APPENDIX B. RTE-V850E/GP1-IE INTERNAL COMMANDS

This appendix describes the RTE-V850E/GP1-IE internal commands. These commands can be used as through commands in the debugger. For an explanation of using through commands, refer to the manual provided with the debugger.

With GHS-Multi

The through commands can be directly input in the target window after RTESERV has been connected.

Commands

Note These commands can be used only if the debugger does not provide equivalent functions. If these commands are issued when the debugger provides equivalent functions, a contention may occur between RTE-V850E/GP1-IE and the debugger, causing either device to malfunction.

Command syntax

The basic syntax for the RTE-V850E/GP1-IE internal commands is described below:

command-name parameter(s)

* In parameter syntax, a parameter enclosed in brackets ([ ]) is omissible. A horizontal line (|) indicates that one of the parameters delimited by it must be selected.

A command name must be an alphabetic character string, and be separated from its parameter(s) by a space or tab. A parameter must be an alphabetic character string or hexadecimal number, and be delimited by a space or tab. (A hexadecimal number cannot contain operators.)

abp, abp1, and abp2 commands

[Format]

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

[Parameters]

abp [or|and|seq]:	Specifies a condition for combination of abp1 and abp2.
or:	Break occurs if either abp1 or abp2 occurs.
and:	Break occurs if both abp1 and abp2 occur at the same time. A mask condition is used.
seq:	Break occurs if abp2 occurs after abp1.
abp{1|2}:	Input before the condition of abp1 or abp2 is specified.
ADDR [AMASK]:	Specifies an address condition.
ADDR:	Specifies addresses in hexadecimal number.
AMASK:	Specifies the mask data of an address in hexadecimal. Bits that are 1 will not be compared.
data DATA [DMASK]:	Specifies a data condition.
DATA:	Specifies data in hexadecimal.
DMASK:	Specifies the mask data of data in hexadecimal. Bits that are 1 will not be compared.
asid ASID|noasid:	For future expansion. Use noasid.
aeq|aneq:	Specifies an address comparison condition.
aeq:	Compares address for equality.
aneq:	Compares address for non-equality.
deq|dneq:	Specifies a data comparison condition.
deq:	Compares data for equality.
dneq:	Compares data for non-equality.
exec|read|write|accs:	Specifies a cycle condition.
exec:	Specifies an executable address. A data condition is ignored.
read:	Specifies a read cycle.
write:	Specifies a write cycle.
accs:	Specifies a read or write cycle.
byte|hword|word|nosize:	Specifies access size.
byte:	Specifies byte access (8 bits).
hword:	Specifies half-word access (16 bits).
word:	Specifies word access (32 bits).
nosize:	Specifies invalidity.
abp{1|2} /del:	Clears a condition.
/del:	Specifies deletion of a condition.

[Function]

These commands set or delete access breakpoints.
Up to two access breakpoints can be set.
They can specify execution addresses.

[Examples]

abp or

abp1 or abp2 is specified.

abp1 1000 aeq exec

A breakpoint for execution of address 1000h is set.

abp2 1000 data 5555 0 aeq deq read hword

Break occurs when 5555h is read in hword from address 1000h.

abp1 /del

The condition set by abp1 is deleted.

acc command

[Format]

acc [byte|hword|word]

[Parameter]

byte|hword|word: Specifies access size.

byte: Specifies byte access (8 bits).

hword: Specifies half-word access (16 bits).

word: Specifies word access (32 bits).

[Function]

Specifies the data size for the fly-by write command (fwr).

[Examples]

```
acc byte
fwr 0 12
    Procedure to write byte data with the fwr command
acc word
fwr 0 12345678
    Procedure to write word data with the fwr command
```

env command

[Format]

env [[!]auto] [[!][verify]] [jtag{25|12|5|2|1|500|250|100}]
[[!]nmi0] [[!]nmi1] [[!]nmi2] [[!]reset] [[!]hldrq] [[!]stopz] [[!]waitz]

[Parameters]

[!]auto: If a breakpoint is set during execution, the breakpoint causes a temporary break. Choose [auto] to automatically perform the subsequent execution. Choose [!auto] to suppress it.

[!]verify: Specifies whether the verification after writing memory is set. Enter ! if it is not to be set.

jtag{25|12|5|2|1|500|250|100}:
Specifies the JTAG clock for internal N-Wire. Each number corresponds to the following JTAG clock.
[25 MHz|12.5 MHz|5 MHz|2 MHz|1 MHz|500 kHz|250 kHz|100 kHz]

Remark Usually, use 25 MHz (default). If the frequency lower than 1 MHz is specified, the debugger might be slowed down in operation speed or might malfunction.

[!]nmi0, [!]nmi1, [!]nmi2:
Specifies whether the nmi pin is to be masked. Enter ! if it is not to be masked. NMI of the external pin is specified by nmi2.

[!]reset: Specifies whether the RESET pin is to be masked. Enter ! if it is not to be masked.

[!]hldrq: Do not change the default value.

[!]stopz: Specifies whether the STOP pin is to be masked. Enter ! if it is not to be masked.

[!]waitz: Do not change the default value.

[Function]

The env command sets the emulation environment and displays the DCU status.
Enter only those parameters that need to be changed. Parameters may be entered in any order.
If the same parameter is entered twice, only the last entry is valid.

The default values are as follows:

CPU Settings:
- Auto Run = ON (auto)
- JTAGCLOCK = 25MHz (jtag25)
- Verify = verify off (!verify)

Signals Mask:
- NMI0 = NO MASK (!nmi0)
- NMI1 = NO MASK (!nmi1)
- NMI2 = NO MASK (!nmi2)
- RESET = NO MASK (!reset)
- HLDRQ = NO MASK (!hldrq)
- STOPZ = NO MASK (!stopz)
- WAITZ = NO MASK (!waitz)

[Examples]

env reset !nmi2

RESET is masked while NMI2 is not masked.

env verify

Sets the Verify function to ON.

emode command

[Format]

emode

[Parameter]

None

[Function]

The emode command displays the event setting status.

[Example]

The initial status is shown below as an example:

```
Event Condition Settings:          <<Displays the setting status of the evt command.
  evt brk       !seq
  evt seqclr    !seq
  evt seq1      !seq
  evt seq2      !seq
  evt seq3      !seq
  evt seq4      !seq
  evt secon     !seq
  evt secoff    !seq
  evt qualify   !seq
  evt tout      !seq
  evt match     !seq
Event Settings (execute):          <<Displays the setting status of the eve command.
      ch Address   ASID   Cmp
  eve  1 /del
  eve  2 /del
  eve  3 /del
  eve  4 /del
  eve  5 /del
  eve  6 /del
  eve  7 /del
  eve  8 /del
Event Settings (access):           <<Displays the setting status of the eva command.
      ch Address       Data     D_Mask    ASID    A_Cmp D_Cmp Kind  Size
  eva  1 /del
  eva  2 /del
  eva  3 /del
  eva  4 /del
  eva  5 /del
  eva  6 /del
Sequence Condition Settings:       <<Displays the setting status of the seq command.
  seq 1 step4
```

eva command

[Format]

eva {1..6} [ADDR] [data DATA [MASK]] [asid ASID|noasid] [eq|lt|gt|neq|lte|gte|ign] [deq|dneq] [read|write|accs] [byte|hword|word|nosize] [/del]

