To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

User's Manual

ASM72 V.1.20

User's Manual

Absolute Assembler for 720 Series

Renesas Electronics
www.renesas.com

Rev.1.00 2004.05

Keep safety first in your circuit designs!

- Renesas Technology Corporation and Renesas Solutions Corporation put the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation, Renesas Solutions Corporation or a third party.
- Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation and Renesas Solutions Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation, Renesas Solutions Corporation or an authorized Renesas Technology product distributor for the latest product information before purchasing a product listed herein. The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors. Please also pay attention to information published by Renesas Technology Corporation and Renesas Solutions Corporation by various means, including the Renesas home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation and Renesas Solutions Corporation assume no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation, Renesas Solutions Corporation or an authorized Renesas Technology product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation and Renesas Solutions Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation or Renesas Solutions Corporation for further details on these materials or the products contained therein.

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

Preface

The PD72 is an emulator debugger for the Renesas 720 Series. This manual describes the functions and operations of the three software packages listed below.
The release notes included with this product are supplements of user's manual. Before using this product, be sure to read them.

1. Absolute Assembler ASM72
2. Cross Referencer CRF72
3. Branch-out Error Correction Utility BRANCH

Organization of this manual
This manual consists of the three parts shown below.

- Part 1 ASM72 Operation Manual
  Describes how to operate absolute assembler ASM72 and how to write the source programs.
- Part 2 CRF72 Operation Manual
  Describes how to operate cross-referencer CRF72.
- Part 3 BRANCH Operation Manual
  Describes how to operate branch-out error correction utility BRANCH.

MEMO

Part 1

Absolute Assembler for 720 Series

ASM72 Operation Manual

Table of Contents

Chapter 1. Organization of ASM72 User's Manual

This manual is organized in the following manner.

Chapter 2. Overview
Description of basic features of the ASM72.

Chapter 3. Source Program Coding Methods
Description of coding a source program with the ASM72.

Chapter 4. Pseudo Instruction
Description of the available pseudo instructions.

Chapter 5. Macro Instruction
Description of the available macro instructions.

Chapter 6. Operating Method
Description of the ASM72 system operation.

Appendix A. Error Message List
Error messages of the ASM72 with explanations and processes.

Appendix B. Pseudo Instruction List
ASM72 pseudo instructions with explanations and processes.

Appendix C. Macro Instruction List
ASM72 macro instructions with explanations.

This manual is compliant with the ASM72 V.1.20xx.

Chapter 2. Overview

The ASM72 is the absolute assembler for the Renesas 720 Series. It converts a source program, (hereafter, source file), written in assembly language into machine language. This process is termed "Assembly".

2.1 Features

The ASM72 has the following features:

1. It generates a Tag file[1] which contains the error descriptions. (It is for efficient modification of assembly errors.)
2. It executes the automatic start-up of an editor or cross-referencer by specifying an assembly parameter.
3. It indicates an output object program by specifying an assembly parameter.

2.2 File Generation

The ASM72 generates the following five types of files:

1. Object file (hereafter, HEX file)
2. Print file (hereafter, PRN file)
3. Tag file (hereafter, TAG file)
4. Symbol file (hereafter, SYM file)
5. Map file (hereafter, MAP file)

The following is a description of each file type.

2.2.1 HEX File

- HEX files have machine language data.
- The ASM72 generates the following format files according to each purpose.

1. HEX file (see Figure 2.1)
   - When a command parameter "-V", "-W" or "-WM" is not specified, the object code for piggybacking is divided into 1 low-order bit and 8 high-order bits. The default HEX file is in the order: low-order and high-order.

2. HEU/HEL File (see Figure 2.1)
   - When a command parameter designates "-V", the object code for evaluation board is divided into 1 high-order bit and 8 low-order bits, which default into the HEU and HEL files respectively.

1. "Tag" is derived from a tag label that indicates the location of an error or warning.

3. OTP[2] File (see Figure 2.2)
   - When a command parameter designates "-W", the object code for debugging with the use of a one-time EPROM is divided into 5 high-order bits and 4 low-order bits, which default into the OTP file in the order: low-order and high-order.
   - All empty areas store as a "1".
   - When a program of ROM capacity less than 2K, using a one-time EPROM, is executing, the data in the address locations from 0000_{16} to $07FF_{16}$ is block copied to 0800_{16} and after, while that from 1000_{16} to $17FF_{16}$ is copied to 1800_{16} and after.

4. CAD File (see Figure 2.3)
   - When a command parameter designates "-WM", the object code for writing into a Renesas one-time EPROM is divided into 1 high-order bit and 8 low-order bits, which default into the CAD file in the order: low-order and high-order.
   - All empty areas stores as a "1".
   - When a program of ROM capacity 2K, using a one-time EPROM, is executing, the data in the address locations from 0000_{16} to $07FF_{16}$ is block copied to 0800_{16} and after, while that from 1000_{16} to $17FF_{16}$ is copied to 1800_{16} and after.
   - When a program of ROM capacity 1K, using a one-time EPROM, is executing, the data in the address locations from 0000_{16} to $03FF_{16}$ is block copied to 0800_{16} and after, while that from 1000_{16} to $13FF_{16}$ is copied to 1800_{16} and after. Each 8 bit area in the address locations from 0400_{16} to $07FF_{16}$ and from $0C00_{16}$ to $0FFF_{16}$ is stored as FF_{16}. The high-order 1 bit area in the address locations from 1400_{16} to $17FF_{16}$ and from $1C00_{16}$ to $1FFF_{16}$ are stored as 01_{16}.

- When the command parameter specifies a "-O", the designated directory is used to store these files, otherwise the current directory is used.
- A coding parameter can designate the file extension of .HEU, .HEL, .OTP, .CAD or .HEX.

