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

M16C R8C FOUSB/UART 软件

瑞萨单片机开发环境系统

瑞萨电子
www.renesas.com

Rev.1.00 2006.10

Keep safety first in your circuit designs!

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

Notes regarding these materials

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

请遵循安全第一进行电路设计

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

关于利用本资料时的注意事项

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

概述

High-performance Embedded Workshop 是一种为了让用户更加快捷、方便的针对瑞萨单片机进行 C/C++/汇编语言开发和调试的用户图形界面，其目标是提供一个强大且直观的方法来调试和修改用户的应用程序。

本篇资料详细阐述了 High-performance Embedded Workshop 作为“调试器”的功能。

目标系统

本调试器应用于 FoUSB/UART 或 Starter-kit。

所支持的 CPU

本篇资料所支持的 CPU 如下所示：

M32C/80，M16C/80 系列

注：在本篇资料中，有关上述 CPU 的信息会被标注为 “对于 M32C”。

M16C/60，M16C/Tiny，M16C/20，M16C/10，R8C/Tiny 系列

注：在本篇资料中，有关上述的 CPU 的信息会被标注为 “对于 M16C/R8C”。

（空白页）

目录

设置调试器

（空白页）

1. 特性

1.1 断点功能

本调试器具有如下功能

1.1.1 软件断点功能

软件断点会在执行某个特定地址的指令前中断调试目标程序。这种断点叫做软件断点。软件断点可以在 Editor（Source）Window 或 S/W Break Point Setting Window 里面进行设置或取消。用户也可以暂时禁用/启用一个软件断点。断点的数量取决于所连接的 MCU 种类。当设置了两个或更多的软件断点后，中断之间是“或”的逻辑关系（目标程序的任何一个断点都会引发中断）。

(1) 设置软件断点

软件断点可以在下列窗口中进行设置：

- Editor（Source）Window
- S/W Break Point Setting Window

用户可以在 Editor（Source）Window 中，通过双击鼠标来设置或取消软件断点。

用户也可以在 S/W Break Point Setting 窗口中暂时地禁用/启用一个软件断点。

(2) 程序中可设置软件断点的位置

可以设置软件断点的位置因 MCU 而异。请参阅如下章节：

“10.1.2 设置软件断点的空间”

1.2 实时操作系统调试功能

该功能用于调试使用实时操作系统的目标程序，可以指示实时操作系统的运行状态。

1.3 用户图形界面输入/输出功能

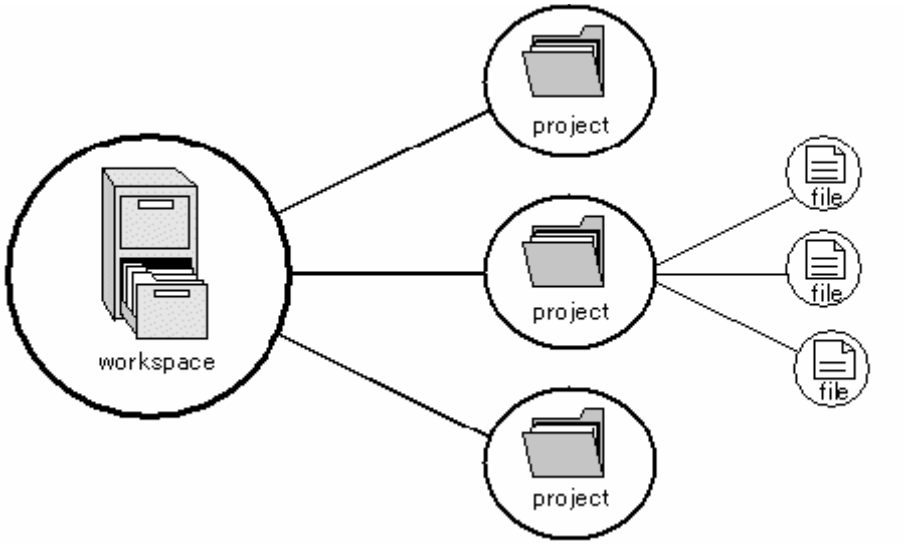
本功能可以将用户系统的键输入和输出在窗口中进行仿真。输入面板中可以使用按钮，输出面板中可以使用字符串和 LED。

2. 使用前的准备

请先运行 High-performance Embedded Workshop 并且连接仿真器。另外，为了使用本产品进行调试，需要创建一个工作空间（Workspace）。

2.1 工作空间、工程和文件

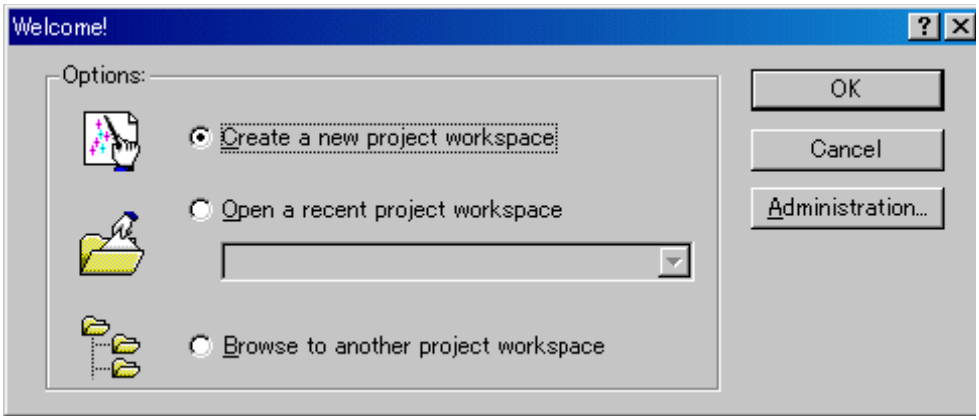
正如一个文字处理程序能让用户创建和修改文档一样，本产品可以让用户创建和修改工作空间。一个工作空间可以被假想为一个装入工程的容器。同样的，一个工程可以被假想为一个装入工程文件的容器。一个工作空间可以包含一个或多个工程，每个工程又可以包含一个或多个文件。



工作空间可以将有关联的工程汇总在一起。举例来说，某一个应用方案可能需要为多种不同的处理器做编译，或者可能某一个应用方案和一个程序库正在同时开发之中。各个工程可以按等级被关联在一个工作空间内，这就意味着，顶级的工程具有高优先级，可以先于另外的工程被编译。无论如何，独立的工作空间并不太常用，必须把一个工程添加到一个工作空间，在把文件添加进这个工程中才能进行开发。

2.2 开始使用 High-performance Embedded Workshop

从“开始”菜单的“程序”里面启动 High-performance Embedded Workshop。首先会看到欢迎对话框。



在这个对话框中，可以创建新工作空间或选择已有工作空间。

[Create a new project workspace] 单选按钮：创建一个新的工作空间。

[Open a recent project workspace] 复选框：打开一个已经创建的工作空间并且显示最近使用过的工作空间。

[Browse to another project workspace] 单选按钮：使用一个已经创建的工程；当上方的历史菜单中没有所需的工程时使用本按钮。

当需要选择一个已经存在的工作空间时，选择 [Open a recent project workspace] 或 [Browse to another project workspace]复选框然后选择工作空间文件（.hws）。

关于如何创建一个新的工作空间请参考以下章节

参考“2.2.1 创建一个新的工作空间（使用工具链）”

参考“2.2.2 创建一个新的工作空间（不使用工具链）”

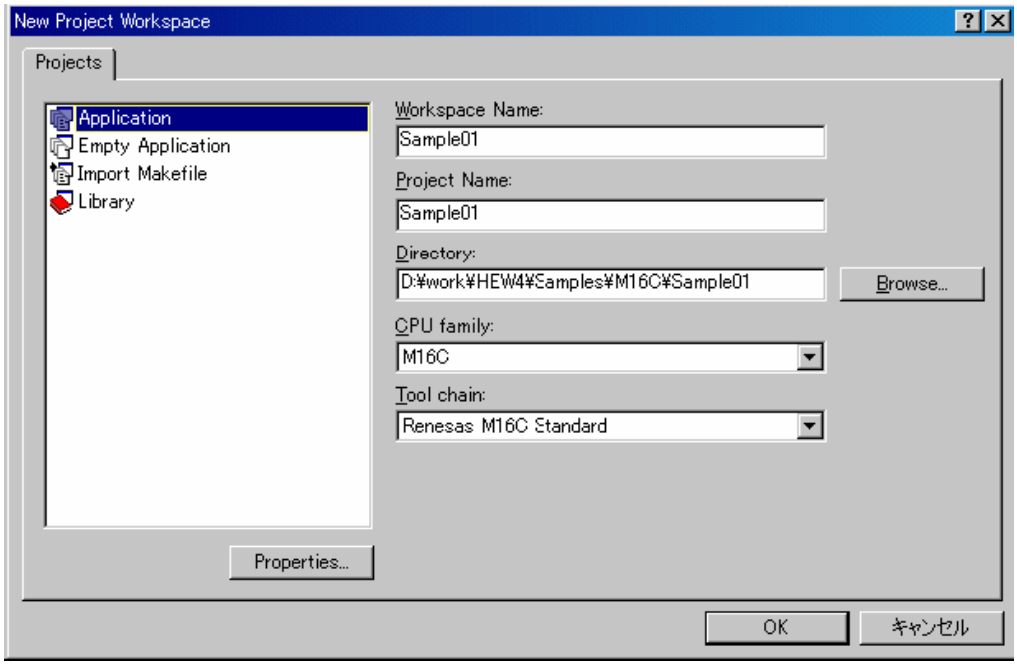
* 当调试现有的加载模块时，使用上述方法创建一个工作空间。

创建一个新工作空间的方法根据是否使用工具链的与否而不同。注意，本产品并不含工具链。当安装了与所使用的 CPU 相对应的 C/C++编译器开发环境以后，才可以使用工具链。具体信息请参考 C/C++编译器所附带的使用手册。

2.2.1 创建一个新的工作空间（使用工具链）

(1) 步骤 1：创建一个新的工作空间

在打开 High-performance Embedded Workshop 后出现的[Welcome!]对话框中，选择[Create a new project workspace]复选框，然后单击[OK]按钮。开始创建一个新的工作空间。将会出现如下对话框：



1、在[CPU family]下拉菜单中选择目标 CPU。

2、在[Tool chain]下拉菜单中选择工具链

3、在[Project type]中选择要使用的工程类型，本例，选择“Application”。

（关于如何选择合适的工程类型，请参考 C/C++编译器中所附带的使用手册）

4、指定工作空间和工程的名称

- 在[Workspace Name]文本框中，键入新的工作空间名称。

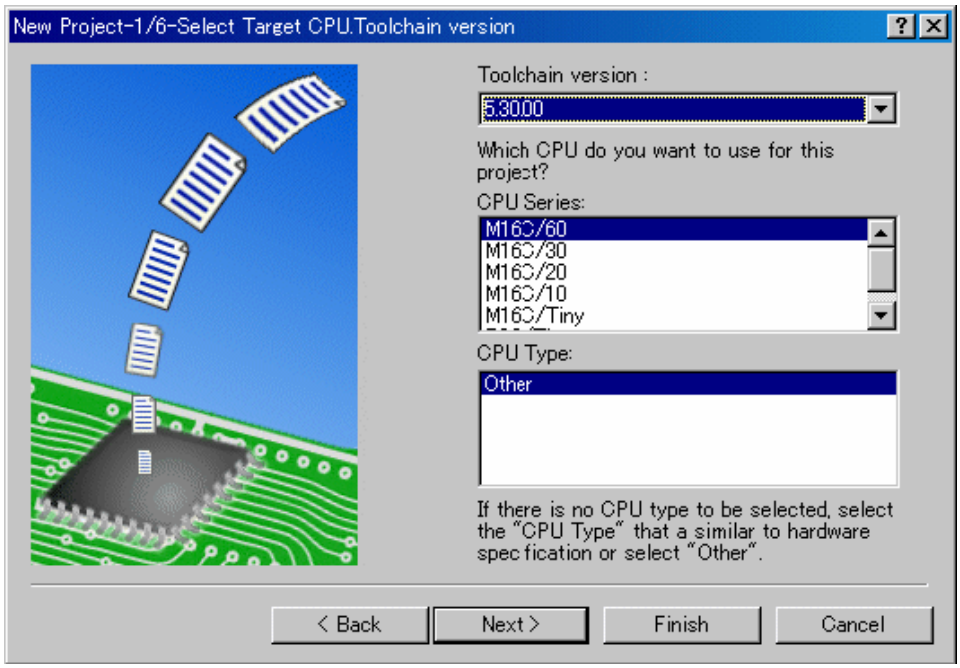
- 在[Project Name]文本框中，键入新的工程名称。当工程名称和工作空间的名称一样时，不需要输入。

- 在[Directory]文本框中，输入要建立工作空间的子目录。单击[Browse...]按钮来选择一个子目录。

设置完成后，单击[OK]按钮。

(2) 步骤 2：设置工具链

下面进入工程创建向导：



在这里，可以设置如下内容：

- 工具链
- 实时操作系统设置（若使用）
- 启动文件、堆（HEAP）空间、堆栈空间等的设置

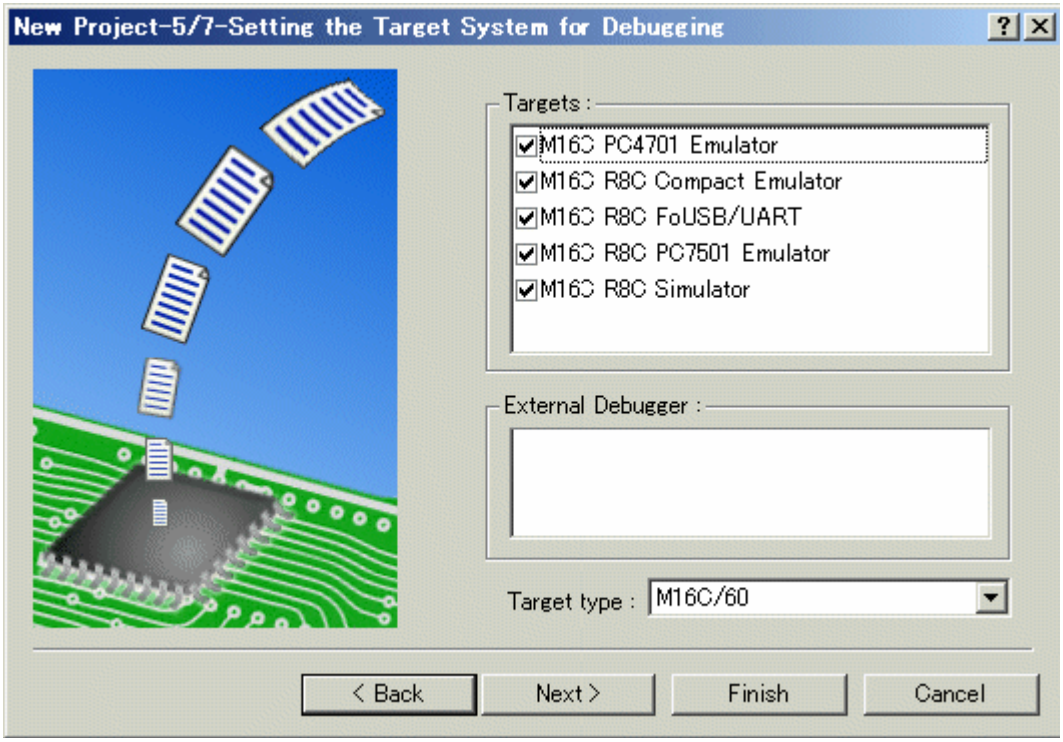
请设置需要的信息并且按[Next]按钮。

这些需要设置的内容随着 C/C++编译器的不同而不同，详情请参考 C/C++编译器中所附带的使用手册。

(3) 步骤 3： 选择目标平台

选择调试所用的目标系统（仿真器；模拟器）。

当工具链的设置完成以后，会显示如下的对话框：



1、在[Target type]下拉菜单中选择目标 CPU 的类型。

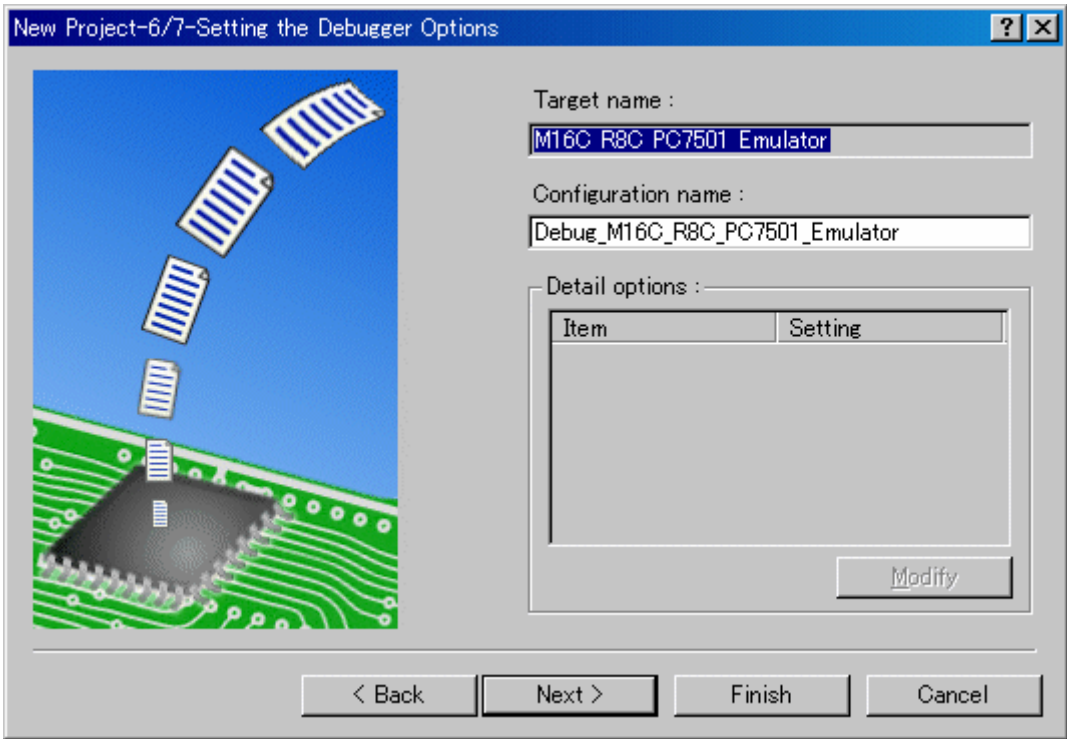
2、在[Targets]中选择目标，必须在这里选择合适的段（Session）文件来配合调试器使用。

选择符合目标平台的复选框（如有需要也可选择其他目标）

然后单击[Next]按钮。

(4) 步骤 4：设定配置文件名

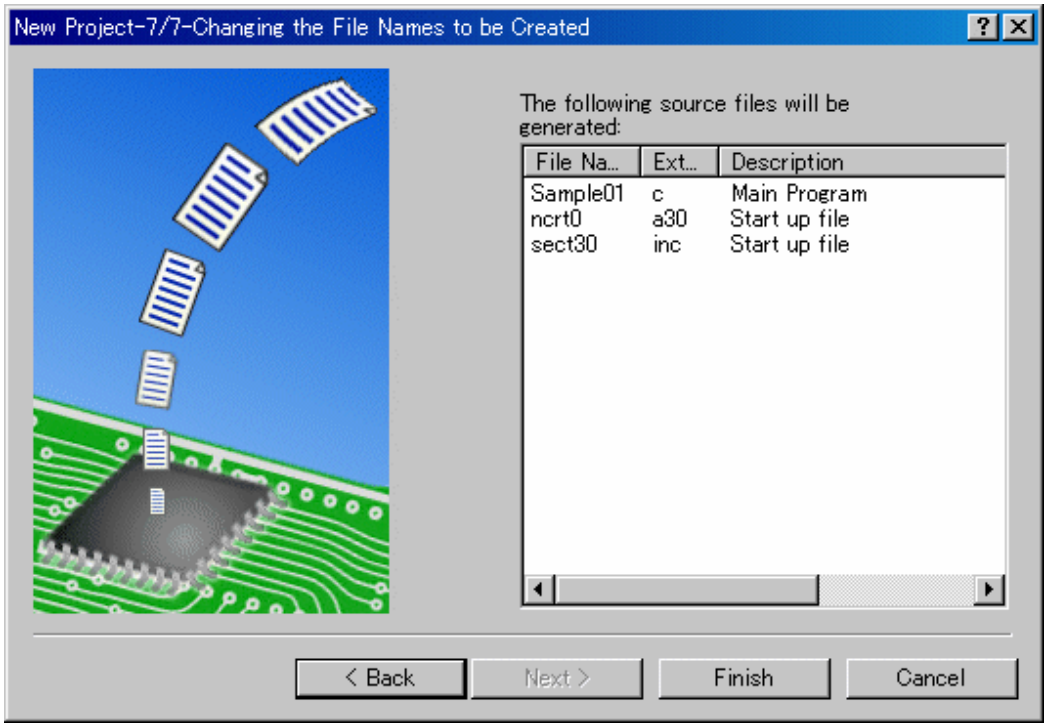
为所有选择的目标设置配置文件名，配置文件存储了除目标调试器（仿真器、调试器）以外的 High-performance Embedded Workshop 的状态。



默认的名称已经填好，如果不想更改，请直接点击[next]。

(5) 步骤 5：检查已创建的文件名

最后，确认创建的文件名。所有将要被 High-performance Embedded Workshop 创建的文件都会被显示出来。如果想要更改文件名，选中并且重新命名。



仿真器设置到此结束，退出工程生成器后请按照屏幕指示操作。

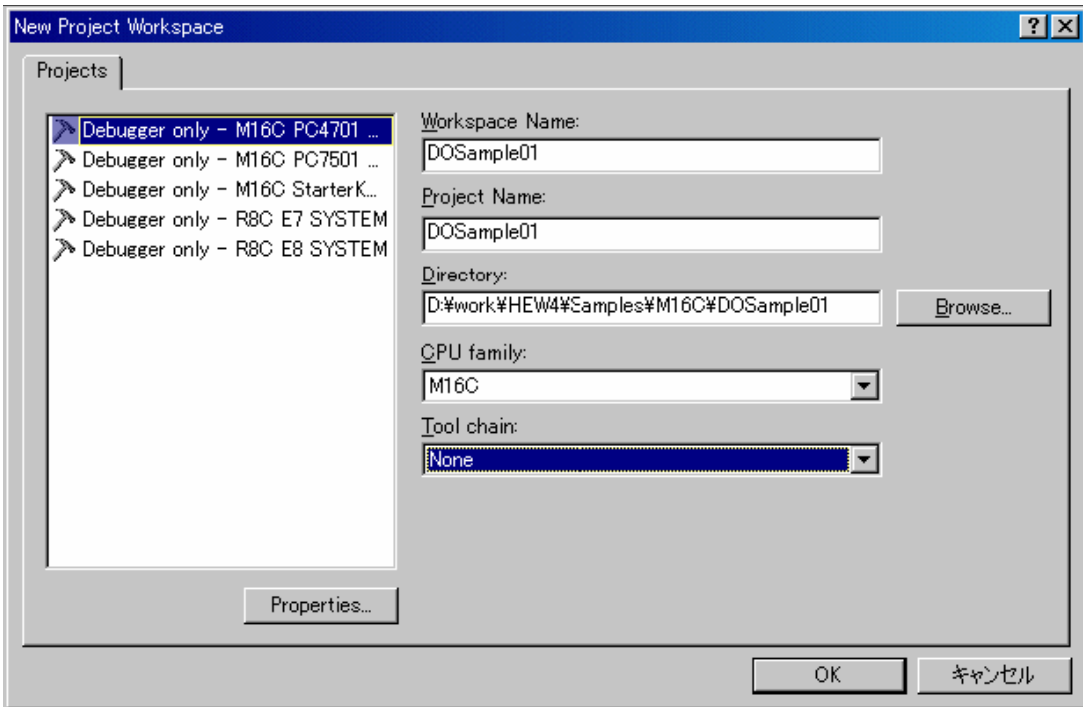
2.2.2 创建一个新的工作空间（不使用工具链）

当调试已经存在的加载模块时，使用此方法创建一个工作空间（即使未安装工具链时也适用）

(1) 步骤 1：创建一个工作空间

在打开 High-performance Embedded Workshop 后出现的[Welcome!]对话框中，选择[Create a new project workspace]复选框，然后单击[OK]按钮。

开始创建一个新的工作空间。将会出现如下对话框：



1、在[CPU family]下拉菜单中选择目标 CPU 的家族。

2、在[Tool chain]复选框中，选择“None”，这样，工具链就不会被使用。

（当未安装任何工具链的时候，Toolchain 复选框值固定为 None）

3、选择工程类型

当未使用工具链时，在[Project Type]列表框里会显示“Debugger only – Target Name”，请选择。（若出现两个或多个工程类型的时候，请选择其中之一。）

