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

RENESAS

User's Manual

M16C R8C 小型仿真调试器 V.1.03

瑞萨单片机开发环境系统

瑞萨电子
www.renesas.com

Rev.1.00　2008.01

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。

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，使用它可以轻松地开发和调试使用 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、检查和修改调试运行应用程序的调试平台。

本帮助手册将介绍 HEW 的 “调试器”功能。

目标系统

此调试器在小型仿真器系统上运行。

支持的 CPU

本帮助手册将介绍与以下 CPU 相对应的调试功能。

- M32C/80、 M16C/80 系列
  注意：在本帮助手册中，与此系列 CPU 有关的信息被形容为 “适用于 M32C”。
- M16C/Tiny、 R8C/Tiny 系列
  注意：在本帮助手册中，与此系列 CPU 有关的信息被形容为 “适用于 M16C/R8C”。

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。

如果要就此文档或此产品内容进行咨询，请填写安装程序在以下目录中生成的文本文件，然后通过电子邮件发送给当地的经销商。

\SUPPORT\Product-name\SUPPORT.TXT

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

目　录

第 1 部分　设置调试器 1

1　功能 3
1.1　RAM 监控功能 3
1.1.1　RAM 监控区 3
1.1.2　采样周期 3
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　硬件断点功能 5
1.2.3　地址中断断点功能 5
1.2.3.1　设置和删除断点 5
1.3　实时跟踪功能 6
1.3.1　跟踪区域 6
1.3.2　跟踪条件设置 7
1.3.3　跟踪数据写条件 7
1.4　实时操作系统调试功能 7
1.5　GUI 输入 / 输出功能 7

2　关于小型仿真器 8
2.1　通信方法 8
2.2　功能表 8

3　启动调试器之前 9
3.1　仿真器使用的通信方法 9
3.1.1　USB 接口 9
3.2　下载固件 9
3.3　仿真器启动前的设置 10
3.3.1　USB 通信 10
3.3.1.1　安装 USB 设备驱动程序 10

4　使用前的准备工作 11
4.1　工作空间、工程和文件 11
4.2　启动 HEW 12
4.2.1　创建新的工作空间 （使用工具链） 13
4.2.1.1　步骤 1：创建新的工作空间 13
4.2.1.2　步骤 2：设置工具链 14
4.2.1.3　步骤 3：选择目标平台 15
4.2.1.4　步骤 4：设置配置文件名 16
4.2.1.5　步骤 5：检查创建的文件名 17
4.2.2　创建新的工作空间 （不使用工具链） 18
4.2.2.1　步骤 1：创建新的工作空间 18
4.2.2.2　步骤 2：选择目标平台 19
4.2.2.3　步骤 3：设置配置文件名 20
4.2.2.4　步骤 4：注册要下载的加载模块 21
4.3　启动调试器 23
4.3.1　连接仿真器 23
4.3.2　终止仿真器 23

5 设置调试器 24
5.1 [Init]（初始化）对话框 24
5.1.1 MCU 选项卡 25
5.1.1.1 指定 MCU 文件 25
5.1.1.2 设置通信接口 25
5.1.1.3 执行自检 26
5.1.1.4 使用 / 不使用地址中断断点功能 26
5.1.1.5 使用 / 不使用看门狗定时器 27
5.1.1.6 选择使用或不使用 CPU 重写模式 27
5.1.1.7 选择是否使用跟踪点设置功能 27
5.1.2 [Debugging Information]（调试信息）选项卡 28
5.1.2.1 显示使用的编译器及其目标格式 28
5.1.2.2 指定如何存储调试信息 28
5.1.2.3 指定是否要显示指令格式说明符 29
5.1.2.4 将枚举类型的大小视为 1 个字节 29
5.1.3 [Emulator]（仿真器）选项卡 30
5.1.3.1 指定目标时钟 30
5.1.3.2 尝试在处于等待 / 停止模式期间存取存储器 30
5.1.4 [Script]（脚本）选项卡 31
5.1.4.1 自动执行脚本命令 31
5.2 设置通信接口 31
5.2.1 设置 USB 接口 31
5.3 设置适用于 M32C 的调试器 32
5.3.1 [Emem] 对话框 32
5.3.1.1 [Status]（状态）选项卡 33
5.3.1.2 [Emulation Memory]（仿真存储器）选项卡 34
5.3.1.3 [Flash Clear]（闪存清除）选项卡 37
5.4 设置适用于 M16C/R8C 的调试器 38
5.4.1 MCU 设置对话框 38
5.4.1.1 MCU 选项卡 39
5.4.1.2 [MAP] 选项卡 41
5.4.1.3 [Flash Clear]（闪存清除）选项卡 42

第 2 部分 教程 43

6 教程 45
6.1 简介 45
6.2 使用 45
6.2.1 步骤 1：启动调试器 45
6.2.1.1 使用前的准备工作 45
6.2.1.2 设置调试器 45
6.2.2 步骤 2：检查 RAM 的运行情况 46
6.2.2.1 检查 RAM 的运行情况 46
6.2.3 步骤 3：下载教程程序 47
6.2.3.1 下载教程程序 47
6.2.3.2 显示源程序 48
6.2.4 步骤 4：设置断点 49
6.2.4.1 设置软件断点 49

6.2.5 步骤 5：执行程序 50
6.2.5.1 复位 CPU 50
6.2.5.2 执行程序 50
6.2.5.3 查看中断原因 51
6.2.6 步骤 6：查看断点 51
6.2.6.1 查看断点 51
6.2.7 步骤 7：查看寄存器 52
6.2.7.1 查看寄存器 52
6.2.7.2 设置寄存器值 52
6.2.8 步骤 8：查看存储器 53
6.2.8.1 查看存储器 53
6.2.9 步骤 9：监视变量 54
6.2.9.1 监视变量 54
6.2.9.2 注册变量 55
6.2.10 步骤 10：逐步执行程序 56
6.2.10.1 执行 [Step In] （跳入）命令 56
6.2.10.2 执行 [Step Out] （跳出）命令 57
6.2.10.3 执行 [Step Over] （跳过）命令 58
6.2.11 步骤 11：强制中断程序执行 58
6.2.11.1 强制中断程序执行 58
6.2.12 步骤 12：显示局部变量 59
6.2.12.1 显示局部变量 59
6.2.13 步骤 13：堆栈跟踪功能 60
6.2.13.1 引用函数调用状态 60
6.2.14 后续内容 60

第 3 部分 参考 61

7 窗口 / 对话框 63
7.1 [RAM Monitor] （RAM 监控）窗口 64
7.1.1 扩展菜单 66
7.1.2 设置 RAM 监控区域 67
7.1.2.1 更改 RAM 监控区域 68
7.1.2.2 添加 RAM 监控区域 68
7.1.2.3 删除 RAM 监控区域 68
7.2 [ASM Watch] （ASM 监视）窗口 69
7.2.1 扩展菜单 70
7.3 [C Watch] （C 监视）窗口 71
7.3.1 扩展菜单 73
7.4 [Script] （脚本）窗口 74
7.4.1 扩展菜单 75
7.5 [S/W Break Point Setting] （软件断点设置）窗口 76
7.5.1 命令按钮 77
7.5.2 从 [Editor (Source)] （编辑器 （源））窗口设置和删除断点 78
7.6 [H/W Break Point Setting] （硬件断点设置）窗口 79
7.6.1 指定断点事件 80
7.6.2 指定组合条件 82
7.6.3 命令按钮 82

7.6.4 指定事件 （取指令） 83
7.6.4.1 从指定地址取指令 83
7.6.4.2 从指定地址范围内取指令 84
7.6.4.3 从指定地址范围以外取指令 85
7.6.4.4 进入 / 退出指定函数 86
7.6.5 指定事件 （存储器存取） 87
7.6.5.1 存储器存取 （用于 M32C 的调试器） 87
7.6.5.2 存储器存取 （用于 M16C/R8C 的调试器） 97
7.6.6 指定事件 （位存取） 106
7.6.6.1 写入 / 读取指定位 106
7.6.6.2 写入 / 读取指定位符号 107
7.6.7 指定事件组合条件 108
7.6.7.1 选择 [AND] （与）、 [OR] （或） 108
7.6.7.2 选择 [AND (Same Time)] （与 （同时）） 109
7.7 [Address Interrupt Break Point Setting] （地址中断断点设置）窗口 109
7.7.1 命令按钮 110
7.7.1.1 从 [Editor (Source)] （编辑器 （源））窗口设置和删除断点 111
7.8 [Trace Point Setting] （跟踪点设置）窗口 112
7.8.1 指定跟踪事件 113
7.8.2 指定组合条件 115
7.8.3 指定跟踪范围 116
7.8.4 指定跟踪写条件 116
7.8.5 命令按钮 117
7.8.6 指定事件 （取指令） 117
7.8.7 指定事件 （存储器存取） 117
7.8.8 指定事件 （位存取） 117
7.8.9 指定事件 （中断） 117
7.8.10 指定写条件 118
7.9 [Trace] （跟踪）窗口 121
7.9.1 配置 “Bus Mode”（总线模式） 121
7.9.2 配置 “Disassemble Mode”（反汇编模式） 123
7.9.3 配置 “Data Access Mode”（数据存取模式） 124
7.9.4 配置 “Source Mode”（源模式） 125
7.9.5 扩展菜单 126
7.9.6 显示 M32C 调试器的总线信息 127
7.9.7 显示 M16C/R8C 调试器的总线信息 129
7.10 [Data Trace] （数据跟踪）窗口 130
7.10.1 扩展菜单 131
7.11 [GUI I/O] 窗口 132
7.11.1 扩展菜单 133
7.12 [MR] 窗口 134
7.12.1 扩展菜单 135
7.12.2 显示任务状态 136
7.12.2.1 显示任务状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 136
7.12.2.2 显示任务状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 138
7.12.3 显示就绪队列状态 140
7.12.3.1 显示就绪队列状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 140
7.12.3.2 显示就绪队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 140
7.12.4 显示超时队列状态 141
7.12.4.1 显示超时队列状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 141
7.12.4.2 显示超时队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 142

7.12.5 显示事件标志状态 143
7.12.5.1 显示事件标志状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 143
7.12.5.2 显示事件标志状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 144
7.12.6 显示信标状态 145
7.12.6.1 显示信标状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 145
7.12.6.2 显示信标状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 146
7.12.7 显示邮箱状态 147
7.12.7.1 显示邮箱状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 147
7.12.7.2 显示邮箱状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 148
7.12.8 显示数据队列状态 149
7.12.8.1 显示数据队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 149
7.12.9 显示周期处理程序状态 150
7.12.9.1 显示周期处理程序状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 151
7.12.9.2 显示周期处理程序状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 151
7.12.10 显示警报处理程序状态 152
7.12.10.1 显示警报处理程序状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 152
7.12.10.2 显示警报处理程序状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 152
7.12.11 显示存储池状态 153
7.12.11.1 显示存储池状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。） 153
7.12.11.2 显示存储池状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。） 154
7.12.12 显示任务上下文 155
7.12.12.1 显示任务上下文 155
7.12.12.2 更改任务上下文 156

8 脚本命令表 157
8.1 脚本命令表 （按功能分类） 157
8.1.1 执行命令 157
8.1.2 文件操作命令 157
8.1.3 寄存器操作命令 158
8.1.4 存储器操作命令 158
8.1.5 汇编 / 反汇编命令 158
8.1.6 软件断点设置命令 159
8.1.7 地址中断断点设置命令 159
8.1.8 硬件断点设置命令 159
8.1.9 实时跟踪命令 159
8.1.10 脚本 / 日志文件命令 160
8.1.11 程序显示命令 160
8.1.12 映像命令 160
8.1.13 时钟命令 160
8.1.14 C 语言调试命令 161
8.1.15 实时操作系统命令 161
8.1.16 实用程序命令 161
8.2 脚本命令表 （依字母顺序） 161

9 编写脚本文件 164
9.1 脚本文件的结构元素 164
9.1.1 脚本命令 165
9.1.2 赋值语句 165
9.1.3 条件语句 165
9.1.4 循环语句（while 及 endw）和中断语句 165
9.1.5 注释语句 166
9.2 编写表达式 166
9.2.1 常数 167
9.2.2 符号和标签 167
9.2.2.1 局部标签符号和作用域 167
9.2.2.2 标签和符号的优先级 168
9.2.3 宏变量 168
9.2.4 寄存器变量 169
9.2.5 存储器变量 169
9.2.6 行号 169
9.2.7 字符常数 170
9.2.8 运算符 170

10 C/C++ 表达式 171
10.1 编写 C/C++ 表达式 171
10.1.1 立即值 171
10.1.2 作用域解析 172
10.1.3 数学运算符 172
10.1.4 指针 172
10.1.5 引用 172
10.1.6 取反 172
10.1.7 使用点运算符的成员引用 173
10.1.8 使用箭头运算符的成员引用 173
10.1.9 指向成员的指针 173
10.1.10 括号 174
10.1.11 数组 174
10.1.12 基本数据类型转换 174
10.1.13 typedef 类型转换 174
10.1.14 变量名 174
10.1.15 函数名称 175
10.1.16 字符常数 175
10.1.17 字符串文字 175
10.2 C/C++ 表达式的显示格式 176
10.2.1 枚举类型 176
10.2.2 基本数据类型 176
10.2.3 指针类型 177
10.2.4 数组类型 178
10.2.5 函数类型 178
10.2.6 引用类型 178
10.2.7 位域类型 178
10.2.8 未找到任何 C 符号时 179
10.2.9 语法错误 179
10.2.10 结构类型和联合类型 179

11 显示程序停止的原因 180

12 注意事项 181
12.1 常见注意事项 181
12.1.1 与 Windows 相关的文件操作 181
12.1.2 可设置软件断点的区域 181
12.1.2.1 适用于 M32C 的调试器 181
12.1.2.2 适用于 M16C/R8C 的调试器 182
12.1.3 获取或设置 C 变量 182
12.1.4 C++ 中的函数名 183
12.1.5 下载模块的选项设置 183
12.1.6 对多个模块进行调试 183
12.1.7 同步调试 183
12.1.8 小型仿真器复位开关 183
12.2 M32C 调试器的注意事项 184
12.2.1 仿真器使用的堆栈区 184
12.2.2 复位目标程序时的中断堆栈指针 184
12.2.3 C 编译器 / 汇编器 / 连接器的选项 184
12.2.4 目标 MCU HOLD 终端 184
12.2.5 硬件事件 185
12.2.6 CPU 重写 185
12.3 M16C/R8C 调试器注意事项 186
12.3.1 小型仿真器使用的堆栈区映射 186
12.3.2 复位目标程序时的中断堆栈指针 186
12.3.3 编译器、汇编器和连接器的选项 186
12.3.4 TASKING C 编译器 186
12.3.5 目标 MCU HOLD 终端 186
12.3.6 硬件断点功能 186
12.3.7 硬件事件 187
12.3.8 存储器空间扩展 187
12.3.9 CPU 重写 188
12.4 编译器、汇编器和连接器的选项 189
12.4.1 使用 NCxx 时 189
12.4.2 使用 IAR C 编译器 (EW) 时 189
12.4.3 使用 IAR EC++ 编译器 (EW) 时 190
12.4.4 使用 IAR C 编译器 (ICC) 时 191
12.4.4.1 指定选项 191
12.4.4.2 命令执行示例 191
12.4.5 使用 TASKING C 编译器 (EDE) 时 192
12.4.6 使用 TASKING C 编译器 (CM) 时 192
12.4.6.1 指定选项 192
12.4.6.2 命令执行示例 192

设置调试器

1 功能

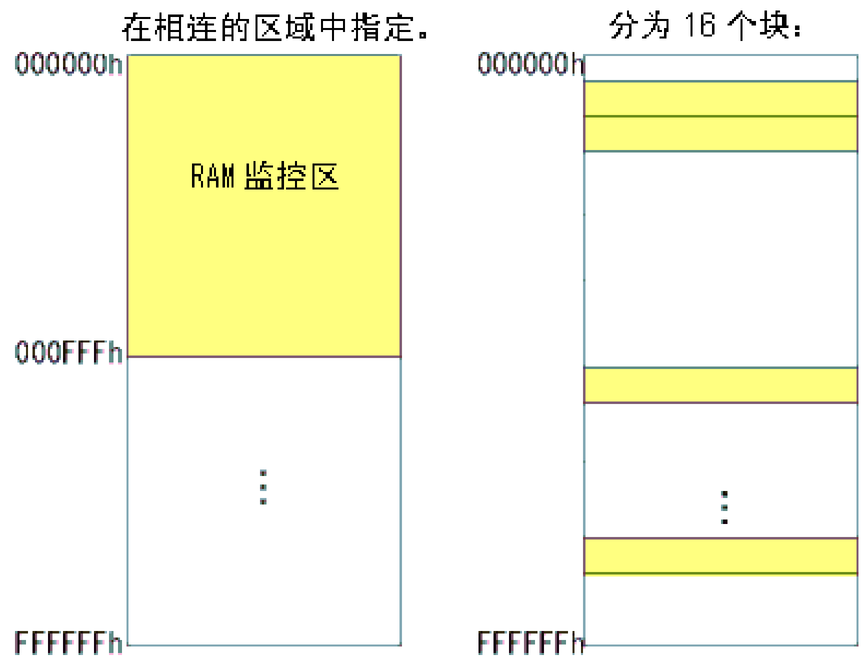
1.1 RAM 监控功能

使用此功能可查看存储器内容的更改，而不影响目标程序的实时执行功能。

小型仿真器系统有一个 1 KB 大小的 RAM 监控区，这个监控区可位于任何连续地址位置上，也可位于 4 个单独的块 （每个块的大小为 256 字节）中。

1.1.1 RAM 监控区

此调试器有一个 1 KB 大小的 RAM 监控区，这个监控区可位于任何连续地址位置上，也可位于 4 个单独的块 （每个块的大小为 256 字节）中。



1.1.2 采样周期

采样周期表示显示更新的间隔。

可以在支持 RAM 监控的任何窗口中指定此功能。（默认情况下设置的间隔为 100 ms。）

实际采样周期可能比指定的周期更长，这取决于操作环境。（采样周期与以下环境有关。）

- 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 监控区内的 ASM 监视点数
- [C Watch] （C 监视）窗口的 RAM 监控区内的 C 监视点数

1.1.3 相关窗口

在下面的窗口中可以使用实时 RAM 监控功能。

- [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

软件断点会在执行指定地址的命令之前中断目标程序。这种断点称作软件断点。

在 [Editor (Source)] （编辑器 （源））窗口或 [S/W Breakpoint Setting] （软件断点设置）窗口中可以设置 / 复位软件断点。还可以临时禁止或允许软件断点。

最多可以指定 64 个软件断点。指定两个或更多个软件断点时，断点组合建立在逻辑或的基础上。（即到达任何一个断点时都会中断目标程序。）

1.2.1.1 设置软件断点

在以下窗口中可以设置软件断点。

- [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。

还可以转到 [S/W Break Point Setting] （软件断点设置）窗口中临时禁止 / 允许软件断点。

1.2.1.2 可设置软件断点的区域

可以设置软件断点的区域因产品而异。

有关可设置软件断点的区域，请参阅以下章节：

“12.1.2 可设置软件断点的区域”

1.2.2 硬件断点功能

此功能会导致目标程序在检测到某些情况时停止，这些情况包括存储器数据读 / 写、指令执行或从外部跟踪电缆所供给的输入信号的上升沿 / 下降沿。

可设置事件的内容在各个目标 MCU 中是不同的。

可以对断点事件使用以下指定方法：

- 地址指定
  - — 取指令
  - — 存储器存取
  - — 位存取
- 外部触发器指定
- 中断

可以指定的事件数为上述所有事件中的 2 个。对于地址指定方法，除普通的单地址指定以外，取指令和存储器存取允许范围指定和逻辑条件指定。此外，取指令还允许指定函数名称。

存储器存取允许指定比较数据，采用与设置硬件断点时一样的方式，对与指定地址相关的读 / 写数据进行比较。还允许对比较数据使用屏蔽指定。

这些断点事件可以按如下方式进行组合：

- 当所有有效事件都建立时进行跟踪 （[AND] （与）条件）
- 当所有有效事件同时建立时进行跟踪 （同时的 [AND] （与）条件）
- 当其中一个有效事件建立时进行跟踪 （[OR] （或）条件）

1.2.3 地址中断断点功能

在指定地址执行指令之前，此功能会立即停止目标程序。此功能是通过使用 MCU 的地址匹配中断实现的。

只有在用户应用程序中未使用地址匹配中断时，才能使用地址中断断点功能。断点个数取决于连接的 MCU。

注意

只有选中 [Init] （初始化）对话框中 MCU 选项卡上的 [Enable the Address Match Interrupt Break Function] （允许地址匹配中断断点功能）复选框时，才能使用地址中断断点功能。（详细信息）。

1.2.3.1 设置和删除断点

在以下窗口中可以设置地址中断断点。

- [Editor (Source)] （编辑器 （源））窗口
- [Address Interrupt Break Point Setting] （地址中断断点设置）窗口

可以在 [Editor (Source)] （编辑器 （源））窗口中双击鼠标来设置 / 复位地址中断断点 （这与软件断点相同）。

还可以转到 [Address Interrupt Break Point Setting] （地址中断断点设置）窗口中临时禁止 / 允许地址中断断点。

1.3　实时跟踪功能

使用实时跟踪功能可记录目标程序的执行历史记录。

最多可以记录 64K 个周期的执行历史记录。使用这个记录可以查看每个周期的总线信息、执行的指令和源程序执行路径。

在跟踪窗口中会引用执行历史记录。

可以采用以下模式引用执行历史记录。

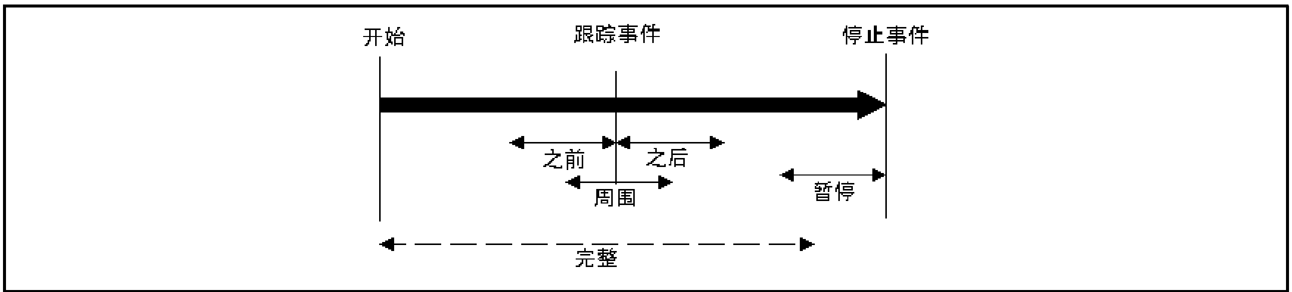
- “BUS mode”（总线模式）
  此模式允许按周期查看总线信息。显示内容取决于使用的 MCU 和仿真器系统。除总线信息外，此模式还允许以组合形式显示反汇编、源行或数据存取信息。
- “Disassemble mode”（反汇编模式）
  此模式允许查看执行的指令。除反汇编信息外，此模式还允许以组合形式显示源行或数据存取信息。
- “Data access mode”（数据存取模式）
  此模式允许查看数据读/写周期。除数据存取信息外，此模式还允许以组合形式显示源行信息。
- “Source mode”（源模式）
  此模式允许查看源程序中的程序执行路径。

1.3.1　跟踪区域

使用此调试器可引用 64K 个周期的执行历史记录。

支持下列 5 种模式的跟踪区域。

- “Break”（暂停）
  目标程序停止之前的 64K 个周期
- “Before”（之前）
  跟踪点之前的 64K 个周期
- “About”（周围）
  跟踪点两侧各 32K 个周期
- “After”（之后）
  跟踪点之后的 64K个周期
- “Full”（完整）
  直到 64K 个周期都写入跟踪存储器时



“Break”（暂停）是默认设置。要引用停止目标程序之前的执行历史记录，请使用 “Break”（暂停）（不需要指定跟踪事件）。

要引用任何位置的执行历史记录，或者要继续执行目标程序，请指定跟踪事件并更改跟踪范围。

1.3.2 跟踪条件设置

以下指定操作可视为跟踪事件：

- 地址指定
  - 取指令
  - 存储器存取
  - 位存取

可以指定的事件数为上述所有事件中的 2 个。这些断点事件可以按如下方式进行组合：

- 当所有有效事件都建立时进行跟踪 （[AND] （与）条件）
- 当所有有效事件同时建立时进行跟踪 （[And (same time)] （与 （同时））条件）
- 当其中一个有效事件建立时进行跟踪 （[OR] （或）条件）

1.3.3 跟踪数据写条件

可以指定跟踪数据写条件。
可以指定以下写条件：

- 无限制的写条件 （默认设置）
- 从建立开始事件到建立结束事件的周期
- 仅建立开始事件的周期
- 从建立开始事件到取消建立开始事件的周期
- 从建立开始事件到建立结束事件的周期之外
- 建立开始事件的周期之外
- 从建立开始事件到取消建立开始事件的周期之外

1.4 实时操作系统调试功能

使用此功能可以对使用实时操作系统的目标程序中与实时操作系统相关的部分进行调试。
此功能可用来显示实时操作系统的状态。

1.5 GUI 输入 / 输出功能

此功能可在窗口上模拟用户目标系统的键盘输入面板 （按钮）和输出面板。
按钮可用来代表输入面板，标签 （字符串）和 LED 可用来代表输出面板。

2　关于小型仿真器

小型仿真器虽然价格便宜、体积又小，但是具备能满足各种规模的开发所需要的调试功能，如实时跟踪和硬件断点等功能。

2.1　通信方法

支持的通信方法如下。

I/F	仿真器
	小型仿真器
USB	支持

2.2　功能表

支持的功能如下。

功能	小型仿真器
软件断点	64 点
硬件断点	2 点
地址中断断点	4 点 *
实时跟踪	64K 个周期
RAM 监控	1K 字节 （256 字节 x 4 块）区域
时间计量	运行到停止

* 取决于使用的目标 MCU。

3 启动调试器之前

3.1 仿真器使用的通信方法

支持的通信方法如下。

- USB

3.1.1 USB 接口

- 符合 USB 标准 1.1。
- 不支持通过 USB 集线器建立的连接。
- 使用 USB 电缆将仿真器连接到主机后，就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。

3.2 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。

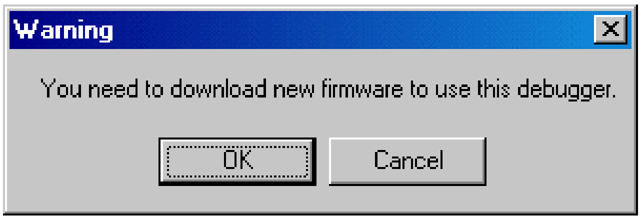
- 首次使用时需要先设置调试器。
- 需要升级仿真调试器。
- 下载到仿真器的固件是未知固件。

在接通小型仿真器电源后两秒钟之内，按系统复位开关，可以建立维护模式。

此调试器在启动时会搜索下载的仿真器固件版本。此外，当下载的仿真器固件为旧版本时，就会建立一种模式来强制此调试器下载固件。

如果仿真器设置为强制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）按钮可下载固件。



3.3 仿真器启动前的设置

3.3.1 USB 通信

Windows 的即插即用功能会检测到连接的 USB 设备。此时会自动安装连接 USB 设备所需的驱动程序。有关详细信息，请参阅 “安装 USB 设备驱动程序”。

3.3.1.1 安装 USB 设备驱动程序

Windows 的即插即用功能会检测连接的 USB 设备。检测到该设备后， USB 设备驱动程序安装向导就会启动。下面说明安装 USB 设备驱动程序的过程。

1. 使用 USB 电缆将仿真器连接到主机。
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. 此时会显示下面的对话框。

按照向导的指示继续操作，就会显示一个要求指定设置信息文件 （inf 文件）的对话框。请指定 musbdrv.inf 文件，该文件位于安装此调试器的目录之下。

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。请先安装此调试器。
- 安装 USB 设备驱动程序的用户需要有管理员权限。
- 在安装过程中，会显示一条信息，指出找不到特定的设备驱动程序 musbdrv.sys。在这种情况下，请指定 musbdrv.sys，该文件与 musbdrv.inf 文件放在同一目录下。

4　使用前的准备工作

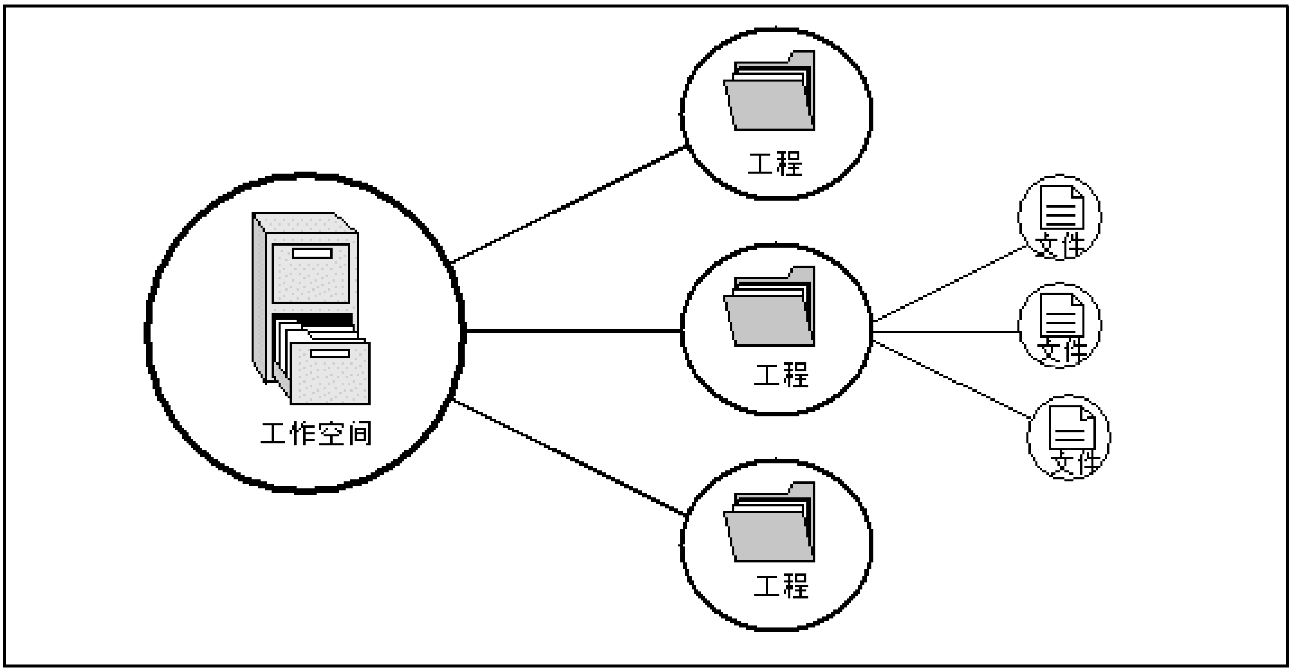
请运行 HEW 并连接仿真器。

此外，为使用此产品进行调试，需要创建一个工作空间。

4.1　工作空间、工程和文件

就像使用文字处理软件可创建和修改文档一样，使用此产品可创建和修改工作空间。

工作空间可视为工程的贮存箱，同样，工程可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，而每个工程可包含一个或多个文件。



