

## 付録. A KIT-NB85E-TP (H) 内部コマンド

本書は、KIT-NB85E-TP (H) の内部コマンドについて記述しています。これらのコマンドは、デバッガの中でスルーコマンドとして使用できます。スルーコマンドの使用方法は各デバッガのマニュアルを参照ください。

### PARTNER/Winの場合

>&	<<	スルーコマンドへの移行します。
>#ENV	<<	内部コマンドの入力です。
>&	<<	スルーコマンドモードを終了します。

### GHS-Multiの場合

RTESERVを接続後、ターゲット・ウインドウで直接入力できます。

### コマンド一覧

コマンド一覧.....	A-1
コマンド書式.....	A-2
アクセス系ブレークポイント : ABP, ABP1, ABP2コマンド.....	A-3
環境設定 : ENV, EMEMSTATコマンド.....	A-3
アクセス系イベントの設定 : EVAコマンド.....	A-7
実行系イベントの設定 : EVEコマンド.....	A-8
イベント統合の設定 : EVTコマンド.....	A-9
外部入力ブレーク : EXTBRKコマンド.....	A-10
ヘルプ : HELPコマンド.....	A-11
INPUT : INB, INH, INWコマンド.....	A-12
初期化 : INITコマンド.....	A-13
JTAGリード : JREADコマンド.....	A-14
デバッガキャッシュ領域の解除 : NCコマンド.....	A-15
デバッガキャッシュ領域の設定 : NCDコマンド.....	A-16
ソフトブレーク禁止領域の設定 : NSBPコマンド.....	A-17
ソフトブレーク禁止領域の解除 : NSBPDコマンド.....	A-18
強制ユーザ領域の設定 : NROMコマンド.....	A-19
強制ユーザ領域の解除 : NROMDコマンド.....	A-20
OUTPUT : OUTB, OUTH, OUTWコマンド.....	A-21
CPUリセット : RESETコマンド.....	A-22
E. ROMの設定 : ROMコマンド (RTE-1000-TP 用コマンド).....	A-23
E. ROMの設定 : ROM1..ROM4 コマンド (RTE-2000 (H)-TP 用コマンド).....	A-24
シーケンシャル条件の設定 : SEQコマンド.....	A-26
SFRアクセス : SFRコマンド.....	A-27
SFRの登録 : SFRFILEコマンド.....	B-28
シンボル : SYMFILE, SYMコマンド.....	A-29
トリガポイント : TPコマンド.....	A-30
トレーススイッチポイント : TSP1, TSP2コマンド.....	A-31
トレースデータ条件 : TD1, TD2コマンド.....	A-32
トレースの設定&開始 : TRONコマンド.....	A-33
トレースの強制終了 : TROFFコマンド.....	A-36
トレースの表示 : TRACEコマンド.....	A-37
トレースデータのディレイ調整 : TDATA_DLYコマンド.....	A-40
バージョン表示 : VERコマンド.....	A-41

ご注意：これらのコマンドは、ご使用になりたい機能がデバッガ本体に有していない場合にのみ補助的にご使用ください。ご使用になるデバッガで同等の機能を有している場合にこれらのコマンドを発行した場合、デバッガとの間で競合をおこし、いずれかの動作が異常になる場合があります。eva/eve/evt/seqの各コマンドは、Rte4win32 V5.10以上で対応されたコマンドです。

## コマンド書式

内部コマンドの基本書式を以下に示します。

コマンド名 パラメータ

\*パラメータ書式で [] は省略可能を示し、| は択一を意味します。

コマンド名はアルファベットの文字列で、パラメータとの間はスペースまたはタブで区切ります。パラメータはアルファベットの文字列または16進数を指定し、各パラメータ間はスペースまたはタブで区切ります。(16進数には演算子は使用できません。)

abp, abp1, abp2コマンド

## [書式]

```
abp [or|and|seq]
abp[1|2] [ADDR [AMASK]] [data DATA [DMASK]] [asid ASID|noasid] [aeq|aneq] [deq|dneq]
[exec|read|write|accs] [byte|hword|word|nosize]
abp[1|2] /del
```

## [パラメータ]

abp [or|and|seq]: abp1とabp2の組み合わせの条件を指定します。

- or: abp1 又は、abp2のどちらかの発生でブレークします。
- and: abp1とabp2が同時に発生した時にブレークします。マスク条件を使用します。
- seq: abp1発生後、abp2が発生した時にブレークします。

abp[1|2]: abp1または、abp2の条件指定に先立ち入力します。

ADDR [AMASK]: アドレス条件の指定

- ADDR: アドレスを16進数で指定します。
- AMASK: アドレスのマスクデータを16進数で指定します。1のビットは、比較の対象になりません。

data DATA [DMASK]: データ条件の指定

- DATA: データを16進数で指定します。
- DMASK: データのマスクデータを16進数で指定します。1のビットは、比較の対象になりません。

asid ASID|noasid: 将来の拡張用です。noasidでご使用ください。

aeq|aneq: アドレスの比較条件を指定します。

- aeq: アドレスをイコールで比較します。
- aneq: アドレスをノットイコールで比較します。

deq|dneq: データの比較条件を指定します。

- deq: データをイコールで比較します。
- dneq: データをノットイコールで比較します。

exec|read|write|accs: サイクルの条件を指定します。

- exec: 実行アドレスを指定します。データ条件は無視されます。
- read: リードサイクルを指定します。
- write: ライトサイクルを指定します。
- accs: リードまたはライトサイクルを指定します。

byte|hword|word|nosize: アクセスサイズの指定します。

- byte: バイトアクセス(8-bit)を指定します。
- hword: ハーフワードアクセス(16-bit)を指定します。
- word: ワードアクセス(32-bit)を指定します。
- nosize: 無効を指定します。

abp[1|2] /del: 条件の解除を行います。

- /del: 解除を指定します。

## [機能]

2点ある、アクセス系のブレークポイントの設定または解除をします。  
実行アドレスの指定もできます。

## [入力例]

```
abp or
    abp1 or abp2 を指定します。
abp1 1000 aeq exec
    1000h番地の実行にブレークを設定します。
abp2 1000 data 5555 0 aeq deq read hword
    1000h番地からhwordで5555hをリードした時にブレークします。
abp1 /del
    abp1の条件を解除します。
```

env, ememstatコマンド

## [書式]

```
env [[!]auto] [[!]verify] [[!]reset] [[!]stopz] [[!]hldrq]
  [jtag{xxx}]. [yyy] {M|K} [[!]nmi0] [[!]nmi1] [[!]nmi2]
  [rtrcb{0|25|50|75}] [nrtrcb{12|25|37|50}] [64m|256m]
  [romless|single0|single1] [d0|d1|d2|dauto] [i0|i1|i2|iauto]
```

## [パラメータ]

- [[!]**auto**: 実行中にブレークポイントを設定した場合一時的にブレークしますが、その後の実行を自動的に行う場合に[Auto], 行わない場合に[!auto]を指定します。
- [[!]**verify**: メモリへの書き込み時にリードアウトしてベリファイするかどうか指定します。!はベリファイしないを意味します。

備考: ROMをエミュレーションしている領域に対しても、CPUからアクセス(jread相当)しますので、ダウンロード時のテストにも有効です。但し、処理速度が遅くなります。

