User's Manual

RENESAS

ASM45 V.1.11

User's Manual

4500 Series Absolute Assembler

Renesas Electronics
www.renesas.com

Rev.1.00 2003.07

Keep safety first in your circuit designs!

Notes regarding these materials

For inquiries about the contents of this document or product, fill in the text file the installer generates in the following directory and email to your local distributor.

¥SUPPORT¥Product-name¥SUPPORT.TXT

Renesas Tools Homepage　http://www.renesas.com/en/tools

Part 1

MELPS 4500 Cross Assembler

ASM45 User's Manual

Renesas Technology Corp.

Table of Contents

Table of Contents

List of Figures

List of Tables

CHAPTER 1. MANUAL ORGANIZATION

The ASM45 User's Manual consists of the following chapters.

- **Chapter 2. Overview**

  This chapter describes the basic function of the ASM45.

- **Chapter 3. Source Program Coding Method**

  This chapter describes how to code the source program that is processed by ASM45.

- **Chapter 4. Pseudo Instructions**

  This chapter describes the pseudo instructions that can be used with ASM45.

- **Chapter 5. Macro Instructions**

  This chapter describes the macro instructions that can be used with ASM45.

- **Chapter 6. Operation Method**

  This chapter describes how to use the ASM45.

- **Appendix A. Instruction List**

  This appendix lists all instructions that can be used with ASM45 and their coding formats.

- **Appendix B. Pseudo Instruction List**

  This appendix lists all pseudo instructions that can be used with ASM45.

- **Appendix C. Macro Instruction List**

  This appendix lists all macro instructions that can be used with ASM45.

- **Appendix D. Error Message List**

  This appendix lists and describes the contents of all error messages output by ASM45.

- **Appendix E. ASM45 Specifications**

  This appendix lists the specifications of the ASM45 such as the number of allowed labels and symbols.

This manual is written for ASM45 V.1.00.00.

CHAPTER 2. OVERVIEW

ASM45 is the cross assembler for the MELPS 4500 series. It converts a source program, (hereafter referred to as source file), written in assembly language into machine language. This process is termed "assembly".

2.1 Features

ASM45 has the following features:

1. A tag file[1] which contains error descriptions is generated. The use of tag file simplifies correction of assembly errors.
2. An editor and cross reference program CRF45 can be invoked with a command parameter.

2.2 Generated Files

ASM45 generates the following four types of files:

1. Object files (hereafter referred to as HEX files)
2. Symbol files (hereafter referred to as SYM files)
3. Print files (hereafter referred to as PRN files)
4. Tag files (hereafter referred to as TAG files)

The following is a description of each file type.

[1]The name tag file is derived from tags indicating the location of errors and warnings.

2.2.1 HEX Files

Hex files contain machine language data. ASM45 generates hex data in the following format.

- A HEX file contains machine language data divided into high-order 5 bits and low-order 5 bits with low-order data stored first.
- The low-order 5 bits are allocated from addresses 0000_{16} to $3FFF_{16}$ and the high-order 5 bits are assigned to addresses 4000_{16} to $7FFF_{16}$.
- 1 is written in the high-order 3 bits of the machine language data output by the assembler.
- If the command parameter "-O" is specified, the file is output to the specified directory. Otherwise, it is output to the current directory.
- The file extension is .HEX.

Figure 2.1 shows the structure of a HEX file.

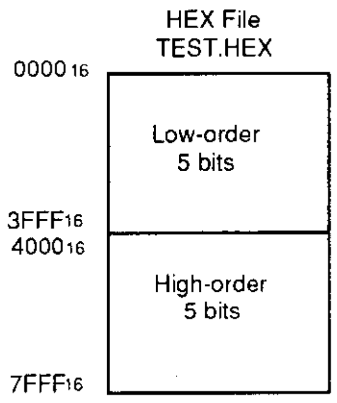

Figure 2.1 HEX File Structure

2.2.2 SYM Files

- The SYM file contains information necessary for symbolic debugging with RTT45.
- The SYM file is output when the command parameter "-S" is specified.
- If the command parameter "-O" is specified, the file is output to the specified directory. Otherwise, it is output to the current directory.
- The file extension is .SYM.
- The information in the SYM file is organized as follows:

  1. Symbol information

  Symbols used to express numbers from 0_{16} to $FFFF_{16}$.

  2. XY symbol information

  Symbols used to express values in registers X and Y.

  3. ZXY register information

  Symbols used to express values in registers Z, X, and Y.

  4. Bit symbol information

  Symbols use to express bit location and values in registers X, Y, and Z. This is used to specify a bit in RAM pointed by the data pointer (DP).

  5. Label information

  Relates labels and addresses.

Figure 2.3 shows an example of a SYM file.

2.2.3 PRN Files

- A PRN file contains the source file to be assembled, its allocated address, and generated data.
- The PRN file can be printed and used for debugging.
- The PRN file is created when the command parameter "-L" is specified.
- If the command parameter "-O" is specified, the file is output to the specified directory. Otherwise, it is output to the current directory.
- The file extension is .PRN.

[PRN File Structure]

The PRN file contains the following information.

- Symbol list information (Figure 2.4)

  The symbols and labels used in the program are listed at the beginning of the file.

- Source file information (Figure 2.5)

$$\underbrace{\texttt{SEQ.}}_{1}\ \underbrace{\texttt{LOC.}}_{2}\ \underbrace{\texttt{OBJ....}}_{3}\ \underbrace{\texttt{DEST.}}_{4}\ \underbrace{\texttt{N}}_{5}\ \underbrace{\texttt{M}}_{6}\ \underbrace{\texttt{....*....1....*....2....*..}}_{7}$$

  1. Source file line information (SEQuence)
  2. Address locations corresponding to the contents of the source file (LOCation)
  3. Object code corresponding to the contents of the source file (OBJect)
  4. Branch destination page and address (DESTination)

     The branch destination page and address are displayed as hexadecimal numbers.
  5. Nesting information (Nest)

     A nest is indicated by '1'.
  6. Macro instruction information (Macro)

     Macro expansion is indicated by a '+'.
  7. Source file column information

     An asterisk or a number is displayed at every fifth column.

- Assembly result information (Figure 2.6)

  Indicates the number of errors, warnings, total lines, comment lines, and memory size.

- When the number of columns is specified as 132 characters with the pseudo instruction .COL, the assembly time is indicated at the top of the PRN file in the following format:

  DATE (Mon Jul 7 15:06:42 1992)

  Figures 2.4 to 2.6 shows the an output example of a PRN file.

2.2.4 TAG Files

- Tag file is used to store the assembly error messages and warning messages which are generated during assembly.
- The TAG jump function can be used when using the editor(ex.MIFES) from Megasoft.
- Use the TAG file as reference when correcting errors with an editor.
- A TAG file is output when the command parameter "-E" is specified.
- If the command parameter "-O" is specified, the file is output to the specified directory. Otherwise, it is output to the current directory.
- The file extension is .PRN.

[TAG File Structure]

The TAG file contains the file name, the line number within the file, line sequence number, error number, and error message for each error and warning.

Figure 2.7 shows an output example of a TAG file.

```
-M34520M6
;********************************************************
;*     M34520 SAMPLE PROGRAM(SAMPLE.ASM)                *
;********************************************************
            .COL  80
            .TTL  SAMPLE PROGRAM      ;SET TITLE
RAM_Z0      .EQU  0                   ;DEFINE SYMBOL
RAM_Z1      .EQU  1
RAM_Z2      .EQU  2
RAM_Z4      .EQU  4
RAM_X0      .EQU  0,15                ;DEFINE XY_SYMBOL
RAM_X1      .EQU  1,15
RAM_X2      .EQU  2,15
RAM_X3      .EQU  3,15

;**************************************************
;*        MAIN ROUTINE                            *
;**************************************************
            .ORG  0,0
MAIN:       DI                        ;DISABLE INTERRUPT
            LZ    RAM_Z0
            BML   RAM_CLEAR
            LZ    RAM_Z1
            BML   RAM_CLEAR
            LZ    RAM_Z2
            BML   RAM_CLEAR
            LZ    RAM_Z4
            BML   RAM_CLEAR
            EI                        ;ENABLE INTERRUPT
LOOP:
            B     LOOP

;****************************************
;*        SUBROUTINE                    *
;****************************************
            .ORG  1,0
RAM_CLEAR
;
CLEAR_X0
            LXY   RAM_X0              ;(X)  <-- 0
            BML   CLEAR_Y
CLEAR_X1
            LXY   RAM_X1              ;(X)  <-- 1
            BML   CLEAR_Y
CLEAR_X2
            LXY   RAM_X2              ;(X)  <-- 2
            B     CLEAR_Y
CLEAR_X3
            LXY   RAM_X3              ;(X)  <-- 3
            BML   CLEAR_Y
            RT
CLEAR_Y
            LA    0                   ;(A)  <--0
            XAMD  0                   ;(A)   <-> (M), Y-1
            B     CLEAR_Y
            RT
;
            .END
;**************************************************
;*       END PROGRAM(SAMPLE.ASM)                  *
;**************************************************
```

Figure 2.2 Example of a Source File

```
#MELPS4500
#EQU                ←Symbol information
RAM_Z0 0000     RAM_Z1 0001     RAM_Z 0002      RAM_Z4 0004
#SYMBOL             ←XY symbol, ZXY symbol, bit symbol information
RAM_X0 **0F     RAM_X1 **1F     RAM_X1 **2F     RAM_X1 **3F
#LABEL              ←Label information
CLEAR_X0 0080    CLEAR_X1 0083    CLEAR_X2 0086    CLEARX_3 0089    CLEAR_Y 008D
MAIN 0000        RAM_CLEAR 0080   LOOP 000E
```

