

C r o s s

D e b u g g e r - M S

# も く じ

## 第1章 は じ め に

1-1	紹 介 .....	1
1-2	デバッグを開始する前の準備 .....	2

## 第2章 シンボル定義ファイル

2-1	シンボル定義ファイルとは? .....	3
2-2	シンボル定義ファイルの作成 .....	4
2-3	定義例 .....	7

## 第3章 デバッガーの実行

3-1	デバッガーの実行方法 .....	8
3-2	スクリーンモードを使用するには .....	9
3-3	スクリーンモードのメリット .....	11
3-4	スクリーンモードの画面説明 .....	12

## 第4章 簡単なデバッグ例

4-1	ノーマルモードでの使用例 .....	15
4-2	スクリーンモードでの使用例 .....	17

## 第5章 コマンドの詳細

5-1	コマンドの実行方法 .....	19
5-2	スクリーンモード関係 .....	21
	スクリーンモードの切り替え .....	21
	メモリ監視エリアの設定 .....	22
5-3	メモリ操作関係 .....	23
	メモリのダンプ表示 .....	23
	メモリ内容の変更 .....	25
	メモリ領域の満たし .....	28

メモリのコピー	29
メモリの比較	30
データの検索	31
アセンブル	32
逆アセンブル	35
5-4 ロード・セーブ関係	39
オブジェクトのロード	39
ロードアドレスの表示	40
モトローラSフォーマットでのセーブ	41
インテルHEXフォーマットでのセーブ	42
5-5 ブレークポイント関係	43
停止するブレークポイントの設定・表示	44
全レジスタを表示するブレークポイントの設定・表示	45
指定レジスタを表示するブレークポイントの設定・表示	46
指定文字列を表示するブレークポイントの設定・表示	47
すべてのブレークポイントの表示	48
ブレークポイントの解除	49
5-6 シンボル関係	50
シンボルフайルのロード	50
シンボルの登録	51
アドレス順でのシンボルの表示	52
シンボル名順でのシンボルの表示	53
シンボルの消去	54
5-7 シミュレーション関係	55
ステップ実行	55
シミュレーション	57
高速シミュレーション	58
スクリーンモード用シミュレーション1	59
スクリーンモード用シミュレーション2	60
5-8 レジスタ関係	62
レジスタの設定・表示	62
現在のレジスタの保存	64
保存したレジスタのロード	65
履歴の設定・表示	66
履歴の表示位置リセット	70
5-9 その他	71
数値計算	71
バッチファイルの実行	74
DOSコマンドの実行	76
ヘルプ表示	77
終了	79

---

## 第6章 各デバッガの詳細

---

6-1	6303編	80
	割り込みコントロール	80
	シミュレートの制限	81
	レジスタの指定法	82
	アセンブルコマンドの書式	83
	C言語を使つてのデバッグ例	86
	アセンブラを使つてのデバッグ例	91
6-2	Z80編	95
	割り込みコントロール	95
	シミュレートの制限	96
	I/Oの書き込み	97
	I/Oの読み込み	99
	レジスタの指定法	101
	アセンブルコマンドの書式	102
	アセンブラを使つてのデバッグ例	104

---

## 付 録

---

付録A	： コマンド一覧表	109
付録B	： ユーティリティープログラム	111
	LOAD (メモリへのオブジェクトのロード)	111
	SAVE (メモリからのオブジェクトのセーブ)	112
	HEXADR (オブジェクトのアドレス確認)	113

---

## 第1章 はじめに

---

### 1-1 紹介

クロス開発環境でプログラムを作成する人にとって、一番大変なものがデバッグ作業です。ICEなどのハードウェアを使用するという方法もありますが、とても高価なものですのでちょっとした仕事や勉強には向きません。

「CROSS DEBUGGER」は、こうした人たちの為に開発されたターゲットCPUをほぼ完全にソフトウェアでシミュレートするデバッガーです。これで、C、アセンブラでの開発がデバッグを含めて全てMS-DOSマシンでのみ行えることになります。最高のコストパフォーマンスなクロス開発ツールです。

本書は、「CROSS DEBUGGER」の共通マニュアルです。

本文中で「S03」と記述しているものは、6301/3のデバッガーを示し、  
「SZ80」と記述しているものは、Z80のデバッガーを示します。

## 1-2 デバッグを開始する前の準備

「S03」では、「S03.EXE」が実行ファイルです。

「SZ80」では、「SZ80.EXE」が実行ファイルです。

読み込めるシンボルファイルの形式は、デフォルトで「CROSS ASSEMBLER」になっています。それ以外のシンボルファイルを読むことも可能ですが、その為にはあらかじめカレントディレクトリに、シンボル定義ファイル「SYMBOL.DEF」を入れて置かなければなりません。

このファイルは、使用しているアセンブラまたはリンカーの出力するシンボルファイルに合わせて作成しなくてはなりません（作成方法については第2章を参照）。

しかし、当社の「Introl Cross C」を使用している場合は、ディスクに含まれる「XC.DEF」「XC3.DEF」を「SYMBOL.DEF」という名前にリネームするだけでOKです。

「？」（ヘルプメッセージを表示する）コマンドを実行したい場合は、カレントディレクトリに「CRSDEBUG.HELP」というファイルを入れて置かなければなりません。

このファイルは、実際にヘルプ表示する為の文章が入っています。ヘルプメッセージは日本語で表示されますので、もし、IBM-PCなどの日本語表示ができない機種では、このファイルの中身をすべて英字に変更してください。詳しくは、「5-9」「ヘルプ表示」を参照してください。

デバッグ作業は、オブジェクトに対して行いますので、あらかじめオブジェクトが必要だということと言うまでもありません。

オブジェクトの形式は、モトローラSフォーマットまたはインテルHEXフォーマットでなくてはなりません。

---

## 第2章 シンボル定義ファイル

---

### 2-1 シンボル定義ファイルとは？

従来のシンボリック・デバッガーでは、デバッグ時にシンボルが使用できるアセンブラ（リンカー）を限定していました。これは、シンボリック・デバッガーの宿命というものでありました。

「CROSS DEBUGGER」では、何とか他のアセンブラの出力したシンボルファイルも使用できないだろうか考えた末、「シンボルファイルの形式を定義する」という方法を考え出しました。この「定義」を行うことによって、他の多くのアセンブラ（リンカー）の出力するシンボルファイルでも使用することが可能になりました。

シンボル定義ファイルは、「SYMBOL. DEF」というファイルです。

まず、ユーザーは自分の使用しているアセンブラ（リンカー）の出力するシンボルファイルの形式を「SYMBOL. DEF」というファイルで定義しなければなりません。

ただし、当社から販売している「Introl Cross C」は、あらかじめ、それように作成してある定義ファイルを「SYMBOL. DEF」という名前に変えるだけでOKです。

また、「CROSS ASSEMBLER」は、この定義ファイルは必要ありません。もし、カレントディレクトリに「SYMBOL. DEF」という名のファイルがあった場合は、そのファイルを削除するかリネームしてください。

「Introl Cross C」のバージョン2用は「XC. DEF」です。

バージョン3用は「XC3. DEF」です。

## 2-2 シンボル定義ファイルの作成

「CROSS ASSEMBLER」や「Introl Cross C」を使用しているユーザーの方は、この章を読む必要はありません。第3章に移ってください。

シンボル定義ファイルは、各種のエディタで作成します。

定義はシンボルのスタート位置、エンド位置、ページスキップ、そしてシンボル、アドレスの1ラインのフォーマットを指定するようにしています。

定義文は必ず最初にスペースなどを入れず、頭から「% x x x」と記述してください。

定義文は以下の4つです。

### 【その1】 シンボルの始まる場所を定義する。

書 式	% S K I P " × × × × × × " ( × × × × × × は文字列を指定)
説 明	「 × × × × × × 」が現われた次の文字からシンボルとして認識します（それまでは、 いかなる文字が現われてもシンボルとして認識しない）。 但し、次の文字がコントロールコード（普通は C R / L F ）の場合は、次の行から認識します。 この指定を省略した場合、ファイルの先頭からシンボルとして認識します。
使用例	% S K I P " * * * S Y M B O L T A B L E * * * "
解 説	「 * * * S Y M B O L T A B L E * * * 」が現れた次の文字からシンボルとして認識します。また、上記の例では「 % S K I P " T A B L E * * * 」と、後ろの方だけ指定して構いません。

### 【その2】 シンボルの終わりを定義する。

書 式	% E N D " × × × × × × " ( × × × × × × は文字列を指定)
説 明	「 × × × × × × 」が現われた時点で終了します。 この指定を省略した場合、ファイルの E O F で終了します。
使用例	% E N D " * * * S Y M B O L E N D * * * "
解 説	「 * * * S Y M B O L E N D * * * 」が現れた時点で、シンボルの読み込みを停止します。



【その 3】 ページング時のスキップ行数を定義する。

書 式	%PAGE スキップ行数
説 明	ファイル中に 0 C (16 進) のコードを見つけると、指定した「スキップ行数」分だけ行をスキップし、シンボルを読み込む行の対象にしません。
使用例	%PAGE 3
解 説	ページング後、3 行をスキップします。

【その 4】 シンボル+アドレスの形式を定義します。

書 式	%FORMAT "##### &&&"
説 明	1 ラインのイメージを定義します。 「#####」はシンボル名、「&&&」はアドレスです。 必ず 1 ラインには、「####」と「&&&」がなくってはなりません。 決して省略はできません。 なお、シンボル名は 11 文字までが入力対象となります。
使用例	%FORMAT "##### &&&"
解 説	シンボル名は第 2 カラムから 8 文字。第 11 カラムから 4 桁の数値がそのアドレスになります。尚、アドレスは 16 進表記でなくてはなりません。

#### ●補足説明

定義文は必ず 1 行（80 文字）以内に記述しなければいけません。

しかし、この FORMAT 文は、シンボルの 1 ラインの形式を定義するものですから、とても 1 行以内には収まらないと思います。その為、この FORMAT 文に関しては、次の行に記述しても良いようになっています。以下にその例を示します。

%FORMAT

##### &&& ##### &&& ##### &&& ##### &&& ##### &&&

この場合、そっくりそのまま、この 1 ラインのイメージでシンボル、アドレスを読みます。

アセンブラの中には、出力するシンボルファイルで、シンボル名とアドレスの間をタブで区切っているものがあります。その場合のフォーマット定義の書き方を説明します。

以下のシンボルテーブルがその一例ですが、画面上ではスペースに見える所でも実際はタブ(9)のコードが入っています。

#### 【画面上】

```
| 0106 ADR1      009A ADR2      009F ADR3      007A ADR4      |
```

#### 【データ中】

```
| 0106 ADR1<tab>009A ADR2<tab>009F ADR3<tab>007A ADR4      |
```

- ・ <tab>はタブ(9)に相当します

その為、そのままスペーススキップの対象にしてしまう（1つのタブを1つのスペースとしてしまう）と、アドレス、シンボル名がずれてしまいます。

フォーマット定義では、タブをサポートしていない為、文中にタブを書くことができません。

しかし、安心してください。画面に表示されたシンボルのイメージ通りに定義しさえすれば、シンボルファイルを読み込む時にタブをスペースに拡張しますからそのまま読み込めます。つまり、上記のシンボルテーブルを読み込む為の定義は以下のように行います。

```
| %FORMAT  
| &&&& ##### &&&& ##### &&&& ##### &&&& #####  
|
```

★決して、シンボルファイルにタブが入っている場合でも、定義ファイルにはタブを使用しないでください。画面に表示されたイメージで定義してください。

★タブは、デフォルトの8スキップを行います。

以上が、定義ファイルの作成法です。

定義ファイルを作成し、自分の使用しているアセンブラ（リンカー）のシンボルファイルが読み込めればOKです。

## 2 - 3 定義例

次のようなシンボルファイルの場合を考えてみましょう。

```
| 0512 DELETE      0641 INKEY      0660 INPUT      0700 KEYBUF      |  
| 0455 PRINT       0440 PUTC       0110 QUIT       0100 STACK      |  
| 0100 START                                              |  
|-----|
```

非常に単純なシンボルファイルです。ただし、最初にアドレスが来て、その後ろにシンボル名が来ています。

この場合は、以下のように定義ファイルを作成すれば良いでしょう。

```
| %FORMAT  
| &&&& ##### &&&& ##### &&&& ##### &&&& #####  
|-----|
```

シンボル名の長さは、最高 8 文字に設定しました。

---

## 第3章 デバッガーの実行

---

### 3-1 デバッガーの実行方法

「CROSS DEBUGGER」は、以下のコマンドラインで起動します。

A>S03	(6301 / 3デバッガーの場合)
A>SZ80	(Z80デバッガーの場合)

★アンダーラインの箇所が実際に入力する所です。

起動時に、デバッグするプログラム（オブジェクト）をロードすることもできます。

A>S03	[ファイル名]
A>SZ80	[ファイル名]

(例) 6301 / 3デバッガーを起動すると同時に、「TEST. S」というモトローラSフォーマットファイルを読み込みます。

A>S03	TEST
-------	------

※「S03」ではファイル名の拡張子を省略すると、自動的に「. S」を付けます。

(例) Z80デバッガーを起動すると同時に、「TEST. HEX」というインテルHEXフォーマットファイルを読み込みます。

A>SZ80	TEST
--------	------

※「SZ80」ではファイル名の拡張子を省略すると、自動的に「. HEX」を付けます。

シンボルファイル名は「. MAP」という拡張子が付きます。

上記の例では、どちらも「TEST. MAP」というシンボルファイルを読み込みます。もちろん、シンボルファイルが見つからない場合は、エラーメッセージを出力し、デバッガーのコマンド入力待ち状態に入ります。

デバッガーは、ファイル名の指定が有る無しに関わらず、起動時に「SYMBOL. DEF」というファイルを探しに行きます。

この「SYMBOL. DEF」ファイルが存在しなかった場合は、「CROSS ASSEMBLER」のシンボルファイルが対象になります。

### 3-2 スクリーンモードを使用するには

「CROSS DEBUGGER」には、スクリーンモードというものがあります。これは、メモリ内容やレジスタ値などを常時表示しているもので、シミュレートやメモリ変更などにとても便利です。

スクリーンモードでは、高速に画面表示をする必要がある為、直接VRAMに表示データを書き込んでいます。これにより非常に高速になりますが、その反面「機種に依存する」という欠点を持つこととなります。

現在の所、以下の機種に対応しています。

●グループA		
日本電気	PC-9801	シリーズ
エプソン	PC-286	シリーズ
●グループB		
富士通	FMR	シリーズ
	FM-16β	シリーズ
松下電機	Panacom	Mシリーズ

※上記以外の機種では、スクリーンモードは使用できませんのでご了承ください。

グループAの機種で、スクリーンモードを使用したい場合は、

A>S03	-PC98
A>SZ80	-PC98

と「-PC98」とオプション指定を行ってください。

グループBの機種で、スクリーンモードを使用したい場合は、

A>S03	-FMR
A>SZ80	-FMR

と「-FMR」とオプション指定を行ってください。

もちろん、このオプションを指定しても、同時にファイルの読み込みも行えます。

```
| A>S 0 3   T E S T   - P C 9 8  
|  
| A>S Z 8 0   - F M R   T E S T  
|
```

上記のようにファイル名もオプションもどちらを先に指定しても構いません。

スクリーンモードのオプションを指定すると、スクリーンモード画面になります。

※使用している機種を無視し、オプションを指定すると予期せぬことが起こりますので、絶対確認をしてください。

### 3-3 スクリーンモードのメリット

スクリーンモードでは、シミュレーション実行時に威力を発揮します。

ノーマルモード（スクリーンモードでない画面モード）では、レジスタ値はいつでも監視できますが、メモリ内容の監視はできません。

それに対しスクリーンモードでは、メモリ内容も常時監視できます。

しかし、メリットばかりではありません。例えば、デバッグ経過をプリンタに印字したり、ファイルに書き出したりすることができません。

「CROSS DEBUGGER」は、スクリーンモード、ノーマルモードをいつでも切り替えることができます（オプション指定で「-PC98」「-FMR」を付けて起動した時のみ）。

※プリンタに印字するには「 P」を押します。次から画面に表示するものがプリンタに出力されます。

※ファイルに書き出すにはデバッガー起動時にリダイレクションを付けて立ち上げます。

（この場合は画面に何も表示されなくなりますのでカンで操作するしかありません。）

3-4 スクリーンモードの画面説明

スクリーンモードでは下記の画面になります。

【SZ80】

①	②										③	④
Cross Debugger	V2.xx	BS[ 0]	BR[ 0]	BQ[ 0]	BT[ 0]	HIST[ ON]	0000-0000(0000)					
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					>
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					⑦
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0040 00 00 00 00 00 00 00 0 ⑤		⑤	0 00 00 00 00 00 00 00 00				.....	⑥	.....			
0050 00 00 00 00 00 00 00 00			00 00 00 00 00 00 00 00 00				.....					
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00							.....					
0000 00	NOP						szhpnc⑨					
0001 00	NOP						F =00 00					
0002 00	NOP						A =00 00					
0003 00	NOP						BC=00 ⑩ 000					
0004 00	NOP						DE=0000 0000					
0005 00	NOP	⑧					HL=0000 0000					
0006 00	NOP						IX=0000					
0007 00	NOP						IY=0000 1 5					
0008 00	NOP						SP=0000 2 6					
0009 00	NOP						PC=0000 3 7					
000A 00	NOP						I =00 4 8					
000B 00	NOP						R =00 ⑪					

【S03】

0000 01	NOP						CC=00 hinzvc⑨					
0001 01	NOP						A =00					
0002 01	NOP						B =00					
0003 01	NOP						X =00 ⑩					
0004 01	NOP						SP=0000					
0005 01	NOP	⑧					PC=0000					
0006 01	NOP											
0007 01	NOP						1 5					
0008 01	NOP						2 6					
0009 01	NOP						3 7					
000A 01	NOP						4 8					
000B 01	NOP						⑪					



#### ①エラーメッセージ表示

通常はタイトルを表示していますが、何等かのエラーが発生した場合、ブザー音と共にこのエリアに表示します。次のコマンドを実行した後に、エラーメッセージは消去されます。