- [[!]**reset**: RESET端子のマスク指定を指定します。!はマスクしないを意味します。
- [[!]**stopz**: STOPZ端子のマスク指定を指定します。!はマスクしないを意味します。
- [[!]**hldrq**: VAREQ端子のマスク指定を指定します。!はマスクしないを意味します。
- [**jtag**{xxx}]. [**yyy**] {M|K}: JTAGクロックの周波数をMHz, またはKHzの単位で指定します。指定は10KHzから125MHzの間の任意の値が可能ですが、設定されるのは指定値以下の以下の値に丸められます。実際の設定値は表示で確認できます。
- RTE-2000-TP : [25MHz, 12.5MHz, 5MHz, 2MHz, 1MHz, 500KHz, 250KHz, 100KHz]
- RTE-2000H-TP: [125MHz, 100MHz, 80MHz, 60MHz, 50MHz, 40MHz, 30MHz, 25MHz, 12.5MHz, 5MHz, 2MHz, 1MHz, 500KHz, 250KHz, 100KHz, 50KHz, 25KHz, 10KHz]

注意: 通常は25MHzまたは、12.5MHzでご使用ください。1MHzより低い周波数を指定した場合、デバッグの動作が著しく遅くなったり、異常になる場合があります。初期値は25MHzを上限とした動作する最高周波数に自動的に設定します。初期値以上の値に設定する場合はCPUの許容範囲内で設定してください。CPUのスペック以上の周波数を設定した場合の動作は保証できません。

- [[!]**nmi**{0|1|2}: NMI0-2端子のマスク指定を指定します。!はマスクしないを意味します。
- rtrcb**{0|25|50|75}: リアルタイムトレース中のオーバフローからの復帰時のバッファの使用率を指定します。通常、初期値で使用してください。
- nrtrcb**{12|25|37|50}: 完全トレースモード中のパイプライン停止要求時にバッファの使用率を指定します。通常は、初期値で使用してください。
- 64m|256m**: CPUのアドレスモードを指定します。
- 64m: 64Mモードを指定します。
- 256m: 256Mモードを指定します。
- romless|single0|single1**: CPUの動作モードを指定します。
- single0m**: シングルモード0 (0番地から内蔵ROM) を指定します。
- single1**: シングルモード1 (100000h番地から内蔵ROM) を指定します。
- Romless**: ロムレスモードを指定します。
- d0|d1|d2|dauto**: データキャッシュの指定をします。
- d0**: データキャッシュなしを指定します。
- d1**: ダイレクトマップのキャッシュを指定します。
- d2**: 2-WAYのキャッシュを指定します。
- dauto**: NB85E-TEGの場合に指定します。自動設定します。
- i0|i1|i2|iauto**: 命令キャッシュを指定します。
- i0**: 命令キャッシュなしを指定します。
- i1**: ダイレクトマップのキャッシュを指定します。
- i2**: 2-WAYのキャッシュを指定します。
- iauto**: NB85E-TEGの場合に指定します。自動設定します。

備考: **dauto**, **iauto**の指定は、NB85E-TEGを使用した評価ボードに限ります。通常は、実際に実装されているキャッシュにあった指定をしてください。

## [機能]

envコマンドは、エミュレーション環境の設定とDCUの状態を表示します。  
 設定は変更が必要なパラメータだけを入力ください。入力の順序は任意です。  
 但し、同じパラメータを2回入力した場合は、後から入力した値が有効です。  
 ememstatコマンドはRTE-2000(H)-TPの場合に、E.MEM基板の実装状態を表示するコマンドです。  
 以下に表示例を示します。

RTE-1000-TPの場合

```
Probe:
Unit      : RTE-1000-TP  << 接続している本体を表示します。
Rom Probe : Extend Type << 接続しているROMプローブをタイプを表示します。
Emem Size : 32Mbyte    << 実装しているエミュレーションメモリの容量を表示します。
CPU Settings:
Auto Run   = ON (auto)
JTAGCLOCK = 12.5MHz (jtag12.5M)
Verify     = verify off (!verify)
CPU Mode   = romless (romless) << rte4win32の設定に依存します。
Space      = 64M Byte Mode (64m) << rte4win32の設定に依存します。
Signals Mask:
NMIO       = NO MASK (!nmi0)
NMI1       = NO MASK (!nmi1)
NMI2       = NO MASK (!nmi2)
RESET      = NO MASK (!reset)
HLDRQ      = NO MASK (!hldrq)
STOPZ      = NO MASK (!stopz)
Trace Buffer Usage Settings:
Realtime   <= 0% (rtrcb0)
None Realtime>= 12% (nrtrcb12)
Trace UNIT:
Cotrol Unit = Enable
Event Unit  = Enable
Execute Event Number = 8
Access Event Number = 4
Sequence Event Number = 1
Sequence Counter Bit = 12
Cache Mode:
Data       = 2Way (d2)
Instruction = Auto Detect (iauto)
```

RTE-2000(H)-TPの場合

```
Probe:
Unit      : RTE-2000(H)-TP          << 接続している本体を表示します。
Rom Probe : (use ememstat command)
Emem Size : (use ememstat command)
CPU Settings:
Auto Run   = ON (auto)
JTAGCLOCK = 25MHz (jtag25M)
Verify     = verify off (!verify)
CPU Mode   = single0 (single0) << rte4win32の設定に依存します。
Space      = 64M Byte Mode (64m) << rte4win32の設定に依存します
Signals Mask:
NMIO       = NO MASK (!nmi0)
NMI1       = NO MASK (!nmi1)
NMI2       = NO MASK (!nmi2)
RESET      = NO MASK (!reset)
HLDRQ      = NO MASK (!hldrq)
STOPZ      = NO MASK (!stopz)
Trace Buffer Usage Settings:
Realtime   <= 0% (rtrcb0)
None Realtime>= 12% (nrtrcb12)
Trace UNIT:
Cotrol Unit = Enable
Event Unit  = Enable
Execute Event Number = 8
Access Event Number = 4
Sequence Event Number = 1
Sequence Counter Bit = 12
```

```
Cache Mode:  
Data      = Auto Detect (dauto)  
Instruction = Auto Detect (iauto)
```

```
ememstat  
Board_num  EMEM_Size  ROM_Probe  
=====
```

ROM1	8Mbyte	Extend Type 2K
------	--------	----------------

[入力例]

```
env reset !nmi0 verify  
    RESETをマスクし、NMIOをマスクしません。Verify機能をONにします。  
env jtag40m  
    JTAGクロックを40MHzに設定します。
```

evaコマンド

## [書式]

```
eva {1..8} [ADDR] [data DATA [MASK]] [asid ASID|noasid] [eq|lt|gt|ne|lte|gte|ign]
[deq|dneq] [read|write|accs] [byte|hword|word|nosize] [/del]
```

