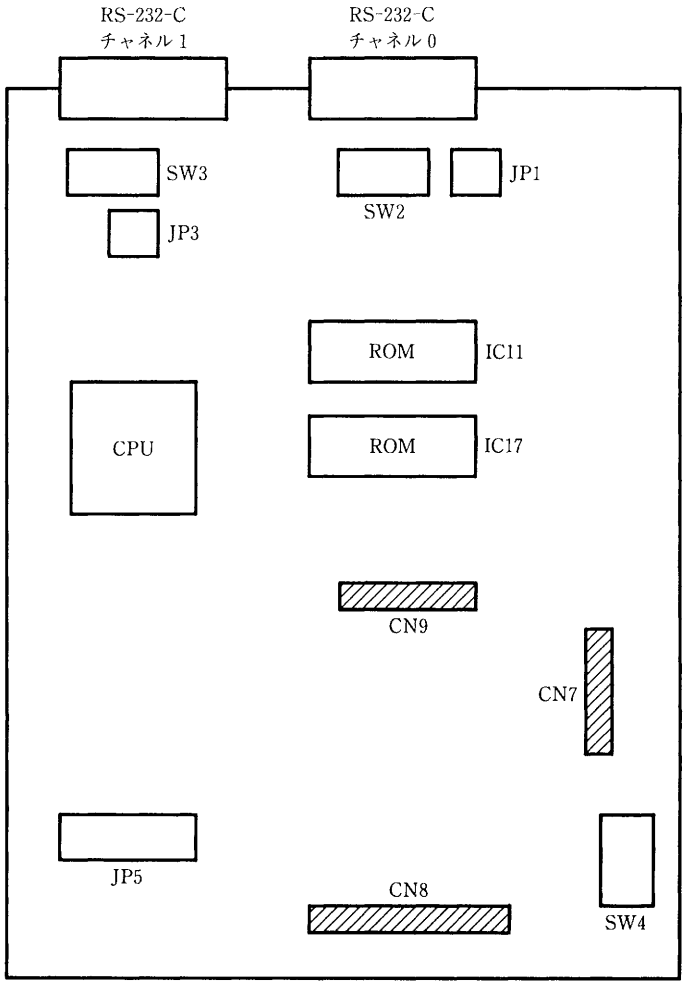
備考　■は出荷時のスイッチの位置

保守／廃止

3.2　コネクタの接続

メイン・ボード上の3つのコネクタ　CN7，CN8，CN9について図3－5に示します（図中の斜線の部分がコネクタです）。

図3－5　メイン・ボード上のコネクタ配置図



CN7，CN8，CN9の3つのコネクタは，SEボードを装着するためのものです。

保守／廃止

3.3 SEボードの実装

出荷時，IE-17K-ET本体には制御用ボードとしてメイン・ボードが搭載されていますが，17Kシリーズの各品種固有に対応したエミュレーション・ボードであるSEボードは搭載されていません。このため，17Kシリーズを開発する際には，IE-17K-ET本体とは別に各品種に対応したSEボードをIE-17K-ETに実装することが必要です。

また，SEボードについての詳しい説明は，各SEボードの**ユーザーズ・マニュアル**を参照してください。

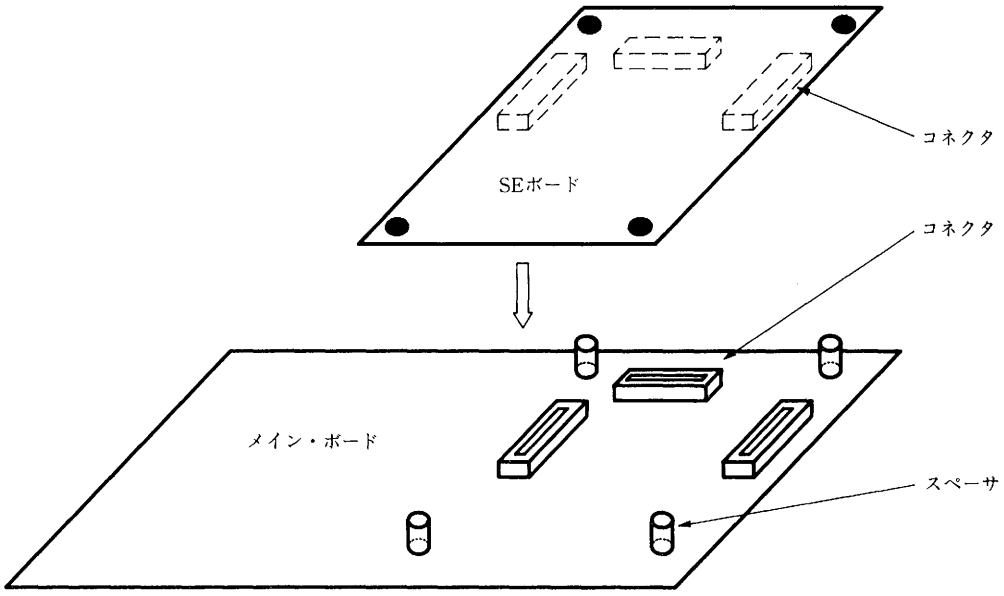
IE-17K-ETに実装して使用する場合の実装手順は次のとおりです。

① IE-17K-ET本体の上面のフタ固定ラッチを引っ張り，上面外ブタを外します。

② メイン・ボード上のコネクタと，SEボード裏面上のコネクタを接続します。

SEボードは，メイン・ボードに対して垂直に押し込むことによって装着され，垂直に引き出すことによって取り外せます。

図3-6 SEボードの装着



③ SEボードとメイン・ボードをネジで固定します。

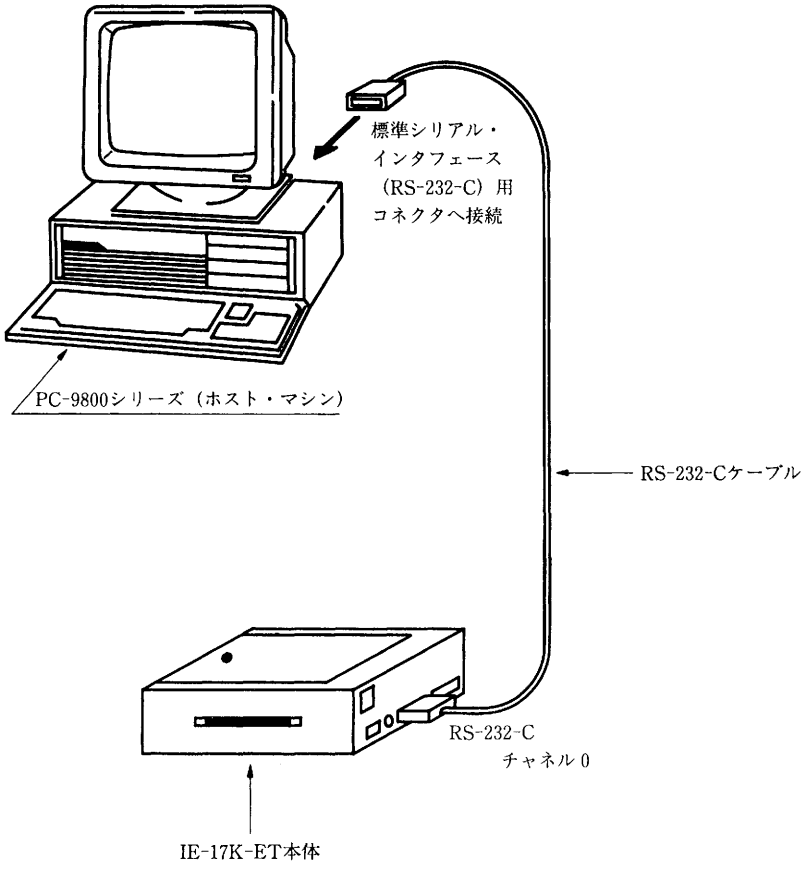
④ 上面内ブタおよび外ブタを元どおりに戻します。

3.4 ホスト・マシンとの接続

ホスト・マシンにPC-9800シリーズを使用したときの例を以下に説明します。

IE-17K-ET本体およびPC-9800シリーズともに電源をOFFにした状態で，IE-17K-ET本体に添付されているRS-232-Cケーブルを使って，IE-17K-ET本体のRS-232-Cのチャネル0のコネクタとPC-9800シリーズの標準シリアル・インタフェース（RS-232-C）用コネクタを接続します。

図3-7　IE-17K-ET本体とPC-9800シリーズの接続

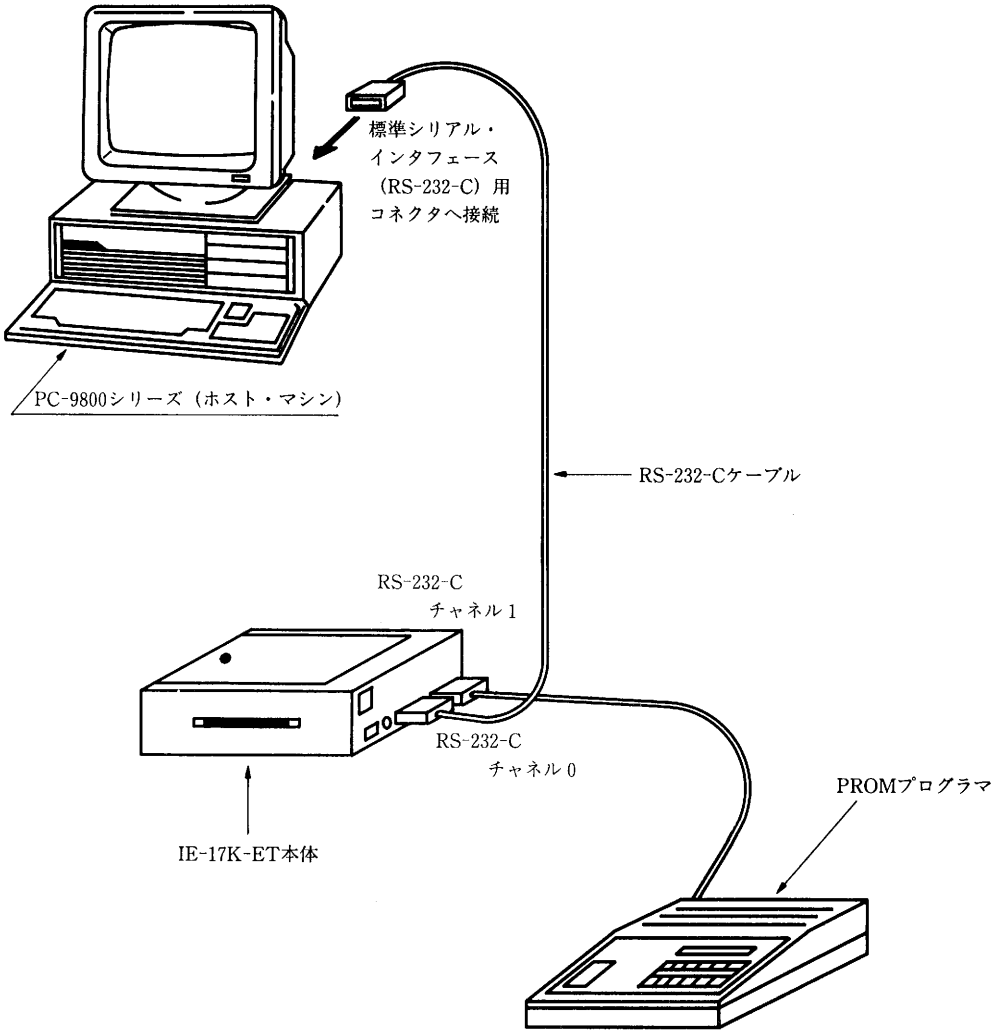


保守／廃止

3.5 PROMプログラマとの接続

IE-17K-ET本体がホスト・マシン（PC-9800シリーズなど）に接続された状態で，IE-17K-ET本体よりPROMプログラマへプログラムをロードするため，IE-17K-ET本体のRS-232-Cのチャネル1のコネクタとPROMプログラマをPROMプログラマ用RS-232-Cケーブルで接続します。

図3-8 IE-17K-ET本体とPROMプログラマの接続

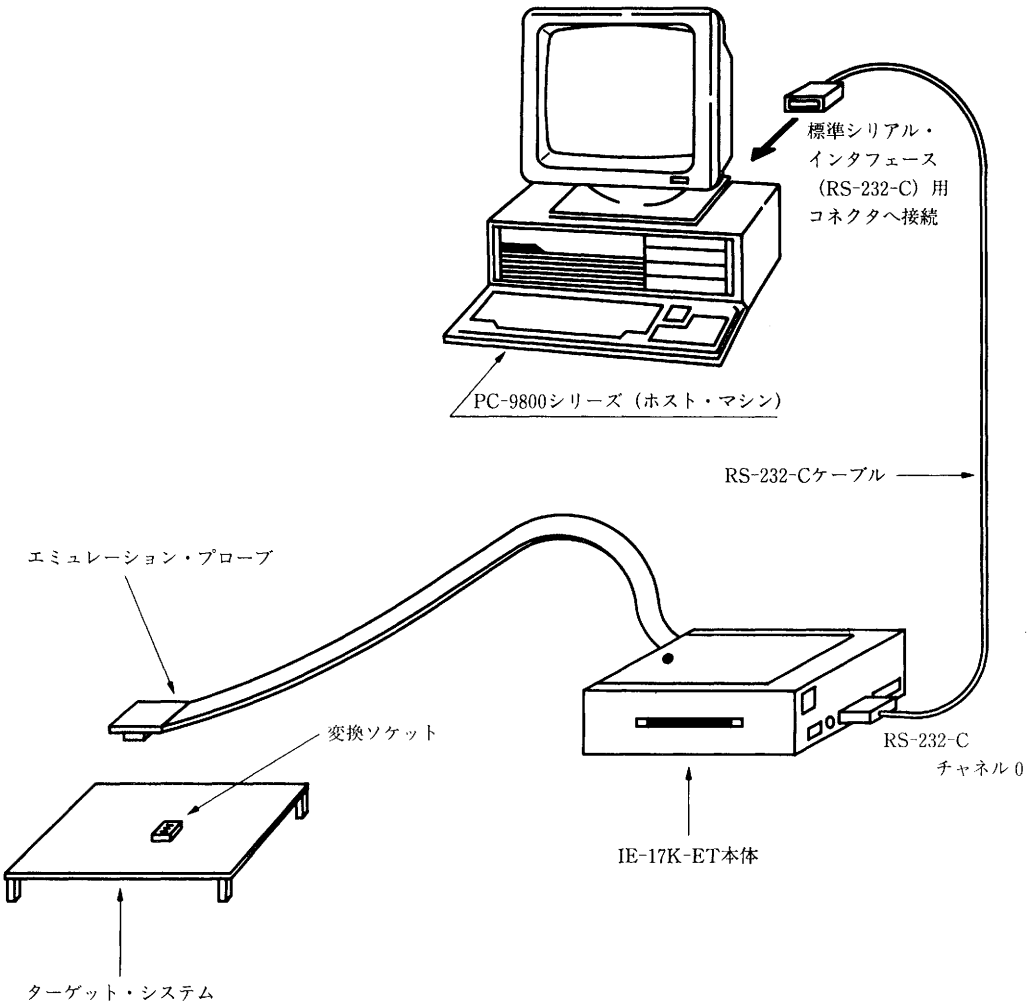


保守／廃止

3.6 ターゲット・システムとの接続

SEボードにエミュレーション・プローブを接続し，ターゲット・システムと接続してください。
詳しくは，各SEボードのユーザーズ・マニュアルを参照してください。

図3-9　IE-17K-ET本体とターゲット・システムとの接続



第４章　起　　動

IE-17K-ETは，ホスト・マシンであるPC-9800シリーズやIBM PC/AT™を，IE-17K-ET専用のRS-232-Cケーブル（IE-17K-ETの付属品）で接続して使用します。

IE-17K-ETとの通信ならびに起動は，RS-232-Cに対応した市販の通信ソフトで行います。

この章では，Windows™ (Version 3.1) のターミナルを用いた通信方法を紹介します。お手持ちの市販の通信ソフトで通信を行われる場合は，この章を参考にして起動を行ってください。

なお，NECではIE-17K-ET専用の通信ソフト（マン・マシン・インタフェース・ソフト）としてWindows上で動作するSIMPLEHOSTを用意しております。

SIMPLEHOSTは，IE-17K-ETとの通信機能を持つほか，ディバグをスムーズにするために，マニュアル・フリーを目指したコマンドのメニュー化，実行結果のグラフィック化などが行われているため，SIMPLEHOSTを使用する場合は，この章を読む必要はありません。

４.１　Windows（Version 3.1）での通信方法

PC-9800シリーズ用Windows (Version 3.1) を使って，IE-17K-ETとPC-9800シリーズとを接続する方法を中心に説明します。

Windowsを起動する前に，SEボードが組み込まれている状態のIE-17K-ETとPC-9800シリーズが，IE-17K-ET専用のRS-232-Cケーブルで接続され，IE-17K-ETに電源が入っていることを確認してください（**第３章　設　　置**参照）。

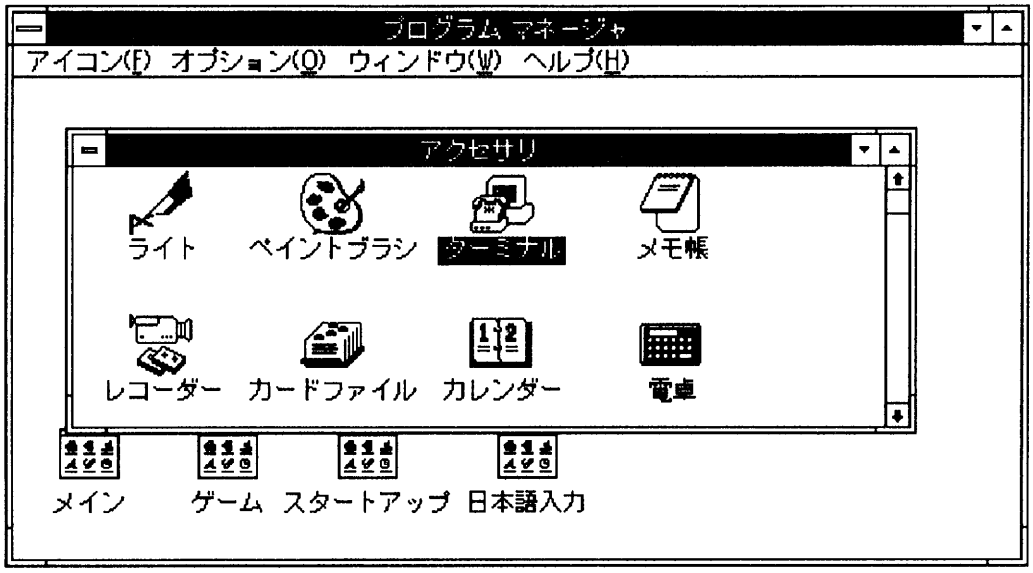
４.１.１　ターミナルの起動

まず，Windowsのオープニング画面の状態から，アイコン表示になっている［アクセサリ］グループを選択（ダブル・クリック）し，さらに［ターミナル］アイコンを選び，実行（ダブル・クリック）します。このときの選択画面の一例を示します。

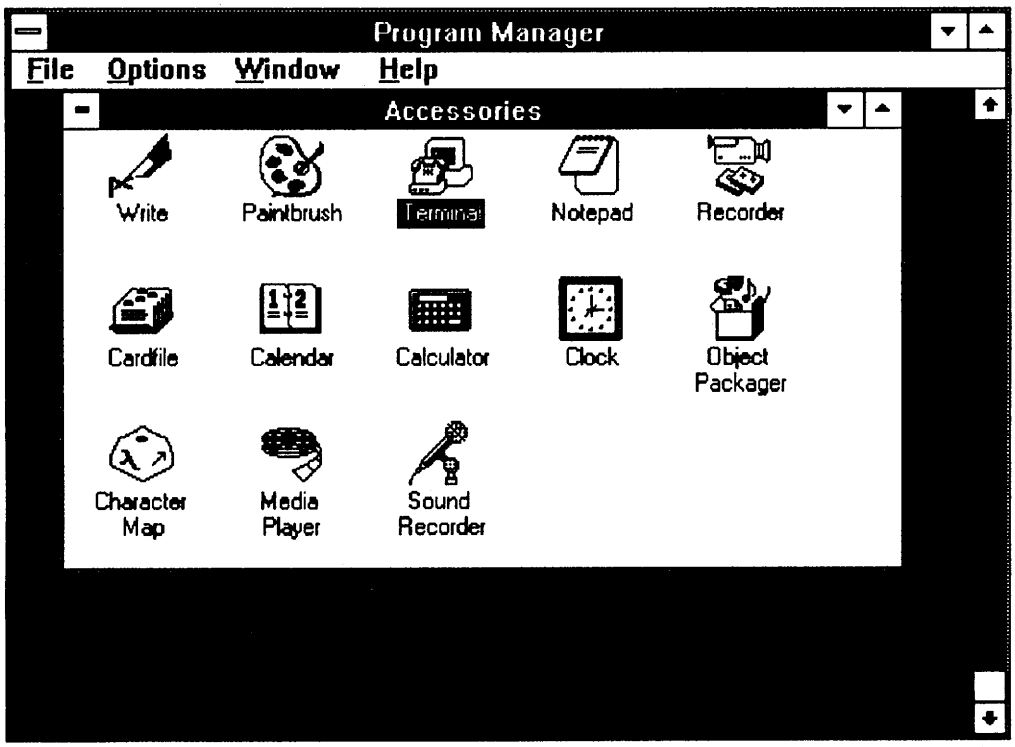
保守／廃止

図４－１　ターミナルの起動（1/2）

(a)　PC-9800シリーズ



(b)　IBM PC/AT

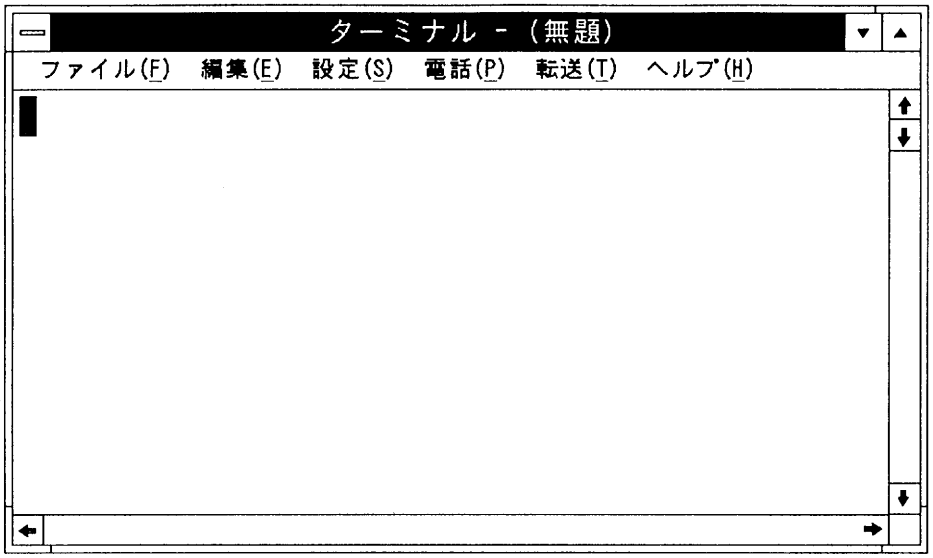


保守／廃止

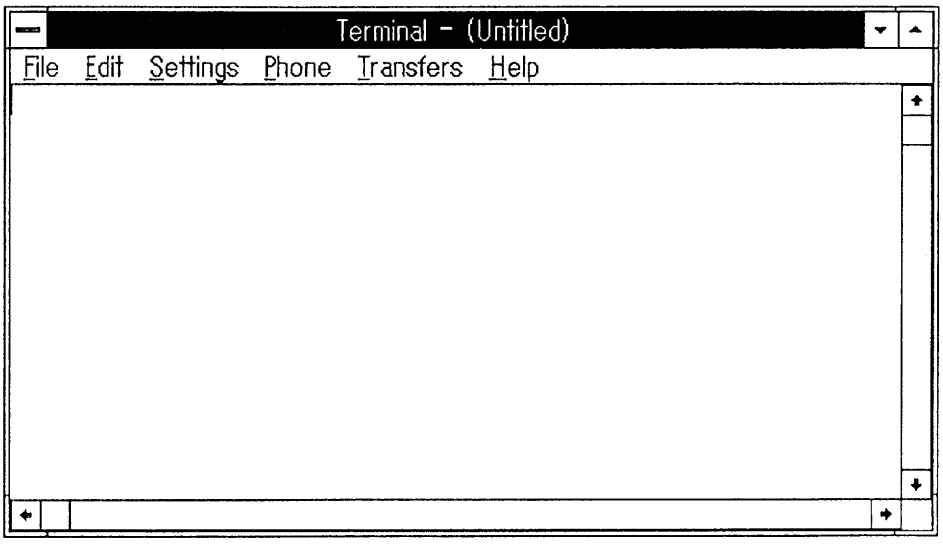
［アクセサリ］グループの［ターミナル］アイコンをダブル・クリックすると，［ターミナル］ウィンドウが開きます。

図4－1　ターミナルの起動（2/2）

(a)　PC-9800シリーズ



(b)　IBM PC/AT



4.1.2 設　定

IE-17K-ETとデータやコマンドのやり取りをするために，通信スピードなどを規定する［通信条件の設定］や通信中の端末の動作を規定する［端末の設定］を設定します。

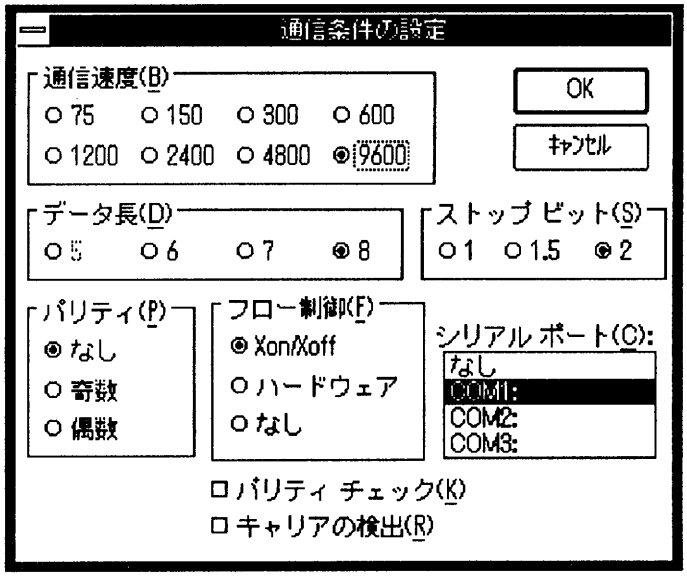
(1) 通信条件の設定

IE-17K-ETの出荷時の通信条件は，通信速度9600ボー，データ長8ビット，ストップ・ビット2ビット，パリティなし，フロー制御Xon/Xoffありとなっています。このためWindowsの設定もこの条件に合うように設定します。