#### ②ブレークポイント数表示

4つのブレークポイントの数を常時表示しています。

#### ③ヒストリフラグ表示

シミュレート実行時（Jコマンドを除く）にヒストリバッファに保存していくかどうかのスイッチです。ヒストリがOFFになっている場合は、ヒストリ機能は働きません。

「H」コマンドで設定できます。

#### ④ロードアドレス表示

プログラム（オブジェクト）のロードアドレスを表示しています。

先頭アドレス及び終了アドレス、そしてカッコの中が実行アドレスです。

#### ⑤ダンプエリア

メモリ内容を160バイト表示しています。

「SZ80」では、I/Oエリアの表示にも切り替わります。

#### ⑥キャラクタエリア

メモリ内容をキャラクタで表示しています。

コントロールコードは「.」で、MS-JISの漢字第1バイト目のコードは「\_」で表示しています。

#### ⑦コマンド入力エリア

コマンドを入力する場所です。9文字以上の入力では横スクロールします。最高79文字まで入力できます。

編集キーとして、「」（バックスペース）」が使用できます。入力のキャンセルは、「」（エスケープ）」または「C（コントロールC）」で行います。

#### ⑧逆アセンブルエリア

メモリの逆アセンブルを表示しています。

また、このエリアはアセンブルエリアとしても使用されます。

## ⑨フラグレジスタの状態

フラグレジスタのビットのオン・オフを表示しています。  
フラグ名が大文字の時はオン、小文字の時はオフです。

## ⑩レジスタエリア

レジスタの内容を表示しています。「S Z 8 0」では、右側に裏レジスタの表示をしています。

## ⑪レジスタ保存状態エリア

レジスタ値の保存は9つまで行えます。  
保存するとその番号がリバーシします。保存内容を取り出すとリバーシを解除します。

スクリーンモード2（「S C 2」コマンド実行）では下記の画面になります。

Cross Debugger	V2.xx	BS[ 0]	BR[ 0]	BQ[ 0]	BT[ 0]	HIST[ ON]	0000-0000(0000)
0000 00 00 00 00	0000 00 00 00 00		WPPTR		1000 (0000)	[0000]	>
0004 00 00 00 00	0004 00 00 00 00						
0008 00 00 00 00	0008 00 00 00 00						
000C 00 00 00 00	000C 00 00 00 00						
0010 0 ⑫ 0 00	0010 0 ⑬ 0 00				⑭		
0014 00 00 00 00	0014 00 00 00 00						
0018 00 00 00 00	0018 00 00 00 00						
001C 00 00 00 00	001C 00 00 00 00						
0020 00 00 00 00	0020 00 00 00 00						
0024 00 00 00 00	0024 00 00 00 00						

## ⑫ダンプエリア1

メモリ内容を40バイト表示しています。

## ⑬ダンプエリア2

メモリ内容を40バイト表示しています。ダンプエリア1とは別のエリアを表示できます。  
「S Z 8 0」では、I/Oエリアの表示にも切り替わります。

## ⑭メモリ監視エリア

データの間接参照を表示しています。  
シンボル名、アドレス、アドレスの内容、アドレスの内容の内容までを表示できます。  
なお、どここのレベルまで表示するかは自由に設定できます。  
最高10個までのメモリ内容を表示できます。

---

## 第4章 簡単なデバッグ例

---

### 4-1 ノーマルモードでの使用例

この章では、ディスクに含まれる「SAMPLE」を実際にロードして、簡単な使用法を示します。ここではノーマルモードでの使用例を、「4-2」ではスクリーンモードでの使用例をそれぞれ述べることにします。

#### ①デバッガーを起動する

A>S03 SAMPLE	(6301/3デバッガーの場合)
A>SZ80 SAMPLE	(Z80デバッガーの場合)

どちらのデバッガーも、オブジェクトファイル名の指定は同じです（拡張子を省略した場合）。

オブジェクトファイルをロードし、デバッガーが起動すればOKです。

オブジェクトファイルが見つからなかった場合や「SYMBOL.DEF」ファイルが異常だった場合は、エラーメッセージを出力しますがそのまま起動します。

>Q
----

「Q」コマンドで終了できますから、エラーが発生した場合はMS-DOSに戻ってください。

なお、シンボルファイル「SAMPLE.MAP」は、「CROSS ASSEMBLER」のもので、SYMBOL.DEF」ファイルは必要ありません。

無事、オブジェクト及びシンボルファイルがロードできればOKです。

#### ②ダンプ表示する

>D 1000
---------

1000番地からのメモリ内容が256バイト表示されます。

1000番地から数バイト値が入っているのがわかると思います。

### ③逆アセンブルする

```
>U 1000
```

1000番地から逆アセンブルして表示されます。

確かにプログラムらしきものが入っています。このプログラムは、100番地から256バイトのエリアに「00,01,02,03 ……」というデータをセットしていくというものです。

### ④プログラムを実行する

```
>G 1000, 100C
```

 (「S03」の場合)

```
>G 1000, 1010
```

 (「SZ80」の場合)

※「S03」と「SZ80」では、プログラムの終了アドレスが異なりますので注意してください。

一拍置くぐらいの感じで戻ってきます。

「S03」の場合は、

```
CC=C4(hinZvc) A=00 B=00 X=0200 SP=0000 PC=100C BRA $100C
```

という表示に、「SZ80」の場合は、

```
sZHpnC A=00 B=0000 D=0000 H=0000 S=0000 P=1010 I=00XX R=00  
szhpnc A'00 B'0000 D'0000 H'0000 X=0200 Y=0000 JP 1010H
```

という表示になれば正常に終了したことを意味します。

### ⑤値を確認する

```
>D 100
```

100番地から「00,01,02,03 ……」とセットされていればOKです。

### ⑥終了する

```
>Q
```

「Q」コマンドで終了します。

## 4-2 スクリーンモードでの使用例

今度は、スクリーンモードで試して見ましょう（必ず「3-2」を最初に読んでください）。

### ①デバッガーを起動する

A>S03	SAMPLE	-PC98	(PC-9801などの機種で実行する場合)
A>S03	SAMPLE	-FMR	(FMRなどの機種で実行する場合)
A>SZ80	SAMPLE	-PC98	(PC-9801などの機種で実行する場合)
A>SZ80	SAMPLE	-FMR	(FMRなどの機種で実行する場合)

ノーマルモードでは、ダンプ表示や逆アセンブル表示をして確認しましたが、このモードでは、立ち上げると既にそれらの表示は完了しています。

### ②プログラムを実行する

さっそく、実行して見ましょう。先ほどの「G」コマンドでも実行できますが、このコマンドではリアルタイムにメモリ内容を見たりすることができません。その為、今回はスクリーンモードのすばらしさを見る為に、リアルタイムに表示しながら実行して見ましょう。

メモリをセットしていくエリア、100番地をメモリダンプエリアに表示させます。

>D	100
----	-----

まだ、プログラムを実行していない為、値はすべて「0」です。

それでは、実行します。

>GS	1000, 100C	(「S03」の場合)
>GS	1000, 1010	(「SZ80」の場合)

実際に、100番地からのメモリが変化している様子が分かります。また、レジスタ、逆アセンブルエリアの表示も変わります。

実行している間にBレジスタを見てください。だんだんと値が減っていくのが分かります。そして「0」になった瞬間、プログラムは停止します。

### ③メモリ内容を確認する

1 A 0 番地からの内容が見えないので、その下も確認してみましょう。

```
>M 1 0 0
```

リバースカーソルがダンプエリアに表示されます。

カーソルキーを使って前後左右に移動できます。「M」コマンドはメモリ変更をするものですから、数値（16進数）を入力すると、その値が変わります。

「（エスケープ）」を押すと、コマンド入力モードに戻ります。

### ④終了する

```
>Q
```

「Q」コマンドで終了します。

---

## 第5章 コマンドの詳細

---

### 5-1 コマンドの実行方法

すべてのコマンドは、

```
[ コマンド名 ]    [ パラメータ 1 ]    [ パラメータ 2 ]    . . . .
```

という形で入力します。

コマンドによっては、パラメータをいくつも指定できるものがあります。その場合でも、1度に入力できる文字数は80文字、パラメータ数は40個までとなっています。

なお、パラメータとパラメータは、必ずスペースまたは「,」（カンマ）で区切ってください。

#### ●コマンド書式の説明

ほとんどのコマンドはパラメータの指定ができます。コマンドによっては、必ず指定しなければならないもの、省略可能なものなどがあります。それらを明確にする為に記号で区別しています。

例えば「D」コマンドを例にとりますと、

```
[ D    [ ADRS1 [ ADRS2 ] ]
```

※ [ ]（大括弧）で囲まれたパラメータは省略可能なものです。

※ < >（鍵括弧）で囲まれたパラメータは省略不可能なものです。

上記の例の指定法は、以下のように3つあります。

```
> D
> D   1 0 0
> D   1 0 0   2 F F
```

これと似た指定で「J」コマンドがありますが、これは少々違います。

```
[ J    [ ADRS1 ] [ ADRS2 ]
```

この場合は、[ADRS1] を省略して、[ADRS2] のみを指定できるようになっています。

「D」コマンドでは、[ADRS2] を指定する場合は、必ず [ADRS1] の指定が必要でした。

実際には、

```

> J
> J 1 0 0
> J 1 0 0 2 F F
> J , 2 F F

```

の4通りの指定が可能です。第1パラメータを指定せず、その次のパラメータを指定する場合は、必ず、「,」を付けてください。

### ●コマンドラインで指定できるもの

コマンドラインで指定できる数値は、デフォルトで16進になります。また、コマンドラインでは、式での表現もできるようになっています。もちろん、シンボルが登録してある場合は、そのシンボルも指定できます。詳しくは、「5-9」「数値計算」の章を読んでください。

例えば、

```

D . S T A R T      (シンボルの指定は、「.」(ピリオド)で行う)
T # 1 0            (10進の指定は「#」で行う)

```

```

| 文章中で「x x x x 番地」と記述しているものは16進表記になっています。
| また、「1 0」を「0 1 0」と頭に強制的に「0」を付けたものや数値の中に「A」～「F」
| の文字があるものも16進表記です。それ以外は10進表記になっています。
|

```



## 5-2 スクリーンモード関係

### スクリーンモードの切り替え

#### 【書 式】

SC [ MODE ]

#### 【説 明】

画面のモードを切り替えます。

「MODE」で0を指定すると、ノーマルモードになります。

「MODE」で1を指定すると、スクリーンモード1になります。

「MODE」で2を指定すると、スクリーンモード2になります。

「MODE」を省略すると、ノーマルモードになります。

機種専用モードで立ち上げないと、このコマンドは実行できません（3-2 参照）。

#### 【使用例】

>SC 0	(ノーマルモードに切り替える)
>SC 1	(スクリーンモード1に切り替える)
>SC 2	(スクリーンモード2に切り替える)
>SC	(ノーマルモードに切り替える)

### メモリ監視エリアの設定

#### 【書 式】

V < ADRS > < MODE >

#### 【説 明】

スクリーンモード2のメモリ監視エリアに表示するアドレスを設定します。

「ADRS」で表示したいアドレスを指定します。

「MODE」で1を指定すると、アドレスの内容を1バイト表示します。

「MODE」で2を指定すると、アドレスの内容を2バイト表示します。

「MODE」で3を指定すると、間接アドレスの内容を1バイト表示します。

「MODE」で4を指定すると、間接アドレスの内容を2バイト表示します。

「MODE」で0を指定すると、指定したアドレスの設定を解除します。

「S 0 3」のアドレス内容の2バイト表示は、カレントアドレスを上位バイト、次のアドレスを下位バイトとします。

「S Z 8 0」では、カレントアドレスを下位バイト、次のアドレスを上位バイトとします。

最高10個までのアドレスが設定できます。

#### 【使用例】

>SC 2	(スクリーンモード2にする)
>K 0 DATA	(0番地に「DATA」というシンボル名を付ける)
>V .DATA 1	(「DATA」をモード1で表示)
>V .DATA 2	(「DATA」をモード2で表示)
>V .DATA 3	(「DATA」をモード3で表示)
>V .DATA 4	(「DATA」をモード4で表示)
>V .DATA 0	(「DATA」の表示をやめる)

DATA	0000 (00)	モード1
DATA	0000 (0000)	モード2
DATA	0000 (0000) [00]	モード3
DATA	0000 (0000) [0000]	モード4
----- -----		
		----- 0番地の内容の内容
		----- 0番地の内容

## メモリのダンプ表示

## ● ノーマルモード時

## 【書 式】

D [ ADRS1 [ ADRS2 ]]

## 【説 明】

メモリ内容を16進で表示します。

右側にそれぞれに対応する文字も表示します。

00～1Fは「.」、80～9F、E0～FFは「\_」で表示します。

パラメータを指定しなかった場合は、カレントアドレスから256バイト表示します。

そして、カレントアドレスは256バイト分増加します。カレントアドレスとは、そのコマンドで次に表示すべきアドレスのことを言い、「D, M, A, U」コマンドがそれぞれ持っています。

また、「ADRS1」のみを指定した場合、そのアドレスがカレントアドレスになり、そこから256バイト分表示します。「ADRS2」も指定した場合は、そのアドレスまで表示します。但し、そのアドレスですぐに終了するのではなく、最低16バイトは表示します。

## 【使用例】

```

>D 100                                     (100番地からダンプ)
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F .....
0110 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F .....
0120 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F !"#$%&'()*+,-./
0130 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
0140 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
0150 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[¥]^_
0160 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F ~abcdefghijklmno
0170 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{|} .
0180 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F _____
0190 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F _____
01A0 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF 。「」、・ヲアイウエオヤユヨツ
01B0 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF ーアイウエオカキクケコサシスセソ
01C0 C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF タチツテトナニヌネノハヒフヘホマ
01D0 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF ミムメモヤユヨラリルレロワヰ°
01E0 E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF _____
01F0 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF _____

```

●スクリーンモード時

【書 式】

D [ ADRS ]

D 2 [ ADRS ]

【説 明】

「D 2」 コマンドは、スクリーンモード 2 のダンプエリア 2 の表示に使用します。

どちらも機能は、ノーマルモードの時と同じです。

ただし、スクリーンモード 2 の時は、キャラクタの表示は行いません。

「ADRS」をトップにダンプエリアを更新します。

【使用例】

```

>SC 1                                (スクリーンモード 1 にする)
>D 100                               (1 0 0 番地からダンプ)

+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F .....
0110 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F .....
0120 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F !"#$%&'()*+,-./
0130 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
0140 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO
0150 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[¥]^_
0160 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F ~abcdefghijklmno
0170 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{|} .
0180 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F _____
0190 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F _____

```

```

>SC 2                                (スクリーンモード 2 にする)
>D2 1000                             (1 0 0 0 番地からダンプ)

0100 00 01 02 03 | 1000 CE 01 00 4F |
0104 04 05 06 07 | 1004 5F A7 00 4C |
0108 08 09 0A 0B | 1008 08 5A 26 F9 |
010C 0C 0D 0E 0F | 100C 20 FE 00 00 |
0110 10 11 12 13 | 1010 00 00 00 00 |
0114 14 15 16 17 | 1014 00 00 00 00 |
0118 18 19 1A 1B | 1018 00 00 00 00 |
011C 1C 1D 1E 1F | 101C 00 00 00 00 |
0120 20 21 22 23 | 1020 00 00 00 00 |
0124 24 25 26 27 | 1024 00 00 00 00 |

```

## メモリ内容の変更

### ●ノーマルモード時

#### 【書 式】

M      [ ADRS ]

#### 【説 明】

メモリ内容を1バイト単位で表示し変更します。

このコマンドもダンプコマンドと同様に、パラメータを指定しなかった場合はカレントアドレスが対象になり、指定した場合はそのアドレスが対象になります。

このデバッガーのメモリチェンジコマンドは、他のものより強力に出来ています。表示は1つのメモリデータに対して、16進、10進、2進、アスキーの4つを、もちろん入力もこの4つのどの形でも行えるようになっています。また、1ラインで1度に入力することも可能です。

「M」コマンドを実行すると、

Adrs	Hex	Dec	Binary	Ascii	Entry
0000	:	41	65	01000001	A [H] =

という表示になります。

左から説明して行きますと、まず、アドレス（ここでは0番地）、次にそのアドレスの内容の16進、10進、2進、アスキーでの表示となっています。アスキー表示では、020未満のコントロールコードは、「^x」（xは、そのコード+040）となっています。07Fは、「del」と表示します。また、80～9F、E0～FFは、「\_」と表示します。

続いて「Entry」ですが、この表示は現在の入力モードを示します。入力モードは以下の通りです。

[H]    16進入力モード  
[D]    10進入力モード  
[B]    2進入力モード  
[A]    アスキー文字入力モード

入力モードの切り替えは「.」（ピリオド）に続いて上記の4種類の文字の1つを指定することにより行います。後は、そのモードで許される数値、文字を入力するだけです。もう1つ、特別な文字があります。それは、「-」（マイナス）です。これは、どのモ

ードでも使用でき、一度これを入力するとリターン後のアドレスが逆に進行します。つまり、100番地で「-」を入力すると次はFF番地になるということです。再度、「-」を入力すると元のプラス方向への進行になります。次の進行方向がどちらかということを見るには、「Adrs」と「Hex」の間に表示している文字で分かります。

「:」が表示している時はプラスの方向、「-」の時は、マイナスの方向に進行するという意味です。

1ラインで入力する方法があると言いましたが、これは次のように行います。

Adrs	Hex	Dec	Binary	Ascii	Entry
0000	:	00	0	00000000	^@ [H] = 1 2 3 .D 10 20 30

つまり、本来リターンを打つ所をスペースで区切って1ラインで入力するという事です。ただし、1ラインに入力できる文字数やパラメータ数が限られています。文字数は38文字、パラメータ数は20個です。もし、それ以上になりそうな場合は、こまめにリターンを打って指定してください。

「M」コマンドから抜け出るには、「.」を入力します。

## 【使用例】

>M 100						(100番地から変更)
Adrs	Hex	Dec	Binary	Ascii	Entry	
0100	:	00	0	00000000	^@ [H] = AB	(16進でABを入力)
0101	:	00	0	00000000	^@ [H] = .D 12	(10進で12を入力)
0102	:	00	0	00000000	^@ [D] = .B 10101010	(2進で10101010を入力)
0103	:	00	0	00000000	^@ [B] = .A T E S T	(文字列でTESTを入力)
0107	:	00	0	00000000	^@ [A] = -	(逆ステップモード)
0106	-	54	84	01010100	T [A] =	
0105	-	53	83	01010011	S [A] =	
0104	-	45	69	01000101	E [A] =	
0103	-	54	84	01010100	T [A] =	
0102	-	AA	170	10101010	エ [A] =	
0101	-	0C	12	00001100	^L [A] =	
0100	-	AB	171	10101011	オ [H] =	

