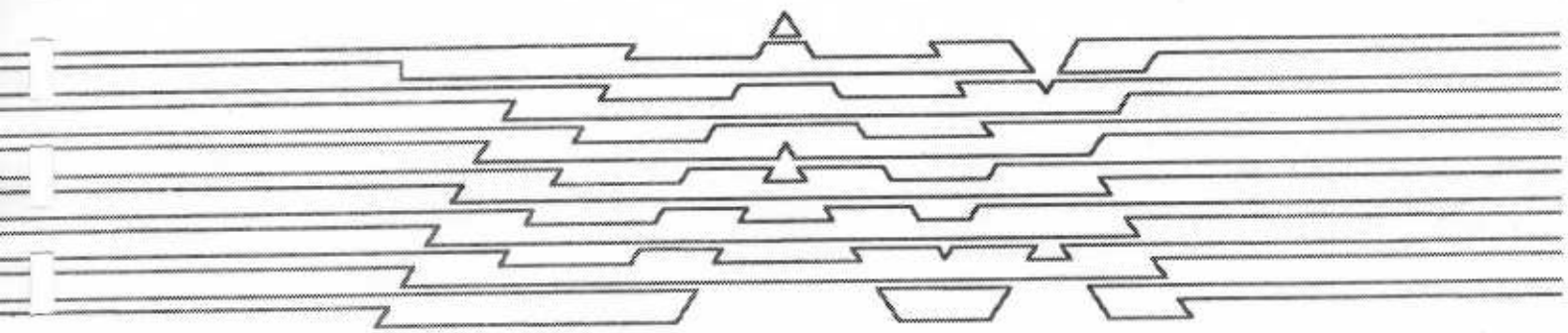
ERX318 for 68000

Command Manual



Zax Corporation

ERX318 for 68000 • 68010

Command Manual

Rev. E1.01

ERX318 for 68000 · 68010 Command Manual

CONTENTS

EVALUATION

O/S INTERFACE

LOGGING/JOURNAL

FILE/CONSOLE/INTERFACE

BATCH/MACRO CONTROL

BATCH/MACRO COMMAND

OTHERS

1. ERX START AND STOP

1.1 File Configuration

The basic file configuration contains the following three files :

a. ERX68K.EXE
b. ERX68K.HLP
c. ERX68K.MAC

EXE file	:	Contains the ERX control program
HLP file	:	Contains the ERX help messages
MAC file	:	Macros are loaded in this file when the control program is started. This happens only when the file name agrees with the control program name.

File names can be changed. If a file is to be renamed, all basic files must be renamed. When files are renamed, the default prompt characters are also modified.

1.2 How to Start

```
> program_name  [batch_name] < CR >
```

program_name	:	Specify the name of the file to be started.
batch_name	:	Specify the name of a batch file to be executed after the file is started.

If this parameter is omitted, the batch file is not executed.

NOTE

1.3 How to Stop

ERX > Q < CR >

ERX can be stopped by the Quit command.
The [Ctrl] + [C] keys are registered as the forced stop function.

NOTE

2. COMMAND INPUT FORMAT

2.1 Command Input Format

The first column to " >" is an ERX prompt message.
The input format is shown below.

```
X ··· X > CMD/MD    P1,P2,···,Pn < CR >
X ··· X > CMD/MD    P1 = P2 < CR >
X ··· X > CMD/MD    SCMD1,SCMD2,···SCMDn < CR >
X ··· X > CMD ; CMD < CR >
```

CMD : Specify the command name.
Specify the command name indicated in Chapter 7, "Command Functions". The lowercase characters contained in the command name can be omitted.

/MD : Specify a command modifier. There are some commands that have no modifier or more than one modifier.

P1~Pn : Specify the command parameter. This parameter is used to detail the command. If more than one parameter is specified, delimit them with commas [,] or space [SPACE].

P1 = P2 : Specify a command parameter. If two parameters have one meaning, specify the equal mark [=] to indicate the relationship.

; : More than one kind of command can be entered on one line. At this time, delimit them with semicolons [;].

SCMD1~SCMDn : Specify the subcommand name. This parameter specifies values specific to the command. If more than one subcommand is to be specified, delimit them with commas [,] or space [SPACE] ordinarily.

< CR > : Press the carriage return key. This key is the terminator for entry of the entered command.

NOTE

2.2 Command Parameter Input Format

Select a parameter that specifies address or data from among the following :

(1) Constant, variable

① Symbol

A label or constant name defined by the source program is called a symbol. (Symbols are defined by the EVent or Load command.) A symbol is an alphanumeric character string or a string of characters combined with _, # $.

② Symbol number (& xx)

When a symbol is defined, an index number is assigned automatically inside. (The index number can be seen by using the EShow command.)

Add & to the beginning of the symbol number.

Entered symbol numbers are converted into symbol names automatically.

③ Memory variable (MB/MW/ML)

The ERX program or user memory content is handled as a variable.

MB ({addr}) : Byte memory variable When a byte memory variable is specified, the content of the specified address is handled as a one byte variable.

MW ({addr}) : Word memory variable When a word memory variable is specified, the content of the specified address is handled as a one-word variable.

ML ({addr}) : Long word memory variable When a long word memory variable is specified, the content of the specified address is handled as a one long word variable.

④ Register variable ({REG})

The content of the emulation CPU register is handled as a variable.

REG ({reg}) : Register variable The content of the specified register is handled as a variable.

Any register name that can be specified by the ERX Register command can be specified.

NOTE

(2) Arithmetic operators

+	: Addition
−	: Subtraction
*	: Multiplication
/	: Division

Up to two terms can be used.

(3) Numerics

Numeric type can be determined by adding a numeric code to the numeric.

H	: Hexadecimal number	(0H~0FFFFFFH)
T	: Decimal number	(0T~0167772157T)
Y	: Binary number	(0Y~11111111111111111111111Y)
O	: Octal number	(0O~777777770)
Q	: Octal number	(0Q~77777777Q)

If a hexadecimal number begins with A to F, add 0 to the beginning of the numeric.
If the numeric code is omitted, the default is a hexadecimal number.

(4) Length specification

Length can be specified in the end addr parameter.
Add plus [+] to the beginning of the parameter.

2.3 Input Statement Modification

[BS]	: Cancels one character on the left hand of the cursor.
[DEL]	: Cancels one character in the cursor position.
[INS]	: Changes insertion/replacement mode. In insertion mode, the specified character is inserted immediately before the cursor. In replacement mode, the character in the cursor is replaced by the specified character.
[←]	: Moves the cursor by one column to left
[→]	: Moves the cursor by one column to right
[↑]	: Recall function (upward), which stocks 16 lines
[↓]	: Recall function (downward), which stocks 16 lines

NOTE

2.4 Special Console Input

The special console input function has the control and function keys. The function keys (F1 to F10) registered by the **FNKey** command can be used. The control keys are listed below.

Ctrl + S	:	Xoff is output and console display is interrupted.
Ctrl + Q	:	Xon is output and console display is restarted.
Ctrl + E	:	Recall function (upward), which stocks 16 lines
Ctrl + X	:	Recall function (downward), which stocks 16 lines
Ctrl + A	:	Recalls. Reinputs the entered command.
Ctrl + Y	:	Equivalent to the **SHEll** command
Ctrl + S	:	Equivalent to the **Quit** command
Ctrl + T	:	Aborts batch and macro file execution
Ctrl + C	:	Aborts ERX execution
ESC	:	Aborts **DISPlay** command execution
SPACE	:	Interrupts/restarts the **DISPlay** command

NOTE

2.5 Formula Input Format

A formula can be used in a macro or batch job.

(1) Constant, variable

① Symbol

A label or constant name defined by the source program is called a symbol. (A symbol is defined by the EVent or Load command.) A symbol is an alphanumeric character string or a string of characters combined with _, # $.

② Memory variable (MB/MW/ML)

MB ({addr }) : Byte memory variable

When a byte memory variable is specified, the content of the specified address is handled as a one - byte variable.

MW ({addr }) : Word memory variable

When a word memory variable is specified, the content of the specified address is handled as a one word variable.

ML ({addr }) : Long word memory variable

When a long word memory variable is specified, the content of the specified address is handled as a one long word variable.

③ Register variable (REG)

The content of the emulation CPU register is handled as a variable.

REG ({reg }) : Register variable

The content of the specified register is handled as a variable. Any register name that can be specified by the ERX Register command can be specified.

④ Internal variable (@)

The internal variable is an integer (32 bits) work variable used for assignment of calculation result and loop count.

@ {n } : Internal variable 0 to 15 can be specified.

⑤ Internal character variable (!)

The internal character is a work variable used for a character string output to the console or ERX command.

! {n } : Internal character variable 0 to 15 or up to 80 characters can be registered.

NOTE

⑥ File number variable (#)
The file number variable is a file assignment number used for file input/output.
{n } : File number variable One to eight can be specified.

⑦ Status variable (# STS)
The status variable is used for returning an execution result of the OPEN, CLOse, GET, PUT, or EOF command.
STS : Status variable (= 0 : normal termination, < >0 : abnormal termination or EOF)
Other variables are used as local variables according to the nesting levels of the macro or batch file, but this variable is used as a global one.

(Example 1) If the content of the memory indicated by symbol TSTFLG has the same content as symbol ON, symbol BEE is displayed on the console.

```
IF MB(TSTFLG)==ON<CR>
LET @1=BEE<CR>
CPUT @1<CR>
ENDI<CR>
```

(Example 2) If the memory content at address 625FH is at least three, the memory content at address 100H or 130H is dumped.

```
IF MB(625FH)>=3<CR>
D 100, 130<CR>
ENDI<CR>
```

(Example 3) The content of register A up to the address indicated by symbol HEXCHK - 1 is emulated repeatedly as a loop counter.

```
REPEAT REG(A)<CR>
G, HEXCHK-1<CR>
ENDR<CR>
```

NOTE

(Example 4) The content of internal variable @1 is assigned to register A. Emulation is repeated until the content of internal variable @1 is 0.

```
WHILE @1<>0<CR>
R A=@1<CR>
G , TEST-1<CR>
LET @1=@1-1<CR>
ENDW<CR>
```

(2) Operators

① Arithmetic operators

+	: Addition
-	: Subtraction
*	: Multiplication
/	: Division

② Bit operators

&	: AND
|	: OR
^	: XOR
~	: NOT

③ Comparison operators

==	: Equal
>	: More than
>=	: Equal or more than
<	: Less than
<=	: equal or less than
<>	: Not equal

Only monomial operators can be used.

(Example 1) An operation result is assigned to internal variable @8.

```
LET @8=10*15H<CR>
```

NOTE

(Example 2) A bit operation result is assigned to internal variable @4.

```
LET @4=MB(INTMSK) & 0FAH<CR>
```

(Example 3) Whether the content of symbol DMABUF is 0E3H is determined.

```
IF MB(DMABUF)==0E5H<CR>L
  :
  :
ENDI<CR>
```

(3) Numerics

Numeric type is determined by adding a numeric code to the end of the numeric.

H	: Hexadecimal number	(0H~0FFFFFFH)
T	: Decimal number	(0T~16777215T)
Y	: Binary number	(0Y~11111111111111111111111Y)
O	: Octal number	(0O~77777770)
Q	: Octal number	(0Q~77777777Q)

If a hexadecimal number begins with A to F, add 0 to the beginning of the numeric. If the numeric code is omitted, the default is a hexadecimal number.

NOTE

3. ERROR MESSAGES

3.1 Error Message Format

```
level : error message
```

level : There are four levels :

CTRL	Batch and macro control errors
ERX	O/S interface command errors
EMU	Emulation command errors
ASM	Assembler errors

error message : Error detail

NOTE

3.2 Batch and Macro Control Errors

CTRL : Argument number error.

The number of arguments is invalid.

CTRL : Batch file read error.

The batch file cannot be read.

CTRL : File close error.

The specified file cannot be closed.

CTRL : File open error.

The specified file cannot be opened.

CTRL : File open mode error.

Other than R or W is specified as **OPEN** command mode.

NOTE

CTRL : Get error.

The specified file cannot be read.

CTRL : Label maximum.

The number of labels used in a macro or batch file exceeds 20.

