致尊敬的顾客

关于产品目录等资料中的旧公司名称

NEC电子公司与株式会社瑞萨科技于2010年4月1日进行业务整合（合并），整合后的新公司暨“瑞萨电子公司”继承两家公司的所有业务。因此，本资料中虽还保留有旧公司名称等标识，但是并不妨碍本资料的有效性，敬请谅解。

瑞萨电子公司网址：http://www.renesas.com

2010年4月1日
瑞萨电子公司

【发行】瑞萨电子公司（http://www.renesas.com）

【业务咨询】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

M3T-SRA74 V.4.10

740族可再定位汇编程序

瑞萨单片机开发环境系统

瑞萨电子
www.renesas.com

Rev.1.00 2005.03

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts 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

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. 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 Corp. or a third party.
2. Renesas Technology Corp. assumes 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.
3. 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 Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. 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 Corp. assumes 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 Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. 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 Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. 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 Corp. or an authorized Renesas Technology Corp. 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.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. 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.
8. Please contact Renesas Technology Corp. 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 of the emulator debugger 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

注意

本文只是参考译文，前页所载英文版“Cautions”具有正式效力。

请遵循安全第一进行电路设计

1. 虽然瑞萨科技尽力提高半导体产品的质量和可靠性，但是半导体产品也可能发生故障。半导体的故障可能导致人身伤害、火灾事故以及财产损害。在电路设计时，请充分考虑安全性，采用合适的如冗余设计、利用非易燃材料以及故障或者事故防止等的安全设计方法。

关于利用本资料时的注意事项

1. 本资料是为了让用户根据用途选择合适的瑞萨科技产品的参考资料，不转让属于瑞萨科技或者第三者所有的知识产权和其它权利的许可。
2. 对于因使用本资料所记载的产品数据、图、表、程序、算法以及其它应用电路的例子而引起的损害或者对第三者的权力的侵犯，瑞萨科技不承担责任。
3. 本资料所记载的产品数据、图、表、程序、算法以及其它所有信息均为本资料发行时的信息，由于改进产品或者其它原因，本资料记载的信息可能变动，恕不另行通知。在购买本资料所记载的产品时，请预先向瑞萨科技或者经授权的瑞萨科技产品经销商确认最新信息。
   本资料所记载的信息可能存在技术不准确或者印刷错误。因这些错误而引起的损害、责任问题或者其它损失，瑞萨科技不承担责任。
   同时也请通过各种方式注意瑞萨科技公布的信息，包括瑞萨科技半导体网站。
   （http：//www.renesas.com）
4. 在使用本资料所记载部分或者全部数据、图、表、程序以及算法等信息时，在最终做出有关信息和产品是否适用的判断前，务必对作为整个系统的所有信息进行评价。由于本资料所记载的信息而引起的损害、责任问题或者其它损失，瑞萨科技不承担责任。
5. 瑞萨科技的半导体产品不是为在可能和人命相关的环境下使用的设备或者系统而设计和制造的产品。在研讨将本资料所记载的产品用于运输、交通车辆、医疗、航空宇宙用、原子能控制、海底中继器的设备或者系统等特殊用途时，请与瑞萨科技或者经授权的瑞萨产品经销商联系。
6. 未经瑞萨科技的书面许可，不得翻印或者复制全部或者部分资料的内容。
7. 如果本资料所记载的某产品或者技术内容受日本出口管理限制，必须在得到日本政府的有关部门许可后才能出口，并且不准进口到批准目的地国家以外的国家。
   禁止违反日本和（或者）目的地国家的出口管理法和法规的任何转卖、挪用或者再出口。
8. 如果需要了解本资料所记载的信息或者产品的详细，请与瑞萨科技联系。

前　言

SRA74 是 740 族专用的结构化描述可再定位的宏汇编程序。SRA74 将 740 族结构化描述语言、汇编语言或者二者混合编写的源程序生成 740 族的机器语言数据文件和调试信息文件等。本书对 SRA74 包含的以下 5 个软件的功能和操作方法进行说明：

1. 结构化描述可再定位的宏汇编程序 SRA74
2. 连接编辑程序 LINK74
3. 库生成程序 LIB74
4. 交叉参考程序 CRF74
5. 用于 M37280 的转换程序 CV74

本书的构成

本书由以下 5 部分构成。手册的各部分尽可能按相同的顺序进行说明。例如，对于环境变量，只要阅读操作说明章节的最后一节就可以了解各软件的信息。

- 第 1 部：SRA74 操作手册
  说明结构化描述可再定位的宏汇编程序 SRA74 的操作方法和源程序记述方法。

- 第 2 部：LINK74 操作手册
  说明连接程序 LINK74 的操作方法和段的功能。

- 第 3 部：LIB74 操作手册
  说明库生成程序 LIB74 的操作方法。

- 第 4 部：CRF74 操作手册
  说明交叉参照程序 CRF74 的操作方法。

- 第 5 部：CV74 操作手册
  说明用于 M37280 的转换程序 CV74 的操作方法。

- S740 族转移指令优化工具 LOOP 操作说明书
  说明 S740 族转移指令优化工具 LOOP 的操作方法。

第 1 部

用于 740 族的
结构化可再定位宏汇编程序

SRA74 操作手册

目　录

图 目 录

表 目 录

第1章　手册的构成

SRA74 操作手册由以下章节构成：

- 第 2 章 概要
  说明 SRA74 的基本功能和 SRA74 生成的文件。
- 第 3 章 源程序的记述方法
  说明 SRA74 处理的源程序的记述方法。
- 第 4 章 汇编语言
  说明 SRA74 能使用的 740 族汇编语言。
- 第 5 章 结构化描述语言
  说明 SRA74 能使用的结构化描述语言。
- 第 6 章 伪指令
  说明 SRA74 能使用的伪指令。
- 第 7 章 宏指令
  说明 SRA74 能使用的宏指令。
- 第 8 章 操作方法
  说明 SRA74 的操作方法。
- 附录 A 错误信息一览表
  对于 SRA74 显示的错误信息，用一览表表示其内容和对策。
- 附录 B 指令一览表
  表示 SRA74 能使用的 740 族汇编语言的全部指令。
- 附录 C 各寻址方式的指令一览表
  按各寻址方式表示 SRA74 能使用的 740 族汇编语言。
- 附录 D 伪指令一览
  对 SRA74 能使用的全部伪指令，表示各伪指令的内容。
- 附录 E 宏指令一览
  对 SRA74 能使用的全部宏指令，表示各宏指令的内容。
- 附录 F 结构化指令一览
  对 SRA74 能使用的结构化指令，表示各结构化指令的内容。
- 保留字一览　汇总了有关 SRA74 的保留字。

注意事项

本手册所载的程序例子中，有一部分用“\”替代表示专用页寻址方式的“¥”符号。这是由于符号根据操作系统而不同，但是代码相同都能使用。

第2章　概要

SRA74 是用于 740 族的结构化可再定位的宏汇编程序。将由汇编语言和结构化描述语言记述的源程序（以下称为源文件）转换为 LINK74[*1] 和 LIB74[*2] 能处理的可再定位文件，以下将该操作称为汇编。可再定位文件通过 LINK74 转换为机器语言数据。

2.1　功能

大规模的软件开发需要能容易进行如多个工程师间的数据交换和既存软件的再利用等程序的共享功能。SRA74 能通过以下功能高效率地进行这些操作：

1. 能通过伪指令.SECTION 给分开的区域指定任意的名（段名）。连接时，能使用在此指定的段名来指定地址。
2. 一个文件中能记述的段数没有限制，因此能对应拥有分为多个区域的 ROM 和 RAM 的用户系统。
3. 能在源程序中用伪指令.LIB 和.OBJ 指定连接对象的文件名（连接时不需要指定连接文件名）。
4. 通过用伪指令.VER 声明版本，连接时能确认可再定位文件间的版本。

其它功能有以下特点：

1. 能汇编由汇编语言和结构化描述语言混合编写的源文件（将结构化描述语言指令转换为汇编语言指令进行汇编）。
2. 能生成将结构化描述语言转换为汇编语言的文件（用户能使用该文件进行优化）。
3. 能生成保存错误内容的标记文件[*3]（能有效地进行汇编错误的修正）。
4. 由于在一个文件中能混有 RAM 区和 ROM 区，因此能用作绝对汇编程序（但是，需要连接）。
5. 能通过宏功能（指令）整理编程环境。
6. 通过指定汇编时的命令参数，能启动编辑程序和交叉参考程序。
7. 通过指定汇编时的命令参数和伪指令，能输出调试信息。

2.2　生成文件

SRA74 生成以下 5 种文件。另外，在没有指定命令参数等的情况下，生成到和源文件相同的目录。

1. 可再定位文件（以下称为 R74 文件）
   - 是保存机器语言数据及其再定位信息的文件。
   - 与命令参数的内容无关生成文件。但是，SRA74 在发生汇编错误等异常结束时不生成该文件。
   - 在指定命令参数“-O”时，输出到所指定的目录。
   - 含有用于符号调试的符号信息。但是，默认值不输出局部符号的信息。
   - 局部符号信息在指定命令参数“-S”时输出到 R74 文件。
   - 源程序行调试信息在指定命令参数“-C”或者伪指令“.FUNC”时输出到 R74 文件。

[*1] 用于 740 族的连接编辑程序。
[*2] 用于 740 族的库生成程序。
[*3] 标记的名称由来于表示错误和警告的位置的标记。

- 通过 LINK74 的处理，生成 Intel HEX 格式的机器语言文件。
- 文件属性为.R74。

2. 汇编语言文件（以下称为 I 文件）
   - 是将源文件中的宏进行展开以及将结构化指令转换成助记符的文件。
   - 在指定命令参数“-I”时生成到和 R74 文件相同的目录。但是，如果同时指定命令参数“A”就不生成该文件。
   - 在没有指定命令参数“-I”时，用于汇编操作的临时文件被生成到和 TMP 文件相同的目录。在汇编结束时自动删除。
   - 作为 SRA74 的路径 2 处理的输入文件（源文件）使用。另外，宏定义部分、宏调用部分以及结构化指令全部作为注解行输出。
   - 在源文件中，通过伪指令.INCLUDE 读取的文件也同样以带有.In（n 为 00～99）属性的文件名输出。
   - 可将 I 文件用于汇编语言级的源文件优化。
   - 文件属性为.I。

3. 打印文件（以下称为 PRN 文件）
   - 是表示处理对象的源文件及其配置地址以及生成数据的文件。
   - 在指定命令参数“-L”时生成 PRN 文件。
   - 在同时指定命令参数“-O”时，输出到所指定的目录。
   - 在同时指定命令参数“-I”时，将结构化指令转换为助记符输出到源程序行的下面。
   - 可将 PRN 文件打印输出用于调试等方面。
   - 文件属性为.PRN。

4. 标记文件（以下称为 TAG 文件）
   - 是保存汇编时发生的汇编错误信息和警告信息的文件。
   - 在指定命令参数“-E”时输出 TAG 文件。
   - 通过使用带标记转移功能的编辑程序，能提高错误修正的效率。
   - 可在通过编辑程序修正错误时参考 TAG 文件。
   - 文件属性为.TAG。

5. 临时文件（以下称为 TMP 文件）
   - 是用于汇编操作的临时文件。
   - 在设定了环境变量 TMP 的情况下，该文件被生成到由 TMP 指定的目录。
   - 该文件在汇编结束时自动删除。
   - 文件属性为.$$n（n 为 1～5）。

注意事项

由于 R74 文件是二进制格式，因此不能输出到打印机和画面。

2.2.1　I 文件的结构

在汇编如**图 2.1** 所示的源文件时生成的 I 文件的输出例子如**图 2.2** 和**图 2.3** 所示。I 文件由以下信息构成：

- 用结构化描述语言指令记述的部分被转换为汇编语言指令，转换后的汇编语言指令被表示在此结构化描述语言指令的下面。另外，结构化描述语言指令和宏指令全部作为注解行被输出。因此，I 文件能作为源文件由 SRA74 汇编。

```
        .INCLUDE    ZERO.H                      ;section z
        .PAGE
        .SECTION    AD_CONVERT
        .ORG        $E000
        .SEXT       GET_AD_DATA                 ;sub routine
        .PUB        START
        .INCLUDE    MACRO.A74
START:
        D = 0
        T = 0
        S = STACK
        RAM_CL   "WORK0, WORK1"                 ;ram clear
        [ fAD_INT_OK ] = OFF
        X = 2                                   ;A/D convert 2 times
        do
                JSR         \GET_AD_DATA
                [ WORK0 ] = [ WORK0 ] + [ AD_RESULT ]
                [ WORK1 ] = [ WORK1 ] + 0 with_c
                X = --X
        while X
        C = 0                                   ;A/D DATA average
        [ WORK1 ] = [ WORK1 ] >> 1
        [ WORK0 ] = [ WORK0 ] >>1 with_c
        [ AD_DATA ] = [ WORK0 ]
        .END
```

图 2.1　源文件的例子

```
        .INCLUDE    ZERO.I00                    ;section z
                          ↑INCLUDE指令行中的文件名被改成SRA74生成的I文件。
        .PAGE
        .SECTION    AD_CONVERT
        .ORG        $E000
        .SEXT       GET_AD_DATA                 ;sub routine
        .PUB        START
        .INCLUDE    MACRO.I01
START:
;       D = 0           ←结构化指令被输出为注解行并被转换成汇编语言。
                CLD
;       T = 0
                CLT
;       S = STACK
        LDX         #STACK
                TXS
;       RAM_CL      "WORK0, WORK1"              ;ram clear
        LDA         #0
;       .REPEATI        ram, WORK0, WORK1           ←宏指令被输出为注解行
;       STA         ram                           并被转换成汇编语言。
;       .ENDM
        STA         WORK0
        STA         WORK1
;       .ENDM
;       [ fAD_INT_OK ] = OFF
                CLB     fAD_INT_OK
;       X = 2                                   ;A/D convert 2 times
                LDX     #2
```

图 2.2　I 文件的例子（源文件前半部分）

```
;           do
            .D0:
                        JSR       ¥GET_AD_DATA
;                       [ WORK0 ] = [ WORK0 ] + [ AD_RESULT ]
                        LDA       WORK0
                        CLC
                        ADC       AD_RESULT
                        STA       WORK0
;                       [ WORK1 ] = [ WORK1 ] + 0 with_c
                        LDA       WORK1
                        ADC       #0
                        STA       WORK1
;                       X = --X
                        DEX
;           while X
                        CPX       #0
                        BNE       .D0
;           C = 0                                          ;A/D DATA average
                        CLC
;           [ WORK1 ] = [ WORK1 ] >> 1
                        LSR       WORK1
;           [ WORK0 ] = [ WORK0 ] >>1 with_c
                        ROR       WORK0
;           [ AD_DATA ] = [ WORK0 ]
                        LDA       WORK0
                        STA       AD_DATA
            .END
```

图 2.3　I 文件的例子（源文件后半部分）

2.2.2　PRN 文件的结构

PRN 文件的输出例子如图 2.4～图 2.11 所示。PRN 文件表示以下信息：

1. 表示源文件的内容和与此对应的地址、生成数据的信息（图 2.4 和图 2.5 的前半部分，图 2.7、图 2.8 以及图 2.9 的前半部分）
   - 对于参照外部标号[*4]的行，在源程序的旁边表示‘E’。
   - 对于参照公共标号[*5]的行，在源程序的旁边表示‘P’。
   - 对于参照局部标号[*6]的行，在源程序的旁边表示‘L’。
   - 对于参照符号[*7]的行，在源程序的旁边表示‘S’。
   - 对于参照位符号[*8]的行，在源程序的旁边表示‘B’。

2. 汇编结果（图 2.5 的后半部分和图 2.9 的后半部分）
   表示错误数、警告数、全部行数、注解行数、每段的存储容量。

3. 符号列表（图 2.6 的前半部分和图 2.10）
   将程序中的符号及其值分为以下 5 种进行表示。
   另外，带符号列表的打印文件能用命令参数“-LS”生成。

[*4] 指在其它文件中定义的标号。另外，将外部标号和公共标号总称为全局标号。
[*5] 指该文件中定义的并能从其它文件参照的标号。
[*6] 指该文件中定义的并只能在该文件中参照的标号。
[*7] 指用命令参数“-D”和.EQU 伪指令定义的符号。
[*8] 指用.EQU 伪指令定义的位符号。

- USED（-d OPTION）
  表示从命令行用命令参数“-D”定义的并在程序中被参照的符号。
- USED（EQUATE）
  表示用伪指令.EQU 定义的并在程序中被参照的符号。
- USED（BIT EQUATE）
  表示用伪指令.EQU 定义的并在程序中被参照的位符号。
- UNUSED（-d or EQUATE）
  表示用上述方法定义的但在程序中没被参照的符号。
- UNUSED（BIT EQUATE）
  表示用伪指令.EQU 定义的但在程序中没被参照的位符号。

4. 标号列表（图 2.6 的后半部分和图 2.11）
   将程序中的标号及其值分为以下 3 种进行表示：
   - USED（USER DEFINED）
     以段单位表示被定义的并在程序中被参照的标号。
   - USED（SYSTEM DEFINED）
     以段单位表示由 SRA74 定义的并在程序中被参照的标号（只在指定命令参数“-I”时输出）。
   - UNUSED（USER DEFINED）
     以段单位表示被定义的但在程序中没被参照的标号。

5. 如果由伪指令.COL 指定的列数为 132 个字符，就在列表的标题部分将执行汇编的时刻表示为以下形式：

```
Sat Jan 16 15:06:42 1993


1                   0           .INCLUDE            ZERO.H          ;section z
                    ↑表示INCLUDE的嵌套。
2                   1           .SECTION            Z
3                   1           .ORG                0
4 0000 (0001)       1 :WORK0:               .BLKB   1
5 0001 (0001)       1 :WORK1:               .BLKB   1
6 0002 (0001)       1 :AD_RESULT:           .BLKB   1
7 0003 (0001)       1 :AD_DATA:             .BLKB   1
          ↑表示被确保的区域容量。
8                   1           .ORG                $FE
9 00FE (0001)       1 :ICON:                .BLKB   1
                    ↓表示参照公共标号。
10 4,00FE           P 1 :fAD_INT_OK         .EQU    4, ICON
   ↑表示位序号。
11                  1           .INCLUDE            DEFINE.H
12 00BF             2 :STACK                .EQU    $BF
13 0000             2 :OFF                  .EQU    0
14 0001             2 :ON                   .EQU    1
```

图 2.4　PRN 文件的例子（源文件前半部分：无命令参数“-I”）

```
15                  0               .PAGE
16                  0               .SECTION            AD_CONVERT
17                  0               .ORG                $E000
18                  0               .SEXT               GET_AD_DATA         ;sub routine
19                  0               .PUB                START
20                  0               .INCLUDE            MACRO.A74
21                  1 RAM_CL:       .MACRO    RAMS
22                  1               LDA       #0
23                  1               .REPEATI            ram, RAMS
24                  1               STA       ram
25                  1               .ENDM
26                  1               .ENDM
27 E000             0 START:
28 E000             0               D = 0
29 E001             0               T = 0
30 E002             0               S = STACK
31 E005             0               RAM_CL    "WORK0, WORK1"            ;ram clear
32 E00B             0               [ fAD_INT_OK ] = OFF
33 E00D             0               X = 2                               ;A/D convert 2 times
34 E00F             0               do
35 E00F 2200      E 0                       JSR         ¥GET_AD_DATA
                    ↑表示参照外部标号。
36 E011             0                       [ WORK0 ] = [ WORK0 ] + [ AD_RESULT ]
37 E018             0                       [ WORK1 ] = [ WORK1 ] + 0 with_c
38 E01E             0                       X = --X
39 E01F             0               while X
40 E023             0               C = 0                               ;A/D DATA average
41 E024             0               [ WORK1 ] = [ WORK1 ] >> 1
42 E026             0               [ WORK0 ] = [ WORK0 ] >>1 with_c
43 E028             0               [ AD_DATA ] = [ WORK0 ]
44                  0               .END

ERROR          COUNT                00000
WARNING        COUNT                00000
STRUCTURED STATEMENT                00014  LINES
TOTAL          LINE ( SOURCE )      00044  LINES
TOTAL          LINE ( OBJECT )      00044  LINES
COMMENT        LINE ( SOURCE )      00000  LINES
COMMENT        LINE ( OBJECT )      00000  LINES
OBJECT         SIZE ( Z              )  00005 (0005)  BYTES
                    ↑表示段名（最多32个字符）。
OBJECT         SIZE ( AD_CONVERT )  00044 (002C) BYTES
```

图 2.5　PRN 文件的例子（源文件后半部分：无命令参数“-l”）

```
*** USED        SYMBOLS ( TYPE = -d OPTION       ) ***

*** USED        SYMBOLS ( TYPE = EQUATE          ) ***
↓符号名最多输出32个字符。
OFF                 0000p           STACK           00BFp
                      ↑表示公共标号。
*** USED        SYMBOLS ( TYPE = BIT EQUATE    ) ***

fAD_INT_OK          4,00FEp
                   ↑以0～7的值表示相应的位。
*** UNUSED      SYMBOLS ( TYPE = -d or EQUATE   ) ***

ON                  0001p

*** UNUSED      SYMBOLS ( TYPE = BIT EQUATE    ) ***


*** USED        LABELS ( TYPE = USER DEFINED   ) ***

EXTERNAL
       ↑表示外部参照。
GET_AD_DATA     0000e
                      ↑表示外部标号。
SECTION : Z
             ↑表示段名（最多122个字符）。
AD_DATA             0003p           AD_RESULT       0002p       ICON        00FEp
WORK0               0000p           WORK1           0001p

SECTION : AD_CONVERT


*** UNUSED      LABELS ( TYPE = USER DEFINED   ) ***

EXTERNAL


SECTION : Z


SECTION : AD_CONVERT

START               E000p
```

图 2.6　PRN 文件的例子（符号和标号列表：无命令参数“-l”）

```
1                   0               .INCLUDE            ZERO.H              ;section z
2                   1               .SECTION            Z
3                   1               .ORG                0
4 0000 (0001)       1 :WORK0:               .BLKB       1
5 0001 (0001)       1 :WORK1:               .BLKB       1
6 0002 (0001)       1 :AD_RESULT:           .BLKB       1
7 0003 (0001)       1 :AD_DATA:             .BLKB       1
8                   1               .ORG                $FE
9 00FE (0001)       1 :ICON:                .BLKB       1
10 4,00FE         P 1 :fAD_INT_OK           .EQU        4, ICON
11                  1               .INCLUDE            DEFINE.H
12 00BF             2 :STACK                .EQU        $BF
13 0000             2 :OFF                  .EQU        0
14 0001             2 :ON                   .EQU        1
```

图 2.7　PRN 文件的例子（源文件最前面部分：有命令参数“-l”）

```
15                  0               .PAGE
16                  0               .SECTION            AD_CONVERT
17                  0               .ORG                $E000
18                  0               .SEXT               GET_AD_DATA         ;sub routine
19                  0               .PUB                START
20                  0               .INCLUDE            MACRO.A74
21                  1 RAM_CL:       .MACRO      RAMS
22                  1               LDA         #0
23                  1               .REPEATI            ram, RAMS
24                  1               STA         ram
25                  1               .ENDM
26                  1               .ENDM
27 E000             0 START:
28                  0 ;             D = 0               ←结构化指令变为注解行。
29 E000 D8          0                           CLD     ←结构化指令被转换成汇编语言。
30                  0 ;             T = 0
31 E001 12          0                           CLT
32                  0 ;             S = STACK
33 E002 A2BF      S 0                           LDX     #STACK
                    ↑表示参照符号。
34 E004 9A          0                           TXS
35 E005             0               RAM_CL      "WORK0, WORK1"          ;ram clear
36 E005 A900        0+              LDA         #0
                      ↑表示宏展开。
37                  0+              .REPEATI            ram, WORK0, WORK1
38                  0+              STA         ram
39                  0+              .ENDM
40 E007 8500      P 0+              STA         WORK0
41 E009 8501      P 0+              STA         WORK1
42                  0+              .ENDM
43                  0 ;             [ fAD_INT_OK ] = OFF
44 E00B 9FFE      B 0                           CLB     fAD_INT_OK
                    ↑表示参照位符号。
45                  0 ;             X = 2                               ;A/D convert 2 times
46 E00D A202        0                           LDX     #2
```

图 2.8　PRN 文件的例子（源文件中间部分：有命令参数“-l”）

```
47                  0 ;             do
48 E00F             0               .D0:
49 E00F 2200     E 0                        JSR     \GET_AD_DATA
50                  0 ;                     [ WORK0 ] = [ WORK0 ] + [ AD_RESULT ]
51 E011 A500     P 0                        LDA     WORK0
52 E013 18          0                      CLC
53 E014 6502     P 0                        ADC     AD_RESULT
54 E016 8500     P 0                        STA     WORK0
55                  0 ;                     [ WORK1 ] = [ WORK1 ] + 0 with_c
56 E018 A501     P 0                        LDA     WORK1
57 E01A 6900        0                       ADC     #0
58 E01C 8501     P 0                        STA     WORK1
59                  0 ;                     X = --X
60 E01E CA          0                      DEX
61                  0 ;             while X
62 E01F E000        0                       CPX     #0
63 E021 D0EC     L 0                        BNE     .D0
64                  0 ;             C = 0                                   ;A/D DATA average
65 E023 18          0                      CLC
66                  0 ;             [ WORK1 ] = [ WORK1 ] >> 1
67 E024 4601     P 0                        LSR     WORK1
68                  0 ;             [ WORK0 ] = [ WORK0 ] >>1 with_c
69 E026 6600     P 0                        ROR     WORK0
70                  0 ;             [ AD_DATA ] = [ WORK0 ]
71 E028 A500     P 0                        LDA     WORK0
72 E02A 8503     P 0                        STA     AD_DATA
73                  0               .END

ERROR        COUNT                  00000
WARNING      COUNT                  00000
STRUCTURED STATEMENT                00014   LINES
TOTAL        LINE ( SOURCE )        00043   LINES
TOTAL        LINE ( OBJECT )        00073   LINES
COMMENT      LINE ( SOURCE )        00000   LINES
COMMENT      LINE ( OBJECT )        00014   LINES
OBJECT       SIZE ( Z                    )  00005 (0005)  BYTES
OBJECT       SIZE ( AD_CONVERT      )  00044 (002C)  BYTES
```

图 2.9　PRN 文件的例子（源文件最后部分：有命令参数“-l”）

```
*** USED     SYMBOLS ( TYPE = -d OPTION         ) ***

*** USED     SYMBOLS ( TYPE = EQUATE            ) ***

OFF                 0000p           STACK       00BFp

*** USED     SYMBOLS ( TYPE = BIT EQUATE      ) ***

fAD_INT_OK      4,00FEp

*** UNUSED SYMBOLS ( TYPE = -d or EQUATE     ) ***

ON                  0001p

*** UNUSED SYMBOLS ( TYPE = BIT EQUATE      ) ***
```

图 2.10　PRN 文件的例子（符号列表：有命令参数“-l”）

```
*** USED       LABELS ( TYPE = USER      DEFINED ) ***

EXTERNAL

GET_AD_DATA    0000e

SECTION : Z

AD_DATA        0003p      AD_RESULT      0002p      ICON        00FEp
WORK0          0000p      WORK1          0001p

SECTION : AD_CONVERT

*** USED       LABELS ( TYPE = SYSTEM  DEFINED ) ***

SECTION : Z


SECTION : AD_CONVERT

.D0                E00F'

*** UNUSED   LABELS ( TYPE = USER      DEFINED ) ***

EXTERNAL


SECTION : Z


SECTION : AD_CONVERT

START              E000p
```

图 2.11　PRN 文件的例子（标号列表：有命令参数“-l”）

2.2.3　TAG 文件的结构

TAG 文件的输出例子如图 2.12 所示。TAG 文件表示以下信息：

- 表示错误或者警告发生位置的文件名及其文件中的行号、连续行号、错误序号和错误信息。

可将 TAG 文件打印输出，在用编辑程序修正错误时进行参考使用。

```
TEST.A74 115 ( TOTAL LINE 115 ) Error 18: Relative jump is out of range
TEST.A74 127 ( TOTAL LINE 127 ) Error 22: Value is out of range "data"
TEST.A74 593 ( TOTAL LINE 593 ) Error 23: "()" format error ";"
```

图 2.12　TAG 文件的例子

第3章　源程序的记述方法

3.1　源程序的结构

源程序的构成以行为单位，行的记述规则如下所示：

1. 各行必须按行结束。因此，一条指令不能跨行记述。
2. 1 行的字符数包括换行码最多为 256 个字符。SRA74 忽视超过 256 个字符的部分。
3. 能按内容分为以下 5 种行：
   - 汇编语言指令行
     是记述 740 族汇编语言指令的行。
     该行生成对应的机器语言数据。
   - 结构化描述语言指令行
     是记述 740 族结构化描述语言指令的行。
     该行生成对应的机器语言数据。
   - 伪指令行
     是记述 740 族伪指令的行。
   - 宏指令行
     是记述 740 族宏指令的行。
   - 注解行
     由于 SRA74 不处理该行，因此用户能自由使用。

3.2　行的结构

本节分别说明各行的结构。另外，书写时所用记号的说明及其规则表示如下：

1. △、▲表示空格或者制表符码。
   △的部分为必需，▲的部分为可省略。
2. 记述标号时不必使用‘:’（冒号），但是，省略‘:’时在各指令间需要空格或者制表符码，汇编时必须指定命令参数“-U”。因此，通常建议用‘:’记述。

3.2.1　汇编语言指令行

汇编语言指令行的结构如下所示。有关该行的详细内容请参照第 4 章、附录 B 以及附录 C。

▲ [标号] :▲ [操作码] △ [操作数] ▲; [注解] <RET>

1. 标号栏
   记述为了能从其它位置参照该行的标号。
2. 操作码栏
   记述 740 族的汇编语言助记符（以下称为操作码）。操作码不区分大小写英文字母。因此，NOP、nop 以及 Nop 中的任何一个都有效。

由于 SRA74 将操作码作为保留字识别，因此在没记述标号时能从行头开始记述。

3. 操作数栏
   记述操作码的处理对象。
   • 操作数有 2 个以上的数据时必须在数据间用‘,’（逗号）隔开。
   • 在逗号的两侧能记述空格或者制表符码。
4. 注解栏
   SRA74 不处理该栏，因此用户能自由使用。

3.2.2　结构化描述语言指令行

结构化描述语言指令（以下称为结构化指令）行的结构如下所示。有关该行的详细内容请参照第 5 章和附录 F。

▲ | 标号 | :▲ | 结构化指令 | △ | 条件表达式 | ▲; | 注解 | <RET>

1. 标号栏
   记述为了能从其它位置参照该行的标号。
2. 结构化指令栏
   记述 740 族的结构化指令。结构化指令不区分大小写英文字母。因此，IF、if 和 If 中的任何一个都有效。
   由于 SRA74 将结构化指令作为保留字识别，因此在没记述标号时能从行头开始记述。
3. 条件表达式栏
   记述 740 族结构化指令的处理对象。
4. 注解栏
   SRA74 不处理该栏，因此用户能自由使用。