## ●スクリーンモード時

### 【書 式】

M [ ADRS ]  
M 2 [ ADRS ]

### 【説 明】

「M 2」コマンドは、スクリーンモード2のダンプエリア2の変更に使います。

実際にダンプエリアに移行して、メモリの変更ができます。

ダンプエリアに表示されているリバーскарソルがカレントアドレスです。

1 6 進数の入力により値の変更が行えます。また、以下のキーが使用できます。

[←]	リバーскарソルを左へ移動
[→]	リバーскарソルを右へ移動
[↑]	リバーскарソルを上へ移動
[↓]	リバーскарソルを下へ移動
[ ]	コマンド入力モードに戻る

## 【使用例】

>SC 1																(スクリーンモード1にする)															
>M 100																(100番地から変更)															
		+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F														
0100	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	.....														
0110	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	.....														
0120	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	!"#\$%&'()*+,-./														
0130	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	0123456789:;<=>?														
0140	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	@ABCDEFGHIJKLMNO														
0150	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	PQRSTUVWXYZ[¥]^_														
0160	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	~abcdefghijklmno														
0170	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	pqrstuvwxyz{ }.														
0180	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	_____														
0190	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	_____														

>SC 2																(スクリーンモード2にする)															
>M2 1000																(1000番地から変更)															
0100				00 01 02 03				1000 CE 01 00 4F																							
0104				04 05 06 07				1004 5F A7 00 4C																							
0108				08 09 0A 0B				1008 08 5A 26 F9																							
010C				0C 0D 0E 0F				100C 20 FE 00 00																							
0110				10 11 12 13				1010 00 00 00 00																							
0114				14 15 16 17				1014 00 00 00 00																							
0118				18 19 1A 1B				1018 00 00 00 00																							
011C				1C 1D 1E 1F				101C 00 00 00 00																							
0120				20 21 22 23				1020 00 00 00 00																							
0124				24 25 26 27				1024 00 00 00 00																							

メモリ領域の満たし

【書 式】

F I < ADRS1 > < ADRS2 > [ DATA ]

【説 明】

このコマンドは指定したデータでメモリを埋めるものですが、通常はメモリをクリアする時に使用します。

「ADRS1」から「ADRS2」までの間を「DATA」で指定した値で満たすのですが、もし「DATA」を省略した場合は値は0になります。

「ADRS1」が「ADRS2」より大きい場合は何も行いません。

【使用例】

>FI 0 FFFF	(0 番地～F F F F 番地をクリア)
>FI 100 1FF 55	(1 0 0 番地～1 F F 番地を5 5で満たす)



## メモリのコピー

### 【書 式】

C < ADRS1 > < ADRS2 > < ADRS3 >

### 【説 明】

メモリ内容のブロックコピーを行うコマンドです。

このコマンドは、全てのパラメータを指定しなければいけません。

「ADRS1」でコピー元の先頭アドレスを、「ADRS2」で最終アドレスを指定します。  
そして、「ADRS3」にコピー先のアドレスを指定します。

「ADRS1」が「ADRS2」より大きい場合は何も行いません。

### 【使用例】

>C 100 103 200 (100番地～103番地を200番地へコピー)

## メモリの比較

### 【書 式】

MC < ADRS1 > < ADRS2 > < ADRS3 >

### 【説 明】

2つのブロックのメモリの比較を行います。

メモリコンペアのパラメータはコピーコマンドと同じで、[ADRS1] から [ADRS2] までの内容を [ADRS3] からの内容と比較します。

もし、メモリ内容が同じならば何も表示しません。違っていれば、

[SSSS : ss] [DDDD : dd]

と表示します。

「SSSS」はソース「ADRS1」側のアドレスでそのメモリ内容が「ss」です。

「DDDD」はディスティネーション「ADRS3」側のアドレスでそのメモリ内容が「dd」です。

スクリーンモード時は、画面を一度消去してから表示します。

何かのキーを押すと、スクリーンモードに戻ります。

### 【使用例】

>MC 100 1FF 200 (100番地～1FF番地を200番地からの内容と比較)

## データの検索

### 【書 式】

F < ADRS1 > < ADRS2 > < LIST >

### 【説 明】

指定した文字列や数値を検索するコマンドです。

「ADRS1」から「ADRS2」までの内容と「LIST」で指定した内容とを1バイトずつ比べ、同じならばそのアドレスを表示します。

「LIST」は以下のように指定します。

```
F 0 FFF 1 2 3                (アンダーラインの所が「LIST」に相当)
```

この例では、「0～FFF番地の中から「1,2,3」と連続してデータが入っているアドレスを探せ」ということです。数値の指定は1バイト単位で区切って指定します。

また、文字列「ABC」を探したい場合は、

```
F 0 FFF "ABC"
```

と、「"」（ダブルコーテーション）で囲んで指定します。

ワイルドカードとして「?」も使用できます。例えば、

```
F 0 FFF "TE?ST"
```

とすると、「TE1ST」でも「TE2ST」でも、3文字目は何でも良いといった指定になります。また、

```
F 0 FFF "TE" ? "ST"
```

と指定しても結果は同じです。ただし、「"」と「?」の間には必ずスペースを空けてください。数値の指定の場合でも同じです。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

```
>F 0 2FF "TEST"                (0番地～2FF番地の中から「TEST」を検索)  
0100                          (100番地にあり)  
0200                          (200番地にあり)
```

## アセンブル

### ●ノーマルモード時

#### 【書 式】

A    [ ADRS ]

#### 【説 明】

アセンブルを行います。

指定した「ADRS」からアセンブリ入力が行えます。

「ADRS」を省略した場合は、前回のアセンブル作業の終了位置からになります。

このコマンドを実行するとアドレスを表示し、アセンブリ言語入力モードになります。  
続いて、命令を入力してください。

アセンブリ入力モードでの書式は以下の通りです。

①シンボル

② [シンボル]    ニーモニック    [オペランド]

①の書式ではカレントアドレスにシンボルを登録します。最初の文字は「.」でないと  
いけません。カレントアドレスはそのままです。「K」コマンドと同様の働きをします。

②の書式では1パスのアセンブルを行います。

ニーモニックは各CPUで異なりますので、第6章を参照してください。

オペランドは、ニーモニックに従属します。普通はアドレス（数値またはシンボル）や  
データを指定します。アドレッシングの詳細も第6章を参照してください。

最初に「    (リターン)」を入力するとアセンブル作業は終了します。

### 【使用例】

「S 0 3」の場合

>A 100	(100番地からアセンブル)
0100 LDX #\$1000	
0103 CLRA	
0104 .LOOP STAA 0,X	
0106 INCA	
0107 INX	
0108 CPX #\$2000	
010B BLO .LOOP	
010D RTS	
010E	(アセンブル作業終了)

「S Z 8 0」の場合

>A 100	(100番地からアセンブル)
0100 LD IX,1000H	
0104 XOR A	
0105 .LOOP LD (IX+0),A	
0108 INC A	
0109 INC IX	
010B CP 0	
010D JR NZ,.LOOP	
010F RET	
0110	(アセンブル作業終了)

### ●スクリーンモード時

#### 【書 式】

A [ ADRS ]

#### 【説 明】

スクリーンモードでも、使用法はノーマルモードとまったく同じです。

逆アセンブルエリアを消去しその中でアセンブル作業が行えます。

1ラインを入力すると、すぐにコードが表示されます。

また、最初に「 (リターン)」を入力すると、カレントアドレスを逆アセンブルして、次のアドレスに進みます。

アセンブルエラーが発生した場合は、ブザー音を鳴らしその行を消去します。

「 (エスケープ)」でアセンブル作業を終了します。

**【使用例】**

「S 0 3」の場合

>A 100	( 1 0 0 番地からアセンブル)
0100 00 LDX #\$1000	(アセンブリ入力)
0100 CE 10 00 LDX   #\$1000 0103 00	( 1 ライン入力するとコードが表示される)

「S Z 8 0」の場合

>A 100	( 1 0 0 番地からアセンブル)
0100 00 LD IX, 1000H	(アセンブリ入力)
0100 DD 21 00 10 LD   IX, 1000H 0104 00	( 1 ライン入力するとコードが表示される)

## 逆アセンブル

### ●ノーマルモード時

#### 【書 式】

```
U      [ ADRS1 [ ADRS2 ]]  
U F    < F-NAME > < ADRS1 > < ADRS2 >
```

#### 【説 明】

メモリの逆アセンブルを行います。

「ADRS1」から「ADRS2」までの間を逆アセンブルして画面に表示します。

「U F」コマンドは画面に表示せず、「F-NAME」で指定したファイル名でディスクに書き出すコマンドです。どのパラメータも省略はできません。

どちらのコマンドも、表示（書き込み）途中でも「 C」を押すと終了します。

「U」コマンドでは、「ADRS1」の指定を省略した場合、前回の逆アセンブルの終了位置から始まります。「ADRS2」を省略した場合は、そこから16ステップ分表示します。

もちろん、「ADRS2」を指定した場合はそこまで逆アセンブル表示を行います。

表示は以下の2タイプがあります。

- ①アドレス オペコード ニーモニック [オペランド]
- ②アドレス オペコード シンボル ニーモニック [オペランド]

②は次の条件の時に表示されます。

- 1) 「L」コマンドや「L S」コマンドでシンボルをロードしたとき
- 2) 「K」コマンドまたは「A」コマンドでシンボルを登録したとき

①は上記以外（つまり一度もシンボルを使用しない）の場合、または「B C」コマンドでシンボルを全て抹消した時に表示されます。

もし、オペコードが不当なコードであればニーモニック欄に「Illegal Opcode」と表示されます。

## 【使用例】

「S 0 3」の場合

>U 100,111		(1 0 0 ~ 1 1 1 番地を表示)
0100 CE 10 00	LDX	#\$1000
0103 4F	CLRA	
0104 A7 00 .LOOP	STAA	0,X
0106 4C	INCA	
0107 08	INX	
0108 8C 20 00	CPX	#\$2000
010B 25 F7	BCS	.LOOP
010D 39	RTS	
010E 00	Illegal Opcode	
010F 00	Illegal Opcode	
0110 00	Illegal Opcode	
0111 00	Illegal Opcode	
>UF TEST.ASM 100 10D		(「TEST.ASM」に書き出し)

「S Z 8 0」の場合

>U 100,111		(1 0 0 ~ 1 1 1 番地を表示)
0100 DD 21 00 10	LD	IX,1000H
0104 AF	XOR	A
0105 DD 77 00 .LOOP	LD	(IX+00H),A
0108 3C	INC	A
0109 DD 23	INC	IX
010B FE 00	CP	00H
010D 20 F6	JR	NZ,.LOOP
010F C9	RET	
0110 00	NOP	
0111 00	NOP	
>UF TEST.ASM 100 10F		(「TEST.ASM」に書き出し)



## ●スクリーンモード時

### 【書 式】

U        [ ADRS1 ]  
U F    < F-NAME >    < ADRS1 >    < ADRS2 >

### 【説 明】

メモリの逆アセンブルを行います。

「U F」コマンドはノーマルモード時とまったく同じです。

「U」コマンドは、スクリーンモード時はかなり操作方法が異なります。  
コマンドを実行すると、逆アセンブルエリアに 12 ステップ分表示します。そして、一番上の行をリバース表示します（そのアドレスをカレントアドレスと言います）。その時点で、以下のキーが使用できます。

[↑]	カレントアドレスを1バイト減らして表示
[↓]	次の行をカレントアドレスにして表示
[←]	カレントアドレスを16バイト減らして表示
[→]	一番下の行をカレントアドレスにして表示
[ ]	コマンド入力モードに戻る

[ ] や [ ] は正常に逆アセンブルできますが、逆にさかのぼると逆アセンブル位置の誤りで正常に表示できない場合が発生します。その場合は [ ] を数回入力して1バイトずつさかのぼると正常な逆アセンブル表示になります。

※ブレークポイントがセットされているアドレスには、逆アセンブルリストの後ろにそれぞれのブレークポイントのマークが表示されます。

# 【使用例】

「S 0 3」の場合

>U 100			
0100 CE 10 00	LDX	#\$1000	CC=C0 hinzvc
0103 4F	CLRA		A =00
0104 A7 00 . LOOP	STAA	0, X	B =00
0106 4C	INCA		X =0000
0107 08	INX		SP=0000
0108 8C 20 00	CPX	#\$2000	PC=0100
010B 25 F7	BCS	. LOOP	
010D 39	RTS		1 5
010E 00	Illegal Opcode		2 6
010F 00	Illegal Opcode		3 7
0110 00	Illegal Opcode		4 8
0111 00	Illegal Opcode		

この状態で [ ] [ ] [ ] [ ] [ ] [ ] が使用できます。

「S Z 8 0」の場合

>U 100			
0100 DD 21 00 10	LD	IX, 1000H	
0104 AF	XOR	A	F =00 00
0105 DD 77 00 . LOOP	LD	(IX+00H), A	A =00 00
0108 3C	INC	A	BC=0000 0000
0109 DD 23	INC	IX	DE=0000 0000
010B FE 00	CP	00H	HL=0000 0000
010D 20 F6	JR	NZ, . LOOP	IX=0000
010F C9	RET		IY=0000 1 5
0110 00	NOP		SP=0000 2 6
0111 00	NOP		PC=0100 3 7
0112 00	NOP		I =00 4 8
0113 00	NOP		R =00

この状態で [ ] [ ] [ ] [ ] [ ] [ ] が使用できます。

## 5-4 ロード・セーブ関係

### オブジェクトのロード

#### 【書 式】

L < F-NAME >

#### 【説 明】

指定したファイルをアドレス情報を表示しながらロードします。

スクリーンモード時は、一度画面を消去して表示します。

ロードするオブジェクト形式は、ファイルで自動的に判別しますので、SフォーマットファイルもインテルHEXファイルもこのコマンドで読み込めます。

拡張子を省略した場合は、「S03」では「.S」を、「SZ80」では「.HEX」を自動的に付けます。

オブジェクトファイルが読み終ると、続いてシンボルファイルのロードに入ります。

シンボルファイルは自動的に「.MAP」を付け読み込みを開始しますが、もちろん別のファイル名であればエラーとなり、その時点で終了します。

その場合は「LS」コマンドでシンボルのロードを行ってください（「5-6」参照）。

#### 【使用例】

```
>L SAMPLE
```

```
1000-1012
```

```
Found End record.
```

```
Top Address = 1000
```

```
End Address = 1012
```

```
Execute Address = 1000
```

スクリーンモード時はこの表示はしない

## ロードアドレスの表示

### 【書 式】

L S

### 【説 明】

ロードしたオブジェクトの先頭アドレス、最終アドレス、実行アドレスを表示します。

スクリーンモード時は、常にこれらの情報が表示されている為、このコマンドは無視されます。

### 【使用例】

>LA	
Top Address = 1000	(先頭アドレスは1 0 0 0 番地)
End Address = 1012	(最終アドレスは1 0 1 2 番地)
Execute Address = 1000	(実行アドレスは1 0 0 0 番地)

## モトローラ S フォーマットでのセーブ

### 【書 式】

S     < F-NAME >   < ADRS1 >   < ADRS2 >   [ ADRS3 ]  
S S   < F-NAME >   < ADRS1 >   < ADRS2 >   [ ADRS3 ]

### 【説 明】

「MOTOROLA EXORMACS」フォーマット（通称 S フォーマット）の形でディスクにセーブします。

「F-NAME」で書き込みたいファイル名を指定し、「ADRS1」で先頭アドレス「ADRS2」で最終アドレスを指定します。「ADRS3」は実行アドレスを指定しますが、省略した場合は「ADRS1」と同じアドレスになります。

ファイル名の拡張子を省略した場合、自動的に「. S」を付けます。

もちろん、このコマンドでセーブしたファイルは、L コマンドで読み込むことができます。

「S」コマンドは、「S O 3」の場合に限り使用できます。

### 【使用例】

>SS TEST.0 1000 4FFF

(「TEST. O」という名で1000番地から  
4FFF番地の内容をSフォーマットでセーブ  
実行アドレスは1000番地)

「S O 3」の場合

>S TEST 0 FF 10

(「TEST. S」という名で0番地からFF番  
地の内容をSフォーマットでセーブ  
実行アドレスは10番地)

## インテルHEXフォーマットでのセーブ

### 【書 式】

S     < F-NAME >   < ADRS1 >   < ADRS2 >   [ ADRS3 ]  
S I   < F-NAME >   < ADRS1 >   < ADRS2 >   [ ADRS3 ]

### 【説 明】

「INTELLEC HEX」フォーマット（通称Sフォーマット）の形でディスクにセーブします。

「F-NAME」で書き込みたいファイル名を指定し、「ADRS1」で先頭アドレス「ADRS2」で最終アドレスを指定します。「ADRS3」は実行アドレスを指定しますが、省略した場合は「ADRS1」と同じアドレスになります。

ファイル名の拡張子を省略した場合、自動的に「. HEX」を付けます。

もちろん、このコマンドでセーブしたファイルは、Lコマンドで読み込むことができます。

「S」コマンドは、「SZ80」の場合に限り使用できます。

### 【使用例】

>SS TEST.0 1000 4FFF

(「TEST. O」という名で1000番地から  
4FFF番地の内容をHEXフォーマットでセーブ。実行アドレスは1000番地)

「SZ80」の場合

>S TEST 0 FF 10

(「TEST. HEX」という名で0番地から  
FF番地の内容をHEXフォーマットでセーブ  
実行アドレスは10番地)

## 5-5 ブレークポイント関係

ブレークポイントは、異なったタイプのものが4種類あります。

「BS」コマンドは、通常のブレークポイントですが、他の3つはブレークポイントと言いながらも停止はしません。

それぞれのブレークポイントで、設定できるアドレスは最高20個までです。

スクリーンモードでは、それぞれのブレークポイントの設定数が常時表示しています。このモードでは、「BS」以外はすべてブレークポイントの名称とアドレスを表示するだけの動作になります。

