AKI-H8マイコン
モニター デバッガ

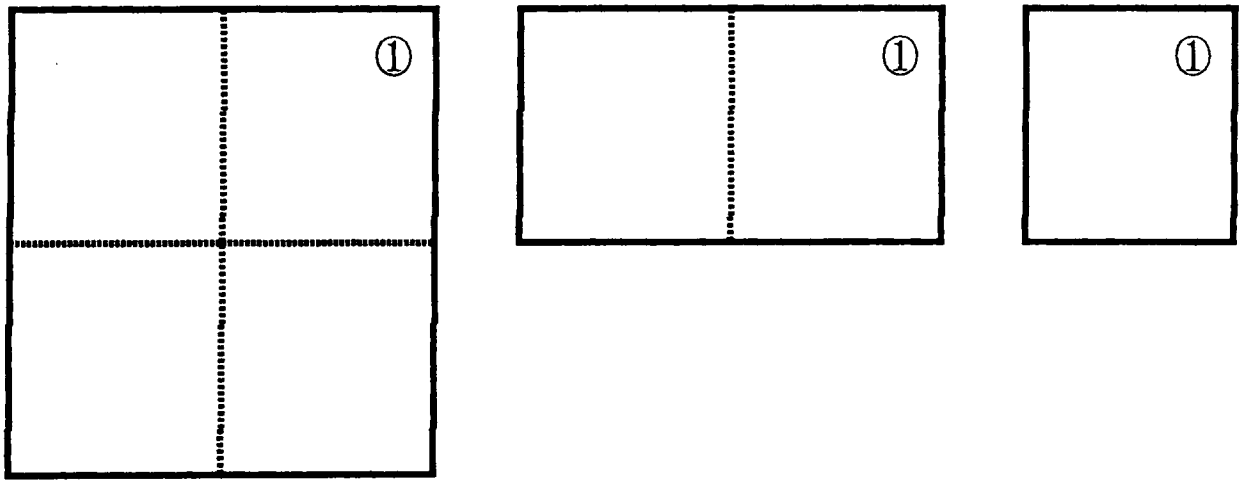
ブレーク機能、メモリーダンプ機能、逆アセンブル機能、
シングルステップ機能、メモリ書替え機能、
内蔵 I/O状態表示機能、など多彩なデバック機能付き。
OS:Windows95, PC/AT, 98

ＡＫＩ－Ｈ８／３０４８Ｆ
フラッシュROM内蔵 マイコンボード

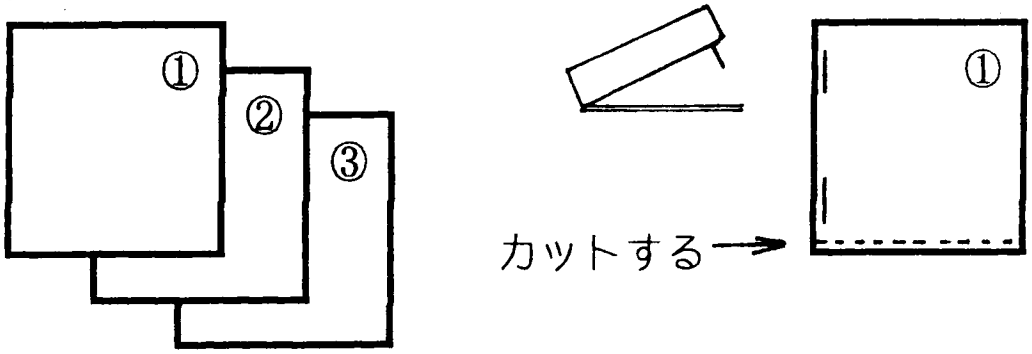
モニター デバッカ

★ブレーク機能・メモリーダンプ機能・逆アセンブル機能
シングルステップ機能・メモリ書替え機能・内蔵Ｉ／Ｏ
状態表示機能・等多彩なデバック機能付き
★ユーザー割り込み対応で割り込みのデバックもできます

■このマニュアルのつくりかた■
①～③の丸付き番号が右上にくるように四折りにします。



①～③の順で重ね合わせホチキスでとめます。



折り返しをカットして出来上がりです。

１．　【機能概要】

１．１　（１）組み込み型モニタの概要

組み込み型モニタは、ユーザの実機システムに組み込まれてユーザプログラムのデバッグを行うソフトウェアを言います。デバッグを行う開発装置と言えば、一般にエミュレータを思い浮かべますが、エミュレータのようにデバッグを行うための機器をほとんど使用しません。組み込み型モニタは実機システムのＲＯＭに格納し、ホスト端末よりダウンロードしたユーザプログラムのデバッグを行うのです。
つまり、エミュレータのようなエバチップは必要とせず、実機システム上の実チップを利用してユーザプログラムのデバッグを行います。このため、エミュレータのようにハードウェアのデバッグは行えませんが、ソフトウェアのデバッグであれば安価にデバッグを行うことができます。

★このモニタプログラムはフロッピーＤＩＳＫ内のＭＯＮＩＴＯＲ．ＭＯＴですのであらかじめＡＫＩ－Ｈ８マイコンにＭＯＮＩＴＯＲ．ＭＯＴを書き込んでください。

★ホスト端末はパソコンの通信ソフトを使用します。
ＷＩＮＤＯＷＳ９５添付の「ハイパーターミナル」、ＷＴＥＲＭ等をご用意ください。

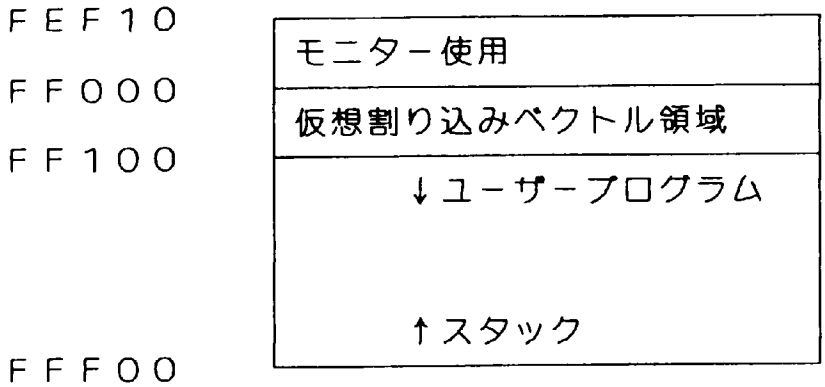
１．２　（２）ユーザープログラムの範囲と制限

このモニタはＲＳ－２３２Ｃを介してホスト端末とインタフェース（コマンドの送受信）を行います。ＡＫＩ－Ｈ８のＳＣＩ－１をホスト端末とのインタフェースとして、モニターが占有します。
ＳＣＩ－０　ユーザー使用可
ＳＣＩ－１　モニターが使用します。

ＲＯＭ領域はモニターが使用します。
ＲＡＭ領域のＦＥＦ１０～ＦＥＦＥ３はモニターが使用します。
ＲＡＭ領域のＦＦＯＯＯ～ＦＦＯＦＦはユーザー用の仮想割込みベクトル領域になります（仮想割り込みベクトル領域の使い方は３．【割込み管理】及びサンプルプログラムをごらんください。）
ユーザープログラムはＲＡＭ領域のＦＦ１００～ＦＦＦＯＯで動作します。