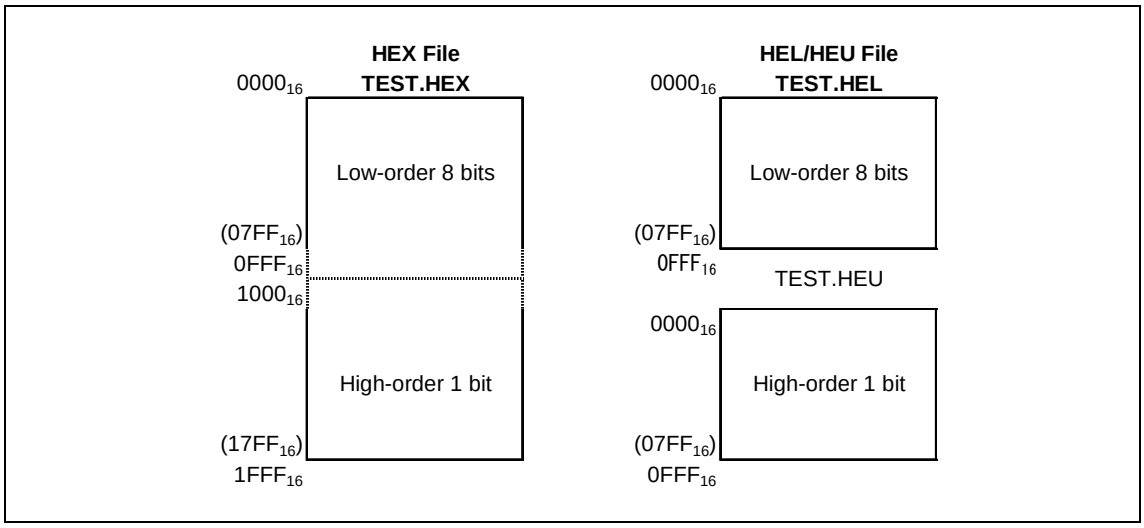


Figure 2.1 HEX File & HEL/HEU File

2. OTP: One-time programmable ROM

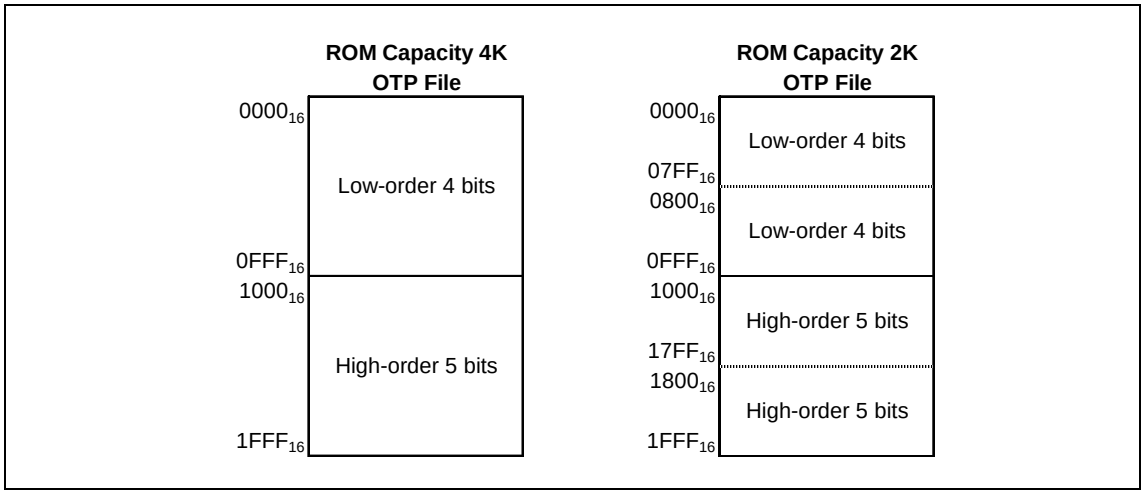


Figure 2.2 OTP File

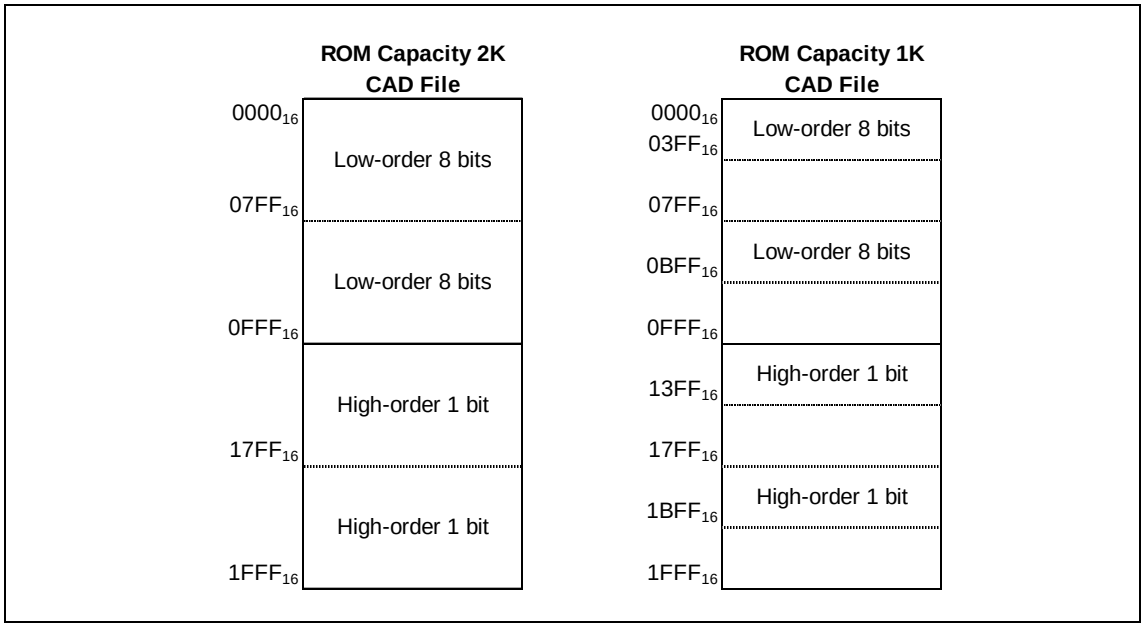


Figure 2.3 CAD File

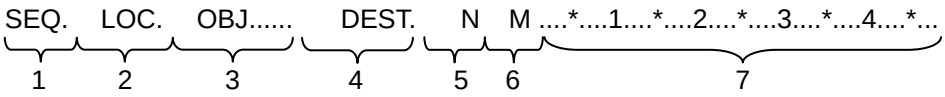
2.2.2 PRN Files

- A PRN file contains the source file for the printing process, the address location, and the generating data.
- The printout of a PRN file can be of use in debugging.
- A PRN file can be created only when the command parameter specifies a "-L".
- When a command parameter contains an "-O", the designated directory is used to store the PRN file, otherwise the current directory is used.
- The file extension is .PRN.

[Configuration of a PRN File]

Figures 2.5 to 2.7 describe the sample output of a PRN file. A PRN file has the following information.

- Symbol list information (Figure 2.5)
  This gives the symbols and labels for the first segment of a PRN file
- Source file information (Figure 2.6)



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. Page and address of the jump destination (DESTINATION)
   The page and address locations of the jump destination are specified as hexadecimal numbers.
5. Nesting information (NEST)
   The nesting location is specified as a '1'.
6. Macro instruction information (MACRO)
   Locations involved in a developing program with the macro instruction is specified with a '+'.
7. Column information of a source file
   Every 5th column, a '*' or number is coded.

- Assembly result information (Figure 2.7)
  The number of errors, warnings, full lines, comment lines, and memory capacity are specified.
- When the pseudo instruction .COL designates the number of columns as 132 characters, the assembly time is specified at the head of the list in the following form:

```
DATE (Mon Jun 3 15:06:42 1989)
```

2.2.3 TAG Files

- Used to store the assembly error messages and warning messages which occur in the assembly process.
- A TAG file can be used as a reference while error correction is performed with the use of an editor.
- A TAG file can be executed only when the command parameter specifies an "-E".
- The file extension is .TAG.

[Configuration of a TAG File]

Figure 2.8 describes a sample output of a TAG file. A TAG file gives the following information.

- Description of the file name, file line numbers, through line numbers, error numbers, and error messages for the location of an error or warning occurrence.

2.2.4 SYM Files

- A SYM file stores the necessary symbolic debugging information for the PD72.
- A SYM file can be executed only when the command parameter specifies an "-S".
- The file extension is .SYM.
- If the command parameter "-O" is also specified, the SYM file is output to the specified directory of the drive; if not specified, it is output to the current directory.
- The following is the configuration for a SYM file.

1. Label information
   Label name and corresponding address value (9 bits).
2. Symbol type 1 information
   Symbols representing the values from 0_{16} to $FFFF_{16}$.
3. Symbol type 2 information
   Symbols representing register values for X and Y.
4. Symbol type 3 information
   Symbols representing register values for Z, X and Y.
5. Symbol type 4 information
   Symbols representing a bit location in a bit symbol and register values for Z, X and Y, which are used to designate a bit in the data page (DP).

Figure 2.9 is a sample output of a SYM file.

2.2.5 MAP Files

- A MAP file contains the memory information for each page.
- A MAP file is output when the command parameter specifies "-A".
- The file extension is .MAP.

```
-M50727
;*******************************************************
;*                                                     *
;*    M50727   TEST  PROGRAM                           *
;*                                                     *
;*******************************************************
;
             .COL   80
             .ORG   0,0
             .TTL   727TEST PROG.                ;SET   TITLE
PORTD1  .EQU  0,0                                ;SET   LABEL
PORTD2  .EQU  0,1
PORTD3  .EQU  0,2
PORTK   .EQU  0,3
;*********************************************
;*    MAIN  ROUTIN                           *
;*********************************************
;
MAIN:        DI                                  ;STOP  INT
             LZ     0                            ;RAM   CLEAR
             LXY    0,0
WAIT:        LA     0
             XAMI   0
             B      WAIT
RAM_X0:      LA     0
             XAMI   0
             B      RAM_X0
             LXY    10
RAM_X1:      LA     0
             XAMI   0
             B      RAM_X1
             LXY    2,0
RAM_X2:      LA     0
             XAMI   0
             B      RAM_X2
             LXY    3,0
             :
             :
```

Figure 2.4 Sample of a Source File

```
M50727 ASSEMBLER SYMBOL TABLES                                          P.001

ATOC      029C  A_D       01B6   COUNT    0100  D0_SET   0262  D10      0253
D10_SET   0280  D1_SET    0265   D2       0206  D2_SET   0268  D3       020F
D3_SET    026B  D4        0218   D4_SET   026E  D5       0221  D5_SET   0271
D6        022A  D6_SET    0274   D7       0233  D7_SET   0277  D8_SET   027A
D9        024A  D9_SET    027D   DH_OUT   025C  DL_OUT   023C  D_OFF1   0083
D_OFF1.1  01A7  D_OFF1.2  01B3   D_OFF2   0093  D_SET1   01C9  D_SET2   01D7
ERR       0191  HENKAN    02B0   IN1      01EE  IN2      01E8  IN3      01E0
LOOP      0027  LOOP1     01BC   LOT      001E  LOTS     0199  MAIN     0000
OUT       01C3  PORTD1    0000   PORTD2   0001  PORTD3   0002  PORTK    0003
RAM_X0    0006  RAM_X1    000A   RAM_X2   000E  RAM_X3   0012  SERIAL   0283
SOUSIN1   028B  SOUSIN2   0295   SUB0     0180  T_1      0109  T_1CNT   010B
WAIT      0003
```