同じアドレスに各々のブレークポイントを設定場合、全ての機能（各々のブレークポイントの動作）を実行しますが、この場合は以下のように優先順位があります。

BT → BQ → BR → BS                      (まず「BT」が働き、最後に「BS」が働く)

### 【注 意】

ブレークポイントは、どこのアドレスに設定しても良い訳ではありません。

「S03」の場合

0100 86 01	LDAA	#\$1
0102 B7 10 00	STAA	\$1000
0105 39	RTS	

「SZ80」の場合

0100 3E 01	LD	A, 01H
0102 32 00 10	LD	(1000H), A
0105 C9	RET	

上記のように100番地から105番地までを実行させたい場合、ブレークポイントは必ず105番地に設定します。

上記のプログラムで実際に設定可能なアドレスは、「100」「102」「105」の3つです。それ以外の場所（「104」など）に設定してもブレークポイントは掛かりません。そればかりではなく、異常な動作をする可能性が大いにあります。

シミュレーション開始時に、ブレークポイントのアドレスに特別なコードをセットします（「S03」では「CF」、「SZ80」では「76」）。

そして、命令を実行している最中に、このコードをフェッチした時、ブレークポイントのアドレスかどうかを判断してそれぞれの動作を行うようにしています。

## 停止するブレークポイントの設定・表示

### 【書 式】

```
B S    [ ADRS1 [ ADRS2 [ ADRSx..... ]]]  
B      [ ADRS1 [ ADRS2 [ ADRSx..... ]]]
```

### 【説 明】

オーソドックスなブレークポイントコマンドです。

「B」コマンドでもパラメータを指定すると、「B S」コマンドと同じ動作をします。

シミュレーションを実行し、指定したアドレスにプログラム・カウンタがぶつかると、そこでシミュレーションを停止します。

「ADRS1」にブレークポイントをかけたアドレスを指定しますが、コマンドラインで一度にいくつかのアドレスが設定できます。

パラメータを指定しなかった場合、「B S」コマンドでセットしたアドレスを全て表示します。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

```
>BS 100 104 106  
>B 200  
>BS  
0100  
0104  
0106  
0200  
>
```



## 全レジスタを表示するブレークポイントの設定・表示

### 【書 式】

B R [ ADRS1 [ ADRS2 [ ADRSx..... ]]]

### 【説 明】

指定したアドレスにぶつかると、全レジスタを表示し、シミュレーションを続行するというブレークポイントです。

パラメータの指定法などは、「B S」コマンドと同じです。

パラメータを指定しなかった場合、「B R」コマンドで設定したアドレスを全て表示します。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

```
>BR 100 104 106 200
>BR
0100
0104
0106
0200
>
```

## 指定レジスタを表示するブレークポイントの設定・表示

### 【書 式】

B Q [ < ADRS > < REG-LIST > ]

### 【説 明】

指定したアドレスにぶつかると、指定したレジスタの値のみを表示し、シミュレーションを続行するというブレークポイントです。

「ADRS」でアドレスを指定し、「REG-LIST」で表示したいレジスタを指定します。

「REG-LIST」では同時にいくつものレジスタが指定できます。

レジスタ名については、第 6 章を参照してください。

パラメータを指定しなかった場合、「B Q」コマンドで設定したアドレスを全て表示します。

スクリーンモード時は、画面を一度消去してから表示します。

指定したレジスタの内容のみを表示しますので、例えば 8 ビットレジスタであれば、「1 2」としか表示しません。レジスタ名も表示させたい場合は、B T コマンドと組み合わせて使用します。

また、改行も行ないませんので改行したい場合は、B T コマンドと組み合わせて使用してください。

例えば、

```
BQ 1000 A
BT 1000 "A="
```

と指定すると、ディスプレイには「A = x x」（x x は 1 6 進数）と表示します。

### 【使用例】

```
>BQ 100 A B
>BQ 200 SP
>BQ
  0100 A B
  0200 SP
>
```

## 指定文字列を表示するブレイクポイントの設定・表示

### 【書 式】

B T    [ < ADRS >   < STRING > ]

### 【説 明】

指定したアドレスにぶつかると、指定した文字列を表示し、シミュレーションを続行するというブレイクポイントです。

パラメータを省略した場合は、「B T」コマンドで設定したアドレスを全て表示します。

スクリーンモード時は、画面を一度消去してから表示します。

「ADRS」でアドレスを指定し、「STRING」で表示したい文字列を指定します。

文字列を表示した後は自動的に「C R / L F」を付けません。

もし、改行したい場合は、

```
BT 1000 Break¥0D¥0A
```

と、「¥」に続いて16進で直接指定してください。数値は必ず2桁で指定します。

また、文字列の中にはスペースを許していません（スペース以降の文字列は無視されます）。

もし、スペースを入れたい場合は、

```
BT 1000 Break¥20Point¥0D¥0A
```

と、これもまた「¥20」と直接コードで指定してください。

### 【使用例】

```
>BT 100 OUTCHAR=  
>BT 1000 Areg=  
>BT  
  0100 OUTCHAR=  
  1000 Areg=  
>
```

## すべてのブレイクポイントの表示

### 【書 式】

B

### 【説 明】

ブレイクポイントが設定されているアドレスをすべて表示します。  
つまり、「BS」「BR」「BQ」「BT」で設定したアドレスです。  
それぞれの名称は以下の通りです。

BS 「Stop Break Point」

BR 「All Register Display Break Point」

BQ 「Register Display Break Point」

BT 「String Display Break Point」

これらのメッセージに続いて、アドレスを表示します。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

```
>B 100
>BS 120 14F
>BR 134
>BQ 150 A
>BT 150 OUTCHAR=
>B
Stop Break Point Address
  0100
  0200
  014F
All Register Display Break Point Address
  0134
Register Display Break Point Address
  0150 A
String Display Break Point Address
  0150 OUTCHAR=
```

## ブレークポイントの解除

### 【書 式】

B C [ ADRS1 [ ADRS2 ]]

### 【説 明】

「B S」「B R」「B Q」「B T」コマンドで設定したブレークポイントをキャンセルします。

パラメータを省略した場合は、全てのブレークポイントをキャンセルします。

「ADRS1」だけを指定した場合は、そのアドレスと同じブレークポイントをキャンセルします。

「ADRS2」を指定した場合は、「ADRS1」から「ADRS2」までの範囲にあるブレークポイントを全てキャンセルします。

### 【使用例】

```
>B 100
>BS 120 14F
>BR 134
>BQ 150 A
>BT 150 OUTCHAR=
>B
Stop Break Point Address
  0100
  0200
  014F
All Register Display Break Point Address
  0134
Register Display Break Point Address
  0150 A
String Display Break Point Address
  0150 OUTCHAR=
>BC 140 200          ( 1 4 0 番地から 2 0 0 番地までのブレークポイントを解除)
>B
Stop Break Point Address
  0100
  0120
All Register Break Point Address
  0134
```

## 5 - 6 シンボル関係

シンボルファイルのロード

### 【書 式】

L S < F-NAME >

### 【説 明】

シンボルファイルをロードします。

拡張子を省略した場合は自動的に「. MAP」を付けます。

「L」コマンドでシンボルファイルをロード出来たにもかかわらず、「LS」コマンドを同じファイルに対して行った場合、シンボルは二重に登録してしまいます。ただし、シンボル格納エリアが減るだけで、デバッグには支障ありません。

シンボルの文字数は最大 11 文字、登録できるシンボル数は最大 4000 個です。

### 【使用例】

>LS TEST

(「TEST. MAP」をロード)

## シンボルの登録

### 【書 式】

K < ADRS > < LABEL >

### 【説 明】

シンボルの登録を行います。

「ADRS」で指定したアドレスに「LABEL」で指定したシンボル名を付けます。

シンボル名は、アルファベット（A～Z，a～z），数字，記号（「?」，「\_」）で構成します。

このコマンドで登録したシンボルは、コマンドライン，アセンブラ，そして計算コマンドで使用できます。その場合は、頭に「.」（ピリオド）を付けて「シンボル名だ」ということを明示します。

例えば、

```
K 1234 ABCDEFG
```

と「A B C D E F G」を登録した場合は、

```
U . ABCDEFG ( 1 2 3 4 番地から逆アセンブルする)
```

と、シンボル名での指定ができます。

同じアドレスに対し2つ以上のシンボル名を登録した場合、逆アセンブルなどアドレスからシンボル名を探すようなコマンドでは、内部テーブルの順によってどちらが採用されるかは明らかではありません。しかし、アセンブルなどのシンボル名からアドレスを探すコマンドなどでは、普通に使用できます。

### 【使用例】

```
>K 100 ABC ( 1 0 0 番地にシンボル「A B C」を設定)
>K 123 AAA ( 1 2 3 番地にシンボル「A A A」を設定)
>K 102 XYZ ( 1 0 2 番地にシンボル「X Y Z」を設定)
>KA (アドレス順にシンボルを表示)
ABC 0100 XYZ 0102 AAA 0123
>KL (シンボル名順にシンボルを表示)
AAA 0123 ABC 0100 XYZ 0102
>
```

## アドレス順でのシンボルの表示

### 【書 式】

K A [ ADRS1 [ ADRS2 ]]

### 【説 明】

登録したシンボルをアドレス順にソートして表示します。

パラメータを省略した場合は、登録したシンボルを全て表示します。

「ADRS1」を指定した場合は、そのアドレス以上のシンボルを表示します。

「ADRS2」も指定した場合は、そのアドレスまでのシンボルを表示します。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

>K 100 ABC	( 1 0 0 番地にシンボル「A B C」を設定)
>K 123 AAA	( 1 2 3 番地にシンボル「A A A」を設定)
>K 102 XYZ	( 1 0 2 番地にシンボル「X Y Z」を設定)
>KA	(アドレス順にシンボルを表示)
ABC            0100    XYZ            0102    AAA            0123	
>KA 110	(アドレス順に 1 1 0 番地以降のシンボルを表示)
AAA            0123	
>	



## シンボル名順でのシンボルの表示

### 【書 式】

K L [ LABEL1 [ LABEL2 ]]

### 【説 明】

登録したシンボルをアルファベット順にソートして表示します。

パラメータを省略した場合は、登録したシンボルを全て表示します。

「LABEL1」を指定した場合はそのシンボル名以上（アルファベット順で）のシンボルを表示します。

「LABEL2」も指定した場合は、そのシンボル名までのシンボルを表示します。

スクリーンモード時は、画面を一度消去してから表示します。

### 【使用例】

>K 100 ABC	( 1 0 0 番地にシンボル「A B C」を設定)
>K 123 AAA	( 1 2 3 番地にシンボル「A A A」を設定)
>K 102 XYZ	( 1 0 2 番地にシンボル「X Y Z」を設定)
>KL	(シンボル名順にシンボルを表示)
AAA            0123    ABC	0100    XYZ            0102
>KL AAB	(シンボル名順に「A A B」以降の名のシンボルを表示)
ABC            0100    XYZ	0102
>	

## シンボルの消去

### 【書 式】

K C

### 【説 明】

登録したシンボルを全て消去します。

確認の問い合わせをしませんので、十分に気を付けて使用してください。

### 【使用例】

>K 100 ABC	( 1 0 0 番地にシンボル「A B C」を設定)
>K 123 AAA	( 1 2 3 番地にシンボル「A A A」を設定)
>K 102 XYZ	( 1 0 2 番地にシンボル「X Y Z」を設定)
>KA	(アドレス順にシンボルを表示)
ABC            0100    XYZ            0102    AAA            0123	
>KL	(シンボル名順にシンボルを表示)
AAA            0123    ABC            0100    XYZ            0102	
>KC	(シンボルを全て消去)
>KA	
>KL	
>	

## ステップ実行

## 【書 式】

T [ COUNT ]

## 【説 明】

ステップごとのシミュレーションを行います。

すなわち、1 ステップ実行してレジスタを表示するということです。

このコマンドでは、ブレークポイントは関係なくそのまま実行します。

パラメータを省略した場合は、1 ステップのみの実行を行います。

「COUNT」を指定した場合、その数だけステップ実行を行います。「COUNT」は、最高 F F F F ( 6 5 5 3 5 ) まで指定できます。

もし、途中でシミュレーションを止めたい場合（一時停止）は、「 S 」を押します。

もう 1 度押すと再開します。

また、ステップ実行を中止させたい場合は、「 C 」を押してください。

連続のステップ実行では、割り込みコントロールもできます。

(割り込みコントロールに付きましては第 6 章を参照してください。)

スクリーンモード時でも同じような動作をしますが、レジスタや逆アセンブルなどは、それぞれの表示エリアに表示します。逆アセンブルエリアでリバース表示している行がカレント位置になります。

## 【使用例】

「 S 0 3 」の場合

```
>T 5                                (現在のプログラムカウンタから 5 ステップ実行)
CC=C8(hiNzvc) A=00 B=00 X=0000 SP=BFFF PC=0303 LDAA #$1
CC=C0(hinzvc) A=01 B=00 X=0000 SP=BFFF PC=0305 LDAB #$FF
CC=C8(hiNzvc) A=01 B=FF X=0000 SP=BFFF PC=0307 LDX  #$8000
CC=C8(hiNzvc) A=01 B=FF X=8000 SP=BFFF PC=030A ADDA #$0
CC=C0(hinzvc) A=01 B=FF X=8000 SP=BFFF PC=030C LDAB $0
```

「S Z 8 0」の場合

>T 5	(現在のプログラムカウンタから 5 ステップ実行)
----- A=00 B=0000 D=0000 H=0000 S=BFFF P=0103 I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD A, 01H	
----- A=01 B=0000 D=0000 H=0000 S=BFFF P=0105 I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD B, FFH	
----- A=00 B=FF00 D=0000 H=0000 S=BFFF P=0107 I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD IX, 8000H	
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=010B I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=8000 Y=0000 ADD A, 00H	
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=010D I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=8000 Y=0000 LD A, (0000H)	

## シミュレーション

### 【書 式】

G [ ADRS1 ] [ ADRS2 ]

### 【説 明】

シミュレーション実行を行います。

最初のデバッグは、この「G」コマンドが無難でしょう。なぜならユーザープログラムが暴走した場合でも「 C」を押すとその時点で止まるからです。また、スペースキーを押すとその時点のレジスタなども確認できます。

また、割り込み関係のキーオペレーションも動作します（第6章で詳しく説明します）。ヒストリスイッチがONの場合（詳しくはHコマンドを参照）、1ステップごとにレジスタ値をヒストリバッファへ格納していきます。

「ADRS1」を指定しなかった場合は、現在のプログラムカウンタから実行を開始します。指定した場合は、そこから実行を開始します。

「ADRS2」を指定すると、一時的にそのアドレスにブレークポイントを設定します。つまり、実行を開始しプログラムカウンタがそのアドレスになった場合、その時点で実行を停止します。もちろん、ブレークポイントコマンドで設定したブレークポイントも効きます。

スクリーンモード時でも同じような動作をします。このコマンドが終了した時、メモリ内容、レジスタ値などを現在の値に表示しなおします。

### 【使用例】

「S O 3」の場合

```
>G 300 30C (300番地から30C番地まで実行)
CC=C0(hinzvc) A=01 B=FF X=8000 SP=BFFF PC=030C LDAB $0
```

「S Z 8 0」の場合

```
>G 100 10D (100番地から10D番地まで実行)
----- A=00 B=FF00 D=0000 H=0000 S=BFFF P=010D I=00XX R=00
----- A'00 B'0000 D'0000 H'0000 X=8000 Y=0000 LD A,(0000H)
```

## 高速シミュレーション

### 【書 式】

J    [ ADRS1 ]    [ ADRS2 ]

### 【説 明】

高速にシミュレーション実行を行います。

パラメータは、「G」コマンドと同じです。

「G」コマンドでは1ステップを実行した後、キーボードの入力状態を見に行きますが、このコマンドではそれを行いません。また、ヒストリ処理も一切行いません。その為、その分高速になっています。ですから、ユーザープログラムが暴走した場合はどうにもなりません。そうなったらリセットボタンを押してください。

その為、必ずブレークポイント（停止するブレークポイント）を設定しておいてください。その他のブレークポイントでは止まりませんが、何かを表示した時にすぐに「 C」を押すと止まる可能性があります（「 C」は、文字を画面に表示した時に押しているかどうか調べている為）。また、イリーガル命令をフェッチしてもシミュレーションを停止し、コマンド入力モードに戻ります。

いずれにしてもこのコマンドは、完全にデバッグが済んでいるルーチンをシミュレートする場合などに適しています。

スクリーンモード時でも同じような動作をします。このコマンドが終了した時、メモリ内容、レジスタ値などを現在の値に表示しなおします。

### 【使用例】

「S 0 3」の場合

```
>J 300 38B (300番地から38B番地まで実行)
CC=C9(hiNzvC) A=80 B=80 X=8000 SP=BFFF PC=038B LDX #8000
```

「S Z 8 0」の場合

```
>J 100 18B (100番地から18B番地まで実行)
----NC A=80 B=8000 D=0000 H=0000 S=BFFF P=038B I=00XX R=00
----- A'00 B'0000 D'0000 H'0000 X=8000 Y=0000 LD IX,8000H
```

# スクリーンモード用シミュレーション 1

## 【書 式】

J S    [ ADRS1 ]    [ ADRS2 ]

## 【説 明】

スクリーンモード用のシミュレーションコマンドです。  
現在実行しているアドレス（P Cレジスタの値）を表示しながら実行します。  
キープレーク、ヒストリ処理なども行います。

パラメータは、「G」コマンドと同じです。

## 【使用例】

「S O 3」の場合

>JS , 605				( 3 0 0 ~ 6 0 5 を実行 )	
0300	8E E0 00	. START	LDS #. STACK	CC=C0 hinzvc	
0303	86 01		LDAA #\$1	A =00	
0305	C6 FF		LDAB #\$FF	B =00	
0307	CE 80 00		LDX #. INDEX	X =0000	
030A	8B 00		ADDA #\$0	SP=0000	
030C	D6 00		LDAB \$0	PC=0300	
030E	9B 00		ADDA \$0		
0310	E6 00		LDAB 0, X	1 5	
0312	AB 00		ADDA 0, X	2 6	
0314	F6 01 00		LDAB . EXTEND	3 7	
0317	BB 01 00		ADDA . EXTEND	4 8	
031A	CB 01		ADDB #\$1		