Figure 2.3 SYM File Example

```
M34520M6 ASSEMBLER SYMBOL TABLES                                          P.001


CLEAR_X0   0080  CLEAR_X1   0083  CLEAR_X2   0086  CLEAR_X3   0089
CLEAR_Y    008D  MAIN       0000  RAM_CLEAR  0080  RAM_X0     000F
RAM_X1     001F  RAM_X2     002F  RAM_X3     003F  RAM_Z0     0000
RAM_X1     0001  RAM_Z2     0002  RAM_Z4     0004  LOOP       000E
```

Figure 2.4 PRN File Example (List of symbols and labels)

Page address
↓

```
* MELPS 4500 ASSEMBLER V.1.00.00C *                                   (00 PAGE) P.002

SEQ.  LOC.  OBJ......  DEST. N M....*....1....*....2....*....3....*....4....*...

   1                          0 ;M34520M6
   2                          0 ;**************************************************
   3                          0 ;*      M34520 SAMPLE PROGRAM(SAMPLE.ASM)
   4                          0 ;**************************************************
   5                          0                .COL     80
   6                          0                .TTL     SAMPLE PROGRAM
sample.asm 6 ( TOTAL LINE 6 ) Error 12: No ';' at the top of comment "SAMPLE"
   7  0000                    0 RAM_Z0         .EQU      0
   8  0001                    0 RAM_Z1         .EQU      1
   9  0002                    0 RAM_Z2         .EQU      2
  10  0004                    0 RAM_Z3         .EQU      4
  11  000P                    0 RAM_Z4         .EQU      0,15
  12  001P                    0 RAM_X1         .EQU      1,15
  13  002P                    0 RAM_X2         .EQU      2,15
  14  003P                    0 RAM_X3         .EQU      3,15
  15                          0 ;**************************************************
  16                          0 ;*             MAIN ROUTINE
  17                          0 **************************************************
  18                          0                .ORG      0,0
  19  0000  004               0  MAIN:         DI
  20  0001  048               0                LZ        RAM_Z0
  21  0002  0C1200     01/00  0                BML       RAM_CLEAR
  22  0004  049               0                LZ        RAM_Z1
  23  0006  061200     01/00  0                BML       RAM_CLEAR
  24  0007  04A               0                LZ        RAM_Z2
  25  0008  061200     01/00  0                BML       RAM_CLEAR
  26  000A  000               0                LZ        RAM_Z4
sample.asm 26 ( TOTAL LINE 26 ) Error 21: Value is out of range "RAM_Z4"
  27  000B  061200     01/00  0                BML       RAM_CLEAR
  28  000D  005               0                EI
  29  000E                    0 LOOP:
  30  000E  18E        00/0E  0                B    LOOP
  31                          0  ;
  32                          0  ;*****************************************
  33                          0  ;*            SUBROUTINE                  *
  34                          0  ;*****************************************
```

Figure 2.5 PRN File Example (first half)

```
* MELPS 4500 ASSEMBLER V.1.00.00C *                                    (01 page) P.003

SEQ.  LOC.  OBJ......  DEST. N M....*....1....*....2....*....3....*....4....*...

  35                         0                    .ORG     1,0
  36  0080                   0  RAM_CLEAR
  37                         0  ;
  38  0080                   0  CLEAR_X0
  39  0080  30P              0                    LXY      RAM_X0
  40  0001  0C120D    01/0D  0                    BML      CLEAR_Y
  41  0083                   0  CLEAR_X0
  42  0083  31P              0                    LXY      RAM_X1
  43  0084  0C120D    01/0D  0                    BML      CLEAR_Y
  44  0086                   0  CLEAR_X0
  45  0086  32F              0                    LXY      RAM_X2
  46  0087  0C120D    01/0D  0                    BML      CLEAR_Y
  47  0089                   0  CLEAR_X0
  48  0089  33F              0                    LXY      RAM_X3
  49  008A  0C120D    01/0D  0                    BML      CLEAR_Y
  50  0086  044              0                    RT
  51  008D                   0  CLEAR_Y
  52  008D  070              0                    LA       0
  53  008E  2F0              0                    XAMD     0
  54  008F  18D       01/0D  0                    B        CLEAR_Y
  55  0090  044              0                    RT
  56                         0  ;
  57                         0                    .END

ERROR   COUNT 0002
WARNING COUNT 0000
SOURCE  LINE  0057  LINES
TOTAL   LINE  0057  LINES
COMMENT LINE  0013  LINES
OBJECT  SIZE  0032  BYTES
```

Figure 2.6 PRN File Example (second half)

```
sample.asm 6 ( TOTAL LINE 6 ) Error 12: No ';' at the top of comment "SAMPLE"
sample.asm 26 ( TOTAL LINE 26 ) Error 21: Value is out of range "RAM_Z4"
```

Figure 2.7 TAG File Example

CHAPTER 3. SOURCE PROGRAM CODING METHOD

3.1 Source Program Organization

The source program consists of lines. The line coding rules are described below.

- Each line must be complete. In other words, an instruction cannot span across more than one lines.
- Each line can contain up to 132 characters. ASM45 ignores characters beyond 132.
- The lines are classified into the following five types:

  1. MCU name specification line
     This line specifies the target MCU of the program to be assembled. It is required at the beginning of the file.
  2. Assembly instruction line
     This line contains an ASM45 assembler instruction. the corresponding machine code is generated after assembly.
  3. Pseudo instruction line
     This line contains an ASM45 pseudo instruction.
  4. Macro instruction line
     This line contains a ASM45 macro instruction. It is expanded into corresponding assembly language instructions.
  5. Comment line
     This line is not processed by ASM45. It can be used freely by the user.

3.2 Source Lines

This section describes the structure of each source line. The following notational conventions are used in the description.

1. A white or black triangle indicate a space or tab code. If white, it is required and if black, it can be omitted.
2. A ':' (colon) is not required when coding a label.
3. A space or tab is required between a label and an instruction.

3.2.1 MCU Name Specification Line

The MCU[2] name must be specified at the first line of the source program. The MCU name must be prefixed with a hyphen.

The structure of the MCU name specification line is as follows:

$\boxed{\text{-MCU name}}$ Δ $\boxed{\text{;comment}}$ <RET>

3.2.2 Assembly Language Instruction Line

The structure of an assembly language instruction line is as follows:

$\boxed{\text{label}}$ Δ $\boxed{\text{operation code}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

The label must be at the beginning of a line.

3.2.3 Pseudo Instruction Line

The structure of a pseudo instruction line is as follows. Refer to Chapter 4 and Appendix B for the details of this line.

$\boxed{\text{symbol}}$ Δ $\boxed{\text{.EQU}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

$\boxed{\text{XY symbol}}$ Δ $\boxed{\text{.EQU}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

$\boxed{\text{ZXY symbol}}$ Δ $\boxed{\text{.EQU}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

$\boxed{\text{bit symbol}}$ Δ $\boxed{\text{.EQU}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

$\boxed{\text{label}}$ Δ $\boxed{\text{pseudo instruction}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

The symbol and label must be at the beginning of a line.

[2] The MCU name can also be specified with the command parameter -M during assembly. Refer to Table 6.2 for details.

3.2.4 Macro Instruction Line

The structure of a macro instruction line is shown below. Refer to Chapter 5 and Appendix C for the details of this line.

$\boxed{\text{label}}$ Δ $\boxed{\text{macro instruction}}$ Δ $\boxed{\text{operand}}$ Δ $\boxed{\text{;comment}}$ <RET>

3.2.5 Comment Line

A comment line must start with a semicolon. The structure of a comment line is shown below.

▲ $\boxed{\text{;comment}}$ <RET>

3.3 Coding of Each Field

This section describes the coding of fields that are common to all instruction lines. Refer to Chapters 4 and 5 for the coding of fields that are unique to each instruction.

3.3.1 Operation Code Field

- The MELPS 4500 assembly language mnemonic (hereafter referred to as operation code) is coded in this field.
- The operation code can be coded either in uppercase or lowercase. Therefore, both NOP and nop are valid.

3.3.2 Operand Field

- The target of the operation code is coded in this field.
- Symbols and labels can be used in the operand[3].
- If there are more than 1 data in the operand, separate each with a comma (,).
- A space or a tab code can be placed on either side of a comma.

3.3.3 Pseudo Instruction Field

- This field contains the ASM45 pseudo instruction.
- Pseudo instructions can be coded either in uppercase or lowercase. Therefore, both .END and .end are valid.

[3] Whether a symbol can be used or a label can be used depends on the type of the operation code. Refer to Appendix A for details.

3.3.4 Macro Instruction Field

- This field contains the ASM45 macro instruction.
- Macro instructions can be coded either in uppercase or lowercase. Therefore, both .CLB and .clb are valid.

3.3.5 Symbol/Label Field

ASM45 manages labels and symbols separately. Symbols are further divided into symbols, XY symbols, ZXY symbols, and bit symbols[4]. The coding rules are described below.