スタックポインタはＦＦＦＯＯになります。

◉ＲＡＭ領域の割り当て



組み込み型モニタは、ユーザプログラムの割込みに対してベクタテーブルの二重化方式をサポートしています。これにより、以下に示す以外の割込みに対し、当該ベクタアドレスを仮想的なベクタテーブルに用意するだけで割込み管理を行うことができます。割込み管理の詳細に関しては【割込み管理】を参照してください。
・リセット　－－＞　組み込み型モニタの起動用割込みとして使用
・ベクタ番号１　－－＞　シングルステップの制御として使用
・ベクタ番号２　－－＞　ブレークポイントの制御として使用

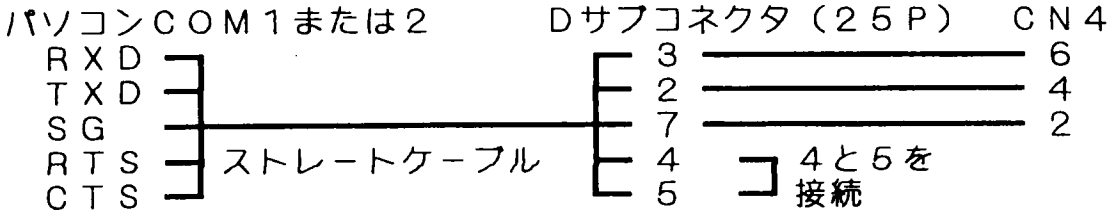
１．３　（３）ホスト端末との接続形態

組み込み型モニタは、ＲＳ－２３２Ｃを介してホスト端末と接続を行います。
以下にＲＳ－２３２Ｃのインタフェース仕様を示します。

・通信速度　　－－＞　９６００Ｂ／Ｓ
・データ長　　－－＞　８ビット長
・パリティ制御　－－＞　なし
・ストップビット長　－－＞　１ビット
・フロー制御　　－－＞　Ｘ－ＯＮ／ＯＦＦあり

■「ハイパーターミナル」の設定例（Ｃｏｍ１の場合）■
● 接続方法　Ｃｏｍ１へダイレクト
● モデムの設定
　ﾋﾞｯﾄ/秒　　９６００
　ﾃﾞｰﾀﾋﾞｯﾄ　　８
　ﾊﾟﾘﾃｨ　　なし
　ｽﾄｯﾌﾟﾋﾞｯﾄ　　１
　ﾌﾛｰ制御　　Ｘｏｎ／Ｘｏｆｆ
● ＡＳＣＩＩ設定
　ＡＳＣＩＩの送信
　☐ 行末に・・・　←選択しない
　☐ ローカルエコー　←選択しない
　ディレイ　行　０　ミリ秒
　　文字　０　ミリ秒
　ＡＳＣＩＩの受信
　☐ 受信データに改行文字を付ける　←選択しない
　☐ 受信データ７ビットに・・・　←選択しない
　☒ 右端で折り返す　←選択する

■ＡＫＩ－Ｈ８とホスト端末（パソコン）との接続■
●接続はマザーボード基板の場合、書き込みの接続（ＣＮ５）と同じです。
●ＡＫＩ－Ｈ８単体で接続する場合は次のようにしてください。



１．４　（４）機能概要

組み込み型モニタの機能（コマンド）を以下に示します。
各コマンドの詳細な使い方は【コマンド詳細】と【割込み管理】を参照してください。

２．１	Breakpoint	ブレークポイントの設定、表示、解除
２．２	Dump	メモリ内容のダンプ
２．３	DisAssemble	逆アセンブル
２．４	Fill	データの書き込み
２．５	Go	ユーザプログラムの実行
２．６	H8 status	内蔵周辺機能の状態表示
２．７	Load	ユーザプログラムのダウンロード
２．８	Memory	メモリ内容の表示、変更
２．９	Register	ＣＰＵレジスタの一覧表示
２．10	Step	シングルステップの実行
２．11	.(register)	ＣＰＵレジスタの表示、変更
２．12	help	コマンドヘルプ
２．13	return	繰り返し実行
２．14	(command).	コマンド履歴
３．４	interrupt	内蔵周辺機能割込み管理
３．５	abort	アボート管理

2.　【コマンド詳細】

組み込み型モニタが持っているコマンドの詳細を以下の形式で説明します。

例　《　B　（ブレークポイント. . . . ）　》　－－＞　コマンド名

（1）コマンドフォーマット　－－＞　コマンドの入力方法について説明しています。（下記のコマンドフォーマットの読み方を参照ください。）

（2）機能　－－＞　コマンドの機能概要を説明しています。

（3）解説　－－＞　コマンド機能の詳細を例を挙げて説明しています。

（4）注意事項　－－＞　コマンドを使用する上で特に注意すべき事項が記載してあります。

（5）備考　－－＞　コマンドを使用する上での補足説明が記載してあります。

コマンドフォーマットの特殊記号は以下に示す意味を持っています。

・'〈パラメータ〉' は〈　〉で囲まれた内容を指定することを意味します。
・'[　]' は[　]で囲まれた内容が省略可能であることを意味します。
・ [RET] は改行キーを入力することを意味します。
・コマンド　パラメータ間、各パラメータ間にはスペースが必要です。
・ブレークポイントのアドレス指定はＦＦＦ１００の様に６桁で指定してください
・ブレークポイント以外のアドレス指定はＦＦ１００の様に５桁で指定してください

----各コマンドの説明----

2．1　《　B　（ブレークポイントの設定、解除、表示）　》

（1）コマンドフォーマット

（a）　：B 〈アドレス〉 [RET]
（b）　：B - [〈アドレス〉] [RET]
（c）　：B [RET]

〈アドレス〉：ブレークポイントを設定、解除するアドレス
－　　：設定値の解除

（2）機能
ユーザプログラムを停止するアドレス（ブレークポイント）を設定、解除、表示します。
（a）最大８個までのブレークポイントが設定できます。
（b）設定されているブレークポイントを解除します。〈アドレス〉を省略すると全て解除します。
（c）設定されているブレークポイントを表示します。

（3）解説

（a）ブレークポイントの設定

：B FFF200 [RET]

H'FF200番地にブレークポイントを設定します。

既に設定されているアドレスを指定した場合は「Duplicate Breakpoint」のエラーメッセージを表示します。

```
: B FFF300 [RET]
: B FFF400 [RET]
```

続けてH'FF300番地、H'FF400番地にブレークポイントを設定します。
設定できるブレークポイントの個数は最大８個までです。ブレークポイントの設定が８個を超えた場合は「Full Breakpoint」のエラーメッセージを表示します。

```
: B FFF15C [RET]
: G FF100 [RET]
  Break at PC=FFF15C
  PC=FFF15C  CCR=0C:....NZ..  SP=000FFF10
  ER0=0000EFEF  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
```

ブレークポイントにユーザプログラムが到達した場合、上記のメッセージを表示してユーザプログラムを停止します。

（ｂ）ブレークポイントの解除

```
: B - FFF200 [RET]
```

H'FF200番地に設定してあるブレークポイントを解除します。
指定されたアドレスにブレークポイントがない場合は「Not Find Breakpoint」のエラーメッセージを表示します。

```
: B - [RET]
```

アドレスを省略すると設定されている全てのブレークポイントを解除します。

（ｃ）ブレークポイントの表示

```
: B [RET]
  (ADDR)
  FF15C
```

ブレークポイントがH'FF15C番地に設定してあります。

（４）注意事項

（ａ）ブレークポイントをＲＯＭ領域、内蔵周辺機能のレジスタ領域、およびメモリが接続されていない領域に設定した場合、Ｇ、Ｓコマンド実行時の動作は保証されません。
（ｂ）ブレークポイントは命令の先頭アドレスにのみ設定可能です。命令の先頭以外に設定した場合、Ｇ、Ｓコマンド実行時の動作は保証されません。
（Ｃ）ブレークポイントのアドレス指定はＦＦＦ１００の様に６桁で指定してください