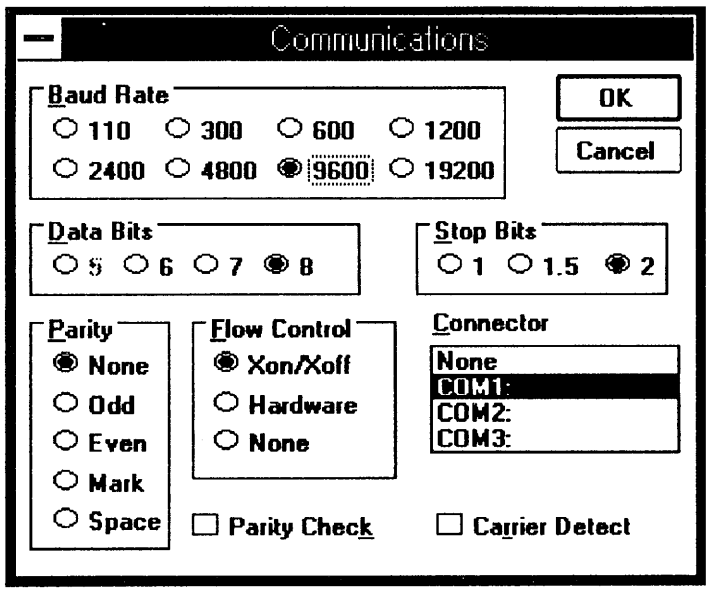
［ターミナル］ウィンドウのメニュー バーの［設定 (S)］，その中の［通信条件］を選びます。すると［通信条件の設定］ダイアログ ボックスが現れますので，図4-2に示すように設定してください。

図４－２　通信条件の設定

(a)　PC-9800シリーズ



(b)　IBM PC/AT

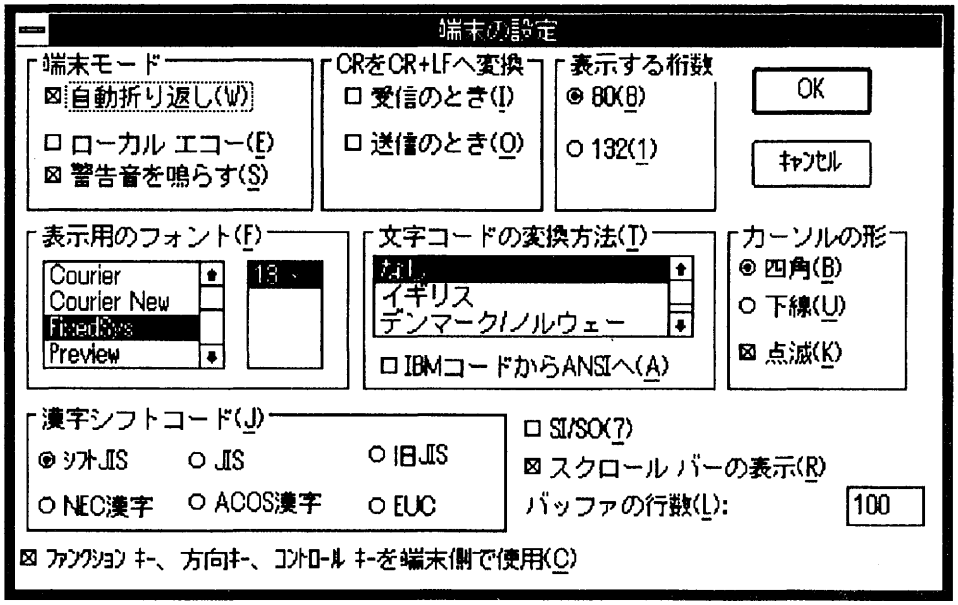


設定が終わったら，［OK］のボタンをクリックして，［通信条件の設定］を終了させます。

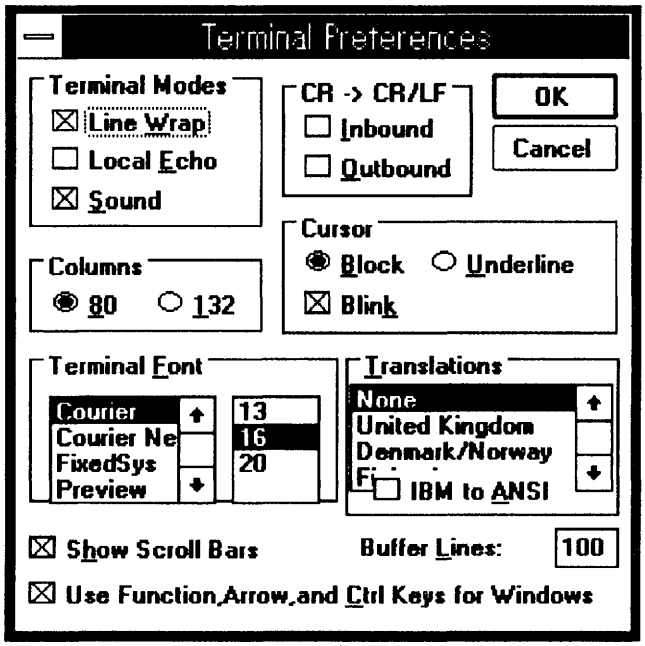
次は［ターミナル］ウィンドウのメニュー バーの［設定 (S)］，その中の［端末の設定］を選び，通信中の端末の動作を規定します。図４－３に［端末の設定］のダイアログ ボックスを示します。［警告音］，［カーソルの形］以外は，図４－３に示すように設定してください。

図４－３　端末の設定

(a)　PC-9800シリーズ



(b)　IBM PC/AT



設定が終わると［OK］のボタンをクリックして，［端末の設定］を終了させます。

保守／廃止

４．１．３　IE-17K-ETへのプログラムのダウンロード

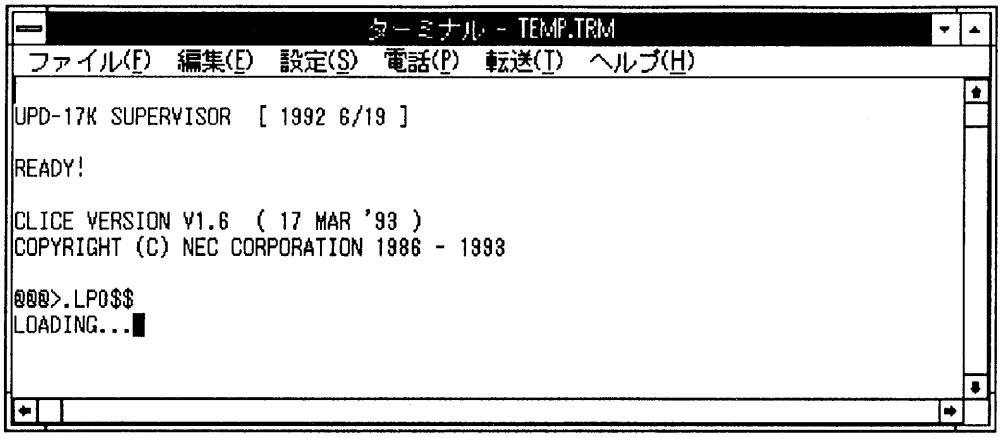
［通信条件の設定］と［端末の設定］が終わると，IE-17K-ETと通信が可能な状態になります。［ターミナル］ウィンドウの画面においてIE-17K-ETのリセット・スイッチを押すと，［ターミナル］ウィンドウの画面にIE-17K-ETよりオープニング・メッセージと＠＠＠＞のプロンプトが返ってきます。このとき，＠＠＠＞のプロンプトが返ってこない場合は，次のような場合が考えられますので，確認してください。

① SEボードが正しく装着されていない
② SEボードに電源が供給されていない（２電源が必要なSEボードも存在します）
③ IE-17K-ET専用のRS-232-Cケーブルが接続されていない

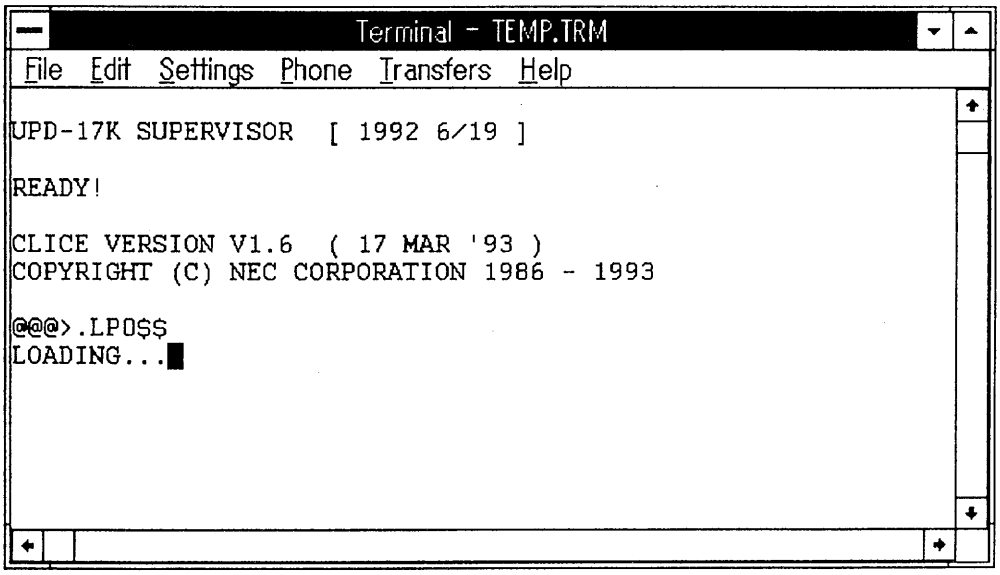
＠＠＠＞のプロンプトが正常に返ってくると，＠＠＠＞のプロンプトに続いて.LP0＄＄とキー入力し，IE-17K-ETをローディング待ち状態にします。図４－４にその様子を示します。

図４－４　ローディング待ちの状態

(a)　PC-9800シリーズ



(b)　IBM PC/AT

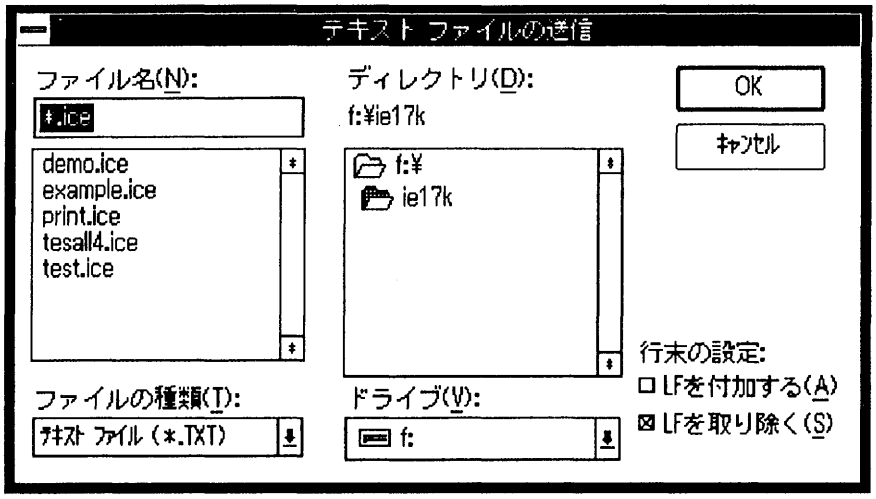


次に，［ターミナル］ウィンドウのメニュー バーの［転送 (T)］を選び，さらにその中の［テキスト ファイルの送信］を選びます。［テキスト ファイルの送信］画面では，［ファイル名 (N)］に＊.ICEと入力します。そのファイルのあるドライブやディレクトリを［ドライブ (V)］［ディレクトリ (D)］で検索，［ファイル名］一覧からファイルを選ぶか，直接テキスト ボックスにファイル名を入力します。ファイルの選択を終えて，［OK］ボタンをクリックするとファイルの送信を始めます。

図４－５に［テキスト ファイルの送信］画面を，図４－６にファイル転送中の画面を示します。

図4-5 テキスト ファイルの送信

(a) PC-9800シリーズ



(b) IBM PC/AT

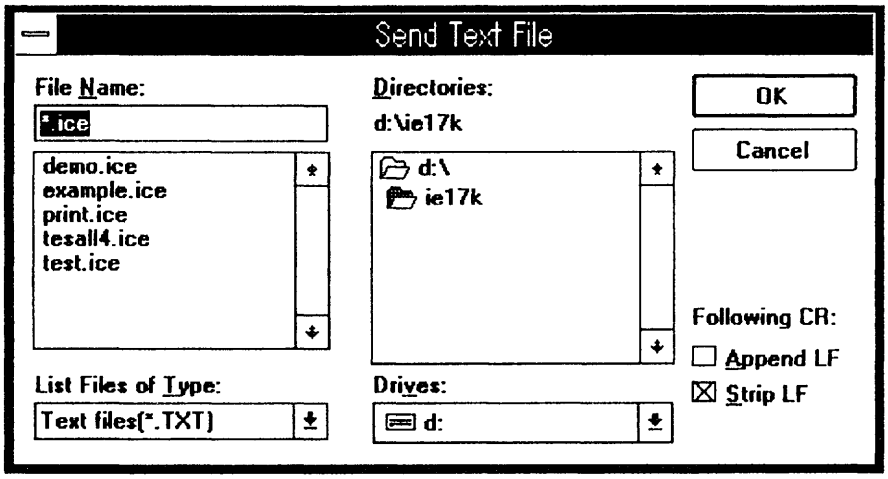
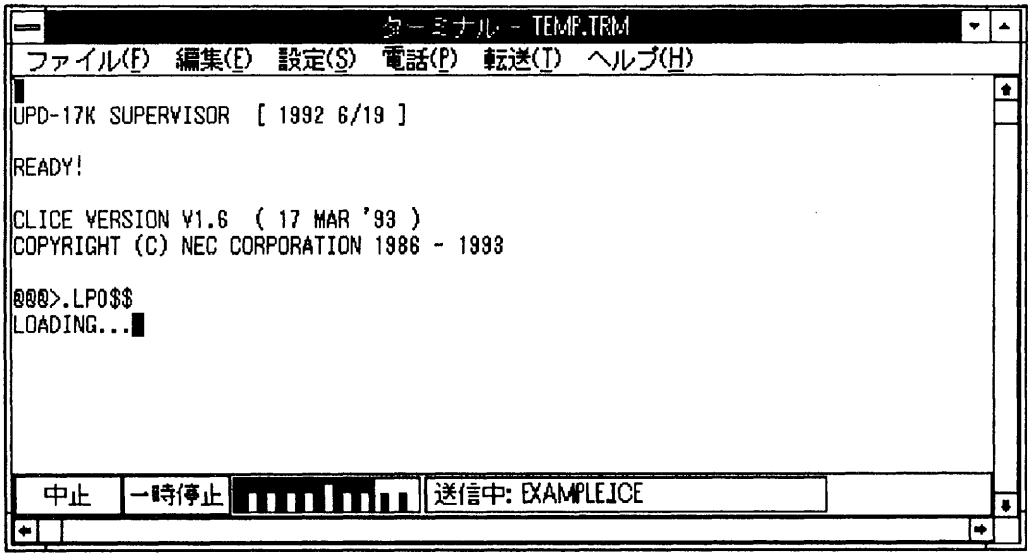
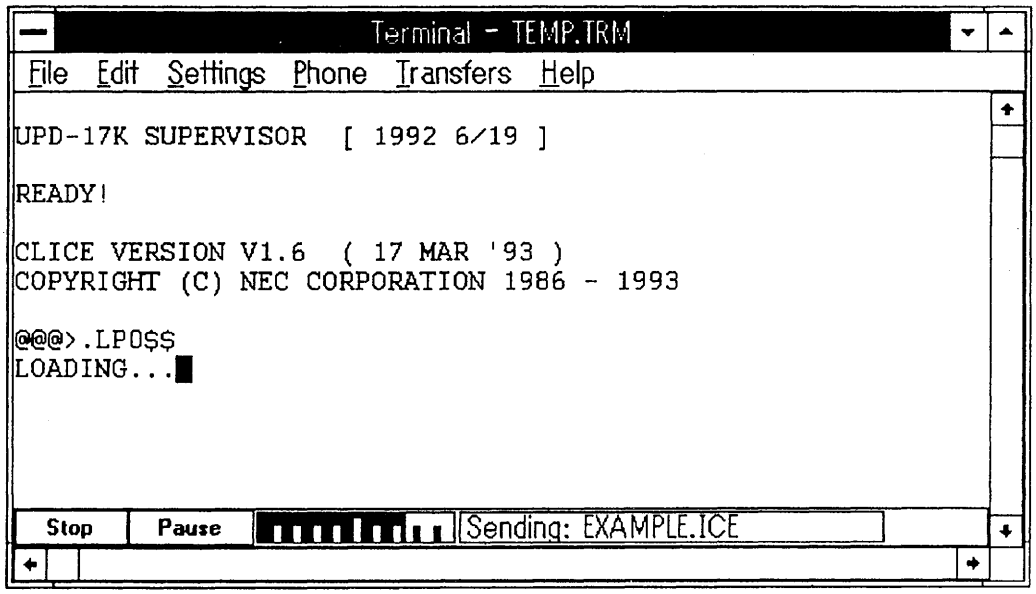


図４－６　ファイル転送

(a)　PC-9800シリーズ



(b)　IBM PC/AT

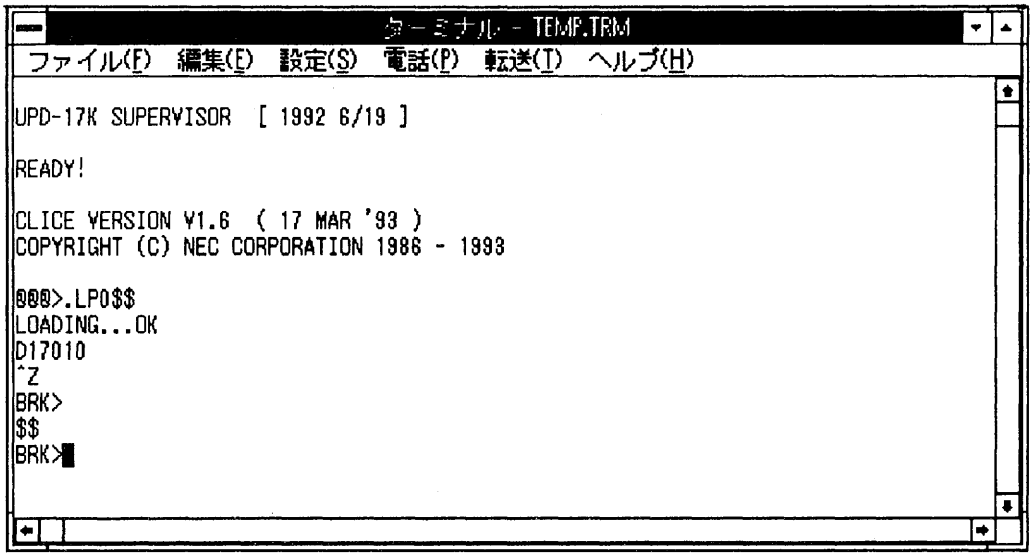


ファイルの転送が完了すると，“OK”と“デバイス・ネーム”とともにBRK>のプロンプトが返ってきます。この状態で，ESCキーを２回または＄＄を入力し，IE-17K-ETがコマンド受け付け可能状態になっていることを確認してください。ESCキー２回または＄＄を入力した結果，BRK>のプロンプトが返ってくれば，.R，.RNなどのIE-17K-ET組み込みマクロ・コマンドが使用できる状態になっています。組み込みマクロ・コマンドについては**5.8　組み込みマクロ・コマンド**を参照してください。

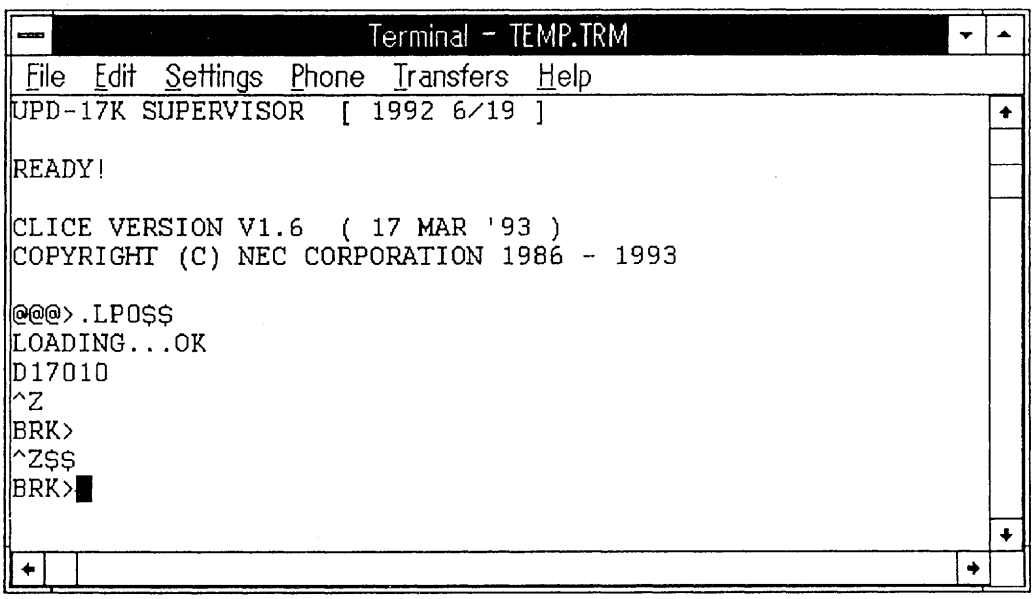
なお，BRK>のプロンプトが返ってこない場合は，SEボード上の本チップに電源が供給されていない，SEボードがリセット状態になっているなどが考えられますので，SEボードのユーザーズ・マニュアルを参照のうえ，設定を確認してください。

図4-7　ファイル転送の終了

(a) PC-9800シリーズ



(b) IBM PC/AT



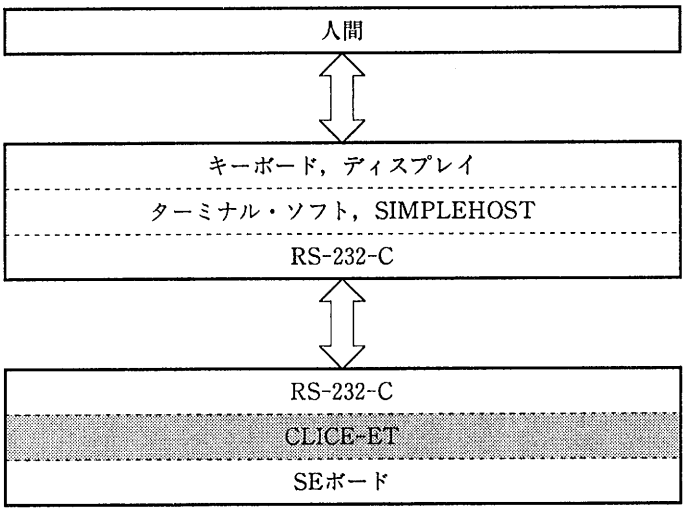
保守／廃止

第５章　コマンド説明

IE-17K-ETは，CLICE（Command Language for In-Circuit Emulator：クライスと読む）-ETと呼ぶコマンド処理系を内蔵しています。

この章では，CLICE-ETがサポートしている全コマンドについて，その表記法，特殊文字などの使用条件および機能の詳細を説明します。

図５－１　CLICE-ETの位置付け



コマンドには組み込みマクロ・コマンドとプリミティブ・コマンドの２種類があります。

組み込みマクロ・コマンドは「．AP」のようにピリオドと２つの英大文字で記述します。組み込みマクロ・コマンドはIE-17K-ETの基本的な機能を使う場合に使用します。

プリミティブ・コマンドは，IE-17K-ETの基本的な機能を用いてプログラム開発を行った経験者に対し，より高度なディバグ方法を提供するためのコマンド群です（**付録Ａ　プリミティブ・コマンド**参照）。

IE-17K-ETの電源を投入しCLICE-ETが起動すると，コンソールにはCLICE-ETのタイトルとバージョンなどが以下のように表示されます。

```
UPD-17K SUPERVISOR[**** **/**]

READY!

CLICE-ET VER. V*.*(** *** ****)
COPYRIGHT (C) NEC CORPORATION 1986-1991
@@@>
```

＊印はバージョン番号および日付を示します。

５.１　プロンプト

CLICE-ETのプロンプトはIE-17K-ET上のターゲット・デバイスの状態を示すものです。また，キー入力が可能な状態にあることを示しています。

プロンプトが表している状態には下記のものがあります。

① ＠＠＠＞ …IE-17K-ETの起動時でHEXファイルのロード待ちを示す。

② BRK>… ターゲット・デバイスが停止中であることを示す。

③ RUN>…ターゲット・デバイスが実行中であることを示す。

④ STP>… ターゲット・デバイスがSTOP命令を実行したことを示す。

⑤ HLT>…ターゲット・デバイスがHALT命令を実行したことを示す。

⑥ DMA>…ターゲット・デバイスがDMAモードで実行中であることを示す。

⑦ DSP>… .DSコマンド実行中を示す。

⑧ RES>… ターゲット・デバイスにリセット信号が入力されたことを示す。

〔注意事項〕

(1) ①のプロンプトが表示されているとき，つまり起動時には，使用する品種がIE-17K-ETに設定されていないため，.Qコマンドを使用してIE-17K-ETを再起動するか，.LP0あるいは.LP1コマンドを入力しプログラムをロードしなければなりません。

(2) コマンド入力中にターゲット・デバイスの状態が遷移したとき（RUNからBRKへプロンプトが変化したときなど）は，新たに出力されたプロンプトに続いてそのときまでに入力されたコマンド列を出力し，引き続きコマンド入力を受け付けます。

（例） 「0,100.DP＄＄」を入力中にRUNからBRKへ変化した場合

```
RUN>0,100          ……ここでRUNからBRKへ変化
BRK>0,100.DP$$……0,100はCLICE-ETが出力
```

上記の例は「0,100」を入力したところで，ターゲット・デバイスの状態がRUNからBRKに変化

したため，自動的に改行され，新しい状態を示すプロンプトと今まで入力されたコマンド列「0，　100」が出力され，キー入力待ちとなった場合を示しています。

(3)　プロンプトがRUN＞からSTP＞，HLT＞，DMA＞，RES＞へ一度変化するとその状態が解除されても自動的にプロンプトが元に戻りません。この場合，次に＄＄が入力された時点でプロンプトが変化します。

保守／廃止

５.２　コマンド行書式

コマンドは以下の形式で入力します。

```
×××>　コマンド$ $
```

×××>の部分はプロンプトと呼び，IE-17K-ETの動作を示します。

コマンドの実行は，プロンプトに続いてコマンドを入力し，最後に ESC キーまたは $ キーを２回入力します。(ESC キーを入力した場合，“$”がエコーバック表示されます)。

コマンドのあとに入力する連続した２つの$記号はコマンドの終了を意味し，ターミネータと呼びます。ターミネータが入力されると，IE-17K-ETはそのコマンドの実行を開始します。

また，コマンドは以下のようにデリミタ（$）で区切ることにより連続していくつでも入力することができます。

×××>コマンド１$コマンド２$……$コマンドn$ $

コマンドには区切り記号としてデリミタ (Delimiter) が必要なものと，そうでないものがあります。コマンドのデリミタには，$ (ESCコード) が使用されます。