CTRL : Label not found.

The label name used by the **GOTO** command is missing.

CTRL : Nest error.

The combination of **IF**, **WHILE** and **REPEAT** is invalid.

CTRL : Nest overflow error.

The total number of **IF**, **WHILE** and **REPEAT** nesting levels exceeds 10.

NOTE

CTRL : Not opened error.

The **GET**, **PUT**, **EOF** or **CLOSE** command was executed for a file for which the open command had not been executed.

CTRL : Number variable error.

The numeric variable name is invalid.

CTRL : Put error.

Data cannot be written in the specified file.

CTRL : Reopen error.

An opened file was reopened.

CTRL : String variable error.

The character – string variable name is invalid.

NOTE

CTRL : Unknown item error.

The specified parameter item is invalid.

NOTE

3.3 O/S Interface Command Errors

ERX : Command error.

The command name is missing.

ERX : Command parameter error.

The parameter is invalid.

ERX : File close error.

The specified file cannot be closed.

ERX : File open error.

The specified file cannot be opened.

ERX : File read error.

The specified file cannot be read.

NOTE

ERX : File write error.

Data cannot be written in the specified file.

ERX : Help not found.

The specified help message is missing.

ERX : Macro not found.

The specified macro command is missing.

ERX : Not enough memory.

The main memory capacity is not enough.

NOTE

3.4 Emulation Command Errors

```
EMU : Check sum error in object file.
```

A check sum error occurred.

```
EMU : Say what ?!?  - Unknown command.
```

The specified command is missing.

```
EMU : Qualifier {command_text} is incorrect  - Check syntax.
```

The specified modifier is missing.

```
EMU : No such parameter {command_text}  - Check syntax.
```

The specified parameter is invalid.

```
EMU : Parameter error. {par} is greater than {par} - Swap or revise.
```

The specified parameter range is invalid.

NOTE

EMU : Parameter error. {par} is less than {par} – Swap or revise.

The specified parameter is invalid.

EMU : Syntax error – Re – enter command.

The specified command syntax is invalid.

EMU : You cannot use a wild card with this command – Be ····

The wild card function (* or ?) cannot be used by the specified command.

EMU : {address} can't be a don't care address !

Address specifying "Don't care (X)" cannot be used.

EMU : No more "COVERAGE" memory available for {address}.

64K bytes × 4 coverage memories are all allocated.

NOTE

EMU : Coverage status busy {status}.

The coverage status has already been set.

EMU : Event or symbol {address} references a duplicate address.

Event address is duplicated.

EMU : Emulation in progress – Type "STOP" first.

An unexecutable command was specified during emulation.

EMU : Emulation memory exceeded at {address}. Remap ····

The emulation memory capacity is not enough.

EMU : There's no emulation memory assigned.

The emulation memory is not mapped (allocated).

NOTE

EMU : User progam 'PC' error.

When emulation was started, the program started at odd address.

EMU : User program stack pointer error.

When emulation was started, a stack pointer at odd address was set.

EMU : Reached maximum definitions on this event - Redefine ...

The number of used channels exceeded the limit.

EMU : Event is not defined - Use EVENT command first.

The specified event is missing.

EMU : Event table is full - not enough host memory.

The event point storage memory capacity is not enough.

NOTE

EMU : {event_symbol} is not an event – Re – enter or define first.

The specified symbol has not been registered as an event.

EMU : External input in progress – Wait a moment.

The external input signal is in use.

EMU : Unable to close file – Check file name,number.

The object or event file cannot be closed.

EMU : Unable to open file – Check file name,mode,etc.

The object or event file cannot be opened.

EMU : Unable to read to file – Check file number,mode,etc.

The object or event file cannot be read.

NOTE

EMU : Unable to write to file – Check file number,mode,etc.

Data cannot be written in the object or event file.

EMU : {function} is not a function.

The parameter function is invalid.

EMU : A memory bus error.

A bus error occurred.

EMU : An illegal memory access was attempted.

Words cannot be accessed from odd address.

EMU : Illegal record in object file.

The object file contains an illegal record.

NOTE

EMU : Illegal address in object file.

The object file contains illegal address.

EMU : Memory guarded access error.

Space specified in the nonexistent memory was accessed.

EMU : Memory timeout error.

Memory access was not acknowledged within a definite time.

EMU : Memory verification error.

A memory verification error occurred.

EMU : Error in downloading object record.

The file does not contain the object record.
This message is also displayed for a file containing symbol records only.

NOTE

-EMU : {address} is not allocated as "COVERAGE" memory.

The coverage memory has not been set (allocated).

EMU : Not enough memory in your computer.

The main memory capacity of the host computer is not enough.

EMU : Exceeded maximum number of "ONBREAK" definitions allowed.

The number of on - break definitions exceeds the limit.

EMU : I can't find that "ONBREAK" !

The specified on - break is missing.

EMU : {address} is out of the "COVERAGE" address range.

When the coverage memory was set (allocated), the 64Kbyte boundary was exceeded.

NOTE

EMU : Out of event address range {address}.

When the address range was set by an event, the 64Kbyte boundary was exceeded.

EMU : Target memory access failure.

The target system whose power was off was accessed.

EMU : I can't find the symbol {symbol _text} – Check spelling.

The specified symbol is missing.

EMU : Illegal character used in symbol name – Restate.

The character cannot be registered as a symbol.

EMU : I can't place a STUB at that location !

The stub function was set in the impermissible memory.

NOTE

```
EMU : No STUB memory avalable.
```

The number of stub definitions exceeds the limits.

```
EMU : I can't find that STUB  – Redirect me.
```

The specified stub is missing.

```
EMU : Fatal system error  – MAJOR PROBLEM !!!
```

An error occurred in the ERX program.

```
EMU : Target fault  – Check power to target.
```

The power of the target processor is off.

```
EMU : Target fault  – Check interface or power.
```

An error occurred in the target processor.

NOTE

3.5 Assembler Errors

ASM : Address error.

The address is invalid.

ASM : Division by zero.

There is a division by zero.

ASM : Duplicate label

The entered symbol has already been defined.

ASM : Event point storage full.

There is no space for storing event symbols.

ASM : Format error.

The operation code and operand format are invalid.

NOTE

ASM : Illegal operand.

The operand is invalid.

ASM : Immediate error.

The number of digits of the numeric exceeds the limit (eight digits).

ASM : Invalid opcode.

The character string cannot be acknowledged as an operation code (command).

ASM : Nesting error.

The number of nesting levels (parentheses) is invalid.

ASM : Not terminated error.

The command format is valid but an excess character is added at its end.

NOTE

ASM : Syntax error.

The operator format is invalid.

ASM : System error.

The ERX program is abnormal.

ASM : Undefined symbol.

An undefined symbol was entered.

ASM : Unknown opcode.

An nonexistent operation code was entered.

NOTE

4. MACRO FUNCTION

The macro function combines registered commands and creates compound commands (which are called macro commands) and registers them in ERX. Control commands are available for macro commands, so the user can create custom executable files for specific applications.

4.1 Macro Control command Types

Table 4 – 1 lists the macro control commands.

Type	Command name
File control command	OPEN/CLOSE GET/PUT CGET/CPUT EOF
Bell control command	BEEP
Screen control command	ECHO
Shell control command	SHELL/EXECUTE
Sequence control command	IF/ELSE/ENDI REPEAT/ENDR WHILE/ENDW PAUSE LOOPOUT LET
Key control command	KEY/FNKEY
Comment control command	REM
Macro control command	MSHOW/MDELETE MSAVE/MLOAD

Table 4 – 1

NOTE

NOTE

5. BATCH FUNCTION

The batch function enables automatic execution of ERX commands from an MS－DOS ASCII file. Control commands are available for macro commands, so the user can create custom executable files for specific applications.

5.1 Batch Control command Types

Table 5－1 lists the macro control commands.

Type	Command name
File control command	OPEN/CLOSE GET/PUT CGET/CPUT EOF
Bell control command	BEEP
Screen control command	ECHO
Shell control command	SHELL/EXECUTE
Sequence control command	IF/ELSE/ENDI PAUSE LOOPOUT LET
Log control command	JOURNAL/NOJOURNAL LOG/NOLOG
Key control command	KEY/FNKEY
Comment control command	REM

Table 5－1

NOTE

NOTE

6. EVENT FUNCTION

6.1 Event Display Contents

```
ERX68K>es
 No.    Module  Symbol      FA Address St Size Data Count EX SEQ B C H P T
&1      ENTRY   ENTRY       -- 002000  -- ---- ---- ----  --  --  - - - - -
&2      ENTRY   LSTOP       -- 002020  -- ---- ---- ----  --  --  - - - - -
&15     sievex  .main       -- 002028  -- ---- ---- ----  --  --  - - - - -
&9      sievex  #14         -- 002034  -- ---- ---- ----  --  --  - - - - -
&10     sievex  #15         -- 00203A  -- ---- ---- ----  --  --  - - - - -
&11     sievex  #16         -- 00203E  -- ---- ---- ----  --  --  - - - - -
&12     sievex  #17         -- 002042  -- ---- ---- ----  --  --  - - - - -
&13     sievex  #18         -- 002074  -- ---- ---- ----  --  --  - - - - -
&14     sievex  #19         -- 002078  -- ---- ---- ----  --  --  - - - - -
&3      sievex  #20         -- 002098  -- ---- ---- ----  --  --  - - - - -
&4      sievex  #21         -- 0020AE  -- ---- ---- ----  --  --  - - - - -
&5      sievex  #23         -- 0020BE  -- ---- ---- ----  --  --  - - - - -
&6      sievex  #24         -- 0020DA  -- ---- ---- ----  --  --  - - - - -
&7      sievex  #26         -- 0020F6  -- ---- ---- ----  --  --  - - - - -
&8      sievex  #27         -- 002106  -- ---- ---- ----  --  --  - - - - -
&16     sievex  #31         -- 002138  -- ---- ---- ----  --  --  - - - - -
&18     sievex  #32         -- 002148  -- ---- ---- ----  --  --  - - - - -
&22     sievex  .OUTCHR     -- 00214E  -- ---- ---- ----  --  --  - - - - -
&19     sievex  #14         -- 002158  -- ---- ---- ----  --  --  - - - - -
&20     sievex  #15         -- 002160  -- ---- ---- ----  --  --  - - - - -
&21     sievex  #16         -- 002168  -- ---- ---- ----  --  --  - - - - -
&23     inchrw  #13         -- 0023DE  -- ---- ---- ----  --  --  - - - - -
 No.    Module   Symbol     FA Address St Size Data Count EX SEQ B C H P T
&26     inchrw  .INCHERW    -- 0023DE  -- ---- ---- ----  --  --  - - - - -
&24     inchrw  #14         -- 0023E2  -- ---- ---- ----  --  --  - - - - -
&25     inchrw  #15         -- 0023E4  -- ---- ---- ----  --  --  - - - - -
&27     SBRK    .sbrk       -- 0023EA  -- ---- ---- ----  --  --  - - - - -
&17     sievex  .flags      -- 009122  -- ---- ---- ----  --  --  - - - - -
ERX68K>
```

Figure 6－1

NOTE

No. : Number registered automatically when an event is registered. Registered event symbols can be used by functions. If this registration number is specified, it is converted into module and symbol names automatically.

Module
Symbol : For a high-level language, module and symbol names can be specified. When an object program is loaded, the symbols in the source file are registered as events automatically. Event information can be added, changed, and deleted freely.
When a program is to be debugged on the basic level, event information can be added by only symbol names because default module names are registered. If the Break command is used, event information is added by the default module name and symbol name {"--" + serial number}.
Note that the event information is not deleted automatically.
If an event symbol name is to be specified, module and symbol names must be specified. Place a period (.) between module and symbol names.

event symbol = {module-name}.{symbol-name}

Note : If module name is not specified in the event symbol parameter, the default module name is set automatically.

FA : Function address information is displayed. The parameters have the following meanings :

SP	: Supervisor program memory
SD	: Supervisor data memory
UP	: User program memory
UD	: User data memory
P	: Program memory
D	: Data memory
S	: Supervisor memory
U	: User memory
--	: Don't care

NOTE

Address : Address information is displayed. Address information is classified into two types ; identity and range conditions. They are explained below.