4、指定工作空间和工程的名称

- 在[Workspace Name]文本框中，键入新的工作空间名称。

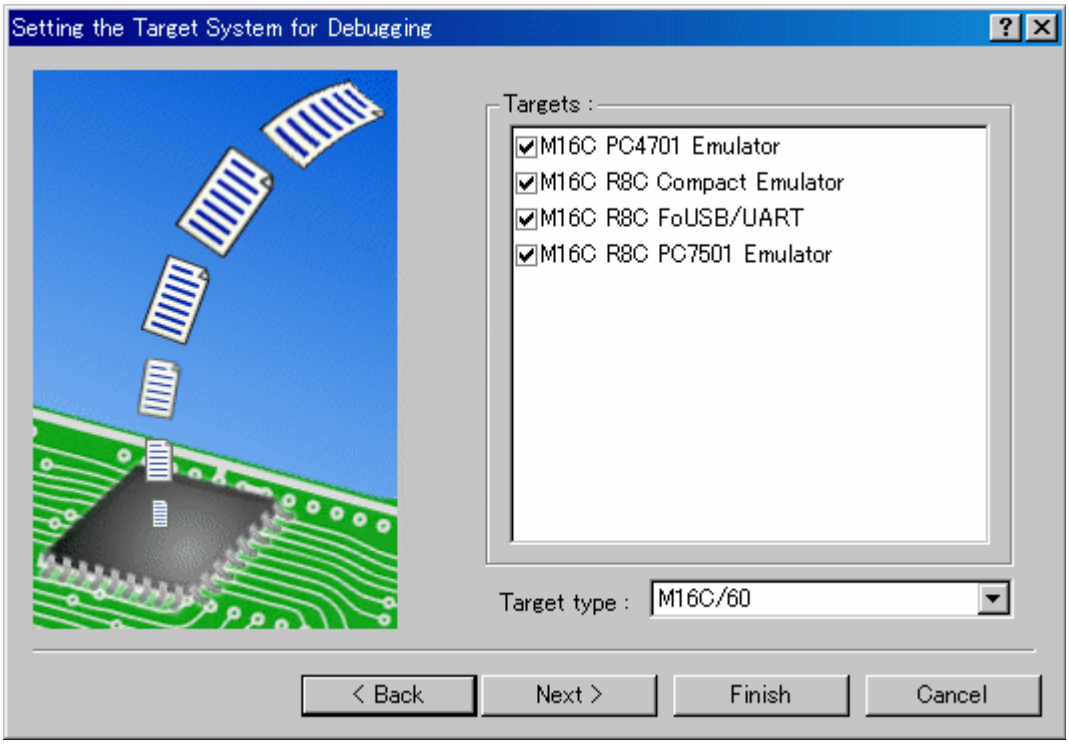
- 在[Project Name]文本框中，键入新的工程名称。当工程名称和工作空间的名称一样时，不需要输入。

- 在[Directory]文本框中，输入要建立工作空间的子目录。单击[Browse...]按钮来选择一个子目录。

设置完成后，单击[OK]按钮。

(2) 步骤 2： 选择目标平台

选择调试所用的目标系统（仿真器、调试器），会显示如下的对话框：



1、在[Target type]下拉菜单中选择目标 CPU 的类型。

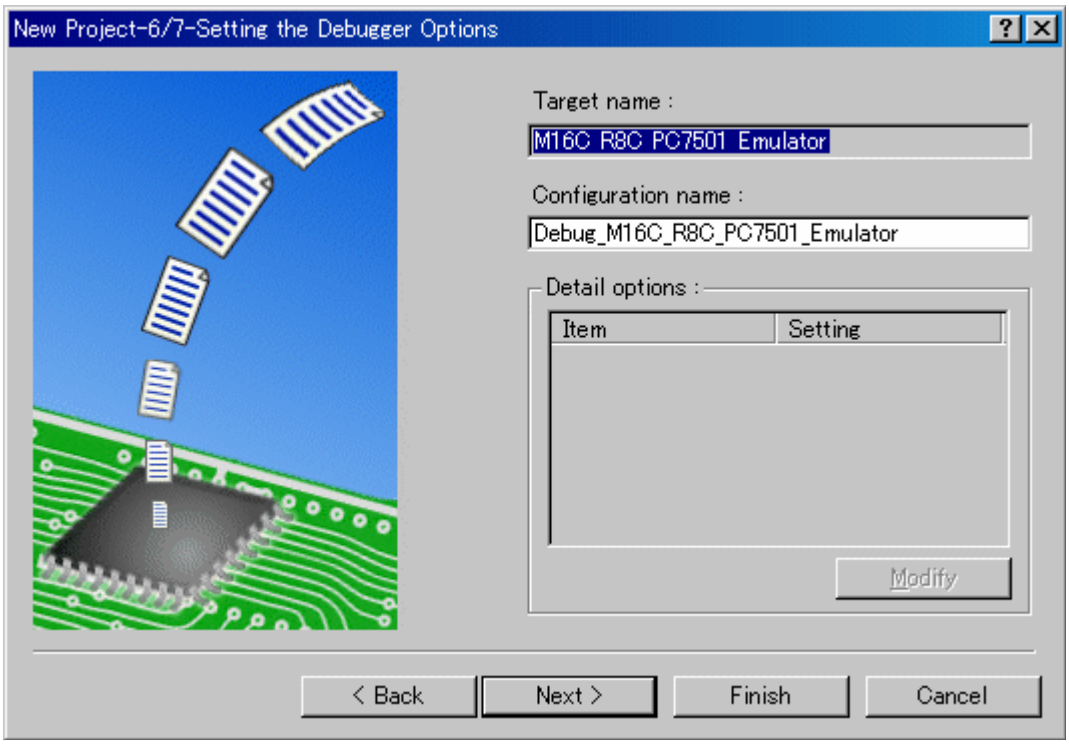
2、在[Targets]中选择目标，必须在这里选择合适的段（Session）文件来配合调试器使用。

选择符合目标平台的复选框（如有需要也可选则其他目标）。

然后单击[Next]按钮。

(3) 步骤 3：设定配置文件名

为所有选择的目标设置配置文件名，配置文件存储了除目标调试器（仿真器、调试器）以外的 High-performance Embedded Workshop 的状态。

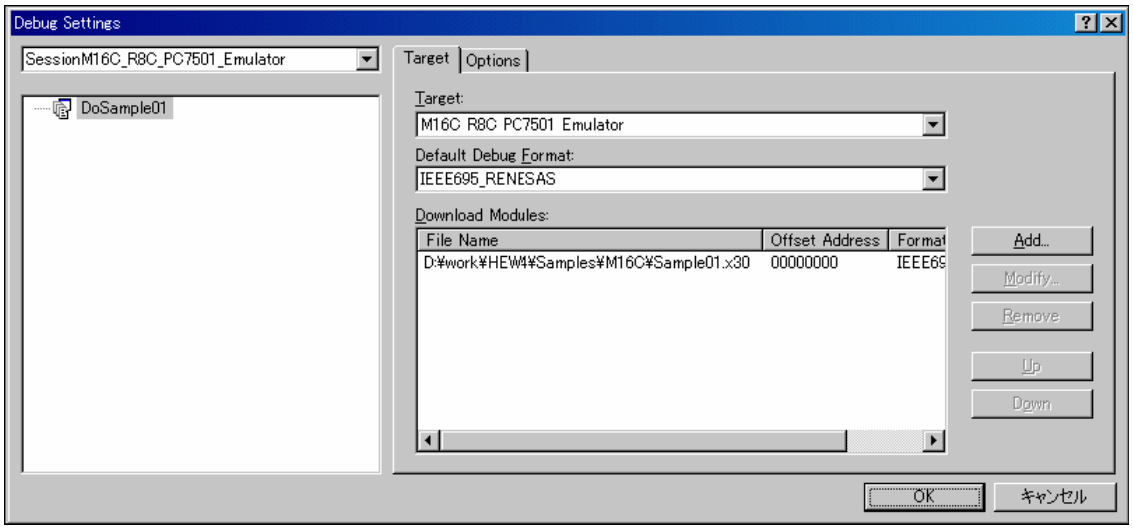


默认的名称已经填好，如果不想更改，请直接点击[Next]。仿真器设置到此结束，请按照屏幕提示退出工程生成器。同时，调试器连接窗口会弹出。

如果调试器已经连接完成，请在对话框中选择调试器设置并且连接。

(4) 步骤 4：注册将被下载的加载模块

最后，注册要下载的加载模块，从[Debug]菜单中选择[Debug Settings...]，打开[Debug Settings]对话框。



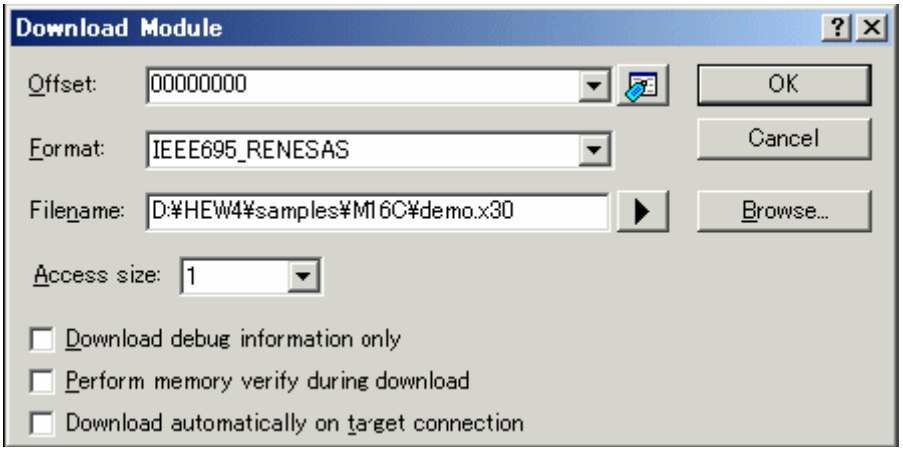
1、在[Target]下拉菜单中选择产品型号。

2、在[Default Debug Format]下拉菜单中选择要下载的模块的格式

本调试器不支持“格式下拉菜单”中没有的文件格式

格式	内容
IEEE695_RENS	IEEE-695 格式文件（当使用 NCxx 时）
ESAIEEE695_IAR	IEEE-695 格式文件（当使用 IAR 关联工具时）
IEEE695_TASKING	IEEE-695 格式文件（当使用 Tasking 关联工具时）
ELF/DWARF2_IAR	ELF/DWARF2 格式文件（当使用 IAR 关联工具时）
ELF/DWARF2_TASKING	ELF/DWARF2 格式文件（当使用 Tasking 关联工具时）

3、然后单击[Add]按钮，在[Download Modules]对话框中添加一个对应的加载模块。



- 在[Offset]一栏中输入加载模块下载后的偏移量。
- 在[Format]下拉菜单中选择加载模块的格式，请参考上一页中的下载模块的格式名称。
- 在[Filename]中输入加载模块的完整路径和文件名。
- 在[Access size]中指定当前的加载模块的 Access Size

然后，单击[OK]按钮。

注意

“Access size”和“Perform memory verify during download”均被忽略

“Access Size”只能设为“1”，并且验证功能目前无法使用。

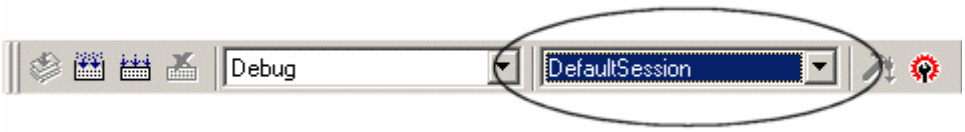
2.3 启动调试器

当仿真器正确连接以后，调试器便可以启动。

2.3.1 连接仿真器

选择一个注册过的段（Session）名称，便可方便地启动调试器连接窗口。

默认将会创建段（Session）文件。当工程创建的时候，段（Session）文件包含了目标的所有信息。在如下的工具栏中选择“段选择”下拉菜单，然后选择和想要连接的目标相对应的段（Session）名称。



当选择了段（Session）名称以后，相对应的调试器对话框将会显示，然后便可以连接仿真器。

2.3.2 断开仿真器

可以用如下方式断开仿真器：

1、当想要断开仿真器连接时，直接在段（Session）选择下拉菜单中选择[DefaultSession]

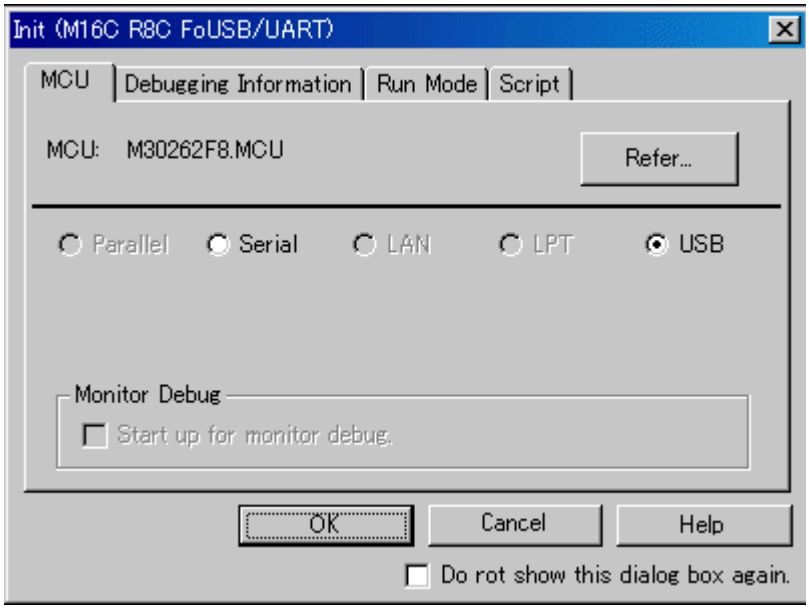
2、从[File]菜单中选择[Exit]，直接退出 High-performance Embedded Workshop

当退出一个段（Session）时，将会弹出一个对话框，询问是否保存，如果需要，单击[YES]进行保存，如果不希望保存，单击[NO]。

3. 设置调试器

3.1 初始化对话框

当调试器启动时，会弹出初始化对话框，来设置一些必要的参数。当下次启动时，这些参数都无需再次设置。



当所用的产品不同时，这个对话框中的选项卡会有多种形式。具体请参考下表：

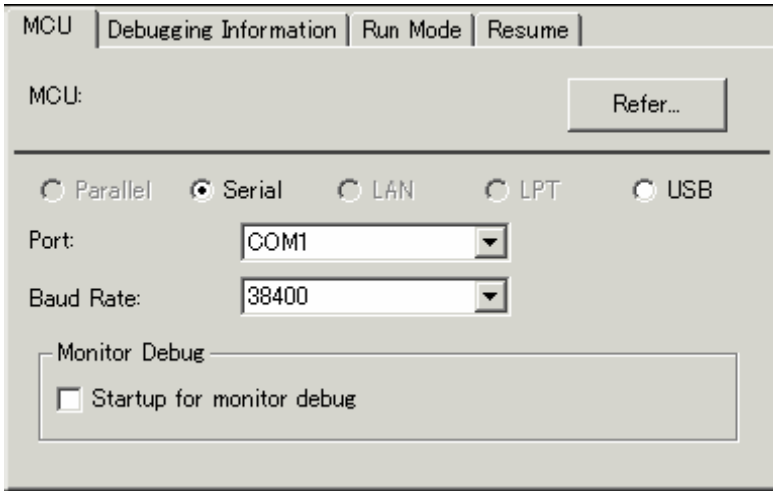
选项卡名称	产品名称	
	M32C 调试器	M16C/R8C 调试器
MCU	有	有
Debugging Information	有	有
Run Mode	有	有
Script	有	有

用户也可以通过如下方式打开本对话框：

- 调试器启动后，从菜单中选择：[Setup] -> [Emulator] -> [System...]。
- 按住 Ctrl 键，启动调试器。

3.1.1 MCU 选项卡

下次启动时，本页的所有设置可以被保留。



(1) 指定 MCU 配置文件

单击“Refer”按钮。

配置文件选择对话框打开后，请选择合适的 MCU 配置文件

配置文件包含了所要被调试的 MCU 的信息。

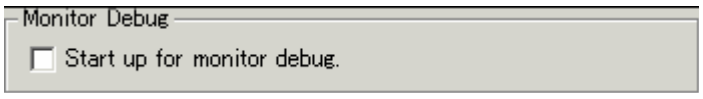
当选择完成以后，所选的 MCU 的信息将会显示在这一页的“MCU:”后边。

(2) 设置通信接口

当所用的通信接口不同时，这个对话框中的内容会有多种形式。可用的通信接口根据所选择的产品的不同而不同。下面的内容讲述了不同的通信接口的设置：

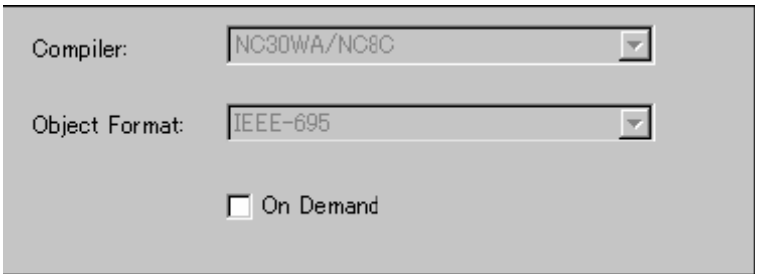
请同时参考：“3.2 设置通信接口”

(3) 设置“调试 Monitor Program”模式



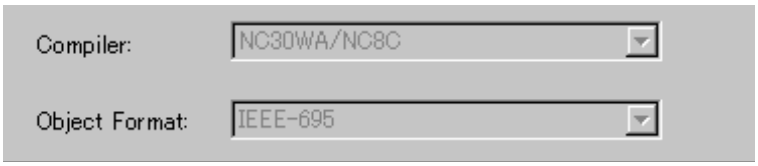
选择“Start up for monitor debug”可以用来调试监控程序。如果这个复选框被选中，本调试器在启动时将不对系统进行初始化。（如果要使用本功能，需要对应的监控程序文件）

3.1.2 调试信息选项卡



当下次下载时，本页所设置的内容才会生效。

(1) 显示所选的编译器和所采用的文件格式



显示所选的编译器和所采用的文件格式

请在菜单中选择[Debug] -> [Debug Settings...]，然后在打开的对话框中设置所选的编译器和所采用的文件格式。

(2) 指定调试信息的存储方式

有两种存储调试信息的方式：存储在存储器中/按需要存储。在两个里面请选择一种（默认为“存储在存储器中”）。若要选择“按需要存储”方式，选择[On Demand]复选框。

“存储在存储器中”方式下，所有调试信息都存储于 PC 中，这种方法适合于目标程序长度较短的场合。

“按需要存储”方式下，所有调试信息都存储于 PC 硬盘中的临时文件内，因为存储的文件可以再次使用，所以下次可以高速下载同样的模块。这种方法适合于目标程序长度较长的场合。

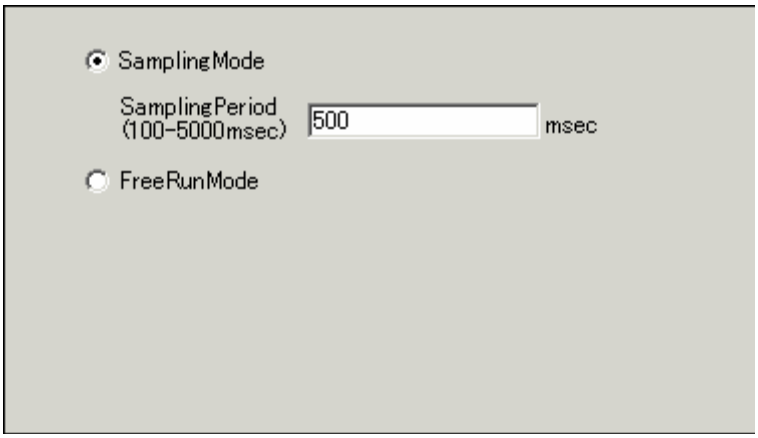
注意

如果目标程序长度较长，“存储在存储器中”方式可能效率较低，因为需要很长时间来进行下载操作。在这种情况下，请选择“按需要存储”方式

“按需要存储”方式下，会在存放加载模块的子目录中创建一个新子目录来存放临时文件。这个子目录的名称是加载模块的名称前加上“~INDEX_”。举例来说，如果加载模块的文件名是“sample.abs”，临时文件夹的名字就会是“~INDEX_sample”。退出调试器这个子目录也不会被删除。

3.1.3 运行模式选项卡

下次调试开始时，所设置的内容才会生效。



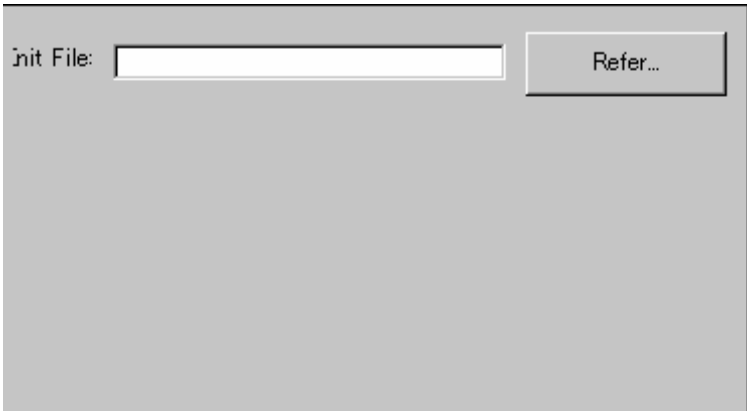
(1) 设置运行模式

设置当发出 Go 或 Come 命令时的系统运行模式。监控程序会定时地检测用户程序是否中断定情况。这意味着监控程序中断了 CPU 处理用户程序的流程，同时意味着，降低了调试用户程序的实时性。为解决这个问题，调试器提供了两种运行模式：

采样模式（Sampling Mode）：当下达了 Go 或者 Come 命令后，用户程序的运行状况会被定时监视，就是说，用户程序是否中断等等信息都可以被监视。定时监视的间隔时间可以在[Sampling period]文本框中输入。请设置一个尽量不影响用户程序实时性的时间。

自由运行模式（Free Run Mode）：当下达了 Go 或者 Come 命令后，用户程序的运行状况不会被定时监视。这就意味着，用户程序的实时性不会被干涉。但是，无法检测用户程序是否发生中断等等。因此，当用户程序已经中断时，调试器仍然会显示“Go”或者“Come”指令正在进行。在按下“Stop”按钮以后，调试器才会中断。同样的，如果在用户程序运行时下达了例如“察看 RAM” 或 “切换源码显示模式”的操作，用户程序运行时的实时性也会被破坏。

3.1.4 脚本选项卡



下次调试开始时，所设置的内容生效。

(1) 自动运行脚本命令

如果想在调试器开始时自动运行脚本命令，单击“Refer”按钮来选择一个脚本文件。

单击“Refer”按钮以后，会打开文件选择对话框，选中的脚本文件会显示在“Init File:”区域中。如果想取消自动脚本运行功能，删除“Init File:”文本框中的内容即可。

3.2 设置通信接口

3.2.1 设置 USB 接口

USB 通信使用 PC 机的 USB 接口，与 USB1.1 协议兼容。在 USB 通信之前，电脑必须先安装对应的驱动程序。如果需要选择 USB 通信连接，单击“MCU”选项卡上的“USB”单选按钮。

(1) 安装 USB 驱动文件

Windows 的即插即用功能可以检测 USB 设备的插拔事件。当设备检测到以后，驱动安装向导将会出现。以下说明讲述了安装驱动的流程：

1、用 USB 电缆连接 PC 机和仿真器

2、将通信接口设置为“USB”，然后打开仿真器电源

3、如下的对话框将会显示

继续安装向导，会弹出对话框要求指出安装文件（.inf 文件）的位置。请选择调试器所安装的目录下的 musbdrv.inf 文件。

注意

必须先安装调试器软件，再安装 USB 设备。

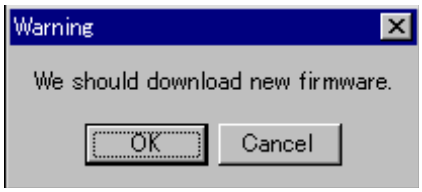
USB 通信方式只支持 Windows Me/98/2000/XP 操作系统，其他操作系统则不被支持。

当用户使用 Windows 2000/XP 时，安装 USB 驱动需要管理员权限。

在安装过程中，有可能探出对话框提示无法找到 musbdrv.sys。在这种情况下，请指名该文件与 musbdrv.inf 位于同一子目录。

(2) 下载监视程序

当所选择的 MCU 配置文件中指明的 MCU 监视程序与目标版上已有的监视软件不同时，调试器会提醒用户下载新的监视程序。如果 MCU 类型选择正确的话，单击[OK]按钮即可。

