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

RENESAS

740 小型仿真调试器V.1.02

瑞萨单片机开发环境系统

瑞萨电子
www.renesas.com

Rev.1.00 2008.01

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

注意

本文只是参考译文，前页所载英文版“Cautions”具有正式效力。

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本公司产品的参考资料，对于本资料中所记载的技术信息，并非意味着对本公司或者第三者的知识产权及其他权利做出保证或对实施权力进行的承诺。
2. 对于因使用本资料所记载的产品数据、图、表、程序、算法及其他应用电路例而引起的损害或者对第三者的知识产权及其他权利造成侵犯，本公司不承担任何责任。
3. 不能将本资料所记载的产品和技术用于大规模破坏性武器的开发等目的、军事目的或其他的军需用途方面。另外，在出口时必须遵守日本的《外汇及外国贸易法》及其他出口的相关法令并履行这些法令中规定的必要手续。
4. 本资料所记载的产品数据、图、表、程序、算法以及其他应用电路例等所有信息均为本资料发行时的内容，本公司有可能在未做事先通知的情况下，对本资料所记载的产品或者产品规格进行更改。所以在购买和使用本公司的半导体产品之前，请事先向本公司的营业窗口确认最新的信息并经常留意本公司通过公司主页(http://www.renesas.com)等公开的最新信息。
5. 对于本资料中所记载的信息，制作时我们尽力保证出版时的精确性，但不承担因本资料的叙述不当而致使顾客遭受损失等的任何相关责任。
6. 在使用本资料所记载的产品数据、图、表等所示的技术内容、程序、算法及其他应用电路例时，不仅要对所使用的技术信息进行单独评价，还要对整个系统进行充分的评价。请顾客自行负责，进行是否适用的判断。本公司对于是否适用不负任何责任。
7. 本资料中所记载的产品并非针对万一出现故障或是错误运行就会威胁到人的生命或给人体带来危害的机器、系统(如各种安全装置或者运输交通用的、医疗、燃烧控制、航天器械、核能、海底中继用的机器和系统等)而设计和制造的，特别是对于品质和可靠性要求极高的机器和系统等（将本公司指定用于汽车方面的产品用于汽车时除外）。如果要用于上述的目的，请务必事先向本公司的营业窗口咨询。另外，对于用于上述目的而造成的损失等，本公司概不负责。
8. 除上述第7项内容外，不能将本资料中记载的产品用于以下用途。如果用于以下用途而造成的损失，本公司概不负责。
   1）生命维持装置。
   2）植埋于人体使用的装置。
   3）用于治疗（切除患部、给药等）的装置。
   4）其他直接影响到人的生命的装置。
9. 在使用本资料所记载的产品时，对于最大额定值、工作电源电压的范围、放热特性、安装条件及其他条件请在本公司规定的保证范围内使用。如果超出了本公司规定的保证范围使用时，对于由此而造成的故障和出现的事故，本公司将不承担任何责任。
10. 本公司一直致力于提高产品的质量和可靠性，但一般来说，半导体产品总会以一定的概率发生故障、或者由于使用条件不同而出现错误运行等。为了避免因本公司的产品发生故障或者错误运行而导致人身事故和火灾或造成社会性的损失，希望客户能自行负责进行冗余设计、采取延烧对策及进行防止错误运行等的安全设计（包括硬件和软件两方面的设计）以及老化处理等，这是作为机器和系统的出厂保证。特别是单片机的软件，由于单独进行验证很困难，所以要求在顾客制造的最终的机器及系统上进行安全检验工作。
11. 如果把本资料所记载的产品从其载体设备上卸下，有可能造成婴儿误吞的危险。顾客在将本公司产品安装到顾客的设备上时，请顾客自行负责将本公司产品设置为不容易剥落的安全设计。如果从顾客的设备上剥落而造成事故时，本公司将不承担任何责任。
12. 在未得到本公司的事先书面认可时，不可将本资料的一部分或者全部转载或者复制。
13. 如果需要了解关于本资料的详细内容，或者有其他关心的问题，请向本公司的营业窗口咨询。

Active X、 Microsoft、 MS-DOS、 Visual Basic、 Visual C++、 Windows 和 Windows NT 是 Microsoft Corporation 在美国和其他国家或地区的注册商标或商标。

IBM 和 AT 是 International Business Machines Corporation 的注册商标。

Intel 和 Pentium 是 Intel Corporation 的注册商标。

Adobe 和 Acrobat 是 Adobe Systems Incorporated 的注册商标。

所有其他品牌和产品名称均为其各自持有者的商标、注册商标或服务标志。

请遵循安全第一进行电路设计

- 虽然瑞萨科技和瑞萨解决方案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失。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

- 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方所有的知识产权和其它权利的许可。
- 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，瑞萨科技和瑞萨解决方案不承担责任。
- 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、瑞萨解决方案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，瑞萨科技和瑞萨解决方案不承担责任。同时也请注意瑞萨科技和瑞萨解决方案通过各种方式公布的信息，包括瑞萨网站 (http：//www.renesas.com)。
- 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，瑞萨科技和瑞萨解决方案不承担责任。
- 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、航天、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、瑞萨解决方案或者经授权的瑞萨科技产品经销商联系。
- 未经瑞萨科技和瑞萨解决方案的书面许可，不得翻印或者复制全部或者部分资料的内容。
- 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以外的国家或地区。禁止违反日本和 （或者）目的地国家或地区的出口管理法和法规的任何转卖、挪用或者再出口。
- 如果需要了解本资料所记载的信息或产品详情，请与瑞萨科技或瑞萨解决方案联系。

有关对本资料内容或产品的查询，请填妥安装程序在下列目录中生成的文本文件，并将它电邮给当地的经销商。

\SUPPORT\Product-name\SUPPORT.TXT

瑞萨工具主页：http://www.renesas.com/en/tools

概要

高性能嵌入式工作区 （High-performance Embedded Workshop，简称 HEW）是一种图形用户界面，使用它可以轻松开发和调试使用 C/C++ 编程语言和汇编语言编写的瑞萨单片机应用程序。它旨在提供一种高性能且直观的方式来存取、观察和修改运行应用程序的调试平台。

本帮助手册说明 HEW 作为一种 “调试器”的功能。

目标系统

该调试器在小型仿真器系统上操作。

支持的 CPU

本帮助手册说明与以下 CPU 相对应的调试功能。

- 740 族

  注意：在本帮助手册中，与此系列 CPU 相关的信息会注明 “用于 740”。

目　录

第 1 部分　设置调试器 1

1　功能 3
1.1　实时 RAM 监控功能 3
1.1.1　RAM 监控区域 3
1.1.2　采样周期 4
1.1.3　相关窗口 4
1.2　断点功能 4
1.2.1　软件断点功能 4
1.2.1.1　设置软件断点 4
1.2.1.2　可设置软件断点的区域 4
1.2.2　硬件断点 4
1.3　实时跟踪功能 5
1.3.1　跟踪区域 5
1.4　GUI 输入 / 输出功能 5

2　启动调试器之前 6
2.1　仿真器使用的通信方法 6
2.1.1　USB 接口 6
2.2　下载固件 6
2.3　仿真器启动前的设置 7
2.3.1　USB 通信 7
2.3.1.1　安装 USB 设备驱动程序 7

3　使用前的准备工作 8
3.1　工作空间、工程和文件 8
3.2　启动 HEW 9
3.2.1　创建新的工作空间 （使用工具链） 10
3.2.1.1　步骤 1：创建新的工作空间 10
3.2.1.2　步骤 2：设置工具链 11
3.2.1.3　步骤 3：设置目标平台 12
3.2.1.4　步骤 4：设置配置文件名 13
3.2.1.5　步骤 5：检查创建的文件名 14
3.2.2　创建一个新的工作空间 （未使用工具链） 15
3.2.2.1　步骤 1：创建新的工作空间 15
3.2.2.2　步骤 2：设置目标平台 16
3.2.2.3　步骤 3：设置配置文件名 17
3.2.2.4　步骤 4：注册要下载的加载模块 18
3.3　启动调试器 20
3.3.1　连接仿真器 20
3.3.2　结束仿真器 20

4　设置调试器 21
4.1　Init （初始化）对话框 21
4.1.1　MCU 选项卡 22
4.1.1.1　指定 MCU 文件 22
4.1.2　Debugging Information （调试信息）选项卡 23
4.1.2.1　显示使用的编译器及其目标格式 24
4.1.2.2　指定调试信息的存储 24

4.1.3 Script （脚本）选项卡 25
4.1.3.1 自动执行脚本命令 25
4.2 设置通信接口 25
4.2.1 设置 USB 接口 25
4.3 设置 740 使用的调试器 26
4.3.1 Map 命令 26
4.4 创建 MCU 文件的方法 27
4.4.1 创建 MCU 文件的方法 （适用于 740 调试器） 27
4.4.1.1 示例 27

第 2 部分 教程 28

5 教程 30
5.1 简介 30
5.2 使用 30
5.2.1 步骤 1：启动调试器 30
5.2.1.1 使用前的准备工作 30
5.2.1.2 设置调试器 30
5.2.2 步骤 2：检查 RAM 的运作情况 31
5.2.2.1 检查 RAM 的运作情况 31
5.2.3 步骤 3：下载教程程序 32
5.2.3.1 下载教程程序 32
5.2.3.2 显示源程序 33
5.2.4 步骤 4：设置断点 34
5.2.4.1 设置软件断点 34
5.2.5 步骤 5：执行程序 35
5.2.5.1 复位 CPU 35
5.2.5.2 执行程序 35
5.2.5.3 查看中断原因 36
5.2.6 步骤 6：查看断点 37
5.2.6.1 查看断点 37
5.2.7 步骤 7：查看寄存器 38
5.2.7.1 查看寄存器 38
5.2.7.2 设置寄存器值 38
5.2.8 步骤 8：查看存储器 39
5.2.8.1 查看存储器 39
5.2.9 步骤 9：监视变量 40
5.2.9.1 监视变量 40
5.2.9.2 注册变量 41
5.2.10 步骤 10：逐步执行程序 42
5.2.10.1 执行 [Step In] （跳入）命令 42
5.2.10.2 执行 [Step Out] （跳出）命令 43
5.2.10.3 执行 [Step Over] （跳过）命令 44
5.2.11 步骤 11：强制暂停程序执行 44
5.2.11.1 强制暂停程序执行 44
5.2.12 步骤 12：显示局部变量 45
5.2.12.1 显示局部变量 45
5.2.13 步骤 13：堆栈跟踪功能 45
5.2.13.1 引用函数调用状态 46
5.2.14 后续内容 46

第 3 部分　参考 47

6　窗口 / 对话框 49
6.1　[RAM Monitor]（RAM 监控）窗口 50
6.1.1　扩展菜单 52
6.1.2　设置 RAM 监控区域 53
6.1.2.1　更改 RAM 监控区域 53
6.2　[ASM Watch]（ASM 监视）窗口 54
6.2.1　扩展菜单 55
6.3　[C Watch]（C 监视）窗口 56
6.3.1　扩展菜单 58
6.4　[Script]（脚本）窗口 59
6.4.1　扩展菜单 60
6.5　[S/W Break Point Setting]（软件断点设置）窗口 61
6.5.1　命令按钮 62
6.5.2　从 [Editor (Source)]（编辑器（源））窗口设置和删除断点 63
6.6　[H/W Break Point Setting]（硬件断点设置）对话框 64
6.6.1　指定断点事件 64
6.6.1.1　取指令 65
6.6.1.2　存储器存取 65
6.6.1.3　位存取 67
6.6.1.4　禁止硬件断点功能 68
6.7　[Trace Point Setting]（跟踪点设置）对话框 68
6.7.1　指定跟踪范围 68
6.8　[Trace]（跟踪）窗口 69
6.8.1　配置“Bus Mode”（总线模式） 69
6.8.2　配置“Disassemble Mode”（反汇编模式） 71
6.8.3　配置“Data Access Mode”（数据存取模式） 72
6.8.4　配置“Source Mode”（源模式） 73
6.8.5　扩展菜单 74
6.8.6　740 调试器上的总线信息的显示 75
6.9　[GUI I/O] 窗口 76
6.9.1　扩展菜单 77

7　脚本命令表 78
7.1　脚本命令表（按功能分类） 78
7.1.1　执行命令 78
7.1.2　文件操作命令 78
7.1.3　寄存器操作命令 79
7.1.4　存储器操作命令 79
7.1.5　汇编 / 反汇编命令 79
7.1.6　软件断点设置命令 80
7.1.7　硬件断点设置命令 80
7.1.8　实时跟踪命令 80
7.1.9　脚本 / 日志文件命令 80
7.1.10　程序显示命令 81
7.1.11　映像命令 81
7.1.12　时钟命令 81
7.1.13　C 语言调试命令 81
7.1.14　实用程序命令 81
7.2　脚本命令表（依字母顺序） 82

8 编写脚本文件 85
8.1 脚本文件的结构元素 85
8.1.1 脚本命令 85
8.1.2 赋值语句 86
8.1.3 条件语句 86
8.1.4 循环语句（while 及 endw）和中断语句 86
8.1.5 注释语句 86
8.2 编写表达式 87
8.2.1 常数 87
8.2.2 符号和标签 88
8.2.2.1 局部标签符号和作用域 88
8.2.2.2 标签和符号的优先级 89
8.2.3 宏变量 89
8.2.4 寄存器变量 90
8.2.5 存储器变量 90
8.2.6 行号 90
8.2.7 字符常数 91
8.2.8 运算符 91

9 C/C++ 表达式 92
9.1 编写 C/C++ 表达式 92
9.1.1 立即值 93
9.1.2 作用域解析 93
9.1.3 数学运算符 93
9.1.4 指针 93
9.1.5 引用 93
9.1.6 取反 94
9.1.7 使用点运算符的成员引用 94
9.1.8 使用箭头运算符的成员引用 94
9.1.9 指向成员的指针 95
9.1.10 括号 95
9.1.11 数组 95
9.1.12 基本数据类型转换 95
9.1.13 typedef 类型转换 96
9.1.14 变量名 96
9.1.15 函数名 96
9.1.16 字符常数 96
9.1.17 字符串文字 96
9.2 C/C++ 表达式的显示格式 97
9.2.1 枚举类型 97
9.2.2 基本数据类型 97
9.2.3 指针类型 98
9.2.4 数组类型 99
9.2.5 函数类型 99
9.2.6 引用类型 99
9.2.7 位域类型 99
9.2.8 未找到任何 C 符号时 100
9.2.9 语法错误 100
9.2.10 结构类型和联合类型 100

10 显示程序停止的原因 101

11 注意事项 102
11.1 通用注意事项 102
11.1.1 与 Windows 相关的文件操作 102
11.1.2 可设置软件断点的区域 102
11.1.3 获取或设置 C 变量 102
11.1.4 C++ 中的函数名 103
11.1.5 调试多个模块 103
11.1.6 同步调试 103
11.1.7 小型仿真器复位开关 103
11.2 740 调试器的注意事项 103
11.2.1 设置存储器映像 103
11.2.2 仿真器使用的堆栈区域 103
11.2.3 看门狗定时器 103
11.2.4 C 编译器 / 汇编器 / 连接器选项 104
11.2.5 调试 16 位定时器功能 104
11.2.6 关于 MCU 的内部 RAM 区域中的单步执行和程序断点功能 104
11.2.7 硬件事件 104
11.3 C 编译器 / 汇编器 / 连接器选项 105
11.3.1 使用 IAR C 编译器 (EW) 105
11.3.2 使用 IAR C 编译器 (ICC) 106
11.3.2.1 指定选项 106
11.3.2.2 命令执行示例 106
11.3.3 使用用于 740 族的汇编器套件时 107
11.3.3.1 命令执行示例 107

设置调试器

1 功能

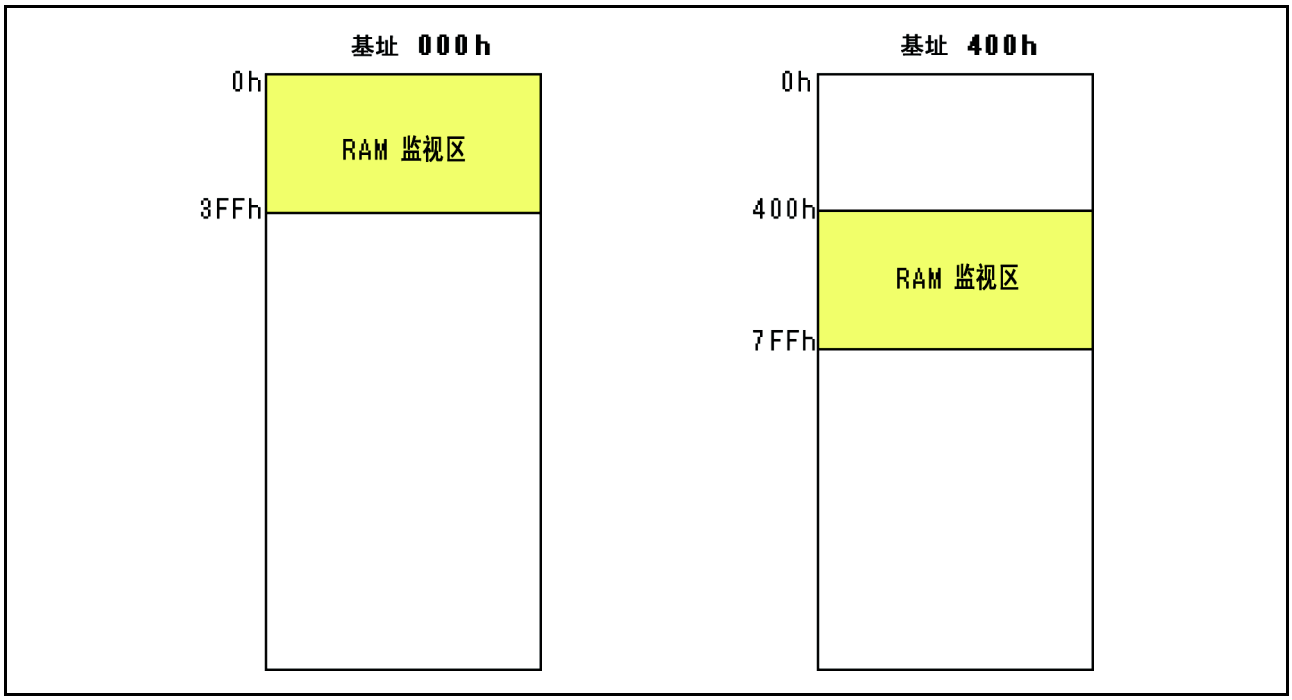
本调试器具有以下功能。

1.1 实时 RAM 监控功能

该功能可在不影响目标程序执行的实时性能的情况下，检查存储器内容的更改。
该小型仿真器系统包含一个 1 KB RAM 监控区域 （不能分成更小的区域）。

1.1.1 RAM 监控区域

本调试器提供一个 1KB 的 RAM 监控区域，该区域可以位于任何连续的地址上。



1.1.2 采样周期

采样周期表示显示更新间隔。

可以在支持 RAM 监控的任何窗口中指定该功能。（默认情况下，设置 100 ms 的间隔。）

根据不同的操作环境，实际采样周期可能比指定的周期更长。（采样周期取决于以下环境。）

- 通信接口
- 显示的 [RAM Monitor] （RAM 监控）窗口数
- 显示的 [RAM Monitor] （RAM 监控）窗口大小
- [ASM Watch] （ASM 监视）窗口的 RAM 监控区域内的 ASM 监视点数
- [C Watch] （C 监视）窗口的 RAM 监控区域内的 C 监视点数

1.1.3 相关窗口

下面显示可以使用实时 RAM 监控功能的窗口。

- [RAM Monitor] （RAM 监控）窗口
- [ASM Watch] （ASM 监视）窗口
- [C Watch] （C 监视）窗口

1.2 断点功能

1.2.1 软件断点功能

软件断点在指定地址执行命令之前中断目标程序。该断点称为软件断点。

软件断点是在 [Editor (Source)] （编辑器 （源））窗口或 [S/W Breakpoint Setting] （软件断点设置）窗口中进行设置 / 复位的。也可以暂时禁止 / 允许软件断点。

最多可以指定 64 个软件断点。指定两个或更多软件断点时，根据 OR 逻辑来组合断点。（到达任一断点均会中断目标程序。）

1.2.1.1 设置软件断点

可以通过以下窗口设置软件断点。

- [Editor (Source)] （编辑器 （源））窗口
- [S/W Break Point Setting] （软件断点设置）窗口

可以在 [Editor (Source)] （编辑器 （源））窗口中双击鼠标来设置 / 复位软件断点。

也可以改为在 [S/W Break Point Setting] （软件断点设置）窗口中暂时禁止 / 允许软件断点。

1.2.1.2 可设置软件断点的区域

可以针对软件断点进行设置的区域会根据产品的不同而有所不同。

有关可用于软件断点的区域，请参阅以下章节：

“11.1.2 可设置软件断点的区域”

1.2.2 硬件断点

此功能使目标程序在检测到存储器的数据读 / 写操作或指令的执行时停止。

可以设置具有通过计数的单地址断点。

1.3 实时跟踪功能

该功能记录目标程序执行历史记录。

最多可以记录 32K 执行周期历史记录。通过该记录，可以检查每个周期的总线信息、执行的指令和源程序执行路径。

实时跟踪功能记录目标程序的执行历史记录。

跟踪窗口会引用执行历史记录。

可以在以下模式中引用执行历史记录。

- “BUS mode”（总线模式）

  该模式用于按周期检查总线信息。显示内容取决于使用的 MCU 和仿真器系统。除总线信息外，该模式还允许以组合形式显示反汇编、源行或数据存取信息。

- “Disassemble mode”（反汇编模式）

  该模式用于检查执行的指令。除反汇编信息外，该模式还允许以组合形式显示源行或数据存取信息。

- “Data access mode”（数据存取模式）

  该模式用于检查数据读/写周期。除数据存取信息外，该模式还允许以组合形式显示源行信息。

- “Source mode”（源模式）

  该模式用于检查源程序中的程序执行路径。

1.3.1 跟踪区域

可以使用此调试器引用 32K 个周期的执行历史记录。

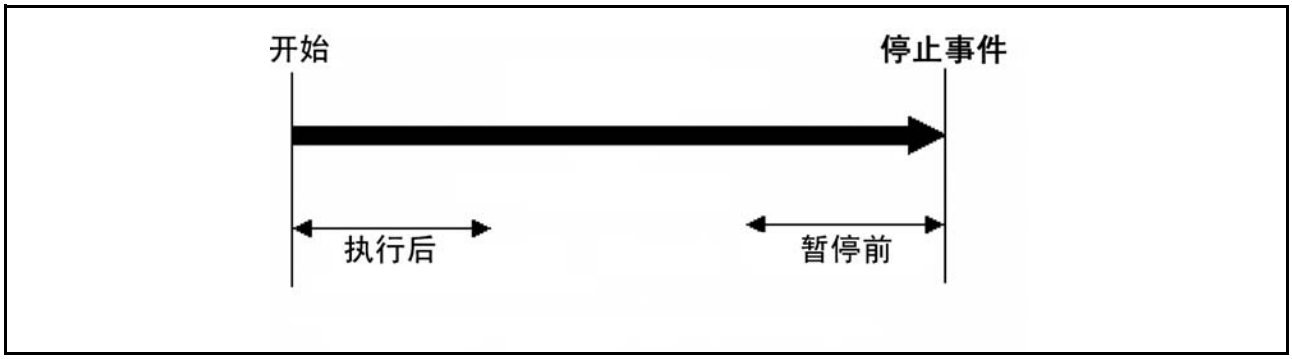
支持下面 5 个模式的跟踪区域。

- “Before Break”（暂停前）

  目标程序停止前的 32K 个周期

- “After Go”（执行后）

  直到 32K 个周期写入跟踪存储器为止



“Before Break”（暂停前）为默认设置。若要在停止目标程序之前引用执行历史记录，请使用 “Before Break”（暂停前）（不需要指定跟踪事件）。

1.4 GUI 输入 / 输出功能

该功能在窗口上模拟用户目标系统的按键输入面板 （按钮）和输出面板。

按钮可以用于输入面板，标签 （字符串）和 LED 可以用于输出面板。

2 启动调试器之前

2.1 仿真器使用的通信方法

支持的通信方法如下。

- M38000T2-CPE 作为通信界面，支持 USB。

2.1.1 USB 接口

- 支持的主机操作系统是 Windows Me/98/2000/XP。不能在其他任何操作系统中使用 USB 通信。
- 符合 USB 标准 1.1。
- 不支持通过 USB 集线器进行的连接。
- 通过使用 USB 电缆连接主机和仿真器，可以使用向导安装支持的设备驱动程序。
- 仿真器附带有所需的电缆。

2.2 下载固件

需要在仿真器上下载固件，该固件对应于启动调试器时连接的小型仿真器。

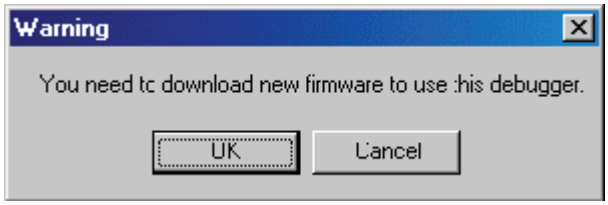
- 下载到仿真器的固件是未知固件。
- 首次设置了调试器。
- 对仿真调试器进行了升级。

在给小型仿真器加电以后的两秒钟内，按系统复位开关，以建立维护模式。

该调试器在启动时会搜索下载到仿真器的固件版本。同时，当下载到仿真器的固件是旧版本时，会设置一种模式，强制让该调试器下载固件。

如果仿真器设置为强制使调试器下载固件的模式，则当该调试器启动时，会打开以下对话框。

单击 [OK]（确定）按钮，下载固件。



2.3 仿真器启动前的设置

2.3.1 USB 通信

USB 设备的连接情况可以由 Windows 的即插即用功能检测到。已连接 USB 设备所需的设备驱动程序会自动进行安装。有关详细信息，请参阅 “安装 USB 设备驱动程序”。

2.3.1.1 安装 USB 设备驱动程序

已连接的 USB 设备可以由 Windows 的即插即用功能检测到。检测到设备以后， USB 设备驱动程序的安装向导会启动。下面说明安装 USB 设备驱动程序的过程。

1. 使用 USB 电缆连接主机和仿真器。
2. 将仿真器的通信接口开关设置到 “USB”位置。然后接通仿真器的电源。
3. 此时会出现下面显示的对话框。

继续按照向导的指示进行操作，此时会显示一个指定设置信息文件 （inf 文件）的对话框。指定 musbdrv.inf 文件，该文件存储在该调试器安装目录下面。

注意事项

- 安装 USB 设备驱动程序之前，必须已安装所用的调试器。首先安装该调试器。
- USB 通信只能用在 Windows Me/98/2000/XP 中，而不能用在其他任何操作系统中。
- 使用 Windows 2000/XP 时，安装 USB 设备驱动程序的用户需要管理员权限。
- 在安装过程中，会输出一条信息，指示找不到适当的设备驱动程序 musbdrv.sys。在这种情况下，指定存储在与 musbdrv.inf 文件相同的目录中的 musbdrv.sys。

3 使用前的准备工作

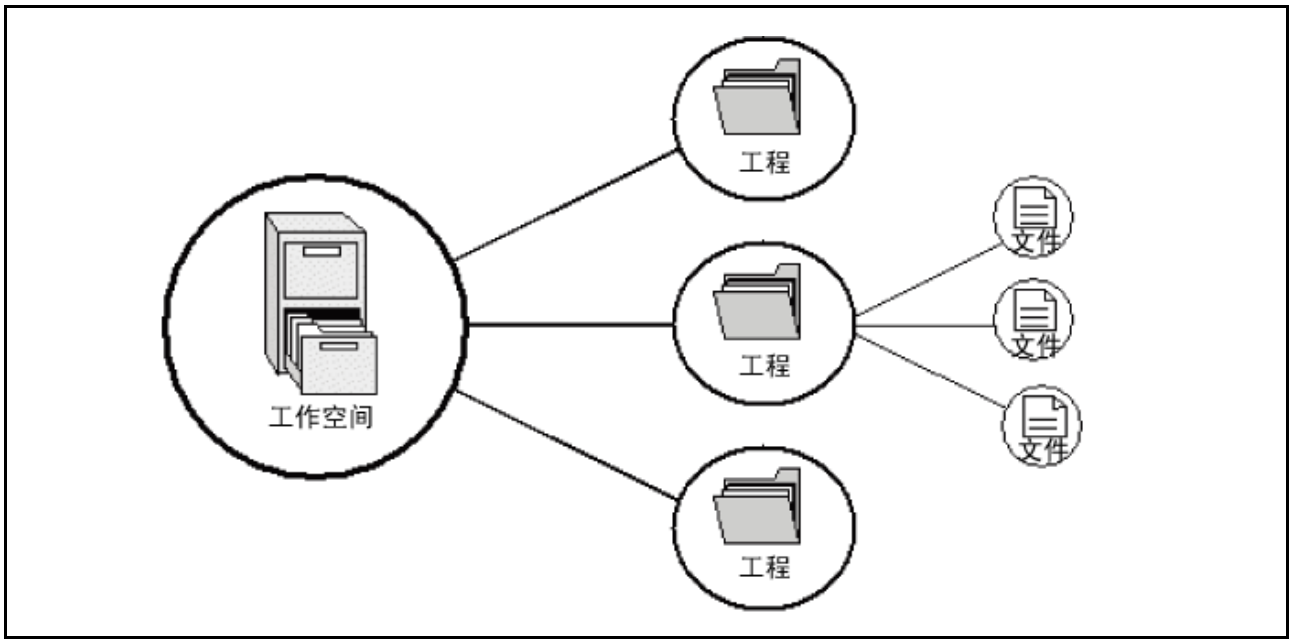
请运行 HEW 并连接仿真器。

此外，为了用该产品进行调试，需要创建工作空间。

3.1 工作空间、工程和文件

就像文字处理软件允许创建和修改文档一样，该产品允许创建和修改工作空间。

可以将工作空间视为工程的贮存箱；同样，可以将工程视为工程文件的贮存箱。因此，每个工作空间可包含一个或多个工程，每个工程可包含一个或多个文件。

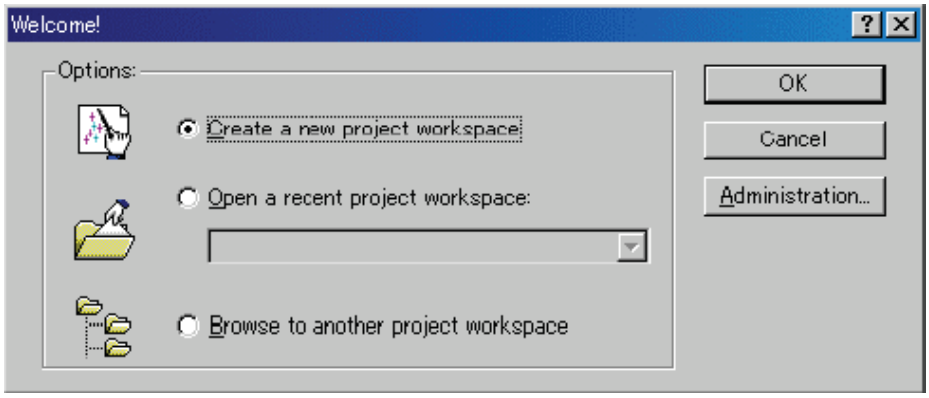


工作空间允许将相关工程组合在一起。例如，可能有一个应用程序，该应用程序需要针对不同处理器进行创建，或者可能正在同时开发一个应用程序和库。在工作空间中，还可以分层连接工程，这意味着创建一个工程时，它的所有 “子”工程都会首先进行创建。