２．２ 《 Ｄ （メモリ内容のダンプ） 》

（１）コマンドフォーマット

```
: D (アドレス１) [(アドレス２)] [;(サイズ)] [RET]
```

(アドレス１) ： ダンプするメモリの先頭アドレス

〈アドレス２〉 ： ダンプするメモリの最終アドレス
〈サイズ〉　　： 表示単位の指定
　　B　　： １バイト単位
　　W　　： ２バイト単位
　　L　　： ４バイト単位
　　省略時 ： １バイト単位

（２）機能

（ａ）指定されたメモリ領域の内容を表示します。〈アドレス２〉が省略された場合、256バイトをダンプします。
（ｂ）メモリダンプの表示は１行16バイト単位です。
（ｃ）[RET] のみの入力で前のＤコマンドで表示したアドレスの次のアドレスから256バイトを連続的にダンプします。

（３）解説

（ａ）メモリ内容のダンプ

```
: D 1000 [RET]
(ADDR)                          〈 D A T A 〉                          〈 ASCII CODE 〉
001000  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  '................'
001010  10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  '................'
001020  20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F  ' !"#$%&'()*+,-./'
001030  30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  '0123456789:;<=>?'
001040  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
001050  50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  'PQRSTUVWXYZ[¥]^_'
001060  60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  '`abcdefghijklmno'
001070  70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  'pqrstuvwxyz{|}~.'
001080  80 81 82 83 84 85 86 87  88 89 8A 8B 8C 8D 8E 8F  '................'
001090  90 91 92 93 94 95 96 97  98 99 9A 9B 9C 9D 9E 9F  '................'
0010A0  A0 A1 A2 A3 A4 A5 A6 A7  A8 A9 AA AB AC AD AE AF  '................'
0010B0  B0 B1 B2 B3 B4 B5 B6 B7  B8 B9 BA BB BC BD BE BF  '................'
0010C0  C0 C1 C2 C3 C4 C5 C6 C7  C8 C9 CA CB CC CD CE CF  '................'
0010D0  D0 D1 D2 D3 D4 D5 D6 D7  D8 D9 DA DB DC DD DE DF  '................'
0010E0  E0 E1 E2 E3 E4 E5 E6 E7  E8 E9 EA EB EC ED EE EF  '................'
0010F0  F0 F1 F2 F3 F4 F5 F6 F7  F8 F9 FA FB FC FD FE FF  '................'
```

H'1000番地よりメモリ内容をダンプします。
最終アドレスを指定しないと256バイトのメモリ内容をダンプします。

```
: D 1030 1058 [RET]
(ADDR)                          〈 D A T A 〉                          〈 ASCII CODE 〉
001030  30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  '0123456789:;<=>?'
001040  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
001050  50 51 52 53 54 55 56 57  58 59 5A 5B              'PQRSTUVWXYZ['
```

最終アドレスを指定すると最終アドレスまでのメモリ内容をダンプします。

```
: D 1030 1058;W [RET]
(ADDR)                          〈 D A T A 〉                          〈 ASCII CODE 〉
001030  3031  3233  3435  3637  3839  3A3B  3C3D  3E3F  '0123456789:;<=>?'
001040  4041  4243  4445  4647  4849  4A4B  4C4D  4E4F  '@ABCDEFGHIJKLMNO'
001050  5051  5253  5455  5657  5859  5A5B              'PQRSTUVWXYZ['
```

サイズをワード単位とすると２バイト単位でメモリ内容をダンプします。

```
: D 1030 105B;L [RET]
  (ADDR)                    ( D A T A )                        ( ASCII CODE )
  001030  30313233    34353637    38393A3B    3C3D3E3F    '0123456789:;<=>?'
  001040  40414243    44454647    48494A4B    4C4D4E4F    '@ABCDEFGHIJKLMNO'
  001050  50515253    54555657    58595A5B                'PQRSTUVWXYZ['
```

サイズをロングワード単位とすると４バイト単位でメモリ内容をダンプします。

（ｂ）繰り返しの表示

```
: D FF0 FFF [RET]
  (ADDR)                                ( D A T A )                                    ( ASCII CODE )
  000FF0  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
: [RET]
  001000  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  '................'
  001010  10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  '................'
  001020  20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F  ' !"#$%&'()*+,-./'
  001030  30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  '0123456789:;<=>?'
  001040  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
  001050  50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  'PQRSTUVWXYZ[¥]^_'
  001060  60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  '`abcdefghijklmno'
  001070  70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  'pqrstuvwxyz{|}~.'
  001080  80 81 82 83 84 85 86 87  88 89 8A 8B 8C 8D 8E 8F  '................'
  001090  90 91 92 93 94 95 96 97  98 99 9A 9B 9C 9D 9E 9F  '................'
  0010A0  A0 A1 A2 A3 A4 A5 A6 A7  A8 A9 AA AB AC AD AE AF  '................'
  0010B0  B0 B1 B2 B3 B4 B5 B6 B7  B8 B9 BA BB BC BD BE BF  '................'
  0010C0  C0 C1 C2 C3 C4 C5 C6 C7  C8 C9 CA CB CC CD CE CF  '................'
  0010D0  D0 D1 D2 D3 D4 D5 D6 D7  D8 D9 DA DB DC DD DE DF  '................'
  0010E0  E0 E1 E2 E3 E4 E5 E6 E7  E8 E9 EA EB EC ED EE EF  '................'
  0010F0  F0 F1 F2 F3 F4 F5 F6 F7  F8 F9 FA FB FC FD FE FF  '................'
```

Ｄコマンド実行後に［RET］のみを入力すると、前回ダンプした次のアドレスから再び256バイト単位で繰り返しメモリ内容をダンプします。

（４）注意事項

（ａ）ワード単位で表示を行う場合、先頭アドレスは偶数番地、最終番地は奇数番地でなければなりません。先頭アドレスが奇数番地の場合は「Invalid Start Address」、最終アドレスが偶数番地の場合は「Invalid End Address」のエラーメッセージを表示します。
また、ロングワード単位で表示を行う場合、先頭アドレスは偶数番地、最終番地は先頭アドレス＋３のＮ倍の番地でなければなりません。

（ｂ）Ｄコマンドで内蔵周辺機能のレジスタ領域を表示した場合、メモリ内容の16進数とASCIIコードの表示が異なることがあります。

（５）備考

Ｄコマンドではメモリの内容をリードする際、サイズで指定された単位でメモリ内容のリードを行います。すなわち、表示単位が１バイトの場合はバイトサイズでリードし、表示単位が２バイトの場合はワードサイズでリードし、表示単位が４バイトの場合はロングワードサイズでリードを行います。

2．3 《 ＤＡ （逆アセンブル） 》

（1）コマンドフォーマット

: DA 〈アドレス１〉 [〈アドレス２〉] [RET]

〈アドレス１〉：逆アセンブルするメモリの先頭アドレス
〈アドレス２〉：逆アセンブルするメモリの最終アドレス

（2）機能

（ａ）指定されたメモリ領域の内容を逆アセンブルします。〈アドレス２〉が省略された場合、16命令文を逆アセンブルします。
（ｂ）[RET] のみの入力で前のＤＡコマンドで表示した次のアドレスから16命令文を連続的に逆アセンブルします。