3.2.3　伪指令行

伪指令行的结构如下所示。有关该行的详细内容请参照第 6 章和附录 D。

▲ | 符号/位符号 | △ | .EQU | △ | 操作数 | ▲; | 注解 | <RET>

▲ | 标号 | :▲ | 伪指令 | △ | 操作数 | ▲; | 注解 | <RET>

1. 符号/位符号栏
   记述由伪指令.EQU 赋值的符号或者位符号。
2. 标号栏
   记述为了能从其它位置参照该行的标号。
3. 伪指令栏
   记述 740 族的伪指令。伪指令不区分大小写英文字母。因此，.END、.end 和.End 中的任何一个都有效。
   由于 SRA74 将伪指令作为保留字识别，因此没有记述标号时能从行头开始记述。
4. 操作数栏
   记述伪指令的处理对象。
   • 操作数有 2 个以上的数据时必须在数据间用‘,’（逗号）隔开。
   • 在逗号的两侧能记述空格或者制表符码。
5. 注解栏
   SRA74 不处理该栏，因此用户能自由使用。

3.2.4　宏指令行

宏指令行的结构如下所示。有关该行的详细内容请参照第 7 章和附录 E。

▲ | 宏名称 | :▲ | MACRO | △ | 操作数 | ▲; | 注解 | <RET>

▲ | 标号 | :▲ | 宏指令 | △ | 操作数 | ▲; | 注解 | <RET>

1. 宏名称栏
   是用于调用宏定义的名称。
2. 标号栏
   记述为了能从其它位置参照该宏指令行的标号（名称）。
3. 宏指令栏
   记述 740 族的宏指令。宏指令不区分大小写英文字母。因此，.REPEATI、.repeati 和.RepeatI 中的任何一个都有效。
   由于 SRA74 将宏指令作为保留字识别，因此在没记述标号时能从行头开始记述。
4. 操作数栏
   记述宏指令的处理对象。
   • 操作数有 2 个以上的数据时必须在数据间用‘,’（逗号）隔开。
   • 在逗号两侧能记述空格或者制表符码。
5. 注解栏
   SRA74 不处理该栏，因此用户能自由使用。

3.2.5　注解行

注解行必须用‘;’（分号）开始行的第一个字符（除了▲）。注解行的结构如下所示：

▲; | 注解 | <RET>

注意事项

如果注解行的第一个字符不为‘;’，SRA74 就不将该行识别为注解行而进行汇编处理。其结果可能会发生汇编错误或者生成预想不到的代码，因此必须注意。

3.3　栏的记述方法

本节仅对各指令行中记述格式相同的栏进行说明。有关记述格式不同的栏请分别参照第 4 章、第 5 章、第 6 章和第 7 章。

3.3.1　符号/位符号/标号栏

在各指令行中，该栏的记述格式都相同。但是，SRA74 区分符号、位符号和标号[*1]进行管理。记述规则如下所示：

1. 名称中能使用英数字、‘_’（下划线）、‘.’（句号）以及‘?’（问号）。但是，能用于第一个字符的字符只限于英文字母和‘_’（下划线）。另外，字符数包括‘:’最多能使用 255 个字符。

[*1] 将由伪指令.EQU 或者命令参数“-D”定义的内容作为符号或者位符号使用，除此之外定义的内容作为标号使用。

2. 区分大小写字母。因此，BIG 和 Big 作为不同的名称来判断。
3. 保留字不能用于名称。SRA74 将寄存器名、标志名、操作码、结构化指令、伪指令以及宏指令作为保留字处理。
4. 用特殊字符‘..’（2 个句号）开头的标号为 SRA74 的保留字，因此不能使用。另外，用 1 个句号或者 3 个以上的句号开头的名称也不能使用。
5. 用“??”（2 个问号）开头的标号作为只在该段内有效的段内局部标号使用。因此，能在不同的段中使用同一名称的标号。参照时，总是只限于参照该段内的标号。

```
例）            .SECTION   PROG1
      ??MAIN:   NOP
                  :
                BRA        ??MAIN      转移到段名为“PROG1”的“??MAIN”。
                .SECTION   PROG2
      ??MAIN:   NOP
                  :
                BRA        ??MAIN      转移到段名为“PROG2”的“??MAIN”。
```

6. 通过在符号、位符号以及标号的前面插入‘:’，能省略.PUB 声明。

```
例）:SYMBOL   .EQU        10
                :
              .SECTION   PROG2
      :LABEL: NOP
                :
```

记述标号时能在名称的后面插入‘:’（冒号）。为了容易和符号进行区分，或者为了有效地进行在编辑程序中的标号检索，建议使用‘:’进行记述。但是，在记述符号和位符号时，如果插入‘:’就会出错，因此必须注意。

3.3.2　注解栏

能记述用户的任意信息。记述格式表示如下：

1. 注解的起始部分必须使用‘;’（分号）记述。
2. 注解栏中能记述任何字符。

第4章　汇编语言

4.1　寻址方式

寻址方式是指令指定处理对象数据的基本方式（模式）。740 族有 19 种寻址方式，各种寻址方式有各自的操作数格式。有关操作数记述的寻址方式如下所示：

1. 隐含
   是只有操作码的指令。对操作数不作任何指定。
   例）　BRK
2. 累加器
   是将处理对象数据作为累加器内容的方法。
   例）　ASL　A
3. 立即
   是用操作数直接指定处理对象数据的方法。操作数所记述的值必须以‘#’开始。
   例）　ADC　#IMMDATA
4. 零页
   将零页区域（00_{16}～FF_{16}）指定为处理对象数据的方法。
   例）　ADC　ZWORK
5. 零页 X
   是用寄存器 X 将零页地址修饰为处理对象数据的方法。必须在‘,’（逗号）后记述寄存器名‘X’。
   例）　ADC　ZWORK, X
6. 零页 Y
   是用寄存器 Y 将零页地址修饰为处理对象数据的方法。必须在‘,’（逗号）后记述寄存器名‘Y’。
   例）　LDX　ZWORK, Y
7. 零页间接
   是用存储器间接表示处理对象数据的方法。给操作数记述存储对象地址的零页地址，在存储器中存储 2 个字节的对象地址。给操作数记述的值必须用“( )”表示。
   例）　JMP　(ZWORK)
8. 零页间接 X
   是表示用存储器间接和寄存器 X 修饰处理对象数据的方法。给操作数记述存储对象地址的零页地址，在存储器中存储 2 个字节的对象地址。给操作数记述的值必须用“( )”表示，并且在‘,’（逗号）的后面记述寄存器名‘X’。
   例）　ADC　(ZWORK, X)
9. 零页间接 Y
   是表示用存储器间接和寄存器 Y 修饰处理对象数据的方法。给操作数记述存储对象地址的零页地址，在存储器中存储 2 个字节的对象地址。必须在‘,’（逗号）的后面记述寄存器名‘Y’，并且给操作数记述的值用“( )”表示。
   例）　ADC　(ZWORK), Y
10. 绝对
    是将一般页区域（0100_{16}～$FFFF_{16}$）指定为处理对象数据的方法。
    例）　ADC　WORK

11. 绝对 X
    是用寄存器 X 将一般页地址修饰为处理对象数据的方法。必须在‘,’（逗号）后记述寄存器名‘X’。
    例）　ADC　WORK, X
12. 绝对 Y
    是用寄存器 Y 将一般页地址修饰为处理对象数据的方法。必须在‘,’（逗号）后记述寄存器名‘Y’。
    例）　ADC　WORK, Y
13. 绝对间接
    是用存储器间接表示处理对象数据的方法。给操作数记述存储对象地址的一般页地址，在存储器中存储 2 字节的对象地址。给操作数记述的值必须用()表示。
    例）　JMP　(WORK)
14. 专用页
    是将专用页区域（$FF00_{16}$～$FFFF_{16}$）指定为处理对象数据的方法。操作数记述的值必须用‘¥’或者‘\’表示。
    例）　JSR　¥WORK
15. 零页位
    是将零页区域（00_{16}～FF_{16}）的特定位指定为处理对象数据的方法。另外，如果给操作数记述位符号，就参照该位符号的位值和地址。
    例）　CLB　0, ZWORK　→　指定‘ZWORK’的位 0。
    例）　CLB　BITSYMBOL
16. 累加器位
    是将累加器的特定位指定为处理对象数据的方法。另外，如果给第一操作数记述位符号，就只参照该位符号的位值。
    例）　CLB　1, A　→　指定累加器的位 1。
    例）　CLB　BITSYMBOL, A
17. 相对
    转移到本指令的起始地址加操作数内容的地址。但是，不能给操作数记述相对值。如果给操作数记述标号或者对象地址，SRA74 就计算相对值。
    例）　BRA　*-12
    例）　BRA　NEXT
18. 零页位相对
    将零页区域（00_{16}～FF_{16}）的特定位指定为处理对象数据。另外，如果给第一操作数记述位符号，就参照该位符号的位值和地址，根据该位的状态转移到本指令的起始地址加最终操作数内容的地址。但是，不能给最终操作数记述相对值。如果记述标号或者对象地址，SRA74 就计算相对值。
    例）　BBC　2, ZWORK, NEXT　→　指定‘ZWORK’的位 2。
    例）　BBC　BITSYMBOL, NEXT
19. 累加器位相对
    将累加器的特定位指定为处理对象数据。另外，如果给第一操作数记述位符号，就只参照该位符号的位值，根据该位的状态转移到本指令的起始地址加最终操作数内容的地址。但是，不能给最终操作数记述相对值。如果记述标号或者对象地址，SRA74 就计算相对值。
    例）　BBC　3, A, NEXT　→　指定累加器的位 3。
    例）　BBC　BITSYMBOL, A, NEXT

各指令的各种寻址方式的记述格式请参照附录 C。

4.2　操作数的数据格式

操作数能记述以下 4 种数据形式：

1. 数值常数
   - 能通过在数值常数前加上‘+’或者‘-’的运算符来表示值的正负。在没有‘+’或者‘-’的情况下，作为正值处理。
   - 不能在表示数值种类的符号和数值间插入空格或者制表符。
     例）　.BYTE　$ 64　→　错误。
   - 数值常数能使用 2 进制数、8 进制数、10 进制数或者 16 进制数。
   - 2 进制数 → 由 2 进制数构成，必须在开头加上‘%’或者在最后加上‘B’、‘b’记述。
     例）　.BYTE　%100110
     例）　.BYTE　100110B
   - 8 进制数 → 由 8 进制数构成，必须在开头加上‘@’或者在最后加上‘O’、‘o’或‘Q’、‘q’记述。
     例）　.BYTE　@70
     例）　.BYTE　70o
     例）　.BYTE　70Q
   - 10 进制数 → 由 10 进制数构成，不用特别指定。只需记述如 23、256 等的整数。
     例）　.BYTE　100
   - 16 进制数→ 由 16 进制数构成，必须在开头加上‘$’或者在最后加上‘H’、‘h’记述。在用英文字符（A～F）开头的情况下，必须在开头加上 0。
     例）　.BYTE　$64
     例）　.BYTE　64H
     例）　.BYTE　0ABH
2. 字符常数
   - 在字符中能记述用 ASCII 码定义的字符。
   - 字符常数必须用‘’’（单引号）或者‘"’（双引号）括起来记述。各字符对应 7 位的 ASCII 码（最高位为 0）。
     例）　.BYTE　'A'　→　设定 41_{16}。
3. 符号常数
   - 符号有符号、位符号、标号以及表示当前语句开始的‘*’4 种。位符号有被分配的位值及其位所属的地址，符号有绝对值，标号和‘*’有相对值或者绝对值。
     例）　.WORD　SUB　→　设定标号 SUB 的地址。
     例）　BRA　*+2　→　转移到当前地址加 2 的地址。
4. 表达式
   - 表达式由数值常数、字符常数、符号常数和运算符组合构成。在运算符和各项之间根据需要能插入空格或者制表符。
     例）　TBL + 1
   - 表达式从左到右计算（运算符没有优先顺序）　。
     例）　2*3　→　结果为 6。
     例）　2+6/2　→　结果为 4。

4.3　运算符

能用于操作数数据记述的运算符一览表如**表 4.1** 所示。

表 4.1　运算符一览表

分类	运算符[*1]	内　容
单项运算符	+	表示正数
	-	表示负数
	!	取 1 的补码
	<	取标号或者符号的高 8 位
	>	取标号或者符号的低 8 位
	SIZEOF[*2]	计算段的大小
	BK[*3]	取标号的存储体值
	BL[*3]	取标号的附加区域值[*4]
双项运算符	+	加法
	-	减法
	*	乘法
	/	除法
	&	各位的与
	|	各位的或

注意事项

*1. V.4.00.00 以上版本的 SRA74 以 32 位带符号整数值处理表达式。但是，也以 32 位带符号整数值处理的表达式结果只限于以下 2 个值：

・.ORG 指示指令的操作数

・指定-BANK 选项时的标号值

在记述如下例表达式的情况下，可能与 V.2.00.10 以前版本的 SRA74 的运算结果不同。

例）　2223h + 0FFFFh / 2

・V.2.00.10 以前版本的运算结果为 1111H

・V.4.00.00 以后版本的运算结果为 9111H

*2. 与“参照段是可再定位还是绝对”无关，在连接时决定 SIZEOF 的值。SIZEOF 运算符只能用于段名。另外，在使用 SIZEOF 运算符时，必须在与段名之间插入空格。

例）　.WORD　SIZEOF　DATA

*3. 必须给运算符 BK 和 BL 的操作数记述在汇编执行时确定值的标号。

*4. 附加区域值为操作数（标号）的低 12 位加上 1000H 后的值。

*5. 从左向右运算（运算符没有优先顺序）。

例）　2+6/2　→　结果为 4。

*6. 不能进行对位符号的运算以及用“.ZEXT”和“.SEXT”参照的标号间的运算。

第5章　结构化描述语言

5.1　结构化指令的功能

在程序的记述中过多使用转移指令和标号会降低程序的可读性。在结构化描述语言中，由于能不使用标号和转移指令来记述条件转移结构和重复结构等，因此能提高程序的可读性。另外，通过直接操作寄存器和标志的指令，能提高程序的效率。

5.2　语句的种类

结构化描述语言有以下 7 种语句。如果附加 l（L），就对各转移指令生成“JMP”的长字转移。记述例子和进行相同处理的汇编程序如下所示。各语句的详细内容请参照附录 F。

1. 赋值语句

   将右边赋值到左边。

```
 [ MEM ] = 10              ; LDM    #10, MEM
 [ MEM ] = [ MEM1 ]        ; LDA    MEM1
                           ; STA    MEM
```

2. (l)if ～ (l)else ～ endif 语句

   if 语句是将控制流程变成 2 个方向的指令，转移的方向由条件表达式决定。

```
if [ MEM ]                 ;        LDA    MEM
        [ WORK ] = 1       ;        BEQ    I0
 else                      ;        LDM    #1,WORK
        [ WORK ] = 2       ;        BRA    I1
 endif                     ; I0:
                           ;        LDM    #2,WORK
                           ; I1:
```

3. (l)for ～ next 语句

   for 语句是控制重复的指令，在指定的条件表达式为真期间重复执行语句。

```
 for [ MEM ]               ; F0:
        jsr output         ;        LDA    MEM
 next                      ;        BEQ    F1
                           ;        JSR    OUTPUT
                           ;        BRA    F0
                           ; F1:
```

4. (l)do ～ while 语句

   do 语句在满足条件表达式（为真）期间重复执行语句。

```
 do                        ; D0:
        jsr output         ;        JSR    OUTPUT
 while [ MEM ]             ;        LDA    MEM
                           ;        BNE    D0
```

5. (l)switch ～ case ～ ends 语句

switch 语句根据条件表达式的值将控制移交给某个语句。

```
switch [ MEM ]          ;       LDA     MEM
        case 1          ;       CMP     #1
                jsr output1     ;       BNE     S0
                break   ;       JSR     OUTPUT1
        case 2          ;       BRA     S3
                jsr output2     ; S0:
                break   ;       CMP     #2
        case 3          ;       BNE     S1
                jsr output3     ;       JSR     OUTPUT2
                break   ;       BRA     S3
        default         ; S1:
                jsr output4     ;       CMP     #3
ends                    ;       BNE     S2
                        ;       JSR     OUTPUT3
                        ;       BRA     S3
                        ; S2:
                        ;       JSR     OUTPUT4
                        ; S3:
```

6. (l)break 语句

break 语句停止执行对应的 for 语句、do 语句或者 switch 语句，将控制移交给下一个执行语句。

```
for [ MEM1 ]            ; F0:
        if [ MEM2 ]     ;       LDA     MEM1
                break   ;       BEQ     F1
        endif           ;       LDA     MEM2
        jsr output      ;       BEQ     I0
next                    ;       BRA     F1
                        ; I0:
                        ;       JSR     OUTPUT
                        ;       BRA     F0
                        ; F1:
```

7. (l)continue 语句

continue 语句假想地在包括自身的最小重复 for 语句或者 do 语句中的最后语句的后面插入空语句，并将控制移交给该空语句。

```
for [ MEM1 ]            ; F0:
        if [ MEM2 ]     ;       LDA     MEM1
           continue     ;       BEQ     F1
        endif           ;       LDA     MEM2
        jsr output      ;       BEQ     I0
next                    ;       BRA     F0
                        ; I0:
                        ;       JSR     OUTPUT
                        ;       BRA     F0
                        ; F1:
```

5.3　记述时的注意事项

说明使用在结构化描述语言编程时的注意事项。

1. 在赋值语句或者各控制语句（if 语句等）的条件表达式记述由 740 族的各寻址方式参照的存储器时，需用“[ ]”或者“{ }”括起来记述。详细内容请参照 F.3.1 的赋值语句的展开例。

2. 在赋值语句或者各控制语句（if 语句等）的条件表达式记述由位符号参照的任意位时，需用“[ ]”或者“{ }”括起来记述。但是，为了参照累加器的任意位，准备了以下的保留字。此时，不需要用“[ ]”或者“{ }”。另外，由于不区分大小写字母，因此 BIT_A0 和 bit_a0 中的任何一个都有效。

BIT_A0	累加器的位 0	BIT_A1	累加器的位 1
BIT_A2	累加器的位 2	BIT_A3	累加器的位 3
BIT_A4	累加器的位 4	BIT_A5	累加器的位 5
BIT_A6	累加器的位 6	BIT_A7	累加器的位 7

详细内容请参照 F.3.1 的 A）寄存器、标志赋值语句的展开例。

3. 在赋值语句或者各控制语句（if 语句等）的条件表达式记述 740 族的各种寄存器时，必须用各寄存器名记述。为此，SRA74 准备了以下名称的保留字。另外，由于不区分大小写字母，因此 A 和 a 中的任何一个都有效。

A	累加器	X	变址寄存器 X
Y	变址寄存器 Y	S	堆栈指针
P	处理器状态寄存器		

详细内容请参照 F.3.1 的 A）寄存器、标志赋值语句的展开例。

4. 在赋值语句或者各控制语句（if 语句等）的条件表达式记述 740 族的状态寄存器内的标志时，必须用各标志名记述。为此，SRA74 准备了以下名称的保留字。另外，由于不区分大小写字母，因此 C 和 c 中的任何一个都有效。

C	进位标志	Z	零标志
I	中断禁止标志	D	10 进制模式标志
T	X 变址运算模式标志	V	溢出标志
N	负标志		

详细内容请参照 F.3.1 的 A）寄存器、标志赋值语句的展开例和 F.3.2 的标志条件表达式的展开例。

5. SRA74 在进行以下记述（先参照）时作为标号处理（通过 LDA 指令等进行代码展开）。因此，如果在后面作为位符号定义“BITSYM”，展开的代码就不符（此时，SRA74 显示错误）。在记述位符号（BITSYM）时，必须按先定义后参照的原则来改写程序。

```
例）            ;if [ BITSYM ]
                LDA     BITSYM      展开代码
                BEQ     .I0         展开代码
                :
                ;endif
                .I0:
                    :
      BITSYM    .equ    1, 80h      位符号的定义
```

详细内容请参照附录 F 的句法图。

6. 生成小代码量的记述方法
   - 在 IF 语句和 FOR 语句的情况下，＞＝比＞效率高，＜比＜＝效率高。在 DO 语句的情况下，＞比＞＝效率高，＜＝比＜效率高。
     详细内容请参照 F.3.2 的存储器条件表达式的展开例。
   - 在进行至少 1 次以上的处理时，DO 语句比 FOR 语句效率高。
   - 在 SWITCH 语句的情况下，必要的地方使用 lbreak 比使用 lswitch 效率高。

LSWITCH语句	汇编语言	LBREAK语句	汇编语言
lswitch　[MEM] 　case　1 　　NOP 　　break 　case　2 　　: 　case　16 　　NOP 　　break 　default 　　NOP ends	CMP　# 1 　　BEQ　.Z0 　　JMP　.S2 .Z0: 　　NOP 　　JMP　.S0 .S2: 　　: .S16: 　　CMP　# 2 　　BEQ　.Z15 　　JMP　.S17 .Z15: 　　NOP 　　JMP　.S0 .S17: 　　NOP .S0:	switch　[MEM] 　case　1 　　NOP 　　lbreak 　Case　2 　　: 　Case　16 　　NOP 　　break 　default 　　NOP ends	CMP　# 1 　　BNE　.S2 　　NOP 　　JMP　.S0 .S2: 　　: .S16: 　　CMP　# 2 　　BNE　.S17 　　NOP 　　BRA　.S0 .S17: 　　NOP .S0:

7. 关于累加器的值
   在结构化描述语言的运算中，大多数情况都借助于累加器。因此，必须注意在执行如下所示的结构化指令后累加器的值发生变化。

```
;[ WORK ] = [ DATA1 ] + [ DATA2 ]
        LDA     DATA1
        CLC
        ADC     DATA2
        STA     WORK
;IF [ WORK ] & 0FH > 4
        LDA     WORK
        AND     #0FH
        CMP     #4
        BEQ     .I0
        BCC     .I0
;   X = ++X
        INX
;ENDIF
.I0:
```

但是，寄存器的传送和比较不使用累加器而被高效率地展开成汇编语言。

```
;X = [ WORK ]
            LDX     WORK
;IF X > [ WORK ]
            CPX     WORK
            BEQ     .I0
            BCC     .I0
;    Y = ++Y
            INY
;ENDIF
.I0:
```

详细内容请参照 F.3.1 的 D）2 项运算赋值语句的展开例。

5.4　结构化运算符

能用于结构化描述语言的运算符一览表如**表 5.1** 所示。

表 5.1　结构化描述语言的运算符一览表

分类	运算符*1	内 容	分类	运算符*1	内 容
单项运算符	+	表示正数	双项运算符	+*1	加法
	-	表示负数		-*1	减法
	~	取 1 的补码		*2	乘法
	++	增量		/ *2	除法
	--	减量		%*2	除法的余数
比较运算符	<	小于		&	各位的与
	>	大于		|	各位的或
	==	等于		^	各位的异或
	!=	不等于		&&*3	逻辑与
	<=	小于等于		||*3	逻辑或
	>=	大于等于		<<	左移
				>>	右移

注意事项

*1. SRA74 的结构化描述运算以 8 位无符号数值处理。运算（只限于‘+’和‘-’）结果不考虑溢出。因此，必须注意在进行如下的大小比较时不能发生溢出（在下面的例子中，如果 work 的内容为 FF_{16}，运算结果就为 00_{16}，条件不成立）。

例）
```
if [ work ] + 2 > 10
            :
else
            :
endif
```

2. 在使用双项运算符‘’、‘/’和‘%’时，必须在汇编后连接附属的子程序库（SRA74.A74）。步骤如下：

(a) 汇编 SRA74.A74，生成可再定位的文件 SRA74.R74。

(b) 将对应运算符的标号指定为外部参照。

由于 SRA74 从乘除法的双项运算符‘*’、‘/’以及‘%’生成调用子程序（包括在 SRA74.R74 中）的机器语言，因此，必须将对应下表所示的运算符的标号指定为外部参照。

运算符	标号	
*	.mult_8	乘法程序
/	.div_8	除法程序
%	.mod_8	取余程序

（指定例）

```
.ext        .mult_8,    .div_8
```

(c) 连接用户程序和 SRA74.R74。

另外，即使在将堆栈区设置为 1 页（0100_{16} 地址以后）的系统中也能使用，但是此时必需进行以下的定义以及确保工作区。

- 设定堆栈页（.SPPAGE 标号的定义）。设定的值=0 时为 0 页，设定的值=1 时为 1 页（但是，必须注意这只是对汇编进行指示而不是设定堆栈）。
  例）　.SPPAGE　　.EQU　　1
  SRA74.A74 根据该符号进行条件汇编。
- 在 0 页内的 RAM 中确保 3 字节的运算工作区（.syswk）。运算结果为乘法时，对.syswk 和.syswk+1 按低位、高位的顺序进行设定（要参照运算结果的高位时，必须由用户自己参照）。
  例）　.syswk:　　.blkb　　3
- 在通过中断处理程序使用运算工作区（.syswk）时必须注意。此时，需要用户自己变更附属的库程序的源程序（SRA74.A74），或者将运算工作区保存到堆栈。

*3. 通过使用双项逻辑运算符‘&&’和‘||’，能最多记述 6 个结构化指令（if、for、while）的条件表达式。
但是，逻辑运算符没有优先顺序，从头开始按序展开。即如下展开：

```
;IF BIT_A1 == 1 || BIT_A2 == 1 && BIT_A3 == 1 || BIT_A4 == 1
        BBS     BIT_A1,A,.I0
        BBC     BIT_A2,A,.I1
        BBS     BIT_A3,A,.I0
        BBC     BIT_A4,A,.I1
.I0:
;       X = ++X
        INX
;ENDIF
.I1:
```

第6章　伪指令

6.1　伪指令的功能

伪指令为了生成机器语言数据而对 SRA74 进行指示[*1]。SRA74 具有 48 条伪指令，能按功能分为以下 6 组：

1. 汇编控制
   - 伪指令本身不生成数据，只是控制汇编处理的流程。
   - 不影响地址的更新。
   - 此组包括以下 7 条伪指令：

.ASSERT	汇编断言声明
.END	程序结束声明
.ERROR	汇编错误声明
.IF (.ELSE) .ENDIF	条件汇编
.INCLUDE	文件读取

2. 地址控制
   - 数据设定伪指令生成常数数据。
   - 更新地址。
   - 此组包括以下 7 条伪指令：

.EQU (=)	同义的定义
.ORG (*=)	地址指定
.BLKB	确保 RAM 区
.BYTE .WORD	数据设定

3. 连接控制
   - 进行与连接处理相关的控制。
   - 此群包括以下 18 条伪指令：

.BEXT	.ZBEXT		外部参照的指定（位符号）
.EXT	.SEXT	.ZEXT	外部参照的指定（符号和标号）
.PUB			公共的指定（位、符号和标号）
.SECTION			ROM和RAM区的指定
.SECTION	P(.PMOD)		ROM区的指定（一般页）
.SECTION	R(.RMOD)		RAM区的指定（一般页）
.SECTION	S(.SMOD)		ROM区的指定（专用页）
.SECTION	Z(.ZMOD)		RAM区的指定（零页）
.OBJ	.LIB		连接文件名的指定
.VER			版本的指定

4. 列表控制
   - 进行与 PRN 文件输出相关的控制。
   - 此组包括以下 7 条伪指令：

[*1] 伪指令的名称将对SRA74进行的指示称为“声明”，将影响输出文件的指示称为“指定”。

.COL	.LINE	列表格式（列数/行数）的指定
.LIST	.NLIST	列表输出/禁止的指定
.LISTM	.NLISTM	宏展开部分列表输出/禁止的指定
.PAGE		改页和标题的指定

5. 调试的支持
   - 进行与源程序行调试相关的控制。
   - 此组包括以下 2 条伪指令：

.FUNC	.ENDFUNC	功能的指定

注意事项

(a) SRA74 在指定“-c”汇编时，不对“.FUNC”和“.END-FUNC”伪指令行进行评价。因此，不能检测有关这些伪指令的错误。

6. 保留伪指令
   - 是用于将来扩展的保留伪指令。这些伪指令不影响汇编处理。
   - 此组包括以下 9 条伪指令：

.PROGNAME	程序名的声明
.IO .ENDIO .RAM .ENDRAM	区域名的声明
.PROCMAIN .PROCSUB .PROCINT	模块名的声明
.ENDPROC	模块结束的声明

以下按组说明伪指令的功能。

6.2　汇编控制

1. 汇编断言声明

   .ASSERT

   在汇编操作中将操作数所记述的字符串显示在画面上。

2. 汇编结束声明

   .END

   声明源程序的结束。SRA74 不处理该行以后的内容。

3. 汇编错误声明

   .ERROR

   将该伪指令的操作数所记述的字符串显示在画面上，并且结束汇编操作。

4. 条件汇编

   .IF (.ELSE) .ENDIF

   根据符号值的内容指定汇编的位置。能用于通过 1 个源程序管理对应多种规格的程序以及控制测试程序的汇编等情况。

5. 文件的读取

   .INCLUDE

   将其它的文件内容读到记述该指令的位置。能用于拆分和编辑大的源程序的情况。

6.3　地址控制

1. 同义的定义

.EQU 或 =

给符号定义绝对值。该伪指令给符号定义 0～7 的位值和 0000_{16}～$FFFF_{16}$ 的值。

2. 地址声明

.ORG 或 *=

声明后面的行地址。记述该伪指令的段为绝对属性，在连接时不能指定地址。能用于中断向量等地址固定的区域。

3. 区域确保

.BLKB

将由操作数指定的容量的存储区确保为 RAM 区。

4. 数据设定

.BYTE .WORD

将由操作数指定的数据生成到 ROM 区。

6.4　连接控制

1. 全局标号的指定

.BEXT .ZBEXT

指定外部参照的位符号名。另外，在此指定的位符号名必须在其它文件中被指定成公共位符号。

.EXT .SEXT .ZEXT

指定外部参照的标号名和符号名。另外，在此指定的标号名必须在其它文件中被指定成公共标号名。

.PUB

指定能从其它文件参照在该文件中定义的标号、符号或者位符号。

2. 区域的指定

.SECTION

指定该行的后面是用操作数指定的名称的区域。另外，通过该指令后的指令，SRA74 自动决定区域属性。该指定在出现其它区域指定指令之前有效。

.SECTION P 或.PMOD

指定该行的后面为一般页的 ROM 区。该指定在出现其它区域指定指令之前有效。

.SECTION R 或.RMOD

指定该行的后面为一般页的 RAM 区。该指定在出现其它区域指定指令之前有效。

.SECTION S 或.SMOD

指定该行的后面为专用页的 ROM 区。该指定在出现其它区域指定指令之前有效。

.SECTION Z 或.ZMOD

指定该行的后面为零页的 RAM 区。该指定在出现其它区域指定指令之前有效。

3. 连接文件名的指定

.OBJ .LIB

指定连接对象的 R74 文件名和库文件名。LINK74（连接程序）自动参照在此声明的文件，因此能简化连接时的命令输入。

4. 版本指定

.VER

指定 R74 文件的版本。如果指定 LINK74 的命令参数“-V”，就能确认 R74 文件间的版本一致性。

6.5　列表控制

1. 改页和标题的指定

.PAGE

指定列表的改页和标题。

2. 列表格式的指定

.COL .LINE

指定列表的列数、行数。这些伪指令只能在源文件中记述一次。

3. 列表输出/禁止的指定

.LIST .NLIST

进行对 PRN 文件的列表输出控制。可在如调试部分程序的情况等只需要一部分列表时使用。

4. 宏展开部分列表输出/禁止的指定

.LISTM .NLISTM

控制对 PRN 文件的宏展开部分的列表输出。

6.6　调试的支持

在指定命令参数“-C”时，SRA74 生成源程序级调试信息，但是也能输出仅用.FUNC～.ENDFUNC 括起来的部分源程序行调试信息。通过只限于必要部分的调试信息，能减少调试时的主机存储器的使用量。另外，即使对使用.FUNC 伪指令的源文件也能通过指定命令参数“-C”生成对于文件整体的源程序行调试信息。