不过，纯粹只是工作空间并不能发挥很大的作用。要进行实际操作，需要将工程添加到工作空间，然后再将文件添加到该工程。

3.2　启动 HEW

从 [Start]（中文系统：[开始]）菜单的 [Programs]（中文系统：[程序]）中激活 HEW。
此时会显示 [Welcome!]（欢迎 !）对话框。



可以在该对话框中创建或显示工作空间。

- [Create a new project workspace]（创建新的工程工作空间）单选按钮：
  创建一个新的工作空间。
- [Open a recent project workspace]（打开最近的工程工作空间）单选按钮：
  使用现有工作空间并显示所打开工作空间的历史记录。
- [Browse to another project workspace]（浏览到另一个工程工作空间）单选按钮：
  使用现有工作空间；
  在所打开的工作空间未保留历史记录时，使用此单选按钮。

在选择现有工作空间的情况下，选择 [Open a recent project workspace]（打开最近的工程工作空间）或 [Browse to another project workspace]（浏览到另一个工程工作空间）单选按钮，并选择工作空间文件 (.hws)。

有关创建新工作空间的方法，请参阅以下章节。
请参阅 “3.2.1 创建新的工作空间 （使用工具链）”
请参阅 “3.2.2 创建一个新的工作空间 （未使用工具链）”
* 当使用该产品调试现有加载模块文件时，该方法会创建一个工作空间。

创建新工作空间的方法取决于是否正在使用工具链。请注意，此产品不包括工具链。在已安装适用于所用 CPU 的 C/C++ 编译器套件的环境中，可以使用工具链。
有关详细信息，请参阅 C/C++ 编译器套件附带的手册。

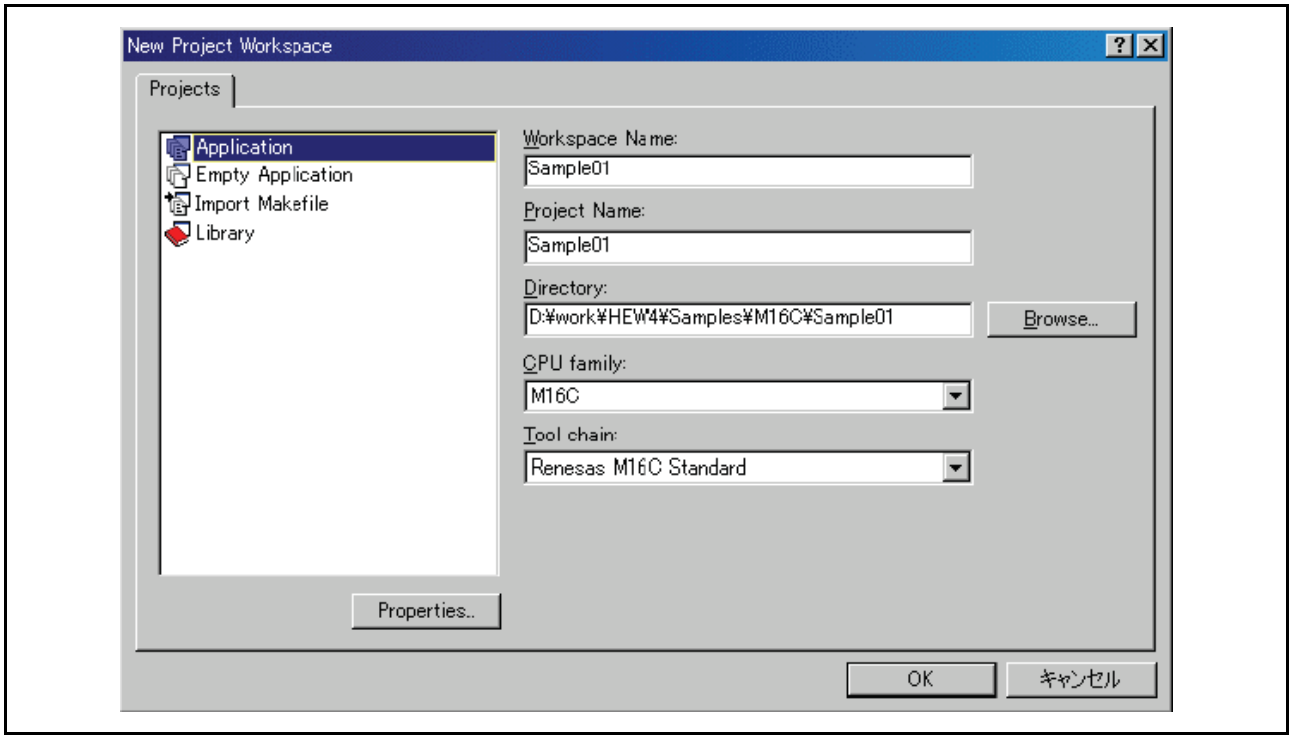
3.2.1　创建新的工作空间 （使用工具链）

3.2.1.1　步骤 1：创建新的工作空间

在激活 HEW 时显示的 [Welcome!] （欢迎！）对话框中，选择 [Create a new project workspace] （创建新的工程工作空间）单选按钮并单击 [OK] （确定）按钮。

此时开始创建新工作空间。

此时会显示下面的对话框。

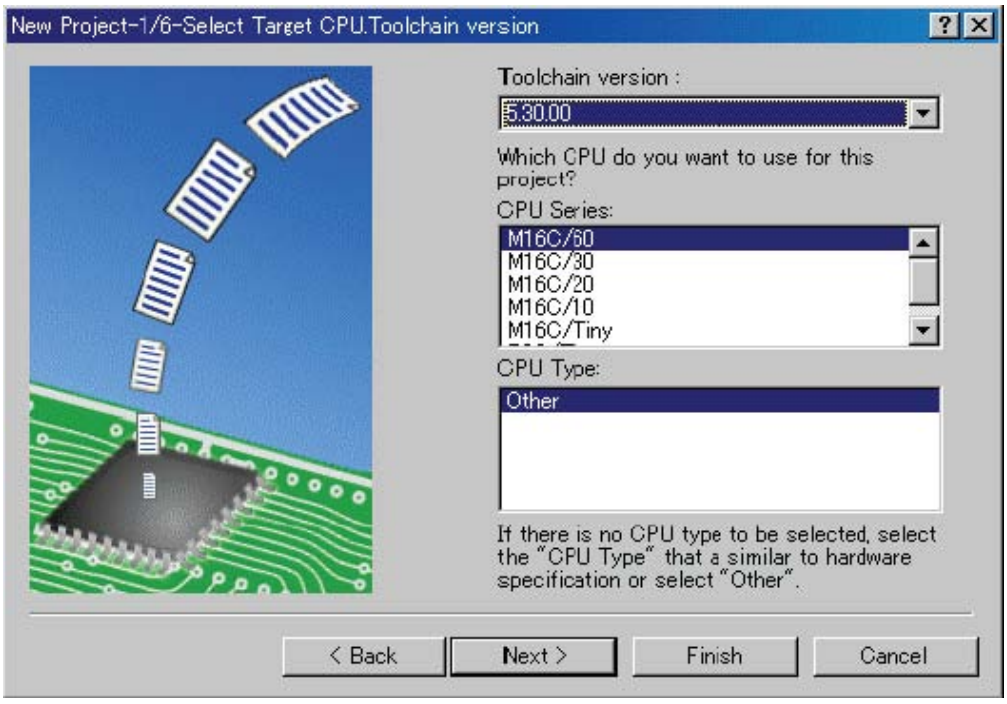


1. 选择目标 CPU 族
   在 [CPU family] （CPU 族）组合框中，选择目标 CPU 族。
2. 选择目标工具链
   在 [Tool chain] （工具链）组合框中，选择使用工具链时的目标工具链名称。
3. 选择工程类型
   在 [Project type] （工程类型）列表框中，选择要使用的工程类型。
   在本例中，选择 “Application”（应用程序）。
   （有关可以选择的工程类型的详细信息，请参阅 C/C++ 编译器套件附带的手册。）
4. 指定工作空间名称和工程名称
   — 在 [Workspace Name] （工作空间名称）编辑框中，输入新的工作空间名称。
   — 在 [Project Name] （工程名称）编辑框中，输入工程名称。如果工程名称与工作空间名称相同，则无需输入。
   — 在 [Directory] （目录）编辑框中，输入将在其中创建工作空间的目录名称。单击 [Browse...] （浏览…）按钮可以选择目录。

设置以后，单击 [OK] （确定）按钮。

3.2.1.2 步骤 2：设置工具链

此时会启动工程创建向导。



在这里设置以下内容。

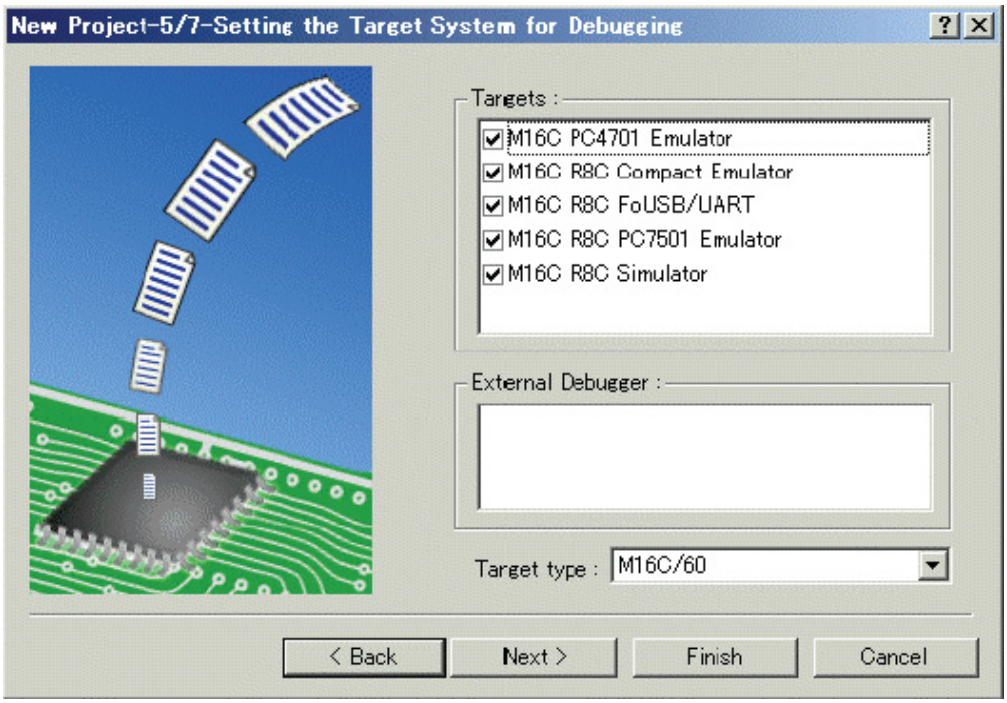
- 工具链
- 实时操作系统 （如果使用的话）的设置
- 启动文件、堆区、堆栈区等等的设置

请设置所需的信息并单击 [Next] （下一步）按钮。

设置的内容会根据所用 C/C++ 编译器套件的不同而有所不同。有关设置内容的详细信息，请参阅 C/C++ 编译器套件附带的手册。

3.2.1.3　　步骤 3：设置目标平台

选择用于调试的目标系统 （仿真器、模拟器）。
工具链设置完成以后，会显示以下对话框。

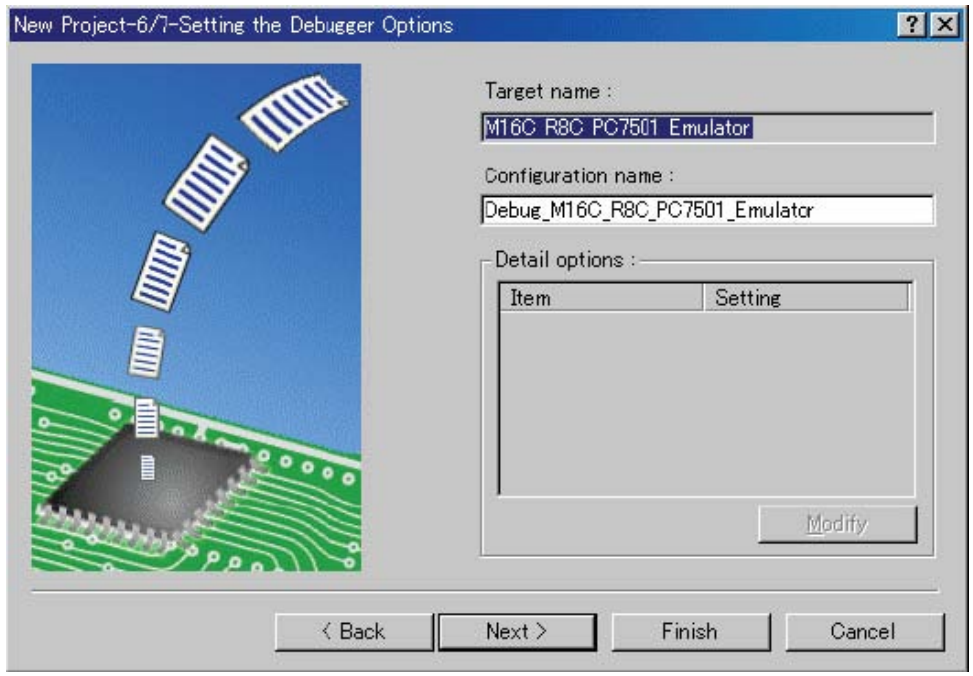


1. 选择目标类型
   在 [Target type] （目标类型）列表框中，选择目标 CPU 类型。
2. 选择目标平台
   在 [Targets] （目标）区域中，必须选择激活本调试器时所用会话文件的目标。
   选中目标平台所对应的框。（根据需要选择其他目标。）

然后单击 [Next] （下一步）按钮。

3.2.1.4 步骤 4：设置配置文件名

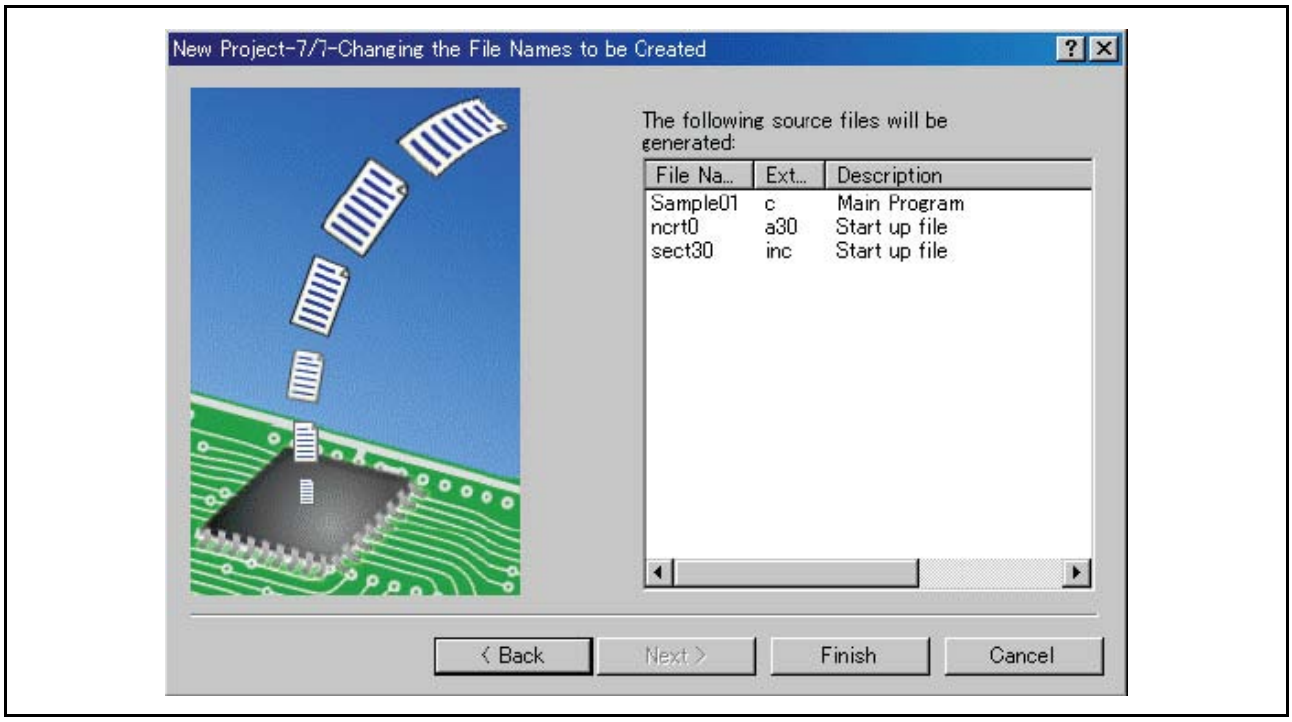
为选择的所有目标逐个设置配置文件名。
配置文件保存 HEW 的状态 （目标仿真器、模拟器除外）。



默认名称已设置。如果不需要更改，请单击 [next] （下一步）按钮 （使用默认名称）。

3.2.1.5 步骤 5：检查创建的文件名

最后，确认创建的文件名。此时会显示将由 HEW 生成的文件；如果要更改文件名，请选择并单击该文件名，然后输入新名称。



仿真器设置到此结束。
按照屏幕上的指示退出工程生成器。

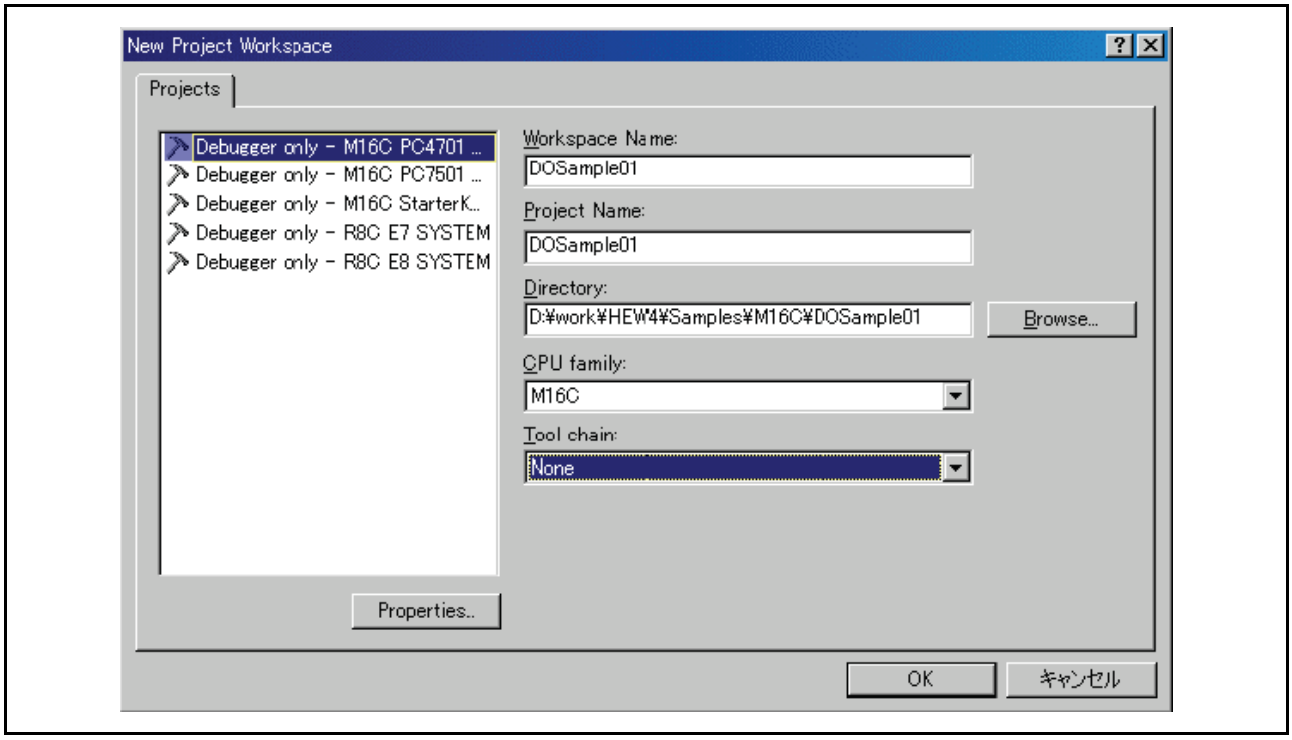
3.2.2　创建一个新的工作空间 （未使用工具链）

当使用该产品调试现有加载模块文件时，该方法会创建一个工作空间。

3.2.2.1　步骤 1：创建新的工作空间

在激活 HEW 时显示的 [Welcome!] （欢迎！）对话框中，选择 [Create a new project workspace] （创建新的工程工作空间）单选按钮并单击 [OK] （确定）按钮。

此时开始创建新工作空间。此时会显示下面的对话框。

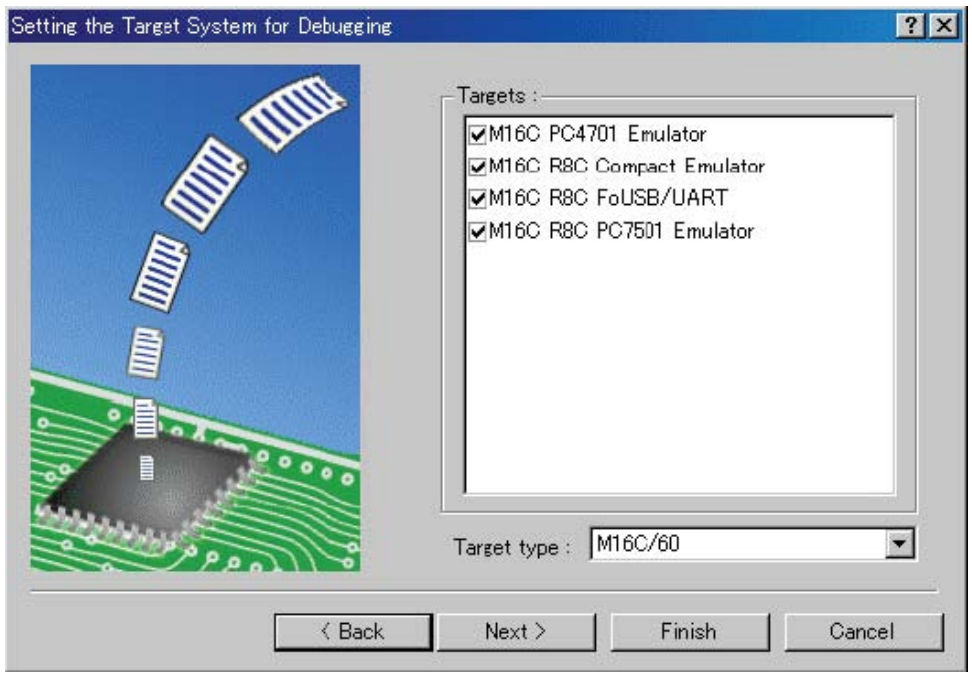


1. 选择目标 CPU 族
   在 [CPU family] （CPU 族）组合框中，选择目标 CPU 族。
2. 选择目标工具链
   在 [Tool chain] （工具链）组合框中，选择 “None”（无）。在这种情况下，不使用工具链。
   （如果未安装工具链，则该组合框中会显示固定信息。）
3. 选择工程类型
   （如果不使用工具链，则 [Project Type] （工程类型）列表框上会显示 “Debugger only - Target Name”（仅限调试器 - 目标名称）。选择该选项。（如果显示两个或更多工程类型，请选择其中的一个。）
4. 指定工作空间名称和工程名称
   — 在 [Workspace Name] （工作空间名称）编辑框中，输入新的工作空间名称。
   — 在 [Project Name] （工程名称）编辑框中，输入工程名称。如果工程名称与工作空间名称相同，则无需输入。
   — 在 [Directory] （目录）编辑框中，输入将在其中创建工作空间的目录名称。单击 [Browse...] （浏览…）按钮可以选择目录。

设置以后，单击 [OK] （确定）按钮。

3.2.2.2　　步骤 2：设置目标平台

选择用于调试的目标系统 （仿真器、模拟器）。
此时会启动一个向导并显示以下对话框。



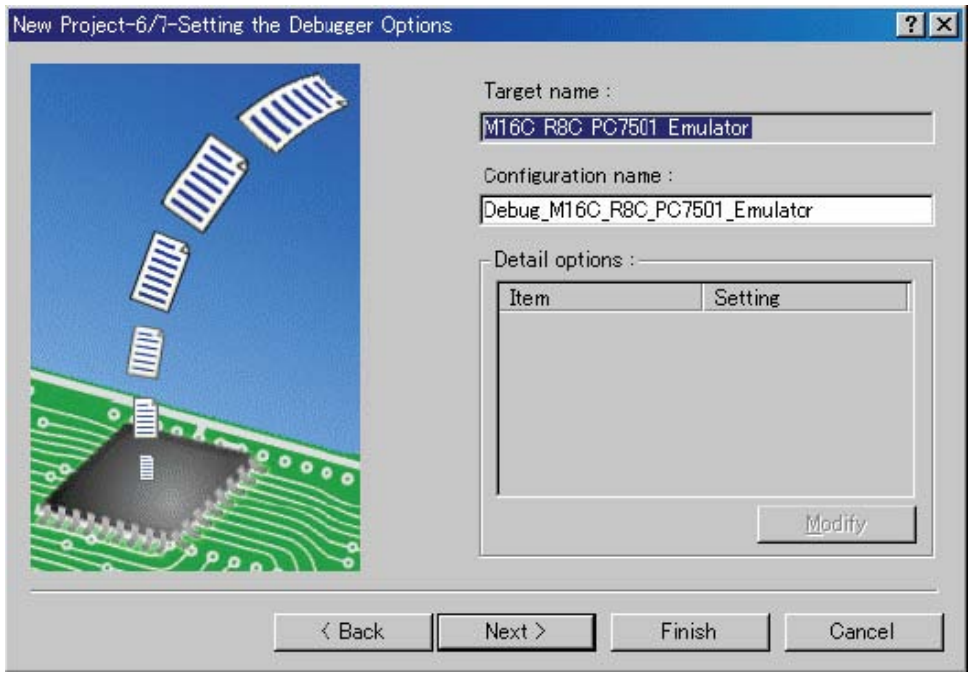
1. 选择目标类型
   在 [Target type] （目标类型）列表框中，选择目标 CPU 类型。
2. 选择目标平台
   在 [Targets] （目标）区域中，必须选择激活本调试器时所用会话文件的目标。
   选中目标平台所对应的框。（根据需要选择其他目标。）

然后单击 [Next] （下一步）按钮。

3.2.2.3 步骤 3：设置配置文件名

为选择的所有目标逐个设置配置文件名。

配置文件保存 HEW 的状态 （目标仿真器、模拟器除外）。



默认名称已设置。如果不需要更改，请单击 [next] （下一步）按钮 （使用默认名称）。

仿真器设置到此结束。

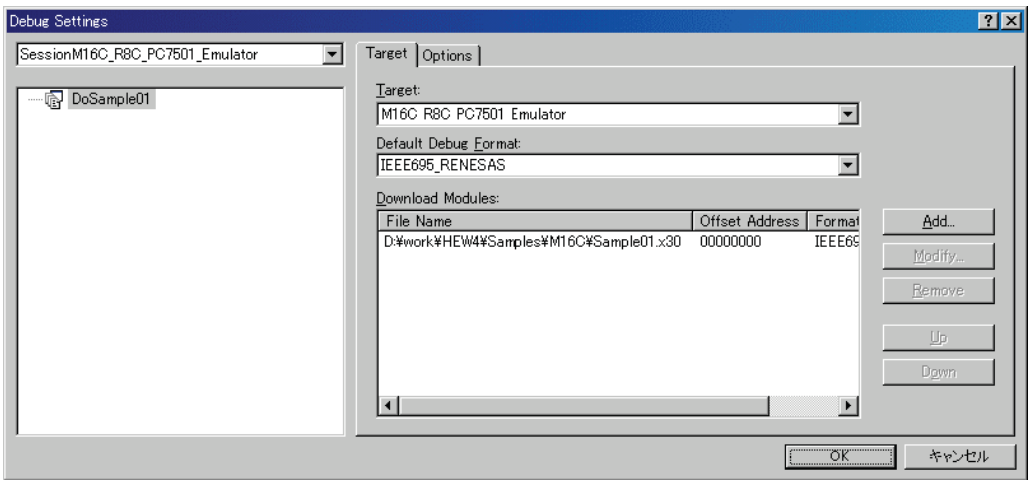
按照屏幕上的指示退出工程生成器。

此时还会显示调试器设置对话框。如果完成了仿真器的准备工作，请在该对话框中设置调试器并连接仿真器。

3.2.2.4　　步骤 4：注册要下载的加载模块

最后，注册要使用的加载模块文件。

从 [Debug] （调试）菜单中选择 [Debug Settings...] （调试设置…），以打开 [Debug Settings] （调试设置）对话框。

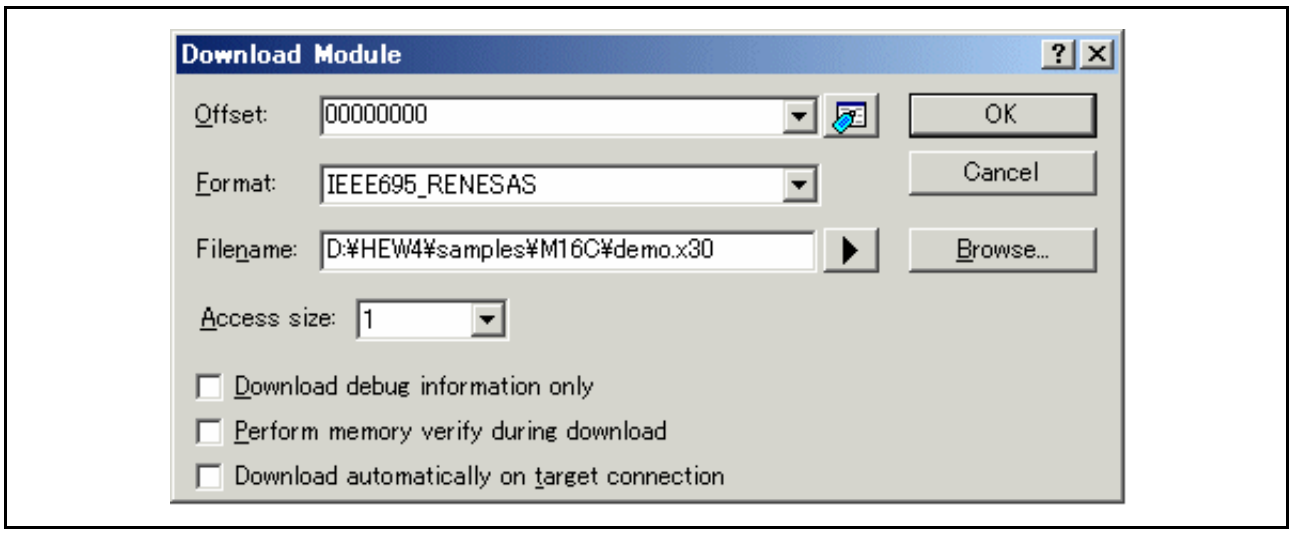


1. 在 [Target] （目标）下拉式列表框中选择要连接的产品名称。
2. 在 [Default Debug Format] （默认调试格式）下拉式列表框中选择要下载的加载模块的格式。

格式名称	内容
Intel-Hex+Sym	Intel Hex 格式文件和 Symbol 格式文件 （使用 SRA74 时）
IEEE695_ICC740	IEEE-695 格式文件 （使用 ICC740 时）

该调试器不支持目标格式，这些格式未显示在下拉式列表中。

3. 然后在 [Download Modules]（下载模块）列表框中注册相应的下载模块。
可以在使用 [Add...]（添加…）按钮打开的对话框中指定下载模块。



— 在 [Offset]（偏移）编辑框中输入加载下载模块的偏移位置。
— 在 [Format]（格式）编辑框中选择下载模块的格式。有关下载模块的格式名称，请参阅上面的表格。
— 在 [Filename]（文件名）编辑框中输入下载模块的完整路径和文件名。
— 在 [Access size]（存取大小）列表框中指定当前下载模块的存取大小。