特にデリミタの必要がないコマンドのあとに入力された１つの$はコマンドの実行に何の影響も与えません。

このためデリミタの必要のないコマンドを複数記述する場合，そのコマンド列の読みやすさのためにコマンド間に$を入れるというような使い方ができます。

コマンド列の実行は連続する２つの$の入力により行われます。つまり，コマンド行は連続する２つの$により終端され，かつ，実行が開始されます。

コマンド列は終端される以前ならば修正することが可能です。

おのおののコマンドには，アーギュメントを持つものと，持たないものがあります。

次にコマンドの基本書式を示します。

① ＜数値アーギュメント＞＜コマンド名＞

② ＜数値アーギュメント＞＜コマンド名＞＜Qレジスタ識別子＞

③ ＜コマンド名＞＜Qレジスタ識別子＞＜文字列＞$

④ ＜コマンド名＞<Qレジスタ識別子＞

⑤ ＜コマンド名＞

⑥ ＜数値アーギュメント＞＜コマンド名＞＜文字列＞$

また，すべてのコマンドにおいて数値アーギュメントを記述する個所には，式を記述できます。

保守／廃止

５.３　コマンド・バッファ

CLICE-ETには，コンソールから入力されたコマンドを格納するコマンド・バッファがあります。

コマンド・バッファには，ほとんどの場合，入力された文字がそのまま格納されます。

ただし，次の場合には入力された文字がそのまま格納されません。

① 特殊制御文字

（５.４.１　特殊制御文字参照）

② プロンプト直後に入力された＊および？

（付録A.５.９　そ の 他　＊コマンド（Qレジスタへの代入），？コマンド（エラー表示）参照）

また，コマンドの実行が正常に終了しプロンプトが表示されると，コマンド・バッファはクリアされます。

保守／廃止

5.4　文字セット

CLICE-ETでは，ASCII文字セットが使用できます。

コントロールA（↑A）からコントロール_（↑^）までの制御文字（ASCIIコードの00から1FH）は，^と英字で表示されます。

また，CLICE-ETでは^とAの2文字を続けて入力した場合にもコントロールA（↑A）が入力されたものとして解釈します。

（例）コントロール・キーを押しながらAキーを同時に押すことをコントロールA（↑Aと略す）と呼びます。このとき，CLICE-ETにはASCIIコードの01Hが入力され，

^A

というように^（ASCIIコードの5EH）とA（ASCIIコードの41H）の2文字が表示されます。

5.4.1　特殊制御文字

CLICE-ETには，次のような特殊制御文字があります。

(1) DEL

DEL（ASCIIコードの7FH）キーを入力すると，直前に入力された1文字分の文字を消去します。

直前の文字が制御文字の場合には，^と英字の2文字を消去します。

(2) ESC

ESC（ASCIIコードの1BH）キーが入力されると，$（ASCIIコードの24H）が表示されます。

コマンド・バッファには1BHが格納されます。

(3) コントロールC

コントロールC(↑C)(ASCIIコードの03H)はコマンドの実行を中断するための特殊制御文字で，^とCの2文字がこの順序で表示されます。

コマンド列が終端される以前に↑Cを1回入力するとそこまで入力したコマンド列がすべて無効となり，プロンプトを表示しコマンド入力待ちの状態になります。

しかし，この場合はコマンド・バッファ内のデータはクリアされません（**付録A.5.9　その他＊コマンド（Qレジスタの代入）**参照）。

コマンドを終端したあと(すなわち，コマンドの実行を開始したあと)，その実行を中断するには↑Cを続けて2回入力します。

コマンド列の実行が中断されると，次のメッセージが表示されます。

ABORTED！

このとき，中断されたコマンド列はコマンド・バッファ内に保存されます。

保守／廃止

(4)　**コントロールX**

コントロールX(↑X)（ASCIIコードの18H)キーは，そのときにカーサが位置する１行が消去されます。

また，↑Xにより消去されたコマンド列は，コマンド・バッファに保存されません。

(5)　**コントロールH**

コントロールH（↑H）またはBSキー（ASCIIコードの08H）は，DELと同一の動作を行います。

(6)　**コントロールU**

コントロールU（↑U）（ASCIIコードの15H）は，機能としては↑Xと同じですが，^Uと改行が表示されます。

↑Uは，CRT以外のTTYタイプのコンソールのためのコマンド消去用制御文字です。

(7)　**コントロールG**

コントロールG（↑G）（ASCIIコードの07H）が入力されると現在行の内容が表示されます。

↑Gは↑Uと同様，TTYタイプのコンソールのための制御文字です。

(8)　**コントロールE**

コントロールE（↑E）（ASCIIコードの05H）は，コマンド列が終端される以前に入力されると，エディット・モードに入ります（**付録A.６　エディタ**参照)。

(9)　**コントロールS**

コントロールS(↑S)（ASCIIコードの13H)が入力されると，そのとき実行中のコマンドの文字表示を一時中断します。

実行の再開はコントロールQ（↑Q）（ASCIIコードの11H）を入力することで行われます。

保守／廃止

５.４.２　特殊文字

CLICE-ETには，次のような特殊文字があります。

(1)　^

^（ASCIIコードの5EH）は必ず^に続く１つの文字とともに使用されます。

^に続く文字は，制御文字と同じ文字をコマンド・バッファへ格納します。

（例）　^に続いて，Bをコンソールから入力すると，↑Bを入力したときと同じコード(ASCIIコードの02H）がコマンド・バッファへ格納されます。

(2)　$

$(ASCIIコードの24H）が入力されると，ESC（ASCIIコードの1BH）がコマンド・バッファへ格納されます。

(3)　コントロールR

コントロールR(↑R)（ASCIIコードの12H)は，↑Rに続く１文字をそのままコマンド・バッファへ格納します。

特殊制御文字や制御文字などのそれだけで動作をしてしまう文字をコマンド・バッファへ格納したい場合に使用します。

（例）　^R↑R…↑R（ASCIIコードの12H）が１文字格納されます。
^R^R…^（ASCIIコードの5EH)とR(ASCIIコードの52H)の２文字格納されます。
↑R↑R…↑R（ASCIIコードの12H）が１文字格納されます。
↑R^R…^（ASCIIコードの5EH)とR(ASCIIコードの52H)の２文字格納されます。

ただし↑CおよびDELは↑Rの適用を受けません，これらの文字はエディタ・コマンドでのみ格納することができます。

(4)　その他の特殊文字

特殊文字には，このほかに，次のものがあります。

コントロールB（↑B)，コントロールD（↑D)，コントロールP（↑P),.（ピリオド）

このうち↑B，↑D，↑Pは，後述する定数の型を示すためのプリフィックスです。

.（ピリオド）は，組み込みコマンドを示すためのプリフィックスです。

保守／廃止

5.4.3　ダミー文字

ダミー文字とは，それ自身はコマンドとして定義されておらず，実行されてもエラーにはならない文字です。

ダミー文字の中にはそれがコンソールへ出力された場合に特別の動作をするものがあります。

このためダミー文字はコマンド列の読みやすさのためにコマンド列に挿入することができます。

(1) CR

CR（↑M）（ASCIIコードの0DH）がコンソールへ出力される場合には，改行が行われます。

(2) LF

LF（↑J）（ASCIIコードの0AH）はこのままではコマンド・バッファには格納されません。

LFをコマンド・バッファに格納するには↑Rのあとに↑Jを入力してください。

(3) 空　白

空白（ASCIIコードの20H）は，それがコンソールへ出力されると単に１文字の空白が表示されます。

(4) NULL

NULL（↑@）（ASCIIコードの00H）はコマンド・バッファには格納されません。

NULLをコマンド・バッファに格納するには↑Rに続いて↑@を入力してください。

(5) TAB

TAB（↑I）（ASCIIコードの09H）は，８カラムの空白を表示します。

コマンド・バッファには↑I（ASCIIコードの09H）が格納されます。

保守／廃止

5.5　式

CLICE-ETで式に使用できる演算子を次に示します。

＋…＋符号または加算記号

－…－符号または減算記号

＊…乗算記号

／…除算記号

^^…剰余記号

&…論理積記号

＃…論理和記号

｜…排他的論理和記号

～…否定記号

｛…左シフト記号

｝…右シフト記号

式の演算に優先順序はなく，式の評価はすべて左から右の方向に行われます。

ただし，優先順位を変更したい場合には（ ）が使用できます。

式の各項の構成要素は定数，変数，関数が使用できます。

（例）　次に示す例の評価値（→の右側）はすべて10進数で示されています。

①×××>3+4	→	7
②×××>10+↑D10＊↑B10	→	52
③×××>10+（↑D10＊↑B10）	→	36
④×××>↑B1010｝1	→	5
⑤×××>↑B1011｝1&↑B1000	→	8
⑥×××>～FFFFFFFC	→	3
⑦×××>5^^3+（5／3）	→	3
⑧×××>5^^3+5／3	→	2
⑨×××>Q1+@↑FDTM↑V	→Qレジスタ１とデータ・メモリの先頭番地に割り当てられたアレイの内容(つまりデータ・メモリの０番地の内容）の和	
⑩×××>_↑FPRM+（100＊2）↑V	→プログラム・メモリの100H番地の内容（16ビット）	

保守／廃止

5.6 定　　数

CLICE-ETでは，定数の表現として16進，10進，２進の整数および17Kシリーズの命令表現形式(1-4-3-4-4ビット形式）が使用できます。

その表現できる数値の範囲は，

10進数：-2^{31}～$(+2^{31}-1)$

16進数：0　～　FFFFFFFFH

の範囲です。また，負の表現には２の補数表現を使用します。

定数の表現方法を次に示します。

２進数定数：２進数の前に↑Bを付けて表します。

（例）　↑B1010は10進の10を意味します。

10進定数：10進数の前に↑Dを付けて表します。

（例）　↑D324は10進の324を表します。

16進定数：16進数のみで表します。

（例）　F1は10進の241を表します。

1-4-3-4-4ビット形式定数：16進数の前に↑Pを付けて表します。

（例）　↑P074F0はNOPを表します。

保守／廃止

５.７　変　　数

CLICE-ETには，変数の概念として，アレイ（Array）とQレジスタ（Q-Register）があります。

５.７.１　ア レ イ

アレイとは，CLICE-ETが管理するすべての資産を表し，このアレイの各要素がターゲット・デバイスの各資源に対応しています。

したがって，CLICE-ETでアレイに値を代入するということは，そのアレイの要素に対応する，ターゲット・デバイスに内蔵されるハードウエアにデータを書き込むことになります。

また，アレイのある要素の内容を参照することは，そのアレイの要素に対応するハードウエアのデータ（状態）を読み出すことになります。

アレイの各要素はポインタにより参照され，XB，XC，XWコマンドによりデータが代入できます。

５.７.２　Qレジスタ

CLICE-ETには，数値または文字列が格納できる，Qレジスタと呼ばれるレジスタがあります。これが一般的なプログラミング言語の変数の概念に相当します。

Qレジスタは，文字Qに続く１文字の英大文字または数字により表記されます（この英大文字をQレジスタ識別子と呼ぶ）。

（例）　Q1，QA，Q8，QZなど

すべてのQレジスタは，数値変数または文字変数のいずれかとして使用できます。

どちらの変数として使用されるかは，そのQレジスタに格納した内容によって決まります。

CLICE-ETの起動時には，すべてのQレジスタの内容は空です。

Qレジスタに格納できる数値の範囲は定数の値の範囲と同じです。Qレジスタに数値を代入した場合，その値を式で使用することができます。

Qレジスタに文字列を格納するコマンドとして，Uコマンド，＊コマンドなどがあります。Qレジスタに文字列を格納した場合，その文字列をマクロ・コマンドとして実行することができます。

保守／廃止

5.8　組み込みマクロ・コマンド

この節では，IE-17K-ETの基本的な機能を使う場合に使用する，組み込みマクロ・コマンドについて説明します。

この節で説明する書式中に使用されている記号は以下の意味を持ちます。

↵	；改行入力です。
{	}	；{　} の中に記述されている文字列のいずれかを選択することを意味しています。
[　]	；入力省略可能を意味します。
_____ (アンダバー)	；コンソール入力を表します。

5

5.8.1　プログラム・メモリ制御コマンド

各コマンドは，実際の手順に従って並べてあります。

(1) プログラム・メモリのロード
.LP (Load Program Memory)

(2) プログラム・メモリのベリファイ
.VP (Verify Program Memory)

(3) プログラム・メモリの初期化
.IP (Initialize Program Memory)

(4) プログラム・メモリの変更
.CP (Change Program Memory)

(5) アセンブル・コマンド
.AP (Assemble Program)

(6) プログラム・メモリのダンプ
.DP (Dump Program Memory)

(7) 逆アセンブル・コマンド
.UP

(8) プログラム・メモリの検索
.FP (Find Program Memory)

(9) プログラム・メモリのセーブ
.SP (Save Program Memory)

(10) PROMデータの出力
.XS (Save PROM Data)

(11) IE-17K-ETの再起動
.Q

保守／廃止

.LP0　.LP1　プログラム・メモリのロード（Load Program Memory）

書　式　：　{.LP0 / .LP1}
RS-232-C　チャネル0：LP0 チャネル1：LP1

〔機　能〕　AS17KのICEファイルの内容を.LP0または.LP1で指定したRS-232-Cのチャネルから入力します。

（例）　チャネル０から，プログラムをロードします。

```
@@@>.LP0$$
```

〔注　意〕
- 電源投入時あるいはIE-17K-ETのリセットを行った場合（プロンプトが＠＠＠＞）は，.LPによりAS17KのICEファイルをロードしてください。
- このコマンドでロードされるプログラムがプログラム・メモリの一部しか占めない場合は，プログラムがロードされていないプログラム・メモリに以前のプログラムが残っています。
- プログラム・カバレージは，クリアされます。

保守／廃止

.VP0　.VP1　プログラム・メモリのベリファイ(Verify Program Memory)

書　式　：　{.VP0 / .VP1}
RS-232-C　チャネル0：VP0 チャネル1：VP1

5

〔機　能〕　プログラム・メモリの内容と.VP0または.VP1で指定したRS-232-Cのチャネルから送られてくるAS17KのICEファイルのデータをベリファイします。

なお，ベリファイすると，ユーザ・プログラム領域外で必ずVerify...NGとなります。これは，IE-17K-ETがICEファイルを格納するメモリ上において，アセンブル環境情報領域の一部とSEボード環境情報領域のデータを加工するためです。

アセンブル環境情報領域ならびにSEボード環境情報領域については，各デバイス・ファイルのユーザーズ・マニュアルの**第５章　ロード・モジュール・ファイルのフォーマット**を参照し，エラーの発生した番地がユーザ・プログラム領域外であることを確認してください。

(例)

```
BRK>.VP0$$
Verify...NG
00
BRK>0000000000303032313130 11
:1007C40038303731441212FF073356202020F0F6
:1007D40000000000••••••••••••••••
:0407FC000100C11522
:00000001FF
```

〔注　意〕　・データ・メモリ情報が異なる場合は，“Verify NG DATA INITIAL VALUE”を表示します。

・EPAが異なる場合は“Verify NG EPA”を表示します。

・IFL，DFLが異なる場合は“Verify NG IFL DFL”を表示します。

保守／廃止

.IP　プログラム・メモリの初期化（Initialize Program Memory）

書　式　：[α], β, γ.IP
α：スタート・アドレス β：エンド・アドレス （α≦βであること，α＞βではエラー） γ：初期化するデータ（1-4-3-4-4ビット形式）

〔機　能〕　α番地からβ番地のプログラム・メモリ内容をγにします。
αが０の場合は，αを省略できます。

（例１）　10H番地から20番地の内容を074F0にします。

```
BRK>10,20,074F0.IP$$
```

（例２）　０H番地から20番地の内容を120FFにします。

```
BRK>,20,120FF.IP$$
```

保守／廃止

.CP　プログラム・メモリの変更（Change Program Memory）

書　式　：［α］.CP
α：変更するプログラム・メモリ・アドレス

〔機　能〕　α番地のプログラム・メモリ内容を変更します。
αが０の場合，αを省略できます。

（例）　100番地からプログラム内容を変更します。

```
BRK>100.CP$$
  0100:074F0-120F5  074F0-14001  074F0-11000  074F0-06100
  0104:074F0-↵
```

↳ ↵の代わりに $$でも可

14344形式の数値を５桁入力すると，カーサは自動的に次の番地に移ります。
終了するときは，数値入力をせず“↵”または“＄＄”で終了します。

```
BRK>100.CP$$
  0100:074F0-SPC                    スペース・キーを入力する

  0100:074F0-074F0 074F0-■          次の番地の変更入力待ちになる
```

数値入力の代わりにスペース･キーを入力すると,プログラム内容を変更せずカーサは次の番地に移ります。
もし，数値入力を間違えた場合“DEL”キーを入力すると修正することができます（“BS”キーも同様)。

```
0100:120AF-120A1 074F0-120_    ” DEL ” ｷｰを押す。
          ↓
0100:120AF-120A1 074F0-12_     ” DEL ” ｷｰを押す。
          ↓
```

0100:120AF-120A1 074F0-1_　　”DEL”ｷｰを押す。

↓

0100:120AF-120AF 074F0-_　　”DEL”ｷｰを押す。

↓

0100:120AF-_　　”DEL”ｷｰを押す。

↓

00FF:120C1-_

備考　__：カーサ

保守／廃止

.AP　アセンブル・コマンド

書　式　：　[α].AP [β]
α：スタート・アドレス β：Qレジスタ名

〔機　能〕　Qレジスタβにかかれたニモニックをアセンブルしてプログラム・メモリ・アドレスαから格納します。

αが０の場合，αを省略できます。

Qレジスタβを省略した場合，プログラム・メモリ・アドレスαのコードを逆アセンブルして表示し，ニモニック入力モードになります。このニモニック入力モードでは，プログラム・メモリの内容をニモニック・レベルで変更することができます。

（例１）

```
BRK>5.AP$$
    0005: MOV 05,#5 - $$                    $$の入力でニモニック入力モードから抜ける
BRK>
```

```
BRK>.AP$$
    0000: MOV 00,#1 - MOV 01,#1↵            ↵の入力でアセンブルを行い，次のアドレスへ
    0001: MOV 10,#2 - ■                     （ＢＳキー使用可）
                      └→ 入力待ち
```

```
BRK>.AP$$
    0000: MOV 00,#1 - MOV 01,#1↵            アセンブル時エラーが発生すると，元のアドレスに戻
    ASSEMBLE ERROR                          り，入力待ちとなる
    0000: MOV 00,#1 - ■
                      └→ 入力待ち
```

```
BRK>.AP$$
    0000: MOV 00,#1 - ↵                ↵のみ入力すると次のアドレスの
    0001: MOV 10,#2 - ■                入力待ちとなる
                  └→ 入力待ち

BRK>.AP$$
    0000: MOV 00,#1 - MOV 01,#1
    0001: MOV 10,#2 - MOV 01,#1■       入力待ち状態で，↑Ｐを入力すると一回前に
                                       入力した文字列（ここでは MOV 01,#1）が表示される
                  └→ ↑Ｐ（CTRL＋Ｐ）
```

（例２）

```
BRK>USMOV 01,#05                   Ｕコマンドで Ｑレジスタ Ｓに文字列を代入する
ADD 1,78
$$
BRK>5.APS$$                        Ｑレジスタ Ｓの内容をアセンブルする
BRK>5,6.DP$$                       アセンブルした結果をダンプする

0005:  1D015    00781
```

〔注　意〕 (1) RFのアドレスは40H～BFHで記述してください。ただし，00H～3FHと書いた場合，80H～BFHと同様に扱われます。また，C0H～FFHは40H～7FHとして扱われます。

(2) ニモニックとオペランドのセパレータはスペースかタブを使用してください。

(3) アドレスは16進数で記述してください。

(4) イミーディエト・データは先頭に＃を付けた16進数で記述してください。

（例）

```
: MOV   11,#0        STOP  0
  ADD   0,11         BR    005F
  POKE  81,WR        BR    @AR
  PUT   01,DBF       MOV   @5,00
```

(5) アセンブルの途中でエラーが発生した場合は，エラー行とエラー行の内容を表示してアセンブルを中止します。このとき，プログラム・メモリにはエラー行の前の行までのコードが格納されます。なお，オペランド範囲のチェックは行いません。

(6) アセンブルする内容がプログラム・メモリの最終アドレスを越えた場合には０番地に戻って格納するので注意してください。

EPAを使用するには8000H番地以降のアドレスを指定してください。

保守／廃止

（例）

：　8000.APD$$　　ＥＰＡにＱレジスタＤの内容をアセンブルして格納する

保守／廃止

.DP　プログラム・メモリのダンプ（Dump Program Memory）

書　式　：[α] [, β].DP
α：スタート・アドレス β：エンド・アドレス　　（α≦βであること，α＞βは，エラー）

〔機　能〕　α番地からβ番地のプログラム内容をダンプします。

αが０の場合，αを省略できます。“，β”が省略されるとエンド･アドレスはα＋3FHになります。

（例１）　10H番地から20H番地までの内容を1-4-3-4-4ビット形式でダンプします。

```
BRK>10,20.DP$$

0010：074F0 074F0 074F0 074F0 074F0 074F0 074F0 074F0
0018：074F0 074F0 074F0 074F0 074F0 074F0 074F0 074F0
0020：074F0
```

（例２）　０番地から10H番地の内容をダンプします。

```
BRK>,10.DP$$

0000：074F0 074F0 074F0 074F0 074F0 074F0 074F0 074F0
0008：074F0 074F0 074F0 074F0 074F0 074F0 074F0 074F0
0010：074F0
```

保守／廃止

（例３）　10H番地以降の内容をダンプします（10Hから10H＋3FH番地をダンプします）。

```
BRK>10.DP$$

0010:1D790 1D7D0 1D7E0 074F0 074F0 074F0 074F0 167E0
0018:1D770 08770 10771 08771 10771 08772 10771 08773
0020:10771 08774 10771 08775 10771 08776 10771 08777
0028:10771 08778 10771 08779 10771 0877A 10771 0877B
0030:10771 0877C 10771 0877D 10771 0877E 10771 0877F
0038:1D000 074F0 074F0 074F0 074F0 074F0 074F0 074F0
0040:1D7F0 00000 074F0 074F0 0B7D0 097E0 0C049 1C146
0048:0C050 097F2 0C170 09000 0C171 18770 09770 0C172
```

保守／廃止

.UP　逆アセンブル・コマンド

書　式　：　[α] [, β]. UP [γ]
α：スタート・アドレス β：エンド・アドレス（α≦βであること，α＞βはエラー） γ：Qレジスタ名

〔機　能〕　γを省略した場合，プログラム・メモリ・アドレスαからβまでの内容を逆アセンブル表示します。このとき，EPA情報も出力します。

γを指定した場合，プログラム・メモリ・アドレスαからβまでの内容を逆アセンブルしてQレジスタγにニモニックを格納します。

αが０の場合αを省略できます。“，β”を省略するとエンド･アドレスはα＋10になります。

（例１）

スタート・アドレス，エンド・アドレス共に省略した場合

```
BRK>.UP$$
EPA  ADDR  CODE   MNEMONIC
     0000  070E0  RET
     0001  007F0  ADD   0,7F
     0002  002A5  ADD   5,2A
     0003  00558  ADD   8,55
     0004  0000F  ADD   F,00
     0005  1000F  ADD   00,#F
     0006  102A5  ADD   2A,#5
     0007  1055A  ADD   55,#A
     0008  107F0  ADD   7F,#0
     0009  027F0  ADDC  0,7F
```

スタート・アドレスのみ指定した場合

```
BRK>8.UP$$
EPA  ADDR  CODE   MNEMONIC
     0008  007F0  ADD   0,7F
     0009  002A5  ADD   5,2A
     000A  00558  ADD   8,55
     000B  0000F  ADD   F,00
     000C  1000F  ADD   00,#F
     000D  102A5  ADD   2A,#5
     000E  1055A  ADD   55,#A
     000F  107F0  ADD   7F,#0
     000E  1055A  ADD   55,#A
     000F  107F0  ADD   7F,#0
```

保守／廃止

```
BRK>44,4B.UP$$
EPA  ADDR  CODE   MNEMONIC
     0044  057F0  XOR   0,7F
 1   0045  052A5  XOR   5,2A
 1   0046  05558  XOR   8,55
 1   0047  0500F  XOR   F,00
     0048  1500F  XOR   00,#F
     0049  152A5  XOR   2A,#5
     004A  1555A  XOR   55,#A
     004B  157F0  XOR   7F,#0
```

スタート・アドレス，エンド・アドレス共に指定した場合

（例２）

```
BRK>10,20.UPB$$
BRK>50.APB$$
```

プログラム・メモリ・アドレス１０Ｈから２０Ｈまでの内容を逆アセンブルし，結果をＱレジスタＢにいれる

ＱレジスタＢの内容をアセンブルしてプログラム・メモリ・アドレス５０Ｈ以降に展開する

（例３）

```
BRK>10,20.UPB$$
BRK>.EDB$$
  >
      •
      •
（編集例は省略）
      •
      •
  >
BRK>10.APB$$
```

プログラム・メモリ・アドレス１０Ｈから２０Ｈまでの内容を逆アセンブルし，ＱレジスタＢにいれる

エディット・コマンドで編集する

ＱレジスタＢの内容をアセンブルしてプログラム・メモリ・アドレス１０Ｈ以降に展開する

〔注　意〕　逆アセンブルできないコードはニモニック欄に「DW」として表示されます。

保守／廃止

.FP　プログラム・メモリの検索（Find Program Memory）

書　式　：[α], β, γ [, δ].FP
α：スタート・アドレス　　δ：マスク・データ β：エンド・アドレス γ：検索したいデータ　　（γ, δは1-4-3-4-4ビット形式）

〔機　能〕　α番地からβ番地のプログラム・メモリ内でδでマスクされたγの内容を検索します。
αが０の場合，αは省略できます。
δが省略されると，マスク・データは1F7FFになります。

（例）　０番地から300H番地内に12×××があるかどうか検索します。

```
BRK>0,300,12000,1F000.FP$$

  0110:12120   0120:12200     0140:12240     0152:12250
  0160:12151   0180:12152
```

〔注　意〕　マスク・データとは検索したいビットに１を設定し，１でも０でもかまわないビットには０を設定した1-4-3-4-4ビット形式のデータです。

保守／廃止

.SP0　.SP1　プログラム・メモリのセーブ（Save Program Memory）

書　式　：　{.SP0 / .SP1}
RS-232-C　チャネル0：SP0 チャネル1：SP1

5

〔機　能〕　プログラム・メモリの内容を.SP0または.SP1で指定したRS-232-Cのチャネルに出力します。出力形式は，AS17KのICEファイル形式と同じです。

（例）プログラム・メモリの内容をチャネル１に出力します。

```
BRK>.SP1$$
:1000000063A03CF061273CF0EFE040423CE0EF04AD
:10001000EF10EF20EF30EF91EF00EF10EF20EF3017
:10002000EF90E8C0E8D0E8D0E8F03CA138A538A6B9
:1000300038A738E0E820E8303CF03CF080219030F0
:10004000F7F4601C38E0B204E8E0E8F038A538E0E6
:10005000E830E82038E08031902038E0F6F4605055
```

保守／廃止

.XSO　.XS1　PROMデータの出力（Save PROM Data）

書　式　：　{.XS0 / .XS1}

RS-232-C　チャネル0：XS0
　　チャネル1：XS1

〔機　能〕　プログラム・メモリの内容を.XS0または.XS1で指定したRS-232-CのチャネルにAS17KのPROMファイル形式で出力します。

（例）プログラム・メモリの内容をチャネル１に出力します。

```
BRK>.XS1$$
:1000000063A03CF061273CF0EFE040423CE0EF04AD
:10001000EF10EF20EF30EF91EF00EF10EF20EF3017
:10002000EF90E8C0E8D0E8E0E8F03CA138A538A6B9
:1000300038A738E0E820E8303CF03CF080219030F0
:10004000F7F4601C38E0B204E8E0E8F038A538E0E6
:10005000E830E82038E08031902038E0F6F4605055
```