Figure 2.5 Sample of a PRN File (Symbol & Label List)

```
* 720 SERIES ASSEMBLER V.1.00.00C *                              (00 page) P.002

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

   1                       0  ;M50727
   2                       0  ;*********************************************
   3                       0  ;*
   4                       0  ;*      M50727   TEST  PROGRAM
   5                       0  ;*
   6                       0  ;*********************************************
   7                       0                .COL    80
   8                       0                .ORG    0,0
   9                       0                .TTL    727TEST PROG.
test.ASM 9 ( TOTAL LINE 9 ) Error 12: No ';' at the top of comment "727TEST"
  10  0000                 0  PORTD1  .EQU   0,0
  11  0001                 0  PORTD2  .EQU   0,1
  12  0002                 0  PORTD3  .EQU   0,2
  13  0003                 0  PORTK   .EQU   0,3
  14                       0  ;*********************************************
  15                       0  ;*      MAIN  ROUTINE
  16                       0  ;*********************************************
  17  0000  004            0  MAIN:          DI
  18  0001  048            0                LZ      0
  19  0002  0C0            0                LXY     0,0
  20  0003  0B0            0  WAIT:         LA      0
  21  0004  068            0                XAM     0
  22  0005  183      00/03 0                B       WAIT
  23  0006  0B0            0  RAM_X0:       LA      0
  24  0007  068            0                XAMI    0
  25  0008  186      00/06 0                B       RAM_X0
  26  0009  000            0                LXY     10
test.ASM 26 ( TOTAL LINE 26 ) Error 5: Improper operand type "10"
  27  000A  0B0            0  RAM_X1:       LA      0
  28  000B  068            0                XAMI    0
  29  000C  18A      00/0A 0                B       RAM_X1
  30  000D  0E0            0                LXY     2,0
  31  000E  0B0            0  RAM_X2:       LA      0
  32  000F  068            0                XAMI    0
  33  0010  18E      00/0E 0                B       RAM_X2
  34  0011  0F0            0                LXY     3,0
  35  0012  0B0            0  RAM_X3:       LA      0
  36  0013  068            0                XAMI    0
  37  0014  192      00/12 0                B       RAM_X3
  38  0015  0C0            0                LXY     0,0
```

Figure 2.6 Sample of a PRN File (first half)

```
* 720 SERIES ASSEMBLER V.1.00.00C *                              (05 page) P.013

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

 413  02BF  0B0            0              LA      0
 414  02C0  082            0              TCA
 415  02C1  044            0              RT
 416                       0              .END

ERROR   COUNT 0003
WARNING COUNT 0000
SOURCE  LINE  0416 LINES
TOTAL   LINE  0416 LINES
COMMENT LINE  0043 LINES
OBJECT  SIZE  0395 BYTES
```

Figure 2.7 Sample of a PRN File (second half)

```
test.ASM 9 ( TOTAL LINE 9 ) Error 12: No ';' at the top of comment "727TEST"
test.ASM 26 ( TOTAL LINE 26 ) Error 5: Improper operand type "10"
test.ASM 53 ( TOTAL LINE 53 ) Error 19: Reference to undefined label "F"
```

Figure 2.8 Sample of a TAG File

```
#MELPS720
#EQU            << Symbol type 1 information
FALSE 0000      TRUE 0001      ZERO 0000
#SYMBOL         << Symbol type 2, 3, 4 information
BANKO *000      BANK1 *100     CNT1 **00       CNT2 **01      EVF 0002
OVF 2002        TSF 3002
#LABEL          << Label information
RAMCL 0125      RAMCL0 0121    RAMCL1 0122     RAMCL2 0123    RAMCL3 0124
```

Figure 2.9 Sample of a SYM File

Chapter 3. Source Program Coding Methods

3.1 Source Program Configuration

A source program is composed of lines put together as a unit. The following are the rules for coding lines.

- An instruction cannot be specified on more than one line.
- The total number of characters on a line can be a maximum of 100. The ASM72 ignores the characters past 100.
- Lines can be classified by contents, into the following six types.

1. MCU Name Specification Line
   This line specifies the MCU name of your 720 Series MCU.
2. Assembly Language Instruction Line
   This line is an ASM72 assembly language instruction, which generates the corresponding machine language.
3. Line of Addition/Subtraction to a Bit Symbol
   This line describes an addition/subtraction to a bit symbol
4. Pseudo Instruction Line
   This line describes a pseudo instruction.
5. Macro Instruction Line
   This line describes an ASM72 macro instruction, which develops into the corresponding assembly language.
6. Comment Line
   Because the ASM72 does not process this line, the user can use this line freely.

3.2 Line Configuration

This section discusses the configuration of each line. The following are explanations, or rules, for the symbols described on a line.

1. A white (△) or black (▲) triangle represents a space or tab code. A white one △ is essential, but you may omit a black one ▲.
2. When you describe a label name, you need not specify the ':' (colon).
3. A space or tab code is required between a label and an instruction.

3.2.1 MCU Name Specification Line

The configuration of an MCU name specification is described as follows:

-MCU name △ ;comment <RET>

1. MCU Name Column
   - Specifies an MCU name.
   - Be sure to describe '-' (hyphen) before the MCU name.
   - The MCU name can be set by the -M option in an assembly process.

2 Comment Column

- The ASM72 does not process this column. It can be used for narrative information.
- The first character of a comment column must be a ';' (semi-colon).
- When an entire statement is a comment, the ';' must be the first character in the line. All character after a ';' are regarded as a comment.

3.2.2 Assembly Language Instruction Line

The configuration of an assembly language instruction is described as follows:

Label △ Op-Code △ Operand △ ;Comment <RET>

1. Label Column
   - Specifies a label used to reference the line from another location.
   - Specifies a character line which contains up to fifteen alphanumeric characters and the special characters, '_' (underscore), '.' (period), and '?' (question mark).
   - The first character of a label must be an alphabet character or a special character, as the '_' (underscore), '.' (period), and '?' (question mark). Other characters, including numeric characters other than the above, cannot be used.
   - Upper-case and lower-case letters of a label can be distinguished.
     Example: 'BIG' and 'big' are judged as different labels.

2. Op-code Column
   - Specifies an assembly language mnemonic (hereafter, Op-code).
   - Upper-case and lower-case letters of an Op-code cannot be distinguished. For example, both 'NOP' and 'nop' are the same.

3. Operand Column
   - Specifies an operand to process.
   - When the operand covers more than two data points, a ',' (comma) must be specified between the data.
   - A space or tab code can be used on both sides of the comma.

4. Comment Column
   - The ASM72 does not process this column. It can be used for narrative information.
   - The first character of a comment must be specified by a ';' (semi-colon).
   - When an entire statement is a comment, the ';' (semi-colon) must be the first character in the line. All characters after a ';' (semi-colon) are regarded as a comment.

3.2.3 Line of Addition/Subtraction to a Bit Symbol

The following is the configuration of a line of addition/subtraction to a bit symbol.

Instruction △ Bit symbol △ Operator △ Bit symbol
Instruction △ Bit symbol △ Operator △ Symbol/numeric ,Symbol/numeric ・・・

1. Instruction Column
   - Describes a macro instruction or mnemonic operable to a bit symbol.
     Macro instruction: .clb .lzxy .seb .szxyb
     Mnemonic: lz lxy rb sb szb

2. Bit Symbol Column After Instruction Column
   - Specifies a bit symbol.

3. Operator Column
   - Operator can be described by addition (+) or subtraction (-).
   - Only one operator can be described in one line.

4. Bit Symbol Column After Operator Column
   - Specifies XY symbol, ZXY symbol or bit symbol.
   - Only one bit symbol can be described in one line.

5. Symbol/Numeric Column after Operator Column
   - Specifies a symbol/numeric.
   - The maximum number of symbols/numerics which can be specified is four.
   - If one symbol/numeric is specified, it is processed as Y register.
   - If two symbols/numerics are specified, the first is processed as Z register, the second as Y register, respectively.
   - If three symbols/numerics are specified, the first is processed as Z register, the second as X register, the third as Y register, respectively.
   - If four symbols/numerics are specified, the first is processed as a bit location, the second as Z register, the third as X register, the forth as Y register, respectively.
   - Describe "," (comma) between numerics.
   - A numeric between commas can be left out.
   - The following cases result in an error.
     - When the result of an addition exceeds each value of registers and a bit location
     - When the result of a subtraction is a minus value
     - When there is no operand after an operator
     - When more than one bit symbol is specified in the bit symbol column after the operator column
     - When more than four symbols/numerics are specified in the symbol/numeric column after the operator column

3.2.4 Pseudo Instruction Line

The following is the configuration of a pseudo instruction line. Refer to Chapter 4 and Appendix B for further information.