[Parameters]

eva {1..6}:	Specifies an access event channel (1 to 6).
ADDR:	Specifies the address in hexadecimal.
data DATA [MASK]:	Specifies a data condition.
DATA:	Specifies data in hexadecimal.
MASK:	Specifies mask data for the data in hexadecimal. Bits that are 1 will not be compared.
asid ASID|noasid:	For future expansion. Use noasid.
eq|lt|gt|neq|lte|gte|ign:	
eq:	Specifies that the condition is satisfied when the event address is equal to the address specified for ADDR.
lt:	Specifies that the condition is satisfied when the event address is smaller than the address specified for ADDR.
gt:	Specifies that the condition is satisfied when the event address is greater than the address specified for ADDR.
neq:	Specifies that the condition is satisfied when the event address is not equal to the address specified for ADDR.
lte:	Specifies that the condition is satisfied when the event address is smaller than or equal to the address specified for ADDR.
gte:	Specifies that the condition is satisfied when the event address is greater than or equal to the address specified for ADDR.
ign:	Specifies that ADDR is not used as a comparison condition.
deq|dneq:	Specifies a data comparison condition.
deq:	Compares data for equality.
dneq:	Compares data for non-equality.
read|write|accs:	Specifies a cycle condition.
read:	Specifies a read cycle.
write:	Specifies a write cycle.
accs:	Specifies a read or write cycle.
byte|hword|word|nosize:	Specifies access size.
byte:	Specifies byte access (8 bits).
hword:	Specifies half-word access (16 bits).
word:	Specifies word access (32 bits).
nosize:	Specifies invalidity.
eva {1..6} /del:	Clears a condition.
/del:	Specifies deletion of a condition.

[Function]

The eva command sets an access event. The specified event can be combined with a condition using the evt command to be used as a break or trace condition.

[Examples]

eva 1 ffff000 data 55 00 byte read

A cycle for reading 0x55 starting at address 0xffff000 is set for eva#1 with using the default values for other parameters.

eva 1 /del

The condition of eva#1 is cleared.

eve command

[Format]

eve {1..8} [ADDR] [asid ASID|noasid] [eq|lt|gt|neq|lte|gte|ign] [/del]

[Parameters]

eve {1..8}: Specifies an execution event channel (1 to 8).

ADDR: Specifies the address in hexadecimal.

asid ASID|noasid: For future expansion. Use noasid.

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

eq: Specifies that the condition is satisfied when the event address is equal to the address specified for ADDR.

lt: Specifies that the condition is satisfied when the event address is smaller than the address specified for ADDR.

gt: Specifies that the condition is satisfied when the event address is greater than the address specified for ADDR.

neq: Specifies that the condition is satisfied when the event address is not equal to the address specified for ADDR.

lte: Specifies that the condition is satisfied when the event address is smaller than or equal to the address specified for ADDR.

gte: Specifies that the condition is satisfied when the event address is greater than or equal to the address specified for ADDR.

ign: Specifies that ADDR is not used as a comparison condition.

eve {1..8} /del: Clears a condition.

/del: Specifies deletion of a condition.

[Function]

The eve command sets an execution event. The specified event can be combined with a condition using the evt command to be used as a break or trace condition.

[Examples]

eve 1 1000

Execution of the instruction at address 0x1000 is set for eve#1 using the default values for other parameters.

eve 1 /del

The condition of eve#1 is cleared.

evt command

[Format]

evt {brk|seqclr|seq1|seq2|seq3|seq4|secon|secoff|qualify|tout|match}
evep{[1][2][3]..[8]} ever{[1][3][5][7]} evap{[1][2][3]..[6]}
evar{[1][3][5]} [[!]seq]

[Parameters]

brk|seqclr|seq1|seq2|seq3|seq4|secon|secoff|qualify|tout|match:
Specifies a condition with which the event is to be combined.

- brk: Specifies a break condition.
- seqclr: Specifies a sequential clear condition.
- seq1: Specifies a first-step sequential condition.
- seq2: Specifies a second-step sequential condition.
- seq3: Specifies a third-step sequential condition.
- seq4: Specifies a fourth-step sequential condition.
- secon: Specifies a trace section on condition.
- secoff: Specifies a trace section off condition.
- qualify: Specifies a trace qualify condition.
- tout: Specifies a trigger output condition.
- match: Specifies a trace trigger condition.

evep{[1][2][3]..[8]}: Specifies the corresponding event specified by the eve command as a point by itself. Specifying this parameter with no numeric characters cancels the setting.

- [1][2][3]..[8]: Each number corresponds to a channel number specified by eve.

ever{[1][3][5][7]}: Specifies each pair of events specified by the eve command as an area. Specifying this parameter with no numeric characters cancels the setting.

- 1: Specifies the conditions of channels 1 and 2 specified by eve as a range (and condition).
- 3: Specifies the conditions of channels 3 and 4 specified by eve as a range (and condition).
- 5: Specifies the conditions of channels 5 and 6 specified by eve as a range (and condition).
- 7: Specifies the conditions of channels 7 and 8 specified by eve as a range (and condition).

evap{[1][2][3]..[6]}: Specifies the corresponding event specified by the eva command as a point by itself. Specifying this parameter with no numeric characters cancels the setting.

- [1][2][3]..[6]: Each number corresponds to a channel number specified by eva.

evar{[1][3][5]}: Specifies each pair of events specified by the eva command as an area. Specifying this parameter with no numeric characters cancels the setting.

- 1: Specifies the conditions of channels 1 and 2 specified by eva as a range (and condition).
- 3: Specifies the conditions of channels 3 and 4 specified by eva as a range (and condition).
- 5: Specifies the conditions of channels 5 and 6 specified by eva as a range (and condition).

[!]seq: Specifies a sequential condition.

- seq: Specifies a sequential condition. Enter ! to cancel the sequential condition. ! cannot be specified for a seq-related condition (seqclr, seq1, seq2, seq3, or seq4).

[Function]

The evt command specifies the use of each event specified by eve or eva.

[Examples]

evt brk evep1234 ever5 evap12 evar3

As break events, the events specified for channels 1 to 4 by eve are used as points; those specified for channels 5 and 6 as a range condition; those specified for channels 1 and 2 by eva as points; and those specified for channels 3 and 4 as a range.

evt brk evep ever evap evar

The events specified for evep, ever, evap, and evar as break events are canceled.

[Remark]

For the details of the sequential conditions, see the description of the seq command.

For the details of the trace section and qualify conditions, see Appendix A, "Details of Trace Functions".

extbrk command

[Format]

extbrk [disable|posi|nega]

[Parameters]

disable:	Disables this capability (default).
posi:	Break request at positive edge detection
nega:	Break request at negative edge detection

[Function]

The extbrk command specifies the break request using external input signal (pin 4 of EXT connector (EXI1)).

[Example]

extbrk posi

A break is requested at positive edge detection.

frd command

[Format]

frd [ADDRESS [LENGTH]]

[Parameters]

ADDRESS: Specifies the start address of the memory to be read, in hexadecimal number. Only the internal RAM address can be specified.

LENGTH: Specify the data to be output, in hexadecimal number (max.: 100).

[Function]

The frd command is used to read the contents of the internal RAM on a fly-by basis during execution.

[Example]

frd ffff0000 100

Reads 0x100 bytes, starting from 0xffff0000.

Contiguous addresses can be referenced by pressing the return key immediately after the display was completed.

fwr command

[Format]

fwr [ADDRESS [DATA]]

[Parameters]

ADDRESS: Specifies the start address of the memory to be written, in hexadecimal number. Only the internal RAM address can be specified.

DATA: Specify the data to be output, in hexadecimal number.

[Function]

The fwr command writes data on a fly-by basis to the internal RAM during execution.

The data and access size are specified by the acc command.