## [パラメータ]

eva {1..8}: アクセス系イベントのチャンネル(Nx85ETで指定できるchは1-4のみ)を指定します。

ADDR: アドレスを16進数で指定します。

data DATA [MASK]: データ条件の指定

DATA: データを16進数で指定します。

MASK: データのマスクデータを16進数で指定します。1のビットは、比較の対象になりません。

asid ASID|noasid: 将来の拡張用です。noasidでご使用ください。

eq|lt|gt|neq|lte|gte|ign:

eq: ADDRで指定したアドレス = イベントアドレスで条件成立を指定します。

lt: ADDRで指定したアドレス > イベントアドレスで条件成立を指定します。

gt: ADDRで指定したアドレス < イベントアドレスで条件成立を指定します。

neq: ADDRで指定したアドレス != イベントアドレスで条件成立を指定します。

lte: ADDRで指定したアドレス => イベントアドレスで条件成立を指定します。

gte: ADDRで指定したアドレス =< イベントアドレスで条件成立を指定します。

ign: ADDRを比較条件として使用しない指定です。

deq|dneq: データの比較条件を指定します。

deq: データをイコールで比較します。

dneq: データをノットイコールで比較します。

read|write|accs: サイクルの条件を指定します。

read: リードサイクルを指定します。

write: ライトサイクルを指定します。

accs: リードまたはライトサイクルを指定します。

byte|hword|word|nosize: アクセスサイズの指定します。

byte: バイトアクセス(8-bit)を指定します。

hword: ハーフワードアクセス(16-bit)を指定します。

word: ワードアクセス(32-bit)を指定します。

nosize: 無効を指定します。

eva {1..8} /del: 条件の解除を行います。

/del: 解除を指定します。

## [機能]

アクセス系のイベントを設定します。指定したイベントは、EVTコマンドで統合して、ブレークやトレースの条件として使用できます。

## [使用例]

```
eva 1 ffff000 data 55 00 byte read
```

デフォルトからの指定で、0xffff000番地から0x55のリードサイクルをEVA ch1に設定します。

```
eva 1 /del
```

EVA ch1の条件を解除します。

e v eコマンド

## [書式]

```
eve {1..16} [ADDR] [asid ASID|noasid] [eq|lt|gt|neq|lte|gte|ign] [/del]
```

## [パラメータ]

eve {1..16}: 実行系イベントのチャンネル(Nx85ETで指定できるchは1-8のみ)を指定します。

ADDR: アドレスを16進数で指定します。

asid ASID|noasid: 将来の拡張用です。noasidでご使用ください。

eq|lt|gt|neq|lte|gte|ign:

eq: ADDRで指定したアドレス = イベントアドレスで条件成立を指定します。

lt: ADDRで指定したアドレス > イベントアドレスで条件成立を指定します。

gt: ADDRで指定したアドレス < イベントアドレスで条件成立を指定します。

neq: ADDRで指定したアドレス != イベントアドレスで条件成立を指定します。

lte: ADDRで指定したアドレス => イベントアドレスで条件成立を指定します。

gte: ADDRで指定したアドレス =< イベントアドレスで条件成立を指定します。

ign: ADDRを比較条件として使用しない指定です。

eve {1..16} /del: 条件の解除を行います。

/del: 解除を指定します。

## [機能]

実行系のイベントを設定します。指定したイベントは、EVTコマンドで統合して、ブレークやトレースの条件として使用できます。

## [使用例]

```
eve 1 1000
```

デフォルトからの指定で、0x1000番地の実行をEVE ch1に設定します。

```
eve 1 /del
```

EVE ch1の条件を解除します。



evt コマンド

## [書式]

```

evt {brk|seqclr|seq1|seq2|seq3|seq4|trcs1|trcs2|trcr|trg|match}
evep{[1][2][3]..[g]} ever{[1][3][5]..[f]} evap{[1][2][3]..[8]}
evar{[1][3][5][7]} [seq|!seq]

```

## [パラメータ]

```
brk|seqclr|seq1|seq2|seq3|seq4|trcs1|trcs2|trcr|trg|match:
```

イベントを統合する対象を指定します。

brk: ブレーク条件を指定します。

seqclr: シーケンシャル条件のクリア条件を指定します。

seq1: シーケンシャル条件の初段の条件を指定します。

seq2: シーケンシャル条件の2段目の条件を指定します。

seq3: シーケンシャル条件の3段目の条件を指定します。

seq4: シーケンシャル条件の4段目の条件を指定します。

trcs1: TSP1(トレーススイッチポイント1)の条件を指定します。

trcs2: TSP2(トレーススイッチポイント2)の条件を指定します。

trcr: 区間トレースの条件を指定します。

trg: EVTTRG信号へのトリガ出力の条件を指定します。

match: トレーストリガの条件を指定します。

evep{[1][2][3]..[g]}: eveコマンドで指定したイベントを単独でポイントとして指定します。  
数字をつけない場合、解除を意味します。(Nx85ETで指定できるchは1-8のみ)

eveで指定したチャンネル番号([1][2][3]..[16])と1対1で対応します。

ever{[1][3][5]..[f]}: eveコマンドで指定したイベントを複合してエリアとして指定します。

数字をつけない場合、解除を意味します。(Nx85ETで指定できるchは1, 3, 5, 7のみ)

1: eveで指定したチャンネル1と2の条件を範囲(and条件)として指定します。

3: eveで指定したチャンネル3と4の条件を範囲(and条件)として指定します。

5: eveで指定したチャンネル5と6の条件を範囲(and条件)として指定します。

7: eveで指定したチャンネル7と8の条件を範囲(and条件)として指定します。

evap{[1][2][3]..[8]}: evaコマンドで指定したイベントを単独でポイントとして指定します。

数字をつけない場合、解除を意味します。(Nx85ETで指定できるchは1-4のみ)

evaで指定したチャンネル番号([1][2][3]..[8])と1対1で対応します。

evar{[1][3][5][7]}: evaコマンドで指定したイベントを複合してエリアとして指定します。

数字をつけない場合、解除を意味します。(Nx85Eで指定できるchは1, 3のみ)

1: evaで指定したチャンネル1と2の条件を範囲(and条件)として指定します。

3: evaで指定したチャンネル3と4の条件を範囲(and条件)として指定します。

[!]seq: シーケンシャル条件を指定します。

seq: シーケンシャル条件を指定します。!でシーケンシャル条件を解除します。

seq関連(secclr, seq1, seq2, ...)の条件には、指定できません。

## [機能]

eve evaで指定したイベントを何に使うかを指定します。

## [使用例]

```
evt brk evep1234 ever5 evap12 evar3
```

ブレーク用のイベントとして、eveで指定した1から4をポイントして、5と6を範囲条件とし、

evaで指定した1と2をポイントとして、3, 4を範囲として使用します。

```
evt brk evep ever evap evar
```

ブレーク用のイベントとして指定した、evep ever evap evarを解除します。

## [備考]

シーケンシャル条件の詳細は、seqコマンドを参照ください。

トレースのセクションやクオリファイに関する詳細は、本編のトレースの章を参照ください。

extbrkコマンド

## [書式]

```
extbrk [disable|posi|nega]
```

## [パラメータ]

disable: 本機能を使用しない時に指定します。(初期値)  
posi: 立ち上がりエッジを検出してブレーク要求を出します。  
nega: 立ち下がりエッジを検出してブレーク要求を出します。

## [機能]

外部入力信号(EXTコネクタ:RSV-INO)から入力する信号を使って、実行をブレークする機能の指定を行う為のコマンドです。

## [使用例]

```
extbrk posi
```

立ち上がりエッジを検出してブレーク要求を出します。

備考: このコマンドは、RTE-100-TPでは無効です。  
この機能を使用する場合は、JTAG/N-WireコネクタのA12ピンがCPUのDBINTに接続されていなければなりません。

## h e l pコマンド

[書式]

help [command]

[パラメータ]

command: コマンド名を指定します。  
コマンド名を省略した場合、コマンドの一覧が表示されます。

[機能]

各コマンドのヘルプメッセージを表示します。

[使用例]

help map  
mapコマンドのヘルプを表示します。

inb, inh, inwコマンド

## [書式]

inb [ADDR]

inh [ADDR]

inw [ADDR]