Symbol type 1-4 △ .EQU △ Operand △ ;Comment <RET>
Label △ Pseudo instruction △ Operand △ ;Comment <RET>

1. Symbol Type 1-4 Column
   - Describes the symbol and bit-symbol which assigns the value by the pseudo instruction EQU.

2. Label Column
   - Refer to the section on assembly language instructions.

3. Pseudo Instruction Column
   - Describes pseudo instructions of the 720 Series.
   - A pseudo instruction does not distinguish between upper-case and lower-case characters. For example, both .END and .end are equivalent.

4. Operand Column
   - Describes the corresponding process target of a pseudo instruction.
   - Refer to "3.2.2 Assembly Language Instruction Line".

5. Comment Column
   - Refer to "3.2.2 Assembly Language Instruction Line".

3.2.5 Macro Instruction Line

The following is the configuration of a macro instruction line. Refer to Chapter 5 and Appendix C for further information.

Label: △ Macro instruction △ Operand △ ;Comment <RET>

1. Label Column
   - Describes the label (name) which can be used to refer to the macro instruction line from another location.

2. Macro Instruction Column
   - Describes a macro instruction of the 720 Series. Macro instructions cannot distinguish between upper-case and lower-case characters. For example, .CLB and .clb are the same.

3. Operand Column
   - Describes the corresponding process target of a macro instruction.
   - Refer to "3.2.2 Assembly Language Instruction Line".

4. Comment Column
   - Refer to "3.2.2 Assembly Language Instruction Line".

3.2.6 Comment Line

The first character of a comment line must be a ';' (semi-colon). The following is the configuration of a comment Line.

▲ ;Comment <RET>

3.3 Column Description

This section discusses the common coding method columns of each instruction line. For more information on the different coding method columns, refer to Chapters 4 and 5.

3.3.1 Symbol Type 1 through 4/Label Column

For these lines, the coding method of an instruction line is common. However, the ASM72 distinguishes between symbol type 1 through 4 and a label[1] for management.

1. The name can be up to fifteen characters, and can contain alphanumeric characters and special characters, such as the '_' (underscore), '.' (period), and '?' (question mark). However, the first character of the line must be an alphabetic character or a special character.
2. Upper-case and lower-case characters can be distinguished. For example, BIG and big are recognized as different names.

When you describe a label, you can add a ':' (colon) after the label. It is recommended that a colon be specified to distinguish the symbols easily, and to make searching for labels with an editor more effective.

3.3.2 Comment Column

Used to describe any type of information. The following is the coding method of the comment column.

1. Must be started with a ';' (semi-colon).
2. Any characters can be used in the comment column.

3.4 Operand Data Format

The following four data formats are possible in the operand notational convention

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

3.4.1 Numeric Constants

- A space or tab key must not be placed between a numeric type symbol and a numeric value.

  Example .DW $ 64 >> Error

- Binary, octal, decimal and hexadecimal notations can be used for a numeric constant.

- When the description is from binary to binary, a '%' must precede the constant or a 'B' or 'b' must follow it.

  Examples .DW %100110
  .DW 100110B

1. Symbol Type 1-4 is defined by pseudo instruction .EQU or parameter "-D". Any others are processed as labels.

- When the description is from octal to octal, an '@' must precede the constant or a 'O', 'o', 'Q', or 'q' must follow it.

  Examples .DW @70
  .DW 70O
  .DW 70Q

- When the description is from decimal to decimal, no designation is necessary. However, an integral number, such as 23 or 256, is required.

  Example .DW 100

- When the description is hexadecimal to hexadecimal, a '$' must precede the constant or an 'H' or 'h' must follow it. When the first letter is an alphabetic A through F, a '0' must precede it.

  Examples .DW $64
  .DW 64H
  .DW 0ABH

3.4.2 Character Constants

- Descriptions of up to 16 characters, defined by the ASCII code, can be used.
- A character constant must be quoted by a ' (a single quotation mark). Each character corresponds to a 7 bit ASCII code (the high-order bit is 0.)

  Example .DW 'A' >> Encode 41H.

3.4.3 Symbol Constants

- Symbols have five kinds: a label and symbol types from 1 to 4.
  - A label has a ROM Address.
  - Symbol type 1 has an absolute value.
  - Symbol type 2 has a symbol name and register values for X and Y.
  - Symbol type 3 has a symbol name and register values for Z, X and Y.
  - Symbol type 4 has a bit location in a bit symbol and register values for Z, X and Y. A bit symbol has an assigned bit value and an address of the bit.

  Example B MAIN >> Branch to MAIN

3.4.4 Expressions

- An expression is configured by the combination of a numeric constant, character constant, symbolic constant and operand. A space or tab can be placed between the operand and each term.

  Example TBL + 1

- An expression is calculated from left to right. Brackets () specify a priority level.

  Example 2+6/2 >> Result is four.
  2+(6/2) >> Result is five.

3.5 Special Characters

Table 3.1 lists the special characters available for specifying operand data.

Table 3.1 Special Character List

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

Note:

When your system does not have the \ (Reverse slash), the ¥ mark can be used instead.

3.6 Operator

Table 3.2 lists the operators available for specifying operand data.

Table 3.2 Operator List

Class	Operator	Definition
Monadic operator	!	Deduct the ones complement.
	<	Retrieve high-order 8 bits from a label or symbol.
	>	Retrieve low-order 8 bits from a label or symbol.
	#	Retrieve page (high-order 1 bit) from a label or symbol.
Binary operator	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	&	AND for each bit
	|	OR for each bit
	^	NOT-IF-THEN operation for each bit
	()	Designation of priority level for an operation

Note:

Operations on bit symbols are not possible.

Chapter 4. Pseudo Instruction

4.1 Pseudo Instruction Features

Pseudo Instructions tell the ASM72[1] to generate machine language as an object. The ASM72 has twelve kinds of pseudo instructions, which are classified into the following three groups.

1. Assembly Control
   - This group of pseudo instructions does not generate data, but controls the flow of the assembly process.
   - They do not update the address.
   - This group includes the following three pseudo instructions.

.END	Declaration of assembly completion
.IF (.ELSE) .ENDIF	Conditional assembly
.INCLUDE	File loading

2. Address Control
   - Data setting pseudo instructions generate constant data
   - They update the address.
   - This group includes the following three pseudo instructions.

.EQU	Synonymous definition
.ORG	Address declaration
.DW	Data establishment

3. List Controls
   - They control the output of a PRN file.
   - This group includes the following six pseudo instructions.

.COL	List column format designation
.LINE	List line format designation
.LIST	List output
.NLIST	List suppress designation
.PAGIE	New page designation
.TTL	Title designation

The following are the features of pseudo instructions, as classified into groups.

4.2 Assembly Control

4.2.1 Declaration of Assembly Completion

.END

- Declares the end of a source program.
- The ASM72 does not process after this line.

---

1. The indication of a pseudo instruction can be classified into a "Declaration" to instruct the ASM72, and a "Command" to effect the output file.

4.2.2 Conditional Assembly

.IF - (.ELSE) - .ENDIF

- According to the specification of the symbolic value, it designates the assembling locations.
- When a source program manages programs that are applicable for multiple specifications, it controls the test routine assembly.

4.2.3 File Loading

.INCLUDE

- Instructs the loading of other files into the, location where this instruction is specified.
- Can be used to partially edit a large source.

4.3 Address Control

4.3.1 Synonymous Definition

.EQU

- Defines an absolute value for a symbol.
- Defines values of X and Y registers for a symbol.
- Defines values of Z, X and Y registers for a symbol.
- Defines bit location of Z, X and Y register values.

4.3.2 Address Declaration

.ORG

- Declares the address after the specified line.

4.3.3 Data Establishment

.DW

- Generates data coded by the operand in the ROM area.

4.4 Listing Control

4.4.1 Page and Title Designation

.PAGE, .TTL

- Designates a new page and the title of a list.

4.4.2 Listing Format Designation

.COL, .LINE

- Designates the number of a column and line.
- You can specify these pseudo instructions only once in a source file.

4.4.3 Listing Output Suppress Designation

.LIST, .NLIST

- Controls the listing output to a PRN file.
- These instructions can be used when only part of a list is required for debugging the program, or for other purposes.