[Examples]

acc hword

The acc command must be specified in advance to access data in half words.

fwr ffff1000 1234

Writes half-word data, 1234h, to 1000H.

help command

[Format]

help [command]

[Parameter]

command: Specifies the name of the command for which you require help.
If this parameter is omitted, a list of commands is displayed.

[Function]

The help command displays a help message for a specified command.

[Example]

help map

A help message for the map command is displayed.

inb, inh, and inw commands

[Format]

inb [ADDR]
inh [ADDR]
inw [ADDR]

[Parameter]

ADDR: Specifies the address of an input port in hexadecimal.

[Function]

The inb, inh, and inw commands read the I/O space in different sizes.
The inb command accesses I/O space in bytes, inh in half words, and inw in words.

[Examples]

inb 1000
The I/O space is read in bytes (8-bit units), starting at 1000H.
inh 1000
The I/O space is read in half words (16-bit units), starting at 1000H.
inw 1000
The I/O space is read in words (32-bit units), starting at 1000H.

init command

[Format]

init

[Parameter]

None

[Function]

The init command initializes RTE-V850E/GP1-IE. All environment values are initialized.
A memory cache rejection area is not initialized.

nc command

[Format]

nc [[ADDR [LENGTH]]

[Parameters]

ADDR: Specifies the start address of a memory cache rejection area.

LENGTH: Specifies the length of the memory cache rejection area in bytes.
The default value is 32 bytes. The allowable minimum value is also 32 bytes.

[Function]

To ensure quick memory access, RTE-V850E/GP1-IE provides a memory read cache of 8 blocks*32 bytes in the ICE. When the same memory address is accessed more than once, the read operation is not actually performed. This cache operation conflicts with the actual operation when an I/O unit is mapped onto memory. In such a case, specify a memory cache rejection area by using the nc command. Up to eight blocks can be specified as a memory cache rejection area. The allowable minimum block size is 32 bytes.

Areas other than the ROM and RAM areas are specified as memory cache rejection areas by default. Usually, the specification of memory cache rejection areas does not need to be changed.

[Example]

The initial status is shown below as an example:

```
>nc
 No Memory Cache Area
 No.  Address     Length
 1    00100000    03ef0000
 2    0ffff000    00001000
```

ncd command

[Format]

ncd block-number

[Parameter]

block-number: Specifies the block number for a memory cache rejection area to be deleted.

[Function]

The ncd command deletes a memory cache rejection area. Specify the block number corresponding to the memory cache rejection area to be deleted. Do not delete any default memory cache rejection area.

If an default memory cache rejection area is deleted, accessing an I/O space by a command may fail to read correct values.

[Example]

ncd 1

Block 1 is deleted from the memory cache rejection area.

>>This is just an example. Do not delete the block actually.

```
>nc
 No Memory Cache Area
 No.  Address     Length
 1    00100000    03ef0000
 2    0ffff000    00001000

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

outb, outh, and outw commands

[Format]

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

[Parameters]

ADDR: Specifies the address of an output port in hexadecimal.
DATA: Specifies the data to be output in hexadecimal.

[Function]

The outb, outh, and outw commands write data to the I/O space in different sizes.
The outb command accesses the I/O space in bytes, outh in half words, and outw in words.

[Examples]

outb 1000 12
Byte data 12h is written to 1000H in the I/O space.
outh 1000 1234
Half-word data 1234h is written to 1000H in the I/O space.
outw 1000 12345678
Word data 12345678h is written to 1000H in the I/O space.

reset command

[Format]

reset

[Parameter]

None

[Function]

The reset command resets the CPU.

seq command

[Format]

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

[Parameters]

PASS: Specifies in decimal the number of times the sequence condition is to be satisfied.

step{1|2|3|4}: Specifies the number of steps in the sequence.

- 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

[Function]

The seq command sets the sequential conditions.

Use eve, eva, and evt to specify conditions for seq1 to seq4.

When the seqclr condition is satisfied during a sequence, the sequence is executed from the beginning.

[Example]

seq 100 step1

A seq event occurs when conditions seq1 -> seq2 -> seq3 -> seq4 are satisfied 100 times.

sswon and sswoff commands

[Format]

ssw{on|off} [{exec_{[0]..[e]}|exec_default}]
[td{1|2|3|4} {none|read|write|accs|readp|writep|accsp}]
[evap{1|2|3|4|5|6} {none|read|write|accs|readp|writep|accsp}]
[evar{1|3|5} {none|read|write|accs|readp|writep|accsp}]
[all_cycle {none|read|write|accs|readp|writep|accsp}]

[Parameters]

sswon: This command specifies a cycle in which trace data is to be loaded when the sub-switch is on.

sswoff: This command specifies a cycle in which trace data is to be loaded when the sub-switch is off.

exec_{[0]...[e]}: Specifies a cycle in which data is to be loaded as execution trace.
Each number corresponds to a cycle as follows. If trace data of some execution cycles is not loaded, disassembled trace data display may not be performed correctly.
0:Interrupt, 1:Exception, 2:RETI, 3:JMP, 4:JR, 5:JARL,
6:Condition Jump(not taken), 7:Condition Jump(taken),
8:CALLT, 9:SWITCH, a:DISPOSE, b:CTRET,
c:tp, d:evt_match, e:opecode

exec_default: Loads trace data in all cycles. (Equivalent to "exec_0123456789abcd".) Usually, use this option.

td{1|2|3|4} {none|read|write|accs|readp|writep|accsp}:
Specifies the type of cycle in which trace data is to be loaded for each condition specified by the td command.

- none: Does not load trace data.
- read: Loads trace data in read cycles only.
- write: Loads trace data in write cycles only.
- accs: Loads trace data in read and write cycles.
- readp: Loads trace data in read cycles and their execution cycles.
- writep: Loads trace data in write cycles and their execution cycles.
- accsp: Loads trace data in read and write cycles, and their execution cycles.

evap{1|2|3|4|5|6} {none|read|write|accs|readp|writep|accsp}:
Specifies the type of cycle in which trace data is to be loaded for each point condition specified by the eva command.

- none: Does not load trace data.
- read: Loads trace data in read cycles only.
- write: Loads trace data in write cycles only.
- accs: Loads trace data in read and write cycles.
- readp: Loads trace data in read cycles and their execution cycles.
- writep: Loads trace data in write cycles and their execution cycles.
- accsp: Loads trace data in read and write cycles, and their execution cycles.

evar{1|3|5} {none|read|write|accs|readp|writep|accsp}:

Specifies the type of cycle in which trace data is to be loaded for each range condition specified by the eva command.

- none: Does not load trace data.
- read: Loads trace data in read cycles only.
- write: Loads trace data in write cycles only.
- accs: Loads trace data in read and write cycles.
- readp: Loads trace data in read cycles and their execution cycles.
- writep: Loads trace data in write cycles and their execution cycles.
- accsp: Loads trace data in read and write cycles, and their execution cycles.

all_cycle {none|read|write|accs|readp|writep|accsp}:

Specifies the type of cycle in which trace data is to be loaded unconditionally.

- none: Does not load trace data.
- read: Loads trace data in read cycles only.
- write: Loads trace data in write cycles only.
- accs: Loads trace data in read and write cycles.
- readp: Loads trace data in read cycles and their execution cycles.
- writep: Loads trace data in write cycles and their execution cycles.
- accsp: Loads trace data in read and write cycles, and their execution cycles.

[Function]

The sswon and sswoff commands specify the types of cycles in which trace data is to be loaded according to the sub-switch status.

[Example]

By default, trace data in all cycles is to beloaded when the sub-switch is on and trace data in no cycles is to be loaded when it is off.

These commands can be used to control the loading of trace data under any desired conditions.

The default settings are shown below.