（3）解説

（ａ）メモリ内容の逆アセンブル

```
: DA FF100
 〈ADDR〉   〈CODE〉          〈MNEMONIC〉  〈OPERAND〉
 FF100    7A07000FFF10     MOV.L       #H'000FFF10:32,ER7
 FF106    F800             MOV.B       #H'00:8,R0L
 FF108    38C5             MOV.B       R0L,@H'FFFC5:8
 FF10A    F8FF             MOV.B       #H'FF:8,R0L
 FF10C    38DA             MOV.B       R0L,@H'FFFDA:8
 FF10E    F8FF             MOV.B       #H'FF:8,R0L
 FF110    38C8             MOV.B       R0L,@H'FFFC8:8
 FF112    F800             MOV.B       #H'00:8,R0L
 FF114    3862             MOV.B       R0L,@H'FFF62:8
 FF116    F8C3             MOV.B       #H'C3:8,R0L
 FF118    3864             MOV.B       R0L,@H'FFF64:8
 FF11A    F8B8             MOV.B       #H'B8:8,R0L
 FF11C    3865             MOV.B       R0L,@H'FFF65:8
 FF11E    79004E20         MOV.W       #H'4E20:16,R0
 FF122    6B80FF6C         MOV.W       R0,@H'FFF6C:16
 FF126    F8B7             MOV.B       #H'B7:8,R0L
```

最終アドレスを指定しないと16命令文の逆アセンブルを行います。

```
: DA 1000 1005 [RET]
 〈ADDR〉   〈CODE〉          〈MNEMONIC〉  〈OPERAND〉
 001000   6CD3             MOV.B       R3H,@-ER5
 001002   0A0B             INC.B       R3L
 001004   7A0100001028     MOV.L       #H'00001028:32,ER1
```

最終アドレスを指定すると最終アドレスを含む命令まで逆アセンブルを行います。

（ｂ）繰り返しの表示

```
: DA 1000 1005 [RET]
 〈ADDR〉   〈CODE〉          〈MNEMONIC〉  〈OPERAND〉
 001000   6CD3             MOV.B       R3H,@-ER5
 001002   0A0B             INC.B       R3L
 001004   7A0100001028     MOV.L       #H'00001028:32,ER1
```

```
: [RET]
  00100A  5C0002DA          BSR       0012E8:16
  00100E  40C8              BRA       000FD8:8
  001010  5C0002C4          BSR       0012D8:16
  001014  7A050020FF56      MOV.L     #H'0020FF56:32,ER5
  00101A  FB0F              MOV.B     #H'0F:8,R3L
  00101C  1A80              SUB.L     ER0,ER0
  00101E  01006DD0          MOV.L     ER0,@-ER5
  001022  1A0B              DEC.B     R3L
  001024  46F8              BNE       00101E:8
  001026  5470              RTS
  001028  0820              ADD.B     R2H,R0H
  00102A  0800              ADD.B     R0H,R0H
  00102C  6E68000A          MOV.B     @(H'00000A:16,ER6),R0L
  001030  474C              BEQ       00107E:8
  001032  0F95              MOV.L     ER1,ER5
  001034  01006D56          MOV.L     @ER5+,ER6
```

②

ＤＡコマンド実行後に［RET］のみを入力すると、前回のＤＡコマンドの最終アドレスを含む次のアドレスから再び16命令文の逆アセンブルを繰り返し行います。

２．４　《　Ｆ　（データの書き込み）　》

（１）コマンドフォーマット

```
: F (アドレス１) (アドレス２) (書き込みデータ) [RET]
```

(アドレス１)　　：書き込みするメモリの先頭アドレス
(アドレス２)　　：書き込みするメモリの最終アドレス
(書き込みデータ)：１バイトの書き込みデータ

（２）機能

指定されたメモリ領域に、(書き込みデータ)で指定された１バイトのデータを書き込みます。

（３）解説

（ａ）データの書き込み

```
: F FF100 FF1FF AA [RET]
```

H'FF100番地からH'FF1FF番地までのメモリ領域に対してH'AAのデータを書き込みます。

（ｂ）データの書き込みの失敗

```
: F FF100 FF1FF AA [RET]
  Failed at FF115 , Write = 55 , Read = 04
```

Ｆコマンドでは書き込みデータのベリファイチェックを行います。ベリファイチェックでエラーが検出された場合は上記のメッセージを表示します。

２．５　《　Ｇ　（ユーザプログラムの実行）　》

（１）コマンドフォーマット

```
: G [(アドレス)] [RET]
```

(アドレス)：実行するユーザプログラムの先頭アドレス

（２）機能

現在のプログラムカウンタ値ないしは、〈アドレス〉で指定したアドレスよりユーザプログラムを実行します。

（３）解説

（ａ）ユーザプログラムの実行

```
: G [RET]
```

現在のプログラムカウンタ値よりユーザプログラムを実行します。

```
: G FF100 [RET]
```

H'FF100番地よりユーザプログラムを実行します。

（ｂ）ユーザプログラムの停止

```
: B FFF15C [RET]
: G FF100 [RET]
  Break at PC=FFF15C
  PC=FFF15C  CCR=0C:....NZ..  SP=000FFF10
  ER0=0000EFEF  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
```

ブレークポイントに到達するとユーザプログラムは停止します。

```
: G FF100 [RET]
NMI (Abort Switch ON) 入力
  Abort at PC=FFF150
  PC=FFF150  CCR=80:I.......  SP=00FFFF00
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
```

ＮＭＩ入力を行うとユーザプログラムを強制停止します。
（ＮＭＩ入力については３．５　ａｂｏｒｔをごらんください。）

（４）注意事項

実行する最初の命令が不当命令の場合は「Invalid Instruction」のエラーメッセージを表示し、ユーザプログラムの実行を行いません。

２．６　《　Ｈ８　（内蔵周辺機能の状態表示）　》

（１）コマンドフォーマット

```
: H8 [(周辺機能名)] [RET]
: H8 ?
 Displays contents of peripheral registers.
      H8 (name) [RET]
   (name) : DMAC0 - Direct Memory Access Controller 0
            DMAC1 - Direct Memory Access Controller 1
            ITU   - 16bit Integrated Timer pulse Unit
            ITU0  - 16bit Integrated Timer pulse Unit 0
            ITU1  - 16bit Integrated Timer pulse Unit 1
            ITU2  - 16bit Integrated Timer pulse Unit 2
            ITU3  - 16bit Integrated Timer pulse Unit 3
```

```
ITU4  - 16bit Integrated Timer pulse Unit 4
TPC   - programmable Timing Pattern Controller
WDT   - Watch Dog Timer
SCI0  - Serial Communication Interface 0
SCI1  - Serial Communication Interface 1
I/O   - I/O port
D/A   - D/A converter
A/D   - A/D converter
INTC  - INTerrupt Controller
BSC   - BuS Controller, etc.
```

（２）機能

内蔵周辺機能のレジスタの状態を表示します。

（３）解説

（ａ）Ｉ／Ｏポートの表示

```
: H8 I/O
(REG)   (ADDR) (CODE)    ( 7      6      5      4      3      2      1      0 )
P1DDR    FFC0 FF
P1DR     FFC2 00000000
P2DDR    FFC1 FF
P2DR     FFC3 00000000
P2PCR    FFD8 00000000
P3DDR    FFC4 FF
P3DR     FFC6 00000000
P4DDR    FFC5 FF
P4DR     FFC7 00000000
P4PCR    FFDA 00000000
P5DDR    FFC8 FF
P5DR     FFCA ....0000
P5PCR    FFDB ....0000
P6DDR    FFC9 FF
P6DR     FFCB .0000000
P7DR     FFCE 11000000   AN7    AN6    AN5    AN4    AN3    AN2    AN1    AN0
                         DA1    DA0
P8DDR    FFCD FF
P8DR     FFCF ...00000                         IRQ3   IRQ2   IRQ1   IRQ0
P9DDR    FFD0 FF
P9DR     FFD2 ..001111                 SCK1   SCK0   RXD1   RXD0   TXD1   TXD0
                                       IRQ5   IRQ4
PADDR    FFD1 FF
PADR     FFD3 00000000    TP7    TP6    TP5    TP4    TP3    TP2    TP1    TP0
                        TIOCB2 TIOCA2 TIOCB1 TIOCA1 TIOCB0 TIOCA0 TEND1  TEND0
                                                    TCLKD  TCLKC  TCLKB  TCLKA
PBDDR    FFD4 FF
PBDR     FFD6 00000000    TP15   TP14   TP13   TP12   TP11   TP10   TP9    TP8
                        DREQ1  DREQ0  TOCXB4 TOCXA4 TIOCB4 TIOCA4 TIOCB3 TIOCA3
                        ADTRG
```