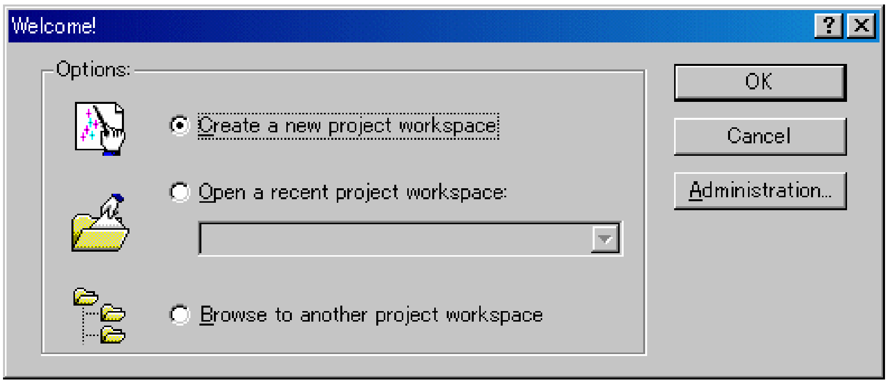
通过工作空间可以将相关的工程组合在一起。例如，可能需要针对不同的处理器创建应用程序，或者可能要同时开发一个应用程序和一个库。在工作空间中，还可以按照层次结构将工程联系起来，这意味着创建一个工程时会首先创建它的所有“子”工程。

但是，工作空间本身并没有什么作用。需要在工作空间中添加工程，然后再在工程中添加文件，才能实际执行各种操作。

4.2 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。



可以在此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；
  在未保留打开工作空间的历史记录时可使用此单选按钮。

在选择现有工作空间时，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，然后选择工作空间文件 (.hws)。

有关创建新工作空间的方法，请参阅以下章节。

参阅 “4.2.1 创建新的工作空间 （使用工具链）”。

参阅 “4.2.2 创建新的工作空间 （不使用工具链）”。

* 当使用此产品对现有加载模块文件进行调试时，此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。在安装了适用于所用 CPU 的 C/C++ 编译器套件的环境中，提供了要使用的工具链。

有关详细信息，请参阅 C/C++ 编译器套件附带的手册。

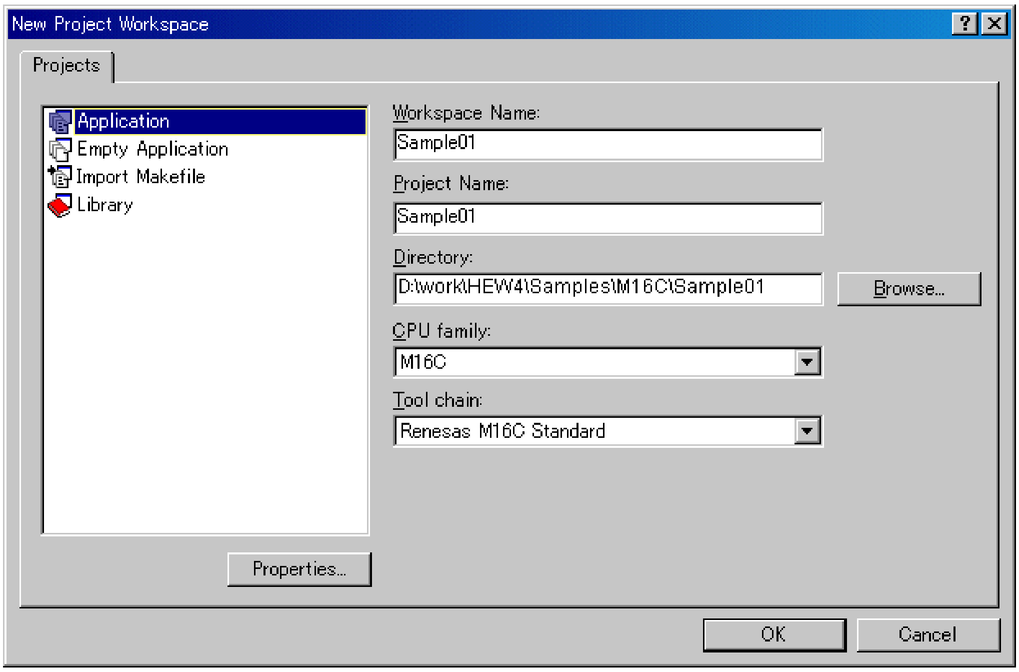
4.2.1 创建新的工作空间（使用工具链）

4.2.1.1 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，然后单击 [OK]（确定）按钮。

此时会开始创建新工作空间。

同时显示以下对话框。



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。

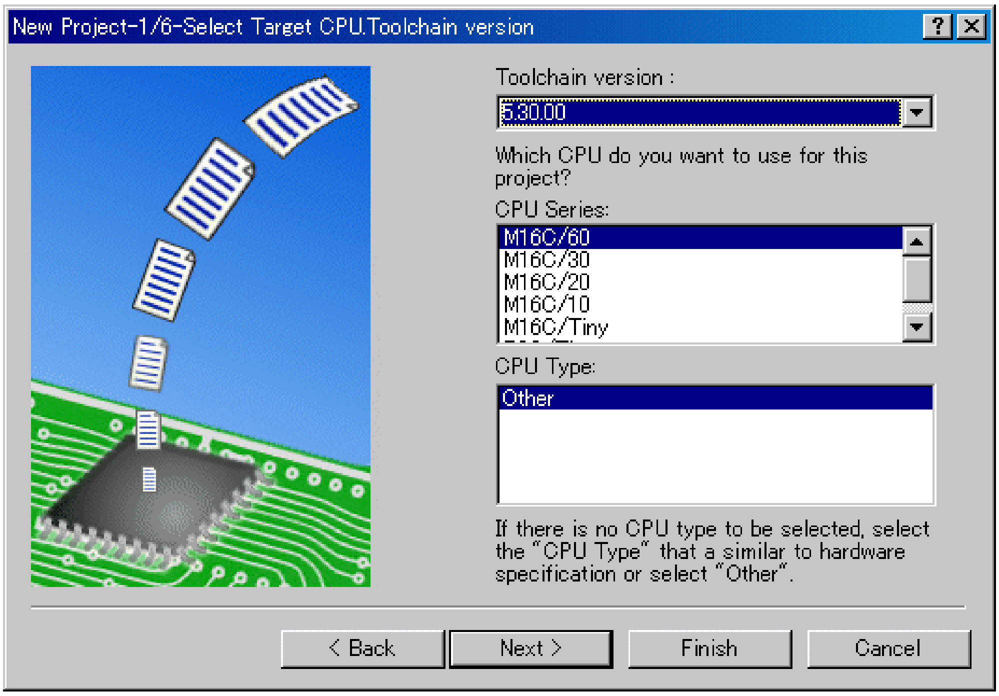
   — 在 [Project Name]（工程名称）编辑框中，输入工程名称。如果工程名称与工作空间名称相同，则无需输入。

   — 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…）按钮可选择目录。

进行设置以后，单击 [OK]（确定）按钮。

4.2.1.2 步骤 2：设置工具链

此时会启动工程创建向导。



在这里可设置以下内容。

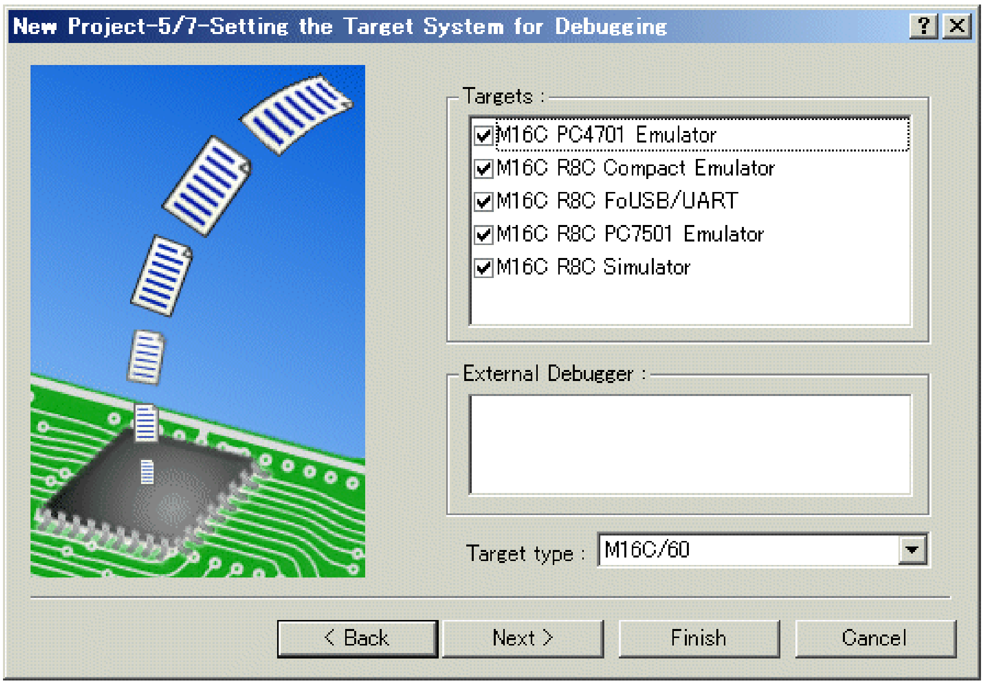
- 工具链
- 实时操作系统的设置 （如果要使用）
- 启动文件、堆区、堆栈区等等的设置

请设置所需信息并单击 [Next] （下一步）按钮。

设置内容会根据所用 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。

4.2.1.3 步骤 3：选择目标平台

选择进行调试的目标系统 （仿真器、模拟器）。
完成工具链的设置后会显示以下对话框。

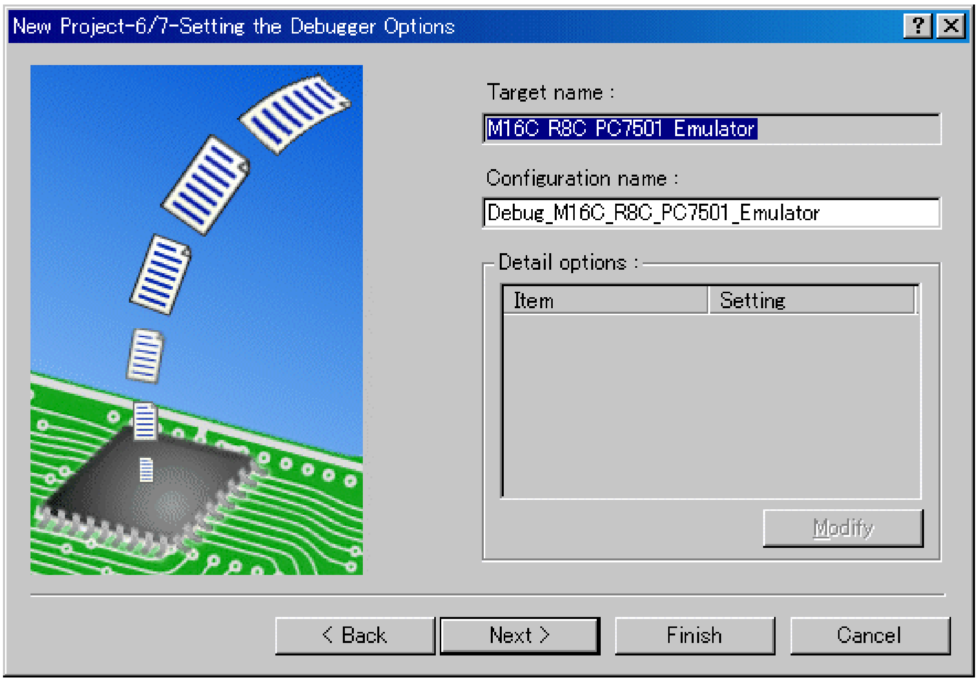


1. 选择目标类型
   在 [Target type] （目标类型）列表框中，选择目标 CPU 类型。
2. 选择目标平台
   在 [Targets] （目标）区域中，必须选择激活此调试器时所用会话文件的目标。
   选中目标平台所对应的框。（根据需要选择其它目标。）

然后单击 [Next] （下一步）按钮。

4.2.1.4　　步骤 4：设置配置文件名

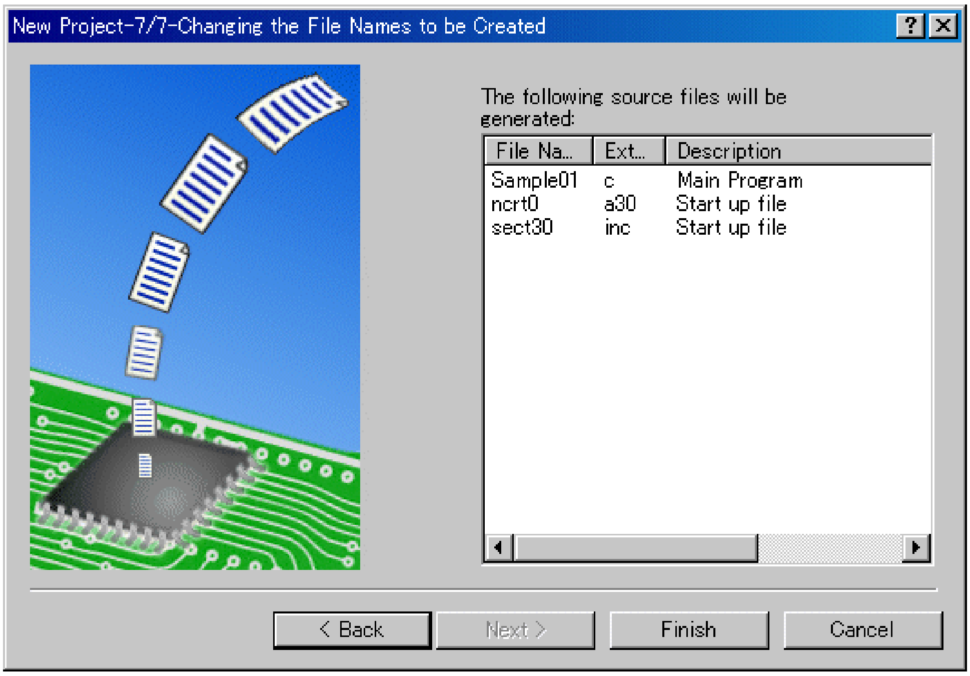
为选择的所有目标逐个设置配置文件名。
配置文件会保存 HEW 的状态 （目标仿真器、模拟器除外）。



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

4.2.1.5 步骤 5：检查创建的文件名

最后，确认所创建的文件名。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。



仿真器设置到此已完成。

按照屏幕指示退出工程生成器 (Project Generator)。

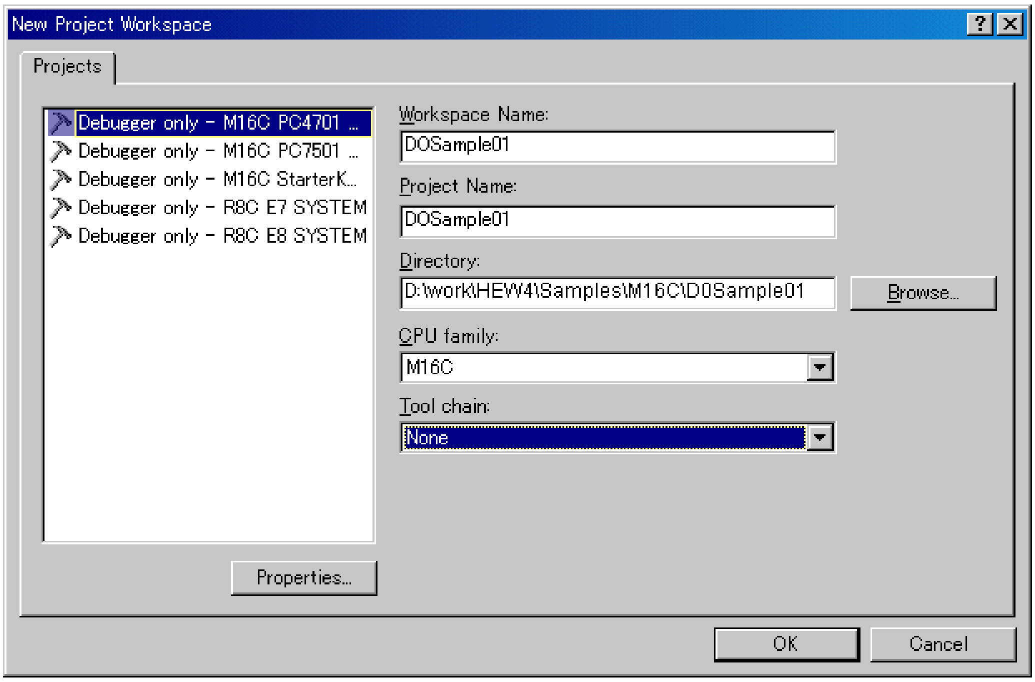
4.2.2 创建新的工作空间 （不使用工具链）

当使用此产品对现有加载模块文件进行调试时，此方法会创建一个工作空间。（即使未安装工具链也可以。）

4.2.2.1 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，然后单击 [OK] （确定）按钮。

此时会开始创建新工作空间。同时显示以下对话框。



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，在这个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，在 [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。（如果显示两个或多个工程类型，请选择其中一个。）

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。

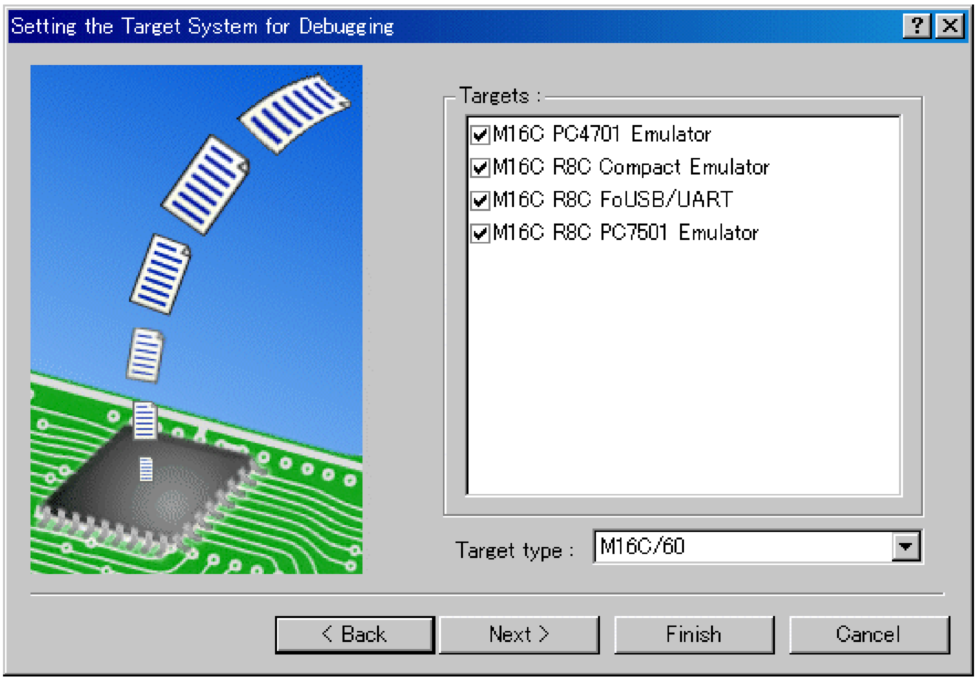
   — 在 [Project Name] （工程名称）编辑框中，输入工程名称。如果工程名称与工作空间名称相同，则无需输入。

   — 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…）按钮可选择目录。

进行设置以后，单击 [OK] （确定）按钮。

4.2.2.2　　步骤 2：选择目标平台

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



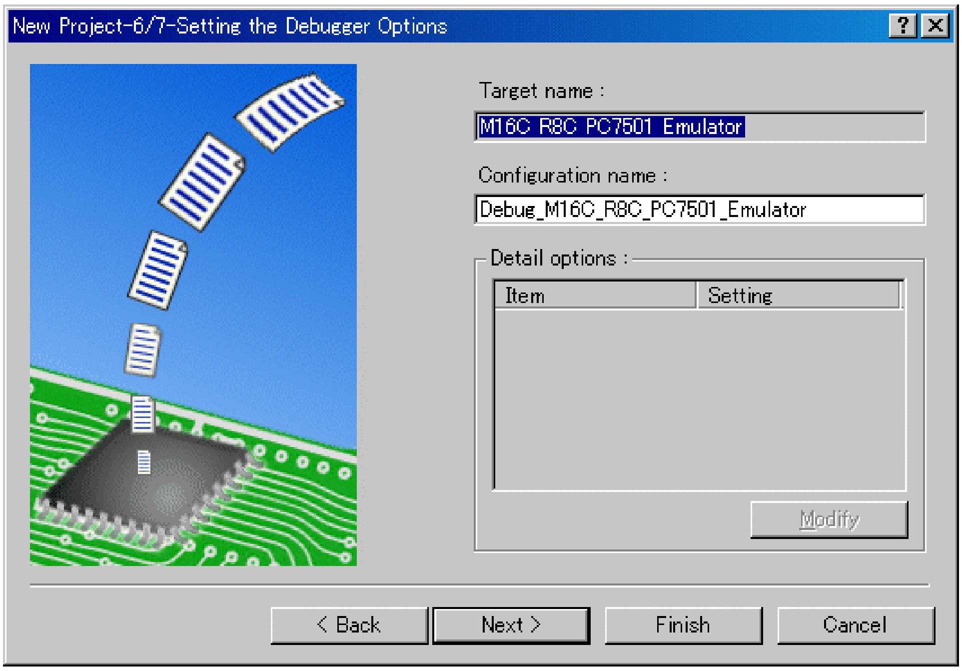
1. 选择目标类型
   在 [Target type] （目标类型）列表框中，选择目标 CPU 类型。
2. 选择目标平台
   在 [Targets] （目标）区域中，必须选择激活此调试器时所用会话文件的目标。
   选中目标平台所对应的框。（根据需要选择其它目标。）

然后单击 [Next] （下一步）按钮。

4.2.2.3 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。

配置文件会保存 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，这样会使用默认名称。

仿真器设置到此已完成。

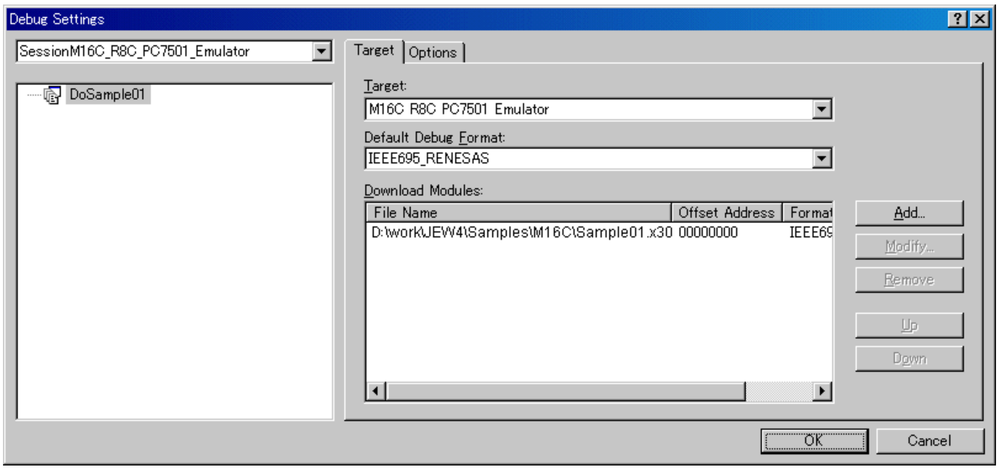
按照屏幕指示退出工程生成器 (Project Generator)。

此时会再次显示设置调试器的对话框。如果完成了仿真器的准备工作，请在此对话框中设置调试器并连接仿真器。

4.2.2.4 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…）打开 [Debug Settings] （调试设置）对话框。

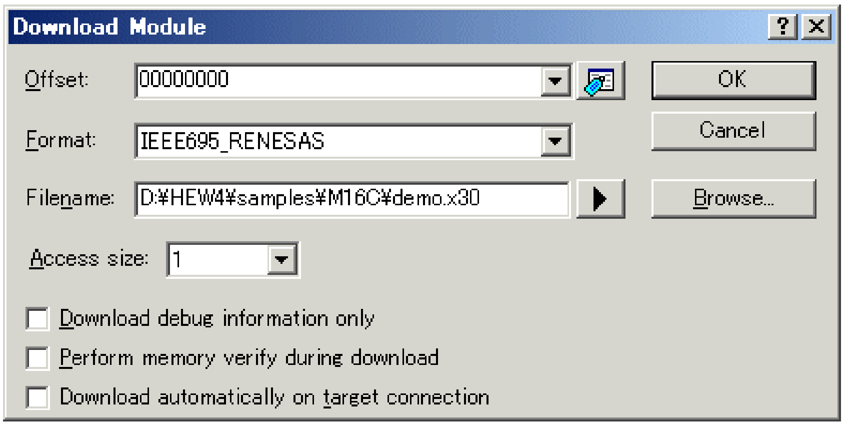


1. 在 [Target] （目标）下拉式列表框中选择要连接的产品名称。
2. 在 [Default Debug Format] （默认调试格式）下拉式列表框中，选择要下载的加载模块的格式。

格式名称	内容
IEEE695_RENESAS	IEEE-695 格式文件 （使用瑞萨工具链时）
IEEE695_IAR	IEEE-695 格式文件 （使用 IAR 工具链时）
IEEE695_TASKING	IEEE-695 格式文件 （使用 Tasking 工具链时）
ELF/DWARF2	ELF/DWARF2 格式文件 （使用瑞萨工具链时）
ELF/DWARF2_IAR	ELF/DWARF2 格式文件 （使用 IAR 工具链时）
ELF/DWARF2_TASKING	ELF/DWARF2 格式文件 （使用 Tasking 工具链时）
ELF/DWARF2_KPIT	ELF/DWARF2 格式文件 （使用 KPIT 工具链时）

此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 [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

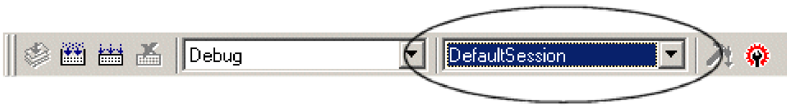
忽略 “Offset”（偏移量）、“Access size”（存取大小）和 “Perform memory verify during download”（下载期间执行存储器检验）。偏移量总是设置为 0，存取大小总是设置为 1，而检验不起作用。

4.3 启动调试器

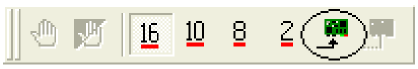
通过与仿真器相连接，可以启动调试器。

4.3.1 连接仿真器

只需将会话文件切换为其中注册了仿真器使用设置的文件，即可连接仿真器。
默认情况下已创建会话文件。会话文件包含关于创建工程时所选目标的信息。
在下面工具栏上被圈中的列表框中，选择包括要连接目标的字符串的会话名称。



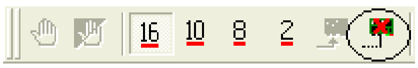
选择会话名称之后，就会显示用于设置调试器的对话框，并且将连接仿真器。
如果未显示该对话框，请从 [Debug] （调试）菜单中选择 [Connect] （连接）。



4.3.2 终止仿真器

可以使用以下方法退出仿真器：

1. 选择 “Disconnect”（断开连接）
   从 [Debug] （调试）菜单中选择 [Disconnect] （断开连接）。



2. 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）。

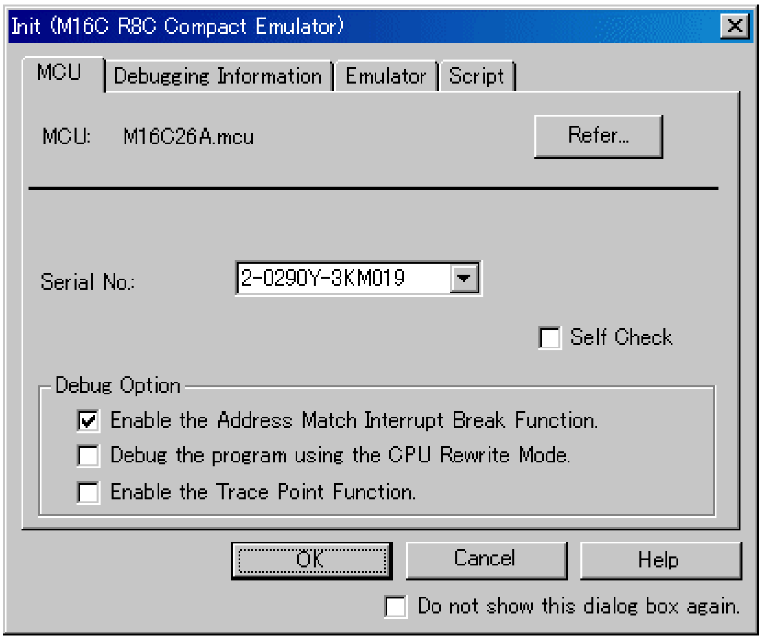
3. 退出 HEW
   从 [File] （文件）菜单中选择 [Exit] （退出）。此时会终止 HEW。