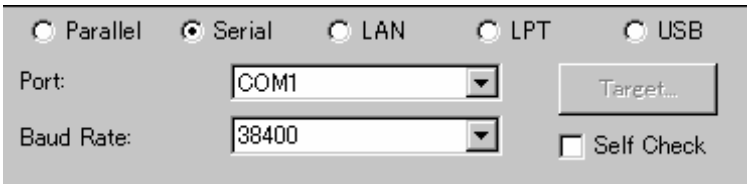


3.2.2 设置串行通信接口

串行通信使用 PC 的串行通信接口（RS-232C），这种方式适合于 PC4701 仿真器和串口调试程序。

(1) 设置串行通信接口

如要选择串行调试接口，单击 MCU 选项卡上的“Serial”按钮，设置画面如下：

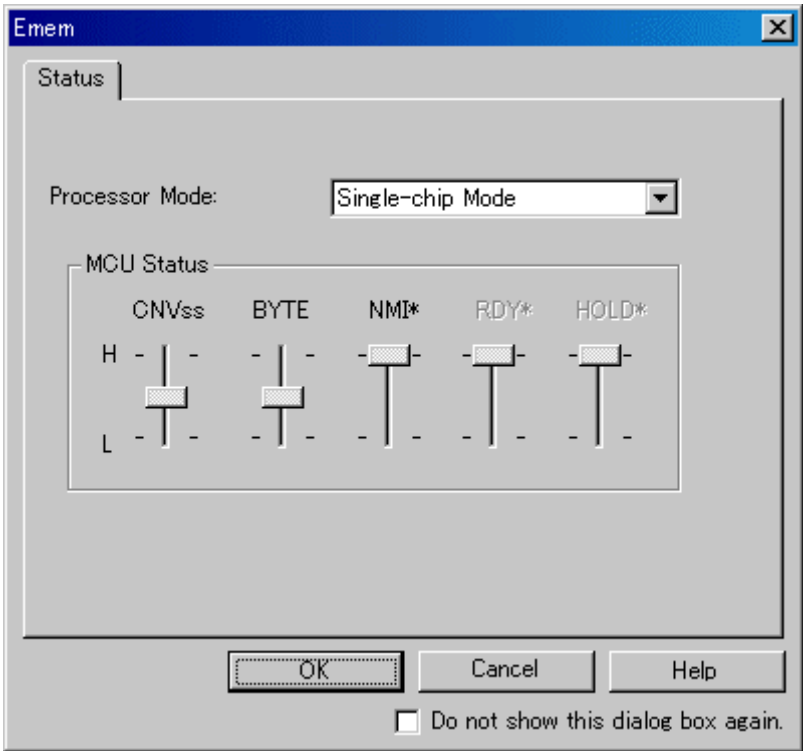


在“Port”下拉菜单中选择所用端口，在“Baud Rate”下拉菜单中选择波特率。

3.3 设置 M32C 调试器

3.3.1 Emem 对话框

在 Emem 对话框中，设置用户目标信息。Emem 对话框将在关闭初始化对话框后出现。



选项卡根据所用产品的不同而不同，具体请参见下表：

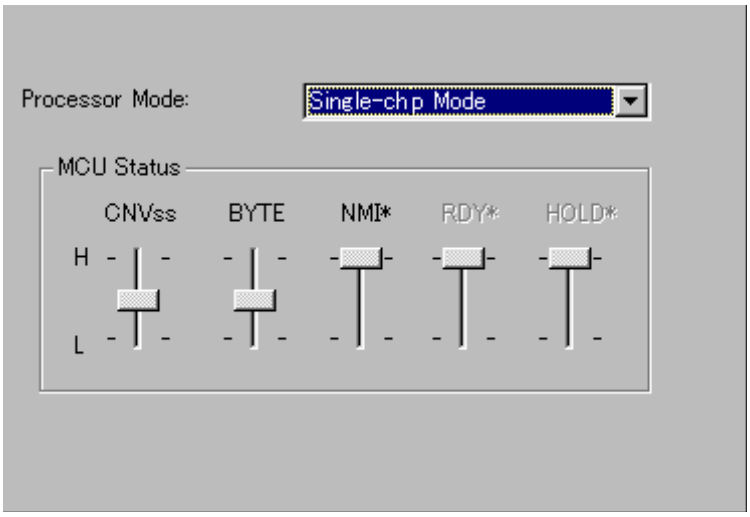
选项卡名称	内容
Status	指定处理器工作模式

如果下次调试器启动时不想打开 Emem 对话框，选中 Emem 对话框下方的“Next Hide”复选框。可以用如下方法再次打开 Emem 对话框：

菜单 - [Setup] -> [Emulator] -> [Target...]。

(1) 状态选项卡

下次调试开始时，所设置的内容生效。



(a) 选择处理器运行模式

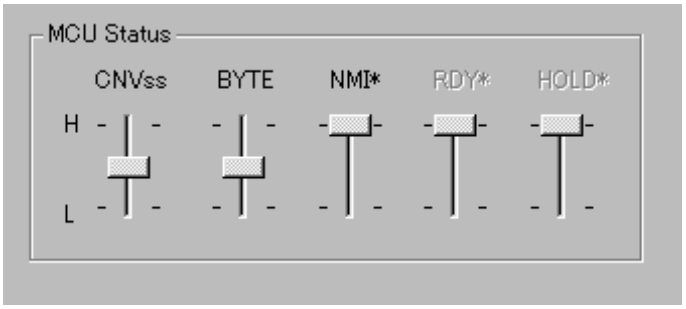
指定目标系统处理器运行模式

可以在下面的几项中选择一项

- 单芯片（Single-chip）模式
- 存储器扩展（Memory Expansion）模式
- 微处理器（Microprocessor）模式

(b) 检查 MCU 状态

选中这个选项卡显示这些 MCU 管脚的电平状态，可以让用户检查 MCU 管脚的当前状态是否与处理器工作模式相对应。



如果滑块在中间，说明此管脚值不定。

4. 示例程序

4.1 简介

本章用一个示例程序描述了本调试器的主要功能，示例程序安装于 High-performance Embedded Workshop 所安装的驱动器的\WorkSpace\Tutorial 目录中。对应各种 MCU 的工作空间也包括在内。请选择符合您系统的程序，然后从菜单[Open Workspace...] 中打开对应的工程空间（.hws）文件。

本示例程序是基于 C 语言的排序程序，将 10 个随机数升序或降序排列。

示例程序进行了以下操作：

- 生成将要被排序的随机数
- 使用排序函数将这些数按升序排列
- 使用改变函数使这些数降序排列

注意

- 重编译之后，所得的地址可能与本章所描述的不同。

4.2 使用

请遵循以下指示：

4.2.1 步骤 1：启动调试器

(1) 使用前的准备

如想要启动 High-performance Embedded Workshop，并且连接仿真器，请参考：“2 使用前的准备”

(2) 设置调试器

如果连接了仿真器，将会显示设置调试器对话框。请在其中设置好调试器。

如想设置调试器，请参考：“3 设置调试器”。

设置完成后，就可以实现 Debugger 的功能。

4.2.2 步骤 2：检查存储器操作

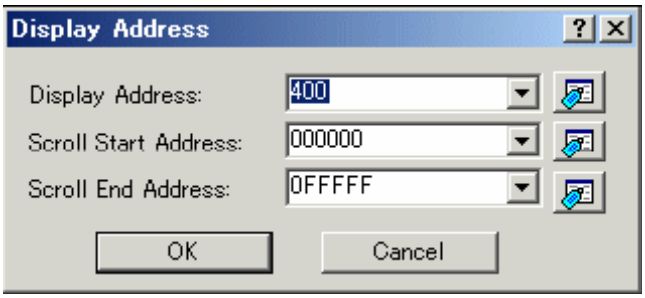
检查存储器操作结果是否正确，可以在[Memory]窗口中显示可修改的存储器的内容，并且检查确认存储器操作结果是否正确。

注意

对于某些单片机，存储器可以板载。在这种情况下，在这种情况下，上述的检查方法可能不适用。建议使用另外的程序来检查存储器中的内容。

(1) 检查存储器操作

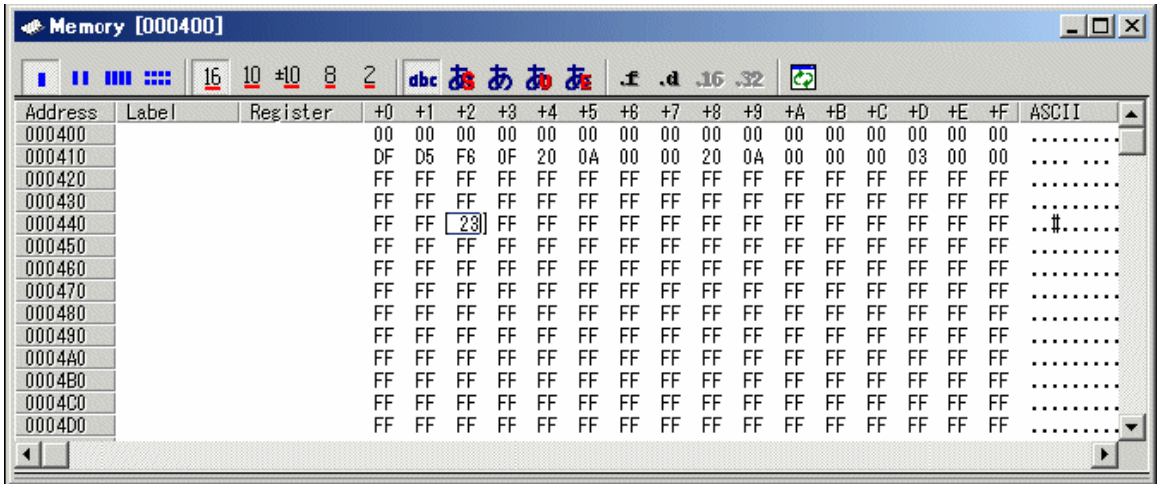
选择[View]菜单下的[CPU]->[Memory]，然后在[Display Address] 编辑框中键入存储器地址（本例中，键入 H’400）。 [Scroll Start Address]和[Scroll End Address]使用默认值即可。（默认情况下，滚动条范围为 0H 到 MCU 存储器的最大地址）



注意

根据产品型号的不同，可设定的存储器范围也不同。具体情况，请参考产品的硬件手册（Hardware Manual）。

单击[OK]按钮，将会显示[Memory]窗口，并且显示特定区域的存储器数值。



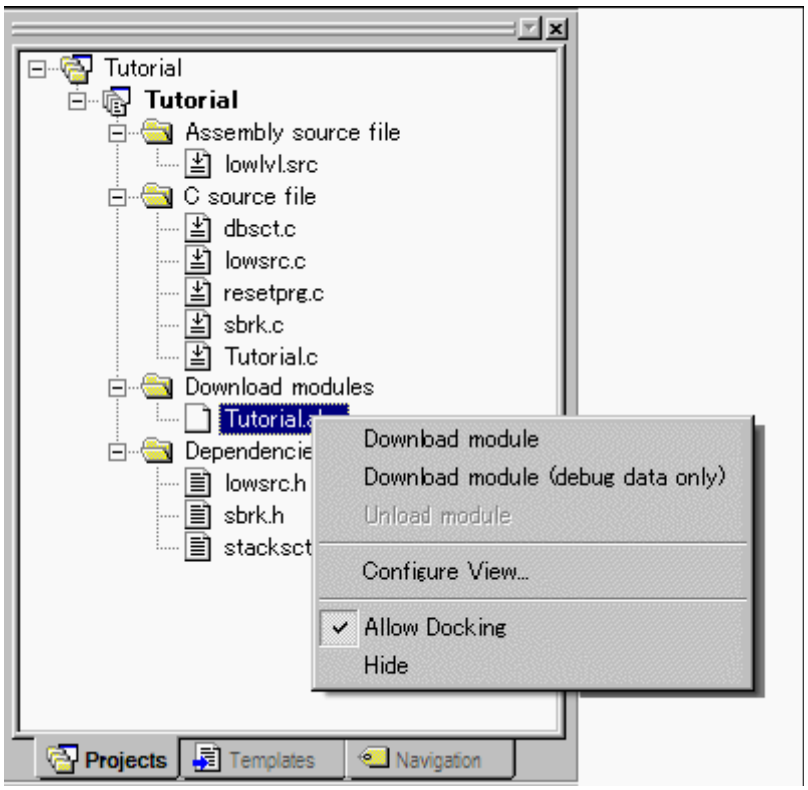
把鼠标移至[Memory]窗口中的数据上方并且双击，就可以直接更改存储器中的数据。

4.2.3 步骤 3：下载示例程序

(1) 下载示例程序

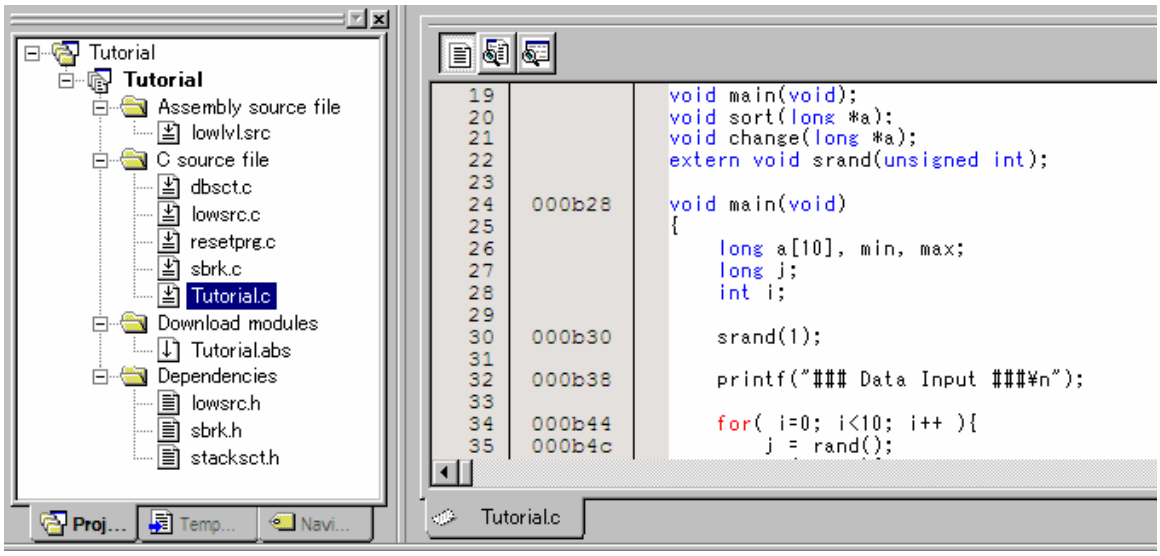
下载将要被调试的目标文件，要下载的文件和下载到的地址随使用的目标 MCU 的型号而不同。请参考屏幕中的设置并且调整为和您的 MCU 相符合的设置。

M16C/R8C 或 M32C 调试器请使用右键菜单中的“Download Module”命令，下载树形目录[Download modules]下的[Tutorial.x30]文件



(2) 显示源代码

本调试器允许用户在源代码基础上进行程序调试，双击树形目录[C source file]下的[tutorial.c]文件，将打开[Editor(Source)]窗口，“Tutorial.c”文件的内容将显示在屏幕上。



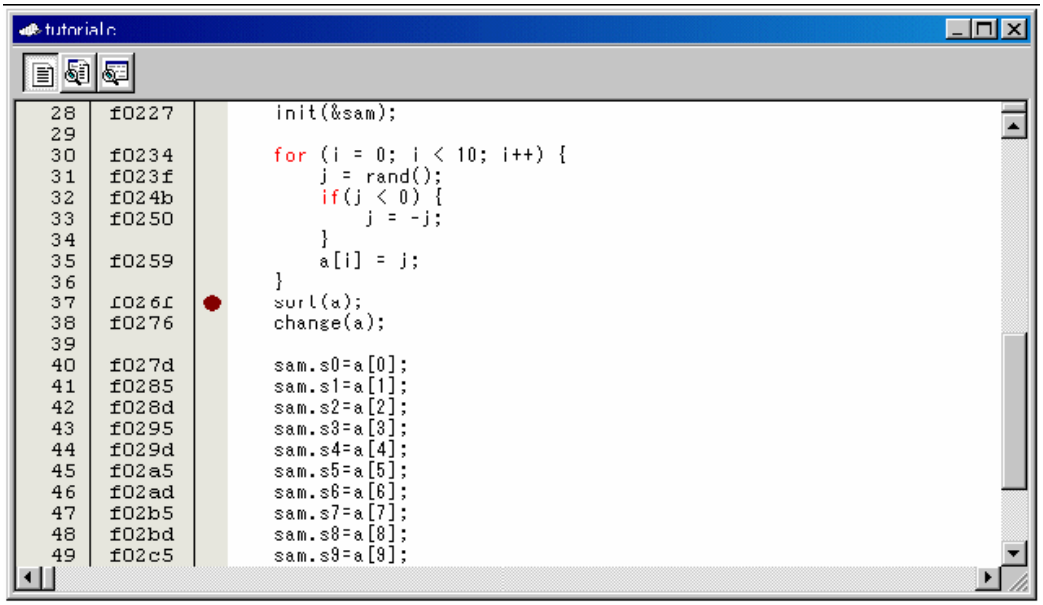
如果需要的话，选择[Setup]菜单中的[Format Views...]选项，并且选择合适的字体大小使阅读清晰容易。[Editor(Source)]窗口显示了用户程序的起始部分，用户也可以拖动滚动条查看整个用户程序或者其他段（Session）。

4.2.4 步骤 4：设置中断

使用软件中断是一种基本的调试方法，通过[Editor(Source)]窗口，提供了一种简单易行的在程序中的任何位置设置断点的办法。

(1) 设置一个软件中断

举例来说，要想在排序程序中设置一个软件断点：在包含排序程序的横行上，双击[S/W breakpoints]一列。



包含排序程序的横行会显示一个红色的标志，这意味着一个软件断点已经被设置了。

4.2.5 步骤 5：执行程序

用如下方法执行程序：

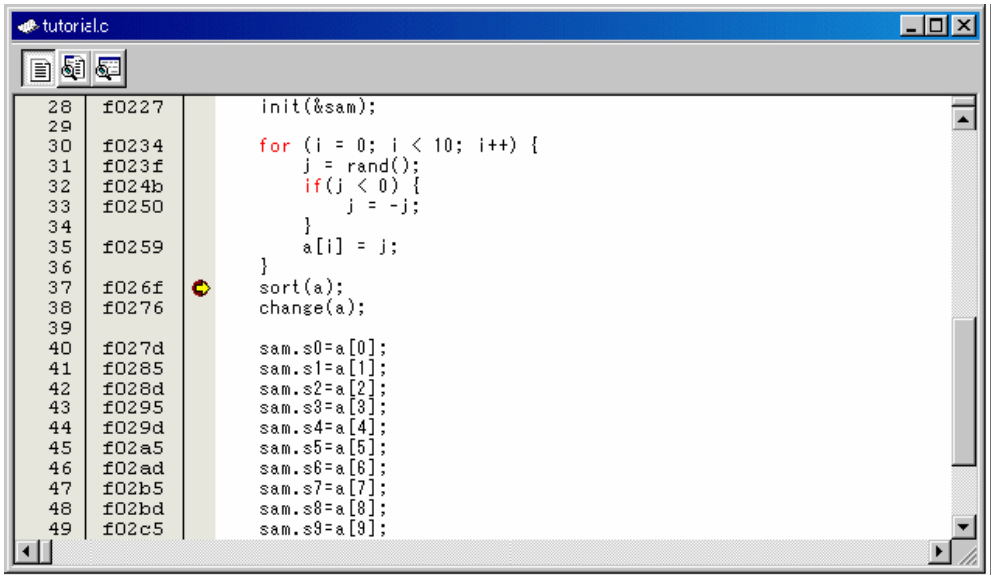
(1) 执行 CPU 复位指令

默认情况下，下载完成后 CPU 不复位。

如果想复位 CPU，选择[Debug]中的[Reset CPU]，或单击工具栏中的[Reset CPU]按钮 

(2) 执行程序

如果想执行程序，选择[Debug]中的[Go]命令，或者单击工具栏中的[Go]按钮 



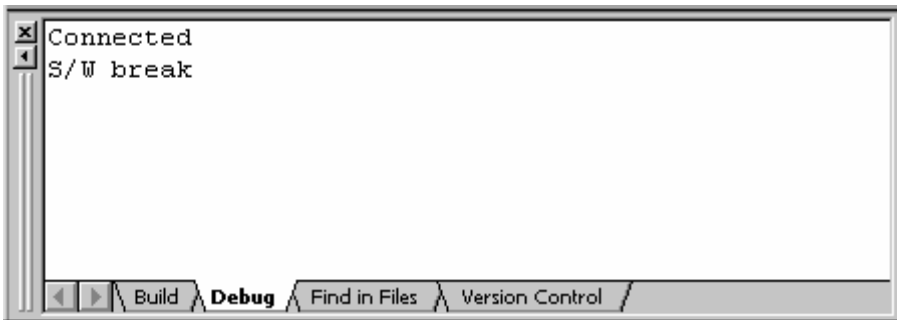
本程序将会执行至所设置的断点处，并且会出现一个箭头指向断点处，表明程序已经中断于此处。

注意

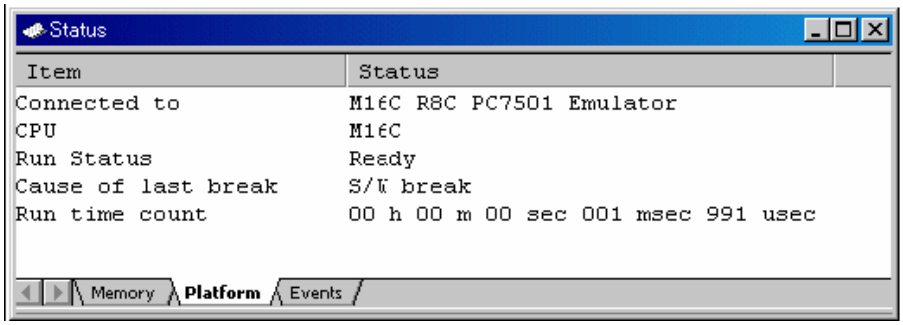
当中断后需要显示源代码时，可能会需要输入源文件路径，此时，请指明源文件的路径。

(3) 查看中断源

中断源可以在[Output]窗口中查看。



在[Status]窗口中，用户也可以查看上次中断的中断源。选择[View]菜单中的[CPU]选项中的[Status]菜单，[Status]窗口打开后，选择[Platform]选项卡，并且检查上次中断的中断源。



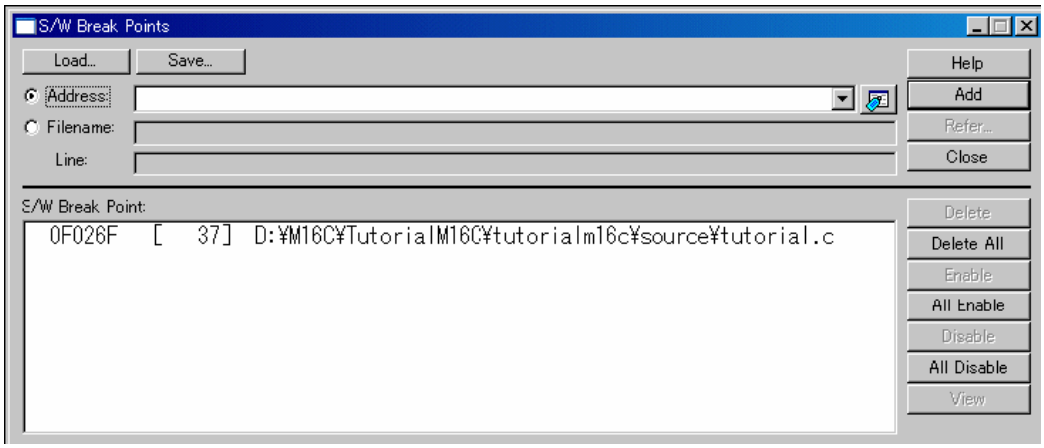
关于中断源，请参考：“9 显示程序停止的原因”