## [パラメータ]

ADDR: 入力ポートのアドレスを16進数で指定します。

## [機能]

inb, inh, inwは、アクセスサイズを区別して、リードを行ないます。

inbはバイト、inhはハーフ・ワード、inwはワード単位でアクセスします。

## [使用例]

inb 1000

1000Hからバイト(8-bit)でリードします。

inh 1000

1000Hからハーフワード(16-bit)でリードします。

inw 1000

1000Hからワード(32-bit)でリードします。

## initコマンド

[書式]

init

[パラメータ]

なし

[機能]

- ICEの環境を起動時の状態に初期化します。  
以下を除き、全ての環境設定値は初期化されます。
- ・メモリキャッシュの除外エリア

j r e a dコマンド

## [書式]

```
jread [ADDR [LENGTH]]
```

## [パラメータ]

ADDR: アドレスを16進数で指定します。

LENGTH: 読み出すバイト数を16進数で指定します。(max 100h)

## [機能]

ROMコマンドで割り付けたROMエミュレーション領域をJTAG(CPU)から読み出すためのコマンドです。

(通常のコマンドでは、ROMエミュレーション領域へのアクセスは内部のメモリに対し直接行っています。)

## [使用例]

```
jread 100000 100
```

100000hから100hバイトをJTAG経由で読み出します。

ncコマンド

## [書式]

```
nc [[ADDR [LENGTH]]]
```

## [パラメータ]

ADDR: メモリキャッシュの除外エリアの開始アドレスを指定します。  
LENGTH: メモリキャッシュの除外エリアのバイト数を指定します。  
デフォルト値 3 2 バイト、最少値 3 2 バイト

## [機能]

メモリ参照の高速化を図るため、ファームウェア内に 8 ブロック \* 3 2 バイトのメモリリードキャッシュを持っています。同一アドレスのメモリ参照などは実際にはメモリをリードしません。I/Oを割り付けている空間では、このキャッシュ機能は実際の動作と矛盾しますので、このコマンドで除外エリアとして指定してください。メモリキャッシュの除外エリアは最大 8 ブロック指定でき、最少のブロックサイズは 3 2 バイトです。尚、ffff000h - ffffffffh と 3fff000h - 3fffffffhは、内蔵のsfr領域ですので、初期値として除外エリアに指定されています。

## [使用例]

```
nc 10000 100
```

10000h番地から100バイトの領域をメモリキャッシュの除外エリアに指定します。

```
>nc 100000 100
No Memory Cache Area
No. Address Length
1 00100000 00000100
2 0ffff000 00001000
3 03fff000 00001000
```

n c dコマンド

## [書式]

ncd ブロック番号

## [パラメータ]

ブロック番号: 削除するメモリアドレスの除外エリアのブロック番号を指定します。

## [機能]

メモリアドレスの除外エリアを削除します。削除は各メモリアドレスの除外エリアのブロック番号を指定します。

## [使用例]

ncd 1

ブロック番号 1 をメモリアドレスの除外エリアから削除します。

```
>nc 100000 100
No Memory Cache Area
No. Address Length
1 00100000 00000100
2 0ffff000 00001000
3 03fff000 00001000
```

```
>ncd 1
No Memory Cache Area
No. Address Length
1 0ffff000 00001000
2 03fff000 00001000
```



nsbpコマンド

## [書式]

```
nsbp [[ADDR [LENGTH]]]
```

## [パラメータ]

ADDR: ソフトウェアブ레이크禁止領域の開始アドレスを指定します。  
LENGTH: ソフトウェアブ레이크禁止領域のバイト数を指定します。  
指定領域の最小単位はハーフワードバウンダリです。  
また、指定できる領域の数は最大4ヶ所です。

## [機能]

ソフトウェアブ레이크を禁止したい領域を指定します。  
ブ레이크ポイントを指定した場合、デバッガは暗黙的に対象アドレスに対し、メモリテスト（ライトアクセス）を行います。  
一部のフラッシュROM等、ライトアクセスを行うことでメモリの状態が変わり、正しいデータの読み出しが行えなくなる場合等に、ライトサイクルを禁止する目的で指定してください。  
通常は、指定する必要はありません。

## [使用例]

```
nsbp 10000 20000  
10000h番地から20000バイトの領域をソフトウェアブ레이크禁止領域に指定します。
```

```
>nsbp 100000 20000  
Num Address Length  
01 00100000 00020000
```

nsbpdコマンド

## [書式]

nsbpd [ブロック番号|/all]

## [パラメータ]

ブロック番号: 削除するソフトウェアブレイク禁止領域のブロック番号を指定します。

/all: 全てのソフトウェアブレイク禁止領域を削除します。

## [機能]

nsbpで指定したソフトウェアブレイク禁止領域を削除します。

## [使用例]

nsbpd 1

ブロック番号 1 をソフトウェアブレイク禁止領域から削除します。

```
nsbp
Num Address Length
01 00100000 00200000
02 00400000 00010000
```

```
>nsbpd 1
Num Address Length
01 00400000 00010000
```

n r o mコマンド

## [書式]

```
nrom [[ADDR [LENGTH]]]
```

## [パラメータ]

ADDR: 強制ユーザ領域の開始アドレスを指定します。  
 LENGTH: 強制ユーザ領域のバイト数を指定します。  
 指定領域の最小単位は、以下の通りです。  
 RTE-1000-TP : 4-byte単位。  
 RTE-2000(H)-TP : エミュレーションしているROMのサイズに応じます。  
     8/16-bit : 128k-byte単位  
     32-bit : 256k-byte単位  
     (64-bit : 512k-byte単位)  
 また、指定できる領域の数は最大4ヶ所です。

## [機能]

ROMコマンドで指定したROMエミュレーション領域内の一部がユーザシステム上の資源にマップされていた場合にその領域を指定します。通常は指定する必要はありません。  
 指定領域に対する動作は以下の通りです。

- ・ デバッガからのアクセスは強制的にユーザシステムに対し行われるようになります。
- ・ 実行中この領域へのアクセスサイクルでEMEMEN-信号はインアクティブ(Highレベル)になります。(RTE-2000(H)-TPのみ)

## [使用例]

```
nrom 0 20000
```

0h番地から20000バイトを強制ユーザ領域に指定します。

```
>nrom 0 20000
```

No.	Address	Length
1	00000000	00020000

```
>nrom 100000 40000
```

No.	Address	Length
1	00000000	00020000
2	00100000	00040000

n r o m dコマンド

## [書式]

nromd [ブロック番号|/all]

## [パラメータ]

ブロック番号: 削除する強制ユーザ領域のブロック番号を指定します。  
/all: 全ての強制ユーザ領域のブロックを削除します。

## [機能]

nromで指定した強制ユーザ領域を削除します。

## [使用例]

ncd 1

ブロック番号 1 を強制ユーザ領域から削除します。

```
>nrom 100000 40000
```

No.	Address	Length
1	00000000	00020000
2	00100000	00040000

```
>nromd 1
```

No.	Address	Length
1	00100000	00040000

outb, outh, outwコマンド

## [書式]

```
outb [[ADDR] DATA]
outh [[ADDR] DATA]
outw [[ADDR] DATA]
```

## [パラメータ]

ADDR: 出力ポートのアドレスを16進数で指定します。  
DATA: 出力するデータを16進数で指定します。

## [機能]

outb, outh, outwは、アクセスサイズを区別して、ライトを行ないます。  
outbはバイト、outhはハーフ・ワード、outwはワード単位でアクセスします。