```
>sswon
Sub-switch ON Settings:
  Trace execute cycle              = exec_0123456789abcd (exec_default)
  td1 Trace cycle   (td1)          = Read and Write cycle with pc value (accsp)
  td2 Trace cycle   (td2)          = Read and Write cycle with pc value (accsp)
  td3 Trace cycle   (td3)          = Read and Write cycle with pc value (accsp)
  td4 Trace cycle   (td4)          = Read and Write cycle with pc value (accsp)
  evap1 Trace cycle (evap1)        = No cycle (none)
  evap2 Trace cycle (evap2)        = No cycle (none)
  evap3 Trace cycle (evap3)        = No cycle (none)
  evap4 Trace cycle (evap4)        = No cycle (none)
  evap5 Trace cycle (evap5)        = No cycle (none)
  evap6 Trace cycle (evap6)        = No cycle (none)
  evar1 Trace cycle (evar1)        = No cycle (none)
  evar3 Trace cycle (evar3)        = No cycle (none)
  evar5 Trace cycle (evar5)        = No cycle (none)
  All access cycle  (all_cycle)    = No cycle (none)

>sswoff
Sub-switch OFF Settings:
  Trace execute cycle              = exec_
  td1 Trace cycle   (td1)          = No cycle (none)
  td2 Trace cycle   (td2)          = No cycle (none)
  td3 Trace cycle   (td3)          = No cycle (none)
  td4 Trace cycle   (td4)          = No cycle (none)
  evap1 Trace cycle (evap1)        = No cycle (none)
  evap2 Trace cycle (evap2)        = No cycle (none)
  evap3 Trace cycle (evap3)        = No cycle (none)
  evap4 Trace cycle (evap4)        = No cycle (none)
  evap5 Trace cycle (evap5)        = No cycle (none)
  evap6 Trace cycle (evap6)        = No cycle (none)
  evar1 Trace cycle (evar1)        = No cycle (none)
  evar3 Trace cycle (evar3)        = No cycle (none)
  evar5 Trace cycle (evar5)        = No cycle (none)
  All access cycle  (all_cycle)    = No cycle (none)
```

[Remark]

For the details of the sub-switch, see Appendix A, "Details of Trace Functions".

sfr and sfr2 commands

[Format]

sfr [reg [VAL]]

sfr2 [reg [VAL]]

[Parameters]

reg: Specifies a relocatable SFR register name.

VAL: Specifies the value for an SFR register in hexadecimal.

The following names can be used as register names:

<Registers that can be accessed by the sfr command>

SFR (R/W):

BPC VSWC IMR0 IMR0L IMR0H IMR1
IMR1L IMR1H IMR2 IMR2L IMR2H IMR3 IMR3L IMR3H IMR4 IMR4L IMR4H IMR5
IMR5L IMR5H IMR6 IMR6L IMR6H VIC512 VIC1M VIC2M VIC4M PIC000 PIC001
PIC002 PIC003 PIC004 PIC005 PIC006 PIC007 PIC008 PIC009 PIC010 PIC011
PIC012 PIC013 PIC014 PIC015 CMRIC00 CMRIC01 CMRIC02 CMRIC03 CMRIC04
CMRIC05 CMRIC06 CMRIC07 CMRIC08 CMRIC09 CMRIC10 CMRIC11 CMRIC12 CMRIC13
CMRIC14 CMRIC15 CMFIC00 CMFIC01 CMFIC02 CMFIC03 CMFIC04 CMFIC05 CMFIC06
CMFIC07 CMFIC08 CMFIC09 CMFIC10 CMFIC11 CMFIC12 CMFIC13 CMFIC14 CMFIC15
CMIC16 CMIC17 CMIC18 CMIC19 CMIC20 CMIC21 CMIC22 CMIC23 CMIC24 CMIC25
CMIC26 CMIC27 CMIC28 CMIC29 CMIC30 CMIC31 PW1IC0 PW1IC1 PW1IC2 PW1IC3
PW1IC4 PW1IC5 PW1IC6 PW1IC7 PW2IC0 PW2IC1 PW2IC2 PW2IC3 PW2IC4 PW2IC5
PW2IC6 PW2IC7 MACIC0 CNERIC1 CNRXIC1 CNTXIC1 CNERIC2 CNRXIC2 CNTXIC2
CSIIC1 CSIIC2 CSIIC3 SRIC1 STIC1 SRIC2 STIC2 SRIC3 STIC3 SRIC4 STIC4
SRIC5 STIC5 SOFTIC0 PIC10 SOFTIC1 ADIC0 ADMO
ADEN ADST ADTR ADIS MAR0 MAR1
MAR2 MAR3 MAR4 MAR5 MAR6 MAR7 MAR8 MAR9 MAR10 MAR11 MAR12 MAR13 MAR14
MAR15 SAR0 SAR1 SAR2 SAR3 SAR4 SAR5 SAR6 SAR7 DTCR1 DTCR2 DTCR3 DTCR4
DTCR5 DTCR6 DTCR7 DTCR8 DTCR9 DTCR10 DTCR11 DTCR12 DTCR13 DTCR14 DTCR15
DMAMC0 DMAMC1 DMAS0 DMAS1 DMAS2 DTFR1 DTFR2 DTFR3 DTFR4 DTFR5 DTFR6
DTFR7 P0 P1 P2 P3 P4 P5 P6 P9
P10 P11 P12 P13 P14 P15 PM0 PM1 PM2 PM3 PM4 PM5 PM6 PM9
PM10 PM11 PM12 PM13 PM14 PM15 PMC0 PMC1 PMC2 PMC3 PMC9
PMC10 PMC11 PMC12 PMC13 PMC14 PMC15 PWMC10
PWMC11 PWMC12 PWMC13 PWMC14 PWMC15 PWMC16 PWMC17 CMC10 CMD10 CMC11
CMD11 CMC12 CMD12 CMC13 CMD13 CMC14 CMD14 CMC15 CMD15 CMC16 CMD16 CMC17
CMD17 PWMC20 PWMC21 PWMC22 PWMC23 PWMC24 PWMC25 PWMC26
PWMC27 CMC20 CMD20 CMC21 CMD21 CMC22 CMD22 CMC23 CMD23 CMC24 CMD24
CMC25 CMD25 CMC26 CMD26 CMC27 CMD27 P16
P17 P18 PM16 PM17 PM18 PMC16
PMC17 PMC18 PFC17 PL0 PL1 PES0
PES1 NRC PHS CKC INTM0 INTM1
ASIM10 ASIM11 ATXB1 ATXBL1 ACKSR1
ABRGC1 ASIM20 ASIM21 PRS2M PRS2CM