- Symbols and labels must be coded at the beginning of a line.
- Symbols can have values assigned with the pseudo instruction .EQU.
- Labels are used to reference that line from other places in the program.
- When coding a label, it can be suffixed with a colon. The use of colon is recommended in order to easily distinguish labels and symbols and simplify label search with an editor.
- Symbols and labels can consist of up to 15 characters. Alphanumeric characters and special characters underscore ('_'), period ('.'), and question mark ('?') are allowed.
- The first character of a symbol or label cannot be a number. It must be an alphabetic character, underscore, period, or question mark.
- Symbols and labels are case sensitive. Therefore, BIG and big are treated as different symbols or labels.

3.3.6 Comment Field

Any user information can be coded in the comment field. The coding format is as follows:

- This field is not processed by ASM45. Therefore, it can be used freely by the user.
- A comment field is started with a semicolon.
- An entire line is assumed to be a comment when a semicolon is placed at the beginning of a line.
- Any character can be used in the comment field.

[4] Symbols defined with the pseudo instruction .EQU are treated as symbols, XY symbols, ZXY symbols, and bit symbols. Symbols defined with the command parameter "-D" are treated as symbols. Others are treated as labels.

3.4 Operand Data Format

An operand can contain any of the following four types of data.

1. Numeric constant
2. Character constant
3. Symbol constant
4. Expression

3.4.1 Numeric Constants

- No space or tab can be inserted between the symbol indicating the type of number and the numeric value.

```
Example)    .DW    $ 64    This is an error.
```

- Numeric constant can be either binary, octal, decimal, or hexadecimal.

1. Binary
Code a binary number prefixed by a percent sign ('%') or suffixed by 'B' or 'b'.

```
Examples)   .DW    %100110

            .DW    100110B
```

2. Octal
Code an octal number prefixed by an at sign ('@') or suffixed by 'O', 'o', or 'Q', 'q'.

```
Examples)   .DW    @70

            .DW    70O

            .DW    70Q
```

3. Decimal
Code a decimal number by itself without any prefix or suffix such as 23 or 256.

```
Example)    .DW    100
```

4. Hexadecimal
Code a hexadecimal number prefixed by a dollar sign ('$') or suffixed by 'H' or 'h'. Prefix the number with 0 if the number begins with an alphabet (A to F).

```
Examples)   .DW    $64

            .DW    64H

            .DW    0ABH
```

3.4.2 Character Constants

- Character constants must be enclosed in single quotes. Each character corresponds to a 7-bit ASCII code (topmost bit is 0).

  Example) `.DW 'A'` Sets to 41H

3.4.3 Symbol Constants

- Symbols are divided into labels, symbols, XY symbols, ZXY symbols, and bit symbols.

  1. Label

     (a) A label represents a ROM address.

     Example) `B MAIN` Branch to address indicated by MAIN

  2. Symbol

     (a) A symbol represents an absolute value.

     Example) `LA DATA` Load value represented by DATA in accumulator

  3. XY symbol

     (a) With the XY symbol, the X represents the value set in the X register and Y represents the value set in the Y register.

     (b) When used as operand of the LXY instruction, X represents the value set in the X register and Y represents the value set in the Y register.

     Example) `LXY XY_DATA` Load the value represented by XY_DATA in registers X and Y

  4. ZXY symbol

     (a) With the ZXY symbol, Z represents the value set in the Z register, X represents the value set in the X register, and Y represents the value set in the Y register.

     (b) When used as the operand of the LZ instruction, Z represents the value set in the Z register and X and Y are ignored.

     (c) When used as the operand of the LXY instruction, X represents the value set in the X register, Y represents the value set in the Y register, and Z is ignored.

     Example) `.LZXY ZXY_DATA` Load the value represented by ZXY_DATA in registers Z, X, and Y

5. Bit symbol

(a) A bit symbol represents the assigned bit position, and Z represents the value set in the Z register, X represents the value set in the X register, and Y represents the value set in the Y register.

(b) When used as the operand of the LZ instruction, Z represents the value set in the Z register and bits X and Y are ignored.

(c) When used as the operand of the LXY instruction, X represents the value set in the X register, Y represents the value set in the Y register, and bit Z is ignored.

(d) When used as the operand of a bit manipulation instruction (SB, RB, SZB), the bits represent the bit positions and Z, X, and Y are ignored.

Example) `.CLB      FLAG`   The bits represented by FLAG are cleared.

- Symbol, XY symbol, ZXY symbol, and bit symbol can be defined with the pseudo instruction .EQU.

3.4.4 Expressions

- An expression consists of combination of numeric constants, character constants, symbol constants, and operators. Spaces and tabs can be included between the operator and each term as necessary.

  Example) TBL + 1

- An expression is evaluated from left to right. Parentheses can be used to change the priority of operation.

  Examples) 2+6/2 The result is 4

  2+(6/2) The result is 5

- Only one monadic operator can be coded in a line.

3.5 Special Characters

Table 3.1 shows a list of special characters that can be used to code operand data.

Table 3.1 Special Characters

Char	Name	Char	Name
	Space	?	Question mark
+	Plus	#	Number sign
-	Minus	(	Left parenthesis
*	Asterisk	)	Right parenthesis
/	Slash	\	Reverse slash
$	Dollar	<	Unequal (Less than)
!	Exclamation mark	>	Unequal (Greater than)
%	Percent	&	Ampersand
,	Comma	:	Colon
'	Single quotation	;	Semicolon
^	High-hat	.	Period
	Horizontal tab		

3.6 Reserved Words

Reserved words are special character strings that are processed by ASM45. Reserved words cannot be used in labels or symbols (including XY symbol, ZXY symbol, bit symbol). Table 3.2 shows a list of characters regarded as reserved words by ASM45.

Table 3.2 Reserved Words

Char	Description
A	Register A
X	Register X
Y	Register Y
Z	Register Z

In addition, instruction mnemonics, pseudo instructions, and macro instructions are also reserved words.

3.7 Operators

Table 3.3 shows a list of operators that can be used in an expression.

Table 3.3 Operators

Type	Operator	Description
Monadic Operator	!	Take the complement of 1
	<	Shift the high-order 8 bits of a label or symbol
	>	Shift the low-order 8 bits of a label or symbol
	#	Shift a page (high-order 5 bits) of a label or symbol
Diadic Operator	+	Add
	-	Subtract
	*	Multiply
	/	Divide
	&	Bitwise AND
	| :	Bitwise OR
	^	Bitwise exclusive OR
	( )	Change operation priority

- The blank bits of a shift operation is filled with zero.
- Only one monadic operator can be coded in a line.

3.7.1 Bit Symbol Operation Rules

The rules for coding an operation line using XY symbol, ZXY symbol, and bit symbol are described below.

[Coding Method 1]

$\boxed{\text{instruction}}$ Δ $\boxed{\text{bit symbol}}$ ▲ $\boxed{\text{operator}}$ ▲ $\boxed{\text{bit symbol}}$

[Coding Method 2]

$\boxed{\text{instruction}}$ Δ $\boxed{\text{bit symbol}}$ ▲ $\boxed{\text{operator}}$ ▲ $\boxed{\text{symbol/numeric value,}}$ ...

Note: "bit symbol" can be XY symbol, ZXY symbol, or bit symbol.

1. Instructions supporting bit symbol operations

   The following instructions can be used to perform bit symbol operation.

   Macro Instruction: .clb .lzxy .seb .szxyb

   Instruction: lz lxy rb sb szb

2. Operand in front of the operator

   XY symbols, ZXY symbols, and bit symbols can be used in the operand in front of the operator.

3. Operators

   - The allowed operators are + (add) and - (subtract).
   - Only one operator can be specified in a line.

4. Operand after the operator

   - For coding method 1, the operand after the operator must be an XY symbol, ZXY symbol, or a bit symbol.
   - For coding method 2, up to four symbols or numeric values can be specified.
   - The specified numeric value is assumed to be a bit position, Z register value, X register value, or Y register value according to the following rules.

     1 operand

     $\boxed{\text{operator}}$ ▲ $\boxed{\text{Y register value}}$

     2 operands

     $\boxed{\text{operator}}$ ▲ $\boxed{\text{Y register value}}$, $\boxed{\text{X register value}}$

     3 operands

     $\boxed{\text{operator}}$ ▲ $\boxed{\text{Z register value}}$, $\boxed{\text{Y register value}}$, $\boxed{\text{X register value}}$

     4 operands

     $\boxed{\text{operator}}$ ▲ $\boxed{\text{bit position}}$, $\boxed{\text{Z register value}}$, $\boxed{\text{Y register value}}$, $\boxed{\text{X register value}}$

   - A comma must be coded between numeric values.
   - A numeric value can be omitted between commas.

5. Errors

The following cases result in error.

- The result of addition exceeds each register or bit position.
- The result of subtraction is negative.
- There is no operand following an operator.
- There is more than one operand for coding method 1.
- There is more than four operand for coding method 2.