然后单击 [OK]（确定）按钮。

注意事项

忽略 “Access size”（存取大小）和 “Perform memory verify during download”（下载期间执行存储器检验）。

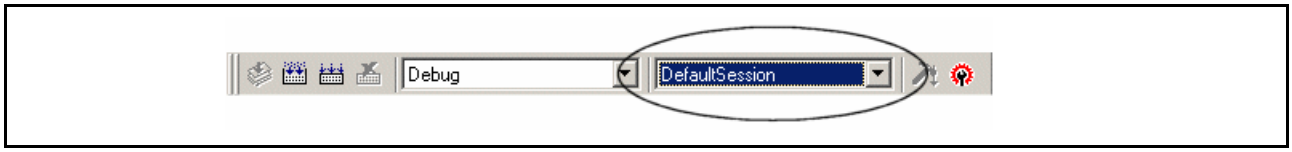
存取大小总是设置为 1，且检验功能不起作用。

3.3　启动调试器

通过与仿真器相连接，可以启动调试器。

3.3.1　连接仿真器

只需将会话文件转换为已在其中注册仿真器使用设置的文件，即可连接仿真器。
默认情况下，会创建会话文件。会话文件含有关于创建工程时所选目标的信息。
在以下工具栏上圈定的列表框中，选择包括要连接的目标字符串的会话名称。



选择会话名称之后，会显示设置调试器的对话框，并且将连接仿真器。

3.3.2　结束仿真器

使用以下方法，可以退出仿真器：

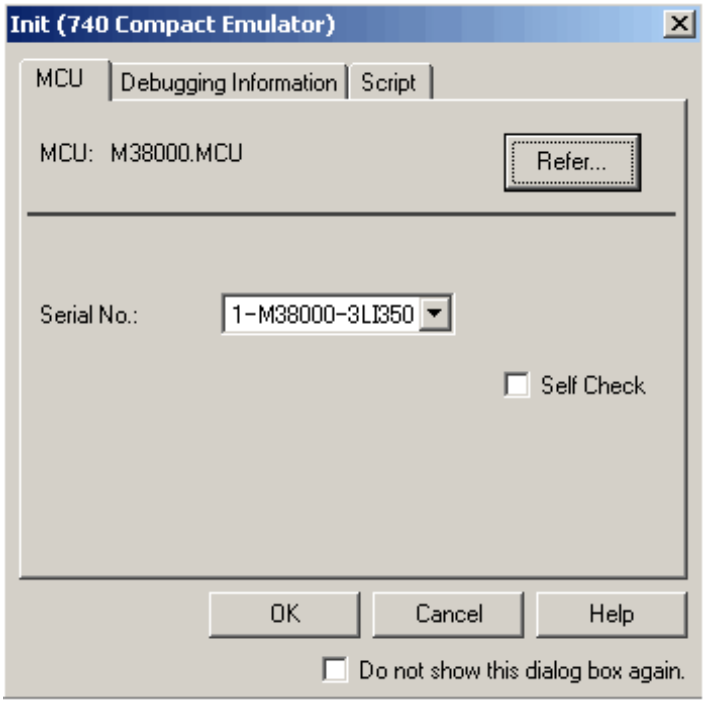
1. 选择 “DefaultSession”（默认会话）
   在列表框中选择连接仿真器时使用的 “DefaultSession”（默认会话）。
2. 退出 HEW
   从 [File] （文件）菜单中选择 [Exit] （退出）。此时会结束 HEW。

退出仿真器时，会显示信息框，询问是否要保存会话。如果需要保存会话，请单击 [Yes] （是）按钮。如果不需要，请单击 [No] （否）按钮。

4 设置调试器

4.1 Init （初始化）对话框

[Init]（初始化）对话框用于设置调试器启动时需要设置的项目。从此对话框设置的内容在调试器下次启动时也是有效的。在此对话框中设置的数据在下次启动时保持有效。

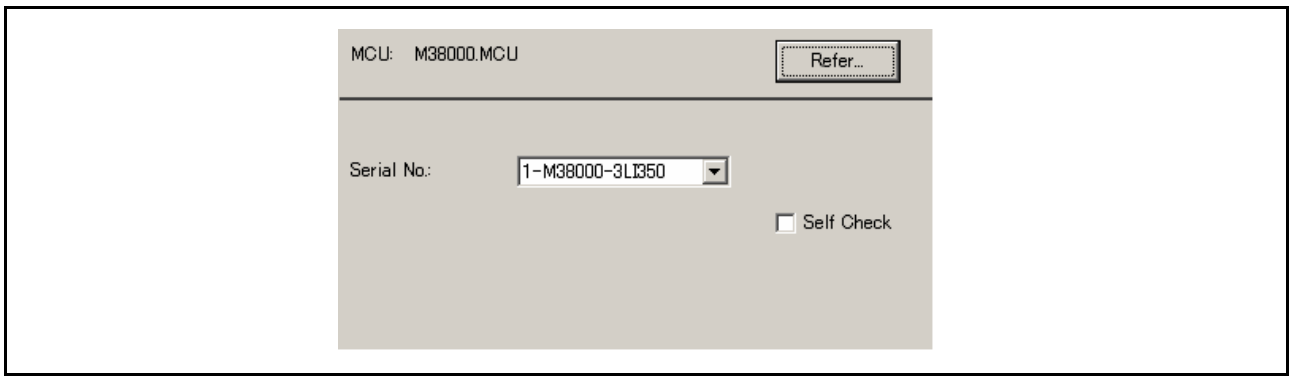


可以使用下列方法中的任意一种来打开 [Init]（初始化）对话框：

- 调试器启动以后，在菜单内选择：[Setup]（设置）→[Emulator]（仿真器）→[System...]（系统…）。
- 在按住 Ctrl 键的同时启动调试器。

4.1.1 MCU 选项卡

指定的内容在下次启动时生效。



4.1.1.1 指定 MCU 文件

单击 [Refer] （引用）按钮。

此时会打开 [File Selection] （文件选择）对话框。指定相应的 MCU 文件。

- MCU 文件包含特定于目标 MCU 的信息。
- 指定的 MCU 文件显示在 [MCU] 选项卡的 [MCU] 区域中。

如果对应的 MCU 文件未包含在调试器 / 仿真主机中，则必须创建新的 MCU 文件。为此，请参阅以下内容：

“4.4.1 创建 MCU 文件的方法 （适用于 740 调试器）”

4.1.1.1.1 设置通信接口

显示的数据因指定的通信接口而异。

可用通信接口因产品而异。

下面显示各个通信接口的设置。

有关详细信息，请参阅 “4.2 设置通信接口”

4.1.1.1.2 执行自检

指定此选项可以在调试器启动时在仿真器上执行自检 *。

仅当要在启动时执行自检时，才有必要选中上面的复选框。出现以下情况时，请指定此选项：

- 无法下载固件
- 虽然成功下载了固件，但调试器无法启动
- MCU 出现异常或跟踪结果有错误，并且希望检查仿真器是否正常运行

选中复选框以关闭 [Init] （初始化）对话框。连接到仿真器并确认固件以后，调试器会在仿真器上立即启动自检。（自检大约花费 30 秒到 1 分钟时间。）

如果在此自检过程中发现错误，则调试器会显示错误的内容并结束。

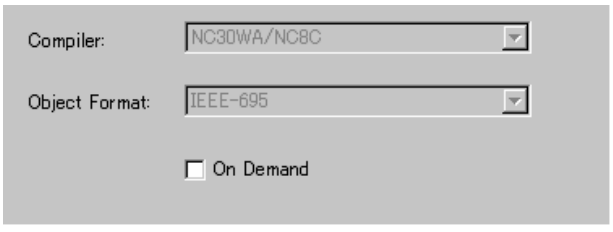
自检正常结束时，会显示下面所示的对话框。单击 [OK] （确定）以后，调试器会在该状态下直接启动。

仅当调试器启动时此项指定才会生效。

* 自检功能将会检查仿真器内部电路板的存储器情况等等。有关自检功能的详细信息，请参阅仿真器的用户手册。

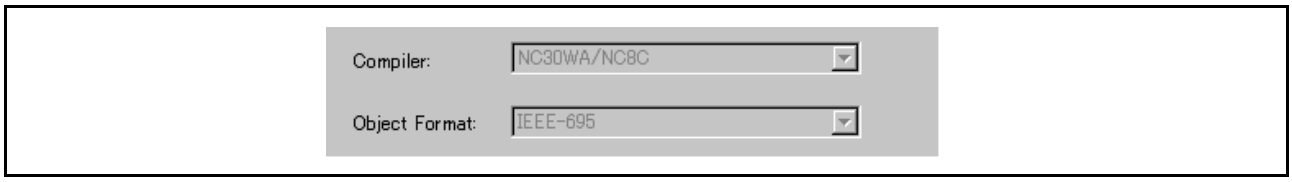
4.1.2 Debugging Information （调试信息）选项卡

指定的内容在下次下载时生效。



4.1.2.1 显示使用的编译器及其目标格式

显示使用的编译器及其目标文件格式。



请在通过菜单 [Debug] （调试）→ [Debug Settings...] （调试设置…）打开的对话框中，指定使用的编译器及其目标文件格式。

4.1.2.2 指定调试信息的存储

调试信息的存储方法有两种：on-memory 和 on-demand。
选择这两种方法中的一种。（默认情况下，选择 on-memory 方法。）
若要选择 on-demand 方法，请单击 [On Demand] （按需）复选框。

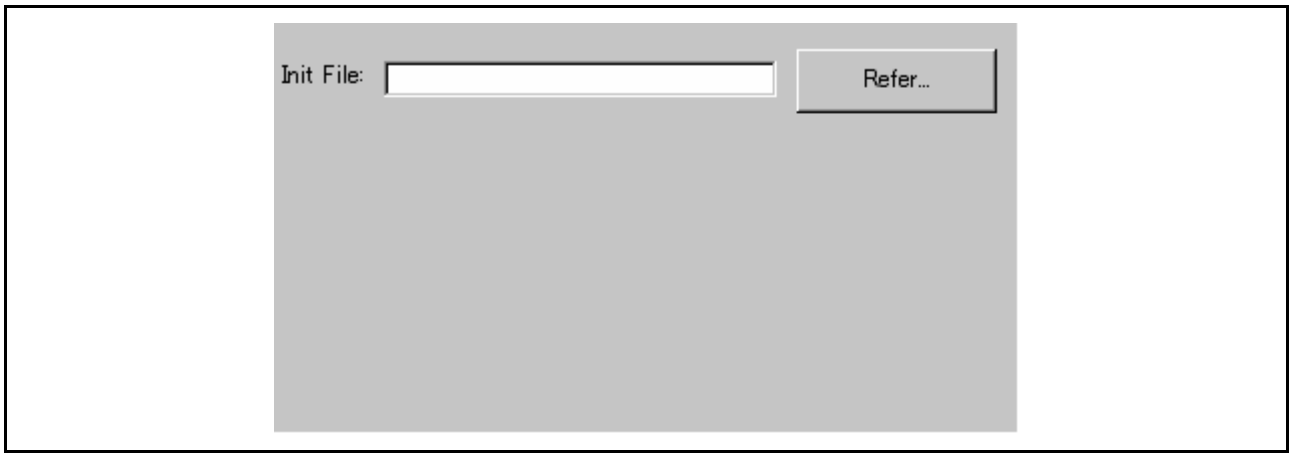
- on-memory 方法
  调试信息存储在计算机的内部存储器中。
  这个方法适用于规模小的加载模块 （目标程序）。
- on-demand 方法
  调试信息存储在计算机硬盘上的可重用临时文件中。
  由于存储的调试信息被重复使用，所以当再次下载相同的加载模块时，将能快速下载。
  这个方法适用于规模较大的加载模块 （目标程序）。

注意

- 如果加载模块很大，则 on-demand 方法的效率可能较低，因为需要大量时间来进行下载。在这种情况下，请选择 on-demand 方法。
- 使用 on-demand 方法时，将在包含已下载的加载模块的文件夹中创建一个文件夹，用于存储可重用的临时文件。此文件夹的命名方法是在加载模块名称之前加上文字 “~INDEX_”。例如，如果加载模块名称为 “sample.abs”，则文件夹名称为 “~INDEX_sample”。即使在退出调试器以后，此文件夹也不会被删除。

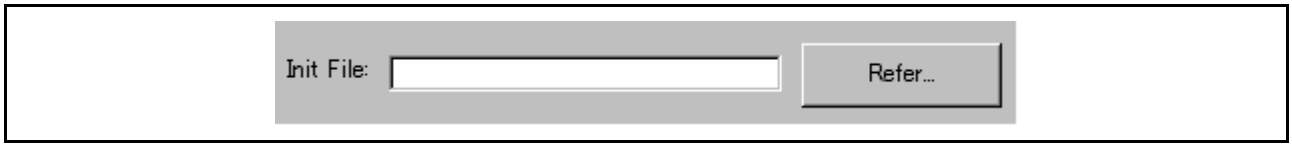
4.1.3 Script （脚本）选项卡

指定的内容在下次启动时生效。



4.1.3.1 自动执行脚本命令

若要在调试器启动时自动执行脚本命令，请单击 [Refer] （引用）按钮，指定要执行的脚本文件。



单击 [Refer] （引用）按钮后，会打开 [File Selection] （文件选择）对话框。
指定的脚本文件会显示在 “Init File:”（初始化文件：）字段中。
若要禁止自动执行脚本命令，请擦除 “Init File:”（初始化文件：）字段中显示的字符串。

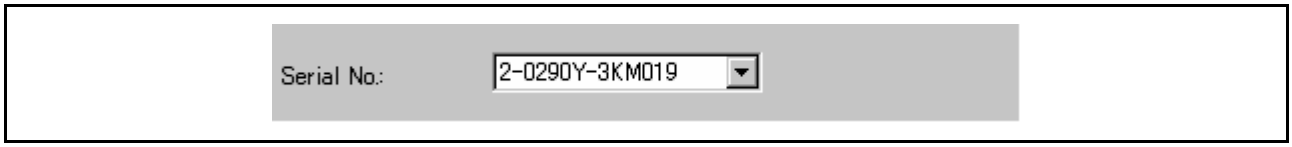
4.2 设置通信接口

4.2.1 设置 USB 接口

USB 通信使用个人计算机的 USB 接口。它符合 USB 1.1 标准。

执行 USB 通信之前，计算机必须已安装专用设备驱动程序。有关如何安装 USB 设备驱动程序的详细信息，请参阅 “2.3.1.1 安装 USB 设备驱动程序”。

当前 USB 连接的仿真器列在 [Serial No.] （序列号）区域中。选择要连接的仿真器的序列号。



4.3　设置 740 使用的调试器

4.3.1　Map 命令

必须使用 Map 命令来改变存储器映像信息，以适合目标单片机的存储器空间。

区域	映像	注意
SFR	External	
RAM	External	
内部 ROM	Internal	
外部 ROM	External	“Memory Expansion Mode”（存储器扩展模式）、“Microprocessor Mode”（微处理器模式）

- Internal

  允许仿真器的内部资源。必须将内部 ROM 区域设置为 Internal，因为它总是使用仿真器的内部资源来仿真。如果存储器未分配在外部区域，则可以通过将内部区域设置为 Internal 使用仿真器的内部存储区。

- External

  允许仿真器外部资源 （包括内部 SFR 区域和 RAM 区域）。必须总是将内部 SFR 区域和内部 RAM 区域设置为 External。要将存储器分配给外部区域，请将该区域设置为 External。

在仿真器启动后，0h-3FFFh 的存储器映像属性为“External”（外部），4000h-FFFFh 的存储器映像属性为“Internal”（内部）。使用 MAP 命令可查找或改变存储器映像信息。从脚本窗口中执行 MAP 命令。

注意事项

[内部 ROM 区域分配给 4000h 前面的地址的情况]

如果目标 MCU 的内部 ROM 区域位于 4000h 前面的地址，请将此区域的映像改为 INTERNAL。

示例）

如果内部 ROM 区域的位置从 1080h 开始：

```
1080 to 3FFF → Internal
```

[关于特殊设置]

应总是将内部 SFR 区域和内部 RAM 区域设置为 External。但是，如果目标 MCU 的 RAM 区域大于仿真器 MCU 中包括的 RAM，请将该区域设置为 Internal。

示例）

如果仿真器 MCU 中包括的 RAM 区域是 40-1FF，目标 MCU 的内部 RAM 区域是 40-2FF

```
40 to 1FF → External
200 to 2FF → Internal
```

4.4 创建 MCU 文件的方法

4.4.1 创建 MCU 文件的方法 （适用于 740 调试器）

MCU 文件依次描述了以下内容。
有关 1-4 项的信息，请参阅所用 MCU 的数据手册。

1. 堆栈页选择位的值
2. CPU 模式寄存器的地址
3. 堆栈的结束地址*1
4. 复位向量的地址
5. POD 编号*2
6. 固件名称*3
7. MCU 信息编号*4

*1 堆栈的结束地址
指定将用作堆栈的区域的最后一个地址。考虑 CPU 模式寄存器中的堆栈页选择位的初始值。（堆栈页选择位的初始值取决于单片机。）对于将堆栈页选择位初始值设置为 “0” 的单片机，允许的指定范围是 0 页地址范围 （0h 到 FFh）。对于将堆栈页选择位初始值设置为 “1” 的单片机，允许的指定范围是 1 页地址范围 （100h 到 1FFh）。

*2POD 编号
在小型仿真器中，请设置 “0”。

*3 固件名称
在小型仿真器中，请设置 “M38000”。

*4MCU 信息编号
在小型仿真器中，请设置 “00”。

4.4.1.1 示例

```
2
3B
FF
FFFC
0
M38000
00
```

教程

5 教程

5.1 简介

本节通过一个教程程序对此调试器的主要功能进行介绍。该教程程序的安装目录为 \WorkSpace\Tutorial，位于已安装 HEW 的驱动器之上。每个目标和 MCU 均有各自的工作空间。请针对系统选择相应的工作空间，然后从 [Open Workspace...]（打开工作空间…）菜单打开工作空间文件 (*.hws)。

该教程程序基于对十个随机数据项进行升序或降序排序的 C 程序。

此教程程序执行下列操作：

- tutorial 函数生成要排序的随机数据。
- sort 函数按升序对生成的随机数据进行排序。
- 然后 change 函数按降序对这些数据进行排序。

注意

重新编译后的地址可能与本章所述的地址不同。

使用用于 740 族的汇编器套件时

740 族汇编器套件的教程程序已准备好。如果使用 740 族的汇编器套件，请使用它

- 请在阅读本教程时用子例程名称替换函数名称。（例如，用 “subroutine sort”替换 “function sort()”）
- 对于源文件名，也请用相应的名称替换它。
- 本教程中的图是针对 C 程序的。汇编器程序的图可能会有所不同。
- 步骤 9 和步骤 12 是 C 程序的说明。

5.2 使用

请遵照以下说明进行操作：

5.2.1 步骤 1：启动调试器

5.2.1.1 使用前的准备工作

要运行 HEW 并连接到仿真器，请参阅 “3 使用前的准备工作”。

5.2.1.2 设置调试器

如果调试器与仿真器相连，则会显示设置调试器的对话框。请在此对话框中设置调试器。
要在此对话框中设置调试器，请参阅 “4 设置调试器”。
调试器设置完成后，即可实现调试器的所有功能。

5.2.2 步骤 2：检查 RAM 的运作情况

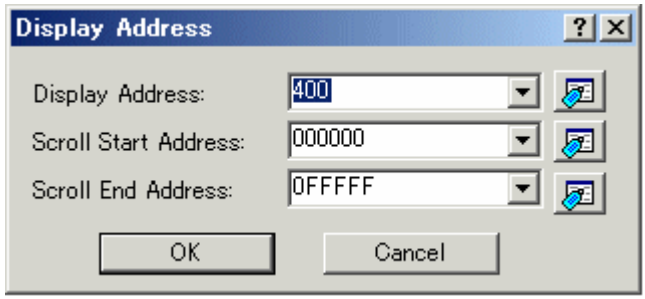
检查 RAM 是否正常运行。在 [Memory]（存储器）窗口中显示和编辑存储器所存储的内容，从而检查存储器是否正常运行。

注意

在某些单片机上，存储器可为板载安装方式。在这种情况下，通过上述方式可能无法检查存储器的工作情况。建议创建一个检查存储器的程序。

5.2.2.1 检查 RAM 的运作情况

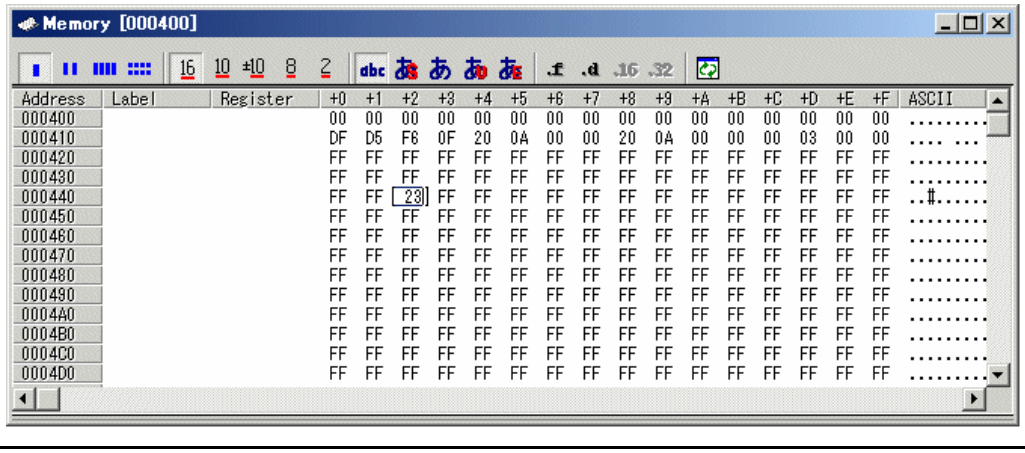
从 [View]（视图）菜单的 [CPU] 子菜单中选择 [Memory]（存储器），然后在 [Display Address]（显示地址）编辑框输入 RAM 地址 （此处输入 H'400）。在 [Scroll Start Address]（滚动起始地址）和 [Scroll End Address]（滚动结束地址）编辑框中保留默认设置。（默认情况下，滚动范围为 0h 至 MCU 的最大地址。）



注意

RAM 区域的设置因产品而异。有关详细信息，请参阅硬件手册。

单击 [OK]（确定）按钮。随即显示 [Memory]（存储器）窗口和指定的存储器区域。



如果将鼠标光标放在 [Memory]（存储器）窗口中的一个数据显示点上并双击，则可以更改该点的值。

5.2.3 步骤 3：下载教程程序

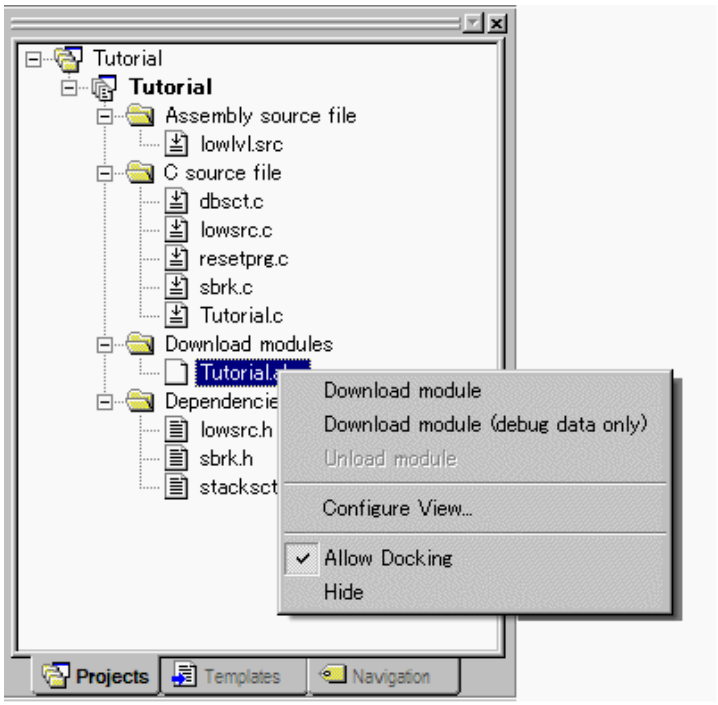
5.2.3.1 下载教程程序

下载要调试的目标程序。下载文件和下载地址取决于所用的目标 MCU。请使用与目标 MCU 相应的图像和地址替换屏幕图像和地址。

- 用于 740 的调试器

  如果对 740 族使用 C 编译器套件，请在 [Download modules] （下载模块）下，从 [Tutorial.695] 选择 [Download module] （下载模块）。

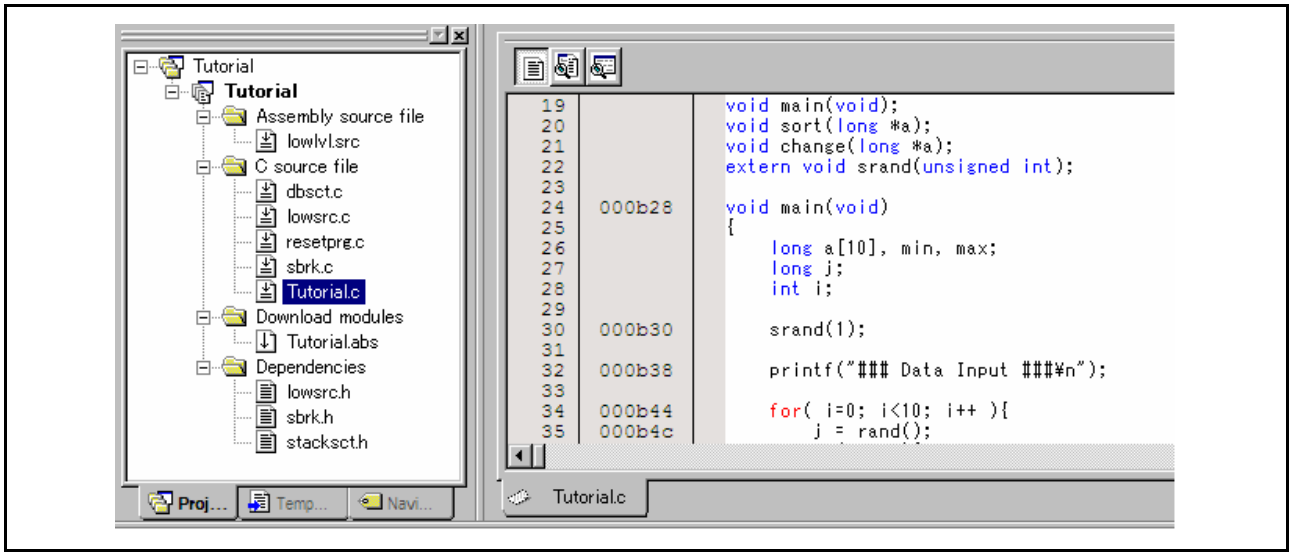
  如果对 740 族使用汇编器套件，请在 [Download modules] （下载模块）下，从 [Tutorial.hex] 选择 [Download module] （下载模块）。



5.2.3.2 显示源程序

使用此调试器，用户可以执行源码级用户程序调试。

双击 [C source file] （C 源文件）下的 [tutorial.c]。此时会打开 [Editor (Source)] （编辑器 （源码））窗口，其中显示 “Tutorial.c”文件的内容。



如果需要，从 [Setup] （设置）菜单选择 [Format Views...] （格式视图…）选项，设置字体和字号以改善显示效果。

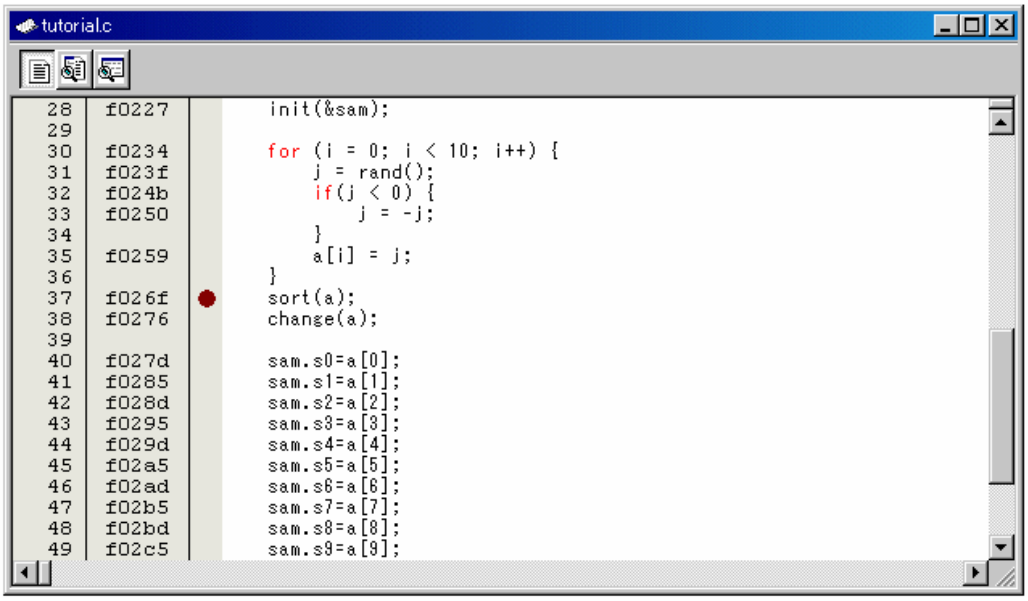
[Editor (Source)] （编辑器 （源码））窗口初始显示用户程序的起始部分，但用户可以使用滚动条查看用户程序中的其他语句。

5.2.4 步骤 4：设置断点

设置软件断点是一项基本调试功能。
通过 [Editor (Source)] （编辑器 （源码））窗口，可在程序中的任意位置轻松设置软件断点。

5.2.4.1 设置软件断点

例如，要在 sort 函数调用处设置一个软件断点，请执行以下操作：
在调用 sort 函数的这一行，双击 [S/W breakpoints] （软件断点）列。



调用 sort 函数的行将显示红色符号。这表示已设置了一个软件断点。

5.2.5 步骤 5：执行程序

按以下过程执行程序：

5.2.5.1 复位 CPU

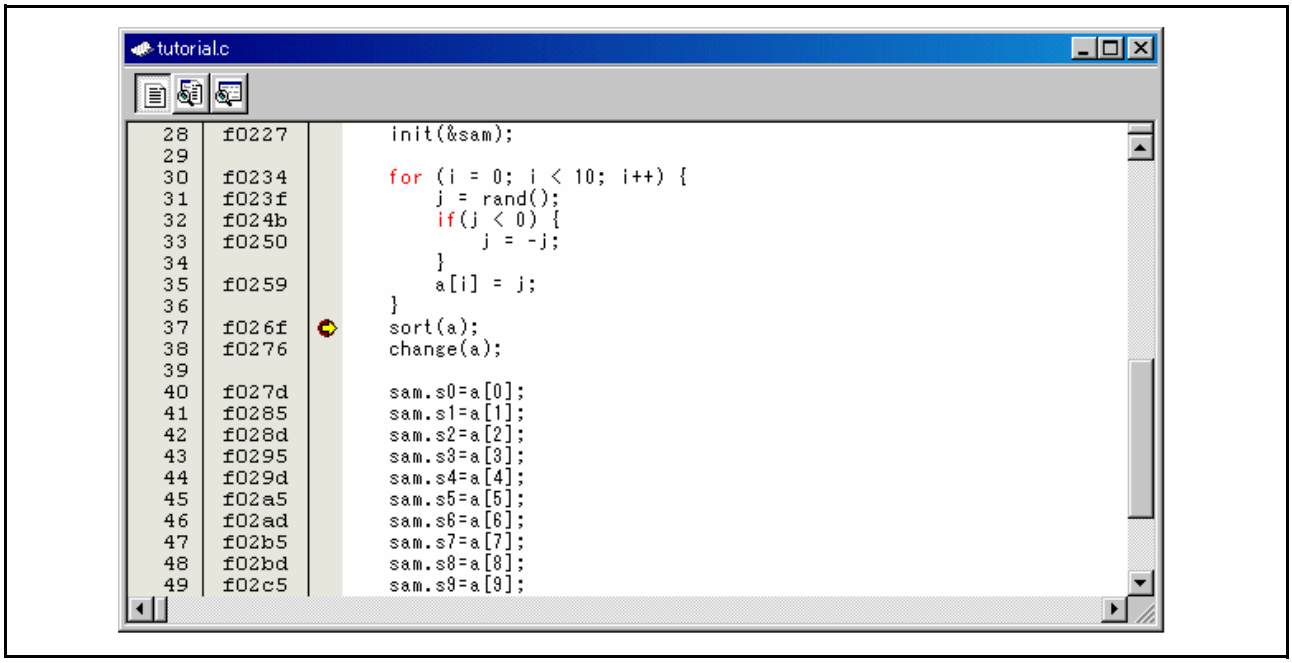
默认情况下，在下载程序后不复位 CPU。

要将 CPU 复位，请从 [Debug]（调试）菜单选择 [Reset CPU]（复位 CPU），或者单击工具栏上的 [Reset CPU]（复位 CPU）按钮  。