4.2.6 步骤 6：查看已设断点

用户可以在[S/W Break Points]窗口中查看所有已经设置的断点。

(1) 查看已设断点

选择菜单 Menu - [View] - [Break]，然后单击[S/W Break Points]，将会显示[S/W Break Points]窗口



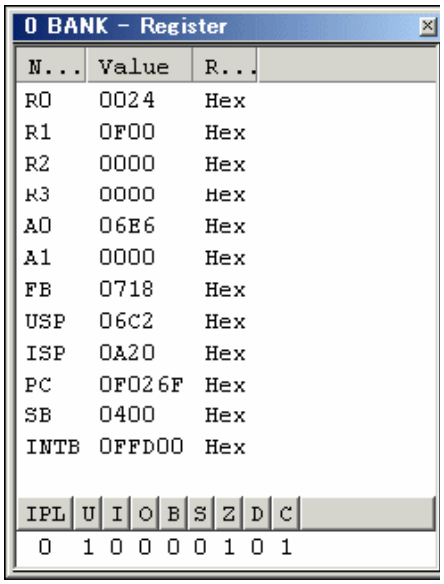
这个窗口允许用户查看或修改中断、定义新的中断，或者删除/禁用/启用中断。

4.2.7 步骤 7：查看寄存器

在[Register]窗口中，用户可以查看所有的寄存器/标志位。

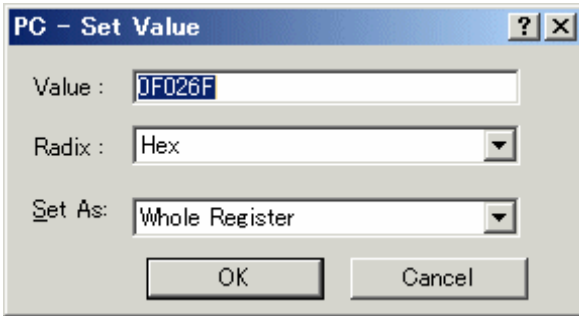
(1) 查看寄存器

选择[View] - [CPU]菜单中的[Registers]子菜单，将会显示[Register]窗口。M16C/R8C 调试器中的寄存器窗口如下所示。



(2) 设置寄存器的值

用户可以在这个窗口中改变寄存器/标志位的值。双击想要改变的一行，将会打开对话框。键入想要改变的值。

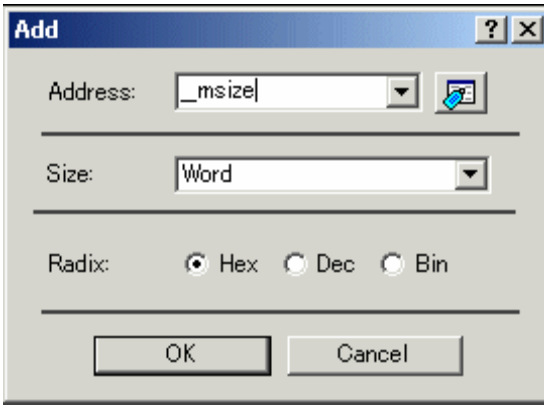


4.2.8 步骤 8：查看存储器

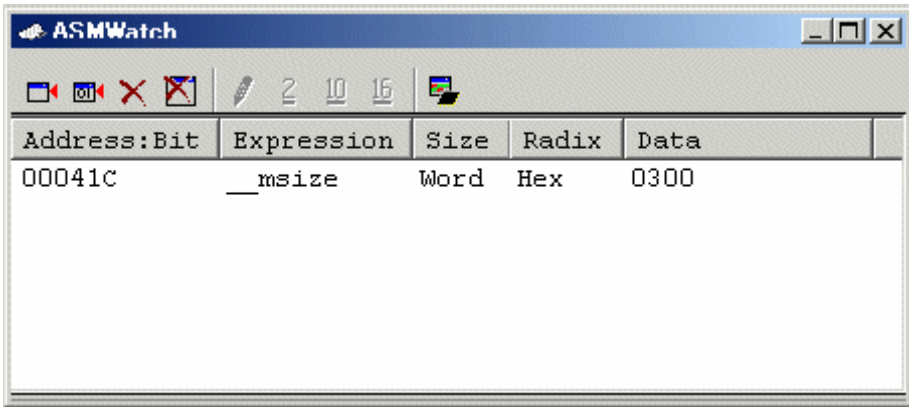
当指明标号以后，用户就可以在[ASM Watch]窗口中查看所指明的标号相关联的存储器区域。

(1) 查看存储器

举例来说，想以双字（Word）的模式查看与标号“__msize”相关的存储器内容：选择[View]-[Symbol]中的[ASM Watch]菜单，打开[ASM Watch]窗口，右键单击[ASM Watch]窗口的空白区域，从弹出菜单中选择[Add...]，在[Address]中键入“__msize”，然后在[Size]下拉菜单中选择[Word]。



单击[OK]按钮，所指定的存储器区域便会显示在[ASM Watch]窗口中。

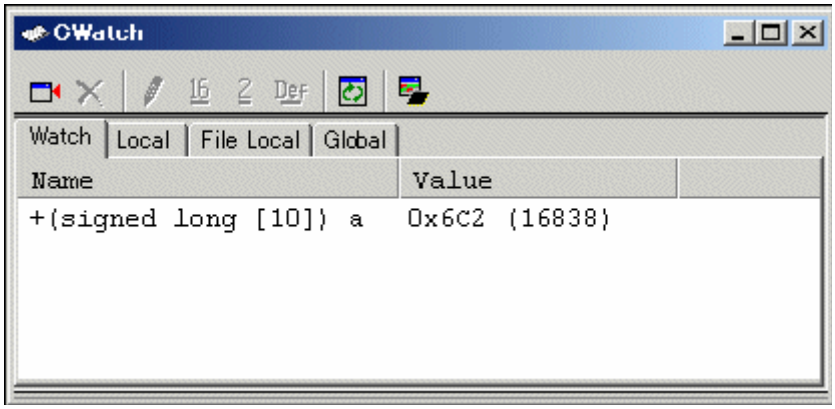


4.2.9 步骤 9：查看变量

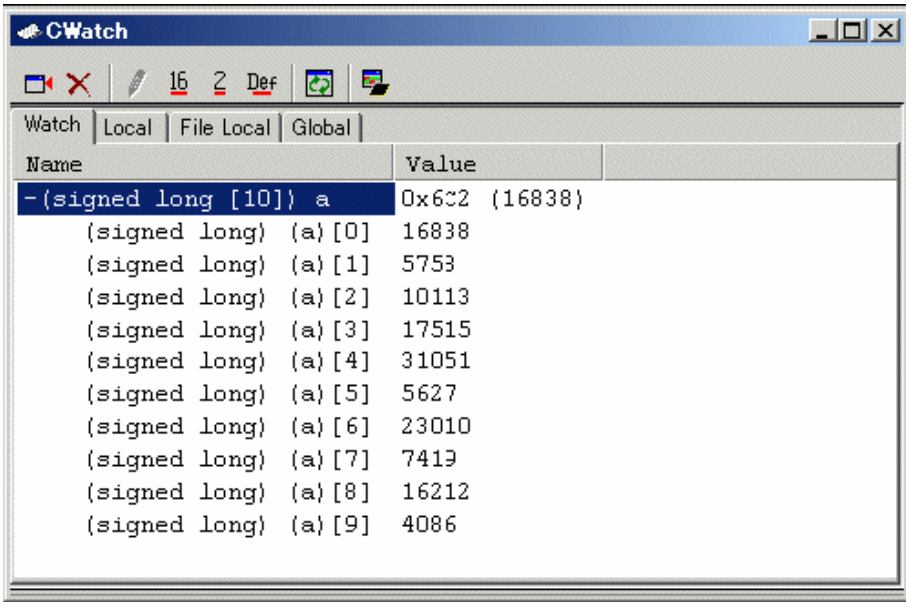
如果用户单步调试程序，可以观察到程序中变量的更改。

(1) 查看变量

举例来说，在程序的起始位置定义一个长整形的数组，并且在 Watch 窗口中查看它，步骤如下：将鼠标指针指向[Editor(Source)]窗口中要观察的变量，右键单击并且选择[Add C Watch...]选项。[Watch]类窗口中的[C watch]将会打开并且显示变量。



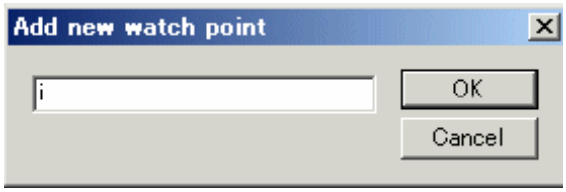
用户可以单击变量左侧的“+”符号观察数组中的所有成员变量。



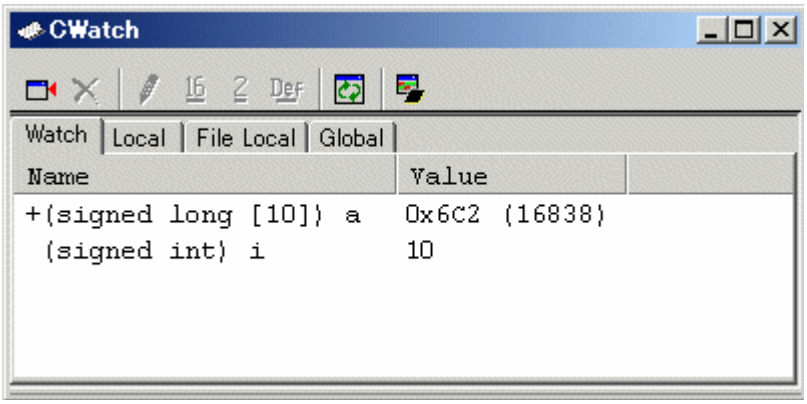
(2) 注册变量

用户也可以通过在[C Watch]窗口中添加变量名来加入观察变量。右键单击[C Watch]窗口的空白处，

选择[Add...]。以下窗口便会显示，键入变量“i”。



单击[OK]按钮，[C Watch]窗口现在也可以显示整型变量“i”了。



4.2.10　　步骤 10：单步调试

本调试器提供了丰富的单步调试命令来协助用户调试程序。

1、Step In 命令执行每条语句，包括函数（子程序）中的语句。

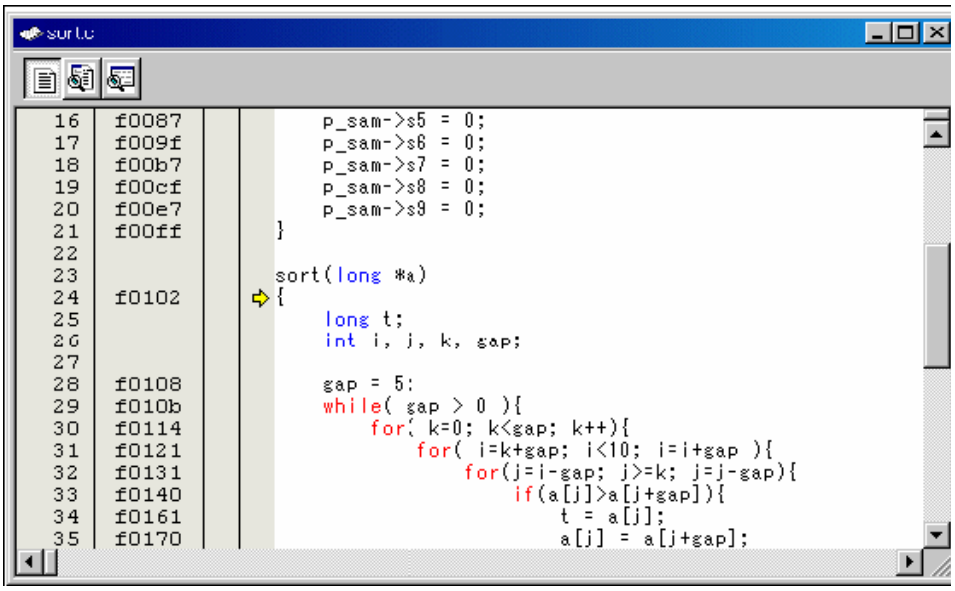
2、Step Out 命令执行完一个函数（子程序），并且停止在调用这个函数（子程序）的语句之后的那一条语句。

3、Step Over 命令一次执行完函数（子程序）的调用。

4、Step 指令按照所设置的步长执行对应长度的语句。

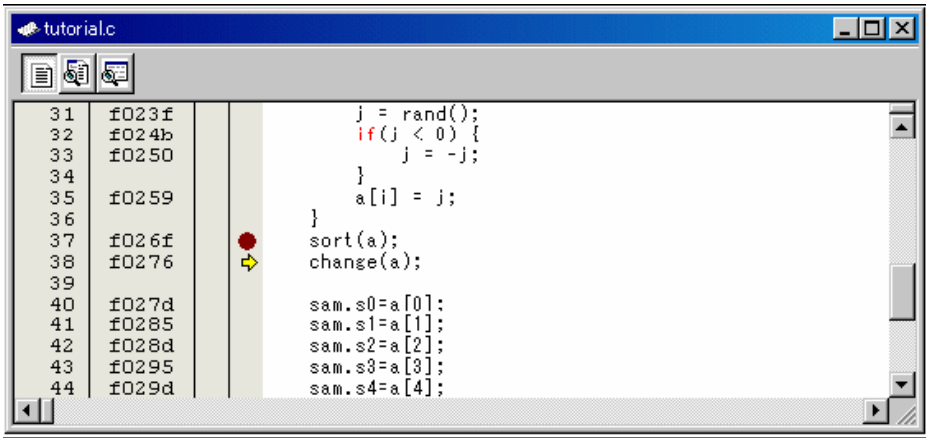
(1)　执行[Step In]命令

[Step In]命令执行进入被调用的函数（子程序），并且停止在被调用的函数（子程序）的第一句。如想单步进入排序程序，选择[Debug]菜单中的[Step In]命令，或者单击工具栏中的[Step In]按钮，[Editor(Source)] 窗口中，PC 指针会停在函数（子程序）的第一句上。



(2) 执行[Step Out]命令

[Step Out]命令执行完一个函数（子程序），并且停止在调用这个函数（子程序）的语句之后的那一条语句。如想单步退出排序程序，选择[Debug]菜单中的[Step Out]命令，或者单击工具栏中的[Step Out]按钮。 PC 指针会移出排序程序，并且停在“change”程序的前方。

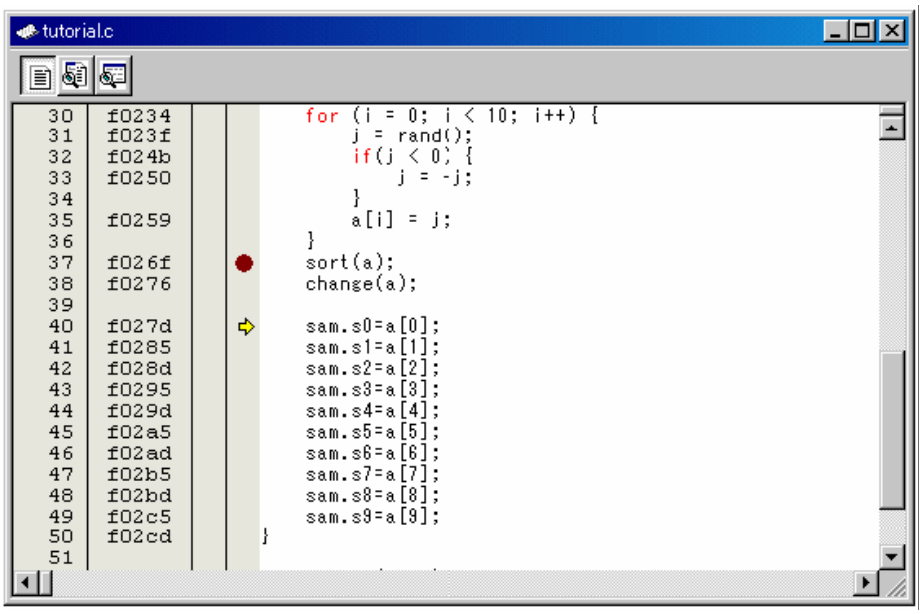


注意

本指令需要较长执行时间，当明确调用的源时，请使用[Go To Cursor]指令。

(3) 执行[Step Over]命令

Step Over 命令一次执行完函数（子程序）的调用，并且停在主程序的下一语句。如想单步执行完 Change 子程序，从[Debug]菜单中选择[Step Over]，或单击工具栏中的[Step Over]按钮。 PC 指针移动到 Change 子程序之后的一句。



4.2.11 步骤 11：强制终止程序执行

本调试器可以强制终止程序运行。

(1) 强制终止程序运行

取消所有中断，如要运行剩下的所有程序，选择[Debug]-[Go]或者单击工具栏上的[Go]按钮。 

程序进入一个无休止的循环。如要强制终止程序运行，选择[Debug]- [Halt Program]或者单击工具栏上的[Halt]按钮。 

4.2.12 步骤 12：显示局部变量

用户可以使用[C Watch]窗口显示局部变量

(1) 显示局部变量

举例来说，我们将要检验示例程序中的局部变量，一共定义了四个变量：a， j， i 和 sam。

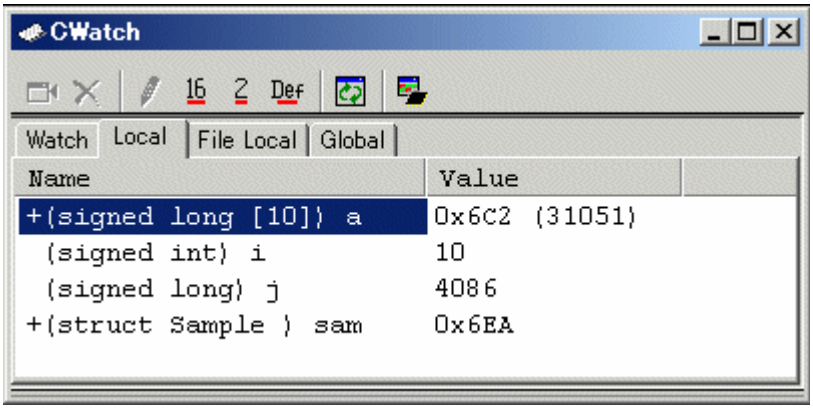
选择[View] - [Symbol] - [CWatch]，将会显示[C Watch]窗口。默认情况下，[C watch]有如下四个选项卡：

- [Watch]选项卡
  只有用户指定的变量将会被显示。
- [Local]选项卡
  所有 PC 指针范围内的局部变量都会被显示在此。

  如果 PC 指针范围随程序的运行而改变，[Local]选项卡页面的内容也会随之改变。
- [File Local]选项卡
  所有 PC 指针范围内的文件局部变量都会被显示在此。

  如果 PC 指针范围随程序的运行而改变，[File Local]选项卡页面的内容也会随之改变
- [Global]选项卡
  当前下载了的程序中的所有全局变量都会被显示于此。

  如果想查看局部变量，请选择[Local]选项卡。

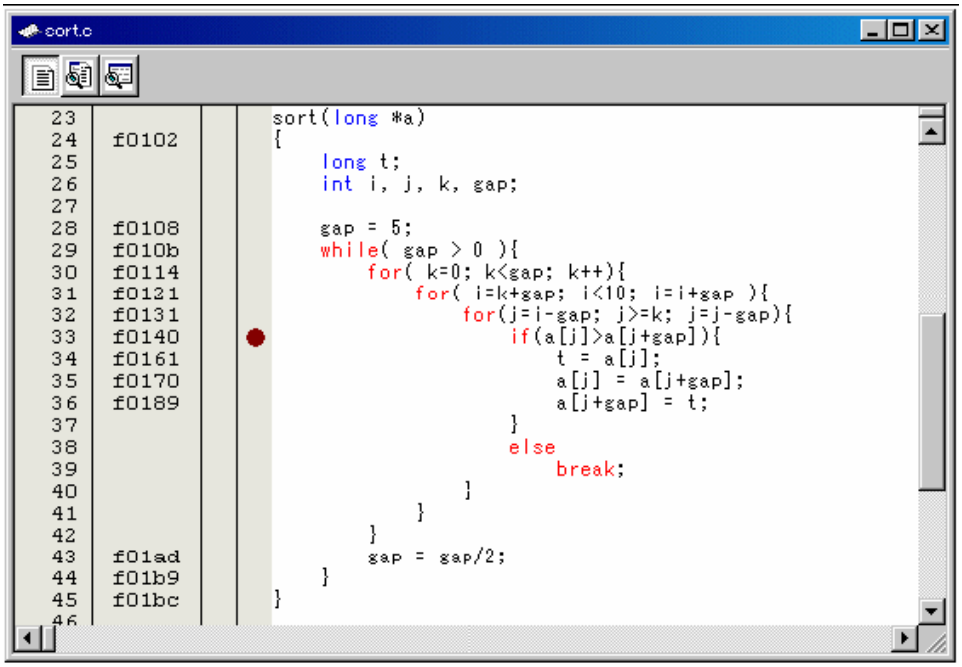


单击[Locals]窗口中数组变量左侧的“+”符号来查看其中的元素。当用户对比排序前的随机数和排序以后的结果，会发现随机数已经按降序排列。

4.2.13 步骤 13：堆栈追踪功能

调试器利用堆栈中的信息来显示调用当前 PC 指向的函数的主函数的名称。

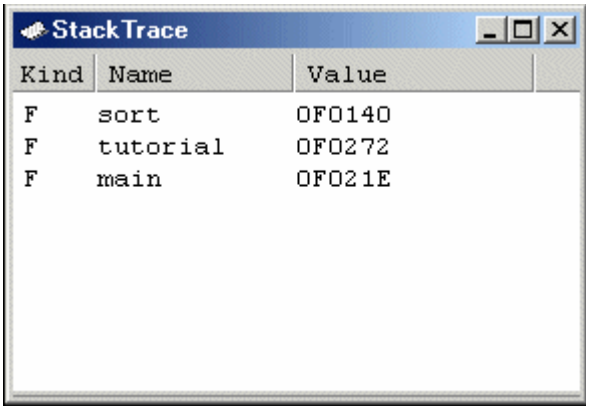
(1) 参考函数调用状态



在[S/W Breakpoints]一列上双击，设置一个软件断点。

如想从复位向量地址开始执行用户程序，在[Debug]菜单中选择[Reset Go]，或者单击工具栏上的[Reset Go]按  钮。

当程序在断点处停止时，从[View] - [Code]菜单中选择[Stack Trace]命令来打开[Stack Trace]窗口。



从上图可以看出，现在程序指针 PC 停止在排序程序的所选行中，并且排序程序是由 tutorial() 函数所调用的。

4.2.14 接下来是？

本指南讲述了调试器的使用方法。复杂的调试可以使用仿真器提供的仿真功能。当有问题发生时，可以迅速有效的定位和隔离软、硬件问题。

参考

（空白页）

5. 窗口/对话框

本调试器的所有窗口如下所列：

当单击窗口名称，将会显示参考信息。

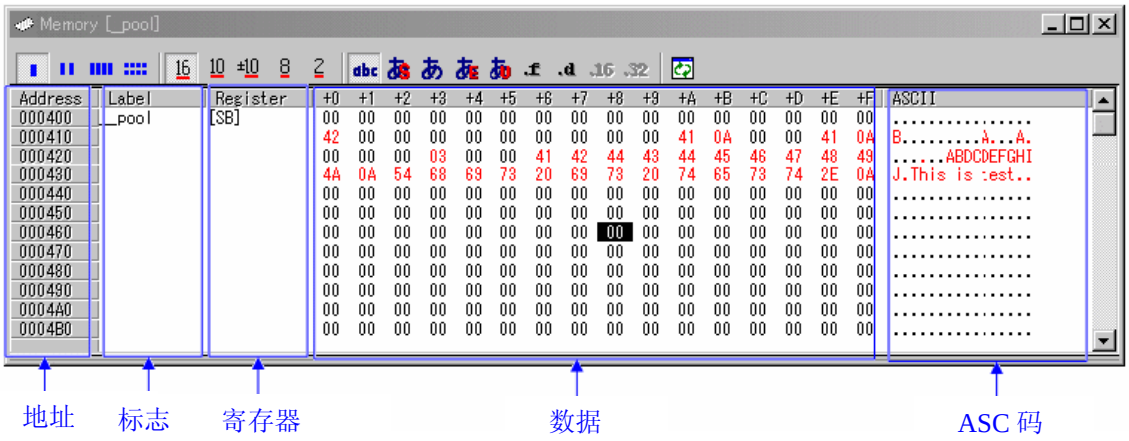
窗口名称	View 菜单
RAM Monitor 窗口	[View]->[CPU]->[RamMonitor]
ASM Watch 窗口	[View]->[Symbol]->[ASMWatch]
C Watch 窗口	[View]->[Symbol]->[CWatch]
Script 窗口	[View]->[Script]
S/W Break Point 设置窗口	[View]->[Break]->[S/W Break Points]
GUI I/O 窗口	[View]->[Graphic]->[GUI I/O]
MR 窗口	[View]->[RTOS]->[MR]

如要获取如下窗口的参考信息，请参阅 High-performance Embedded Workshop 所附带的帮助文件

- Differences 窗口
- Map 窗口
- Command Line 窗口
- Workspace 窗口
- Output 窗口
- Disassembly 窗口
- Memory 窗口
- IO 窗口
- Status 窗口
- Register 窗口
- Image 窗口
- Waveform 窗口
- Stack Trace 窗口

5.1 RAM Monitor 窗口

RAM monitor 窗口是显示目标程序运行时存储器变化的窗口。当目标程序运行时，显示刷新时间可以根据所设定的值而变化。



本系统提供了 1K 字节的 RAM 显示范围，并且可以定义为任何连续的地址。

RAM Monitor 可以被设定为任意想要的地址范围，参照“5.1.2 设定 RAM Monitor 范围”来更改 RAM Monitor 范围。默认范围映射于内部 RAM 的前 1K 字节范围。