(Identify condition) A=2000
..... Address .....
..... 002000 .....

Information is displayed on one line.

(Range condition) A=2000, 2FFF
..... Address .....
..... 002000 .....
002FFE

Information is displayed on two lines.

(Invalid) A=1XXXXX
..... Address .....
..... 1XXXXX .....

Information is displayed on one line.

Note : If x is contained in address information, the digits (four bits) means "Don't care".

St : Status information is displayed. The parameters have the following meanings :

M	:	Memory access
MR	:	Memory read access
MW	:	Memory write access
IA	:	Interrupt acknowledgment
--	:	Don't care

NOTE

Size : Data size information is displayed. The parameters have the following meanings :

BYTE	: Byte count
WORD	: Word size
- - - -	: Don't care

Data : Data information is displayed. Data information is classified into two types ; identity and range conditions.

(Identity condition) D=1234

```
........ Data ........
........ 1234 ........
```

Data information is displayed on one line.

(Range condition) D=1000, 1FFF

```
........Data ........
........1000 ........
        1FFF
```

Data information is displayed on two lines.

(Invalid) D000, 0FFFF

```
........ Data ........
........      ........
```

Data information is displayed on one line.

NOTE

Count : Displays pass count information. Pass count information is classified into two types; specification and execution count information. Execution count information is displayed after at least once counting down is executed.

(No counting down) CNT=10

```
..... Count .....
..... 0010 .....
```

Count information is displayed on one line.

(Counting down)

```
..... Count .....
..... 0010 .....
      0007
```

Count information is displayed on two lines.

(Invalid)

```
..... Count .....
..... ____  .....
```

Count information is displayed on one line.

EX : External trigger information is displayed. The parameters have the following meanings :

HI	: High active signal
LO	: Low active signal
NE	: Negative edge signal
PE	: Positive edge signal
--	: Don't care

NOTE

SEQ : Sequential information is displayed.

B : Break function set state is displayed. The parameters have the following meanings :

*	: If the condition is met, emulation is broken.
–	: Don't care

C : Coverage function set state is displayed. The parameters have the following meanings :

*	: Used as the coverage condition
–	: Don't care

H : History function set state is displayed. The parameters have the following meanings :

S	: Used as the start trigger
E	: Used as the end trigger
–	: Don't care

P : Performance function set state is displayed. The parameters have the following meanings :

S	: Used as the start trigger
E	: Used as the end trigger
–	: Don't care

T : External trigger output set state is displayed. The parameters have the following meanings :

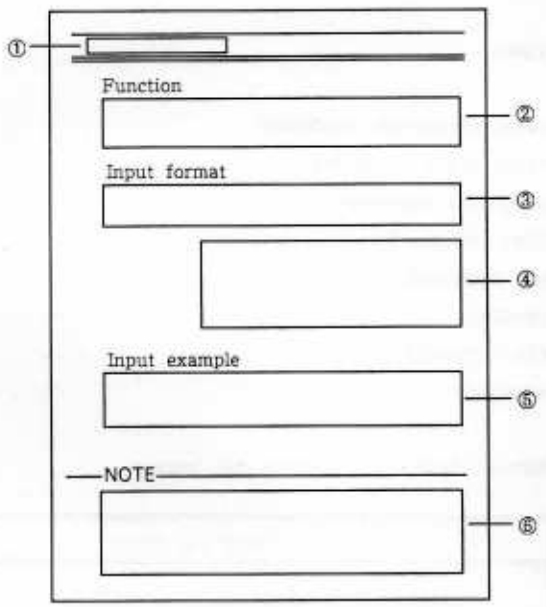
*	: Outputs the external trigger
–	: Don't care

NOTE

7. COMMAND FUNCTIONS

This chapter explains the commands in alphabetical order.
The commands are explained in the following format.

7.1 Command Function



①	Command name	: Name of the command to be input. Uppercase characters cannot be omitted.(Lower case characters can be omitted.)
②	Function	: Command function
③	Input format	: How to activate the command The command is indicated in abbreviation. A lowercase character string indicates a parameter.
④	Input parameters	: Lowercase characters in the input format.
⑤	Input example	: Actual input example
⑥	NOTE	: Note (s) on use

NOTE

7.2 Function Address Specification

Function address can be specified in the address input field of the command parameter. (Example : SP : 1000) The specifiable function address types and default function address types by command types are listed below.

Function address types

SP : Supervisor program memory
SD : Supervisor data memory
UP : User program memory
UD : User data memory
P : Program memory
D : Data memory
S : Supervisor memory
U : User memory

Default function address types by command types

Default value	Command name
SP or UP *1	ASSEMBLE,DISASSEMBLE,SAVE,STUB
SD or UD *2	COMPARE,DUMP,EXAMINE,FILL,MOVE,SEARCH
ANY	BREAK,EVENT,HISTORY,MAP

*1 If supervisor mode is used, SP is specified by the supervisor state flag of the status register. If user mode is used, UP is specified.
*2 If supervisor mode is used, SD is specified by the supervisor state flag of the status register. If user mode is used, UD is specified.

NOTE

Assemble

Function

The **Assemble** command converts assembler codes into machine codes and changes the memory contents.

Input format

```
> A  mem_addr < CR >
XXXXXX    {Assemble_code}  < CR >
XXXXXX    {Assemble_code}  < CR >
XXXXXX    < CR >
```

mem_addr	:	specify the change start memory address.
XXXXXX	:	The address of the memory to be changed by the line assembler is displayed.
{Assemble_Code}	:	Specify the mnemonic code used by the target processor.
< CR >	:	When only the Carriage Return (CR) key is pressed, the **Assemble** command is terminated.

Input example

```
ERX>A 1000
ERX>A &1
ERX>A MAIN. START
```

NOTE

The operand number representation system conforms to the assembler language of the target processor.

BAtch

Function

The **BAtch** function executes a batch file from a command line.

Input format

```
> BA    file_name < CR >
```

file_name : Specify the name of a batch file to be executed.
Its directory and device names can be specified in the file name.
If the extension is omitted, **.CMD** is the default.

Input example

```
ERX>BA TEST
ERX>BA TEST.CMD
```

NOTE

Up to ten levels of the **BAtch** command can be nested.
When the **BAtch** command is executed, the prompt characters become the batch file name.
Execution of the **BAtch** command on the current level can be stopped by control T.

BEEP

Function

The **BEEP** command outputs the bell code to the console.

Input format

```
> BEEP < CR >
```

Input example

```
MACRO>BEEP
```

NOTE

The **BEEP** command is used for confirmation in macro or batch processing or notification of termination.

The bell code is output irrespective of controlling the console by the **ECho** command.

Break

Function

The **Break** function displays the condition of the current break – point.

Input format

```
> B < CR >
```

Input example

```
ERX>B
```

NOTE

Break

Function

The **Break** function sets an event as a break condition.

Input format

```
> B    event_symbol [= switch] < CR >
```

event_symbol : Specify the symbolic name of the event used as a break condition.

- & When a symbol number is entered, it is converted into a symbolic name automatically.

switch : Specify a switch code.

- ON Sets the break condition
- OFF Releases the break condition

If this parameter is omitted, the default is **ON**.

Input example

```
ERX>B MAIN. START
ERX>B MAIN. END
ERX>B &1
ERX>B WORK. BUFF?
ERX>B PROG. *
ERX>B PROG. LOOP??=OFF
ERX>B *
ERX>B *=OFF
```

The wild card [*] [?] function can be specified in the symbolic name.

NOTE

Break

Function

The **Break** command sets address, status, and pass count as break conditions.

Input format

```
> B    address [,[status] [,passcount]] < CR >
```

address : Specify break address.

status : Specify a break status.

M	Memory
MR	Memory read
MW	Memory write
IA	Interrupt acknowledgement
ANY	Don't care

If this parameter is omitted, "Don't care" is the default.

passcount : Specify break pass count.
If this parameter is omitted, 1 is the default.

Input example

```
ERX>B 1000
ERX>B 1FFF, MR
ERX>B 0E000, MW, 1F
ERX>B 0EFFF, , 65535T
ERX>B 0XX55, M
```

NOTE

If the **Break** command is set, a unique event name (&&. n) is registered automatically.
Four - bit "Don't care" using X can be specified in the address parameter.

Break

Function

The **Break** command sets break points for all event symbols defined in the memory space in the specified range.

Input format

```
> B/M    beg_addr,end_addr [= switch] < CR >
```

M	:	Specifies the multi point break function
beg_addr	:	Specify the break range begin address.
end_addr	:	Specify the break range end address.
switch		ON The break conditions are set in the hardware.
		OFF The break conditions are released from the hardware.
		If this parameter is omitted, ON is the default.

Input example

```
ERX>B/M PROG. LOPP10, PROG. LOOP50
ERX>B/M MAIN. START, MAIN. END=OFF
ERX>B/M 1000, 1FFF
ERX>B/M &1, &5=OFF
```

NOTE

Break

Function

The Break command sets the break function specifying the address range.

Input format

```
> B/R    beg_addr,end_addr [,[status] [,passcount]] < CR >
```

Range	:	Specifies the range break function
beg_addr	:	Specify the lower limit of the break address range.
end_addr	:	Specify the upper limit of the break address range.
status	:	Specify a break status.
		M Memory access
		MR Memory read
		MW Memory write
		IA Interrupt acknowledge
		ANY Don't care
		If this parameter is omitted, "Don't care" is the default.
passcount	:	Specify the break pass count.
		If this parameter is omitted, 1 is the default.

Input example

```
ERX>B/R 1000, 1FFF
ERX>B/R 1000, 1FFF, , 3
ERX>B/R 1000, 1FFF, , 65535T
ERX>B/R 1000, 1FFF, MR
ERX>B/R &1, &2, M, 2
ERX>B/R WORK. BUFF1, WORK. BUFF2-1, MW
ERX>B/R PROG. LOOP10, PROG. LOOP20, M
```

NOTE

If the break function is set, a unique event name ($ $. n) is registered automatically.

Calculate

Function

The **Calculate** command calculates numeric characters or symbol values using an expression and displays a result.

Input format

```
> C  expression < CR >
```

expression : Specify an expression.

+	Addition
−	Subtraction
*	Multiplication
/	Division
&	AND
|	OR
^	XOR
~	Complement

Input example

```
ERX>C 1000
ERX>C MAIN.START
ERX>C 1000H+1FFFH
ERX>C WORK.BUFF1+WORK.BUFF_SIZE
ERX>C 1000H*2
ERX>C 0E000H/1000H
ERX>C (1000H+0E000H)/3
```

NOTE

CGET

Function

The **CGET** command assigns a numeric or character string entered from the console into an internal variable or character variable.

Input format

```
> CGET    @n < CR >
> CGET    !n < CR >
```

@n : Specify an internal variable. (n = 0 to 15)
!n : Specify an internal character variable. (n = 0 to 15)

Input example

```
MACRO>CGET @1
MACRO>CGET !2
```

NOTE

CLock

Function

The **CLock** command selects a clock mode for the target processor.

Input format

```
> CL  [clock_mode] < CR >
```

clock_mode : Specify a clock mode.

0	External clock by TTL logic
1	10MHz internal clock
2	5MHz internal clock
3	2.5MHz internal clock

If this parameter is omitted, the clock mode is not changed.

Input example

```
ERX>CL 3
ERX>CL
Clock Mode is 3
0. External (TTL)
1. Internal (10MHz)
2. Internal (5MHz)
3. Internal (2.5MHz)
Select (0-3)? 1
```

NOTE

If the power of the target processor is on, external clock (TTL clock) mode is selected at initialization.
If not, internal clock (10MHz) mode is selected.
When the clock mode is switched over, the target processor is reset.
The external clock means the target system clock.
The internal clock means the clock generated by ERX.

CLOSE

Function

The **CLOSE** command closes the file with the specified number.

Input format

```
> CLOSE   # n < CR >
```

n : Specify a file number variable.
If 0 is returned as a status variable (# STS), it means that close process has terminated normally.

Input example

```
MACRO> CLOSE #1
```

NOTE

COmpare

Function

The **COmpare** command compares memories and displays inconsistent data.