保守／廃止

.Q　IE-17K-ETの再起動をする

書　式　：　.Q

〔機　能〕　IE-17K-ETのリセット・スイッチを押したときや，電源を切ったあと再び電源を入れたときに前の状態から再起動します。再起動に成功した場合，前と同じプログラムをロードする必要はありません。

（例）
```
@@@>.Q$$
17XXX
BRK>
```
．Qコマンド入力の2～3秒後，品名が表示される

〔注　意〕　.Qコマンド実行後，２～３秒しても品名表示されない場合はプログラムが失われていますので，IE-17K-ETをリセット後，プログラムを再ロードしてください。

また，品名表示の代わりに

```
Your program must be lost. Please load it again!
```

のメッセージがでた場合もプログラムが失われていますので，IE-17K-ETをリセット後プログラムを再ロードしてください。

なお，プログラムが保存されているか破壊されているかの判断は，プログラムのチェックサムが一致したか否かで判断しているため，たまたま品名表示した場合でもプログラムの一部が失われている場合がありますので注意してください。

IE-17K-ETの電源を切ったあと，プログラムが保存されている時間はSEボードにより異なります。

保守／廃止

5.8.2　データ・メモリ制御コマンド

(1)　データ・メモリの初期化

.ID (Initialize Data Memory)

(2)　データ・メモリの変更

.CD (Change Data Memory)

(3)　データ・メモリのダンプ

.DD (Dump Data Memory)

(4)　すべてのデータ・メモリのダンプ

.D (Dump All Data Memory)

保守／廃止

.ID　データ・メモリの初期化（Initialize Data Memory）

書　式　：［α］，β，γ.ID
α：スタート・アドレス β：エンド・アドレス　　（$\alpha \leqq \beta$であること，$\alpha > \beta$は，エラー） γ：内容

〔機　能〕　α番地からβ番地のデータ・メモリ内容をγにします。
αが０の場合，αを省略できます。

（例１）　10H番地から20H番地の内容を０にします。

```
BRK>10,20,0.ID$$
```

（例２）　０番地から20H番地の内容を１にします。

```
BRK>,20,1.ID$$
```

保守／廃止

.CD　データ・メモリの変更（Change Data Memory）

書　式　：[α].CD
α：変更するデータ・メモリ・アドレス

〔機　能〕　α番地のデータ・メモリ内容を変更します。
αが０の場合，省略できます。

（例）　０番地からデータ内容を変更します。

```
BRK>.CD$$
  0000  0-0  1-0  2-0  3-0  4-0  5-0  6-0  7-0
  0008  8-0  9-↵
                  └→ ↵の代わりに $$でも可
```

データを１つ入力すると，カーサが次の番地に自動的に移ります。終了するときは数値入力をせず，“↓”または“＄＄”を入力してください。

```
BRK>100.CD$$
  0100  3-SPC                    スペース・キーを入力する

  0100  3-3  2-■                 次の番地の変更入力待ちになる
```

数値入力のかわりにスペース・キーを入力すると，データ内容を変更せず，次の番地にカーサが移ります。
もし，数値入力を間違った場合，“DEL”キーを入力すると修正することができます（“BS”キーでも同様）。

保守／廃止

```
0010:2-3  4-5  6-_   “ＤＥＬ”キーを押す。
  ↓
0010:2-3  5-_        “ＤＥＬ”キーを押す。
  ↓
0010:3-_             “ＤＥＬ”キーを押す。
  ↓
000F:4-_
```

注)　__：カーサ

保守／廃止

.DD　データ・メモリのダンプ（Dump Data Memory）

書　式　：[α] [, β].DD
α：スタート・アドレス β：エンド・アドレス　　（α≦βであること，α＞βは，エラー）

〔機　能〕　α番地からβ番地のデータ・メモリ内容をダンプします。
αが0の場合，αを省略できます。

（例1）　0番地から80H番地までのデータ・メモリ内容をダンプします。

```
BRK>0,80.DD$$

0000:0 1 2 3 4 5 6 7  8 9 A B C D E F
0010:0 1 2 3 4 5 6 7  8 9 A B C D E F
0020:0 1 2 3 4 5 6 7  8 9 A B C D E F
0030:0 1 2 3 4 5 6 7  8 9 A B C D E F
0040:0 1 2 3 4 5 6 7  8 9 A B C D E F
0050:0 1 2 3 4 5 6 7  8 9 A B C D E F
0060:0 1 2 3 4 5 6 7  8 9 A B C D E F
0070:0 1 2 3 4 5 6 7  8 9 A B C D E F

0080:0
```

（例2）　30H番地の内容をダンプします（30Hから7FH番地の内容をダンプします）。

```
BRK>30.DD$$

0030:0 0 0 3 0 5 0 7  8 9 0 1 4 6 0 F
0040:0 1 2 3 4 5 6 0  8 3 A B C D E F
0050:0 0 2 D F F F 0  4 9 A 0 C 0 0 F
0060:0 1 2 3 4 5 6 0  8 9 A 0 0 D 0 F
0070:0 C 2 B 4 5 6 0  8 7 9 B C 0 0 0
```

〔注　意〕　・“, β”を省略した場合は，α番地からα番地が割り当てられているバンクの最終番地までのデータをダンプします。
レジスタ・ファイルのα番地を指定した場合は，同様にα番地からレジスタ・ファイルの最後までをダンプします。
・実装されていないデータ・メモリの内容は“—”で表示されます。
・0080番地から00BF番地のダンプは，レジスタ・ファイルの内容のダンプです。レジスタ・ファイルが実装されていない場合は，内部バスの状態が表示されます。

保守／廃止

.D　すべてのデータ・メモリのダンプ（Dump All Data Memory）

書　式　：　.D

〔機　能〕　　すべてのデータ・メモリ内容をダンプします。

（例）　BRK>.D$$

```
0000:0 0 1 C 6 0 0 0  0 0 0 0 0 0 0 0
0010:0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0020:0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0030:0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0040:0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0050:0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0060:4 0 8 0 0 0 0 0  0 0 0 0 0 0 0 0
0070:5 0 F 0 0 0 0 0  1 0 0 7 0 0 C 0

0080:2 5 2 3 4 5 6 1  7 A A B C D E 1
0090:0 0 0 3 4 0 6 0  0 9 A B C D E 0
00A0:0 1 2 3 4 5 6 1  0 9 A B C D E 0
00B0:0 0 2 3 4 7 7 F  0 0 A B C D E 2
```

〔注　意〕　　レジスタ・ファイルもダンプされます。

保守／廃止

５.８.３　周辺回路制御コマンド

(1)　周辺レジスタの内容表示

　　. GD (Get & Display)

(2)　周辺レジスタの内容読み出し

　　. GE (GET)

(3)　周辺レジスタへの書き込み

　　. PD (Put Direct)

(4)　周辺レジスタへの間接書き込み

　　. PU (PUT)

保守／廃止

．GD　ゲット＆ディスプレイ・コマンド

書　式　：　α，GD
α：周辺アドレス

〔機　能〕　　周辺アドレスαの内容をHEXで表示します。

（例１）

```
BRK>1.GD$$                    周辺アドレス１の内容を表示する。
00000002
```

（例２）

```
BRK>UP1.GD^A TIME1 ^A2.GD^A TIME2 ^A3.GD^A SIO ^A$$      マクロＰを定義する
BRK>MP$$                                                 マクロＰを実行する
00000034 TIME1
00000022 TIME2
00000004 SIO
```

〔注　意〕　　周辺アドレスαがその品種にない値のときは

```
?POS INVALID ADDRESS
```

を表示します。

保守／廃止

．GE　ゲット・コマンド

書　式　：　α．GEβ
α：周辺アドレス β：Qレジスタ名

〔機　能〕　周辺アドレスαの値をQレジスタβに数値で代入します。

（例１）

```
BRK>1.GEA$$          周辺アドレス１の内容をＱレジスタＡに代入する。
BRK>QA=H$$IO         ＱレジスタＡの内容をＨＥＸで表示する。
```

（例２）

```
BRK>UQ1.GEA2.GEB3.GEC^A SIO= ^AQA=H^A TM1= ^AQB=H^A TM2= ^AQC=H$$     マクロＱを定義する
BRK>MQ$$ SIO= 34 TM1= 66 TM2= 2                                      マクロＱを実行する
BRK>
```

〔注　意〕　例２で定義しているマクロQにおいて，「^A」は文字列をそのまま表示させるための命令です。マクロ定義したコマンドの実行時に表示させたい文字列がある場合，その文字列を「^A」で囲んでください。

周辺アドレスαがその品種にない値のときは

```
?POS INVALID ADDRESS
```

を表示します。

保守／廃止

. PD　プット・ダイレクト・コマンド

書　式　：　α, β. PD
α：周辺アドレス β：データ

〔機　能〕　周辺アドレスαに数値βを代入します。

（例）　BRK>1,55.PD$$　　周辺アドレス１に５５Ｈを代入する。

〔注　意〕　周辺アドレスαがその品種にない値のときは

?POS INVALID ADDRESS

を表示します。

保守／廃止

．PU　プット・コマンド

書　式　：　α．PUβ
α：周辺アドレス β：Qレジスタ名

〔機　能〕　周辺アドレスαにQレジスタβの内容（数値）を代入します。

（例）　BRK>55UA$$　　Qレジスタ A に５５Ｈを代入する。
　　BRK>1.PUA$$　　周辺アドレス１にQレジスタ A の値を代入する。

〔注　意〕　周辺アドレスαがその品種にない値のときは

?POS INVALID ADDRESS

を表示します。

保守／廃止

5.8.4　エミュレーション・コマンド

(1)　リセット

.R (Reset)

(2)　プログラムの実行

.RN (Run)

(3)　プログラムの実行（コンディションをリセット）

.BG (Run Beginning Condition)

(4)　ブレーク

.BK (Break)

(5)　プログラム・スタート・アドレスの変更

.CA (Change Start Address)

(6)　ステップ動作

.S (Step)

(7)　ディスプレイ

.DS (Display)

保守／廃止

.R　リセット（Reset）

書　式　：　.R

〔機　能〕　SEボードのリセットを行います。

（例）　BRK>.R$$

〔注　意〕　・レジスタ・ファイル，データ・メモリの内容は対象製品のリセット状態の内容と同じになります。
・データ・カバレージの内容はクリアされます。
・実行開始アドレスは0H番地になります。

保守／廃止

.RN　プログラムの実行（Run）

書　式　：　.RN

〔機　能〕　現在，指定されているプログラムの実行開始アドレスからプログラムを実行します。
ブレーク，トレースに使用するコンディションは変化しません。

（例）
```
BRK>.RN$$
RUN>
```

保守／廃止

.BG　プログラムの実行（Run Beginning Condition）

書　式　：　.BG

〔機　能〕　現在，指定されているプログラムの実行開始アドレスからプログラムを実行します。
ただし，ブレーク，トレースに使用する以下のコンディションはリセットされます。

＜リセットされる内容＞

・レベル１に使用するカウンタ（リセット値は０）
・レベル２に使用するシーケンシャル・スタック（初期値に）
・トレース・オン，トレース・ワン・ショット，トレース・オフの指定（すべてトレース状態に）
・レベル１の条件

（例）
```
BRK>.BG$$
RUN>
```

保守／廃止

.BK　ブレーク（Break）

書　式　：　.BK

〔機　能〕　プログラムの実行を停止させます。

このときシステム・レジスタ，ジェネラル・レジスタの内容を表示します。

ブレーク状態でも受け付け可能です。


```
(例) RUN>.BK$$
        ADDR INSTRUCTION
        0002 074F0     BREAK    ·····  ブレーク処理を行った命令
        0003 074F0     OVERRUN  ·····  最後に実行した命令
        0004 0C004     NEXT     ·····  次に実行する命令
         PC  SP  AR  WR  BR  MP  IX      ┐
        0004  3 0700  0   0  *** 000      │ システム・レジスタ
        PSW: DB  CP  CY   Z  IXE MPE JG   │
             0   0   0   0   0   0   0   ┘
         RP  0123456789ABCDEF
        000  0000000000000320 ·····  ジェネラル・レジスタ
```

保守／廃止

.CA　プログラム・スタート・アドレスの変更（Change Start Address）

書　式　：　α.CA
α：実行開始アドレス

〔機　能〕　　プログラムの実行開始アドレスを変更します。

（例）　プログラムの実行開始アドレスを100H番地に変更します。

```
BRK>100.CA$$
```

保守／廃止

.S　ステップ動作（Step）

書　式　：[α].S
α：回数

〔機　能〕　　指定された回数プログラムを実行します。

（例１）ステップ動作を行います。

```
BRK>.S$$
    BR RP  PC   INST   MNEMONIC
     0 00 0000 074F0  NOP       :SPC        スペース・キー入力で１ステップ進む
     0 00 0001 1D000  MOV 00,#0 :SPC        〃
     0 00 0002 1D011  MOV 01,#1 :SPC        〃
     0 00 0003 1000A  ADD 00,#A :$$         $$入力でステップ動作終了
```

（例２）４回ステップ動作を行います。

```
BRK>4.S$$                                  ステップ数指定
    BR RP  PC   INST   MNEMONIC
     0 00 0000 074F0  NOP       :
     0 00 0001 1D000  MOV 00,#0 :          ┐
     0 00 0002 1D011  MOV 01,#1 :          │ ４ステップ実行
     0 00 0003 1000A  ADD 00,#A :          ┘
     |  |  |  |  |       |
     |  |  |  |  |       命令ｺｰﾄﾞ ﾆｰﾓﾆｯｸ
     |  |  |  |  ﾌﾟﾛｸﾞﾗﾑ･ｶｳﾝﾀ
     |  |  |  ﾚｼﾞｽﾀ･ﾎﾟｲﾝﾀ
     |  |  ﾊﾞﾝｸ
```

〔注　意〕　・ステップ動作は，＄＄あるいは↵（リターン）入力で終了します。

・αが０または省略された場合は，各命令ごとに１ステップずつ動作します。

保守／廃止

.DS　ディスプレイ（Display）

書　式　：　.DS

〔機　能〕　ブレーク中に液晶表示を表示可能にします（LCDコントローラを持っている製品のみ）。

製品によっては，ブレーク中液晶表示が消えるものがあるため，ブレーク中に液晶の表示を見たい場合に使用します。

（例）
```
BRK>.DS$$
DSP>
```

〔注　意〕
- エミュレーション状態としては，RUN状態(BR命令を繰り返し実行)であるため，トレース，カバレージの内容は，このコマンド使用後，保証されません。
- 任意のキー入力により元の状態（ブレーク状態）に戻ります。

保守／廃止

5.8.5　ブレーク／トレース・コンディション制御コマンド

(1)　ブレーク／トレース条件の変更

.CC (Change Break/Trace Condition)

(2)　トレース・オン／オフ条件の変更

.CT (Change Trace Condition)

(3)　ブレーク／トレース条件のダンプ

.DC (Dump Break Condition)

(4)　トレース・テーブルのダンプ

.DT (Dump Trace Table)

(5)　ブレーク／トレース条件のセーブ

.SC (Save Break/Trace Condition)

(6)　ブレーク／トレース条件のロード

.LC (Load Break/Trace Condition)

(7)　ブレーク／トレース条件のベリファイ

.VC (Verify Break/Trace Condition)

保守／廃止

.CC　ブレーク／トレース条件の変更(Change Break/Trace Condition)

書　式　：　.CC

〔機　能〕　ブレーク条件とトレース条件の設定，変更を行います。

〔説　明〕　ブレーク／トレース条件は，同時に独立して４つ設定できます。つまり，４つのユニットと呼ばれる条件設定単位があります。４つのユニット内の項目を設定する場合は，.CCコマンドのレベル１を使用します。各ユニットをブレーク条件として設定する場合は，.CCコマンドのレベル２を使用し各ユニットをトレース条件として設定する場合は，.CTコマンドを使用します。

.CCコマンドは対話方式で条件設定を行います。

次に設定項目を示します。

また，↵（リターン）はC）～L）（ただし，H）は無効）の項目の既定値をそのまま設定する場合に入力し，設定から抜け出すときは＄を入力します。

＜ブレーク条件設定項目（レベル１について)＞

●A）LEVEL (1,2)：？　1

レベルの選択を行います。レベルは１と２の２つです。

●B）UNIT (0-3)：？

ユニットの選択を行います。ユニットは0～3の４つです。各ユニットで設定できる項目については表５-１を参照してください。

CATG (C-L)：？

C）～L）のどの設定項目から条件設定を行うかを指定します。ユニットによっては，設定項目がないものがあります。ない項目を指定した場合は，その項目以降次に存在する項目から設定可能となります。

●C）CONDITION AND(1)/OR(0)　：　既定値　？

D）からK）までの設定項目のANDをとるかORをとるかの選択を行います。

AND条件を選択した場合，ユニット内の全設定項目が満足されたあとユニットのブレーク条件が成立します。このため，AND条件からある条件を除きたい場合は，常にその条件を満たすよう条件を設定してください。

ただし，E）とI）の条件成立にはタイミング信号が関与しているためAND条件から除くにはRELEASE～FROM AND～で１を設定してください。

●D）PROG ADDR UPPER　：　既定値　?

プログラム・アドレスのブレーク／トレース条件範囲の上限を指定します。

PROG ADDR LOWER　：　既定値　?

プログラム・アドレスのブレーク／トレース条件範囲の下限を指定します。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記プログラム・アドレス範囲内がブレーク／トレース条件となります。

UNMATCHを指定すると上記プログラム・アドレス範囲外がブレーク／トレース条件となります。

●E）RELEASE DATAMEMORY FROM AND YES(1)/NO(0)　：　既定値　?

データ・メモリに関する項目E）をD）～K）のAND条件（(C)で１を選んだ場合）から削除するときは，１を入力します。また，D）～K）のOR条件（(C)で０を選んだ場合）が選択されているときは，この設定内容は無視されるため１でも０でもかまいません。

データ・メモリに関する条件は，DATA ADDR, CURRENT DATA, PREVIOUS DATA（ユニットによってはないものもある）の３つの条件のAND条件となっています。

DATA ADDR　：　既定値　?

データが書き込まれたデータ・メモリ・アドレスによるブレーク／トレース条件の設定を行います。

DATA ADDR MASK　：　既定値　?

ブレーク／トレース条件になるデータ・メモリ・アドレスに対してのマスク・データの設定を行います。マスク・データは，ブレーク／トレース条件にしたいデータ・メモリ・アドレスのビットには１を設定し，１でも０でもかまわないビットには０を設定した16進データです。

ユニット２には，この項目がないのでデータ・メモリ・アドレスのブレーク／トレース条件を無効にできません。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記DATA ADDR値がブレーク／トレース条件となります。

UNMATCHを指定すると上記DATA ADDR値以外の値がブレーク／トレース条件となります。

CURRENT DATA　：　既定値　?

書き込まれたデータ・メモリの値によるブレーク／トレース条件の設定を行います。

CURRENT MASK　：　既定値　?

ブレーク／トレース条件になるデータ・メモリの値に対してのマスク・データの設定を行います。

ユニット２には，この項目がないので，データ・メモリのブレーク／トレース条件を無効にできません。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記CURRENT DATA値がブレーク／トレース条件となります。

UNMATCHを指定すると上記CURRENT DATA値以外の値がブレーク／トレース条件となります。

PREVIOUS DATA DISABLE YES(1)/NO(0)　：　既定値　?

DATA ADDR, CURRENT DATA, PREVIOUS DATAに関するブレーク／トレース条件は，AND条件であるため，PREVIOUS DATAに関してのブレーク／トレース条件をE）の項目から除きたいときには１を入力します。

PREVIOUS DATA　：　既定値　?

データが書き込まれる前のデータ・メモリの値によるブレーク／トレース条件の設定を行います。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記PREVIOUS DATA値がブレーク／トレース条件となります。

UNMATCHを指定するとPREVIOUS DATA値以外の値がブレーク／トレース条件となります。

●F）SP LEVEL UPPER　：　既定値　?

スタック・ポインタのブレーク／トレース条件範囲の上限を指定します。

SP LEVEL LOWER　：　既定値　?

スタック・ポインタのブレーク／トレース条件範囲の下限を指定します。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記スタック・ポインタの範囲内がブレーク／トレース条件となります。

UNMATCHを指定すると上記スタック・ポインタの範囲外がブレーク／トレース条件となります。

●G）INST CODE　：　既定値　?

実行される命令コードによるブレーク／トレース条件の設定を行います。

命令コードの表記形式は1-4-3-4-4ビット形式です。

INST MASK　：　既定値　？

ブレーク／トレース条件になる命令コードに対してのマスク・データの設定を行います。

MATCH(1)/UNMATCH(0)：　既定値　？

MATCHを指定すると上記命令コードがブレーク／トレース条件となります。

UNMATCHを指定すると上記命令コード以外のコードがブレーク／トレース条件となります。

●I)　現在，この項目はサポートされていません。

このため，この項目のブレーク／トレース条件を下記のように常に無効にしておいてください。

```
I) RELEASE MAR FROM AND YES(1) / NO(0) : 0 ? 1
   MAR  DATA              : 0 ? 0
   MAR  MASK              : 0 ? 1

   MATCH(1) / UNMATCH(0): 0 ? 0
```

●J)　INTERRUPT ACKNOWLEDGE　：　既定値　？

インタラプトの発生によるブレーク／トレース条件の設定を行います。

１を設定した場合，プログラム実行中にインタラプトが発生するとブレーク／トレース条件が成立します。

インタラプトの発生によるブレークおよびトレース開始するアドレスは，対応するベクタ・アドレスです。

INTERRUPT MASK　：　既定値　？

ブレーク／トレース条件になるインタラプトに関しての既定値に対してのマスク・データの設定を行います。

MATCH(1)/UNMATCH(0)　：　既定値　？

MATCHを指定すると上記で設定されたインタラプトに関しての設定値がブレーク／トレース条件値となります。

UNMATCHを指定すると上記で設定されたインタラプトに関しての設定値以外がブレーク／トレース条件値となります。

●K)　DMA　：　既定値　？

DMA (Direct Memory Access) の発生によるブレーク／トレース条件の設定を行います。

DMAが発生したときをブレーク／トレース条件にする場合には１を設定し，DMA

が発生していないときをブレーク／トレース条件にする場合には，0を設定します。
DMAが発生しているときには，ブレークがかかりませんのでご注意ください。

DMA MASK　：　既定値　?

ブレーク／トレース条件になるDMAに関しての設定値に対してのマスク・データの設定を行います。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記DMAに関しての設定値がブレーク／トレース条件となります。

UNMATCHを指定すると上記DMAに関しての設定値以外がブレーク／トレース条件となります。

●L)　COUNTER SOURCE SELECT

NO(0)/INST(1)/CONDITION(2)/INST AFTER CONDITION(3)　：　0　?

カウンタのオーバフローによるブレーク／トレース条件の設定を行います。
カウンタは0を初期値として+1ずつインクリメントするアップ・カウンタになっています。

・NO(0)　……カウンタを使用しない。
・INST(1)　……無条件に命令の実行数をカウントする。
・CONDITION(2)　…C) ～K) で設定したユニットとしてのブレーク／トレース条件を満足する命令の実行回数をカウントする。
・INST AFTER CONDITION(3)
　…C) ～K) までの項目の条件成立後，実行された命令の実行数をカウントする。

TERMINAL COUNTER　：　既定値　?

カウンタの最終値を設定します。

COUNTER MASK　：　既定値　?

ブレーク／トレース条件になるカウンタの設定値に対してのマスク・データ値の設定を行います。

MATCH(1)/UNMATCH(0)　：　既定値　?