退出仿真器时，将会显示一个信息框，询问是否需要保存会话。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。

5 设置调试器

5.1 [Init]（初始化）对话框

在 [Init]（初始化）对话框中可设置调试器启动时所需设置的项目。在此对话框中设置的内容在调试器下次启动时还会有效。在此对话框中设置的数据在下次启动时继续有效。



此对话框上提供的选项卡因所用产品而异。有关详细信息，请单击下表中显示的所需选项卡名称。

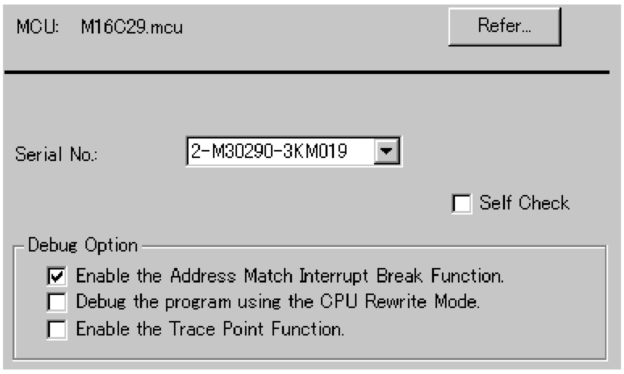
选项卡名称	产品名称	
	适用于 M32C 的调试器	适用于 M16C/R8C 的调试器
MCU	存在	存在
Debugging Information（调试信息）	存在	存在
Emulator（仿真器）	存在	存在
Script（脚本）	存在	存在

可以使用下列任一种方法打开 [Init]（初始化）对话框：

- 启动调试器以后，选择菜单 - [Setup]（设置）→ [Emulator]（仿真器）→ [System...]（系统…）。
- 按住 Ctrl 键的同时启动调试器。

5.1.1 MCU 选项卡

指定内容在下次启动时生效。



5.1.1.1 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。

5.1.1.2 设置通信接口

显示的数据因指定的通信接口而异。

可用的通信接口因产品而异。

下面显示各种通信接口的设置。

参阅 “5.2.1 设置 USB 接口”

5.1.1.3 执行自检

指定此选项可在调试器启动时在仿真器上执行自检 *。

只有在启动时要执行自检，才有必要选中上面的复选框。请在出现以下情况时指定此选项：

- 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时，要检查仿真器运行是否正常

选中该复选框可关闭 [Init] （初始化）对话框。连接到仿真器并确认固件之后，调试器会立即在仿真器上启动自检。（自检大约花费 30 秒到 1 分钟时间。）

如果在这个自检过程中发现错误，调试器会显示错误内容，然后结束。

自检正常结束时，会显示下面的对话框。单击 [OK] （确定）后，调试器会直接在该状态下启动。

此设置只在调试器启动时才会生效。

* 所谓自检功能指的是检查仿真器内部电路板的存储器状况等。有关自检功能的详细信息，请参阅仿真器用户手册。

5.1.1.4 使用 / 不使用地址中断断点功能

指定是否要使用地址中断断点功能。

- 使用地址中断断点功能 （默认）
  选中上面显示的复选框。
  在这种情况下，地址中断断点功能由仿真器使用，而不能在用户程序中使用。
- 不使用地址中断断点功能
  取消选中上面显示的复选框。
  在这种情况下，地址中断断点功能可在用户程序中使用。

这里设置的内容只在启动时才会反映出来。

5.1.1.5　使用 / 不使用看门狗定时器

指定是否使用看门狗定时器。（默认情况下，不使用看门狗定时器。）
此设置选择只在 M32C 调试器中才存在。

在对使用看门狗定时器的目标系统进行调试时，请选中上面显示的复选框。

5.1.1.6　选择使用或不使用 CPU 重写模式

指定是否要使用 “CPU rewrite mode”（CPU 重写模式）。（默认情况下，不使用 “CPU rewrite mode”（CPU 重写模式）。）

当对使用 “CPU rewrite mode”（CPU 重写模式）的目标系统进行调试时，请选中上面的复选框。
此设置只在启动调试器时才可设置或更改。

补充说明

如果允许在 “CPU rewrite mode”（CPU 重写模式）下进行调试，则以下限制适用：

- 不能设置地址匹配断点。
- 不能在内部 ROM 区中设置软件断点。
- 不能在内部 ROM 区中执行 COME 命令。

5.1.1.7　选择是否使用跟踪点设置功能

指定是否要使用跟踪点设置功能。（默认情况下，不使用跟踪点功能。）

当使用小型仿真器的事件作为跟踪点时，请选中上面的复选框。

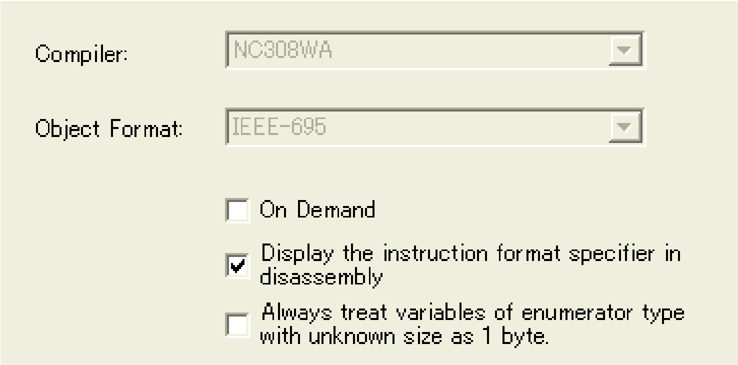
补充说明

启用跟踪点设置功能时，以下限制适用：

- 不能使用硬件断点功能。

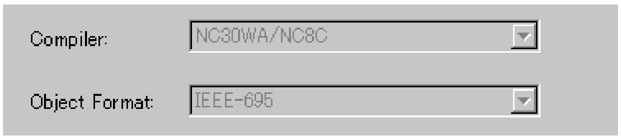
5.1.2　[Debugging Information] （调试信息）选项卡

指定内容在下次启动时生效。



5.1.2.1　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。

5.1.2.2　指定如何存储调试信息

存储调试信息的方法有两种：on-memory 和 on-demand。
请选择其中一种方法。（默认情况下已选择 on-memory 方法。）
要选择 on-demand 方法，请单击 [On Demand] （按需）复选框。
指定内容在下次下载时生效。

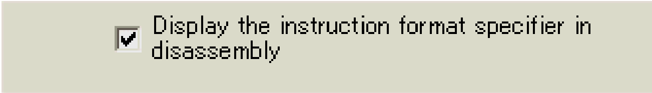
- On-memory 方法
  调试信息存储在计算机内部存储器中。
  通常选择此方法。
- On-demand 方法
  调试信息存储在计算机硬盘上的可重用临时文件中。
  由于可以重用存储的调试信息，因此下次下载同一加载模块时，下载速度会更快。
  当下载调试信息需要花费很长时间时适合选择此方法，因为 PC 的存储器比加载模块文件大小要小。

注意

- 如果加载模块很大，则使用 on-memory 方法时效率较低，因为需要花很长时间来下载。在这种情况下，请选择 on-demand 方法。
- 使用 on-demand 方法时，将在已下载的加载模块的文件夹中创建一个文件夹，用来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 “~INDEX_”。例如，如果加载模块名为 “sample.abs”，则此文件夹名为 “~INDEX_sample”。即使在退出调试器之后，也不会删除此文件夹。

5.1.2.3　　指定是否要显示指令格式说明符

指定是否要在反汇编显示中显示指令格式说明符。



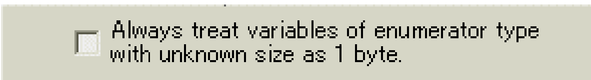
如果要显示指令格式说明符，请选中上面的复选框。

此设置只在启动调试器时才可设置或更改。

5.1.2.4　　将枚举类型的大小视为 1 个字节

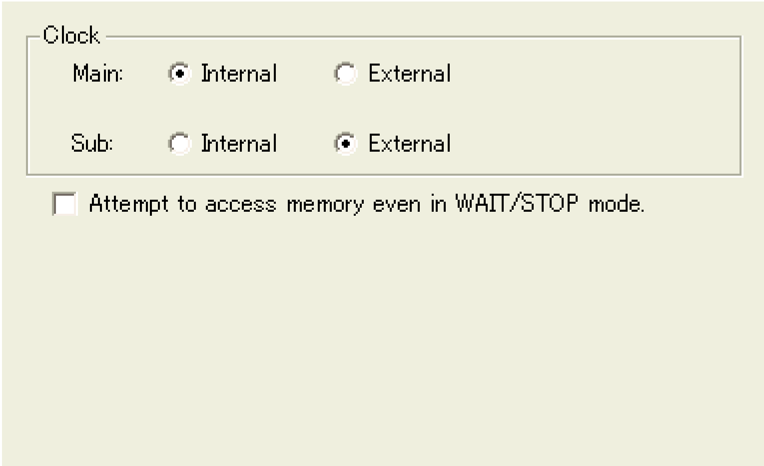
可以指定调试器是否将调试信息中大小未知的枚举类型的全部大小视为 1 个字节。为了减少存储器的消耗， NC30 和 NC308 提供了一个选项，可将枚举符类型的大小视为 1 个字节，而不是与 “int”的大小相同。请注意， NC30 和 NC308 不会在调试信息中显示枚举符类型的大小，调试器将该大小视为与 “int”的大小相同。

因此，可能无法在使用以上选项编译的目标程序中正确地引用枚举类型的值。此功能为上面的问题提供了解决方法。有关以上选项的详细信息，请参阅各种编译器的用户手册。



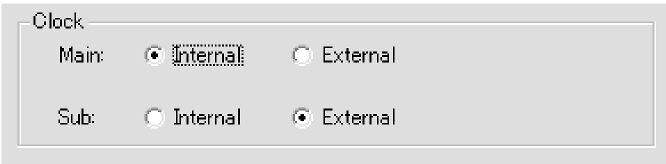
如果要将枚举类型的全部大小视为 1 个字节，请选中上面的复选框。为了反映此设置，需要再次加载调试信息。

5.1.3 [Emulator] （仿真器）选项卡



5.1.3.1 指定目标时钟

通过与目标单片机使用的时钟保持同步可更改设置。（[Internal] （内部）为默认设置。）



选择 [Internal] （内部）可设置内部时钟，选择 [External] （外部）可设置外部时钟。
指定内容在下次启动时生效。

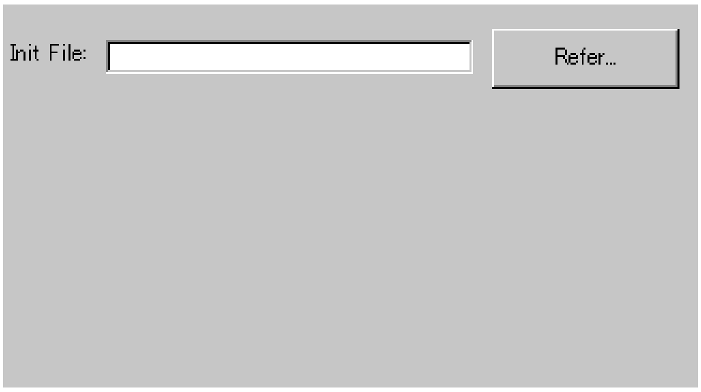
5.1.3.2 尝试在处于等待 / 停止模式期间存取存储器

如果 MCU 需要在处于 “WAIT/STOP mode”（等待 / 停止模式）期间存取存储器，请选中此项。

选中此项后，调试器将等待大约 5 秒钟，一直到 MCU 从 “WAIT/STOP mode”（等待 / 停止模式）返回，才尝试存取存储器。如果 MCU 在此期间保持 “WAIT/STOP mode”（等待 / 停止模式），操作将会出错。如果未选中此项，调试器在未真正存取存储器情况下就会出错。

5.1.4 [Script]（脚本）选项卡

指定内容在下次启动时生效。



5.1.4.1 自动执行脚本命令

要在启动调试器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。
指定的脚本文件显示在 [Init File:]（初始化文件：）字段中。
要禁止自动执行脚本命令，请删除在 [Init File:]（初始化文件：）字段中显示的字符串。

5.2 设置通信接口

5.2.1 设置 USB 接口

USB 进行通信时使用个人计算机的 USB 接口。该接口符合 USB 1.1 标准。

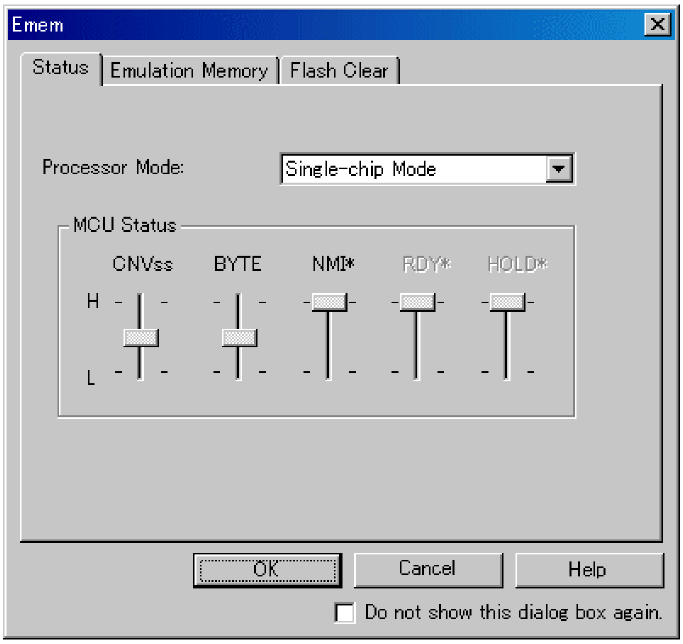
可以执行 USB 通信之前，计算机必须安装了专用设备驱动程序。有关如何安装 USB 设备驱动程序的详细信息，请参阅 “3.3.1.1 安装 USB 设备驱动程序”。

在 [Serial No.]（序列号）区域中会列出当前 USB 可连接的仿真器。请选择要连接的仿真器的序列号。

5.3 设置适用于 M32C 的调试器

5.3.1 [Emem] 对话框

在 [Emem] 对话框中，可设置有关用户目标的信息。关闭 [Init] （初始化）对话框后可打开 [Emem] 对话框。



此对话框上提供的选项卡因所用产品而异。有关详细信息，请单击下表中显示的所需选项卡名称。

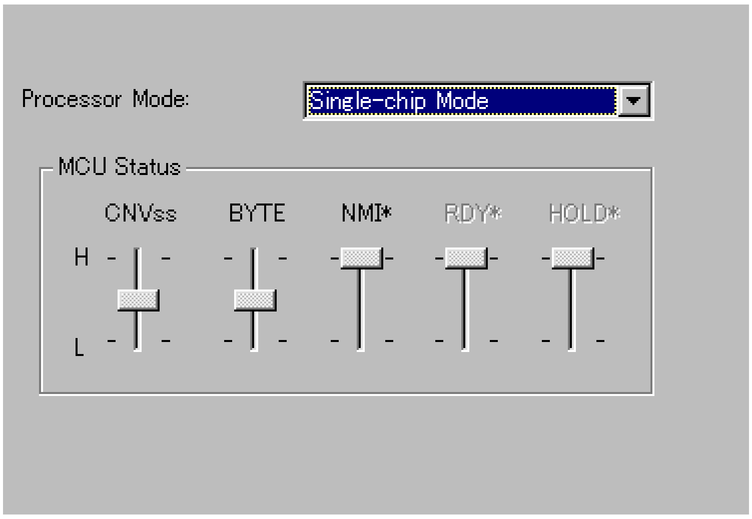
选项卡名称	内容
Status （状态）	指定处理器模式。
Emulation Memory （仿真存储器）	指定仿真存储器区域。
Flash Clear （闪存清除）	指定是否要清除 MCU 的内部闪存 ROM 中的内容。

要在下次启动调试器时让 [Emem] 对话框处于关闭状态，请选中 [Emem] 对话框底部的 “Next Hide” （下次隐藏）。可以使用下列任一种方法打开 [Emem] 对话框：

- 调试器启动以后，选择菜单 - [Setup] （设置） → [Emulator] （仿真器） → [Target...] （目标…）。

5.3.1.1　[Status]（状态）选项卡

指定内容在下次启动时生效。



5.3.1.1.1　选择处理器模式

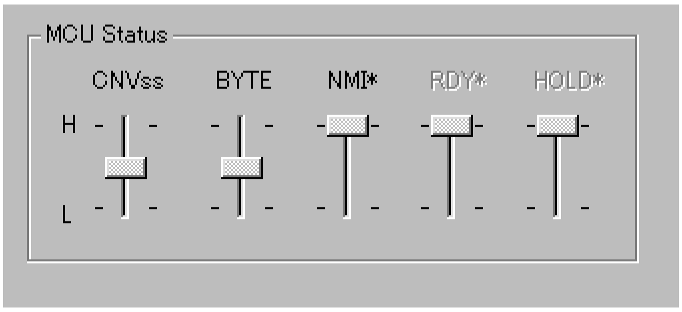
指定目标系统的处理器模式。

可以指定以下任意一项。

- Single-chip Mode
  单片模式
- Memory Expansion 8 Bit
  存储器扩展模式（8 位总线宽度）
- Memory Expansion 16 Bit
  存储器扩展模式（16 位总线宽度）

5.3.1.1.2　　检查 MCU 状态

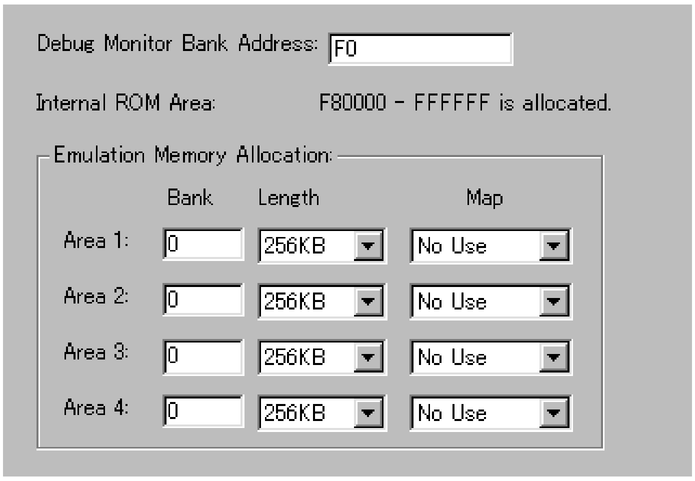
单击此选项卡可显示 MCU 各引脚的状态。可以检查 MCU 引脚的状态是否与要设置的处理器模式相匹配。



如果滑块处于中间位置，则表示值不确定。

5.3.1.2　　[Emulation Memory]（仿真存储器）选项卡

指定内容在下次启动时生效。



5.3.1.2.1 调试监控器的存储体地址设置

此产品分配一个 64 KB 的连续地址区作为仿真器的工作区，以供调试监控器使用。
指定目标系统不使用的任何存储体。调试监控器使用从指定存储体起始地址开始的 64 KB 区域。
（示例：如果指定存储体为 “F0”，则调试监控器使用从地址 F000000h 开始的 64 KB 区域。）

- 不能引用或设置此处指定存储体中的内容。
  此区域中的内容显示在 [Memory] （存储器）窗口中或 [Program/Source] （程序/源）窗口的反汇编显示模式下时可能不正确。
- 不能指定以下存储体地址：
  — MCU 内部资源 （例如， SFR 和 RAM 区）
  — DRAM 区和多路复用区
  — 中断向量区

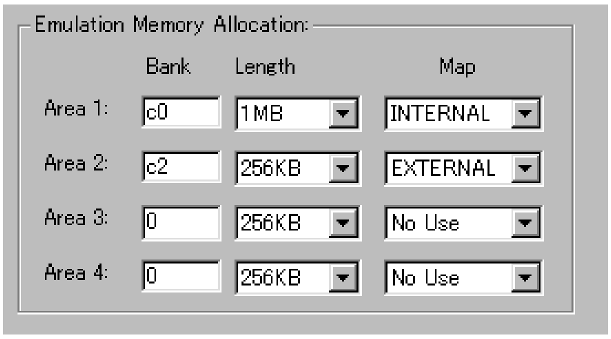
5.3.1.2.2 内部 ROM 的自动仿真存储器分配

选择单片模式或存储器扩展模式时，仿真存储器会自动分配到内部 ROM 区。
自动分配的内部 ROM 地址范围显示在此字段中。

5.3.1.2.3 扩展区的仿真存储器分配

选择存储器扩展模式或微处理器模式时，仿真存储器可以分配到要调试的扩展区 （最多四个区）。
（当仿真存储器板未连接到仿真器时，不能分配仿真存储器。）

此时，可分配调试目标区的存储器并指定映射信息。



按照下面说明的过程进行操作。

Bank （存储体） （设置存储体地址）	指定要分配的调试目标区的存储体地址 （十六进制）。 如果指定为 C0，则 C00000h 为调试目标区的起始地址。
Length （长度） （指定区域大小）	指定调试目标区的大小 （256 字节或 1 MB）。 如果 [Length] （长度）指定为 “256 字节”，则会为 [Bank] （存储体）指定 00、 04、 08 到 FC （每四个存储体）；如果 [Length] （长度）指定为 “1 MB”，则会为 [Bank] （存储体）指定 00、 10、 20 到 F0 （每 16 个存储体）。
Map （映射） （指定区域映射）	为指定区域指定映射信息 （[Internal] （内部）或 [External] （外部））。 如果不指定任何区域，则选择 “No Use”（未使用）。 Internal （内部）：指定为 “Internal”（内部）的区域会映射到内部区域 （仿真存储器）。 External （外部）：指定为 “External”（外部）的区域会映射到外部区域 （目标系统中的外部资源）。

- 在 [Map] （映射）中选择 “No Use”（未使用）的区域和这里未指定的区域都会映射到外部区域。
  与明确指定为 “External”（外部）的区域相比，这种区域的唯一区别是下载速度。（到这些区域的下载速度低于到指定为 “External”（外部）的区域的下载速度。）
- 内部 ROM 区会自动映射到仿真存储器。因此无需在这里进行设置。
- 请注意调试区不会重叠。
- 请确保指定调试目标区的全部大小不超过所用仿真存储器板上仿真存储器的大小。

仿真存储器区域的设置会根据指定处理器模式的不同而有所不同。

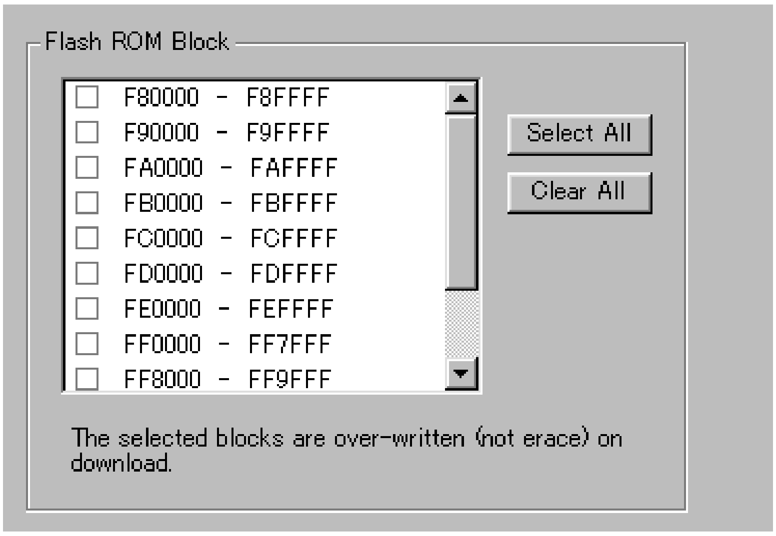
- “Single-chip Mode”（单片模式）
  无需指定要作为仿真存储器分配的区域。
  内部 ROM 区会自动映射到仿真存储器。自动映射区的地址范围显示在 [Internal ROM Area:] （内部 ROM 区：）字段中。
- “Memory Expansion Mode”（存储器扩展模式）（8 位和 16 位）
  如果除内部 ROM 区之外还有一个区域要分配为仿真存储器，请单独指定它。
  内部 ROM 区会自动映射到仿真存储器。自动映射区的地址范围显示在 [Internal ROM Area:] （内部 ROM 区：）字段中。
- “Microprocessor Mode”（微处理器模式）（8 位和 16 位）
  指定要单独分配的区域。（没有自动分配的区域。）

注意事项

- 使用 Map 命令指定的映射设置数据不会反映在 [Emem] 对话框中。
- 按照使用优先级顺序设置仿真存储器区域。
  通过 Map 命令设置的仿真存储器区域都带有编号 （未使用的区域不算）。
  因此，在 [Emem] 对话框中设置的仿真存储器区域和通过 Map 命令设置的仿真存储器区域的编号不匹配。

5.3.1.3 [Flash Clear] （闪存清除）选项卡

指定内容在下次启动时生效。



5.3.1.3.1 设置清除 MCU 的内部闪存 ROM

指定下载目标程序或数据时是否要清除 MCU 的内部闪存 ROM 中的内容。

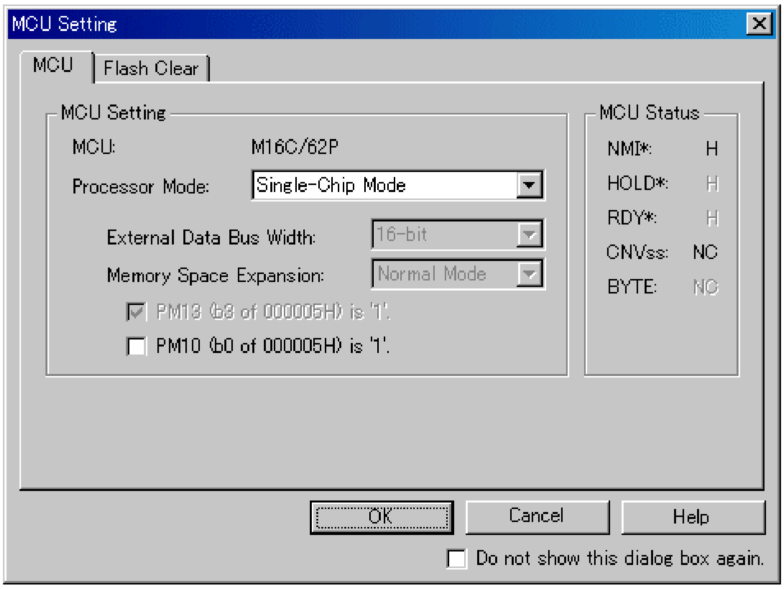
MCU 的内部闪存 ROM 会逐块显示在列表视图中。

- 选中其复选标记的那些块不会在下载时清除其闪存内容。处于不盖写状态的存储内容在下载时会保持原样。
- 未选中其复选标记的那些块会在下载时清除其闪存内容。
- 按 [Select All] （全选）按钮可防止在下载时清除所有块的闪存内容。
- 按 [Clear All] （全部清除）按钮可在下载时清除所有块的闪存内容。

5.4 设置适用于 M16C/R8C 的调试器

5.4.1 MCU 设置对话框

在 [MCU Setting] （MCU 设置）对话框中，可设置有关用户目标的信息。关闭 [Init] （初始化）对话框后可打开 [MCU Setting] （MCU 设置）对话框。



此对话框上提供的选项卡因所用产品而异。有关详细信息，请单击下表中显示的所需选项卡名称。

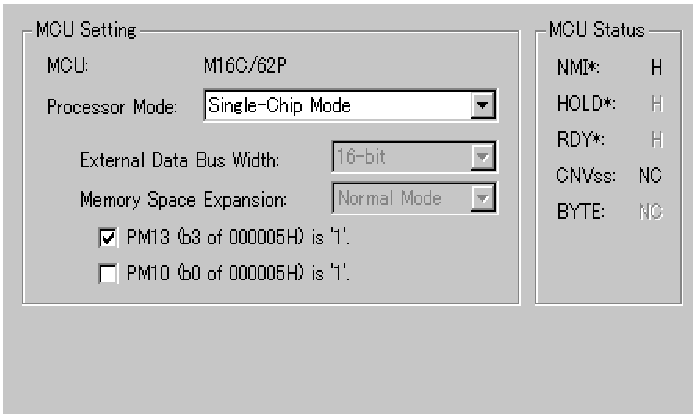
选项卡名称	内容
MCU	指定 MCU 的处理器模式、调试选项等等。
MAP	设置仿真存储器映射到的存储区。 （* 要使用此选项卡，必须有仿真存储器板。）
Flash Clear （闪存清除）	指定是否要清除 MCU 的内部闪存 ROM 中的内容。

要在下次启动调试器时让 [MCU Setting] （MCU 设置）对话框处于关闭状态，请选中 [MCU Setting] （MCU 设置）对话框底部的 “Next Hide”（下次隐藏）。可以使用下列任一种方法打开 [MCU Setting] （MCU 设置）对话框：

- 调试器启动以后，选择菜单 - [Setup] （设置）→ [Emulator] （仿真器）→ [Target...] （目标…）。

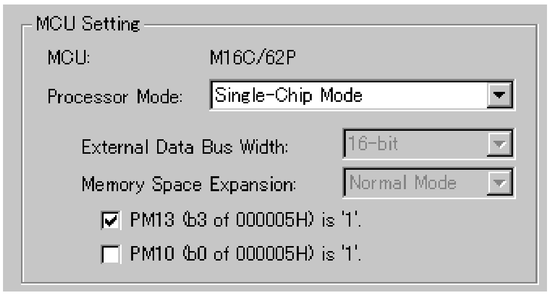
5.4.1.1 MCU 选项卡

指定内容在下次启动时生效。



5.4.1.1.1 选择处理器模式

指定目标系统的处理器模式。



可以指定以下任意一项。

- Single-chip Mode
  单片模式
- Memory Expansion Mode
  存储器扩展模式
- Microprocessor Mode
  微处理器模式