5.2.5.2 执行程序

要执行程序，请从 [Debug]（调试）菜单选择 [Go]（执行），或者单击工具栏上的 [Go]（执行）按钮  。

程序将执行至已设置的断点位置， [S/W Breakpoints]（软件断点）列中显示一个箭头，指示程序暂停的位置。

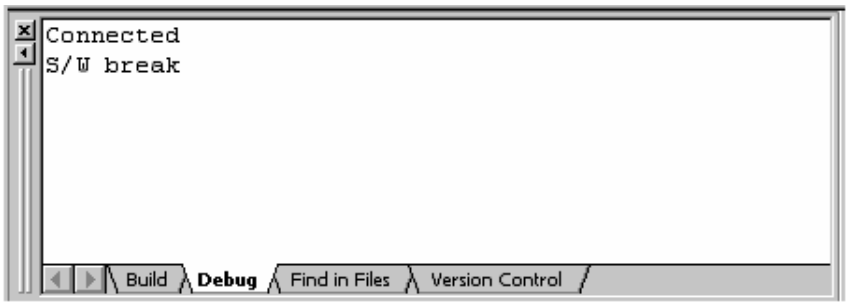


注意

暂停之后显示源文件时，可能会查询源文件的路径。如遇到这种情况，请指定源文件的位置。

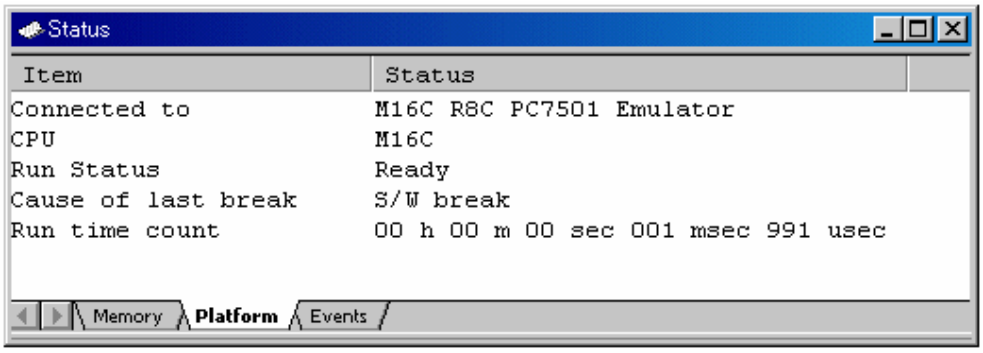
5.2.5.3 查看中断原因

中断原因显示于 [Output] （输出）窗口中。



用户也可以在 [Status] （状态）窗口中查看上一次出现中断的原因。

从 [View] （视图）菜单的 [CPU] 子菜单中选择 [Status] （状态）。显示 [Status] （状态）窗口后，打开 [Platform] （平台）页，检查 Cause of last break （上次中断的原因）的 Status （状态）。



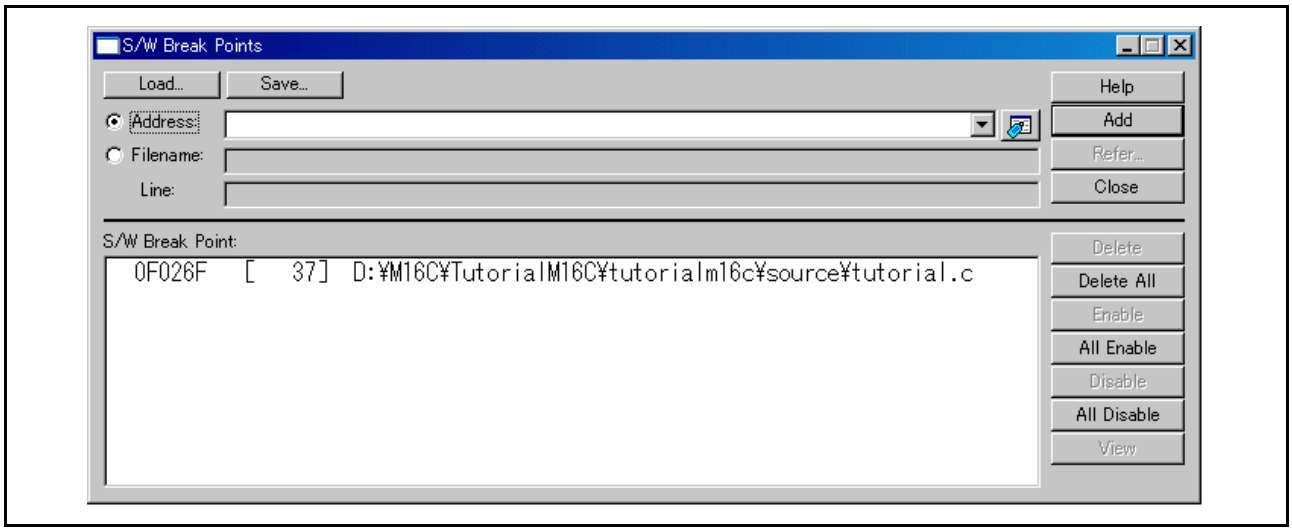
请参阅 “10 显示程序停止的原因”，了解中断原因的表示法。

5.2.6 步骤 6：查看断点

用户可在 [S/W Break Points] （软件断点）窗口中看到程序中设置的所有断点。

5.2.6.1 查看断点

从 [View] （视图）菜单的 [Break] （断点）子菜单中选择 [S/W Break Points] （软件断点）。将显示 [S/W Break Points] （软件断点）窗口。



用户可通过此窗口设置或更改断点，定义新断点，以及删除、允许或禁止断点。

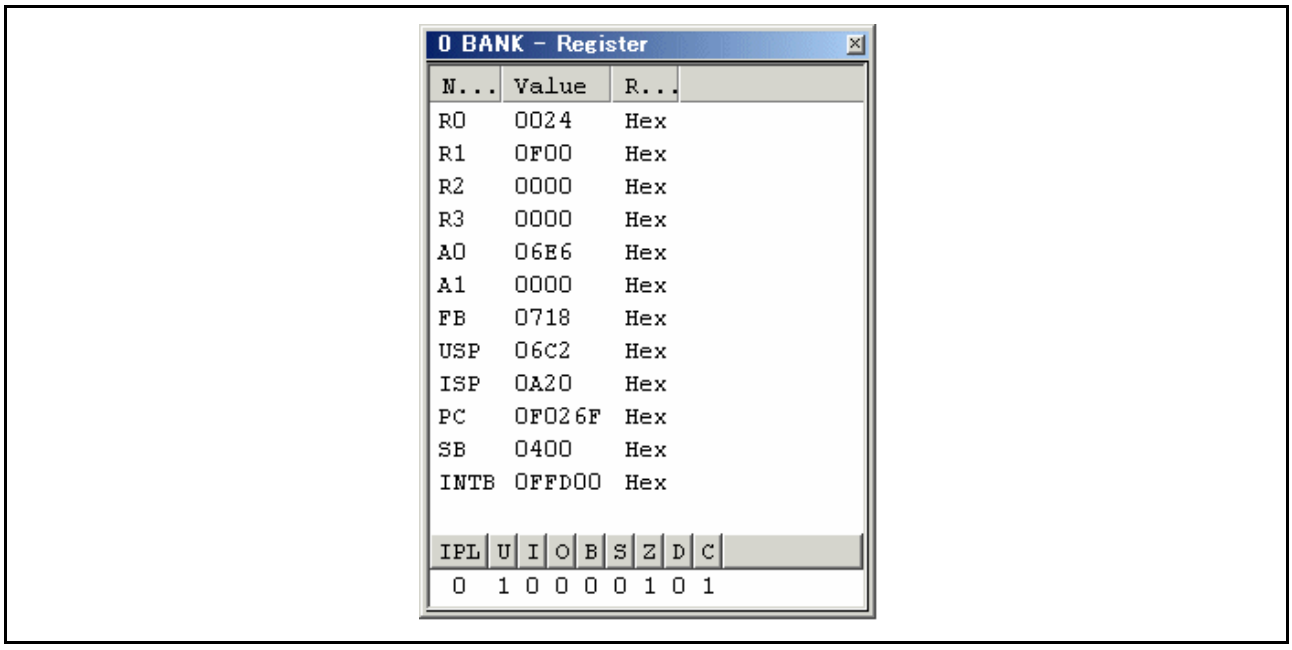
5.2.7 步骤 7：查看寄存器

用户可以在 [Register] （寄存器）窗口中看到所有寄存器 / 标志值。

5.2.7.1 查看寄存器

从 [View] （视图）菜单的 [CPU] 子菜单中选择 [Registers] （寄存器）。此时显示 [Register] （寄存器）窗口。

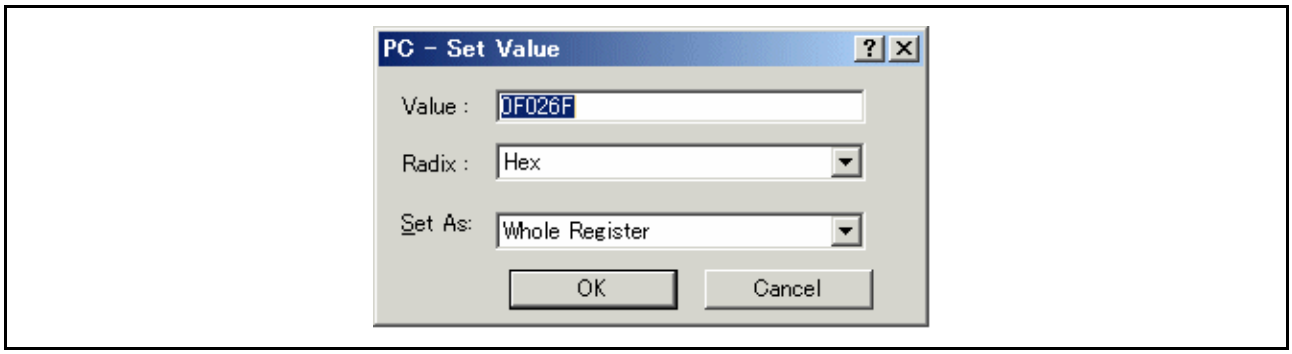
下图所示为用于 M16C/R8C 的调试器的 [Register] （寄存器）窗口。



5.2.7.2 设置寄存器值

可以从此窗口更改寄存器 / 标志值。

双击相关行以更改其寄存器。此时会打开对话框。输入要更改的值。



5.2.8 步骤 8：查看存储器

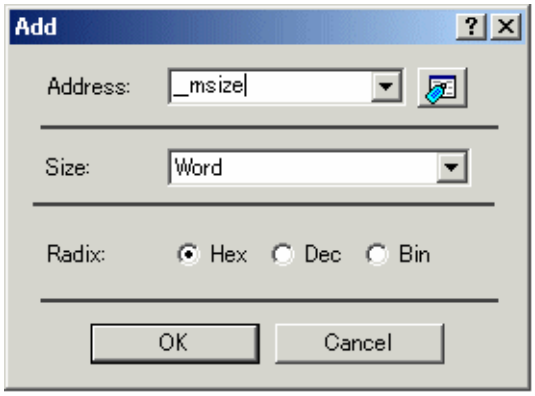
指定标签名称后，用户可以在 [ASM Watch] （ASM 监视）窗口中查看该标签已注册的存储内容。

5.2.8.1 查看存储器

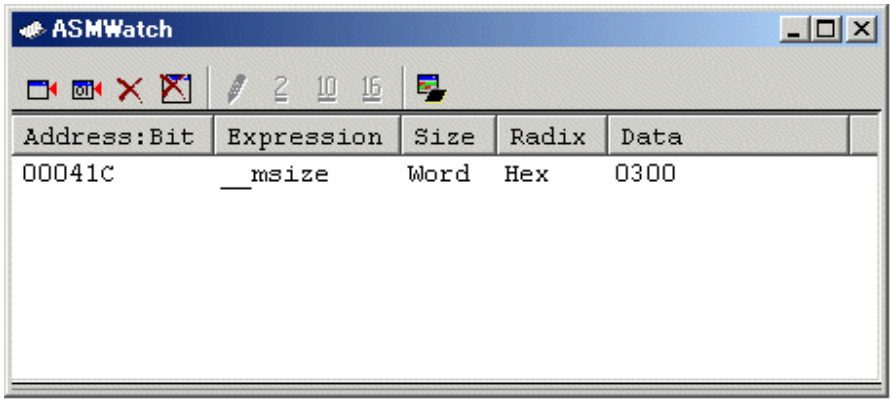
例如，要查看与 __msize 对应的、字大小的存储内容：

从 [View] （视图）菜单的 [Symbol] （符号）子菜单中选择 [ASM Watch] （ASM 监视），打开 [ASM Watch] （ASM 监视）窗口。

随后用鼠标右键单击 [ASM Watch] （ASM 监视）窗口，从弹出菜单中选择 [Add...] （添加…），在 [Address] （地址）编辑框中输入 __msize，并在 [Size] （大小）组合框中设置 “Word”（字）。



单击 [OK] （确定）按钮。此时的 [ASM Watch] （ASM 监视）窗口会显示所指定的存储区域。



5.2.9 步骤 9：监视变量

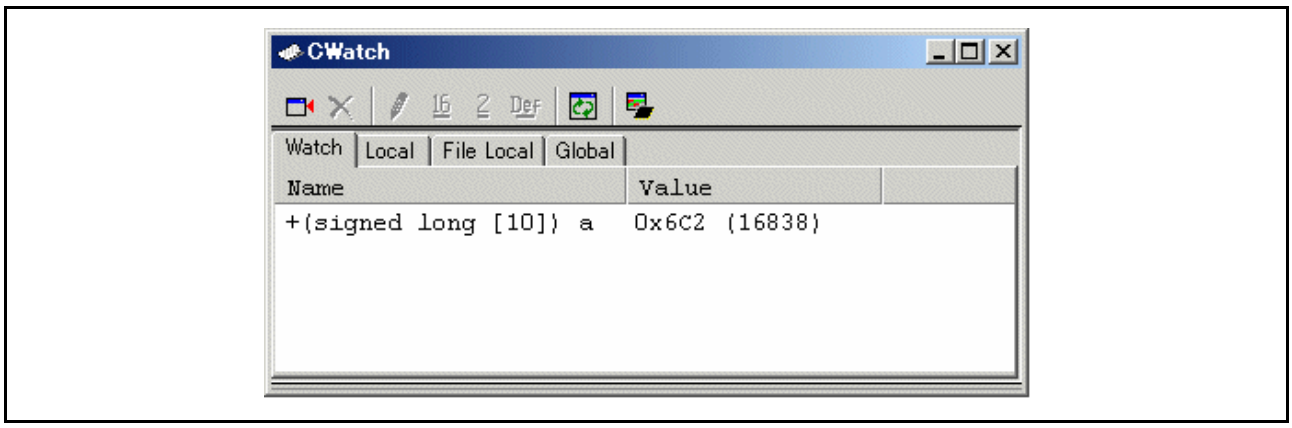
当用户逐步执行程序时，可以看到用户程序中所用变量的值的变化。

如果下载的程序是用于 740 族的汇编器套件生成的程序，则不能在 [C watch] （C 监视）窗口中监视变量。

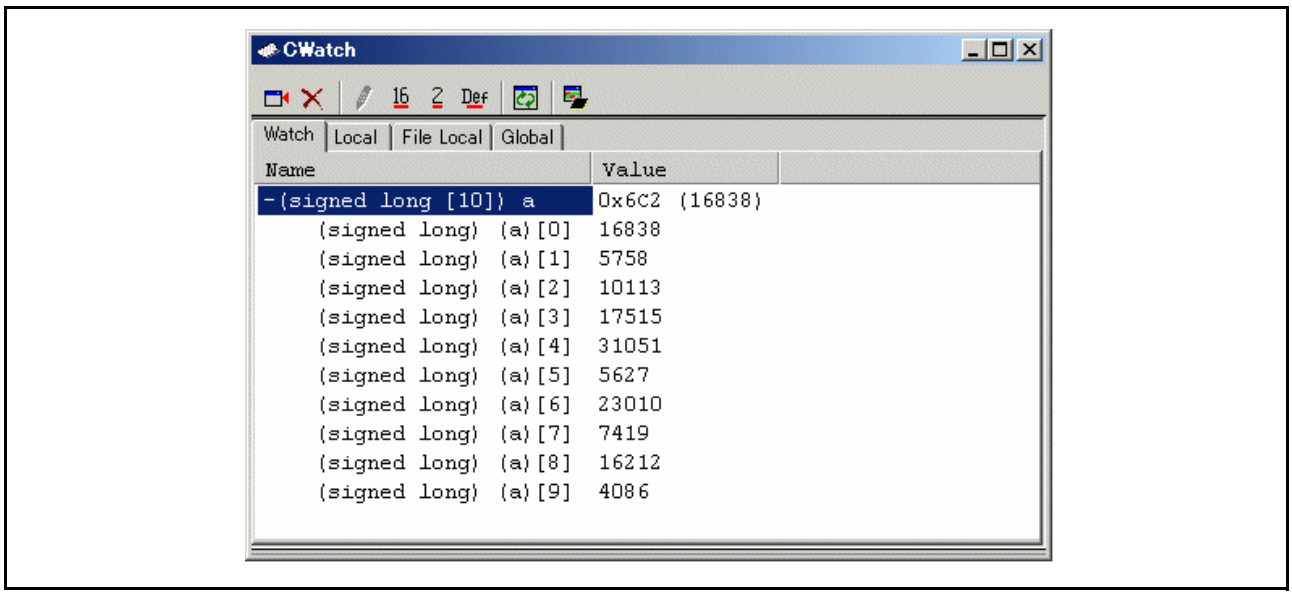
5.2.9.1 监视变量

例如，对于程序起始处声明的 long 类型的数组 a，可以通过以下步骤设置对其的监视：

在 [Editor (Source)] （编辑器 （源码））窗口中，单击显示的数组 a 的左侧以定位光标，然后从鼠标右键菜单中选择 [Add C Watch...] （添加 C 监视…）。此时会打开 [C watch] （C 监视）窗口的 [Watch] （监视）选项卡，其中显示变量。



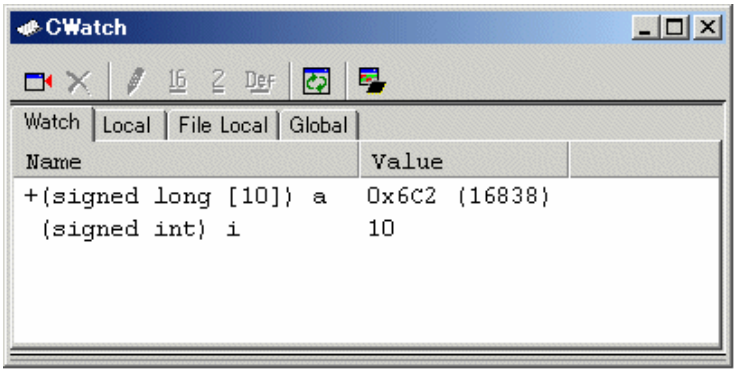
在 [C watch] （C 监视）窗口中，用户可以单击数组 a 左侧的 “+”标记来监视所有元素。



5.2.9.2 注册变量

用户可以通过指定变量的名称将其添加到 [C watch] （C 监视）窗口。
鼠标右键单击 [C Watch] （C 监视）窗口，并从弹出菜单中选择 [Add...] （添加…）。
此时显示以下对话框。输入变量 i。

单击 [OK] （确定）按钮。现在 [C Watch] （C 监视）窗口会显示 int 类型的变量 i。



5.2.10 步骤 10：逐步执行程序

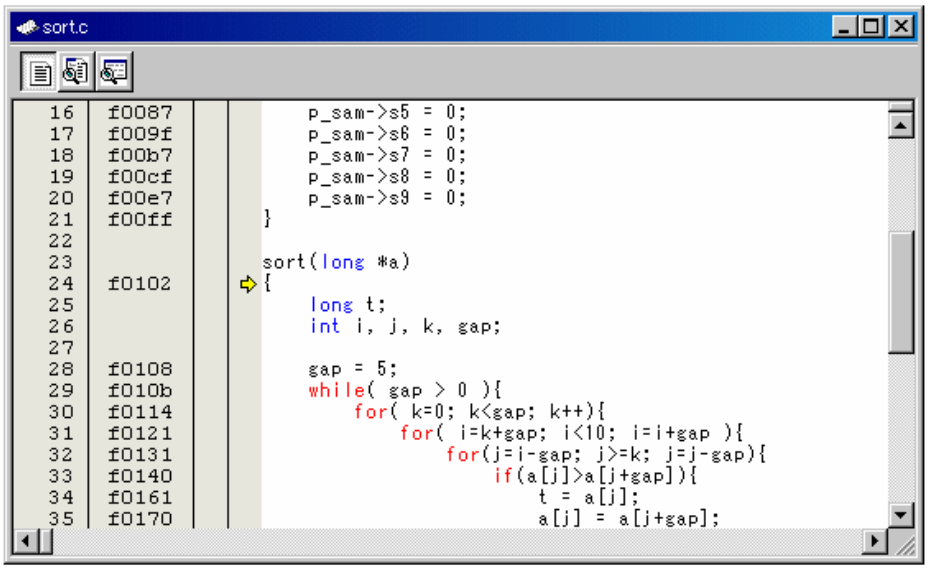
此调试器提供多个逐步执行菜单命令，从而实现高效率的程序调试。

1. [Step In] （跳入）
   执行每个语句，包括函数 （子例程）内的语句。
2. [Step Out] （跳出）
   跳出函数 （子例程），并在程序中调用该函数 （子例程）的语句的下一句处停止。
3. [Step Over] （跳过）
   在单步中执行函数 （子例程）调用。
4. [Step...] （单步…）
   以指定的速率和次数重复单步执行。

5.2.10.1 执行 [Step In] （跳入）命令

[Step In] （跳入）命令跳入被调用的函数 （子例程），并在该函数 （子例程）的第一个语句处停止。
要逐步执行 sort 函数，请从 [Debug] （调试）菜单选择 [Step In] （跳入），或者单击工具栏上的 [Step In] （跳入）按钮  。

在 [Editor (Source)] （编辑器 （源码））窗口中， PC 光标将移动到 sort 函数的第一个语句。

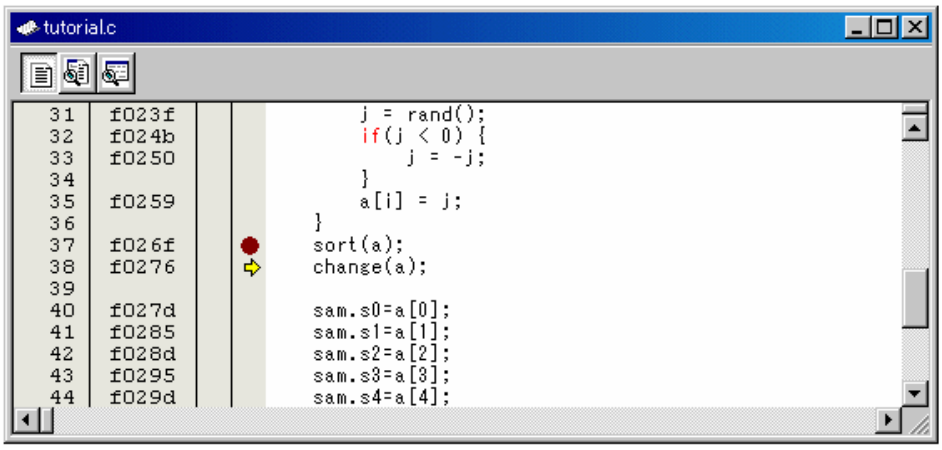


5.2.10.2 执行 [Step Out]（跳出）命令

[Step Out]（跳出）命令跳出所调用的函数（子例程），并在主程序中调用语句的下一语句处停止。

要跳出 sort 函数，请从 [Debug]（调试）菜单选择 [Step Out]（跳出），或者单击工具栏上的 [Step Out]（跳出）按钮  。

PC 光标将移出 sort 函数，并移动到 change 函数前方的位置。



注意

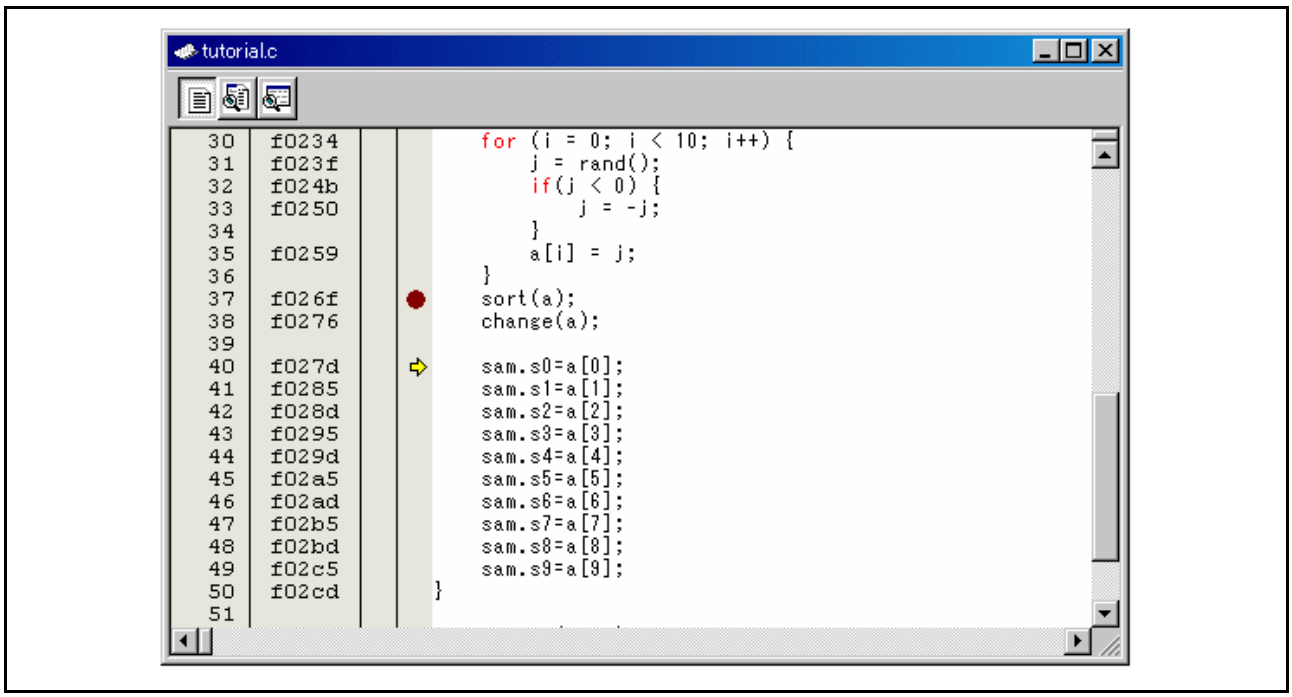
执行此函数需要一些时间。查明调用源后，使用 [Go To Cursor]（转至光标）。

5.2.10.3 执行 [Step Over]（跳过）命令

[Step Over]（跳过）命令将函数（子例程）调用作为单步执行，并在主程序的下一语句处停止。

要一次性逐步执行 change 函数中的所有语句，请从 [Debug]（调试）菜单选择 [Step Over]（跳过），或者单击工具栏上的 [Step Over]（跳过）按钮  。

PC 光标将移动到与 change 函数相邻的位置。



5.2.11 步骤 11：强制暂停程序执行

此调试器可以强制暂停程序执行。

5.2.11.1 强制暂停程序执行

取消所有暂停。

要执行主函数的其他部分，请从 [Debug]（调试）菜单选择 [Go]（执行），或者单击工具栏上的 [Go]（执行）按钮  。

程序进入死循环。要在执行时强制暂停，请从 [Debug]（调试）菜单选择 [Halt Program]（暂停程序），或者单击工具栏上的 [Halt]（暂停）按钮  。

5.2.12 步骤 12：显示局部变量

用户可以通过 [C Watch] （C 监视）窗口显示函数中的局部变量。

如果下载的程序是用于 740 族的汇编器套件生成的程序，则不能在 [C watch] （C 监视）窗口中监视变量。

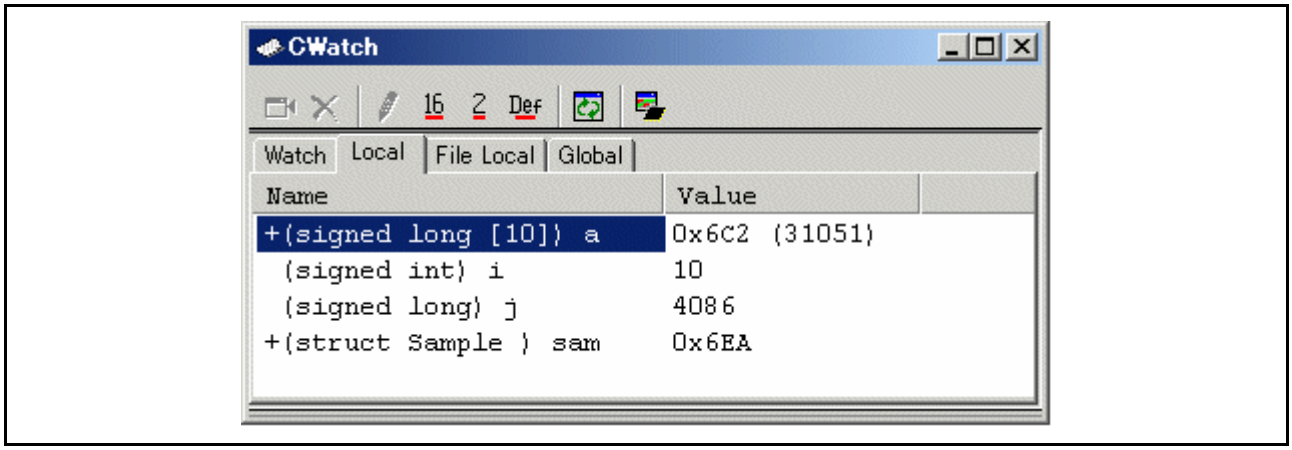
5.2.12.1 显示局部变量

例如，我们将检查 tutorial 函数中的局部变量，该函数声明四个局部变量：a、 j、 i 和 sam。

从 [View] （视图）菜单的 [Symbol] （符号）子菜单中选择 [C Watch] （C 监视）。此时显示 [C Watch] （C 监视）窗口。默认情况下， [C watch] （C 监视）窗口有四个选项卡，如下所示：

- [Watch] （监视）选项卡：
  仅显示用户已注册的变量。
- [Local] （局部变量）选项卡
  显示 PC 所在作用域可以引用的所有局部变量。如果作用域因程序的执行发生改变，则 [Local] （局部变量）选项卡的内容也会随之更改。
- [File Local] （文件局部变量）选项卡
  显示 PC 所在的文件作用域的所有文件局部变量。如果文件作用域因程序的执行发生改变，则 [File Local] （文件局部变量）选项卡的内容也会随之更改。
- [Global] （全局变量）选项卡
  显示已下载程序当前使用的所有全局变量。