MATCHを指定すると上記カウンタ値がブレーク／トレース条件となります。
UNMATCHを指定すると上記カウンタ値以外の値がブレーク／トレース条件となります。

（各ユニットの出力例）

＜ユニット０＞

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) : 0 ?
D) PROG ADDR UPER     : FFFF ?  ┐
   PROG ADDR LOWER    : 0000 ?  │ プログラム・メモリ
   MATCH(1) / UNMATCH(0) : 0 ?  ┘
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ?  ┐
   DATA ADDR          : 000 ?                      │
   DATA ADDR MASK     : 000 ?                      │ データ・メモリ
   MATCH(1)/UNMATCH(0)-1 : 0 ?                     │
   CURRENT   DATA     : 0 ?                        │
   CURRENT   MASK     : 0 ?                        ┘
   MATCH(1) / UNMATCH(0) : 0 ?
F) SP LEVEL UPER      : 0 ?  ┐
   SP LEVEL LOWER     : 0 ?  │ スタック・ポインタ
   MATCH(1) / UNMATCH(0) : 0 ?  ┘
J) INTERRUPT ACKNOWLEDGE : 0 ?  ┐
   INTERRUPT MASK     : 0 ?     │ インタラプト
   MATCH(1) / UNMATCH(0) : 0 ?  ┘
K) DMA                : 0 ?  ┐
   DMA MASK           : 0 ?  │ DMA
   MATCH(1) / UNMATCH(0) : 0 ?  ┘
L) COUNTER SOURCE SELECT
   NO(0) / INST(1) / CONDITION(2) / INST AFTER CONDITION(3) : 0 ?  ┐
   TERMINAL CONTER    : 0000 ?                                    │ カウンタ
   COUNTER  MASK      : 0000 ?                                    │
   MATCH(1) / UNMATCH(0) : 0 ?                                    ┘
```

保守／廃止

＜ユニット１＞

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) : 0 ?
D) PROG ADDR UPER     : FFFF ?  ┐
   PROG ADDR LOWER    : 0000 ?  ┘ プログラム・メモリ
   MATCH(1) / UNMATCH(0) : 0 ?
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? ┐
   DATA ADDR          : 000 ?                    │
   DATA ADDR MASK     : 000 ?                    │
   MATCH(1) / UNMATCH(0) : 0 ?                   │
   CURRENT   DATA     : 0 ?                      │ データ・メモリ
   CURRENT   MASK     : 0 ?                      │
   MATCH(1) / UNMATCH(0) : 0 ?                   │
   PREVIOUS DATA DISABLE YES(1) / NO(0) : 0 ?    │
   PREVIOUS DATA      : 0 ?                      │
   MATCH(1) / UNMATCH(0) : 0 ?                   ┘
I) RELEASE MAR FROM AND YES(1) / NO(0) : 0 ?  ┐   *
   MAR   DATA         : 0 ?                   │ MAR
   MAR   MASK         : 0 ?                   │
   MATCH(1) / UNMATCH(0) : 0 ?                ┘
L) COUNTER SOURCE SELECT
   NO(0) / INST(1) / CONDITION(2) / INST AFTER CONDITION(3) : 0 ? ┐
   TERMINAL CONTER    :   00 ?                                   │ カウンタ
   COUNTER  MASK      :   00 ?                                   │
   MATCH(1) / UNMATCH(0) : 0 ?                                   ┘
```

＊ 現在サポートされていません。

＜ユニット２＞

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 2
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) : 0 ?
D) PROG ADDR UPER     : FFFF ?  ┐
   PROG ADDR LOWER    : 0000 ?  ┘ プログラム・メモリ
   MATCH(1) / UNMATCH(0) : 0 ?
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : 0 ?  ┐
   DATA ADDR          : 000 ?                        │
   MATCH(1) / UNMATCH(0) : 0 ?                       │ データ・メモリ
   CURRENT   DATA     : 0 ?                          │
   MATCH(1) / UNMATCH(0) : 0 ?                       ┘
L) COUNTER SOURCE SELECT
   NO(0) / INST(1) / CONDITION(2) / INST AFTER CONDITION(3) : 0 ?  ┐
   TERMINAL CONTER    :   00 ?                                    │ カウンタ
   COUNTER  MASK      :   00 ?                                    │
   MATCH(1) / UNMATCH(0) : 0 ?                                    ┘
```

＜ユニット３＞

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 3
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) : 0 ?
D) PROG ADDR UPER     : FFFF ?  ┐
   PROG ADDR LOWER    : 0000 ?  ┘ プログラム・メモリ
   MATCH(1) / UNMATCH(0) : 0 ?
G) INST CODE          : 00000 ?  ┐
   INST MASK          : 00000 ?  ┘ 命令コード
   MATCH(1) / UNMATCH(0) : 0 ?
```

＜ブレーク条件設定項目（レベル２について）＞

レベル１で設定した各ユニット（ユニット0～3）をブレーク条件として設定する場合は，レベル２を使用します。

設定はDEPTHという概念で４段階に階層化されています。

１つのDEPTH内で４つのユニットのOR条件を設定でき，１を指定したユニットはOR条件の対象ユニットとなり，０の場合無効となります。

DEPTH内のOR条件が成立すると次のDEPTHの条件待ちとなりDEPTH-0の条件が成立すると，ブレークします。

条件待ちの順序はDEPTH3→DEPTH0の順です。

INITIAL DEPTHの設定で開始するDEPTHを指定できます。

```
(例)  BRK>.CC$$
      A) LEVEL(1 , 2) : ? 2
      B) LEVEL2       : 0123
           DEPTH-3    : 0101   0000
           DEPTH-2    : 1101 ? 1111
           DEPTH-1    : 1101 ? 1010
           DEPTH-0    : 1101 ? 0001
         INITIAL DEPTH : 0 ? 1
```

レベル１で設定したユニット０またはユニット２の条件が成立したあと，レベル１で設定したユニット３の条件が成立したらブレークするように設定しています。

保守／廃止

表５－１　ブレーク／トレース条件一覧表

項　　目	UNIT0	UNIT1	UNIT2	UNIT3
C) CONDITION AND(1)/OR(0)	○	○	○	○
D) PROG ADDR UPER PROG ADDR LOWER MATCH(1)/UNMATCH(0)	○	○	○	○
E) RELEASE DATA MEMORY FROM AND YES(1)/NO(0) DATA ADDR	○	○	○	×
DATA ADDR MASK	○	○	×	×
MATCH(1)/UNMATCH(0) CURRENT DATA	○	○	○	×
CURRENT MASK	○	○	×	×
MATCH(1)/UNMATCH(0)	○	○	○	×
PREVIOUS DATA DISABLE YES(1)/NO(0) PREVIOUS DATA MATCH(1) UNMATCH(0)	×	○	×	×
F) SP LEVEL UPER SP LEVEL LOWER MATCH(1)/UNMATCH(0)	○	×	×	×
G) INST CODE INST MASK MATCH(1)/UNMATCH(0)	×	×	×	○
I) RELEASE MAR FROM AND YES(1)/NO(0) MAR DATA MAR MASK MATCH(1)/UNMATCH(0)	×	○*	×	×
J) INTERRUPT ACKNOWLEDGE INTERRUPT MASK MATCH(1)/UNMATCH(0)	○	×	×	×
K) DMA DMA MASK MATCH(1)/UNMATCH(0)	○	×	×	×
L) COUNTER SOURCE SELECT NO(0)/INST(1)/CONDITION(2)/INST AFTER CONDITION(3) TERMINAL COUNTER COUNTER MASK MATCH(1)/UNMATCH(0)	○	○	○	×

○…設定可能
×…設定不可
＊現在サポートされていません

保守／廃止

.CT　トレース・オン／オフ条件の変更（Change Trace Condition）

書　式　：　.CT

〔機　能〕　トレース・オン／オフ条件の変更を行います。

〔説　明〕　.CCのレベル１で設定した各ユニットをトレース条件として設定します。トレース・オン／オフ条件の設定は下記のとおりです。

```
BRK>.CT$$
      TRACE CONDITION MODE
D : TRACE DON'T CARE          ----①
T : TRACE ON                  ----②
U : TRACE OFF                 ----③
S : TRACE ONE SHOT            ----④
LEVEL 1 UNIT : 0123
             : DDDD ?
               ┗━┛
              既定値
```

① ユニットのトレース条件が成立しても，トレースに何ら影響を与えません。

② ユニットのトレース条件が成立すると，トレースをON（開始）します。

③ ユニットのトレース条件が成立すると，トレースをOFF（終了）します。

④ ユニットのトレース条件が成立しているところだけをトレースします（トレース・ワンショット）。

（例）
```
BRK>.CT$$
      TRACE CONDITION MODE
D : TRACE DON'T CARE
T : TRACE ON
U : TRACE OFF
S : TRACE ONE SHOT
LEVEL 1 UNIT : 0123
             : DDDD ? TUSS
```

ユニット０の条件が成立するとトレースを開始します。

ユニット１の条件が成立するとトレースを終了します。

ユニット２またはユニット３の条件が成立している間，トレースを行います。

保守／廃止

〔注　意〕　・同一箇所で複数のユニットによるトレース条件が成立する場合，トレース条件の有効となる優先順位は次のとおりです。

TRACE ON＞TRACE ONE SHOT＞TRACE OFF

・トレースには，アドレス・トレースとステータス・トレースがあり，アドレス・トレースは本コマンドの設定とは無関係にトレースされます。

・.R入力後の実行および.BGで実行を開始した場合は，トレース・オンとなります。

・.R入力後の実行および.BGで実行を開始した場合は，.CTで設定した内容は影響を受けません。

・トレース・オン，トレース・オフ後は，TRACE DON'T CAREに設定しても，その状態が保持されます。

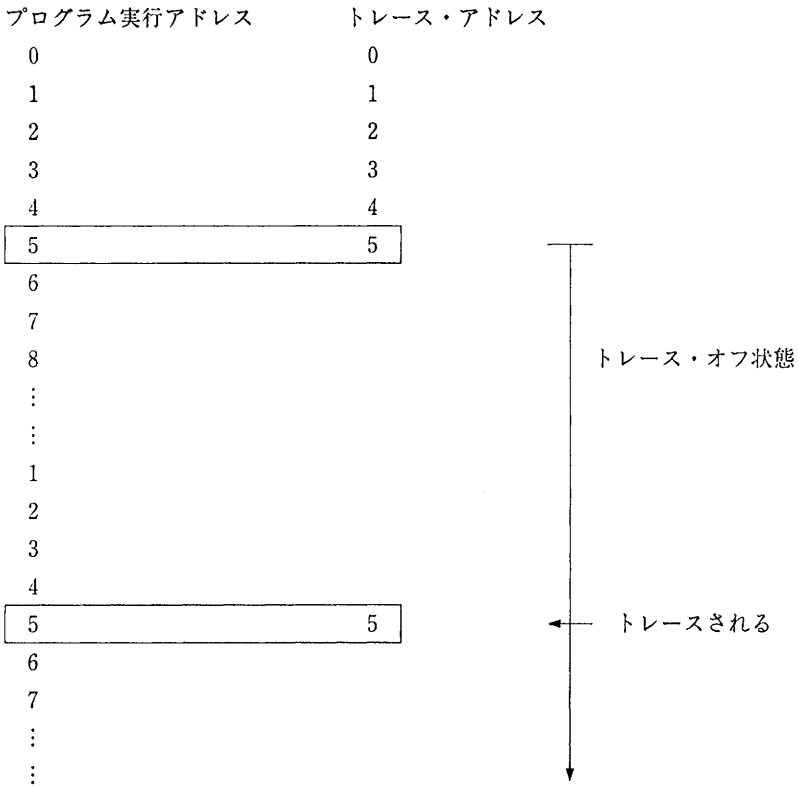
TRACE ONE SHOTは，トレース・オフの状態のときしか有効に働きません。トレース・オフ状態でTRACE ONE SHOTを指定すると，条件成立したアドレスのみトレースされます。

また，TRACE ONE SHOT指定後，TRACE DON'T CAREにしてもTRACE ONE SHOTの指定が保持されるのではなくTRACE ONE SHOT指定前の状態であるトレース・オフの状態となっています。

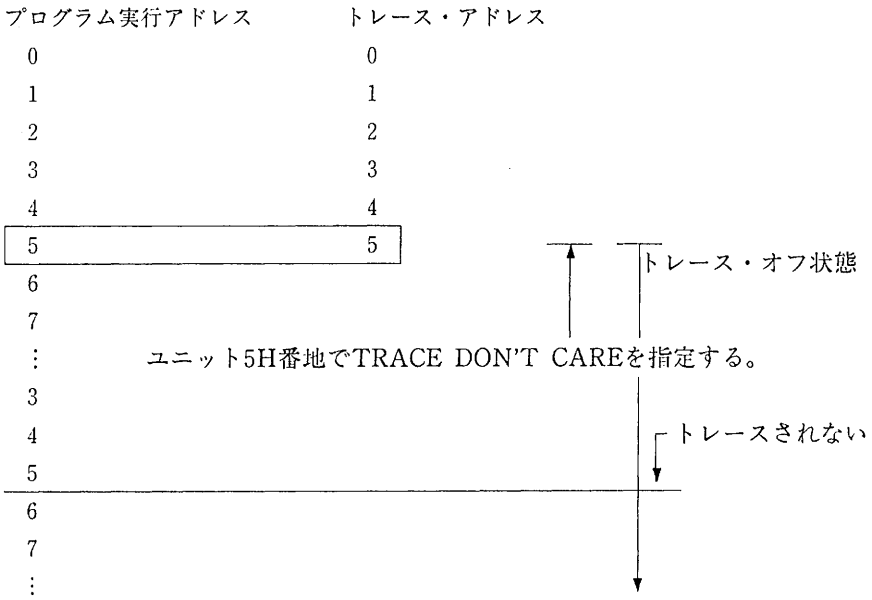
・トレース・オフを指定すると，トレース・オフ後のトレースは実行されませんが，トレース・オフの開始を決定する実行アドレスはトレースされます。この場合，TRACE ONE SHOTの指定の場合と同じように動作します。

保守／廃止

（例１）5H番地でトレース・オフを開始したあと，トレース・オフの状態が続く場合



（例２）5H番地でトレース・オフを開始したあと，同じ5H番地でTRACE DON'T CAREを指定した場合



保守／廃止

表５－２　トレース状態の遷移

現在のトレース状態 / 成立条件	トレース・オン中	トレース・オフ中	トレース・ワンショット中
トレース・オン	トレースを継続	トレースを開始	トレースを開始
トレース・オフ	トレースを終了	トレース・オフを継続	トレース・ワンショットを終了
トレース・ワンショット	トレースを継続（ワンショットは無効）	トレース・ワンショットを開始	トレース・ワンショットを継続（最新の条件が有効）

保守／廃止

.DC　ブレーク／トレース条件のダンプ（Dump Break Condition）

書　式　：　.DC

〔機　能〕　ブレーク／トレース条件をダンプします。

（例）　ユニット０～３のブレーク／トレース条件をダンプします。

```
BRK>.DC$$
UNIT (0 - 3) ? 0
CONDITION  : OR
PROG ADDR  : FFFF - 0000 UNMATCH
DATA ADDR  : 000 <000> UNMATCH
     CRNT  :  0  < 0 > UNMATCH
SP  LEVEL  : F - 0 UNMATCH
INTERRUPT  : 0 <0> UNMATCH
DMA        : 0 <0> UNMATCH
COUNT SEL  : NO         0000 <0000> UNMATCH
TRACE SEL  : TRACE ON

BRK>.DC$$
UNIT (0 - 3) ? 1
CONDITION  : OR
PROG ADDR  : FFFF - 0000 UNMATCH
DATA ADDR  : 000 <000> UNMATCH
     CRNT  :  0  < 0 > UNMATCH
     PRVS  :  0 UNMATCH
MAR  DATA  : 0 <0> UNMATCH
COUNT SEL  : NO           00  <  00> UNMATCH
TRACE SEL  : TRACE OFF

BRK>.DC$$
UNIT (0 - 3) ? 2
CONDITION  : OR
PROG ADDR  : FFFF - 0000 UNMATCH
DATA ADDR  : 000 <000> UNMATCH
     CRNT  :  0  < 0 > UNMATCH
COUNT SEL  : NO           00  <  00> UNMATCH
TRACE SEL  : TRACE DON'T CARE

BRK>.DC$$
UNIT (0 - 3) ? 3
CONDITION  : OR
PROG ADDR  : FFFF - 0000 UNMATCH
INST CODE  : 0000 <0000> UNMATCH
TRACE SEL  : TRACE DON'T CARE
```

〔注　意〕　＜　＞内はマスク・データです。

保守／廃止

.DT　トレース・テーブルのダンプ（Dump Trace Table）

書　式　：[α, β].DT
α：ダンプ開始トレース番号　　（α≦βであること β：ダンプ終了トレース番号　　α＞βは，エラー）

〔機　能〕　指定されたトレース番号αからβまでのトレース内容をダンプします。α,β両方を省略すると，トレース・テーブルの最後の内容が表示されます。システム・リセットによりトレース・テーブルは初期化され，トレース・カウンタが0になります。

32 K(32768　10進数)ステップ以内のトレースでは，トレース･カウンタはトレース･テーブルの最後を示しています。

32Kステップを越えるトレースでは，最新の32Kステップのトレース内容がトレース･テーブルに格納され，トレース・カウンタは7FFFH（32767）になります。

トレースには，アドレス・トレースとステータス・トレースがあります。

アドレス・トレースはトレース条件に関係なく，最新のプログラム実行内容をトレースします。

ステータス・トレースはトレース条件で指定した範囲をトレースします。

また，ステータス・トレースはアドレス・トレースよりも多くの情報を含んでいます。

（例1）　トレース番号0から10のアドレス・トレース結果をダンプします。

```
BRK>0,^D10.DT$$
ADDRESS(1) / STATUS (0) TRACE?  1 ↵

 TR_NO        ADDR      MNEMONIC    INST
 0000  0000   0000   MOV 00 ,#A    1D00A
 0001  0001   0001   MOV 01 ,#B    1D01B
 0002  0002   0002   MOV 02 ,#C    1D02C
 0003  0003   0003   MOV 03 ,#D    1D03D
 0004  0004   0004   MOV 04 ,#E    1D04E
 0005  0005   0005   MOV 05 ,#F    1D05F
 0006  0006   0006   MOV 06 ,#0    1D060
 0007  0007   0007   MOV 07 ,#1    1D071
 0008  0008   0008   MOV 08 ,#2    1D082
 0009  0009   0009   MOV 09 ,#3    1D093
 0010  000A   000A   NOP           074F0
 (1)   (2)    (3)      (4)          (5)
```

（例２）　０から10Hまでステータス・トレース結果をダンプします。

```
BRK>0,10.DT$$
ADDRESS(1) / STATUS (0) TRACE? 0 ↵
```

TR_NO		ADDR	INSTRUCTION		WA	DB	JG	TIME
0000	0000	0000	MOV 00 ,#A	1D00A	000	A	0	0000001
0001	0001	0001	MOV 01 ,#B	1D01B	001	B	0	0000002
0002	0002	0002	MOV 02 ,#C	1D02C	002	C	0	0000003
0003	0003	0003	MOV 03 ,#D	1D03D	003	D	0	0000004
0004	0004	0004	MOV 04 ,#E	1D04E	004	E	0	0000005
0005	0005	0005	MOV 05 ,#F	1D05F	005	F	0	0000006
0006	0006	0006	MOV 06 ,#0	1D060	006	0	0	0000007
0007	0007	0007	MOV 07 ,#1	1D071	007	1	0	0000008
0008	0008	0008	MOV 08 ,#2	1D082	008	2	0	0000009
0009	0009	0009	MOV 09 ,#3	1D093	009	3	0	0000010
0010	000A	000A	NOP	074F0	04F	0	0	0000011
0011	000B	000B	NOP	074F0	04F	0	0	0000012
0012	000C	000C	NOP	074F0	04F	0	0	0000013
0013	000D	000D	NOP	074F0	04F	0	0	0000014
0014	000E	000E	NOP	074F0	04F	0	0	0000015
0015	000F	000F	BR 000F	0C00F	000	F	0	0000016
0016	0010	000F	BR 000F	0C00F	000	F	0	0000017
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

(1)　トレース番号の10進表示

(2)　トレース番号の16進表示

(3)　プログラム・アドレス（プログラム・カウンタの値）

(4)　命令のニモニック表示

(5)　命令コード（1-4-3-4-4ビット形式）

(6)　データ・メモリの書き込みアドレス

データ・メモリにデータを書き込まれたときのみ有効です。[注]

(7)　データ・バス

データ・メモリにデータを書き込まれたとき[注]，その書き込まれた値を示します。

(8)　スキップ命令実行時にスキップされた命令には＊が表示されます。

(9)　タイム・スタンプ

.RNコマンドにより１になり，１命令実行するごとに１カウント・アップします（ただし，MOVT命令を実行すると２カウント・アップします）。

保守／廃止

注　データ・メモリに書き込まれたときのみ有効です。レジスタ・ファイルへの書き込み（PEEK命令），周辺レジスタへの書き込み（PUT命令）で表示されている内容はIE-17K-ET上のバスの内容を表示しているため，表示されている内容は無効です。

保守／廃止

.SC0　.SC1　ブレーク／トレース条件のセーブ(Save Break/Trace Condition)

書　式　：　{.SC0 / .SC1}
RS-232-C　チャネル０：SC0 チャネル１：SC1

〔機　能〕　.CCのレベル１で設定したブレーク・トレース条件を.SC0または.SC1で指定したRS-232-Cのチャネルにインテル・ヘキサ形式で出力します。

（例）　ブレーク／トレース条件をチャネル０に出力します。

```
BRK>.SC0$$
:104143000B0B0B0B00FF020202000000000001003A
:1041530000010100000000FFFF0000100000FF0746
:10416300B000F0400000F00040000FFFF00C0011C
:1041730000001000F000001010001010000000FF29
:10418300FF000010000000FFFF0000100000FF0709
:104193000B000F0400040F00000000FFFF000001EC
:1041A3000001000F040001000001000000000FFF7
:1041B300FF000008000000FFFF0000100000FF07E1
:1041C3000B000F0400000F00000000FFFF000001C0
:1041D3000001000F000001000001000000000FFCB
:1041E300FF000008000000FFFF0000100000FF07B1
:1041F30000000F0000000F00000000FFFF1000018F
:104203000001000F000001000001000000000FF9A
:07421300FF0000000000A5
:00000001FF
```

保守／廃止

.LC0　.LC1　ブレーク／トレース条件のロード(Load Break/Trace Condition)

書　式　：{.LC0 / .LC1}
RS-232-C　チャネル０：LC0 チャネル１：LC1

〔機　能〕　IE-17K-ETに送られてきたデータを，IE-17K-ET内部のブレーク／トレース条件を格納するエリアに格納します。

（例）チャネル０からブレーク／トレース条件を入力します。

```
BRK>.LC0$$
```

保守／廃止

.VC0　.VC1　ブレーク／トレース条件のベリファイ(Verify Break/Trace Condition)

書　式　：{.VC0 / .VC1}
RS-232-C　チャネル０：VC0 チャネル１：VC1

〔機　能〕　IE-17K-ET内部のブレーク／トレース条件と，IE-17K-ETに送られてくるデータをベリファイします。

もし，同じものであれば，

Verify OK

異なれば，

Verify NG

のメッセージを出力します。

(例)　チャネル０から入力されたブレーク／トレース条件をベリファイします。

```
BRK>.VC0$$
Verify OK
```

保守／廃止

5.8.6　カバレージ表示コマンド

(1)　カバレージ・メモリのダンプ

.DM (Dump Coverage Memory)

保守／廃止

.DM　カバレージ・メモリのダンプ（Dump Coverage Memory）

書　式　：[α, β].DM
α：スタート・アドレス　　（α≦βであること β：エンド・アドレス　　α＞βは，エラー）

〔機　能〕　カバレージ・メモリの内容をダンプします。

カバレージの対象となるPC（プログラム・カウンタ）とDATAの２つがあります。

・PCカバレージは実行したプログラム・アドレスに対して実行回数を0～FFHまで記録します。FFH以上はFFHが表示されます。

・DATAカバレージは，データ・メモリの書き込み状態（ビット単位）を表示します。表示の意味は次のとおりです。

＜表示の意味＞

— …… 一度も書き込まれていないビット

＊ …… ０と１が書き込まれたビット

０…… ０のみ書き込まれたビット

１…… １のみ書き込まれたビット

・PCカバレージを選択した場合，α，βを省略するとプログラム・メモリの0～7FH番地を表示します。

・DATAカバレージを選択した場合，α，βを省略するとデータ・メモリの0～3FH番地を表示します。レジスタ・ファイルはカバレージ対象外となります。

（例１）　PCカバレージの内容を表示します。

```
BRK>.DM$$
PC (1) / DATA (0)  COVERAGE : ? 1↵
ADDR  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

（例２）　DATAカバレージの内容を表示します。

```
BRK>.DM$$
PC (1) / DATA (0)  COVERAGE : ? 0↵
ADDR  0/8   1/9   2/A   3/B   4/C   5/D   6/E   7/F
0000  ----  ----  ----  ----  ----  ----  ----  ----
0008  ----  ----  ----  ----  ----  ----  ----  ----
0010  ----  ----  ----  ----  ----  ----  ----  ----
0018  ----  ----  ----  ----  ----  ----  ----  ----
0020  ----  ----  ----  ----  ----  ----  ----  ----
0028  ----  ----  ----  ----  ----  ----  ----  ----
0030  ----  ----  ----  ----  ----  ----  ----  ----
0038  ----  ----  ----  ----  ----  ----  ----  ----
```

〔注　意〕　　デバイスによっては，α，βを省略した場合エラーとなるものがあります。

保守／廃止

５.８.７　ヘルプ・コマンド

(1)　全コマンドの表示

.H（Help）

保守／廃止

.H　全コマンドの表示（Help）

書　式　：　.H

〔機　能〕　　コマンド一覧を表示します。

（例）　コマンド一覧を表示します。

```
BRK>.H$$
.IP .CP .DP .FP .SP0 .SP1 .LP0 .LP1 .VP0 .VP1 .XS0 XS1
                    << PROGRAM MEMORY COMMAND  >>
.ID .CD .DD .D
                    << DATA MEMORY COMMAND  >>
.R  .RN .BG .BK .CA .S
                    << EMULATION COMMAND  >>
.CC .CT .DC .DT .DM .SC0 .SC1 .LC0 .LC1 .VC0 .VC1
                    << BREAK , TRACE CONDITION COMMAND  >>
```

保守／廃止

第６章　プログラムの実行

プログラムの実行には以下の方法があります。

(1)　リアルタイム・エミュレーション

(2)　１ステップ・エミュレーション

６.１　リアルタイム・エミュレーション

実際の製品と同じ速さでプログラムを実行させたい場合，.RNを使用します。ブレークポイントを設定することにより，任意の条件でブレークすることが可能です。

また，.BKにより実行を強制的にブレークすることも可能です。

（例１）　CPUにリセットをかけたあと実行します。

```
BRK>.R$$
BRK>.RN$$
RUN>
```

（例２）　リアルタイム・エミュレーションを中断したあと，再開します。

```
RUN>.BK$$
ADDR INSTRUCTION
0027 1E7F2   BREAK
0028 0C026   OVERRUN
0029 070E0   NEXT
 PC  SP   AR  WR BR  MP  IX
0029  0  9999  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   1   0   1  *   *   0
 RP  0123456789ABCDEF
 *0  8D98D99999FFAD9D