「S Z 8 0」の場合

>JS , 1F				( 0 ~ 1 F を実行 )	
0000	00	. XZ80	NOP	szhpnc	
0001	01 84 05		LD BC, . NN	F =00 00	
0004	02		LD (BC), A	A =00 00	
0005	03	. INDEX	INC BC	BC=0000 0000	
0006	04		INC B	DE=0000 0000	
0007	05		DEC B	HL=0000 0000	
0008	06 20		LD B, 20H	IX=0000	
000A	07		RLCA	IY=0000 1 5	
000B	08		EX AF, AF'	SP=0000 2 6	
000C	09		ADD HL, BC	PC=0000 3 7	
000D	0A		LD A, (BC)	I =00 4 8	
000E	0B		DEC BC	R =00 ⑪	

## スクリーンモード用シミュレーション 2

### 【書 式】

G S    [ ADRS1 ]    [ ADRS2 ]

### 【説 明】

スクリーンモード用のシミュレーションコマンドです。

すべての変更内容（メモリ内容、レジスタ値など）をリアルタイムに表示しながら実行します。

キーブレイク、ヒストリ処理なども行います。

パラメータは、G コマンドと同じです。

### 【使用例】

「S 0 3」の場合

>GS , 100C																( 1 0 0 0 ~ 1 0 0 C を実行 )													
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0140	00	00	00	00	00	00	00	0	①	0	00	00	00	00	00	00	.....	②											
0150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
0190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....												
1000	CE	01	00	. START				LDX	#. RAMTOP				CC=C0 hinzvc																
1003	4F					CLRA				A =00																			
1004	5F					CLRB				B =00 ④																			
1005	A7	00	. LOOP				STAA	0, X				X =0000																	
1007	4C					INCA				SP=0000																			
1008	08					INX				③				PC=1000															
1009	5A					DECB																							
100A	26	F9					BNE	. LOOP								1 5													
100C	20	FE					BRA	\$100C								2 6													
1009	00					Illegal Opcode												3 7											
100A	00					Illegal Opcode												4 8											
100B	00					Illegal Opcode																							

※①～④のエリアがリアルタイムに変化します。



「S Z 8 0」の場合（スクリーンモード2）

>GS ,1010				(1 0 1 0 ~ 1 0 0 Cを実行)			
0100 00 00 00 00	0118 00 00 00 00	RAMTOP	0100 (0000) [0000]				
0104 00 00 00 00	011C 00 00 00 00						
0108 00 00 00 00	0120 00 00 00 00						
010C 00 00 00 00	0124 00 00 00 00						
0110 0 ① 0 00	0128 0 ② 0 00		③				
0114 00 00 00 00	012C 00 00 00 00						
0118 00 00 00 00	0130 00 00 00 00						
011C 00 00 00 00	0134 00 00 00 00						
0120 00 00 00 00	0138 00 00 00 00						
0124 00 00 00 00	013C 00 00 00 00						
1000 DD 21 00 01 . START	LD IX, RAMTOP		⑤ szhpnc				
1004 3E 00	LD A, 00H		F =00 00				
1006 06 00	LD B, 00H		A =00 00				
1008 DD 77 00	LD (IX+00H), A		BC=0000 0000				
100B 3C	INCA		DE=0000 0000				
100C DD 23	INC IX		HL=0000 0000				
100E 10 F8	DJNZ . LOOP ④		IX=0000				
1010 C3 10 10	JP 1010H		IY=0000 1 5				
1013 00	NOP		SP=0000 2 6				
1014 00	NOP		PC=1000 3 7				
1015 00	NOP		I =00 4 8				
1016 00	NOP		R =00				

※①～⑤エリアがリアルタイムに変化します。

## レジスタ値の設定・表示

## 【書 式】

```
R    [ REG1=VAL1    [ REG2=VAL2 ] .....]
```

## 【説 明】

レジスタの設定を行います。

「レジスタ名=数値」または「レジスタ名 数値」と指定します（レジスタ名については第6章を参照してください）。

1回のコマンドラインで、複数のレジスタの設定もできます。

```
R A=1 B=2 X=3
R BC=2345 IX=5678
```

パラメータを指定しなかった場合は、全レジスタの表示を行います。

ただし、スクリーンモード時は、全レジスタの設定を行います。

レジスタ表示エリアにカーソルが移動します。カーソルキーでカーソルの移動が行え、16進数値の入力で値の設定ができます。

また、フラグの設定では、「0」または「1」でオン、オフが行えます。

「 」でコマンド入力モードに戻ります。

## 【使用例】

「S 0 3」の場合

```
>R                                (レジスタを表示)
CC=C0(hinzvc) A=00 B=00 X=0000 SP=0000 PC=0000 LDAA #$1
>R PC=100                        (PCレジスタに100をセット)
>R                                (レジスタを表示)
CC=C0(hinzvc) A=00 B=00 X=0000 SP=0000 PC=0100 LDX  #$1000
```

「SZ80」の場合

>R	(レジスタを表示)
----- A=00 B=0000 D=0000 H=0000 S=0000 P=0000 I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD A, 01H	
>R PC=100	(PCレジスタに100をセット)
>R	(レジスタを表示)
----- A=00 B=0000 D=0000 H=0000 S=0000 P=0100 I=00XX R=00	
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD IX, 0100H	

「S03」の場合 (スクリーンモード)

>R	
CC=C0 hinzvc	
A =00	
B =00	
X =0000	
SP=0000	
PC=1000	
	で自由に移動ができる
	「 」で終了

「SZ80」の場合 (スクリーンモード)

>R	
szhpnc	
F =00 00	
A =00 00	
BC=0000 0000	
DE=0000 0000	
HL=0000 0000	
IX=0000	
IY=0000 1 5	
SP=0000 2 6	
PC=1000 3 7	
I =00 4 8	
R =00	
	で自由に移動ができる
	「 」で終了

## 現在のレジスタの保存

### 【書 式】

R S [ No. ]

### 【説 明】

現在のレジスタ値を保存します。

繰り返し同じ所をシミュレートする場合に便利です。

「No.」には、0～8までの数値が指定できます。つまり、保存場所を9つ持っているのです。異なった値をそれぞれの場所に保存できます。

「No.」を省略すると0が選ばれます。

デバッガを立ち上げた後のレジスタ値は、保存域0に格納されます。

スクリーンモード時では、レジスタ保存状態エリアで保存状態を知らせます。

### 【使用例】

「S 0 3」の場合

```
>R PC=100                (P Cレジスタに1 0 0をセット)
>R                        (レジスタを表示)
CC=C0(hinzvc) A=00 B=00 X=0000 SP=0000 PC=0100 LDX  #$1000
>RS 1                    (現在のレジスタ値を保存)
>R A=1 B=2 X=3 SP=BFFF PC=5000 (各レジスタに値をセット)
>R                        (レジスタを表示)
CC=C0(hinzvc) A=01 B=02 X=0003 SP=BFFF PC=5000 ADDA  #$1
```

「S Z 8 0」の場合

```
>R PC=100                (P Cレジスタに1 0 0をセット)
>R                        (レジスタを表示)
----- A=00 B=0000 D=0000 H=0000 S=0000 P=0100 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   IX, 0100H
>RS 2                    (現在のレジスタ値を保存)
>R A=1 B=2 X=3 SP=BFFF PC=5000 (各レジスタに値をセット)
>R                        (レジスタを表示)
----- A=01 B=0200 D=0000 H=0000 S=BFFF P=5000 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0003 Y=0000 ADD  A, 01H
```

保存したレジスタのロード

【書 式】

R L [ No. ]

【説 明】

R S コマンドで保存したレジスタ値をロードします。

「No.」には、0～8までの数値を指定します。

「No.」を省略すると0が選ばれます。

【使用例】

「S 0 3」の場合

```
>R (レジスタを表示)
CC=C0(hinzvc) A=01 B=02 X=0003 SP=BFFF PC=5000 ADDA #$1
>RL 1
>R (レジスタを表示)
CC=C0(hinzvc) A=00 B=00 X=0000 SP=0000 PC=0100 LDX #$1000
```

「S Z 8 0」の場合

```
>R (レジスタを表示)
----- A=01 B=0200 D=0000 H=0000 S=BFFF P=5000 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0003 Y=0000 ADD A, 01H
>RL 2 (保存したレジスタ値をロード)
>R (レジスタを表示)
----- A=00 B=0000 D=0000 H=0000 S=0000 P=0100 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD IX, 0100H
```

## 履歴の設定・表示

### ● ノーマルモード時

#### 【書 式】

H [ SWITCH ]  
H [ COUNT ]

#### 【説 明】

履歴のオン、オフ、クリア。履歴の表示を行います。

履歴とは、シミュレート実行した過程を過去に逆昇って表示するものです。かんたんに説明するとトレースを逆に実行したようなものです。ただし、履歴はトレースなどとは違って実際に命令をシミュレートするのではなく、「T, G, GS, JS」コマンドであらかじめ1ステップごとにレジスタ値を保存したものを表示するだけのものです。ですから、「T, G, GS, JS」を実行しなかった場合には、「H」コマンドは動作しないということ言うまでもありません。

履歴の為のレジスタ保存バッファは、1024ステップ分確保しています。それ以上のステップ数は、一番古い内容のものから捨てていきます。

「H」コマンドには2種類のパラメータがあり、1つは履歴を行うかどうかのスイッチを指定するもの、もう1つは実際に履歴表示を行うものです。

H OFF

とすると、履歴機能は停止します。つまり、G, GS, JS, Tを実行しても履歴は表示できません。

ただし、「H OFF」を行う前までの履歴は表示できます。

H ON

とすると、履歴機能を有効にします。立ち上げ時にはこのモードになっています。

H CLR

とすると、履歴バッファをすべて消去します。

「COUNT」で数値を指定すると、その分だけ履歴表示を行います。「COUNT」を省略すると16ステップ分だけ表示します。

もう1度、「H」コマンドを実行すると最後に表示した次の部分から表示を開始します。

## 【使用例】

「S 0 3」の場合

```

>H ON                                (ヒストリ動作を行う)
>T 5                                (5ステップ実行)
CC=C8(hiNzvc) A=00 B=00 X=0000 SP=BFFF PC=0303 LDAA #$1
CC=C0(hinzvc) A=01 B=00 X=0000 SP=BFFF PC=0305 LDAB #$FF
CC=C8(hiNzvc) A=01 B=FF X=0000 SP=BFFF PC=0307 LDX  $$8000
CC=C8(hiNzvc) A=01 B=FF X=8000 SP=BFFF PC=030A ADDA #$0
CC=C0(hinzvc) A=01 B=FF X=8000 SP=BFFF PC=030C LDAB $0
>H 2                                (ヒストリを2つだけ表示)
CC=C8(hiNzvc) A=01 B=FF X=8000 SP=BFFF PC=030A ADDA #$0
CC=C8(hiNzvc) A=01 B=FF X=0000 SP=BFFF PC=0307 LDX  $$8000
>H CLR                              (ヒストリバッファ消去)
>H                                  (ヒストリバッファは空)
>H OFF                              (ヒストリ動作を行わない)

```

「S Z 8 0」の場合

```

>H ON                                (ヒストリ動作を行う)
>T 4                                (4ステップ実行)
----- A=00 B=0000 D=0000 H=0000 S=BFFF P=0103 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   A, 01H
----- A=01 B=0000 D=0000 H=0000 S=BFFF P=0105 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   B, FFH
----- A=00 B=FF00 D=0000 H=0000 S=BFFF P=0107 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   IX, 8000H
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=010B I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=8000 Y=0000 ADD   A, 00H
>H 2                                (ヒストリを2つだけ表示)
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0107 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   IX, 8000H
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0105 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   B, FFH
>H CLR                              (ヒストリバッファ消去)
>H                                  (ヒストリバッファは空)
>H OFF                              (ヒストリ動作を行わない)

```

●スクリーンモード時

【書 式】

H [ SWITCH ]  
H

【説 明】

履歴のオン、オフ、クリア。履歴の表示を行います。

履歴のオン、オフ、クリアの方法は、ノーマルモード時とまったく同じです。なお、スクリーンモードでは、常に履歴のモードが確認できます。

パラメータを指定しなかった場合は、履歴表示モードに入ります。

逆アセンブルエリアに履歴の逆アセンブルリストを表示し（カレント位置をリバース表示）、レジスタ表示エリアにその時点でのレジスタ値を表示します。

以下のキーで履歴のカレント位置の移動が行えます。

「 」	1 ステップ前を表示
「 」	1 ステップ後を表示
「 」	16 ステップ前を表示
「 」	16 ステップ後を表示

「 」キーでコマンド入力モードに戻ります。

【使用例】

「S 0 3」の場合

>H ON >G ,100C >H		(履歴動作を行う) (シミュレート実行) (履歴表示モード)
100A 26 F9	BNE .LOOP	CC=C4 hinZvc
<div style="border: 1px dashed black; width: 150px; height: 80px; margin: 10px auto;"></div> でカレント位置移動 「 」で終了		A =00
		B =00
		X =0200
		SP=0000
		PC=100A *
		1 5
		2 6
		3 7
		4 8



「S Z 8 0」の場合

<div>&gt;H ON</div> <div>&gt;G ,1010</div> <div>&gt;H</div>		<div>(ヒストリ動作を行う)</div> <div>(シミュレート実行)</div> <div>(ヒストリ表示モード)</div>
<div>100E 10 F8</div> <div><div></div></div>	<div>DJNZ .LOOP</div> <div>でカレント位置移動</div> <div>「 」で終了</div>	<div>sZHpnc</div> <div>F =50 00</div> <div>A =00 00</div> <div>BC=0100 0000</div> <div>DE=0000 0000</div> <div>HL=0000 0000</div> <div>IX=0200</div> <div>IY=0000 1 5</div> <div>SP=0000 2 6</div> <div>PC=100E 3 7</div> <div>I =00 4 8</div> <div>R =00</div>

## 履歴の表示位置リセット

### 【書 式】

H C

### 【説 明】

履歴の表示位置のリセットを行います。

ノーマルモード時の「H」コマンドで見た履歴は、同じ所を二度と見ることはできませんが、もし同じ所を見たい場合は、このコマンドを実行してください。

履歴表示のポインタを元の位置に戻します。

### 【使用例】

「S 0 3」の場合

```
>H 2                                (履歴を2つだけ表示)
CC=C8(hiNzvc) A=01 B=FF X=8000 SP=BFFF PC=030A ADDA #$0
CC=C8(hiNzvc) A=01 B=FF X=0000 SP=BFFF PC=0307 LDX  #$8000
>HC                                (履歴ポインタを元に戻す)
>H 3                                (履歴を3つ表示)
CC=C8(hiNzvc) A=01 B=FF X=8000 SP=BFFF PC=030A ADDA #$0
CC=C8(hiNzvc) A=01 B=FF X=0000 SP=BFFF PC=0307 LDX  #$8000
CC=C0(hinzvc) A=01 B=00 X=0000 SP=BFFF PC=0305 LDAB  #$FF
```

「S Z 8 0」の場合

```
>H 2                                (履歴を2つだけ表示)
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0107 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   IX, 8000H
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0105 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   B, FFH
>HC                                (履歴ポインタを元に戻す)
>H 3                                (履歴を3つ表示)
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0107 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   IX, 8000H
----- A=01 B=FF00 D=0000 H=0000 S=BFFF P=0105 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   B, FFH
----- A=00 B=0000 D=0000 H=0000 S=BFFF P=0103 I=00XX R=00
----- A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000 LD   A, 01H
```

## 数値計算

### 【書 式】

#  
C A

### 【説 明】

「#」も「C A」も同じコマンドで、計算を行います。

これらを実行すると計算モードに入り、「=」プロンプトを表示します。

その状態で、単に「 」を入力すると元のコマンド入力モードに戻ります。

計算結果は、16進、10進、2進の順序で表示します。

計算モードでは、以下の式が実行できます。また、コマンドラインでも同様な式の入力が可能です。

スクリーンモード時は、画面を一時消去します。

### ●数 値

<16進数> デフォルトで、全て16進と見なします。  
また、数値の頭に「\$」を付けても16進になります。  
使用できる文字は0～9，A～F，a～fです。

<10進数> 数値の頭に「#」を付けます。使用できる文字は0～9です。

<8進数> 数値の頭に「@」を付けます。使用できる文字は0～7です。

<2進数> 数値の頭に「%」を付けます。使用できる文字は0，1です。

<文 字> 文字の頭に「'」を付けます。  
文字は、アスキーコードに変換されます。

### ●ラベル

「.」に続いて指定します。

もし、指定したシンボルが登録されていなかった場合はエラーになります。

## ●演算子

<単項演算子> 数値の先頭に演算子を置きます。

+	正の数
－	負の数
^	NOT
!	NOT
{	上位バイト
}	下位バイト

<算術演算子> 数値と数値の間に演算子を置きます。

+	加算
－	減算
*	乗算
/	除算
¥	除算の剰余

<論理演算子> 数値と数値の間に演算子を置きます。

>>	シフトライト
<<	シフトレフト
&	論理積
	論理和
—	排他的論理和

## ●「(,)」カッコ

優先順位を付けます。

演算子の優先順位は、以下のようになっています。(数字が小さい方が高い)

- 1) カッコで囲まれた式
- 2) 単項演算子 (「+」「-」)
- 3) シフト演算子
- 4) 乗算、除算、除算の剰余
- 5) 加算、減算
- 6) 単項演算子 (「^」「!」)
- 7) 論理積、論理和、排他的論理和
- 8) 単項演算子 (「{」「}」)

## 【使用例】

>K 1234 ABCD	(シンボルの登録)
>#	(計算モードに入る)
=#1+#2	
\$0003 : #3 : %0000000000000011	
=100+100	(16進同士の加算)
\$0200 : #512 : %0000001000000000	
=%10101010+%1010101	(2進同士の加算)
\$00FF : #255 : %0000000011111111	
=#2*(2+3)	
\$000A : #10 : %0000000000001010	
=( %1010 >> 1 ) * 2	
\$000A : #10 : %0000000000001010	
=( { . ABCD }   { } . ABCD ) << 8	
\$3412 : #13330 : %0011010000010010	
=\$100+#100+@100+%100	
\$01A8 : #424 : %0000000110101000	
='a&^20	
\$0041 : #65 : %0000000001000001	
=	(計算モードから抜け出る)
>	