1. 功能开始的指定

.FUNC

指定功能开始。

2. 功能结束的指定

.ENDFUNC

指定功能结束。

注意事项

1. 在指定“-C”进行汇编时，SRA74 不对“.FUNC”以及“.ENDFUNC”伪指令行进行评价。因此，必须注意不能检测与这些伪指令行相关的错误。

6.7　保留伪指令

保留伪指令是 SRA74 为将来的扩展而保留的伪指令。这些伪指令即使记述到源文件中也不会发生错误，并且也不影响汇编结果。一般来说，能结合市场上出售的字符串检索程序而用于确认源文件的内容等情况。

如果源文件中记述了如下的伪指令.PROCMAIN，就能通过字符串检索程序检索“.PROCMAIN”，进行主程序的检索。

例）　.PROCMAIN　KEY_SCAN　；键扫描程序部分的项目

第7章　宏指令

7.1　宏指令的功能

通过将使用 740 族的汇编语言和结构化描述语言的程序定义为宏，用户能将此时所起的名称（宏名）与操作码和结构化指令一样记述到源程序的操作数段。因此，如果事先准备好各种宏定义，就能在编程时将 740 族作为扩展后的新 CPU 来利用。这样，宏功能就提供用户能调整自身编程环境的功能。

7.2　宏指令的种类

宏指令能分为由 SRA74 包含的宏和用户定义的宏 2 种。

1. 系统宏指令
   - .REPEATI～.ENDM
     以记述在操作数中的参数个数为重复次数进行重复处理。
   - .REPEATC～.ENDM
     以操作数中作为参数的字符个数为重复次数进行重复处理。
   - .REPEAT～.ENDM
     只进行指定次数的重复处理。
   - .EXITM
     强制中止宏扩展。

2. 用户宏指令
   - .MACRO～.ENDM
     对宏指令进行定义。
   - .EXITM
     强制中止宏扩展。
   - .LOCAL
     将宏定义中使用的标号转换为宏内的局部标号。

详细内容请参照附录 E。

注意事项

1. 使用用户宏时必须事先定义宏。因此，宏定义通常在程序的开头进行定义或者在开头用.INCLUDE 伪指令读取宏定义的文件。另外，用户宏定义的嵌套最大为 20 层（但是，取决于主机的存储器容量）。
2. 如果将宏定义的文件作为其它文件（宏程序库），只要在程序的开头进行文件的包含，就能使用宏，此时不必要对各程序记述宏定义。
3. 在打印文件中的源程序旁边用‘+’表示被宏扩展的部分。
4. 对于被 LOCAL 声明的标号，按该顺序分配..n（n 为 10 进制数 ..1～..65535）的标号进行汇编。
5. 以句号‘.’开始的名称不能用于宏名和标号名等。
6. 对宏调用时的宏参数，在使用表示结构化描述的存储器参照的“[ ]”和“{ }”时，必须用‘"’（双引号）括起来。

[例]　mac: .macro para1, para2
　　para1 = para2
　　.endm
　　mac　"[work]", 10h

7.3　宏运算符

能用于宏指令的运算符一览表如表 7.1 所示。

表 7.1　宏运算符一览表

运算符	内　容
¥*1	在不能用作宏参数的特殊字符前面插入‘¥’后的字符被作为参数识别。 [格式] ¥字符
;; *2	在宏定义中定义不展开的注解。 [格式] ;;注解
"*3	在宏调用时，用于参数包含空格、制表符、逗号(,)以及保留字等的情况。 [格式] "字符串"
$*4	用于在参数前后连接字符串时。 [格式] (1)参数 $ 字符串 (2)字符串 $ 参数

注意事项

*1. 让 ESC 字符发挥作用，消除以下字符的特殊意义。

例）

```
[宏定义]
    DATA:   .MACRO    VAL
            .BYTE     VAL
            .ENDM
[调用例]
            DATA      "¥"HELLO !¥""
[宏扩展]
            .BYTE     "HELLO !"
            .ENDM
```

*2. 在将宏扩展结果输出到打印文件的情况下，当每次进行宏扩展时输出以 1 个分号（‘;’）开始的注解，但是不输出以 2 个分号（“;;”）开始的注解。

例）

```
[宏定义]
    LOOP:   .MACRO
            .LOCAL    LOOP1
            LDA       #20           ; comment
    LOOP1:  DEC       A             ;; comment
            BNE       LOOP1         ;; comment
            .ENDM
[调用例]
            LOOP
[宏扩展]
            LDA       #20           ; comment
     ..0:   DEC       A
            BNE       ..0
            .ENDM
```

*3. 在由操作数给予的参数中包含空格、制表符、逗号（,）以及保留字等时，用双引号（"）将整个参数括起来表示。

例）

[源记述]

```
SUB:    .REPEATI   INST, "NOP", "LDA #1", "JSR SUB1", "RTS"
        INST
        .ENDM
```

[宏扩展]

```
        SUB:
        NOP
        LDA    #1
        JSR    SUB1
        RTS
    .ENDM
```

*4. 用于在参数前后连接字符串并更改由参数给予的名称时。‘$’和字符串之间不能插入空格或者制表符。

例）

[源记述]

```
ADDWI:  .MACRO     MEM, IMM
        CLC
        LDA        MEM$_2
        ADC        #>IMM
        STA        MEM$_1
        LDA        #<IMM
        ADC        MEM$_2+1
        STA        MEM$_1+1
        .ENDM
```

[调用例]

```
        ADDWI      RAM, 1000H
```

[宏扩展]

```
        CLC
        LDA        RAM_2
        ADC        #>1000H
        STA        RAM_1
        LDA        #<1000H
        ADC        RAM_2+1
        STA        RAM_1+1
```

*5. 用‘( )’括起来的字符串作为一个参数处理。

例）

[宏定义]

```
ADDI:   .MACRO     MEM, IMM
        LDA        MEM
        CLC
        ADC        #IMM
        STA        MEM
        .ENDM
```

[调用例]

```
        ADDI       ( RAM,X ), 5
```

[宏扩展]

```
        LDA        ( RAM,X )
        CLC
        ADC        #5
        STA        ( RAM,X )
```

第8章　操作方法

8.1　启动方法

为了执行 SRA74，必须输入以下信息（输入参数）：

1. 源文件名（必须项目）
2. 命令参数

在 SRA74，从操作系统的命令行指定这些信息。另外，也能用环境变量 SRA74 定义。
在 8.2 节说明输入参数，在 8.3 节举例说明命令行输入，在 8.5 节举例说明环境变量 SRA74 的定义。

8.2　输入参数

8.2.1　源文件名

1. 指定汇编对象的源文件名。只能指定 1 个源文件名。
2. 在省略文件属性（.A74）时，作为默认值选择属性.A74。
3. 通过用全名指定文件名，也能汇编.A74 以外属性（例.ASM）的文件。
4. 对文件名能指定目录路径。在只指定文件名时，处理当前驱动器的当前目录中的文件。
   例）　A>SRA74 C:¥WORK¥TEST<RET>

8.2.2　命令参数

1. 命令参数由负号（-）及其后面的字符构成。
2. 大小写字母中的任何一个字符都有效。
3. 各参数能同时指定多个参数。此时，必须用空格将各参数隔开输入。
4. 通过连续 2 个负号能使该命令参数无效。例如，--L 能使 PRN 文件的生成无效。
5. 命令参数首先从环境变量 SRA74 处理。

命令参数的内容如**表 8.1** 所示。

表 8.1　命令参数一览表

命令参数	内　容
-.	禁止将错误信息以外的信息输出到画面。在用批文件等执行 SRA74 时，用于在画面上只显示错误信息的时候。
-I8	对‘!’运算符的运算进行‘!’运算后，只取低 8 位的值作为运算结果。
-A	指定只用汇编语言记述程序。*1
-BANK	将地址空间的上限从 FFFFH 扩展到 1FFFFH。运算符 BK 和 BL 有效。段 E 的信息不输出到可再定位文件和列表文件。
-C	对于文件中的全部行，输出源程序行调试信息。
-D	给符号设定数值。命令功能和伪指令.EQU 相同。指定的格式如下（同时定义多个符号时必须用‘:’隔开）： -D 符号=数值[:符号=数值....:符号=数值] 例）A>SRA74 SRCFILE -DS1=10:S2=20<RET>
-E	生成 TAG 文件以及启动编辑程序。编辑程序的程序名的指定格式如下：*2 -E[编辑程序名] 例）A>SRA74 SRCFILE -EMI<RET> “[ ]”部分可以省略。省略时只生成 TAG 文件。指定编辑程序名时，在汇编结束后将 TAG 文件作为参数启动编辑程序。但是，即使指定编辑程序名，在不发生错误时也不启动编辑程序。
-I	生成将结构化描述语言指令扩展为汇编语言指令的源文件（.I）。另外，该参数和-L 同时被指定时，扩展部分也被输出到生成的打印文件。
-K	为了不将 SRA74 生成的标号（“.I0”等）信息输出到 R74 文件。由此，用-S 选项输出的局部标号只限于用户定义的信息。
-L	生成 PRN 文件。没有该指定时，不生成 PRN 文件。
-LS	生成带符号列表的 PRN 文件。
-M	将宏扩展结果输出到 PRN 文件。没有该指定时，宏扩展结果不输入到列表。
-O	指定生成文件的输出路径。路径能指定目录或者驱动器名。没有该指定时，输出到和源文件相同的路径。指定格式如下： -O 路径名 例）A>SRA74 SRCFILE -OB: ¥USR<RET>
-S	给 R74 文件输出局部位符号、符号以及标号。
-U	忽视标号记述的‘:’（冒号）。没有该指定时，在程序中记述标号的情况下，如果没有‘:’就出错误。
-X	在汇编结束后，启动交叉参考程序 CRF74。*2 例）A>SRA74 SRCFILE -X<RET>

注意事项

1. 如果进行该指定，SRA74 在汇编时就不建立暂时文件。因此，能缩短汇编处理时间。另外，在将 I 文件作为源文件输入时也必须指定此命令参数。
2. 如果在当前目录或者命令路径中没有 CRF74，就出现系统错误。

8.3　输入方法

SRA74 通过在操作系统的提示符状态下输入命令行启动。启动命令的输入例子如图 8.1 所示。

如果对命令行输入检测到错误，就显示如图 8.2 所示的帮助画面，并且停止汇编。如果命令行输入正常，就开始汇编。如果汇编结束，就将错误数、警告数、全行数、注解行数以及段单位的存储容量输出到画面。汇编正常结束时的画面显示例子如图 8.3 所示。

```
A>SRA74 FILENAME -L -E <RET>
          ↓        ↓
汇编对象的源文件名    命令参数
```

图 8.1　启动命令的输入例子

```
A>SRA74<RET>
740 Family SRA74 V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

Usage: sra74 <filename> [ options ]
        -.      : all messages suppressed
        -a      : assemble assembly language source file
        -c      : source line data output to .R74 file
        -d      : define symbol ( syntax -ds1=1:s2=2 )
        -e      : make tag file
        -i      : make assembler source file
        -k      : suppress system label information to R74 file
        -l[s]   : make list file or symbol list file
        -m      : macro listing
        -o      : select drive and directory for output ( syntax -otmp )
        -q      : .EQU symbol multi define warning message output
        -s      : local symbol data output to .R74 file
        -u      : don't care ':' at end of label
        -x      : execute crf74
A>
```

图 8.2　命令错误时的帮助画面

```
A>SRA74 TEST<RET>
740 Family SRA74 V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

now processing pass 1 ( TEST.A74 )
----*----
now processing pass 2 ( TEST.A74 )
----*----

ERROR       COUNT                 00000
WARNING     COUNT                 00000
STRUCTURED STATEMENT              00702   LINES
TOTAL       LINE    ( SOURCE  )   00994   LINES
TOTAL       LINE    ( OBJECT  )   00994   LINES
COMMENT     LINE    ( SOURCE  )   00247   LINES
COMMENT     LINE    ( OBJECT  )   00147   LINES
OBJECT      SIZE    ( Z       )   00010   (000A)  BYTES
OBJECT      SIZE    ( R       )   00053   (0035)  BYTES
OBJECT      SIZE    ( P       )   01938   (0792)  BYTES

A>
```

图 8.3　正常结束时的画面表示

8.4　错误

8.4.1　错误的种类

执行 SRA74 时产生错误的原因有以下 3 个：

1. 与操作系统有关的错误
   是与磁盘和存储容量不够等执行 SRA74 的操作系统环境有关的错误。请在参照“**附录 A 错误信息一览表**”的基础上，利用操作系统的命令来对应。
2. 与 SRA74 的命令行输入有关的错误
   是与 SRA74 启动时的命令行输入有关的错误。请在确认本章内容的基础上重新输入命令。
3. 与汇编对象的源文件内容有关的错误
   是与标号的双重定义、未定义符号的参照等源文件内容有关的错误。请修正该处的源文件内容后再次进行汇编。
   在检测到汇编错误时，不生成 R74 文件。

SRA74 如果检测到错误和警告，就以**图 8.4** 的格式将错误内容（文件名、文件中的行号、连续行号、错误序号以及错误信息）输出到画面和 PRN 文件。请在参照附录 A 的按错误序号顺序的错误一览表的基础上进行对应。

```
A>SRA74 TEST<RET>
740 Family SRA74 V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

now processing pass 1 ( TEST.A74 )
----*----
now processing pass 2 ( TEST.A74 )
-
   115 E025   EAEA          BCC     LOOP2
TEST.A74 115 ( TOTAL LINE 115 ) Error 18: Relative jump is out of range
   127 E031   EAEA          LDA     #data
TEST.A74 127 ( TOTAL LINE 127 ) Error 22: Value is out of range "data"
---*
   551 E42B   EAEA          BRA     TEST2
TEST.A74 551 ( TOTAL LINE 551 ) Error 18: Relative jump is out of range
   593 E4FC   EAEAEA        LDA     (work,x                 ; data set
TEST.A74 593 ( TOTAL LINE 593 ) Error 23: "()" format error ";"
----

ERROR       COUNT              00004
WARNING     COUNT              00000
STRUCTURED STATEMENT           00702   LINES
TOTAL       LINE ( SOURCE  )   00994   LINES
TOTAL       LINE ( OBJECT  )   00994   LINES
COMMENT     LINE ( SOURCE  )   00247   LINES
COMMENT     LINE ( OBJECT  )   00147   LINES
OBJECT      SIZE ( Z       )   00010   (000A)   BYTES
OBJECT      SIZE ( R       )   00053   (0035)   BYTES
OBJECT      SIZE ( P       )   01938   (0792)   BYTES

A>
```

图 8.4　错误显示例子

8.4.2　给操作系统的返回值

在用操作系统的批文件等记述执行命令时，有时需要根据执行结果改变处理内容。SRA74 将执行结果分为表 8.2 的 4 个错误级返回给操作系统。

表 8.2　错误级一览表

错误级	执行结果的内容
0	正常结束
1	与汇编对象的源文件内容有关的错误
2	与 SRA74 的命令输入有关的错误
3	与操作系统有关的错误
4	根据^C（Ctrl + C）输入的强制结束

8.5　环境变量

SRA74 使用以下的环境变量：

- TMP

  指定在汇编时建立生成临时文件的目录名。没设定该环境变量时，临时文件被生成在和源文件相同的目录内。环境变量的设定例子如下所示：

  例）set TMP=A:¥TMP

- INC

  指定汇编时的包括文件的目录名。没设定该环境变量时，从当前目录检索由.INCLUDE 伪指令读取的文件。环境变量的设定例子如下所示：

  例）set INC=A:¥INC74

- SRA74

  能利用环境变量 SRA74 在命令行以外的地方指定文件名和命令参数。环境变量的设定例子如下所示：

  例）SET SRA74=-L -I

  因此，通过事先定义频繁使用的命令参数，不需要对每次汇编的命令行进行输入。另外，不想使用环境变量 SRA74 所定义的命令参数时，能通过命令行使其无效。
  环境变量 SRA74 所定义的信息比通过命令行指定的信息优先处理。例如，由以下格式指定时，(PC)

  ```
  SET    SRA74=-L    -I
  SRA74    FILE    --L    -S
  ```

  和从命令行输入以下指定时一样。

  ```
  SRA74    -L    -I    FILE    --L    -S
  ```

即通过命令行的指定使命令参数“-L”无效，并且不生成 PRN 文件。

附录 A　错误信息一览表

A.1　　系统错误一览表

如果在汇编中检测到系统错误，就在画面显示错误信息，并且中止汇编。系统错误一览表如表 A.1 所示。

表 A.1　系统错误一览表

错误信息	错误内容和对策
Usage: sra74 <filename> [options]	命令输入错误。 → 请参照帮助画面，重新输入命令。
Can't open xxx *1	没有找到该文件。 → 请确认源文件名，重新输入。
Can't create xxx	不能生成该文件。 → 请确认命令参数“-O”的指定，重新输入。
Out of disk space	文件输出所需要的磁盘容量不足。 → 请扩大磁盘容量。
Out of heap space	汇编程序运行所需要的存储容量不足。*2 → 请减少符号或标号的个数，或者拆分文件。
Can't find crf74.exe	没有找到 CRF74。 → 请将 CRF74 复制到当前目录或操作系统的命令路径中的目录中。
Can't find command.com for execute xxx	没有找到启动命令参数“-E”指定的编辑程序所需要的文件 COMMAND.COM。 → 请确认操作系统的命令路径的指定。

注意事项

*1.　SRA74 可处理的符号和标号的总数取决于执行 SRA74 的系统的可用内存容量。

A.2　　汇编错误一览表

如果检测出汇编错误，就将错误信息输出到画面和 PRN 文件。

汇编错误及其内容如表 A.2 所示。

表 A.2　汇编错误一览表

错误序号	错误信息	错误内容和对策
1	Already had same statement	在源文件中只能使用 1 次的伪指令被使用了 2 次以上。 例）　　.LINE　60 　　: 　　.LINE　80 → 请将声明改为 1 次。
2	Reference to forward label or symbol	伪指令参照了后面的标号或符号。 例）　　.ORG　TOP 　TOP: → 请在参照前定义标号或符号。
3	Division by 0	被 0 除。 → 请确认数值表达式的记述。

错误序号	错误信息	错误内容和对策
4	Illegal operand	操作数中含有不能使用的字符。 例）　　LDA　#&10 → 请确认操作数的记述。
5	Improper operand type	助记符和操作数的组合错误。 例）　　ASL　work,Y → 请确认指令的记述格式。
6	Invalid label definition	在不能定义标号的位置定义了标号。 例 1）　　LABEL1: .LINE　60 → 请删除标号。 例 2）　　LABEL2: .EQU　100 → 请将标号改为符号。
7	Invalid symbol definition	在不能定义符号的位置定义了符号。 例）　　SYMBOL　.LINE　60 → 请删除符号。
8	Out of maximum program size	地址超过 $0FFFF_{16}$。 例）　　.ORG　　0FFF0H 　　.WORD　1,2,3,4,5,6,7,8,9 → 请更改程序使地址在范围之内。
9	Label or symbol is multiple defined	同一标号或符号被定义了 2 次以上。 例）　　MAIN: NOP 　　MAIN: NOP → 请确认标号名或符号名。
10	Nesting error	超过嵌套层的界限。 → 请不要超过嵌套层的界限。
11	No .END statement	源文件中没有.END 语句。 → 请在程序的最后记述.END 语句。
12	No symbol definition	没有记述符号。 例）　　.EQU　60 → 请记述符号。
13	No ';' at the top of comment	注解栏的开头没有“;”（分号）。 例）　　LDA　#CNT counter set → 请在注解栏的开头插入“;”。
14	Not in conditional block	没有对应的.IF 语句却记述了.ELSE 或.ENDIF。 （在对应的.IF 语句有错时，也发生此错误。） 例）　　.IF DATA1 　　: 　　.ENDIF 　　: 　　.ELSE 　　: 　　.ENDIF → 请确认.IF 语句。
15	Operand is expected	指令需要的操作数不足。 例）　　.BYTE → 请确认操作数的记述。
16	Questionable syntax	助字符的拼写错误。 例）　　ADD　#DATA → 请确认助字符的拼写。

错误序号	错误信息	错误内容和对策
17	Reference to multi defined label or symbol	参照了被重复定义的标号或符号。 例）　MAIN: NOP MAIN: NOP BRA　MAIN → 请确认标号名或符号名。
18	Relative jump is out of range	相对转移指令的转移目标地址超过范围。 → 请重新编排程序或者更改指令。
19	Label or symbol is reserved word	标号或符号使用了和保留字相同的名称。 例）　A　.EQU　1FFH → 请更改标号名或符号名。
20	Reference to undefined label or symbol	参照了未定义的标号或符号。 → 请确认标号或符号。
21	Value error	数据记述格式错误。 例）　ADC　#'A → 请确认数据记述格式。
22	Value is out of range	数据范围超过了容许值。 例）　ADC　#100H → 请确认操作数记述格式。
23	"（）" format error	左括弧‘(’和右括弧‘)’的个数不一致。 例）　ADC　(WORK → 请确认操作数记述格式。
24	Relocatable error	在可再定位段中记述了伪指令.ORG。 例）　.SECTION　PROG LDA　WORK : .ORG　1000H : → 请分段。
25	No .SECTION statement	未记述伪指令.SECTION。 → 请在记述程序前记述伪指令.SECTION。
26	Reference to undefined section	参照了未定义的段名。 例）　LDA　#SIZEOF　UNDEF → 请确认该段。
27	Section type mismatch	指令或数据设定伪指令（.BYTE 等）与区域确保指令（.BLKB 等）混在一起。 例）　LDA　#WORK .BLKB 1 → 请分段。
28	Constant value is required	不能使用相对属性的标号和外部参照符号。 → 请指定绝对值。
30	else not associated with if	没有和 else 对应的 if。 → 请确认程序。
31	endif not associated with if	没有和 endif 对应的 if。 → 请确认程序。
32	next not associated with for	没有和 next 对应的 for。 → 请确认程序。

错误序号	错误信息	错误内容和对策
33	while not associated with do	没有和 while 对应的 do。 → 请确认程序。
34	break not inside for, do or switch	break 的使用位置不合适。 → 请确认程序。
35	case not inside switch	case 在 switch 的范围之外。 → 请确认程序。
36	deplicate case value	重复使用了相同的 case 值。 → 请确认程序。
37	more than one default	在同一 switch 语句中有 2 个以上的 default。 → 请确认程序。
38	default not inside switch	default 在 switch 语句的范围之外。 → 请确认程序。
39	ends not associated with switch	没有和 ends 对应的 switch。 → 请确认程序。
40	continue not inside for or do	continue 的使用位置不合适。 → 请确认程序。

A.3　警告一览表

如果检测出警告，就将警告信息输出到画面和 PRN 文件。警告及其内容如表 A.3 所示。

表 A.3　警告一览表

警告序号	警告信息	警告内容和对策
1	Phase warning	1)由伪指令.ORG 指定的地址先于此行以前的地址。 例）　　.ORG　0E000H MAIN:　LDA　　WORK : .ORG　0C000H
		2) 指令参照了在此行之后定义的标号或符号。 例）　　LDA　WK,X : WK　.EQU　80H → 请在参照的行之前定义标号或符号。
2	.END statement in include file	在包含文件中记述了伪指令.END。 → 请在源文件中记述.END。
3	statement has no effect	作为语句无意义。 → 请确认程序。
4	not case values for switch statement	在 switch 语句中无 case。 → 请确认程序。
5	statement not preceded by case or default	在 switch 语句中，语句出现在 case 和 default 前。 → 请确认程序。
6	.EQU symbol is multiple defined	用伪指令.EQU 将同一符号定义了 2 次以上。
7	'.SECTION E' requires command option '-BANK'	段 E 已被记述。请指定命令选项-BANK 进行汇编。

附录 B 指令一览表

B.1 符号表

在指令一览表中所用符号的意思如**表 B.1** 所示。

表 B.1 符号表

符号	内容	符号	内容
A	累加器	X	变址寄存器 X
Y	变址寄存器 Y	imm	立即数
zz	零页地址	hhll	一般页地址
i	位值（0～7）	bitsym	位符号
#	立即方式	¥	专用页方式

B.2 指令一览表

SRA74 可使用的全部指令如**表 B.2** 所示。在各指令的旁边表示寻址方式名和记述格式。

表 B.2 指令一览表

指令名	寻址方式名	记述格式
ADC	立即	ADC #imm
	零页	ADC zz
	零页 X	ADC zz,X
	零页间接 X	ADC (zz,X)
	零页间接 Y	ADC (zz),Y
	绝对	ADC hhll
	绝对 X	ADC hhll,X
	绝对 Y	ADC hhll,Y
AND	立即	AND #imm
	零页	AND zz
	零页 X	AND zz,X
	零页间接 X	AND (zz,X)
	零页间接 Y	AND (zz),Y
	绝对	AND hhll
	绝对 X	AND hhll,X
	绝对 Y	AND hhll,Y
ASL	累加器	ASL A
	零页	ASL zz
	零页 X	ASL zz,X
	绝对	ASL hhll
	绝对 X	ASL hhll,X
BBC	累加器位相对	BBC i,A,hhll BBC bitsym,A,hhll
	零页位相对	BBC i,zz,hhll BBC bitsym,hhll
BBS	累加器位相对	BBS i,A,hhll BBS bitsym,A,hhll
	零页位相对	BBS i,zz,hhll BBS bitsym,hhll

指令名	寻址方式名	记述格式
BCC	相对	BCC　hhll
BCS	相对	BCS　hhll
BEQ	相对	BEQ　hhll
BIT	零页	BIT　zz
	绝对	BIT　hhll
BMI	相对	BMI　hhll
BNE	相对	BNE　hhll
BPL	相对	BPL　hhll
BRA	相对	BRA　hhll
BRK	隐含	BRK
BVC	相对	BVC　hhll
BVS	相对	BVS　hhll
CLB	累加器位	CLB　i,A CLB　bitsym,A
	零页位	CLB　i,zz CLB　bitsym
CLC	隐含	CLC
CLD	隐含	CLD
CLI	隐含	CLI
CLT	隐含	CLT
CLV	隐含	CLV
CMP	立即	CMP　#imm
	零页	CMP　zz
	零页 X	CMP　zz,X
	零页间接 X	CMP　(zz,X)
	零页间接 Y	CMP　(zz),Y
	绝对	CMP　hhll
	绝对 X	CMP　hhll,X
	绝对 Y	CMP　hhll,Y
COM	零页	COM　zz
CPX	立即	CPX　#imm
	零页	CPX　zz
	绝对	CPX　hhll
CPY	立即	CPY　#imm
	零页	CPY　zz
	绝对	CPY　hhll
DEC	累加器	DEC　A
	零页	DEC　zz
	零页 X	DEC　zz,X
	绝对	DEC　hhll
	绝对 X	DEC　hhll,X
DEX	隐含	DEX
DEY	隐含	DEY
DIV	零页 X	DIV　zz,X
EOR	立即	EOR　#imm
	零页	EOR　zz
	零页 X	EOR　zz,X
	零页间接 X	EOR　(zz,X)
	零页间接 Y	EOR　(zz),Y
	绝对	EOR　hhll
	绝对 X	EOR　hhll,X
	绝对 Y	EOR　hhll,Y

指令名	寻址方式名	记述格式
FST	隐含	FST
INC	累加器	INC　A
	零页	INC　zz
	零页 X	INC　zz,X
	绝对	INC　hhll
	绝对 X	INC　hhll,X
INX	隐含	INX
INY	隐含	INY
JMP	绝对	JMP　hhll
	零页间接	JMP　(zz)
	绝对间接	JMP　(hhll)
JSR	绝对	JSR　hhll
	专用页	JSR　¥hhll
	零页间接	JSR　(zz)
LDA[*1]	立即	LDA　#imm
	零页	LDA　zz LDA　bitsym
	零页 X	LDA　zz,X
	零页间接 X	LDA　(zz,X)
	零页间接 Y	LDA　(zz),Y
	绝对	LDA　hhll LDA　bitsym
	绝对 X	LDA　hhll,X
	绝对 Y	LDA　hhll,Y
LDM	零页	LDM　#imm,zz
LDX	立即	LDX　#imm
	零页	LDX　zz
	零页 Y	LDX　zz,Y
	绝对	LDX　hhll
	绝对 Y	LDX　hhll,Y
LDY	立即	LDY　#imm
	零页	LDY　zz
	零页 X	LDY　zz,X
	绝对	LDY　hhll
	绝对 X	LDY　hhll,X
LSR	累加器	LSR　A
	零页	LSR　zz
	零页 X	LSR　zz,X
	绝对	LSR　hhll
	绝对 X	LSR　hhll,X
MUL	零页 X	MUL　zz,X
NOP	隐含	NOP

注意事项

*1.　在给 LDA 指令的操作数记述位符号时，只有该位符号的地址有效。

指令名	寻址方式名	记述格式
ORA	立即	ORA　#imm
	零页	ORA　zz
	零页 X	ORA　zz,X
	零页间接 X	ORA　(zz,X)
	零页间接 Y	ORA　(zz),Y
	绝对	ORA　hhll
	绝对 X	ORA　hhll,X
	绝对 Y	ORA　hhll,Y
PHA	隐含	PHA
PHP	隐含	PHP
PLA	隐含	PLA
PLP	隐含	PLP
ROL	累加器	ROL　A
	零页	ROL　zz
	零页 X	ROL　zz,X
	绝对	ROL　hhll
	绝对 X	ROL　hhll,X
ROR	累加器	ROR　A
	零页	ROR　zz
	零页 X	ROR　zz,X
	绝对	ROR　hhll
	绝对 X	ROR　hhll,X
RRF	零页	RRF　zz
RTI	隐含	RTI
RTS	隐含	RTS
SBC	立即	SBC　#imm
	零页	SBC　zz
	零页 X	SBC　zz,X
	零页间接 X	SBC　(zz,X)
	零页间接 Y	SBC　(zz),Y
	绝对	SBC　hhll
	绝对 X	SBC　hhll,X
	绝对 Y	SBC　hhll,Y
SEB	累加器位	SEB　i,A SEB　bitsym,A
	零页位	SEB　i,zz SEB　bitsym
SEC	隐含	SEC
SED	隐含	SED
SEI	隐含	SEI
SET	隐含	SET
SLW	隐含	SLW
STA[*1]	零页	STA　zz STA　bitsym
	零页 X	STA　zz,X
	零页间接 X	STA　(zz,X)
	零页间接 Y	STA　(zz),Y
	绝对	STA　hhll STA　bitsym
	绝对 X	STA　hhll,X
	绝对 Y	STA　hhll,Y

指令名	寻址方式名	记述格式
STP	隐含	STP
STX	零页	STX　zz
	零页 Y	STX　zz,Y
	绝对	STX　hhll
STY	零页	STY　zz
	零页 X	STY　zz,X
	绝对	STY　hhll
TAX	隐含	TAX
TAY	隐含	TAY
TST	零页	TST　zz
TSX	隐含	TSX
TXA	隐含	TXA
TXS	隐含	TXS
TYA	隐含	TYA
WIT	隐含	WIT