Input format

```
> CO    beg_addr,end_addr,cmp_addr [,direction] < CR >
```

beg_addr : Specify the comparison begin memory address.
end_addr : Specify the comparison end memory address.
cmp_addr : Specify the address of the memory to be compared.
direction : Specify a comparison condition.

- UE — The beg addr parameter is specified in the user memory.
  The cmp addr parameter is specified in the emulation memory.
  Data is compared between the user and emulation memories.
- EU — The beg addr parameter is specified in the emulation memory.
  The cmp addr parameter is specified in the user memory.
  Data is compared between the emulation and user memories.

If this parameter is omitted, data is compared according to the current mapping state.

Input example

```
ERX>CO 1000, 1FFF, 1000
ERX>CO 1000, 1FFF, 1000, UE
ERX>CO &1, &2, &1, EU
ERX>CO WORK. BUFF, +WORK. BUFF1_SIZE, WORK. BUFF2
```

NOTE

COVerage

Function

The **COVerage** command sets the set event symbol address (including the range) and status as coverage conditions.

Input format

```
> COV    event_symbol = switch < CR >
```

event_symbol : Specify an event symbolic name.
switch : Specify a switch code.
ON The coverage conditions are set.
OFF The coverage conditions are released.

Input example

```
ERX>COV  *=ON
ERX>COV  MAIN.*=ON
ERX>COV  *=OFF
```

NOTE

The wild card [*] [?] function can be specified in the symbolic name.

COVerage

Function

The **COVerage** command displays the coverage conditions and coverage measurement result.

Input format

```
> COV [/pass]     [event_symbol] < CR >
```

pass : Specify a pass code.
- P — Passed symbols are displayed.
- U — Unpassed symbols are displayed.
- D — A detailed measurement result is displayed.

If this parameter is omitted, symbols are displayed irrespective of whether they are passed.

event_symbol : Specify the symbolic name of the event in which the coverage function is set.
If this parameter is omitted, all event symbols are specified.

Input example

```
ERX>COV
ERX>COV/U
ERX>COV/P MAIN.*
ERX>COV/D &1
```

NOTE

The wild card [*] [?] function can be specified in the symbolic name.

COVerage

Function

The **COVerage** command initializes the coverage measurement function.

Input format

```
> COV/CL   < CR >
```

CL : Specifies that the coverage function is to be initialized

Input example

```
ERX>COV/CL
```

NOTE

CPUT

Function

The **CPUT** command displays the content of an internal variable or internal character variable on the console.

Input format

```
> CPUT    var [,var....] < CR >
```

var : Specify a variable to be output to the console.
@ {n} Internal variable (n = 0 to 15)
! {n} Internal character variable (n = 0 to 15)
"strings" Character strings
More than one variable can be specified.

Input example

```
MACRO>CPUT @1
MACRO>CPUT !1
MACRO>CPUT "This is message \n"
MACRO>CPUT !1, "=", @1, "\n"
```

NOTE

DEFModule

Function

The DEFModule command sets the default module name when a symbol is input.

Input format

```
> DEFM    [module_name] < CR >
```

module_name : If an event symbol without module name is input, the module name specified by this command is used.
If this parameter is omitted, the current default module name is displayed.

Input example

```
>ERX>DEFM<CR>
>ERX>DEFM MAIN<CR>
```

NOTE

The initial default module name is $ $.

Disassemble

Function

The **Disassemble** command displays memory contents in disassemble mode.

Input format

```
> DI  [beg_addr] [,end_addr] < CR >
```

beg_addr : Specify the disassemble begin memory address.
If this parameter is omitted, the value of the current program counter is the default.

end_addr : Specify the disassemble end memory address.
If this parameter is omitted, memory contents are displayed in ten line assemble mode.

Input example

```
ERX>DI
ERX>DI ,+16
ERX>DI 1000,102F
ERX>DI 1000,+8
ERX>DI &1
ERX>DI MAIN.START
ERX>DI PROG.LOOP10,PROG.LOOP50
```

NOTE

If disassemble mode is disabled, ****is displayed in the command expansion field. The displayed disassembler codes conform to the assembler language of the target processor.

Use the **DISPlay** command to enable/disable symbolic display.

DISPlay

Function

The DISPlay command specifies whether symbols are displayed on the console.

Input format

```
> DISP  switch < CR >
```

switch : Specify a switch code.
- ON Symbols are displayed.
- OFF Symbols are not displayed.

: If this parameter is omitted, the current state is displayed.

Input example

```
ERX>DISP ON
ERX>DISP OFF
```

NOTE

If symbolic display is disabled, symbols are not displayed but the symbol in the input command can be used.

A symbolic name must be specified with uppercase and lowercase characters.

Dump

Function

The **Dump** command displays the memory contents in a byte or word hexadecimal number.

Input format

```
> D [/size]    beg_addr [,end_addr] < CR >
```

size : Specify a data size.
- B Byte access
- W Word access
- L Long word access

If this parameter is omitted, word access is the default.

beg_addr : Specify the display begin memory address.

end_addr : Specify the display end memory address.
If this parameter is omitted, 16byte display is the default.

Input example

```
ERX>D 0E000
ERX>D 0E000, +100
ERX>D/L 1000, 10FF
ERX>D WORK. BUFF1
ERX>D &1, &2
ERX>D WORK. BUFF1, +WORK. BUFF1_SIZE
ERX>D MW(1000)
ERX>D R(SSP), +4
ERX>D/B MW(R(SSP)+8), +20
```

NOTE

Dump

Function

The **Dump** command displays the memory contents at each symbol in a byte hexadecimal number.

Input format

```
> D/S    beg_addr [,end_addr] < CR >
```

beg_addr : Specify the display begin memory address.

end_addr : Specify the display end memory address.
If this parameter is omitted, the contents of a symbol memory are displayed.

Input example

```
ERX>D/S 0E000
ERX>D/S 0E000, +100
ERX>D/S 1000, 10FF
ERX>D/S WORK. BUFF1
ERX>D/S &1, &2
ERX>D/S WORK. BUFF1, +WORK. BUFF1_SIZE
ERX>D/S MW(1000)
ERX>D/S R(SSP), +4
ERX>D/S MW(R(SSP)+8), +20
```

NOTE

ECho

Function

The **ECho** command enables/disables screen output.

Input format

```
> EC    [switch] < CR >
```

switch : Specify a switch code.

ON	Enables screen output.
OFF	Disables screen output.(Only ERX messages are displayed.)
KILL	Disables screen output.(ERX messages are not displayed, either.)

If this parameter is omitted, the current state is displayed.

Input example

```
MACRO>EC OFF
MACRO>EC ON
```

NOTE

When another macro or batch file is activated from one macro or batch file, echo state continues.
Control returns to the echo state of the original (calling) macro or batch file.
If screen output is disabled by the **ECho** command, echo back screen output is enabled by using the **CPUT** command and key entry.

EDelete

Function

The EDelete command deletes events.

Input format

```
> ED    beg_addr [,end_addr] < CR >
```

beg_addr : Specify the event range begin address.
end_addr : Specify the event range end address.
If this parameter is omitted, only the even in the address specified by the beg addr parameter is deleted.

Input example

```
ERX>ED &1
ERX>ED MAIN. START
ERX>ED 1000, 1FFF
ERX>ED MAIN. START, MAIN. END
ERX>ED MAIN. *
ERX>ED *. LOPP??
ERX>ED WORK. BUFF*
ERX>ED *
```

NOTE

The wild card [*] [?] function can be specified in the symbolic name.

EMSelect

Function

The EMSelect command displays the set state of machine cycle operation for the current target system.

Input format

```
> EMS  [select] < CR >
```

select : Specify an emulation select switch.

A	When emulation is broken, the AS signal is output to the target system.
B	When emulation is broken, the R/W signal is output to the target system.
C	In a write cycle of the emulation memory, the DC signal is output to the target system.
D	In a read cycle of the emulation memory, the DC signal is output to the target system.
E	In a ready cycle of the emulation memory, the D0 to D15 signal is output to the target system.
F	When the emulation memory is accessed, the BERR signal is accepted.
G	When the emulation memory is accessed, the DTACK signal is accepted.
H	The double bus fault break signal is generated.
I	When a memory is accessed, the wait signal generated in the ICE is used.
J	When a memory is accessed, the time out signal is generated.
K	Number of clock pulses of the wait signal generated in the ICE
L	Number of clock pulses of the time out signal

NOTE

Input example

```
ERX>EMS
```

NOTE

EMSelect

Function

The **EMSelect** command displays the set state of machine cycle operation for the current target system.

Input format

```
> EMS select = switch < CR >
```

select : Specify an emulation select switch.

- A — When emulation is broken, the AS signal is output to the target system.
- B — When emulation is broken, the R/W signal is output to the target system.
- C — In a write cycle of the emulation memory, the DC signal is output to the target system.
- D — In a read cycle of the emulation memory, the DC signal is output to the target system.
- E — In a read cycle of the emulation memory, the D0 to D15 signal is output to the target system.
- F — When the emulation memory is accessed, the BERR signal is accepted.
- G — When the emulation memory is accessed, the DTACK signal is accepted.
- H — The double bus fault break signal is generated.
- I — When a memory is accessed, the wait signal generated in the ICE is used.
- J — When a memory is accessed, the time out signal is generated.
- K — Number of clock pulses of the wait signal generated in the ICE.
- L — Number of clock pulses of the time out signal

NOTE

switch : Specify a switch code.

DI	Disables the set
EN	Ignores the set
1-7	Sets the number of clock pulses of the wait signal
8-32K	Sets the number of clock pulses of the time out signal

Input example

```
ERX>EMS  C=EN
ERX>EMS  K=3
```

NOTE

If G and I are both enabled, a greater one among the numbers of wait state signals (set or generated) when the memory is used.
When this command is executed, the target system is reset automatically.

EOF

Function

The **EOF** command checks to see whether the file with the specified number reaches the end of file (EOF).

Input file

```
> EOF   # n < CR >
```

n : Specify the file number variable.

Input example

```
MACRO>EOF #1
```

NOTE

EOF can be determined by the # STS variable.

(1) # STS = = 0 indicates EOF.

(2) # STS < > 0 indicates non − EOF.

ESAve

Function

The **ESAve** command saves the current event information in a file.

Input format

```
> ESA  file_name < CR >
```

file_name : Specify the event file name.
If the extension is omitted, **.EVT** is the default.

Input example

```
ERX>ESA FILE
ERX>ESA FILE.EVT
ERX>ESA A:/FILE.EVT
```

NOTE

EShow

Function

The **EShow** command displays event set state.

Input format

```
>ES  [beg_addr [,end_addr]] <CR>
```

beg_addr : Specify the event range begin address.
If this parameter is omitted, the states of all events are displayed.

end_addr : Specify the even range end address.
If this parameter is omitted, the state of the event in the address specified by the beg addr parameter is displayed.

Input example

```
ERX>ES
ERX>ES MAIN.START
ERX>ES 1000,1FFF
ERX>ES MAIN.START,MAIN.END
ERX>ES MAIN.*
ERX>ES *.LOOP??
ERX>ES WORK.BUFF*
```

NOTE

The wild card [*] [?] function can be specified in the symbolic name.

EVent

Function

The **EVent** command sets the event conditions.

Input format

```
> EV  [event_symbol] [FA = func_addr] [A = beg_addr [,end_addr]]
[ST = status]  [SI = size]  [D = beg_data [,end _data]]
[CNT = passcount]  [EXT = external]  [SEQ = seqid] < CR >
```

event_symbol : Specify the event symbolic name to be set.
(1) If an existing event is to be used, change the event conditions.
(2) If a new event is to be used, set the event conditions.
If this parameter is omitted, a unique event symbolic name ($$.--n) is registered.

FA func_addr : Specify a event function address.

SP	Supervisor program memory
SD	Supervisor data memory
UP	User program memory
UD	User data memory
P	Program memory
D	Data memory
S	Supervisor memory
U	User memory
ANY	Don't care

If this parameter is omitted, "Don't care" is the default.

A beg_addr : Specify the event address.
If this parameter is omitted, all address spaces are set automatically.

end_addr : If the address range is specified, specify the address upper limit.
If this parameter is omitted, the identity conditions of the beg addr parameter are used.