## バッチファイルの実行

### 【書 式】

X < F-NAME >

### 【説 明】

ファイルよりコマンドラインを読み込みます。

コマンドラインをキーボードから入力する代わりに、指定したファイルより読み込み実行します。

「X」コマンドで「X」コマンドを実行することはできません。

スクリーンモード時でも実行できますが、「M」コマンドのようにカーソルキーなどを受けるモードに入ってしまうものでは、そこで「X」コマンドの実行を中断します。

### 【使用例】

「B A T C H. X」の内容が以下の時、

```
FI 0 FFFF
M 100
. A T E S T
.
D 100 10F
C 100 103 200
MC 100 1FF 200
F 0 2FF "TEST"
#
#1+#2
100+100
%10101010+%1010101
#2*(2+3)
(%1010>>1)*2
```

Xコマンドを実行すると、

```

| >X BATCH.X
| >FI 0 FFFF
| >M 100
| Adrs   Hex Dec  Binary  Ascii Entry
| 0100 :  00   0 00000000   ^@   [H] = .A T E S T
| 0104 :  00   0 00000000   ^@   [A] = .
| >D 100 10F
|       +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
| 0100 54 45 53 54 00 00 00 00 00 00 00 00 00 00 00 TEST.....
| >C 100 103 200
| >MC 100 1FF 200
| >F 0 2FF "TEST"
| 0100
| 0200
| >#
| =#1+#2
| $0003 : #3 : %00000000000000011
| =100+100
| $0200 : #512 : %00000001000000000
| =%10101010+%1010101
| $00FF : #255 : %00000000011111111
| =#2*(2+3)
| $000A : #10 : %00000000000001010
| =( %1010 >> 1)*2
| $000A : #10 : %00000000000001010
| =
| >

```

## DOS コマンドの実行

### 【書 式】

.  
OS

### 【説 明】

OS コマンド実行モードに入ります。

MS-DOS コマンド実行モードに入ります。

このモードから抜け出るには、「.」（ピリオド）を入力します。

通常は、「TYPE」や「DIR」などのビルトインコマンドを実行するのに使用しますが、「. EXE」ファイルも実行できます。但し、大きいプログラムは実行できません。

このコマンドは、MS-DOS の「COMMAND. COM」を実行している為、必ず、このファイルを入れておく必要があります。

実行するプログラム（コマンド）によっては、暴走する可能性があります。このコマンドの実行には十分に注意してください。また、正常に戻ってきた場合でも本プログラムの一部が壊されている場合もあります。どうも動作がおかしいと思った場合は、もう一度立ち上げ直してください。

スクリーンモード時は、画面を一時消去します。

### 【使用例】

```
>.
OS>DIR

ドライブ B: のディスクのボリュームラベルは CROSS DEBUG
ディレクトリは B:¥

S03      EXE      5xxxx  90-01-01  02:10
SAMPLE   S        52    90-01-01  02:10
SAMPLE   MAP      75    90-01-01  02:10
          4 個のファイルがあります.
          xxxxxx バイトが使用可能です.

OS>.
>
```



## ヘルプ表示

### 【書 式】

? [ CMD-NAME ]

### 【説 明】

コマンドサマリを表示します。

スクリーンモード時は、画面を一時消去します。

尚、「 S 」で一時停止します。また、「 C 」で表示を中止することもできます。

ヘルプメッセージは、プログラム中に組み込んでいません。「CRSDEBUG.HLP」というファイルに入っています。

その為、自分の好きなように変更が可能です。IBM-PCなど、日本語が表示できない機種に対してはこのファイルを英字にしなければ正しく表示されません。

「CRSDEBUG.HLP」の中身

(パラメータを指定しなかった場合は、冒頭部分のみを表示する)

#<CMD-NAME>

(ここでコマンド名を指定)

(パラメータでコマンド名を指定した場合は、ここの中を表示する)

#<CMD-NAME>

## 【使用例】

>? SC (「S C」コマンドの説明を表示)

【概要】 スクリーンモードの切り替え  
([-PC98]か[-FMR]を付けて起動した場合のみ使用可能)

【書式】 S C [MODE]

【説明】 [MODE] = 0 : ノーマルモード (省略は[MODE]=0)  
[MODE] = 1 : スクリーンモード 1  
[MODE] = 2 : スクリーンモード 2

【参照】 P. 21

>

終 了

【書 式】

Q

【説 明】

デバッグを終了します（MS-DOSに戻ります）。

今までの全てのコードは失われますので、必要であれば「S」コマンドなどでセーブしてから終了してください。

【使用例】

```
>Q  
A>
```

6-1 6303編

割り込みコントロール
------------

「T, G, GS, JS」コマンドでのシミュレーションは、実行中に割り込みを受けることができます。それぞれの割り込みはキーボードに割り当てており、ただ単にそのキーを押すことによって、その割り込みベクターの示すアドレスにサブルーチンコールします。もちろん、割り込みルーチンからの復帰は「RTI」命令です。

キー	割り込みタイプ	ベクター
「9」	TRAP	FFEE, FFEF
「8」または「N」	NMI	FFFC, FF FD
「7」または「I」	IRQ 1	FFF 8, FFF 9
「6」	ICI	FFF 6, FFF 7
「5」	O C I	FFF 4, FFF 5
「4」	T O I	FFF 2, FFF 3
「3」	C M I	FFEC, FFED
「2」	IRQ 2	FFEA, FFEB
「1」	S I O	FFF 0, FFF 1

割り込みタイプについての詳しいことは、後述の資料を参考にしてください。

# シミュレータの制限

6301／3などの組み込みの周辺 I／Oはメモリとして扱います。

6800／1／2／3などの命令は全て網羅していますが、6301／3の命令をイリーガルな命令コードとは解釈しませんので注意してください。

6800/2では本来使用できない命令

コード	命 令
3A	ABX
C3 xx xx	ADDD #IMMED
D3 xx	ADDD DIRECT
E3 xx	ADDD INDEX, X
F3 xx xx	ADDD EXTND
05	ASLD
CC xx xx	LDD #IMMED
DC xx	LDD DIRECT
EC xx	LDD INDEX, X
FC xx xx	LDD EXTND
04	LSRD
3D	MUL
3C	PSHX
38	PULX
DD xx	STD DIRECT
ED xx	STD INDEX, X
FD xx xx	STD EXTND
83 xx xx	SUBD #IMMED
93 xx xx	SUBD DIRECT
A3 xx	SUBD INDEX, X
B3 xx xx	SUBD EXTND
21	BRN LOOP

6800/1/2/3では本来使用できない命令

コード	命 令
71 xx xx	AIM #IMM, DIRECT
61 xx xx	AIM #IMM, INDEX, X
72 xx xx	OIM #IMM, DIRECT
62 xx xx	OIM #IMM, INDEX, X
75 xx xx	EIM #IMM, DIRECT
65 xx xx	EIM #IMM, INDEX, X
7B xx xx	TIM #IMM, DIRECT
6B xx xx	TIM #IMM, INDEX, X
18	XGDX
1A	SLP

レジスタの指定法

6 3 0 1 / 3 には、「A」「B」「X」「S P」「P C」「C C」と 6 つのレジスタがあります。また、A レジスタと B レジスタを合わせて「D」レジスタとしても使用できます。

レジスタの設定（R コマンド）では、上記の 7 種類のレジスタ名を指定します。

ただし、6 3 0 1 / 3 では頭の 1 文字でレジスタ名を認識できる為、1 文字しか見ていません。ですから、S P レジスタは、「S P」でも「S」でも構いません。ただし、アセンブルコマンドでは省略できませんので注意してください。

例)    > R    S = B F F F  
         > R    S P = B F F F

また、表示するレジスタを指定する「B Q」コマンドでも同じです。

例)    > B Q   1 0 0 0   S  
         > B Q   1 0 0 0   S P

レジスタの設定ではレジスタ名に続き数値を指定しますが、1 つだけ例外のレジスタがあります。それは、「C C」レジスタです。このレジスタは、フラグレジスタですので、数値を入れるよりもどのフラグをセットするかリセットするかを指定できた方が便利だと思い、セットまたはリセットするフラグを指定するようにしました。

フラグ名は、「H」「I」「N」「Z」「V」「C」の 6 つで、セットの場合は大文字で指定し、リセットの場合は小文字で指定します。また、表示（「R, T」コマンドなど）の方でもこれらの表記方法を使っています。ただし、「B Q」コマンドでは、C C レジスタも数値で表示します。

## アセンブルコマンドの書式

### ニーモニック一覧

以下のニーモニックが使用できます。

NOP	LSRD	ASLD	TAP	TPA	INX	DEX	CLV	SEV	CLC	SEC	CLI	SEI	SBA
CBA	TAB	TBA	XGDX	DAA	SLP	ABA	BRA	BRN	BHI	BLS	BCC	BCS	BNE
BEQ	BVC	BVS	BPL	BMI	BGE	BLT	BGT	BLE	TSX	INS	PULA	PULB	DES
TXS	PSHA	PSHB	PULX	RTS	ABX	RTI	PSHX	MUL	WAI	SWI	NEG	NEGA	NEGB
AIM	OIM	COM	COMA	COMB	LSR	LSRA	LSRB	EIM	ROR	RORA	RORB	ASR	ASRA
ASRB	ASL	ASLA	ASLB	ROL	ROLA	ROLB	DEC	DECA	DECB	TIM	INC	INCA	INCB
TST	TSTA	TSTB	JMP	CLR	CLRA	CLRB	SUBA	SUBB	CMPA	CMPB	SBCA	SBCB	SUBD
ADDD	ANDA	ANDB	BITA	BITB	LDAA	LDAB	STAA	STAB	EORA	EORB	ADCA	ADCB	ORAA
ORAB	ADDA	ADDB	CPX	LDD	BSR	JSR	STD	LDS	LDX	STS	STX		
BLO	BRANCH LOWER			BCSと同様のコードを発生									
BHS	BRANCH HIGHER SAME			BCCと同様のコードを発生									

【注意】 「LDAA」などは「LDA A」と第二オペランドを分離しての入力是不可能的。

### アドレッシングによるオペランドの記述

＜アドレス＞や＜数値＞には、10進数値、16進数値、シンボルが指定できます。

- ①「100」と数値のみを指定すると10進数として解釈されます。
- ②「\$1000」と数値の前に「\$」を付加すると16進数として解釈されます。
- ③「.LOOP」と文字列の前に「.」を付加するとシンボル名として解釈されます。

#### インヘレント・アドレッシング

#### アキュムレータ・アドレッシング

オペランド部の記述はありません。

#### リラティブ・アドレッシング

- オペランド部に飛び先アドレスを指定します。

＜アドレス＞

【注意】 リラティブ・アドレッシングのときは、アドレスの変位が-128から+127までの値でないといけません。アドレスがこれ以上離れているとエラーになります。

## エックステンド・アドレッシング

- オペランド部にアドレス（数値）を指定します。

<アドレス>

## イミーディエイト・アドレッシング

- 数値の前に「#」を付けます。

#<数値>

【注意】 8ビット系のイミーディエイト命令（例えば「LDAA #\$80」）では値が8ビット以上のときはエラーになります。

16ビット系のイミーディエイト命令（例えば「LDX #\$1000」）ではエラーになりません。

## インデックス・アドレッシング

- オフセット（数値）の後に「, X」を付けます。

<数値>, X

【注意】 オフセット値は0から255（10進）の間の数値でないといけません。256以上の値ではエラーになります。

また、オフセット値が0の場合でも、0の省略はできませんので注意してください。

## ダイレクト・アドレッシング

- オペランド部にアドレス（数値）を指定します。

<アドレス>

「LDAA \$80」などと100番地（16進数）より小さいアドレスの指定はダイレクト・アドレッシングでアセンブルします。

強制的にエックステンド・アドレッシングにしたい場合は「LDAA \$080」のように、数値の頭に0を付加してください。



## A I M, O I M, E I M, T I M命令

- 「#」の後に数値、オフセット（, 数値）を付けます。  
また、その後ろに「, X」を付けることもできます。

# <数値>, <数値>

# <数値>, <数値>, X

「AIM #30,\$10,X」または「AIM #30,\$80」のように、イミーディエイト・インデックスと、イミーディエイト・ダイレクトの合成的な記述をします。

★ニーモニックやその動作などについての詳細は以下の書籍・資料を参考にしてください。

「日立マイクロコンピュータ」

「マイクロコンピュータソフトウェア基礎技術」

「6800／6809による図解マイコンアセンブラ入門」

日立製作所

ラジオ技術社

オーム社

## C 言語を使つてのデバッグ例

簡単な使用例として、C 言語（当社の『Introl Cross C「C03」』）で作った関数の動作チェックを行つて見ます。

### ● 10 進表示を行う「putdec」関数のテスト

#### 【PUTDEC. C】

```
main()
{
    putdec(12345);
    putdec(-12345);
}

putdec(n)
int n;
{
    int i;

    if (n < 0){
        putchar('-');
        n = -n;
    }
    if ((i = n/10)) putdec(i);

    putchar(n % 10 + '0');
}

/* 1 文字出力を行うダミーの関数 */

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

```
A>ICC -g03 PUTDEC -r
```

(コンパイル)

```
A>IMERGE -o PUTDEC.CLS PUTDEC.C PUTDEC.S03
```

(C＋アセンブル混合リストの生成)

```
A>TYPE PUTDEC.CLS
```

(混合リストの表示)

```
*      main()
main:   fbegin
*      {
*          putdec(12345);
*          ldd     #12345
*          bsr     putdec
*          putdec(-12345);
*          ldd     #-12345
*          bsr     putdec
*          rts
*          fend
*          sttl     Function: putdec
*      }
*
*      putdec(n)
*      int n;
putdec: fbegin
*      pshb
*      psha
*      pshx
*      {
*          int i;
*
*          if (n < 0) {
*              tsx
*              ldd     2, x
*              subd    #0
*              bge     ?1.6
*              putchar(' -');
*              ldd     #45
*              bsr     putchar
*              n = -n;
*              tsx
*              ldd     2, x
*              nega
*              negb
*              sbca    #0
*              std     2, x
```

```

?1.6
*      }
*      if ((i = n/10)) putdec(i);
      tsx
      ldd      2, x
      pshb
      psha
      ldd      #10
      jsr      idiv$
      tsx
      std      0, x
      subd     #0
      beq      ?1.8
      tsx
      ldd      0, x
      bsr      putdec
?1.8
*
*      putchar(n % 10 + '0');
      tsx
      ldd      2, x
      pshb
      psha
      ldd      #10
      jsr      imod$
      addd     #48
      bsr      putchar
      pulx
      pulx
      rts
      fend
      sttl     Function: putchar
*      }
*
*      /* 1文字出力を行うダミーの関数 */
*
*      putchar(c)
*      char c;
*      {
putchar:      fbegin
      pshb
*      ;

```

```

*      }
*
      ins
      rts
      fend
      import idiv$
      import imod$

A>ILD -g03 PUTDEC.0 -gc -o PUTDEC.OUT          (リンク)
A>IHEX PUTDEC.OUT                             (Sフォーマットの生成)
A>ISYM -o PUTDEC.MAP PUTDEC.OUT               (シンボルファイルの生成)
A>TYPE B:PUTDEC.MAP                           (シンボルファイルの表示)

      idiv$      006C
      imod$      0052
      main       0000
      putchar    004F
      putdec     000B
      udiv$      007F
      umod$      0062

A>S03 PUTDEC.0                                (デバッガー起動)
+-----+
|      Cross Debugger Version 2.10      |
|                                       |
|      Copyright(c) Arcpit Co.,LTD.1990  |
+-----+

0000-00D2
Found End record.