ASIM30 ASIM31 ATXB3 ATXBL3 ACKSR3
ABRGC3 ASIM40 ASIM41 ATXB4 ATXBL4 ACKSR4
ABRGC4 ASIM50 ASIM51 ATXB5 ATXBL5 ACKSR5
ABRGC5 TOC0 TOC1 TOC2 TOC3 TMDL0
TMDL1 TMDL2 TMDL3 OSI0 OSI1 CPCNT8 CPCNT9 CPCNT10 CPCNT11 CPCNT12 CPCNT13
CPCNT14 CPCNT15 CMR0 CMR1 CMR2
CMR3 CMR4 CMR5 CMR6 CMR7 CMR8 CMR9 CMR10
CMR11 CMR12 CMR13 CMR14 CMR15 CMF0 CMF1 CMF2
CMF3 CMF4 CMF5 CMF6 CMF7 CMF8 CMF9 CMF10
CMF11 CMF12 CMF13 CMF14 CMF15 CM16 CM17 CM18
CM19 CM20 CM21 CM22 CM23 CM24 CM25 CM26
CM27 CM28 CM29 CM30 CM31 SES0 SES1 SES2 SES3 CTXB1
CTXB10 CTXBL10 CTXB11 CSIM10 CSIM11 CSIL1 CSIC1 CSIS1 CBRGC1 ITC1 CTXB2
CTXB20 CTXBL20 CTXB21 CSIM20 CSIM21 CSIL2 CSIC2 CSIS2 CBRGC2 ITC2 CTXB3
CTXB30 CTXBL30 CTXB31 CSIM30 CSIM31 CSIL3 CSIC3 CSIS3 CBRGC3 ITC3 DMAWC0
DMAWC1

SFR (W):
PRCMD ATXS2 ATXSL2

SFR (R):
ISPR ADCR0 ADCR1 ADCR2 ADCR3
ADCR4 ADCR5 ADCR6 ADCR7 ADCR8 ADCR9 ADCR10 ADCR11 ADCR12 ADCR13 ADCR14
ADCR15 CKDR ARXB1 ARXBL1 ASIS1
ASIF1 ASIS2 ARXB2 ARXBL2 ARXB3
ARXBL3 ASIS3 ASIF3 ARXB4 ARXBL4
ASIS4 ASIF4 ARXB5 ARXBL5 ASIS5
ASIF5 GTM CPR0 CPF0 CPR1 CPF1
CP2 CP3 CP4 CP5 CP6 CP7 CP8 CP9
CP10 CP11 CP12 CP13 CP14 CP15 CRXB1
CRXB10 CRXBL10 CRXB11 CRXB2 CRXB20
CRXBL20 CRXB21 CRXB3 CRXB30 CRXBL30
CRXB31

<Registers that can be accessed by the sfr2 command>

SFR (R/W):
M_DLC00 M_CTRL00 M_TIME00 M_DATA000
M_DATA001 M_DATA002 M_DATA003 M_DATA004 M_DATA005 M_DATA006 M_DATA007
M_ID00 M_IDL00 M_IDH00 M_CONF00 M_DLC01 M_CTRL01 M_TIME01 M_DATA010
M_DATA011 M_DATA012 M_DATA013 M_DATA014 M_DATA015 M_DATA016 M_DATA017
M_ID01 M_IDL01 M_IDH01 M_CONF01 M_DLC02 M_CTRL02 M_TIME02 M_DATA020
M_DATA021 M_DATA022 M_DATA023 M_DATA024 M_DATA025 M_DATA026 M_DATA027
M_ID02 M_IDL02 M_IDH02 M_CONF02 M_DLC03 M_CTRL03 M_TIME03 M_DATA030
M_DATA031 M_DATA032 M_DATA033 M_DATA034 M_DATA035 M_DATA036 M_DATA037
M_ID03 M_IDL03 M_IDH03 M_CONF03 M_DLC04 M_CTRL4 M_TIME04 M_DATA040
M_DATA041 M_DATA042 M_DATA043 M_DATA044 M_DATA045 M_DATA046 M_DATA047
M_ID04 M_IDL04 M_IDH04 M_CONF04 M_DLC05 M_CTRL05 M_TIME05 M_DATA050
M_DATA051 M_DATA052 M_DATA053 M_DATA054 M_DATA055 M_DATA056 M_DATA057
M_ID05 M_IDL05 M_IDH05 M_CONF05 M_DLC06 M_CTRL06 M_TIME06 M_DATA060

M_DATA061 M_DATA062 M_DATA063 M_DATA064 M_DATA065 M_DATA066 M_DATA067
M_ID06 M_IDL06 M_IDH06 M_CONF06 M_DLC07 M_CTRL07 M_TIME07 M_DATA070
M_DATA071 M_DATA072 M_DATA073 M_DATA074 M_DATA075 M_DATA076 M_DATA077
M_ID07 M_IDL07 M_IDH07 M_CONF07 M_DLC08 M_CTRL08 M_TIME08 M_DATA080
M_DATA081 M_DATA082 M_DATA083 M_DATA084 M_DATA085 M_DATA086 M_DATA087
M_ID08 M_IDL08 M_IDH08 M_CONF08 M_DLC09 M_CTRL09 M_TIME09 M_DATA090
M_DATA091 M_DATA092 M_DATA093 M_DATA094 M_DATA095 M_DATA096 M_DATA097
M_ID09 M_IDL09 M_IDH09 M_CONF09 M_DLC10 M_CTRL10 M_TIME10 M_DATA100
M_DATA101 M_DATA102 M_DATA103 M_DATA104 M_DATA105 M_DATA106 M_DATA107
M_ID10 M_IDL10 M_IDH10 M_CONF10 M_DLC11 M_CTRL11 M_TIME11 M_DATA110
M_DATA111 M_DATA112 M_DATA113 M_DATA114 M_DATA115 M_DATA116 M_DATA117
M_ID11 M_IDL11 M_IDH11 M_CONF11 M_DLC12 M_CTRL12 M_TIME12 M_DATA120
M_DATA121 M_DATA122 M_DATA123 M_DATA124 M_DATA125 M_DATA126 M_DATA127
M_ID12 M_IDL12 M_IDH12 M_CONF12 M_DLC13 M_CTRL13 M_TIME13 M_DATA130
M_DATA131 M_DATA132 M_DATA133 M_DATA134 M_DATA135 M_DATA136 M_DATA137
M_ID13 M_IDL13 M_IDH13 M_CNF13 M_DLC14 M_CTRL14 M_TIME14 M_DATA140
M_DATA141 M_DATA142 M_DATA143 M_DATA144 M_DATA145 M_DATA146 M_DATA147
M_ID14 M_IDL14 M_IDH14 M_CONF14 M_DLC15 M_CTRL15 M_TIME15 M_DATA150
M_DATA151 M_DATA152 M_DATA153 M_DATA154 M_DATA155 M_DATA156 M_DATA157
M_ID15 M_IDL15 M_IDH15 M_CONF15 M_DLC16 M_CTRL16 M_TIME16 M_DATA160
M_DATA161 M_DATA162 M_DATA163 M_DATA164 M_DATA165 M_DATA166 M_DATA167
M_ID16 M_IDL16 M_IDH16 M_CONF16 M_DLC17 M_CTRL17 M_TIME17 M_DATA170
M_DATA171 M_DATA172 M_DATA173 M_DATA174 M_DATA175 M_DATA176 M_DATA177
M_ID17 M_IDL17 M_IDH17 M_CONF17 M_DLC18 M_CTRL18 M_TIME18 M_DATA180
M_DATA181 M_DATA182 M_DATA183 M_DATA184 M_DATA185 M_DATA186 M_DATA187
M_ID18 M_IDL18 M_IDH18 M_CONF18 M_DLC19 M_CTRL19 M_TIME19 M_DATA190
M_DATA191 M_DATA192 M_DATA193 M_DATA194 M_DATA195 M_DATA196 M_DATA197
M_ID19 M_IDL19 M_IDH19 M_CONF19 M_DLC20 M_CTRL20 M_TIME20 M_DATA200