注意事项

*1.　在给 STA 指令的操作数记述位符号时，只有该位符号的地址有效。

附录 C 各寻址方式的指令一览表

C.1 各寻址方式的指令一览表

表示各寻址方式的指令记述格式和对应指令。说明中使用的符号列于附录 B.1 的符号表。

1. 隐含

对应指令	记述格式
BRK、CLC、CLD、CLI、CLT、CLV、DEX、DEY INX、INY、NOP、PHA、PHP、PLA、PLP、RTI RTS、SEC、SED、SEI、SET、SLW、STP、TAX TAY、TSX、TXA、TXS、TYA、WIT	BRK

2. 立即

对应指令	记述格式
ADC、AND、CMP、CPX、CPY、EOR、LDA LDX、LDY、ORA、SBC	ADC #imm

3. 累加器

对应指令	记述格式
ASL、DEC、INC、LSR、ROL、ROR	ASL A

4. 零页

对应指令	记述格式
ADC、AND、ASL、BIT、CMP、COM、CPX、CPY DEC、EOR、INC、LDA、LDX、LDY、LSR、ORA ROL、ROR、RRF、SBC、STA、STX、STY、TST	ADC zz
LDM	LDM #imm,zz

5. 零页变址 X

对应指令	记述格式
ADC、AND、ASL、CMP、DEC、DIV、EOR、INC LDA、LDY、LSR、MUL、ORA、ROL、ROR、SBC STA、STY	ADC zz,X

6. 零页变址 Y

对应指令	记述格式
LDX、STX	LDX zz,Y

7. 零页间接

对应指令	记述格式
JMP、JSR	JMP (zz)

8. 零页间接变址 X

对应指令	记述格式
ADC、AND、CMP、EOR、LDA、ORA、SBC、STA	ADC　(zz,X)

9. 零页间接变址 Y

对应指令	记述格式
ADC、AND、CMP、EOR、LDA、ORA、SBC、STA	ADC　(zz),Y

10. 绝对

对应指令	记述格式
ADC、AND、ASL、BIT、CMP、CPX、CPY、DEC EOR、INC、JMP、JSR、LDA、LDX、LDY、LSR ORA、ROL、ROR、SBC、STA、STX、STY	ADC　hhll

11. 绝对变址 X

对应指令	记述格式
ADC、AND、ASL、CMP、DEC、EOR、INC、LDA LDY、LSR、ORA、ROL、ROR、SBC、STA	ADC　hhll,X

12. 绝对变址 Y

对应指令	记述格式
ADC、AND、CMP、EOR、LDA、LDX、ORA、SBC STA	ADC　hhll,Y

13. 绝对间接

对应指令	记述格式
JMP	JMP　(hhll)

14. 专用页

对应指令	记述格式
JSR	JSR　¥hhhll

15. 零页位

对应指令	记述格式
CLB、SEB	CLB　i,zz CLB　bitsym

16. 累加器位

对应指令	记述格式
CLB、SEB	CLB　i,A CLB　bitsym,A

17. 相对

对应指令	记述格式
BCC、BCS、BEQ、BMI、BNE、BPL、BRA、BVC BVS	BCC　hhll

18. 零页位相对

对应指令	记述格式
BBC、BBS	BBC　i,zz,hhll BBC　bitsym,hhll

19. 累加器位相对

对应指令	记述格式
BBC、BBS	BBC　i,A,hhll BBC　bitsym,hhll

附录 D 伪指令一览

D.1 伪指令一览的说明

按照字母顺序记载了 SRA74 能使用的伪指令。书写规则如下所示：

1. 说明中“[ ]”内的部分表示可省略。
2. △、▲表示空格或者制表符码。△的部分为必需，▲的部分为可省略。
3. 在以下说明中，标号和伪指令之间用▲表示。记述标号不必使用‘:’（冒号），但是在省略‘:’时，标号和伪指令之间必需空格或者制表符码。

D.2 伪指令一览

.ASSERT　　声明汇编断言

格式

▲.ASSERT △'字符串'

内容

・在汇编操作中，将操作数所记述的字符串显示在画面上。

记述例

```
        .IF         MODE
          :
        .ELSE
        .ASSERT  'MODE is FALSE'
        .ENDIF
```

.BEXT　　指定外部参照（一般页位符号）

格式

▲.BEXT △位符号[,位符号,..,位符号]

内容

・指定对操作数指定的位符号进行外部参照。
・解释为由本伪指令指定的位符号在一般页内。
・本伪指令必须记述在参照标号的行之前。

记述例

```
        .BEXT    BIT0,BIT1,BIT2
          :
```

.BLKB　　确保 RAM 区（单位：字节）

格式

▲[标号:] ▲.BLKB △数值表达式

内容

- 确保由数值表达式指定大小的区域（单位:1 字节）。
- 数值表达式中使用的标号或者符号必须定义在该行之前。
- 不能给操作数记述带可再定位值的标号。

记述例

```
label:    .BLKB    10          ; 确保 10 字节的区域。
            :
```

.BYTE　　设定字节数据

格式

▲[格式:] ▲.BYTE △数值表达式

内容

- 设定 1 字节的常数数据。
- 在设定 2 个以上的数据时，必须用‘,’隔开指定各个数据。能指定的个数最多为 255 个。
- 能给操作数记述全局标号。

记述例

```
label1:   .BYTE    10          ; 设定 0AH。
label2:   .BYTE    'A','B'     ; 设定 41H 和 42H。
  :
```

.COL　　指定列数（默认值为 512）

格式

▲.COL　△数值表达式

内容

- 指定列表的 1 行的字符数（80～512）。
- 在指定 79 以下时为 80 个字符，在指定 513 以上时为 512 字符。
- 本伪指令在程序中只能记述 1 次。

记述例

```
.COL    80      ; 设定列数为 80。
  :
```

.END　　声明程序结束

格式

▲.END

内容

- 指定源程序的结束。
- 不汇编本伪指令以后的行。

记述例

```
  :
.END            ; 声明程序结束。
```

.ENDFUNC　　指定功能结束

格式

▲.ENDFUNC △标号

内容

- 指定功能（子程序）的结束。
- 通过该指令能调试源程序行。

记述例

```
        .FUNC     SUB
SUB:    LDA       #1
          :
          :
        RTS
        .ENDFUNC  SUB     ; 指定功能结束。
```

.EQU（或者‘=’）　　同义的定义

格式 1

符号△.EQU（或者‘=’）△数值表达式

格式 2

位符号△.EQU（或者‘=’）△数值表达式, 数值表达式

内容

- 给左边的符号赋值。
- 格式 1 给符号赋带符号的 16 位整数值。格式 2 给符号赋 0～7 的位值和带符号的 16 位整数值（地址）。
- 在数值表达式中使用的标号或者符号必须定义在该行之前。
- 不能给操作数记述带可再定位值的标号。
- 符号能被重新设定。
- 通过指定命令参数“-Q”，在重新设定符号时输出警告。

记述例

```
symbol   .EQU   1          ; 给 symbol 设定 1
  :
symbol   .EQU   2          ; 给 symbol 设定 2
  :
symbol   .EQU   3          ; 给 symbol 设定 3
  :
bitsym   .EQU   1, 23H     ; 给 bitsym 设定 23H 的位 1
  :
```

.ERROR　　声明断言错误

格式

▲.ERROR　△'字符串'

内容

- 与条件汇编指令（.IF）一起使用。
- 如果指定了不可能存在的条件，就在画面上显示操作数所记述的字符串，并且结束汇编操作。

记述例

```
.IF          MODE
        :
.ELSE
.ERROR   'undefined assemble mode'
.ENDIF
```

.EXT　　指定外部参照（一般页）

格式

▲.EXT　△标号或者符号[,标号或者符号,..,标号或者符号]

内容

- 指定在绝对寻址方式对操作数指定的标号进行外部参照。如果希望在零页寻址方式汇编时必须用.ZEXT 指定，如果希望在专用页寻址方式汇编时必须用.SEXT 指定。
- 本伪指令必须记述在参照标号的行之前。

记述例

```
.EXT      WORK1, WORK2, WORK3
  :
```

.FUNC　　指定功能开始

格式

▲.FUNC △标号

内容

- 指定功能（子程序）开始。
- 通过该指令能调试源程序行。
- 用于“.FUNC”伪指令的标号名不能用于其它“.FUNC”伪指令。

记述例

```
        .FUNC     SUB     ; 指定功能开始。
SUB:    LDA       #1
          :
          :
        RTS
        .ENDFUNC  SUB
```

.IF (.ELSE) .ENDIF　　条件汇编

格式

▲.IF　△表达式
　　<语句 1>
▲.ELSE
　　<语句 2>
▲.ENDIF

内容

- 操作数的表达式能记述数值表达式或者字符串表达式。
- 如果操作数的数值表达式为真（不为 0）则汇编语句 1，如果为假（0）则汇编语句 2。
- 如果操作数的字符串表达式为真（有字符串数据）则汇编语句 1，如果为假（没有字符串数据）则汇编语句 2。
- 能最大嵌套 20 层（但是，这取决于主机的存储容量）。
- 语句 1 和语句 2 能记述多行。
- 不能给操作数记述带可再定位值的标号。
- 操作数能记述条件表达式判定的比较运算符。能记述的条件判定表达式如表所示：

<	小于
>	大于
==	等于
!=	不等于
<=	小于等于
>=	大于等于

- 如果在数值表达式中使用未定义符号，就作为 0 值的符号处理。另外，如果在字符串表达式中记述未定义符号，就作为字符串处理。
- 通过逻辑运算符（‘||’、‘&&’）能对操作数组合最多 6 个条件表达式。但是，逻辑运算符之间没有优先顺序，条件表达式从左开始依次评价。

记述例

（1）

```
.IF     FLAG      ; 如果 FLAG 的值为真则汇编到.ELSE。
  :
  :
.ELSE             ; 如果为假则汇编到.ENDIF。
  :
  :
.ENDIF
```

（2）

```
ADD:  .MACRO   OP1,OP2,OP3
      .IF      "OP3"        ; 参数 OP3 存在时
                            ; 汇编到.ELSE。
      ADC      OP1,OP2,OP3
      .ELSE
      ADC      OP1,OP2
      .ENDIF
```

（3）

```
.IF     FLAG1 && FLAG2 || FLAG3 && FLAG4
  :       根据下表所示，决定此部分是否为汇编的对象。
  :
.ENDIF
```

FLAG1	FLAG2	FLAG3	FLAG4	结果
真	真	-	-	真
真	假	真	真	真
真	假	真	假	假
真	假	假	-	假
假	-	-	-	假

无论在真假任何一种情况下，‘-’都表示相同结果。

.INCLUDE　　读文件

格式

▲.INCLUDE　△文件名

内容

- 在记述本伪指令的位置，读取由操作数指定的文件内容。
- 必须用全称指定文件名。
- 能最大嵌套 4 层。
- 将嵌套层输出到打印文件。

记述例

```
.INCLUDE      TEST.INC;  读取 TEST.INC 的内容
  :
```

.LIB　　指定库文件名

格式

▲.LIB　△文件名[,文件名,..,文件名]

内容

- 指定连接对象的库文件名。
- 能指定的文件仅为.LIB 属性的文件。在指定除此以外的文件的情况下，连接时发生错误。
- 不能嵌套本伪指令。
- 不能给文件名记述目录路径和文件属性（.LIB）。

记述例

```
.LIB      LIB1,LIB2,LIB3
  :
```

.LINE　　指定每一页的行数（默认值为 54）

格式

▲.LINE △数值表达式

内容

- 指定每一页列表的行数（5～255）。
- 本伪指令只能在程序中记述 1 次。

记述例

```
.LINE    60      ; 设定行数为 60。
  :
```

.LIST　　开始列表输出（默认值）

格式

▲.LIST

内容

- 列表输出到 PRN 文件。
- 用于在通过伪指令.NLIST 停止对 PRN 文件的输出后重新开始列表输出的情况。

记述例

```
.NLIST      ; 停止列表输出。
  :         ; ".LIST"为止的内容不输出到 PRN 文件。
  :
.LIST       ; 开始列表输出。
  :         ; 本伪指令以后的内容输出到 PRN 文件。
  :
```

.LISTM　　开始宏扩展部分的列表输出（默认值）

格式

▲.LISTM

内容

- 将宏指令的扩展部分输出 PRN 文件。
- 用于在通过伪指令.NLISTM 停止宏扩展部分的输出后重新开始列表输出的情况。
- 在通过伪指令.NLIST 停止全部列表输出的情况下，.LISTM 指令无效。

记述例

```
.NLISTM     ; 停止宏扩展部分的列表输出。
  :         ; “.LISTM”为止的宏扩展部分不被输出。
  :
.LISTM      ; 开始宏扩展部分的列表输出。
  :         ; 输出本伪指令以后的宏扩展部分。
  :
```

.NLIST　　停止列表输出

格式

▲.NLIST

内容

- 停止对 PRN 文件的输出。
- 此状态能通过伪指令.LIST 解除。

记述例

```
.NLIST      ; 停止列表输出。
  :         ; “.LIST”为止的内容不输出到 PRN 文件。
  :
.LIST       ; 开始列表输出。
  :         ; 将本伪指令以后的内容输出到 PRN 文件。
  :
```

.NLISTM　　停止宏扩展部分的列表输出

格式

▲.NLISTM

内容

- 不对 PRN 文件输出宏指令的扩展部分。
- 此状态能通过伪指令.LISTM 解除。

记述例

```
.NLISTM     ; 停止宏扩展部分的列表输出。
  :         ; ".LISTM"为止的宏扩展部分不被输出。
  :
.LISTM      ; 开始宏扩展部分的列表输出。
  :         ; 输出本伪指令以后的宏扩展部分。
  :
```

.OBJ　　指定可再定位的文件名

格式

▲.OBJ △文件名[,文件名,..,文件名]

内容

- 指定连接对象可再定位的文件名。
- 能指定的文件仅为.R74 属性的文件。在指定除此以外的文件的情况下，连接时发生错误。
- 本伪指令不能嵌套。
- 不能给文件名记述目录路径和文件属性（.R74）。

记述例

```
.OBJ    OBJ1,OBJ2,OBJ3
  :
```

.ORG（或者"*="）　声明地址（默认值为 0000H）

格式

▲.ORG（或者"*="） △数值表达式

内容

- 声明该行以后的开始地址。
- 没有指定时，开始地址为 0000_{16}。
- 记述本伪指令的段为绝对属性。不包含本伪指令的段为相对属性。
- 数值表达式中使用的标号或者符号必须定义在该行之前。
- 不能给操作数记述带可再定位值的标号。

记述例

```
.ORG    0C000H      ; 将位置设定为 C000H。
  :
*=      0E000H      ; 将位置设定为 E000H。
```

.PAGE　列表改页和标题指定

格式

▲.PAGE △['标题']

内容

- 在该指令前进行列表改页，将由操作数指定的标题输出到列表的标头部分。标题必须用‘’（单引号）或者‘"’（双引号）括起来记述。
- 在指定列数为 80 列时，标题的字符数最多为 20 个字符，在指定列数为 105 到 512 时最多为 45 个字符，在指定列数为 81 到 104 时最多为列数减去 60 后的字符数。另外，省略标题时只改页。

记述例

```
.PAGE   'PROG1'     ; 将 PROG1 输出到 PRN 文件的标头部分。
  :
```

.PMOD　　指定 ROM 区（一般页）

格式

▲.PMOD

内容

- 指定该指令以后部分为一般页的 ROM 区。
- 该指令与给伪指令.SECTION 的操作数指定‘P’时相同。
- 该指定在其它的区域指定指令出现之前有效。
- 文件开头没有区域指定指令时，.PMOD 作为默认值被选择。因此，只限于文件开头的 ROM 区才能省略区域指定指令。

记述例

```
                .PMOD
                .EXT    MAIN,SUB
                .PUB    ENZAN
ENZAN:
                CLT
                  :
```

.PUB　　公共指定

格式

▲.PUB △位符号或者符号或者标号,....

内容

- 能从其它源文件参照操作数所指定的位符号、符号或者标号。
- 在仅由 Z 段和 R 段构成的 RAM 区的文件中，所有标号都作为全局属性处理。因此，能省略对标号的.PUB 指定。
- 本伪指令必须记述在定义标号或者符号的行之前。

记述例

```
                .RMOD
                .PUB    WORK1, WORK2, WORK3
WORK1:  .BLKB   1
WORK2:  .BLKB   1
WORK3:  .BLKB   1
                  :
```

.RMOD　　指定 RAM 区（一般页）

格式

▲.RMOD

内容

- 指定该指令以后部分为一般页的 RAM 区。
- 该指令与给伪指令.SECTION 的操作数指定‘R’时相同。
- 该指定在其它的区域指定指令出现之前有效。
- 在仅由 Z 段和 R 段构成的 RAM 区的文件中，所有标号都作为全局属性处理。因此，能省略对标号的.PUB 指定。

记述例

```
        .RMOD
WORK1:  .BLKB   1
WORK2:  .BLKB   1
WORK3:  .BLKB   1
          :
```

.SECTION　　指定区域

格式

▲.SECTION　△段名

内容

- 指定该行以后部分为操作数所指定的名称的区域。
- 在给操作数记述保留段名（P、R、S、Z）时，SRA74 将本伪指令以后部分作为各自的区域属性来识别处理。但是在记述任意的段名时，由该指令以后的指令决定区域属性，此时仅作为一般页的 ROM 区或者一般页的 RAM 区识别。
- 该指定在其它的区域指定指令出现之前有效。
- 文件中存在多个同名的段也无妨。
- 文件开头没有区域指定指令时，.SECTION P 作为默认值被选择。因此，只限于文件开头的 ROM 区才能省略区域指定指令。

记述例

```
            .SECTION    DATA    ; 开始 DATA 段。
datatop:
nulldt:     .BLKB 8
              :
            .SECTION    STACK   ; 开始 STACK 段。
            .BLKB       16
stacktop:
              :
            .SECTION    PROG    ; 开始 PROG 段。
_init:
            LDX         #stacktop
            TXS
            LDA         #SIZEOF DATA
            LDX         #datatop
              :
```

.SEXT　　指定外部参照（专用页）

格式

▲.SEXT △标号或者符号[, 标号或者符号,.., 标号或者符号]

内容

- 指定用专用页寻址方式（只限于 JSR ¥）或者绝对寻址方式对操作数指定的标号进行外部参照。希望在零页寻址方式汇编时必须用.ZEXT 指定。
- 本伪指令必须记述在参照标号的行之前。

记述例

```
                .SECTION P
                .EXT    WORK1, WORK2
                .SEXT   SUB
START:
                LDA     WORK1
                JSR     ¥SUB
```

.SMOD　　指定 ROM 区（专用页）

格式

▲.SMOD

内容

- 指定该指令以后部分为专用页的 ROM 区。
- 该指令与给伪指令.SECTION 的操作数指定‘S’时相同。
- 该指定在其它的区域指定指令出现之前有效。

记述例

```
                .SMOD
                .EXT    MAIN,SUB
                .PUB    ENZAN
ENZAN:
                CLT
                  :
```

.VER　　指定程序版本

格式

▲.VER △'字符串'

内容

- 指定可再定位的文件版本。
- 如果指定命令参数“-V”，LINK74 就确认各可再定位文件中的版本指定是否一致。由此可确认可再定位文件间的版本匹配性。有关命令参数“-V”的详细内容请参照 LINK74 操作手册。
- 通过比较字符串来确认版本是否一致。由于区分大小写字母，因此记述时必须注意。
- 本伪指令只能在程序中记述一次。

记述例

```
        .VER    'V.1.0'     ; 指定版本“V.1.0”。
          :
```

.WORD　　设定字数据

格式

▲[标号:] ▲.WORD △数值表达式

内容

- 设定数值表达式的值（单位：字）。
- 在设定 2 个以上的数据时，必须用‘,’隔开指定各数据。1 行能指定的个数最多为 16 个。
- 从低位字节开始设定数据。
- 能给操作数记述全局标号。

记述例

```
label:  .WORD   0E000H      ; 设定 00H、E0H。
        .WORD   symbol      ; 从低位字节开始设定 symbol 的值。
          :
```

.ZBEXT　　指定外部参照（零页位符号）

格式

▲.ZBEXT　△位符号[,位符号,..,位符号]

内容

- 指定对操作数指定的位符号进行外部参照。
- 解释为由本伪指令指定的位符号在零页内。
- 本伪指令必须记述在参照标号的行之前。

记述例

```
        .ZBEXT    BIT0,BIT1,BIT2
          :
```

.ZEXT　　指定外部参照（零页）

格式

▲.ZEXT　△标号或者符号[,标号或者符号,..,标号或者符号]

内容

- 指定用零页寻址方式对操作数指定的标号进行外部参照。希望在绝对寻址方式汇编时必须用.EXT 指定，希望在专用页寻址方式汇编时必须用.SEXT 指定。
- 本伪指令必须记述在参照标号的行之前。

记述例

```
              .SECTION Z
              .ZEXT    WORK1,WORK2
              .SEXT    SUB
START:
              LDA      WORK1
              JSR      ¥SUB
```

.ZMOD　　指定 RAM 区（零页）

格式

▲.ZMOD

内容

- 指定该指令以后部分为零页的 RAM 区。
- 该指令与给伪指令.SECTION 的操作数指定‘Z’时相同。
- 该指定在其它的区域指定指令出现之前有效。
- 在仅由 Z 段和 R 段构成的 RAM 区的文件中，所有标号都作为全局属性处理。因此，能省略对标号的.PUB 指定。

记述例

```
        .ZMOD
WORK1:  .BLKB   1
WORK2:  .BLKB   1
WORK3:  .BLKB   1
          :
```

D.3　保留伪指令一览

以下的伪指令为了将来的扩展被保留。即使记述这些伪指令也不影响汇编。

.ENDIO　　声明 I/O 区结束（保留）

格式

▲.ENDIO

内容

- 声明 I/O 区的结束。
- 将在.IO 和.ENDIO 之间记述的标号和符号作为 I/O 区来解释。

记述例

```
        .IO
port0   .EQU    00H
port1   .EQU    01H
        .ENDIO
```

.ENDPROC　　声明程序模块结束（保留）

格式

▲.ENDPROC

内容

- 声明主程序、子程序和中断处理程序的各模块的结束。
- 将从.PROCINT、.PROCMAIN 或者.PROCSUB 到.ENDPROC 的范围作为一个程序模块来解释。

记述例

```
        .PROCMAIN
MAIN:
          :
        JMP     MAIN
        .ENDPROC
```

.ENDRAM　　声明 RAM 区结束（保留）

格式

▲.ENDRAM

内容

- 声明 RAM 区的结束。
- 将在.RAM 和.ENDRAM 之间记述的标号和符号作为 RAM 区来解释。

记述例

```
        .RAM
work0:  .BLKB   1
work1:  .BLKB   1
        .ENDRAM
```

.IO　　声明 I/O 区开始（保留）

格式

▲.IO

内容

- 声明 I/O 区的开始。
- 将在.IO 和.ENDIO 之间记述的标号或者符号作为 I/O 区来解释。

记述例

```
        .IO
port0   .EQU    00H
port1   .EQU    01H
        .ENDIO
```

.PROCINT　　声明中断处理程序开始（保留）

格式

▲.PROCINT　△[标号]

内容

- 声明中断处理程序的开始。
- 将从.PROCINT 到.ENDPROC 的范围作为中断处理程序来解释。
- SRA74 将操作数记述的标号作为该行的标号处理。

记述例

```
.PROCINT    INT
  :
  :
.ENDPROC
```

.PROCMAIN　　声明主程序开始（保留）

格式

▲.PROCMAIN　△[标号]

内容

- 声明主程序的开始。
- 将从.PROCMAIN 到.ENDPROC 的范围作为主程序来解释。
- SRA74 将操作数记述的标号作为该行的标号处理。

记述例

```
.PROCMAIN    MAIN
  :
.ENDPROC
```

.PROCSUB　　声明子程序开始（保留）

格式

▲.PROCSUB　△[标号]

内容

- 声明子程序的开始。
- 将从.PROCSUB 到.ENDPROC 的范围作为子程序来解释。
- SRA74 将操作数记述的标号作为该行的标号处理。

记述例

```
.PROCSUB    SUB
  :
  :
.ENDPROC
```

.PROGNAME　　声明程序名（保留）

格式

▲.PROGNAME　△程序名

内容

- 声明程序名。
- 将操作数内容作为程序标题来解释。

记述例

```
.PROGNAME    打印机控制程序
```

.RAM　　声明 RAM 区开始（保留）

格式

▲.RAM

内容

- 声明 RAM 区的开始。
- 将在.RAM 和.ENDRAM 之间记述的标号和符号作为 RAM 区来解释。

记述例

```
        .RAM
work0:  .BLKB   1
work1:  .BLKB   1
        .ENDRAM
```

附录 E　宏指令一览

E.1　宏指令一览的说明

按字母顺序记载了 SRA74 能使用的宏指令。书写的规则如下所示：

1. 说明中“[ ]”内的部分表示可省略。
2. △、▲表示空格或者制表符码。△的部分为必需，▲的部分为可省略。
3. 在下面说明中，标号和宏指令之间用▲表示。记述标号时不必使用‘:’（冒号），但是在省略‘:’时，标号和宏指令之间必需空格或者制表符码。

E.2　宏指令一览

.ENDM　　ENDM 宏指令

格式

▲.ENDM

内容

・此指令指定全部宏定义部分的结束。

记述例

```
[宏定义]
ADD:    .MACRO   VAL
        CLC
        ADC      VAL
[调用例]
        ADC      #10
[宏扩展]
        CLC
        ADC      #10
        .ENDM
```

.EXITM　　EXITM 宏指令

格式

▲.EXITM

内容

・此指令中止全部的宏扩展，将控制移交给最近的.ENDM。

记述例

[宏定义]

```
DATA1:  .MACRO   VAL
        .IF      LABEL
        .BYTE    VAL
        .EXITM
        .ENDIF
        .WORD    VAL
        .ENDM
```

[调用例]

```
LABEL   .EQU     1
        DATA1    10
```

[宏扩展]

```
        .IF      LABEL
        .BYTE    10
        .EXITM
        .ENDIF
        .ENDM
```

.LOCAL　　LOCAL 宏指令

格式

▲.LOCAL　△标号,[标号,..,标号]

内容

- 此指令将在宏定义中定义的标号作为宏内的局部标号。
- 对于被.LOCAL 声明的标号，按其顺序分配称为..n（n 为 10 进制数..1～..65535）的标号进行汇编。因此用户必须注意不能使用..n 标号。

记述例

```
[宏定义]
    LOOP:   .MACRO
            .LOCAL   LOOP1
            LDA      #20
    LOOP1:  DEC      A
            BNE      LOOP1
            .ENDM
[调用例]
            LOOP
[宏扩展]
            LDA      #20
      ..0:  DEC      A
            BNE      ..0
            .ENDM
```

.MACRO～.ENDM　　MACRO 宏指令

格式

▲[宏名:] ▲.MACRO △[参数 1，参数 2,..，参数 n]

内容

- 从记述.MACRO 的行开始宏定义，在记述.ENDM 的行结束该宏定义。
- 给.MACRO 指令行起的标号成为此宏定义的名称。
- 宏定义有带参数和不带参数的宏，在使用带参数的宏时，必须用宏调用语句传递需要的参数。
- 在进行宏调用和展开参数时，根据宏定义记述的形式参数的顺序传递参数。
- 在.MACRO～.ENDM 之间，可以记述汇编语言指令、结构化描述指令、宏指令以及.INCLUDE 以外的伪指令。但是，宏定义的嵌套最大为 20 层。
- 如果宏调用在宏定义之后，就可以在程序的任意位置进行宏调用。如果宏定义只出现在宏调用之后，就出错。
- 由宏调用指定在操作数中的参数从左起按顺序替换对应的宏定义的形式参数。另外，参数的个数和定义的形式参数的个数不一致也无妨。但是，如果参数的个数多于形式参数的个数，其多余的部分就被忽视；如果少于形式参数的个数，就将不足的形式参数视为空字符（长度为 0 的字符串）。
- 与宏定义中使用的形式参数无关，参数可以指定任意的个数，但是必须全部写在一行以内。由于参数和参数之间用‘,’分隔，所以在将逗号或空格作为参数传递时必须用‘"’括起来。但是，‘()’内的逗号不作为参数的分隔处理。
- 宏扩展部分在列表文件中用‘+’表示。
- 能用同一个宏名进行多次宏定义。此时，如果进行宏调用，调用前刚被定义的宏定义就成为调用对象。

记述例

例 1）不带操作数的宏定义

[宏定义]

```
ADDa:   .MACRO
        LDA     ABC
        LDX     #DEF
        CLC
        ADC     TABLE,X
        STA     GHI
        .ENDM
```

[调用例]

```
        ADDa
```

[宏扩展]

```
        LDA     ABC
        LDX     #DEF
        CLC
        ADC     TABLE,X
        STA     GHI
        .ENDM
```

记述例

例 2）带操作数的宏定义

[宏定义]

```
ADDb:   .MACRO   V1,IMM,V2      ; 形式参数（V1,IMM,V2）
        LDA      V1
        LDX      #IMM
        CLC
        ADC      TABLE,X
        STA      V2
        .ENDM
```

[调用例]

```
        ADDb     WORK1,10,WORK2
```

[宏扩展]

```
        LDA      WORK1
        LDX      #10
        CLC
        ADC      TABLE,X
        STA      WORK2
        .ENDM
```

例 3）宏的嵌套

[宏定义]

```
ADD:    .MACRO   SRC
        CLC
        ADC      SRC
        .ENDM
```

[调用例]

```
ADDW:   .MACRO   SRC
        ADD      SRC            ; 在宏内的宏调用
        ADD      SRC+1
        .ENDM
```

例 4）宏的递归调用

[宏定义]

```
MAC:    .MACRO                  ; 第一次定义
DATA:   .BLKB    1
MAC:    .MACRO   VALUE          ; 第二次定义
        LDM      #VALUE,DATA
        .ENDM
        .ENDM
```

[调用例]

```
        MAC                     ; 区域的确保和新的宏定义
          :
        MAC      10H            ; 被重新定义的宏调用
```

.REPEAT～.ENDM　　REPEAT 宏指令

格式

▲[标号:] ▲.REPEAT △次数

内容

- 以操作数指定的次数，重复汇编在.REPEAT 和.ENDM 之间的 740 族的指令。
- .REPEAT 指令行的标号成为被生成行的最初标号。
- 在.REPEAT～.ENDM 之间，能记述汇编语言指令、结构化描述指令、宏指令以及.INCLUDE 以外的伪指令。但是，宏的嵌套最大为 20 层。
- 能给操作数记述数值常数和符号常数（标号），但是不能记述带可再定位值的标号。
- 能给操作数记述的值的范围是 0～32767。如果指定超过此范围的值就出错。

记述例

[源程序记述例]

```
TIME5:  .REPEAT  5
        NOP
        .ENDM
```

[宏扩展后]

```
TIME5:  .REPEAT  5
        NOP
        .ENDM
TIME5:
        NOP
        NOP
        NOP
        NOP
        NOP
```