此外，还需要根据选择的处理器模式指定以下信息。

- “External Data Bus Width”（外部数据总线宽度）

  如果选择了存储器扩展模式或微处理器模式，请指定外部总线宽度为 “16-bit”（16 位）或 “8-bit”（8 位）。请确保指定的外部总线宽度与 BYTE 引脚的设置匹配。

- “Memory Space Expansion”（存储器空间扩展）

  如果选择了存储器扩展模式或微处理器模式，请指定是否要使用存储器空间扩展工具。如果要使用存储器空间扩展工具，请选择 “4MB Mode”（4MB 模式）；如果不使用，请选择 “Normal Mode”（正常模式）。

- PM13(b3 of 000005H)

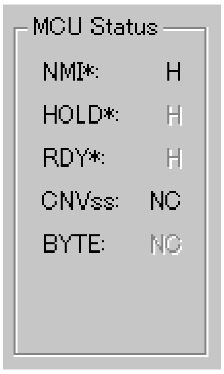
  指定是否设置 PM13 位 （000005H 的 b3）。如果在 PM13 的设置为 1 时使用目标系统，请选择此选项。

- PM10(b0 of 000005H)

  指定是否设置 PM10 位 （000005H 的 b0）。如果在 PM10 的设置为 1 时使用目标系统，请选择此选项。

5.4.1.1.2 检查 MCU 状态

单击此选项卡可显示 MCU 各引脚的状态。可以检查 MCU 引脚的状态是否与要设置的处理器模式相匹配。

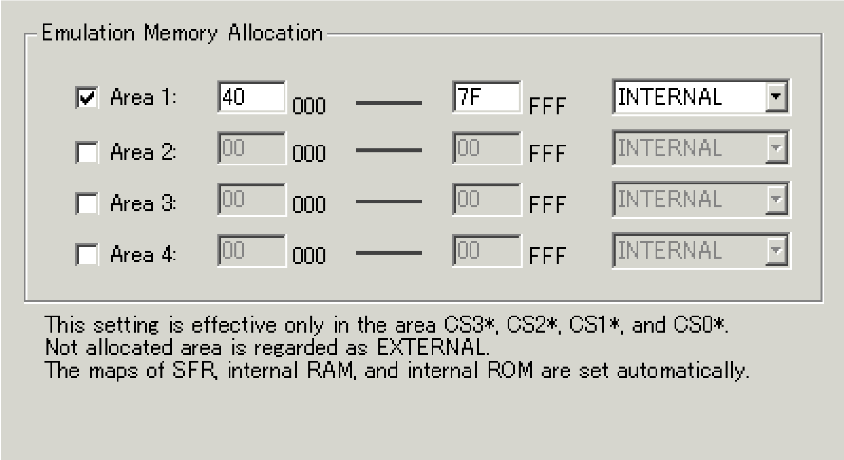


“NC”表示值不确定。

5.4.1.2 [MAP] 选项卡

要使用此指令，必须有仿真存储器板。

指定内容在下次启动时生效。



5.4.1.2.1 仿真存储器分配

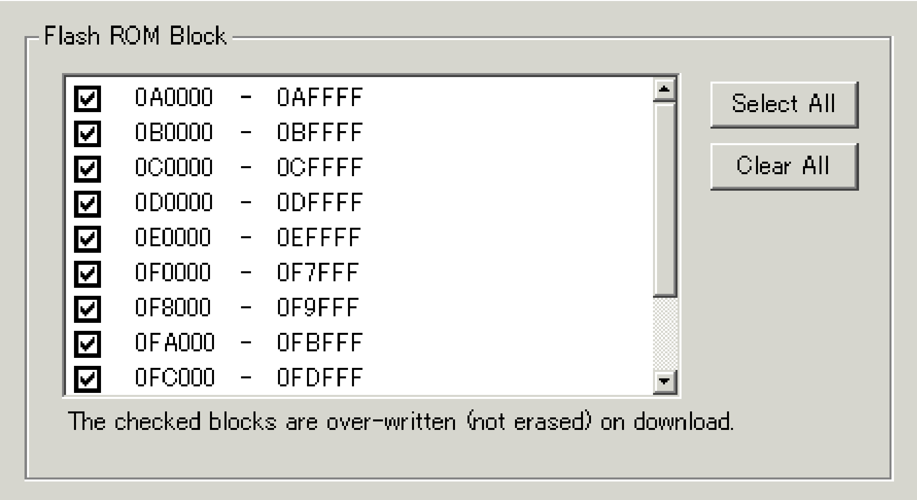
以 4 KB 为单位设置仿真存储器要映射到的存储区。可以设置四个这样的存储区。

仿真存储器会映射到标志为 “Internal”（内部）的区域。未选中区域和未指定任何内容的区域会分配到外部区域。

请注意，[MAP]（映射）设置仅对 CS3*、CS2*、CS1* 和 CS0* 区域有效。SFR、内部 ROM 和内部 RAM 区会自动映射。

5.4.1.3 [Flash Clear] （闪存清除）选项卡

指定内容在下次启动时生效。



5.4.1.3.1 设置清除 MCU 的内部闪存 ROM

指定下载目标程序或数据时是否要清除 MCU 的内部闪存 ROM 中的内容。 MCU 的内部闪存 ROM 会逐块显示在列表视图中。

- 选中其复选标记的那些块不会在下载时清除其闪存内容。处于不盖写状态的存储内容在下载时会保持原样。
- 未选中其复选标记的那些块会在下载时清除其闪存内容。
- 按 [Select All] （全选）按钮可防止在下载时清除所有块的闪存内容。
- 按 [Clear All] （全部清除）按钮可在下载时清除所有块的闪存内容。

教程

6 教程

6.1 简介

本章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。每个目标和每个 MCU 都有各自的工作空间。请选择与您的系统相适应的工作空间，然后从 [Open Workspace...]（打开工作空间…）菜单中打开工作空间文件 (*.hws)。

此教程程序是使用 C 程序编写的，可用来对十个随机数据项按升序或降序进行排序。

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，有关地址可能与本章提供的地址不同。

6.2 使用

请遵照以下说明进行操作：

6.2.1 步骤 1：启动调试器

6.2.1.1 使用前的准备工作

要运行 HEW 并连接仿真器，请参阅 “4 使用前的准备工作”。

6.2.1.2 设置调试器

如果调试器与仿真器相连，则会显示用于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 “5 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。

6.2.2 步骤 2：检查 RAM 的运行情况

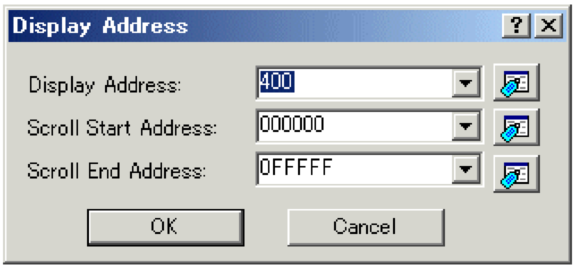
检查 RAM 是否正常运行。在 [Memory] （存储器）窗口中显示和编辑存储器的内容，检查存储器是否正常运行。

注意

存储器可安装在某些单片机的板上。但是，在这种情况下，上述检查存储器操作的方法可能不适当。建议创建一个检查存储器的程序。

6.2.2.1 检查 RAM 的运行情况

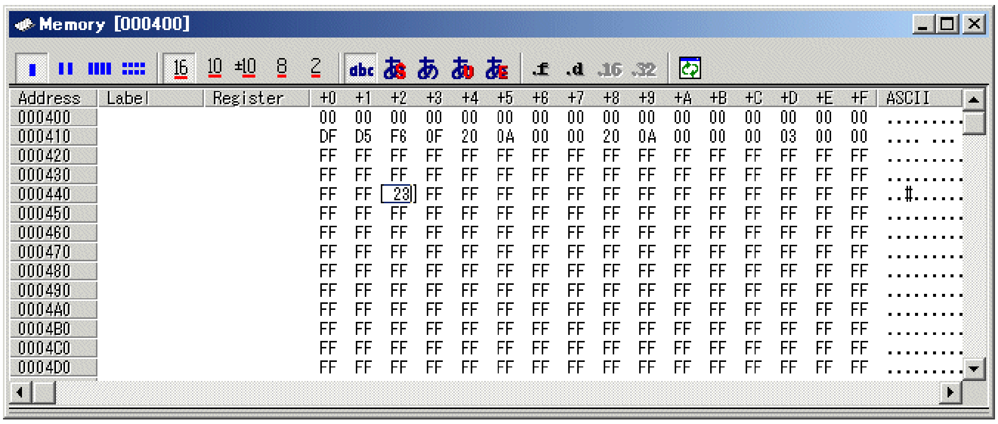
从 [View] （视图）菜单的 [CPU] 子菜单中选择 [Memory] （存储器），然后在 [Display Address] （显示地址）编辑框输入 RAM 地址 （此处输入 “400”）。在 [Scroll Start Address] （滚动起始地址）和 [Scroll End Address] （滚动结束地址）编辑框中保留默认设置。（默认情况下，滚动范围为 0h 到 MCU 的最大地址。）



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。此时显示 [Memory] （存储器）窗口，窗口中显示指定的存储区。



将鼠标光标放在 [Memory] （存储器）窗口中数据显示区的某个点，然后双击即可更改该点处的值。

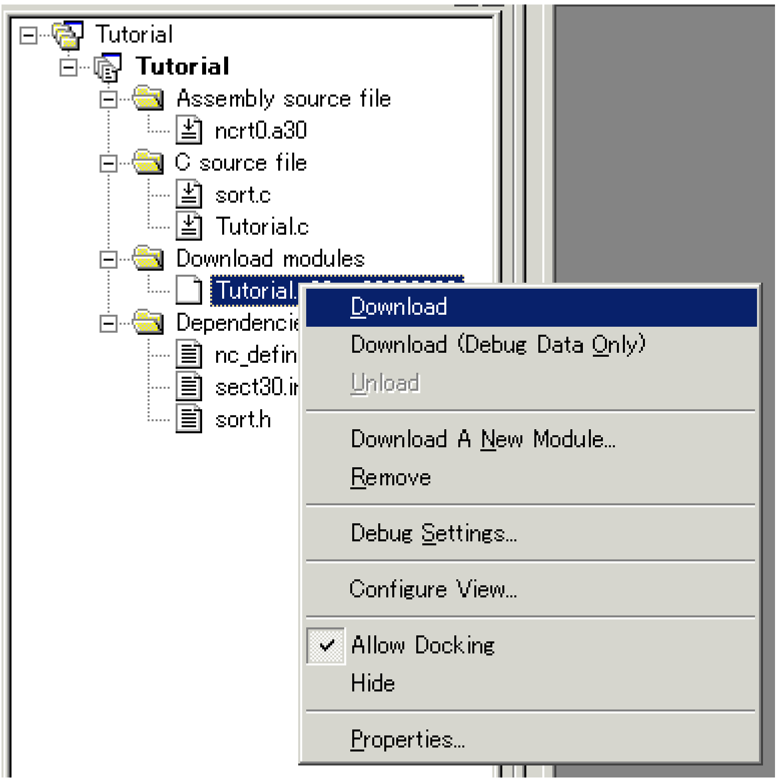
6.2.3 步骤 3：下载教程程序

6.2.3.1 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 对应的图像和地址来替换屏幕上的图像和地址。

- 适用于 M16C/R8C 或 M32C 的调试器

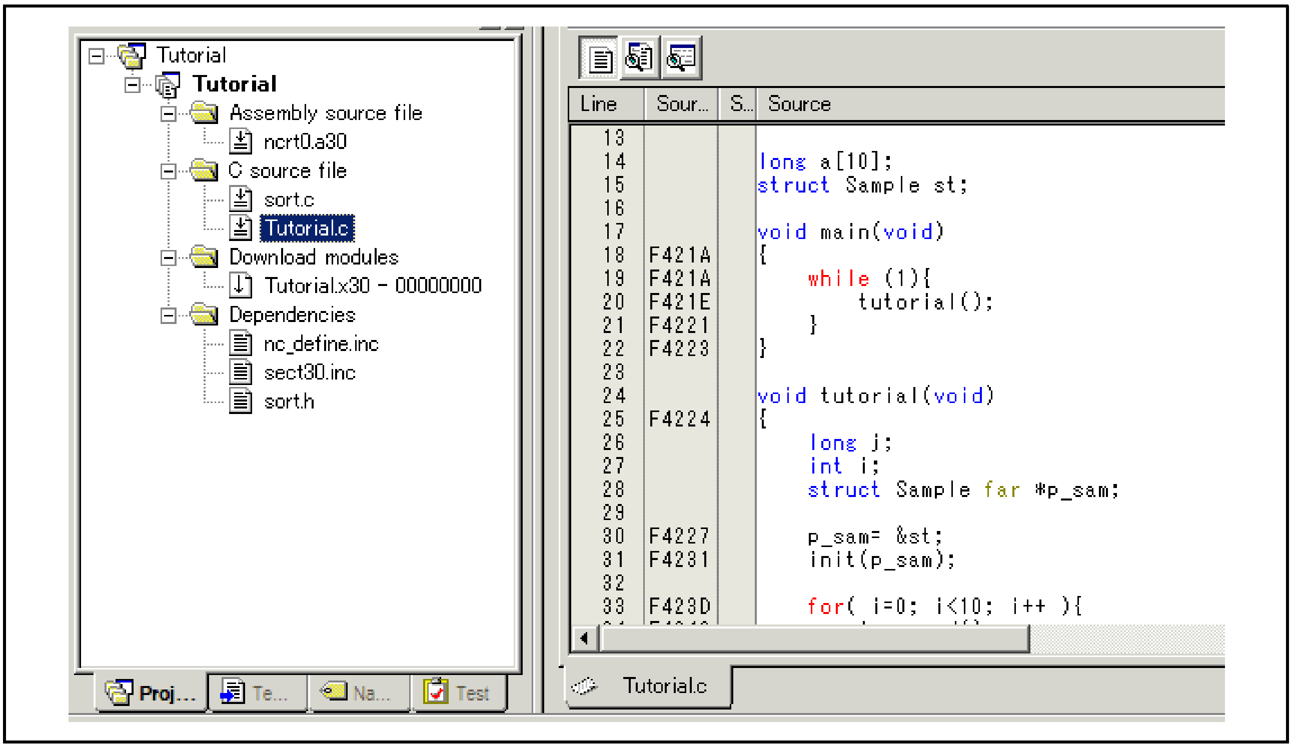
  在 [Download modules] （下载模块）下的 [Tutorial.x30] 中选择 [Download module] （下载模块）。



6.2.3.2 显示源程序

此调试器允许用户在源码级对用户程序进行调试。

双击 [C source file] （C 源文件）下的 [tutorial.c]。此时会打开 [Editor (Source)] （编辑器 （源））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…）选项来设置易于阅读的字体和字号。

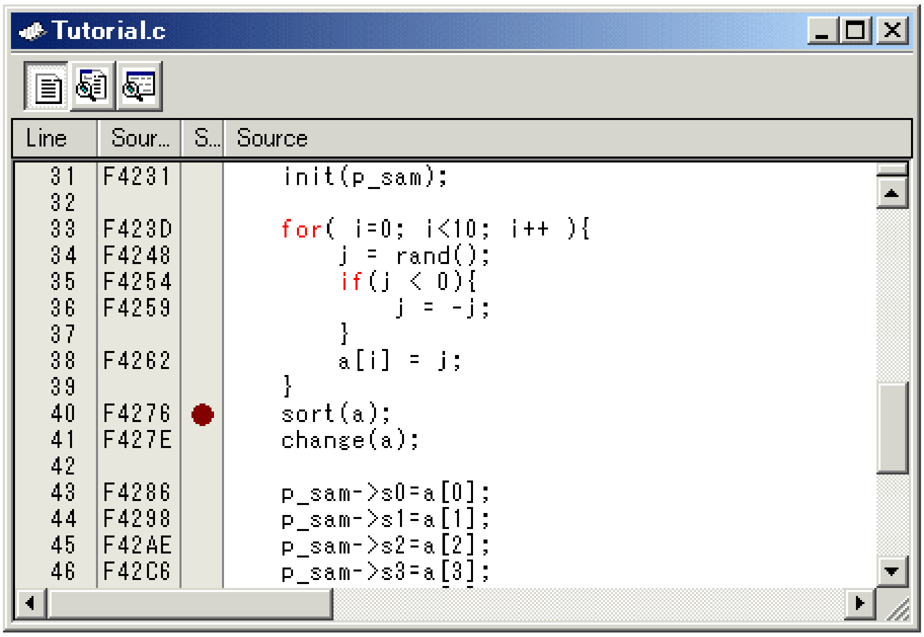
最初，在 [Editor (Source)] （编辑器 （源））窗口中会显示用户程序的起始部分，但用户可以使用滚动条查看用户程序中的其他语句。

6.2.4 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）） 窗口，可在程序中的任意位置轻松设置软件断点。

6.2.4.1 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 函数的这一行上，双击 [S/W breakpoints] （软件断点）列。



调用 sort 函数的行上会显示一个红色符号。这表示已设置一个软件断点。

6.2.5 步骤 5：执行程序

按以下描述执行程序：

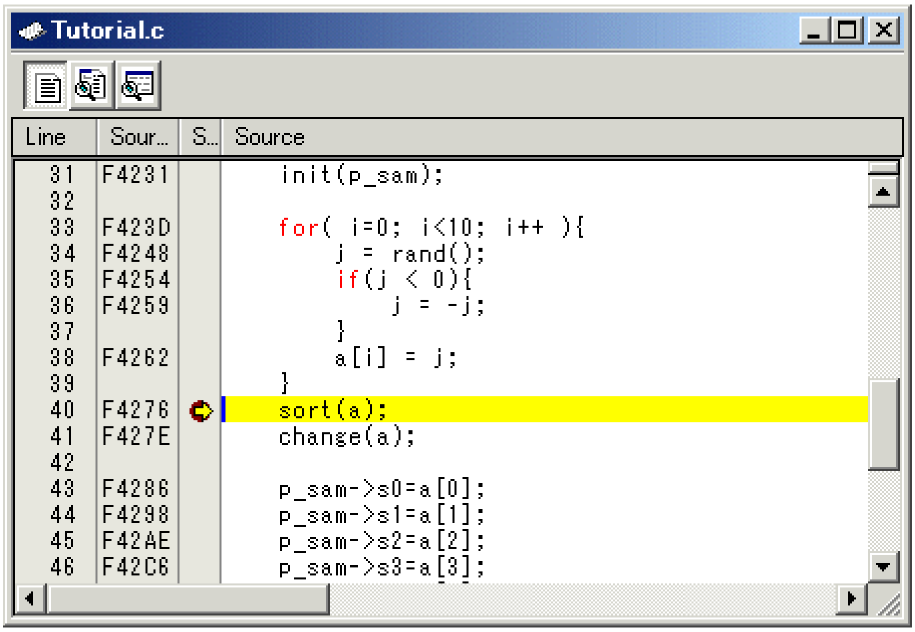
6.2.5.1 复位 CPU

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  。

6.2.5.2 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  。

程序将执行到已设置断点的位置，在 [S/W Breakpoints] （软件断点）列中会显示一个箭头，指示程序暂停的位置。

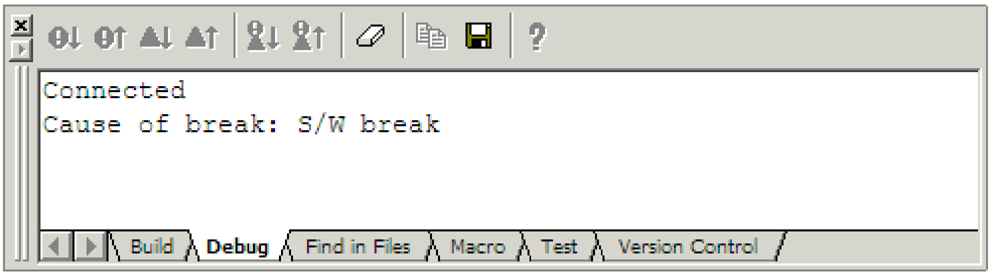


注意

如果源文件在某个断点之后显示，则可能需要查询源文件的路径。如遇到这种情况，请指定源文件的位置。

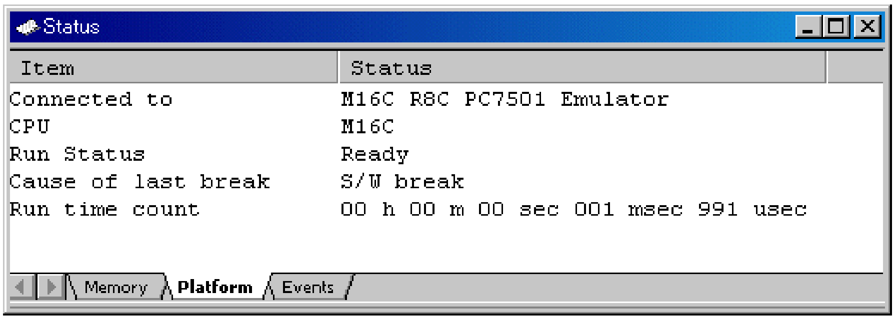
6.2.5.3 查看中断原因

中断原因会显示在 [Output]（输出）窗口中。



用户还可以在 [Status]（状态）窗口中查看上一次中断的原因。

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]（上次中断原因）对应的 [Status]（状态）。



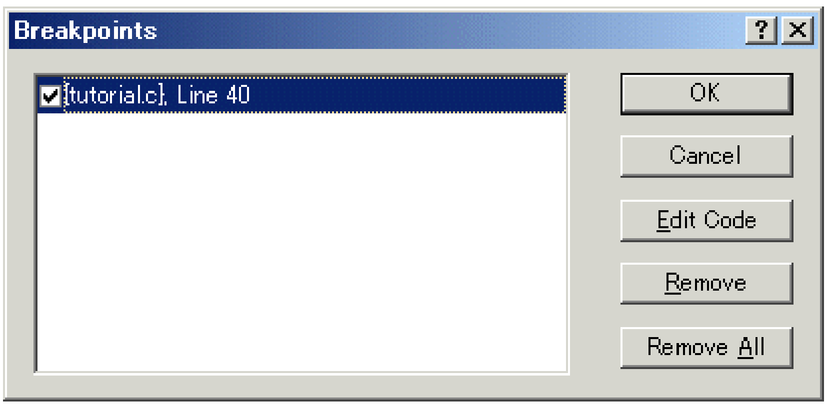
关于中断原因的表示法，请参阅“11 显示程序停止的原因”。

6.2.6 步骤 6：查看断点

用户在 [Breakpoints]（断点）对话框中可查看程序中设置的所有断点。

6.2.6.1 查看断点

按 Ctrl+B 键后会显示 [Breakpoints]（断点）对话框。



用户在此窗口中可删除、允许或禁止断点。

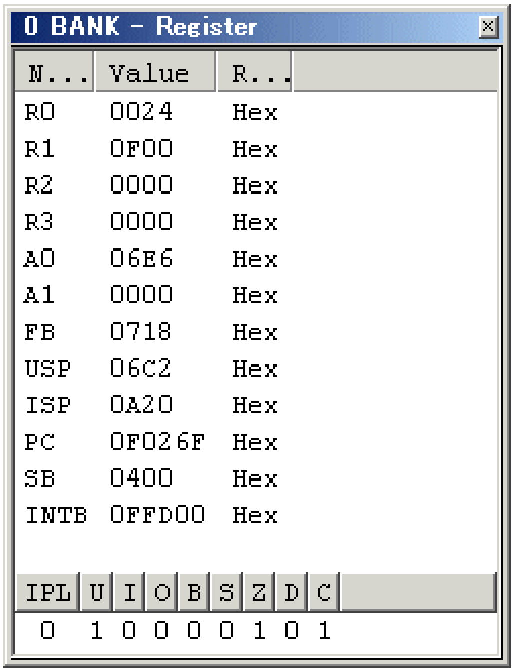
6.2.7 步骤 7：查看寄存器

用户在 [Register] （寄存器）窗口中可查看所有寄存器 / 标志值。

6.2.7.1 查看寄存器

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

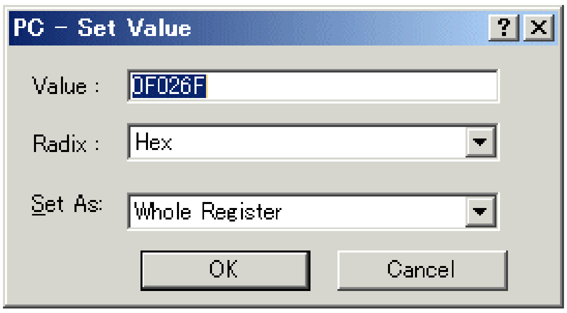
下图为适用于 M16C/R8C 的调试器中的 [Register] （寄存器）窗口。



6.2.7.2 设置寄存器值

在此窗口中可更改寄存器 / 标志值。

双击要更改的寄存器行。此时会打开该对话框。输入要更改的值。



6.2.8 步骤 8：查看存储器

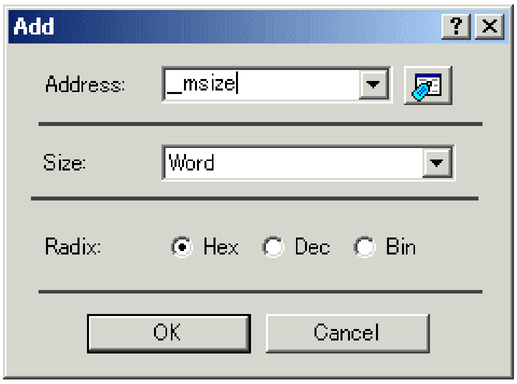
指定标签名称后，用户可以在 [ASM Watch] （ASM 监视）窗口中查看已注册标签的存储器内容。

6.2.8.1 查看存储器

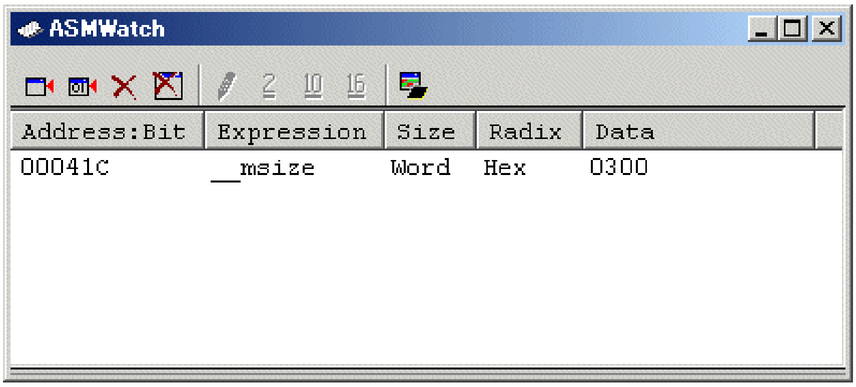
例如，要查看与 __msize 对应的采用字大小表示的存储器内容：

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。

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



单击 [OK] （确定）按钮。此时在 [ASM Watch] （ASM 监视）窗口中会显示存储器的指定区域。



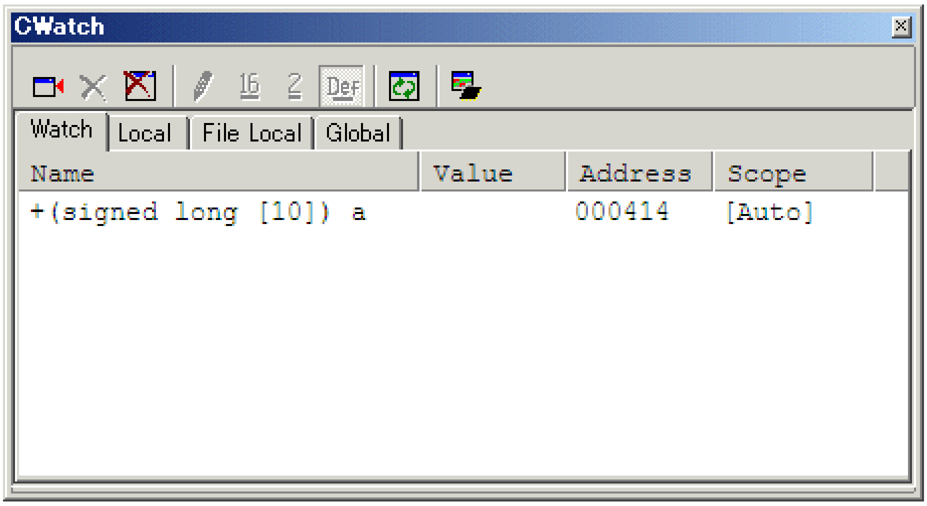
6.2.9　步骤 9：监视变量

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。

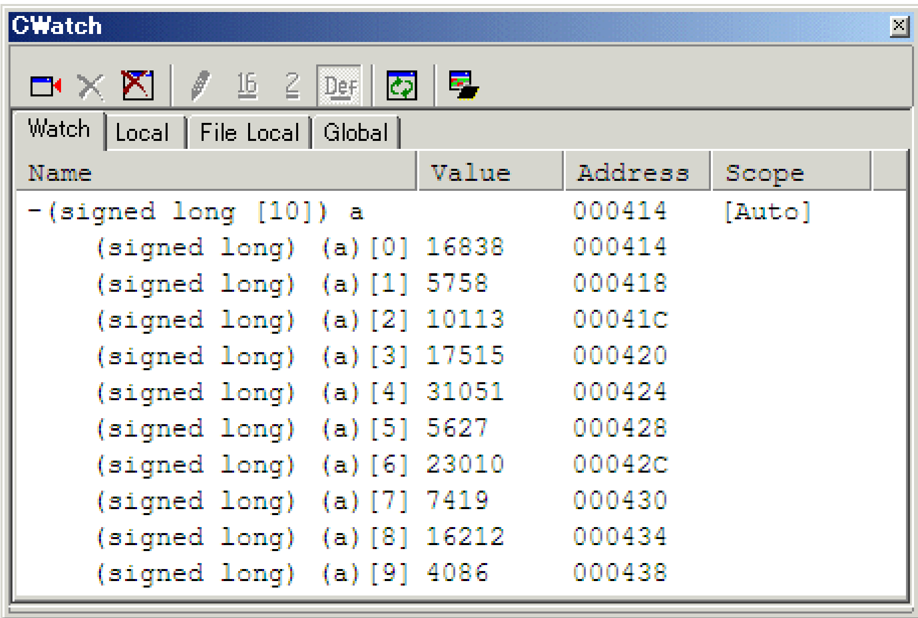
6.2.9.1　监视变量

例如，通过执行以下过程，在程序开头声明的 long 类型的数组 a 上设置一个监视点：

在 [Editor (Source)]（编辑器（源））窗口中，单击显示数组 a 的左侧以定位光标，然后单击鼠标右键选择 [Add C Watch...]（添加 C 监视…）。此时 [C watch]（C 监视）窗口中显示变量的 [Watch]（监视）选项卡会打开。