BRK>.RN$$

RUN>
```

保守／廃止

6.2　ブレークポイントの設定方法

ブレークポイントを設定することにより，任意の条件でブレークさせることができます。ブレーク条件にはプログラム・メモリ・アドレス，データ・メモリ・アドレス，データ・メモリへのデータの書き込みなどを設定できます（図6－1参照）。

また，ブレーク条件は単独で使用するだけでなく，複数のブレーク条件が同時に成立したときブレークするように設定したり，複数のブレーク条件が続けて成立したときブレークするように設定できます（図6－2参照）。

保守／廃止

図６－１　ブレーク条件の設定

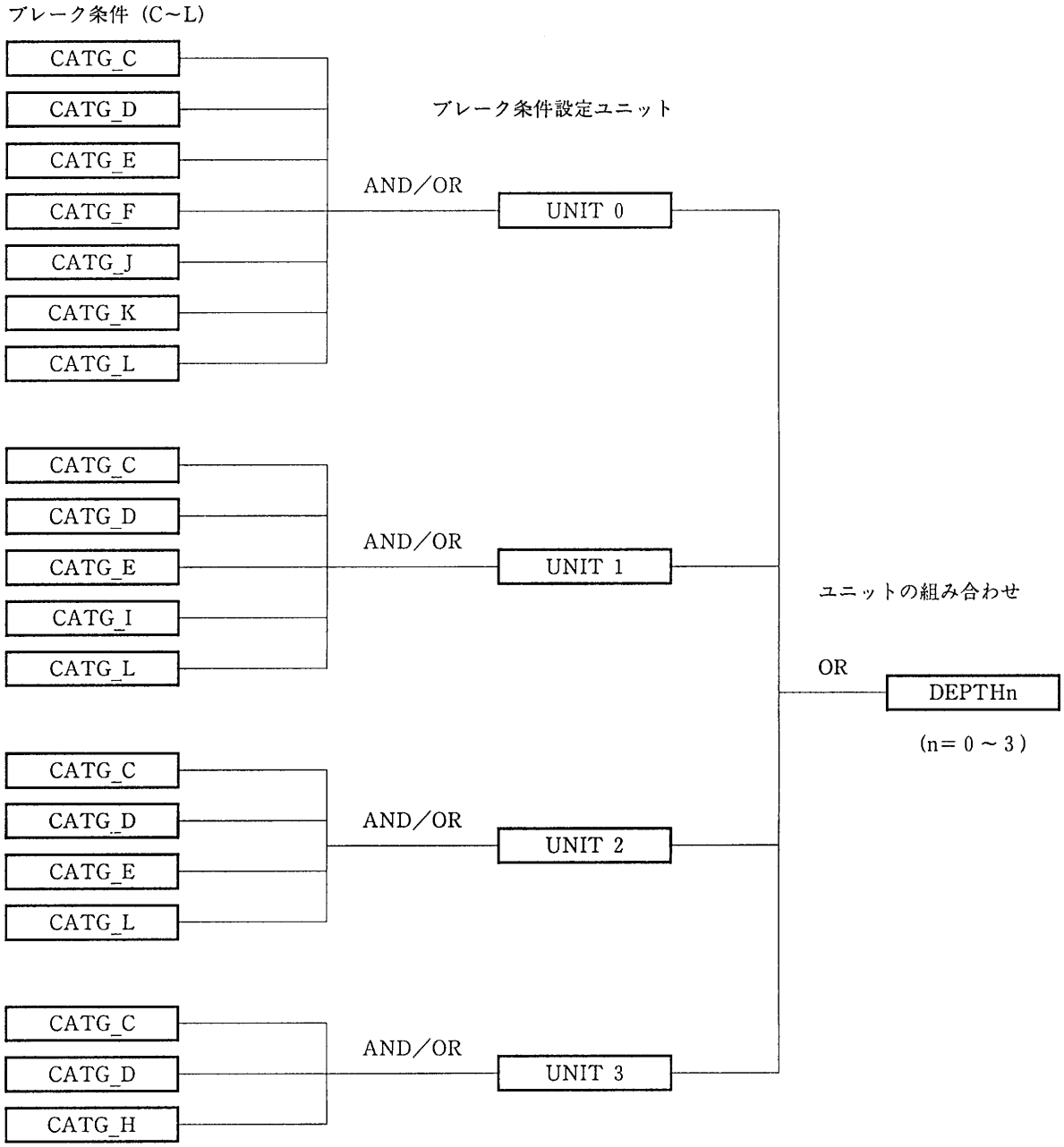


図６－２　ブレーク条件のシーケンスによるブレーク

６.２.１　プログラム・アドレスによるブレーク

プログラム・アドレスによるブレークは，.CCコマンドでアドレスを指定することにより行います。

（例１）　プログラム・アドレスが00F0H番地になったときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00F0
   PROG ADDR LOWER    : 0000 ? 00F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2         : 0123
     DEPTH-3      : 1101 ? 0000
     DEPTH-2      : 1101 ? 0000
     DEPTH-1      : 1101 ? 0000
     DEPTH-0      : 1101 ? 1000
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D069   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

（例２）　プログラム・アドレスが00F0H～00FFH番地の範囲に入ったときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00FF
   PROG ADDR LOWER    : 0000 ? 00F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2         : 0123
     DEPTH-3      : 1101 ? 0000
     DEPTH-2      : 1101 ? 0000
     DEPTH-1      : 1101 ? 0000
     DEPTH-0      : 1101 ? 1000
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D069   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

（例３）　プログラム・アドレスが0000H～00EFH番地の範囲を越えたときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00EF
   PROG ADDR LOWER    : 0000 ? 0000
   MATCH(1) / UNMATCH(0) 0 ? 0
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2         : 0123
     DEPTH-3      : 1101 ? 0000
     DEPTH-2      : 1101 ? 0000
     DEPTH-1      : 1101 ? 0000
     DEPTH-0      : 1101 ? 1000
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D069   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

（例４）　プログラム・アドレス00F0H番地を５回実行したときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00F0
   PROG ADDR LOWER    : 0000 ? 00F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0
   CATG (C - L) : ? L
L) COUNTER SOURCE SELECT
   NO(0)/INST(1)/CONDITION(2)/INST AFTER CONDITION(3) : 0 ? 2
   TERMINAL COUNTER   : 00 ? 5          上記の条件が５回成立した
   COUNTER MASK       : 00 ? FF         ときブレーク
   MATCH(1)/UNMATCH(0) : 0 ? 1

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2          : 0123
     DEPTH-3       : 1101 ? 0000
     DEPTH-2       : 1101 ? 0000
     DEPTH-1       : 1101 ? 0000
     DEPTH-0       : 1101 ? 1000
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D069   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0 *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

６.２.２　データ・メモリの変更によるブレーク

データ・メモリの変更によるブレークは，.CCコマンドでデータ・メモリ・アドレスや変更するデータを指定することにより行います。

（例１）　データ・メモリ・アドレス1.30H番地が変更されたときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1
   CATG (C - L) : ? E
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? 0
   DATA ADDR           : 000 ? 130     ﾃﾞｰﾀ･ﾒﾓﾘ･ｱﾄﾞﾚｽを1.30Hにする
   DATA ADDR MASK      : 000 ? F7F     ｱﾄﾞﾚｽ指定の全ﾋﾞｯﾄが有効
   MATCH(1)/UNMATCH(0) : 0 ? 1         上記ｱﾄﾞﾚｽになった時ブレーク
   CURRENT DATA        : 0 ? 0
   CURRENT MASK        : 0 ? 0         書き込むデータには無関係
   MATCH(1)/UNMATCH(0) : 0 ? 1
   PREVIOUS DATA DISABLE YES(1)/NO(0) : 0 ? 1
   PREVIOUS DATA       : 0 ? 0         変更前のデータには無関係
   MATCH(1)/UNMATCH(0) : 0 ? 1

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2          : 0123
     DEPTH-3       : 1101 ? 0000
     DEPTH-2       : 1101 ? 0000
     DEPTH-1       : 1101 ? 0000
     DEPTH-0       : 1101 ? 0100
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D309    BREAK
00F1 1D059    OVERRUN
00F2 1D045    NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  0  1  000 000
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  0   0   0
RP   0123456789ABCDEF
00   0000099990008005
BRK>
```

（例２）　データ・メモリ・アドレス1.30H番地に５Hが書き込まれたときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1
   CATG (C - L) : ? E
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? 0
   DATA ADDR           : 000 ? 130     ﾃﾞｰﾀ･ﾒﾓﾘ･ｱﾄﾞﾚｽを1.30Hにする
   DATA ADDR MASK      : 000 ? F7F     ｱﾄﾞﾚｽ指定の全ﾋﾞｯﾄが有効
   MATCH(1)/UNMATCH(0) : 0 ? 1
   CURRENT DATA        : 0 ? 5         書き込む値が5Hの時ブレーク
   CURRENT MASK        : 0 ? F
   MATCH(1)/UNMATCH(0) : 0 ? 1
   PREVIOUS DATA DISABLE YES(1)/NO(0) : 0 ? 1
   PREVIOUS DATA       : 0 ? 0         変更前のデータには無関係
   MATCH(1)/UNMATCH(0) : 0 ? 1

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2          : 0123
     DEPTH-3       : 1101 ? 0000
     DEPTH-2       : 1101 ? 0000
     DEPTH-1       : 1101 ? 0000
     DEPTH-0       : 1101 ? 0100
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D305   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  0  1  000 000
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  0   0   0
RP   0123456789ABCDEF
00   0000099990008005
BRK>
```

（例３）　データ・メモリ・アドレス1.3×H番地のビット０がセットされたときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1
   CATG (C - L) : ? E
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? 0
   DATA ADDR           : 000 ? 130     ﾃﾞｰﾀ･ﾒﾓﾘ･ｱﾄﾞﾚｽを1.30Hにする
   DATA ADDR MASK      : 000 ? F70     ｶﾗﾑ･ｱﾄﾞﾚｽは何でもかまわない
   MATCH(1)/UNMATCH(0) : 0 ? 1
   CURRENT DATA        : 0 ? 1         ﾋﾞｯﾄ0がｾｯﾄされた時ﾌﾞﾚｰｸ
   CURRENT MASK        : 0 ? 1
   MATCH(1)/UNMATCH(0) : 0 ? 1
   PREVIOUS DATA DISABLE YES(1)/NO(0) : 0 ? 1
   PREVIOUS DATA       : 0 ? 0         変更前のデータには無関係
   MATCH(1)/UNMATCH(0) : 0 ? 1

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2         : 0123
     DEPTH-3      : 1101 ? 0000
     DEPTH-2      : 1101 ? 0000
     DEPTH-1      : 1101 ? 0000
     DEPTH-0      : 1101 ? 0100
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D309    BREAK
00F1 1D059    OVERRUN
00F2 1D045    NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  0  1  000 000
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  0   0   0
RP   0123456789ABCDEF
00   0000099990008005
BRK>
```

（例４）　データ・メモリ・アドレス1.30H番地が１から５に変更されたときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1
   CATG (C - L) : ? E
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? 0
   DATA ADDR           : 000 ? 130     ﾃﾞｰﾀ･ﾒﾓﾘ･ｱﾄﾞﾚｽを1.30Hにする
   DATA ADDR MASK      : 000 ? F7F     ｱﾄﾞﾚｽ指定の全ﾋﾞｯﾄが有効
   MATCH(1)/UNMATCH(0) : 0 ? 1
   CURRENT DATA        : 0 ? 5         書き込む値が5Hの時ブレーク
   CURRENT MASK        : 0 ? F
   MATCH(1)/UNMATCH(0) : 0 ? 1
   PREVIOUS DATA DISABLE YES(1)/NO(0) : 0 ? 0
   PREVIOUS DATA       : 0 ? 1         変更前のデータが1であること
   MATCH(1)/UNMATCH(0) : 0 ? 1

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2          : 0123
     DEPTH-3       : 1101 ? 0000
     DEPTH-2       : 1101 ? 0000
     DEPTH-1       : 1101 ? 0000
     DEPTH-0       : 1101 ? 0100
   INITIAL DEPTH : 0 ? 0

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D305   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  0  1  000 000
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  0   0   0
RP   0123456789ABCDEF
00   0000099990008005
BRK>
```

６.２.３　複数のブレーク条件によるブレーク

（例１）プログラム・アドレスが00F0H番地か01F0H番地になったときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0              ユニット０に００Ｆ０Ｈを設定
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00F0
   PROG ADDR LOWER    : 0000 ? 00F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1              ユニット１に０１Ｆ０Ｈを設定
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 01F0
   PROG ADDR LOWER    : 0000 ? 01F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2          : 0123
     DEPTH-3       : 1101 ? 0000
     DEPTH-2       : 1101 ? 0000
     DEPTH-1       : 1101 ? 0000
     DEPTH-0       : 1101 ? 1100  ユニット０またはユニット１の条件
   INITIAL DEPTH : 0 ? 0          が成立したときブレーク

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
00F0 1D069   BREAK
00F1 1D059   OVERRUN
00F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
00F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

（例２）　プログラム・アドレス00F0H番地を実行したあと，01F0H番地を実行したときブレークします。

```
BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 0              ユニット０に００Ｆ０Ｈを設定
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 00F0
   PROG ADDR LOWER    : 0000 ? 00F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 1
B) UNIT (0 - 3) : ? 1              ユニット１に０１Ｆ０Ｈを設定
   CATG (C - L) : ? C
C) CONDITION AND(1) / OR(0) 0 : ? 0
D) PROG ADDR UPER     : FFFF ? 01F0
   PROG ADDR LOWER    : 0000 ? 01F0
   MATCH(1) / UNMATCH(0) 0 ? 1
E) RELEASE DATAMEMORY FROM AND YES(1) / NO(0) : ? $

BRK>.CC$$
A) LEVEL(1 , 2) : ? 2
B) LEVEL2         : 0123
     DEPTH-3      : 1101 ? 0000
     DEPTH-2      : 1101 ? 0000
     DEPTH-1      : 1101 ? 1000   ﾕﾆｯﾄ0が成立したときDEPTH0の条件へ
     DEPTH-0      : 1101 ? 0100   ﾕﾆｯﾄ1が成立したときﾌﾞﾚｰｸ
   INITIAL DEPTH : 0 ? 1

BRK>.R$$
BRK>.RN$$
ADDR INSTRUCTION
01F0 1D069   BREAK
01F1 1D059   OVERRUN
01F2 1D045   NEXT
 PC  SP   AR  WR BR  MP  IX
01F2  0  0000  *  *  *** ***
PSW :DB  CP  CY   Z IXE MPE JG
      0   0   0   0  *   *   0
RP   0123456789ABCDEF
*0   0000099990008005
BRK>
```

保守／廃止

６.３　１ステップ・エミュレーション

プログラムを１命令ずつ実行させて，処理の流れを確認したいときに使用します。

（例１）　プログラムを１命令実行させます。

```
BRK>.S$$
BR RP  PC  INSTRUCTION
 * *0 0034 0C03D   $$

BRK>
```

表示された0034H番地の命令はまだ実行されていません。

.Sコマンドの前に数値を記述するとステップ回数を指定できます。

（例２）　33H番地と34番地の２命令を実行します。

```
BRK>33.CA$2.S$$

BR RP  PC  INSTRUCTION
 * *0 0034 0C03D
 * *0 003D 11001   $$

BRK>
```

.Sコマンドを実行したあと，スペースを入力することで次の命令を実行します。

（例３）　33H番地と34H番地の２命令を実行します。

```
BRK>33.CA$.S$$

BR RP  PC  INSTRUCTION
 * *0 0034 0C03D   _          ……  ｽﾍﾟｰｽを入力すると1ｽﾃｯﾌﾟ 実行する
 * *0 003D 11001   $$

BRK>
```

第７章　SEボード用PROMの作成

IE-17K-ETで修正したプログラムは，.XS0あるいは.XS1を使用して，AS17Kの出力するPROMファイル形式のHEXコードをチャネル０あるいはチャネル１に出力できます。したがって，チャネル１にPROMプログラマを接続しておけば，SEボード用のPROMを作成することができます。

保守／廃止

第8章　エラー・メッセージ

IE-17K-ETが発生するエラー・メッセージには，コマンドの使用方法の誤りに対するメッセージとIE-17K-ETのハードウエアの動作異常が起こった場合のメッセージがあります。

8.1　コマンド・エラー

コマンド・エラーはコマンド入力時にコマンド名の誤りやアーギュメント数の不一致があったとき表示されるエラー・メッセージです。以下にエラー・メッセージと意味を示します。

(1)　? ANF ARY NOT FOUND

アレイが見つかりません。

正しいアレイを指定してください。

(2)　? ARG IMPROPER ARGUMENT

アーギュメントが間違っています。

正しいアーギュメントを入力してください。

(3)　? ASC ARGUMENT STACK COUNTER ERROR

アーギュメントの数が間違っています。

アーギュメントの数が正しくなるように各コマンドを調整してください。

(4)　? AST ARGUMENT STACK OVERFLOW

アーギュメントの数が多すぎます。

アーギュメントの数を減らしてください。

(5)　? AUN ARGUMENT STACK UNDERFLOW

アーギュメント・スタックに値がありません。

アーギュメント・スタックに値を代入してください。

(6)　? ILL ILLEGAL COMMAND

サポートされていないコマンドを使用しているか，または使い方が間違っています。

正しいコマンドを使用してください。

保守／廃止

(7)　?INA ILLEGAL NUMBER OF ARGUMENTS

マクロ・コマンドのアーギュメント数が少なくなっています。

マクロ・コマンドのアーギュメントを正しく入力してください。

(8)　?IPE INPUT ERROR

.CCコマンドで不適当な値を設定しました。

正しい値を設定し直してください。

(9)　?IQN ILLEGAL Q-REGISTER NAME

Qレジスタ名が間違っています。

正しいQレジスタ名を使用してください。

(10)　?IVA INVALID ARGUMENT

アーギュメントに不適当な値が指定されています。

正しいアーギュメントを指定してください。

(11)　?IWT ILLEGAL WAIT TIME VALUE

Zコマンドのアーギュメントが間違っています。

正しい値を指定してください。

(12)　?MLA MISSING<

ループにおいて'<'が'>'の数より少なくなっています。

<　>の対応を正しくしてください。

(13)　?MLP MISSING)

'('が')'の数より少なくなっています。

(　)の対応を正しくしてください。

(14)　?MNF MACRO COMMAND NOT FOUND

'.'から始まる文字列が組み込みマクロ・コマンドとして存在しません。

正しい組み込みマクロ・コマンドを入力してください。

(15)　?MRA MISSING>

ループにおいて'>'が'<'の数より少なくなっています。

<　>の対応を正しくしてください。

保守／廃止

(16)　? MRP MISSING (

') ' が' (' の数より少なくなっています。

(　) の対応を正しくしてください。

(17)　? MVQ NO VALUE IN Q-REGISTER

マクロ定義されていないQレジスタをマクロとして実行しました。

マクロ定義した正しいQレジスタ名を指定してください。

(18)　? NAE NO ARGUMENT BEFORE=

=コマンドのアーギュメントがありません。

アーギュメントを入力してください。

(19)　? NAQ NO ARGUMENT BEFORE"

" コマンドのアーギュメントがありません。

アーギュメントを入力してください。

(20)　? NVQ NO VALUE IN Q-REGISTER

Qレジスタに値（数値または文字列）がありません。

Qレジスタに値を代入してください。

(21)　? POS INVALID ADDRESS

アドレス指定が対象製品のプログラム・メモリ・アドレスの範囲を越えています。

アドレス指定の最終プログラム・メモリ・アドレス以内の値にしてください。

(22)　? QNF QUARTATION NOT FOUND

' が見つかりません。

' を入力してください。

(23)　? QRO Q-REGISTER AREA OVERFLOW

Qレジスタ領域がいっぱいです。

不要なQレジスタの内容を削除してください。

(24)　? QST Q-STACK OVERFLOW

Qスタック領域がいっぱいです。

不要なQレジスタの内容を削除してください。

保守／廃止

(25)　? QUN Q-STACK UNDERFLOW

Qスタック領域に値がありません。

Qスタック領域に値を代入してください。

(26)　? RNE CPU RUN ERROR

プログラムの実行中，実行できないコマンドを入力しました。

.BKコマンドでプログラムを停止させてもう一度コマンドを入力してください。

(27)　? RNG NUMERIC RANGE OVER

数値が有効範囲を越えています。

数値を有効範囲内にしてください。

(28)　? RSE CPU RESET ERROR

プログラムの実行中，実行できないコマンドを入力しました。

.BKコマンドでプログラムを停止させてもう一度コマンドを入力してください。

(29)　? RTE CPU RESET ERROR

プログラムの実行中，実行できないコマンドを入力しました。

.BKコマンドでプログラムを停止させてもう一度コマンドを入力してください。

(30)　? SYN INVALID SYNTAX

文法が間違っています。

正しい文法を使用してください。

(31)　? TAG MISSING TAG　! XXX !

タグが見つかりません。

正しいタグを指定してください。

(32)　? UPE UNPRINTABLE ERROR

CLICE-ETシステムに用意されていないエラーが発生しました。

システムに異常があります。再起動してください。

(33)　? UPO UNTERMINATED POINTER @ OR _

@コマンドまたは_コマンドが↑Vで終端されていません。

↑Vで終端してください。

保守／廃止

(34)　? UTG UNTERMINATED TAG

タグが！で囲まれていません。

タグは！で囲んでください。

(35)　? WRE WRITE ERROR

メモリに書き込みを行うときにベリファイ・エラーが発生しました。

もう一度コマンドを実行し直してください。

8.2　ハードウエア・エラー

ハードウエア・エラーは，プログラム実行時にIE-17K-ETが異常動作した場合に表示されるエラー・メッセージです。以下にメッセージと意味を示します。

(1)　SYSTEM REGISTER ACCESS ERROR

ARレジスタを除くシステム・レジスタで実装されていないビットをセットしたときに表示されます。

(2)　STACK OVER/UNDER FLOW

スタック・ポインタがオーバフローかアンダフローしたとき表示されます。

(3)　RAM NOT INITIALIZE

ポートなどを除くメモリで一度も書き込みを行っていないデータ・メモリあるいは，初期値の決まっていないデータ・メモリを読み出す命令を実行したとき表示されます。

(4)　ILLEGAL RAM WRITE

存在しないデータ・メモリに書き込みを行った場合に表示されます。

(5)　?IOS INVALID OPTION SWITCH AT 0000

IE-17K-ETにプログラムをロードしたとき，あるいはプログラムの実行時にアセンブラで記述したオプション・スイッチの指定がSEボード上のオプション・スイッチの設定と異なるとき表示されます。最後の数字は設定の異なるSEボードのスイッチの番号です。

(6)　?ISE INVALID SE BOARD NUMBER [00-00]

IE-17K-ETにプログラムをロードしたときあるいはプログラムの実行時にアセンブラで使用したデバイス・ファイルとSEボードが異なるとき表示されます。SEボードの装着状態が悪いときにも表示されます。最後の数字の左側はSEボードの番号，右側はデバイス・ファイルに含まれる番号です。

(7)　?IDI INVALID DEVICE ID NUMBER [00-00]

IE-17K-ETにプログラムをロードしたときにアセンブラで使用したデバイス・ファイルとSEボード上のデバイスが異なるとき表示されます。SEボード上のデバイスの装着状態が悪いときにも表示されます。最後の数字の左側はSEボード上のデバイスの番号，右側はデバイス・ファイルに含まれる番号です。

保守／廃止

(8) ----- NO SWITCH OPTION -----

IE-17K-ETにプログラムをロードするとき，オプション情報が正常にロードされなかった場合に表示されます。

(9) PC ERROR !

SEボード上のデバイスの異常動作により期待するとおりにプログラム・カウンタが動作しなかったときに表示されます。

次の(10)から(15)に示すエラー・メッセージは，電源投入時またはリセット時に行われるIE-17K-ETの自己診断結果にエラーがあった場合に表示されます。これらのエラー・メッセージが表示された場合は，ハードウエアが故障しているので修理が必要です。

(10) MEMORY ERROR → 0000 : 0000-7000 : FFFF

IE-17K-ETが使用するメモリに異常があるときに表示されます。

(11) DEVICE ERROR → PTC (UPD71054) #0

プログラマブル・タイマ0 (μPD71054) に異常があるときに表示されます。

(12) DEVICE ERROR → PTC (UPD71054) #1

プログラマブル・タイマ1 (μPD71054) に異常があるときに表示されます。

(13) DEVICE ERROR → SCU (UPD71051) #0

シリアル・コントロール・ユニット0 (μPD71051) に異常があるときに表示されます。

(14) DEVICE ERROR → SCU (UPD71051) #1

シリアル・コントロール・ユニット1 (μPD71051) に異常があるときに表示されます。

(15) DEVICE ERROR → ICU (UPD71059)

割り込みコントローラ (μPD71059) に異常があるときに表示されます。

次の(16)から(21)に示すエラー・メッセージは，IE-17K-ETのCPUが暴走などの異常動作をしたときに表示されます。

(16) <<DIVIDE BY ZERO>>

'0'で割算を行った。

(17) ＜＜CHECK FIELD＞＞

メモリ境界をオーバした。

(18) ＜＜SINGLE STEP＞＞

シングル・ステップを行った。

(19) ＜＜BREAK MODE＞＞

ブレーク命令を実行した。

(20) ＜＜OVERFLOW＞＞

演算時にオーバフローした。

(21) ＜＜NMI＞＞

NMIが発生した。

保守／廃止

付録A　プリミティブ・コマンド

プリミティブ・コマンドは，ユーザ・オリジナルなマクロ命令を作る際に使用します。ただし，IE-17K-ETのCLICE-ET 1.6は，このプリミティブ・コマンドを組み合わせた組み込みマクロ命令を52命令サポートしています。したがって，通常の作業をされる場合，プリミティブ・コマンドは特に必要ありません。組み込みマクロ・コマンドをさらに充実させたいときに，この章を読んでください。

保守／廃止

A.1　プリミティブ・コマンド一覧

表A-1　コマンド一覧表（1/2）

コマンド	ASCII コード	機　　能	コマンド	ASCII コード	機　　能
↑@	00	ダミー	SPACE	20	ダミー
↑A	01	文字列表示	!	21	タグ
↑B	02	2進定数プリフィクス	"	22	条件分岐
↑C	03	強制停止	#	23	論理和演算子
↑D	04	10進定数プリフィクス	$	24	ダミー
↑E	05	エディタの起動	%	25	Qレジスタのインクリメント
↑F	06	アレイ・アドレス関数	&	26	論理積演算子
↑G	07	現在行の表示	'	27	条件分岐
↑H	08	1文字削除	(	28	優先順位子
↑I	09	ダミー	)	29	優先順位子
↑J	0A	ダミー	*	2A	積算演算子
↑K	0B	〈未定義〉	+	2B	加算演算子
↑L	0C	〈未定義〉	,	2C	ダミー
↑M	0D	ダミー	−	2D	減算演算子
↑N	0E	〈未定義〉	.	2E	組み込みマクロ・プリフィクス
↑O	0F	〈未定義〉	／	2F	除算演算子
↑P	10	14344定数プリフィクス	0	30	数値定数
↑Q	11	ユーザ定義関数呼び出し	1	31	数値定数
↑R	12	制御文字入力プリフィクス	2	32	数値定数
↑S	13	画面表示の一時中断	3	33	数値定数
↑T	14	キー入力関数	4	34	数値定数
↑U	15	1行削除	5	35	数値定数
↑V	16	ポインタ	6	36	数値定数
↑W	17	〈未定義〉	7	37	数値定数
↑X	18	1行削除	8	38	数値定数
↑Y	19	〈未定義〉	9	39	数値定数
↑Z	1A	〈未定義〉	:	3A	アーギュメント・ポップ
↑[	1B	ダミー	;	3B	ループの強制終了
↑¥	1C	〈未定義〉	<	3C	ループの開始
↑]	1D	〈未定義〉	=	3D	表示コマンド
↑^	1E	剰余演算子	>	3E	ループの終了
↑_	1F	〈未定義〉	?	3F	エラーの表示

保守／廃止

表A-1　コマンド一覧表 (2/2)

コマンド	ASCII コード	機　　能	コマンド	ASCII コード	機　　能
@	40	8ビット・ポインタ		60	〈未定義〉
A	41	16進定数		61	〈未定義〉
B	42	16進定数		62	〈未定義〉
C	43	16進定数		63	〈未定義〉
D	44	16進定数		64	〈未定義〉
E	45	16進定数		65	〈未定義〉
F	46	16進定数		66	〈未定義〉
G	47	〈未定義〉		67	〈未定義〉
H	48	〈未定義〉		68	〈未定義〉
I	49	〈未定義〉		69	〈未定義〉
J	4A	〈未定義〉		6A	〈未定義〉
K	4B	〈未定義〉		6B	〈未定義〉
L	4C	〈未定義〉		6C	〈未定義〉
M	4D	マクロ定義コマンド		6D	〈未定義〉
N	4E	〈未定義〉		6E	〈未定義〉
O	4F	GOTO		6F	〈未定義〉
P	50	〈未定義〉		70	〈未定義〉
Q	51	Qレジスタ・プリフィクス		71	〈未定義〉
R	52	〈未定義〉		72	〈未定義〉
S	53	〈未定義〉		74	〈未定義〉
T	54	〈未定義〉		74	〈未定義〉
U	55	Qレジスタ代入		75	〈未定義〉
V	56	〈未定義〉		76	〈未定義〉
W	57	〈未定義〉		77	〈未定義〉
X	58	アレイ代入		78	〈未定義〉
Y	59	〈未定義〉		79	〈未定義〉
Z	5A	ウェイト		7A	〈未定義〉
[	5B	Qレジスタのポップ	{	7B	左シフト演算子
¥	5C	数値関数	|	7C	排他的論理和演算子
]	5D	Qレジスタのプッシュ	}	7D	右シフト演算子
^	5E	制御文字プリフィクス	~	7E	否定演算子
_	5F	16ビット・ポインタ	DEL	7F	1文字削除

A

保守／廃止

A.2　アレイ一覧

表A－2に示すアレイを参照する場合，↑FVER＝H（CLICE-ETのバージョンを16進表現で表示する）などを使用してください。

表A－2　アレイ一覧表（1/2）

アレイ名	機　　能
VER	CLICEのバージョン
WRK	CLICEのワーク・エリア先頭アドレス
PRM	プログラム・メモリ先頭アドレス
TRM	トレース・メモリ先頭アドレス（現在のバンクに注意すること） BANK#0：PC，BANK#1：PORT，BANK#2：WA，DB，JG，BANK#3：PC， BANK#4：TM0，BANK#5：TM1
PCV	PCカバレージ・メモリ先頭アドレス
DCV	データ・カバレージ・メモリ先頭アドレス
CND	コンディション・メモリ先頭アドレス
ERG	エミュレータ・レジスタ先頭アドレス
CRG	コンディション・レジスタ先頭アドレス
CR0	コンディション・レジスタ・ユニット#0
CR1	コンディション・レジスタ・ユニット#1
CR2	コンディション・レジスタ・ユニット#2
CR3	コンディション・レジスタ・ユニット#3
SRG	システム・レジスタ先頭アドレス
DTM	データ・メモリ先頭アドレス
RGF	レジスタ・ファイル先頭アドレス
RNI	RAM NOT INITIALIZEブレーク・レジスタ
SPE	STACK OVERFLOW/UNDERFLOWブレーク・レジスタ
IRW	ILLEGAL RAM WRITEブレーク・レジスタ
PC□	システム・レジスタ　PCアドレス
SP□	システム・レジスタ　SPアドレス
AR0	システム・レジスタ　AR0アドレス
AR1	システム・レジスタ　AR1アドレス
AR2	システム・レジスタ　AR2アドレス
AR3	システム・レジスタ　AR3アドレス
WR□	システム・レジスタ　WRアドレス
BNK	システム・レジスタ　BANKアドレス
IXH	システム・レジスタ　IXHアドレス

備考　□はスペースを表します。

表A-2　アレイ一覧表（2/2）

アレイ名	機　　能
IXM	システム・レジスタ　IXMアドレス
IXL	システム・レジスタ　IXLアドレス
RPH	システム・レジスタ　RPHアドレス
RPL	システム・レジスタ　RPLアドレス
PSW	システム・レジスタ　PSWアドレス
JG□	システム・レジスタ　JGアドレス
ICU	INTERRUPT CONTROL　(μPD71059)
TC0	TIMER CONTROL#0　(μPD71054)
TC1	TIMER CONTROL#1　(μPD71054)
SC0	SERIAL CONTROL#0　(μPD71051)
SC1	SERIAL CONTROL#1　(μPD71051)
DSW	DIP SWITCH
RES	RESET Mam' Chip
SEN	SE BOARD NUMBER

保守／廃止

A.3　コンディション・レジスタ・オフセット・アドレス

表A－3はアレイCRGに対するオフセット・アドレスを示しています。

アレイを参照する場合には，↑FCRG＋↑FDP0＝H（ブレーク条件のDEPTH0の条件を格納するアドレスを16進表現で表示する）などを使用してください。

表A－3　コンディション・レジスタ・オフセット・アドレス

アレイ名	機　　能
DP0	ブレーク条件　DEPTH0
DP1	ブレーク条件　DEPTH1
DP2	ブレーク条件　DEPTH2
DP3	ブレーク条件　DEPTH3
LVC	ブレーク条件　DEPTHカウンタ
CNB	条件ブレーク許可フラグ
RIB	「RAM NOT INITIALIZE」ブレーク許可フラグ
IWB	「ILLEGAL RAM WRITE」ブレーク許可フラグ
SOB	「SP OVER/UNDERFLOW」ブレーク許可フラグ
TTM	トレース・タイマ
TTP	トレース・タイマ・プリスケーラ
TAD	ステータス・トレース／アドレス・トレース切り替えフラグ
PDB	PREVIOUS DATAブレーク許可フラグ
PCA	PCブレーク・アドレス
PCB	PCブレーク許可フラグ

A.4　コンディション・ユニット・レジスタ・オフセット・アドレス

表A－4はアレイCR0～CR3に対するオフセット・アドレスを示しています。

アレイを参照する場合には，↑FCR0＋↑FPAU＝H（プログラム・アドレスのブレーク／トレース条件範囲の上限を格納するユニット0のアドレスを16進表現で表示する）などを使用してください。

表A－4　コンディション・ユニット・レジスタ・オフセット・アドレス（1/2）

アレイ名	機　　能
PAU	プログラム・アドレスのブレーク／トレース条件範囲の上限
PAL	プログラム・アドレスのブレーク／トレース条件範囲の上限
PAM	プログラム・アドレスのブレーク／トレース条件範囲のMATCH／UNMATCH指定
DTA	データ・メモリ・アドレスのブレーク／トレース条件
DMK	データ・メモリ・アドレスのブレーク／トレース条件のマスク
DTU	データ・メモリ・アドレスのブレーク／トレース条件のMATCH／UNMATCH指定
CRD	データ・メモリ書き込みデータのブレーク／トレース条件
CRM	データ・メモリ書き込みデータのブレーク／トレース条件のマスク
CRU	データ・メモリ書き込みデータのブレーク／トレース条件のMATCH／UNMATCH指定
PDT	データ・メモリに書き込まれる前のデータのブレーク／トレース条件
PDM	データ・メモリに書き込まれる前のデータのブレーク／トレース条件のMATCH／UNMATCH指定
SPU	スタック・ポインタのブレーク／トレース条件範囲の上限
SPL	スタック・ポインタのブレーク／トレース条件範囲の下限
SPM	スタック・ポインタのブレーク／トレース条件範囲のMATCH／UNMATCH指定
ISD	指定した命令の実行によるブレーク／トレース条件
ISM	指定した命令の実行によるブレーク／トレース条件のマスク
ISU	指定した命令の実行によるブレーク／トレース条件のMATCH／UNMATCH指定
MAR	MAR（モニタ・アドレス・レジスタ）で指定したデータのブレーク／トレース条件
MAM	MARで指定したデータのブレーク／トレース条件のマスク
MAU	MARで指定したデータのブレーク／トレース条件のMATCH／UNMATCH指定
INT	インタラプトによるブレーク／トレース条件
INM	インタラプトによるブレーク／トレース条件のマスク
INU	インタラプトによるブレーク／トレース条件のMATCH／UNMATCH指定
DMA	DMAによるブレーク／トレース条件
DMM	DMAによるブレーク／トレース条件のマスク
DMU	DMAによるブレーク／トレース条件のMATCH／UNMATCH指定
AND	ブレーク／トレース条件のAND／OR指定
CNT	ブレーク／トレース条件の成立回数カウント条件
CND	ブレーク／トレース条件の成立回数によるブレーク／トレース条件
CNM	ブレーク／トレース条件の成立回数のマスク

表A-4　コンディション・ユニット・レジスタ・オフセット・アドレス（2/2）

アレイ名	機　　能
CDR	現在のブレーク／トレース条件の成立回数
CNU	ブレーク／トレース条件の成立回数のMATCH／UNMATCH指定
TRS	トレース条件のON／OFF条件の指定

A.5　プリミティブ・コマンド説明

この節では，プリミティブ・コマンドの機能について説明します。プリミティブ・コマンドは，IE-17K-ETを使ってプログラム開発を行った経験者に対し，より高度なディバグ方法を提供します。

また，IE-17K-ETに新しいコマンド（マクロ・コマンド）を追加したり，一連のディバグ手順をプログラムしたりできます。

A.5.1　ポインタ

ポインタは，アレイの要素が示す資源を参照する（読み出す）ものです。

(1)　バイト・ポインタ

@↑V

(2)　ワード・ポインタ

_↑V

保守／廃止

@↑V　バイト・ポインタ

書　式　：　@α↑V
α：式

〔機　能〕　アレイの要素から1バイト読み出します。

バイト・ポインタは，@と↑Vの間に記述された式の評価値で示されるアレイの要素から1バイト（8ビット）分のデータを読み出します。

（例）　@100↑V　　…アレイの要素番号100Hの内容を参照。

@↑FDTM↑V　…データ・メモリの先頭番地（0番地）の内容を参照。

ここで↑FDTMはデータ・メモリの先頭番地を示すアレイ・アドレスを返す関数です。

保守／廃止

_↑V　ワード・ポインタ

書　式　：　_α↑V
α：式

〔機　能〕　アレイ要素から1ワード読み出します。

ワード・ポインタは，_と↑Vの間に記述された式の評価値で示されるアレイの要素から1バイト（8ビット）分のデータを読み出しこれを上位とし，評価値＋1で指定されるアレイの要素から1バイト分のデータを読み出しこれを下位とした1ワード(16ビット)分のデータを読み出します。

（例1）　_↑FDTM↑V　…データ・メモリの先頭番地（0番地）の内容を上位バイトとし，1番地の内容を下位バイトとする16ビット・データを参照。
ここで↑FDTMはデータ・メモリの先頭番地を示すアレイ・アドレスを返す関数です。

（例2）　_↑FPRM＋（10H｛1)↑V＝P　…プログラム・メモリの10H番地の内容を表示します。
ここで↑FPRMはプログラム・メモリの先頭番地を示すアレイ・アドレスを返す関数です。

保守／廃止

A.5.2　関　　数

CLICE-ETには，システムに組み込まれた「組み込み関数」とユーザが定義する「ユーザ定義関数（↑Qコマンド)」の2種類があります。

組み込み関数には「アレイ・アドレス関数（↑Fコマンド)」，「キー入力関数（↑Tコマンド)」，「数値関数（¥コマンド)」などがあります。

(1)　ユーザ定義関数

↑Q

(2)　アレイ・アドレス関数

↑F

(3)　キー入力関数

↑T

(4)　数値関数

¥

保守／廃止

↑Q　ユーザ定義関数

書　式　：　↑Qα
α：Qレジスタ名

〔機　能〕　Qレジスタの内容を実行し，値を返します。

↑Qコマンドは，αで示されるQレジスタに格納されたコマンド列（文字列）を実行し，この中で定義された値を返します。

このコマンドの詳しい説明は，**付録A.5.6　マクロ**を参照してください。

保守／廃止

↑F　アレイ・アドレス関数

書　式　：　↑Fα
α：アレイ要素名

〔機　能〕　アレイ要素名のアレイ・アドレスを返します。

↑Fコマンドは，それに続く3文字で指定されたアレイ要素名αが示すアレイ・アドレスを返します。

付録A.2にアレイ要素名一覧を示します。

(例)　↑FPRM

これは，プログラム・メモリの0番地に対応するアレイ・アドレスを返します。したがって，プログラム・メモリの番地はアレイ・アドレスを使用すると次のようになります。

↑FPRM＋1 …0番地の下位8ビット

↑FPRM＋n …nが偶数のとき：n／2番地の上位8ビット

nが奇数のとき：(n－1)／2番地の下位8ビット

A

保守／廃止

↑T　キー入力関数

書　式　：　↑T

〔機　能〕　キー入力された文字のコードを返します。

キー入力された文字のASCIIコードを返します。もし，キー入力がない場合にはキーが入力されるまで待ちます。

また，↑Tコマンド実行中であっても↑Cキーが2回連続して入力されると↑Tコマンドは中断されます。

（例）　↑TU1↑TU2↑TU3
①　②　③

上記のコマンド列が実行されると，まず①の↑Tでキー入力待ちになり，ここで↑Cをキー入力すると，①の↑Tは03Hを返し，Q1に03Hが代入されます（**付録A.5.3　代入　Uコマンド（Qレジスタへの代入）**参照）。
そして，コマンドの実行は②の↑Tに移り，ここで↑Cをキー入力すると，コマンド列の実行は，②の↑Tで中断されます。
つまり，↑Tでは2つの連続する↑Cは入力することはできません。

入力された文字は自動的に表示されます。

保守／廃止

¥　数値関数

書　式　：　¥α
α：Qレジスタ名

〔機　能〕　文字列を数値に変換します。

αで指定されるQレジスタに格納された文字列が表現する数値を返します。

このとき，数値の桁数は指定されたQレジスタの先頭から最初に数値以外の文字が現れる所までか，Qレジスタの先頭からQレジスタの最後の文字までかのいずれかです。

（例）Qレジスタ１の内容が次のような場合，¥1は，

①　↑B1234　…１を返す（２は２進数ではない）。

②　BACK　…0BACHを返す（Kは16進数ではない）。

③　↑D16＊2　…10Hを返す（＊は10進数ではない）。

次の例のように１文字も数値として変換できない場合や，定数として表現できる範囲を超える場合はエラーになります。

④　↑DECIMAL　…エラー

⑤　12345678901234567890　…エラー

A

A.5.3　代　　入

CLICE-ETには変数への数値または文字列の代入のためのコマンドとして，Uコマンドおよび XB，XC，XWコマンドが用意されています。

(1)　Qレジスタへの代入

U

(2)　アレイへのバイト・データ代入

XB

(3)　アレイへのワード・データ代入

XW

(4)　アレイへの文字列代入

XC

保守／廃止

U　Qレジスタへの代入

書　式　：　①　$\beta U\alpha$
②　$U\alpha\gamma\$$ (マクロの定義)
③　δ, $\varepsilon U\alpha$

α：Qレジスタ名
β：式
γ：文字列
δ：アレイ先頭アドレス
ε：アレイ最終アドレス

〔機　能〕　Qレジスタに数値や文字列を代入します。
書式①は，式βの評価値をαで指定されるQレジスタに代入します。

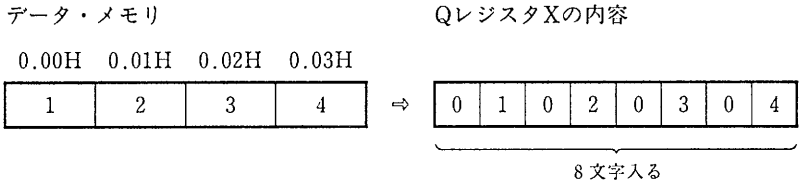
(例)　3UA　　…QレジスタAに3を代入。
@↑FDTM+1↑VU3　…データ・メモリの1番地の内容をQレジスタ3に代入。

書式②は，$までの文字列$\gamma$を$\alpha$で指定されるQレジスタに代入します。
文字列γとしてコマンド列を代入するとマクロとして使用することができます。

(例)　U1ABC$　…Qレジスタ1にABCを代入。

書式③は，式δの評価値で指定されるアレイ・アドレスから式εの評価値で指定されるアレイ・アドレスまでの内容を文字列としてαで指定されるQレジスタに代入します。

(例)　↑FDTM，↑FDTM+3UX　…データ・メモリの0番地から，3番地までの内容をQレジスタXに文字列として代入。



保守／廃止

XB　アレイへのバイト・データ代入

書　式　：　α, βXB
α：式（下位8ビットのみ有効） β：アレイ・アドレス

〔機　能〕　アレイβに8ビットのデータαを代入します。

式αの評価値の下位8ビットを式βの評価値が示すアレイ・アドレスの要素へ代入します。

（例1） Q1, ↑FDTMXB　…Qレジスタ1の下位8ビットをデータ・メモリの0番地へ代入。

（例2） 次にキー入力された文字をアレイのワーク領域にストアするプログラム例を示します。