Chapter 5. Macro Instruction

5.1 Macro Instruction Features

Multiple assembly language instructions for the 720 Series are available through the use of a macro instruction, in the same way as for an Op-code. The ASM72 has four pseudo instructions, which are classified into the following two groups.

1. Bit Macro Instructions
   - Operates on the bits of a bit symbol designated by the operand.
   - This group includes the following three macro instructions

   .CLB Clearing of the specified bit
   .SEB Setting of the specified bit
   .SZXYB Changing of the program sequence in accordance with the specified bits

2. Register Macro Instructions
   - Sets values of Z, X and Y to each register, in accordance with the bit symbols specified by the operand.

   .LZXY Setting of the specified value of Z, X and Y registers.

Macro instruction features are classified into the following groups and are discussed respectively.

5.2 Bit Macro Instructions

5.2.1 Clearing a Specified Bit

.CLB

- Clears the bit of a bit symbol specified by the operand.

5.2.2 Setting a Specified Bit

.SEB

- Sets the bit of a bit symbol specified by the operand.

5.2.3 Changing the Program Sequence According to a Specified Bit

.SZXYB

- If the test performed on the bit of a bit symbol specified by the operand should result in a zero, the program sequence is changed.

5.3 Macro Register Instructions

5.3.1 Setting Values of Z, X and Y

.LZXY

- Sets values of Z, X and Y to each register in accordance with the specified operand.

Chapter 6. Operating Method

6.1 Getting Started

Before starting the ASM72, the following information (input parameters) should be set.

1. Name of the source file (essential)
2. Command parameters

The ASM72 requires this information in a DOS command line. Section 6.2 discusses the input parameters, while Section 6.3 describes how to input the command line by giving examples.

6.2 Parameter Input

6.2.1 Source Filename

1. The source filename for assembly must be designated. There can be only one
2. When the file attribute (.ASM) is omitted, the ASM72 selects .ASM as the default attribute value.
3. By specifying the full name, the assembly of files from extension .ASM (example: .SRC) is also possible.
4. The designation of a directory path, included in the file name, is possible. When you only indicate a file name, the file processed is the one in the current directory of the current drive.

   Example `A>ASM72 C:¥WORK¥TEST<RET>`

6.2.2 Command Parameter

1. Upper or lower case letters can be used for command parameters
2. Multiple definitions are possible in a command parameter. In this case, a space should be inserted between the parameters.

Tables 6.1 and 6.2 list the definitions of command parameters.

Table 6.1 Command Parameter List

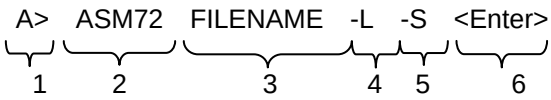
Command Parameter	Definition
-A	Generates a MAP file. Example: A>ASM72 FILENAME -A<RET>
-C	Outputs a source line debug information to a symbol file. Example: A>ASM72 FILENAME –C -S<RET>
-D	Defines an assembler symbol. This parameter has the same function as pseudo instruction .EQU. The syntax is formed as follows. (When you define many symbols simultaneously. Insert a ':' between each symbol) -D SYMBOL=VALUE[:SYMBOL=VALUE.....:SYMBOL=VALUE] Example: A>ASM72 FILENAME -DSYMBOL1=10:SYMBOL2=20<RET>
-E	Generates a tag file and starts the editor. The following syntax specifies the program name of the editor. -E[Program Name of Editor] Example: A>ASM72 FILENAME -EMl<RET> You can omit the program name of the editor inside the [ ]. When the name is omitted, it generates a tag file. When the name is specified, it starts the editor with the option of a tag file after assembly completion. However, if no error has occurred, it suppresses the start of the designated editor.
-L	Generates a PRN file. Without this parameter, the ASM72 omits the generation of a PRN File.
-M	Designates the assembly MCU's name. Without this parameter, the ASM72 refers to the indication of the MCU specified in the first line of the source file. Example: A>ASM72 FILENAME -M50720<RET>
-O	Designates the path where the generated file is targeted. A directory or drive name can be indicated in the path. However, without this parameter, the ASM72 targets the same path as the source file. -O path name Example: A>ASM72 FILENAME -OB:¥USR<RET>
-P	Designates the directory and drive where the data file (MXXXXX.DAT) is stored. -P drive name Example: A>ASM72 FILENAME -PB:¥USR<RET>
-R	Outputs the program developed by bit macro pseudo instructions into a PRN file. Example: A>ASM72 FILENAME -R<RET>
-S	Outputs a symbol file for the PD72. Example: A>ASM72 FILENAME -S<RET>
-V	Outputs HEU (high-order 1 bit of a HEX file) and HEL (low-order 8 bits of a HEX file) for an evaluation board. Example: A>ASM72 FILENAME -V<RET>
-W	Outputs an EPROM OTP file. Example: A>ASM72 FILENAME -W<RET>
-WM	Outputs the high-order 1 bit and the low-order 8 bits of a CAD file of the EPROM version. Example: A>ASM72 FILENAME -WM<RET>
-X	After assembly completion, starts cross-referencer CRF72[1]. Example: A>ASM72 FILENAME -X<RET>

[Notes]

1. When the CRF72 does not exist in the current directory or command path, a system error occurs.
2. The combination of command parameters -V, -W, and -WM is not permitted.

6.3 Input Method

Starting the ASM72 is performed by the input of a command line at the DOS prompt. The following is an example of a boot-up command.



1. DOS prompt
2. ASM72
3. Source filename of the assembly target
4. Command parameter -L generates a PRN file.
5. Command parameter -S generates a SYM file.
6. Return key

When an error is detected in the command line, assembly is suppressed, with the help display as shown below.

```
C:¥>ASM72<Enter>
720 SERIES ASSEMBLER V.1.20.02C
COPYRIGHT(C) 1989 (1989-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

Usage: ASM72 <filename> [Options...]

 -A  : make memory Area information.( output MAP file )
 -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 -M50720 )
 -O  : select drive and directory for output ( use -Oa:¥work )
 -P  : select directory(drive) of mxxxxx.dat file.( use -P¥work )
 -R  : output bit macro expansion
 -S  : make symbol file for symbolic debugger
 -V  : make file for eva.chip
 -W  : output hex data for EPROM version ( lower 4bit, upper 5bit)
 -WM : output hex data for EPROM version ( lower 8bit, upper 1bit)
 -X  : execute crf72
```

Figure 6.1 Help Display after a Command Error

When assembly is complete, the number of errors, warning lines, comment lines, and memory capacity are displayed. Figure 6.2 is an example of a display after a normal assembly completion.

```
D:¥>ASM72 TEST<Enter>
720 SERIES ASSEMBLER V.1.20.02C
COPYRIGHT(C) 1989 (1989-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
now processing pass 1
----*----*
now processing pass 2
----*----*
ERROR   COUNT 0000
WARNING COUNT 0000
SOURCE  LINE  1028 LINES
TOTAL   LINE  1028 LINES
COMMENT LINE  0000 LINES
OBJECT  SIZE  1024 BYTES
```

Figure 6.2 Display after a Normal Completion

6.4 Errors

6.4.1 Error Types

Errors which occur in the ASM72 assembling are classified into the following causes.

1. Errors related to DOS
   This relates to the lack of memory capacity, DOS operation environment when executing the ASM72, etc. Please refer to the error messages listed in Appendix A and process them with a DOS command.

2. Errors related to ASM72 command indications
   This relates to the command line indications when starting the ASM72. Please refer to this chapter and execute the command line again.

3. Errors related to the source file
   This relates to the contents of a source file; as multiply defined, reference to undefined symbols, etc. Please modify the indicated part of the source file and assemble again. If the ASM72 detects an error, HEX files will not be generated properly.

If the ASM72 detects an error or warning, the following error content (filename, line number in the file, through line number, error number and error message) is output to the display and PRN file. Please apply the necessary procedure by referring to the error messages listed in Appendix A.

```
D:¥>ASM72 TEST<Enter>
720 SERIES ASSEMBLER V.1.20.02C
COPYRIGHT(C) 1989 (1989-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
now processing pass 1
----*----*
now processing pass 2
----*----*
1028  0401  102       08/02 0         BM sub1
TERST.ASM 1028 ( TOTAL LINE 1028 ) Error 21: Value is out of range "sub1"

ERROR   COUNT 0001
WARNING COUNT 0000
SOURCE  LINE  1031 LINES
TOTAL   LINE  1031 LINES
COMMENT LINE  0000 LINES
OBJECT  SIZE  1026 BYTES
```

Figure 6.3 Example of an Error Description

6.4.2 Return Value to DOS

When you describe an execution command in a DOS batch file, sometimes you have to change the transaction process according to the results of the execution. The ASM72 classifies these execution results into four error levels, as shown below.