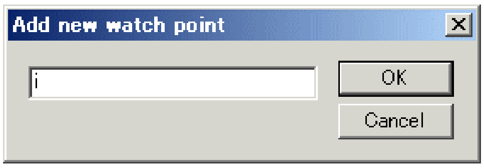


在 [C watch]（C 监视）窗口中，用户可以单击数组 a 左侧的“+”标记来监视所有元素。

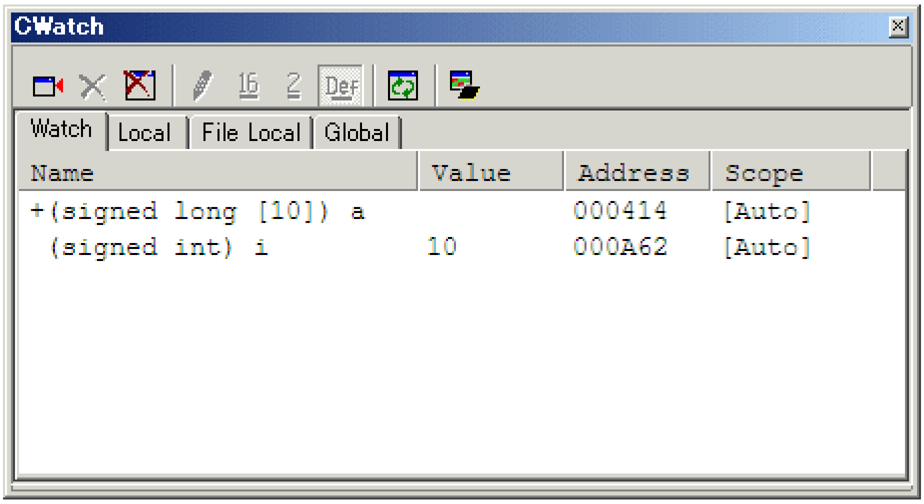


6.2.9.2　　注册变量

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。
在 [C Watch]（C 监视）窗口中单击鼠标右键，从弹出菜单中选择 [Add...]（添加…）。
此时显示以下对话框。请输入变量 i。



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



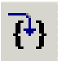
6.2.10 步骤 10：逐步执行程序

此调试器提供了多个逐步执行菜单命令，可以有效地对程序进行调试。

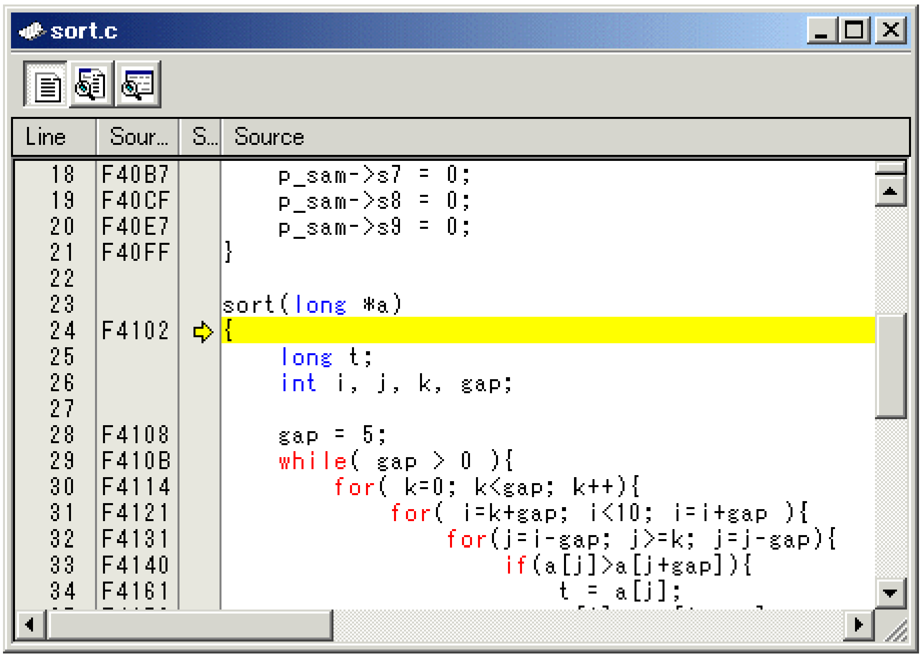
1. [Step In] （跳入）
   执行每条语句，包括函数 （子例程）内的语句。
2. [Step Out] （跳出）
   跳出某一函数 （子例程），在程序中调用该函数 （子例程）的语句下一句处停止。
3. [Step Over] （跳过）
   单步执行一个函数 （子例程）调用。
4. [Step...] （单步…）
   以指定的速率和次数重复单步执行。

6.2.10.1 执行 [Step In] （跳入）命令

[Step In] （跳入）命令会跳入被调用的函数 （子例程）并停止于该函数 （子例程）的第一条语句。

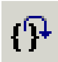
要逐步执行 sort 函数，请从 [Debug] （调试）菜单选择 [Step In] （跳入），或者单击工具栏上的 [Step In] （跳入）按钮  。

PC 光标会移到 [Editor (Source)] （编辑器 （源））窗口中 sort 函数的第一条语句。

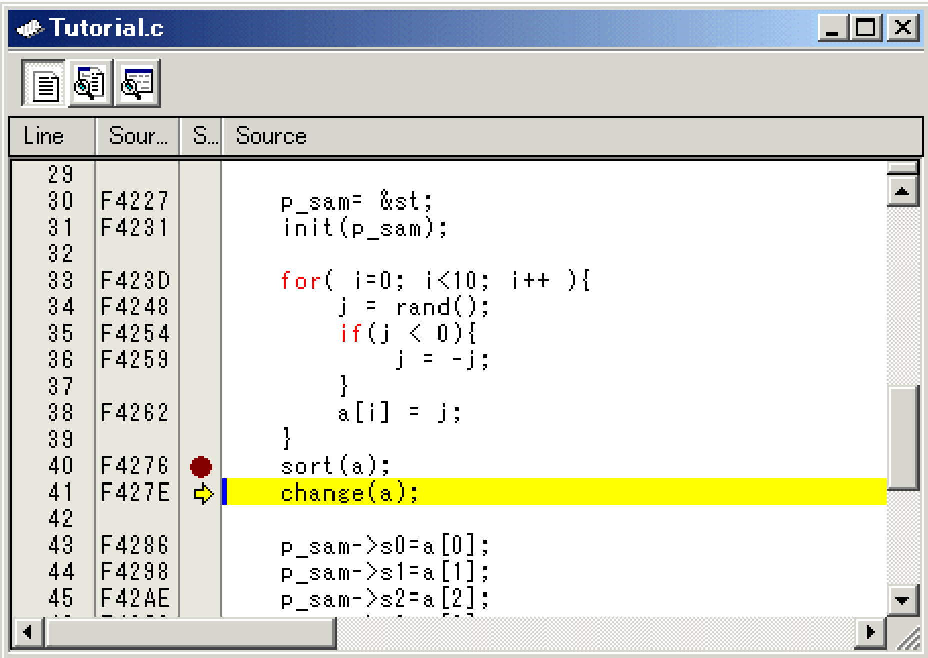


6.2.10.2 执行 [Step Out] （跳出）命令

[Step Out] （跳出）命令会跳出被调用的函数（子例程），并停止于 main 函数中调用语句的下一条语句。

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 光标移出 sort 函数后会移到 change 函数前面的位置。

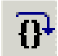


注意

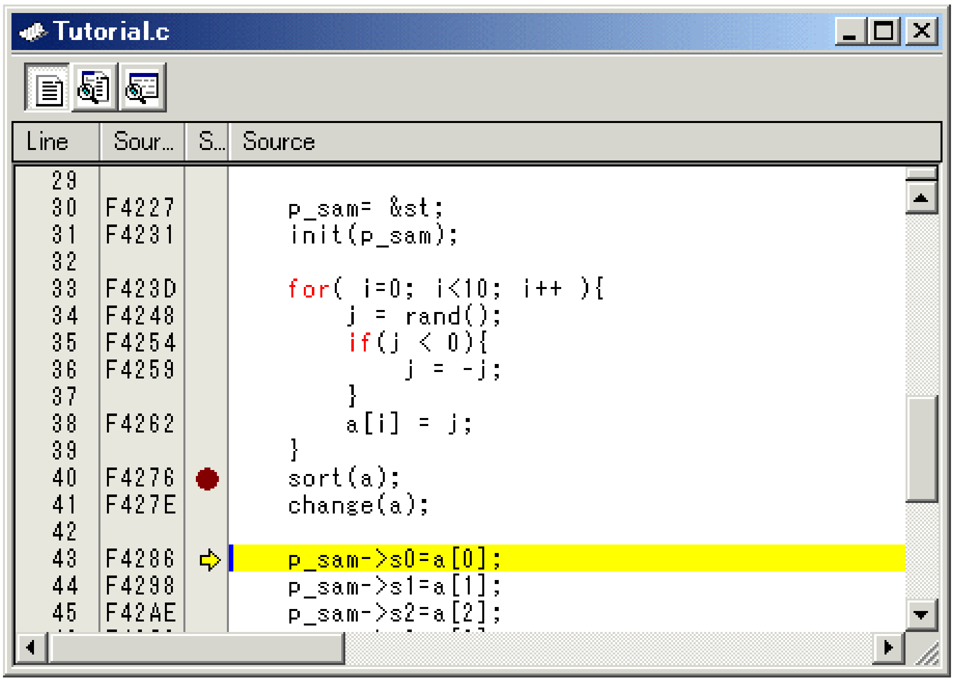
执行此函数需要一些时间。查明调用源后，可使用 [Go To Cursor] （转至光标）。

6.2.10.3 执行 [Step Over] （跳过）命令

[Step Over] （跳过）命令会以单步形式执行函数 （子例程）调用并停止于主程序的下一条语句。

要逐步执行 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 光标会移到 change 函数中的下一个位置。



6.2.11 步骤 11：强制中断程序执行

此调试器可以强制中断程序执行。

6.2.11.1 强制中断程序执行

取消所有断点。

要执行 main 函数的余下部分，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  。

程序会进入死循环。要强制中断执行，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  。

6.2.12　步骤 12：显示局部变量

用户可以在 [C Watch] （C 监视）窗口中显示函数中的局部变量。

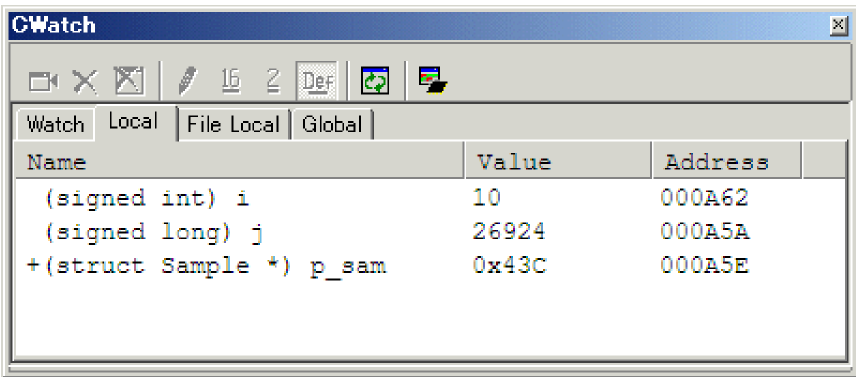
6.2.12.1　显示局部变量

例如，我们将查看 tutorial 函数中的局部变量，该函数声明了三个局部变量：i、 j 和 p_sam。

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

- [Watch] （监视）选项卡
  仅显示用户已注册的变量。
- [Local] （局部）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而改变，则 [Local] （局部）选项卡的内容也会随之改变。
- [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而改变，则 [File Local] （文件局部变量）选项卡的内容也会随之改变。
- [Global] （全局）选项卡
  显示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）选项卡。



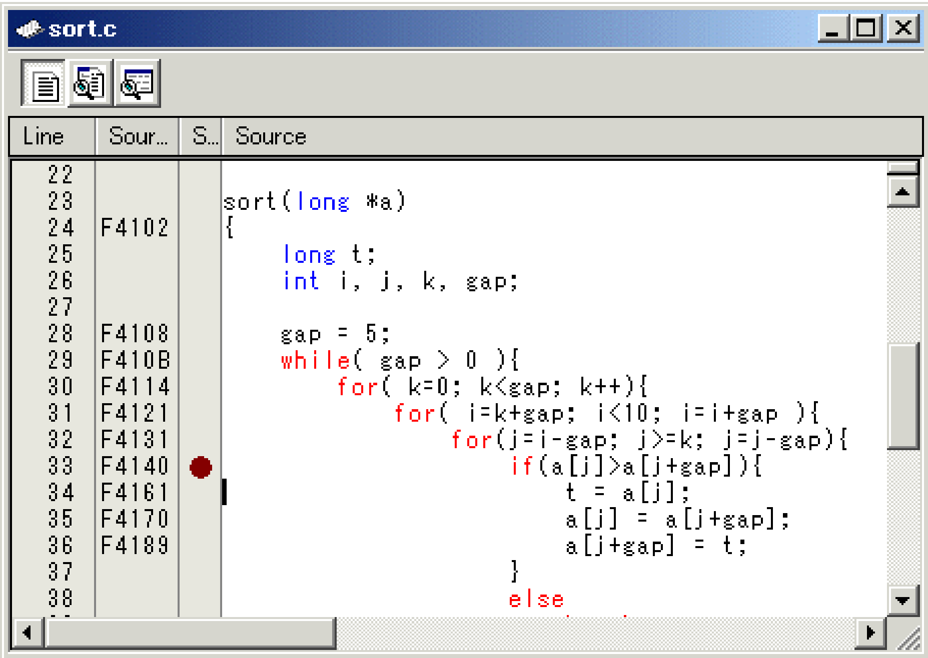
在 [Local] （局部）窗口中，双击指针 p_sam 左侧的 “+” 标记来显示结构 *(p_sam)。
用户在 Tutorial 函数末尾引用这一结构的成员时，随机数据会按降序排列。

6.2.13 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。

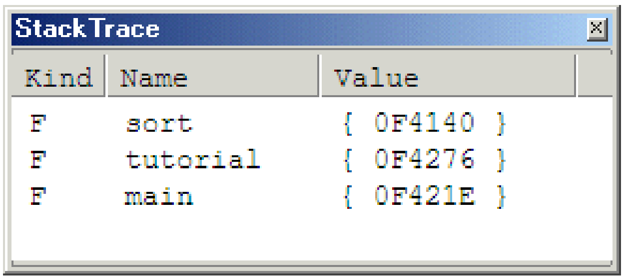
6.2.13.1 引用函数调用状态

双击 sort 函数中的 [S/W Breakpoints] （软件断点）列并设置一个软件断点。



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  。

程序执行中断后，从 [View] （视图）菜单的 [Code] （代码）子菜单中选择 [Stack Trace] （堆栈跟踪）可打开 [Stack Trace] （堆栈跟踪）窗口。



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

6.2.14 后续内容

此教程对此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。通过准确地隔离和确定出现硬件和软件问题的各种情况，可以有效地对此类问题进行研究。

参考

7 窗口 / 对话框

此调试器的窗口如下所示。

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）窗口	[View] （视图）→ [Break] （断点）→ [H/W Break Points] （硬件断点）
[Address Interrupt Break Point Setting] （地址中断断点设置）窗口	[View] （视图）→ [Break] （断点）→ [Address Interrupt Break Points] （地址中断断点）
[Trace Point Setting] （跟踪点设置）窗口	[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）
[Data Trace] （数据跟踪）窗口	[View] （视图）→ [Trace] （跟踪）→ [Data Trace] （数据跟踪）
[GUI I/O] 窗口	[View] （视图）→ [Graphic] （图形）→ [GUI I/O]
[MR] 窗口	[View] （视图）→ [RTOS] → [MR]

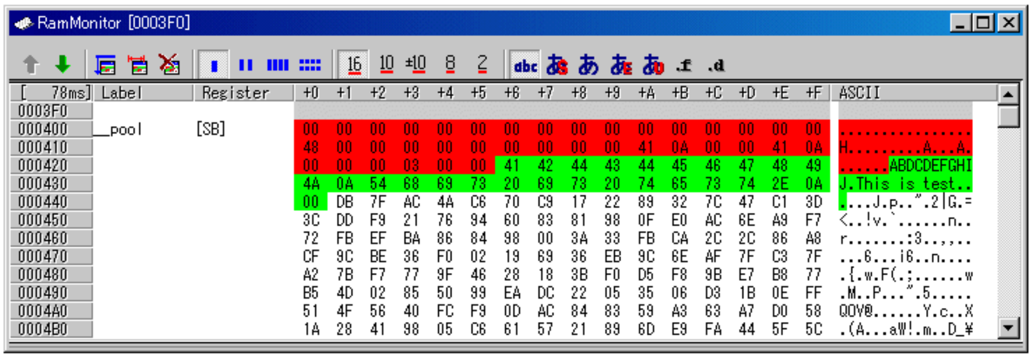
有关以下窗口的参考内容，请参阅 HEW 主部件的帮助文件。

- [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

7.1 [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。



- 此系统具有 1 KB 大小的 RAM 监控区域，该区域可位于任何连续的地址区域，也可以由 4 个大小为 256 字节的单独块组成。
- 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 “7.1.2 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。

```
001B  00C8  00D2  0000  007C
0000  0000  0000  0000  0000
0000 (007C) FF8C  0000  0000
0000  0000  0000  0050  0000
```

- 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

7.1.1 扩展菜单

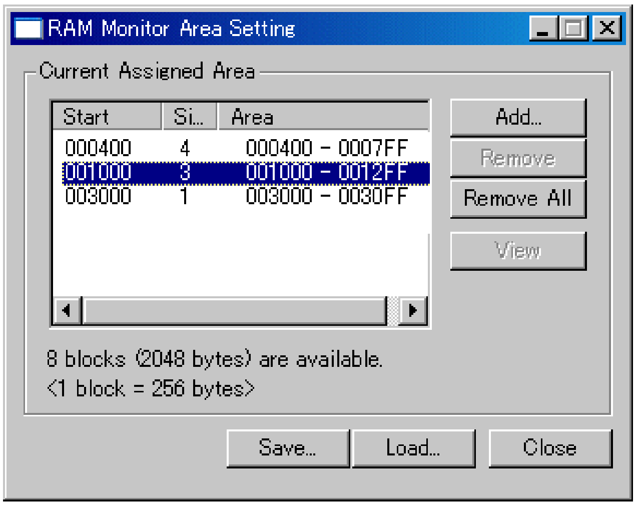
此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

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。

7.1.2 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 [RAM monitor area setup] （RAM 监控区域设置）窗口。此窗口中列出当前设置的 RAM 监控区域。



使用此窗口可添加、删除或更改 RAM 监控区域。

- 通过指定 RAM 监控区域的起始地址和大小 （大小通过块数指定）对 RAM 监控区域进行指定。
- 起始地址可以按 0x100 字节单位指定。

  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 0x100 字节单位的地址值，然后才能用于设置。
- 通过指定块的数目指定 RAM 监控区域的大小。

  对于小型仿真器，一个块的大小为 256 字节。最多可指定 4 个块。
- 可以一直添加 RAM 监控区域，直到使用的总块数达到 4。

  （当前可用的块的数目 （和大小）显示于列表下方。）

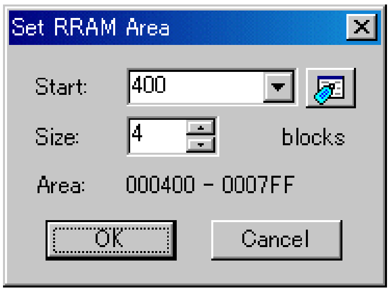
7.1.2.1 更改 RAM 监控区域

RAM 监控区域的起始地址和大小可以更改。

- 从对话框更改

  从 RAM 监控区域列表中选择要更改的 RAM 监控区域，并在其上双击。

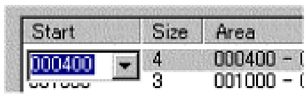
  此时将出现如下所示的 [Set RRAM Area] （设置 RRAM 区域）对话框。在此对话框的 [Start] （起始）和 [Size] （大小）字段中指定 RAM 监控区域的起始地址和大小 （通过块数指定）。



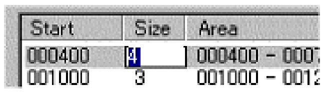
- 直接在窗口中更改

  从 RAM 监控区域列表中选择要更改的 RAM 监控区域，并在其 [Start] （起始）或 [Size] （大小）显示列中再次单击。

  在随后显示的编辑框中指定要用来更改的新起始地址或大小。按 Enter 键确认输入的更改，或按 Esc 键取消更改。



更改地址



更改大小

7.1.2.2 添加 RAM 监控区域

单击 [Add...] （添加…）按钮。

此时将出现 [Set RRAM Area] （设置 RRAM 区域）对话框。在此对话框的 [Start] （起始）和 [Size] （大小）字段中指定新的 RAM 监控区域的起始地址和大小 （通过块数指定）。

7.1.2.3 删除 RAM 监控区域

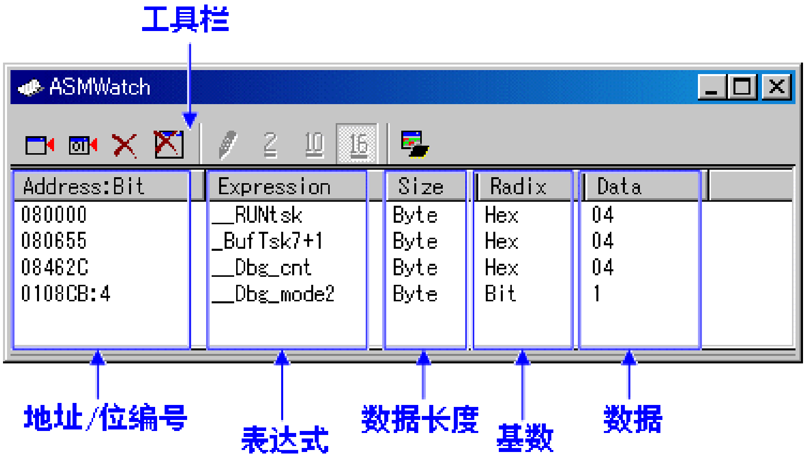
从 RAM 监控区域列表中选择要删除的 RAM 监控区域，然后单击 [Remove] （移除）按钮。

要删除所有 RAM 监控区域，请单击 [Remove All] （全部移除）按钮。

7.2　[ASM Watch] （ASM 监视）窗口

在 [ASM watch] （ASM 监视）窗口中，可以将特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）窗口后，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。

7.2.1　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。

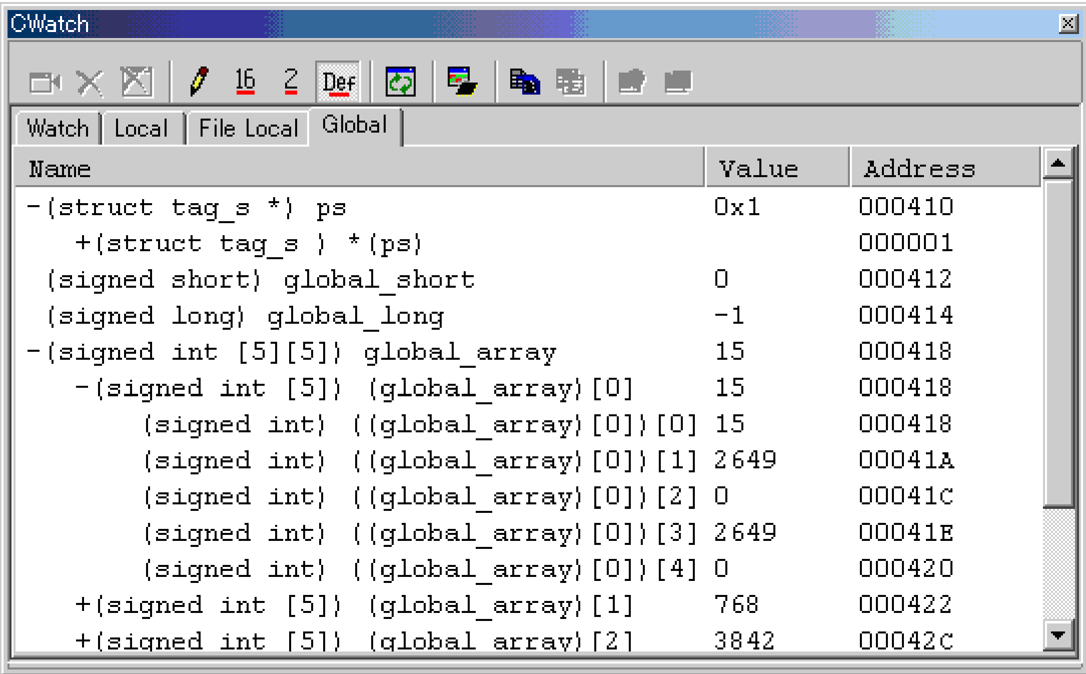
<table>
<tr><th colspan="2">菜单</th><th>功能</th></tr>
<tr><td colspan="2">[Add...]（添加…）</td><td>添加监视点。</td></tr>
<tr><td colspan="2">[Add Bit...]（添加位…）</td><td>添加位标签监视点。</td></tr>
<tr><td colspan="2">[Remove]（移除）</td><td>移除选定的监视点。</td></tr>
<tr><td colspan="2">[Remove All]（全部移除）</td><td>移除全部监视点。</td></tr>
<tr><td colspan="2">[Set...]（设置…）</td><td>对选定监视点设置新数据。</td></tr>
<tr><td rowspan="3">[Radix]（基数）</td><td>[Bin]（二进制）</td><td>以二进制显示。</td></tr>
<tr><td>[Dec]（十进制）</td><td>以十进制显示。</td></tr>
<tr><td>[Hex]（十六进制）</td><td>以十六进制显示。</td></tr>
<tr><td colspan="2">[Refresh]（刷新）</td><td>刷新存储器数据。</td></tr>
<tr><td rowspan="2">[Layout]（布局）</td><td>[Address Area]
（地址区域）</td><td>在显示或不显示地址区域之间切换。</td></tr>
<tr><td>[Size Area]
（大小区域）</td><td>在显示或不显示大小区域之间切换。</td></tr>
<tr><td rowspan="2">[RAM Monitor]
（RAM 监控）</td><td>[Enable RAM Monitor]
（允许 RAM 监控）</td><td>在允许或禁止 RAM 监控功能之间切换。</td></tr>
<tr><td>[Sampling Period...]
（采样周期…）</td><td>设置 RAM 监控采样周期。</td></tr>
<tr><td colspan="2">[Toolbar display]（工具栏显示）</td><td>显示工具栏。</td></tr>
<tr><td colspan="2">[Customize toolbar...]（自定义工具栏…）</td><td>打开工具栏自定义对话框。</td></tr>
<tr><td colspan="2">[Allow Docking]（允许入坞）</td><td>允许窗口入坞。</td></tr>
<tr><td colspan="2">[Hide]（隐藏）</td><td>隐藏窗口。</td></tr>
</table>

7.3 [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。
  — 初始显示基数可以更改。
  — 对于十六进制显示，前导零抑制为可选。
- 任何变量均可注册到 [Watch] （监视）选项卡，这样该变量即可始终显示：
  — 各个工程的已注册内容将分别保存。
  — 同时打开两个或两个以上的 [C watch] （C 监视）窗口时， [Watch] （监视）选项卡中的注册内容将在多个窗口间共享。
  — 变量的引用作用域可从当前作用域、全局作用域以及各文件的作用域中选择。
- 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。
- RAM 监控功能可置于指定变量的地址处。

注意事项

- 以下所列 C 监视点的值无法更改：
  - — 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）、 [File Local] （文件局部变量）和 [Global] （全局）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。

有关 C 变量的更多信息，请参阅 “12.1.3 获取或设置 C 变量”。

7.3.1 扩展菜单

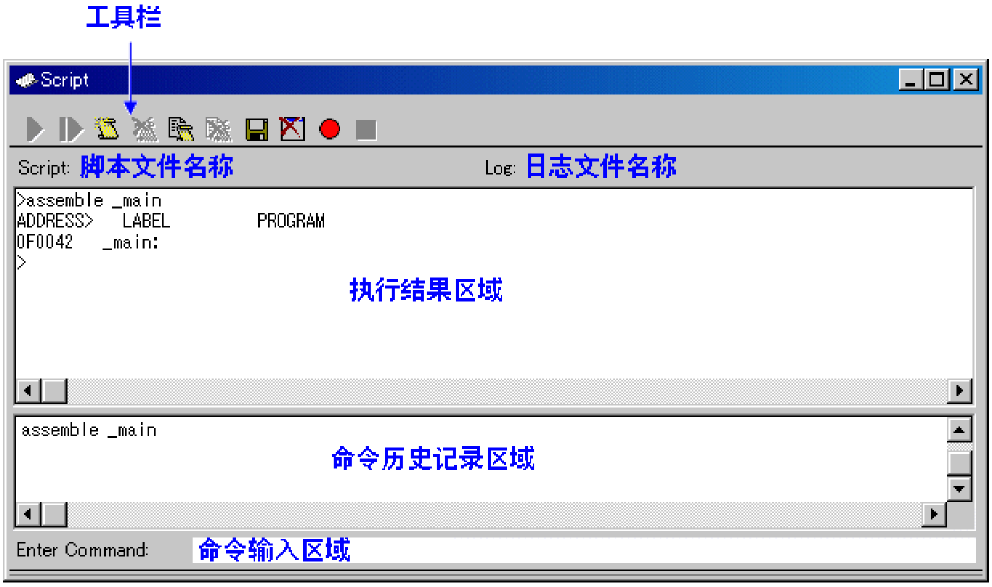
此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