显示局部变量时请选择 [Local] （局部变量）选项卡。



单击 [Locals] （局部变量）窗口中数组 a 左侧的 “+” 标记将显示元素。

用户在执行 sort 函数前后检查数组 a 的元素时，可以确认随机数据是按降序排序的。

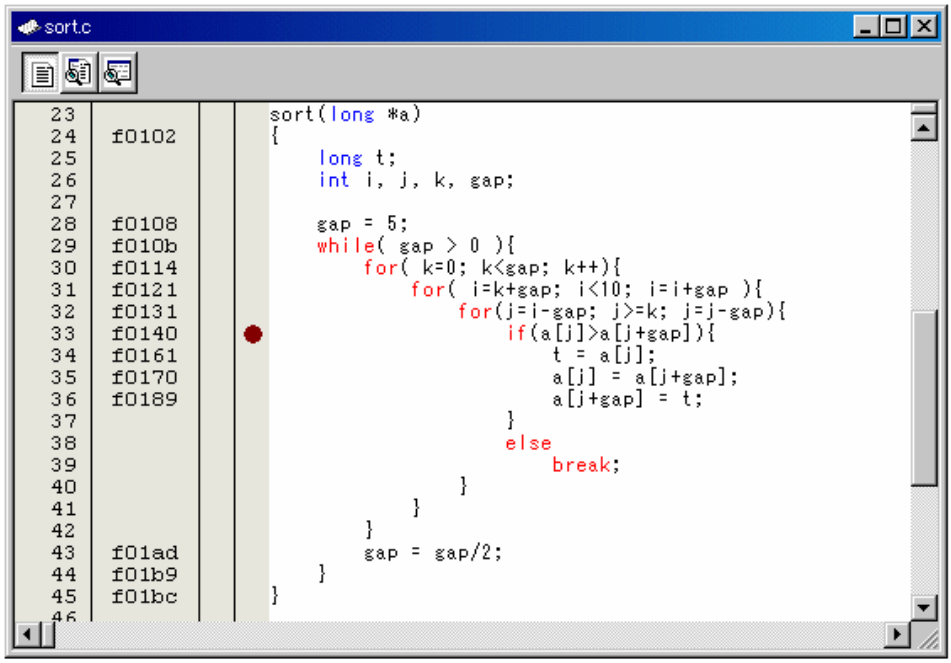
5.2.13 步骤 13：堆栈跟踪功能

调试器使用堆栈信息来显示调用序列中的函数名称，此调用序列最终到达程序计数器当前所指向的函数。

用于 740 的调试器不支持堆栈跟踪功能。

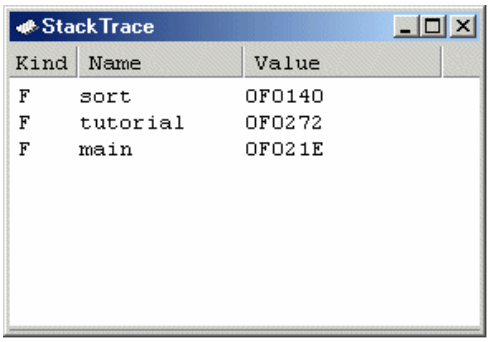
5.2.13.1 引用函数调用状态

在 sort 函数中，双击 [S/W Breakpoints] （软件断点）列设置一个软件断点。



要从复位向量地址执行用户程序，请从 [Debug] （调试）菜单选择 [Reset Go] （复位执行），或者单击工具栏上的 [Reset Go] （复位执行）按钮  。

程序执行暂停后，从 [View] （视图）菜单的 [Code] （代码）子菜单中选择 [Stack Trace] （堆栈跟踪）打开 [Stack Trace] （堆栈跟踪）窗口。



上图显示程序计数器当前位于 sort() 函数的选定行，而 sort() 函数从 tutorial() 函数中调用。

5.2.14 后续内容

此教程已对该调试器的使用进行了介绍。

使用仿真器提供的仿真功能可执行复杂调试。通过准确隔离和识别发生硬件和软件问题的条件，它可以有效调查此类问题。

参考

6 窗口 / 对话框

此调试器的窗口如下所示。
单击窗口名称后会显示参考内容。

窗口名称	[View]（视图）菜单
[RAM Monitor]（RAM 监控）窗口	[View]（视图）→ [CPU] → [RamMonitor]（RAM 监控）
[ASM Watch]（ASM 监视）窗口	[View]（视图）→ [Symbol]（符号）→ [ASMWatch]（ASM 监视）
[C Watch]（C 监视）窗口	[View]（视图）→ [Symbol]（符号）→ [CWatch]（C 监视）
[Script]（脚本）窗口	[View]（视图）→ [Script]（脚本）
[S/W Break Point Setting]（软件断点设置）窗口	[View]（视图）→ [Break]（断点）→ [S/W Break Points]（软件断点）
[H/W Break Point Setting]（硬件断点设置）对话框	[View]（视图）→ [Break]（断点）→ [H/W Break Points]（硬件断点）
[Trace Point Setting]（跟踪点设置）对话框	[View]（视图）→ [Trace]（跟踪）→ [Trace Points]（跟踪点）
[Trace]（跟踪）窗口	[View]（视图）→ [Trace]（跟踪）→ [Trace]（跟踪）
[GUI I/O] 窗口	[View]（视图）→ [Graphic]（图形）→ [GUI I/O]

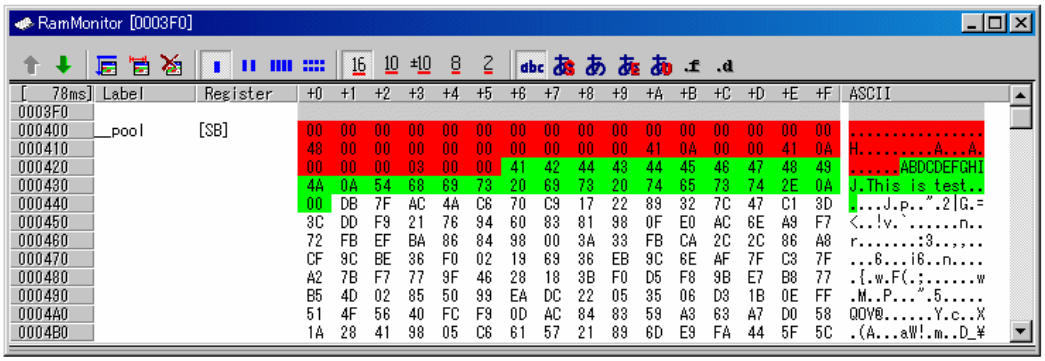
有关以下窗口的参考内容，请参阅 HEW 主部件的帮助文件。

- [Differences]（差异）窗口
- [Map]（映像）窗口
- [Command Line]（命令行）窗口
- [Workspace]（工作空间）窗口
- [Output]（输出）窗口
- [Disassembly]（反汇编）窗口
- [Memory]（存储器）窗口
- [IO] 窗口
- [Status]（状态）窗口
- [Register]（寄存器）窗口
- [Image]（图像）窗口
- [Waveform]（波形）窗口
- [Stack Trace]（堆栈跟踪）窗口

6.1　[RAM Monitor]（RAM 监控）窗口

目标程序运行时，[RAM monitor]（RAM 监控）窗口会显示存储器内容的变化。

使用实时 RAM 监控功能，相关存储器内容将通过转储形式显示在 RAM 监控区域。目标程序运行时，将按指定间隔（默认情况下为 100 ms）对所显示的内容进行更新。



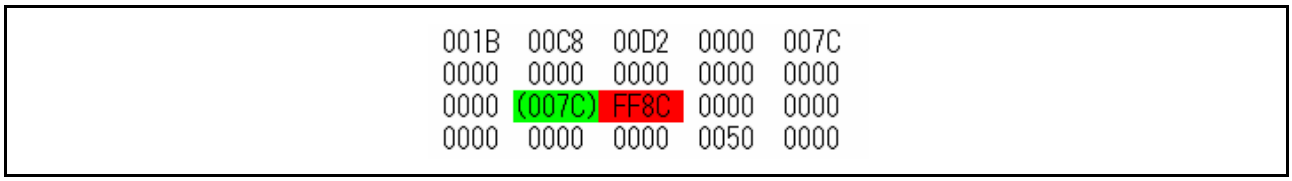
- 本系统提供一个 1KB 的 RAM 监控区域，该区域可以位于任何连续的地址上。
- RAM 监控区域可变换到任何需要的地址范围。
  有关如何更改 RAM 监控区域的详细信息，请参阅 “6.1.2 设置 RAM 监控区域”。
  默认 RAM 监控区域映射到一个 1 KB 大小的区域，该区域的起始地址为内部 RAM 的起始地址。
- 各个窗口可分别设置显示内容更新间隔。
  在地址显示区域的标题字段中，显示了目标程序运行时显示内容发生实际更新的实际更新间隔。
- 数据显示区域和代码显示区域的背景颜色由存取属性预先确定，如下所示。

存取属性	背景颜色
读地址	绿
写地址	红
非存取地址	白

背景颜色可以更改。

注意事项

- [RAM monitor]（RAM 监控）窗口显示通过总线存取的数据。因此，显示的数据不反映所做的更改，除非目标程序对这些数据进行了存取，例如，从外部 I/O 直接盖写存储器时。
- 如果 RAM 监控区域中的数据不以字节长度显示，则该数据中各字节可能具有不同的存储器存取属性。如果数据中的字节具有不同的存取属性（如在上述情况下），则这类数据在窗口中显示时将置于括号中。在这种情况下，从背景颜色可以看出数据中第一个字节的存取属性。



- 显示的存取属性在下载目标程序时初始化。
- 在如下所示的某些工作条件下，更新显示的间隔时间可能会比指定间隔长。
  - 主机执行/加载条件
  - 通信接口
  - 窗口大小（存储器显示范围）或显示的窗口数

6.1.1 扩展菜单

此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

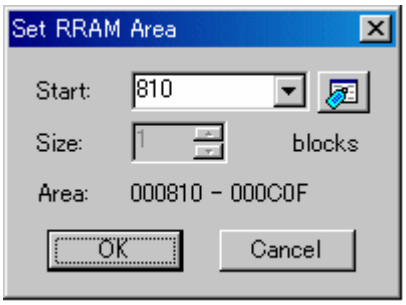
菜单		功能
RAM Monitor Area... （RAM 监控区域…）		设置 RAM 监控基址。
Sampling Period... （采样周期…）		设置 RAM 监控采样周期。
Clear （清除）		清除存取属性。
Up （上移）		将显示位置移至紧邻的上一 RAM 监控区域 （较小地址）
Down （下移）		将显示位置移至紧邻的下一 RAM 监控区域 （较大地址）
Address... （地址…）		从指定地址显示。
Scroll Area... （滚动区域…）		指定滚动范围。
Data Length （数据长度）	1byte （1 字节）	以 1 字节为单位显示。
	2bytes （2 字节）	以 2 字节为单位显示。
	4bytes （4 字节）	以 4 字节为单位显示。
	8bytes （8 字节）	以 8 字节为单位显示。
Radix （基数）	Hex （十六进制）	以十六进制显示。
	Dec （十进制）	以十进制显示。
	Signed Dec （带符号的十进制）	以带符号的十进制显示。
	Oct （八进制）	以八进制显示。
	Bin （二进制）	以二进制显示。
Code （代码）	ASCII	显示为 ASCII 字符。
	SJIS	显示为 SJIS 字符。
	JIS	显示为 JIS 字符。
	UNICODE	显示为 UNICODE 字符。
	EUC	显示为 EUC 字符。
	Float （浮点值）	显示为浮点值。
	Double （双精度值）	显示为双精度浮点值。
Layout （布局）	Label （标签）	在显示或不显示标签区域之间切换。
	Register （寄存器）	在显示或不显示寄存器区域之间切换。
	Code （代码）	在显示或不显示代码区域之间切换。
Column... （列…）		设置一行所显示的列数。
Split （拆分）		拆分窗口。
Toolbar display （工具栏显示）		显示工具栏。
Customize toolbar... （自定义工具栏…）		打开工具栏自定义对话框。
Allow Docking （允许入坞）		允许窗口入坞。
Hide （隐藏）		隐藏窗口。

6.1.2　设置 RAM 监控区域

在 [RAM monitor] （RAM 监控）窗口中选择弹出菜单 [RAM Monitor Area...] （RAM 监控区域…）。

此时将出现如下所示的 [Set RRAM Area] （设置 RRAM 区域）对话框。

在此对话框的 [Start] （起始）和 [Area] （区域）字段中，显示了当前设置的 RAM 监控区域的起始地址和 RAM 监控区域的范围。（在 [Size] （大小）字段中不能输入任何值。）



使用此对话框可以更改 RAM 监控区域的位置。

- 通过指定 RAM 监控区域的起始地址来指定此区域。大小不能更改 （固定为 1 KB）。
- 起始地址可以按 0x10 字节单位来指定。

  如果指定了一个不符合上述要求的地址值，该值会就近舍入为符合 0x10 字节单位的地址，然后才能用于设置。

6.1.2.1　更改 RAM 监控区域

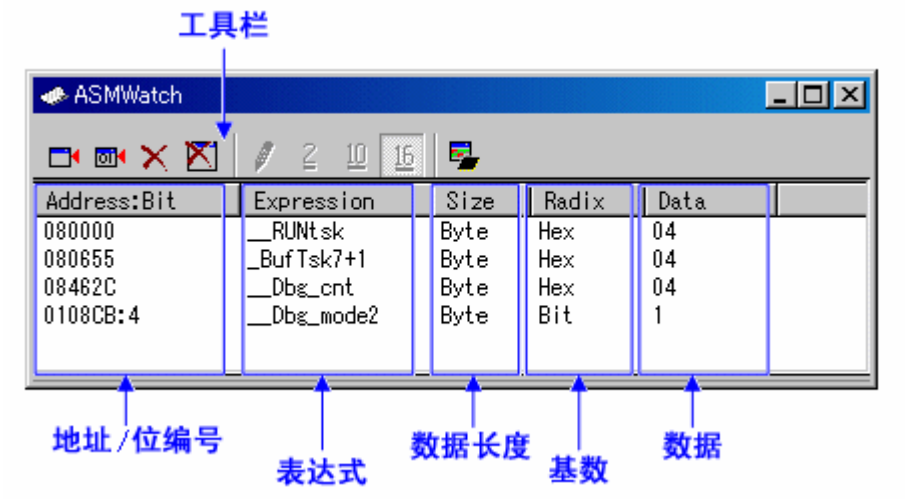
RAM 监控区域的起始地址可以更改。

在 [Set RRAM Area] （设置 RRAM 区域）对话框的 [Start] （起始）字段中指定 RAM 监控区域的起始地址。（在 [Size] （大小）字段中不能输入任何值。）

6.2 [ASM Watch] （ASM 监视）窗口

在 [ASM watch] （ASM 监视）窗口中，可以将指定地址注册为监视点，并检查这些地址处的存储器内容。

如果已注册的地址驻留在 RAM 监控区域内，那么在程序执行过程中，该地址处的存储器内容将按给定间隔进行更新 （默认情况下间隔时间为 100 ms）。



- 要注册的地址称为 “监视点”。可以注册下列内容之一：
  - 地址 （可使用符号指定）
  - 地址 + 位号
  - 位符号
- 关闭 [ASM watch] （ASM 监视）窗口时，已注册的监视点将保存在调试器中，并在再次打开窗口时自动完成注册。
- 如果为监视点指定了符号或位符号，则下载目标程序时会重新计算监视点地址。
- 屏幕上所显示的无效监视点将以 “-<not active>-” 标识。
- 监视点的列表顺序可通过拖放操作更改。
- 监视点的表达式、大小、基数和数据可通过在原位置编辑来更改。

注意事项

- RAM 监控获取通过总线存取的数据。除了从目标程序进行存取外，不会反映其他任何更改。
- 如果 RAM 监控区域的数据显示长度不是 1 字节，该数据中各字节可能具有不同的存储器存取属性。如果一组数据中的存取属性不统一，则该数据的存取属性可能不会正确显示。在这种情况下，从背景颜色可以看出数据中第一个字节的存取属性。

6.2.1 扩展菜单

此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

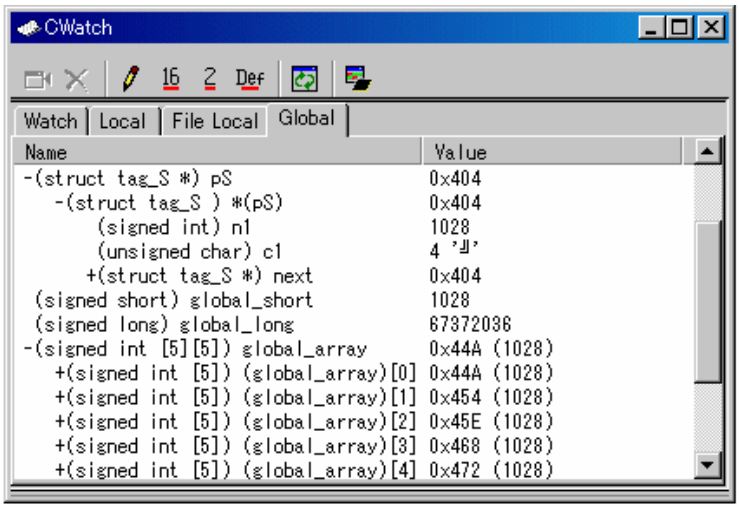
菜单		功能
Add... （添加…）		添加监视点。
Add Bit... （添加位…）		添加位标签监视点。
Remove （移除）		移除选定的监视点。
Remove All （全部移除）		移除全部监视点。
Set... （设置…）		对选定监视点设置新数据。
Radix （基数）	Bin （二进制）	以二进制显示。
	Dec （十进制）	以十进制显示。
	Hex （十六进制）	以十六进制显示。
Refresh （刷新）		刷新存储器数据。
Layout （布局）	Address Area （地址区域）	在显示或不显示地址区域之间切换。
	Size Area （大小区域）	在显示或不显示大小区域之间切换。
RAM Monitor （RAM 监控）	Enable RAM Monitor （允许 RAM 监控）	在允许或禁止 RAM 监控功能之间切换。
	Sampling Period... （采样周期…）	设置 RAM 监控采样周期。
Toolbar display （工具栏显示）		显示工具栏。
Customize toolbar... （自定义工具栏…）		打开工具栏自定义对话框。
Allow Docking （允许入坞）		允许窗口入坞。
Hide （隐藏）		隐藏窗口。

6.3 [C Watch] （C 监视）窗口

[C Watch] （C 监视）窗口将显示 C/C++ 表达式及其值 （计算结果）。

[C Watch] （C 监视）窗口中显示的 C/C++ 表达式通常称为 C 监视点。 C 监视点的计算结果显示在每次执行命令时更新。

如果 RAM 监控功能有效，并且 C 监视点位于 RAM 监控区域内，则显示的值在目标程序执行期间更新。



- 可以按作用域 （局部、文件局部或全局）检查变量。
- 显示内容在 PC 值发生改变时自动更新。
- 变量值可以更改。
- 可分别更改各个变量的显示基数。
- 任何变量均可注册到 [Watch] （监视）选项卡，这样该变量即可总是显示：
  - 各个工程的已注册内容将分别保存。
  - 如果同时打开两个或两个以上的 [C watch] （C 监视）窗口， [Watch] （监视）选项卡中的注册内容将在多个窗口间共享。
- 通过添加 [Watch] （监视）选项卡，可将 C 监视点注册到不同的目标中。
- 通过拖放操作，可从另一窗口或编辑器注册变量。
- C 监视点可以按名称或地址排序。
- 使用 RAM 监控功能，在程序执行过程中可以对值进行实时检查。

注意事项

- 以下所列 C 监视点的值无法更改：
  - 位字段变量
  - 寄存器变量
  - 未指示地址的 C 监视点 （无效的 C 监视点）
- 如果某 C/C++ 语言表达式无法正确计算 （例如，存在未定义的 C/C++ 符号），此表达式将注册为无效的 C 监视点，显示为 “--<not active>--”。

  如果第二次这个 C/C++ 语言表达式可以进行正确计算，那么它将变为有效的 C 监视点。
- [Local] （局部变量）、 [File Local] （文件局部变量）和 [Global] （全局变量）选项卡的显示设置无法保存。 [Watch] （监视）选项卡的内容和新添加的选项卡将得以保存。
- RAM 监控获取通过总线存取的数据。除了从目标程序进行存取外，不会反映其他任何更改。
- 实时更改的变量只有全局变量和文件局部变量。
- 如果 RAM 监控区域的数据显示长度不是 1 字节，该数据中各字节可能具有不同的存储器存取属性。如果一组数据中的存取属性不统一，则该数据的存取属性可能不会正确显示。在这种情况下，从背景颜色可以看出数据中第一个字节的存取属性。

6.3.1 扩展菜单

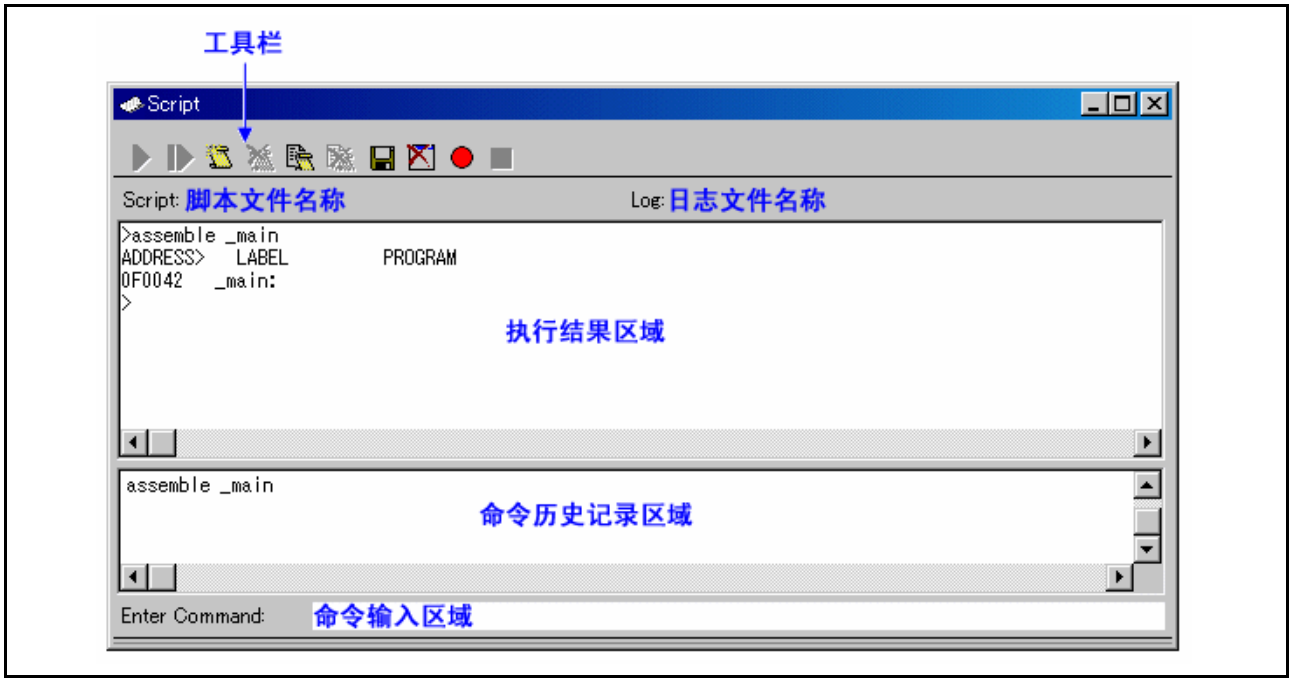
此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

菜单		功能
Add... （添加…）		添加 C 监视点。
Remove （移除）		移除选定的 C 监视点。
Remove All （全部移除）		移除全部 C 监视点。
Initialize （初始化）		对选定的 C 监视点重新求值。
Set New Value... （设置新值…）		对选定的 C 监视点设置新数据。
Radix （基数）	Hex （十六进制）	以十六进制显示。
	Bin （二进制）	以二进制显示。
	Default （默认值）	显示默认基数。
	Toggle(All Variables) （切换 （全部变量））	更改基数 （切换）。
Refresh （刷新）		刷新存储器数据。
Hide type name （隐藏类型名称）		隐藏变量的类型名称。
Show char* as string （将 char* 显示为 string）		选择是否将 char* 显示为 string。
Sort （排序）	Sort by Name （按名称排序）	按变量名称对变量排序。
	Sort by Address （按地址排序）	按变量地址对变量排序。
RAM Monitor （RAM 监控）	Enable RAM Monitor （允许 RAM 监控）	在允许或禁止 RAM 监控功能之间切换。
	Sampling Period... （采样周期…）	设置 RAM 监控采样周期。
Add New Tab... （添加新选项卡…）		添加新选项卡。
Remove Tab （移除选项卡）		移除选定的选项卡。
Toolbar display （工具栏显示）		显示工具栏。
Customize toolbar... （自定义工具栏…）		打开工具栏自定义对话框。
Allow Docking （允许入坞）		允许窗口入坞。
Hide （隐藏）		隐藏窗口。

6.4 [Script] （脚本）窗口

[Script] （脚本）窗口显示文本形式脚本命令的执行及执行的结果。

脚本命令可通过脚本文件执行，或者以交互方式执行。也可以将脚本命令写入脚本文件，从而自动执行脚本命令。脚本命令的执行结果还可以存储到之前指定的日志文件中。



- [Script] （脚本）窗口有一个显示缓冲区，其中存储最后 1000 行的执行结果。因此执行结果可存储在文件 （显示文件）中，无需指定日志文件。
- 打开脚本文件后，命令历史区域将变为脚本文件显示区域，并显示脚本文件的内容。当脚本文件为嵌套文件时，则会显示最后打开的脚本文件的内容。脚本文件显示区域将以反色形式显示当前正在执行的行。
- 打开脚本文件后，如果脚本文件尚未执行，可以从命令输入区域调用脚本命令。
- [Script] （脚本）窗口可将已执行的历史命令记录到文件。此功能与日志功能并不相同。它只记录已执行的命令，并不记录命令执行结果，因此所保存的文件可用作脚本文件。

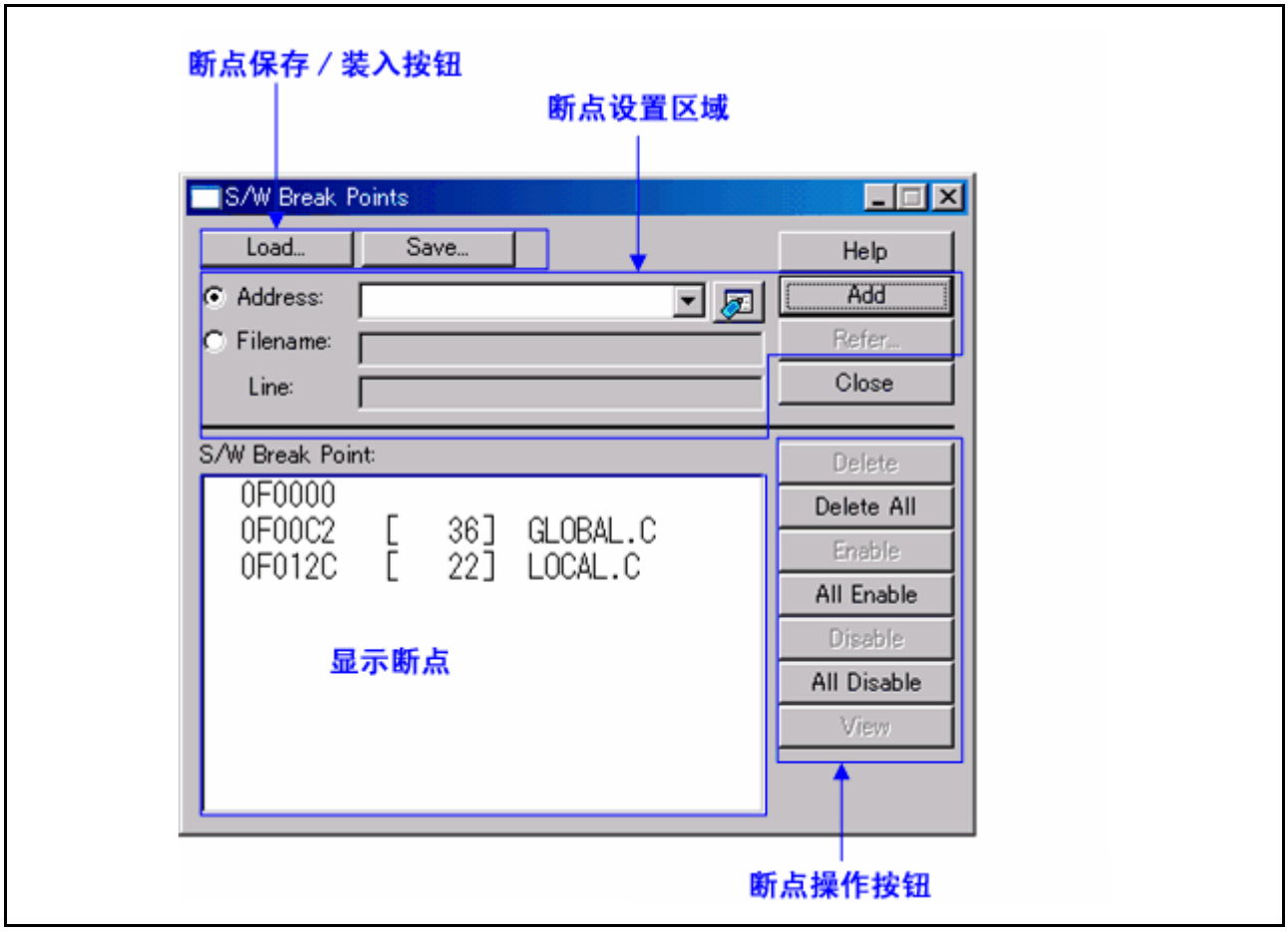
6.4.1 扩展菜单

此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

菜单		功能
Script （脚本）	Open... （打开…）	打开脚本文件。
	Run （运行）	运行脚本文件。
	Step （步进）	单步执行脚本文件。
	Close （关闭）	关闭脚本文件。
View （视图）	Save... （保存…）	将显示缓冲区保存到文件。
	Clear （清除）	清除显示缓冲区。
Log （日志）	On... （开…）	打开日志文件并开始记录 （开始输出到文件）。
	Off （关）	关闭日志文件并结束记录 （停止输出到文件）。
Record （记录）	On... （开…）	将已执行的命令记录到文件。
	Off （关）	停止记录已执行的命令。
Copy （复制）		复制所选内容并将其置于剪贴板。
Paste （粘贴）		插入剪贴板内容。
Cut （剪切）		剪切所选内容并将其置于剪贴板。
Delete （删除）		擦除所选内容。
Undo （撤消）		撤消上一操作。
Toolbar display （工具栏显示）		显示工具栏。
Customize toolbar... （自定义工具栏…）		打开工具栏自定义对话框。
Allow Docking （允许入坞）		允许窗口入坞。
Hide （隐藏）		隐藏窗口。

6.5 [S/W Break Point Setting] （软件断点设置）窗口

可以通过 [S/W Break Point Setting] （软件断点设置）窗口设置软件断点。
软件断点会在指定的断点之前立即停止执行指令。



- 如果设置了多个软件断点，则遇到其中任一软件断点地址 （或条件）时都会停止执行程序。
- 在单击 “Close”（关闭）按钮关闭 [S/W Break Point Setting] （软件断点设置）窗口之前，可以连续设置软件断点。
- 在软件断点显示区域可通过单击操作清除、允许或禁止所选软件断点。也可以双击软件断点将其允许或禁止。
- 单击 [Save] （保存）按钮，可将软件断点保存到文件。要从已保存的文件重新加载软件断点设置，请单击 [Load] （加载）按钮。如果从文件加载软件断点，这些断点将被添加到任何现有断点中。