```
↑FWRKUP < ↑TU2   D−Q2 ; Q2, QPXB      %P>
QP=^FWRK  Q2=key  if CR ? ARRAY (Q1)=Q2 QP=QP+1
```

この例は，キー入力された文字がASCIIコードの0DH以下の場合ループから抜け出すプログラムです（**付録A.5.7　制御　< > コマンド（ループ）**も参照してください）。

保守／廃止

XW　アレイへのワード・データ代入

書　式　：　α, βXW
α：式（下位16ビットのみ有効） β：アレイ・アドレス

〔機　能〕　アレイβに16ビットのデータαを代入します。

式αの評価値の下位16ビットを式βの評価値が示すアレイ・アドレスの要素へ代入します。

（例） Q1, ↑FPRMXW　…Qレジスタ1の下位16ビットをデータ・メモリの0番地と1番地へ代入します。

この例は，XBコマンドを使って次のように記述することができます。

Q1} 8, ↑FDTMXBQ1, ↑FDTM+1XB

A

保守／廃止

XC　アレイへの文字列代入

書　式　：　βXCα
α：Qレジスタ名 β：アレイ・アドレス

〔機　能〕　アレイβにQレジスタαに格納されている文字列を代入します。

αで指定されるQレジスタに格納されている文字列を式βの評価値が示すアレイ・アドレスの要素から順にその文字数分のアレイの範囲に代入します。

（例）　U1ABC①$↑FWRKXC1②

この例は，①でQレジスタ1にABCという文字列を代入し，②でその値をアレイのワーク領域の先頭から順にA，B，Cと代入します。

つまり，アレイ・アドレスの↑FWRKには，A，↑FWRK+1にはB，↑FWRK+2にはCがそれぞれ代入されます。

また，この例はXBコマンドを使って次のように記述できます。

41，↑FWRKXB42，↑FWRK+1XB43，↑FWRK+2XB

ただし，41H，42H，43HはそれぞれASCIIコードでA，B，Cを示しています。

保守／廃止

A.5.4　アーギュメント・スタック

CLICE-ETには,コマンドの数値アーギュメントの受け渡し用一時記憶としてアーギュメント・スタックがあります。

数値アーギュメントは，定数，関数値などの式の評価値からなり，これがアーギュメント・スタックに格納され，コマンドにより読み出されて使用されます。

次にアーギュメント・スタックの使われ方を説明します。

3，4＊2，5＋1＄

数値アーギュメントの3がアーギュメント・スタックにプッシュされ，カンマ（,）で区切られた次の数値アーギュメント8と6も順にアーギュメント・スタックにプッシュされます。

そして，コマンドがアーギュメント・スタックより6，8，3の順でポップして使用されます。

また，アーギュメント・スタックは，コマンド列の実行が終了すると初期化されます。

(1)　アーギュメントのプッシュ

　　数値

(2)　アーギュメントのポップ

　　：

保守／廃止

アーギュメント・プッシュ

書　式　：　α [, α]
α：式

〔機　能〕　アーギュメント・スタックに数値をプッシュ（push）します。
式の値をアーギュメント・スタックにプッシュします。
アーギュメント・スタックのレベルは+1（インクリメント）されます。

（例） 12, 34, 56 $

12Hと34Hと56Hが順にアーギュメント・スタックにプッシュされます。

保守／廃止

:　アーギュメント・ポップ

書　式　:　:α
α：Qレジスタ名

〔機　能〕　アーギュメント・スタックから数値をポップ（pop）し，Qレジスタに格納します。

アーギュメント・スタックから数値をポップして，αで指定されたQレジスタに格納します。

アーギュメント・スタックのレベルは−1（デクリメント）されます。

（例）
```
UM：A：BQA＊QB＝D$$
↑D12，3MM$$36
```

QレジスタMにコマンド列を格納して，次に12と3をアーギュメントとするマクロMを実行します。

この結果36が表示されます。

（使用しているコマンドについては**付録A.5.6　マクロ，付録A.5.8　表示**を参照してください）。

保守／廃止

A.5.5　Qスタック

Qスタックは，Qレジスタに格納されている内容をセーブするためのスタック機構です。

Qスタックはコマンド列の実行が終了すると初期化されます。

(1)　Qレジスタのプッシュ

[

(2)　Qレジスタのポップ

]

保守／廃止

[　Qレジスタのプッシュ

書　式　：　[α
α：Qレジスタ名

〔機　能〕　Qレジスタαの内容をQスタックへプッシュ（push）します。

αで指定されるQレジスタの内容（数値または文字列）をQスタックへプッシュします。

Qスタックのレベルは+1（インクリメント）されます。

（例）　34UN [N

QレジスタNに34Hを代入するとともにQスタックへ34Hをプッシュします。

A

保守／廃止

]　Qレジスタのポップ

書　式　：　] α
α：Qレジスタ名

〔機　能〕　Qスタックからポップ（pop）した内容をQレジスタαに格納します。

Qスタックからポップした内容（数値または文字列）をαで指定されるQレジスタに格納します。

Qスタックのレベルは−1（デクリメント）されます。

（例）] NQN＝H

QレジスタNにポップした内容を16進数値として表示します。

保守／廃止

A.5.6　マ ク ロ

マクロとはQレジスタに格納された文字列をコマンドとして実行させる機能です。

マクロの実行にはマクロ・コマンド（Mコマンド）とユーザ定義関数（↑Qコマンド）の2種類があります。

(1)　マクロ・コマンドの実行

M

(2)　ユーザ定義関数の実行

↑Q

保守／廃止

M　マクロ・コマンドの実行

書　式　：　Mα
α：Qレジスタ名

〔機　能〕　Qレジスタαの内容をコマンド列として実行します。

αで指定されるQレジスタに格納されている内容をコマンドとして実行します。

Qレジスタ内の文字列（コマンド）へのパラメータ（引き数）の受け渡し方法には，変数，Qスタック，アーギュメント・スタックを介して行います。

なお，マクロ・コマンドはネスティングすることができます。

以下に2つの引き数の加算結果を16進表示するマクロの例を示します。

（例1） 変数を介してのパラメータの受け渡し。

```
UMQ1+Q2=H$$
100U1200U2MM$$300
```

QレジスタMに，Qレジスタ1の内容とQレジスタ2の内容を加算して16進表示を行うコマンド列を格納します。

次に，Qレジスタ1とQレジスタ2にそれぞれ100Hと200Hを代入して，マクロMを実行することでパラメータの受け渡しができます。

その結果，300が表示されます。

上記の例と同一の動作をアレイを介して行うと次のようになります。

```
UM_↑FWRK↑V+_↑FWRK+2↑V=H$$
100,↑FWRKXW200,↑FWRK+2XWMM$$300
```

（例2） Qスタックを介してのパラメータの受け渡し。

変数を介してパラメータを受け渡す方法は，マクロがネスティングするような場合には，使用する変数（Qレジスタなど）が重複してしまうことがあります。

これを解決する1つの方法としてQスタックを使用する方法があります。

```
UM]1Q1+]1Q1=H$$
100U1[1 200U1[1 MM$$300
```

QレジスタMにQレジスタ1へポップした数値同士を加算をして16進表示を行うコマン

保守／廃止

ド列を格納します。

次に，Qレジスタ１に代入した数値をQスタックにプッシュし，マクロMを実行することでパラメータの受け渡しができます。

（例３）アーギュメント・スタックを介してのパラメータの受け渡し。

この方法がパラメータの受け渡しとしてはもっとも簡単な方法です。

```
UM：1Q1＋：1Q1＝H$$
100，200MM$$300
```

QレジスタMにQレジスタ１へポップした数値同士を加算をして16進表示を行うコマンド列を格納します。

次に，数値をアーギュメント・スタックにプッシュし，マクロMを実行することでパラメータの受け渡しができます。

保守／廃止

↑Q　ユーザ定義関数の実行

書　式　：　↑Qα
α：Qレジスタ名

〔機　能〕　Qレジスタαの内容をコマンド列（関数）として実行し，値を返します。

αで指定されるQレジスタに格納されている内容をコマンドとして実行します。

このコマンドの中で関数として返すべき値をQスタックにプッシュします。

こうすることにより，この関数の実行が完了した時点でQスタックから自動的に値が取り出されます。

Qレジスタ内のコマンド列へのパラメータ（引き数）の受け渡し方法はマクロ・コマンドと同様に，変数，Qスタック，アーギュメント・スタックを介して行います。

なお，ユーザ定義関数はネスティングすることができます。

（例） プログラム・メモリのアドレスを返す関数の定義例を示します。

この関数は，プログラム・メモリ・アドレス（1アドレス／16ビット）を入力とし，アレイ上の論理アドレス（1アドレス／8ビット）を返します。