Ｉ／Ｏポートの状態表示では、現状の動作モードで使用可能なものを表示します。(REG)ではレジスタの名称、(ADDR)ではレジスタの番地、(CODE)ではレジスタの値を表示します。なお、(CODE)では各レジスタの仕様に合わせて、バイト単位またはビット単位で値を表示します。また、(76543210)では各Ｉ／Ｏポートとの兼用端子機能を表示します。

（ｂ）Ｉ／Ｏポート以外の表示

```
: H8 ITU0
(REG)   (ADDR) (CODE)    ( 7      6      5      4      3      2      1      0 )
TCR      FF64 .0000000           CCLR1  CCLR0  CKEG1  CKEG0  TPSC2  TPSC1  TPSC0
TIOR     FF65 .000.000           IOB2   IOB1   IOB0           IOA2   IOA1   IOA0
TIER     FF66 .....000                                         OVIE   IMIEB  IMIEA
TSR      FF67 .....000                                         OVF    IMFB   IMFA
TCNT     FF68 0000
GRA      FF6A FFFF
GRB      FF6C FFFF
: H8 SCI0
(REG)   (ADDR) (CODE)    ( 7      6      5      4      3      2      1      0 )
SMR      FFB0 00000000   C/A    CHR    PE     O/E    STOP   MP     CKS1   CKS0
BRR      FFB1 FF
SCR      FFB2 00000000   TIE    RIE    TE     RE     MPIE   TEIE   CKE1   CKE0
TDR      FFB3 FF
SSR      FFB4 10000100   TDRE   RDRF   ORER   FER    PER    TEND   MPB    MPBT
RDR      FFB5 00
```

Ｉ／Ｏポート以外の状態表示では、指定された周辺機能の状態を表示します。(REG)ではレジスタの名称、(ADDR)ではレジスタの番地、(CODE)ではレジスタの値を表示します。なお、(CODE)では各レジスタの仕様に合わせて、ロングワード単位、ワード単位、バイト単位またはビット単位で値を表示します。また、(76543210)ではビット単位に意味のあるレジスタのみ対応するビットの意味を表示します。

２．７　《　Ｌ　（ユーザプログラムのダウンロード）　》

（１）コマンドフォーマット

```
: L [RET]
```

（２）機能

ホスト端末より、指定されたＳタイプフォーマットのロードモジュールをメモリ上にダウンロードします。

（３）解説

（ａ）ユーザプログラムのダウンロード

```
: L [RET]      (-- コマンド投入後、ホスト端末よりプログラムをファイル転送する。
  Top Address=FF070
  End Address=FF1A5
```

（ハイパーターミナルの場合は　転送（Ｔ）－テキストファイルの送信で　．ＭＯＴファイルを転送してください。）

（ｂ）ダウンロードの失敗

```
: L [RET]
  ********  S Type Format Error  ********
```

Ｓタイプフォーマットでないロードモジュールを転送すると上記のエラーメッセージを表示します。

```
: L [RET]
******** Check Sum Error ********
```

チェックサムが不正の場合は上記のエラーメッセージを表示します。

（4）注意事項

（a）Lコマンド投入後、ホスト端末の通信ソフトを利用してロードモジュールをファイル転送（テキストファイルの転送）してください。
（b）LコマンドはSタイプフォーマットのロードモジュールしかダウンロードできません。

2．8 《 M （メモリ内容の表示、変更） 》

（1）コマンドフォーマット

: M 〈アドレス〉 [;〈サイズ〉] [RET]

〈アドレス〉：表示、変更を行うメモリの先頭アドレス
〈サイズ〉　：表示、変更の単位の指定
　　B　　：1バイト単位
　　W　　：2バイト単位
　　L　　：4バイト単位
　　省略時：1バイト単位

（2）機能

指定されたアドレスのメモリ内容を〈サイズ〉で指定した単位で表示、変更します。
コマンド投入後は下記の操作が可能です。
- [RET] を入力すると次のメモリ内容を表示します。
- ^ [RET] を入力すると前のメモリ内容を表示します。
- 〈データ〉 [RET] を入力するとメモリの内容を〈データ〉に変更します。
- . [RET] を入力するとMコマンドを終了します。

（3）解説

（a）バイト単位の表示、変更

```
: M FF12D  [RET]
FF12D   6F  ? [RET]
FF12E   79  ? [RET]
FF12F   00  ? [RET]
FF130   00  ? [RET]
FF131   64  ?  FF[RET]
FF132   6B  ?  .[RET]
```

H'FF131の内容をH'FFに変更します。

（b）ワード単位の表示、変更

```
: M FF130 ;W [RET]
FF130   00FF  ? 1234[RET]
FF132   6B80  ?  .[RET]
```

H'FF130番地からの内容をH'1234に変更します。

（ｃ）ロングワード単位の表示、変更

```
: M FF100 ;L [RET]
  FF100  BCD567D1  ?  12345678 [RET]
  FF104  B80AABCD  ?  . [RET]
```

H'FF100番地からの内容をH'12345678に変更します。

（ｄ）ベリファイチェック

```
: M 1000 [RET]
  01000  BC  ?  34 [RET]
  01001  D1  ?  AB [RET]
  01002  B8  ?  12 [RET]
  **** Verify Error ****
  01002  B8  ?
```

Ｍコマンドではメモリ内容の変更の際にベリファイエラーが検出されると、再び当該アドレスの内容を表示してコマンド待ち状態となります。なお、内蔵周辺機能のレジスタ領域に対してはベリファイチェックを行いません。

（４）注意事項

ワード単位、ロングワード単位でメモリの表示、変更を行う場合は、アドレスは偶数番地でなければなりません。アドレスを奇数番地とした場合は「Invalid Start Address」のエラーメッセージを表示します。

（５）備考

Ｍコマンドではメモリの内容をリード／ライトする際、サイズで指定された単位でリード／ライトを行います。すなわち、サイズが１バイトの場合はバイトサイズでリード／ライトし、サイズが２バイトの場合はワードサイズでリード／ライトし、サイズが４バイト単位の場合はロングワードサイズでリード／ライトを行います。

２．９ 《 Ｒ （ＣＰＵレジスタの一覧表示） 》

（１）コマンドフォーマット

```
: R [RET]
```

（２）機能

ＣＰＵのコントロールレジスタ、汎用レジスタの一覧を表示します。

（３）解説

```
: R [RET]
  PC=000000  CCR=FF:IUHUNZVC  SP=00FFFF00
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
  PC  : プログラムカウンタ
  CCR : コンディションコードレジスタ
  [IUHUNZVC] :  I : 割込みマスク・ビット      U : ユーザ・割込みマスク・ビット
                H : ハーフ・キャリー・フラグ  U : ユーザ・ビット
                N : ネガティブ・フラグ        Z : ゼロ・フラグ
                V : オーバフロー・フラグ      C : キャリー・フラグ
  SP : スタックポインタ
  ER0～ER7 : 汎用レジスタ
```