.REPEATC～.ENDM　　REPEATC 宏指令

格式

▲[标号:] ▲.REPEATC △形式参数,实际参数

内容

- 以操作数中作为实际参数的字符个数为重复次数进行.ENDM 为止的重复汇编。
- 每次重复时从实际参数中将字符逐一取出传递给形式参数。
- .REPEATC 指令行的标号成为被生成行的最初标号。
- 在.REPEATC～.ENDM 之间，能记述汇编语言指令、结构化描述指令、宏指令以及.INCLUDE 以外的伪指令。但是，宏的嵌套最大为 20 层。
- 如果字符串含有空格、制表符、‘,’（逗号）等的特殊字符，就用‘"’将全部字符括起来指定字符串。此时使用除去‘"’的字符串。

记述例

例 1）

[源程序记述例]

```
DATA:   .REPEATC    VAL,ABCDE
                      ↑      ↑
                  形式参数  实际参数
        .BYTE       'VAL'
        .ENDM
```

[宏扩展后]

```
DATA:   .REPEATC    VAL,ABCDE
        .BYTE       'VAL'
        .ENDM
DATA:
        .BYTE       'A'
        .BYTE       'B'
        .BYTE       'C'
        .BYTE       'D'
        .BYTE       'E'
```

记述例

例 2）

[源程序记述例]

```
DATA:   .REPEATC    VAL, "ABC,;"
                      ↑       ↑
                  形式参数  实际参数
        .BYTE       'VAL'
        .ENDM
```

[宏扩展后]

```
DATA:   .REPEATC    VAL, "ABC,;"
        .BYTE       'VAL'
        .ENDM
DATA:
        .BYTE       'A'
        .BYTE       'B'
        .BYTE       'C'
        .BYTE       ','
        .BYTE       ';'
```

.REPEATI～.ENDM　　REPEATI 宏指令

格式

▲[标号:] ▲.REPEATI △形式参数,实际参数[,实际参数,..,实际参数]

内容

- 以操作数中记述的参数个数为重复次数进行.ENDM 为止的重复汇编。
- 每次重复时从操作数记述的参数中逐一取出传递给形式参数。
- .REPEATI 指令的标号成为被生成行的最初标号。
- 在.REPEATI～.ENDM 之间，能记述汇编语言指令、结构化描述指令、宏指令以及.INCLUDE 以外的伪指令。但是，宏的嵌套最大为 20 层。
- 能给实际参数记述数值常数、字符常数、符号常数（标号）以及字符串常数，但是不能记述其它的宏指令。
- 如果实际参数含有空格、制表符、‘,’（逗号），就用‘"’将全部字符括起来表示。

记述例

例 1）

[源程序记述例]

```
SUB:    .REPEATI    INST,"NOP","LDA #1","JSR SUB1","RTS"
                      ↑              ↑
                  形式参数        实际参数
        INST
        .ENDM
```

[宏扩展后]

```
SUB:    .REPEATI    INST,"NOP","LDA A,#1","JSR SUB1","RTS"
        INST
        .ENDM
SUB:
        NOP
        LDA         #1
        JSR         SUB1
        RTS
```

例 2）

[源程序记述例]

```
DATA:   .REPEATI    VAL,0,1,2,'"HELLO !!"'
                      ↑      ↑
                  形式参数  实际参数
        .BYTE       VAL
        .ENDM
```

[宏扩展后]

```
DATA:   .REPEATI    VAL,0,1,2,'"HELLO !!"'
        .BYTE       VAL
        .ENDM
SUB:
        .BYTE       0
        .BYTE       1
        .BYTE       2
        .BYTE       'HELLO !!'
```

附录 F　结构化指令一览

F.1　结构化指令一览的说明

按字母顺序记载了 SRA74 能使用的结构化指令。书写的规则如下所示：

1. 说明中“[ ]”内的部分表示可省略。
2. △、▲表示空格或者制表符码。△的部分为必需，▲的部分为可省略。
3. 记述标号时不必使用‘:’（冒号），但是在省略‘:’时，各结构化指令之间必需空格或者制表符码。通常推荐附加‘:’记述。

F.2　结构化指令一览

BREAK　　BREAK 语句

格式

▲[标号:] ▲（L）BREAK

内容

- break 语句停止对应的 for 语句、do 语句或 switch 语句的执行，将控制移交给下一个执行语句。
- 只能用于 for 语句、do 语句和 switch 语句。

CONTINUE　　CONTINUE 语句

格式

▲[标号:] ▲（L）CONTINUE

内容

- continue 语句假想地在包括自身的最小重复的 for 语句和 do 语句中的最后语句的后面插入空语句，并将控制移交给该空语句。
- 只能用于 for 语句和 do 语句。

DO～WHILE　　DO 语句

格式

▲[标号:] ▲（L）DO
　　<语句>
▲[标号:] ▲WHILE △条件表达式

内容

- do 语句在满足条件表达式（真）期间重复执行语句，在执行语句后判断是否重复。因此，如果在某种条件下结束重复时希望结束前再次执行后退出重复，使用 do 语句就比较方便。
- 如果在条件表达式中记述 ever，就成为无限循环。
- 条件表达式的详细内容请参照句法图。

FOR～NEXT　　FOR 语句

格式

▲[标号:] ▲（L）FOR △条件表达式
　　<语句>
▲[标号:] ▲NEXT

内容

- for 语句是控制重复的指令，在指定的条件表达式为真期间重复执行语句。
- 如果在条件表达式中记述 ever，就成为无限循环。
- 条件表达式的详细内容请参照句法图。

IF ～ (ELSE) ～ ENDIF　　IF 语句

格式

▲[标号:] ▲（L）IF △条件表达式
　　<语句>
▲[[标号:] ▲（L）ELSE]
　　<语句>
▲[标号:] ▲ENDIF

内容

- if 语句是将控制的流程变成 2 个方向的指令，转移的方向由条件表达式决定。根据条件表达式的运算结果是“0”（假）还是“1”（真）判断转移。如果是真，就将控制移交给紧接其后的指令；如果是假并且有 else 时就将控制移交给紧接 else 之后的指令，没有 else 时就移交给 endif 以后的指令。
- else 的部分可以省略。
- if 语句的嵌套数没有限制。
- 在 if 语句为复杂的嵌套时，if 和 else 的对应关系是以最近的 if 和 else 按顺序配对。
- 条件表达式的详细内容请参照句法图。

SWITCH ～ CASE ～ENDS　　SWITCH 语句

格式

▲[标号:] ▲（L）SWITCH △条件表达式
▲[标号:] ▲CASE △常数
　　<语句>
▲[标号:] ▲CASE △常数
　　<语句>
　　：
▲[标号:] ▲DEFAULT
　　<语句>
▲[标号:] ▲ENDS

内容

- switch 语句根据条件表达式的值将控制移交给某个语句。表达式的值和写在语句前的 case 前缀的常数进行比较，将控制移交给一致的语句。在表达式的值和 case 指定的值不一致时，如果有 default 前缀，就将控制移交给该语句。如果没有 default 就不执行任何语句而退出 switch 语句。
- default 的部分可以省略。
- 将控制移交给一致的 case 语句后，在其后记述的 case 语句也被顺序执行。如果不转移到下个 case 语句而要退出 switch 语句时，就用 break 语句退出 switch 语句。
- 不能将相对值和外部参照值用作 case 的值。
- switch 语句的嵌套数没有限制。
- 条件表达式和常数的详细内容请参照句法图。

赋值　　**赋值语句**

格式

▲[标号:] ▲左边▲=▲右边

内容

- 将右边赋值给左边。另外，对存储器位变量、寄存器位变量、标志变量的赋值，只能使用带 0 或 1 的数值常数和符号常数。
- 左边和右边的详细内容请参照句法图。

F.3　展开例

说明将结构化描述语言展开为汇编语言的例子。

F.3.1　赋值语句的展开例

说明条件转移指令以外的汇编语言和结构化描述语言的对应关系。在 F.3.2 的条件表达式的展开中说明条件转移。另外，在赋值语句的展开例中所使用的符号的意思如下所示：

表 F.1　符号表

符号	内容	符号	内容
A	累加器	X	变址寄存器 X
Y	变址寄存器 Y	S	堆栈指针
P	处理器状态寄存器	C	进位标志
Z	零标志	I	中断禁止标志
D	10 进制模式标志	T	X 变址运算模式标志
V	溢出标志	N	负标志
imm	立即数据	zz	零页地址
hhll	一般页地址	#	立即方式
MEM	存储器	FLAG	存储器位

(A) 寄存器和标志赋值语句的展开例

表 F.2　寄存器和标志赋值语句的展开例

分类	结构化描述语言	汇编语言
Bit 处理	BIT_A0 = 0	CLB　BIT_A0
	BIT_A0 = 1	SEB　BIT_A0
标志设定	C = 0	CLC
	C = 1	SEC
	D = 0	CLD
	D = 1	SED
	I = 0	CLI
	I = 1	SEI
	T = 0	CLT
	T = 1	SET
	V = 0	CLV

分类	结构化描述语言	汇编语言
数据传送指令	A = imm	LDA　# imm
	X = imm	LDX　# imm
	Y = imm	LDY　# imm
	X = A	TAX
	A = X	TXA
	Y = A	TAY
	A = Y	TYA
	X = S	TSX
	S = X	TXS
	[S] = A	PHA
	[S] = P	PHP
	A = [S]	PLA
	P = [S]	PLP
运算指令	A = A + imm WITH_C	ADC　# imm
	A = A - imm WITH_C	SBC　# imm
	A = ++ A	INC　A
	A = -- A	DEC　A
	X = ++ X	INX
	X = -- X	DEX
	Y = ++ Y	INY
	Y = -- Y	DEY
	A = A & imm	AND　# imm
	A = A | imm	ORA　# imm
	A = A ^ imm	EOR　# imm
	A = A << imm	ASL　A
	A = A >> imm	LSR　A
	A = A << imm WITH_C	ROL　A
	A = A >> imm WITH_C	ROR　A

(B) 存储器赋值语句的展开例

表 F.3　存储器赋值语句的展开例

分类	结构化描述语言	汇编语言
数据传送指令	A = [zz]	LDA　zz
	X = [zz]	LDX　zz
	Y = [zz]	LDY　zz
	[zz] = imm	LDM　# imm, zz
	[zz] = A	STA　zz
	[zz] = X	STX　zz
	[zz] = Y	STY　zz
运算指令	A = A + [zz] WITH_C	ADC　zz
	A = A - [zz] WITH_C	SBC　zz
	[zz] = ++ [zz]	INC　zz
	[zz] = -- [zz]	DEC　zz
	A = A & [zz]	AND　zz
	A = A | [zz]	ORA　zz
	A = A ^ [zz]	EOR　zz
	[zz] = [zz] << imm	ASL　zz
	[zz] = [zz] >> imm	LSR　zz
	[zz] = [zz] << imm WITH_C	ROL　zz
	[zz] = [zz] >> imm WITH_C	ROR　zz
	[zz] = ~ [zz]	COM　zz
Bit 处理	[FLAG] = 0	CLB　FLAG
	[FLAG] = 1	SEB　FLAG

(C) 寻址方式赋值语句的展开例

表 F.4　寻址方式赋值语句的展开例

分类	结构化描述语言	汇编语言
装入指令	A = [zz,X]	LDA　zz,X
	A = [hhll]	LDA　hhll
	A = [hhll,X]	LDA　hhll,X
	A = [hhll,Y]	LDA　hhll,Y
	A = [(zz,X)]	LDA　(zz,X)
	A = [(zz),Y]	LDA　(zz),Y
	X = [zz,Y]	LDX　zz,Y
	X = [hhll]	LDX　hhll
	X = [hhll,Y]	LDX　hhll,Y
	Y = [zz,X]	LDY　zz,X
	Y = [hhll]	LDY　hhll
	Y = [hhll,X]	LDY　hhll,X
存储指令	[zz,X] = A	STA　zz,X
	[hhll] = A	STA　hhll
	[hhll,X] = A	STA　hhll,X
	[hhll,Y] = A	STA　hhll,Y
	[(zz,X)] = A	STA　(zz,X)
	[(zz),Y] = A	STA　(zz),Y
	[zz,Y] = X	STX　zz,Y
	[hhll] = X	STX　hhll
	[zz,X] = Y	STY　zz,X
	[hhll] = Y	STY　hhll
加减法指令	A = A + [zz,X] WITH_C	ADC　zz,X
	A = A + [hhll] WITH_C	ADC　hhll
	A = A + [hhll,X] WITH_C	ADC　hhll,X
	A = A + [hhll,Y] WITH_C	ADC　hhll,Y
	A = A + [(zz,X)] WITH_C	ADC　(zz,X)
	A = A + [(zz),Y] WITH_C	ADC　(zz),Y
	A = A - [zz,X] WITH_C	SBC　zz,X
	A = A - [hhll] WITH_C	SBC　hhll
	A = A - [hhll,X] WITH_C	SBC　hhll,X
	A = A - [hhll,Y] WITH_C	SBC　hhll,Y
	A = A - [(zz,X)] WITH_C	SBC　(zz,X)
	A = A - [(zz),Y] WITH_C	SBC　(zz),Y
	[zz,X] = ++ [zz,X]	INC　zz,X
	[hhll] = ++ [hhll]	INC　hhll
	[hhll,X] = ++ [hhll,X]	INC　hhll,X
	[zz,X] = -- [zz,X]	DEC　zz,X
	[hhll] = -- [hhll]	DEC　hhll
	[hhll,Y] = -- [hhll,Y]	DEC　hhll,X
逻辑运算指令	A = A & [zz,X]	AND　zz,X
	A = A & [hhll]	AND　hhll
	A = A & [hhll,X]	AND　hhll,X
	A = A & [hhll,Y]	AND　hhll,Y
	A = A & [(zz,X)]	AND　(zz,X)
	A = A & [(zz),Y]	AND　(zz),Y
	A = A | [zz,X]	ORA　zz,X
	A = A | [hhll]	ORA　hhll
	A = A | [hhll,X]	ORA　hhll,X
	A = A | [hhll,Y]	ORA　hhll,Y
	A = A | [(zz,X)]	ORA　(zz,X)
	A = A | [(zz),Y]	ORA　(zz),Y

分类	结构化描述语言	汇编语言
逻辑运算指令	A = A ^ [zz,X]	EOR　zz,X
	A = A ^ [hhll]	EOR　hhll
	A = A ^ [hhll,X]	EOR　hhll,X
	A = A ^ [hhll,Y]	EOR　hhll,Y
	A = A ^ [(zz,X)]	EOR　(zz,X)
	A = A ^ [(zz),Y]	EOR　(zz),Y
循环／移位指令	[zz,X] = [zz,X] << imm	ASL　zz,X
	[hhll] = [hhll] << imm	ASL　hhll
	[hhll,X] = [hhll,X] << imm	ASL　hhll,X
	[zz,X] = [zz,X] >> imm	LSR　zz,X
	[hhll] = [hhll] >> imm	LSR　hhll
	[hhll,X] = [hhll,X] >> imm	LSR　hhll,X
	[zz,X] = [zz,X] << imm WITH_C	ROL　zz,X
	[hhll] = [hhll] << imm WITH_C	ROL　hhll
	[hhll,X] = [hhll,X] << imm WITH_C	ROL　hhll,X
	[zz,X] = [zz,X] >> imm WITH_C	ROR　zz,X
	[hhll] = [hhll] >> imm WITH_C	ROR　hhll
	[hhll,X] = [hhll,X] >> imm WITH_C	ROR　hhll,X

* 与结构化描述语言不对应的有：

MUL、DIV、BIT、TST、RRF、JSR、RTI、RTS、NOP、FST、SLW、WIT、STP、BRA、JMP

(D) 双项运算赋值语句的展开例

表 F.5　双项运算赋值语句的展开例

分类	结构化描述语言	汇编语言
循环／移位指令	[zz] = [zz] << 2	ASL　zz ASL　zz
	[zz] = [zz] << 2 WITH_C	ROL　zz ROL　zz
	[zz] = [zz] >> 2	LSR　zz LSR　zz
	[zz] = [zz] >> 2 WITH_C	ROR　zz ROR　zz
加减法指令	[zz] = [zz] + 4	LDA　zz CLC ADC　#4 STA　zz
	[zz] = [zz] + 4 WITH_C	LDA　zz ADC　#4 STA　zz
	[zz] = [zz] + [zz]	LDA　zz CLC ADC　zz STA　zz
	[zz] = [zz] - 4	LDA　zz SEC SBC　#4 STA　zz
	[zz] = [zz] - 4 WITH_C	LDA　zz SBC　#4 STA　zz

分类	结构化描述语言	汇编语言
减法指令	[zz] = [zz] - [zz]	LDA　zz SEC SBC　zz STA　zz
乘除法指令	[zz] = [zz] * 4	LDA　zz JSR　.mult_8 .BYTE 4 STA　zz
	[zz] = [zz] / 4	LDA　zz JSR　.div_8 .BYTE 4 STA　zz
	[zz] = [zz] % 4	LDA　zz JSR　.mod_8 .BYTE 4 STA　zz
逻辑运算指令	[zz] = [zz] & 4	LDA　zz AND　#4 STA　zz
	[zz] = [zz] | 4	LDA　zz ORA　#4 STA　zz
	[zz] = [zz] ^ 4	LDA　zz EOR　#4 STA　zz

* 关于.mult_8、.div_8 以及.mod_8 请参照 5.4 注意事项之 2。

F.3.2　条件表达式的展开例

• 表示 IF 语句、DO 语句、FOR 语句中的条件表达式的展开例。
另外，标号名对应以下的展开例。

- if ～（else） ～ endif

```
if   [ MEM ] == 2        ;       LDA     MEM
     JSR     out         ;       CMP     #2
endif                    ;       BNE     hhll
                         ;       JSR     out
                         ; hhll:
```

- do ～ while

```
do                       ; hhll:
     JSR     out         ;       JSR     out
while   [ MEM ] == 2     ;       LDA     MEM
                         ;       CMP     #2
                         ;       BEQ     hhll
```

- for ～ next

```
for   [ MEM ] == 2       ; hhll1:
     JSR     out         ;       LDA     MEM
next                     ;       CMP     #2
                         ;       BNE     hhll2
                         ;       JSR     out
                         ;       BRA     hhll1
                         ; hhll2:
```

表 F.6　标志条件表达式的展开例

结构化描述语言	汇编语言	长字转移
• if　[FLAG] == 1 • while　[FLAG] == 0 • for　[FLAG] == 1	BBC　FLAG,hhll	BBS　FLAG, .Z0 JMP　hhll .Z0:
• if　[FLAG] == 0 • while　[FLAG] == 1 • for　[FLAG] == 0	BBS　FLAG,hhll	BBC　FLAG, .Z0 JMP　hhll .Z0:
• if　C == 1 • while　C == 0 • for　C == 1	BCC　hhll	BCS　.Z0 JMP　hhll .Z0:
• if　C == 0 • while　C == 1 • for　C == 0	BCS　hhll	BCC　.Z0 JMP　hhll .Z0:
• if　Z == 1 • while　Z == 0 • for　Z == 1	BNE　hhll	BEQ　.Z0 JMP　hhll .Z0:
• if　Z == 0 • while　Z == 1 • for　Z == 0	BEQ　hhll	BNE　.Z0 JMP　hhll .Z0:
• if　N == 1 • while　N == 0 • for　N == 1	BPL　hhll	BMI　.Z0 JMP　hhll .Z0:
• if　N == 0 • while　N == 1 • for　N == 0	BMI　hhll	BPL　.Z0 JMP　hhll .Z0:
• if　V == 1 • while　V == 0 • for　V == 1	BVC　hhll	BVS　.Z0 JMP　hhll .Z0:
• if　V == 0 • while　V == 1 • for　V == 0	BVS　hhll	BVC　.Z0 JMP　hhll .Z0:

表 F.7　存储器条件表达式的展开例

分类	结构化描述语言	汇编语言	长字转移
IF 语句	If　[MEM] == 2	LDA　MEM CMP　#2 BNE　hhll	LDA　MEM CMP　#2 BEQ　.Z0 JMP　hhll .Z0:
	if　[MEM] != 2	LDA　MEM CMP　#2 BEQ　hhll	LDA　MEM CMP　#2 BNE　.Z0 JMP　hhll .Z0:
	If　[MEM] > 2	LDA　MEM CMP　#2 BEQ　hhll BCC　hhll	LDA　MEM CMP　#2 BEQ　.Z0 BCS　.Z1 .Z0: JMP　hhll .Z1:
	if　[MEM] >= 2	LDA　MEM CMP　#2 BCC　hhll	LDA　MEM CMP　#2 BCS　.Z0 JMP　hhll .Z0:
	If　[MEM] < 2	LDA　MEM CMP　#2 BCS　hhll	LDA　MEM CMP　#2 BCC　.Z0 JMP　hhll .Z0:
	if　[MEM] <= 2	LDA　MEM CMP　#2 BEQ　.Z0 BCS　hhll .Z0:	LDA　MEM CMP　#2 BEQ　.Z0 BCC　.Z0 JMP　hhll .Z0:

分类	结构化描述语言	汇编语言	长字转移
DO 语句	while [MEM] == 2	LDA MEM CMP #2 BEQ hhll	LDA MEM CMP #2 BNE .Z0 JMP hhll .Z0:
	while [MEM] != 2	LDA MEM CMP #2 BNE hhll	LDA MEM CMP #2 BEQ .Z0 JMP hhll .Z0:
	while [MEM] > 2	LDA MEM CMP #2 BEQ .Z0 BCS hhll .Z0:	LDA MEM CMP #2 BEQ .Z0 BCC .Z0 JMP hhll .Z0:
	while [MEM] >= 2	LDA MEM CMP #2 BCS hhll	LDA MEM CMP #2 BCC .Z0 JMP hhll .Z0:
	while [MEM] < 2	LDA MEM CMP #2 BCC hhll	LDA MEM CMP #2 BCS .Z0 JMP hhll .Z0:
	while [MEM] <= 2	LDA MEM CMP #2 BEQ hhll BCC hhll	LDA MEM CMP #2 BEQ .Z0 BCS .Z1 .Z0: JMP hhll .Z1:

分类	结构化描述语言	汇编语言	长字转移
FOR 语句	for　[MEM] == 2	LDA　MEM CMP　#2 BNE　hhll2	LDA　MEM CMP　#2 BEQ　.Z0 JMP　hhll2 .Z0:
	for　[MEM] != 2	LDA　MEM CMP　#2 BEQ　hhll2	LDA　MEM CMP　#2 BNE　.Z0 JMP　hhll2 .Z0:
	for　[MEM] > 2	LDA　MEM CMP　#2 BEQ　hhll2 BCC　hhll2	LDA　MEM CMP　#2 BEQ　.Z0 BCS　.Z1 .Z0: JMP　hhll2 .Z1:
	for　[MEM] >= 2	LDA　MEM CMP　#2 BCC　hhll2	LDA　MEM CMP　#2 BCS　.Z0 JMP　hhll2 .Z0:
	for　[MEM] < 2	LDA　MEM CMP　#2 BCS　hhll2	LDA　MEM CMP　#2 BCC　.Z0 JMP　hhll2 .Z0:
	for　[MEM] <= 2	LDA　MEM CMP　#2 BEQ　.Z0 BCS　hhll2 .Z0:	LDA　MEM CMP　#2 BEQ　.Z0 BCC　.Z1 .Z0: JMP　hhll2 .Z1:

表 F.8　寄存器条件表达式的展开例

分类	结构化描述语言	汇编语言	长字转移
A 寄存器	if　A == [MEM]	CMP　MEM BNE　hhll	CMP　MEM BEQ　.Z0 JMP　hhll .Z0:
	if　A != [MEM]	CMP　MEM BEQ　hhll	CMP　MEM BNE　.Z0 JMP　hhll .Z0:
	if　A > [MEM]	CMP　MEM BEQ　hhll BCC　hhll	CMP　MEM BEQ　.Z0 BCS　.Z1 .Z0: JMP　hhll .Z1:
	if　A >= [MEM]	CMP　MEM BCC　hhll	CMP　MEM BCS　.Z0 JMP　hhll .Z0:
	if　A < [MEM]	CMP　MEM BCS　hhll	CMP　MEM BCC　.Z0 JMP　hhll .Z0:
	if　A <= [MEM]	CMP　MEM BEQ　.Z0 BCS　hhll .Z0:	CMP　MEM BEQ　.Z0 BCC　.Z0 JMP　hhll .Z0:

分类	结构化描述语言	汇编语言	长字转移
X 寄存器	if　X == [MEM]	CPX　MEM BNE　hhll	CPX　MEM BEQ　.Z0 JMP　hhll .Z0:
	if　X != [MEM]	CPX　MEM BEQ　hhll	CPX　MEM BNE　.Z0 JMP　hhll .Z0:
	if　X > [MEM]	CPX　MEM BEQ　hhll BCC　hhll	CPX　MEM BEQ　.Z0 BCS　.Z1 .Z0: JMP　hhll .Z1:
	if　X >= [MEM]	CPX　MEM BCC　hhll	CPX　MEM BCS　.Z0 JMP　hhll .Z0:
	if　X < [MEM]	CPX　MEM BCS　hhll	CPX　MEM BCC　.Z0 JMP　hhll .Z0:
	if　X <= [MEM]	CPX　MEM BEQ　.Z0 BCS　hhll .Z0:	CPX　MEM BEQ　.Z0 BCC　.Z0 JMP　hhll .Z0:

分类	结构化描述语言	汇编语言	长字转移
Y 寄存器	if　Y == [MEM]	CPY　MEM BNE　hhll	CPY　MEM BEQ　.Z0 JMP　hhll .Z0:
	if　Y != [MEM]	CPY　MEM BEQ　hhll	CPY　MEM BNE　.Z0 JMP　hhll .Z0:
	if　Y > [MEM]	CPY　MEM BEQ　hhll BCC　hhll	CPY　MEM BEQ　.Z0 BCS　.Z1 .Z0: JMP　hhll .Z1:
	if　Y >= [MEM]	CPY　MEM BCC　hhll	CPY　MEM BCS　.Z0 JMP　hhll .Z0:
	If　Y < [MEM]	CPY　MEM BCS　hhll	CPY　MEM BCC　.Z0 JMP　hhll .Z0:
	If　Y <= [MEM]	CPY　MEM BEQ　.Z0 BCS　hhll .Z0:	CPY　MEM BEQ　.Z0 BCC　.Z0 JMP　hhll .Z0:

F.4　结构化指令的句法图

用句法图表示 SRA74 可使用的结构化指令的语法。
说明中使用的主要术语的意思如下所示：

- 变量

  将存储器变量、存储器位变量、寄存器变量、寄存器位变量、标志变量总称为变量。

- 存储器变量

  指在 740 族的各寻址方式中被参照的任意存储器或堆栈。用“[ ]”或“{ }”括起来记述。

[ZZ]	零页	[ZZ,X]	零页 X
[ZZ,Y]	零页 Y	[HHLL]	绝对
[HHLL,X]	绝对 X	[HHLL,Y]	绝对 Y
[(ZZ,X)]	零页间接 X	[(ZZ),Y]	零页间接 Y
[S]	堆栈		

- 存储器位变量

  指在 740 族的位寻址方式中被参照的任意位。用“[ ]”或“{ }”括起来记述。但是，在参照前需要通过.EQU 伪指令对该变量进行如下定义：

  例）

```
BITSYM      .EQU 1,1000H        位符号的定义
            if [ BITSYM ]       位符号的参照
                  :
            else
                  :
            endif
```

- 寄存器变量

  指 740 族的各种寄存器。不需要用“[ ]”或“{ }”而用各寄存器名记述。为此，SRA74 准备了以下名称的保留字。另外，由于不区分大小写字母，因此 A 和 a 都有效。

A	累加器	X	变址寄存器 X
Y	变址寄存器 Y	S	堆栈指针
P	处理器状态寄存器		

- 寄存器位变量

  指在 740 族的累加器位寻址方式中被参照的累加器内的任意位。为此，SRA74 准备了以下名称的保留字。另外，由于不区分大小写字母，因此 BIT_A0 和 bit_a0 都有效。

BIT_A0	累加器的位 0	BIT_A1	累加器的位 1
BIT_A2	累加器的位 2	BIT_A3	累加器的位 3
BIT_A4	累加器的位 4	BIT_A5	累加器的位 5
BIT_A6	累加器的位 6	BIT_A7	累加器的位 7

- 标志变量

  指 740 族的状态寄存器内的标志。为此，SRA74 准备了以下名称的保留字。另外，由于不区分大小写字母，因此 C 和 c 中的任何一个都有效。

C	进位标志	Z	零标志
I	中断禁止标志	D	10 进制模式标志
T	X 变址运算模式标志	V	溢出标志
N	负标志		

・WITH C

指带进位运算。为此，SRA74 准备了保留字。另外，由于不区分大小写字母，因此 WITH_C 和 with_c 都有效。

例）

[work] = [work] << 2 with_c　　包含进位将 work 的内容向左移位 2 次。

・EVER

指无限循环。为此，SRA74 准备了保留字。另外，由于不区分大小写字母，因此 EVER 和 ever 都有效。

例）

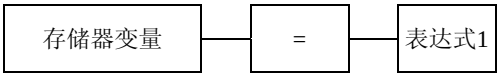
```
for ever          无限循环的指定
  :
next
```