M_DATA201 M_DATA202 M_DATA203 M_DATA204 M_DATA205 M_DATA206 M_DATA207
M_ID20 M_IDL20 M_IDH20 M_CONF20 M_DLC21 M_CTRL21 M_TIME21 M_DATA210
M_DATA211 M_DATA212 M_DATA213 M_DATA214 M_DATA215 M_DATA216 M_DATA217
M_ID21 M_IDL21 M_IDH21 M_CONF21 M_DLC22 M_CTRL22 M_TIME22 M_DATA220
M_DATA221 M_DATA222 M_DATA223 M_DATA224 M_DATA225 M_DATA226 M_DATA227
M_ID22 M_IDL22 M_IDH22 M_CONF22 M_DLC23 M_CTRL23 M_TIME23 M_DATA230
M_DATA231 M_DATA232 M_DATA233 M_DATA234 M_DATA235 M_DATA236 M_DATA237
M_ID23 M_IDL23 M_IDH23 M_CONF23 M_DLC24 M_CTRL24 M_TIME24 M_DATA240
M_DATA241 M_DATA242 M_DATA243 M_DATA244 M_DATA245 M_DATA246 M_DATA247
M_ID24 M_IDL24 M_IDH24 M_CONF24 M_DLC25 M_CTRL25 M_TIME25 M_DATA250
M_DATA251 M_DATA252 M_DATA253 M_DATA254 M_DATA255 M_DATA256 M_DATA257
M_ID25 M_IDL25 M_IDH25 M_CONF25 M_DLC26 M_CTRL26 M_TIME26 M_DATA260
M_DATA261 M_DATA262 M_DATA263 M_DATA264 M_DATA265 M_DATA266 M_DATA267
M_ID26 M_IDL26 M_IDH26 M_CONF26 M_DLC27 M_CTRL27 M_TIME27 M_DATA270
M_DATA271 M_DATA272 M_DATA273 M_DATA274 M_DATA275 M_DATA276 M_DATA277
M_ID27 M_IDL27 M_IDH27 M_CONF27 M_DLC28 M_CTRL28 M_TIME28 M_DATA280
M_DATA281 M_DATA282 M_DATA283 M_DATA284 M_DATA285 M_DATA286 M_DATA287
M_ID28 M_IDL28 M_IDH28 M_CONF28 M_DLC29 M_CTRL29 M_TIME29 M_DATA290
M_DATA291 M_DATA292 M_DATA293 M_DATA294 M_DATA295 M_DATA296 M_DATA297

M_ID29 M_IDL29 M_IDH29 M_CONF29 M_DLC30 M_CTRL30 M_TIME30 M_DATA300
M_DATA301 M_DATA302 M_DATA303 M_DATA304 M_DATA305 M_DATA306 M_DATA307
M_ID30 M_IDL30 M_IDH30 M_CONF30 M_DLC31 M_CTRL31 M_TIME31 M_DATA310
M_DATA311 M_DATA312 M_DATA313 M_DATA314 M_DATA315 M_DATA316 M_DATA317
M_ID31 M_IDL31 M_IDH31 M_CONF31 CSTOP
CGST CGIE CGCS CGINTP C1INTP
C2INTP C1MASK0 C1MASKL0 C1MASKH0
C1MASK1 C1MASKL1 C1MASKH1 C1MASK2 C1MASKL2 C1MASKH2 C1MASK3 C1MASKL3
C1MASKH3 C1CTRL C1DEF C1IE C1BRP C1SYNC C2MASK0
C2MASKL0 C2MASKH0 C2MASK1 C2MASKL1 C2MASKH1 C2MASK2 C2MASKL2 C2MASKH2
C2MASK3 C2MASKL3 C2MASKH3 C2CTRL C2DEF C2IE C2BRP C2SYNC

SFR (W):

SC_STAT00 SC_STAT01 SC_STAT02
SC_STAT03 SC_STAT04 SC_STAT05
SC_STAT06 SC_STAT07 SC_STAT08
SC_STAT09 SC_STAT10 SC_STAT11
SC_STAT12 SC_STAT13 SC_STAT14
SC_STAT15 SC_STAT16 SC_STAT17
NSCST18 SC_STAT19 SC_STAT20 SC_STAT21
SC_STAT22 SC_STAT23 SC_STAT24
SC_STAT25 SC_STAT26 SC_STAT27
SC_STAT28 SC_STAT29 SC_STAT30
SC_STAT31 CGMSS

SFR (R):

M_STAT00 M_STAT01 M_STAT02 M_STAT03
M_STAT04 M_STAT05 M_STAT06 M_STAT07
M_STAT08 M_STAT09 M_STAT10 M_STAT11
M_STAT12 M_STAT13 M_STAT14 M_STAT15
M_STAT16 M_STAT17 M_STAT18 M_STAT19
M_STAT20 M_STAT21 M_STAT22 M_STAT23
M_STAT24 M_STAT25 M_STAT26 M_STAT27
M_STAT28 M_STAT29 M_STAT30 M_STAT31
CCINTP CGTSC CGMSR C1LAST C1ERC
C1BA C1DINF C2LAST C2ERC C2BA
C2DINF

[Function]

The sfr command sets and displays the value of the SFR register.

[Examples]

sfr P10

The value of the P10 register is displayed.

sfr PM10 0

Value 0h is set in the PM10 register.

sfr2 C2CTRC

Refers the C2CTRL register.

symfile and sym commands

[Format]

symfile FILENAME

sym [NAME]

[Parameters]

FILENAME: Specifies file name.

NAME: Specifies first character string in the symbols to be displayed.

[Function]

The symfile command reads symbols from the elf file specified by the FILENAME parameter.

Only global symbols can be read.

The sym command displays up to 30 symbols that have been read.

[Examples]

symfile c:\test\dry\dry.elf

Symbols are read from the elf file dry.elf in the c:\test\dry directory.

sym m

Up to 30 symbols that begin with "m" are displayed.

time command

[Format]

time [sysclk]

[Parameter]

sysclk: Specifies the system clock of the CPU in MHz. The value is valid at the second place below the decimal point. The default value is 64 MHz.

[Function]

This time command displays the result of measuring the execution time. The timer that measures the execution time is initialized to the default value each time the CPU has started execution, and counts during CPU execution. The timer value is counted once on the CPU clock.

[Remark]

The measured value includes the overhead times (error of several clocks) at the start of execution and breaks.

[Examples]