NOTE

ST status : Specify the status of the event to be set.

M	Memory access
MR	Memory read
MW	Memory write
IA	Interrupt acknowledgement
ANY	Don't care

If this parameter is omitted, "Don't care" is the default.

SI size : Specify the event access size.

B	Byte access
W	Word access
ANY	Don't care

If this parameter is omitted, "Don't care" is the default.

D beg_date : Specify event data
If this parameter is omitted, "Don't care" is the default.

end_date : If the data range is specified, specify the address upper limit. If this parameter is omitted, the identity conditions of the beg addr parameter are used.

CNT passcount : Specify the event pass count.
If this parameter is omitted, 1 is the default.

EXT external : Specify one of the following external signals :

HI	High active signal
LO	Low active signal
NE	Negative edge signal
PE	Positive edge signal
OFF	External signals are disabled.

If this parameter is omitted, OFF is the default.

SEQ seqid : Specify this parameter if the event is used as a sequential trigger.
(# 1 to # 4, $ 1 to $ 4, OFF)
If OFF is specified, the specified sequential trigger can be released.
If this parameter is omitted, OFF is the default.

NOTE

Input example

```
ERX>EV A=0E000
ERX>EV A=0000, S=MR
ERX>EV A=0XX55, S=MW
ERX>EV PROG. LOOP10, A=1FFF
ERX>EV WORK. BUFF1, A=0E000, 0EFFF
ERX>EV WORK. FLAG, S=MW
ERX>EV PROG. LOOP20, C=10
ERX>EV WORK. BUFF1, D=55
ERX>EV WORK. FLAG, D=55, 0AA
ERX>EV EXT=PE
ERX>EV PROG. LOOP30, EXT=L0
ERX>EV PROG. LOOP40, SEQ=$1
ERX>EV PROG. LOOP50, A=1000, S=MR, D=0FF, C=8
ERX>EV WORK. BUFF*, S=MW
ERX>EV MAIN. *, C=1
ERX>EV *, EXT=0FF
ERX>EV *, SEQ=0FF
```

NOTE

When an existing event is to be changed, if each parameter is omitted, the already set value is the default.
4bit "Don't care" using X can be specified in the address parameter.

EVent

Function

The **EVent** command sets the event conditions.

Input format

```
> EV  [event_symbol] < CR >
Symbol    : event_symbol
  = [event_symbol] [ctrl_char] < CR >
Faddress  : func_addr  = [func_addr] [ctrl_char] < CR >
Address   : beg_addr,end_addr
  = [beg_addr [,end_addr]] [ctrl _char] < CR >
Status    : status  = [status] [ctrl_char] < CR >
Size      : size  = [size]  [ctrl_char] < CR >
Data      : beg_data,end_data
  = [beg_data  [,end_data]] [ctrl_char] < CR >
Passcount : passcount  = [passcount] [ctrl_char] < CR >
External  : external  = [external] [ctrl_char] < CR >
Sequential : seqid  = [seqid] [ctrl_char] < CR >
```

event_symbol : Specify the event symbolic name to be set.
(1) If an existing event is to be used, change the event conditions.
(2) If a new event is to be used, set the event conditions.
If this parameter is omitted, a unique event symbolic name ($$.--n) is registered.

func_addr : Specify event function address.

SP	Supervisor program memory
SD	Supervisor data memory
UP	User program memory
UD	User data memory

NOTE

P Program memory
D Data memory
S Supervisor memory
U User memory
ANY Don't care
If this parameter is omitted, "Don't care" is the default.

beg_addr : Specify the event address.
If this parameter is omitted, all address spaces are set automatically.

end_addr : If the address range is specified, specify the address upper limit.
If this parameter is omitted, the identity conditions of the beg_addr parameter are used.

status : Specify an event status.
M Memory access
MR Memory read
MW Memory write
IA Interrupt acknowledgement
ANY Don't care
If this parameter is omitted, "Don't care" is the default.

size : Specify an event access size.
B Byte access
W Word access
ANY Don't care
If this parameter is omitted, "Don't care" is the default.

beg_date : Specify event data.
If this parameter is omitted, "Don't care" is the default.

end_date : If the data range is specified, specify the address upper limit.
If this parameter is omitted, the identity conditions of the beg_addr parameter are used.

passcount : Specify the event pass count.
If this parameter is omitted, 1 is the default.

external : Specify an external signal.
HI High active signal
LO Low active signal

NOTE

NE Negative edge signal
PE Positive edge signal
OFF External signals are inhibited.

seqid : Specify this parameter if the event is to be used as a sequential trigger. (#1 to #4, $1 to $4, OFF)
Specify OFF if the specified sequential trigger is to be released.
If this parameter is omitted, OFF is the default.

ctrl_char : Specify an item control character.
, The same item is displayed.
- The preceding item is displayed.
/ The command is terminated.
If this parameter is omitted, the next item is displayed.

Input example

```
ERX>EV
Symbol     :            = MAIN.START
Faddress   : --         = SD
Address    : XXXXXX     = 1000
Status     : --         = MR
Size       : ----       = B
Data       : ----       = /
ERX>EV PROG.LOOP20
Symbol     : PROG.LOOP20
Faddress   : --         =
Address    : 001000     =0E000
Status     : --         =
Size       : BYTE       = W
Data       : ----       =
Passcount  : 0001       = 5
External   : --         =
Sequential : --         = $2
```

NOTE

When an existing event is to be changed, if each parameter is omitted, the already set value is the default.
4bit "Don't care" using X can be specified in the address parameter.

Examine

Function

The **Examine** command writes data in the specified address.

Input format

```
> E [/size] [/verify]    addr = data_line < CR >
```

size	:	Specify a data size.
		B Byte access
		W Word access
		L Long word access
		If this parameter is omitted, work access is the default.
verify	:	Specify a verification code.
		N Data is not verified.
		If this parameter is omitted, data is verified.
addr	:	Specify the change memory address.
data_line	:	Specify memory change data or data row. (Hexadecimal 8/16/32bit data or ASCII data of at most 32bytes)

Input example

```
ERX>E 1000=20
ERX>E &1=55
ERX>E WORK.FLAG=00
ERX>E/L WORK.BUFF1_SIZE=30004000
ERX>E/B/N 1000=20
ERX>E MW(1000)=55
ERX>E MW(0E000)=MB(1000)
ERX>E/B 1FFF=R(D0)
```

NOTE

If data row is assigned to the data line parameter, place a delimiter, comma [,] between hexadecimal and ASCII data.
Enclose ASCII data with quotation marks ['].
If more than one piece of data is specified, data to the specified number of pieces is written in the specified address.

Examine

Function

The **Examine** command changes the memory contents.

Input format

```
> E [/size] [/verify] [/read]    beg_addr < CR >
XXXXXX    XXXX = [data] [ctrl_char] < CR >
```

size	:	Specify a data size. B Byte access W Word access L Long word access If this parameter is omitted, word access is the default.
verify	:	Specify the verification code. N Data is not verified. If this parameter is omitted, data is verified.
read	:	Specify the read code. NOR The contents of the current memory are not displayed. If this parameter is omitted, the memory contents are displayed.
beg_addr	:	Specify the memory address to be changed.
XXXXXX	:	The memory address to be changed is displayed.
XXXX	:	The contents of the current memory are displayed. (If /B is specified, data is displayed by bytes. If /L is specified, data is displayed by long words.)
data	:	Specify memory change data. (Hexadecimal 8/16/32bit data or ASCII data of at most 32bytes) If this parameter is omitted, the memory contents are not changed.

NOTE

ctrl_char : Specify a control character.

, The same address and data are displayed.

^ The preceding address and data are displayed.

/ The command is terminated.

If this parameter is omitted, the next address and data are displayed.

Input example

```
ERX>E/B 1000
001000 B6 = 0
001001 C0 = 55,
001001 55 = 0AA
001002 00 =
001003 95 =
001004 02 =^
001003 95 =^
001002 00 =^
001001 AA =^
001000 00 = /
```

NOTE

EXECute

Function

The **EXECute** command executes the DOS command when ERX is in start state.

Input format

```
> EXEC    dos_command < CR >
```

dos_command : Specify the DOS command.

Input example

```
ERX>EXEC DIR *.LOG
ERX>EXEC TYPE FILE.SRC
```

NOTE

When the **EXECute** command is executed, the DOS command is displayed.
When a series of data related to the DOS command is displayed and processing ends, control returns to the ERX prompt character.

Fill

Function

The **Fill** command changes the specified memory area.

Input format

```
>F [/size] [/verify]  beg_addr,  [end_addr],fill_data < CR >
```

size	:	Specify a data size. B Byte access W Word access L Long word access If this parameter is omitted, word access is the default.
verify	:	Specify the verification code. N Data is not verified. If this parameter is omitted, data is verified.
beg_addr	:	Specify the change begin memory address.
end_addr	:	Specify the change end memory address. If this parameter is omitted, the byte count of the memory change data is the default.
fill_data	:	Specify the memory change data or data row. (Hexadecimal 8/16/32bit data or ASCII data of at most 32bytes)

Input example

```
ERX>F 1000, 1FFF, 0FF
ERX>F &1, &2, 0AA
ERX>F WORK. BUFF1, +WORK. BUFF1_SIZE, 0
ERX>F/L WORK. BUFF1, WORK. BUFF2-1, 0
ERX>F/B/N 1000, , 'This is message'
```

NOTE

If data row is assigned to the data line parameter, place a delimiter, comma [,] between hexadecimal and ASCII data.
Enclose ASCII data with quotation marks ['].

FNkey

Function

The **FNkey** command executes the command defined in the function key.

Input format

```
> FN    func_no < CR >
```

func_no : Specify the function key number in a decimal number ranging from 1 to 10.

Input example

```
ERX>FN 1
ERX>FN 2
```

NOTE

GET

Function

The **GET** command assigns one line data from the file with the specified number to the specified numeric variable.

Input format

```
> GET   # n,! n < CR >
```

# n	:	Specify the file number variable. (n = 1 to 8)
! n	:	Specify the internal character variable. (n = 0 to 15) If the status variable (# STS) is 0, it indicates that the file could be accessed normally.

Input example

```
MACRO>GET #1, !1
```

NOTE

Go

Function

The **Go** command executes a user program.

Input format

```
> G [/wait] [beg _addr] [,end_addr_1] [,end_addr_2] < CR >
```

wait : Specify the wait code.
W After execution, the user program enters the wait state.
If this parameter is omitted, the user program does not enter the wait state after execution.

beg_addr : Specify the execution begin memory address.
If this parameter is omitted, the value of the current program counter is the default.

end_addr : Specify the execution end memory address.
If this parameter is omitted, the execution end memory address is not specified.
Up to two end addresses can be specified.

Input example

```
ERX>G
ERX>G 1000
ERX>G 1000, 1FFF
ERX>G &1
ERX>G MAIN. START
ERX>G MAIN. START, MAIN. END
ERX>G , MAIN. END
ERX>G , PROG. LOPP10, PROG. LOOP20
```

NOTE

Before emulation is executed, the hardware break (> B end_addr) function is set in the end_addr 1 and 2 parameters automatically and after emulation is broken, it is released automatically.
If there is no space for the break function, ERX displays the following error message without executing emulation :

EMU: Reached maximum definitions on this event.

If the end addr 1 and 2 parameters are omitted or the set break signal does not occur, emulation can be broken by the **STOP** command.

NOTE

The ERX break signal is generated after the command is executed, so the state after command execution is displayed.
During emulation, the plus [+] sign is displayed before the prompt character.
The following commands can be executed during emulation :
EVent. History. Performance. Pin. RESet. SCOPE. STOp. WAIT. and ICRESet.
All other commands are also executable but may cause errors because of hardware restrictions.

GOTO

Function

The **GOTO** command moves macro or batch processing control to the label specified by the parameter of the **GOTO** command.

Input format

```
> GOTO    label < CR >
```

label : Specify the label name in the position to which control is to be moved in the macro or batch process.

Input example

```
MACRO>GOTO LABEL
```

NOTE

HElp

Function

The **HElp** command explains the ERX command functions.