6.5.1 命令按钮

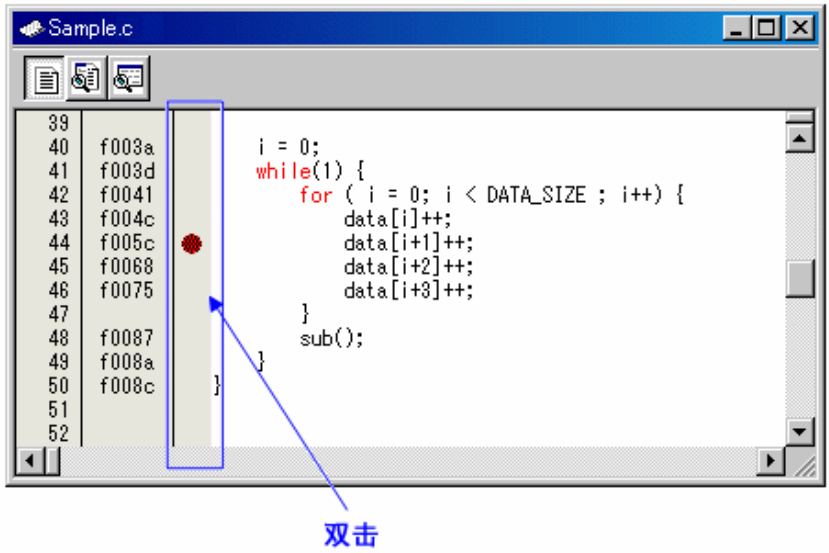
此窗口中的按钮具有以下含义。

按钮	功能
Load... （加载…）	从保存了设置信息的文件加载设置信息。
Save... （保存…）	将窗口中设置的内容保存到文件中。
Help （帮助）	显示此窗口的帮助信息。
Add （添加）	添加断点。
Refer... （引用…）	打开文件选择对话框。
Close （关闭）	关闭窗口。
Delete （删除）	移除选定断点。
Delete All （全部删除）	移除所有断点。
Enable （允许）	允许选定的断点。
All Enable （全部允许）	允许所有断点。
Disable （禁止）	禁止选定的断点。
All Disable （全部禁止）	禁止所有断点。
View （视图）	在 [Editor (Source)] （编辑器 （源））窗口中显示选定断点位置。

6.5.2 从 [Editor (Source)] （编辑器 （源））窗口设置和删除断点

可设置软件断点的区域因产品而异。请参阅 “11.1.2 可设置软件断点的区域”。

可以在 [Editor (Source)] （编辑器 （源））窗口中设置断点。为此，对于要在其中设置断点的行，双击对应的断点设置区域 （“S/W breakpoints”（软件断点）列）。（设置了断点的行将显示一个红色标记。）

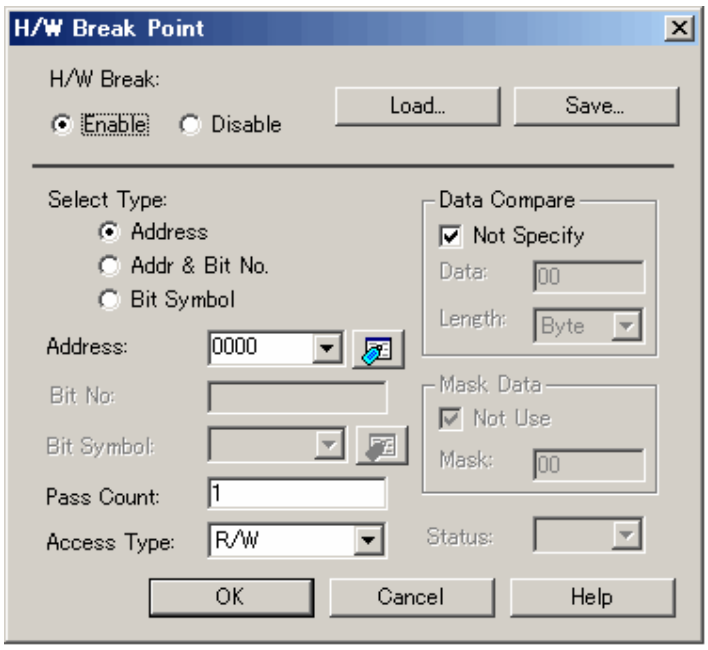


在断点设置区域 （“S/W breakpoints”（软件断点）列）中再次双击即可删除该断点。

在 [Editor (Source)] （编辑器 （源））窗口中，“S/W breakpoints”（软件断点）列显示在默认情况下设置为 “Enable”（允许）。要删除此列显示，请选择主菜单 - [Edit] （编辑）→ [Define Column Format] （定义列格式），在打开的对话框中取消选择 [S/W breakpoints] （软件断点）复选框。此时将从所有 [Editor (Source)] （编辑器 （源））窗口删除 “S/W breakpoints”（软件断点）列显示。也可以在 [Editor (Source)] （编辑器 （源））窗口中选择弹出菜单 - [Columns] （列）→ [S/W breakpoints] （软件断点），对各 [Editor (Source)] （编辑器 （源））窗口添加这一列。

6.6 [H/W Break Point Setting] （硬件断点设置）对话框

可以通过 [H/W Break Point Setting] （硬件断点设置）对话框设置硬件断点。
如果使用仿真器，则可以设置具有通过计数的单地址断点。



- 因为是地址断点存取类型，所以可以指定将数据写入地址断点 (Write)、从地址断点读取数据 (Read)、读写数据 (R/W) 以及取指令 (Fetch)。
- 还可以指定：如果在地址断点中读取或写入的数据具有一个特定的值，则执行中断。此外，还可以为该特定值指定有效位和无效位。
- 可以通过单击 [Save] （保存）将硬件断点保存到文件中。要从保存的文件中读取硬件断点设置，请单击 [Load] （加载）。

6.6.1 指定断点事件

指定 [H/W Break Point Setting] （硬件断点设置）对话框

- 可以指定存取类型 （Read、 Write、 R/W、 Fetch）。
- 可以指定要读取/写入的数据。还可以对数据指定 “Enable Bit/Disable Bit”（允许位/禁止位），即指定掩码。
- 可以指定通过计数 （1 到 255）。

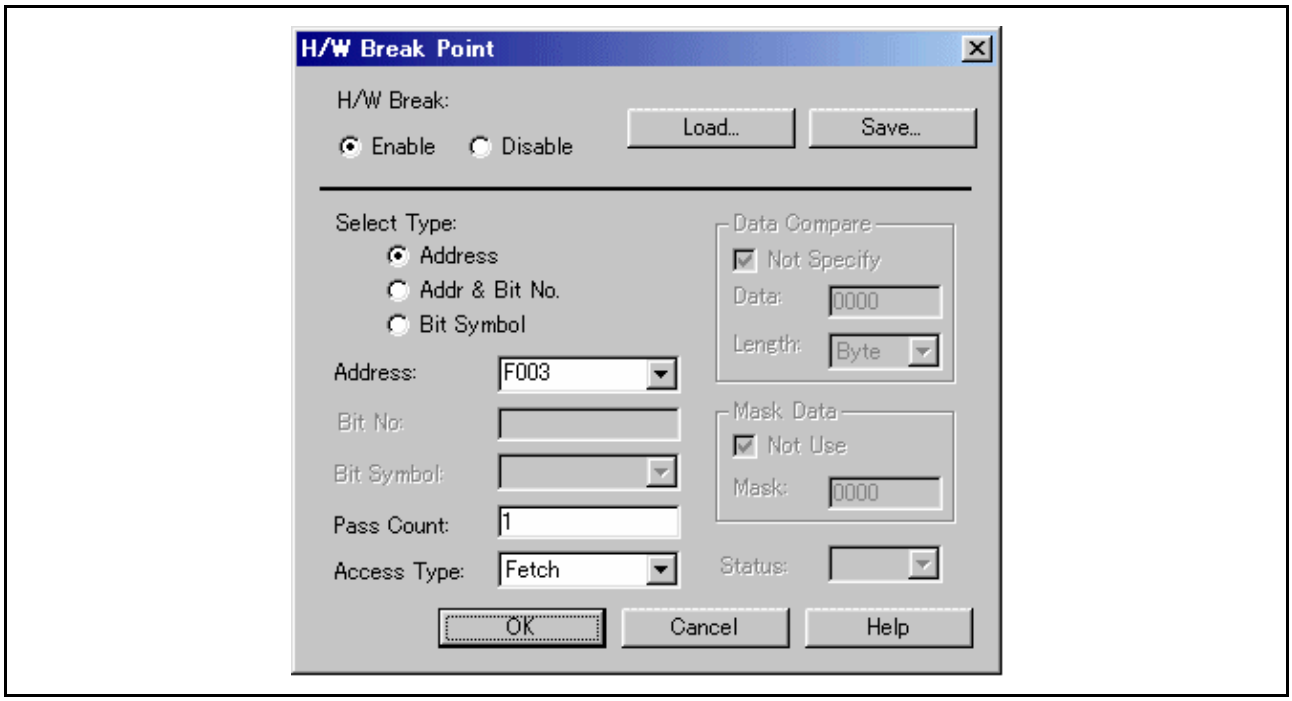
注意事项

无法检测到向奇地址写入字长度数据的操作。
对于 740，还无法检测到向偶地址写入字长度数据的操作。

6.6.1.1 取指令

设置如下所示。

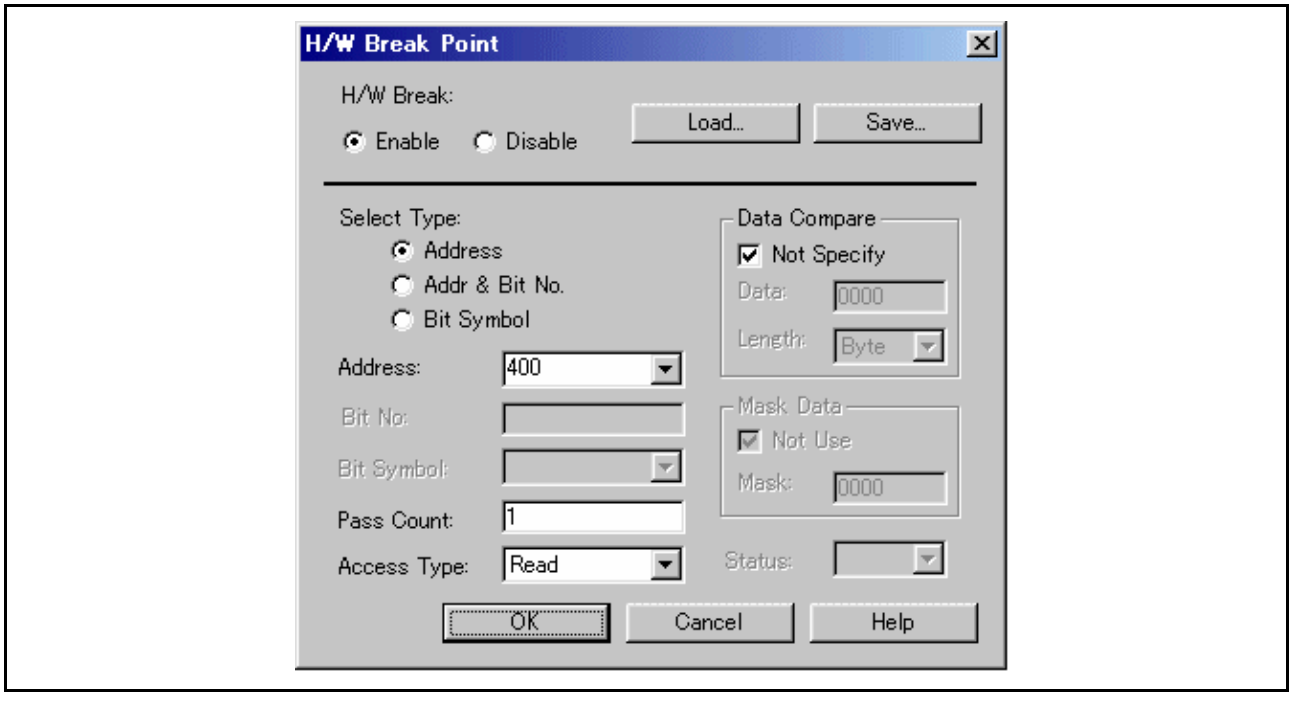
示例）在地址 F0003h 处执行指令。



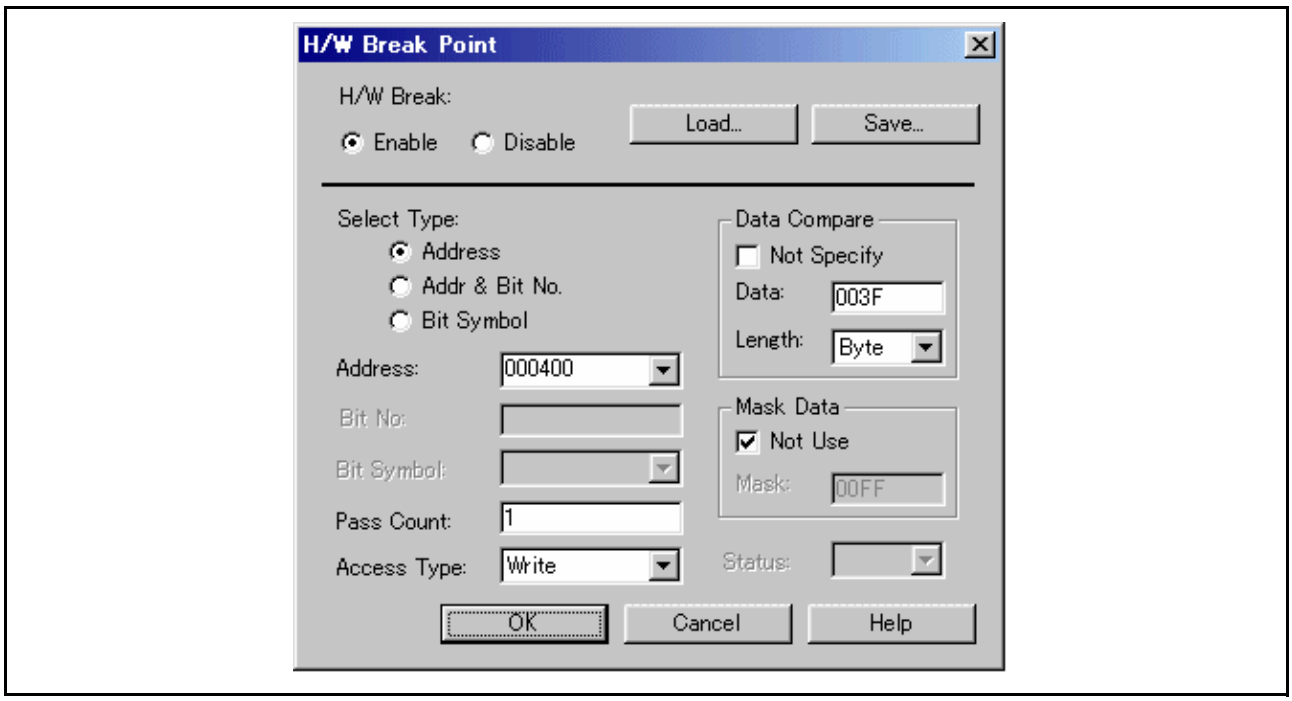
6.6.1.2 存储器存取

设置如下所示。

示例）读取偶地址 400h。



示例）将字节长度数据 3Fh 写入偶地址 400h。



注意事项

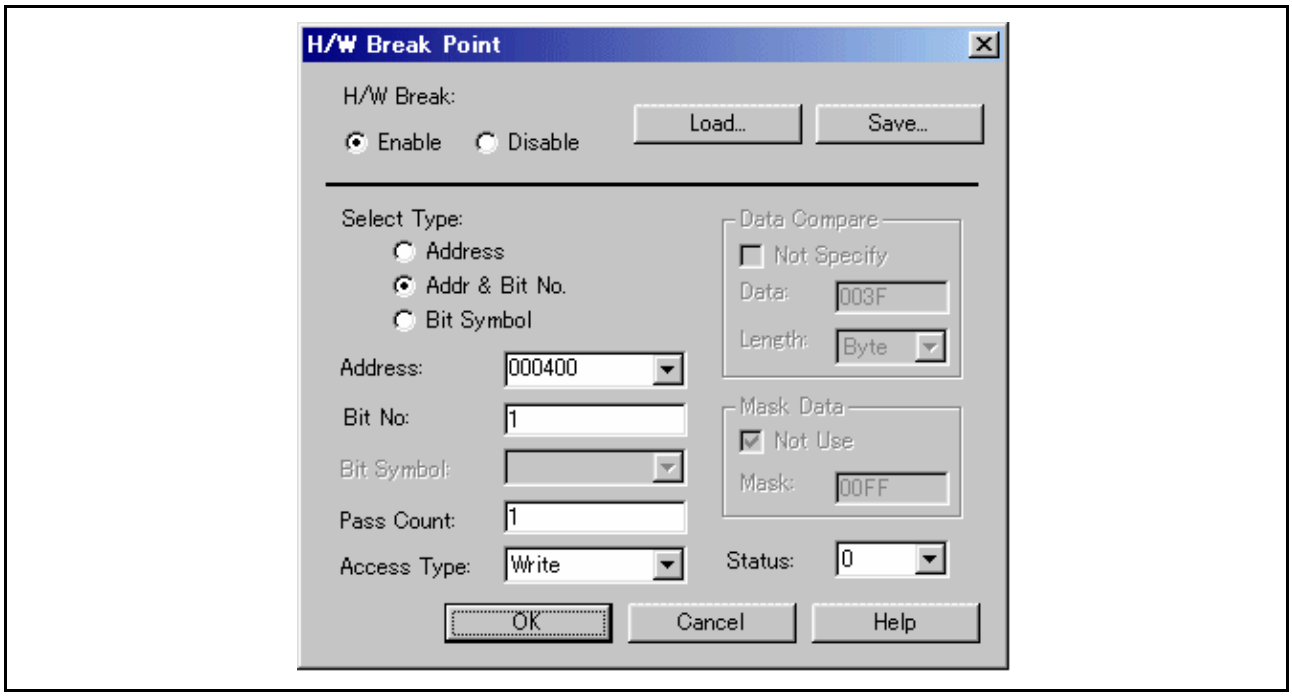
在具有 16 位存储器存取总线宽度的区域的某个奇地址处指定字节存取类型的断点时，需要进行如下设置：

- 在比较数据的高 8 位中设置要比较的数据。
- 在 [Length] （长度）列表框中设置 “Word”（字）。
- 在掩码数据的低位字节中设置 “00”。

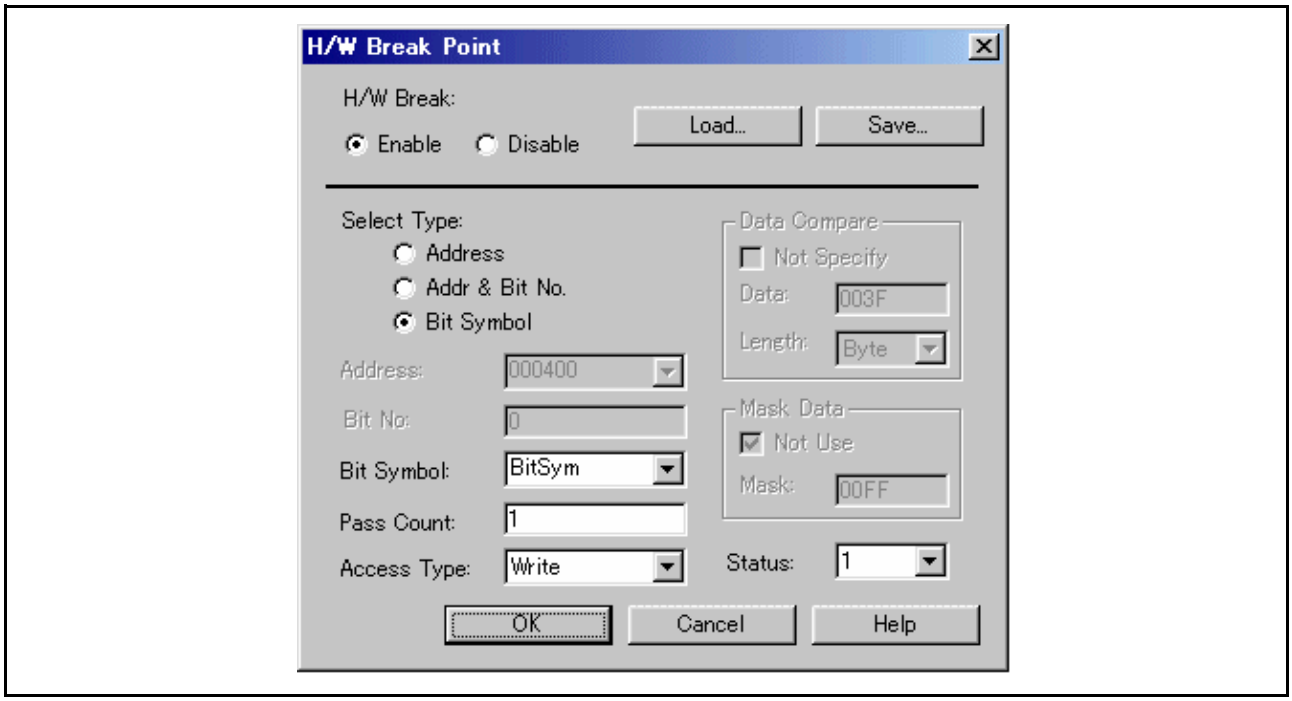
6.6.1.3 位存取

设置如下所示。

示例）将数据 0 写入地址 400h 的位 1。

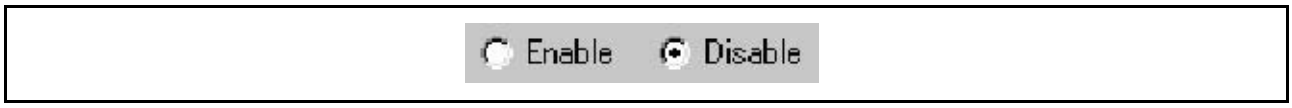


示例）将数据 0 写入位符号 “BitSym”。



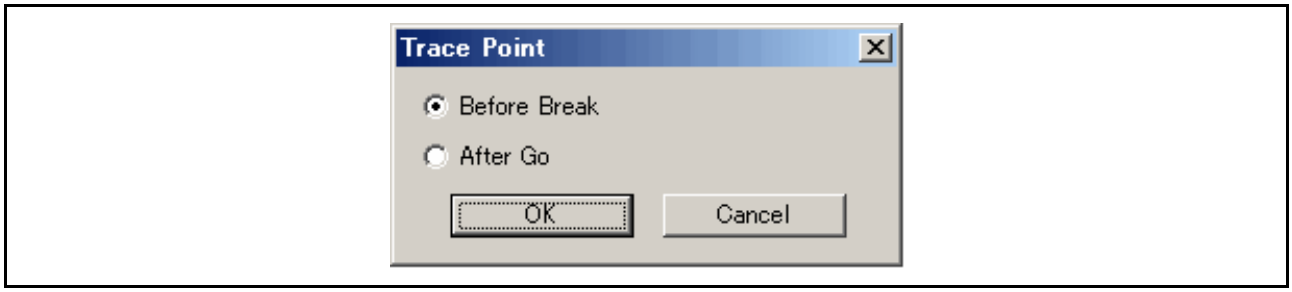
6.6.1.4 禁止硬件断点功能

单击 [Disable] （禁止）按钮。



6.7 [Trace Point Setting] （跟踪点设置）对话框

[Trace Point Setting] （跟踪点设置）对话框用于设置跟踪点。



6.7.1 指定跟踪范围

对于小型仿真调试器，可以记录相当于 32K 周期的数据。

Before Break （暂停前）	存储目标程序停止点之前的 32K 个周期 （-32K 到 0 周期）。
After Go （执行后）	存储跟踪开始之后的 32K 个周期 （-32K 到 0 周期）的跟踪数据。

6.8 [Trace] （跟踪）窗口

[Trace] （跟踪）窗口用于显示实时跟踪计量的结果。可以通过以下显示模式显示计量结果。

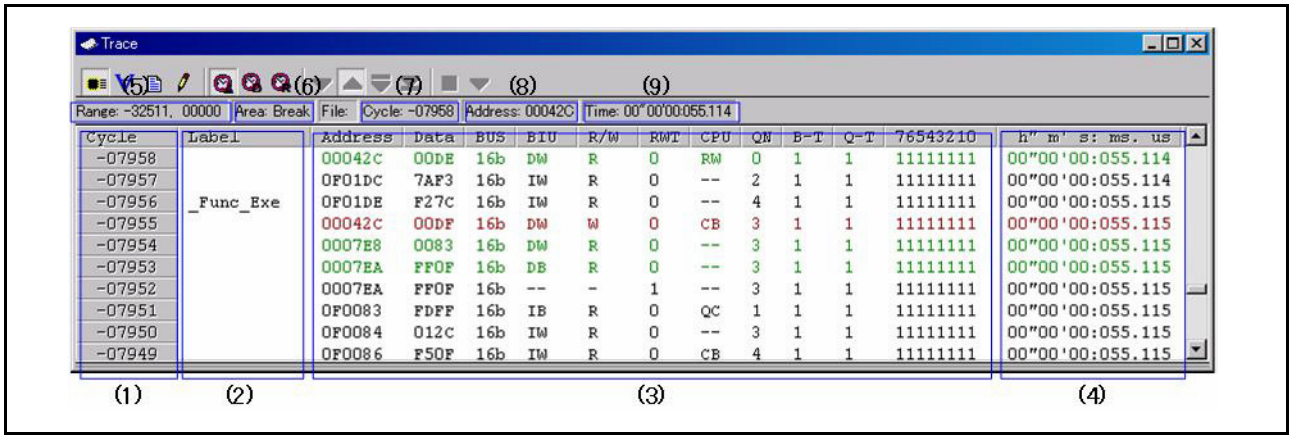
- “Bus mode”（总线模式）
  该模式用于按周期检查总线信息。显示内容取决于使用的 MCU 和仿真器系统。除总线信息外，该模式还允许以组合形式显示反汇编、源行或数据存取信息。
- “Disassemble mode”（反汇编模式）
  该模式用于检查执行的指令。除反汇编信息外，该模式还允许以组合形式显示源行或数据存取信息。
- “Data access mode”（数据存取模式）
  该模式用于检查数据读/写周期。除数据存取信息外，该模式还允许以组合形式显示源行信息。
- “Source mode”（源模式）
  该模式用于检查源程序中的程序执行路径。

跟踪计量结束时，会显示计量结果。当跟踪计量再启动时，会清除窗口显示内容。

可以在 [Trace Point Setting] （跟踪点设置）窗口中改变跟踪计量的范围。有关此窗口的详细信息，请参阅“6.7 [Trace Point Setting] （跟踪点设置）对话框”。使用默认设置时，会记录程序停止之前的跟踪信息。

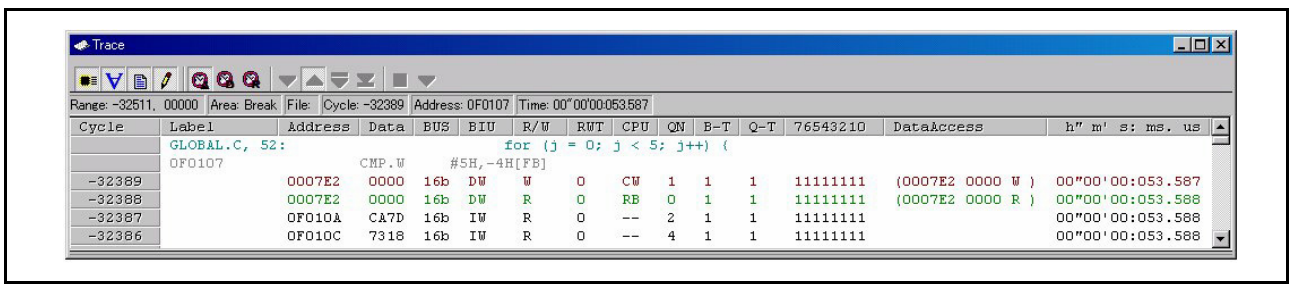
6.8.1 配置 “Bus Mode”（总线模式）

选择 “bus mode”（总线模式）后，跟踪信息会以总线模式显示。总线模式的配置如下所示。
总线模式下的显示内容因所用的 MCU 或仿真器系统而异。



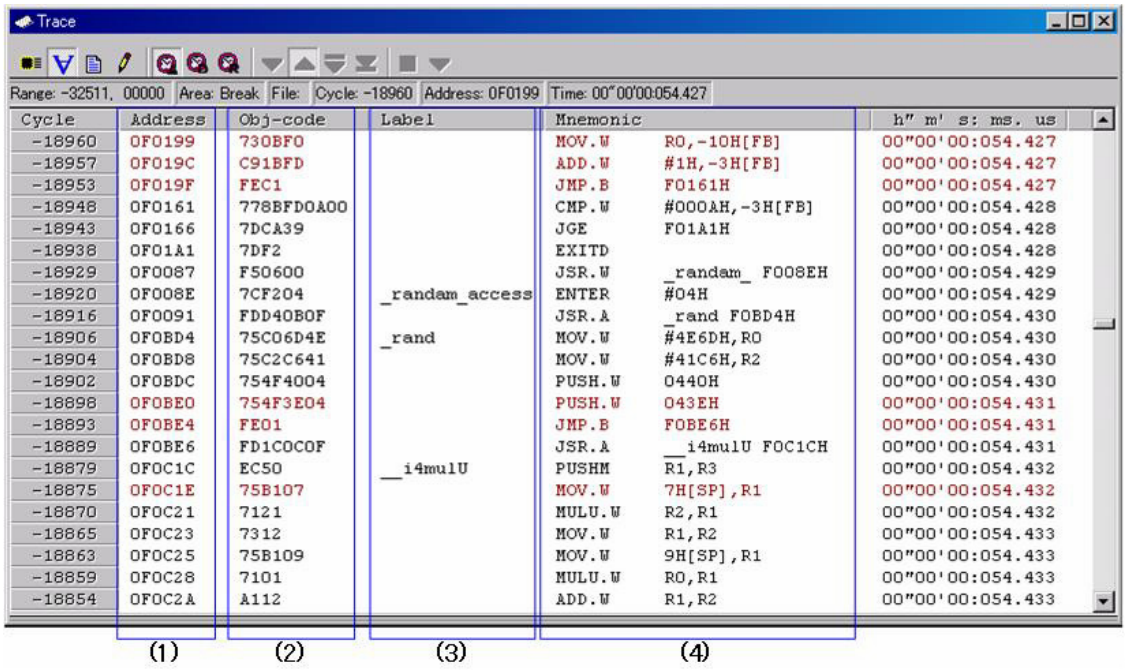
1. 周期显示区：
   显示跟踪周期。双击此处将弹出用于更改所显示周期的对话框。
2. 标签显示区：
   显示与地址总线信息对应的标签。双击此处将弹出用于搜索地址的对话框。
3. 总线信息显示区：
   这里显示的内容因所用的 MCU 或仿真器系统而异。
   — 请参阅 “6.8.6 740 调试器上的总线信息的显示”
4. 时间信息显示区：
   显示跟踪计量结果的时间信息。可以从菜单中选择以下三种模式之一。（不支持小型仿真调试器。）
   — “Absolute Time”（绝对时间）：按照绝对时间显示从程序开始运行的时间到现在所经过的时间（默认）。
   — “Differences”（差异）：显示与相邻的前一个周期的时间差异。
   — “Relative Time”（相对时间）：显示与选择的周期的相对时间。不过，请注意，更新跟踪计量结果时，此模式会更改为绝对时间显示模式。
5. 获取的跟踪计量结果范围：
   显示当前获取的跟踪计量结果范围。
6. 跟踪计量范围：
   显示当前设置的跟踪计量范围。
7. 第一行周期：
   显示所示的第一行的周期。
8. 第一行地址：
   显示所示的第一行的地址。
9. 第一行时间：
   显示所示的第一行的时间信息。
10. 窗口拆分框：
    双击此框可以将窗口拆分为若干部分。