每个窗口的更新周期可以自由设定。程序运行中实际的更新时间显示在 Address 显示区域的标题中。

注意

显示的更新时间可能比所指定的更新时间要长，这取决于当前的操作环境：

- 上位机的速度/负载情况
- 通信接口
- 窗口大小（存储器显示范围）或者显示窗口的数量。

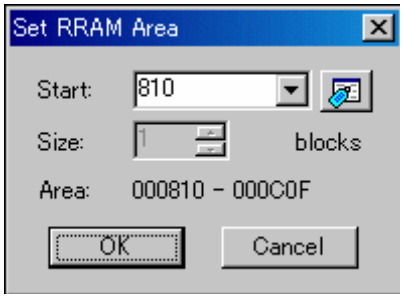
5.1.1 扩展菜单

在这个窗口内右键单击时会弹出菜单：

菜单		功能
RAM Monitor Area…		设置 RAM monitor 起始地址
Sampling Period…		设置 RAM monitor 采样周期
Clear		清除访问属性
Up		将显示窗口位置向存储器地址较低的方向滚动
Down		将显示窗口位置向存储器地址较高的方向滚动
Address…		从所指定的地址开始显示
Scroll Area…		指定滚动范围
Data Length	1byte	以一字节为单位显示
	2bytes	以两字节为单位显示
	4bytes	以四字节为单位显示
	8bytes	以八字节为单位显示
Radix	Hex	以十六进制显示
	Dec	以十进制显示
	Signed Dec	以有符号十进制显示
	Oct	以八进制显示
	Bin	以二进制显示
Code	ASCII	以 ASCII 码显示
	SJIS	以 SJIS 码显示
	JIS	以 JIS 码显示
	UNICODE	以 UNICODE 码显示
	EUC	以 EUC 码显示
	Float	以浮点方式显示
	Double	以双精度浮点方式显示
Layout	Label	选择是否显示 Label 区域
	Register	选择是否显示 Register 区域
	Code	选择是否显示 Code 区域
Column…		设置每一行能够显示的列的数目
Split		分割窗口
Toolbar display		显示工具栏
Customize toolbar...		打开自定义工具栏对话框
Allow Docking		允许窗口停靠
Hide		隐藏窗口

5.1.2 设置“RAM monitor”范围

在 RAM monitor 窗口中选择 [RAM Monitor Area...]右键弹出菜单，如下所示，“Set RAM Area”对话框将会显示。当前选择的 RAM 区域将会显示在 Area 之后（Size 一栏无法输入）



使用这个对话框来改变 RAM monitor 的显示范围

通过指定起始地址来指定 RAM monitor 的区域。“Size”一栏不能被改变（固定于“1Kbyte”）

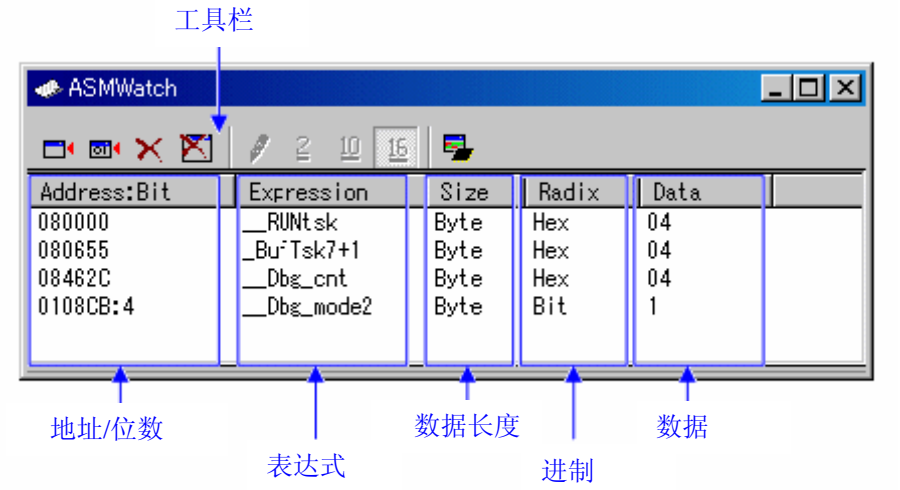
起始地址可以以 0x10 字节为单位改变，如果指定了不符合单位的地址，会在设置前自动更改为最近的 0x10 字节为单位的地址。

(1) 改变 RAM Monitor 范围

RAM monitor 的起始地址可以更改，在 RAM monitor 中的“Set RAM Area”对话框中设置起始地址（Size 一栏无法输入）

5.2 ASM Watch 窗口

ASM watch 窗口可以将特定地址设定为 watchpoint 并且查看这些地址数值。



要被观察的地址叫做“watchpoints” 如下项目可以被选择：

- 地址（可以使用符号）
- 地址 + 位数
- 位符号

ASM watch 窗口关闭时被加入的 watchpoints 会被自动保存，下次窗口打开时会被自动加载。

如果将标号或者位符号设置为 watchpoints，watchpoint 地址会在下载到目标单片机时重新计算。

无效的 watchpoints 在显示时会被标记为“-<not active>-”

所列出的 watchpoints 的次序可以通过鼠标拖放来更改

Watchpoint 的表达式、大小、进制和数值都可以在运行中更改。

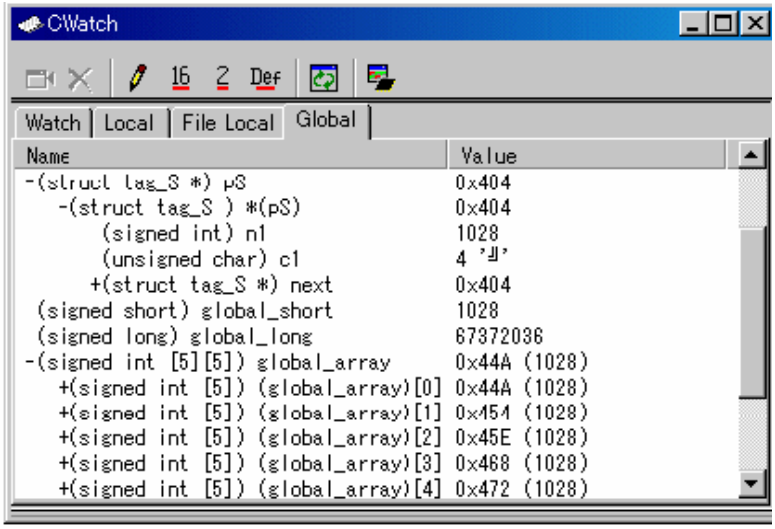
5.2.1 扩展菜单

在这个窗口内右键单击时会弹出菜单：

菜单		功能
Add...		添加一个 watchpoint。
Add Bit...		添加 bit-label watchpoint。
Remove		删除所选择的 watchpoint。
Remove All		删除所有的 watchpoints。
Set...		给所选择的 watchpoint 设置新的值
Radix	Bin	以二进制显示
	Dec	以十进制显示
	Hex	以十六进制显示
Refresh		刷新存储器数值
Layout	Address Area	选择是否显示 Address 栏
	Size Area	选择是否显示 Size 栏
RAM Monitor	Enable RAM Monitor	选择启用或禁用 RAM moniter 功能
	Sampling Period...	设置 RAM monitor 采样周期
Toolbar display		显示工具栏
Customize toolbar...		打开自定义工具栏窗口
Allow Docking		允许窗口停靠
Hide		隐藏窗口

5.3 C Watch 窗口

C Watch 窗口显示 C/C++ 语法表达式和它们的值（计算的结果）。显示在 C Watch 窗口中的 C/C++ 表达式被叫做 C watchpoints。当每一条命令执行后，C watchpoints 计算结果将会更新。



- 变量可以按照范围加以区分（Local、File Local 或者 Global）
- 显示会随着 PC 值的改变而自动更新
- 变量的值可以被更改
- 变量的进制可以分别被设置
- 任何变量都可以被加入到 Watch 选项卡中，所以它们总可以被显示：
  - 所登记的内容会按照每个工程分别存储
  - 如果两个或以上的 Watch 窗口同时打开，将要添加的变量会在所有 Watch 窗口中同时出现
- 增加 Watch 选项卡可以将 C watchpoints 分组
- 可以通过鼠标的拖放操作从别的窗口或编辑器中添加变量到 Watch 窗口
- C watchpoints 可以按名称或地址存储

注意

下列 C watch points 不能被更改：

- Bit 变量
- 寄存器变量
- 不关联地址的 C watch point （无效的 C watch point）

如果 C/C++表达式无法被正确计算，（举例来说，如果 C/C++符号未被定义），将会被表示为无效的 C watch point。它将会被显示为“--<not active>--”。但如果这个 C/C++表达式之后可以被正确计算，它将会变为有效的 C watch point。

Local，File Local 和 Global 选项卡的显示设置无法被存储，Watch 选项卡和新加入的选项卡可以被存储。

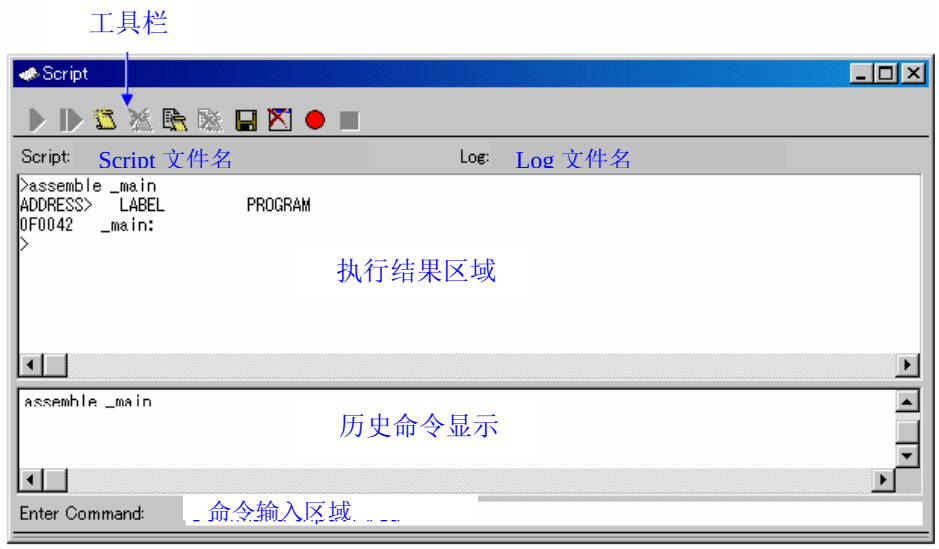
5.3.1 扩展菜单

在这个窗口内右键单击时会弹出菜单：

菜单		功能
Add...		添加一个 C watchpoint。
Remove		删除所选择的 C watchpoint。
Remove All		删除所有的 C watchpoints。
Initialize		重新对所选择的 C watchpoint 求值
Set New Value...		给所选择的 C watchpoint 设置新的值
Radix	Hex	以十六进制显示
	Bin	以二进制显示
	Default	以默认进制显示
	Toggle(All Variables)	改变进制（所有变量）
Refresh		刷新存储器数值
Hide type name		隐藏变量类型
Show char* as string		选择是否把 char* 类型的变量显示为字符串
Sort	Sort by Name	按名称排列变量
	Sort by Address	按地址排列变量
RAM Monitor	Enable RAM Monitor	选择启用或禁用 RAM moniter 功能
	Sampling Period...	设置 RAM monitor 采样周期
Add New Tab...		加入新的选项卡
Remove Tab		删除所选择的选项卡
Toolbar display		显示工具栏
Customize toolbar...		打开自定义工具栏窗口
Allow Docking		允许窗口停靠
Hide		隐藏窗口

5.4 Script 窗口

Script 窗口显示文本格式的脚本命令和执行结果。脚本命令可以通过脚本文件或者手工键入来执行。用户也可以在脚本文件中写入脚本命令并且自动执行。脚本命令的执行结果也可以存储在一个 Log 文件中。



Script 窗口有一个可以记录 1000 行运行结果的显示缓冲存储器。运行结果可以存储于一个文件中而不需指定 Log 文件

当一个脚本文件打开时，历史命令显示区域改变为脚本文件显示区域，并且显示脚本文件内容。当脚本文件嵌套时，最后一个打开的脚本文件将会显示。脚本文件显示区域显示当前正在执行的命令行。

当一个脚本文件打开但是未执行时，用户也可以从命令输入区域输入命令并执行

Script 窗口可以将执行过的命令记录在文件中，它不同于记录为 Log 文件：这个功能并不记录命令执行的结果，仅记录所执行的命令。所以记录下的文件也可以当作脚本文件使用。

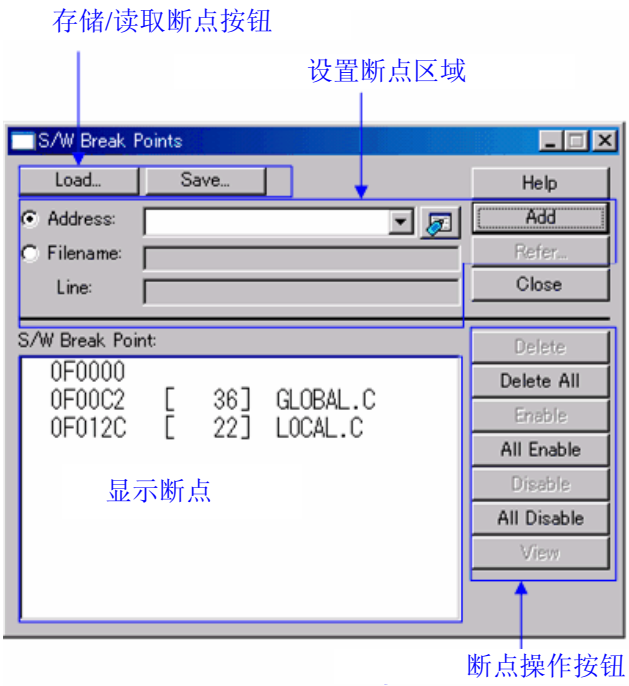
5.4.1 扩展菜单

在这个窗口内右键单击时会弹出菜单：

菜单		功能
Script	Open...	打开脚本文件
	Run	运行脚本文件
	Step	脚本文件单步运行
	Close	关闭脚本文件
View	Save...	将缓冲区存为文件
	Clear	清除缓冲区
Log	On...	打开 Log 文件并开始记录（开始输出到文件）
	Off	关闭 Log 文件并停止记录（停止输出到文件）
Record	On...	将执行过的命令存储到文件
	Off	停止记录执行过的命令
Toolbar display		显示工具栏
Customize toolbar...		打开自定义工具栏窗口
Allow Docking		允许窗口停靠
Hide		隐藏窗口

5.5 S/W Break Point 设置窗口

S/W Break Point 设置窗口允许用户设置软件断点，程序会在所指定的断点前立即停止。



- 如果用户设置了多个断点，程序会在任何一个执行到的断点停止（“或”关系）
- 用户可以连续设置断点直到点击“Close”按钮为止
- 用户可以在软件断点区域内单击来清除/启用或禁用软件断点，也可以双击它们来启用/禁用软件断点。
- 点击“Save”按钮把设置的软件断点存储到文件中。按“Load” 按钮从已存的文件中重新装载断点。如果从一个文件中读取断点，它们会与已设置的断点合并在一起

5.5.1 命令按钮

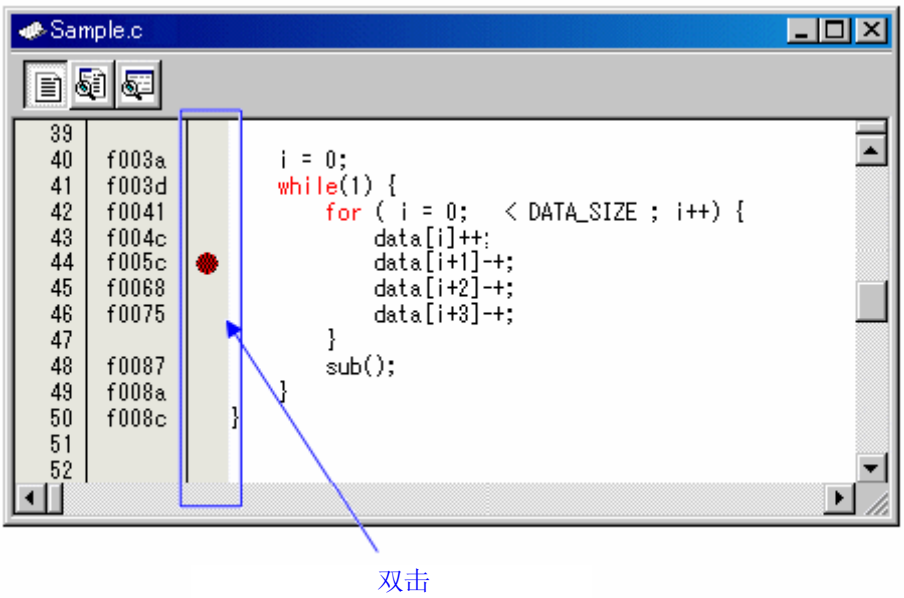
本窗口的命令按钮有如下功能：

按钮	功能
Load...	从一个已保存的文件中读取设置信息
Save...	把窗口中的内容保存到一个文件
Help	显示帮助
Add	添加断点
Refer...	打开文件选择对话框
Close	关闭窗口
Delete	移除所选择的断点
Delete All	移除所有断点
Enable	启用所选择的断点
All Enable	启用所有的断点
Disable	禁用所选择的断点
All Disable	禁用所有的断点
View	在编辑窗口中显示所选择的断点

5.5.2 在编辑窗口中设置、删除断点

可以设置断点的区域随产品型号而不同，请参考第“10.1.2 可设置软件断点的区域”。

用户可以在编辑窗口中设置断点，如果想这么做，在想设置断点一行的断点设置区域（“软件断点”一列）双击，（设置好断点后，将会出现一个红色的标记）



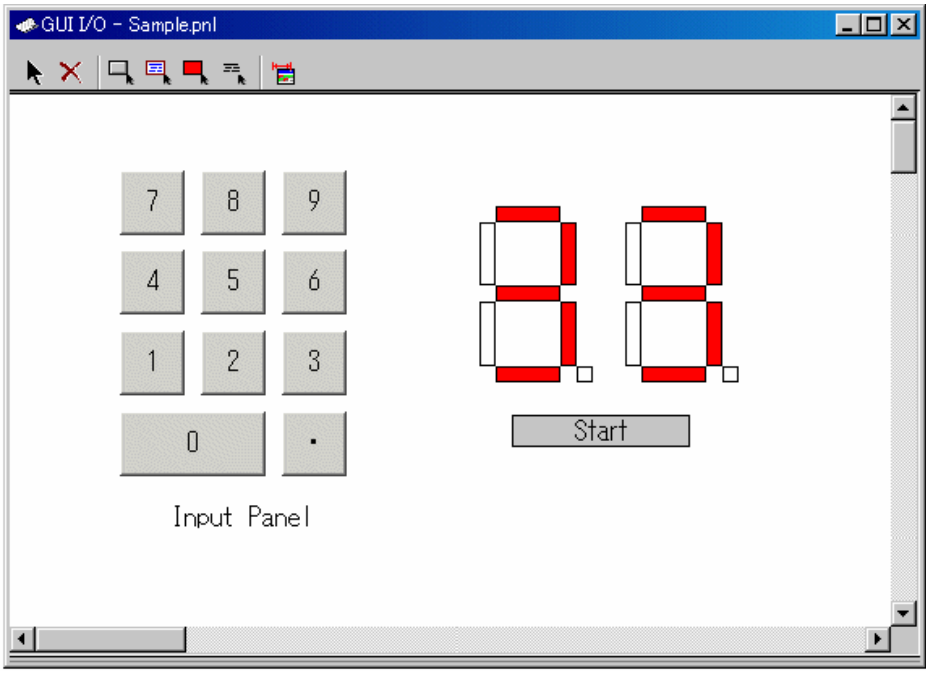
双击

用户也可以在编辑窗口中删除断点，在想删除断点一行的断点设置区域（“软件断点”列）双击即可。

在编辑窗口中，一个“软件断点”列默认设置为开启。如果想删除此列，请在菜单中选择：[Edit] -> [Define Column Format]，去除打开的对话框中的[S/W breakpoints]复选框。“软件断点”一栏会从所有编辑窗口中移除。在编辑窗口中右键单击选择[Columns] -> [S/W breakpoints]可以独立设置每一个编辑窗口的断点栏。

5.6 GUI I/O 窗口

GUI I/O 窗口允许用户创建一个用户目标键输入面板来输入到端口，然后点击创建的按钮。并且这个窗口还允许将用户目标系统的输出显示在面板中。



用户可以在本窗口中加入如下组件：

- 选项卡（字符串），根据一个指定的地址（Bit）显示/删除一个用户设定的字符串
- LED，当任意值被写入指定的地址（Bit）时，改变显示颜色
- 按钮，当按钮被按下的时候，执行一个虚拟的端口输入
- 文本框，显示字符串

用户也可以存储已经创建的面板并且重新加载它

用户可以给已经创建的部件设定 200 个地址，如果给每个部件分配不同的地址， 用户可以安排 200 个部件

5.6.1 扩展菜单

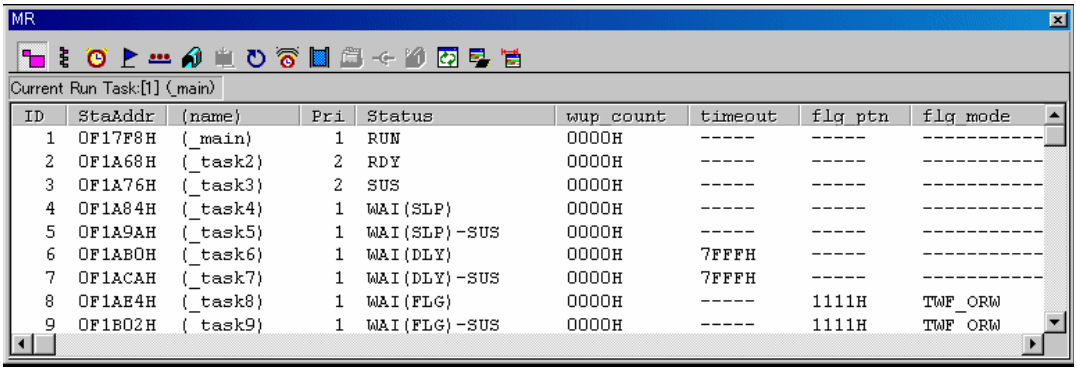
在这个窗口内右键单击时会弹出菜单：

菜单	功能
Select Item	选择一个 I/O 部件
Delete	删除所选择的 I/O 部件
Copy	复制所选择的 I/O 部件
Paste	粘贴已复制的 I/O 部件
Create Button	创建新的按钮
Create Label	创建新的选项卡
Create LED	创建新的 LED
Create Text	创建新的文本框
Display grid	显示栅格线
Save...	存储 I/O 面板文件
Load...	读取 I/O 面板文件
Sampling Period...	设置 RAM Monitor 采样周期
Toolbar display	显示工具栏
Customize toolbar...	显示自定义工具栏对话框
Allow Docking	允许窗口停靠
Hide	隐藏窗口

5.7 MR 窗口

使用 MR 来显示当前实时操作系统的状态。

仅当使用实时操作系统时，才可以使用 MR 窗口（如果下载的程序中没有使用 MR，打开 MR 窗口时将不会显示任何内容）



- 用户可以打开和显示模式一样多的 MR 窗口
- 点击特定的按钮，可以改变 MR 窗口的显示模式，显示的数据也会随之改变
- 双击想要查看的任务行，可以显示任务的内容
- 用户可以用过拖动来改变每一列的宽度
- 如果下载的程序没有使用 MR，所有可以选择显示模式的菜单都无法被选择
- 可以支持以下的显示模式

如果由 MR30 或 MR308 编译的目标程序符合 uITRON4，本窗口支持以下列表

- Task status
- Ready queue status
- Timeout queue status
- Event flag status
- Semaphore status
- Mailbox status
- Data queue status
- Cyclic handler status
- Alarm handler status
- Memory pool status

如果由 MR30 或 MR308 编译的目标程序符合 uITRON3，本窗口支持以下列表

如果基于 MR30 V1.00 的目标程序被下载，MR30 不能使用 MPL 模式。（不能从菜单中选择改变模式为 MPL 模式）

- Task status
- Ready queue status
- Timeout queue status
- Event flag status
- Semaphore status
- Mailbox status
- Cyclic handler status
- Alarm handler status
- Memory pool status