3.7.2 Operation Examples

• XY Symbol

```
XY_DATA  .EQU     1,0 ;XY symbol definition
```

```
         Coding Example          After Expansion
     LXY  XY_DATA = 0,3     ->   LXY 1,3
     LXY  XY_DATA = 1,0     ->   LXY 2,0
```

```
xy_symbol + 1          --> x  , y+1
xy_symbol + 1,2        --> x+1, y+2
xy_symbol + 1,2,3      --> x+2, y+3
xy_symbol + 1,2,3,4    --> x+3, y+4
xy_symbol + 1,         --> x+1, y
xy_symbol + ,2,        --> x+2, y
xy_symbol + ,,3,       --> x+3, y
xy_symbol + ,2,3       --> x+2, y+3
xy_symbol + ,,3,4      --> x+3, y+4
```

• ZXY Symbol

```
ZXY_DATA .EQU     2,1,0    ;ZXY symbol definition
```

```
         Coding Example          After Expansion
     LZXY  ZXY_DATA + 1,1,1    ->  LZ  3
                               ->  LXY 2,1
```

```
zxy_symbol + 1         --> z  , x  , y+1
zxy_symbol + 1,2       --> z  , x+1, y+2
zxy_symbol + 1,2,3     --> z+1, x+2, y+3
zxy_symbol + 1,2,3,4   --> z+2, x+3, y+4
zxy_symbol + ,,,4      --> z  , x  , y+4
zxy_symbol + 1,        --> z  , x+1, y
zxy_symbol + ,2,       --> z  , x+2, y
zxy_symbol + ,,3,      --> z  , x+3, y
zxy_symbol + 1,,       --> z+1, x  , y
zxy_symbol + ,2,,      --> z+2, x  , y
```

- Bit symbol

```
BIT_DATA   .EQU    1,2,1,0  ;Symbol
```

```
        Coding Example            After Expansion
          .SZXYB BIT_DATA + 1,0,1,1 ->  LXY 2,1
                                    ->  SZB 2
```

```
bit_symbol bit,z,x,y
bit_symbol + 1        --> bit  , z  , x  , y+1
bit_symbol + 1,2      --> bit  , z  , x+1, y+1
bit_symbol + 1,2,3,4  --> bit+1, z+2, x+3, x+4
bit_symbol + 1,,,     --> bit+1, z  , x  , y
```

CHAPTER 4. PSEUDO INSTRUCTIONS

4.1 Function of Pseudo Instructions

Pseudo instructions instruct[5] ASM45 to generate machine language data as objects. ASM45 has twelve pseudo instructions and these are divided into the following three groups by function.

1. Assembly control

   - This type of pseudo instruction does not generate data. Instead, it controls the assembly flow.
   - The address counter is not affected.
   - The following pseudo instructions fall in this group

`.EQU`	Equate
`.END`	End of program
`.IF (.ELSE) .ENDIF`	Conditional assembly
`.INCLUDE`	Load a file

2. Address control

   - This type of pseudo instruction updates the address counter.
   - Data definition pseudo instruction (.DW) generates constant data.
   - The following two pseudo instructions fall in this group.

`.ORG`	Set address
`.DW`	Define data

[5] Pseudo instructions instructing ASM45 to perform something are referred to as declarations and those that affect the output file are referred to as commands.

3. List control

- This type of pseudo instruction controls output to the PRN file.
- The following six pseudo instructions fall in this group.

`.COL`	Specifies the number of columns
`.LINE`	Specifies the number of lines
`.LIST`	Starts list output
`.NLIST`	Suppresses list output
`.PAGE`	Skip to new page
`.TTL`	Specifies the list title

The function of each pseudo instruction is described below by group.

4.2 Assembly Control

4.2.1 Equation

.EQU

- Defines an absolute value to a symbol.
- Defines X and Y registers to an XY symbol.
- Defines the value of Z, X, and Y registers to an ZXY symbol.
- Defines the value of bit position BIT, X, Y, and Z registers to a bit symbol.

4.2.2 End of Assembly

.END

- Declares the end of the source program.
- ASM45 does not process any source lines beyond this line.

4.2.3 Conditional Assembly

.IF (.ELSE) .ENDIF

- Specifies the flow of assembly according to the value of a symbol.
- This can be used when coding a single source program to support several specifications or when controlling the flow of a test routine.

4.2.4 Loading a File

.INCLUDE

- The contents of the specified file is loaded where this instruction is coded.
- This instruction is useful when dividing large source program into small parts.

4.3 Address Control

4.3.1 Address Declaration

.ORG

- Declares the address of the subsequent lines.

4.3.2 Data Definition

.DW

- Generates the data specified as operand in ROM.

4.4 List Control

4.4.1 Page and Title Specification

.PAGE, .TTL

- Specifies skip to new page and the list title.

4.4.2 List Format Specification

.COL, .LINE

- Specifies the number of columns and lines on the list.
- These pseudo instructions can be coded only once in a source file.

4.4.3 List Output/Suppress Specification

.LIST, .NLIST

- Starts/suppresses list output to the PRN file.
- This is useful when only a part of the list is required such as during debugging.

CHAPTER 5. MACRO INSTRUCTIONS

5.1 Macro Instruction Functions

Macro instructions are used to generate several MELPS 4500 assembly language instructions from a single instruction. ASM45 provides four macro instructions. These instructions are divided into the following two groups by function.

1. Bit macro instructions

   - These macro instructions manipulate the bit indicated by the bit symbol specified as operand.

   - The following three macro instructions fall into this group.

.CLB	Clear the specified bit
.SEB	Set the specified bit
.SZXYB	Change the program flow according to the status of the specified bit

2. Register macro instruction

   - This macro instruction sets the values of operands Z, X, and Y specified in the operand to the respective register.

.LZXY	Set the value in registers Z, X, and Y

The function of each macro instruction is described below by group.

5.2 Bit Macro Instructions

5.2.1 Clear a Specified Bit

.CLB

- Clears the bit indicated by the bit symbol specified as operand.

5.2.2 Set a Specified Bit

.SEB

- Sets the bit indicated by the bit symbol specified as operand.

5.2.3 Change Program Flow According to Specified Bit

.SZXYB

- Checks the bit indicated by the bit symbol specified as operand and changes the program flow if it is zero.

5.3 Register Macro Instructions

5.3.1 Set Values Z, X, Y

.LXYZ

- Sets the values Z, X, and Y specified as operand in the respective registers.

CHAPTER 6. OPERATION METHOD

6.1 Getting Started

The following information (input parameters) is required to execute ASM45.

1. Source file name (must be specified)
2. Command parameters

ASM45 receives these information from the MS-DOS command line. Section 6.2 describes the input parameters and section 6.3 describes how to enter commands with some examples.

6.2 Input Parameters

6.2.1 Source File Name

1. Specify the name of the source file to be assembled. Only one name can be specified.
2. If the file extension (.ASM) is omitted, .ASM is assumed by default.
3. Extension other than .ASM (e.g. .SRC) can be used if the file name is specified in full.
4. The file name can include a directory path specification. If only the file name is specified, the current directory in the current drive is searched.

6.2.2 Command Parameters

1. Command parameters can be specified in uppercase or lowercase.
2. More than one parameter can be specified at the same time. In this case, the parameters must be separated by a space.

Tables 6.1 and 6.2 describe each command parameter.

Table 6.1 Command Parameters (1/2)

Command Parameter	Description
-D	Sets a numeric value to a symbol. The function of this command is equivalent to the pseudo instruction .EQU. Decimal is assumed if no radix is specified. The specification format is as follows (separate each symbol with a colon when defining more than one symbol simultaneously). -D symbol =numeric value[: symbol=numeric value...:symbol=numeric value] Example) `A>ASM45 FILENAME -DSYMBOL1=10:SYMBOL2=20<RET>` This command cannot be used to set bit symbols.
-E	Generates a TAG file and starts the editor[1]. The name of the editor is specified as follows. -E[editor name] Example) `A>ASM45 FILENAME -EMI<RET>` The item enclosed in brackets can be omitted. If omitted, only the TAG file is generated. If the editor name is specified, the editor is invoked using the TAG file as argument. However, if no error occurs, the editor is not invoked even if it is specified.
-L	Generates a PRN file. No PRN file is generated if this parameter is not specified.
-M	Specifies the name of the MCU to be used. If this parameter is not specified, the MCU name in the first line of the source code is used. Example) `A>ASM45 FILENAME -M34550M6<RET>` • The MCU name can also be specified in the source code. Refer to section 3.2.1 for details. • An error will occur if the MCU name specified in the source file does not match the MCU name specified with this parameter.

Table 6.1 Command Parameters (2/2)

Command Parameter	Description
-O	Specifies the output path of the generated file. A directory or drive name can be specified for path. If this parameter is not specified, output is directed to the same path as the source file. The specification format is as follows: -O path name Example) `A>ASM45 FILENAME -OB:\USR<RET>`
-P	Specifies the directory and drive containing the data file (MXXXXXXX.DAT). -P drive name Example) `A>ASM45 FILENAME -PB:\USR<RET>`
-R	Specifies that the expansion of bit macro pseudo instruction is to be output to the PRN file. Example) `A>ASM45 FILENAME -L -R<RET>`
-S	Specifies output of RTT45 symbol file. Example) `A>ASM45 FILENAME -S<RET>`
-X	Invokes the cross reference program CRF45 after assembly[2]. Example) `A>ASM45 FILENAME -X<RET>`

Notes:

1. The editor is invoked indirectly through MS-DOS COMMAND.COM. Therefore, make sure that COMMAND.COM exists in the MS-DOS command path. When working on drive other than the one containing COMMAND.COM, add the following line in the CONFIG.SYS file.

   SHELL = A:\COMMAND.COM A:\ /P   (if the startup drive is A)

   If the editor is not in the current directory or in the command path, an MS-DOS error will occur.