除总线信息外，该窗口还能够以组合形式显示反汇编、源行或数据存取信息。在这种情况下，显示内容将与如下内容相似。



6.8.2　配置 “Disassemble Mode”（反汇编模式）

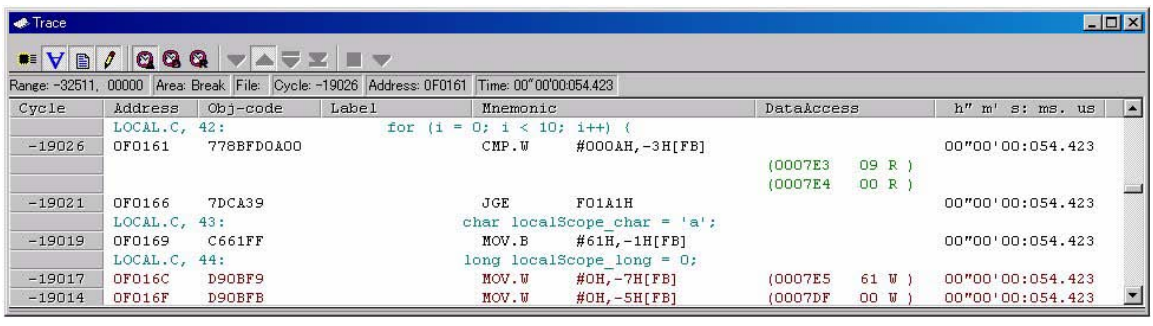
当选择 “disassemble mode”（反汇编模式）而取消选择 “bus mode”（总线模式）时，跟踪信息会以反汇编模式显示。反汇编模式的配置如下所示。



1. 地址显示区：
   显示与指令相对应的地址。双击此处将弹出用于搜索地址的对话框。
2. 目标代码显示区：
   显示指令的目标代码。
3. 标签显示区：
   显示与指令地址相对应的标签。双击此处将弹出用于搜索地址的对话框。
4. 助记符显示区：
   显示指令的助记符。

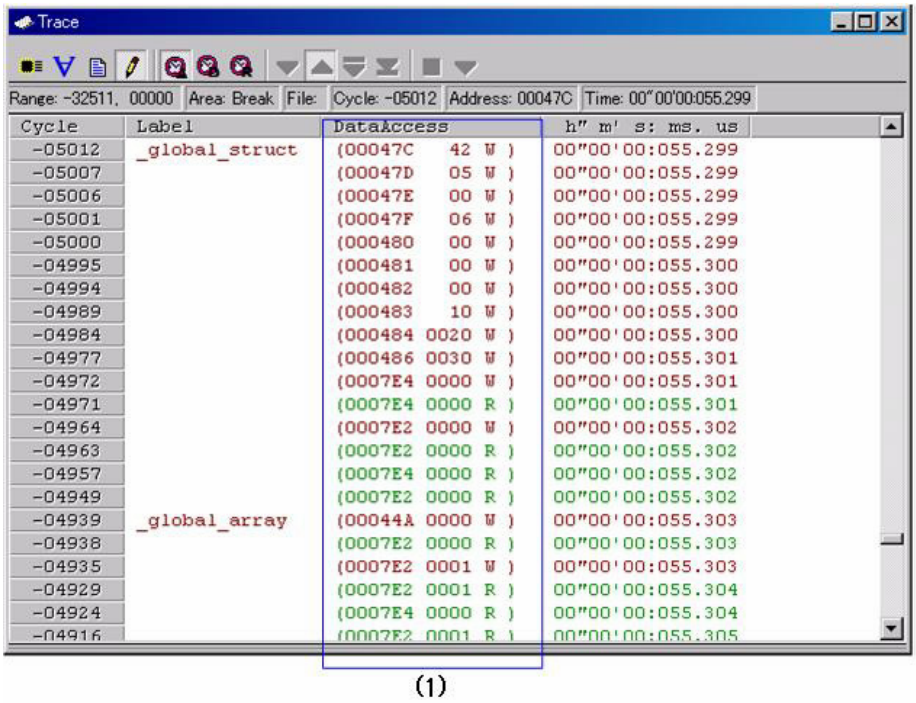
其他显示区与总线模式的显示区相同。

除反汇编信息外，该窗口还能够以组合形式显示源行或数据存取信息。在这种情况下，显示内容将与如下内容相似。



6.8.3　配置 “Data Access Mode”（数据存取模式）

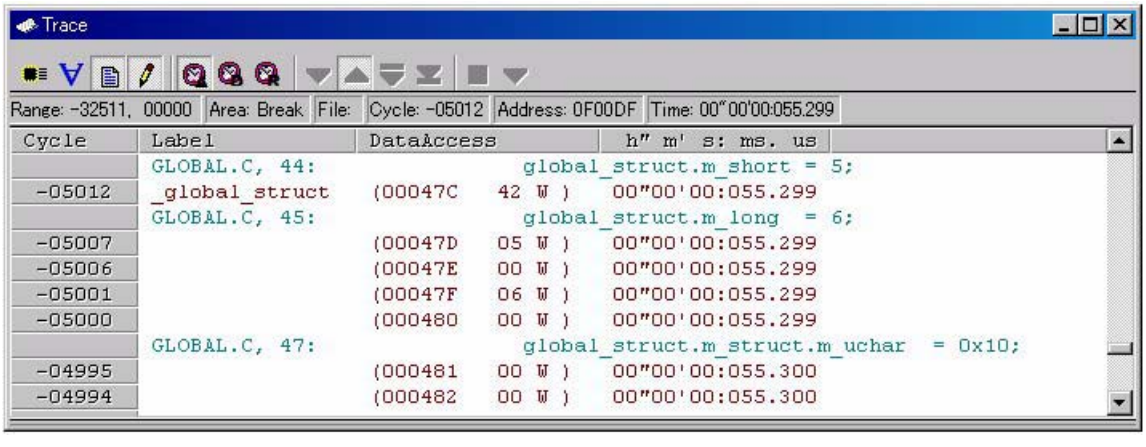
当选择 “data access mode”（数据存取模式）而取消选择 “bus mode”（总线模式）和 “disassemble mode”（反汇编模式）时，跟踪信息会以数据存取模式显示。数据存取模式的配置如下所示。



1. 数据存取显示区：
   显示数据存取信息。例如，如果这里显示的信息是 “000400 1234 W”，它表示数据 “1234H”以 2 个字节的宽度写入地址 000400H。

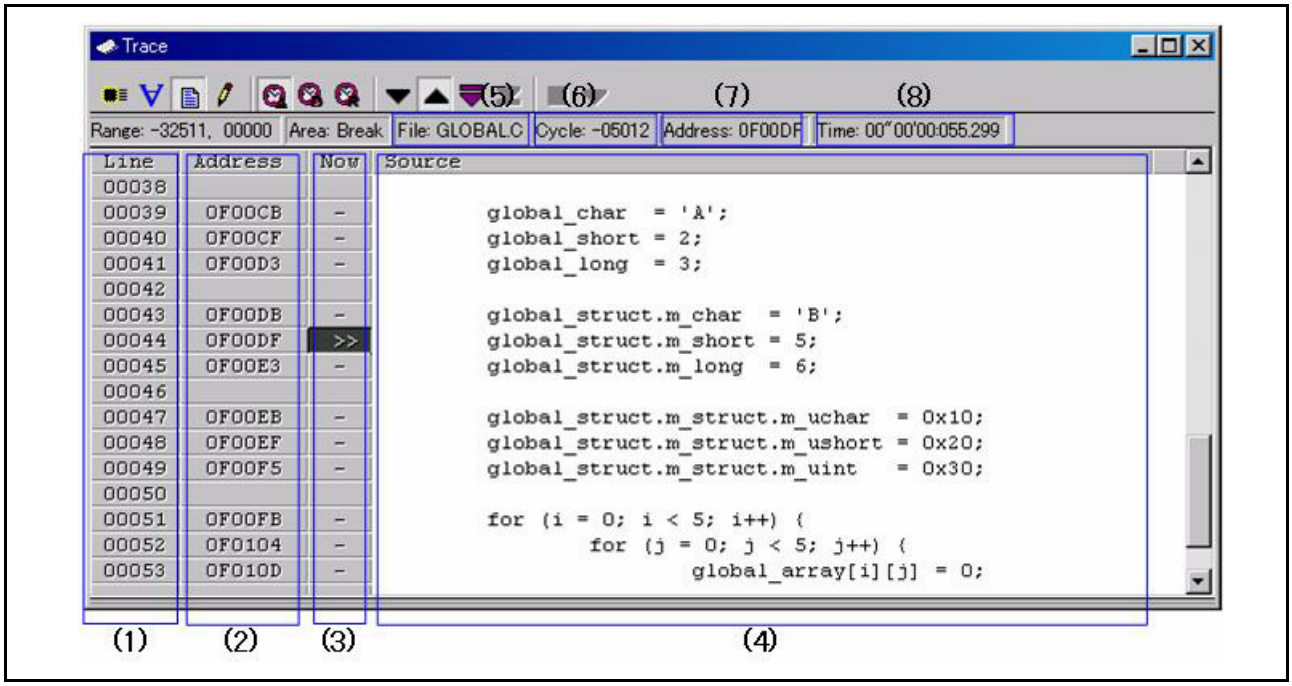
其他显示区与总线模式的显示区相同。

除数据存取信息外，该窗口还能够以组合形式显示源行信息。在这种情况下，显示内容将与如下内容相似。



6.8.4 配置 “Source Mode”（源模式）

当仅选择 “source mode”（源模式）时，跟踪信息会以源模式显示。源模式的配置如下所示。



1. 行号显示区：
   显示所示文件的行号信息。双击这里可以打开更改显示文件的对话框。
2. 地址显示区：
   显示与源行相对应的地址。双击此处将弹出用于搜索地址的对话框。
3. 引用周期显示区：
   显示当前引用的周期，该周期带有 “>>”标志。此外，如果有与源行相对应的地址，则该地址会带有 “-”标志。
4. 源显示区：
   显示源文件的内容。
5. 文件名：
   显示当前显示的源文件的文件名。
6. 引用周期：
   显示当前引用的周期。
7. 引用地址：
   显示与当前引用的周期相对应的地址。
8. 引用时间：
   显示与当前引用的周期相对应的时间信息。

其他显示区与总线模式的显示区相同。

6.8.5 扩展菜单

此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

菜单		功能
BUS		显示 “BUS mode”（总线模式）的信息。
DIS		显示 “Disassemble mode”（反汇编模式）的信息。
SRC		显示 “Source mode”（源模式）的信息。
DATA		显示 “Data access mode”（数据存取模式）的信息。
View （视图）	Cycle... （周期…）	通过指定周期来更改显示的位置。
	Address... （地址…）	通过搜索地址来更改显示的位置。
	Source... （源…）	显示选择的源文件。
Time （时间）	Absolute Time （绝对时间）	按照绝对时间显示从程序开始运行的时间到现在所经过的时间。
	Differences （差异）	显示与相邻的前一个显示周期的时间差异。
	Relative Time （相对时间）	显示与当前选择的周期的相对时间。
Trace （跟踪）	Forward （正向）	将搜索方向更改为正向。
	Backward （反向）	将搜索方向更改为反向。
	Step （步进）	以 “Step”（步进）模式在指定方向进行搜索。
	Come （指定位置）	以 “Come”（指定位置）模式在指定方向进行搜索。
	Stop （停止）	在中间停止跟踪计量并显示当前时间点的计量内容。
	Restart （再启动）	再启动跟踪计量。
Layout... （布局…）		更改当前视图的布局。
Copy （复制）		复制选择的行。
Save... （保存…）		将跟踪数据保存到文件。
Load... （加载…）		从文件加载跟踪数据。
Toolbar display （工具栏显示）		显示工具栏。
Customize toolbar... （自定义工具栏…）		打开工具栏自定义对话框。
Allow Docking （允许入坞）		允许窗口入坞。
Hide （隐藏）		隐藏窗口。

6.8.6 740 调试器上的总线信息的显示

从左到右的内容如下：

- Address

  地址总线的状态

- Data

  数据总线的状态

- Sync

  取指令操作码时会输出此信号。取操作码时，此信号指示逻辑 1。此同步值有时显示为 “(1)”。在这种情况下，它表示虚设的同步，即实际并未在线路上执行该指令。

- Read

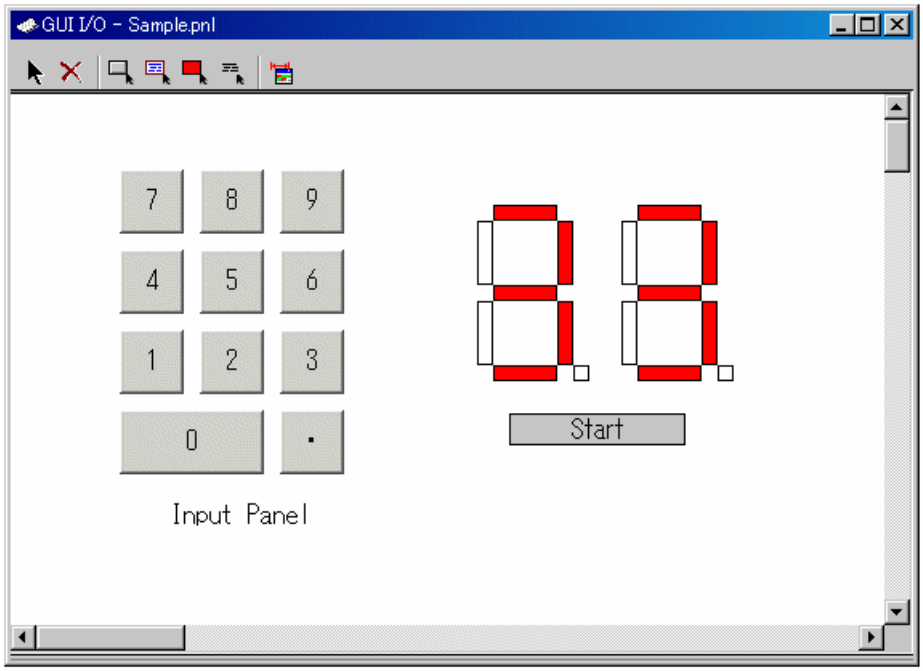
  此信号决定了数据总线的方向。要读取数据时，此信号指示逻辑 0。

- Write

  此信号决定了数据总线的方向。要写入数据时，此信号指示逻辑 0。

6.9 [GUI I/O] 窗口

在 [GUI I/O] 窗口中，可以创建用户目标系统键盘输入面板 （按钮），并单击所创建的按钮来进行端口输入。另外，还可以在此窗口中创建并使用用户目标系统输出面板。



- 可以在窗口上布置下列部件。
  - — 标签 （字符串）
    如果有任意值写入指定地址 （位），则显示/擦除用户指定的字符串。
  - — 指示灯
    如果有值写入指定地址 （或指定位），则更改区域的显示颜色。（取代 “指示灯亮起”）
  - — 按钮
    可在按下按钮时执行虚拟端口输入。
  - — 文本
    显示文本字符串。
- 也可将创建的面板保存到文件中并重新加载。
- 最多可为创建的部件设置 200 个地址点。如果为各个部件设置不同的地址，则最多可布置 200 个部件。

6.9.1 扩展菜单

此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

菜单	功能
Select Item （选择项目）	选择 I/O 项目。
Delete （删除）	删除选定的 I/O 项目。
Copy （复制）	复制选定的 I/O 项目。
Paste （粘贴）	粘贴复制的 I/O 项目。
Create Button （创建按钮）	创建新按钮项目。
Create Label （创建标签）	创建新标签项目。
Create LED （创建指示灯）	创建新指示灯项目。
Create Text （创建文本）	创建新文本项目。
Display grid （显示网格）	显示网格线。
Save... （保存…）	保存 I/O 面板文件。
Load... （加载…）	加载 I/O 面板文件。
Sampling Period... （采样周期…）	设置 RAM 监控采样周期。
Toolbar display （工具栏显示）	显示工具栏。
Customize toolbar... （自定义工具栏…）	打开工具栏自定义对话框。
Allow Docking （允许入坞）	允许窗口入坞。
Hide （隐藏）	隐藏窗口。

7 脚本命令表

我们准备了以下脚本命令。
带黄色背景的命令可以在运行时执行。
该产品不支持后面带有 “*” 的命令。

7.1 脚本命令表 （按功能分类）

7.1.1 执行命令

命令名称	缩写名称	内容
Go	G	带有断点的程序执行
GoFree	GF	自由运行程序执行
GoProgramBreak*	GPB	运行带有软件断点的目标程序
GoBreakAt*	GBA	运行带有软件断点的目标程序
Stop	-	停止程序执行
Status	-	检查 MCU 的工作状况
Step	S	暂停以便用户输入，直到指定的时间过去为止
StepInstruction	SI	单步执行指令
OverStep	O	跳过执行源行
OverStepInstruaction	OI	跳过执行指令
Return	RET	执行源行返回
ReturnInstruction	RETI	执行指令返回
Reset	-	复位目标 MCU
Time	-	设置运行时显示并检查当前设置

7.1.2 文件操作命令

命令名称	缩写名称	内容
Load	L	下载目标程序
LoadHex	LH	下载 Intel HEX 格式文件
LoadMot*	LM	下载 Motorola S 类型格式文件
LoadSymbol	LS	加载源行 /ASM 符号信息
LoadIeee*	LI	下载 IEEE-695 绝对格式文件
Reload	-	重新下载目标程序
UploadHex	UH	将数据输出到 Intel HEX 格式文件
UploadMot*	UM	将数据输出到 Motorola S 类型格式文件

7.1.3 寄存器操作命令

命令名称	缩写名称	内容
Register	R	检查和设置寄存器值

7.1.4 存储器操作命令

命令名称	缩写名称	内容
DumpByte	DB	显示存储器的内容 （以 1 个字节为单位）
DumpWord*	DW	显示存储器的内容 （以 2 个字节为单位）
DumpLword*	DL	显示存储器的内容 （以 4 个字节为单位）
SetMemoryByte	MB	检查和更改存储器内容 （以 1 个字节为单位）
SetMemoryWord*	MW	检查和更改存储器内容 （以 2 个字节为单位）
SetMemoryLword*	ML	检查和更改存储器内容 （以 4 个字节为单位）
FillByte	FB	用指定数据填充存储块 （以 1 个字节为单位）
FillWord*	FW	用指定数据填充存储块 （以 2 个字节为单位）
FillLword*	FL	用指定数据填充存储块 （以 4 个字节为单位）
Move	-	移动存储块 （以 1 个字节为单位）
MoveWord*	MOVEW	移动存储块 （以 2 个字节为单位）

7.1.5 汇编 / 反汇编命令

命令名称	缩写名称	内容
Assemble	A	逐行汇编
DisAssemble	DA	逐行反汇编存储器内容
Module	MOD	显示模块名称
Scope	-	设置和检查有效的局部符号作用域
Section	SEC	检查段信息
Bit*	-	检查和设置位符号
Symbol	SYM	检查汇编器符号
Label	-	检查汇编器标签
Express	EXP	显示汇编器表达式

7.1.6 软件断点设置命令

命令名称	缩写名称	内容
SoftwareBreak	SB	设置和检查软件断点
SoftwareBreakClear	SBC	清除软件断点
SoftwareBreakClearAll	SBCA	清除所有软件断点
SoftwareBreakDisable	SBD	禁止软件断点
SoftwareBreakDisableAll	SBDA	禁止所有软件断点
SoftwareBreakEnable	SBE	允许软件断点
SoftwareBreakEnableAll	SBEA	允许所有软件断点
BreakAt	-	通过指定行号设置软件断点
BreakIn	-	通过指定函数设置软件断点

7.1.7 硬件断点设置命令

命令名称	缩写名称	内容
HardwareBreak	HB	设置和检查硬件断点
BreakMode	BM	设置和检查硬件断点模式

7.1.8 实时跟踪命令

命令名称	缩写名称	内容
TracePoint	TP	设置和检查跟踪点
TraceData*	TD	实时跟踪数据显示
TraceList*	TL	显示反汇编的实时跟踪数据

7.1.9 脚本 / 日志文件命令

命令名称	缩写名称	内容
Script	-	打开和执行脚本文件
Exit	-	退出脚本文件
Wait	-	在输入命令之前等待事件发生
Pause	-	等待用户输入
Sleep	-	暂停以便用户输入，直到指定的时间过去为止
Logon	-	将屏幕显示内容输出到日志文件
Logoff	-	停止将屏幕显示内容输出到日志文件
Exec	-	执行外部应用程序

7.1.10 程序显示命令

命令名称	缩写名称	内容
Func	-	检查函数名称和显示函数内容
Up*	-	显示调用函数
Down*	-	显示被调用函数
Where*	-	显示函数调用状态
Path	-	设置和检查搜索路径
AddPath	-	添加搜索路径
File	-	检查文件名和显示该文件的内容

7.1.11 映像命令

命令名称	缩写名称	内容
Map*	-	检查和设置映像数据

7.1.12 时钟命令

命令名称	缩写名称	内容
Clock	CLK	检查和更改时钟

7.1.13 C 语言调试命令

命令名称	缩写名称	内容
Print	-	检查指定 C 变量表达式的值
Set	-	在指定 C 变量表达式中设置指定数据

7.1.14 实用程序命令

命令名称	缩写名称	内容
Radix	-	设置和检查数字输入的基数
Alias	-	指定和检查命令别名定义
UnAlias	-	取消为命令定义的别名
UnAliasAll	-	取消为命令定义的所有别名
Version	VER	显示版本号
Date	-	显示日期
Echo	-	显示消息
CD	-	窗口打开

7.2 脚本命令表 （依字母顺序）

命令名称	缩写名称	内容
AddPath	-	添加搜索路径
Alias	-	指定和检查命令别名定义
Assemble	A	逐行汇编
Bit*	-	检查和设置位符号
BreakAt	-	通过指定行号设置软件断点
BreakIn	-	通过指定函数设置软件断点
BreakMode	BM	设置和检查硬件断点模式
CD	-	指定和检查当前目录
Clock	CLK	检查和更改时钟
Date	-	显示日期
DisAssemble	DA	逐行反汇编存储器内容
Down*	-	显示被调用函数
DumpByte	DB	显示存储器的内容 （以 1 个字节为单位）
DumpLword*	DL	显示存储器的内容 （以 4 个字节为单位）
DumpWord*	DW	显示存储器的内容 （以 2 个字节为单位）
Echo	-	显示消息
Exec	-	执行外部应用程序
Exit	-	退出脚本文件
Express	EXP	显示汇编器表达式
File	-	检查文件名和显示该文件的内容
FillByte	FB	用指定数据填充存储块 （以 1 个字节为单位）
FillLword*	FL	用指定数据填充存储块 （以 4 个字节为单位）
FillWord*	FW	用指定数据填充存储块 （以 2 个字节为单位）
Func	-	检查函数名称和显示函数内容
Go	G	带有断点的程序执行
GoBreakAt*	GBA	运行带有软件断点的目标程序
GoFree	GF	自由运行程序执行
GoProgramBreak*	GPB	运行带有软件断点的目标程序
HardwareBreak	HB	设置和检查硬件断点
Label	-	检查汇编器标签
Load	L	下载目标程序
LoadHex	LH	下载 Intel HEX 格式文件
LoadIeee*	LI	下载 IEEE-695 绝对格式文件
LoadMot*	LM	下载 Motorola S 格式文件
LoadSymbol	LS	加载源行 /ASM 符号信息
Logoff	-	停止将屏幕显示内容输出到日志文件
Logon	-	将屏幕显示内容输出到日志文件
Map*	-	检查和设置映像数据
Module	MOD	显示模块名称

命令名称	缩写名称	内容
Move	-	移动存储块 （以 1 个字节为单位）
MoveWord*	MOVEW	移动存储块 （以 2 个字节为单位）
OverStep	O	跳过执行源行
OverStepInstruaction	OI	跳过执行指令
Path	-	设置和检查搜索路径
Pause	-	等待用户输入
Print	-	检查指定 C 变量表达式的值
Radix	-	设置和检查数字输入的基数
Register	R	检查和设置寄存器值
Reload	-	重新下载目标程序
Reset	-	复位目标 MCU
Return	RET	执行源行返回
ReturnInstruction	RETI	执行指令返回
Scope	-	设置和检查有效的局部符号作用域
Script	-	打开和执行脚本文件
Section	SEC	检查段信息
Set	-	在指定 C 变量表达式中设置指定数据
SetMemoryByte	MB	检查和更改存储器内容 （以 1 个字节为单位）
SetMemoryLword*	ML	检查和更改存储器内容 （以 4 个字节为单位）
SetMemoryWord*	MW	检查和更改存储器内容 （以 2 个字节为单位）
Sleep	-	暂停以便用户输入，直到指定的时间过去为止
SoftwareBreak	SB	设置和检查软件断点
SoftwareBreakClear	SBC	清除软件断点
SoftwareBreakClearAll	SBCA	清除所有软件断点
SoftwareBreakDisable	SBD	禁止软件断
SoftwareBreakDisableAll	SBDA	禁止所有软件断点
SoftwareBreakEnable	SBE	允许软件断点
SoftwareBreakEnableAll	SBEA	允许所有软件断点
Status	-	检查 MCU 的工作状况
Step	S	单步执行源行
StepInstruction	SI	单步执行指令
Stop	-	停止程序执行
Symbol	SYM	检查汇编器符号
Time	-	设置运行时显示并检查当前设置
TraceData*	TD	实时跟踪数据显示
TraceList*	TL	显示反汇编的实时跟踪数据
TracePoint	TP	设置和检查跟踪点
UnAlias	-	取消为命令定义的别名
UnAliasAll	-	取消为命令定义的所有别名
Up*	-	显示调用函数

命令名称	缩写名称	内容
UploadHex	UH	将数据输出到 Intel HEX 格式文件
UploadMot*	UM	将数据输出到 Motorola S 格式文件
Version	VER	显示版本号
Wait	-	在输入命令之前等待事件发生
Where*	-	显示函数调用状态

8 编写脚本文件

此调试器允许在 [Script] （脚本）窗口中运行脚本文件。脚本文件包含自动执行脚本命令所需的控制。

8.1 脚本文件的结构元素

可以在脚本文件中包括以下内容：

- 脚本命令
- 赋值语句
- 条件语句 （if、 else、 endi）
  程序执行转移到要根据条件表达式的结果执行的语句。
- 循环语句 （while、 endw）
  根据表达式，重复执行一条或多条语句构成的块。
- 中断语句
  从最里面的循环退出。
- 注释语句
  可以在脚本文件中包括注释。执行脚本命令时，会忽略注释语句。

在脚本文件的每一行上，只能指定一条语句。不能在一行上指定多条语句，也不能编写跨越两行或更多行的语句。

注意

- 不能在与脚本命令相同的行上加插注释。
- 最多可以将脚本文件嵌套五层。
- 最多可以将 if 语句和 while 语句嵌套 32 层。
- 在每个脚本文件中， if 语句必须与 endi 语句成对出现， while 语句必须与 endw 语句成对出现。
- 脚本文件中包含的表达式将作为无符号类型进行计算。因此，如果在 if 或 while 语句中使用负值来进行比较，将无法保证正常运算。
- 每行最多可以指定 4096 个字符。如果某行超过此字符数，会发生错误。
- 当自动执行某个包含非法命令的脚本文件时 （打开脚本文件后，从 [Script] （脚本）窗口菜单中选择 [Option] （选项）→[Script] （脚本）→[Run] （运行），或者在 [Script] （脚本）窗口中单击该按钮），即使在检测到错误以后，脚本文件也会继续执行，除非是无法读取脚本行本身。如果检测到错误并且脚本文件继续执行，则无法保证检测到错误以后的运算。因此，在检测到错误以后，执行的结果并不可靠。

8.1.1 脚本命令

可以使用在 [Script] （脚本）窗口中输入的相同脚本命令。也可以在其他脚本文件内部调用脚本文件（最多嵌套 10 层）。

8.1.2 赋值语句

赋值语句定义并初始化宏变量以及进行赋值。以下所示为使用的格式。

```
% 宏变量 = 表达式
```

- 可以在宏变量名称中使用字母数字和下划线 (_)。不过，不能在宏变量名称开头使用数字。
- 可以在宏变量中指定分配任何表达式，该表达式的值是一个介于 0h 和 FFFFFFFFh 之间的整数。如果指定负数，它会被当作二的补码来处理。
- 可以在表达式内部使用宏变量。
- 请总是在宏变量之前添加 “%”符号。

8.1.3 条件语句

在条件语句中，会根据条件真伪来执行不同的语句。以下所示为使用的格式。

```
if （表达式）
     语句 1
 else
     语句 2
 endi
```

- 如果表达式为真 （非 0），则执行 **语句** 1。如果为伪 (0)，则执行 **语句** 2。
- 可以省略 else 语句。如果省略并且表达式为伪，则执行会跳到 endi 语句之后的行。
- if 语句可以嵌套 （最多 32 层）。

8.1.4 循环语句 （while 及 endw）和中断语句

在循环语句中，当表达式为真时，重复执行一组语句。以下所示为使用的格式。

```
while （表达式）
     语句
 endw
```

- 如果表达式为真，则重复该组语句。如果为伪，则退出循环 （执行 endw 语句之后的语句）。
- 最多可以将 while 语句嵌套 32 层。
- 使用中断语句可以强制退出 while 循环。如果 while 语句是嵌套的，则中断会从最里面的循环退出。

8.1.5 注释语句

可以在脚本文件中包括注释。请使用以下格式。

```
; 字符串
```

- 在分号 (;) 后面编写语句。分号前面只能包括空格和制表符。
- 执行脚本文件时，会忽略注释语句行。

8.2 编写表达式

此调试器允许使用表达式来指定地址、数据和传递数等等。以下所示为使用了表达式的示例命令。

```
>DumpByte TABLE1
>DumpByte TABLE1+20
```

可以在表达式中使用以下元素：

- 常数
- 符号和标签
- 宏变量
- 寄存器变量
- 存储器变量
- 行号
- 字符常数
- 运算符

8.2.1 常数

可以使用二进制、八进制、十进制或十六进制数。附加到数值上的前缀或后缀符号指示使用的是哪一种基数。

M32C、 M16C/R8C 和 740 的调试器

	十六进制	十进制	八进制	二进制 *
前缀	0x、 0X	@	无	%
后缀	h、 H	无	o、 O	b、 B
示例	0xAB24 AB24h	@1234	1234o	%10010 10010b

* 当预先确定的基数是十六进制数时，只能指定 %。

- 如果输入的是一个与预先确定的基数匹配的基数，则可以省略指示该基数 （二进制数除外）的符号。
- 使用 RADIX 命令可以设置预先确定的基数值。不过，在下面显示的情况中，无论在 RADIX 命令中指定什么内容，基数都是固定的。

类型	基数
地址	十六进制
行号 执行次数 通过次数	十进制

8.2.2 符号和标签

可以包括目标程序中定义的符号和标签，或者包括使用汇编命令定义的符号和标签。

- 可以在符号和标签中包括字母数字、下划线 (_)、句点 (.) 和问号 (?)。不过，请不要以数字开头。
- 符号和标签最多可以由 255 个字符组成。
- 有大写字母和小写字母的区分。

产品名称	说明
用于 740 的调试器	• 不能使用寄存器名称。（A、 X、 Y、 S、 PC、 PS、 P） • 不能包括汇编器结构化指令、伪指令、宏指令、操作码或保留字（.SECTION、 .BYTE、 switch、 if 等等）。 • 不能使用以两个句点 (..) 开头的字符串作为符号或标签。 D0 到 .D65535、 .F0 到 .F65535、 .I0 到 .I56635、 .S0 到 .S65535、 ..0 到 ..65535、 ??0 到 ??65535

8.2.2.1 局部标签符号和作用域

此调试器支持全局标签符号 （可以从整个程序区引用全局标签符号）和局部标签符号 （只能在声明局部标签符号的文件内部引用局部标签符号）。局部标签符号的有效范围就是通常所说的作用域，它是以目标文件为单位计量的。在以下情况下，作用域会在此调试器中发生转变：

- 输入命令时

  包括程序计数器所指示地址的目标文件会成为当前作用域。当使用 SCOPE 命令设置作用域时，指定的作用域为有效作用域。

- 在命令执行过程中

  当前作用域会自动根据命令正在处理的程序地址而转变。

8.2.2.2 标签和符号的优先级

将值转换为标签或符号 （反之亦然）时，应遵从以下优先级：

- 转换地址值

1. 局部标签
2. 全局标签
3. 局部符号
4. 全局符号
5. 作用域之外的局部标签
6. 作用域之外的局部符号

- 转换数据值

1. 局部符号
2. 全局符号
3. 局部标签
4. 全局标签
5. 作用域之外的局部符号
6. 作用域之外的局部标签

- 转换位值

1. 局部位符号
2. 全局位符号
3. 作用域之外的局部位符号

8.2.3 宏变量

宏变量是由脚本文件中的赋值语句定义的。有关详细信息，请参阅 “参考”部分中的 “8.1.2 赋值语句”一节。请在变量之前添加 “%”，以便用作宏变量。

- 可以在百分比符号 (%) 后面的变量名称中指定字母数字和/或下划线 (_)。不过，请不要以数字作为名称的开头。
- 不能使用寄存器的名称作为变量名称。
- 变量名称区分大写字母和小写字母。
- 最多可以定义 32 个宏变量。一旦定义，宏变量将保持有效，直到退出调试器。

宏变量用于指定 while 语句的重复次数。

8.2.4 寄存器变量

如果使用寄存器变量，则可以在表达式中使用寄存器的值。在寄存器名称之前添加 “%”可以将其用作寄存器变量。请使用以下格式。（用于 740 的调试器可使用 “_”，而不使用 “%”。）

产品名称	寄存器名称
用于 740 的调试器	PC、 A、 X、 Y、 S、 PS

在寄存器名称中不区分大小写字母。可以指定任一形式的字母。

8.2.5 存储器变量

如果使用存储器变量，则可以在表达式中使用存储器值。格式如下：

[地址]. 数据大小

- 可以在地址中指定表达式 （也可以指定存储器变量）。
- 如下表所示指定数据长度。（用于 740 的调试器不支持四字节长度。）

数据长度	调试器	规格
1 个字节	全部	B 或 b
2 个字节	用于 M32R 的调试器	H 或 h
	其他	W 或 w
4 个字节	用于 M32R 的调试器	W 或 w
	用于 M32R、 M16C/R8C 的调试器	L 或 l

示例：引用位于地址 8000h 的存储器的内容 （数据长度为 2 个字节）

```
[0x8000].W
```

- 如果没有指定，则默认数据长度为字。

8.2.6 行号

指源文件行号。行号格式如下：

 # 行号
 # 行号 . “源文件名称”

- 使用十进制数指定行号。
- 只能指定可以在其中设置软件断点的行号。不能指定未生成任何汇编器指令的行，包括注释行和空白行。
- 如果省略源文件名称，则对应源文件中的行号会显示在有效 [Editor (Source)] （编辑器 （源））窗口中。
- 在源文件名称中包含文件属性。
- 请不要在行号和源文件名称之间添加任何空格。

8.2.7 字符常数

指定的字符或字符串会转换为 ASCII 码并作为常数处理。

- 将字符括在单引号中。
- 将字符串括在双引号中。
- 字符串必须由一个或两个字符组成 （最多 16 位）。如果指定的字符超过两个，则会处理该字符串的后两个字符。例如，“ABCD”会当作 “CD”或值 4344h 来处理。

8.2.8 运算符

下表列出可以在表达式中使用的运算符。

- 运算符的优先级由级别表示，级别 1 最高，级别 8 最低。如果两个或更多个运算符具有相同优先级，则从表达式左边开始依次进行计算。

运算符	功能	优先级
( )	括号	级别 1
+、 -、 ~	一元正、一元负、一元逻辑非	级别 2
*、 /	二元乘法、二元除法	级别 3
+、 -	二元加法、二元减法	级别 4
>>、 <<	右移、左移	级别 5
&	二元逻辑 “与”	级别 6
|、 ^	二元逻辑 “或”、二元逻辑 “异 （或）”	级别 7
<、 <=、 >、 >=、 ==、 !=	二元比较	级别 8

9 C/C++ 表达式

9.1 编写 C/C++ 表达式

使用由如下所示标记组成的 C/C++ 表达式，可以将 C 监视点写入寄存器，并指定要赋给这些监视点的值。

标记	示例
立即值	10、 0x0a、 012、 1.12、 1.0E+3
作用域	::name、 classname::member
数学运算符	+、 -、 *、 /
指针	*、 **、 ...
引用	&
取反	-
使用点运算符的成员引用	Object.Member
使用箭头运算符的成员引用	Pointer → Member、 this → Member
指向成员的指针	Object.*var、 Pointer → *var
括号	(,)
数组	Array[2]、 DArray[2] [3]...
基本数据类型转换	(int)、 (char*)、 (unsigned long *)...
typedef 类型转换	(DWORD)、 (ENUM)...
变量名和函数名	var、 i、 j、 func...
字符常数	'A'、 'b'...
字符串文字	"abcdef"、 "I am a boy."...