２．１０ 《 Ｓ （シングルステップの実行） 》

②′

（１）コマンドフォーマット

：Ｓ［(実行ステップ数)］［RET］

(実行ステップ数) ： 実行する命令数（10進数２桁で表現）

（２）機能

ユーザプログラムのシングルステップ実行を行い、レジスタ内容と実行した命令を表示します。

（ａ）現在のプログラムカウンタ値から指定された命令数分のユーザプログラムを実行します。
（ｂ）実行ステップ数が省略されると１命令だけユーザプログラムを実行します。
（ｃ）実行ステップ数で０を指定すると100命令分ユーザプログラムを実行します。
（ｄ）［RET］のみの入力で前のＳコマンドで実行した命令の次の命令から再び指定された命令数分ユーザプログラムを実行します。

（３）解説

（ａ）１命令だけの実行

```
: S [RET]
PC=FFF106  CCR=80:I.......  SP=000FFF10
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
FF100   7A07000FFF10          MOV.L      #H'000FFF10:32,ER7
```

実行ステップ数が省略されると１命令だけユーザプログラムを実行します。

（ｂ）複数命令の実行

```
: S 3 [RET]
PC=FFF108  CCR=84:I....Z..  SP=000FFF10
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
FF106   F800                  MOV.B      #H'00:8,ROL       0

PC=FFF10A  CCR=84:I....Z..  SP=000FFF10
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
FF108   38C5                  MOV.B      ROL,@H'FFFC5:8    1

PC=FFF10C  CCR=88:I...N...  SP=000FFF10
ER0=000000FF  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
FF10A   F8FF                  MOV.B      #H'FF:8,ROL
```

実行ステップ数を指定すると指定された命令数分だけ連続的にユーザプログラムを実行します。

（ｃ）繰り返しの実行

```
: S [RET]
  PC=FFF10E   CCR=88:I...N...   SP=000FFF10
  ER0=000000FF   ER1=00000000   ER2=00000000   ER3=00000000
  ER4=00000000   ER5=00000000   ER6=00000000   ER7=000FFF10
  FF10C    38DA                   MOV.B       ROL,@H'FFDA:8
: [RET]
  PC=FFF110   CCR=88:I...N...   SP=000FFF10
  ER0=000000FF   ER1=00000000   ER2=00000000   ER3=00000000
  ER4=00000000   ER5=00000000   ER6=00000000   ER7=000FFF10
  FF10E    F8FF                   MOV.B       #H'FF:8,ROL
: [RET]
  PC=FFF112   CCR=88:I...N...   SP=000FFF10
  ER0=000000FF   ER1=00000000   ER2=00000000   ER3=00000000
  ER4=00000000   ER5=00000000   ER6=00000000   ER7=000FFF10
  FF110    38C8                   MOV.B       ROL,@H'FFC8:8
```

Ｓコマンド実行後に［RET］のみを入力すると、前回のＳコマンドで実行した次の命令から再び指定された命令数分（上記の場合１命令）ユーザプログラムを実行します。

（４）注意事項

実行する命令が不当命令の場合は「Invalid Instruction」のエラーメッセージを表示しユーザプログラムの実行を行いません。

２．１１　《　.(register)　（ＣＰＵレジスタの表示、変更）　》

（１）コマンドフォーマット

: .　(レジスタ名)　[(データ)]　[RET]

(レジスタ名)　：表示、変更を行うＣＰＵのレジスタ名
(データ)　　：設定値

（２）機能

ＣＰＵのコントロール／汎用レジスタの内容を表示、変更します。
(データ)を指定すると当該のレジスタのみ変更を行います。
(データ)を省略すると当該のレジスタから順番に会話形式でレジスタ値の表示、変更を行います。

- [RET] を入力すると次のレジスタ内容を表示します。
- ^ [RET] を入力すると前のレジスタ内容を表示します。
- (データ) [RET] を入力するとレジスタの内容を(データ)に変更します。
- . [RET] を入力するとコマンドを終了します。

なお、レジスタの表示順は以下の通りです。

```
ER0, ER1, ER2, ER3, ER4, ER5, ER6, ER7, PC, CCR, SP
R0,  R1,  R2,  R3,  R4,  R5,  R6,  R7
E0,  E1,  E2,  E3,  E4,  E5,  E6,  E7
R0L, R1L, R2L, R3L, R4L, R5L, R6L, R7L
R0H, R1H, R2H, R3H, R4H, R5H, R6H, R7H
```

（３）解説

（ａ）レジスタの変更

```
: R [RET]
  PC=000000  CCR=FF:IUHUNZVC  SP=00FFFF00
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
: .ER0 12345678 [RET]
: .R1 ABCD [RET]
: .R2L EF [RET]
: R [RET]
  PC=000000  CCR=FF:IUHUNZVC  SP=00FFFF00
  ER0=12345678  ER1=0000ABCD  ER2=000000EF  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
```

〈データ〉を指定すると当該レジスタのみ変更を行います。

（ｂ）レジスタの表示、変更

```
: R [RET]
  PC=000000  CCR=80:I.......  SP=00FFFF00
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
: .ER6 [RET]
  ER6=00000000  ?  12345678 [RET]
  ER7=00FFFF00  ?  [RET]
  PC=000000  ?  200100 [RET]
  CCR=80  ?  [RET]
  SP=00FFFF00  ?  ' [RET]
  CCR=80  ?  ' [RET]
  PC=200100  ?  ' [RET]
  ER7=00FFFF00  ?  FFFF10 [RET]
  PC=200100  ?  . [RET]
: R [RET]
  PC=200100  CCR=80:I.......  SP=00FFFF10
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=12345678  ER7=00FFFF10
```

〈データ〉を省略すると会話形式でレジスタの表示、変更を行います。

２．１２　《 help （コマンドヘルプ） 》

（１）コマンドフォーマット

```
: [〈コマンド名〉] ? [RET]
```

〈コマンド名〉：使用方法を表示したいコマンドの名称

（２）機能

コマンドの使用方法を表示します。
（ａ）〈コマンド名〉を省略するとモニタが持っているコマンドの一覧を表示します。
（ｂ）〈コマンド名〉を指定すると該当のコマンドの使用方法を表示します。

（３）解説

（ａ）コマンドの一覧表示

```
: ? [RET]

  Monitor Vector 00000 - 000FF
  Monitor ROM    00100 - 04773
  Monitor RAM    FEF10 - FEFE3
  User    Vector FF000 - FF0FF

  .   : Changes contents of H8/300H registers.
  B   : Sets or displays or clear breakpoint(s).
  D   : Displays memory contents.
  DA  : Disassembles memory contents.
  F   : Fills specified memory range with data.
  G   : Executes real-time emulation.
  H8  : Displays contents of peripheral registers.
  L   : Loads user program into memory from host system.
  M   : Changes memory contents.
  R   : Displays contents of H8/300H registers.
  S   : Executes single emulation(s) and displays instruction and registers.
```