2. A system error will occur if the CRF45 program does not exist in the current directory or in the command path.

6.3 Input Method

ASM45 is started by entering the command at the MS-DOS prompt. The following is an example of how the command is entered.

$$\underbrace{\texttt{A>}}_{1}\underbrace{\texttt{ASM45}}_{2}\ \underbrace{\texttt{FILENAME}}_{3}\ \underbrace{\texttt{-L}}_{4}\ \underbrace{\texttt{-S}}_{5}\underbrace{\texttt{<RET>}}_{6}$$

1. MS-DOS prompt
2. ASM45
3. Name of the source file to be assembled
4. Command parameter -L specifies the generation of PRN file
5. Command parameter -S specifies the generation of SYM file
6. Return key

If there is an error in the command line, the help screen shown in Figure 6.1 is displayed and assembly is canceled. If there is no error in the command line, assembly starts.

```
C>ASM45<Enter>
4500 SERIES ASSEMBLER V.1.11.01C
COPYRIGHT(C) 1990 (1990-2003)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

Usage: ASM45 [Options...] <filename>

  -.  : all messages suppressed.
  -A  : make memory Area information.( output MAP file )
  -B  : execute Brn instruction optimize.
  -C  : output source line information.( output SYM file )
  -D  : define symbol ( use -Ds1=1:s2=2 )
  -E  : make tag file and start editor ( use -E or -Eeditor name )
  -L  : make list file
  -M  : define CPU name ( use -M34550M8 )
  -O  : select drive and directory for output ( use -Oa:\work )
  -P  : select directory(drive) of M345XXXX.dat file.( use -P\work )
  -R  : output bit macro expansion
  -S  : make symbol file for symbolic debugger
  -VER: display version.
  -X  : execute crf45
```

Figure 6.1 Help Screen when a Command Error Occurs

When assembly completes, the number of errors, warnings, total lines, comment lines, and object size are output to the screen. Figure 6.2 shows the output to screen when assembly completes normally.

```
C>ASM45 TEST<Enter>
4500 SERIES ASSEMBLER V.1.11.01C
COPYRIGHT(C) 1990 (1990-2003)
RENESAS TECHNOLOGY CORPORATION                          :ON
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

now processing pass 1
----*----*
now processing pass 2
----*----*
ERROR   COUNT 0000
WARNING COUNT 0000
SOURCE  LINE  1079 LINES
TOTAL   LINE  1079 LINES
COMMENT LINE  0092 LINES
OBJECT  SIZE  0980 BYTES
```
Figure 6.2 Normal Termination Screen

6.4 Errors

6.4.1 Types of Error

Errors during ASM45 execution may be caused by any of the following reasons:

1. MS-DOS related errors

   These are errors related to the MS-DOS environment such as insufficient disk or memory. Refer to the list of error messages in Appendix D and correct the error using MS-DOS commands.

2. ASM45 command line input errors

   These are errors resulting from errors in the ASM45 invocation command. Check the content of this chapter and reenter the command.

3. Source file errors

   These are errors caused by the code in the source file such as duplicate label definition or reference to undefined symbol. Correct the lines causing the error and reassemble. Correct HEX file is not generated when there is an assembly error.

When ASM45 detects an error or warning condition, it outputs the error contents (file name, line number within file, line sequence number, error number and error message) to the screen and PRN file. Refer to the error message table in Appendix D for the description of errors.

```
C>ASM45 TEST<Enter>
4500 SERIES ASSEMBLER V.1.11.01C
COPYRIGHT(C) 1990 (1990-2003)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

now processing pass 1
----*----*
now processing pass 2

  89  0187  106         00/06 0       BM LOOP2
test.ASM 89 ( TOTAL LINE 89 ) Error 21: Value is out of range "LOOP2"
  91  0189  106         00/06 0       BM LOOP2
test.ASM 91 ( TOTAL LINE 91 ) Error 21: Value is out of range "LOOP2"
----*----*
ERROR   COUNT 0002
WARNING COUNT 0000
SOURCE  LINE  1086 LINES
TOTAL   LINE  1086 LINES
COMMENT LINE  0093 LINES
OBJECT  SIZE  0982 BYTES
```

Figure 6.3 Error Display

6.4.2 Value Returned to MS-DOS

When the command to execute ASM45 is coded within a batch file, it may be desirable to change the process flow according to the execution result. ASM45 classifies the execution result into four error levels shown in Table 6.3 and returns it to MS-DOS. Refer to the MS-DOS manual for description on how to use the error level.

Table 6.3 Error Levels

Error Level	Description
0	Normal termination
1	Error in assembly source file
2	ASM45 command input error
3	MS-DOS error

6.5 Environment Variables

The following environment variables are used when executing ASM45.

- Data file search path specification

  When invoked, ASM45 refers to a data file containing information specific to each MCU. The path to this data file can be set with the environment variable 4500DAT. The following is an example of setting 4500DAT.

```
A>SET 4500DAT=A:\USR\DAT
          ↑         ↑
Environment variable  Search Path
```

Figure 6.4 Setting Environment Variable 4500DAT

  The data file is searched in the following order.

  1. Current directory
  2. Path specified with command parameter -P
  3. Path set with 4500DAT

- CRF45 and editor path specification

  ASM45 can be instructed to invoke CRF45 and an editor after assembly. CRF45 and the editor are invoked indirectly through the MS-DOS COMMAND.COM file. Therefore, make sure that COMMAND.COM is available in the MS-DOS command path. When working from drive other than the one containing COMMAND.COM, add the following line in the CONFIG.SYS file. Refer to the MS-DOS manual for the details concerning command path. The following is a coding example when the COMMAND.COM file is in the root directory of drive A.

```
SHELL = A:\COMMAND.COM A:\ /P
```

  MS-DOS will return an error if CRF45 or the editor is not in the current directory or command path.

APPENDIX A. INSTRUCTION LIST

A.1 Symbol Table

Table A.1 shows the meaning of symbols used in the instruction list.

Table A.1 Symbols

Symbol	Description	Symbol	Description
a	General page address	p	Page address
x	Value of register X	y	Value of register Y
z	Value of register Z	n	Immediate value
j	Bit value	label	Label
symbol	Symbol value	xy_symbol	XY symbol value
zxy_symbol	ZXY symbol value	bit_symbol	Bit symbol value

A.2 Instruction Table

Tables A.2 and A.3 shows all of the core instructions available with ASM45. The coding format is shown beside each instruction.

A.2.1 Implied Instruction Table

The following table lists the available implied instructions.

Table A.2 Implied Instructions

ADST	AM	AMC	AND	CLD	CMA
DEY	DI	EI	IAP0	IAP1	IAP2
IAP3	IAP4	IAP5	IAP6	IAP7	INY
NOP	OP0A	OP1A	OP2A	OP3A	OP4A
OP5A	OP6A	OP7A	OR	POF	POF2
RAR	RBK	RC	RC3	RC4	RD
RT	RTI	RTS	SBK	SC	SC3
SC4	SD	SEAM	SNZ	SNZ1	SNZ2
SNZ3	SNZ4	SNZ5	SNZ6	SNZI	SNZP
SNZT1	SNZT2	SPCR	SST	STCR	SZAD
SZC	SZD	SZSI	T1AB	T2AB	T3AB
T4AB	T5AB	T6AB	TAB	TAB1	TAB2
TAB3	TAB4	TAB5	TAB6	TABE	TACP
TAH	TAHA	TAI1	TAI2	TAJ1	TAJ2
TAL	TALA	TAL1	TAMR	TABN1	TABN2
TAQ1	TAQ2	TAQ3	TAQ4	TAW1	TAW2
TAW3	TAW4	TAW5	TAW6	TAD	TASP
TAX	TAW	TAY	TAZ	TAV1	TAV2
TBA	TC1A	TC2A	TCPA	TDA	TEAB
TFR0A	TFR1A	TFR2A	TFR3A	TFR4A	THA
THAA	TI1A	TI2A	TIRI	TJ1A	TJ2A
TL1A	TL2A	TL3A	TLA	TLAA	TLCA
TPU3	TPAA	TPBA	TPTA	TPU0A	TPU1A
TPU2A	TQ1A	TQ2A	TQ3A	TQ4A	TR1A
TR1AB	TV1A	TV2A	TW1A	TW2A	TW3A
TW4A	TW5A	TW6A	TYA	TWA	TZCA
WRST					

A.2.2 List of Instructions with Operands

The following is a list of instructions with operands. The format of the allowed symbols and labels is also shown.

Instruction	Operand	Operand Range	Code	Format
A	n	0 to 0F	A	symbol
B	a	Absolute address in the same page	B	label
BL	p,a	p: Depends on ROM address[1] a: 0 to 7F	BL BL	symbol,label label
BLA	p	p: Depends on ROM address[1]	BLA BLA	symbol label
BM	a	Absolute address within 2 pages	BM	label
BML	p,a	p: Depends on ROM address[1] a: 0 to 7F	BML BML	symbol,label label
BMLA	p	p: Depends on ROM address[1]	BMLA BMLA	symbol label
LA	n	0 to 0F	LA	symbol
LXY	x,y	x: 0 to 0F y: 0 to 0F	LXY LXY LXY LXY	symbol,symbol xy_symbol zxy_symbol bit_symbol
LZ	z	Depends on RAM address[2]	LZ LZ LZ	symbol zxy_symbol bit_symbol
RB	j	0 to 3	RB	bit_symbol
SB	j	0 to 3	SB	bit_symbol
SEA	n	0 to 0F	SEA	symbol
SNZ	n	0 to 1	SNZ	symbol
SNZI	n	0 to 1	SNZI	symbol
SZB	j	0 to 3	SZB	bit_symbol
TABP	p	Depends on ROM address[1]	TABP	symbol
TAM	j	0 to 0F	TAM	symbol
TMA	j	0 to 0F	TMA	symbol
XAM	j	0 to 0F	XAM	symbol
XAMD	j	0 to 0F	XAMD	symbol
XAMI	j	0 to 0F	XAMI	symbol

[1] The range of operand depends on the ROM address of the MCU. If the MCU type is M8 it is 0 to 3F, if it is M6 it is 0 to 2F, and if it is M4 it is 0 to 1F.
[2] The operand range is 0 to 3. However, when register Z is used as a pointer to RAM, it depends on the RAM address of the MCU.