## [使用例]

```
outb 1000 12
    1000Hへバイトデータ : 12hを1ライトします。
outh 1000 1234
    1000Hへハーフワードデータ : 1234hをライトします。
outh 1000 12345678
    1000Hへワードデータ : 12345678hをライトします。
```

resetコマンド

[書式]

reset

[パラメータ]

なし

[機能]

エミュレーションCPUをリセットします。

romコマンド(RTE-1000-TP用コマンド)

## [書式]

```
rom [ADDRESS [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
[bus8|bus16|bus32]
```

## [パラメータ]

ADDR [LENGTH]: エミュレーションする領域を指定します。

ADDR: 開始アドレスを指定します。エミュレートするROMの最下位のアドレス (ROMのバウンダリ)に合致していない場合、エラーになります。

LENGTH: エミュレートするROMのバイト数を指定します。(4バイトの境界で指定)

512k|1m|2m|4m|8m|16m|32m|64m|128m|256m:

1本のROMプローブでエミュレートするROMのBit容量を指定します。512K-bitから256M-bitまでの値が指定できます。

例えば、27C1024の場合は、1Mを指定します。

rom8|rom16: エミュレートするROMのデータビット数を指定します。

8bitと16bitが指定できます。DIP32-ROMケーブルを使用する場合はrom8、DIP-40/42-ROMケーブル、16bit-標準ROMケーブルを使用する場合は、rom16を指定します。

bus8|bus16|bus32: エミュレートするシステムの中でのROMのバスサイズを指定します。

8bit, 16bit, 32bitが指定できます。

## [機能]

RTE-1000-TPのROMのエミュレーション環境の設定を行います。設定はADDRとLENGTHをペアで入力する以外は変更が必要なパラメータだけ入力できます。入力の順序は任意です。但し、同じパラメータを2回入力した場合は、後から入力した値が有効です。初期値は、LENGTH=0 (使用しない) になっています。

## [入力例]

```
rom 100000 40000 1m rom16 bus16
```

27C1024 (1M-bitの16bit-ROM) を100000hから256Kバイト (40000h) エミュレートします。

この場合、結果的に16bit-romを2個使用してエミュレートします。

```
rom 0 40000 2m rom rom16 bus32
```

27c2048 (2M-bitの16bit-ROM) を0x0から256Kバイト (40000h) エミュレートします。

この場合、結果的に16bit-ROMを1個使用してエミュレートします。

## &lt;備考&gt;

romコマンドで指定した領域における注意事項

romコマンドで指定した範囲へのデバッグからのアクセスは、ツール内部のエミュレーションメモリに対し直接アクセスしています。その結果、プロセッサから正しくROMにアクセスできない状態においても表示は正しく行われますので、デバッグ初期の段階ではjreadコマンド (CPUのバス経由で読み出すコマンド) を使用して読み出し確認するか、envコマンドでverifyをONにして書き込み (ダウンロード) を行うことをお勧めします。

rom1..rom4コマンド(RTE-2000(H)-TP用コマンド)

## [書式]

```
rom1 [ADDRESS [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16|bus32|bus64] [[!]wren]
rom2 [ADDRESS [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16] [[!]wren]
rom3 [ADDRESS [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16|bus32] [[!]wren]
rom4 [ADDRESS [LENGTH]] [512k|1m|2m|4m|8m|16m|32m|64m|128m|256m] [rom8|rom16]
      [bus8|bus16] [[!]wren]
```

rom1: スロット#3に実装されたEMEM基板を含むモジュールに対する設定コマンドです。  
rom2: スロット#4に実装されたEMEM基板を含むモジュールに対する設定コマンドです。  
rom3: スロット#5に実装されたEMEM基板を含むモジュールに対する設定コマンドです。  
rom4: スロット#6に実装されたEMEM基板を含むモジュールに対する設定コマンドです。

## [パラメータ]

ADDR [LENGTH]: エミュレーションする領域を指定します。  
ADDR: 開始アドレスを指定します。  
エミュレートするROMの最下位のアドレス (ROMのバウンダリ) に合致していない場合、指定アドレス以下のアドレス領域は非エミュレーション領域になります。  
LENGTH: エミュレートするROMのバイト数を指定します。

備考: ADDR, LENGTHで指定できる領域の最小単位は、エミュレーションしているROMのサイズに応じ、以下の通りです。

- ・ 8/16-bit : 128k-byte単位
- ・ 32-bit : 256k-byte単位
- ・ 64-bit : 512k-byte単位

512k|1m|2m|4m|8m|16m|32m|64m|128m|256m:

1本のROMプローブでエミュレートするROMのBit容量を指定します。512K-bitから256M-bit(32M-Byte)までの値が指定できます。  
例えば、27C1024の場合は、1mを指定します。

rom8|rom16: エミュレートするROMのデータビット数を指定します。  
8bitと16bitが指定できます。DIP32のアダプタを使用する場合はrom8、DIP-40/42のアダプタ、及び16bit-標準ROMケーブルをそのまま使用する場合は、rom16を指定します。

bus8|bus16|bus32|bus64: エミュレートするシステムの中でのROMのバスサイズを指定します。  
8bit, 16bit, 32bit, 64bitが指定できます。

>> [64-bit]は将来のためのパラメータです。(本KITでは使用しません)

[[!]wren]: Write Enable:エミュレーションメモリをRAMとして使用する場合の設定です。  
wrenで書込み許可、!wrenで書込み禁止です。初期値は!wrenです。

## [機能]

ROMエミュレーション環境の設定を行います。設定はADDRとLENGTHをペアで入力する以外は必要なパラメータだけ入力できます。入力の順序は任意です。但し、同じパラメータを2回入力した場合は、後から入力した値が有効です。初期値は、LENGTH = 0 (使用しない) になっています。



[入力例]

>rom1 100000 300000 32m rom16 bus16 !wren

対象EMEM基板 スロット位置	アドレス範囲	バス幅	ROM		ライトイェ-ブル
			バス幅	Bit数	
#3	100000 - 3ffffff	16-bit	16-bit	32M-Bit	禁止

>rom2 140000 40000 2m rom16 bus16 wren

対象EMEM基板 スロット位置	アドレス範囲	バス幅	ROM		ライトイェ-ブル
			バス幅	Bit数	
#4	140000 - 17ffff	16-bit	16-bit	2M-Bit	許可

>rom1 0 80000 2m rom rom16 bus32 !wren

対象EMEM基板 スロット位置	アドレス範囲	バス幅	ROM		ライトイェ-ブル
			バス幅	Bit数	
#3+#4	000000 - 07ffff	32-bit	16-bit	2M-Bit	禁止