```
>time
Time = 1,139,655,796 (ns) (64.000MHz) [Counter=0x000458f1f3]>
```

Displays time from execution immediately before to break.

```
>time 32
```

Changes the default value 64 MHz of the CPU operating clock (to 32 MHz in this example).

td1, td2, td3, and td4 commands

[Format]

td{1|2|3|4} [ADDR [MASK]] [asid ASID|noasid] [/del]

[Parameters]

td{1|2|3|4}:	Input before the condition of a command from td1, td2, td3, and td4 is specified.
ADDR:	Specifies an address in hexadecimal.
MASK:	Specifies the mask data of an address in hexadecimal. Bits that are 1 are not subject to comparison. Only bits 9 through 2 are valid.
asid ASID|noasid:	For future expansion. Use noasid.
/del:	Clears the specified address.

[Function]

The td1, td2, td3, and td4 commands set the conditions of the data access cycles to be recorded by trace. These conditions can be used as trace loading conditions and triggers.

[Example]

td1 100000 ff

The access cycle of address 1000xxh is loaded to trace.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

tenv command

[Format]

tenv [subor|suband] [[!]dmatrc] [[!]sfrtrc] [[!]tendbrk] [[!]stop]
[rtrcb{0|8|16}]|[nrtrcb{0|4|8|12|16|20|24}]
[nonbranchNN] [[!]phold] [[!]once] [[!]debug]

[Parameters]

subor:	Specifies OR of the section and qualify conditions as the sub-switch.
suband:	Specifies AND of the section and qualify conditions as the sub-switch.
[!]dmatrc:	Traces cycle of DMA. Enter ! to inhibit tracing.
[!]sfrtrc:	Always use as !sfrtrc.
[!]tendbrk:	Break occurs upon completion of trace. Enter ! to cause no break.
[!]stop:	Stops trace in the stop mode. Enter ! not to stop trace.
rtrcb{0|8|16}:	Specifies the number of packages used of the buffer when execution restores from overflow during real-time trace. Usually, use the default value "0".
nrtrcb{0|4|8|12|16|20|24}:	Specifies the number of packets used of the buffer when a request to stop the pipeline is made in the complete trace mode. Usually, use the default value "0".
nonbranchNN:	Specifies the trace loading interval for PC information when the execution of the instructions at contiguous addresses is continued. For NN, specify a value between 0 and fff. NN = 0 means infinity. Usually, use the default value (0).
[!]phold:	Loads a packet indicating that execution is stopped during trace in the complete (non-real-time) mode. Enter ! to load no packet.
[!]once:	Outputs trigger output once when the trigger condition is satisfied. Enter ! to output trigger output each time the trigger condition is satisfied.
[!]debug:	Use the default value (!debug).

[Function]

The tenv command sets the trace environment.

[Example]

tenv subor dmatrc

Sub-switch is ORed with section and quality and traces DMA cycles.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

tp command

[Format]

tp [ADDR] [asid ASID|noasid] [/del]

[Parameters]

ADDR:	Specifies an even-numbered address in hexadecimal. (A0 is always corrected to 0.)
asid ASID|noasid:	For future expansion. Use noasid.
/del:	Clears the specified address.

[Function]

The tp command specifies a trace trigger point.

Trace is used to monitor the execution status before and after a trigger point.

[Example]

tp 100000

The execution of the instruction at 100000h is specified as a trigger point.

[Note]

If delay mode is specified with the tron command, the trigger point specification is ignored.

Delay mode can be canceled by entering tron !delay.

For the details of the trace, see Appendix A, "Details of Trace Functions".

tsp1 and tsp2 commands

[Format]

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

[Parameters]

tsp{1|2}:	Input before the condition of tsp1 or tsp2 is specified.
ADDR:	Specifies an execution address in hexadecimal.
asid ASID|noasid:	For future expansion. Use noasid.
/del:	Clears the specified address.

[Function]

The tsp1 and tsp2 commands specify the section points (addresses) of the two trace points.
The cycle in which the trace information is to be loaded can be changed by using the specified point.
(For information on how to specify the loading condition, see the description of the sswon and sswoff commands.)

[Example]

tsp1 100

The execution of the instruction at 100h is specified to section point 1.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

tmode command

[Format]

tmode

[Parameter]

None

[Function]

The tmode command displays the trace setting status.

[Example]

The default status is shown below as an example:

```
>tmode
  Trace Settings (tron):
  Delay Count    = 0000ffff
  Trace Mode     = Real Time (real)
  Start Mode     = Force Start (force)
  Delay Mode     = Disable (!delay)
  Ext Trigger    = Disable (noext)
  TD1 Trigger    = Disable (!td1)
  TD2 Trigger    = Disable (!td2)
  TD3 Trigger    = Disable (!td3)
  TD4 Trigger    = Disable (!td4)
 Trace Settings (tenv):
  Sub switch     = <section> or  <qualify> (subor)
  DMA Trace      = Enable (dmatrc)
  SFR Trace      = Disable (!sfrtrc)
  Trace End BRK = Disable (!tendbrk)
  STOP Mode      = Enable (stop)
  Non-branch     = None (nonbranch0)
  Realtime       = 0 (rtrcb0)
  No Realtime    = 0 (nrtrcb0)
  PHOLD          = Disable (!phold)
  ONCE           = Disable (!once)
  Debug Mode     = Disable (!debug)
 Trace Switch Point Settings:
        Address   ASID
  tsp1 /del
  tsp2 /del
 Trigger Point Settings:
        Address   ASID
  tp /del
 Data Trace Settings:
        Address   A_Mask   ASID
  td1 /del
```

td2 /del
td3 /del
td4 /del

[Remark]
For the details of the trace, see Appendix A, "Details of Trace Functions".

tron command

[Format]

tron [DELAY] [[!]delay] [[!]real] [[!]force] [noext|posi|nega]
[[!]td{1|2|3|4}]

[Parameters]

DELAY = 0..1ffff delay counter
Specifies the number of frames in memory that are to be loaded in response to a trigger, in hexadecimal.

[!]delay: Specifies forced delay mode. Enter !delay to return to normal mode.
In forced delay mode, trace is forcibly terminated when the number of frames specified by the delay counter are traced after trace starts. In this mode, trigger events are ignored.

[!]real: Specifies the execution mode during trace. real specifies the real-time execution mode. The trace information may overflow in real-time execution mode. Enter ! to specify the non-real-time execution mode. An overflow does not occur in this mode, but the execution speed drops.

[!]force: Specifies the forced tracestart immediately after the tron command is issued as the trace start condition. Enter ! to cancel the forced start. In this case, trace is started according to the condition specified by tsp1. When forced start is specified, tsp1 and tsp2 are also valid.

noext|posi|nega: Specifies an external input pin (EXI0) as a trigger.
- noext: Does not use EXI0 as a trigger.
- posi: Uses the rising edge of EXI0 as a trigger.
- nega: Uses the falling edge of EXI0 as a trigger.

[!]td1: Specifies Trace Data Condition 1 (td1) as a trigger. ! clears the setting.

[!]td2: Specifies Trace Data Condition 2 (td2) as a trigger. ! clears the setting.

[!]td3: Specifies Trace Data Condition 3 (td3) as a trigger. ! clears the setting.

[!]td4: Specifies Trace Data Condition 4 (td4) as a trigger. ! clears the setting.

[Function]

The tron command clears the trace buffer and the settings of trace, and begins loading trace data.

[Examples]

tron

When tron is specified using the default values, trace is forcibly started and continues until forcibly terminated. Trace data displayed after a break shows the execution status until the execution immediately before the break.

tron delay 1ffff

Trace is started in the forced delay mode (delay=on) with using the default values for other parameters. Trace data in as many cycles as s pecified by the delay counter value (0x1ffff) is loaded immediately after the start of execution, and trace is automatically terminated. In the forced delay mode, trigger events are ignored.

td1 3ffb800 0

tron !delay td1 ffff

Trace is started when the condition of td1 is satisfied as a trigger point. !delay does not need to be specified if not changed. After the trigger condition is satisfied, trace data in as many cycles as the delay counter value (0xffff) is loaded, and trace is automatically terminated. As the result, trace data in each 0xffff cycles before and after the trigger point is loaded.

tp 1000
tron !delay ffff

Trace is started when the condition specified by tp is satisfied as the trigger point. !delay does not need to be specified if not changed. After the trigger condition is satisfied, trace data in as many cycles as the delay counter value (0xffff) is loaded, and trace is automatically terminated. As the result, trace data in each 0xffff cycles before and after the trigger point is loaded.

tsp1 1000
tsp2 2000
tp 1800
tron !force

As the trace packet loading condition, the value specified in the sswon command is used after the condition specified by tsp1 is satisfied and the value specified in the sswoff command is used after the condition specified by tsp2 is satisfied. By default, the sswon command specifies packet loading and the sswoff command specifies the stop of loading. According to this setting, trace loading is started immediately after the execution of the instruction at address 0x1000 specified by tsp1 and is temporarily stopped at the execution of the instruction at address 0x2000 specified by tsp2. During this period, the instruction at address 0x1800 specified by tp may be executed. In this case, the execution is used as the trigger point. As many packets as the delay cycle value (default value: 0xffff) are traced and loading is terminated.

tsp1 /del
tsp2 /del
tron force

tsp1 and tsp2 are canceled and trace is started in the forced start mode.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

troff command

[Format]

troff

[Parameter]

None

[Function]

The troff command forcibly terminates the loading of trace data.

trace command

[Format]

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

[Parameters]

POS=±0..1ffff:	Specifies the trace display start position in hexadecimal, assuming the vicinity of a trigger cycle or the ending cycle to be 0.
all|pc|data:	Specifies the cycle in loaded trace information that is to be displayed.
all:	All cycles
pc:	Execution cycles only
data:	Data cycles only
asm|ttag1|ttag2:	Specifies the display type.
asm:	Displays assembled listing.
ttag1:	Displays assembled listing and Time Tag in absolute time format.
ttag2:	Displays assembled listing and Time Tag in relative time format.
subNN:	Number of instructions to be disassembled in succession from an information item to actually be loaded (hexadecimal). The initial value is 80h (sub80).

[Function]

The trace command displays the contents of the trace buffer.

Issuing this command during trace forcibly terminates the loading process.

[Display]