APPENDIX B. PSEUDO INSTRUCTIONS

B.1 Notational Convention

The pseudo instructions available with ASM45 are described in alphabetical order. The following notational conventions are used.

1. The item enclosed in brackets [ ] can be omitted.
2. A white (Δ) or black triangle (▲) indicate a space or tab code. If white, it is required and if black, it can be omitted.
3. A space or tab is required between a label and pseudo instruction.
4. A ':' (colon) is not required when coding a label.
5. Numeric value Z indicates the value to be set in register Z.
6. Numeric value X indicates the value to be set in register X.
7. Numeric value Y indicates the value to be set in register Y.

B.2 Pseudo Instruction Description

COL — Specify number of columns (default 132)

Syntax

Δ.COLΔvalue

Description

- Specifies the number of characters in a line (80 or 132).
- If a value less than 80 is specified, 80 is assumed, if a value greater than 80 is specified 132 is assumed.
- This instruction can be used only once in a program.
- A symbol can be used as the value.

Example

```
            .COL 80       ;Set number of columns to 80.
               :
```

.DW — Set data

Syntax

[label:]Δ.DWΔvalue

Description

- Defines the specified value as data (unit: 10 bits).
- When defining more than one data, separate each data with a comma.
- The maximum number of values that can be specified in a line is sixteen.
- A symbol can be used as the value.

Example

```
label:    .DW      1FFH      ; Set 1FFH.
          .DW      symbol    ; Set value of symbol.
           :
           :
```

Syntax

Δ.END

Description

- This instruction indicates the end of a source program.
- This pseudo instruction must be coded at the end of a source program.
- ASM45 does not assemble any lines after this pseudo instruction.

Example

```
          :
         .END                        ; Declares the end of program.
```

Syntax 1

symbolΔ.EQUΔvalue

Syntax 2

XY symbolΔ.EQUΔvalue X,value Y

Syntax 3

ZXY symbolΔ.EQUΔvalue Z,value X,value Y

Syntax 4

bit symbolΔ.EQUΔbit location,value Z,value X,value Y

Description

- Assigns a value to the symbol on the left side.
- Syntax 1 assigns a sixteen bit integer to a symbol.
- Syntax 2 assigns the values of registers X and Y to an XY symbol. This XY symbol can be used in an LXY instruction.
- Syntax 3 assigns the values of registers Z, X, and Y to a ZXY symbol. This symbol can be used in an LXY instruction, LZ instruction, or macro instruction (.LZXY).
- Syntax 4 assigns bits 0 to 3, the value of registers Z, X, and Y to a bit symbol. This bit symbol can be used in a LXY, LZ, SB, RB, SZB or macro pseudo instruction (.LZXY, .CLB, .SZXYB, .SEB).
- Symbols used as values must be defined before this line.

Example

```
COUNT  .EQU  4        ;Syntax1
TYPE   .EQU  3,0      ;Syntax2
DATE5  .EQU  0,3,4    ;Syntax 3 (No bit specification)
FLAG0  .EQU  1,0,3,4  ;Syntax3 (Bits specified)
       LXY   TYPE     ;X=3,Y=0
       LXY   DATE5    ;X=3,Y=0
       LXY   FLAG0    ;X=3,Y=4
       LZ    DATE5    ;Z=0
       LZ    FLAG0    ;Z=0
       SB    FLAG0    ;Set the 1st bit in DP to 1.
       RB    FLAG0    ;Set the 1st bit in DP to 0.
       SZB   FLAG0    ;If the 1st bit in DP is 0, skip next inst.
```

Syntax

Δ.IFΔexpression

<statement 1>

Δ.ELSE

<statement 2>

Δ.ENDIF

Description

- A label, symbol defined with the .EQU pseudo instruction, or a symbol defined with the command parameter -D can be used for expression.
- If the expression following .IF is true (not 0), <statement 1> is assembled. If it is false (0), <statement 2> is assembled.
- This pseudo instruction can not be nested.
- Statements 1 and 2 can be multiple instructions on multiple lines.
- Symbols can be used in values.
- Labels and symbols used in operand must be defined before this line.

Example

```
.IF     FLAG     ;If the value of FLAG is true, statements up
   :             ;to ELSE are assembled.
   :
.ELSE        ;if false, statements up to .ENDIF are assembled.
   :
.ENDIF
```

.INCLUDE — Load a file

Syntax

Δ.INCLUDEΔfilename

Description

- The specified file is loaded at the location of this pseudo instruction.
- The filename must be specified in full.
- The filename can contain a path specifier.
- This pseudo instruction cannot be nested.

Example

```
.INCLUDE TEST.INC ;Load the contents of TEST.INC.
    :
    :
```

.LINE — Specify number of lines per page (default 54)

Syntax

Δ.LINEΔvalue

Description

- This instruction specifies the number of lines per page (5 to 255).
- The 4 header lines are included in the line number.
- This pseudo instruction can be used only once in a program.
- A symbol can be used for value.
- The symbol used as operand must be defined before this line.

Example

```
LINE  60    ;Set the number of lines to 60
 :
 :
```

.LIST — Start list output (default)

Syntax

Δ.LIST

Description

- This pseudo instruction starts list output to PRN file.
- This pseudo instruction is used to restart output to the .PRN file after stopping output with the pseudo instruction .NLIST.

Example

```
.NLIST   ; Suppress list output.
 :       ; No file is output to PRN file until ".LIST"
 :
.LIST    ; Start list output.
 :       ; Output subsequent pseudo instructions to PRN file.
```

.NLIST — Suppress list output

Syntax

Δ.NLIST

Description

- This pseudo instruction suppresses list output to PRN file.
- List output can be restarted with the pseudo instruction .LIST.

Example

```
.NLIST   ; Suppress list output.
 :       ; No file is output to PRN file until ".LIST"
 :
.LIST    ; Start list output.
 :       ; Output subsequent pseudo instructions to PRN file.
 :
```

.ORG Address declaration (default 0000H)

Syntax 1

Δ.ORGΔaddress

Syntax 2

Δ.ORGΔpage,offset

Description

- Declares the start address after this line.
- If the start address is not specified, 0000_{16} is assumed.
- A symbol can be used for page and a label can be used for offset.
- The label or symbol used as operand must be defined before this line.

Example

```
        .ORG        780H         ;Set to address of page 15.
          :
          :
        .ORG        5,10         ;Set to address 10 at page 5.
```

.PAGE Advance the list page and specify a title

Syntax

Δ.PAGEΔ['Title']

Description

- The list is advanced just before this instruction and the title specified as operand is output in the header.
- The title is output only on the advanced page.
- The title must be enclosed in single quotes.
- If the number of columns is 80, the title can be up to 16 characters long. If the number of columns is 132, it can be up to 30 characters.
- If the title is omitted, the page is simply advanced.

Example

```
.PAGE 'PROG1'     ;Output PROG1 at the head of the PRN file.
  :
```

Syntax

Δ.TTLΔ['title']

Description

- Outputs the title specified as operand in the header of the list.
- The title is output on every page of the list file.
- The title must be enclosed in single quotes.
- The title can be up to 16 characters long.
- If a title is specified with the pseudo instruction .PAGE, the specified title is output only on that page.
- If the .TTL pseudo instruction is coded more than once, the last processed .TTL takes effect.

Example

```
.TTL     'PROG1'  ;Output PROG1 at the head of the PRN file.
  :
```

APPENDIX C. MACRO INSTRUCTION LIST

C.1 Notational Convention

The ASM45 macro instructions are listed in alphabetical order. The following notational conventions are used.

1. The item enclosed in brackets [ ] can be omitted.
2. A white (Δ) or black triangle (▲) indicate a space or tab code. If white, it is required and if black, it can be omitted.
3. A space or tab is required between a label and macro instruction.
4. A ':' (colon) is not required when coding a label.

C.2 Macro Instruction Description

.CLB Clear a bit

Syntax

[label:]Δ.CLBΔbit symbol

Description

- Sets the bit specified by the bit symbol to "0".

Example

[Coding Example]

```
FLAG0    .EQU          1,0,3,4
         .CLB          FLAG0
```

[Expansion Example]

```
         LXY      3,4
         RB       1
         :
```

.LZXY Set register

Syntax 1

[label:]Δ.LZXYΔZXY symbol

Syntax 1

[label:]Δ.LZXYΔbit symbol

Description

- Sets the file group (Z), file (X), and column (Y) indicated by the specified symbol in each register.
- ZXY symbol and bit symbol can be used.

Example

[Coding Example]

```
FLAG0    .EQU     1,0,3,4
         .LZXY    FLAG0
         :
```

[Expansion Example]

```
         LZ       0
         LXY      3,4
         :
```

.SEB Set bit

Syntax

[label:]Δ.SEBΔbit symbol

Description

- Sets the bit specified by the bit symbol to "1".