Table 6.3 Error Level List

Error Level	Execution Result
0	Normal completion
1	ASM72 source file error
2	ASM72 command description error
3	DOS error

Appendix A. Error Message List

A.1 System Errors

When the ASM72 detects a system error during assembly, it shows an error message on the display and suppresses the assembly. Table A.1 is a list of the system error messages.

[Note]
The total number of symbols and labels that can be used in an ASM72 assembly depends on the system memory capacity available to execute the assembly.

Table A.1 System Error Message List

Error Message	Troubleshooting
Usage: ASM72 <filename> [Options...]	Syntax error in the command line. >> Refer to the help display and input a new command line.
Cannot open xxx	Cannot find a source file. >> Make sure of the source file name and input again.
Cannot create xxx	Indicates the file can not be generated. >> Make sure of the -O parameter and input again.
Out of disk space	Not enough disk capacity to output the file. >> Delete an unnecessary file from the diskette.
Out of heap space	Not enough memory to operate the assembler. >> Reduce the number of symbols or labels.
Can't find crf72.exe	CRF72 can not be found. >> Copy the CRF72.exe file into the current directory or DOS command path directory.
Can't find command.com to execute xxx	COMMAND .COM needed to boot the designated editor by the -E option, cannot be found. >> Make sure of the directory path description for DOS commands.
Multiply defined CPU name	Multiply defined CPU name both in command line and the source file. >> Eliminate all but one of the descriptions.
M34xxx.DAT not found	A data file in the indicated directory for the specified MCU was not found. >> Make sure that the specified data file exists.

A.2 Assembly Errors

When the ASM72 detects an assembly error, it outputs an error message on the display and in the PRN file. Tables A.2 through A.6 show the assembly errors and their descriptions.

Table A.2 Assembly Error List (No. 1)

Error Number	Error Message	Troubleshooting

1	Already had same statement	Pseudo instruction is used more than two times. Can only be used once in a source file.
		Example `.LINE 60`
		`:`
		`.LINE 80`
		>> Declaration should be used only one time.

2	Reference to backward label or symbol	Pseudo instruction refers to a later symbol or label. Example `.ORG TOP` `TOP:` >> Reference of label and symbol is set after definition.
3	Division by 0	A numeric expression includes division by 0. >> Make sure of the numeric expression value.
4	Illegal operand	An operand includes an unavailable character. Example `LDA #&10` >> Make sure of the operand description.
5	Improper operand type	The combination of mnemonic and operand is incorrect. Example `ASL work,Y` >> Make sure of the instruction syntax form.
6	Invalid label definition	Defined label is executed at an unavailable location. Example 1) `LABEL1: .LINE 60` >> Delete the label. Example 2) `LABEL2: .EQU 100` >> Change the label to a symbol.
7	Out of maximum program size	Address is out of the maximum program size. Example `.ORG 0FFFFH` `.WORD 1,2,3,4,5,6,7,8,9` >> Change the program address to be within the maximum program size.
8	Label is multiply defined	The same label or symbol is defined more than once. Example `MAIN: NOP` `MAIN: NOP` >> Make sure of the label and symbol names.

Table A.3 Assembly Error List (No. 2)

Error Number	Error Message	Troubleshooting

9	Nesting error	The pseudo instruction .IF or .INCLUDE is nested.
		Example.
		`IF DATA1`
		`.IF DATA2`
		`:`
		`.ENDIF`
		`.ELSE`
		`:`
		`.ENDIF`
		>> Change the program to avoid nesting.

10	No .END statement	No .END pseudo is stated in the source file. >> Place an .END pseudo at the end of the program.
11	No label definition	Symbol is not defined. Example `.EQU 60` >> Define the symbol.
12	No ‘;’ at the start of a comment	No ';' (semi colon) is at the start of a comment. Example `LDA #CNT counter set` >> Place a ';' at the start of a comment.

13	Not in conditional block	.ELSE or .ENDIF pseudo is placed in a block where the corresponding .IF pseudo is not defined.
		(If the corresponding .IF pseudo is defined as an error, this error message will appear.)
		Example
		`.IF DATA1`
		`:`
		`.ENDIF`
		`:`
		`.ELSE`
		`:`
		`.ENDIF`
		>> Make sure of the .IF pseudo condition.

14	Operand is expected	Operand for an instruction is missing. Example `.BYTE` >> Make sure of the instruction operand.
15	Questionable syntax	A mnemonic instruction is misspelled. Example `ADD #DATA` >> Make sure of the mnemonic instruction spelling.

16	Reference to a multiply defined label	Reference is made to multiply defined label or symbol.
		Example
		`MAIN: NOP`
		`MAIN: NOP`
		`BRA MAIN`
		>> Make sure of the label or symbol name.

17	Relative jump is out of range	The location addressed by a relative jump instruction is out of range. >> Relocate the program or change the instruction.
18	Label is a reserved word	A label and symbol have the same reserved word. Example `A .EQU 1FFH` >> Change the name of the label or symbol.

Table A.4 Assembly Error List (No. 3)

Error Number	Error Message	Troubleshooting
19	Reference to an undefined label	Reference is made to an undefined label or symbol. >> Make sure of the label or symbol.
20	Value error	Data value is invalid. Example `ADC #'A` >> Make sure of the data description form.
21	Value is out of range	Data value is out of range. Example `ADC #100H` >> Make sure of the operand description form.
22	"()" format error	The number of open and close parenthesis is unequal. Example `ADC (WORK` >> Make sure of the operand description form.
23	Label error	A label has more than sixteen characters or includes invalid characters. Example `L123456789012345:` >> Make sure of the label.

A.3 Warnings

When the ASM72 detects a warning, it outputs a warning message to the display and PRN file. Table A.5 is a list of the warnings and transactions with their descriptions.

Table A.5 Warning List

Warning Number	Warning Message	Troubleshooting

1	Phase warning	1) The address designated by an .ORG pseudo instruction is bigger than the address designated by an .ORG pseudo instruction
		Example
		`.ORG 0E000H`
		`MAIN: LDA WORK`
		`:`
		`.ORG 0C000H`

		2) The instruction refers to the label or symbol after this line.
		Example
		`LDA WK,X`
		`:`
		`WK .EQU 80H`
		>> Define the label or symbol before the line referred.

2	.END statement in included file	Pseudo instruction .END is stated in included file. >> Place .END in the source file only.

Appendix B. Pseudo Instruction List

B.1 Overview of the Pseudo Instruction List

ASM72 Pseudo Instructions are listed in alphabetical order. The following are notational rules for pseudo instructions.

1. The portion inside the [ ] can be omitted.
2. A white triangle △ means a space or tab code.
3. In the following description, a white triangle △ is inserted between the label and pseudo instructions. A label description does not always require a ':' (colon).
4. A space or tag code should be inserted between a label and a pseudo instruction.

B.2 Pseudo Instruction List

.COL Indicating Columns (default 132)

Syntax

△ .COL △ Value

Description

- Designates the number of characters in a line (80 or 132).
- Numbers designated below 80 are regarded as 80, while those beyond 81 are regarded as 132.
- This instruction can be used only once in a program.

Example

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

DW Setting up data

Syntax

△ Label △ .DW △ Expression

Description

- Sets up the value of an expression (Unit: word).
- When you set more than two data values, each must be separated by a comma ','.
- The maximum number of values in a line is sixteen.

Example

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

.END Declaring the End of a Program

Syntax

△ .END

Description

- This instruction indicates the end of a source program.
- The ASM72 does not assemble any lines after this pseudo instruction.

Example

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

Syntax 1

Symbol type 1 △ .EQU △ Value

Syntax 2

Symbol type 1 △ .EQU △ Value X, Value Y

Syntax 3

Symbol type 1 △ .EQU △ Value Z, Value X, Value Y

Syntax 4

Symbol type 1 △ .EQU △Bit Location, Value Z, Value X, Value Y

Description

- Assign a value to the symbol on the left side.
- Syntax 1 assigns an integral value of sixteen bits for the symbol code.
- Syntax 2 assigns the register values of X and Y for the symbol code to be used by the LXY instruction.
- Syntax 3 assigns the register values of Z, X and Y. LXY, LZ and macro pseudo (.LZXY) instructions can be used.
- Symbol 4 assigns a bit value from 0 to 3, register values of Z, X and Y. LXY, LZ, SB, RB, SZB and macro instructions (.LZXY, .CLB, .SZXYB, .SEB) can be used.
- Symbols for the values input must be defined before this line.