注意

当创建下载程序时，如果 Startup 文件与所用的 MRxx 版本号不符，MR 窗口和 MR 指令将无法正常工作。

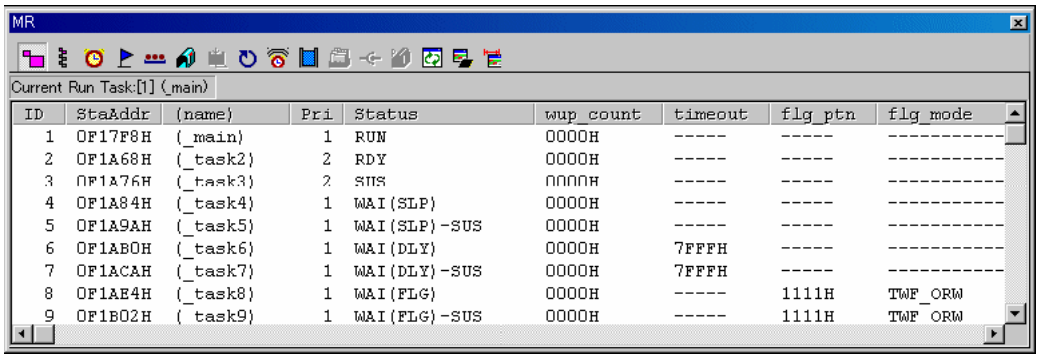
5.7.1 扩展菜单

在这个窗口内右键单击时会弹出菜单：

菜单		功能
Mode	Task	显示任务状态
	Ready Queue	显示 Ready 状态
	Timeout Queue	显示 Timeout 状态
	Event Flag	显示 Event Flag 状态
	Semaphore	显示 Semaphore 状态
	Mailbox	显示 Mailbox 状态
	Data Queue	显示 Data Queue 状态
	Cyclic Handler	显示 Cyclic Handler 状态
	Alarm Handler	显示 Alarm Handler 状态
	Memory Pool	显示 Memory Pool 状态
	Message Buffer	显示 Message Buffer 状态
	Port	显示 Port 状态
	Mailbox(with Priority)	显示 Mailbox（附带等级）状态
Context...		显示 Context
Layout	Status Bar	是否显示状态栏
Refresh		刷新存储器数据
RAM Monitor	Enable RAM Monitor	开启或关闭 RAM Monitor 功能
	Sampling Period...	设置 RAM Monitor 采样周期
Toolbar display		显示工具栏
Customize toolbar...		开打自定义工具栏对话框
Allow Docking		允许窗口停靠
Hide		隐藏窗口

5.7.2 显示任务状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Task]。



双击任何一行，任务的信息显示在 Context 对话框中。关于 Context 对话框的具体信息，请参考“5.7.12 显示 Task Context” 下列数据显示于状态栏中

(1) 显示任务状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有任务都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	功能
ID	任务 ID
StaAddr	任务起始地址
(name)	任务名
Pri	级别
Status*1	任务状态
wup_count	Wake-up 计数
timeout	Timeout 数值
flg_ptn	Event flag 的 Wait bit pattern
flg_mode*2	Event flag 的取消等待条件

*1 任务状态

显示	状态
RUN	运行状态
RDY	准备好状态
SUS	休眠状态
DMT	静止状态
WAI(SLP)	睡眠状态
WAI(SLP)-SUS	睡眠状态（双等待）
WAI(DLY)	因 dly_tsk 的时间等待状态
WAI(DLY)-SUS	因 dly_tsk（双等待）的时间等待状态
WAI(FLG)	Event flag 等待状态
WAI(FLG)-SUS	Event flag 等待状态（双等待）
WAI(SEM)	Semaphore 等待状态
WAI(SEM)-SUS	Semaphore 等待状态（双等待）
WAI(MBX)	Message 等待状态
WAI(MBX)-SUS	Message 等待状态（双等待）
WAI(SLP-TMO)	Sleep 附带超时状态
WAI(SLP-TMO)-SUS	Sleep 附带超时状态（双等待）
WAI(FLG-TMO)	Event flag 附带超时等待状态
WAI(FLG-TMO)-SUS	Event flag 附带超时等待状态（双等待）
WAI(SEM-TMO)	Semaphore 附带超时等待状态
WAI(SEM-TMO)-SUS	Semaphore 附带超时等待状态（双等待）
WAI(MBX-TMO)	Message 附带超时等待状态
WAI(MBX-TMO)-SUS	Message 附带超时等待状态（双等待）

*2 显示 Event Flag 的取消等待条件

flg_mode	状态
TWF_ANDW	等待直到 Wait Pattern 的所有位被置起（“与”等待）
TWF_ANDW+TWF_CLR	当“与”等待发生时，将 event flag 清零，并且任务的等待状态被取消
TWF_ORW	等待 Wait Pattern 的任何一位被置起（“或”等待）
TWF_ORW+TWF_CLR	当“或”等待发生时，将 event flag 清零，并且任务的等待状态被取消

(2) 显示任务状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有任务都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	任务 ID
Name	任务名称
Pri	级别
Status*1	任务状态
Wupcnt	Wake-up 计数
Actcnt	Activated 计数
Tmout	Timeout 值
Flgptn	Event flag 的 Wait Bit Pattern
Wfmode*2	Event flag 的等待取消条件

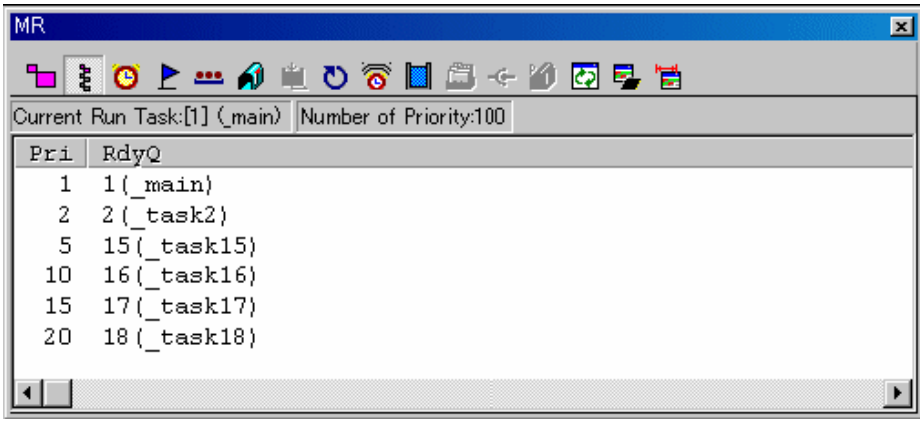
*1 任务状态

显示	状态
RUN	Run 状态
RDY	Ready 状态
SUS	Suspend 状态
DMT	Dormant 状态
WAI(SLP)	Sleep 状态
WAI(SLP)-SUS	Sleep 状态（双等待）
WAI(DLY)	因 dly_tsk 的时间等待状态
WAI(DLY)-SUS	因 dly_tsk 的时间等待状态（双等待）
WAI(FLG)	Event flag 等待状态
WAI(FLG)-SUS	Event flag 等待状态（双等待）
WAI(SEM)	Semaphore 等待状态
WAI(SEM)-SUS	Semaphore 等待状态（双等待）
WAI(MBX)	Message 等待状态
WAI(MBX)-SUS	Message 等待状态（双等待）
WAI(SDTQ)	Transmission data 等待状态
WAI(SDTQ)-SUS	Transmission data 等待状态（双等待）
WAI(RDTQ)	Reception data 等待状态
WAI(RDTQ)-SUS	Reception data 等待状态（双等待）
WAI(VSDTQ)	Transmission extended data 等待状态
WAI(VSDTQ)-SUS	Transmission extended data 等待状态（双等待）
WAI(VRDTQ)	Reception extended data 等待状态
WAI(VRDTQ)-SUS	Reception extended data 等待状态（双等待）
WAI(MPF)	定长度存储器池等待
WAI(MPF)-SUS	定长度存储器池等待（双等待）
WAI(SLP-TMO)	附带超时睡眠状态
WAI(SLP-TMO)-SUS	附带超时睡眠状态（双等待）
WAI(FLG-TMO)	Event flag 附带超时等待状态
WAI(FLG-TMO)-SUS	Event flag 附带超时等待状态（双等待）
WAI(SEM-TMO)	Semaphore 附带超时等待状态
WAI(SEM-TMO)-SUS	Semaphore 附带超时等待状态（双等待）
WAI(MBX-TMO)	Message 附带超时等待状态
WAI(MBX-TMO)-SUS	Message 附带超时等待状态（双等待）
WAI(SDTQ-TMO)	Transmission data 附带超时等待状态
WAI(SDTQ-TMO)-SUS	Transmission data 附带超时等待状态（双等待）
WAI(RDTQ-TMO)	Reception data 附带超时等待状态
WAI(RDTQ-TMO)-SUS	Reception data 附带超时等待状态（双等待）
WAI(VSDTQ-TMO)	Transmission extended data 附带超时等待状态
WAI(VSDTQ-TMO)-SUS	Transmission extended data 附带超时等待状态（双等待）
WAI(VRDTQ-TMO)	Reception extended data 附带超时等待状态
WAI(VRDTQ-TMO)-SUS	Reception extended data 附带超时等待状态（双等待）
WAI(MPF-TMO)	定长度存储器池附带超时等待
WAI(MPF-TMO)-SUS	定长度存储器池附带超时等待（双等待）

*2 显示 Event Flag 的等待取消条件

Wfmode	状态
TWF_ANDW	等待直到 Wait Pattern 的所有位被置起（“与”等待）
TWF_ORW	等待 Wait Pattern 的任何一位被置起（“或”等待）

5.7.3 显示 Ready Queue 状态



在 MR 窗口中，选择弹出菜单 - [Mode] -> [Ready Queue]。

下列数据会显示在状态栏中

(1) 显示 Ready Queue 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
Pri	显示优先级
RdyQ	显示 ready queue 中的 ID 号和任务名称

RdyQ 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

(2) 显示 Ready Queue 状态

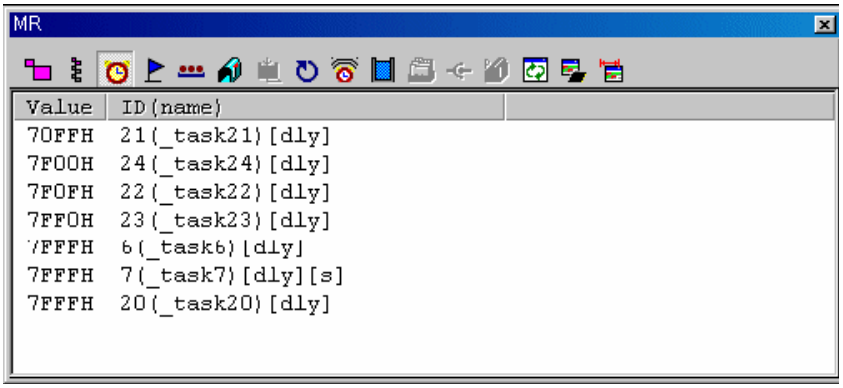
各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
Pri	显示优先级
RdyQ	显示 ready queue 中的 ID 号和任务名称

RdyQ 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

5.7.4 显示 Timeout Queue 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Timeout Queue]。



(1) 显示 Timeout Queue 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

各项目功能如下所述，当前等待中的任务按照超时时间降序排列 （当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
Value	显示每个任务的超时值
ID(name)	显示 timeout queue 中的任务 ID

下面的字符串用于指示等待状态

字符串	等待状态
[slp]	因 tslp_tsk 等待
[dly]	因 dly_tsk 等待
[flg]	因 twai_flg 等待
[sem]	因 twai_sem 等待
[mbx]	因 trcv_msg 等待

当连接到 Timeout queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 ID 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]

(2) 显示 Timeout Queue 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

各项目功能如下所述，当前等待中的任务按照超时时间降序排列 （当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
Tmout	显示每个任务的超时时间（毫秒）
ID(Name)	显示 timeout queue 中的任务 ID

下面的字符串用于指示等待状态

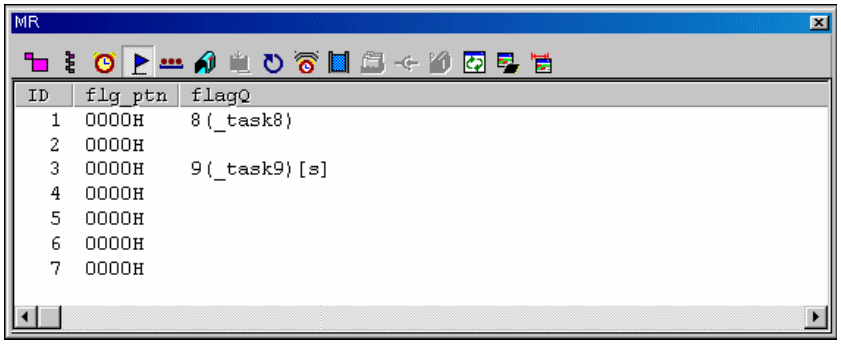
字符串	等待状态
[slp]	因 tslp_tsk 等待
[dly]	因 dly_tsk 等待
[flg]	因 twai_flg 等待
[sem]	因 twai_sem 等待
[mbx]	因 trcv_msg 等待
[mpf]	因 tget_mpf 等待
[sdtq]	因 tsnd_dtq 等待
[rdtq]	因 trcv_dtq 等待
[vsdtq]	因 vtsnd_dtq 等待
[vrdtq]	因 vtrcv_dtq 等待

当连接到 Timeout queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 ID 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]

5.7.5 显示 Event Flag 状态

在 MR 窗口中，选择弹出菜单- [Mode] -> [Event Flag]。



(1) 显示 Event Flag 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有任务都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	event flag 的 ID 编号
flg_ptn	每个 event flag 的 Bit pattern
flagQ	event flag 队列中的任务 ID 和任务名称

当连接到 event flag 队列中的任务启用了 Timeout（在 twai_flg 中等待），将会有一个字符串“[tmo]”显示在 Flag Q 中，表明启用了 Timeout。当连接到 Timeout queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 Flag Q 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

FlagQ 一栏中的任务名只显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

(2) 显示 Event Flag 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有任务都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	event flag 的 ID
Flgatr	每个 event flag 的属性
Flgptn	每个 event flag 的 Bit pattern
Flag Queue	event flag 队列中的任务 ID 和任务名称

下列项目显示在 Flgatr 区域中：

TA_TFIFO	任务等待队列（按先进先出顺序）
TA_TPRI	任务等待队列（按任务级别顺序）
TA_WSGL	只允许一个任务进入等待 event flag 状态
TA_WMUL	允许多个任务进入等待 event flag 状态
TA_CLR	当一个任务从等待 event flag 状态脱离时，Eventflag 的 bit pattern 被清除

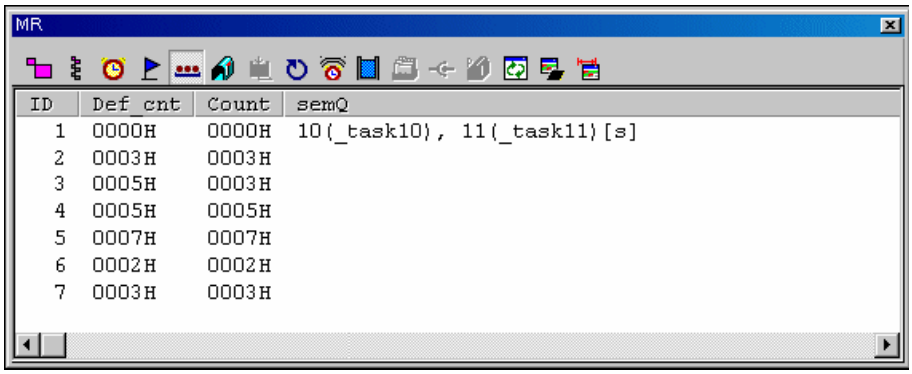
当连接到 event flag 队列中的任务启用了 Timerout（在 twai_flg 中等待），将会有一个字符串“[tmo]”显示在 Flag Q 中，表明启用了 Timeout。当连接到 Timeout queue 的任务实强制等待的任务时，会显示一个字符“[s]”，表明 Flag Q 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

FlagQ 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

5.7.6 显示 Semaphore 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Semaphore]。



(1) 显示 Semaphore 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有 SEMs 任务都以任务 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	semaphore 的 ID 编号
Def_cnt	semaphore 计数器的默认值
Count	semaphore 计数
semQ	semaphore 队列中的任务 ID 和任务名称

当连接到 event flag 队列中的任务启用了 Timeout（在 twai_sem 中等待），将会有一个字符串“[tmo]”显示在 semQ 中，表明启用了 Timeout。当连接到 SEM queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 semQ 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

semQ 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

(2) 显示 Semaphore 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有 SEMs 任务都以任务 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	semaphore 的 ID 编号
Sematr	每个 semaphore 的状态
Semcnt	semaphore 计数
semQ	semaphore 队列中的任务 ID 和任务名称

下列项目在 Sematr 区域中显示：

TA_TFIFO	任务等待队列（按先进先出顺序）
TA_TPRI	任务等待队列（按任务级别顺序）

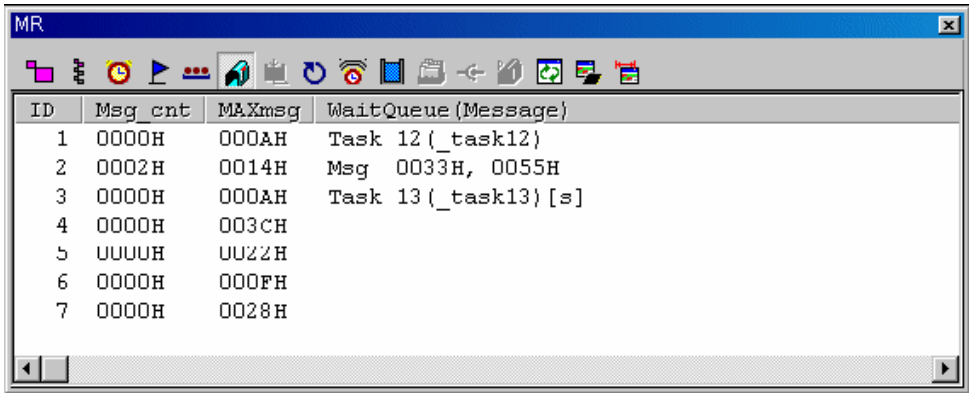
当连接到 event flag 队列中的任务启用了 Timeout（在 twai_sem 中等待），将会有一个字符串“[tmo]”显示在 semQ 中，表明启用了 Timeout。当连接到 SEM queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 semQ 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

semQ 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

5.7.7 显示 Mailbox 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Mailbox]。



(1) 显示 Mailbox 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有 MailBox 都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	mailbox 的 ID 编号
Msg_cnt	每一个 mailbox 的消息数量
MAXmsg	每一个 mailbox 能存储的最大消息数
Wait Queue(Message)	mailbox 存储器存储的消息或者等待消息的任务的 ID 号码

如果有消息存储时，WaitQueue (Message)区域显示一个字符串“Msg”（当如上所述的 Msg_cont 不为零时），然后显示已经存储的消息。当没有消息被存储时（当 Msg_cont 为零时），WaitQueue 区域显示一个字符串“Task”。如果有任务正在等待消息，就显示任务的 ID 号。

当连接到 mail box 队列中的任务启用了 Timeout（在 trcv_msg 中等待），将会有一个字符串“[tmo]”显示在 WaitQueue 中，表明启用了 Timeout。

当连接到 mail box queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 WaitQueue 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

WaitQueue 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

(2) 显示 Mailbox 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有 MailBox 都以任务 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	mailbox 的 ID 编号
Mbxatr	每个 mailbox 的属性
Mailbox Queue (Wait)	mailbox 存储器存储的消息或者等待消息的任务的 ID 号码
Mailbox Queue (Message)	mailbox 中存储的消息

以下内容显示在 Mbxatr 区域中：

TA_TFIFO	任务等待队列以先入先出优先级排列
TA_TPRI	任务等待队列以任务优先级排列
TA_MFIFO	消息队列以先入先出排列
TA_MPRI	消息队列以消息优先级排列

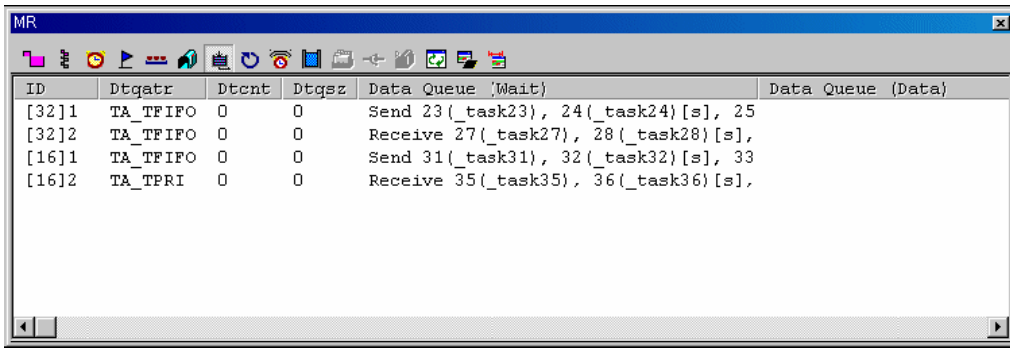
当连接到 mail box 队列中的任务启用了 Timeout（在 trcv_msg 中等待），将会有一个字符串“[tmo]”显示在 WaitQueue 中，表明启用了 Timeout。当连接到 mail box queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 WaitQueue 一栏中所列的任务被强制等待

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

WaitQueue 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

5.7.8 显示 Data Queue 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Data Queue]。



(1) 显示 Data Queue 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有 data queues 都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	data queue 的 ID 号
Dtqatr	每一个 date queue 的属性
Dtcnt	每一个 data queue 中的消息数量
Dtqsz	每一个 data queue 中可以容纳的最大消息数量
Data Queue (Wait)	正在等待接受或传送消息的任务的 ID 号
Data Queue (Data)	data queue 中存储的消息

ID 栏显示的内容取决于显示的是标准数据（32 位）还是扩展数据（16 位）

MR308/4

- 如果选择了标准数据（32 位），ID 栏显示字符串“[32]”和 data queue 的 ID 号
- 如果选择了扩展数据（16 位），ID 栏显示字符串“[16]”和 data queue 的 ID 号

MR30/4

- 如果选择了标准数据（16 位），ID 栏显示字符串“[16]”和 data queue 的 ID 号
- 如果选择了扩展数据（32 位），ID 栏显示字符串“[32]”和 data queue 的 ID 号

以下内容显示在 Dtqatr 区域中：

TA_TFIFO	任务等待队列以先入先出优先级排列
TA_TPRI	任务等待队列以任务优先级排列

如果一个任务正在等待发送消息，Data Queue 区域会显示字符串“Send”，然后显示任务的 ID 和能带发送消息的任务的名称。同样的，如果一个任务正在等待接受消息，会显示字符串“Receive”然后显示任务的 ID 和等待发送消息的任务的名称。

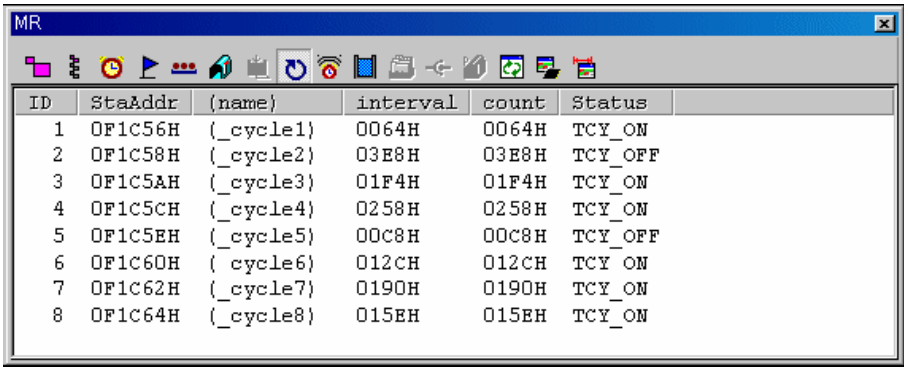
当连接到 date queue 队列中的任务启用了 Timeout（在 trcv_msg 中等待），将会有一个字符串“[tmo]”显示在 Data Queue 中，表明启用了 Timeout。当连接到 date queue 的任务是强制等待的任务时，会显示一个字符“[s]”，表明 Date Queue 一栏中所列的任务被强制等待。

通常显示	26(_task26)
WAIT-SUSPEND 状态时显示	26(_task26)[s]
WAIT-SUSPEND with time out 状态时显示	26(_task26)[tmo][s]

Date Queue 一栏中的任务名最多能显示 8 个字符，当任务名长度超过 8 个字符时，多余的字符将被忽略

5.7.9 显示 Cycle Handler 状态

在 MR 窗口中，选择弹出菜单 - [Mode] - [Cyclic Handler]



(1) 显示 Cycle Handler 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有 cycle handlers 都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	cycle handler 的 ID 号
StaAddr	cycle handler 的起始地址
(name)	cycle handler 的名称
interval	中断周期
count	中断计数
Status	cycle start handler 动作状态

以下内容显示在状态区域中

TCY_ON	Cycle handler 允许
TCY_OFF	Cycle handler 禁止

(2) 显示 Cycle Handler 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有 cycle handlers 都以 ID 号列出。各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