Example

[Coding Example]

```
FLAG0    .EQU     1,0,3,4
         .SEB     FLAG0
          :
```

[Expansion Example]

```
         LXY      3,4
         SB       1
         :
```

.SZXYB Test bit

Syntax

[label:]Δ.SZXYBΔbit symbol

Description

- Tests the bit specified by the bit symbol and skips the next instruction if "0".

Example

[Coding Example]

```
FLAG0    .EQU     1,0,3,4
         .SZYB    FLAG0
         :
```

[Expansion]

```
         LXY      3,4
         SZB      1
         :
```

APPENDIX D. ERROR MESSAGE LIST

D.1 Input Errors

Figure D.1 shows the screen when there is an input error.

```
Usage:   ASM45 <filename> [-D] [-E] [-L] [-M] [-O] [-P] [-R] [-S] [-X]
```

Figure D.1 Input Error Messages

- Error Description

  There is an error in the input command.

- User Action

  Refer to the Help screen and reenter the command.

D.2 System Errors

When a system error is detected during assembly, an error message is displayed on the screen and assembly is canceled. Table D.1 shows a list of possible system errors.

Note: The total number of symbols and labels that can be used in an ASM45 assembly depends on the amount of system memory available for assembly. Refer to Appendix E for the number of symbols and labels that can be used under the standard environment.

Table D.1 System Errors

Error Message	Error Description and User Action
Can't open xxx	The specified file cannot be found. ⇛ Check the source file name and reenter. Also increase the value of the FILES parameter in CONFIG.SYS.
Can't create xxx	The file cannot be created. ⇛ Check the specification of the command parameter -O and reenter. Also increase the value of the FILES parameter in CONFIG.SYS.
Out of disk space	There is insufficient disk space to output the file. ⇛ Create sufficient free area on disk.
Out of heap space	There is insufficient memory for the assembler to execute. ⇛ Reduce the number of symbols or labels. Also increase the MS-DOS application memory.
Cannot find crf45.exe	The CRF45 cannot be found. ⇛ Copy CRF45 to the current directory or a directory within the MS-DOS command path.
Can't find command.com for execute xxx	The COMMAND.COM file necessary to start the editor specified with the -E option cannot be found. ⇛ Check the MS-DOS command path specification.
Multiple define of MCU name	An MCU name is specified at the beginning of command input and source file. ⇛ Delete one of the two.
M345XXXX.DAT not found	There is no data file for the specified MCU. ⇛ Check whether the specified data file is available.
M345XXXX.DAT file Version mismatch. please use V.X.X	The data file version conflicts with the assembler version. ⇛ Use the same version data file.

D.3 Assembly Errors

When an assembly error is detected, an error message is output to the screen and the PRN file. Tables D.2 to D.6 list the possible assembly errors.

Table D.2 Assembly Errors (1/5)

Error No.	Error Message	Error Description and User Action

1	Already had same statement	A pseudo instruction that can only be used once in a source file is used more than once.
		Example
		`.LINE    60`
		`:`
		`.LINE    80`
		⇒ Delete all but one declaration.

2	Reference to backward label or symbol	A pseudo instruction is referencing a backward label or symbol.
		Example
		`        .IF     FLAG`
		`FLAG    .EQU    0`
		⇒ Declare the label or symbol before it is referenced.

3	Division by 0	An expression contains division by zero. ⇒ Check the expression.
4	Illegal operand	An operand contains an illegal character. Example `.CLB     =FLAG` ⇒ Check the operand.
5	Improper operand type	The specified operand is not allowed for the this mnemonic. Example `.IF      SYMBOL` ⇒ Check the instruction.

Table D.3 Assembly Errors (2/5)

Error No.	Error Message	Error Description and User Action

6	Invalid label definition	A label is defined where it is not allowed.
		Example 1
		`LABEL   .COL     80`
		Delete the label.
		Example 2
		`LABEL2  .EQU     100`
		⇛ Change the label to symbol.

7	Out of maximum program size	The address exceeds the ROM size. Example `        .ORG     10000H` ⇛ Change the program so that the address is within ROM size.
8	Label is multiple defined	The same label or symbol is defined more than once. Example `MAIN:   NOP` `MAIN:   NOP` ⇛ Check the label or symbol.

9	Nesting error	The pseudo instruction .IF or .INCLUDE is nested.
		Example
		`.IF     DATA1`
		`        .IF      DATA2`
		`          :`
		`        .ENDIF`
		`.ELSE`
		`          :`
		`.ENDIF`
		⇛ Change the program so that the pseudo instruction is not nested.

Table D.4 Assembly Errors (3/5)

Error No.	Error Message	Error Description and User Action
10	No .END statement	There is no .END statement in the source file. ⇒ Code a .END statement at the end of the program.
11	No symbol definition	No symbol is coded. Example) `        .EQU    60` ⇒ Code a symbol.
12	No ';' at the top of a comment	There is no semicolon at the beginning of a comment field. Example) `MAIN3:  LXY  2,0   FLAG?` Add a semicolon at the beginning of a comment field.

13	Not in conditional block	There is a .ELSE or .ENDIF statement without a corresponding .IF statement.
		(This error also occurs when there is an error in the corresponding .IF statement.)
		
		Example) `.IF    DATA1`
		`    :`
		`.ENDIF`
		`    :`
		`.ELSE`
		`    :`
		`.ENDIF`
		
		⇒ Check the corresponding .IF statement.

14	Operand is expected	A required operand is missing. Example) `.CLB` ⇒ Check the operand.

Table D.5 Assembly Errors (4/5)

Error No.	Error Message	Error Description and User Action
15	Questionable syntax	The mnemonic is spelled incorrectly. Example) `LZY 2,8` ⇒ Check the spelling of the mnemonic.

16	Reference to a multi defined label	A duplicately defined label or symbol is referenced.
		
		Example) `MAIN: NOP`
		`MAIN: NOP`
		`BL MAIN`
		
		⇒ Check the label or symbol.

17	Relative jump is out of range	The destination address of a relative jump instruction is out of range. ⇒ Reallocate the program or change the instruction.
18	Label is a reserved word	A reserved word is used as label or symbol. Example) `A .EQU 1FFH` ⇒ Change the label or symbol.
19	Reference to undefined label	An undefined label or symbol is referenced. ⇒ Check the label or symbol.

Table D.6 Assembly Errors (5/5)

Error No.	Error Message	Error Description and User Action
20	Value error	There is an error in the data coding format. Example) `.DW '123456789012345678'` ⇛ Check the data coding format.
21	Value is out of range	The data exceeds the allowed range. Example) `LA $10` ⇛ Check the operand coding format.
22	"()" format error	The number of left and right parentheses does not match. Example) `LXY (35/4,8+` ⇛ Check the operand coding format.
23	Label error	The label is longer than 16 characters or contains invalid character. Example) `L123456789012345:` ⇛ Check the label.

D.4 Other Errors

Table D.7 shows other errors.

Table D.7 Other Errors

Error No.	Error Message	Error Description and User Action
50	Can't execute CRF45	CRF45 cannot be started because there is an error in the source file. ⇛ Correct the error in the source file and reassemble.

D.5 Warnings

When a warning is detected, a warning message is output to the screen and PRN file. Table D.8 describes the meaning of each warning message.

Table D.8 Warnings

Warning No.	Warning Message	Warning Description and User Action
1	Phase warning	The address specified with the pseudo instruction .ORG is smaller than the address previously specified with .ORG. Example) `.ORG 100H` / `MAIN: .SEB FLAG` / `:` / `.ORG 80H` ⇛ Change the address specified with .ORG so that it is greater than the previous address.
2	.END statement in include file	An included file contains a .END statement. ⇛ Code the .END statement in the source file.

APPENDIX E. ASM45 SPECIFICATIONS

E.1 Standard Environment

Table E.1 shows the MS-DOS standard environment used to determine the specification.

Table E.1 MS-DOS Standard Environment

Item	Specification
MS-DOS Version	V.3.X or above
Memory Size	User memory: 256K bytes This value is determined by the MS-DOS standard command CHKDSK.

E.2 ASM45 Specifications

Table E.2 shows the ASM45 specification under the standard MS-DOS environment. The calculated estimate is shown for some values that cannot be measured.

Table E.2 ASM45 Specification

Item	Specification
Number of characters per line	Up to 132 characters. (Subsequent characters are ignored)
Label and symbol length	Up to 15 characters.
Number of labels and symbols (assuming each contains 15 characters)	Total of 1500 symbols and labels. The size depends on the available MS-DOS user memory.

Part 2

MELPS 4500 Cross Reference Program

CRF45 User's Manual

Renesas Technology Corp.

Table of Contents

List of Figures

List of Tables

CHAPTER 1. CRF45 MANUAL ORGANIZATION

The CRF user's manual consists of the following chapters.

- **Chapter 2. Overview**

  This chapter describes the basic functions, CRF45 input files, and generated files.

- **Chapter 3. Operation Method**

  This chapter describes how to enter the CRF commands.

- **Appendix A. Error Messages**

  The description and user action are listed for all error messages output by CRF45.

- **Appendix B. CRF45 Specification**

  The specifications of CRF45 such as the number of labels and symbols are described.

This manual is written for CRF45 V.1.00.00.

CHAPTER 2. OVERVIEW

CRF45 generates a cross reference (hereafter referred to as cross reference list) between the labels and symbols in the source file[1] or the print file[2]. This list can be used to determine the relationship between each part of the source file during program debugging.

2.1 Function

CRF45 provides the following functions to simplify program debugging.