9.1.1 立即值

可以将十六进制数、十进制数和八进制数用作立即值。以 0x 开头的值将视为十六进制数进行处理，以 0 开头的值被视为八进制数，不使用这两种前缀的值被视为十进制数。

还可以使用浮点数给变量赋值。

注意

- 不能只注册立即值作为 C 监视点。
- 仅当立即值在用于指定 C/C++ 监视点的 C/C++ 语言表达式中使用，或者用于指定要赋给这些表达式的值时，该立即值才有效。无法对类似于 1.0+2.0 这样的使用浮点数的表达式执行运算。

9.1.2 作用域解析

作用域解析运算符 :: 的使用方式如下。

全局作用域：::valiable name

```
::x、 ::val
```

类作用域：class name::member name, class name::class name::member name。例如：

```
T::member, A::B::member
```

9.1.3 数学运算符

可以使用加号 (+)、减号 (-)、乘号 (*) 和除号 (/) 数学运算符。如下所示为上述数学运算符用于求值时的优先级顺序。

```
(*), (/), (+), (-)
```

注意

- 目前不支持对浮点数使用数学运算符。

9.1.4 指针

指针由星号 (*) 指示。可以使用指向指针的指针 **，以及指向指针的指针的指针 ***，依次类推。

示例：“* 变量名称”、“ 变量名称”等**

注意

- 不能把立即值当作指针进行处理。例如，不能指定 *0xE000。

9.1.5 引用

引用由 “&” 符指示。只能指定 “& 变量名称” 这种形式。

9.1.6 取反

取反由减号 (-) 指示。只能指定 “- 立即值”或 “- 变量名称”这种形式。如果指定了 2 个 （或任意偶数个）减号，则不会执行取反。

注意

- 目前不支持对浮点数取反。

9.1.7 使用点运算符的成员引用

使用点运算符时只能通过 “变量名称 . 成员名称”的形式引用结构及联合的成员。
示例：

```
class T {
public:
int member1;
char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

在这种情况下， t_cls.member1 和 (*pt_cls).member2 均可正确引用成员。

9.1.8 使用箭头运算符的成员引用

使用箭头运算符时只能通过 “变量名称→成员名称”的形式引用结构及联合的成员。
示例：

```
class T {
public:
int member1;
char member2;
};
class T t_cls;
class T *pt_cls = &t_cls;
```

在这种情况下， (&t_cls) → member1 和 pt_cls → member2 均可正确引用成员。

9.1.9 指向成员的指针

对于使用 “.*” 或 “→ *” 运算符指向成员的指针，只能通过如下形式进行引用：变量名称 .* 成员名称或变量名称 → * 成员名称。

示例：

```
class T {
public:
int member;
};
class T t_cls;
class T *pt_cls = &t_cls;
int T::*mp = &T::member;
```

在这种情况下， t_cls.*mp 和 tp_cls → *mp 可正确引用成员指针类型的变量。

注意

- 请注意，不能将表达式 *mp 视为成员指针类型的变量。

9.1.10 括号

使用 “(” 和 “)” 可指定表达式中计算的优先级。

9.1.11 数组

可以使用 “[” 和 “]” 指定数组的元素。可以通过如下方式指定数组：“变量名称 [(元素序号或变量)]”、“变量名称 [(元素序号或变量)] [(元素序号或变量)]”，等等。

9.1.12 基本数据类型转换

可以对 C 基本数据类型 （如 char、 short、 int 和 long）执行转换，还可以将指针类型转换成这些基本数据类型。执行指针数据类型转换时，可以使用的指针数据类型包括指向指针的指针，以及指向指针的指针的指针，等等。

请注意，如果未指定是否带符号，则默认值如下所示：

基本数据类型	默认值
char	无符号
short	带符号
int	带符号
long	带符号

注意

- 在 C++ 的基本数据类型中，无法对 bool 类型、 wchar_t 类型和 floating-point 类型 （浮点值或双精度值）数据进行转换。
- 无法执行对寄存器变量的转换。

9.1.13 typedef 类型转换

可以执行对 typedef 类型 （除 C 基本数据类型以外的类型）的转换，以及将指针类型转换为 typedef 类型。执行指针数据类型转换时，可以使用的指针数据类型包括指向指针的指针，以及指向指针的指针的指针，等等。

注意

- 无法执行对 struct 类型或 union 类型的转换，无法将指针类型转换成这些类型。

9.1.14 变量名

可以根据 C/C++ 惯例使用以英文字母开头的变量名。
变量名允许的最大字符数为 255。
可以使用 “this”指针。

9.1.15 函数名

可以根据 C 惯例使用以英文字母开头的函数名。

注意

- 对于 C++，不能使用任何函数名。

9.1.16 字符常数

可将由单引号 (') 引住的字符用作字符常数。例如， 'A'、 'b' 等等。这些字符常数将转换为 ASCII 码，并用作单字节的立即值。

注意

- 不能只是注册字符常数作为 C 监视点。
- 仅在指定 C 监视点的 C/C++ 表达式中作为字符常数使用，或者用于指定要赋予的值时，这些字符常数才有效 （处理字符常数的方式与处理立即值的方式相同）。

9.1.17 字符串文字

可将由双引号 (“ ”) 引住的字符串用作字符串文字。例如，“abcde”、“I am a boy.”等等。

注意

- 在表达式中，字符串文字只能位于赋值运算符的右侧。使用字符串文字的前提是赋值运算符左侧必须为 char 数组类型或 char 指针类型。否则会导致语法错误。

9.2 C/C++ 表达式的显示格式

在 [C Watch] （C 监视）窗口的数据显示区域中， C/C++ 表达式将显示为类型名称、 C/C++ 表达式（变量名）以及计算结果 （值），如下所示。

下文将分别介绍各类型的显示格式。

9.2.1 枚举类型

- 如果计算结果 （值）已确定，则显示其名称。

```
(DATE) date = Sunday （所有基数） _
```

- 如果计算结果 （值）尚未确定，则其显示如下：

```
(DATE) date = 16 （如果基数为初始状态）
(DATE) date = 0x10 （如果基数为十六进制数）
(DATE) date = 0000000000010000B （如果基数为二进制数）
```

9.2.2 基本数据类型

- 当计算结果为除 char 类型或浮点类型以外的基本数据类型时，其显示如下：

```
(unsigned int) i = 65280 （如果基数为初始状态）
(unsigned int) i = 0xFF00 （如果基数为十六进制数）
(unsigned int) i = 1111111100000000B （如果基数为二进制数）
```

- 当计算结果为 char 类型时，其显示如下：

```
(unsigned char) c = 'J' （如果基数为初始状态）
(unsigned char) c = 0x4A （如果基数为十六进制数）
(unsigned char) c = 10100100B （如果基数为二进制数）
```

- 当计算结果为浮点类型时，其显示如下：

```
(double) d = 8.207880399131839E-304 （如果基数为初始状态）
(double) d = 0x10203045060708 （如果基数为十六进制数）
(double) d = 0000000010.....1000B （如果基数为二进制数）
(..... 表示省略的内容）
```

9.2.3 指针类型

- 当计算结果为非 char* 类型的指针数据类型时，其将以十六进制形式显示如下：

  (unsigned int *) p = 0x1234 （所有基数）

- 当计算结果为 char* 类型时，可以在 [C Watch] （C 监视）窗口的 [Display String] （显示字符串）菜单中选择字符串或字符形式的显示格式。

  — 字符串类型

  (unsigned char *) str = 0x1234 "Japan" （所有基数）

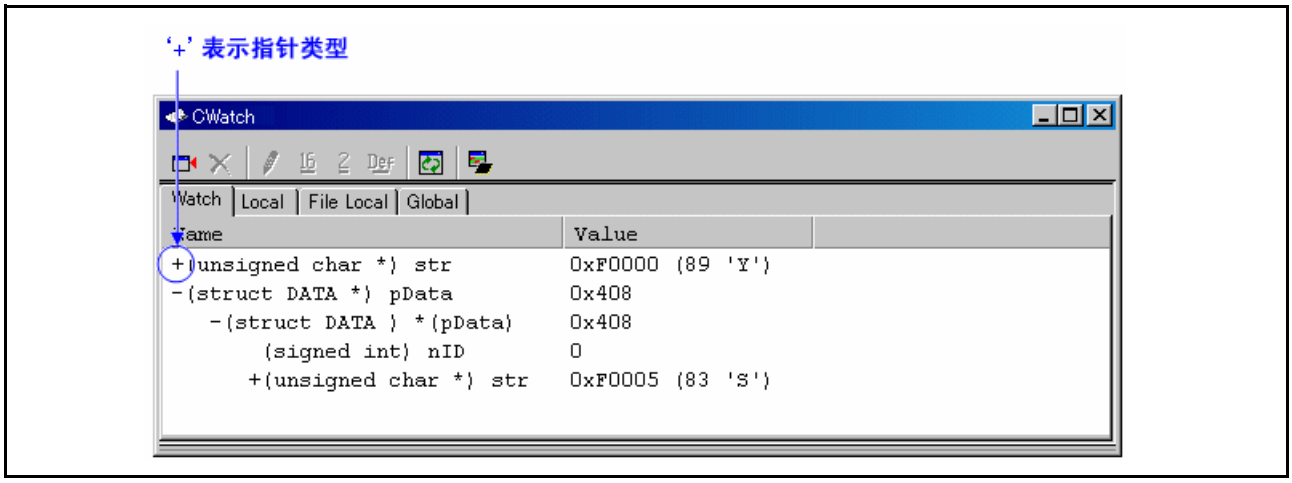
  — 字符类型

  (unsigned char *) str = 0x1234 (74 'J') （所有基数）

当计算结果为 char* 类型时，其显示如下：

(unsigned char *) str = 0x1234 "Jap （所有基数）

如果字符串中某个符号前包含一个非打印符号，用以显示字符串结束符 (0)，则字符串仅显示到此非打印字符位置，并且不显示右引号。



可以双击由 “+” 指示的行，查看该结构或联合的成员。显示成员时 “+” 变为 “-”。要恢复初始显示，双击现由 “-” 指示的那一行。

9.2.4 数组类型

- 当计算结果为非 char [ ] 类型的数组数据类型时，起始地址将按以下形式以十六进制数显示：

  (signed int [10]) z = 0x1234 （所有基数）

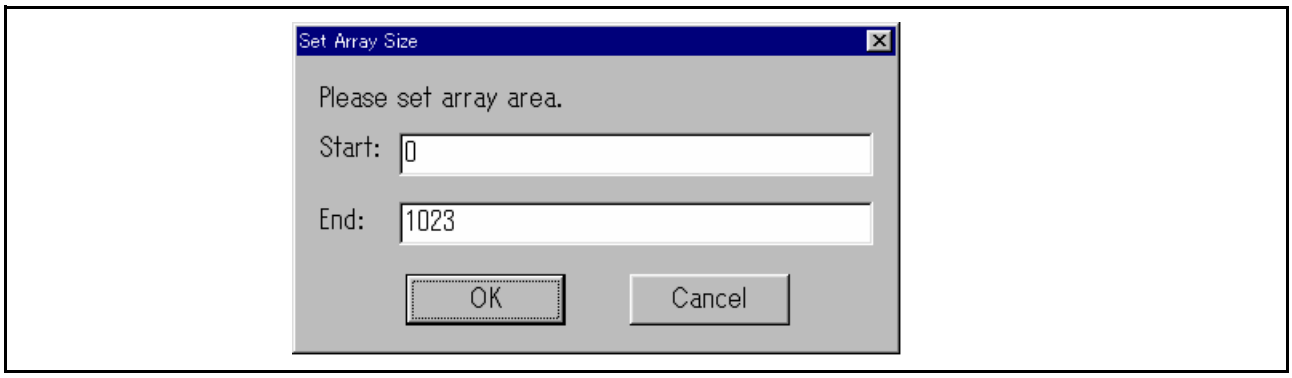
- 当计算结果为 char [ ] 类型时，其显示如下：

  (unsigned char [10]) str = 0x1234 "Japan" （所有基数）

如果字符串中某个符号前包含一个非打印符号，用以显示字符串结束符 (0)，则字符串仅显示到此非打印字符位置，并且不显示右引号。

(unsigned char [10]) str = 0x1234 "Jap （所有基数）

如果字符串字符个数超过 80，同样也不会显示右引号。当 C/C++ 表达式是数组类型时，与指针类型一样，类型名称左侧将显示 “+”。可以使用此指示符号查看数组的元素。（有关详细信息，请参阅 “9.2.3 指针类型”）当数组元素个数超过 100 时，将打开如下对话框。请在对话框中指定元素个数。



将会显示此处指定的从 “Start”（开始）至 “End”（结束）的下标所对应的元素。如果指定的值超出该数组的最大下标，则该值将被视作数组的最大下标。如果单击 “Cancel”（取消）按钮，则不会显示元素。

9.2.5 函数类型

- 当计算结果为函数数据类型时，起始地址将以十六进制形式显示如下：

  (void()) main = 0xF000 （所有基数）

9.2.6 引用类型

- 当计算结果为引用数据类型时，引用地址将以十六进制形式显示如下：

  (signed int &) ref = 0xD038 （所有基数）

9.2.7 位域类型

- 当计算结果为位域数据类型时，其显示如下：

  (unsigned int :13) s.f = 8191 （如果基数为初始状态）
  (unsigned int :13) s.f = 0x1FFF （如果基数为十六进制数）
  (unsigned int :13) s.f = 1111111111111B （如果基数为二进制数）

9.2.8 未找到任何 C 符号时

- 如果所计算的表达式包含找不到的 C 符号，其显示如下：

```
() x = <not active>（所有基数）
```

9.2.9 语法错误

- 当所计算的表达式包含语法错误时，其显示如下：

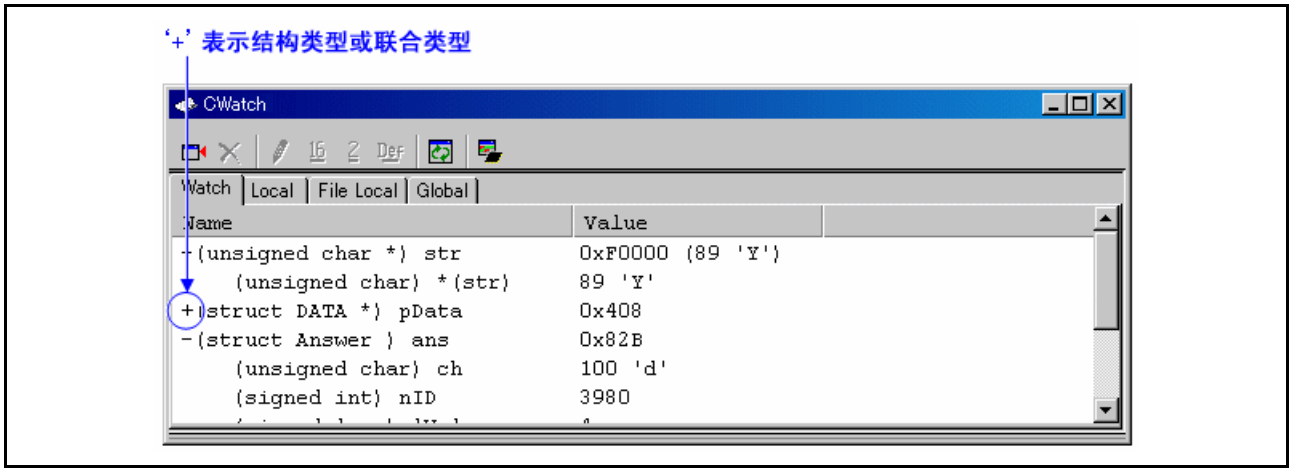
```
() str*(p = <syntax error>（所有基数）
（其中 str*(p 为该语法错误）
```

9.2.10 结构类型和联合类型

- 当计算结果为结构或联合数据类型时，地址将以十六进制形式显示如下：

```
(Data) v = 0x1234（所有基数）
```

如果 C/C++ 表达式像结构或联合一样包含成员，则类型名称（标记名称）左侧会显示“+”。



可以双击由“+”指示的行，查看该结构或联合的成员。显示成员时“+”变为“-”。要恢复初始显示，双击现由“-”指示的那一行。可以使用此功能查看结构或联合的成员。

注意事项

如果声明的变量与通过 typedef 声明的类型定义同名，则无法引用该变量。

- 寄存器变量

当计算结果为寄存器变量时，类型名称左侧将显示“register”，如下所示：

```
(register signed int) j = 100
```

10 显示程序停止的原因

如果程序被调试功能停止，则停止原因将显示在 [Output] （输出）窗口或 [Status] （状态）窗口 （[Platform] （平台）页）中。

显示内容及 “停止原因”说明如下所示：

显示内容	停止原因
Halt	由 [Halt Program] （暂停程序）按钮 / 菜单停止
S/W break	软件断点
Address match interrupt break	地址中断断点
H/W event, Combination	硬件断点，符合逻辑组合 [AND] （与）或 [AND(same time)] （与 （同时））条件
H/W event, Combination, Ax	硬件断点，符合逻辑组合 [OR] （或）条件 （Ax：符合条件的事件编号。）
H/W event, State transition, from xx	硬件断点，符合 [State Transition] （状态转换）条件 （from xx：上一状态 （start、 state1、 state2））
H/W event, State transition, Timeout	硬件断点，符合 [State Transition] （状态转换）和 [Time Out] （超时）条件
H/W event, Access protect error	中断保护

注意

能否显示中断原因取决于所连接的目标。一些目标可能总是显示 “Halt”（暂停）或者显示 “---”。

11 注意事项

11.1 通用注意事项

11.1.1 与 Windows 相关的文件操作

应注意下面几点：

1. 文件名和目录名称
   - 不要使用包含空格的目录名或文件名。
   - 如果目录名和文件名中包含日文汉字，则不保证操作正常。
   - 文件名中只能有一个扩展名分隔符号 (.)。
2. 指定文件和目录
   - 不能使用 “…”指定两层以上的目录。
   - 不能使用网络路径。必须分配一个驱动器。

11.1.2 可设置软件断点的区域

可设置软件断点的区域因 MCU 类型的不同而异。

只有 ROM 区域是在设置为 Internal 的区域中进行存储映射的，才能在该区域设置软件断点。

如果 ROM 区域是在 SFR 区域、 RAM 区域或其他设置为 External 的区域中进行存储器映像的，则不能在该区域中设置软件断点。

11.1.3 获取或设置 C 变量

- 如果声明的变量与通过 typedef 声明的类型定义同名，则无法引用该变量。
- 无法更改寄存器变量和位字段的值。
- 无法更改 64 位宽变量 （long long、 double 等）的值。
- 无法更改未指定存储地址和大小的 C/C++ 表达式的值。
- 出于优化考虑， C 编译器可能在相同地址放置不同的变量。在这种情况下， C 变量的值可能无法正常显示。
- 文字字符串只能代换 char 数组和 char 指针类型的变量。
- 对浮点类型无法执行任何数学运算。
- 对浮点类型无法执行任何取反操作。
- 对浮点类型无法执行转换。
- 对寄存器变量无法执行转换。
- 无法转换结构类型或联合类型，无法将指针类型转换成结构类型或联合类型。
- 字符常数和文字字符串不能包含转义序列。

11.1.4 C++ 中的函数名

- 进行显示地址、断点等设置时，如果使用函数名输入地址，则不能指定类的成员函数、运算符函数和过载函数。
- 不能在 C/C++ 表达式中使用函数名。
- 为参数所指定的函数名中不能使用任何脚本命令 （例如， breakin 和 func）。
- 在对话框的地址值指定列中，任何地址均不能使用函数名进行指定。

11.1.5 调试多个模块

如果在单个会话中注册两个或两个以上绝对模块文件，则一次只能下载一个文件。
如果在单个会话中注册一个绝对模块文件和一个或多个机器语言文件，则可以同时下载所有文件。

11.1.6 同步调试

同步调试功能不可用。

11.1.7 小型仿真器复位开关

如果小型仿真器的系统复位功能不正常，请终止调试器，再次开启小型仿真器，然后重新启动调试器。
重新下载程序。

11.2 740 调试器的注意事项

11.2.1 设置存储器映像

在仿真器启动后， 0h-3FFFh 的映像属性为 “External”（外部）， 4000h-FFFFh 的映像属性为 “Internal”（内部）。必须改变存储器映像信息，以适合目标单片机的存储器空间。这与过去 ROM 区域内部的单片机从 4000h 开始的情况类似。

有关详细信息，请参阅 “参考”部分中的 “4.3 设置 740 使用的调试器”一节。

11.2.2 仿真器使用的堆栈区域

仿真器将用户堆栈区域用作工作区域 （3 字节）。
开始调试之前，请务必保留用户堆栈区域，再外加 3 字节区域。

11.2.3 看门狗定时器

允许看门狗定时器时，目标程序自由运行以外的操作都会被禁止。开始调试之前，应禁止使用看门狗定时器。

11.2.4 C 编译器 / 汇编器 / 连接器选项

根据编译器、汇编器和连接器的选项指定，有时无法正常下载 / 调试信息。
请参阅以下内容了解选项的指定。
请参阅 “11.3 C 编译器 / 汇编器 / 连接器选项”

740 调试器可使用的编译器：

- 用于 740 族的汇编器套件 SRA74
- IAR C 编译器

11.2.5 调试 16 位定时器功能

支持 16 位定时器的单片机 （38B5 组等等）具有下述限制。

- [注意事项 1]
  在写入 16 位定时器的高/低位字节的过程中，如果程序执行由于中断或单步执行而暂停，则 16 位定时器的输出可能在 [Dump] （转储）窗口等位置无效。
  例如：

```
[TIMER_LOW] = [DATA1]
[TIMER_HIGH] = [DATA2] <- 在此处发生中断时
```

- [注意事项 2]
  在写入 16 位定时器的高/低位字节的过程中，如果程序执行由于中断或单步执行而暂停，则向 16 位定时器写入值 （利用 [Dump] （转储）窗口）的操作将失败。
  例如：

```
[DATA1] = [TIMER_LOW]
[DATA2] = [TIMER_HIGH] <- 在此处发生中断时
```

这些问题是由于单片机的规格而导致的。在具有 16 位定时器的单片机中，定时器按先低位字节再高位字节的顺序写入。定时器读取操作则按相反的顺序进行。

因此，如果发生如注意事项 1 或 2 所示的中断，则在 [Dump] （转储）窗口中显示或设置 16 位定时器的值时，将读取或写入错误的值。

11.2.6 关于 MCU 的内部 RAM 区域中的单步执行和程序断点功能

使用仿真主机 M38000L2-FPD 进行调试时，内部 RAM 区域中的单步执行和程序断点功能将不可用。
调试传送到内部 RAM 区域的程序时，请使用自由运行功能和跟踪功能。

11.2.7 硬件事件

在位存取过程中，如果对包含指定位的地址中的其它位进行了存取，则事件可能会在以下数据存取中生效。

- 硬件断点事件

11.3 C 编译器 / 汇编器 / 连接器选项

根据编译器、汇编器和连接器的选项指定，有时无法正常下载 / 调试信息。
请参阅以下内容了解选项的指定。

在除上述选项之外的选项中，不会进行操作检查。请注意，除上述介绍的选项外不推荐使用其他选项。

11.3.1 使用 IAR C 编译器 (EW)

请按如下步骤指定工程设置：

1. IAR Embedded Workbench 中的设置

   选择 [Project]（工程）→ [Options...]（选项…）菜单时，将打开 “Options For Target “ target”” （目标 “target”的选项）对话框。在此对话框中，请选择 “XLINK”类别，然后设置工程设置。

   — [Output]（输出）选项卡

   在 “Format”（格式）区域，选中 “Other”（其他）选项，然后选择 “ieee-695”作为 “Output Format”（输出格式）。

   — [Include]（包含）选项卡

   在 “XCL File Name”（XCL 文件名）区域中，指定 XCL 文件 （例如， lnkm16c.xcl）。

2. 编辑 XCL 文件

   将命令行选项 “-y”添加到 XCL 文件。对 “-y”选项的指定因产品而异。

产品名称	-y 选项
用于 740 的调试器	-ylmba

3. 完成上述设置后创建程序。

在除上述选项之外的选项中，不会进行操作检查。请注意，除上述介绍的选项外不推荐使用其他选项。

11.3.2 使用 IAR C 编译器 (ICC)

11.3.2.1 指定选项

请根据以下步骤进行编译和连接。

- 编译时

指定 “-r” 选项

- 连接前

进行连接时，请打开要读取的连接器选项定义文件 (extension .xcl) 并添加 “-FIEEE695” 和 “-y” 选项。对 “-y” 选项的指定因产品而异。

产品名称	-y 选项
用于 740 的调试器	-ylmba

- 连接时

使用 “-f” 选项指定连接器的选项定义文件。

在除上述选项之外的选项中，不会进行操作检查。请注意，除上述介绍的选项外不推荐使用其他选项。

11.3.2.2 命令执行示例

以下示例显示如何针对不同的产品输入不同的命令

- 用于 740 的调试器

```
>ICC740 -r file1.c<Enter>
>ICC740 -r file2.c<Enter>
>XLINK -o filename.695 -f lnk7400t.xcl file1 file2<Enter>
```

XCL 文件名因不同的产品和存储器型号而异。有关详细信息，请参阅 ICCxxxx 手册。

11.3.3 使用用于 740 族的汇编器套件时

请根据以下步骤进行汇编和连接。

- 汇编时
  - — “-c”选项
    将与源代码行有关的调试信息输出到可重新定位的文件。

    注意
    为源文件中的某个函数指定了指令 .FUNC 时，如果使用 “-c”选项，该函数的名称将不可用。若要使名称可用，则不要使用此选项。
  - — “-s”选项
    将局部标签、局部 .equ 符号和局部 .bequ 符号输出到可重新定位的文件。
- 连接时
  - — “-s”选项
    生成符号文件。

在除上述选项之外的选项中，不会进行操作检查。请注意，除上述介绍的选项外不推荐使用其他选项。

11.3.3.1 命令执行示例

以下示例显示如何针对不同的产品输入不同的命令

- 用于 740 的调试器

```
>sra74 -c -s main.a74<Enter>
>sra74 -c -s sub.a74<Enter>
>link74 main sub ,,,-s<Enter>
```

[备注]

修订记录	740 小型仿真调试器 V.1.02

Rev.	发行日	修订内容	
		页	修订处
1.00	2008.01.10	—	初版发行

740 小型仿真调试器 V.1.02 用户手册

Publication Date:	Rev.1.00, Jan. 10, 2008
Published by:	Sales Strategic Planning Div. Renesas Technology Corp.
Edited by:	Customer Support Department Global Strategic Communication Div. Renesas Solutions Corp.

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 (Shanghai) Co., Ltd.
Unit 204, 205, AZIACenter, No.1233 Lujiazui Ring Rd, Pudong District, Shanghai, China 200120
Tel: <86> (21) 5877-1818, Fax: <86> (21) 6887-7898

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

Renesas Technology Korea Co., Ltd.
Kukje Center Bldg. 18th Fl., 191, 2-ka, Hangang-ro, Yongsan-ku, Seoul 140-702, Korea
Tel: <82> (2) 796-3115, Fax: <82> (2) 796-2145

Renesas Technology Malaysia Sdn. Bhd
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No.18, Jalan Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: <603> 7955-9390, Fax: <603> 7955-9510

Colophon 6.0

740 小型仿真调试器V.1.02

RENESAS

瑞萨电子株式会社

RCJ10J0078-0100