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）	显示默认基数。
	[Toggle(All Variables)]（切换 （全部变量））	更改基数 （切换）。
	[Set initial...] （设置初始基数…）	设置初始基数。
[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。
[Zero suppress in Hex display] （十六进制显示中消零）		十六进制显示中消零。
[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。
	[Arrange a RAM monitor area around this variable]（在此变量旁布置 RAM 监控区域）	在此变量旁布置 RAM 监控区域。
	[Start Recording...] （开始记录…）	开始记录更新的值。
	[Stop Recording] （停止记录）	停止记录更新的值。
[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。
[Copy] （复制）		将选定项目复制到剪贴板。
[Copy All] （全部复制）		将页中所有项目复制到剪贴板。
[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。

7.4 [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 （视图文件）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。

7.4.1 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）		复制选定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。

7.5　[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。

7.5.1　命令按钮

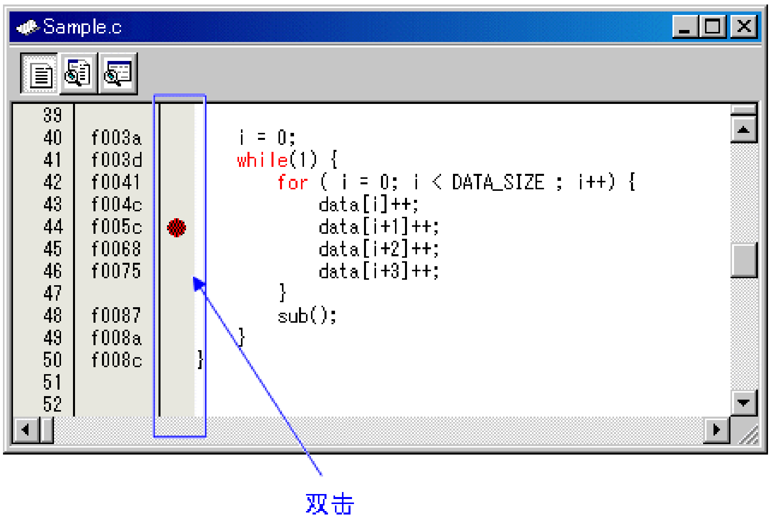
此窗口中的按钮具有以下含义。

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。

7.5.2 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。有关详细信息，请参阅 “12.1.2 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。）

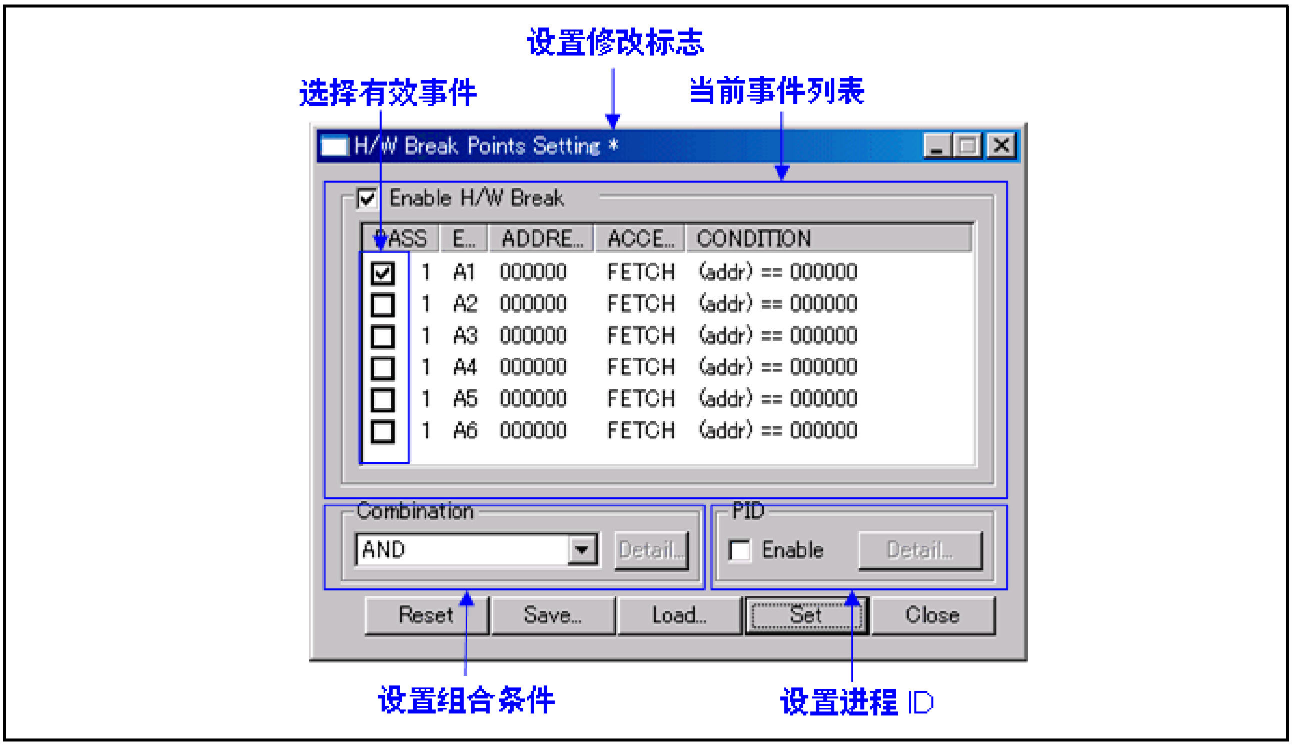


在断点设置区域 （“S/W breakpoints”（软件断点）列）中再次双击即可删除该断点。

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中 [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。

7.6 [H/W Break Point Setting] （硬件断点设置）窗口

[H/W Breakpoint Setting] （硬件断点设置）窗口用于为仿真器设置硬件断点。



- 下列事件可指定为断点事件。如果事件内容发生变化，则事件的标题栏上会标上一个星号 (*)。设置仿真器后不再显示星号 (*)。

  [Fetch] （取）、 [Memory Access] （存储器存取）、 [Bit Access] （位存取）

- 最多可使用两个点处的事件。

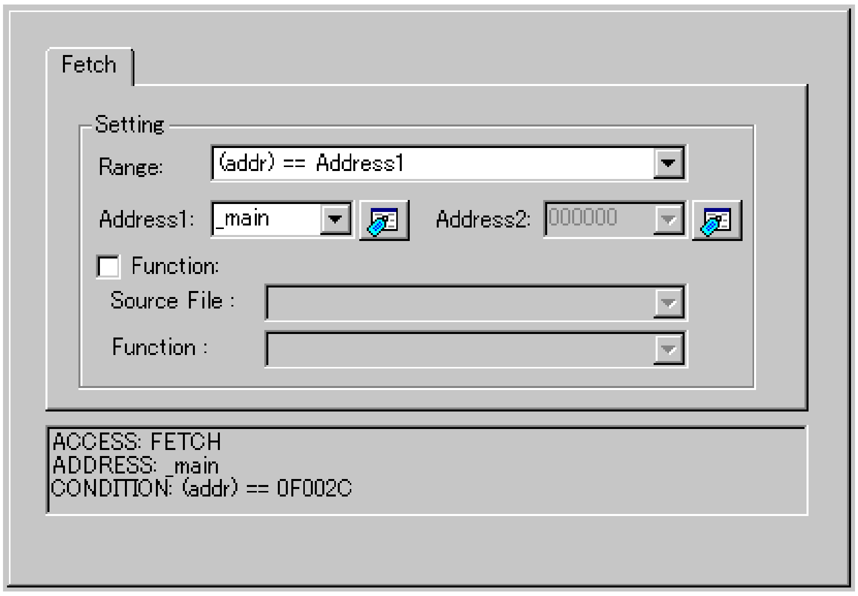
  [H/W Break Point Setting] （硬件断点设置）窗口和 [Trace Point Setting] （跟踪点设置）窗口使用相同的仿真器资源。使用 [Init] （初始化）对话框中的 [MCU] 选项卡可以指定资源用于哪个功能。在此选项卡上，取消选中 [Enable the Trace Point Function] （允许跟踪点功能）复选框。
- 可以通过下列方式之一组合这些事件：
  — 建立所有有效事件时中断 （[AND] （与）条件）
  — 同时建立所有有效事件时中断 （[Simultaneous AND] （同时的 [AND] （与）条件）
  — 建立有效事件之一时暂停 （[OR] （或）条件）
- 启动调试器时硬件断点不起作用。

7.6.1 指定断点事件

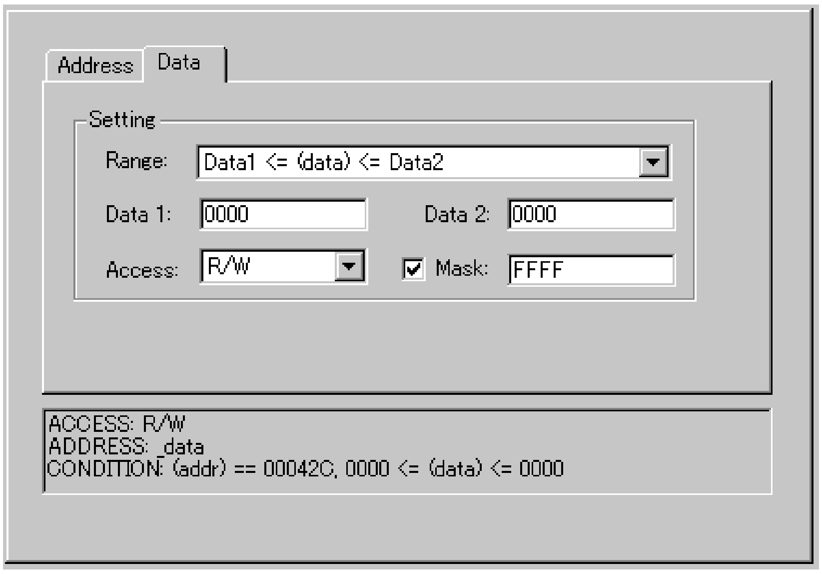
要设置事件，请从 [H/W Break Point Setting] （硬件断点设置）窗口的事件设置区域双击以选中要设置的事件。此时会打开如下对话框。

通过在此对话框中指定 [Event Type] （事件类型），可以设置以下事件。

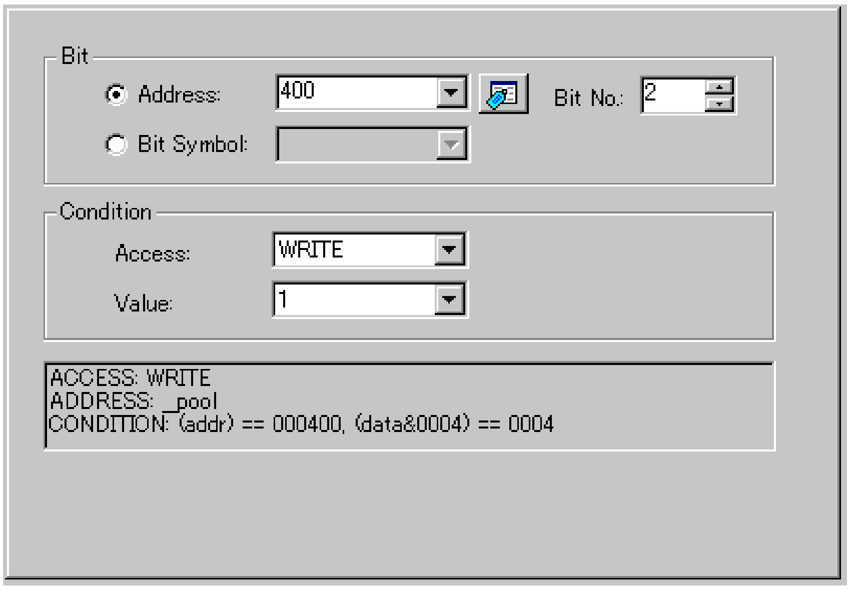
- 选择 [FETCH] （取）时
  取指令的断点。



- 选择 [DATA ACCESS] （数据存取）时
  存储器存取的断点。



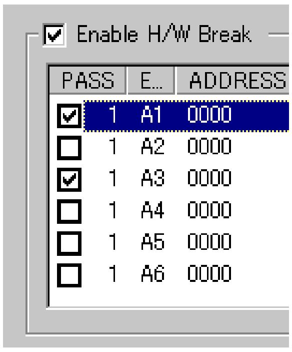
- 选择 [BIT SYMBOL] （位符号）时
  位存取的断点。



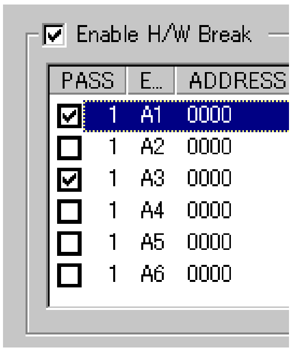
7.6.2 指定组合条件

要指定组合条件，请从组合条件指定区域选择所需条件。

- 选择 [AND] （与）或 [OR] （或）时
  在事件指定区域，可以指定所用事件和该事件的通过计数。要更改通过计数，请在要更改的事件处于选中状态时，单击该事件的通过计数值。



- 选择 [AND (Same Time)] （与 （同时））时
  在事件指定区域，可以指定所用事件。不能指定通过计数。



7.6.3 命令按钮

此窗口中的按钮具有以下含义。

按钮	功能
[Reset] （复位）	删除窗口中显示的内容，并从设置了内容的仿真器加载内容。
[Save...] （保存…）	将窗口中设置的内容保存到文件中。
[Load...] （加载…）	从保存了事件信息的文件加载事件信息。
[Set] （设置）	将窗口中设置的内容发送到仿真器。
[Close] （关闭）	关闭窗口。

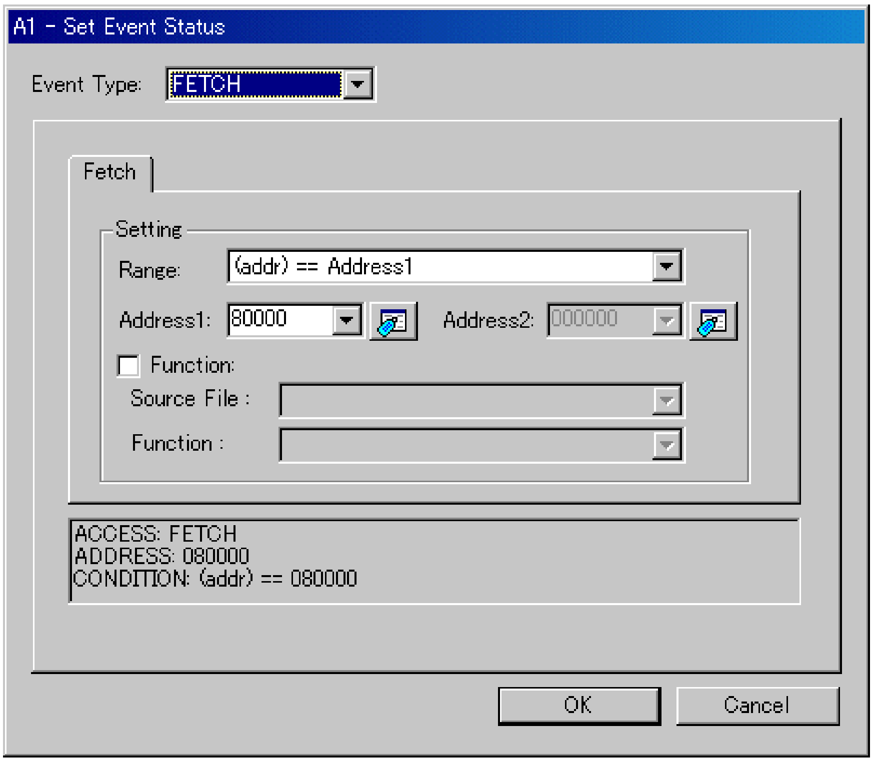
7.6.4 指定事件 （取指令）

要指定取指令事件，请将事件选择对话框的 [Event Type] （事件类型）更改为 [FETCH] （取）。将在从指定地址或指定地址范围中的任意地址取指令时建立事件。

7.6.4.1 从指定地址取指令

设置如下所示。

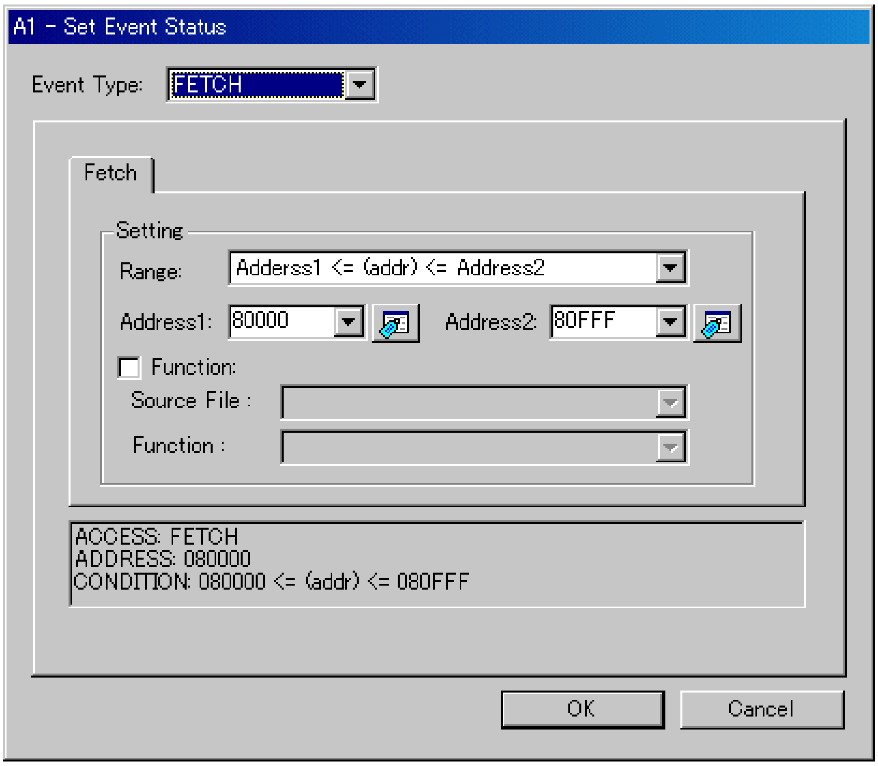
示例）从地址 80000h 取指令



7.6.4.2 从指定地址范围内取指令

设置如下所示。

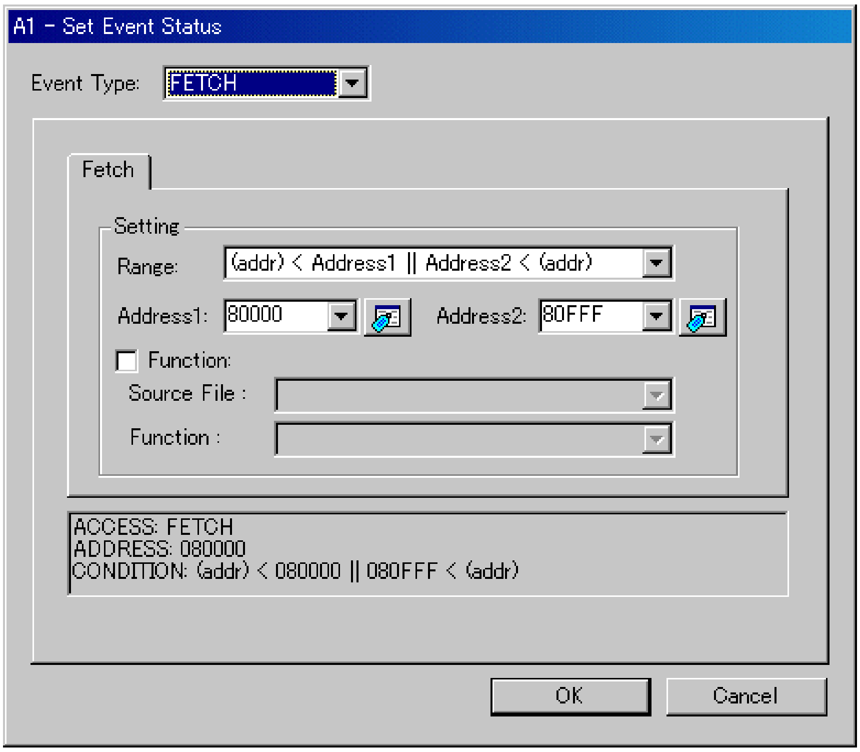
示例）从地址 80000h 至 80FFFh 取指令



7.6.4.3 从指定地址范围以外取指令

设置如下所示。

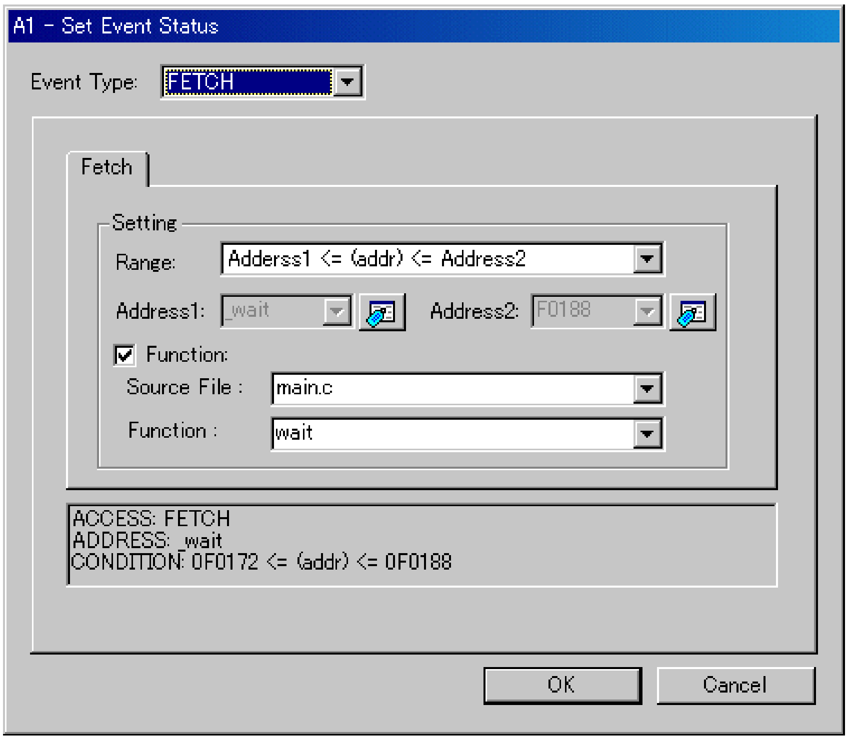
示例）从 80000h 至 80FFFh 范围以外的任意地址取指令



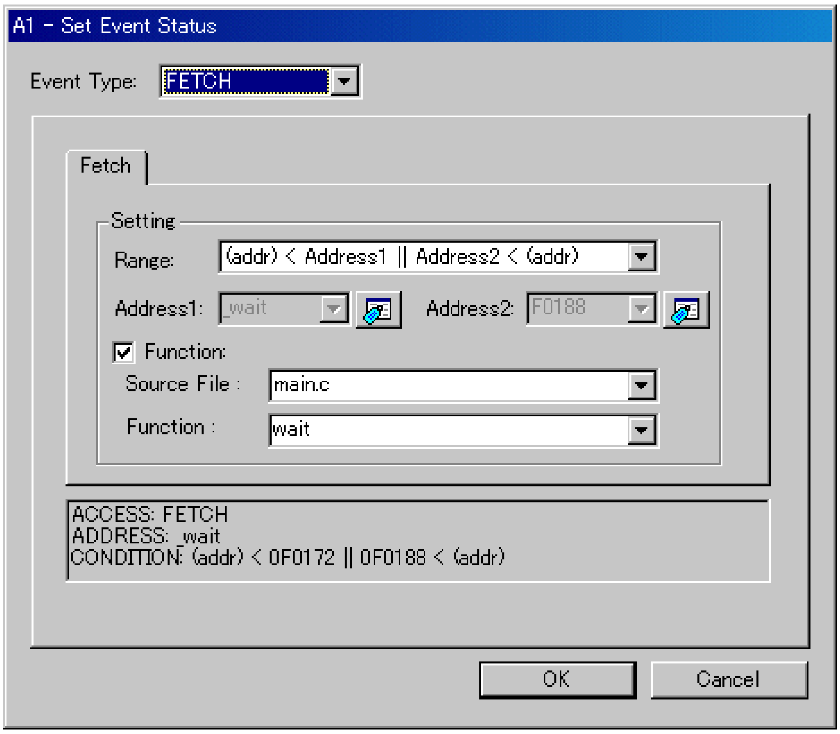
7.6.4.4 进入 / 退出指定函数

设置如下所示。

示例）对名为 “wait”的函数输入一个断点



示例）从名为 “wait”的函数退出



7.6.5　指定事件（存储器存取）

要指定存储器存取事件，请将事件选择对话框的 [Event Type]（事件类型）更改为 “DATA ACCESS”（数据存取）。在指定地址处存取存储器时，或在为指定地址范围设置的条件下存取存储器时建立事件。