・常数

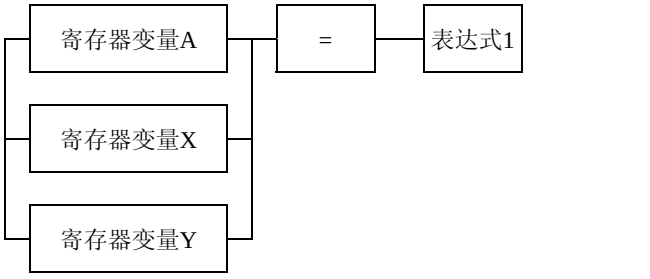
数值常数、字符常数、符号常数或者用运算符将它们组合的数统称为常数。

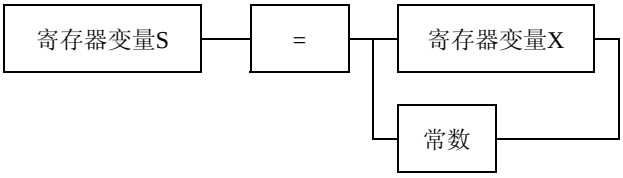
●赋值语句

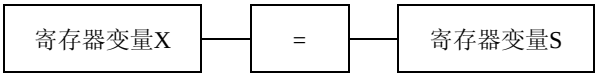
・存储器变量赋值语句

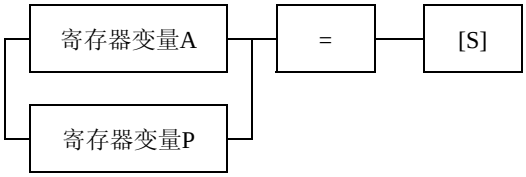


・寄存器变量赋值语句

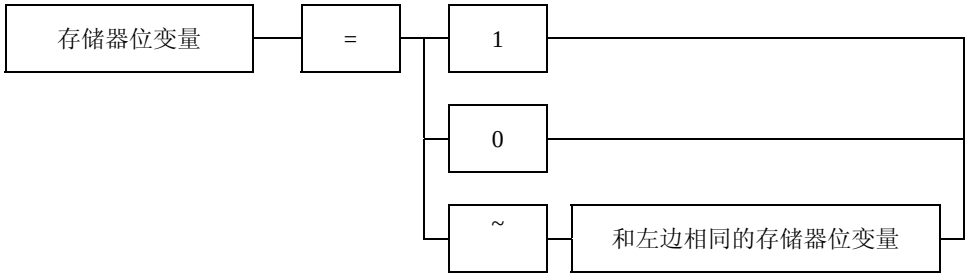




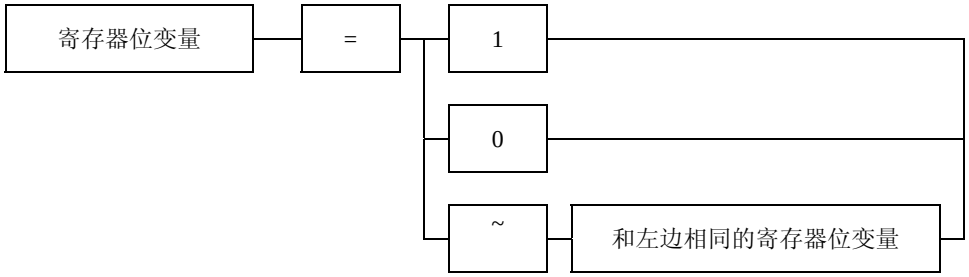




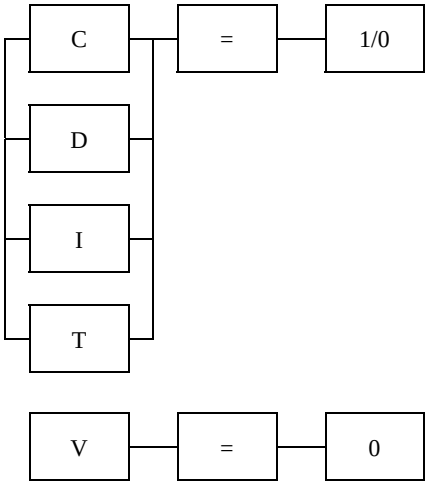
• 存储器位变量赋值语句



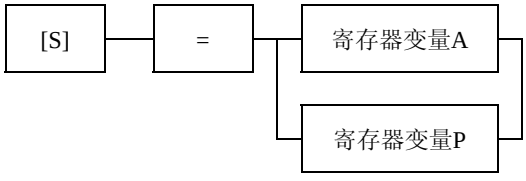
• 寄存器位变量赋值语句



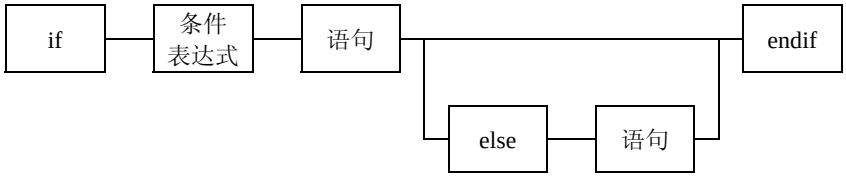
• 标志变量赋值语句



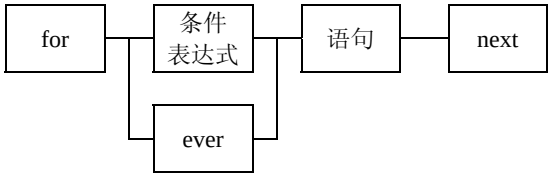
• 堆栈帧变量赋值语句



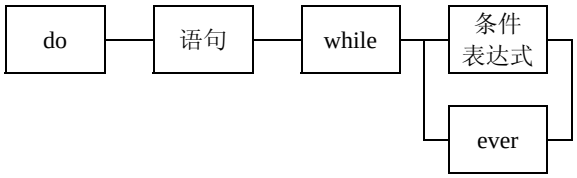
●if 语句



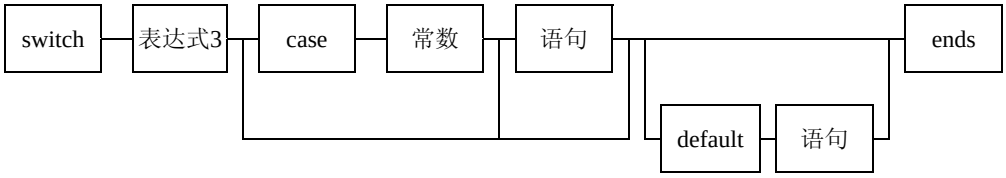
●for 语句



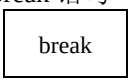
●do 语句



●switch 语句

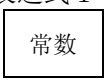


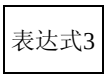
●break 语句



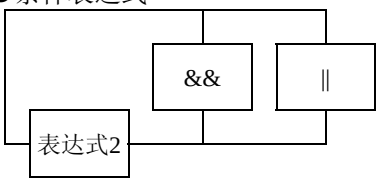
●continue 语句

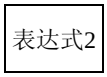
●表达式 1





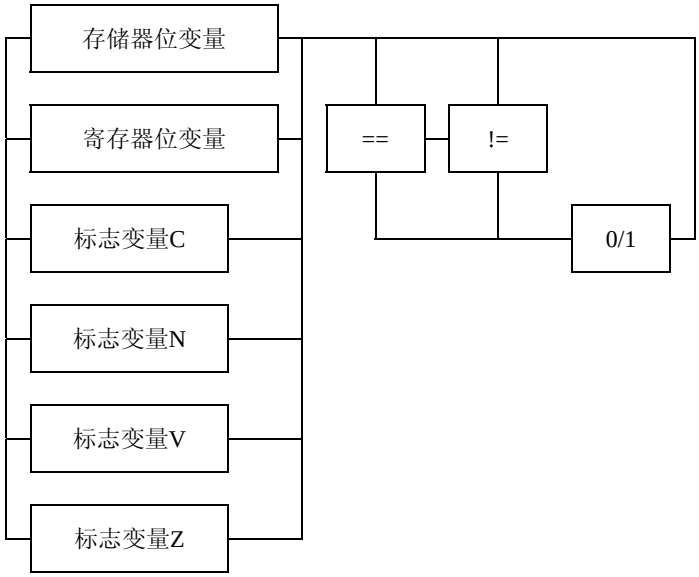
●条件表达式

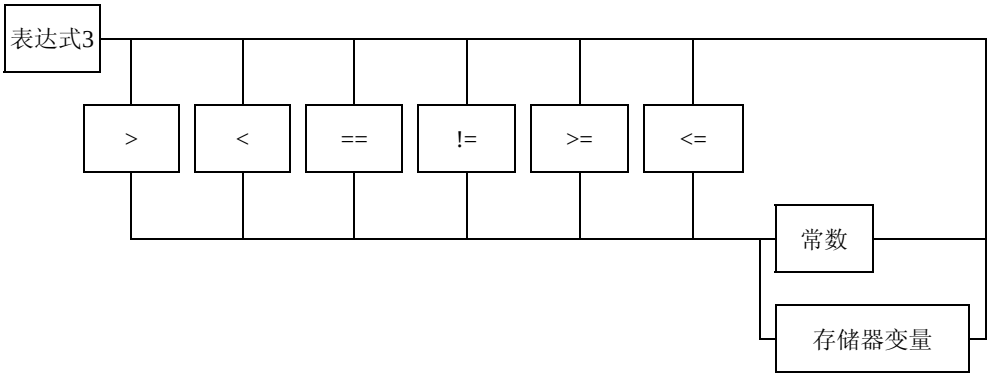




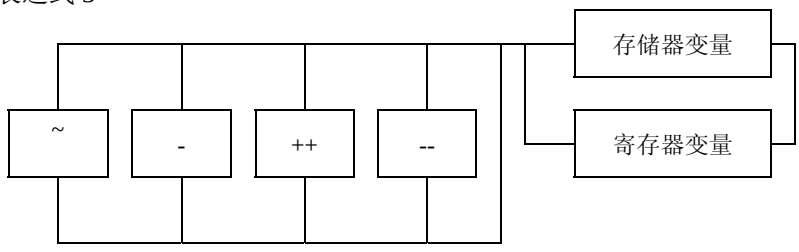
最多能连结6个

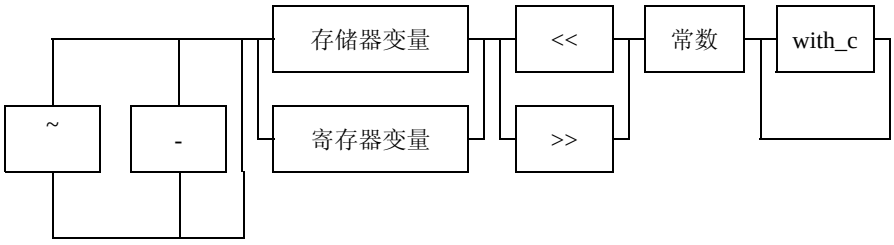
●表达式 2

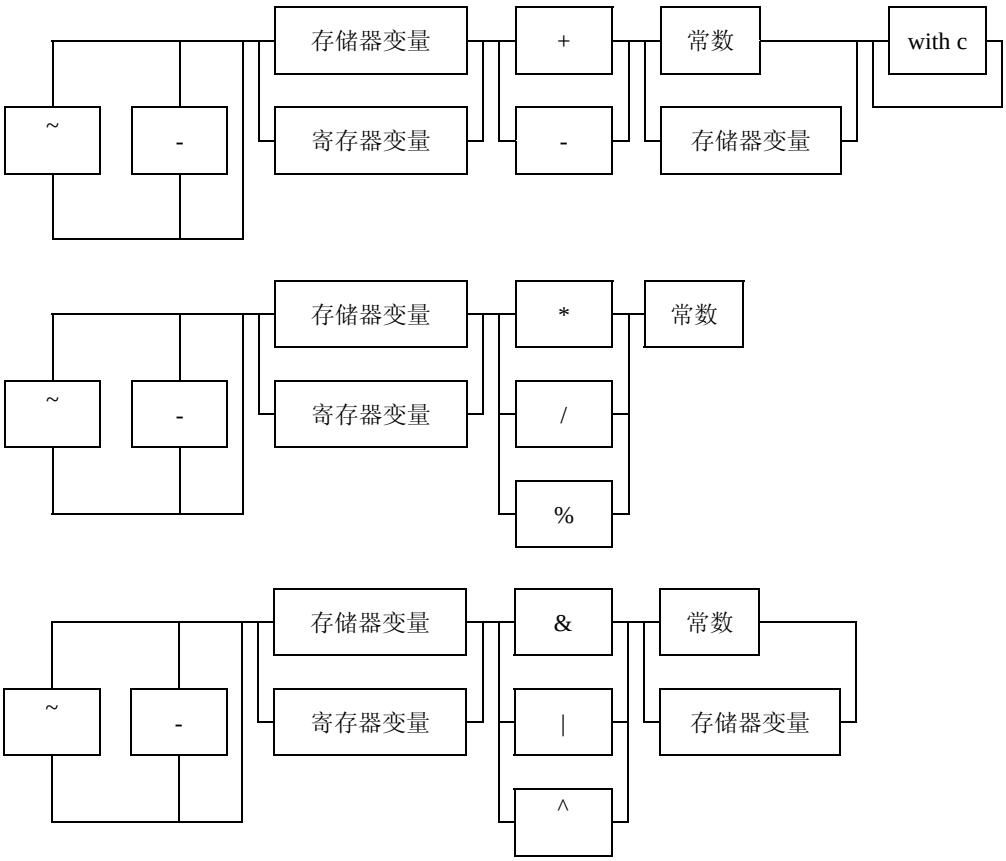

存储器位变量
寄存器位变量
标志变量C
标志变量N
标志变量V
标志变量Z
==
!=
0/1


表达式3
>
<
==
!=
>=
<=
常数
存储器变量

●表达式 3

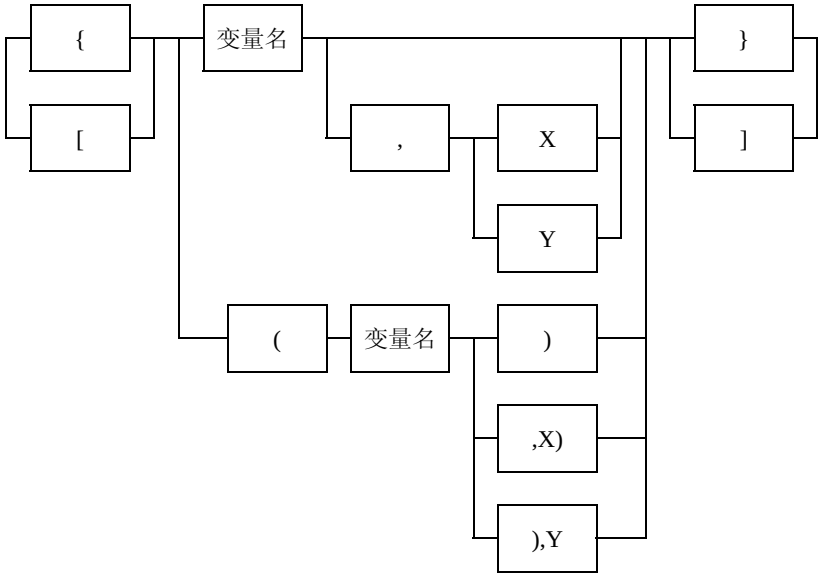

~
-
++
--
存储器变量
寄存器变量


~
-
存储器变量
寄存器变量
<<
>>
常数
with_c

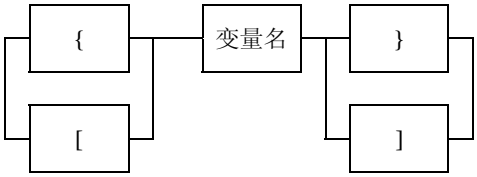

~
-
存储器变量
寄存器变量
+
-
常数
存储器变量
with c
~
-
存储器变量
寄存器变量
*
/
%
常数
~
-
存储器变量
寄存器变量
&
|
^
常数
存储器变量

●变量

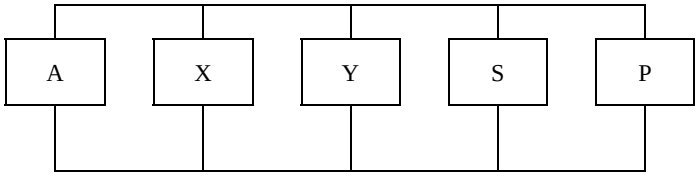
• 存储器变量


{
[
变量名
,
X
Y
}
]
(
变量名
)
,X)
),Y

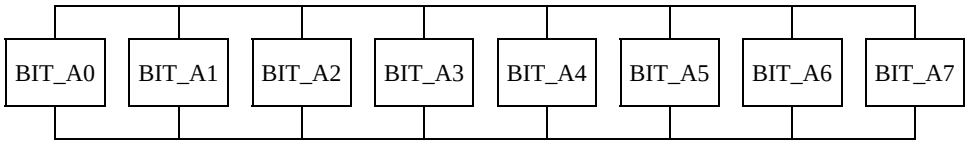
• 存储器位变量



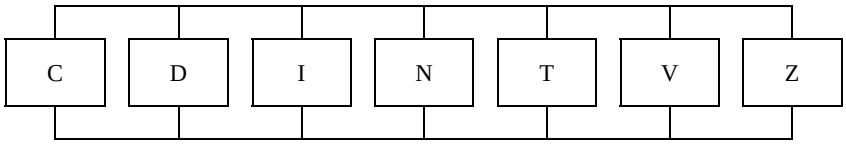
• 寄存器变量



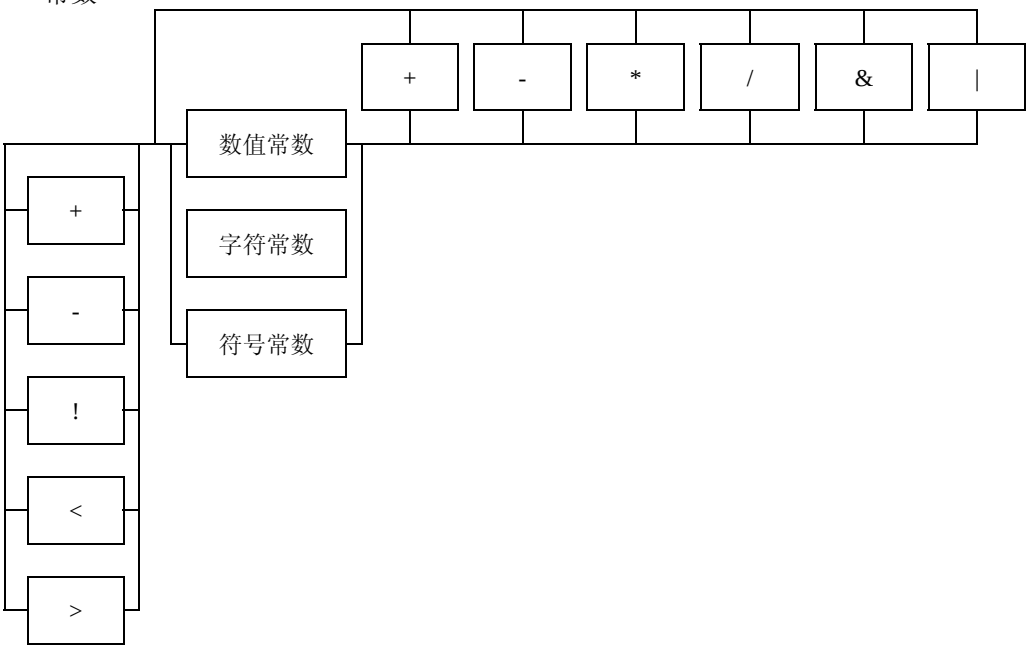
• 寄存器位变量



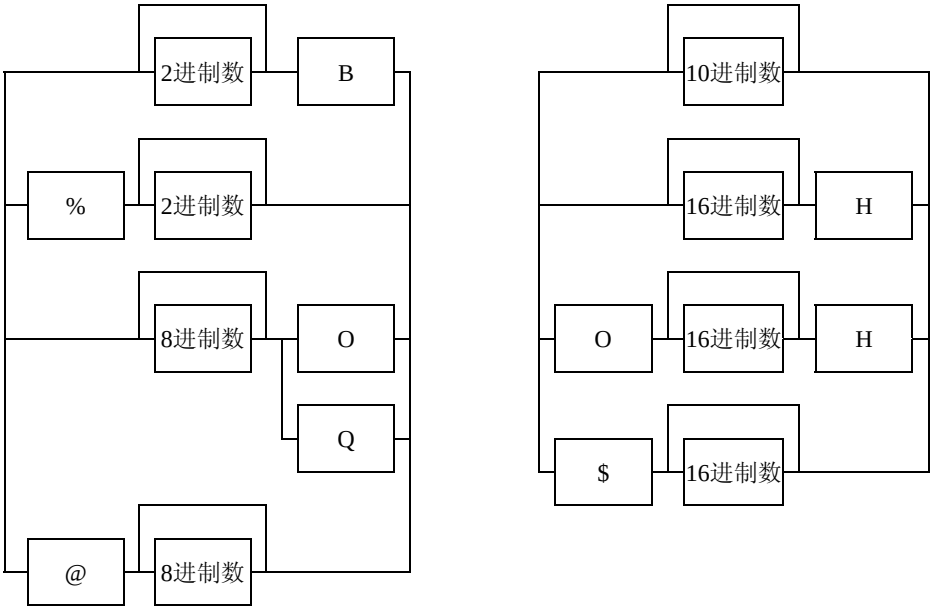
• 标志变量



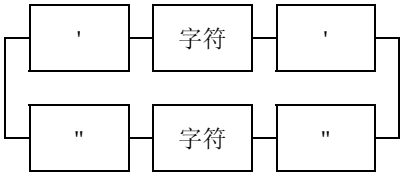
• 常数



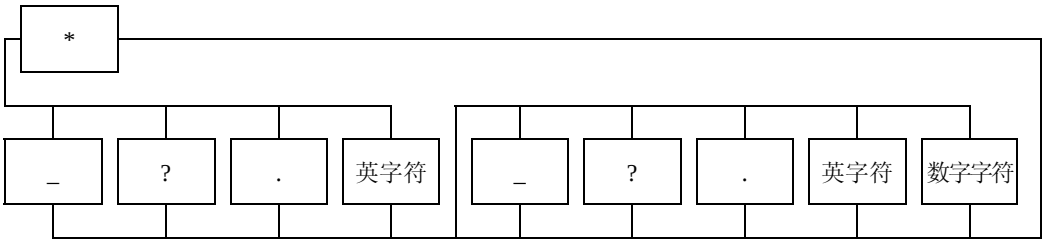
• 数值常数



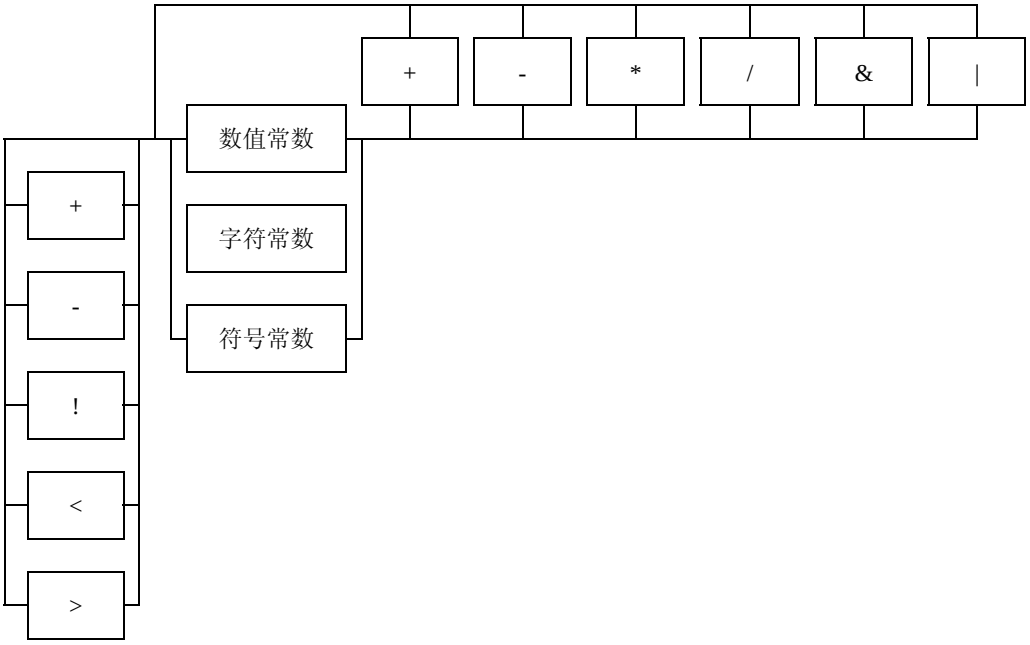
• 字符常数



• 符号常数



• 变量名



保留字一览

符号

..0～..65535　标号
.ASSERT　伪指令
.BEXT　伪指令
.BLKB　伪指令
.BYTE　伪指令
.COL　伪指令
.D0～.D65535　标号
.ELSE　伪指令
.END　伪指令
.ENDFUNC　伪指令
.ENDIF　伪指令
.ENDIO　保留伪指令
.ENDM　宏指令
.ENDPROC　保留伪指令
.ENDRAM　保留伪指令
.EQU　伪指令
.ERROR　伪指令
.EXITM　宏指令
.EXT　伪指令
.F0～.F65535　标号
.FUNC　伪指令
.I0～.I65535　标号
.IF　伪指令
.INCLUDE　伪指令
.IO　保留伪指令
.LIB　伪指令
.LINE　伪指令
.LIST　伪指令
.LISTM　伪指令
.LOCAL　宏指令
.MACRO　宏指令
.NLIST　伪指令
.NLISTM　伪指令
.OBJ　伪指令
.ORG　伪指令
.PAGE　伪指令
.PMOD　伪指令
.PROCINT　保留伪指令
.PROCMAIN　保留伪指令
.PROCSUB　保留伪指令
.PROGNAME　保留伪指令
.PUB　伪指令
.RAM　保留伪指令
.REPEAT　宏指令
.REPEATC　宏指令
.REPEATI　宏指令
.RMOD　伪指令
.S0～.S65535　标号
.SECTION　伪指令
.SEXT　伪指令
.SMOD　伪指令
.VER　伪指令
.WORD　伪指令
.ZBEXT　伪指令
.ZEXT　伪指令
.ZMOD　伪指令
??0～??65535　标号

A

A　累加器
ADC　助记符
AND　助记符
ASL　助记符

B

BBC　助记符
BBS　助记符
BCC　助记符
BCS　助记符
BEQ　助记符
BIT　助记符
BIT_A0　累加器的位0
BIT_A1　累加器的位1
BIT_A2　累加器的位2
BIT_A3　累加器的位3
BIT_A4　累加器的位4
BIT_A5　累加器的位5
BIT_A6　累加器的位6
BIT_A7　累加器的位7
BMI　助记符
BNE　助记符
BPL　助记符
BRA　助记符
BREAK　结构化指令
BRK　助记符
BVC　助记符
BVS　助记符

C

C　进位标志
CASE　结构化指令
CLB　助记符
CLC　助记符
CLD　助记符
CLI　助记符
CLT　助记符
CLV　助记符
CMP　助记符
COM　助记符
CONTINUE　结构化指令
CPX　助记符
CPY　助记符

D

D　10进制模式标志
DEC　助记符
DEX　助记符

第 2 部

用于 740 族的连接编辑程序

LINK74 操作手册

目　录

图 目 录

表 目 录

第1章　LINK74 操作手册的构成

LINK74 操作手册由以下章节构成：

- 第 2 章　概要
  说明 LINK74 的基本功能和 LINK74 生成的文件。
- 第 3 章　段的功能
  说明作为 LINK74 基本操作单位的段。
- 第 4 章　操作方法
  说明 LINK74 的命令输入方法。
- 附录 A　错误信息一览表
  对于 LINK74 显示的错误信息，用一览表表示其内容和对策。

注意事项

本手册所载的部分程序例子中，有一部分用“\”替代表示专用页寻址方式的“¥”符号。这是由于符号根据操作系统而不同，但是代码相同都能使用。

第2章　概要

LINK74 将 SRA74 生成的可再定位文件与库文件连接，生成 740 族机器语言数据文件。

2.1　功能

LINK74 能与 740 族的调试程序一起使用。另外，为了发挥这些功能，准备了以下功能：

1. 连接配置具有多个可再定位的文件中的同一段[*1] 名的区域。
2. 能指定段的配置顺序和各段单位的开始地址。
3. 能利用由 LIB74 建立的库文件。
4. 生成便于调试的映象文件。
5. 生成调试所需的符号文件。

2.2　生成文件

LINK74 生成以下 3 种文件：

1. 机器语言数据文件（以下称为 HEX 文件）
   - 以 intel16 进制格式输出。
   - 文件属性为.HEX。

2. 映象文件（以下称为 MAP 文件）
   - 表示各段最终配置的地址信息。
   - 将该文件输出到打印机，可用于调试和掌握各段的存储容量。
   - 在指定命令参数“-M”时输出 MAP 文件。
   - 文件属性为.MAP。

3. 符号文件（以下称为 SYM 文件）
   - 是含有调试所需的各种信息的文件。
   - 该文件不为列表格式，因此不能打印输出。
   - 在指定命令参数“-S”时输出 SYM 文件。
   - 文件属性为.SYM。

2.3　MAP 文件的结构

MAP 文件的输出例子如**图 2.1** 所示。MAP 文件表示以下信息：

1. 有关从哪个可再定位的文件连接多少数据的段单位的信息。输出以下信息：
   - ATR 部：表示相对属性或者绝对属性[*2]。REL 表示相对，ABS 表示绝对。

[*1] 段是指具有不同物理性质的要素单位（如构成程序的 ROM 区和 RAM 区）。详情内容请参照第 4 章。
[*2] 汇编语言源程序中，由伪指令.ORG（或者“*=”）指定的开始地址为绝对属性。

- TYPE 部：表示 RAM 区或者 ROM 区。
- START 部：表示开始地址。
- LENGTH 部：以字节数表示区域大小。
- 在连接库文件时，表示库文件名和可再定位文件名。可再定位文件名表示在“()”内。

2. 全局标号列表
   表示程序中的全局标号[*3]及其绝对地址。该部分只在指定命令参数“-MS”时才输出。

3. 全局符号列表
   表示程序中的全局符号[*4]及其绝对地址。该部分只在指定命令参数“-MS”时才输出。

4. 全局位符号列表
   表示程序中的全局位符号[*5]、位值以及绝对地址。该部分只在指定命令参数“-MS”时才输出。

```
SECTION         FILENAME        ATR.  TYPE  START   LENGTH

WORKRAM         MAIN.R74        ABS   RAM   0000    0080
                SUB.R74         REL   RAM   0080    0100
                UTIL.LIB        REL   RAM   0180    0008
                (CALC.R74)
PROM            MAIN.R74        REL   ROM   C000    1800
                SUB.R74         REL   ROM   D800    1500
                UTIL.LIB        REL   ROM   ED00    0820
                (CALC.R74)
DROM            MAIN.R74        REL   ROM   F520    0023
                SUB.R74         REL   ROM   F544    0030

GLOBAL LABEL INFORMATION

ADCNT           0030    COUNT       009C    DATA0       00A4
DATA1           00A6    MAIN        C000    TIME        00C6

GLOBAL SYMBOL INFORMATION

GLOBAL BIT SYMBOL INFORMATION
```

图 2.1　MAP 文件的输出例子

[*3] 指用伪指令.PUB 声明的标号。
[*4] 指用伪指令.PUB 声明的符号。
[*5] 指用伪指令.PUB 声明的位符号。

第3章　段的功能

3.1　段的作用

用汇编语言编写的程序一般由 RAM 区、程序区以及固定数据区构成。用 SRA74 汇编源文件时生成可再定位文件，但是可再定位文件含有 1 个以上这样的区域。段是表示这些区域的单位。以下用具体例子说明段的内容及其用法：

作为最简单的例子，设想 2 个可再定位文件 MAIN.R74 和 SUB.R74。如图 3.1 所示，各可再定位文件有 RAM 区、程序区、固定数据区。

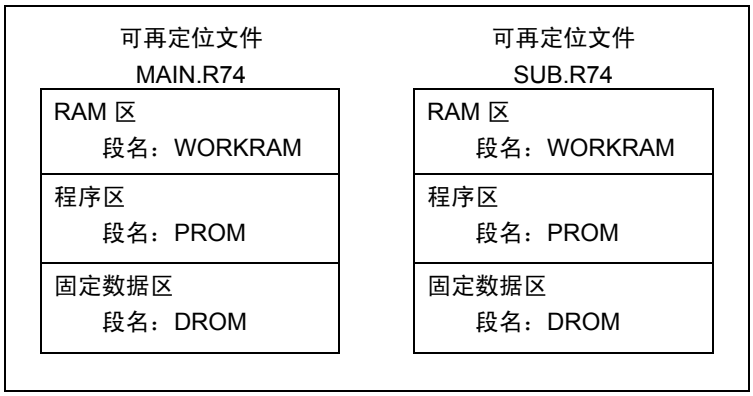


图 3.1　可再定位文件结构图

要将这些可再定位文件配置成如图 3.2 的存储空间时，预先通过汇编伪指令.SECTION 给要结合的段指定相同的名称。这样，在连接时各段就被配置为连续区域。在连接时 LINK74 能指定各段的开始地址。