この時、rom2コマンドは発行しないでください。

<備考>

romコマンドで指定した領域における注意事項

rom1..rom4コマンドで指定した範囲へのデバッグからのアクセスは、ツール内部のエミュレーションメモリに対し直接アクセスしています。その結果、プロセッサから正しくROMにアクセスできない状態においても表示は正しく行われますので、デバッグ初期の段階ではjreadコマンド(CPUのバス経由で読み出すコマンド)を使用して読み出し確認するか、envコマンドでverifyをONにして書き込み(ダウンロード)を行うことをお勧めします。

romコマンドとEMEM基板の関係

romコマンド	バス幅	対象EMEM基板の スロット位置	使用できないromコマンド
rom1	8-bit	#3	
	16-bit	#3	
	32-bit	#3+#4	rom2
	64-bit	#3+#4+#5+#6	rom2, rom3, rom4
rom2	8-bit	#4	
	16-bit	#4	
rom3	8-bit	#5	
	16-bit	#5	
	32-bit	#5+#6	rom4
rom4	8-bit	#6	
	16-bit	#6	

seqコマンド

## [書式]

```
seq [PASS] [step{1|2|3|4}]
```

## [パラメータ]

PASS: シーケンス条件の成立回数を10進数で指定します。

step{1|2|3|4}: シーケンスの段数を指定します。

step1: seq4→pass\_count\_decrement

step2: seq3→seq4→pass\_count\_decrement

step3: seq2→seq3→seq4→pass\_count\_decrement

step4: seq1→seq2→seq3→seq4→pass\_count\_decrement

## [機能]

シーケンシャル条件の設定をします。

seq1～seq4の条件は、eve, eva, evtで指定します。

シーケンス途中でseqclr条件が成立した場合、そのシーケンスは最初に戻ります。

## [使用例]

```
seq 100 step1
```

seq1→seq2→seq3→seq4の条件が100回成立した時にseqイベントが発生します。

s f r コマンド

## [書式]

sfr [reg [VAL]]

## [パラメータ]

VAL: SFRのレジスタ値を16進数で指定します。

reg: SFRレジスタ名を指定します。

レジスタとして使用できる名称は以下の通りです。

リード・ライトレジスタ:

CSC0 CSC1 BPC BSC BEC BHC VSWC  
 DSA0L DSA0H DDA0L DDA0H DSA1L DSA1H DDA1L DDA1H  
 DSA2L DSA2H DDA2L DDA2H DSA3L DSA3H DDA3L DDA3H  
 DBC0 DBC1 DBC2 DBC3  
 DADC0 DADC1 DADC2 DADC3 DCHC0 DCHC1 DCHC2 DCHC3  
 DRST IMR0 IMR1 IMR2 IMR3  
 PIC0..PIC63  
 PSC BCT0 BCT1 DWCO DWC1 BCC ASC PRC RWC  
 DRC0 SCR0 RFC0 RFS0 DRC1 SCR1 RFC1 RFS1  
 DRC2 SCR2 RFC2 RFS2 DRC3 SCR3 RFC3 RFS3  
 DRC4 SCR4 RFC4 RFS4 DRC5 SCR5 RFC5 RFS5  
 DRC6 SCR6 RFC6 RFS6 DRC7 SCR7 RFC7 RFS7  
 ICG IC1 ICD

ライトオンリーレジスタ:

PRCMD

リードオンリーレジスタ:

DDIS ISPR

## [機能]

SFRレジスタ値の設定と表示を行います。

## [使用例]

sfr PIC0

PIC0レジスタの値を表示します。

sfr PIC0 2

PIC0レジスタに2hを設定します。

s f r f i l eコマンド

## [書式]

```
sfrfile {/|FILENAME}
```

## [パラメータ]

/: sfrの登録を初期値に戻します。  
FILENAME: NEC社が提供するデバイスファイル名を指定します。

## [機能]

sfrfile コマンドは、FILENAMEで指定したデバイスファイルからSFR情報を抽出し登録します。  
抽出したSFR情報はSFRコマンドで使用できるようになります。  
抽出前に登録されていた情報は削除されます。

## [使用例]

```
sfrfile c:¥test¥d800299.800  
c:¥test¥のディレクトリからデバイスファイル: d800299.800を読み込みます。
```

symfile, symコマンド

## [書式]

symfile FILENAME

sym [NAME]

## [パラメータ]

symfile: ファイル名を指定します。

sym: シンボルの先頭文字列を指定します。

## [機能]

symfile コマンドは、FILENAMEで指定したelfファイルからシンボルを読み込みます。  
対象となるのはグローバルシンボルだけです。

Symコマンドは、読み込んだシンボルの表示（最大30個）をします。

## [使用例]

symfile c:%test%dry%dry.elf

c:%test%dryのディレクトリからelfファイル:dry.elfのシンボルを読み込みます。

sym m

mから始まるシンボルを最大30個表示します。

tpコマンド

## [書式]

tp [ADDR]

## [パラメータ]

ADDR: 偶数アドレスを16進数で指定します。(A0は、常に0に補正されます)

## [機能]

トレースのトリガポイントを指定します。

トレースは、トリガポイントを基点にしてその前後の実行状態を取り込むことができます。

(トリガポイントの使用方法はtronコマンドを参照ください)

## [使用例]

tp 100000

100000hの命令実行をトリガポイントとして指定します。

## [注意事項]

tronコマンドでdelay modeが指定されている場合、トリガポイントの指定は無視されます。

この場合、tron !delayと入力してdelay modeを解除してください。

t s p 1 , t s p 2 コマンド

## [書式]

tsp{1|2} [ADDR] [asid ASID|noasid] [/del]

## [パラメータ]

tsp{1|2}: tsp1または、tsp2の条件指定に先立ち入力します。  
ADDR: 実行アドレスを16進数で指定します。  
asid ASID|noasid: 将来の拡張用です。noasidでご使用ください。  
/del: 指定したアドレスを解除します。

## [機能]

2点あるトレースのスイッチ・ポイント（アドレス）を指定します。  
指定したスイッチ・ポイントにより、トレース情報の取り込み条件をかえることができます。  
（取り込み条件の指定方法は、tronコマンドを参照ください）

## [使用例]

tsp1 100000  
100000hの命令実行をトレースのスイッチ・ポイントとして指定します。

## [備考]

このコマンドで指定したスイッチ・ポイントは、tronコマンドを発行した時点で有効になります。

td1, td2コマンド

## [書式]

```
td{1|2} [ADDR [MASK]] [asid ASID|noasid] [/del]
```

## [パラメータ]

td{1|2}: td1または、td2の条件指定に先立ち入力します。

ADDR: アドレスを指定します。

MASK: アドレスのマスクデータを16進数で指定します。1のビットは、比較の対象になりません。  
有効なのは、bit9-bit2のみです。

asid ASID|noasid: 将来の拡張用です。noasidでご使用ください。

/del: 指定したアドレスを解除します。

## [機能]

トレースに取り込むデータアクセスサイクルの条件を設定します。トレースは、実行履歴情報と、ここで指定したアドレスへのアクセスサイクルを取り込みます。

## [使用例]

```
td1 100000 ff
```

1000xxh番地のアクセスサイクルをトレースに取り込みます。



tronコマンド

## [書式]

```
tron [DELAY] [[!]delay] [[!]real] [[!]force] {[[!]evttcrs1] [[!]evttcrs2] |
  [[!]evttcr]} [tr1_{[0]..h}|tr1_all] [tr2_{[0]..h}|tr2_all]
  [[!]clock2] [[!]stop] [noext|posi|nega] [[!]td1] [[!]td2] [[!]debug]
```

## [パラメータ]

DELAY = 0..*xxxx*: デレイカウンタ <注意> (RTE-1000-TPでは、0..*1ffffd*です。)

トリガ成立後にメモリに取り込むフレーム数を16進数で指定します。

[[!]*delay*: 強制デレイモードを指定します。!で通常のモードの指定に戻ります。

強制デレイモードでは、トレース開始後、デレイカウンタ数分のトレースをした時点で強制的にトレースを終了するモードです。このモード中は、トリガイベントは無視されます。