Input format

```
> HE    [command_name] < CR >
```

command_name : Specify a command name.
If this parameter is omitted, command names are listed.

Input example

```
ERX>HE
ERX>HE BREAK
ERX>HE HISTORY
ERX>HE STUB
```

NOTE

History

Function

The **History** command displays the trigger mode set conditions.

Input format

```
>H/S<CR>
```

S : Specifies that the trigger mode set status is to be displayed

Input example

```
ERX>H/S
```

NOTE

History

Function

The **History** command sets the trigger mode conditions.

Input format

```
>H  [S = event_symbol [,switch]] [E = event_symbol [,switch]]
[A = switch] [L = length] [M = switch] [F = switch] [B = switch] < CR >
```

S	:	Specify the trace start trigger event condition.
E	:	Specify the trace end trigger event condition.
A	:	A specifies auto start. If auto start is set, trace starts when emulation starts.
L	:	Specify the trace storage count condition. If the $ code is specified, the length condition is ignored. If the trace start trigger is specified the length condition is validated.
M	:	Specify the activation condition of the length con dition. ON The length condition activated by more than one start trigger is activated. OFF Only the length condition activated by the first start trigger is activated.
F	:	Specify the freeze conditions. If the freeze conditions are set, when all trace memories are full, trace stops automatically and the traced contents are guaranteed.
B	:	B specifies history break. If the number of storage times reaches 8191, emulation is broken.
event_symbol	:	Specify the set event symbolic name.
switch	:	Specify a switch code. ON The specified conditions are set. OFF The specified conditions are released. If this parameter is omitted, ON is the default.
length	:	Specify the number of trace storage times in a decimal number ranging from 1 to 8191. If the $ code is specified, the length condition is ignored.

NOTE

Input example

```
ERX>H S=MAIN. START
ERX>H S=MAIN. START, E=MAIN. END
ERX>H A=ON
ERX>H L=100
ERX>H A=ON, M=OFF
ERX>H A=OFF L=20 M=ON F=ON
ERX>H S=MAIN. START, E=*, OFF, A=OFF
ERX>H S=*, OFF, E=MAIN. END, L=$, M=OFF, F=OFF
ERX>H S=PROG. LOOP20, A=OFF, L=256, M=ON, F=OFF
ERX>H S=PROG. LOOP20, E=PROG. LOOP50, A=OFF, L=256, M=ON
ERX>H PROG. LOPP20, E=PROG. LOOP40, A=OFF, L=$, M=OFF, F=OFF
```

NOTE

History

Function

The History command sets the trigger mode conditions.

Input format

```
>H<CR>
Start Event   = [event_symbol [,switch]]  [ctrl_char] <CR>
End Event     = [event_symbol [,switch]]  [ctrl_char] <CR>
Auto Start    = [switch]  [ctrl_char] <CR>
Length        = [length]  [ctrl_char] <CR>
Multi         = [switch]  [ctrl_char] <CR>
Freeze        = [switch]  [ctrl_char] <CR>
Break         = [switch]  [ctrl_char] <CR>
```

Start_Event : Specify the event conditions of the trace start event trigger.

End_Event : Specify the event conditions of the trace end event trigger.

Auto_Start : This parameter specified auto start.
If auto start is set, when emulation is started, trace starts.

Length : Specify the number of traced data storage times.
If the $ code is specified, the length condition is ignored.
If the Start Event parameter is specified, the length condition is validated.

Multi : Specify the activation condition of the length condition.
- ON The length condition activated by more than on start trigger is activated.
- OFF Only the length condition activated by the first start trigger is activated.

Freeze : Specify the freeze condition.
If the freeze condition is set, when all trace memories are full, trace stops automatically and traced data is guaranteed.

Break : This parameter specifies history break.
If the number of storage times reaches 8191, emulation is broken.

NOTE

event_symbol : Specify the set event symbolic name.
switch : Specify a switch code
ON The specified conditions are set.
OFF The specified conditions are released.
If this parameter is omitted, ON is the default.
length : Specify the number of traced data storage times in a decimal number ranging from 1 to 8191.
If the $ code is specified, the length condition is ignored.
ctrl_char : Specify a control character.
, The same address and data are displayed.
^ The preceding address and data are displayed.
/ The command is terminated.
If this parameter is omitted, the next item is displayed.

Input example

```
ERX>H
Start Event        = MAIN.START
End  Event         = MAIN.END
Auto Start   : ON  = OFF
Length       : $   =
MultiE       : ON  =
Freeze       : OFF =
Break        : OFF =
```

NOTE

History

Function

The **History** command displays traced data storage size.

Input format

```
> H/SI < CR >
```

SI : Specify the traced data storage size.

Input example

```
ERX>H/SI
```

NOTE

History

Function

The **History** command displays the traced data in machine cycle or disassemble display mode.

Input format

```
> H/mode  [beg_point] [,end_point] [FA = func_addr] [A = address]
[ST = status] [SI = size] [D = data] [E = switch] < CR >
```

mode : Specify a display condition.
- M Machine cycle display mode
- D Disassemble display mode

FA func_addr : Specify the event function address.
- SP Supervisor program memory
- SD Supervisor data memory
- UP User program memory
- UD User data memory
- P Program memory
- D Data memory
- S Supervisor memory
- U User memory
- ANY Don't care

If this parameter is omitted, the function address condition is ignored.

A Address : Specify the address to be searched for.
If this parameter is omitted, the address condition is ignored.

D data : Specify the data to be searched for.
If this parameter is omitted, the data condition is ignored.

NOTE

ST status : Specify the status to be searched for.
- M Memory
- MR Memory read
- MW Memory write
- IA Interrupt acknowledgement
- ANY Don't care

If this parameter is omitted, the status condition is ignored.

SI size : Specify the access size to be searched for.
- B Byte access
- W Word access
- ANY Don't care

If this parameter is omitted, the access size condition is ignored.

E switch : Specify whether the trigger point is to be searched for.
- ON The trigger point is searched for.
- OFF The trigger point is not searched for.

If this parameter is omitted, OFF is the default.

beg_point : Specify the search begin point in a decimal number ranging from 1 to 8191.
If this parameter is omitted, the currently traced data storage size is the default.

end_point : Specify the search end point in a decimal number ranging from 1 to 8191.
If this parameter is omitted, 1 is the default.

Input example

```
ERX>H/M
ERX>H/M 100
ERX>H/M 1000, 1
ERX>H/M 1000, 1, A=0E000, 0EFFF
ERX>H/M A=0E000, 0EFFF
ERX>H/M ST=IA
ERX>H/M A=0XX55 ST=MW
ERX>H/M 2000, , A=WORK. BUFF1, ST=MW, D=OFF
ERX>H/M E=ON
```

NOTE

When machine cycle display mode is set, traced data in the cycles that satisfy the condition in the specified range is displayed.
When disassemble display mode is set, traced data from the cycle that satisfies the condition to the last one in the specified range is displayed in disassemble mode.
4byte "Don't care" can be specified using X in the address parameter.
The read modification write cycle is displayed as RMW.

ICERESet

Function

The **ICERESet** command stops emulation state forcedly.

Input format

```
> ICERES < CR >
```

Input example

```
ERX>ICERES
ERX+>ICERES
```

NOTE

If the target system operates abnormally, the **STOp** command may be unexecutable. If so, the **ICERESet** command can stop the target system forcedly.
If the **ICERESet** command is used, the target processor is reset forcedly, so this causes the same state as when the RESet command is executed.

IDentification

Function

The **IDentification** command displays the ERX model and software version.

Input format

```
> ID < CR >
```

Input example

```
ERX>ID
```

NOTE

IF

Function

The IF command evaluates an expression. If the result is true, the IF command executes the commands up to ELSE or ENDI.
If the result is false, the IF command executes the commands from ELSE to ENDI.

Input format

```
IF  {exp} < CR >
{command] < CR >
    :
ELSE < CR >
{command} < CR >
    :
ENDI < CR >
```

{exp}	:	Specify an expression.	
		{symbol}	Symbolic constant
		MB ({addr})	Byte memory variable
		MW ({addr})	Word memory variable
		ML ({addr})	Long word memory variable
		REG ({reg})	Register variable
		@ {n}	Internal variable
		! {n}	Internal character variable
		# STS	Status variable
		+, −, *, /	Operators
		&, |, ^, ~	Operators
		= =, >, > =, < =, < >	Operators
{command}	:	The ERX command can be entered.	

NOTE

Input example

```
MACRO>IF #STS == 0
MACRO> GET #1,!1
MACRO>ELSE
MACRO> LET @1=0
MACRO>ENDI
```

NOTE

Journal

Function

The **Journal** command catalogs the commands and parameters entered from the keyboard during operation in the specified disk file sequentially.

Input format

```
> J  file_name < CR >
```

file_name : Specify the catalog file name.
If the extension is omitted, **.CMD** is the default.

Input example

```
ERX>J TEST
ERX>J TEST.CMD
```

NOTE

If cataloging in the disk file is to be ended, use the **NOJornal** command.

Key

Function

The **Key** command assigns a command to a function key.

Input format

```
> K    fun_name = command < CR >
```

fun_name : Specify a function key number in a decimal number ranging 1 to 10.

command : Specify the command to be assigned.

Input example

```
ERX>KEY 1=R RES\n
ERX>KEY 2="R RES;G\n"
ERX>KEY 3=E 1000=
```

NOTE

If " \n" is added to the end of the command, the command can be assigned as an executable command.

If a command assigned to a function key is to be executed, press the function key (F1 to F10) or use the **FNkey** command.

LET

Function

The **LET** command calculates an expression represented by exp and assigns the result to the variable represented by var.

Input format

```
> LET  var = exp   < CR >
```

var	:	Specify the name of the variable to be assigned.	
		@ {n}	Internal variable
		! {n}	Internal character variable
exp	:	Specify an expression.	
		{symbol}	Symbolic constant
		MB ({addr})	Byte memory variable
		MW ({addr})	Word memory variable
		ML ({addr})	Long word memory variable
		REG ({reg})	Register variable
		@ {n}	Internal variable
		! {n}	Internal character variable
		+, −, *, / &, |, ^, ~	Operators

Input example

```
MACRO>LET @1=1000
MACRO>LET @2=MB(1000)
MACRO>LET !1="This is message.\n"
```

NOTE

Load

Function

The Load command loads an object program.

Input format

```
> L [/format]     [file_name] [,event_file_name] < CR >
```

format : Specify a format type.
- I Intel hex format
- M Motorola hex format
- D Dump command image format

If this parameter is omitted, the format conforms to the standard assembler format.

file_name : Specify the name of the object program to be loaded.
If the extension is omitted, .**ABS** is the default.
If this parameter is omitted, the object program is not loaded.

event_file_name: Specify the name of the event file to be loaded.
If the extension is omitted, .**EVT** is the default.
If this parameter is omitted, the event file is not loaded.

Input example

```
ERX>L SAMPLE
ERX>L SAMPLE. ABS
ERX>L A:\SAMPLE
ERX>L/I SAMPLE. ABS
ERX>L/M SAMPLE. ABS
ERX>L/D SAMPLE. ABS
ERX>L , EVENT. EVT
ERX>L SAMPLE. ABS, EVENT. EVT
```

NOTE

Object file format	intel – HEX S – format
Symbolic file format	ZAX – format
Event setting	For already set event symbols, only the event conditions are updated.
Start Address display	The value of the start address code in the object file is displayed and set in the program counter (PC).

LOG

Function

The **LOG** command catalogs all operation contents sequentially in the specified disk file.

Input format

```
LOG    file_name < CR >
```

file_name : Specify the catalog file name.
If the extension is omitted, .**LOG** is specified.

Input example

```
ERX>LOG TEST
ERX>LOG TEST.LOG
```

NOTE

If cataloging in the disk file is to be terminated, use the **NOLog** command.
More than one disk file cannot be loaded.

LOOPOUT

Function

The **LOOPOUT** command moves control out of the inner most loop controlled by the **WHILE** or **REPEAT** command.

Input format

```
> LOOPOUT < CR >
```

Input example

```
MACRO>LOOPOUT
```

NOTE

MACro

Function

The **MACro** command catalogs a new macro.