Example

```
COUNT   .EQU    4         ; Syntax 1
TYPE    .EQU    3,0       ; Syntax 2
DATE5   .EQU    0,3,4     ; Syntax 3 (No indication of bit location)
FLAG0   .EQU    1,0,3,4   ; Syntax 3 (Indication of bit location)
        LXY     TYPE      ; X=3,Y=0
        LXY     DATE5     ; X=3,Y=4
        LXY     FLAG0     ; X=3,Y=4
        LZ      DATE5     ; Z=0
        LZ      FLAG0     ; Z=0
        SB      FLAG0     ; Set the 1st bit to 1 in DP.
        RB      FLAG0     ; Set the 1st bit to 0 in DP.
        SZB     FLAG0     ; If the 1st bit in DP is 0, skip the next instruction.
```

.IF (.ELSE) .ENDIF — Conditional Assembly

Syntax

△ .IF △ Expression
Instruction 1
△ .ELSE
Instruction 2
△ .ENDIF

Description

- A label defined by a synonymous instruction of syntax type 1 and the -D parameter can be described in the expression.
- If the expression following .IF is true, the ASM72 assembles instruction 1. If false, it assembles instruction 2.
- This pseudo instruction can not be nested.
- You can describe instructions in a column.
- Labels or symbols using in the expression must be defined before this line.

Example

```
.IF     FLAG    ;       If the value of FLAG is true, the ASM72
  :                       performs the assembly until .ELSE.
  :
.ELSE           ;       If false, it assembles until .ENDIF
  :
  :
.ENDIF
```

.INCLUDE — Loading a File

Syntax

△ .INCLUDE △ Filename

Description

- The ASM72 loads the contents of the file specified by an operand to the location this pseudo instruction is described.
- The filename must be specified in full.
- A path indication can be used in the filename.

Example

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

.LINE Indicates the Number of Lines per page (Default 54)

Syntax

△ .LINE △ Value

Description

- This instruction designates the number of lines per page (5 to 255).
- 4 header lines are included.
- This pseudo instruction can be used only once in a program
- Symbols for an operand must be defined before this line.

Example

```
.Line   60      ;       Set the number of lines to 60
  :
  :
```

.LIST List output start (Default)

Syntax

△ .LIST

Description

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

Example

```
.NLSIT          ;       Output of list stopped.
  :             ;       No output to PRN file until ".LIST" is designated.
  :
.LIST           ;       Output of lists started.
  :             ;       Data following this pseudo instruction is output to PRN file.
  :
```

.NLIST List Output stop

Syntax

△ .NLIST

Description

- This instruction stops output to the list file.
- If you want to start output to the list file again, pseudo .LIST can be used.

Example

```
.NLIST          ;       Output of list stopped.
  :             ;       No output to PRN file until ".LIST" is designated.
  :
.LIST           ;       Output of lists started.
  :             ;       Data following this pseudo instruction is output to PRN file.
  :
```

.ORG Declaring Address (Default 0000H)

Syntax 1

△ .ORG △ Address

Syntax 2

△ .ORG △ Page, Offset

Description

- Declares the starting address after this line.
- With no indication, it uses a starting address of 0000H.
- Labels or symbols used in the expression must be defined before this line.

Example

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

.PAGE Forward the list page and specify a Title

Syntax

△ .PAGE △ ['Title']

Description

- Forwards the list page just before this instruction and outputs the title indicated by the operand at the head of the list.
- The title must be specified with ' (single quotations) or " (double quotations).
- If the column is designated as 80, the maximum number of characters is 20. If 132, the maximum number of characters is 30. If the title is omitted, only a page forward occurs.

Example

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

.TTL Indicating a Title

Syntax

△ .TTL △ ['Title']

Description

- Forwards a page just before this instruction and outputs the title indicated by the operand at the head of the list.
- Title must be specified with a ' (single quotations).
- The maximum number of title characters is 16.

Example

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

Appendix C. Macro Instruction List

C.1 Overview of the Macro Instruction List

The macro instructions used by the ASM72 are listed in alphabetical order.
The following are the notational convention rules.

1. The portion enclosed by [ ] can be omitted.
2. A white triangle △ indicates a space or tab code.
3. In the following description, a white triangle △ is inserted between the label and the macro instruction.
4. A ':' (colon) is not always required when specifying a label.
5. A space or tab code is required between a label and a macro instruction.

C.2 Macro instruction list

.CLB Clearing a Bit

Syntax

△ Label △ .CLB △ Symbol type 4

Description

- Sets the bit specified by symbol type 4 to "0".

Example

[Specifying Example]

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

[After Macro Generation]

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

.LZXY — Setting a Register

Syntax

△[Label] △ .LZXY △ Bit symbol

Description

- Sets the specified value of Z, X and Y to each register.
- If register Z does not exist, you cannot use this instruction.

Example

[Specifying Example]

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

[After Macro Generation]

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

.SEB — Setting a Bit

Syntax

△[Label:]△ .SEB △Symbol type 4

Description

- Sets the bit specified by symbol type 4 to "1".

Example

[Specifying Example]

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

[After Macro Generation]

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

Syntax

△[Label]△ .SZXYB△ Bit symbol

Description

- Tests the bit specified by the bit symbol. If it results in a "0", it skips to the next instruction.

Example

[Specifying Example]

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

[After Macro Generation]

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

Part 2

Cross-Referencer for 720 Series

CRF72 Operation Manual

Table of Contents

Chapter 1. Organization of the CRF72 Operation Manual

This CRF72 Operation Manual is organized by the following chapters.

Chapter 2. Overview
This chapter describes the basic functions and files the CRF72 generates.

Chapter 3. Operation Method
This chapter describes the command input method.

Appendix A. Error Message List
Lists the descriptions and transactions of the error messages displayed by the CRF72.

This manual is compliant with the CRF72 V.1.xx.xxC.

Chapter 2. Overview

CRF72 creates a cross-reference list of labels and symbols (hereafter, cross-reference list) for the source file. By using this list to correct a program, it becomes easy to understand the interactive relationship of the source files.

2.1 Features

The CRF72 can be used with the ASM72[1]. The CRF72 performs the following functions to effective understand the source files.

1. Displays the various types of label reference instructions in the reference line number.
2. Creates reference information of the file defined in the Source file by the pseudo instruction .INCLUDE.
3. Outputs a header with the pseudo instruction .PAGE.

2.2 File Generation

The CRF72 creates the following files.

1. Cross-reference file (hereafter, CRF file)
   - Shows a cross-reference list of labels and symbols.
   - The number of columns is fixed to 80, and the number of lines per page to 57.
   - By printing this file, it can be used for debugging or editing.
   - The file extension is .CRF.

2.3 CRF File Configuration

Figure 2.1 shows an example of a CRF file. The following information is included in a CRF file.

1. A numeric character beside the label and symbol name indicates the line number of the assembly list where the label is used.
2. A '#' (sharp) indicates that the label is defined in the line.
3. An '&' (ampersand) indicates the reference line is executed by a BM, BML and BMLA instruction.
4. Label and symbol names can be up to eight characters; excessive characters will be ignored.
5. The title indicated by the .PAGE pseudo instruction will be specified in the header of the list.
6. As CRF72 does not check the values of labels and symbols in the source program. Note that it cannot perform conditional assembly.

1. ASM72: Absolute assembler for the 720 Series

```
M50720 CROSS REFERENCE V1.00C 727TEST PROG.              P.001

ATOC     177&  396#
A_D      118   174#
COUNT     95#  154&    166&     201&     214&     379&     388&
D0_SET   250&  333#
D10      318   320#
D10_SET  322&  363#
D1_SET   254&  336#
D2       257   259#
D2_SET   261&  339#
D3       264   266#
D3_SET   268&  342#
D4       271   273#
D4_SET   275&  345#
D5       278   280#
D5_SET   282&  348#
D6       285   287#
         :
```

Figure 2.1 Example of a CRF File

Chapter 3. Operation Method

3.1 Getting Started

Starting CRF72 requires the input of the following information (input parameters).

1. Source filename (essential)
2. Command parameters

3.2 Input Parameters

3.2.1 Source Filename

1. Always input a source filename.
2. Without a file extension (.ASM), the CRF72 sets the extension to .ASM.
3. By describing the name and extension together in a filename, the CRF72 can process files other than .ASM.
4. A drive name can be also specified in the filename. Without a drive name, the CRF72 processes files in the current directory. The directory path cannot be used.
5. The source filename can include up to 16 characters.

3.2.2 Command Parameters

Command parameters indicate the .INCLUDE pseudo instruction and the drive name of the output file for the source file. Table 3.1 describes the command parameters.