7.6.5.1　存储器存取（用于 M32C 的调试器）

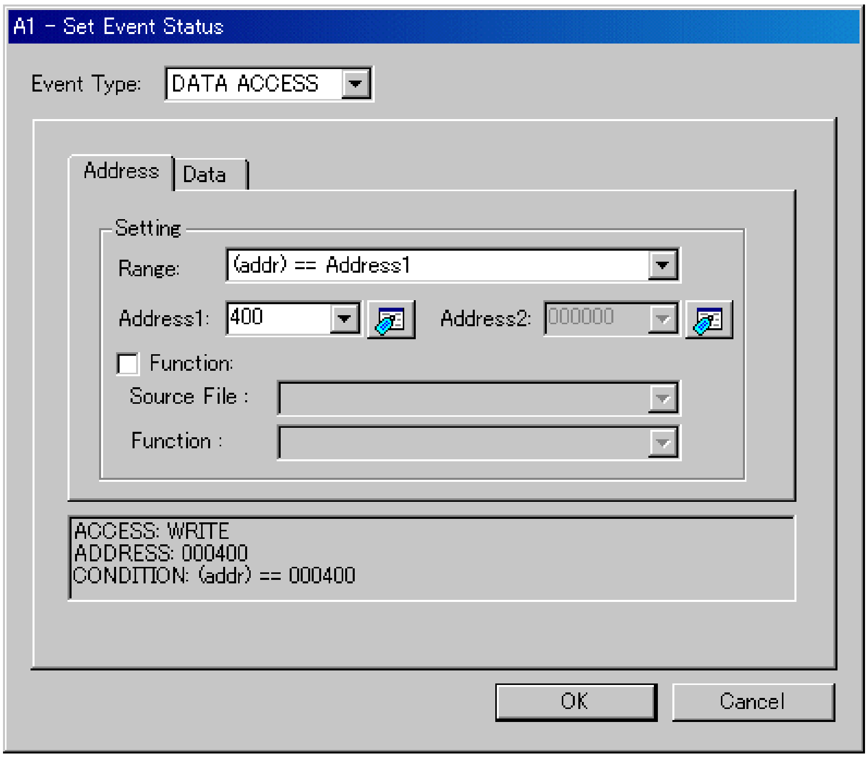
注意事项

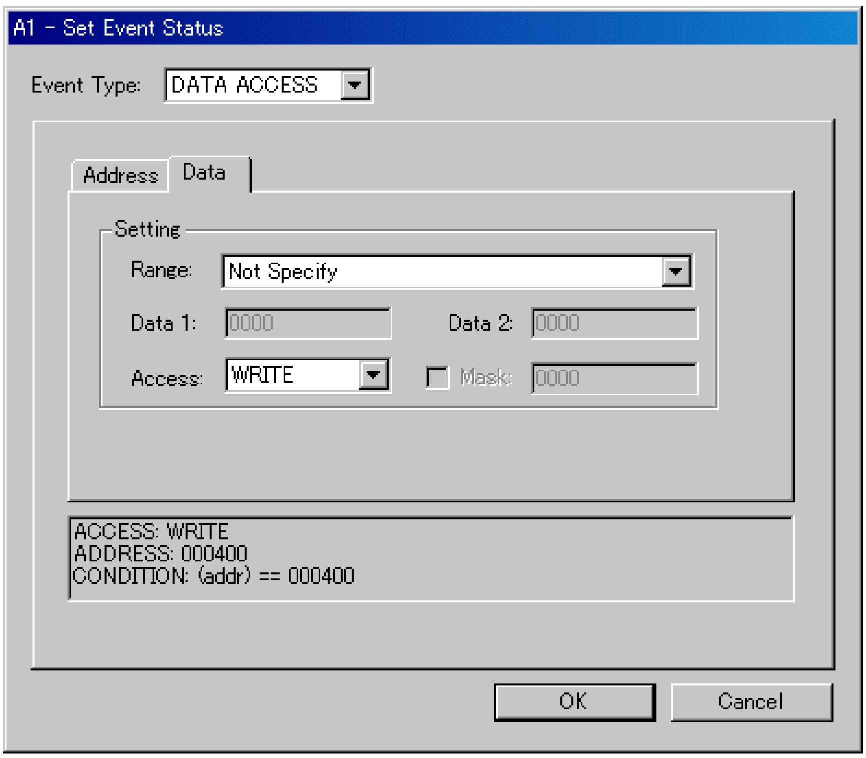
从奇地址以字大小存取数据时不会被检测到。

7.6.5.1.1　　写入 / 读取指定地址

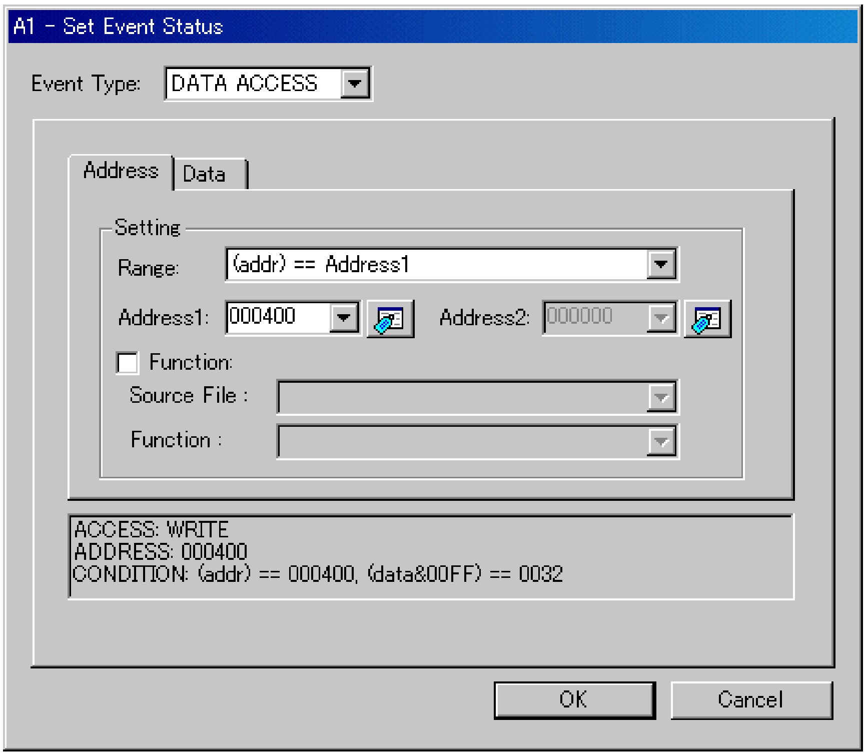
设置如下所示。

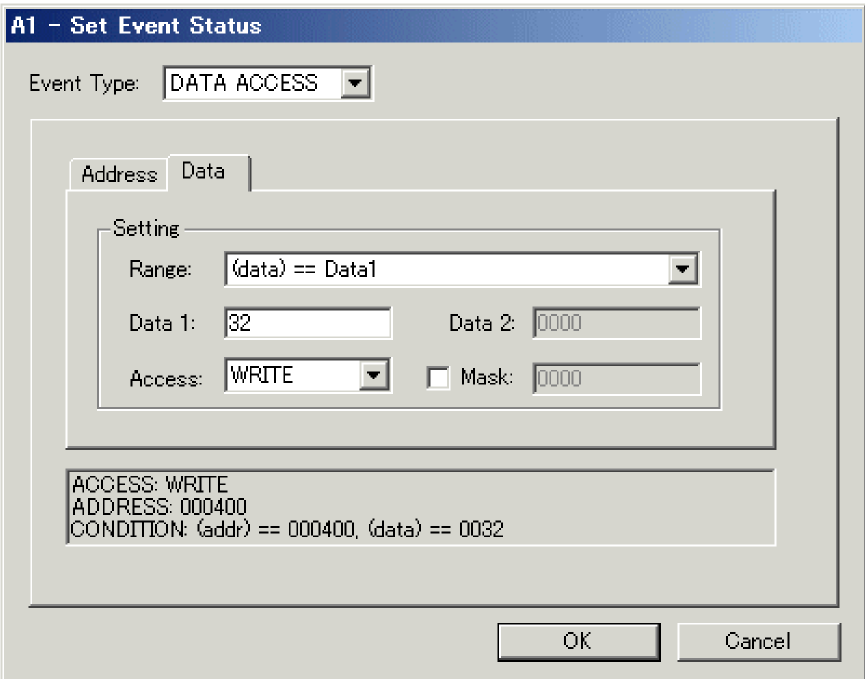
示例）写入偶地址 400h



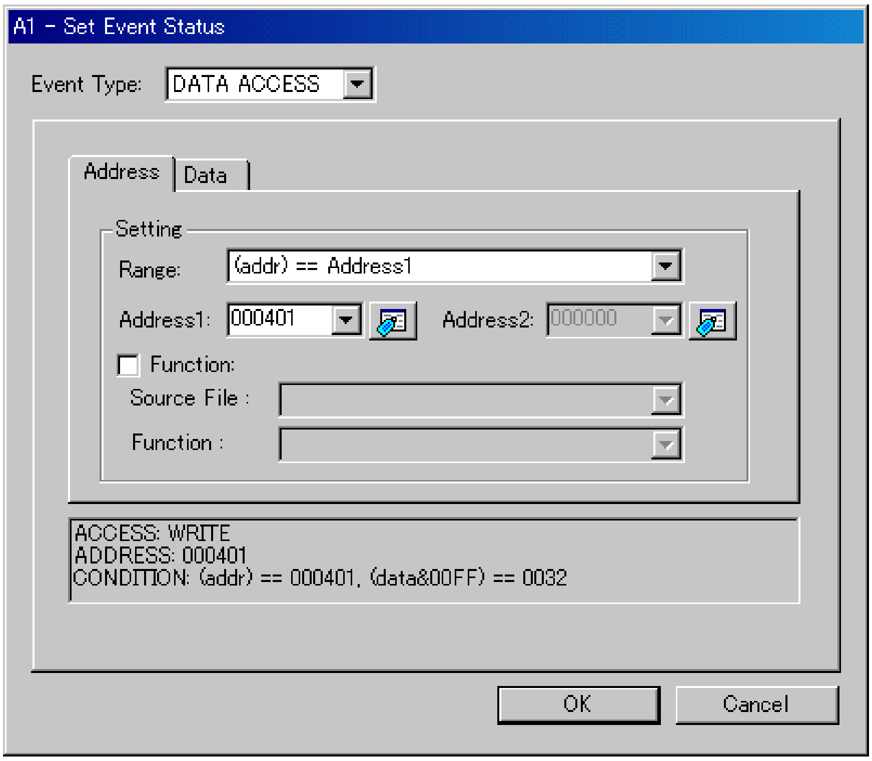


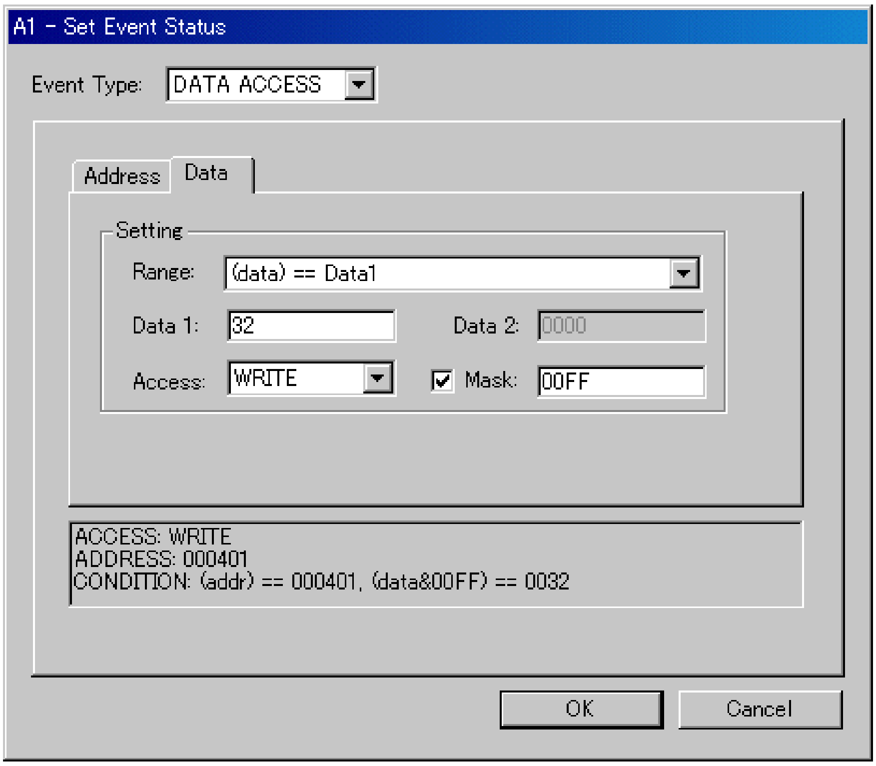
示例）将字节长度的数据 32h 写入偶地址 400h



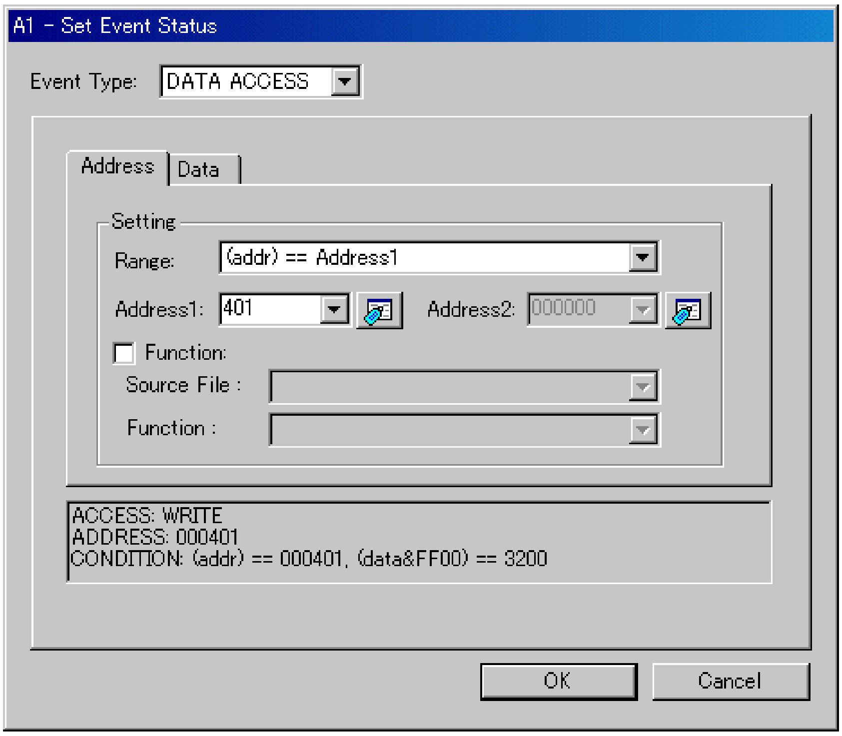


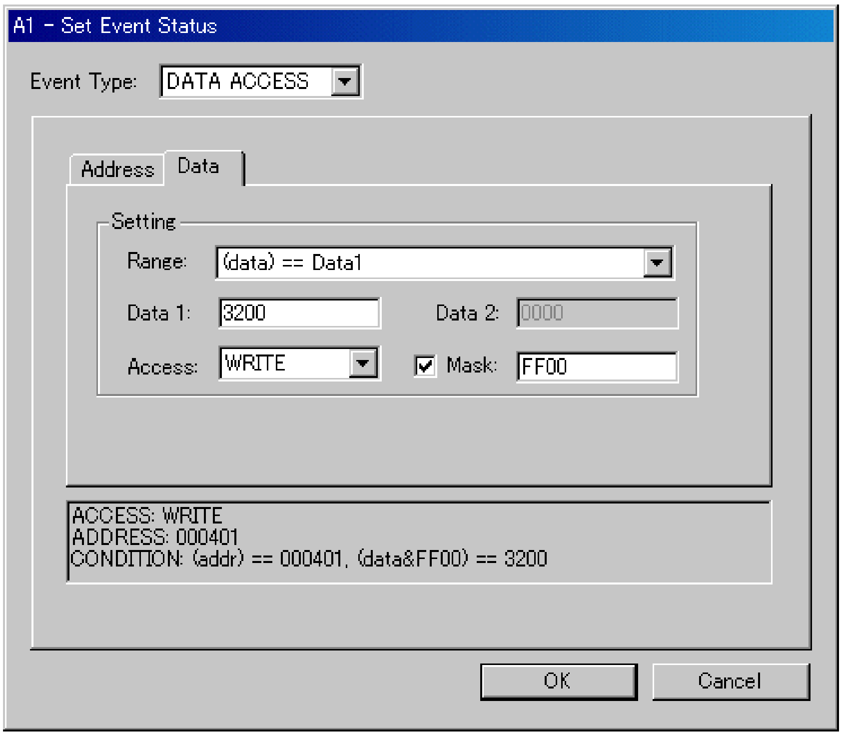
示例）将字节长度的数据 32h 写入奇地址 401h
设置内容因产品和总线宽度而异。
（8 位总线宽度）



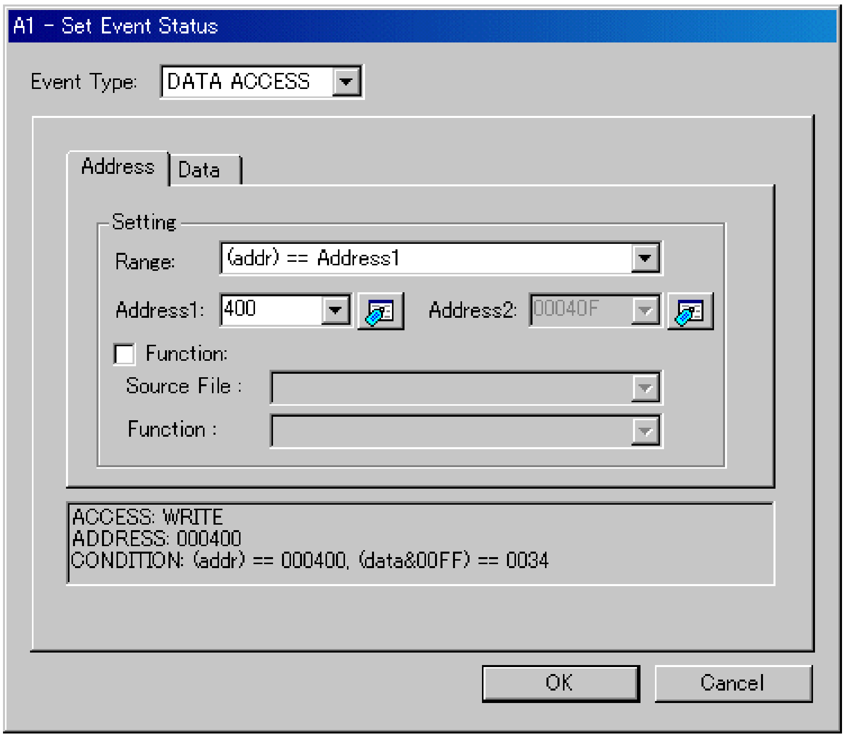


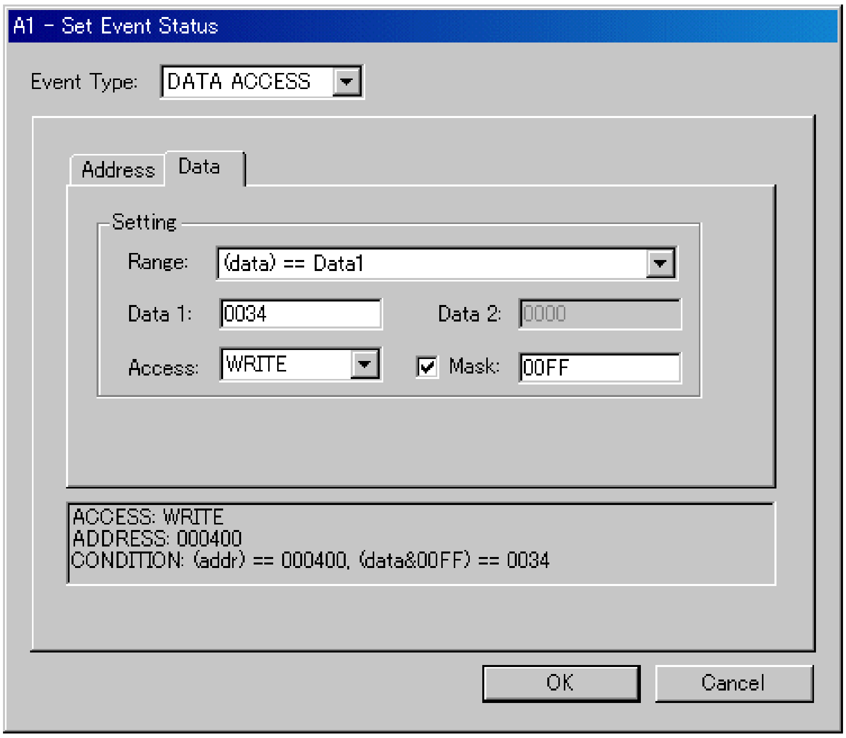
（16 位总线宽度）


A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (addr) == Address1
Address1: 401
Address2: 000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
OK
Cancel

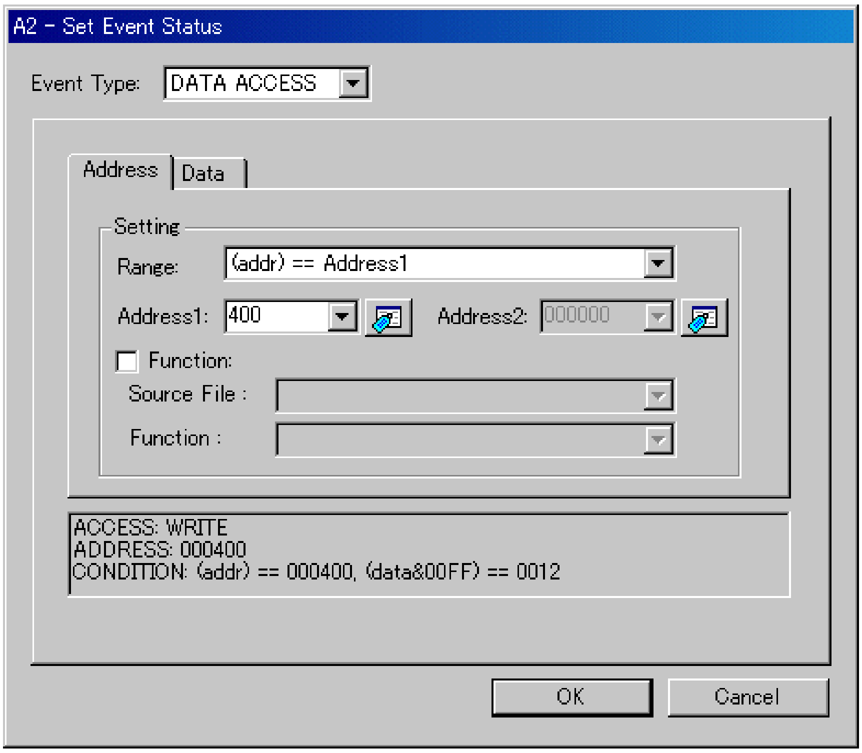

A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (data) == Data1
Data 1: 3200
Data 2: 0000
Access: WRITE
Mask: FF00
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
OK
Cancel

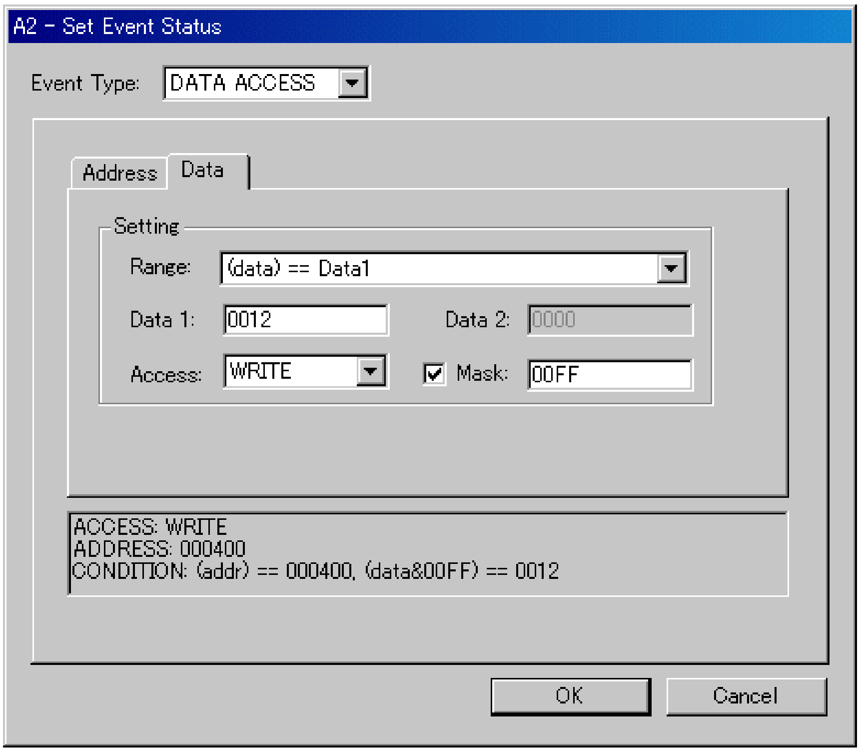
示例）将字长度的数据 1234h 写入偶地址 400h
设置内容因产品和总线宽度而异。
（8 位总线宽度）
请指定第一个事件点和第二个事件点的逻辑与，作为组合条件。
第一个事件点



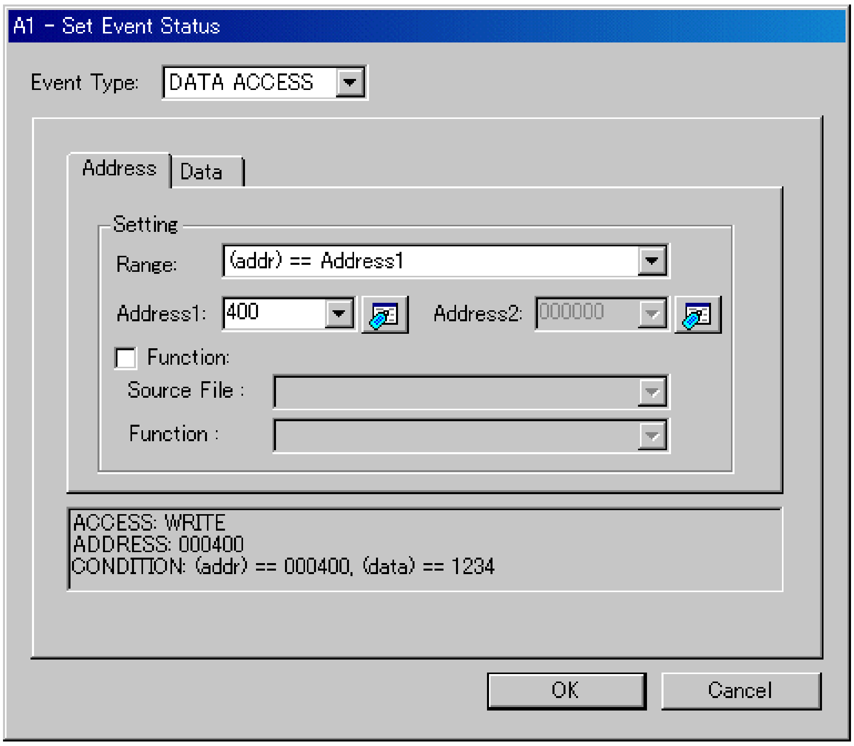


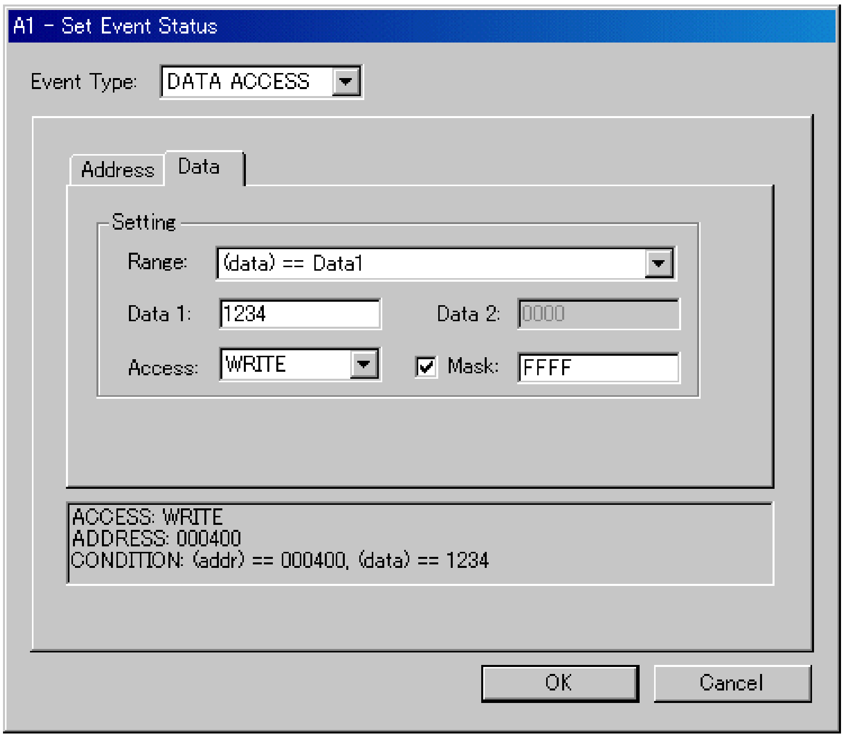
第二个事件点


A2 - Set Event Status
Event Type:
DATA ACCESS
Address
Data
Setting
Range:
(addr) == Address1
Address1:
400
Address2:
000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
OK
Cancel


A2 - Set Event Status
Event Type:
DATA ACCESS
Address
Data
Setting
Range:
(data) == Data1
Data 1:
0012
Data 2:
0000
Access:
WRITE
Mask:
00FF
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
OK
Cancel

（16 位总线宽度）

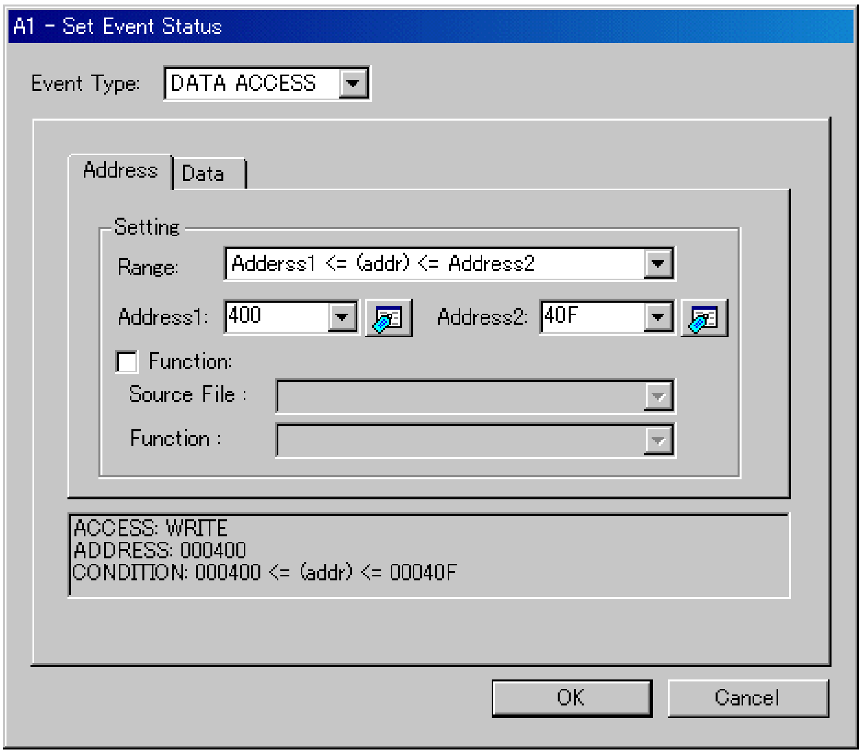

A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (addr) == Address1
Address1: 400
Address2: 000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
OK
Cancel

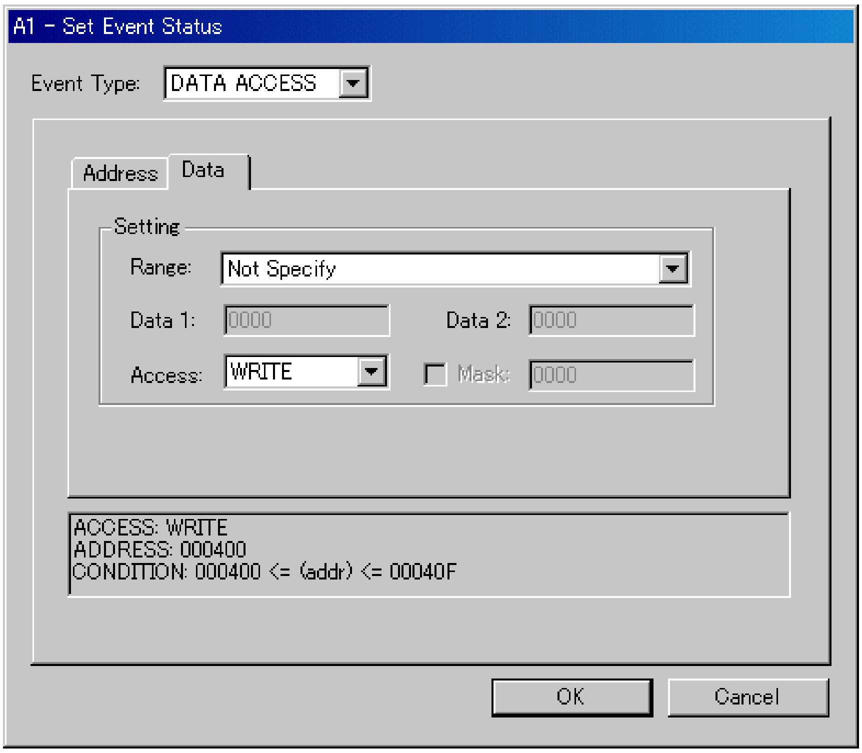

A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (data) == Data1
Data 1: 1234
Data 2: 0000
Access: WRITE
Mask: FFFF
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
OK
Cancel