Input format

```
> MAC    mac_name < CR >
MACRO > [command] < CR >
```

mac_name : Specify the macro name.
command : Specify the command.
If this parameter is omitted, cataloging **MACro** commands is terminated.

Input example

```
ERX>MACRO DEBUG
MACRO>REM Sample Macro File
MACRO>EC OFF
MACRO>OPEN #1,%1,R
MACRO>IF #STS == 0
MACRO> OPEN #2,%2,W
MACRO> WHILE #STS == 0
MACRO> GET #1,!1
MACRO> EOF #1
MACRO> IF #STS == 0
MACRO>  PUT #2,!1
MACRO> ENDI
MACRO> ENDW
MACRO>CLOSE #2
MACRO>ENDI
MACRO>CLOSE #1
MACRO>
```

NOTE

A macro can use nine parameters form %1 to %9 as variables.
These variable parameters must be specified when a macro is to be activated.
The format is as follows :

> mac_name [Value_1 [,Value_2 [...]]] < CR >
Value_n Specify the numeric corresponding to %n.

MAp

Function

The MAp command displays the current mapping state.

Input format

```
> MA    [beg_addr] [,end_addr] < CR >
```

beg_addr : Specify the display begin address.
If this parameter is omitted, 0 is the default.

end_addr : Specify the display end address.
If this parameter is omitted, 0FFFFFF is the default.

Input example

```
ERX>MA
ERX>MA 0E000, 0EFFF
```

NOTE

MAp

Function

The MAp command, which sets a memory map, allocates each memory to a 4Kbyte space.

Input format

```
> MA    beg_addr  [,end_addr] = area < CR >
```

beg_addr	:	Specify the mapping specification begin address.
end_addr	:	Specify the mapping specification end address. If this parameter is omitted, each memory is allocated to a 4Kbyte space.
area	:	Specify an area code.
		RO The emulation memory is specified as a read only memory.
		RW The emulation memory is specified as a read/write memory.
		US The target system memory is specified.
		NO No memory is allocated.

Input example

```
ERX>MA 0E000=US
ERX>MA 1000, 1FFF=RO
ERX>MA 0, 0FFFF=RW
```

NOTE

If the end_addr parameter is omitted, 4Kbytes from the value specified by the beg_addr parameter is the default.

If the emulation memory is smaller than the address space of the target processor, auto allocation is executed.

In mapping initialization, all address spaces are US.

MDelete

Function

The MDelete commands deletes a cataloged macro.

Input format

```
> MD    macro_name < CR >
```

macro_name : Specify the macro name.

Input example

```
ERX>MD DEBUG
```

NOTE

MLoad

Function

The **MLoad** command loads a macro file and catalogs macros automatically.

Input format

```
> ML    file_name < CR >
```

file_name : Specify the macro file name.
If the extension is omitted, **.MAC** is the default.

Input example

```
ERX>ML DEBUG
ERX>ML DEBUG.MAC
ERX>ML A:\MACRO \DEBUG.MAC
```

NOTE

MODlen

Function

The MODlen command displays the number of the module name.

Input format

```
> MOD < CR >
```

Input example

```
ERX>MOD
```

NOTE

MODlen

Function

The MODlen command specifies number of the module name characters to be displayed.

Input format

```
> MOD  number < CR >
```

number : Specify the number of display columns in a decimal number ranging from 0 to 31.

Input example

```
ERX>MOD 5
ERX>MOD 10
```

NOTE

Move

Function

The **Move** command moves data between the emulation and user memories. This command also moves data in the emulation or user memory.

Input format

```
> M [/verify] beg_addr,end_addr,mov_addr [,direction] < CR >
```

verify : Specify a verification code.
- N Data is not verified.

If this parameter is omitted, data is verified.

beg_addr : Specify the move begin memory address.

end_addr : Specify the move end memory address.

mov_addr : Specify the leading address of the destination memory.

direction : Specify the move condition.
- UE The value specified by the beg_addr parameter is specified in the user memory and the value specified by the mov_addr parameter is specified in the emulation memory. So, data is moved from the user to emulation memory.
- EU The value specified by the beg_addr parameter is specified in the emulation memory and the value specified by the mov_addr parameter is specified in the user memory. So, data is moved from the emulation to user memory.

If this parameter is omitted, data is moved in the current mapping state.

Input example

```
ERX>M 1000, 1FFF, 1000
ERX>M 1000, 1FFF, 1000, UE
ERX>M &1, &2, &1, EU
ERX>M MAIN. START, MAIN. END, MAIN. START, UE
ERX>M WORK. BUFF1, +WORK. BUFF_SIZE, WORK. BUFF_SIZE, WORK. BUFF2, EU
```

NOTE

MSAve

Function

The **MSAve** command saves the cataloged macro in a disk file.

Input format

```
> MSA  file_name < CR >
```

file_name : Specify the macro file name.
If the extension is omitted, **.MAC** is the default.

Input example

```
ERX>MSA DEBUG
ERX>MSA DEBUG.MAC
```

NOTE

MShow

Function

The **MShow** command displays the contents of all macros cataloged currently.

Input format

```
> MS    [mac_name] < CR >
```

mac_name : Specify the macro name.
If this parameter is omitted, the contents of all macros are displayed.

Input example

```
ERX>MS DEBUG
```

NOTE

NOJournal

Function

The **NOJournal** command terminates cataloging in a disk file by the **Journal** command.

Input format

```
> NOJ < CR >
```

Input example

```
ERX>NOJ
```

NOTE

NOLog

Function

The **NOLog** command terminates cataloging in a disk file by the **LOG** command.

Input format

```
> NOL < CR >
```

Input example

```
ERX>NOL
```

NOTE

ONbreak

Function

The **ONbreak** command displays on – break set state.

Input format

```
> ON < CR >
```

Input example

```
ERX>ON
```

NOTE

ONbreak

Function

The **ONbreak** command defines postprocessing if the event break signal is generated during ERX emulation.

Input format

```
> ON    event_symbol,"command"  < CR >
```

event_symbol : Specify the event symbolic name.
command : All ERX commands (including batch and macro) are usable.

Input example

```
ERX>ON MAIN.FLAG, "D/S MAIN.FLAG"
ERX>ON *.##, DI
ERX>ON *, "H/D"
```

NOTE

The wild card [*] [?] function can be specified in the symbolic name.
If on break is duplicated by using the wild card function, the specified last one has the priority.
Up to 32 on break points can be set.
When the **SUTB** command is activated, the **History** and **Performance** commands are stopped as when the **STOp** command is executed or the break signal is generated.

ONbreak

Function

The **ONbreak** command specifies on break.

Input format

```
> ON    address = switch < CR >
```

address	:	Specify the set on break symbol.
switch	:	Specify a switch code.
		ON The set on break function is validated.
		OFF The set on break function is ignored.
		CLR The set on break function is released.

Input example

```
ERX>ON MAIN.FLAG=ON
ERX>ON *,##=OFF
ERX>ON *=CLR
```

NOTE

OPEN

Function

The **OPEN** command opens the specified file.

Input format

```
> OPEN    # n,file_name,mode < CR >
```

# n	:	File number variable ranging from one to eight determined when the file is opened. This value is used for accessing the file. If −1 is returned to the status variable (# STS), it indicates that the file could not be opened normally.
file_name	:	Name of the file to be opened. A pass name can be specified.
mode	:	Specify open mode.

W Write mode access
The file is opened as a write − only file.
The contents of the file opened in this mode cannot be read by using the **GET** command.
Only the **PUT** and **CLOSE** commands can be used.

R Read mode access
The file is opened as a read − only file.
No data can be written in the file opened in this mode by using the **PUT** command.
Only the **GET**, **CLOSE** and **EOF** commands can be used.

Input example

```
MACRO>OPEN #1, FILE. SRC, R
MACRO>OPEN #2, A:\FILE. SRC, W
```

NOTE

PAUSE

Function

The **PAUSE** command interrupts macro or batch processing.
When a key on the keyboard is pressed, processing is restarted.

Input format

```
> PAUSE < CR >
```

Input example

```
MACRO>PAUSE
```

NOTE

PErformance

Function

The **PErformance** command displays an event for measuring performance.

Input format

```
> PE < CR >
```

Input example

```
ERX>PE
```

NOTE

PErformance

Function

The PErformance command sets the event set status for measuring performance.

Input format

```
> PE   S = event_symbol [,switch]  E = event_symbol [,switch] < CR >
```

S	:	Specify the start trigger for measuring performance.
E	:	Specify the end trigger for measuring performance.
event_symbol	:	Specify the set event symbolic name.
switch	:	Specify a switch code.
		ON The specified event is set.
		OFF The specified event is released.
		If this parameter is omitted, ON is the default.

Input example

```
ERX>PE S=&1
ERX>PE S=MAIN.START
ERX>PE S=MAIN.END
ERX>PE S=MAIN.START, E=MAIN.END
ERX>PE S=*, OFF
ERX>PE E=MAIN.*, OFF
```

NOTE

PErformance

Function

The PErformance command displays the result of performance measurement.

Input format

```
> PE/D < CR >
```

D : Specifies that the result of performance measurement is to be displayed

Input example

```
ERX>PE/D
```

NOTE

PIn

Function

The **PIn** command displays the state of the input signal from the current target system and mask set state.

Input format

```
> PI  [signal] < CR >
```

signal : Specify a target system signal.

BR	Bus request signal
HALT	Halt signal
BERR	Bus error signal
RESET	Reset signal
L7-L1	Interrupt signal

If this parameter is omitted, the states of all pins that can be monitored are displayed.

Input example

```
ERX>PI
ERX>PI BERR
ERX>PI RESET
```

NOTE

The [*] code in the displayed data indicates a negative logical signal.

PIn

Function

The **PIn** command masks/unmasks a target system control signal.

Input format

```
> PI signal = switch < CR >
```

signal	:	Specify a target CPU signal.
		BR Bus request signal
		HALT Halt signal
		BERR Bus error signal
		RESET Reset signal
		L7-L1 Interrupt signal
		ALL All signals are masked.
switch	:	Specify a switch code.
		DI Pins are disabled (masked).
		EN Pins are enabled (unmasked).

Input example

```
ERX>PI ALL=EN
ERX>PI RESET=DI
ERX>PI L7=DI
```

NOTE

If the power of the target system is on, the pin mask initial value is "enable" (unmasked) ; if the power is off, it is "disable" (masked).

PROmpt

Function

The **PROmpt** command renames the prompt character.

Input format

```
> PRO    pro_char < CR >
```

pro_char : Specify the new prompt characte.

Input example

```
ERX>PRO ERX
```

NOTE

PUT

Function

The **PUT** command outputs one line data to the file with the specified number.

Input format

```
> PUT   # n,! n < CR >
```

# n	:	Number of the file to which data is to be output Use a file number variable. (n = 1 to 8)
! n	:	Specify the internal character variable. (n = 0 to 15) If the value of the status variable (# STS) is 0, it indicates that the file could be accessed normally.

Input example

```
MACRO>PUT #1, !1
MACRO>PUT #2, !2
```

NOTE

Quit

Function

The **Quit** command terminates ERX and returns control to DOS.

Input format

```
> Q < CR >
```

Input example

```
ERX>Q
```

NOTE

Register

Function

The Register command displays the contents of the CPU registers (including the flag).

Input format

```
> R < CR >
```

Input example

```
ERX>R
```

NOTE

Register

Function

The **Register** command resets all register values to 0.(If a register has a reset value, the command sets the reset value.)

Input format

```
> R  RES < CR >
```

RES : Specifies that all registers are to be reset

Input example

```
ERX>R RES
```

NOTE

Register

Function

The **Register** changes the contents of a CPU register.

Input format

```
> R reg = data < CR >
```

reg : Specify a register name.

D0 D1 D2 D3 D4 D5 D6 D7		Generalpurpose registers
A0 A1 A2 A3 A4 A5 A6 A7		
PC	Program counter	
SSP	Supervisor stack pointer	
USP	User stack pointer	
SR	Status register	
CCR	Condition code register	
T	Trace mode bit	
S	Supervisor state bit	
I	Interrupt mask	
X	Extend flag	
N	Negative flag	
Z	Zero flag	
V	Overflow flag	
C	Carry flag	
VBR	Vector base register	
SFC	Source function code	
DFC	Destination function code	

data : Specify register change data.