```
> trace asm -5
    Cycle     Sub  Address       Code       Instruction              EXT   Stat
    -000006   ----   00:00001806 4200       mov    +00h,r8           0000  JMPD  JMP
    -000006   0001                00:00001808     8f260005          ld.w  +04h[r6],r17  0000
              SUB
    -000006   0002                00:0000180c     89e0              cmp   zero,r17      0000
              SUB
    -000004   ----   00:0000180e 259a       bne    00001850h         0000  JMPS  BcondNT
    -000002   ----   00:00001810 4f26000d   ld.w   +0ch[r6],r9       0000  JMPD  BcondNT
    -000002   0001                00:00001814     17860015          ld.bu +0014h[r6],r2 0000
              SUB
  * +000000   ----    --:00001818 5f260011   ld.w   +010h[r6],r11     0000  MATCH
    +000001   ----   00:0000181c 0df5       br     0000183ah         0000  JMPS  Bcond
    +000002   ----   00:0000183a 700b       mov    r11,r14           0000  JMPD  Bcond
    +000002   0001                00:0000183c     5a5f              add   -01h,r11      0000
              SUB

> trace ttag1
    Cycle     Sub  Address       Code       Instruction              EXT   Stat
    -000006   ----   00:00001806 4200       mov    +00h,r8           0000  JMPD  JMP
                                   time = 000,000,284,368.8uS
    -000006   0001                00:00001808     8f260005          ld.w  +04h[r6],r17  0000
              SUB
    -000006   0002                00:0000180c     89e0              cmp   zero,r17      0000
              SUB
    -000004   ----   00:0000180e 259a       bne    00001850h         0000  JMPS  BcondNT
                                   time = 000,000,284,368.9uS
    -000002   ----   00:00001810 4f26000d   ld.w   +0ch[r6],r9       0000  JMPD  BcondNT
```

```
                                    time = 000,000,284,368.9uS
   -000002  0001                    00:00001814     17860015        ld.bu  +0014h[r6],r2 0000
            SUB
 * +000000  ----    --:00001818  5f260011  ld.w    +010h[r6],r11    0000  MATCH
                                    time = 000,000,284,368.9uS
   +000001  ----    00:0000181c  0df5        br      0000183ah      0000  JMPS  Bcond
                                    time = 000,000,284,368.9uS

> trace ttag2
   Cycle     Sub  Address       Code        Instruction             EXT   Stat
   -000006   ----   00:00001806  4200        mov   +00h,r8         0000  JMPD  JMP
                                    time = 000,000,000,000.0uS
   -000006   0001                   00:00001808     8f260005        ld.w   +04h[r6],r17   0000
             SUB
   -000006   0002                   00:0000180c     89e0            cmp    zero,r17       0000
             SUB
   -000004   ----   00:0000180e  259a        bne   00001850h       0000  JMPS  BcondNT
                                    time = 000,000,000,000.1uS
   -000002   ----   00:00001810  4f26000d  ld.w    +0ch[r6],r9     0000  JMPD  BcondNT
                                    time = 000,000,000,000.0uS
   -000002   0001                   00:00001814     17860015        ld.bu  +0014h[r6],r2 0000
             SUB
 * +000000   ----    --:00001818  5f260011  ld.w    +010h[r6],r11   0000  MATCH
                                    time = 000,000,000,000.0uS
   +000001   ----   00:0000181c  0df5        br      0000183ah      0000  JMPS  Bcond
                                    time = 000,000,000,000.0uS
```

Cycle: Relative positions in the trace buffer are displayed in hexadecimal. The vicinity of the trigger point or the trace end frame is assumed to be 0.

Sub: Cycle numbers generated by analyzing branching and number-of-executed-instruction information.

Address: Execution addresses or bus cycle addresses are displayed.

Code: Instruction code or bus cycle data is displayed.

Instruction: Instruction mnemonics or bus types are displayed.

EXT: The states of external input pins EXI3 to EXI0 are displayed as bit strings.

Stat: The types of trace packets on which display is based are displayed.

TRGSTARTON	START packet is generated. Sub-switch is set to ON.
TRGSTARTOFF	START packet is generated. Sub-switch is set to OFF.
MATCH	MATCH packet is generated.
OVF	Overflow occurs.
TRCEND	TRCEND packet is generated.
JMPD <>	JMPD packet is generated. (< > will be explained later.)
JMPDS <>	JMPDS packet is generated. (< > will be explained later.)
JMPS <>	JMPS packet is generated. (< > will be explained later.)
OPCODE	Op code access (execution) occurs.
DATAW	Memory write occurs (trace packet).
DATAR	Memory read occurs (trace packet).
SFRW	SFR write occurs (bus trace).
SFRR	SFR read occurs (bus trace).
DMAW	iRAM writing by DMA occurs (bus trace).
DMAR	iRAM reading by DMA occurs (bus trace).

SUB Sub-cycle

"<>" indicates the following character strings. It indicates an instruction or an event that has caused branch.

NMI/INT	By occurrence of interrupt
EXP/TRAP	By occurrence of exception
RETI	By corresponding instruction
JMP	By corresponding instruction
JR	By corresponding instruction
JARL	By corresponding instruction
BcondNT	By corresponding instruction
Bcond	By corresponding instruction
CALLT	By corresponding instruction
SWITCH	By corresponding instruction
DISPOSE	By corresponding instruction
CTRET	By corresponding instruction
STORE	When WithPC is specified for data trace
LOAD	When WithPC is specified for data trace
FSTART	Forced start of trace

* mark: Trigger point (may shift slightly).

time = Displays Time Tag

Remark The Time Tag reflects a value when the CPU outputs branch information. The output of branch information has some delay from the time of actual execution, and the delay is not constant. Thus, the measurement value of the Time Tag has some error. Especially, please ignore the measurement result immediately after the execution, as it is unreliable.

[Remark]

For the details of the trace, see Appendix A, "Details of Trace Functions".

ftrace command

[Format]

ftrace statpos endpos filname [trace_options]

[Parameters]

statpos:	Start trace position to be written into a file
endpos:	End trace position to be written into a file
filname:	Input a file name
trace_options:	The following parameters are available. The meaning is the same as for the trace command. [all|pc|data] [asm|ttag1|ttag2] [subNN]

[Function]

The ftrace command writes the contents of the trace buffer into a file.

[Note]

Carefully enter the parameters for this command because the command cannot be canceled once executed.

ver command

[Format]

ver

[Parameter]

None

[Function]

The ver command displays the version of RTE-V850E/GP1-IE.