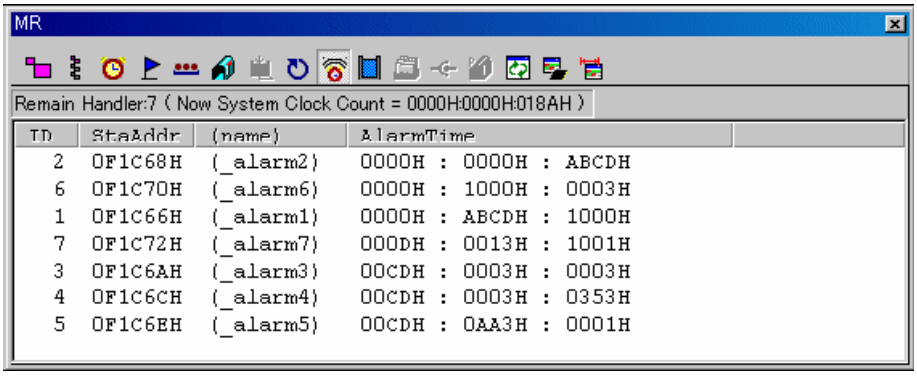
项目	内容
ID	cycle handler 的 ID 号
Name	cycle handler 的名称
Cycphs	激活相位(以毫秒为单位）
Cyctim	激活循环时间(以毫秒为单位）
Tmout	距离下次激活的时间（以毫秒为单位）
Status	cycle start handler 活动状态

以下内容显示在状态区域中

TCY_ON	Cycle handler 允许
TCY_OFF	Cycle handler 禁止

5.7.10 显示 Alarm Handler 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Alarm Handler]。



当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时，如下信息显示于状态栏：

(1) 显示 Alarm Handler 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在所有设置中定义了的 cycle start handlers 中，只有目前还没有开始的项目会以开始时间降序排列。各项功能如下所示：（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	alarm handler 的 ID 号
StaAddr	alarm handler 的起始地址
(name)	alarm handler 的名称
AlarmTime	alarm handler 的开始时间

(2) 显示 Alarm Handler 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在所有设置中定义了的 cycle start handlers 中，只有目前还没有开始的项目会以开始时间降序排列。各项功能如下所示：（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

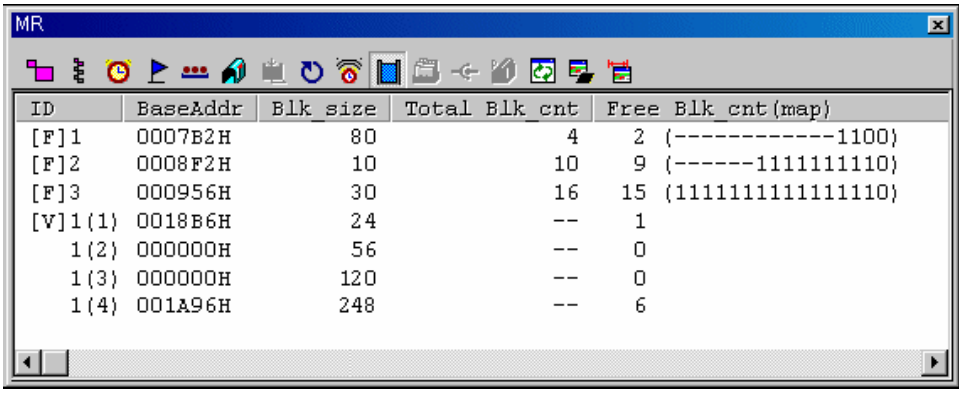
项目	内容
ID	alarm handler 的 ID 号
Name	alarm handler 的名称
Almtim	距离 alarm handler 下次激活的时间（以毫秒为单位）
Status	alarm handler 活动状态

以下内容显示在状态区域中

TALM_STA	Alarm handler 处于操作状态
TALM_STP	Alarm handler 处于非操作状态

5.7.11 显示 Memory Pool 状态

在 MR 窗口中，选择弹出菜单 - [Mode] -> [Memory Pool]。



(1) 显示 Memory Pool 状态（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

在设置中定义的所有 memory pools 都以 ID 号列出。（首先是固定长度的数据，其次是可选长度的数据）各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V3.0 标准时）

项目	内容
ID	memory pool 的 ID 号
BaseAddr	memory pool 起始地址
Blk_Size	memory pool 的区块大小
Total Blk_cnt	memory pool 的所有区块大小
Free Blk_cnt(map)	存储器中未使用的区块数量信息（bit information）

ID 一栏显示的内容和所选项有关，固定长度或可选长度。

- 如果数据是固定长度，ID 一栏会显示字符串“[F]”和 memory pool 的 ID 号

- 如果是可变长度，第一行的内容是字符串“[V]”，memory pool 的 ID 号，和一个区块的 ID 号。显示 在第二到第四行的是 memory pool 的 ID 号和区块的 ID 号。区块的 ID 号在圆括号之中

当指定了可选长度的 memory pool 时，“—”显示在 Total Mlk_cut 一栏中。在 Free Blk_cnt (map) 一栏中不显示 Bit 信息。

当指定了固定长度的 memory pool 时，Free Blk_cnt (map)一栏中的显示格式如下：

项目	内容
’0’	Memory block 正在使用（忙）
’1’	Memory block 未使用（就绪）
’-’	没有 memory block

(2) 显示 Memory Pool 状态（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

在设置中定义的所有 memory pools 都以 ID 号列出，各项目功能如下所述（当实时操作系统是 MRxx 并符合 uITRON V4.0 标准时）

项目	内容
ID	memory pool 的 ID 号
Mplatr	每一个 memory pool 的属性
Mpladr	memory pool 起始地址
Mplsz	memory pool 的大小
Blkcnt	固定长度 memory pool 的总区块数
Fblkcnt	未使用的区块的数量和信息
Memory Pool Queue	显示在 memory pool 中等待的任务的 ID 号和名称

以下内容显示在 Mplatr 区域中：

TA_TFIFO	任务等待队列以先入先出优先级排列
TA_TPRI	任务等待队列以任务优先级排列

ID 一栏显示的内容和所选项有关，固定长度或可选长度。

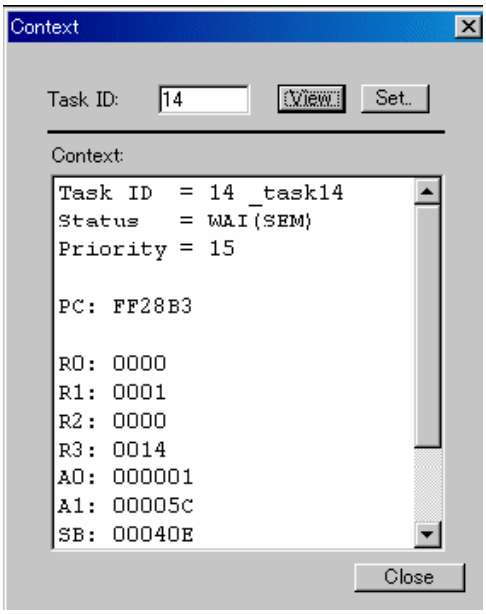
- 如果数据是固定长度，ID 一栏会显示字符串“[F]”和 memory pool 的 ID 号

- 如果是可变长度，第一行的内容是字符串“[V]”，memory pool 的 ID 号，和一个区块的 ID 号。显示在第二到第四行的是 memory pool 的 ID 号和区块的 ID 号。区块的 ID 号在圆括号之中。

5.7.12 显示 Task Context

(1) 显示 Task Context

在 MR 窗口中，选择弹出菜单 - [Context...]，打开 Context 对话框。Context 对话框用于参考/指定特定任务的背景。用户也可以通过 task state display mode 时双击 data display 区域来打开此



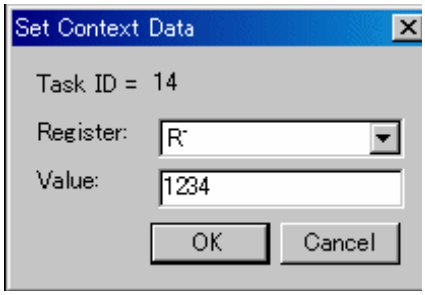
对话框。

在任务 ID 中输入任务的 ID 号，然后单击 View 按钮（或者按回车键）。 所选任务的背景会显示在 Context field 栏中。

- 如果单击“VIEW”时，在任务 ID 一栏输入的是“RUN”或者“DMT”，背景就不会被显示。（Context 区域中，只显示任务 ID 和任务状态）
- 如果单击“VIEW”时，在任务 ID 一栏中输入了不存在的任务 ID 号，会出现错误提示

(2) 改变 task context

在任务 ID 一栏中输入任务 ID 号，然后单击 Set 按钮。Set Context 对话框将打开。Set Context 对话框用来给特定的任务输入特定寄存器的值。



在 Register 一栏中选择要更改的寄存器，在“Value:”一栏中输入值。如果在“Value:”中输入的值错误，或者超过范围，会出现错误。

6. 脚本命令列表

有如下命令：

显示为黄色的命令可以在运行时执行。

本产品不支持附加有“*”的命令

6.1 脚本命令列表(按功能分类）

6.1.1 执行命令

命令名称	代码	功能
Go	G	带断点执行程序
GoFree	GF	自由执行程序
Stop	-	停止程序执行
Status	-	检查 MCU 执行状况
Step	S	中断用户输入直到经过了指定的时间以后
StepInstruction	SI	步进执行
OverStep	O	执行源文件 Overstep
OverStepInstruaction	OI	执行指令 Overstep
Return	RET	执行源文件 return
ReturnInstruction	RETI	执行指令 return
Reset	-	MCU 复位

6.1.2 文件操作命令

命令名称	代码	功能
Load	L	下载目标程序
LoadHex	LH	下载 Intel HEX-格式文件
LoadMot*	LM	下载 Motorola S-格式文件
LoadSymbol	LS	载入源程序/ASM 符号信息
Reload	-	重新下载目标程序
UploadHex	UH	输出数据到 Intel HEX-格式文件

6.1.3 寄存器操作指令

命令名称	代码	功能
Register	R	检查和设置寄存器的值

6.1.4 存储器操作命令

命令名称	代码	功能
DumpByte	DB	显示存储器的内容（以 1 字节为单位）
DumpWord*	DW	显示存储器的内容（以 2 字节为单位）
DumpLword*	DL	显示存储器的内容（以 4 字节为单位）
SetMemoryByte	MB	检查或改变存储器的内容(以 1 字节为单位)
SetMemoryWord*	MW	检查或改变存储器的内容(以 2 字节为单位)
SetMemoryLword*	ML	检查或改变存储器的内容(以 4 字节为单位)
FillByte	FB	用指定的内容填充存储器区块（以 1 字节为单位）
FillWord*	FW	用指定的内容填充存储器区块（以 2 字节为单位）
FillLword*	FL	用指定的内容填充存储器区块（以 4 字节为单位）
Move	-	移动存储器区块
MoveWord*	MOVEW	移动存储器区块（以 2 字节为单位）

6.1.5 汇编/反汇编指令

命令名称	代码	功能
Assemble	A	逐行汇编
DisAssemble	DA	存储器中逐行反汇编
Module	MOD	显示模块名称
Scope	-	设置和检查有效的局部符号范围
Section	SEC	检查段信息
Bit*	-	检查和设置位符号
Symbol	SYM	检查汇编符号
Label	-	检查汇编选项卡
Express	EXP	显示汇编表达式

6.1.6 软件中断设定指令

命令名称	代码	功能
SoftwareBreak	SB	设置和检查软件中断
SoftwareBreakClear	SBC	清除软件中断
SoftwareBreakClearAll	SBCA	清除所有软件中断
SoftwareBreakDisable	SBD	禁用软件中断
SoftwareBreakDisableAll	SBDA	禁用所有软件中断
SoftwareBreakEnable	SBE	启用软件中断
SoftwareBreakEnableAll	SBEA	启用所有软件中断
BreakAt	-	在指定行号设置软件中断
BreakIn	-	在指定函数处设置软件中断

6.1.7 脚本/Log 文件命令

命令名称	代码	功能
Script	-	打开和运行一个脚本文件
Exit	-	退出脚本文件
Wait	-	输入命令前等待事件发生
Pause	-	等待用户输入
Sleep	-	中断用户输入直到经过了指定的时间以后
Logon	-	输出屏幕内容到一个 Log 文件
Logoff	-	停止输出屏幕内容到一个 Log 文件
Exec	-	执行外部程序

6.1.8 程序显示命令

命令名称	代码	功能
Func	-	检查函数名并且显示函数内容
Up*	-	显示正在调用的函数
Down*	-	显示调用过的函数
Where*	-	显示函数调用状态
Path	-	设置和检查搜索路径
AddPath	-	加入搜索路径
File	-	检查一个文件/显示文件内容

6.1.9 C 语言调试命令

命令名称	代码	功能
Print	-	检查指定的 C 变量表达式的值
Set	-	设置指定的 C 变量表达式的值

6.1.10 实时操作系统命令

命令名称	代码	功能
MR*	-	显示实时操作系统状态（MRxx）

6.1.11 功能命令

命令名称	代码	功能
Radix	-	设置和查看数字输入的进制
Alias	-	设置和查看命令对齐方式
UnAlias	-	取消命令对齐方式
UnAliasAll	-	取消所有命令对齐方式
Version	VER	显示版本号
Date	-	显示日期
Echo	-	显示消息
CD	-	打开窗口

6.2 脚本命令列表（按字母顺序排列）

命令名称	代码	功能
AddPath	-	加入搜索路径
Alias	-	设置和查看命令对齐方式
Assemble	A	逐行汇编
Bit*	-	检查和设置位符号
BreakAt	-	在指定行号设置软件中断
BreakIn	-	在指定函数处设置软件中断
CD	-	打开窗口
Date	-	显示日期
DisAssemble	DA	存储器中逐行反汇编
Down*	-	显示调用过的函数
DumpByte	DB	显示存储器的内容（以 1 字节为单位）
DumpWord*	DW	显示存储器的内容（以 2 字节为单位）
DumpLword*	DL	显示存储器的内容（以 4 字节为单位）
Echo	-	显示消息
Exec	-	执行外部程序
Exit	-	退出脚本文件
Express	EXP	显示汇编表达式
File	-	检查一个文件/显示文件内容
FillByte	FB	用指定的内容填充存储器区块（以 1 字节为单位）
FillWord*	FW	用指定的内容填充存储器区块（以 2 字节为单位）
FillLword*	FL	用指定的内容填充存储器区块（以 4 字节为单位）
Func	-	检查函数名并且显示函数内容
Go	G	带断点执行程序
GoFree	GF	自由执行程序
Label	-	检查汇编选项卡
Load	L	下载目标程序
LoadHex	LH	下载 Intel HEX-格式文件
LoadMot*	LM	下载 Motorola S-格式文件
LoadSymbol	LS	载入源程序/ASM 符号信息
Logoff	-	停止输出屏幕内容到一个 Log 文件
Logon	-	输出屏幕内容到一个 Log 文件
Module	MOD	显示模块名称
Move	-	移动存储器区块
MoveWord*	MOVEW	移动存储器区块（以 2 字节为单位）
MR*	-	显示实时操作系统状态（MRxx）
OverStep	O	执行源文件 Overstep
OverStepInstruaction	OI	执行指令 Overstep
Path	-	设置和检查搜索路径
Pause	-	等待用户输入
Print	-	检查指定的 C 变量表达式的值
Radix	-	设置和查看数字输入的进制
Register	R	检查和设置寄存器的值
Reload	-	重新下载目标程序

Reset	-	MCU 复位
Return	RET	执行源文件 return
ReturnInstruction	RETI	执行指令 return
Scope	-	设置和检查有效的局部符号范围
Script	-	打开和运行一个脚本文件
Section	SEC	检查段信息
Set	-	设置指定的 C 变量表达式的值
SetMemoryByte	MB	检查或改变存储器的内容（以 1 字节为单位）
SetMemoryWord*	MW	检查或改变存储器的内容（以 2 字节为单位）
SetMemoryLword*	ML	检查或改变存储器的内容（以 4 字节为单位）
Sleep	-	中断用户输入直到经过了指定的时间以后
SoftwareBreak	SB	设置和检查软件中断
SoftwareBreakClear	SBC	清除软件中断
SoftwareBreakClearAll	SBCA	清除所有软件中断
SoftwareBreakDisable	SBD	禁用软件中断
SoftwareBreakDisableAll	SBDA	禁用所有软件中断
SoftwareBreakEnable	SBE	启用软件中断
SoftwareBreakEnableAll	SBEA	启用所有软件中断
Status	-	检查 MCU 执行状况
Step	S	中断用户输入直到经过了指定的时间以后
StepInstruction	SI	步进执行
Stop	-	停止程序执行
Symbol	SYM	检查汇编符号
UnAlias	-	取消命令对齐方式
UnAliasAll	-	取消所有命令对齐方式
Up*	-	显示正在调用的函数
UploadHex	UH	输出数据到 Intel HEX-格式文件
UploadMot*	UM	输出数据到 Motorola S-格式文件
Version	VER	显示版本号
Wait	-	输入命令前等待事件发生
Where*	-	显示函数调用状态

7. 编写脚本文件

本调试器允许用户在脚本窗口中运行脚本文件。脚本文件包含了自动运行脚本命令所必须的控制指令。

7.1 脚本文件的构成元素

脚本文件中可以包含以下内容：

- 脚本命令
- 赋值语句
- 条件语句（if，else，endi）程序执行到这些语句时按条件分支。
- 循环语句（while，endw）在这些语句中，一组语句循环执行。
- 中断语句，退出内部循环
- 注释语句；用户可以在脚本文件中加入注释。当脚本命令执行的时候注释会被忽略。

脚本文件的每一行只能包含一个语句。用户不能在一行中写入多个语句或把一个语句写在多行上。

注意

- 不能在写有脚本命令的同一行中加入注释
- 脚本命令文件可以嵌套 5 层
- If 语句和 while 语句可以嵌套 32 层
- 每个脚本文件中，If 语句必须和 endi 语句配对，while 语句必须和 endw 语句配对
- 脚本文件中的表达式可被看作无符号类型。因此，如果在 while 或者 if 语句中使用负数进行比较可能会出现问题。
- 每一行可以容纳 4096 个字符，如果超过的话将会出现错误
- 当包含非法命令的脚本文件被自动执行时（当用户选择[Option] -> [Script]-> [Run] 的时候，或者单击脚本窗口中的按钮时），即使错误被检测到，执行仍然会继续，除非脚本文件本身发生错误无法读取。如果发生了错误，但是执行继续进行的话，后续的操作无法保证正确，换句话说，无法保证正确性。

7.1.1 脚本命令

用户可以使用与脚本窗口中一样的命令。也可以在脚本命令中嵌套另外的脚本命令（最大 10 层）

7.1.2 赋值语句

赋值语句定义和初始化宏变量，下例说明了所用的格式：

```
%macro-variable = expression
```

- 变量名中可以使用数字/字母/下划线（_）。但是，首字母不能是数字。
- 可以将 0h 到 FFFFFFFFh 之间的整数赋给变量，如果赋一个负数，会按照 2 的补数进行处理
- 可以在语句中使用宏变量
- 必须加入前缀“%”符号

7.1.3 条件语句

在条件语句中，根据条件的真假执行不同的语句。下例说明了所用的格式：

```
if ( expression )
    statement 1
else
    statement 2
endi
```

- 如果表达式为真（不为 0）执行语句 1，如果为假（0）执行语句 2
- 可以不使用 else 一段。如果没有 else 而且条件为假，程序会跳至 endi 之后的一句执行。
- If 语句可以嵌套（最多 32 层）

7.1.4 循环语句（while，endw）和 Break 语句

在循环语句中，当表达式为真时一组语句被循环执行。下例说明了所用的格式：

```
while ( expression )
    statement
endw
```

- 如果表达式为真，这组语句被重复执行；如果为假，则退出循环（并且执行 endw 语句之后的命令）。
- while 语句可以嵌套 32 级
- 使用 Break 语句强制退出 while 循环，如果 while 语句嵌套，break 语句退出最内侧的循环

7.1.5 注释语句

用户可以在脚本文件中插入注释语句，下例说明了所用的格式：

```
;character string
```

- 在分号（;）后写入语句。分号前只允许有空格和 TAB
- 当执行的时候，注释语句被忽略

7.2 编写表达式

本调试器允许使用表达式来指明地址/数据/和通过断点的次数等等。下例描述了如何使用表达式：

>DumpByte TABLE1

>DumpByte TABLE1+20

在表达式中可以使用如下内容：

- 常量
- 符号和标号
- 宏变量
- 寄存器的值
- 存储器变量
- 行号
- 字符常量
- 操作符

7.2.1 常量

用户可以使用二进制/八进制/十六进制和十进制。前缀和后缀表示了所用数字的进制。

M32C / M16C / R8C 和 740 调试器

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

* 当预定的进制是 16 进制时，只能指定 %

如果正在输入的数和指定的进制一样，就可以省略符号（二进制除外）。

使用 RADIX 命令来设置预定的进制。但是，如下的情况，进制被固定，不能用 RADIX 指令更改。

类型	进制
地址	Hex
行号/执行序号/通过中断次数的序号	Dec

7.2.2 符号和标号

在用户的目标程序中可以加入符号和标号，或汇编语言中定义的符号和标号。

- 用户可以在标号中使用字母/数字/下划线（_），点（.），和问号（?），但是，首字母不能是数字。
- 符号和标号可以使用最多 255 个字符。
- 大写和小写字符均可。

产品名称	注意
M32R 调试器， M32C 调试器， M16C/R8C 调试器	不能包含汇编结构指令，伪指令，宏指令，操作代码或者保留字（.SECTION，.BYTE，switch，if，等等）。 不能使用开头为两个点的字符串(..)，作为符号或者标号

(1) 局部标号和范围

本调试器支持可以在整个程序中使用的全局标号，和只能在声明的文件中使用的局部标号。局部标号的有效范围叫做“scope”，以 obj 目标文件作为单位。下列情况时，范围有效：

- 当一条指令执行到 PC 指针当前指向的 obj 文件中时，这个文件就叫做“scope”，当使用 SCOPE 命令指定一个“scope”时，所指定的“scope”就是活动的“scope”。
- 命令执行过程中，当前的 scope 会自动进行切换

(2) 标号和符号的优先级

无论是把数值转换为标号或符号，或者反之，都受以下优先级限制：

- 地址转换

1、局部标号

2、全局标号

3、局部符号

4、全局符号

5、scope 外的全局标号

6、scope 外的全局符号

- 数值转换

1、局部符号

2、全局符号

3、局部标号

4、全局标号

5、scope 外的局部标号

6、scope 外的局部符号

- 位转换

1、局部位符号

2、全局位符号

3、scope 外的局部位符号

7.2.3 宏变量

在脚本文件中的赋值语句中可以定义宏变量。详情请参考“7.1.2 赋值语句”使用宏变量时请加入前缀“%”

- 用户可以在变量前缀“%”后使用字母/数字/下划线(_)，但是，首字母不能是数字。
- 不能使用寄存器名称来当作变量名称
- 变量名称中区分大小写
- 最多可以定义 32 个宏变量，一旦定义，直到调试器退出，变量都有效

宏变量可以用于在迭代的 while 语句中设定循环次数

7.2.4 寄存器变量

寄存器变量用于在表达式中使用寄存器。使用时需加入前缀“%”，按如下格式使用：（740 系列的调试器可以使用“_”替代“%”）。

产品名称	寄存器名称
M32C 调试器	PC，USP，ISP，INTB，FLB，SVF，SVP，VCT，DMD0,DMD1，DCT0，DCT1，DRC0，DRC1，DMA0,DMA1，DCA0，DCA1，DRA0，DRA1，0R0，0R1，0R2，0R3，0A0，0A1，0FB，0SB <- Bank 0 寄存器 1R0，1R1，1R2，1R3，1A0，1A1，1FB，1SB <- Bank 1 寄存器
M16C/R8C 调试器	PC，USP，ISP，SB，INTB，FLG 0R0，0R1，0R2，0R3，0A0，0A1，0FB <- Bank 0 寄存器 1R0，1R1，1R2，1R3，1A0，1A1，1FB <- Bank 1 寄存器

变量名称不区分大小写，两者都可以使用。

7.2.5 存储器变量

存储器变量用于在表达式中使用存储器中的数值。格式如下：

[Address].data-size

- 可以在表达式中指定地址（也可以指定存储器变量）。
- 所指定的数据大小由下表所示（740 系列调试器不支持 4 字节 长度）。

数据长度	调试器	代码
1 字节	所有	B 或 b
2 字节	M32R 调试器	H 或 h
	其它	W 或 w
4 字节	M32R 调试器	W 或 w
	M32R，M16C/R8C 调试器	L 或 l

例：调用存储器地址 8000h，以 2 字节为单位

[0x8000].W

- 如果不指定的话，默认的数据长度为2字节

7.2.6 行号

源文件的行号，格式如下：

#line_no

#line_no."source file name"