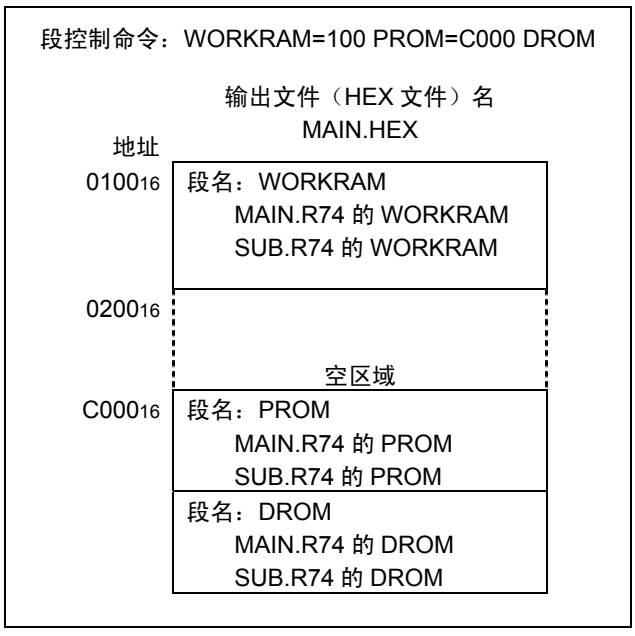


图 3.2　系统存储器映象

如上所述，如果使用 LINK74，就能通过连接时的命令来建立对应系统最终地址的机器语言数据。

3.2　段的属性

各段有地址配置相关的信息（能重新配置或者固定）和物理配置相关的信息（ROM 或者 RAM）。前者称为地址属性，后者称为物理属性。以下说明各属性的内容。

3.2.1　地址属性

地址属性有相对属性和绝对属性。根据在源程序的相应段中有无伪指令.ORG（或者“*=”）决定这些属性。各属性的特征如下：

1. 相对属性
   - 段内没有伪指令.ORG 时，该段为相对属性。相对属性的段是可重新配置的段，连接时能指定开始地址。
2. 绝对属性
   - 段内有伪指令.ORG 时，该段为绝对属性。绝对属性的段被配置为用伪指令.ORG 指定的固定地址。
   - 对绝对属性的段，连接时不能指定开始地址。

存在于多个可再定位文件中的同名段能具有不同的地址属性。

3.2.2　物理属性

物理属性有 ROM 属性和 RAM 属性。这些属性表示配置段的区域的物理性质。物理属性的特征如下：

1. ROM 属性
   - 汇编语言源程序中记述产生代码的语句（LDA 等指令或者伪指令.BYTE 等）的段为 ROM 属性。
   - ROM 属性的段生成连接的结果 HEX 文件。
   - 为了避免和地址属性混淆，该属性称为 ROM 类型。
2. RAM 属性
   - RAM 属性的段不生成连接的结果 HEX 文件。
   - 汇编语言源程序中记述区域确保伪指令.BLKB 的段为 RAM 属性。
   - 为了避免和地址属性混淆，该属性称为 RAM 类型。

同名的段必须都为相同的物理属性。

3.2.3　保留段

LINK74 将段 Z、S、P、R 作为保留段处理。以下说明各保留段：

1. Z 段

   Z 段为配置于 740 族零页（从 00_{16} 地址到 FF_{16} 地址）的 RAM 属性的段。用 SRA74 的伪指令.SECTION 声明段名‘Z’（也可以小写字母）时或者声明伪指令.ZMOD 时，该段变为 Z 段。

   Z 段为相对属性时，能通过输入 LINK74 命令指定起始地址。省略起始地址时，从 00_{16} 地址开始配置。

2. S 段
   S 段为配置于 740 族专用页（从 FF00$_{16}$ 地址到 FFFF$_{16}$ 地址）的 ROM 属性的段。用 SRA74 的伪指令.SECTION 声明段名‘S’（也可以小写字母）时或者声明伪指令.SMOD 时，该段变为 S 段。
   S 段为相对属性时，能通过输入 LINK74 命令指定起始地址。省略起始地址时，从 FF00$_{16}$ 地址开始配置。

3. P 段
   P 段为具有程序或者固定数据的 ROM 属性的段。能配置于程序区内（从 00$_{16}$ 地址到 FFFF$_{16}$ 地址）的任何区域。用 SRA74 的伪指令.SECTION 声明段名‘P’（也可以小写字母）时或者声明伪指令.PMOD 时，该段变为 P 段。
   P 段为相对属性时，如果省略指定命令就从 00$_{16}$ 地址开始配置。

4. R 段
   R 段为 RAM 属性的段。能配置于程序区内（从 00$_{16}$ 地址到 FFFF$_{16}$ 地址）的任何区域。用 SRA74 的伪指令.SECTION 声明段名‘R’（也可以小写字母）时或者声明伪指令.RMOD 时，该段变为 R 段。
   R 段为相对属性时，如果省略指定命令就从 00$_{16}$ 地址开始配置。

5. E 段
   不能将 E 段内定义的符号和段内的源程序行调试信息输出到可再定位文件。用 SRA74 的伪指令.SECTION 声明段名‘E’（也可以小写字母）时，该段变为 E 段。E 段必须预先用伪指令.ORG 指定段的配置地址。另外，执行 SRA74 时必须指定-BANK 选项来汇编。不指定命令选项时，在 E 段的定义行输出警告。

3.3　段的基本功能

1. 连接时连续配置同名段。同名段之间不配置其它名称的段。
2. 同名段中的配置顺序按照连接命令中的可再定位文件名的指定顺序。

第4章　操作方法

4.1　启动方法

为了执行 LINK74，必须输入以下信息（输入参数）：

1. 可再定位文件名（必需项目）
2. 库文件名
3. 段控制信息
4. 命令参数

作为输入这些信息的方法，LINK74 准备了以下 3 种适应操作环境的方式：

1. 对话式输入
2. 命令行输入
3. 命令文件输入

这些输入方式使用相同的输入参数。另外，无论使用哪种方式都能执行相同的命令。4.2 节说明该输入参数，4.3 节举例说明各输入方式。

4.2　输入参数

4.2.1　可再定位文件名

1. 必须输入可再定位文件名。
2. 可再定位文件只处理具有属性.R74 的文件。命令输入时，能省略文件属性。
3. 文件名能指定目录路径。在只记述文件名时，处理当前驱动器的当前目录中的文件。
4. 最初指定的可再定位文件名为输出文件名。用命令参数“-F”指定输出文件名时，命令参数“-F”优先。
5. 将输出文件输出到最初指定的可再定位文件的目录。用命令参数“-O”指定目录时，命令参数“-O”优先。

4.2.2　库文件名

1. 能省略库文件名。
2. 库文件只处理具有属性.LIB 的文件。命令输入时，能省略文件属性。
3. 文件名能指定目录路径。在只指定文件名时，处理当前目录的文件。
4. 只在可再定位文件中有不能解决的全局标号或者全局符号时，才参照库文件。

4.2.3　段控制

1. 能省略段信息。此时，按照在读取的可再定位文件中所遇到的段顺序来配置。
2. 必须对相对属性的段指定段信息。绝对属性的段与有无段指定无关，从用伪指令.ORG 指定的固定地址开始配置。

3. 段配置的指定必须从配置成低位地址的段开始顺序用空格将段名隔开记述。
4. 必须在该段名后的‘=’（等于）之后记述各段的开始地址。地址必须用 16 进制数指定。此时，不需要起始的‘0’和最后的‘H’。
5. 段名区分大小写字母。
6. 省略相对属性的段的开始地址时，从 0000$_{16}$ 地址开始配置。另外，保留段 S 从 FF00$_{16}$ 开始配置。
7. 段的地址重叠会产生错误。
   但是，如果指定命令参数“-A”就能允许绝对地址重叠。由此，不需要对 SFR 区等绝对地址标号进行外部参照指定，而能通过伪指令“.INCLUDE”读取多个程序文件。

4.2.4　命令参数

命令参数控制连接的输出文件和版本确认等。命令参数的内容如**表 4.1** 所示。

表 4.1　命令参数一览表

命令参数	内　容
-A	允许同名的绝对属性段的重叠。能用于共享全局存储区。
-BANK	地址空间的上限从 FFFFH 扩展到 1FFFFH。
-F	指定输出文件名。指定格式如下： -Fsample
-M	输出 MAP 文件（只限于段信息）。
-MS	以带全局标号、全局位符号以及全局符号列表输出 MAP 文件。
-N	忽视在源文件中由伪指令.OBJ、.LIB 指定的.R74 和.LIB 文件的参照指定信息。
-O	指定输出文件的目录。指定格式如下： -OC:¥USRY¥WORK 将 C 驱动器的¥USR¥WORK 指定成输出目录。
-S	输出 SYM 文件。
-V	确认可再定位文件间的版本匹配性。但是，必须在汇编语言源程序中用伪指令.VER 指定版本。

4.3　输入方法

4.3.1　对话式输入

对话式输入的特征如下：

1. 是按照可再定位文件、库文件、段控制命令、命令参数的顺序以对话式输入的方式。
2. 是便于可再定位文件和段的个数少的情况或者要尝试设定地址的情况等方式。
3. 在命令行输入或者命令文件输入，如果命令个数不足，就自动转移到该方式。

在操作系统的提示符状态下，通过输入 LINK74<RET>启动对话式输入。LINK74 启动后输出如下的画面：

```
A>LINK74 <RET>
740 Family LINKER V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

Relocatable files (.R74) >>
```

图 4.1　LINK74 启动画面

图 4.1 的最后一行表示等待可再定位文件的输入。请在“>>”后输入连接对象的可再定位文件。由于对话式输入依次等待库文件名、段控制、命令参数的输入，因此必须如图 4.2 所示进行输入（输入多个文件时，各文件名必须用空格或者制表符隔开。有关库文件名的输入、段信息、命令参数也相同）。

```
A>LINK74 <RET>
740 Family LINKER V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

Relocatable files (.R74) >> MAIN SUB<RET>
Libraries          (.LIB ) >> UTIL1 UTIL2<RET>
Section information      >> WORKRAM=100 PROM=C000 DROM<RET>
Command parameter   >> -O¥WORK -M -S<RET>
```

图 4.2　对话式输入时的画面

4.3.2　命令行输入

命令行输入的特征如下：

1. 是在操作系统的命令输入状态下输入全部命令的方式。
2. 由于命令行的字符数取决于 OS，因此能用于可再定位文件或段的个数少的情况。
3. 能用于通过批文件和 make 文件记述执行命令的情况。
4. 4 种输入参数的信息必须用‘,’（逗号）隔开输入。在命令行输入和图 4.2 相同命令的例子如图 4.3 所示。
5. 不需要库文件名后的参数时，也必须输入逗号。在图 4.3 的例子中不需要库文件的情况如图 4.4 所示。
6. 只在省略命令参数的特殊情况下，为了明确没有命令参数，需要 2 个逗号。在图 4.4 中省略命令参数的例子如图 4.5 所示。
7. 在输入参数的个数不足时，自动转移到对话式输入状态。

```
A>LINK74 MAIN SUB, UTIL1 UTIL2, WORKRAM=100 PROM=C000 DROM, -O¥WORK -M -S<RET>
```

图 4.3　命令行输入例子 1

```
A>LINK74 MAIN SUB,, WORKRAM=100 PROM=C000 DROM, -O¥WORK -M -S<RET>
```

图 4.4　命令行输入例子 2（省略库文件名）

```
A>LINK74 MAIN SUB,, WORKRAM=100 PROM=C000 DROM,,<RET>
```

图 4.5　命令行输入例子 3（省略命令参数）

4.3.3　命令文件输入

命令文件输入的特征如下：

1. 是预先通过编辑程序将连接命令建立成命令文件并在启动时指定该文件名的方式。
2. 是便于因命令字符数多而不能使用命令行输入时的方式。
3. 从操作系统启动 LINK74 时，如**图 4.6** 所示，在文件名前加上‘@’指定命令文件名。在图 4.6 的情况下，文件名 CMD.DAT 的内容作为命令被执行。
4. 命令文件所记述的内容与命令行输入相同（但是，不需要指示 LINK74 启动的 LINK74 部分）。由于忽视换行码，因此能将长命令分成多行记述。用命令文件记述**图 4.5** 内容的例子如**图 4.7** 所示。
5. 命令参数不够时，LINK74 转移到对话式输入状态。例如，在没有**图 4.7** 的最后行的第 2 个逗号时，LINK74 变为命令参数对话式输入画面。

```
A>LINK74 @CMD.DAT<RET>
```

图 4.6　指定命令文件

```
MAIN SUB
,
,WORKRAM=100 PROM=C000 DROM
,,
```

图 4.7　命令文件记述例子

4.4　错误

4.4.1　错误种类

LINK74 执行时发生的错误由以下原因引起：

1. 与操作系统有关的错误
   是磁盘或者存储容量不足等与执行 LINK74 的操作系统环境有关的错误。请在参照“**附录 A 错误信息一览表**”的基础上，用操作系统的命令对应。

2. 与 LINK74 命令行输入有关的错误
   是与启动 LINK74 时的命令行输入有关的错误。请在确认本章内容的基础上，重新输入命令。

3. 与连接对象的可再定位文件内容有关的错误
   是与全局标号的双重定义、外部标号的未定义等可再定位文件内容有关的错误。请确认该处的源程序等，如果有必要请从汇编开始重新执行。

4. 与 LINK74 功能有关的错误
   是由于与 SRA74、LIB74 的版本不匹配等而引起的错误。如果需要更详细的内容，请与本公司联系。

LINK74 以图 4.8 的格式在画面上显示错误内容。请在参照附录 A 的按错误序号顺序的错误一览表的基础上进行处理。

```
A>LINK74 MAIN SUB,,WORKRAM=100 PROM=C000 DROM,,
740 Family LINKER V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

now processing pass 1
processing "MAIN.R74"
ERROR NO.2: Out of heap space

A>
```

图 4.8　错误显示例子

4.4.2　给操作系统的返回值

在用操作系统的批文件等记述执行命令时，有可能要根据执行结果改变处理内容的情况。LINK74 将执行结果分为表 4.2 的 5 个错误级返回给操作系统。

表 4.2　错误级

错误级	执行结果的内容
0	正常结束
1	与连接对象的可再定位文件内容有关的错误
2	与 LINK74 的命令输入有关的错误
3	与操作系统有关的错误
4	与 LINK74 的功能有关的错误

4.5　环境变量

在读取库文件时，LINK74 参照环境变量 LIB74。通过将库文件所在的目录路径名设定给环境变量 LIB74，能在命令输入时省略指定路径名。

环境变量设定方法的详情内容请参照 OS 手册。

附录 A 错误信息一览表

表 A.1 错误信息一览表

错误序号	错误信息	错误内容和对策
0	xxx file not found	没有找到被输入指定的文件。输入指定也包括由伪指令.OBJ 或者.LIB 指定的内容。 → 请正确输入文件名。
1	Invalid command input	输入命令的参数个数在 5 个以上或者命令的输入参数的字符数在 2048 个以上。 → 请在上述范围内重新输入。
2	Out of heap space	连接程序运行所需的存储容量不足。 → 请减少公共符号的个数。
3	Invalid section information	段信息的指定错误。 → 请用“段名=地址”的格式重新输入。
4	Invalid parameter input "xxx"	命令参数的指定错误。 → 请正确输入。
5	Non relocatable file name	没有输入可再定位文件名。 → 请输入文件名。
6	Internal error	发生了 LINK74 内部错误。 → 请与 LINK74 经销商联系。
7	xxx relocatable format is mismatch	.R74 文件的可再定位格式版本不同。 → 该错误是使用中的汇编程序或者库生成程序与连接程序不匹配时发生的错误。请用和 LINK74 相同版本的 SRA74 或者 LIB74 建立连接对象.R74 及.LIB 文件。
8	Program version is different	用伪指令.VER 声明的程序版本不一致。 → 请使在连接对象的可再定位文件中由伪指令.VER 指定的版本一致，或者解除命令参数“-V”的指定。
9	Unresolved label "xxx" in xxx	在该段中没有定义被外部参照声明的标号或者符号。 → 必须连接对该标号或者符号进行公共声明的程序。
10	"xxx" is multiple defined in xxx. others in xxx	该标号或者符号为双重定义。 → 请更改该标号或者符号名。
11	Location overlap. SECTION=xxx ADDRESS=xxx in xxx	该段的地址空间重叠。 → 请确认该段的地址配置，注意不能使地址重叠（在绝对属性的段的情况下，请更改源文件中的伪指令.ORG）。
12	SECTION xxx is an absolute	对于绝对属性的段，用段控制命令指定了起始地址。 → 请不要用命令输入指定地址或者将该段更改为相对属性。
14	Can't find SECTION xxx	该段不存在。 → 请正确指定段信息（请注意：段名区分大小写字母）。
15	Can't create xxx	不能生成该文件。 → 请确认命令参数“-O”的指定，然后重新输入。

错误序号	错误信息	错误内容和对策
16	File seek error xxx	不能寻找该文件。 → 这是与操作系统有关的错误。一般认为大多是由于磁盘装置的硬件出现异常而引起的错误。
17	Expression value is out of range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	该位置的运算结果超过了限制值（作为错误位置的信息，显示段名、绝对地址和从段头开始的偏移量）。 → 请更改程序，使其不超过限制值。
18	Out of disk space	输出文件所需的磁盘容量不足。 → 请扩大磁盘容量。
19	Relative jump is out of range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	不能到达该位置的相对转移目标的存取对象。 → 请更改程序，使转移目标的标号在范围之内（作为错误位置的信息，显示段名、绝对地址和从段头开始的偏移量）。
22	Out of maximum program size	程序区超出了 64K 字节（$FFFF_{16}$）的最大区域。 → 请减小程序容量。
23	Section type mismatch in SECTION xxx	在该段中 ROM 类型和 RAM 类型混在一起。 → 同一段必须统一为 ROM 类型或者 RAM 类型。
25	Expression is out of ZERO page. SECTION=xxx ADDRESS=xxx OFFSET=xxx	由零页寻址处理的计算表达式的结果超出了 00_{16} 到 FF_{16} 的范围。 → 请更改使计算结果在零页范围内。
26	Expression is out of SPECIAL page. SECTION=xxx ADDRESS=xxx OFFSET=xxx	由专用页寻址处理的计算表达式的结果超出了 $FF00_{16}$ 到 $FFFF_{16}$ 的范围。 → 请更改使计算结果在专用页范围内。
27	label "xxx" type is mismatch.	外部参照的标号类型（标号或者位符号）与声明的类型不同。 → 请正确声明外部参照的伪指令。
28	Section "xxx" information is out of range.	由命令输入指定的段的开始地址在范围外。 → 请正确指定开始地址。
29	Bit value is out of range. SECTION=xxx ADDRESS=xxx OFFSET=xxx	位值的计算表达式的结果超出了 0 到 7 的范围。 → 请更改使计算结果在范围之内。
30	Extended area requires command option '-BANK'	定义了扩展区域。 → 请指定命令选项‘-BANK’进行连接。

第 3 部

用于 740 族的库生成程序

LIB74 操作手册

目　录

图 目 录

表 目 录

第1章　LIB74 操作手册的构成

LIB74 操作手册由以下章节构成：

- 第 2 章 概要
  说明 LIB74 的基本功能和 LIB74 生成的文件。
- 第 3 章 操作方法
  说明 LIB74 的命令输入方法。
- 附录 A 错误信息一览表
  对于 LIB74 显示的错误信息，用一览表表示其内容和对策。

第2章　概要

LIB74 是以程序库形式管理 SRA74 生成的可再定位文件的软件。能通过将常用的子程序转换成程序库来缩短汇编时间及进行程序的再利用。

2.1　功能

LIB74 能处理 SRA74 生成的可再定位文件。另外，LIB74 生成的文件能被 LINK74[*1] 参照。为了充分发挥这些功能，准备了以下功能：

1. 建立和修正能被 LINK74 参照的库文件。
2. 将可再定位文件注册到库文件。
3. 删除库文件中不需要的可再定位文件。
4. 将库文件中的旧的可再定位文件更新为新建的可再定位文件。
5. 抽出注册在库文件中的可再定位文件。
6. 显示库文件中的可再定位文件信息。

2.2　特点

1. 实现连接处理的高速化。
   通过将可再定位文件转换成程序库，在连接时能快速取出需要的信息，使 LINK74 的连接处理高速化。

2. 在更新库文件中的可再定位文件时，通过比较文件的更新日期，只更新最新版本的可再定位文件（指定命令参数“-U”时）。

2.3　生成文件

LIB74 生成以下 4 种文件：

1. 库文件
   - 是编辑 SRA74 生成的可再定位文件并附加标号和符号的索引部分的文件。
   - 在库文件中，将可再定位文件作为 1 个模块来管理（以下将库文件中可再定位文件称为模块）。
   - 模块名与含有文件属性的可再定位文件名相同。
   - 文件属性为.LIB。

2. 列表文件（以下称为 LST 文件）
   - 在指定命令参数“-L”时生成此文件。
   - 表示库文件中可再定位文件名、全局标号以及符号等的一览表。
   - 文件属性为.LST。
   - 在下一节说明该文件的结构。

[*1] 是用于 740 族的连接编辑程序的程序名。

3. 可再定位文件
   - 在指定命令参数“-X”时生成此文件。
   - 是重建库文件中的可再定位文件的文件。
   - 重建的可再定位文件是与 SRA74 生成的程序库被注册前的可再定位文件相同。
   - 文件属性为.R74。

4. 备份文件
   - 此文件的生成时与命令参数的内容无关。但是，在 LIB74 异常结束时不生成此文件。
   - 在更改库文件时，将更改前的库文件保留为备份文件。
   - 文件属性为.BAK。

注意事项

由于列表文件以外的文件是二进制格式，因此请不要输出到打印机及画面。

2.4　LST 文件的结构

LST 文件的输出例子如图 2.1、图 2.2、图 2.3 所示。LST 文件表示以下信息：

1. 模块名一览表（图 2.1）

   输出注册在库文件中的以下信息：
   - Module name 部分：表示注册在库文件中的模块名。输出顺序为注册在库文件中的顺序。
   - Offset 部分：表示从库文件起始位置到模块起始位置的字节数（以 16 进制表示）。
   - Module size 部分：表示模块的存储容量（以 16 进制表示）。

```
LIB74 librarian V.4.00.00                    date 1992-Dec-10 15:30 page 1

Library file name :          CALC8.LIB
Relocatable format:          VER.A
Last update time :           1992-Dec-10 15:30
Number of module :           3
Number of global symbol:     10

Module_name:
key_scan ..... Offset:   00000000H Module size: 00000100H
multiply ..... Offset:   00000180H Module size: 00000080H
division ..... Offset:   00000200H Module size: 000000A5H
```

图 2.1　LST 文件的输出例子 1（模块名一览表）

2. 全局标号和符号一览表（按字母顺序）（图 2.2）

   在此，输出以下 2 个表：
   - 公共标号列表（PUBLIC symbol table）

     Symbol_name 部分：表示公共标号、公共符号或者公共位符号。给用伪指令.EQU 声明的公共符号附加“(e)”后输出。给公共位符号附加“(b)”后输出。

     Module_name 部分：表示含有公共标号、公共符号或者公共位符号的模块名。
   - 外部参照标号列表（EXTERN symbol table）

     Symbol_name 部分：表示外部标号或者外部符号。

     Module_name 部分：表示进行外部参照指定的模块名。

```
LIB74 librarian V.4.00.00                    date 1989-Jun-30 15:30 page 2

PUBLIC symbol table (symbol count = 0010)

  Symbol_name       Module_name       Symbol_name       Module_name

  _dividel.....  division        _division.....  division
  _key_flg1(b)..  key_scan       _key_flg2(b)..  key_scan
  _key_scan.....  key_scan       _key_scn1.....  key_scan
  _multi1.......  multiply       _multiply.....  multiply
  _one(e).......  key_scan       _two(e).......  key_scan

LIB74 librarian V.4.00.00                    date 1992-Dec-10 15:30 page 3

EXTERN symbol table (symbol count = 0010)

  Symbol_name       Module_name       Symbol_name       Module_name

  _div_ansl.....  division        _div_ansh..... division
  _key_buff1....  key_scan       _key_buff2.....  key_scan
  _key_buff3....  key_scan       _key_buff4.....  key_scan
  _mul_ansl.....  multiply       _mul_ansh.....  multiply
```

图 2.2　LST 文件的输出例子 2（全局标号和符号一览表）

3. 按全局标号和符号的模块分类的一览表（图 2.3）
   在此，输出 PUBLIC symbol table 和 EXTERN symbol table 的 2 个表。任何一个表都在表示模块名的后面列举其模块内的全局标号和全局符号。

```
LIB74 librarian V.4.00.00                    date 1992-Dec-10 15:30 page 4

 PUBLIC symbol table

 Module name: key_scan (symbol count = 0006)

 _key_flg1(b)     _key_flg2(b)     _key_scan     _key_scn1
 _one(e)          _two(e)

 Module name: multiply (symbol count = 0002)

 _multi1          _multiply

 Module name: division (symbol count = 0002)

 _divide1         _division


LIB74 librarian V.4.00.00                    date 1992-Dec-10 15:30 page 5

 EXTERN symbol table

 Module name: key_scan (symbol count = 0006)

 _key_buff1       _key_buff2       _key_buff3     _key_buff4
 _linecnty        _linecntx

 Module name: multiply (symbol count = 0002)

 _mul_ansl        _mul_ansh

 Module name: division (symbol count = 0002)

 _div_ansl        _div_ansh
```

图 2.3　LST 文件的输出例子 3（按全局标号和符号的模块分类的一览表）

第3章　操作方法

3.1　启动方法

为了执行 LIB74，必须输入以下信息：

1. 库文件名（必需项目）
2. 可再定位文件名
3. 命令参数

作为输入这些信息的方法，LIB74 准备了以下 2 种方式：

1. 命令行输入
2. 命令文件输入

LIB74 在任何一种输入方式中都能使用相同的输入参数以及执行相同的命令。在下一节中说明各输入参数的功能，最后举例说明各输入方式。

3.2　输入参数

3.2.1　库文件名

1. 必须输入库文件名。
2. 必须在命令参数“-O”之后加上空格或者制表符，然后输入编辑对象的库文件名。
3. 文件名能指定目录路径名。没有指定路径时，首先参照当前目录，然后参照由环境变量“LIB74”指定的目录路径。没有设定环境变量时，处理当前目录的文件。
4. 能省略文件属性.LIB。

3.2.2　可再定位文件名

1. 能用空格或者制表符隔开指定多个可再定位文件名。
2. 必须在命令参数“-F”之后用空格或者制表符隔开输入处理对象的可再定位文件名。
3. 文件名能指定目录路径名。没有指定路径时，处理当前驱动器的当前目录中的文件。
4. 能省略文件属性.R74。

3.2.3　命令参数

命令参数控制库文件操作内容的指定、输出文件的指定等。命令参数一览表如**表 3.1** 所示。

表 3.1　命令参数一览表

序号*1	命令参数*2	内　容
1	-O (Output)	指定编辑对象的库文件名。
2	-F (Files)	指定对库文件追加、更新或者抽出的可再定位文件名，或者指定从库文件删除的模块名。
3	-A (Append)	对库文件追加可再定位文件。
	-R (Replace)	更新库文件中的模块。
	-D (Delete)	从库文件中删除指定的模块。
	-L (List)	输出与注册在库文件中的模块相关的一览表。
	-X (eXtract)	从库文件中抽出指定的模块。抽出的模块作为能被 LINK74 处理的可再定位文件生成到当前目录。
4	-V (Verbose)	在画面上显示处理中的文件名。
	-U (Update)	在更新库文件中的模块时，只更新与最新的可再定位文件对应的模块。

注意事项

*1. 序号表示以下内容：

1：是必需项目。必须指定编辑对象的库文件名。

2：在指定命令参数“-A”、“-R”或者“-X”中的任何一个来追加、更新或者抽出可再定位文件的情况下，必须指定操作对象的可再定位文件名。在删除模块时，必须指定删除对象的模块名。
如果同时指定多个对象，就出错并中止处理。

3：只能且必须指定 5 个中的一个。

4：请根据需要指定。

*2. 不区分命令参数的大小写字母，因此“-A”、“-a”中的任何一个都有效。

3.2.4　命令参数的详细说明

命令参数的详情说明如下：

1. -O
   - 指定编辑对象的库文件名。
   - 库文件名能指定目录路径名。
     例)　A>LIB74 -LO B:¥USR¥TEST<RET>
   - 省略目录路径时，首先参照当前目录，然后参照由环境变量“LIB74”指定的目录路径。在环境变量进行如下设定时，处理 B 驱动器的¥USR¥LIB 目录中的文件。
     例)　A>SET LIB74=B:¥USR¥LIB
   - 在省略目录路径并且也没有设定环境变量时，处理当前驱动器的当前目录中的文件。
   - 省略文件属性时，作为默认值选择属性.LIB。
   - 也能通过用全称记述文件来指定“.LIB”以外的属性文件。

2. -F
   - 指定对库文件追加、更新、抽出的可再定位文件名或者删除的模块名。
   - 能用空格或者制表符隔开指定多个文件名。
   - 文件名能指定目录路径名。
     例)　A>LIB74 -AO TEST.LIB -F B:¥WORK¥SUB1 C:SUB2<RET>
   - 省略目录路径时，处理当前驱动器的当前目录中的文件。
   - 省略文件属性时，作为默认值选择属性.R74。
   - 也能通过用全称记述文件来指定“.R74”以外的属性文件。

3. -A
   - 新建库文件或者对库文件追加可再定位文件。
   - 新建库文件时，按照指定的顺序从库文件的起始位置开始注册由“-F”指定的可再定位文件。
   - 追加可再定位文件时，按照指定的顺序将由“-F”指定的可再定位文件追加到库文件的最后。
   - 指定的可再定位文件名和已注册的模块名不进行重复检查。
   - 如果同时指定同一模块名，就出现标号或者符号的双重定义错误并中止处理。

4. -R
   - 更新库文件中的模块。
   - 如果和“-U”一起使用，就将可再定位文件的更新日期和库文件内的模块的更新日期进行比较，只更新新的模块。

5. -D
   - 从库文件删除指定的模块。

6. -L
   - 输出与注册在库文件中的模块相关的一览表（LST 文件）。
   - 用“-F”指定文件名时，将指定的可再定位文件的信息输出到 LST 文件。
   - 没有用“-F”指定文件名时，将有关库文件中的所有模块的信息输出到 LST 文件。
   - 文件属性为.LST。

7. -X
   - 从库文件中抽出指定的模块，成为给库文件注册前的可再定位文件的状态。
   - 可再定位文件的更新日期为抽出时的日期。
   - 抽出的可再定位文件能被 LINK74 处理。
   - 抽出的可再定位文件被生成到当前目录。
   - 没有用“-F”指定可再定位文件名时，抽出库文件中的所有模块。
   - 库文件内容不发生变化。
   - 文件属性为.R74。

8. -V
   - 在画面上显示 LIB74 正在处理的文件。

9. -U
   - 更新库文件中的模块时（指定“-R”时），只更新最新的文件。
   - 更新库文件中的模块时，将由“-F”指定的可再定位文件的更新日期和注册在库文件中的模块的更新日期比较，在由“-F”指定的可再定位文件为新的情况下，就进行更新处理。
   - 可再定位文件的更新日期取决于操作系统的文件管理信息。