1. Lists instructions referencing labels in reference line number sequence.
2. Processes source file included with the pseudo instruction .INCLUDE.
3. Prints the header specified with the pseudo instruction .PAGE in the cross reference list.
4. Allows specification of source file or print file.

2.2 Input Files

CRF45 uses the following input files.

1. Source file

   If the target file is a source file that can be assembled by ASM45, a cross reference of coded labels, symbols, target files, and physical source line information are generated. The file included with the pseudo instruction .INCLUDE is also processed.

2. Print file

   If the target file is a print file that is output by ASM45, cross reference is made using the line number information output on the left side. The line number output on the cross reference list is the line number output on the left side of the print file (the physical line number of the print file is not output). The file specified with the pseudo instruction .INCLUDE is not processed in the case of print file.

[1] File that is assembled by ASM45.
[2] File generated by ASM45. Contains the source code, allocated addresses, and generated data.

Note: CRF45 distinguishes between a print file and a source file using the file extension of the input file. Print file is assumed if the extension of the input file is .PRN and source file is assumed if any other extension is used.

If the extension of the input file is omitted, .ASM is assumed.

2.3 Generated Files

Upon completion, CRF45 generates the following files.

1. Cross reference file (hereafter referred to as CRF file).
    - Shows the cross reference for symbols and labels
    - The list page is 80 characters wide and 57 lines long.
    - This file can be output to the printer to be used for debugging and editing.
    - The file extension is .CRF.

2.4 CRF File Organization

Figure 2.1 shows an output example of a CRF file. The CRF file contains the following information.

1. Labels and symbols output in ASCII code sequence.
2. The string and number beside labels and symbols are the file name and line number of the assembly list containing that label or symbol.
3. A number sign ('#') indicates that the label or symbol is defined in that line.
4. An ampersand ('&') indicates that the label or symbol is referenced by a BM, BML, or BMLA instruction.
5. File name is output up to 12 characters. Remaining characters are omitted.
6. Labels and symbols are output up to 15 characters. Remaining characters are omitted.
7. The title specified with the pseudo instruction .PAGE is output in the list header (the first 25 characters are output).
8. If a label or symbol is used in more than one file, a plus ('+') is output in front of the file name and the line number for each file is output.
9. CRF45 does not check the value of symbols and labels in the source program. Therefore, it cannot process conditional assembly.

```
                                                    Title (specified with .PAGE)
                                                           ↓
MELPS 4500 CROSS REFERENCE V.1.00.00C    4500TST PROGRAM1         PAGE. 001

ATOC      filename.asm 396#
               ↑      (source file)
          File name
               ↓      (include file)
          + inc_file.inc 177&
                          ↑
                     line number


A_D         filename.asm 118
          + inc_file.inc 174#
COUNT       filename.asm 154&    166&    201&    214&    379&    388&
          + inc,file.inc  95#
D0_SET      filename.asm 250&    333#
D10         filename.asm 318     320#
D10_SET     filename.asm 322&    363#
D1_SET      filename.asm 254&    336#
D2          filename.asm 257     259#
D2_SET      filename.asm 261&    339#
D3          filename.asm 264     266#
D3_SET      filename.asm 268&    342#
D4          filename.asm 271
          + inc_file.inc 273#
D4_SET      filename.asm 275&
          + inc_file.inc 345#
D5          filename.asm 278
          + inc_file.inc 280#
D5_SET      filename.asm 282&
          + inc_file.inc 348#
D6          filename.asm 285     287#
    :
```

Figure 2.1 CRF File Example

CHAPTER 3. OPERATION METHOD

3.1 Getting Started

The following information (input parameter) must be entered to start CRF45.

1. Input file name (source file or print file) (required)
2. Command parameter

3.2 Input Parameters

3.2.1 Input File Name

1. The input file name must be specified.
2. The input file can be either a source file or a print file.
3. If the file extension (.ASM) is omitted, .ASM is assumed.
4. File with attribute other than .ASM can be processed by specifying the full file name.
5. The file name can include the drive name and directory name. If only the file name is specified, the current directory in the current drive is searched.

Note: CRF45 distinguishes between a print file and a source file using the file extension of the input file. Print file is assumed if the extension of the input file is .PRN and source file is assumed if any other extension is used.

3.2.2 Command Parameter

Command parameters are used to specify whether to process the pseudo instruction .INCLUDE and to specify the drive of the output file. Table 3.1 describes the command parameters.

Table 3.1 Command Parameters

Command Parameter	Description
-.	Suppresses output of all messages to screen.
-I	Suppresses processing of include file specified with -I. Up to 16 include files can be specified. This parameter is specified as follows: -I include_filename Example) `A>CRF45 SRCFILE -ISMP1.INC,SMP2.INC<RET>` Suppresses processing of include files SMP1.INC and SMP2.INC.
-O	Specifies the path of the generated file. Directory or drive can be specified as path. If this is omitted, the generated file is output to the same path as the input file. The path is specified as follows: -O path_name Example) `A>CRF45 SRCFILE -OB:\WORK<RET>` The CRF files are output to the WORK directory on drive B.

3.3 Input Method

3.3.1 Command Line Input

CRF45 is started by entering the command name from the MS-DOS prompt line. Figure 3.1 shows how to enter the command name.

```
A>CRF45 TEST<RET>
```

Figure 3.1 Command Line Input Example

If there is an error in the command, a help screen shown in Figure 3.2 is output and processing stops.

```
C>CRF45<Enter>
4500 SERIES CROSS REFERENCE V.1.00.01C
COPYRIGHT(C) 1992 (1992-2003)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

Usage: crf45 <filename> [-.] [-ifilename,..] [-opath]
       -. : all messages supressed
       -i : not include specified files ( use -ifilename,.... )
       -o : select drive and directory for output ( use -otmp )
```

Figure 3.2 Help Screen for Command Line Error

3.4 Errors

3.4.1 Types of Error

Errors that occur during CRF45 execution are caused by any of the following reasons:

1. MS-DOS related errors

   These are errors such as insufficient disk or memory which are related to the MS-DOS environment in which CRF45 is executed. Refer to the error message in Appendix A and correct the error using MS-DOS commands.

2. CRF45 command line related errors

   These errors occur when entering command name from the MS-DOS prompt line. Check the contents of this chapter and reenter the command.

3. Target input file related error

   This error occurs when the target file does not exist.

When CRF45 detects an error, it is displayed on the screen in the format shown in Figure 3.3. Correct the error referring to the error message in Appendix A and reexecute.

```
C>CRF45 TEST<Enter>
4500 SERIES CROSS REFERENCE V.1.00.01C
COPYRIGHT(C) 1992 (1992-2003)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

now making cross reference.
----*----*
Error 3:Can't create file ( TEST.CRF )
```

Figure 3.3 Error Display Example

3.4.2 Values Returned to MS-DOS

When the command to execute CRF45 is coded within a batch file, it may be desirable to change the process flow according to the execution result. CRF45 classifies the execution result into four error levels shown in Table 3.2 and returns it to MS-DOS. Refer to the MS-DOS manual for description on how to use the error level.

Table 3.2 Error Levels

Error Level	Execution Result
0	Normal termination
1	CRF45 target file related error
2	CRF45 command input related error
3	MS-DOS related error

3.5 Environment Variable

CRF45 does not use any environment variables.

APPENDIX A. LIST OF ERROR MESSAGES

A.1 Error Messages

Table A.1 Error Messages

Error No.	Error Message	Description and User Action
1	Can't make cross reference.	CRF45 cannot be invoked. ⇛ Check the memory size.
2	Can't open file (filename).	The file cannot be opened. ⇛ Check the file. ⇛ Check the memory size.
3	Can't create file (filename).	The file cannot be created. ⇛ Check the file. ⇛ Check the memory size.
4	Out of disk space.	The result cannot be output to CRF file. ⇛ Check the availability of free disk space.
5	Out of heap space.	Insufficient work memory. ⇛ Increase the memory by removing device drivers for instance.
6	Can't close file (filename).	The file cannot be closed. ⇛ Check the availability of free disk space.
7	Illegal token (token).	Data in the file is invalid. ⇛ Check the content of the file.

A.2 Warning Messages

Table A.2 Warning Messages

Warning Message	Description and User Action
Nesting error	Include file is nested. ⇛ Change the program to avoid nesting of include files.
Can't open include file (filename).	The include file cannot be opened. ⇛ Check the include file.

APPENDIX B. CRF45 SPECIFICATIONS

B.1 Standard Environment

Table B.1 shows the MS-DOS standard environment used to determine the specification.

Table B.1 MS-DOS Standard Environment

Item	Specification
MS-DOS Version	V.3.1 or above
Memory Size	User memory: 256K bytes This value is determined by the MS-DOS standard command CHKDSK.

B.2 CRF45 Specifications

Table B.2 shows the CRF45 specification under the standard MS-DOS environment. The calculated estimate is shown for some values that cannot be measured.

Table B.2 CRF45 Specification

Item	Specification
Number of labels and symbols	Up to 4900 assuming each label and symbol is referenced once. Up to 1300 assuming each label and symbol is referenced average of ten times. These values depend on the MS-DOS available user memory.

- CRF45 is processed in one pass. A label reference table is created while reading the input file and alphabetically sorted cross reference file is output when the file is read to the end.

Index

MEMO

Renesas Technology Corp.

ASM45 V.1.11 User’s Manual

Rev. 1.00
July 1, 2003
REJ10J0162-0100Z

ASM45 V.1.11 User's Manual

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ10J0162-0100Z