Top Address = 0000
End Address = 00D2
Execute Address = 0000
>KA                                           (ロードしたシンボルを確認)
main      0000  putdec    000B  putchar    004F  imod$      0052
umod$     0062  idiv$     006C  udiv$      007F

```

>U .main	(逆アセンブル表示)
0000 CC 30 39 .main LDD #\$3039	
0003 8D 06 BSR .putdec	
0005 CC CF C7 LDD #\$CFC7	
0008 8D 01 BSR .putdec	
000A 39 RTS	
000B 37 .putdec PSHB	
000C 36 PSHA	
000D 3C PSHX	
000E 30 TSX	
000F EC 02 LDD 2,X	
0011 83 00 00 SUBD #.main	
0014 2C 0E BGE \$0024	
0016 CC 00 2D LDD #\$2D	
0019 8D 34 BSR .putchar	
001B 30 TSX	
001C EC 02 LDD 2,X	
>B A	(終了アドレスのブレークポイントをセット)
>BT 5 ¥0D¥0A	(5 番地に来たら「C R / L F」)
>BT A ¥0D¥0A	(A 番地に来たら「C R / L F」)
>BQ .putchar D	(「putchar」に来たらDレジスタを表示)
>R S=BFFF	(S PレジスタにB F F Fをセット)
>R	(現在のレジスタを確認)
CC=C0(hinzvc) A=00 B=00 X=0000 SP=BFFF PC=0000 .main LDD #\$3039	
>G	(シミュレーション実行開始)
0031 0032 0033 0034 0035	(「1 2 3 4 5」と出力している)
002D 0031 0032 0033 0034 0035	(「- 1 2 3 4 5」と出力している)
CC=C0(hinzvc) A=00 B=35 X=3039 SP=BFFF PC=000A RTS	
>Q	(正常に動作したので終了する)
A>	

## アセンブラを使つてのデバッグ例

次に、アセンブラ（当社の『CROSS ASSEMBLER「X6801」』）で作ったサンプルプログラムの動作チェックを行つて見ます。

●データ群の中から最小値、最大値のデータを見つけ出すプログラムのテスト

### 【MINMAX. TXT】

	ORG	\$100
LARGE	RMB	1
SMALL	RMB	1
	ORG	\$200
MINMAX	LDX	#DATA
	CLR	LARGE
	LDAA	#\$FF
	STAA	SMALL
LOOP	LDAA	0, X
	CMPA	LARGE
	BHS	CONT2
	STAA	LARGE
CONT2	CMPA	SMALL
	BHS	CONT3
	STAA	LARGE
CONT3	INX	
	CPX	#DATEND
	BNE	LOOP
	RTS	
DATA	FCB	2, 54, 76, 32, 12, 87, 55, 6
DATEND	EQU	*
	END	MINMAX

A>X6801 MINMAX,,,MINMAX /SL

(アセンブル)

6801 Cross Assembler Version 3.10

Copyright(C) Arcpit Co.,LTD.1990. All rights reserved.

A>S03

(デバッガー起動)

```
+-----+
|               |
| Cross Debugger Version 2.10 |
|               |
| Copyright(c) Arcpit Co.,LTD.1990 |
|               |
+-----+
```

>L MINMAX

(オブジェクト&シンボルのロード)

0200-022B

Found End record.

Top Address = 0200

End Address = 022B

Execute Address = 0200

>KA

(シンボルの確認)

LARGE	0100	SMALL	0101	MINMAX	0200	LOOP	020B
CONT2	0215	CONT3	021D	DATA	0224	DATEND	022C

(逆アセンブル表示)

>U .MINMAX

```
0200 CE 02 24 .MINMAX      LDX  #.DATA
0203 7F 01 00              CLR  .LARGE
0206 86 FF                LDAA  #$FF
0208 B7 01 01              STAA  .SMALL
020B A6 00      .LOOP      LDAA  0,X
020D B1 01 00              CMPA  .LARGE
0210 24 03                BCC   .CONT2
0212 B7 01 00              STAA  .LARGE
0215 B1 01 01 .CONT2      CMPA  .SMALL
0218 24 03                BCC   .CONT3
021A B7 01 00              STAA  .LARGE
021D 08      .CONT3      INX
021E 8C 02 2C              CPX   #.DATEND
0221 26 E8                BNE   .LOOP
0223 39                  RTS
0224 02      .DATA      Illegal Opcode
```

>B 223

(223番地にブレークポイントをセット)

>R S=BFFF

(SPレジスタにBFFFをセット)

>R

(レジスタの確認)

CC=C0(hinzvc) A=00 B=00 X=0000 SP=BFFF PC=0200 .MINMAX LDX #.DATA

>RS

(レジスタ内容の待避①)

>G

(シミュレーション実行)

CC=C4(hinZvc) A=06 B=00 X=022C SP=BFFF PC=0223

RTS



>M .LARGE	(「LARGE」 と 「SMALL」 の確認)
Adrs Hex Dec Binary Ascii Entry	
0100 : 06 6 00000110 ^F [H] =	
0101 : FF 255 11111111 _ [H] = .	(「SMALL」 にデータ設定なし)
>A 21A	(2 1 A 番地からアセンブル入力)
021A STAA .SMALL	(「STAA .LARGE」 を 「STAA .SMALL」 に訂正)
021D	
>RL	(①で待避したレジスタを復帰)
>G	(再度シミュレーション実行)
CC=C4(hinZvc) A=06 B=00 X=022C SP=BFFF PC=0223	RTS
>M .LARGE	(「LARGE」 と 「SMALL」 の確認)
Adrs Hex Dec Binary Ascii Entry	
0100 : 00 0 00000000 ^@ [H] =	(「LARGE」 にデータ設定なし)
0101 : 02 2 00000010 ^B [H] = .	
>BQ 212 A	(2 1 2 番地でAレジスタを表示)
>BT 212 LARGE=	(2 1 2 番地で「LARGE=」を表示)
>BQ 21A A	(2 1 A 番地でAレジスタを表示)
>BT 21A SMALL=	(2 1 A 番地で「SMALL=」を表示)
>BT 223 ¥0D¥0A	(2 2 3 番地で「C R / L F」を表示)
>RL	(①で待避したレジスタを復帰)
>J	(再度シミュレーション実行)
SMALL=02	(2 1 2 番地の通過がない)
CC=C4(hinZvc) A=06 B=00 X=022C SP=BFFF PC=0223	RTS
>RL	(①で待避したレジスタを復帰)
>T 10	(ステップ実行)
CC=C0(hinzvc) A=00 B=00 X=0224 SP=BFFF PC=0203	CLR .LARGE
CC=C4(hinZvc) A=00 B=00 X=0224 SP=BFFF PC=0206	LDAA #\$FF
CC=C8(hiNzvc) A=FF B=00 X=0224 SP=BFFF PC=0208	STAA .SMALL
CC=C8(hiNzvc) A=FF B=00 X=0224 SP=BFFF PC=020B .LOOP	LDAA 0, X
CC=C0(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=020D	CMPA .LARGE
CC=C0(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=0210	BCC .CONT2
CC=C0(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=0215 .CONT2	CMPA .SMALL
CC=C1(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=0218	BCC .CONT3
CC=C1(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=021A	STAA .SMALL
CC=C1(hinzvc) A=02 B=00 X=0224 SP=BFFF PC=021D .CONT3	INX
CC=C1(hinzvc) A=02 B=00 X=0225 SP=BFFF PC=021E	CPX #.DATEND
CC=C9(hiNzvc) A=02 B=00 X=0225 SP=BFFF PC=0221	BNE .LOOP
CC=C9(hiNzvc) A=02 B=00 X=0225 SP=BFFF PC=020B .LOOP	LDAA 0, X
CC=C1(hinzvc) A=36 B=00 X=0225 SP=BFFF PC=020D	CMPA .LARGE
CC=C0(hinzvc) A=36 B=00 X=0225 SP=BFFF PC=0210	BCC .CONT2
CC=C0(hinzvc) A=36 B=00 X=0225 SP=BFFF PC=0215 .CONT2	CMPA .SMALL

```

>A 210 (210番地が条件に合わない為、
0210 BLS .CONT2 「BCC .CONT2」を「BLS .CONT2」に訂正)
0212
>RL
>J (再度シミュレーション)
LARGE=02 SMALL=02 LARGE=36 LARGE=4C LARGE=57
CC=C4(hinZvc) A=06 B=00 X=022C SP=BFFF PC=0223 RTS
>M .LARGE (「LARGE」と「SMALL」の確認)
Adrs Hex Dec Binary Ascii Entry
0100 : 57 87 01010111 W [H] = (「LARGE」OK)
0101 : 02 2 00000010 ^B [H] = . (「SMALL」OK)
>M .DATA (他のデータで再度テスト)
Adrs Hex Dec Binary Ascii Entry
0224 : 02 2 00000010 ^B [H] = 3 2 1 0 FF FE FD .
>RL
>J
LARGE=03 SMALL=03 SMALL=02 SMALL=01 SMALL=00 LARGE=FF
CC=C4(hinZvc) A=06 B=00 X=022C SP=BFFF PC=0223 RTS
>M .LARGE (「LARGE」と「SMALL」の確認)
Adrs Hex Dec Binary Ascii Entry
0100 : FF 255 11111111 _ [H] = (「LARGE」OK)
0101 : 00 0 00000000 ^@ [H] = . (「SMALL」OK)
>Q (終了)
A>

```

エディターを立ち上げてソースプログラムを訂正する。

割り込みコントロール

「T, G, GS, JS」コマンドでのシミュレーションは、実行中に割り込みを受けることができます。それぞれの割り込みはキーボードに割り当てており、ただ単にそのキーを押すことによって、その割り込みベクターの示すアドレスにサブルーチンコールします。もちろん、割り込みルーチンからの復帰は「RETI」や「RETN」命令です。

キー	割り込みタイプ	実行アドレス
「I」	INT	モード0 :    x x x x モード1 :    3 8 H モード2 :    I x x
「N」	NMI	6 6 H

マスクابل割り込み（INT）は、「EI」命令で「IFF」を1にしなくては実行できません。割り込みが発生すると（「I」をキーイン）、現在の割り込みモード（「IM」命令で設定）により、以下の動作を行います。

●モード0（「IM 0」実行）の場合

指定したアドレスへ制御を移します。

ノーマルモード時は、「Mode0 interrupt = vector 16bit?」と聞いてきます。

スクリーンモード時は、「INT Mode0 Lower 16」と聞いてきます。

16ビットのベクターアドレスを指定してください。

●モード1（「IM 1」実行）の場合

アドレス38Hに制御を移します。

●モード2（「IM 2」実行）の場合

Iレジスタの値を上位アドレスとし、指定したアドレスを下位アドレスとして、そのアドレスにセットされているアドレスに制御を移します（間接コール）。

ノーマルモード時は、「Mode2 interrupt = vector 8bit?」と聞いてきます。

スクリーンモード時は、「INT Mode2 Lower 8」と聞いてきます。

8ビットのベクターアドレスを指定してください。

「RETI」命令をフェッチすると、元のアドレスに制御が戻ります。

ノンマスカブル割り込み（NMI）は、割り込みを禁止することができず、「N」をキーインすると割り込みが発生します。

アドレス 6 6 H に制御を移します。

「RET N」命令をフェッチすると、元のアドレスに制御が戻ります。

割り込みについての詳しいことは、後述の資料を参考にしてください。

シミュレータの制限
-----------

Z 8 0 の命令は全て網羅していますが、I / O 命令は特別に割り当てられた 2 5 6 バイトの空間に対して行なわれます。

I / O 空間に対して出力した値は、「I」コマンドで出力した値を確認することができます。また、「O」コマンドで、直接 I / O 空間に書き込みができます。

## I / Oの書き込み

### ●ノーマルモード時

#### 【書 式】

O [ IO-ADRS ]

#### 【説 明】

I / Oメモリ内容を1バイト単位で表示し、変更もできます。

パラメータを指定しなかった場合はカレントI / Oアドレスが対象になり、指定した場合は、そのI / Oアドレスが対象になります。

I / Oアドレスは、0から255までの範囲で指定します。

Oコマンドを実行すると、

```
00 : 00 =
```

という表示になります。

左の「00」はI / Oアドレスであり、右の「00」はそのI / Oの値です。

```
00 : 00 = FF
```

と出力したい値を入力すれば、I / Oアドレスに書き込まれます。

もちろん、本来のI / O（8086の持っているもの）へ出力する訳ではありませんから、どんな値でも書き込めます。

```
00 : 01 = 1 2 3 4 5
```

と「M」コマンドのように、連続して値を書き込むこともできます。最高20バイトまでの値を指定することができます。

単に「 」のみを入力すると、「O」コマンドは終了します。

#### 【使用例】

```
>O 10
10 : 00 = FF                (10にFFを書き込む)
11 : 00 = 1 2 3 4 5        (11から連続して書き込む)
16 : 00 =                  (Oコマンド終了)
>
```

## ●スクリーンモード時

### 【書 式】

O [ IO-ADRS ]

### 【説 明】

スクリーンモード1の場合は、ダンプエリアがI/Oダンプ表示に切り替わります。  
そして、そのエリアに移行して、I/Oの変更が行えます。  
操作方法は、「M」コマンドとまったく同じです。

スクリーンモード2の場合は、ダンプエリア2がI/Oダンプ表示に切り替わります。  
そして、そのエリアに移行して、I/Oの変更が行えます。  
操作方法は、「M2」コマンドとまったく同じです。

### 【使用例】

>SC 1	(スクリーンモード1にする)
>O 10	(10番地から表示)
<pre> +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F --00 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ..... --10 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F ..... --20 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F  !"#\$%&amp;'()*+,-./ --30 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 0123456789:;&lt;=&gt;? --40 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F @ABCDEFGHIJKLMNO --50 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F PQRSTUVWXYZ[¥]^_ --60 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F ~abcdefghijklmno --70 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F pqrstuvwxyz{ } . --80 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F _____ --90 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F _____ </pre>	

>SC 2	(スクリーンモード2にする)
>O 10	(10番地から表示)
<pre> 0100 00 01 02 03   --00 CE 01 00 4F   0104 04 05 06 07   --04 5F A7 00 4C   0108 08 09 0A 0B   --08 08 5A 26 F9   010C 0C 0D 0E 0F   --0C 20 FE 00 00   0110 10 11 12 13   --10 00 00 00 00   0114 14 15 16 17   --14 00 00 00 00   0118 18 19 1A 1B   --18 00 00 00 00   011C 1C 1D 1E 1F   --1C 00 00 00 00   0120 20 21 22 23   --20 00 00 00 00   0124 24 25 26 27   --24 00 00 00 00   </pre>	

## I / O の読み込み

### ● ノーマルモード時

#### 【書 式】

I

#### 【説 明】

I / O メモリ内容を表示するコマンドです。

0 から 2 5 5 までの I / O エリアを一括してダンプ表示します。

#### 【使用例】

```
>I
+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

## ●スクリーンモード時

### 【書 式】

I        [ IO-ADRS ]

### 【説 明】

スクリーンモード 1 の場合は、ダンプエリアが I / O ダンプ表示に切り替わります。  
スクリーンモード 2 の場合は、ダンプエリア 2 が I / O ダンプ表示に切り替わります。

それぞれ、「IO-ADRS」を指定するとそのアドレスを先頭に表示します。

「D」や「M」コマンドでメモリ表示に変更するまで I / O 表示を行っています。

### 【使用例】

```
>SC 1                                (スクリーンモード 1 にする)
>I 10                                (10 番地から変更)
```

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
--10	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
--20	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
--30	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
--40	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
--50	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
--60	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
--70	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
--80	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
--90	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
--A0	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F

```
>SC 2                                (スクリーンモード 2 にする)
>I 10                                (10 番地から変更)
```

0100	00	01	02	03	--10	CE	01	00	4F
0104	04	05	06	07	--14	5F	A7	00	4C
0108	08	09	0A	0B	--18	08	5A	26	F9
010C	0C	0D	0E	0F	--1C	20	FE	00	00
0110	10	11	12	13	--20	00	00	00	00
0114	14	15	16	17	--24	00	00	00	00
0118	18	19	1A	1B	--28	00	00	00	00
011C	1C	1D	1E	1F	--2C	00	00	00	00
0120	20	21	22	23	--30	00	00	00	00
0124	24	25	26	27	--34	00	00	00	00



## レジスタの指定法

Z80には、「A」「B」「C」「D」「E」「H」「L」「IX」「IY」「SP」「PC」「F」「I」「R」、そして裏レジスタ「A'」「B'」「C'」「D'」「E'」「H'」「L'」「F'」と22個のレジスタがあります。また、ペアレジスタとして、「BC」「DE」「HL」と、「BC'」「DE'」「HL'」としても使用できます。

レジスタの設定（Rコマンド）では、上記のレジスタ名で指定します。ただし、省略できるものは省略して、「IX」は「X」、「IY」は「Y」、「SP」は「S」、「PC」は「P」と指定できます。ただし、アセンブルコマンドでは省略できませんので注意してください。

例) >R IX=1000  
>R X=1000

また、表示するレジスタを指定する「BQ」コマンドでも同じです。

例) >BQ 1000 IX  
>BQ 1000 X

レジスタの設定では、レジスタ名に続き数値を指定しますが、1つだけ例外のレジスタがあります。それは、「F」レジスタです。このレジスタはフラグレジスタですので、数値を入れるよりもどのフラグをセットするかにリセットするかを指定できた方が便利だと思い、セットまたはリセットするフラグを指定するようにしました。

フラグ名は、「S」「Z」「H」「P」「N」「C」の6つで、セットの場合は大文字で指定し、リセットの場合は小文字で指定します。また、表示（「R, T」コマンドなど）の方でもこれらの表記方法を使っています。ただし、「BQ」コマンドでは、Fレジスタも数値で表示します。

## アセンブルコマンドの書式

### ニーモニック一覧

以下のニーモニックが使用できます。

ADC	ADD	AND	BIT	CALL	CCF	CP	CPD	CPI	CPIR	CPL	DAA	DEC	DI
DJNZ	EI	EX	EXX	HALT	IM	IN	INC	IND	INDR	INI	INIR	JP	JR
LD	LDD	LDDR	LDI	LDIR	NEG	NOP	OR	OTDR	OTIR	OUT	OUTD	OUTI	POP
PUSH	RES	RET	RETN	RL	RLA	RLC	RLCA	RLD	RR	RRA	RRC	RRCA	RRD
RST	SBC	SCF	SET	SLA	SRA	SRL	SUB	XOR					

### アドレッシングによるオペランドの記述

＜アドレス＞や＜数値＞が指定できる所は、10進数値、16進数値、シンボルが指定できます。

- ①「100」と数値のみを指定すると10進数として解釈されます。
- ②「1000H」と数値の後に「H」を付加すると16進数として解釈されます。  
数値の最初が「A」～「F」で始まるものは、必ずその前に「0」を指定しなくてはなりません（通常のアセンブラと同様）。
- ③「. LOOP」と文字列の前に「.」を付加するとシンボル名として解釈されます。

基本的には、通常のZ80のアセンブラと記述方法は同じです。

特に注意することのみをここでは記載します。

### 相対アドレッシング

- オペランド部に飛び先アドレスを指定します。

また、オフセット数値も指定可能です。

＜アドレス＞

\$ +＜数値＞

\$ -＜数値＞

「\$ + 0」のように、自分からのオフセット値は直接指定することもできます。

「\$」に続いて、「+」または「-」、そしてオフセット値を指定します。

**【注意】** アドレスの変位がオペコードのアドレスから計算すると、-126から+129までの範囲でないといけません。アドレスがこれ以上離れているとエラーになります。

## インデックス・アドレッシング

- インデックスレジスタの後に「+＜数値＞」を付けます。

I X +＜数値＞

I Y +＜数値＞

【注意】オフセット値は－１２８から＋１２７（１０進）の間の数値でないといけません。  
この範囲を越えるとエラーになります。

またオフセット値が０の場合でも、０の省略はできませんので注意してください。

- ★ニーモニックやその動作などについての詳細は以下の書籍・資料を参考にしてください。

「Ｚ８０プログラミングマニュアル」

SHARP

「Ｚ８０テクニカルマニュアル」

SHARP

「マイコンピュータ⑦ Ｚ８０アセンブラ言語入門」

CQ出版社

## アセンブラを使つてのデバッグ例

アセンブラ（当社の『CROSS ASSEMBLER「XZ80」』）で作つたサンプルプログラムの動作チェックを行つて見ます。

●データ群の中から最小値、最大値のデータを見つけ出すプログラムのテスト

### 【MINMAX. TXT】

	ORG	100H
LARGE	DS	1
SMALL	DS	1
	ORG	200H
MINMAX	LD	HL, DATA
	LD	IX, SMALL
	LD	IY, LARGE
	LD	B, DATEND-DATA
	LD	(IY+0), 0
	LD	(IX+0), 0FFH
LOOP	LD	A, (HL)
	CP	(IY+0)
	JR	NC, CONT2
	LD	(IY+0), A
CONT2	CP	(IX+0)
	JR	NC, CONT3
	LD	(IY+0), A
CONT3	INC	HL
	DJNZ	LOOP
	RET	
DATA	DB	2, 54, 76, 32, 12, 87, 55, 6
DATEND	EQU	\$
	END	MINMAX