〈コマンド名〉を省略するとモニタのメモリマップ及びコマンドの一覧を表示します。

（ｂ）コマンドの詳細表示

```
: B ? [RET]
 1. Sets breakpoint at specified address.
      B 〈address〉 [RET]
 2. Displays breakpoint(s).
      B [RET]
 3. Clear breakpoint(s).
      B - [〈address〉] [RET]
   〈address〉 : address of breakpoint
: D ? [RET]
 Displays memory contents.
      D 〈address1〉 [〈address2〉] [;〈size〉] [RET]
   〈address1〉 : dump area start address
   〈address2〉 : dump area end   address
   〈size〉     : B -- byte
                W -- word
                L -- long word
: F ? [RET]
 Fills specified memory range with data.
      F 〈address1〉 〈address2〉 〈data〉 [RET]
   〈address1〉 : filling area start address
   〈address2〉 : filling area end   address
   〈data〉     : filling byte data
: H8 ? [RET]
 Displays contents of H8/3003 peripheral registers.
      H8 〈name〉 [RET]
   〈name〉 : DMAC0 - Direct Memory Access Controller 0
            DMAC1 - Direct Memory Access Controller 1
            DMAC2 - Direct Memory Access Controller 2
            DMAC3 - Direct Memory Access Controller 3
            ITU   - 16bit Integrated Timer pulse Unit
            ITU0  - 16bit Integrated Timer pulse Unit 0
            ITU1  - 16bit Integrated Timer pulse Unit 1
```

```
ITU2  - 16bit Integrated Timer pulse Unit 2
ITU3  - 16bit Integrated Timer pulse Unit 3
ITU4  - 16bit Integrated Timer pulse Unit 4
TPC   - programmable Timing Pattern Controller
WDT   - Watch Dog Timer
SCI0  - Serial Communication Interface 0
SCI1  - Serial Communication Interface 1
I/O   - I/O port
A/D   - A/D converter
INTC  - INTerrupt Controller
BSC   - BuS Controller, etc.
```

〈コマンド名〉を指定すると当該コマンドの詳細な使い方を表示します。

２．１３　《 return （繰り返し実行） 》

（１）コマンドフォーマット

```
: [RET]
```

（２）機能

Ｄ、ＤＡ、Ｓコマンドの [RET] のみによる繰り返し実行を行います。

（３）解説

（ａ）Ｄコマンドの繰り返し実行

```
: D FF0 FFF [RET]
  〈ADDR〉                    〈 D  A  T  A 〉                      〈 ASCII CODE 〉
  000FF0  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
: [RET]
  001000  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  '................'
  001010  10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  '................'
  001020  20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F  ' !"#$%&'()*+,-./'
  001030  30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  '0123456789:;<=>?'
  001040  40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  '@ABCDEFGHIJKLMNO'
  001050  50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  'PQRSTUVWXYZ[¥]^_'
  001060  60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  '`abcdefghijklmno'
  001070  70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  'pqrstuvwxyz{|}~.'
  001080  80 81 82 83 84 85 86 87  88 89 8A 8B 8C 8D 8E 8F  '................'
  001090  90 91 92 93 94 95 96 97  98 99 9A 9B 9C 9D 9E 9F  '................'
  0010A0  A0 A1 A2 A3 A4 A5 A6 A7  A8 A9 AA AB AC AD AE AF  '................'
  0010B0  B0 B1 B2 B3 B4 B5 B6 B7  B8 B9 BA BB BC BD BE BF  '................'
  0010C0  C0 C1 C2 C3 C4 C5 C6 C7  C8 C9 CA CB CC CD CE CF  '................'
  0010D0  D0 D1 D2 D3 D4 D5 D6 D7  D8 D9 DA DB DC DD DE DF  '................'
  0010E0  E0 E1 E2 E3 E4 E5 E6 E7  E8 E9 EA EB EC ED EE EF  '................'
  0010F0  F0 F1 F2 F3 F4 F5 F6 F7  F8 F9 FA FB FC FD FE FF  '................'
```

Ｄコマンド実行後に [RET] のみを入力すると、前回ダンプした次のアドレスから再び256バイト単位で繰り返しメモリ内容をダンプします。

（b）ＤＡコマンドの繰り返し実行

```
: DA 1000 1005 [RET]
  (ADDR)   (CODE)              (MNEMONIC)  (OPERAND)
  001000   6CD3                MOV.B       R3H,@-ER5
  001002   0A0B                INC.B       R3L
  001004   7A0100001028        MOV.L       #H'00001028:32,ER1
: [RET]
  00100A   5C0002DA            BSR         0012E8:16
  00100E   40C8                BRA         000FD8:8
  001010   5C0002C4            BSR         0012D8:16
  001014   7A050020FF56        MOV.L       #H'0020FF56:32,ER5
  00101A   FB0F                MOV.B       #H'0F:8,R3L
  00101C   1A80                SUB.L       ER0,ER0
  00101E   01006DD0            MOV.L       ER0,@-ER5
  001022   1A0B                DEC.B       R3L
  001024   46F8                BNE         00101E:8
  001026   5470                RTS
  001028   0820                ADD.B       R2H,R0H
  00102A   0800                ADD.B       R0H,R0H
  00102C   6E68000A            MOV.B       @(H'00000A:16,ER6),R0L
  001030   474C                BEQ         00107E:8
  001032   0F95                MOV.L       ER1,ER5
  001034   01006D56            MOV.L       @ER5+,ER6
```

ＤＡコマンド実行後に［RET］のみを入力すると、前回のＤＡコマンドの最終アドレスを含む次のアドレスから再び16命令文の逆アセンブルを繰り返し行います。

（c）Ｓコマンドの繰り返し実行

```
: S [RET]
  PC=200106  CCR=80:I.......  SP=00FFFF00
  ER0=01234567  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
  200100  7A0001234567        MOV.L       #H'01234567:32,ER0
: [RET]
  PC=20010C  CCR=88:I...N...  SP=00FFFF00
  ER0=01234567  ER1=89ABCDEF  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
  200106  7A0189ABCDEF        MOV.L       #H'89ABCDEF:32,ER1
: [RET]
  PC=20010E  CCR=80:I.......  SP=00FFFF00
  ER0=01234567  ER1=89ABCDEF  ER2=01234567  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
  20010C  0F82                MOV.L       ER0,ER2
```

Ｓコマンド実行後に［RET］のみを入力すると、前回のＳコマンドで実行した次の命令から再び指定された命令数分（上記の場合１命令）ユーザプログラムを実行します。

２．１４　《 (command). （コマンド履歴） 》

（１）コマンドフォーマット

：〈コマンド名〉．[RET]

〈コマンド名〉：前回の内容を表示したいコマンドの名称
Ｄ、ＤＡ、Ｆ、Ｈ８、Ｍが指定可能

（２）機能

指定されたコマンドの前回の内容を表示し、キーボード入力待ち状態となります。

（３）解説

（ａ）コマンド履歴表示

```
: DA 1000 1005 [RET]
  〈ADDR〉  〈CODE〉                〈MNEMONIC〉  〈OPERAND〉
  001000  6CD3                  MOV.B       R3H,@-ER5
  001002  0A0B                  INC.B       R3L
  001004  7A0100001028          MOV.L       #H'00001028:32,ER1
: DA. [RET]
: DA 1000 1005
```

指定されたコマンドの前回の内容を表示し、キーボード入力待ち状態となります。

（ｂ）別コマンド実行後のコマンド履歴表示

```
: DA 1000 1005 [RET]
  〈ADDR〉  〈CODE〉                〈MNEMONIC〉  〈OPERAND〉
  001000  6CD3                  MOV.B       R3H,@-ER5
  001002  0A0B                  INC.B       R3L
  001004  7A0100001028          MOV.L       #H'00001028:32,ER1
: R [RET]
  PC=000000  CCR=80:I.......  SP=00FFFF00
  ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
  ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
: DA. [RET]
: DA 1000 1005
```

他のコマンドを実行後でもコマンド履歴を使用できます。

３．　【割込み管理】

組み込み型モニタは、ユーザプログラムの割込みに対してベクタテーブルの二重化方式をサポートしています。これにより、以下に示す組み込み型モニタ予約の割込み以外に対し、当該ベクタアドレスを仮想的なベクタテーブルに用意するだけで割込み管理を行うことができます。