- 以十进制指定行号，
- 只能指定可以设置软件断点的行号，不能指定没有生成汇编指令的行，包括注释行和空行。
- 如果用户省略了源文件名称，那么指定的行号就对应着当前打开的 Editor（Source）窗口中的文件。
- 源文件的名称中包含文件属性。
- 行号和源文件名称中不能夹入空格。

7.2.7 字符常量

指定的字符或字符串会被转换为 ASCII 编码，然后按照常量处理。

- 用单引号将字符括起
- 用双引号将字符串括起
- 字符串必须由 1 或者 2 个字符构成（最长 16 位）。如果字符串长度大于 2 个字符，只有最后两个字符有效。举例来说，“ABCD”将会按照“CD”处理，即数值 4344h。

7.2.8 操作符

下表列出了表达式中可以使用的操作符

- 操作符的级别取决于优先级，1 级最高，8 级最低。如果两个操作符同级，左边的先被处理。

操作符	功能	优先级
( )	括号	1 级
+，-，~	正/负/逻辑“非”	2 级
*，/	带符号乘法，除法	3 级
+，-	加法，减法	4 级
<<，>>	左移，右移	5 级
&	逻辑“与”	6 级
|，^	逻辑“或”，逻辑“异或”	7 级
<，<=，>，>=，==，!=	比较	8 级

8. C/C++表达式

8.1 编写 C/C++表达式

用户可以使用由以下标识组成的 C/C++语句来注册 C watchpoints 和指定将要赋给 C watchpoints 的数值

令牌	例子
立即数	10，0x0a，012，1.12，1.0E+3
Scope（范围）	::name，classname::member
数学运算	+，-，*，/
指针	*，**，...
取址	&
符号取反	-
使用点操作符参考成员	Object.Member
使用箭头操作符参考成员	Pointer->Member，this->Mmber
成员指针	eObct.*var，Pointer->*var
圆括号	je(，)
数组	Array[2]，DArray[2] [3] ，...
指定基本类型	(int)，(char*)，(unsigned long *)，...
指定 typedef 类型	(DWORD)，(ENUM)，...
变量和函数名称	var，i，j，func，...
字符常量	'A'，'b'，...
字符串	“abcdef”，“I am a boy."，…

8.1.1 立即数

用户可以使用十六进制/十进制/八进制作为立即数。前缀为 0x 的数字被认为是十六进制。前缀为“O”的数字为八进制，没有前缀的数字为 10 进制。浮点数也可以用来给变量赋值。

注意

不能把立即数注册为 C watchpoints。

当使用 C/C++表达式来指明 C/C++ watchpoints 时，或者给这些表达式赋值时，立即数有效。当使用浮点数时，不能使用像“1.0+2.0”这样的表达式。

8.1.2 Scope 拆分

scope 拆分操作符 :: 如下所示：

全局 scope： ::变量名

::x, ::val

类 scope： 类名称::成员名称,，类名称::类名称::成员名称，等等

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

8.1.3 数学运算

用户可以使用加法（+），减法（-），乘法（*），和除法（/）这些数学运算。以下为运算时的优先级：

（*），（/），（+），（-）

注意

- 现在数学运算不支持浮点数

8.1.4 指针

星号（*）代表指针运算符。用户也可以使用双重指针 **，或者三重指针 ***，等等。

例子：“*variable_name”，“**variable_name”，等等。

注意

立即数不能进行指针操作。换句话说，用户不能使用诸如“*0xE000”这样的操作

8.1.5 取址

“与”符号（&）代表取址运算符。只能使用“&variable_name”此种用法。

8.1.6 符号取反

负号（-）代表符号取反操作。只能够使用“- 立即数”或者“- 变量名”。如果使用了两个（或者偶数个）取反符号的时候，相当于没有取反。

注意

- 现在符号取反运算不支持浮点数

8.1.7 使用点操作符指向成员变量

只能使用“变量名.成员名”的格式来用点操作符检查结构体和联合体的成员变量。

举例来说：

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

在此例中，t_cls.member1,（*pt_cls).member2 正确地指向了成员变量

8.1.8 使用箭头操作符指向成员变量

只能使用“变量名 -> 成员名”的格式来用点操作符检查结构体和联合体的成员变量。

举例来说：

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

在此例中，(&t_cls)->member1, pt_cls->member2 正确地指向了成员变量

8.1.9 指向成员变量的指针

使用“.*”或“->*”操作符的指向成员变量的指针可以使用：“变量名 .* 成员变量名”或者“变量名 ->* 成员变量名”这两种格式。

举例来说：

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

在此例中，t_cls.*mp and tp_cls->*mp 正确地指向了指针成员变量。

注意

- 表达式 *mp 不是指向成员变量的指针

8.1.10 圆括号

使用“(”和“)”来指定一个表达式中的运算优先级

8.1.11 数组

用户可以使用“[”和“]”来指定数组的元素。可以用如下方式声明数组：“变量名 [（元素数量或变量）]” ，“变量名 [（元素数量或变量）] [（元素数量或变量）]”，等等。

8.1.12 指定基本类型

用户可以使用 C 中的基本类型，char，short，int 和 long，并且可以为这些基本类型指定指针。当指定指针变量时，用户也可以使用二重或三重指针变量等等。注意，如果没有指定符号，默认的符号如下：

基本类型	默认符号
char	unsigned
short	signed
int	signed
long	signed

注意

- C++的基本类型中，bool 类型，wchar_t 类型，浮点类型（float 或 double）不能使用
- 不能为寄存器变量指定类型

8.1.13 指定 typedef 类型

用户可以指定 typedef 类型（C 基本类型以外的类型）和指针类型。当指定指针类型时，用户也可以使用二重或三重指针变量等等。

注意

- 不能给结构体和联合体指定指针类型。

8.1.14 变量名称

变量名可以使用英文字母开头，遵循 C/C++习惯。

变量名的最大长度是 255 个字符

并且“this”指针有效

8.1.15 函数名

函数名可以使用英文字母开头，遵循 C/C++习惯。

注意

- C++状态下不支持函数名

8.1.16 字符常量

单引号（’）中包含的内容可以作为字符常量。举例来说：’A’，’b’等等。这些字符被转换为 ASCII 码，并且被当作 1-字节立即数使用。

注意

- 不能将字符常量注册为 C watchpoints
- 字符常量只在用 C/C++表达式声明 C watchpoints 并赋值时有效（字符串常量与立即数等效）

8.1.17 字符串

用户可以使用两个双引号（"）中的字符作为字符串。举例来说： "abcde"，"I am a boy."，等等。

注意

- 字符串只能放在赋值语句的右侧，只能用于等号左边是字符串或者字符指针的场合。否则的话会造成语法错误。

8.2 C/C++表达式显示格式

C Watch 窗口的数据显示区中的 C/C++表达式以它们的类型名显示，C/C++表达式（变量名称），和计算结果（值），下文说明了每种不同类型的显示方式：

8.2.1 枚举类型

- 当运算结果（值）被定义的时候，名称会被显示：
  (DATE) date = Sunday（所有进制）_
- 当运算结果（值）未被定义的时候，会有如下显示：
  (DATE) date = 16（当进制是初始值）

  (DATE) date = 0x10（当进制是 16 进制）

  (DATE) date = 0000000000010000B（当进制是 2 进制）

8.2.2 基本类型

- 当计算的结果是除了浮点和字符类型以外的其他类型时，如下显示：
  (unsigned int) i = 65280（当进制是初始值）

  (unsigned int) i = 0xFF00（当进制是 16 进制）

  (unsigned int) i = 1111111100000000B（当进制是 2 进制）
- 当计算的结果是字符类型时，如下显示：
  (unsigned char) c = 'J'（当进制是初始值）

  (unsigned char) c = 0x4A（当进制是 16 进制）

  (unsigned char) c = 10100100B（当进制是 2 进制）
- 当计算的结果是浮点类型时，如下显示：
  (double) d = 8.207880399131839E-304（当进制是初始值）

  (double) d = 0x10203045060708（当进制是 16 进制）

  (double) d = 0000000010.....1000B（当进制是 2 进制）

  (..... 标志缩写)

8.2.3 指针类型

- 当计算的结果是除了 char* 以外的类型时，以下列十六进制的格式显示：
  (unsigned int *) p = 0x1234（所有进制）
- 当计算的结果是 char*的类型时，用户可以在 C Watch 窗口的菜单[Display String]中选择是按照字符串类型显示还是按照字符类型显示。
  - 字符串类型

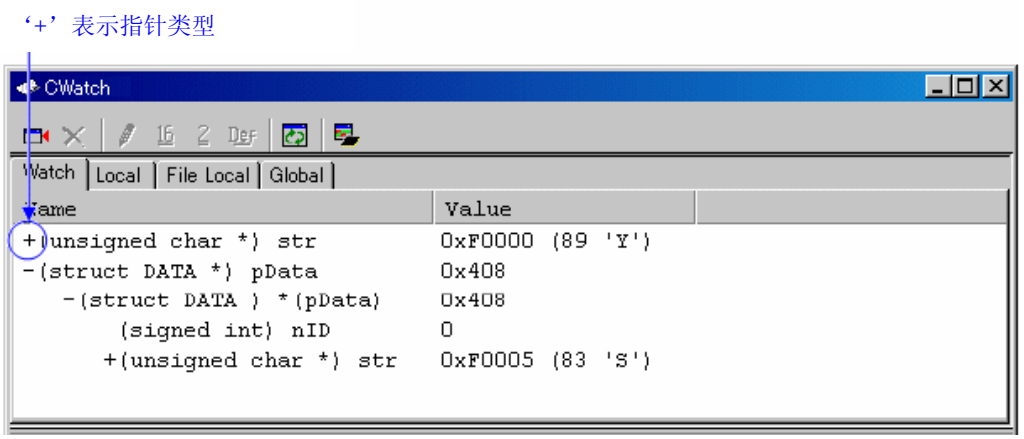
    (unsigned char *) str = 0x1234 "Japan"（所有进制）
  - 字符类型

    (unsigned char *) str = 0x1234 (74 'J')（所有进制）

当计算的结果是 char*的类型时，以下列格式显示：

(unsigned char *) str = 0x1234 "Jap（所有进制）

如果字符串包含有不可打印的字符，则只显示到不可打印字符的前一个字符，并且结尾的双引号不会显示。



用户可以双击带有“+”的行来查看结构体或联合体的成员。当成员被显示的时候，“+”号会变为“-”号。想返回原始的显示状态，双击已经变为“-”的行。

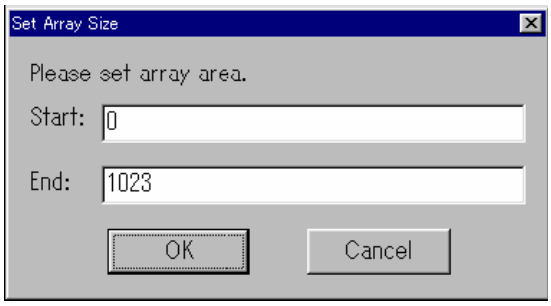
8.2.4 数组类型

- 当计算的结果是除了 char [ ] 类型以外的数组类型时，起始地址会以 16 进制显示：
  (signed int [10]) z = 0x1234（所有进制）
- 当计算的结果是 char [ ] 类型时，会按如下显示：
  (unsigned char [10]) str = 0x1234 "Japan"（所有进制）

如果字符串包含有不可打印的字符，则只显示到不可打印字符的前一个字符，并且结尾的双引号不会显示。

(unsigned char [10]) str = 0x1234 "Jap（所有进制）

同样的，如果一个字符串包含有 80 个以上的字符，结尾的双引号不会显示。当 C/C++表达式是数组类型或者指针类型时，类型名称左边会显示“+”。用户可以用这种方法来查看数组的元素（具体请参考“8.2.3 指针类型”）。当数组中的元素超过 100 个时，会显示以下对话框，指定元素的数量。



由“Start:”和“End:”指定的区间内的元素将会被显示。如果指定的值超过了数组的界限，这个值将会被当作数组的最大界限。当单击“Cancel”按钮时，元素不会被显示。

8.2.5 函数类型

- 当运算结果是函数类型时，起始地址会以 16 进制显示：
  (void()) main = 0xF000（所有进制）

8.2.6 取址类型

- 当运算结果是取址类型时，取得的地址会以 16 进制显示：
  (signed int &) ref = 0xD038（所有进制）

8.2.7 位区域类型

- 当运算结果是位区域类型时，按照如下格式显示：
  (unsigned int :13) s.f = 8191（当进制是初始值）

  (unsigned int :13) s.f = 0x1FFF（当进制是 16 进制）

  (unsigned int :13) s.f = 1111111111111B（当进制是 2 进制）

8.2.8 当未找到 C 符号时

- 当运算表达式包含了无法被找到的 C 符号时，会有如下显示：

() x = <not active>（所有进制）

8.2.9 语法错误

- 当运算表达式有语法错误时，会有如下显示：

() str*(p = <syntax error>（所有进制）

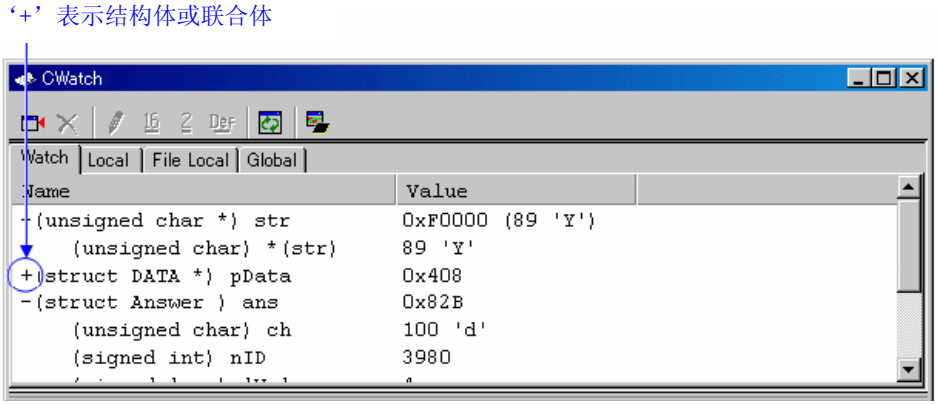
（str*(p 处有语法错误）

8.2.10 结构体和联合体类型

- 当运算结果是结构体和联合体类型时，起始地址会以 16 进制显示：

(Data) v = 0x1234 （所有进制）

如果 C/C++表达式中，像结构体或者联合体一样由成员组成，“+”号会显示在名称（tag 名称）的左边。



用户可以双击带有“+”的行来查看结构体或联合体的成员。当成员被显示的时候，“+”号会变为“-”号。想返回原始的显示状态，双击已经变为“-”的行。本功能允许用户检查结构体或联合体的成员。

注意

如果一个变量的名称和 typedef 中定义的类型名一样时，就不能引用这个变量。

- 寄存器变量

当运算结果是寄存器变量时，“register”会显示在类型名称的左边。

(register signed int) j = 100

9. 显示程序停止的原因

如果程序被调试功能停止，停止的原因会显示在 Output 窗口或者 Status 窗口中（[Platform]页）。显示的内容和“停止的原因”如下所述：

Halt	点击了[Halt Program] 按钮或菜单
S/W break	软件中断
Address match interrupt break	地址一致中断
H/W event，Combination	硬件中断，满足逻辑‘与’（同时）的条件
H/W event，Combination，Ax	硬件中断，满足逻辑‘或’的条件（Ax: 满足条件的次数）
H/W event，State transition，from xx	硬件中断，满足状态转移条件（from xx: 上一个状态（start，state1，state2））
H/W event，State transition，Timeout	硬件中断，状态转移，满足超时条件
H/W event，Access protect error	保护中断

注意

显示的中断的原因根据所连接的目标的不同而不同，有的目标系统只能显示“Halt”或者显示“---”。

10. 注意事项

10.1 一般注意事项

10.1.1 Windows 系统中的文件操作

应该注意以下几点：

1、文件名和子目录名

- 不要使用带空格的文件名或子目录名
- 当文件名或子目录名中包含汉字的时候不能保证正确操作
- 文件名中只能使用一个“.”符号

2、指定文件和目录

- 不可以使用“...”来指定向上 2 级的子目录
- 不可以使用网络路径，必须先把远程路径挂载到本地的网络驱动器上

10.1.2 可以设置软件断点的区域

内部的 RAM 和 ROM 区域可以设置软件断点。

10.1.3 读取或设置 C 变量

- 如果一个变量的名称和 typedef 中定义的类型名一样时，就不能引用这个变量。
- 寄存器变量和 bit fields 变量的值不能改变。
- 64 位宽度的变量的值不能改变（long long，double 等等）。
- 如果在 C/C++表达式中没有指明存储器地址和大小的话，变量的值不能改变
- 有时出于优化的需要，C 编译器会将不同的变量放置在同一个地址。在这种情况下，C 变量的值有可能显示不正确。
- 字符串只能被字符数组或者字符指针变量替换
- 不能对浮点变量进行数学运算
- 不能对浮点变量进行符号取反运算
- 不能对浮点变量指定类型
- 不能对寄存器变量指定类型
- 不能对结构体/联合体/指向结构体或联合体的指针指定类型
- 字符常量和字符串不能包含 ESC

10.1.4 C++中的函数名

- 当用户在设置显示地址时使用变量名或设置断点等等时，不能使用某一类中的成员函数/操作符函数和重载函数。
- C/C++ 表达式中不能使用函数名
- 当函数的名称被指定为参数时，可以使用脚本命令（例，breakin 和 func）
- 当在对话框中指定地址时，不能使用函数名来指定地址。

10.1.5 多个模块调试

如果在一个段中定义了两个或以上的绝对地址模块，同一时间只能有一个文件被下载。如果在一个段中注册了一个绝对地址模块和一个或多个机器码文件，用户可以同时下载所有文件。

10.1.6 同步调试

目前暂无同步调试功能

10.1.7 RAM 监控功能

RAM 监控功能通过存储器 dump 来实现。当使用此功能时，会降低实时性。而且，根据访问属性设置不同颜色的功能无法使用。

10.1.8 逐行汇编功能

可以为逐行汇编功能指定从 00000H 到 FFFFFH 的 RAM 空间

10.1.9 不支持的调试功能

本调试器不支持硬件断点、保护、覆盖、跟踪、时间测量等其它仿真器支持的功能。

10.1.10 软件中断功能

- 软件中断功能使用 MCU 的地址一致中断功能。因此，能用的断点数量取决于所使用的 MCU。
- 如果双击 Source 窗口的 PASS 一列，可以设置经过某一断点的次数。

10.2 M32C 调试器注意事项

10.2.1 C 编译器/汇编程序和连接器的选项

信息有可能不能正常下载/调试，这取决于 C 编译器/汇编程序和连接器的设置。请参考以下内容进行设置：

请参考“10.4 编译器、汇编程序和连接器的选项”进行设置

可以和M32C调试器配合使用的编译器：

- NCxx
- the IAR EC++ Compiler
- the IAR C Compiler

10.3 M16C/R8C 调试器注意事项

10.3.1 编译器、汇编程序和连接器的选项

信息有可能不能正常下载/调试，取决于 C 编译器/汇编程序和连接器的设置。请参考以下内容进行设置：

参考“10.4 编译器、汇编程序和连接器的选项”

可以和M16C/R8C调试器配合使用的编译器：

- NCxx
- the IAR EC++ Compiler
- the IAR C Compiler
- the TASKING C Compiler

10.3.2 TASKING C 编译器

当用户调试经过 TASKING C 编译器“CCM16”编译的程序时， bit field 的类型会固定为“unsigned short int”。因为 CCM16 会将 bit field 类型的调试信息以“unsigned short int”的格式进行输出。

10.3.3 R8C/Tiny 系列的注意事项

- 如果在 FFFFH 的第 0 位写入了“0”，导致无法使用 R8C/12-17 群的 MCU，请用 flash 写入器擦除整个 Flash 区域。
- 如果目标 MCU 是 R8C/12-17 群的 MCU，不能调试使用了看门狗定时器的程序。

10.4 编译器、汇编程序和连接器的选项

由于没有评估其他设置，所以不推荐附加选择其他选项。

10.4.1 当使用 NCxx 时

当编译时选择了 -O，-OR 或 -OS 选项时，源代码的信息可能因为优化而不能正确生成，导致不能正常单步执行。为避免这个问题，在选择了-O，-OR 或者-OS 选项时，请同时选择 -ONBSD（或者 -Ono_Break_source_debug）选项。

由于未测试以上所提及的选项以外的选项。因此不推荐选择以上提及的选项以外的选项。

10.4.2 当使用 IAR C 编译器（EW）时

请遵循以下步骤指定项目的参数

(1) IAR Embedded Workbench 的设置

当用户从菜单中选择 [Project] -> [Options...]，“Options For Target”的对话框将会打开。在这个对话框中，请选择“XLINK”，并且设置项目参数。

- Output 选项卡

在“Format”区域中，选择“Other”选项，并且将“ieee-695”设置为“Output Format”。

- Include 选项卡

在“XCL File Name”区域中，选择用户的 XCL 文件（例如: lnkm16c.xcl）。

(2) 编辑 XCL 文件

加入命令行参数“-y”到用户的 XCL 文件中，根据产品的不同指定“-y”选项。

产品名称	-y 选项
M32C 调试器	-ylmb
M16C/R8C 调试器	-ylmb

(3) 在经过上述设置后重新生成程序

未测试上面所提到的选项以外的其它选项。所以请注意，以上提及的选项以外的选项不推荐选择。

10.4.3 当使用 IAR C 编译器（ICC）时

(1) 指定选项

请按照以下过程编译和连接程序

- 编译时

指定“-r”选项

- 连接前

打开连接选项定义文件（扩展名 .xcl）并且加入“-FIEEE695”和“-y”选项，根据产品的不同指定“-y”选项。

产品名称	-y 选项
M32C 调试器	-ylmb
M16C/R8C 调试器	-ylmb

- 连接时

使用“-f”选项来选中连接选项定义文件

未测试上面所提到的选项以外的其它选项。所以请注意，以上提及的选项以外的选项不推荐选择。

(2) 命令行示例

以下为不同产品的命令行示例

- M32C 调试器

  >ICCMC80 -r file1.c<回车>

  >ICCMC80 -r file2.c<回车>

  >XLINK -o filename.695 -f lnkm80.xcl file1 file2<回车>

- M16C/R8C 调试器

  >ICCM16C -r file1.c<回车>

  >ICCM16C -r file2.c<回车>

  >XLINK -o filename.695 -f lnkm16c.xcl file1 file2<回车>

XCL 文件名根据所选型号的不同而不同。详情请参阅 ICCxxxx 手册

10.4.4 当使用 TASKING C 编译器（EDE）时

请遵循以下步骤指定项目的设置

1、从菜单中选择 - [EDE]->[C Compiler Option]->[Project Options...],“M16C C Compiler Options [Project Name]”对话框将会打开，请按如下设置

- Optimeze 选项卡

请在优化设置中选择“No optimization”

- Debug 选项卡

请只选择“Enable generation of any debug information(including type checkeing)”和“Genarate symbolic debug information”。

2、从菜单中选择- [EDE]->[Linker/Locator Options...]，“M16C Linker/Locator Options [Project Name]”对话框将会打开，请按如下设置

- Format 选项卡

请指定“IEEE 695 for debuggers(abs)”为输出格式

3、在经过上述设置后重新生成程序

未测试上面所提到的选项以外的其它选项。所以请注意，以上提及的选项以外的选项不推荐选择。

10.4.5 当使用 TASKING C 编译器（CM）时

(1) 指定选项

请在编译时指定“-g”和“- O0”选项。

未测试上面所提到的选项以外的其它选项。所以请注意，以上提及的选项以外的选项不推荐选择。

(2) 命令行示例

以下为命令行示例

>CM16 -g -O0 file1.c<回车>

10.4.6 当使用 IAR EC++ 编译器（EW）时

请遵循以下步骤指定项目的设置

(1) IAR Embedded Workbench 中的设置

从菜单中选择 [Project] -> [Options...]，“Options For Target”对话框将会打开。在这个对话框中，请选择“XLINK”，并且设定项目设置。

- Output 选项卡

在“Format”区域里，选中“Other”选项，然后选择“elf/dwarf”作为“Output Format”。

- Include 选项卡

在“XCL File Name”区域，指定用户的 XCL 文件（例：lnkm32cf.xcl）。

(2) 加入命令行参数“-y”到用户的 XCL 文件中，根据产品的不同指定“-y”选项。

产品名称	-y 选项
M32C 调试器	-yspc
M16C/R8C 调试器	-yspc

(3) 在经过上述设置后重新生成程序

未测试上面所提到的选项以外的其它选项。所以请注意，以上提及的选项以外的选项不推荐选择。

[备注]

M16C R8C FoUSB/UART 软件
用户手册

Publication Date: Rev.1.00 Oct. 27, 2006

Published by: Renesas Technology Corp.

Edited by: MCU Application Engineering Dept.

Renesas Solutions Corp.

M16C R8C
FOUSB/UART

RENESAS

瑞萨电子株式会社

RCC10J0005-0100