APPENDIX D. KIT-V850E/PG2-IE INTERNAL COMMANDS

This appendix describes the KIT-V850E/PG2-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 KIT-V850E/PG2-IE and the debugger, causing either device to malfunction.

Command syntax

The basic syntax for the KIT-V850E/PG2-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.

env and ifromenv command

[Format]

```
env  [[!]auto] [[!][verify]] [jtag[xxx][.[yyy]]{M|K}] [[!]reset]
ifromenv [[!]tracecache]
```

[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[xxx][.[yyy]]{M|K}:	The frequency of a JTAG clock is specified in the unit of MHz or KHz. Any value can be specified between 125MHz from 10kHz. However, being set up is rounded by the following values below a specification value. Actual set value can be checked by display. 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] **Remark** Usually, please use it by initial value (in the case of this ICE, it is 12.5MHz).
[!]reset:	Specifies whether the RESET pin is to be masked. Enter ! if it is not to be masked.
[!]tracecache	It is specification whether to enable a trace display during execution. Enter ! if it is not to be set. Initial value is enabling.

[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 ifromenv command specifies the setting parameter for the trace display under execution.

The default values are as follows:

```
Probe:
CPU Settings:
 Auto Run     = ON (auto)
 JTAGCLOCK    = 12.5MHz
 Verify       = verify off (!verify)
Signals Mask:
RESET         = NO MASK (!reset)
```

[Examples]

env reset

RESET is 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".

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 KIT-V850E/PG2-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, KIT-V850E/PG2-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 be loaded 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:

[Function]

The sfr(sfr2) command sets and displays the value of the SFR register.
The SFR command is targeted at the internal register assigned to the fixed area.
SFR2 command is targeted at the internal registers (register for CAN controllers etc.) assigned to the relocatable area.
The register name which can be used by each command can be checked in inputting a command without a parameter.

[Examples]

sfr P1
The value of the P1 register is displayed.

sfr PM1 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.

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] [[!]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.
[!]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

Sub-switch is ORed with section and quality.

[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)
  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 trace start 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 specified 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).
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.

time command

[Format]

time

[Parameter]

None

[Function]

The time command displays the time as the result of execution time measurement. The execution time measurement timer is initialized each time the CPU starts execution and is keeping the time during the execution of the CPU. The measurement clock frequency is 50 MHz and the resolution is 20 ns. The time converted to ns units is displayed.

[Remark]

The measurement time includes the overhead times (several clocks) at the start of execution and breaks.

[Example]

```
>time
 Time = 6,600 (ns) (50.000000MHz) [Counter=0000014a]
                    |               |
                    |               |_Counter value (hexadecimal)
                    |_Measurement clock frequency (always 50 MHz)
```

ver command

[Format]

ver

[Parameter]

None

[Function]

The ver command displays the version of KIT-V850E/PG2-IE.