・リセット　　－－＞　組み込み型モニタの起動用割込みとして使用
・ベクタ番号１　－－＞　シングルステップの制御として使用
・ベクタ番号２　－－＞　ブレークポイントの制御として使用

３．１　（１）ベクタテーブルの二重化方式

ベクタテーブルの二重化方式とは、本来 H'000000 番地から配置すべきベクタテーブルを別のＲＡＭ領域から配置することを言います。つまり、実機システムでは H'000000 番地以降の数バイトはベクタテーブルを配置するためＲＯＭ領域となります。組み込み型モニタを使用した場合もこれを変更することはできません。しかし、ユーザプログラムをホスト端末よりダウンロードしてデバックする場合、ユーザプログラムのベクタテーブルもホスト端末よりダウンロードすることになります。すると、ＲＯＭ領域にユーザプログラムのベクタテーブルをダウンロードすることになってしまいます。当然の事ながらＲＯＭ領域にユーザプログラムをダウンロードはできません。そこで、組み込み型モニタではユーザプログラムのベクタテーブルを別のＲＡＭ領域（仮想ベクタ領域）から配置し、割込み発生時はそのベクタテーブルを参照してユーザの割込みプログラムに起動をかけます。
このモニターは仮想ベクタ領域を H'FFF000 番地から配置しています。

	本来のベクタアドレス	仮想ベクタアドレス
リセットＰＣ初期値	H'000000 ～ H'000003	H'FFF000 ～ H'FFF003
（システム予約）	H'000004 ～ H'00001B	H'FFF004 ～ H'FFF01B
外部割込み　ＮＭＩ	H'00001C ～ H'00001F	H'FFF01C ～ H'FFF01F
ＴＲＡＰＡ　＃０命令	H'000020 ～ H'000023	H'FFF020 ～ H'FFF023
ＴＲＡＰＡ　＃１命令	H'000024 ～ H'000027	H'FFF024 ～ H'FFF027
ＴＲＡＰＡ　＃２命令	H'000028 ～ H'00002B	H'FFF028 ～ H'FFF02B
ＴＲＡＰＡ　＃３命令	H'00002C ～ H'00002F	H'FFF02C ～ H'FFF02F

３．２　（２）組み込み型モニタ予約の割込み

以下に示す割込みは組み込み型モニタ予約の割込みです。これらの割込みに対してユーザが割込みプログラムを記述することはできません。

・リセット　　－－＞　組み込み型モニタの起動用割込みとして使用
・ベクタ番号１　－－＞　シングルステップの制御として使用
・ベクタ番号２　－－＞　ブレークポイントの制御として使用

これらの割込みが発生した場合、組み込み型モニタは以下のメッセージを表示します。

（ａ）リセット

```
H8/3048 Serise Advanced Mode Monitor Ver. 2.0C
AKI-H8 3048/F
1998
```

リセットが入力されると組み込み型モニタは初期化処理を行い上記のメッセージを表示し、コマンド待ち状態に入ります。

（ｂ）ベクタ番号１

```
@@H'04:8 Addressing !!
PC=000000  CCR=80:I.......  SP=00FFFF00
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
```

ユーザプログラム実行中にベクタ番号１の割込み（メモリ間接）が発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

（ｃ）ベクタ番号２

```
@@H'08:8 Addressing !!
PC=000000  CCR=80:I.......  SP=00FFFEFC
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFEFC
```

ユーザプログラム実行中にベクタ番号２の割込み（メモリ間接）が発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

３．３　（３）未定義割込み

組み込み型モニタ予約の割込み以外の割込みは、仮想ベクタテーブルに割込みプログラムのアドレスを記述するだけでサポートすることができます。ただし、このベクタアドレスが記述されていない割込みが発生した場合、組み込み型モニタは以下に示すような割込み対応のメッセージを表示してユーザプログラムの実行を停止します。

（ａ）ＪＭＰ命令のメモリ間接

```
@@H'10:8 Addressing !!
PC=000000  CCR=80:I.......  SP=00FFFF00
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFF00
```

未定義のメモリ間接を利用したＪＭＰ命令が発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

（ｂ）ＪＳＲ命令のメモリ間接

```
@@H'14:8 Addressing !!
PC=000000  CCR=80:I.......  SP=00FFFEFC
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFEFC
```

未定義のメモリ間接を利用したＪＳＲ命令が発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

（ｃ）ＴＲＡＰＡ

```
TRAPA #0 Occur !!
PC=000000  CCR=80:I.......  SP=00FFFEFC
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFEFC
```

未定義のＴＲＡＰＡ命令が発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

３．４　《 interrupt （内蔵周辺機能割込み管理） 》

（１）機能

組み込み型モニタはベクタ番号12以降の割込みに対しても、前ページまでの仮想ベクタ方式をサポートしています。以下に未定義割込み表示（仮想ベクタに定義していない割り込みが発生した場合）を示しておきます。

（２）解説

（ａ）ITU0 の IMIA 割込みが発生した場合

```
ITU0 IMIA Occur !!
PC=000000  CCR=80:I.......  SP=00FFFEFC
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFEFC
```

未定義の ITU0 IMIA 割込みが発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

（ｂ）A/D の ADI 割込みが発生した場合

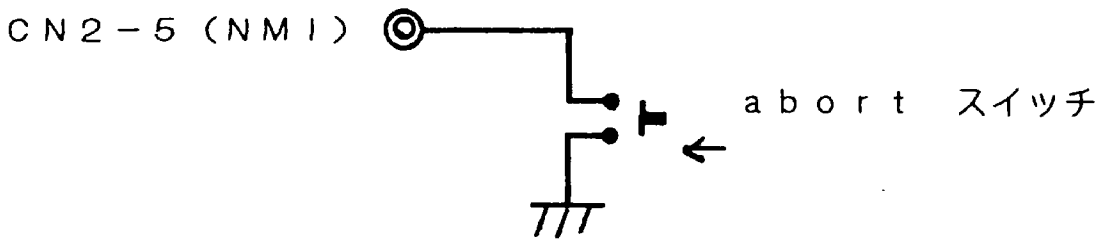
```
A/D ADI Occur !!
PC=000000  CCR=80:I.......  SP=00FFFEFC
ER0=00000000  ER1=00000000  ER2=00000000  ER3=00000000
ER4=00000000  ER5=00000000  ER6=00000000  ER7=00FFFEFC
```

未定義の A/D ADI 割込みが発生すると、上記のメッセージを表示してユーザプログラムの実行を停止します。

３．５ 《 abort （アボート管理） 》

（１）機能

組み込み型モニタはＮＭＩ割込みに対して、ユーザプログラムを強制停止させるアボート機能をサポートしています。
ＡＫＩ－Ｈ８キットでは、コネクタ２の５番ピンにＮＭＩピンが出ています。
ＮＭＩは抵抗でプルアップされていますので、ＧＮＤ間にスイッチを付けてください。
ユーザープログラム実行時にスイッチを押すと強制停止し、モニタープログラムにもどります。



（２）解説

（ａ）アボート機能

```
: G FF100 [RET]
NMI (Abort Switch ON) 入力
: Abort at PC=FFF14E
 PC=FFF14E  CCR=08:....N...  SP=000FFF10
 ER0=0000FFFF  ER1=00000000  ER2=00000000  ER3=00000000
 ER4=00000000  ER5=00000000  ER6=00000000  ER7=000FFF10
```

NMI入力を行うとユーザプログラムを強制停止します。