[[!]*real*: トレース中の実行モードを指定します。*real*でリアルタイム実行モードです。

リアルタイム実行モードでは、トレース情報がオーバーフローする場合があります。

!で非リアルタイム実行モードになります。このモードでは、オーバーフローは発生しませんが、実行速度が低下します。

[[!]*force*: トレースの強制開始を指定します。!で強制開始を解除です。その場合は、*tsp1*の条件によります。

[[!]*evttcrs1*] [[!]*evttcrs2*] | [[!]*evttcr*]: TSP1のイベント条件を指定します。

*evttcrs1*と*evttcrs2*のペアと*evttcr*は排他的な設定となります。

[[!]*evttcrs1*: *trcs1*をTSP1の条件として使用します。!で使用しません。

[[!]*evttcrs2*: *trcs2*をTSP2の条件として使用します。!で使用しません。

[[!]*evttcr*: *trcr*の条件成立中をTSP1、非成立中をTSP2の条件として使用します。

!で使用しません。

*tr1*\_{[0]..*h*}|*tr1\_all*: *tsp1*のスイッチポイント以降に取り込むトレース情報を指定します。

通常は*tr1\_all*を指定することでトレースの取り込みを開始する目的で使用します。

*tr1*\_{[0]..*h*}: 0:Interrupt, 1:Exception, 2:RET1, 3:JMP, 4:JR, 5:JARL,  
6:Condition Jump(not taken), 7:Condition Jump(taken),  
8:CALLT, 9:SWITCH, a:DISPOSE, b:CTRET,  
c:td1 read cycle, d:td1 write cycle,  
e:td2 read cycle, f:td2 write cycle,  
g:tp, h:evt\_match

*tr1\_all*: 全てのトレース情報を取り込みます。

*tr2*\_{[0]..*h*}|*tr1\_all*: *tsp2*のスイッチポイント以降に取り込むトレース情報を指定します。

通常は何も指定しないことでトレースの取り込みを一時的に停止する目的で使用します。

*tr2*\_{[0]..*h*}: 0:Interrupt, 1:Exception, 2:RET1, 3:JMP, 4:JR, 5:JARL,  
6:Condition Jump(not taken), 7:Condition Jump(taken),  
8:CALLT, 9:SWITCH, a:DISPOSE, b:CTRET,  
c:td1 read cycle, d:td1 write cycle,  
e:td2 read cycle, f:td2 write cycle,  
g:tp, h:evt\_match

*tr2\_all*: 全てのトレース情報を取り込みます。

[[!]*clock2*: トレースのサンプリングクロックを指定します。*clock2*でVBCLKの1/2を指定します。!で1/1になります。通常!*clock2*でご使用ください。

[[!]*stop*: *stop*モード中のトレース出力を指定します。*stop*で*stop*モード中のトレースを停止します。!で停止しないの指定になります。

*noext*|*nega*|*posi*: トリガとして外部入力端子(EX10)を指定します。

*noext*: EX10をトリガとして使用しません。

*posi*: EX10の立ち上がりエッジをトリガとして指定します。

*nega*: EX10の立ち下がりがエッジをトリガとして指定します。

[[!]*td1*: トレースデータ条件1(*td1*)をトリガとして指定します。!で解除します。

[[!]*td2*: トレースデータ条件2(*td2*)をトリガとして指定します。!で解除します。

備考: [[!]*td1*] [[!]*td2*]は、RTE-100-TPでは、無効です。

*td1*と*td2*の条件が重複するサイクルを指定している場合、トリガの条件は、*td1*を指定してください。*td2*では、トリガがかからない場合があります。

[[!]*debug*: 常時初期値(!*debug*)でご使用ください。

## [機能]

トレースの諸設定とトレースバッファをクリアし、トレースの取り込みを開始します。

## [使用例]

delayモードでtron直後より、無条件に3ffffdサイクル分トレースします。

```
>tron delay 3ffffd                <<トレース開始
Trace Settings:
Delay Count   = 0003ffffd
Trace Mode    = Real Time (real)
Start Mode    = Force Start (force)
Delay Mode    = Enable (delay)
Event trcs1   = Disable (!evttrcs1)
Event trcs2   = Disable (!evttrcs2)
Event trcr    = -----
Sampling cond1= tr1_0123456789abcdefgh
Sampling cond2= tr2_0123456789abcdefgh
Trace Clock   = VBCLK (!clock2)
STOP Mode     = Disable (!stop)
Ext Trigger   = Disable (noext)
TD1 Trigger   = Disable (!td1)
TD2 Trigger   = Disable (!td2)
Debug Mode    = Disable (!debug)
```

100000h番地の命令実行をトリガにして、トリガ後の取り込を1ffffサイクルでトレースを行います。

```
>tp 100000                        <<トリガの指定
Trigger Point Settings:
Address  AISD
tp 00100000 noasid

>tron !delay 1ffff                <<トレース開始
Trace Settings:
Delay Count   = 0001ffff
Trace Mode    = Real Time (real)
Start Mode    = Force Start (force)
Delay Mode    = Disable (!delay)
Event trcs1   = Disable (!evttrcs1)
Event trcs2   = Disable (!evttrcs2)
Event trcr    = -----
Sampling cond1= tr1_0123456789abcdefgh
Sampling cond2= tr2_0123456789abcdefgh
Trace Clock   = VBCLK (!clock2)
STOP Mode     = Disable (!stop)
Ext Trigger   = Disable (noext)
TD1 Trigger   = Disable (!td1)
TD2 Trigger   = Disable (!td2)
Debug Mode    = Disable (!debug)
```

tsp1をトレース開始条件、tsp2をトレース停止条件にして、100000h番地の実行から100100h番地を実行するまでの間の実行履歴をトレースします。

```

>tsp1 100000                                <<開始条件に使用するポイントの設定
Trace Switch Point Settings:
  Address  AISD
tsp1 00100000 noasid
tsp2 /del

>tsp2 100100                                <<停止条件に使用するポイントの設定
Trace Switch Point Settings:
  Address  AISD
tsp1 00100000 noasid
tsp2 00100100 noasid

>tron !force tr1_all tr2_                   <<tsp1でall、tsp2でnoneを指定
Trace Settings:
Delay Count   = 0000ffff
Trace Mode    = Real Time (real)
Start Mode    = Start by tsp1 or evttrcs1 or evttrcr (!force)
Delay Mode    = Disable (!delay)
Event trcs1   = Disable (!evttrcs1)
Event trcs2   = Disable (!evttrcs2)
Event trcr    = -----
Sampling cond1= tr1_0123456789abcdefgh
Sampling cond2= tr2_
Trace Clock   = VBCLK (!clock2)
STOP Mode     = Disable (!stop)
Ext Trigger   = Disable (noext)
TD1 Trigger   = Disable (!td1)
TD2 Trigger   = Disable (!td2)
Debug Mode    = Disable (!debug)

```

t r o f f コマンド

[書式]

troff

[パラメータ]

なし

[機能]