NOTE

VBR, SFC, and DFC are registers for 68010 only.

Input example

```
ERX>R D0=7F
ERX>R PC=1000
```

NOTE

Register

Function

The **Register** command changes the contents of a CPU register.

Input format

```
>R reg <CR>
XX YYYY = [data] [ctrl_char] <CR>

XX YYYY =/<CR>
```

reg : Specify a register name.

D0 D1 D2 D3 D4 D5 D6 D7 A0 A1 A2 A3 A4 A5 A6 A7	]	General purpose registers
PC		Program counter
SSP		Supervisor stack pointer
USP		User stack pointer
SR		Status register
CCR		Condition code register
T		Trace mode bit
S		Supervisor state bit
I		Interrupt mask
X		Extend flag
N		Negative flag
Z		Zero flag
V		Overflow flag
C		Carry flag
VBR		Vector base register
SFC		Source function code
DFC		Destination function code

NOTE

XX	:	The name of the register whose contents are to be changed is displayed.
YYYY	:	The current register content is displayed.
data	:	Specify new data. If this parameter is omitted, the register contents are not changed.
ctrl_char	:	Specify a control character. , The same register and data are displayed. ^ The preceding register and data are displayed. / The command is terminated. If this parameter is omitted, the next register and data are displayed.

Input example

```
ERX>R PC
PC   00000000 = 2000
SSP  00000010 = 10000
USP  003F0000 = ^
SSP  00010000 = ^
PC   00002000 = /
```

NOTE

VBR, SFC, and DFC are registers for 68010 only.

REM

Function

The REM command adds a comment in the batch macro.

Input format

```
> REM  [comment] < CR >
```

comment : Specify a comment.

Input example

```
MACRO>REM
MACRO>REM This is message
```

NOTE

The REM command has no control function.

REPEAT

Function

The **REPEAT** command executes the commands from **REPEAT** [(**exp**)] **to ENDR** repeatedly the number of times specified by the exp parameter.

Input format

```
>REPEAT    [exp] <CR>
> {command] <CR>
      :
>ENDR <CR>
```

exp : Specify an expression.

{symbol}	Symbolic constant
MB ({addr})	Byte memory variable
MW ({addr})	Word memory variable
ML ({addr})	Long word memory variable
REG ({reg})	Register variable
@ {n}	Internal variable
! {n}	Internal variable
# STS :	Status variable
+, −, *, /	Operators
&, |, ^, ~	Operators
==, >, >=, <=, <>	Operators

If this parameter is omitted, infinity is the default.

{command} : ERX and macro console commands

NOTE

Input example

```
MACRO>REPEAT 5
MACRO> CPUT  "\n"
MACRO>ENDR
```

NOTE

RESet

Function

The **RESet** command resets the target processor.

Input format

```
> RES < CR >
```

Input example

```
ERX>RES
ERX+>RES
```

NOTE

SAve

Function

The SAve command saves the object program.

Input format

```
> SA [/format] file_name,beg_addr,end_addr [,start_addr] < CR >
```

format	:	Specify a format type. I Intel hex format M Motorola hex format D Dump command image format If this parameter is omitted, the standard assembler format is used.
file_name	:	Specify the name of the file to be created. If the extension is omitted, .ABS is the default.
beg_addr	:	Specify the object program generation begin memory address.
end_addr	:	Specify the object program generation end memory address.
start_addr	:	Specify the user program start address. If this parameter is omitted, the value of the beg_addr parameter is the default.

Input example

```
ERX>SA SAMPLE, 1000, 1FFF
ERX>SA SAMPLE. ABS, 1000, 1FFF
ERX>SA A: \SAMPLE. ABS, 1000, 1FFF, 1000
ERX>SA/I SAMPLE. ABS, 1000, 1FFF, 1000
ERX>SA/M SAMPLE. ABS, 1000, 1FFF, 1000
ERX>SA/D SAMPLE. ABS, 1000, 1FFF, 1000
ERX>SA/M SAMPLE. ABS, MAIN. START, MAIN. END, MAIN. START
```

NOTE

SCOpe

Function

The **SCOpe** command displays the value of the program counter during emulation in real time mode.

Input format

```
> SCO  [step] < CR >
```

step : Specify the number of steps to be displayed in a decimal number ranging from 1 to 65535.
If the number is to be displayed continuously, specify the $ code.
If this parameter is omitted, 1 is the default.

Input example

```
ERX+>SCO
ERX+>SCO 5
ERX+>SCO $
```

NOTE

SEarch

Function

The SEarch command searches the memory contents and displays consistent or inconsistent data.

Input format

```
SE [/size] [/match]  beg_addr,[end_addr],search_data < CR >
```

size	:	Specify the data size. B Byte search W Word search L Long word search If this parameter is omitted, word search is the default.
match	:	Specify a search condition. D Inconsistent data is searched for.
beg_addr	:	Specify the search begin memory address.
end_addr	:	Specify the search end memory address. If this parameter is omitted, the byte count of search data.
search_data	:	Specify search data or search data row. (Hexadecimal 8/16/32bit data or ASCII data of at most 32bytes)

Input example

```
ERX>SE  1000, 1FFF, 55
ERX>SE/B 1000, 1FFF, 'This is message'
ERX>SE 1000, 1FFF, 0FFFF
ERX>SE/D 1000, +1000, 0
ERX>SE/L/D WORK. BUFF, +WORK. BUFF1_SIZE, 0
```

NOTE

If the search_data parameter is specified by data row, place a delimiter, comma [,] between hexadecimal data or between hexadecimal and ASCII data.
Enclose the ASCII data with quotation marks ['].

SHEll

Function

The **SHEll** command executes DOS system command interpreter (shell), keeping all symbols and the ERX environment.

Input format

```
> SHE < CR >
{dos Command}
> EXIT < CR >
```

EXIT : Use this parameter when the DOS command operation is terminated and control is to be returned to ERX.

Input example

```
ERX)SHE

To return to ERX, use DOS command EXIT.

Command v. 3.10 (C)Copyright Microsoft Corp 1981,1985

A:/)DIR
 Volume in drive A has no label
 Directory of  A: /

TEST    LOG     1190  11-10-86   9:59a
DEBUG   MAC       11  11-10-86  10:01a
SAMPLE  ABS     1572  11-10-86  10:06a
SAMPLE  EVT     1122  11-10-86  10:08a
ERX     EXE   301882  11-10-86  10:30a
ERX     HLP   15280  11-10-86  10:30a
      4 File(s)   267776 bytes free
A:\)
A:\)EXIT
ERX)
```

NOTE

Step

Function

The Step command starts step trace from the value of the current program counter (PC).

Input format

```
> S [/jump]  [step] < CR >
```

step : Specify the number of execution program steps in a decimal number ranging 1 to 65535.
If the $ code is entered, step trace continues permanently.
If this parameter is omitted, 1 is the default.

jump : Specify the step condition.
J Only the branch command is displayed.
If this parameter is omitted, all steps are displayed.

Input example

```
ERX>S
ERX>S $
EEX>S 10
ERX>S/J
ERX>S/J $
ERX>S/J 10
```

NOTE

STOp

Function

The **STOp** command stops emulation.

Input format

```
> STO < CR >
```

Input example

```
ERX+>STO
```

NOTE

There is a possibility that the **STOp** command cannot be executed if the target system operates abnormally.
If so, the **ICERESet** command can stop emulation forcedly.

STUB

Function

The **STUB** command displays stub.

Input format

```
> STUB < CR >
```

Input example

```
ERX>STUB
```

NOTE

STUB

Function

The **STUB** command sets stub.

Input format

```
> STUB    address,"command"  < CR >
```

address : Specify the address where the stub function is to be set.
command : All ERX commands (including batch and macro) can be used.

Input example

```
ERX>STUB 1000, "DI;R;PIN;G"
ERX>STUB &1, "DI;R;PIN;G"
ERX>STUB PROG.LOPP10, "DI;R;PI;G"
ERX>STUB PROG.LOOP10, "R;R SP=REG(SSP)-1;R;EX REG(SSP)=REG(D0);D REG(SSP);G"
```

NOTE

The **STUB** command can be set in the emulation memory or RAM of the target system.
The command in the address in which the **STUB** command is stored is not executed.
If the **STUB** command is activated, the program counter proceeds to the next command.
When the **STUB** command is activated, the **History** and **Performance** commands are stopped as when the **STOp** command is executed or break signal is generated.
Up to 32 stub points can be set.

STUB

Function

The **STUB** command specifies stub.

Input format

```
> STUB    address = switch < CR >
```

address : Specify the set stub address.
switch : Specify a switch code.
ON The set stub function is validated.
OFF The set stub function is ignored.
CLR The set stub function is released.

Input example

```
ERX>STUB 1000=OFF
ERX>STUB &1=ON
ERX>STUB PROG.LOOP10=OFF
ERX>STUB PROG.LOOP10=ON
ERX>STUB PROG.LOOP10=CLR
```

NOTE

SYMlen

Function

The SYMlen command displays symbolic names.

Input format

```
> SYM < CR >
```

Input example

```
ERX>SYM
```

NOTE

SYMlen

Function

The SYMlen command specifies the number of display columns of symbolic names.

Input format

```
> SYM  number < CR >
```

number : Specify the number of display columns in a decimal number ranging from 0 to 31.

Input example

```
ERX>SYM 10
```

NOTE

TRigger

Function

The TRigger command displays external output trigger events.

Input format

```
> TRI < CR >
```

Input example

```
ERX>TRI
```

NOTE

TRigger

Function

The **TRigger** command sets an event for an external output trigger.

Input format

```
> TRI    event_symbol [= switch] < CR >
```

event_symbol : Specify the symbol of an event that outputs an external trigger.
switch : Specify a switch code.
ON The external trigger output is validated.
OFF The external trigger output is ignored.
If this parameter is omitted, ON is the default.

Input example

```
ERX>TRI &1
ERX>TRI MAIN.START
ERX>TRI WORK.FLAG
ERX>TRI WORK.FLAG=OFF
```

NOTE

VErify

Function

The **VErify** command compares the object file and memory contents and displays inconsistent data.

Input format

```
> VE [/format]    file_name < CR >
```

file_name : Specify the name of the object file to be verified.

format : Specify a format type.

- I Intel hex format
- M Motorola hex format
- D **Dump** command image format

If this parameter is omitted, the format conforms to the standard assembler format.

Input example

```
ERX>VE SAMPLE
ERX>VE SAMPLE.ABS
ERX>VE A:\SAMPLE
ERX>VE/I SAMPLE.ABS
ERX>VE/M SAMPLE.ABS
ERX>VE/D SAMPLE.ABS
```

NOTE

WAIT

Function

After emulation is started, the **WAIT** command puts the break or stub function in operation wait state.

Input format

```
> WAIT < CR >
```

Input example

```
ERX+>WAIT
```

NOTE

The wait state is released on the following two conditions:

(1) Activation of the break function because the break condition is met.

(2) Activation of the stub function because the stub condition is met.

When the [ESC] key on the keyboard is pressed, the command entry wait state is released forcedly.

WHILE

Function

If an expression is evaluated and the result is true, the WHILE command executes the commands between **WHILE** and **ENDW** repeatedly.
If the result is false, the WHILE command is terminated.

Input format

```
> WHILE  {exp} < CR >
> {command} < CR >
      :
> ENDW < CR >
```

{exp}	:	Specify an expression.	
		{symbol}	Symbolic constant
		MB ({addr})	Byte memory variable
		MW ({addr})	Word memory variable
		ML ({addr})	Long word memory variable
		REG ({reg})	Register variable
		@ {n}	Internal variable
		! {n}	Internal character variable
		# STS	Status variable
		+, −, *, /	Operators
		&, |, ^, ~	Operators
		==, >, >=, <=, <>	Operators
{command}	:	The ERX commands and macros can be specified.	

Input example

```
MACRO>WHILE @1==0
MACRO> LET @1=@1-1
MACRO>ENDW
```

NOTE

NOTE

ERX318 for 68000・68010

COMMAND LIST