3.3　输入方法

3.3.1　命令行输入

命令行输入的特征如下：

1. 是在操作系统的命令输入状态下输入全部命令的方式。

2. 能用于指定的文件数或者命令参数的个数少的情况。
3. 能用于通过批文件或者 make 文件记述执行命令的情况。

从库文件 TEST.LIB 删除模块 FILE1.R74 的例子如图 3.1 所示。将可再定位文件 FILE2.R74、FILE3.R74 追加到库文件 TEST.LIB 的例子如图 3.2 所示。

```
A>LIB74 -D -O TEST.LIB -F FILE1<RET>
```

图 3.1　命令行输入例子 1（删除模块）

```
A>LIB74 -AO TEST.LIB -F FILE2 FILE3<RET>
```

图 3.2　命令行输入例子 2（追加可再定位文件）

3.3.2　命令文件输入

命令文件输入的特征如下：

1. 是预先通过编辑程序将 LIB74 的处理内容建立为命令文件并在启动时指定该文件名的方式。
2. 是便于因指定的文件名或者命令的字符多而不能使用命令行输入时的方式。
3. 在启动 LIB74 时，如图 3.3 所示，用‘@’符号指定命令文件名。图 3.3 是执行文件 CMD.DAT 内容的情况。
4. 命令文件所记述的内容和命令行输入相同（但是，不需要指示 LIB74 启动的 LIB74 部分）。由于换行码被认为是空格，因此能将长命令分成多行记述。如果用命令文件记述图 3.2 的内容，就能按图 3.4 那样记述。

如果正确输入命令，LIB74 就开始处理。如果 LIB74 的各命令处理结束，画面上就显示结束信息，并结束程序处理。LIB74 正常结束处理时的画面显示例子如图 3.5 所示。

```
A>LIB74 @CMD.DAT<RET>
```

图 3.3　命令文件的指定

```
-AO
TEST.LIB
-F
FILE2
FILE3
```

图 3.4　命令文件记述例子

```
A>LIB74 -AVO TEST.LIB -F SUB_1 SUB_100<RET>
740 Family LIBRARY MANAGER V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED
< test.lib > Create
APPEND FILE = sub_1,sub_100
MODULE COUNT              000002
GLOBAL SYMBOL COUNT   000019
A>
```

图 3.5　LIB74 正常结束时的画面显示例子

3.4　错误

3.4.1　错误的种类

LIB74 执行时发生的错误由以下原因引起：

1. 与操作系统有关的错误
   是磁盘或者存储容量不足等与执行 LIB74 的操作系统环境有关的错误。请在参照“附录 A 错误信息一览表”的基础上，用操作系统的命令对应。

2. 与 LIB74 命令输入有关的错误
   是与启动 LIB74 时的命令输入有关的错误。命令输入有误时，画面上显示如图 3.6 所示的帮助画面。请在确认本章内容的基础上，重新输入命令。

3. 与处理对象的可再定位文件内容有关的错误
   是与公共标号被双重定义等可再定位文件内容有关的错误。请参照 LST 文件来修正该位置内容。

```
A>LIB74<RET>
740 Family LIBRARY MANAGER V.4.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

  Usage: A>LIB74 -[ardxluv]o <filename> [-f <filename> ...]

        -o :  library file name
        -f :  relocatable file names
        -a :  append command
        -r :  replace command
        -d :  delete command
        -x :  extract command
        -l :  listout command
        -u :  update check (option)
        -v :  verbose (option)
```

图 3.6　命令行错误时的帮助画面

如果在 LIB74 运行中发生错误，画面上就显示错误信息。请在参照“附录 A 错误信息一览表”的基础上进行处理。

3.4.2　给操作系统的返回值

在用批文件等记述执行命令时，有可能要根据执行结果改变处理内容的情况。LIB74 将执行结果分为表 3.2 所示的 4 个错误级返回给操作系统。

表 3.2　错误级

错误级	执行结果的内容
0	正常结束
1	与处理对象的库文件或者可再定位文件有关的错误
2	与 LIB74 的命令输入有关的错误
3	与操作系统有关的错误

3.5　环境变量

LIB74 总是优先处理当前目录内的库文件。

在当前目录内不存在库文件时，并且在给环境变量“LIB74”设定了目录路径时，处理环境变量设定的目录内的库文件。

例)　A>SET LIB74=B:¥USR¥LIB<RET>

附录 A　错误信息一览表

A.1　　系统错误一览表

如果在 LIB74 运行中检测到系统错误，画面上就显示错误信息，并且中止处理。系统错误一览表如**表** A.1 所示。

表 A.1　系统错误一览表

错误信息	错误内容和对策
Usage:A>LIB74 -[adluvxz]o <filename> [-f <filename>...]	命令输入错误。 → 请参照帮助画面，重新输入命令。
Can't open xxx	没有找到该文件。 → 请确认由命令参数“-O”或“-F”指定的文件是否在指定的目录中。
Can't create xxx	不能建立该文件。 → 请确认命令参数“-O”的指定，重新输入。
Out of disk space*1	输出文件所需要的磁盘容量不足。 → 请扩大磁盘容量。
Input file read error xxx	在读取该文件期间发生了错误。 → 这是与操作系统有关的错误。一般认为大多是由于磁盘装置的硬件出现异常而引起的错误。
Internal error	发生了 LIB74 的内部错误。 → 请与 LIB74 经销商联系。
File seek error xxx	不能找到该文件。 → 这是与操作系统有关的错误。一般认为大多是由于磁盘装置的硬件出现异常而引起的错误。

注意事项

*1. 在执行 LIB74 时，由于在执行过程中建立中间文件，大约需要库文件的 2～3 倍的磁盘空间。

A.2　　库生成程序错误一览表

如果在 LIB74 运行中检测到处理错误，画面上就显示错误信息，并且中止处理。库生成程序错误一览表如**表** A.2 所示。

表 A.2　库生成程序错误一览表

错误信息	错误内容和对策
xxx is a multiple defined in xxx. others in xxx	双重定义了公共标号或者公共符号。 → 请用 LST 文件确认库文件内的公共标号或者符号。
xxx module is not in the library	库文件中不存在该模块。 → 请用 LST 文件确认库文件中的模块名。
Invalid module or library	库文件中的模块和指定的可再定位文件的格式不同。 → 请用相同版本的 SRA74 来建立编辑对象的库文件和可再定位文件。
xxx command file not found	不存在该命令文件。 → 请确认指定的命令文件。
Out of heap space	运行库生成程序所需的存储容量不足。 → 请减少全局标号的个数。
Too many object module	库文件中的模块数过多。 → 请将库文件分开。一个库文件中的模块数最多为 500 个。
CPU number error	指定的库文件或者可再定位文件不是由 SRA74 生成的文件。 → 请确认指定的库文件或者可再定位文件。

第 4 部

用于 740 族的交叉参考程序

CRF74 操作手册

目　录

图 目 录

表 目 录

第1章　CRF74 操作手册的构成

CRF74 操作手册由以下章节构成：

- 第 2 章 概要
  说明 CRF74 的基本功能和 CRF74 生成的文件。
- 第 3 章 操作方法
  说明 CRF74 的命令输入方法。
- 附录 A 错误信息一览表
  对于 CRF74 显示的错误信息，用一览表表示其内容和对策。

第2章　概要

CRF74 生成源文件中的标号和符号的相互参照列表（称为交叉参考列表）。如果使用此列表，就能简单地掌握在程序修正时源文件各部分之间的依赖关系。

2.1　功能

CRF74 能与 SRA74[*1] 一起使用。CRF74 能通过以下功能有效地掌握源文件：

1. 将标号的参照指令种类显示为参照行序号。
2. 能通过伪指令.INCLUDE 进行文件读取。
3. 能通过伪指令.PAGE 进行标头输出。

2.2　生成文件

CRF74 生成交叉参考列表（以下称为 CRF 文件）。

- 表示标号和符号名的相互参照列表。
- 每行的列数固定成 80 列，每页的行数固定成 57 行。
- 能将该文件输出到打印机，用于调试和编辑。
- 文件属性为.CRF。

2.3　CRF 文件的结构

CRF 文件的输出例子如**图 2.1** 所示。CRF 文件表示以下信息：

1. 标号和符号名及其参照行和定义行。
   对定义行附加‘#’表示，对参照行中的子程序调用附加‘&’表示。

2. 显示标号和符号名的前 32 个字符。另外，CRF 文件按最长的名称进行格式化。

3. 在列表的标头显示由伪指令.PAGE 指定的标题（但是，只显示前 30 个字符，超过的字符不显示）。

4. CRF74 不对源程序中的标号和符号值进行判断。因此，必须注意不能进行带条件汇编的判断。

[*1] 是用于 740 族的可再定位汇编程序名。

```
740 Family CROSS REFERENCE V.1.00.10                              P. 001

A0              3926    4285    8549    9079    9100
AA              3884    5545    5668
ABEND           9396&   9465&   9587#
ABEND10         9588    9593#
ABENDRT         9590#   9605
ACCHK           1201#   1408
ACCHK5          1213    1237#
ACCHKE          1235    1239    1241    1249#
ADDING          4994    5006#
ADDING0         5013    5014    5016#
ADDING1         5015    5019#
ADDRESS          300     318    1025
ADR_CHK         9154#
ADR_OUT         8302    8336    9145#
ADR_PNT         8157#
ADR_PNT2        8149#
```

図 2.1　CRF 文件的例子

第3章　操作方法

3.1　　启动方法

为了执行 CRF74，必须输入以下信息（输入参数）：

1. 源文件名（必需项目）
2. 命令参数

3.2　　输入参数

3.2.1　　源文件名

1. 必须输入源文件名。
2. 省略文件属性（.A74）时，作为默认值选择属性.A74。
3. 也能通过用全称指定文件名来处理.A74 以外的属性（例如.ASM）文件。
4. 文件名能指定驱动器名。只记述文件名时，处理当前驱动器中的文件。
5. 源文件名能最多指定 16 个。

3.2.2　　命令参数

命令参数指定源文件中有无检测伪指令.INCLUDE 以及指定输出文件的驱动器名。命令参数的内容如表 3.1 所示。

表 3.1　命令参数一览表

命令参数	内容
-O	指定输出 CRF 文件的驱动器名和目录路径。指定格式如下： 例) A>CRF74 SRCFILE -OC:\TMP<RET> （‘¥’符号用‘\’表示。这是根据操作系统而不同，但是代码相同都能使用） 将 CRF 文件输出到 C 驱动器的 TMP 目录。
-I	忽视伪指令.INCLUDE。

3.3　　输入方法

3.3.1　　命令行输入

CRF74 通过在操作系统的提示符状态下输入命令行来启动。启动命令的输入例子如图 3.1 所示。如果在命令行输入期间检测到错误，就显示如图 3.2 所示的帮助画面，并且中止处理。

```
A>CRF74 SRCFILE1 SRCFILE2 SRCFILE3<RET>
```

图 3.1　命令行输入例子

```
A>CRF74<RET>
740 Family CROSS REFERENCE V.1.00.10
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

Usage: crf74 <filename> [-ifilename,..] [-opath]
        -i : not include specified files ( use -ifilename,.... )
        -o : select drive and directory for output ( use -otmp )
```

图 3.2　命令行错误时的帮助画面

3.4　错误

3.4.1　错误的种类

CRF74 执行时发生的错误由以下原因引起：

1. 与操作系统有关的错误
   是磁盘和存储容量不足等与执行 CRF74 的操作系统环境有关的错误。请在参照“**附录 A 错误信息一览表**”的基础上，用操作系统的命令对应。

2. 与 CRF74 的命令行输入有关的错误
   是与启动 CRF74 时的命令行输入有关的错误。请在确认本章内容的基础上，重新输入命令。

3. 与处理对象的源文件内容有关的错误
   在由伪指令.INCLUDE 指定的源文件不存在时发生此错误。

如果检测到错误，CRF74 就在画面上以**图 3.3** 的格式显示错误内容。请在参照附录 A 的错误一览表的基础上进行处理。

```
A>CRF74 SRCFILE1<RET>
740 Family CROSS REFERENCE V.1.00.10
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

now processing pass 1
now making cross reference ( SRCFILE1.A74 )
-----+
Out of heap space

A>
```

图 3.3　错误显示例子

3.4.2　给操作系统的返回值

在用操作系统的批文件等记述执行命令时，有可能要根据执行结果改变处理内容的情况。CRF74 将执行结果分为如**表 3.2** 所示的 3 个错误级返回给操作系统。

表 3.2　错误级

错误级	执行结果的内容
0	正常结束
1	在由伪指令.INCLUDE 指定的源文件不存在时发生的错误
2	与 CRF74 的命令输入有关的错误
3	与操作系统有关的错误

3.5　环境变量

CRF74 没有使用操作系统的环境变量。

附录 A　错误信息一览表

A.1　　系统错误一览表

如果在 CRF74 运行中检测到系统错误，画面上就显示错误信息，并且中止处理。系统错误一览表如表 A.1 所示。

表 A.1　系统错误一览表

错误信息	错误内容和对策
Usage: crf74 <filename> [-ifilename] [-opath]	命令输入错误。 → 请参照帮助画面，重新输入命令。
Can't open xxx	没有找到该文件。 → 请确认源文件名，重新输入。
Can't create xxx	不能生成该文件。 → 请扩大磁盘容量。
Out of disk space	输出文件所需要的磁盘容量不足。 → 请扩大磁盘容量。
Out of heap space	交叉参照程序运行所需的存储容量不足。*1 → 请减少符号或者标号个数。

注意事项

1. CRF74 处理的符号和标号的总数取决于执行 CRF74 的系统所能使用的存储容量。

A.2　　交叉参考错误一览表

如果检测到与建立交叉参考有关的错误，就在画面上输出错误信息，处理继续进行。交叉参考错误及其内容如表 A.2 所示。

表 A.2　交叉参考错误一览表

错误信息	错误内容和对策
Can't open include file xxx	不存在由伪指令.INCLUDE 指定的源文件。 → 请确认目录内容。

第 5 部

用于 M37280 的文件转换工具

CV74 操作手册

目　录

图 目 录

表 目 录

第1章　CV74 操作手册的构成

CV74 操作手册由以下章节构成：

- 第 2 章 概要
  说明 CV74 的基本功能和 CV74 生成的文件。
- 第 3 章 操作方法
  说明 CV74 的命令输入方法。
- 附录 A 错误信息一览表
  对于 CV74 显示的错误信息，用一览表表示其内容和对策。

第2章　概要

CV74 将由对应 M37280 的汇编系统（SRA74 V.3.00 以上）生成的 HEX 文件和符号文件转换为调试程序能利用的数据格式。

可利用于使用调试程序对 M37280 的具有超过 64KB 地址空间数据的程序进行调试时。

在使用 CV74 之前，必须事先用 LINK74 建立 HEX 文件和符号文件。

2.1　功能

CV74 能将由对应 M37280 的汇编系统（SRA74 V.3.00 以上）生成的 HEX 文件和符号文件转换为调试程序能利用的数据格式。

1. 将 HEX 文件分成超出 64KB 的地址空间数据和 64KB 以内的地址空间数据。
2. 将符号文件转换为调试程序能调试的格式。

注）由 CV74 转换前的 HEX 文件和符号文件不能正常调试。

2.2　生成文件

CV74 中生成以下 3 种文件：

1. 保存 64KB 以内的地址空间数据的 HEX 文件
   - 是调试程序能调试的 HEX 文件。
   - 是从 LINK74 生成的 HEX 文件中抽出并保存 64KB 以内的地址空间数据的文件。
   - 扩展名为.HEX。

2. 保存超出 64KB 的地址空间数据的 HEX 文件
   - 是从 LINK74 生成的 HEX 文件中抽出并保存超出 64KB 的地址空间数据的文件。
   - 扩展名为.HXH。

3. 调试程序能调试的符号文件
   - 扩展名为.SYM。

第3章　操作方法

3.1　启动方法

为了启动 CV74，必须输入以下信息（输入参数）：

1. 转换对象文件名
2. 命令参数

3.2　输入参数

3.2.1　转换对象文件名

1. 必须输入转换对象文件名。
2. 必须将转换对象 HEX 文件和符号文件准备在同一目录。
3. 不能指定多个文件名。
4. 包括相对目录，不能指定含有句点的文件名。

3.2.2　命令参数

命令参数指定输出文件名和转换对象文件。命令参数的内容指定如**表 3.1** 所示。

表 3.1　命令参数一览表

命令参数	内容
-O	指定输出文件名。 必须指定此选项。
-H	只转换 HEX 文件。 （不能和-S 选项同时指定。）
-S	只转换符号文件。 （不能和-H 选项同时指定。）

另外，在没有指定-H 或-S 时，进行 HEX 和符号文件的转换。

注意事项

在使用调试程序进行调试时，必须下载 CV74 生成的 HEX 和符号文件。

3.3　输入方法

CV74 通过在操作系统的提示符状态下输入命令来启动。

将由 LINK74 生成的 HEX 文件（master.hex）和符号文件（master.sym）转换为调试程序能调试的 HEX 文件（change.hex）和符号文件（change.sym）的例子如**图 3.1** 所示。

```
A> CV74 master –Ochange <RET>
HEX, SYM file converter for 7200 Series V.1.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED


== HEX file convert now == ( master.hex --> change.hex )
== SYM file convert now == ( master.sym --> change.sym )
```

图 3.1　CV74 的启动例子

3.4　错误

3.4.1　错误的种类

CV74 执行时发生的错误由以下原因引起：

1. 与操作系统有关的错误
   是磁盘和存储容量不足等与执行 CV74 的操作系统环境有关的错误。请在参照“**附录 A 错误信息一览表**”的基础上，用操作系统的命令对应。

2. 与 CV74 的命令行输入有关的错误
   是与启动 CV74 时的命令行输入有关的错误。请在确认本章内容基础上，重新输入命令。

3. 与处理对象的源文件内容有关的错误
   是与处理对象的 HEX 文件和符号文件内容有关的错误。必须在确认文件内容的基础上，重新输入。

```
A> CV74 <RET>
HEX, SYM file converter for 7200 Series V.1.00.00
COPYRIGHT(C) 1989-1998 RENESAS TECHNOLOGY CORPORATION
ALL RIGHTS RESERVED AND RENESAS SOLUTIONS CORPORATION
ALL RIGHTS RESERVED

  Error : No input files specified

    Usage: cv74 Input-filename -Ooutput-filename [-S or -H]<cr>
```

图 3.2　命令行错误时的帮助画面

如果在 CV74 运行中发生错误，画面上就显示错误信息。请在参照“**附录 A 错误信息一览表**”的基础上进行处理。

3.4.2　给操作系统的返回值

在用 OS 的批文件等记述执行命令时，有可能要根据执行结果改变处理内容的情况。CV74 将执行结果分为如表 3.2 所示的 5 个错误级返回给操作系统。关于错误级的利用方法，请参照市场出售的 OS 等的说明书。

表 3.2　错误级

错误级	执行结果的内容
0	正常结束
1	根据^C（Ctrl + C）输入的强制结束
2	与操作系统有关的错误
3	与处理对象的源文件内容有关的错误
4	与 CV74 的命令输入有关的错误

附录 A　错误信息一览表

表 A.1　错误信息一览表

错误信息	错误内容和对策
'-H ' and '-S ' are used	同时指定了-H 选项和-S 选项。 → 请指定其中某一选项。
. is specified in the filename	指定的文件名中有句点。 → 请确认文件名。
Can't create 'xx' file	不能生成'xx'文件。 → 请确认目录容量。
Can't create temporary file	不能生成临时文件。 → 为了在当前目录以外建立临时文件，请给环境变量'TMP'指定目录。
Can't open 'xx' file	不能打开'xx'文件。 → 请确认文件名。
Command line is too long	命令行的字符数过多。 → 请重新输入命令。
File format error	输入文件的格式不正确。 → 请确认文件格式。
Hex files number exceed 1	指定了多个转换对象文件。 → 只能指定一个转换对象文件。
Invalid option 'xx' is used	使用了无效的命令选项'xx'。 → 不存在指定的选项。请重新输入命令。
No input files specified	没有指定输入文件。 → 请指定输入文件。
No output files specified	没有指定输出文件名。 → 请指定输出文件名。
Not enough memory	存储容量不足。 → 请将文件分开后重新执行，或者请扩大存储容量。
Option 'xx' is multiple defined	双重定义了命令选项'xx'。 → 请重新输入命令。
Option 'xx' is not appropriate	命令选项'xx'的记述不正确。 → 请重新指定命令选项。

表 A.2　警告信息

警告信息	内容和对策
Output data to the 'xx' file doesn't exist	由于指定区域的数据为空，因此不生成文件'xx'。 → 请确认输入文件的内容。

S740 族转移指令优化工具 LOOP 操作说明书

1. 功能

这是将汇编程序的转移指令转换为最适合的转移指令的程序。

2. 生成文件

LOOP74 根据命令选项的指定将转换结果进行如下的输出：

在指定-ASM 命令选项时：

输出到 SRA74 生成的 xxx.i 文件（xxx.i 文件被输出到和输入文件相同的目录）。

在没有指定-ASM 选项时：

输出到 LOOP74 指定的结构化源文件。

注）附加-ASM 命令选项转换的文件不能用调试程序对源程序级进行调试。

3. 启动

3.1 输入参数

为了执行 LOOP74，必须输入以下信息（输入参数）：

• 源文件名

指定转换对象的源文件名（能指定的个数为 1 个）。
必须输入含有扩展名的文件名（能用的扩展名只限于.A74）。
能给文件名指定目录路径。
只指定文件名时，处理当前驱动器的当前目录中的文件。

• LOOP74 命令选项

选项	内容
一.	画面上不输出执行状况（错误信息除外）
-ASM	将结构化指令展开为汇编程序助记符的转移指令为对象，进行转移指令的转换处理。 （使用此选项时，不能进行源程序级的调试。）
-V	在显示 LOOP74 版本后结束。
-COUNTn	在达到 n（10 进制数值）指定的转换次数时结束转换。 n 必须指定在 1～100 的范围内。 默认值为 100 次的转换次数。
-COMMENT	将表示 LOOP74 转换的注解附加到转换对象行。

由于上述的选项辨别大小写字母，因此必须如上所述用大写字母指定。

・SRA74 命令选项
指定 LOOP74 在启动 SRA74 时指定的用于 SRA74 的命令选项。
LOOP74 将除自己支持的命令选项以外的选项都判断为 SRA74 的命令选项而交给 SRA74。

3.2 启动方法

LOOP74 用以下格式启动：

A> LOOP74 SAMPLE.A74 -COMMENT -S -C -L<RET>

在上述情况下，将“SAMPLE.A74”指定为进行转换的汇编程序源文件，在转换行指示 LOOP74 附加注解，同时指定将“-S -C -L”作为命令选项交给 SRA74。
另外，上述的指定即使变为如下的指定也被同样处理：

A> LOOP74 -L SAMPLE.A74 -S -COMMENT -C <RET>

即 LOOP74 与命令参数的指定顺序无关。
※LOOP74 自动生成标记文件，因此没必要记述 SRA74 的“-E”选项（即使记述也无妨）。

4. 输入源文件格式

LOOP74 处理的源文件格式符合 SRA74 的输入源文件格式。
对于 SRA74 不能处理的源文件，LOOP74 也不能处理。
另外，对于 LOOP74，有以下的限制事项：

・不对应用.EXT 等声明的标号。
・不对应标号定义中没有记述‘:’的源程序。
・不对应如.if 伪指令的条件“伪”部分等 SRA74 不处理的部分。
・如果转移指令的操作数记述了如下的相对地址就出错。

另外，LOOP74 中有无相对地址记述的判断是根据转移指令操作数中是否存在‘*’的记述，如果存在就判断为相对地址的指定。
（例）：

```
BRA * + 5
BRA * + label
```

5. 输出文件格式

LOOP74 用以下格式输出转换对象的转移指令。
通常，LOOP74 删除转换前的指令行，但是在指定命令选项-COMMENT 时，作为注解保留原来的指令行。

●没有指定命令参数-COMMENT 时（默认值）
・转换对象行①

```
BEQ     L1
```

• LOOP74 的转换结果

▲BNE ▲..lop1▲; This is the line which loop74 generated

▲JMP ▲L1▲; This is the line which loop74 generated

..lop1：

• 转换对象行②

BEQ L1 ;goto L1

• LOOP74 的转换结果

▲BNE ▲..lop1 ▲; This is the line which loop74 generated

▲JMP ▲L1 ▲;goto L1

..lop1：

如上所述，在 LOOP74 生成的转移指令行附加了表示 LOOP74 生成的注解。
另外，在转换对象行有注解记述时，LOOP74 将对象行记述的注解输出到最后输出的转移指令。

●指定命令参数-COMMENT 时

• 转换对象行①

BEQ L1

• LOOP74 的转换结果

； BEQ L1

▲BNE ▲..lop1▲; This is the line which loop74 generated

▲JMP ▲L1▲; This is the line which loop74 generated

..lop1：

• 转换对象行②

BEQ L1 ;goto L1

• LOOP74 的转换结果

； BEQ L1 ;goto L1

▲BNE ▲..lop1 ▲; This is the line which loop74 generated

▲JMP ▲L1 ▲; This is the line which loop74 generated

..lop1：

如上所述，指定命令选项-COMMENT 时，作为注解输出转换对象行。
另外，在转换对象行有注解记述时，LOOP74 不将对象行记述的注解输出到最后输出的转移指令。

6. 转换规则

LOOP74 根据以下规则转换转移指令：

输入行		输出行	
BEQ	L1	BNE	..lop1
		JMP	L1
		..lop1:	
BNE	L1	BEQ	..lop1
		JMP	L1
		..lop1:	
BCC	L1	BCS	..lop1
		JMP	L1
		..lop1:	
BCS	L1	BCC	..lop1
		JMP	L1
		..lop1:	
BMI	L1	BPL	..lop1
		JMP	L1
		..lop1:	
BPL	L1	BMI	..lop1
		JMP	L1
		..lop1:	
BVC	L1	BVS	..lop1
		JMP	L1
		..lop1:	
BVS	L1	BVC	..lop1
		JMP	L1
		..lop1:	
BBC	BIT,MEM,L1	BBS	BIT,MEM,..lop1
		JMP	L1
		..lop1:	
BBS	BIT,MEM,L1	BBC	BIT,MEM,..lop1
		JMP	L1
		..lop1:	
BRA	L1	JMP	L1

LOOP74 生成的标号以..lop1～..lop65535 格式输出。

另外，在连续多行记述转移到同一转移目标的条件转移指令时，根据以下规则转换：

输入行		输出行	
Bcnd_1	L1	Bcnd_1	..lop1
Bcnd_2	L1	Bcnd_2	..lop1
:		:	
:		:	
Bcnd_n	L1	BcndR_n	..lop2
		..lop1:	
		JMP	L1
		..lop2:	

上述 Bcnd_1、Bcnd_2、Bcnd_n 表示以下的条件转移指令：

BEQ、BNE、BCC、BCS、BMI、BPL、BVC、BVS

BcndR_n 表示 Bcnd_n 转移指令取反的条件转移指令。

如上所述，在转移到同一转移目标的条件转移指令连续时，LOOP74 只将最后记述的条件转移指令取反并展开条件。

具体例子如下：

输入行		输出行	
BEQ	L1	BEQ	..lop1
BCS	L1	BCC	..lop2
		..lop1:	
		JMP	L1
		..lop2:	

7. 错误

如果在执行 LOOP74 期间检测到错误，画面上就显示错误信息，并且中止 LOOP74 的执行。错误一览表如下所示：

错误信息	内容和对策
Can't create file 'filename'	不能生成'filename'文件。 请确认磁盘容量。
Can't create Temporary file	不能生成临时文件。 请确认目录是否处于写禁止状态。
Can't open file 'filename'	不能打开'filename'文件。 请确认指定的文件名。
Can't translate	即使是最大的转移指令也无法到达转移地址。 请确认源程序内容。
Command line is too long	命令行的字符数过多。 LOOP74 的命令行最多只能接受 255 个字节。请减少输入的字符数。

错误信息	内容和对策
Error occurred in executing 'sra74'	发生了 SRA74 的执行错误。 请重新执行 SRA74。
Illegal source file	为 LOOP74 不能解析的源文件格式。 请指定正确格式的源文件。
Illegal TAG file	为 LOOP74 不能解析的标记文件格式。 请指定正确格式的标记文件。
No input files specified	没有指定输入文件。 请指定输入文件。
Not enough memory	存储容量不足。 请将输入文件分开或者增加存储容量。
Option 'xx' is not appropriate	命令选项 xx 的记述不正确。 请重新指定命令选项。
Source files number exceed 1	指定了多个源文件。 请指定 1 个源文件。
Too many branch error	转换的转移指令过多。 请将源文件分开，减少 1 个文件中的 LOOP74 处理的转移指令个数。

8. 警告

如果在执行 LOOP74 期间检测到警告，画面上就显示警告信息，LOOP74 的处理继续进行。警告一览表如下所示：

警告信息	内容和对策
Relative address is specified	对转移目标指定了相对地址。 不进行该行的转换处理。

9. 给 OS 的返回值

在结束处理时，LOOP74 按如下内容给 OS 返回值。

返回值	内容
0	正常结束。
1	与 LOOP74 读取的源文件和标记文件内容有关的错误。
2	对于命令行输入，发生了错误。
3	与操作系统有关的错误。
4	根据^C（Ctrl + C）输入的强制结束。

修订记录	M3T-SRA74 V.4.10 用户手册

Rev.	发行日	修订内容	
		页	修订处
1.00	2005.03.04	—	初版发行

M3T-SRA74 V.4.10 用户手册

发行年月日	2005 年 03 月 04 日 Rev.1.00
发　　行	Sales Strategic Planning Div. Renesas Technology Corp. 株式会社瑞萨科技 营业企画综合部
编　　辑	Technical Documentation & Information Department Renesas Kodaira Semiconductor Co., Ltd. 株式会社瑞萨小平半导体技术文献部

Renesas Technology Corp. Sales Strategic Planning Div. Nippon Bldg., 2-6-2, Ohte-machi, Chiyoda-ku, Tokyo 100-0004, Japan

RENESAS SALES OFFICES

http://www.renesas.com

Refer to "**http://www.renesas.com/en/network**" for the latest and detailed information.

Renesas Technology America, Inc.
450 Holger Way, San Jose, CA 95134-1368, U.S.A
Tel: <1> (408) 382-7500, Fax: <1> (408) 382-7501

Renesas Technology Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.
Tel: <44> (1628) 585-100, Fax: <44> (1628) 585-900

Renesas Technology Hong Kong Ltd.
7th Floor, North Tower, World Finance Centre, Harbour City, 1 Canton Road, Tsimshatsui, Kowloon, Hong Kong
Tel: <852> 2265-6688, Fax: <852> 2730-6071

Renesas Technology Taiwan Co., Ltd.
10th Floor, No.99, Fushing North Road, Taipei, Taiwan
Tel: <886> (2) 2715-2888, Fax: <886> (2) 2713-2999

Renesas Technology (Shanghai) Co., Ltd.
Unit2607 Ruijing Building, No.205 Maoming Road (S), Shanghai 200020, China
Tel: <86> (21) 6472-1001, Fax: <86> (21) 6415-2952

Renesas Technology Singapore Pte. Ltd.
1 Harbour Front Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: <65> 6213-0200, Fax: <65> 6278-8001

Colophon 2.0

M3T-SRA74 V.4.10

740族可再定位汇编程序

瑞萨电子株式会社

RCJ10J0008-0100