```
UM：1↑FPRM+ (Q1*2) ↑VU1 [1$
```

このマクロMは，次のように使います。

```
_100↑QM↑V=H$$
```

これは，プログラム・メモリの100H番地の内容を16進表示します。

保守／廃止

A.5.7　制　　御

CLICE-ETは各コマンドの実行順序を制御することができます。

(1)　ループ

　　＜＞

(2)　ループの強制終了

　　；

(3)　タグ

　　！！

(4)　無条件分岐

　　0＄

(5)　条件分岐

　　”，

保守／廃止

<>　ループ

書　式　：　[α]　<β>
α：繰り返し回数 β：コマンド列

〔機　能〕　コマンド列βをα回繰り返し実行します。

αが$2^{32}-1$のときαを省略できます。

<>で囲まれたコマンド列βが，αで指定された数値アーギュメントの値だけ繰り返し実行されます。

ループはネスティングすることができます。

繰り返し回数αが省略された場合には$2^{32}-1$回繰り返します。

（例） 10<コマンド列A　5<コマンド列B>　コマンド列C>

この場合，コマンド列Aを1回実行し，コマンド列Bを5回実行し，コマンド列Cを1回実行するという処理が16回繰り返されます。

保守／廃止

；　ループの強制終了

書　式　：　α；
α：終了条件式

〔機　能〕　繰り返しループを強制終了します。

繰り返しループで指定された繰り返し回数の終了以前にαで指定される数値アーギュメントの値が0以上になった場合，コマンド列の繰り返しを終了しループの外に抜け出します。

すなわち，；を含むループの終了を表す＞のすぐ右側に記述されたコマンドに制御を移します。

また，αで指定される数値アーギュメントが0より小さい場合，このコマンドは無視されます。

ループ外での；コマンドはエラーになります。

（例）　100＜1F－↑T；コマンド列＞

ループを256回繰り返すか，制御キー（ASCIIコードの00H～1FH）が入力されるまで，コマンド列を繰り返し実行します。

A

保守／廃止

！！　タグ

書　式　：　！α！
α：タグ

〔機　能〕　コマンド列中のタグを示します。

コマンド列中に処理内容の説明のためのコメントを記述したり，Oコマンドの飛び先の指定を行います。

！に囲まれた文字列はコマンドの実行に影響を与えません。

（例）　↑FWRKU1！Q1＝WORK POINTER！

Qレジスタ１にアレイWRKのアドレスを代入します。

！で囲まれた文字列はコメントとして扱われ，コマンドとして実行されません。

保守／廃止

O$　無条件分岐

書　式　：　Oα$
α：タグ

〔機　能〕　無条件に指定したタグに制御を移します。

Oコマンドはその右側に記述された文字列（タグ）と同じ文字列の記述されたタグにコマンドの流れを変えます。

同一タグが記述された場合は，はじめに現れるタグが有効です。

（例）　!LOOP!コマンド列OLOOP$

コマンド列を無限に繰り返し実行します。

保守／廃止

”’ 条件分岐

書　式　：　α”β　γ’δ
α：条件式 β：条件 γ：コマンド列1 δ：コマンド列2

〔機　能〕　条件により実行するコマンドを変更します。

条件式αの値を判断して，コマンドの実行の流れを制御します。

αで指定された数値アーギュメントの値が ”の右側に記述された条件βを満足した場合，条件に続くコマンド列γを実行し，さらにコマンド列δを実行します。

もし条件を満足しない場合は，’に続くコマンド列δのみを実行します。

すなわちコマンド列γがスキップされます。

この条件分岐コマンドはネスティングすることができます。

条件として次の4種類が記述できます。

E：Equal zero　　（n＝0）

N：Not equal zero　　（n≠0）

L：Less than zero　　（n＜0）

G：Greater than zero　（n＞0）

（例） Q1−3” E0U2’Q1＋Q2U1

この例では，Q1−3が0（すなわちQ1＝3）ならば，Qレジスタ2に0を代入したあと，次のコマンドであるQ1＋Q2U1を実行します。

もし，Q1≠3ならば，0U2は実行せず ’の次のQ1＋Q2U1を実行します。

保守／廃止

A.5.8　表　　示

CLICE-ETには，文字や数値を表示するコマンドがあります。

(1)　数値の2進表示

＝B

(2)　数値の10進表示

＝D

(3)　数値の16進表示

＝H

(4)　Qレジスタの内容の文字表示

＝C

(5)　文字列表示

↑A

(6)　1−4−3−4−4ビット形式表示

＝P

保守／廃止

＝B　数値の２進表示

書　式　：　[β,] α＝B
α：数値データ β：表示桁数

〔機　能〕　数値データαで指定される数値アーギュメントの値を２進表示します。

βで指定される数値アーギュメントの値は表示桁数を示し，0または省略したとき先頭の0をサプレスして表示します。

（例１）　A＝B$$1010

0AHを２進表示します。このとき，＄＄により実行されますが改行されずにそのまま表示されます。

（例２）　8，A＝B$$00001010

0AHを８桁で２進表示します。

保守／廃止

＝D　数値の10進表示

書　式　：　[β,] α＝D
α：数値データ β：表示桁数

〔機　能〕　数値データαで指定される数値アーギュメントの値を10進表示します。

βで指定される数値アーギュメントの値は表示桁数を示し，0または省略したとき先頭の0をサプレスして表示します。

（例1）　A＝D$$10

0AHを10進表示します。このとき，＄＄により実行されますが改行されずにそのまま表示されます。

（例2）　4，A＝D$$0010

0AHを4桁で10進表示します。

保守／廃止

＝H　数値の16進表示

書　式　：　[β,] α＝H
α：数値データ β：表示桁数

〔機　能〕　数値データαで指定される数値アーギュメントの値を16進数で表示します。

βで指定される数値アーギュメントの値は表示桁数を示し，0または省略したとき先頭の0をサプレスして表示します。

（例1）　↑D10＝H$$A

10進数の10を16進表示します。このとき，＄＄により実行されますが改行されずにそのまま表示されます。

（例2）　4，A＝H$$000A

0AHを4桁で16進表示します。

保守／廃止

＝C　Qレジスタの内容の文字表示

書　式　：　＝Cα
α：Qレジスタ名

〔機　能〕　Qレジスタαに格納されている内容を文字列として表示します。

（例）　U1ABC$＝C1$$ABC

保守／廃止

↑A　文字列の表示

書　式　：　＝↑Aα↑A
α：文字列

〔機　能〕　↑Aで囲まれた文字列αを表示します。

（例）↑ABELL　↑R↑GBELL　TWICE↑R↑G↑R↑G↑A$$

BELLと表示されビープ音が鳴り，BELL TWICEと表示されビープ音が2回鳴ります。

保守／廃止

＝P　1-4-3-4-4ビット形式表示

書　式　：　α＝P
α：数値データ

〔機　能〕　αで指定される数値アーギュメントの値を1-4-3-4-4ビット形式で表示します。

（例）　↑B0011110011110000＝P$$078F0

0011110011110000B (NOP) を1-4-3-4-4形式で表示します。このとき，＄＄により実行されますが改行されずにそのまま表示されます。

A.5.9　そ の 他

CLICE-ETには前記のほかに以下のコマンドがあります。

(1) Qレジスタのインクリメント

%

(2) ウエイト

Z

(3) Qレジスタへの代入

*

(4) エラー表示

?

(5) デバイスからのデータ・メモリ読み出し

Y

(6) デバイスへのデータ・メモリ書き込み

W

保守／廃止

%　Qレジスタのインクリメント

書　式　：　%α
α：Qレジスタ名

〔機　能〕　Qレジスタの内容をイクリメント（+1）します。
Qレジスタαに格納されている内容を数値としてインクリメント（+1）します。

（例） 100U1%1

Qレジスタ1に100Hを代入したあと%コマンドでQレジスタ1の内容をインクリメント（+1）します。
この結果，Qレジスタ1の内容は101Hになります。

保守／廃止

Z　ウエイト

書　式　：　αZ
α：待ち時間

〔機　能〕　αで指定される数値アーギュメントの値が示す時間だけ，次のコマンドの実行を停止します。時間の単位は10 msecです。

（例）　↑D1000Z

10秒間実行を停止します。

保守／廃止

＊　Qレジスタへの代入

書　式　：　＊α
α：Qレジスタ名

〔機　能〕　コマンド・バッファの内容をQレジスタαへ代入します。

コマンド列入力の↑Cによる中断や，コマンド実行の↑C↑Cによる中断，あるいは，コマンド実行中のエラーによる中断によりプロンプトが表示されている場合に，このプロンプトに続く第1文字目に＊を入力しQレジスタを指定するとそのQレジスタへコマンド・バッファの内容を格納できます。

（例）　×××＞：1＿↑FPRM＋（Q1｛1）↑V=H$↑C　…　①
×××＞＊2　…　②
×××＞M2$$　…　③

①でコマンド列を入力し$でコマンドを終端し↑Cにより入力を中断します。
②で＊2でQレジスタ2に①のコマンド列が代入されます。
このとき，代入が正常に行われた場合には③のようにプロンプトが表示されます。Qレジスタ2に格納されたコマンド列はMコマンドによって実行できます。

このように，＊コマンドを使用することによりマクロを簡単に作成できます。

エラーによりコマンドの実行が中断した場合，＊コマンドでコマンド・バッファの内容をQレジスタに代入することにより，.EDコマンドでエラーを修正して再実行することができます。

この機能により，長いコマンド列を入力してエラーが発生したとしても，最初からコマンド列を再入力する必要がなく効率よくコマンド入力できます。

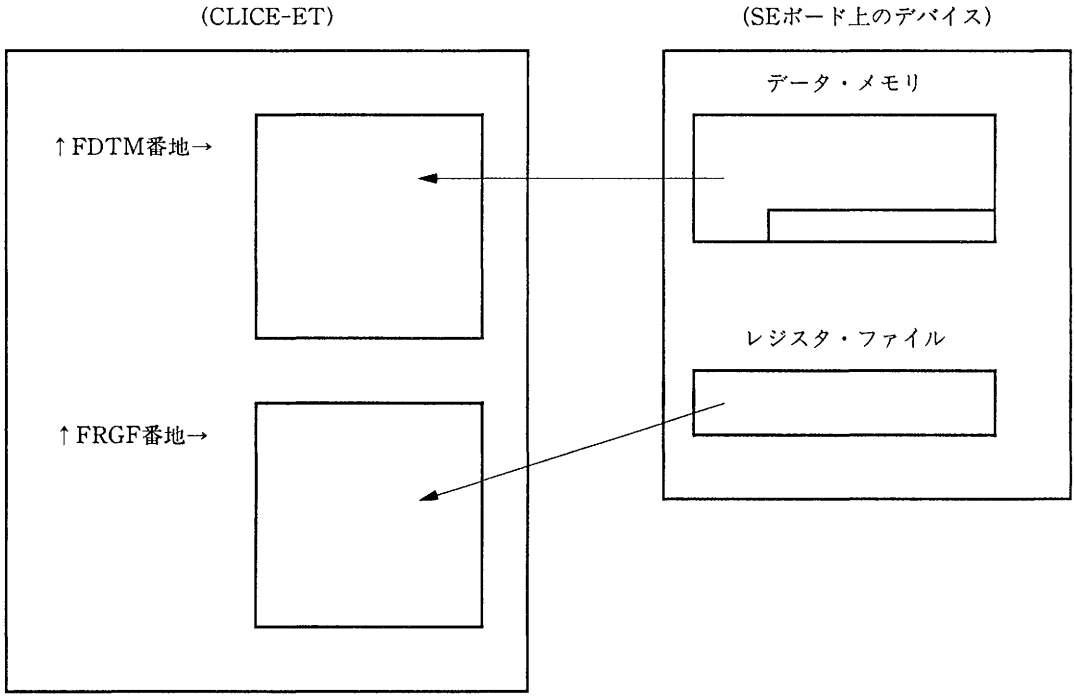
保守／廃止

Y　データ・メモリ読み出しコマンド

書　式　：　Y

〔機　能〕　SEボード上のデバイスのデータ・メモリとレジスタ・ファイルの内容をCLICE-ET上に読み出します。

SEボード上のデバイスのデータ・メモリの内容をアレイDTMで示されるアドレス以降に読み出します。また，レジスタ・ファイルの内容をアレイRGFで示されるアドレス以降に読み出します。



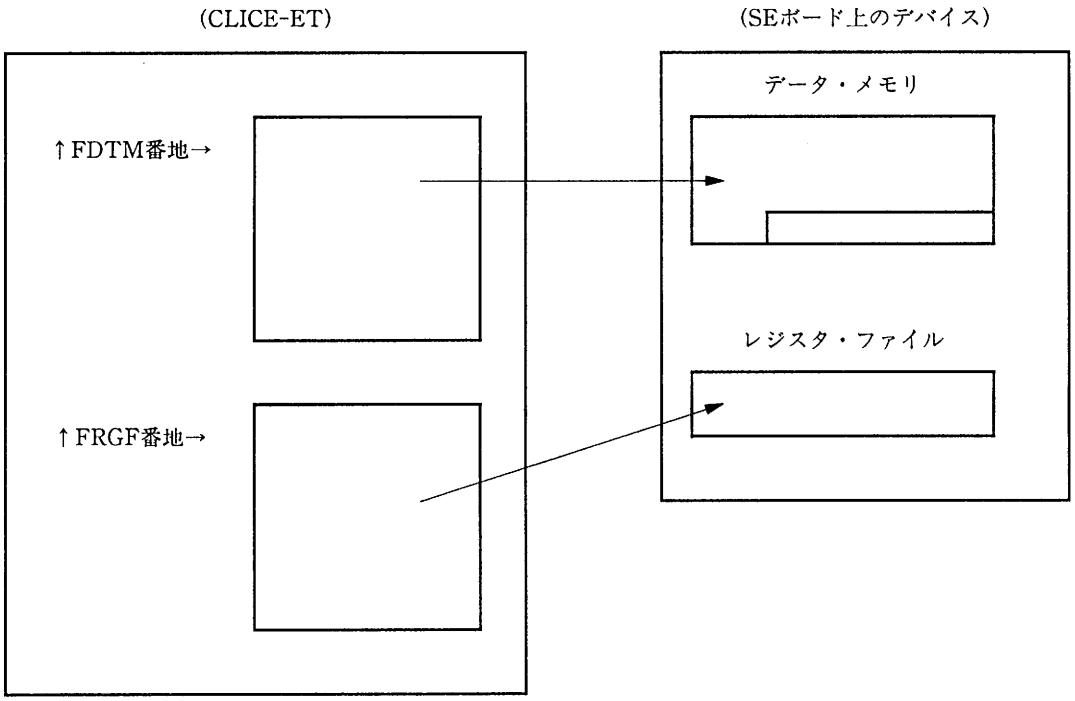
保守／廃止

W　データ・メモリ書き込みコマンド

書　式　：　W

〔機　能〕　CLICE-ET上のデータを，SEボード上のデバイスのデータ・メモリおよびレジスタ・ファイルに書き込みます。

アレイDTMで示されるアドレス以降のデータをSEボード上のデバイスのデータ・メモリに書き込みます。また，アレイRGFで示されるアドレス以降のデータをSEボード上のデバイスのレジスタ・ファイルに書き込みます。



保守／廃止

A.6　エディタ

CLICE-ETには，コマンド列を編集するエディタ機能があります。

エディタはコマンド列（文字列）に対してその一部または全部を追加，変更，削除することができます。

A.6.1　コマンド・バッファの編集

〔機　能〕　コマンド・バッファの編集をします。

〔書　式〕　↑E

コマンド列の入力中に↑Eを入力すると入力したコマンド列を編集するエディタ・モードになります。

エディタ・モード中では後述のエディタ・コマンドが使用できます。

（例）　×××>U1↑ASTRINGS↑A$↑E

>　　…エディタ・モードになります。

A.6.2　Qレジスタの編集

〔機　能〕　Qレジスタの内容の編集をします。

〔書　式〕　. EDα

αで指定されたQレジスタに格納された内容を文字列として編集するエディタ・モードになります。

エディタ・モード中では，後述のエディタ・コマンドを使用します。

（例）　×××>U1↑ASTRINGS↑A$$

×××>. ED1$$

>　　…エディタ・モードになります。

〔注　意〕

Qレジスタαが未定義の場合

?NVQ NO VALUE IN Q-REGISTER

というエラー・メッセージを表示します。

保守／廃止

A.6.3　エディタ・コマンド

エディタ・コマンドには移動，挿入，削除，検索，置換，表示，終了などがあります。

これらのコマンドの中にはコマンドの前にそのコマンドを実行する回数を指定できるものがあります。

この回数は10進数の２桁で指定し，０または省略されると１が指定されたことになります。

各エディタ・コマンドは英大文字１文字からなります。エディタ・コマンドを入力しても表示されませんので注意してください。

エディタ・コマンドがエラーのとき，ビープ音が鳴り入力したコマンドは無視されます。

(1)　カーサの移動

スペース・キーによりカーサは右に移動し編集対象の文字が表示されます。

BSキーまたはDELキーによりカーサは左に移動し編集対象の表示文字を消します。

この操作により編集したい位置にカーサを移動します。

編集はカーサが示す位置に対して行われます。

（例）　＞■　　カーサが先頭の文字を編集対象にしている。

＞S■　　スペース・キーによりカーサを右に移動すると文字列が表示されていく。

＞ST■

＞STRING■

＞STRIN■　　BSまたはDELキーによりカーサを左に移動すると表示していた文字が消える。

■は，カーサを示します。

(2)　挿　　入

〔コマンド〕　I

現在のカーサ位置にキー入力した文字列を挿入します。

ESCキーにより挿入が終了します。

（例）　＞ABCGHI■　　元の文字列。

＞ABC■　　カーサをGに移動する。

＞ABCDEF■　　IDEF（ESC）を入力。

＞ABCDEFGHI■　　カーサを移動すると文字列が挿入されたのが分かる。

A

〔コマンド〕　X

カーサを行の終端に移動したあと，Iコマンドと同じ動作をします。

ESCキーにより挿入が終了します。

(例)　＞■　　元の文字列。
　　＞ABCDEF■　　Xを入力すると文字列が最後まで表示される。
　　＞ABCDEFGHI■　GHI（ESC）を入力し挿入を終了する。

〔コマンド〕　P

現在のカーサ位置にキー入力した１文字を挿入します。

(例)　＞ABCEF■　元の文字列。
　　＞ABC■　　カーサをEに移動する。
　　＞ABCD■　　PDを入力。
　　＞ABCDEF■　カーサを移動すると文字が挿入されたのが分かる。

(3) 削　除

〔コマンド〕　D

現在のカーサ位置から１文字削除します。

回数が指定されている場合はその回数分の文字を削除します。

(例１)　＞ABCDEF■　　元の文字列。
　　＞ABCD■　　カーサをEに移動する。
　　＞ABCD¥E¥■　Dを入力すると削除された文字が表示される。
(例２)　＞ABCDEF■　　元の文字列。
　　＞ABC■　　カーサをDに移動する。
　　＞ABC¥DEF¥■　3Dを入力すると削除された３文字が表示される。

〔コマンド〕　H

現在のカーサ位置から行の終端までの文字を削除し，Iコマンドと同じ動作をします。

ESCキーにより挿入が終了します。

(例)　＞ABC■　　元の文字列。
　　＞¥ABC¥■　　Hコマンドにより削除された文字が表示される。
　　＞¥ABC¥DEF■　DEF（ESC）を入力

保守／廃止

(4) 検　索

〔コマンド〕　S

キー入力した１文字を現在のカーサ位置から検索し，見つかった位置にカーサを移動します。
見つからなかった場合，カーサは終端に移動してビープ音が鳴ります。
回数が指定されている場合は，その回数分の文字を検索します。

（例１）＞ABCDEFGHI■　元の文字列。
＞■　カーサを先頭に移動する。
＞ABCDEF■　SGを入力するとGにカーサが移動する。
（例２）＞ABCABCABC■　元の文字列。
＞■　カーサを先頭に移動する。
＞ABCABCA■　3SBを入力すると３番目のBにカーサが移動する。

(5) 置き換え

〔コマンド〕　C

キー入力した１文字を現在のカーサ位置の文字と置き換えます。
回数が指定されている場合，その回数分のキー入力により文字を置き換えます。

（例１）＞ABC＊EF■　元の文字列。
＞ABC■　カーサを＊に移動する。
＞ABCD■　CDを入力すると＊がDに置き換わる
（例２）＞ABC1234HI■　元の文字列。
＞ABC■　カーサを１に移動する。
＞ABCDEFG■　4CDEFGを入力すると1234がDEFGに置き換わる。

〔コマンド〕　R

現在のカーサ位置以降の文字列をキー入力した文字列で置き換えます。
ESCキーにより置き換えが終了します。

（例）＞ABC123GHI■　元の文字列。
＞ABC■　カーサを１に移動する。
＞ABCDEF■　RDEF（ESC）を入力。
＞ABCDEFGHI■　カーサを移動すると文字列置き換えられたのが分かる。

(6) 表　示

〔コマンド〕　L

現在のカーサ位置から行の終端までを表示し，カーサを行の先頭に移動します。

（例）＞ABCDEFG■　元の文字列。
＞ABC■　カーサを移動。
＞ABCDEFG　Lコマンドにより文字列を表示する。
＞■　自動的に改行されカーサが先頭に移動する。

〔コマンド〕　T

現在のカーサ位置から，編集中の最終文字列までを表示し，カーサを編集中の文字列の先頭に移動します。

（例）＞ABCDEFG■
HIJKL■　元の文字列。
＞ABC■　カーサを移動。
＞ABCDEFG
HIJKL　Tコマンドにより文字列を表示する。
＞■　自動的に改行されカーサが先頭に移動する。

(7) 終　了

ESCキーによりエディタ・モードを終了します。

保守／廃止

付録B　組み込みマクロ・コマンド

組み込みマクロ・コマンドを表形式でまとめました。プログラム開発の際にお役立てください。

この一覧は，索引としても使用できます。

なお，IE-17K-ETはPPG（プログラム・パターン・ジェネレータ）をサポートしていないため，PPG制御コマンドがありません。

B.1　プログラム・メモリ制御コマンド

コマンド名	機　能	概　要	参照頁
. LP0	プログラム・メモリのロード（チャネル0）	IE-17K-ETの起動時，RS-232-Cのチャネル0を使用して，×××.ICEファイルをダウンロードします。	p.46
. LP1	プログラム・メモリのロード（チャネル1）	IE-17K-ETの起動時，RS-232-Cのチャネル1を使用して，×××.ICEファイルをダウンロードします。	p.46
. VP0	プログラム・メモリのベリファイ（チャネル0）	RS-232-Cのチャネル0を使用し，IE-17K-ETにダウンロードされたICEファイルと，プログラム・メモリのベリファイをします。	p.47
. VP1	プログラム・メモリのベリファイ（チャネル1）	RS-232-Cのチャネル1を使用し，IE-17K-ETにダウンロードされたICEファイルとプログラム・メモリのベリファイをします。	p.47
. IP	プログラム・メモリの初期化	IE-17K-ETのプログラム領域の任意の範囲を，指定されたデータに一度に書き換えます（1-4-3-4-4形式）。	p.48
. CP	プログラム・メモリの変更	IE-17K-ETのプログラム領域の任意の番地を，1命令ごとに書き換えます（1-4-3-4-4形式）。	p.49
. AP	アセンブル・コマンド（ニモニックによる変更）	IE-17K-ETのプログラム領域の任意の番地を，ニモニックで書き換えます。UPコマンドと組み合わせて使用すると便利です。	p.51
. DP	プログラム・メモリのダンプ	IE-17K-ETのプログラム領域の任意の番地を，最大3FH番地分画面に表示します（1-4-3-4-4形式）。	p.54
. UP	逆アセンブル・コマンド（ニモニックによるダンプ）	IE-17K-ETのプログラム領域の任意の番地を，最大10ステップ画面に表示（ニモニック形式）。	p.56
. FP	プログラム・メモリの検索	プログラム・メモリの内容を1-4-3-4-4形式で検索します。	p.58
. SP0	プログラム・メモリのセーブ（チャネル0）	RS-232-Cのチャネル0を使用し，IE-17K-ET中のプログラムをセーブします。パッチを当てたプログラムのセーブに便利です。	p.59
. SP1	プログラム・メモリのセーブ（チャネル1）	RS-232-Cのチャネル1を使用し，IE-17K-ET中のプログラムをセーブします。パッチを当てたプログラムのセーブに便利です。	p.59
. XS0	PROMデータの出力（チャネル0）	RS-232-Cのチャネル0を使用し，IE-17K-ET中のプログラムをPROM用のファイルに変換し，出力します。	p.60

コマンド名	機　能	概　要	参照頁
. XS1	PROMデータの出力（チャネル1）	RS-232-Cのチャネル1を使用し，IE-17K-ET中のプログラムをPROM用のファイルに変換し，出力します。	p.60
. Q	再起動	プロンプトが＞@@@のときに，このコマンドを実行すると，一度ダウンロードしたプログラムを再起動できます。	p.61

B.2　データ・メモリ制御コマンド

コマンド名	機　能	概　要	参照頁
. ID	データ・メモリの初期化	データ・メモリの任意の範囲を，指定されたデータに一度に初期化します。	p.63
. CD	データ・メモリの変更	データ・メモリの任意の番地を，1データごとに書き換えます。	p.64
. DD	データ・メモリのダンプ	任意の範囲のデータ・メモリをダンプします。	p.66
. D	すべてのデータ・メモリのダンプ	すべてのデータ・メモリをダンプします。	p.67

B.3　周辺回路制御コマンド

コマンド名	機　能	概　要	参照頁
. GD	周辺レジスタの内容表示	周辺レジスタの内容を，画面に表示します。	p.69
. GE	周辺レジスタの内容読み出し	周辺レジスタの内容を，Qレジスタに代入します。プリミティブ・コマンドを用い，ユーザ・マクロを作成するときに使用します。	p.70
. PD	周辺レジスタへの書き込み	周辺レジスタに，数値を代入します。	p.71
. PU	周辺レジスタへの間接書き込み	Qレジスタの内容を周辺レジスタに代入します。プリミティブ・コマンドを用い，ユーザ・マクロを作成するときに使用します。	p.72

B.4　エミュレーション・コマンド

コマンド名	機　能	概　要	参照頁
. R	リセット	SEボードにリセットをかけます。プログラム実行アドレスは0H，データ・メモリ，レジスタ・ファイルなどが，対象製品のリセット状態と同じ状態になります。	p.74，109
. RN	プログラムの実行	指定されているプログラム実行開始アドレスからプログラムを実行します。ブレーク，トレース条件は変化しません。	p.75，109
. BG	プログラムの実行(ブレーク，トレース条件の一部リセット)	指定されているプログラム実行開始アドレスからプログラムを実行します。ブレーク，トレース条件が一部リセットされます。	p.76
. BK	ブレーク	プログラム実行を停止します。	p.77，109
. CA	プログラム・スタート・アドレスの変更	プログラムの実行開始アドレスを変更します。	p.78
. S	ステップ動作	スペース・キーにより1命令ごと，数値のキー入力により指定された回数分，プログラムを実行します。	p.79，122
. DS	ディスプレイ	LCDコントローラを持っている一部の製品が対象のコマンドです。ブレーク中に液晶表示を可能にします。	p.80

B.5　ブレーク／トレース・コンディション制御コマンド

コマンド名	機　能	概　要	参照頁
. CC	ブレーク／トレース条件の変更	ブレーク／トレース条件の設定および，変更を行います。	p.82，112
. CT	トレース・オン／オフ条件の変更	トレースの開始，終了の設定およびトレース・ワン・ショットの設定を行います。	p.92
. DC	ブレーク／トレース条件のダンプ	ブレーク／トレース条件を画面にダンプします。	p.96
. DT	トレース・テーブルのダンプ	トレース結果をダンプします。	p.97
. SC0	ブレーク／トレース条件のセーブ（チャネル0）	.CCのレベル1で設定したブレーク／トレース条件をRS-232-Cのチャネル0に出力します。	p.100
. SC1	ブレーク／トレース条件のセーブ（チャネル1）	.CCのレベル1で設定したブレーク／トレース条件をRS-232-Cのチャネル1に出力します。	p.100
. LC0	ブレーク／トレース条件のロード（チャネル0）	RS-232-Cのチャネル0から，ブレーク／トレース条件をダウン・ロードします。	p.101
. LC1	ブレーク／トレース条件のロード（チャネル1）	RS-232-Cのチャネル1から，ブレーク／トレース条件をダウン・ロードします。	p.101
. VC0	ブレーク／トレース条件のベリファイ（チャネル0）	ブレーク／トレース条件をRS-232-Cのチャネル0によるベリファイに使用します。	p.102
. VC1	ブレーク／トレース条件のベリファイ（チャネル1）	ブレーク／トレース条件をRS-232-Cのチャネル1によるベリファイに使用します。	p.102

B.6　カバレージ表示コマンド

コマンド名	機　能	概　要	参照頁
. DM	カバレージ・メモリのダンプ	プログラム・メモリの通過履歴，データ・メモリの書き込み履歴を表示します。	p.104

B.7　ヘルプ・コマンド

コマンド名	機　　能	概　　要	参照頁
.H	サポート・コマンドの表示	コマンド一覧を表示します。	p.107

保守／廃止

アンケート記入のお願い

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

［ドキュメント名］ IE-17K-ET CLICE-ET V1.6 ユーザーズ・マニュアル
（EEU-931（第1版））

［お名前など］（さしつかえのない範囲で）

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

1．ご評価（各欄に○をご記入ください）

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

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

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

4．ご意見，ご要望

5．このドキュメントをお届けしたのは
NEC 販売員，特約店販売員，NEC 半応技本部員，その他（　　）

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

NEC 半導体応用技術本部インフォメーションセンター
FAX：(044)548-7900

キリトリ

保守／廃止

保守／廃止

お問い合わせは，最寄りのNECへ

本　　社	〒108-01 東京都港区芝五丁目7番1号 (NEC本社ビル)
コンシューマ半導体販売事業部 OA半導体販売事業部 インダストリ半導体販売事業部	〒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
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 (06)945-3178 大　阪 (06)945-3200 大　阪 (06)945-3208

北海道支社	札　幌 (011)231-0161	立川支社	立　川 (0425)26-5981
東北支社	仙　台 (022)261-5511	千葉支社	千　葉 (043)238-8116
岩手支店	盛　岡 (0196)51-4344	静岡支社	静　岡 (054)255-2211
山形支店	山　形 (0236)23-5511	沼津支店	沼　津 (0559)63-4455
郡山支店	郡　山 (0249)23-5511	浜松支店	浜　松 (053)452-2711
いわき支店	いわき (0246)21-5511	北陸支社	金　沢 (0762)23-1621
長岡支店	長　岡 (0258)36-2155	福井支店	福　井 (0776)22-1866
水戸支店	水　戸 (0292)26-1717	富山支店	富　山 (0764)31-8461
神奈川支社	横　浜 (045)324-5511	京都支社	京　都 (075)344-7824
群馬支店	高　崎 (0273)26-1255	神戸支社	神　戸 (078)332-3311
太田支店	太　田 (0276)46-4011	中国支社	広　島 (082)242-5504
宇都宮支店	宇都宮 (0286)21-2281	鳥取支店	鳥　取 (0857)27-5311
小山支店	小　山 (0285)24-5011	岡山支店	岡　山 (086)225-4455
長野支社	長　野 (0262)35-1444	四国支社	高　松 (0878)36-1200
松本支店	松　本 (0263)35-1666	新居浜支店	新居浜 (0897)32-5001
上諏訪支店	諏　訪 (0266)53-5350	松山支店	松　山 (0899)45-4111
甲府支店	甲　府 (0552)24-4141	九州支社	福　岡 (092)271-7700
埼玉支社	大　宮 (048)641-1411	北九州支店	北九州 (093)541-2887

(技術お問い合わせ先)

半導体応用技術本部　マイクロコンピュータ技術部	〒210　川崎市川崎区駅前本町15番5号 (十五番館)	川　崎 (044)246-3923
半導体応用技術本部 中部応用システム技術部	〒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
半導体応用技術本部 西日本応用システム技術部	〒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

半導体応用技術本部
インフォメーションセンター
FAX(044)548-7900
(FAXで対応させていただいております)