トレースの取り込みを強制的に終了します。

traceコマンド

## [書式]

```
trace [POS] [all|pc|data] [asm] [asm|ttag1|ttag2] [subNN]
```

## [パラメータ]

POS=±0..xxxx: トリガサイクル近辺または終了サイクルを0として、トレースの表示開始位置を16進数で指定します。<注意>(RTE-1000-TPでは、0..1ffffdです)

all|pc|data: 取り込んだトレース情報の中から選択して表示するサイクルの指定します。

all: 全てのサイクル

pc: 実行サイクルのみ

data: データサイクルのみ

asm|ttag1|ttag2: 表示種別を指定します。

asm: アセンブラ表示のみ

ttag1: アセンブラ表示+絶対時間でのタイムタグ表示

ttag2: アセンブラ表示+相対時間でのタイムタグ表示

備考: ttag1|ttag2の指定は、RTE-100-TPでは無効です。

subNN: 実際に取り込まれる一つの分岐情報から連続して逆アセンブルする命令数を16進数で指定します。初期値は80h(ex:sub80)です。

## [機能]

トレースバッファの内容を表示します。

トレース中にこのコマンドを発行した場合、強制的に取り込みを終了します。

## [表示内容]

```
>trace asm -15
```

Cycle	Sub	Address	Code	Instruction	EXT	Stat
-00001e	----	00:0010558e	ffbf7da	jarl 00100d68h	1111	JMPS JARL
-000014	----	00:00100d68	3f460000	st. b r7, +00h[r6]	1111	JMPD JARL
* 000000	----	--:00100d6c	007f	jmp [lp]	1111	MATCH
000002	----	00:00105592	664003d0	movehi 03d0h, zero, r12	1111	JMPD JMP
000002	0001	00:00105596	672ca4b2	ld. h -05b4eh[r12], r12	1111	SUB
000002	0002	00:0010559a	6ecc0010	andi 0010h, r12, r13	1111	SUB
000002	0003	00:0010559e	69e0	cmp zero, r13	1111	SUB
00000c	----	00:001055a0	1d92	be 001055d2h	1111	JMPS BcondNT
000016	----	00:001055a2	16400380	movehi 0380h, zero, r2	1111	JMPD BcondNT

```
>trace -15 ttag1
```

Cycle	Sub	Address	Code	Instruction	EXT	Stat
-00001e	----	00:0010558e	ffbf7da	jarl 00100d68h	1111	JMPS JARL
				time = 000, 001, 448, 264. 9uS		
-000014	----	00:00100d68	3f460000	st. b r7, +00h[r6]	1111	JMPD JARL
				time = 000, 001, 448, 265. 3uS		
* 000000	----	--:00100d6c	007f	jmp [lp]	1111	MATCH
				time = 000, 001, 448, 265. 7uS		
000002	----	00:00105592	664003d0	movehi 03d0h, zero, r12	1111	JMPD JMP
				time = 000, 001, 448, 267. 4uS		
000002	0001	00:00105596	672ca4b2	ld. h -05b4eh[r12], r12	1111	SUB
000002	0002	00:0010559a	6ecc0010	andi 0010h, r12, r13	1111	SUB
000002	0003	00:0010559e	69e0	cmp zero, r13	1111	SUB
00000c	----	00:001055a0	1d92	be 001055d2h	1111	JMPS BcondNT
				time = 000, 001, 448, 268. 5uS		
000016	----	00:001055a2	16400380	movehi 0380h, zero, r2	1111	JMPD BcondNT

time = 000,001,448,268.9uS

>trace -15 ttag2

Cycle	Sub	Address	Code	Instruction	EXT	Stat
-00001e	----	00:0010558e	ffbf7da	jarl 00100d68h	1111	JMPS JARL
				time = 000,000,000,002.6uS		
-000014	----	00:00100d68	3f460000	st.b r7,+00h[r6]	1111	JMPD JARL
				time = 000,000,000,000.4uS		
* 000000	----	--:00100d6c	007f	jmp [lp]	1111	MATCH
				time = 000,000,000,000.2uS		
000002	----	00:00105592	664003d0	movehi 03d0h,zero,r12	1111	JMPD JMP
				time = 000,000,000,001.7uS		
000002	0001	00:00105596	672ca4b2	ld.h -05b4eh[r12],r12	1111	SUB
000002	0002	00:0010559a	6ecc0010	andi 0010h,r12,r13	1111	SUB
000002	0003	00:0010559e	69e0	cmp zero,r13	1111	SUB
00000c	----	00:001055a0	1d92	be 001055d2h	1111	JMPS BcondNT
				time = 000,000,000,001.1uS		
000016	----	00:001055a2	16400380	movehi 0380h,zero,r2	1111	JMPD BcondNT
				time = 000,000,000,000.4uS		

Cycle:        トレースバッファ内の位置を16進数で相対的に表示しています。トリガポイント位置の近辺または、トレースの最終フレームを0としています。

Sub:            分岐や実行命令数などの情報から解析して生成したサイクルの番号です。

Address:        実行アドレスまたは、バスサイクルのアドレスを表示します。

Code:            命令コードまたは、バスサイクルのデータを表示します。

Instruction:    命令のニーモニックまたは、バスの種類を表示します。

EXT:            外部入力端子EX13.0の状態をビット列で表示します。

Stat:            表示にもとになるトレースパケットの種別を表示します。

TRGSTART0	STARTパケット発生、サブスイッチがONになった
TRGSTART1	STARTパケット発生、サブスイッチがOFFになった
MATCH	MATCHパケット発生
OVF	オーバーフロー発生
TRCEND	TRCENDパケット発生
JMPD <>	JMPDパケット発生 (<>は後述)
JMPDS <>	JMPDSパケット発生 (<>は後述)
JMPS <>	JMPSパケット発生 (<>は後述)
OPCODE	オペコード・アクセス(実行)発生
DATAW1,2	メモリ書き込み発生(トレース・パケット)
DATAR1,2	メモリ読み出し発生(トレース・パケット)
SUB	サブサイクル

上記の“<>”部分は次の文字列が入ります。これは、分岐要因となった命令もしくは事象です。

NMI/INT	割り込みの発生によるもの
EXP/TRAP	例外の発生によるもの
RETI	当該命令によるもの
JMP	当該命令によるもの
JR	当該命令によるもの
JARL	当該命令によるもの
BcondNT	当該命令によるもの
Bcond	当該命令によるもの
CALLT	当該命令によるもの
SWITCH	当該命令によるもの
DISPOSE	当該命令によるもの
CTRET	当該命令によるもの
FSTART	トレースの強制スタート

\*:                   トリガポイント（多少ずれる場合があります）  
time =               タイムタグの表示

備考：タイムタグは、CPUから分岐情報が出力された時点のものです。分岐情報の出力は、実際の実行時間に対し遅れがあり、この遅れは一定ではありません。したがって、測定値には潜在的な誤差があります。また、実行直後の測定値は不定ですので無視してください。

t d a t a d l yコマンド

## [書式]

tdata\_dly [off|small|medium|large]

## [パラメータ]

off: 補正しません。  
small: 最小の補正をします。  
medium: 中程度の補正をします。(初期値)  
large: 最大の補正をします。

## [機能]

トレースクロックに対するトレースデータのセットアップ時間を調整するためのコマンドです。セットアップ時間はoffが一番小さく、largeが一番大きくなります。なお、実際のセットアップ値は使用するRTE-xxxx-TP本体やケーブルに依存しますので、各本体の仕様を確認ください。

## [補足]

通常は初期値から変更する必要はありませんが、CPUやボードの状態によっては調整が必要になる場合があります。  
このコマンドはRTE-2000(H)-TPでのみ使用できるコマンドです。



verコマンド

[書式]

ver

[パラメータ]

なし

[機能]

ICE制御のファームウェアのバージョンを表示します。