A>XZ80 MINMAX,, ,MINMAX /SL	(アセンブル)
Z80 Cross Assembler Version 3.10 Copyright(C) Arcpit Co.,LTD. All rights reserved.	
A>SZ80	(デバッガー起動)
<div> <div> +-----+   Cross Debugger Version 2.10   Copyright(c) Arcpit Co.,LTD.1990   +-----+ </div> </div>	
>L MINMAX 0200-0231 Found End record.	(オブジェクト&シンボルのロード)
Top Address = 0200 End Address = 0231 Execute Address = 0200	
>KA	(シンボルの確認)
LARGE        0100    SMALL            0101    MINMAX        0200    LOOP            0215 CONT2        021E    CONT3            0226    DATA           022A    DATEND           0232	
>U .MINMAX	(逆アセンブル表示)
0200 21 2A 02        .MINMAX        LD    HL,. DATA	
0203 DD 21 01 01        LD    IX,. SMALL	
0207 FD 21 00 01        LD    IY,. LARGE	
020B 06 08            LD    B, 08H	
020D FD 36 00 00        LD    (IY+00H), 00H	
0211 DD 36 00 FF        LD    (IX+00H), FFH	
0215 7E                . LOOP        LD    A, (HL)	
0216 FD BE 00           CP    (IY+00H)	
0219 30 03            JR    NC,. CONT2	
021B FD 77 00           LD    (IY+00H), A	
021E DD BE 00        . CONT2        CP    (IX+00H)	
0221 30 03            JR    NC,. CONT3	
0223 FD 77 00           LD    (IY+00H), A	
0226 23                . CONT3        INC   HL	
0227 10 EC            DJNZ . LOOP	
0229 C9                RET	

```

>B 229                                ( 2 2 9 番地にブレークポイントをセット)
>R S=BFFF                             ( S レジスタに B F F F をセット)
>R                                    ( レジスタの確認)
szhpnc A=00 B=0000 D=0000 H=0000 S=BFFF P=0200 I=00XX R=00
szhpnc A'00 B'0000 D'0000 H'0000 X=0000 Y=0000 .MINMAX      LD   HL,.DATA
>RS                                    ( レジスタ内容の待避①)
>G                                    ( シミュレーション実行)
szHpNC A=06 B=0000 D=0000 H=0232 S=BFFF P=0229 I=00XX R=00
szhpnc A'00 B'0000 D'0000 H'0000 X=0101 Y=0100              RET
>M .LARGE                             ( 「LARGE」 と 「SMALL」 の確認)
Adrs   Hex Dec  Binary  Ascii Entry
0100 :  06   6 00000110  ^F   [H] =
0101 :  FF 255 11111111  _    [H] = .      ( 「SMALL」 にデータ設定なし)
>A 223                                ( 2 2 3 番地からアセンブリ入力)
0223 LD (IX+0),A                    ( 「LD (IY+0),A」 を 「LD (IX+0),A」 に訂正)
0226
>RL                                    ( ①で待避したレジスタを復帰)
>G                                    ( 再度シミュレーション実行)
szhpNc A=06 B=0000 D=0000 H=0232 S=BFFF P=0229 I=00XX R=00
szhpnc A'00 B'0000 D'0000 H'0000 X=0101 Y=0100              RET
>M .LARGE                             ( 「LARGE」 と 「SMALL」 の確認)
Adrs   Hex Dec  Binary  Ascii Entry
0100 :  00   0 00000000  ^@   [H] =      ( 「LARGE」 にデータ設定なし)
0101 :  02   2 00000010  ^B   [H] = .
>BQ 21B A                            ( 2 1 B 番地で A レジスタを表示)
>BT 21B LARGE=                       ( 2 1 B 番地で 「LARGE=」 を表示)
>BQ 223 A                            ( 2 2 3 番地で A レジスタを表示)
>BQ 223 SMALL=                      ( 2 2 3 番地で 「SMALL=」 を表示)
>BT 229 ¥0D¥0A                      ( 2 2 9 番地で 「C R / L F」 を表示)
>RL                                    ( ①で待避したレジスタを復帰)
>J                                    ( 再度シミュレーション実行)
SMALL=02                            ( 2 1 B 番地の通過がない)
szhpNc A=06 B=0000 D=0000 H=0232 S=BFFF P=0229 I=00XX R=00
szhpnc A'00 B'0000 D'0000 H'0000 X=0101 Y=0100              RET
>RL                                    ( ①で待避したレジスタを復帰)

```

>T 10	(ステップ実行)	
szhpnc A=00 B=0000 D=0000 H=022A S=BFFF P=0203 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0000 Y=0000	LD	IX, . SMALL
szhpnc A=00 B=0000 D=0000 H=022A S=BFFF P=0207 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0000	LD	IY, . LARGE
szhpnc A=00 B=0000 D=0000 H=022A S=BFFF P=020B I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	LD	B, 08H
szhpnc A=00 B=0800 D=0000 H=022A S=BFFF P=020D I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	LD	(IY+00H), 00H
szhpnc A=00 B=0800 D=0000 H=022A S=BFFF P=0211 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	LD	(IX+00H), FFH
szhpnc A=00 B=0800 D=0000 H=022A S=BFFF P=0215 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100 . LOOP	LD	A, (HL)
szhpnc A=02 B=0800 D=0000 H=022A S=BFFF P=0216 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	CP	(IY+00H)
szhpnc A=02 B=0800 D=0000 H=022A S=BFFF P=0219 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	JR	NC, . CONT2
szhpnc A=02 B=0800 D=0000 H=022A S=BFFF P=021E I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100 . CONT2	CP	(IX+00H)
szHpNC A=02 B=0800 D=0000 H=022A S=BFFF P=0221 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	JR	NC, . CONT3
szHpNC A=02 B=0800 D=0000 H=022A S=BFFF P=0223 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	LD	(IX+00H), A
szHpNC A=02 B=0800 D=0000 H=022A S=BFFF P=0226 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100 . CONT3	INC	HL
szHpNC A=02 B=0800 D=0000 H=022B S=BFFF P=0227 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	DJNZ	. LOOP
szHpNC A=02 B=0700 D=0000 H=022B S=BFFF P=0215 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100 . LOOP	LD	A, (HL)
szHpNC A=36 B=0700 D=0000 H=022B S=BFFF P=0216 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	CP	(IY+00H)
szhpnc A=36 B=0700 D=0000 H=022B S=BFFF P=0219 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	JR	NC, . CONT2
>A 219	( 2 1 9 番地が条件に合わない為、	
0219 JR C, . CONT2	「JR NC, . CONT2」を「JR C, . CONT2」に訂正)	
021B		
>RL		
>J	(再度シミュレーション)	
LARGE=02 BFFF LARGE=36 LARGE=4C LARGE=57		
szhpnc A=06 B=0000 D=0000 H=0232 S=BFFF P=0229 I=00XX R=00		
szhpnc A' 00 B' 0000 D' 0000 H' 0000 X=0101 Y=0100	RET	

```

>M .LARGE                                     (「LARGE」 と 「SMALL」 の確認)
Adrs  Hex Dec  Binary  Ascii Entry
0100 :  57  87 01010111    W   [H] =      (「LARGE」 OK)
0101 :  02   2 00000010    ^B   [H] = .    (「SMALL」 OK)
>M .DATA                                     (他のデータで再度テスト)
Adrs  Hex Dec  Binary  Ascii Entry
022A :  02   2 00000010    ^B   [H] = 3 2 1 0 FF FE FD .
>RL
>J
LARGE=03 BFFF BFFF BFFF BFFF LARGE=FF
szhpNc A=06 B=0000 D=0000 H=0232 S=BFFF P=0229 I=00XX R=00
szhpnc A'00 B'0000 D'0000 H'0000 X=0101 Y=0100          RET
>M .LARGE                                     (「LARGE」 と 「SMALL」 の確認)
Adrs  Hex Dec  Binary  Ascii Entry
0100 :  FF 255 11111111    _   [H] =      (「LARGE」 OK)
0101 :  00   0 00000000    ^@   [H] = .    (「SMALL」 OK)
>Q                                           (終了)
A>

```

エディターを立ち上げてソースプログラムを訂正する。



付録A：コマンド一覧表

● ノーマルモード時に使用できるコマンド

名前	書式	機能	備考	頁
?	? [CMD-NAME]	ヘルプ表示		77
SC	SC [MODE]	スクリーンモードの切り替え		21
LA	LA	ロードアドレスの表示		40
D	D [ADR1 [ADR2]]	メモリの16進ダンプ		23
M	M [ADR]	メモリの表示と変更		25
FI	FI ADR1 ADR2 [DATA]	メモリのデータセット		28
C	C ADR1 ADR2 ADR3	メモリのコピー		29
MC	MC ADR1 ADR2 ADR3	メモリの比較		30
F	F ADR1 ADR2 LIST	メモリのサーチ		31
L	L F-NAME	ファイルとシンボルのロード		39
LS	LS F-NAME	シンボルのロード		50
SS	SS F-NAME ADR1 ADR2 [ADR3]	Sフォーマットでのセーブ		41
SI	SI F-NAME ADR1 ADR2 [ADR3]	インテルHEXでのセーブ		42
B	B [ADR ...]	ブレークポイントの表示と設定		48
BS	BS [ADR ...]	停止するブレークポイントの設定		44
BR	BR [ADR ...]	全レジスタを表示するブレークポイント		45
BQ	BQ [ADR REG-LIST]	指定レジスタを表示するブレークポイント		46
BT	BT [ADR STRING]	指定文字列を表示するブレークポイント		47
BC	BC [ADR1 [ADR2]]	ブレークポイントの解除		49
K	K ADR LABEL	シンボルの登録		51
KA	KA [ADR1 [ADR2]]	アドレス順シンボルの表示		52
KL	KL [LABEL1 [LABEL2]]	名前順シンボルの表示		53
KC	KC	シンボルの全解除		54
T	T [COUNT]	ステップ実行		55
G	G [ADR1 [ADR2]]	シミュレーション実行		57
J	J [ADR1 [ADR2]]	高速シミュレーション実行		58
R	R [REG-NAME=VALUE]	レジスタの表示と設定		62
RS	RS [No.]	今のレジスタ値の退避		64
RL	RL [No.]	退避したレジスタ値の復帰		65
H	H [ON OFF CLR COUNT]	履歴のオン、オフ、クリア、表示		66
HC	HC	履歴の表示位置のリセット		70
A	A [ADR]	アセンブル		32
U	U [ADR1 [ADR2]]	逆アセンブル		35
UF	UF F-NAME ADR1 ADR2	逆アセンブル（ファイルに書き出す）		35
I	I	I/Oエリアのダンプ表示	「SZ80」	99
O	O [I/O-ADR]	I/Oエリアへの書き込み	「SZ80」	97
#	#(CA)	計算		71
X	X F-NAME	ファイルからコマンド実行		74
.	.(OS)	MS-DOSコマンドの実行モード		76
Q	Q	終了		79

●スクリーンモード時に使用できるコマンド

?	? [CMD-NAME]	ヘルプ表示		77
S C	SC [MODE]	スクリーンモードの切り替え		21
D	D [ADR]	メモリの16進ダンプ		24
D 2	D2 [ADR]	メモリの16進ダンプ (エリア2)		24
M	M [ADR]	メモリの表示と変更		27
M 2	M2 [ADR]	メモリの表示と変更 (エリア2)		27
F I	FI ADR1 ADR2 [DATA]	メモリのデータセット		28
C	C ADR1 ADR2 ADR3	メモリのコピー		29
M C	MC ADR1 ADR2 ADR3	メモリの比較		30
F	F ADR1 ADR2 LIST	メモリのサーチ		31
L	L F-NAME	ファイルとシンボルのロード		39
L S	LS F-NAME	シンボルのロード		50
S S	SS F-NAME ADR1 ADR2 [ADR3]	Sフォーマットでのセーブ		41
S I	SI F-NAME ADR1 ADR2 [ADR3]	インテルHEXでのセーブ		42
B	B [ADR ...]	ブレークポイントの表示と設定		48
B S	BS [ADR ...]	停止するブレークポイントの設定		44
B R	BR [ADR ...]	全レジスタを表示するブレークポイント		45
B Q	BQ [ADR REG-LIST]	指定レジスタを表示するブレークポイント		46
B T	BT [ADR STRING]	指定文字列を表示するブレークポイント		47
B C	BC [ADR1 [ADR2]]	ブレークポイントの解除		49
K	K ADR LABEL	シンボルの登録		51
K A	KA [ADR1 [ADR2]]	アドレス順シンボルの表示		52
K L	KL [LABEL1 [LABEL2]]	名前順シンボルの表示		53
K C	KC	シンボルの全解除		54
V	V ADR MODE	メモリ監視エリアの設定		22
T	T [COUNT]	ステップ実行		55
G	G [ADR1] [ADR2]	シミュレーション実行		57
J	J [ADR1] [ADR2]	高速シミュレーション実行		58
G S	GS [ADR1] [ADR2]	全値を表示しながらのシミュレート		60
J S	JS [ADR1] [ADR2]	PC値を表示してのシミュレート		59
R	R [REG-NAME=VALUE]	レジスタの表示と設定		62
R S	RS [No.]	今のレジスタ値の退避		64
R L	RL [No.]	退避したレジスタ値の復帰		65
H	H [ON OFF CLR]	履歴のオン, オフ, クリア, 表示		68
H C	HC	履歴の表示位置のリセット		70
A	A [ADR]	アセンブル		34
U	U [ADR]	逆アセンブル		37
U F	UF F-NAME ADR1 ADR2	逆アセンブル (ファイルに書き出す)		37
I	I [I/O-ADR]	I/Oエリアのダンプ表示	「SZ80」	100
O	O [I/O-ADR]	I/Oエリアへの書き込み	「SZ80」	98
#	#(CA)	計算		71
X	X F-NAME	ファイルからコマンド実行		74
.	.(OS)	MS-DOSコマンドの実行モード		76
Q	Q	終了		79

---

## 付録B：ユーティリティープログラム

---

LOAD (メモリへのオブジェクトのロード)

このユーティリティーは、オブジェクト（モトローラSフォーマットやインテルHEXフォーマット）をメモリ上にロードするものです。

LOAD ファイル名 物理アドレス [オフセット]

- ファイル名           モトローラSフォーマットかインテルHEXファイルです。  
                      自動的にフォーマットを認識します。  
                      拡張子を省略した場合は、「.S」または「.HEX」を自動的に付けます。
- 物理アドレス        何番地からロードするかを指定します。  
                      セグメントアドレス、オフセットアドレスを合わせての物理アドレスで指定します。  
                      例えば、セグメントB000Hで、オフセット0の場合は、「B0000」と指定します。  
                      ロード範囲の対象は、「B0000」～「BFFFF」になります。
- オフセット          アセンブラでのアドレスにオフセット値を与えたい場合に指定します。  
                      マイナスのオフセットも「-100」のように指定できます。  
                      この場合は、「FF00」を指定したのと同じです。

例) A>LOAD TEST B0000 -100

「TEST」プログラムが、100番地から1FF番地のものだとする、メモリのB0000～B00FFまでのエリアにロードされます。

★このユーティリティーは、直接メモリにロードしますので、MS-DOSなどで使用しているアドレスを指定すると、システムを破壊する可能性があります。アドレスの指定には十分注意してください。

SAVE (メモリからのオブジェクトのセーブ)

このユーティリティーは、メモリ上にあるコードをモトローラSフォーマットやインテルHEXフォーマットでファイルにセーブするものです。

SAVE モード ファイル名 物理アドレス1 物理アドレス2 [アドレス]

モード 「S」 モトローラSフォーマットでセーブします。  
「I」 インテルHEXフォーマットでセーブします。

ファイル名 セーブしたいファイル名を指定します。  
拡張子を省略した場合で「モード」が「S」の場合は「.S」を付けます。  
「I」の場合は「.HEX」を付けます。

物理アドレス1 メモリ上の何番地からセーブするかを指定します。  
セグメントアドレス、オフセットアドレスを合わせて物理アドレスで指定します。  
例えば、セグメントB000Hで、オフセット0の場合は、「B0000」と指定します。

物理アドレス2 メモリ上の何番地までセーブするかを指定します。  
指定法は「物理アドレス1」と同じです。

アドレス 出力する先頭アドレスを指定します。  
「物理アドレス1」を実際の何番地にするかを指定します。  
0からFFFFの範囲です。

例) A>SAVE S TEST B0000 B00FF 100

「TEST.S」という名で、セグメントB000、オフセット0から同じセグメントのFF番地までのコードを100番地としてセーブします。オブジェクトタイプはSフォーマットです。

## HEXADR (オブジェクトのアドレス確認)

このプログラムは、出力したオブジェクトのロードアドレスを表示するものです。

プログラム及び初期化されたデータのアドレスを詳細に表示します。ROM化を対象としている場合は、必ず表示されたアドレスはROM部でないといけません。

## HEXADR ファイル名

ファイル名            SフォーマットかインテルHEXファイルです。  
自動的にフォーマットを認識します。  
拡張子を省略した場合は、「.S」または「.HEX」を自動的に付けます。

例)    A>HEXADR TEST

「TEST」のロードアドレスを確認します。

「スタートアドレス」－「エンドアドレス」

という具合に連続して表示します。

---

CROSS DEBUGGER ユーザーズ・マニュアル

初 版 1987年 9月  
第2版 1988年11月  
第3版 1989年 6月  
第4版 1990年10月  
第5版 1997年 4月

発行責任 株式会社アークビット

---

- ・本書は改善のため事前連絡なしに変更することがあります。
- ・無断転載を禁じます。
- ・乱丁、落丁本はお取り替えします。