Table 3.1 Command Parameter List

Command Parameter	Description
-O	Indicates the directory name for the output file. The following is an example of the syntax. Example A>CRF72 TEST -OC:¥TEMP<RET> Output the cross-reference file to C¥TEMP.

3.3 Input Method

3.3.1 Command Line Input

Starting the CRF72 is performed with the input of a command line at the DOS prompt display. Figure 3.1 shows an example of a starting command input.

```
A>CRF72 TEST<RET>
```

Figure 3.1 Example of a Command Line Input

If the CRF72 detects an error in the command line, it suppresses the process and outputs the following display; Figure 3.2.

```
C:¥>CRF72<Enter>

720 SERIES CROSS REFERENCE V.1.01.01C
COPYRIGHT(C) 1992 (1992-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

Usage: crf72 <filename> [-.] [-opath]
      -. : all messages supressed
      -o : select drive and directory for output ( use -otmp )
```

Figure 3.2 Help Display for a Command Line Error

3.4 Errors

3.4.1 Types of Errors

The following are the causes of errors while executing the CRF72.

1. DOS errors
   Relates to the DOS environment while executing the CRF72, lack of hard disc or memory capacity, etc. By referring to the Error Message List of Appendix A, process the necessary transaction with a DOS command.
2. CRF72 command line input errors
   Relates to the command line input starting the CRF72. Refer to this section and input the command line again.
3. Content of the source file in process errors
   Occurs when a specified source file does not exist.

```
C:¥>CRF72 TEST<Enter>

720 SERIES CROSS REFERENCE V.1.01.01C
COPYRIGHT(C) 1992 (1992-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

Error 2:Can't open file ( TEST.ASM )
```

Figure 3.3 Example of an Error Display

3.4.2 Returned Value to DOS

The returned value to DOS is 0 in the CRF72.

3.5 Environmental Variables

The CRF72 does not use the DOS environmental variables.

Appendix A. Error Message List

Table A.1 Error Message List

Error Number	Error Message	Troubleshooting
1	Can't make cross reference	CRF72 start-up is rejected. >> Check the memory capacity.
2	Can't open XXX.ASM	Cannot open the file. >> Make sure XXX.ASM exists and check the memory capacity.
3	Can't close XXX.ASM	Cannot close the file. >> Check the memory capacity.
4	Out of disk space	Cannot output a CRF file. >> Check the memory capacity.

Part 3

Branch-out Error Correction Utility BRANCH for 720 Series

BRANCH Operation Manual

Table of Contents

Chapter 1. Organization of the BRANCH Operation Manual

This BRANCH Operation Manual is organized by the following chapters.

Chapter 2. Overview
This chapter describes the basic functions and files the BRANCH generates.

Chapter 3. Operation Method
This chapter describes the command input method.

Appendix A. Error Message List
Lists the descriptions and transactions of the error messages displayed by the BRANCH.

This manual is compliant with the BRANCH V.1.xx.xx.

Chapter 2. Overview

The BRANCH is a software tool for correcting branch-out errors which occur at assembling. Supported MCUs are the 4500 and 720 Series of 4-bit microcomputers.

2.1 Features

The BRANCH has the following functions.

1. Automatically starts up assembler ASM45/ASM72.
2. Converts all branch instructions in a page (B) where a branch-out error occurs to branch instructions out of the page (BL).
3. Saves the assembler source file which has not been converted.

2.2 File Generation

The BRANCH generates the following files.

1. Corrected assembler source file
   This file is an assembler source file whose branch-out errors are corrected. Its file name is the same as a file name which is specified at start-up. If an extension is not specified, ".ASM" is its extension.
2. Assembler source file before correcting
   This is an assembler source file before correcting. Its extension is ".ORG". If a branch-out error does not occur, it is not generated.

Chapter 3. Operation Method

3.1 Getting Started

Starting the BRANCH requires the input of the following information (input parameters).

1. Source filename (essential)
2. Command parameters

3.2 Input Parameters

3.2.1 Source Filename

1. Specify the name of an assembler source file to be corrected.
2. You can specify only one source file.
3. Without a file extension (.ASM), the BRANCH sets the extension to .ASM.
4. By describing the name and extension together in a filename, the BRANCH can process files other than .ASM.
5. A directory path can be also specified in the filename. When only a file name is specified, the BRANCH processes files in the current directory.

3.2.2 Command Parameters

Command parameters indicate the .INCLUDE pseudo instruction and the drive name of the output file for the source file. Table 3.1 describes the command parameters.

Table 3.1 Command Parameter List

Command Parameter	Description
-.	Suppresses all the messages displayed on the screen.
-45/72	Specifies an MCU series. "-45" is for the 4500 Series; "-72" is for the 720 Series. The default setting is "-72".
-ASM	Specifies options for the ASM45/ASM72. Set options after "-ASM".

3.3 Input Method

3.3.1 Command Line Input

Starting the BRANCH is performed with the input of a command line at the DOS prompt display. An example of a starting command input is shown below.

```
A>BRANCH SAMPLE<RET>
```

If the BRANCH detects an error in the command line, it suppresses the process and outputs the following display; Figure 3.1.

```
C:¥>BRANCH<Enter>
720/4500 Series Branch Instruction Change Tool V.1.01.00
COPYRIGHT(C) 1994 (1994-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED


The file name is not specified.

Usage: branch <ASMfilename> [-.][-45/72][-ASM asmOption]

      -.     : all messages suppressed.
      -45/72 : set MCU type(4500 or 720(default)).
      -ASM   : set assembler option after -ASM.
```

Figure 3.1 Help Display for a Command Line Error

When the conversion is completed, the result of assemble and number of errors are displayed. Figure 3.2 shows an example of window display when the conversion is completed normally.

```
C:¥>BRANCH TEST<Enter>
720/4500 Series Branch Instruction Change Tool V.1.01.00
COPYRIGHT(C) 1994 (1994-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED

[1]720 SERIES ASSEMBLER V.1.20.02C
COPYRIGHT(C) 1989 (1989-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
now processing pass 1

now processing pass 2
TEST.ASM 6 ( TOTAL LINE 6 ) Error 17: Relative jump is out of range "start"

ERROR   COUNT 0001
WARNING COUNT 0000
SOURCE  LINE  0008 LINES
TOTAL   LINE  0008 LINES
COMMENT LINE  0000 LINES
OBJECT  SIZE  0002 BYTES
[2]720 SERIES ASSEMBLER V.1.20.02C
COPYRIGHT(C) 1989 (1989-2004)
RENESAS TECHNOLOGY CORPORATION
AND RENESAS SOLUTIONS CORPORATION ALL RIGHTS RESERVED
now processing pass 1

now processing pass 2

ERROR   COUNT 0000
WARNING COUNT 0000
SOURCE  LINE  0008 LINES
TOTAL   LINE  0008 LINES
COMMENT LINE  0000 LINES
OBJECT  SIZE  0003 BYTES

Conversion was completed.
```

Figure 3.2 Display When the Conversion is Completed Normally

3.4 Errors

3.4.1 Types of Errors

The following are the causes of errors while executing the BRANCH.

1. Error Concerning Command Line Input
   Relates to the command line input at BRANCH start-up. Input the command line again.
2. Error Concerning an Operation Environment
   Relates to an operating environment of the BRANCH such as shortage of hard disk or memory space.

3.4.2 Value Returned to DOS

To describe an execution file in a DOS batch file, you may want to change its process according the result of the execution. The BRANCH can return the result to DOS with a label listed below. For details on the usage of error labels, refer to a manual of DOS commercially available.

Table 3.2 List of Error Level

Error Level	Execution Result
0	Normal completion
1	Target file error
2	Operating environment error

Appendix A. Error Message List

A.1 Error Message List

If an error is detected when executing the BRANCH, an error message is displayed and the conversion is canceled. The list of error message is shown below.

Error Messages
An error does not exist.
Conversion was completed.
The instruction for conversion does not exist.
Conversion was completed, the error exists. Please correct the error of an ORG file and execute the BRANCH again.
Option specification of the BRANCH is wrong.
Option specification of an assembler is wrong.
The file name is not specified.
An assembler cannot be executed. Please check a PATH, and an environmental setup of DOS.
Option specification of an assembler is wrong.
The system error occurred in the assembler. Please execute an assembler and check a cause.
Out of heap space.
Can't open source file.
Can't open temporary file.
The source file has broken.
Can't write temporary file.
The specified editor does not execute.
CRFxx cannot be executed.
Execute of cross referencer was completed.
Can't open nul device.

ASM72 V.1.20 User's Manual

Rev.1.00
May 1, 2004
REJ10J0486-0100Z

ASM72 V.1.20
User's Manual

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

REJ10J0486-0100Z