7.6.5.1.2 从指定地址范围读取 / 写入数据

设置如下所示。

示例）将数据写入 400h 至 40Fh 的地址范围

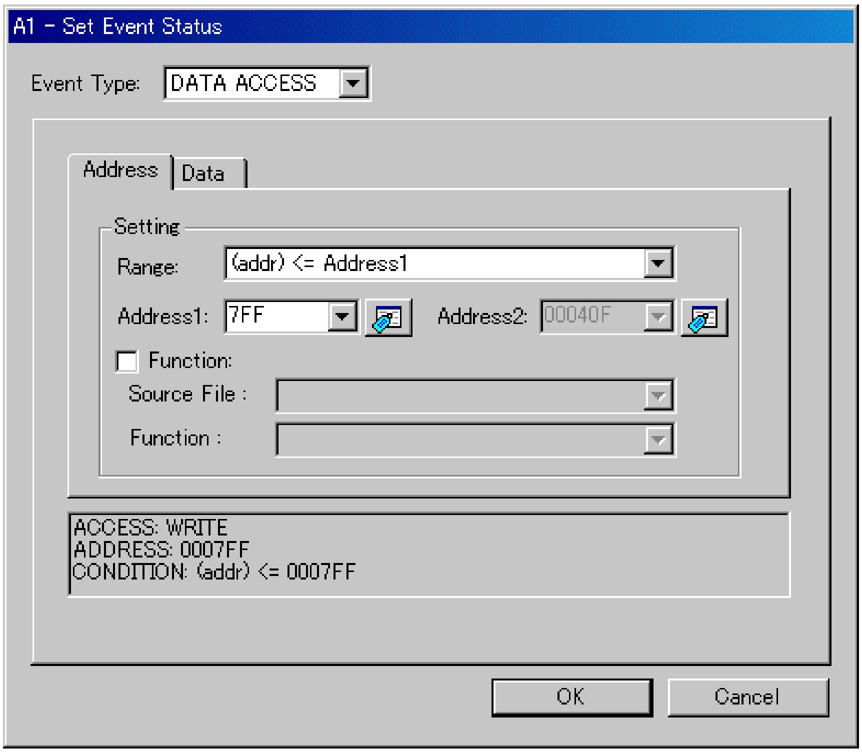


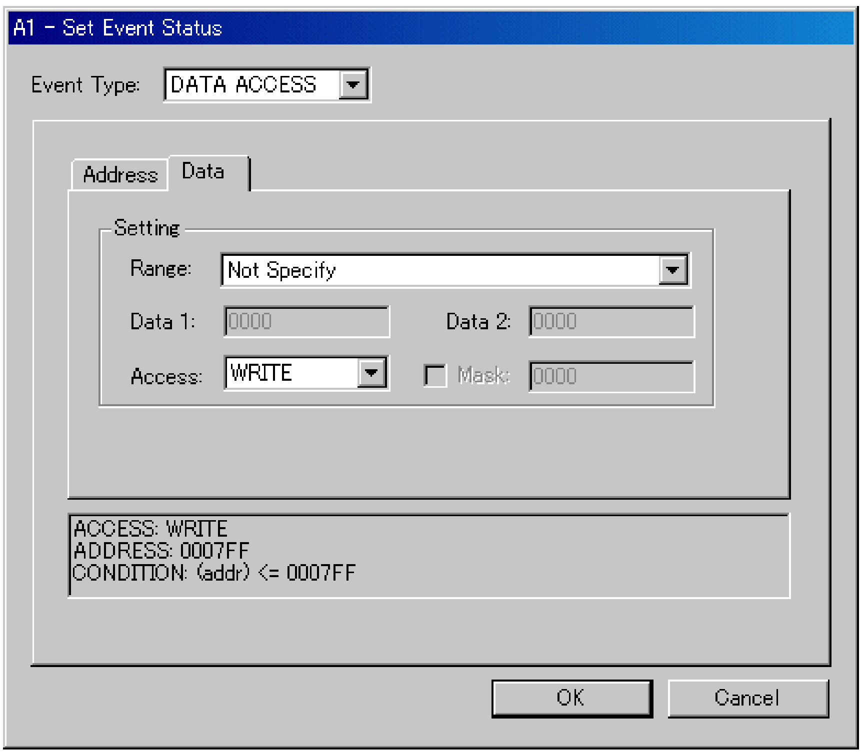


7.6.5.1.3 从指定地址范围以外读取 / 写入数据

设置如下所示。

示例）将数据写入小于 7FFh 的地址





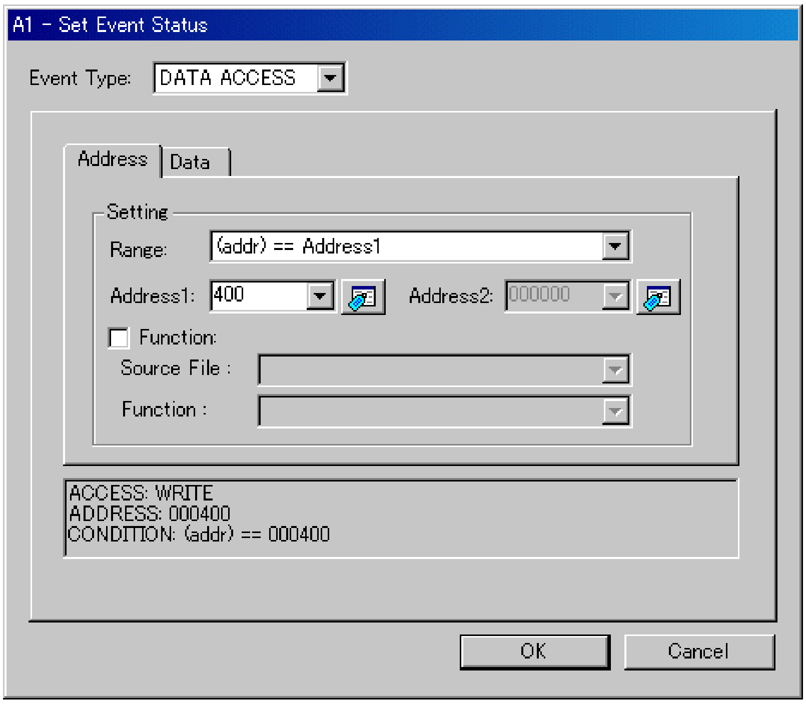
7.6.5.2　　存储器存取 （用于 M16C/R8C 的调试器）

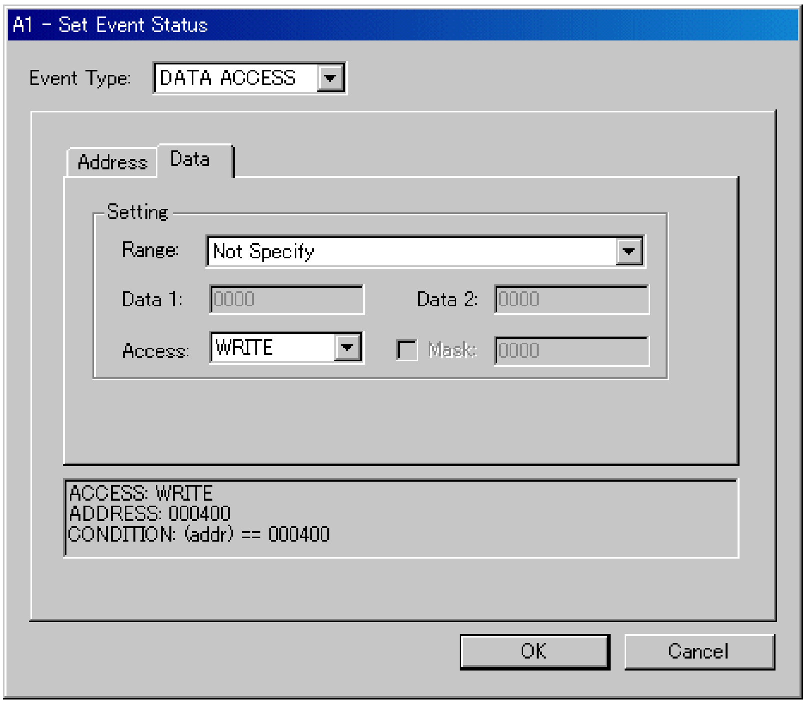
注意事项

从奇地址以字大小存取数据时不会被检测到。

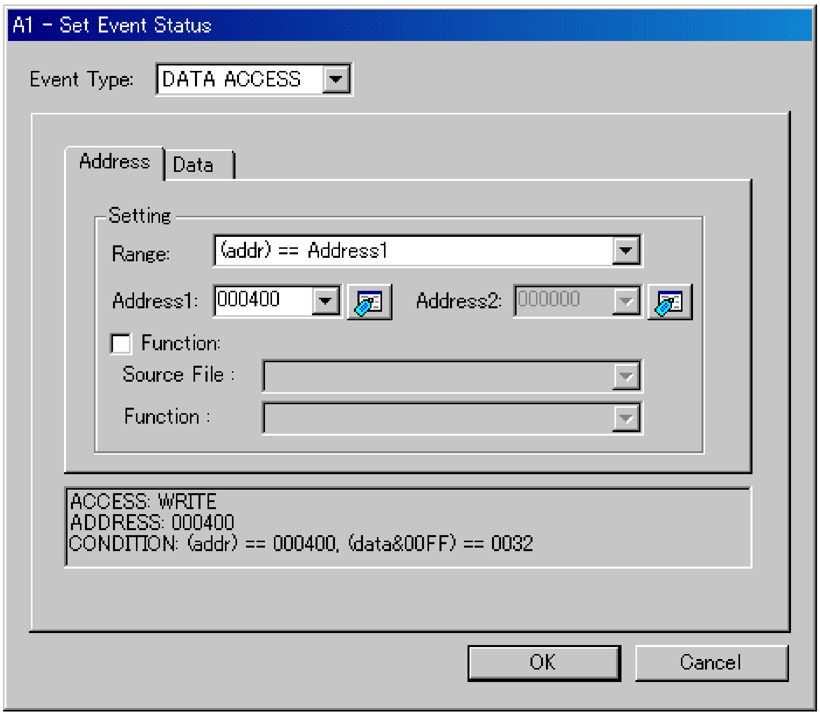
7.6.5.2.1　　写入 / 读取指定地址

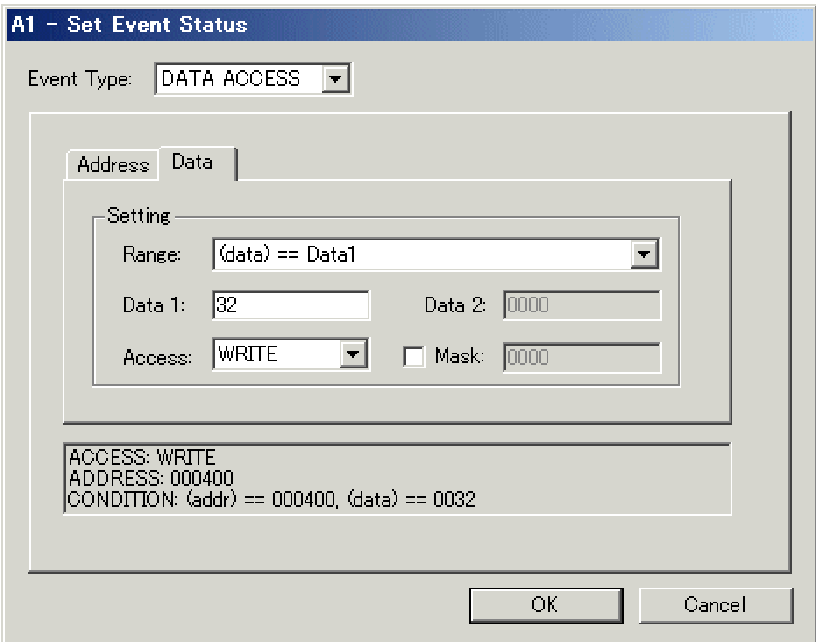
设置如下所示。
示例）写入偶地址 400h



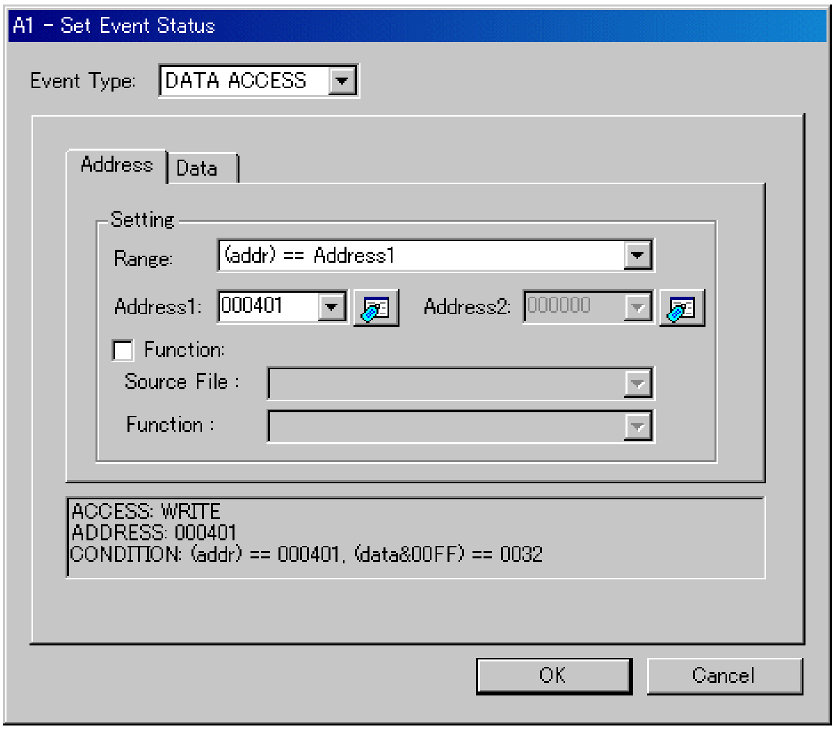


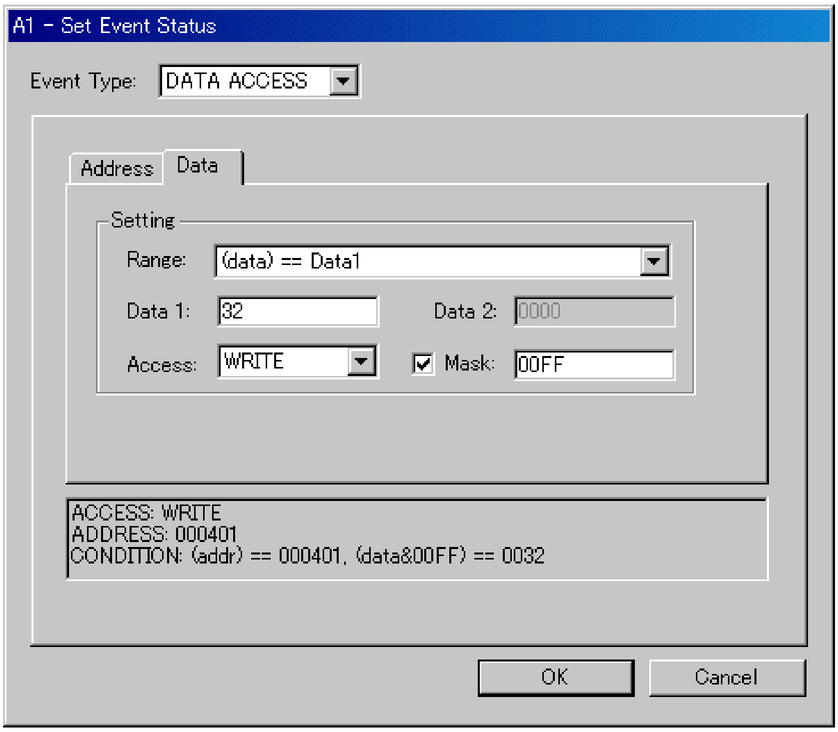
示例）将字节长度的数据 32h 写入偶地址 400h



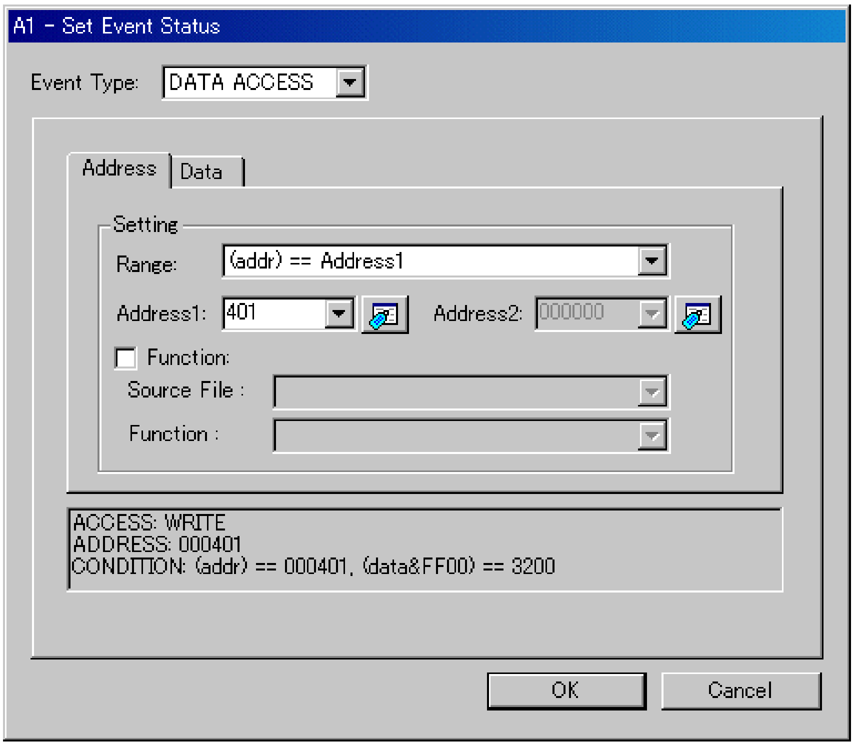


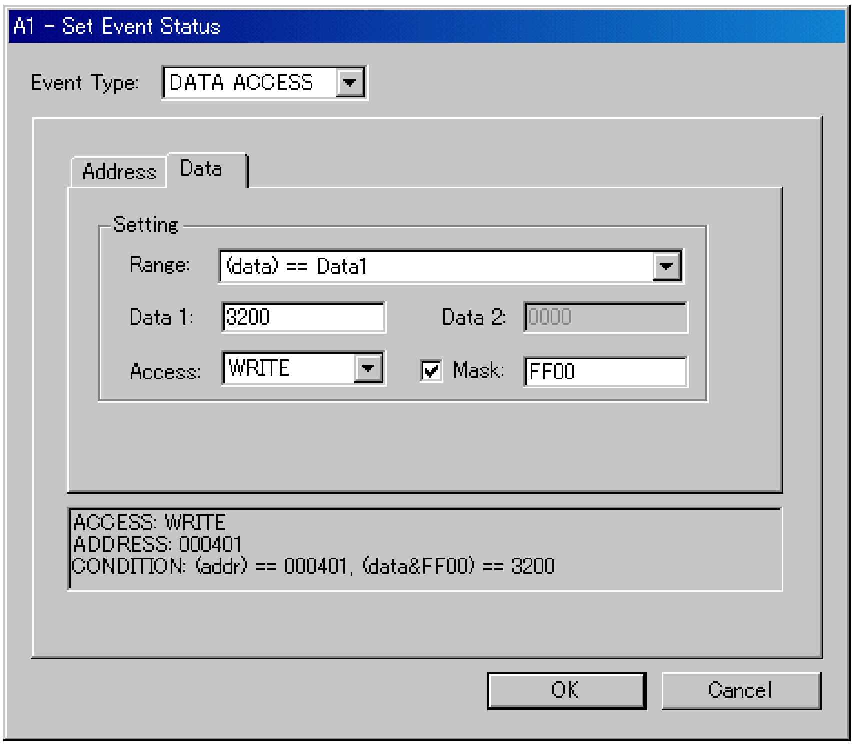
示例）将字节长度的数据 32h 写入奇地址 401h
设置内容因产品和总线宽度而异。
（8 位总线宽度）



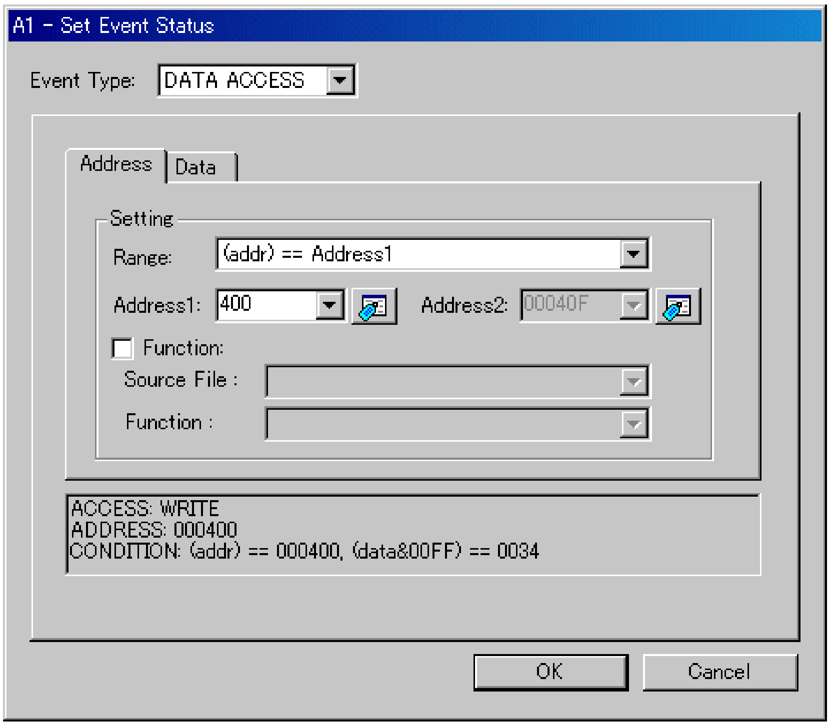


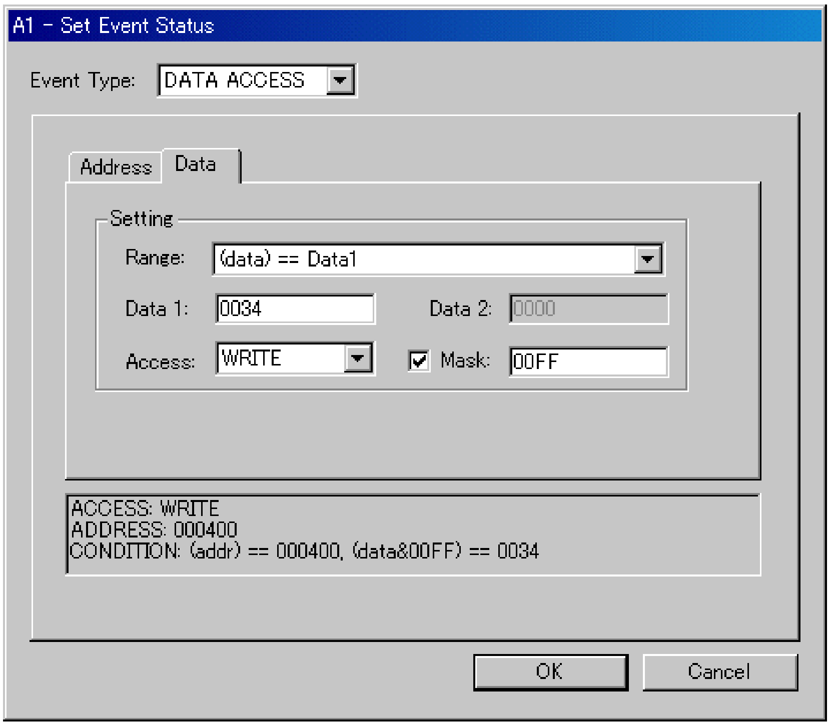
（16 位总线宽度）


A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (addr) == Address1
Address1: 401
Address2: 000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
OK
Cancel

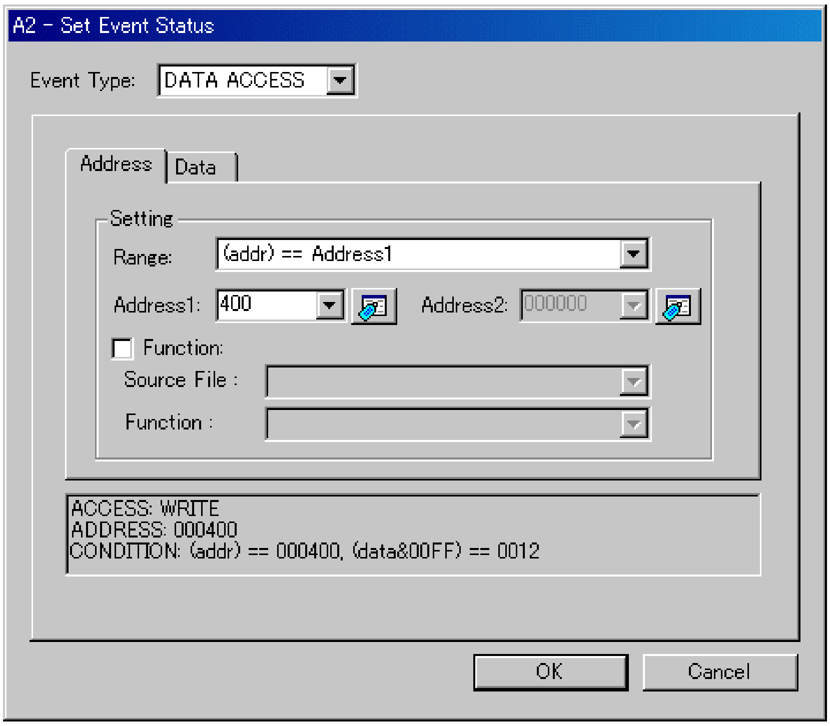

A1 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (data) == Data1
Data 1: 3200
Data 2: 0000
Access: WRITE
Mask: FF00
ACCESS: WRITE
ADDRESS: 000401
CONDITION: (addr) == 000401, (data&FF00) == 3200
OK
Cancel

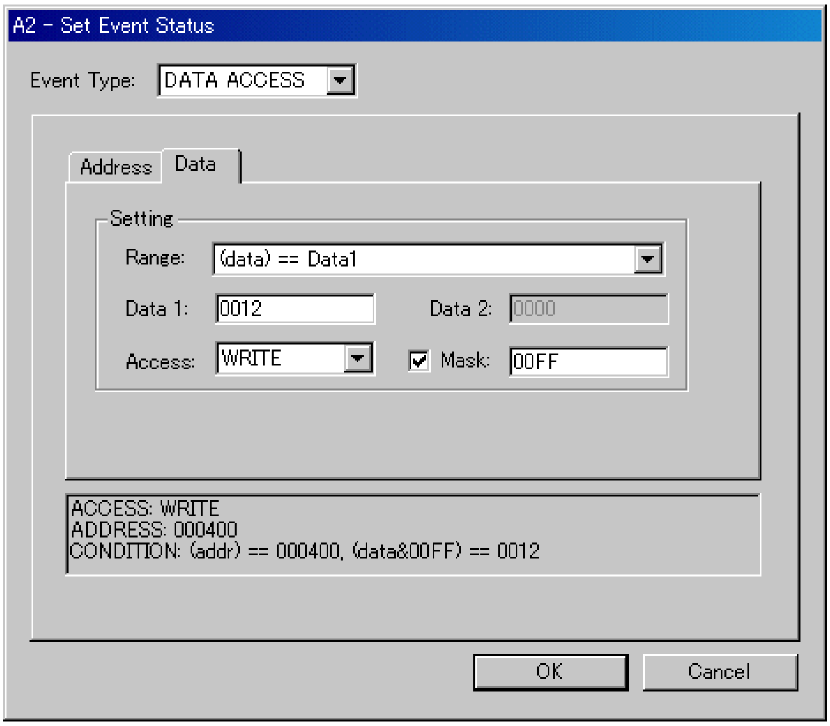
示例）将字长度的数据 1234h 写入偶地址 400h
设置内容因产品和总线宽度而异。
（8 位总线宽度）
请指定第一个事件点和第二个事件点的逻辑与，作为组合条件。
第一个事件点



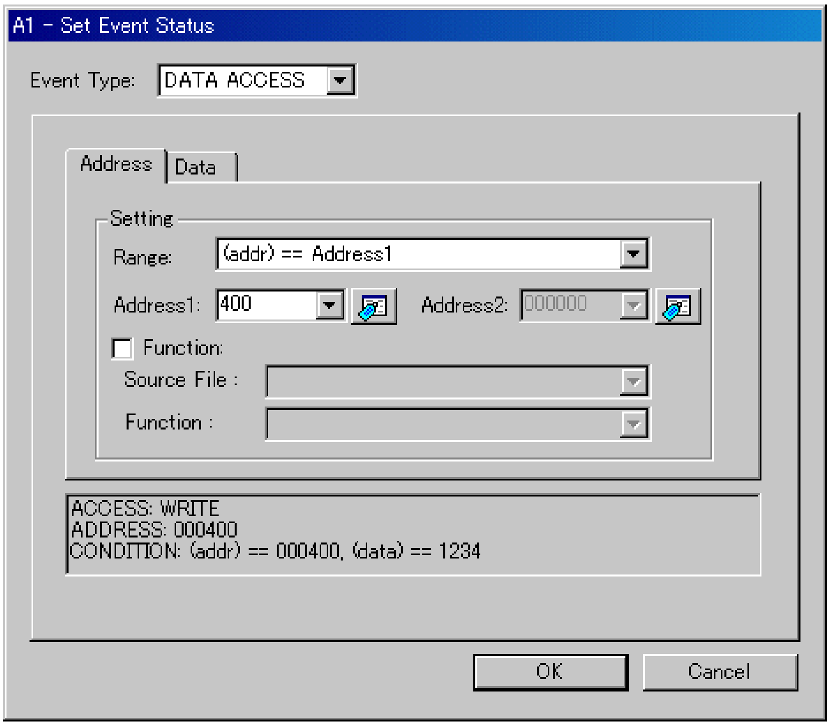


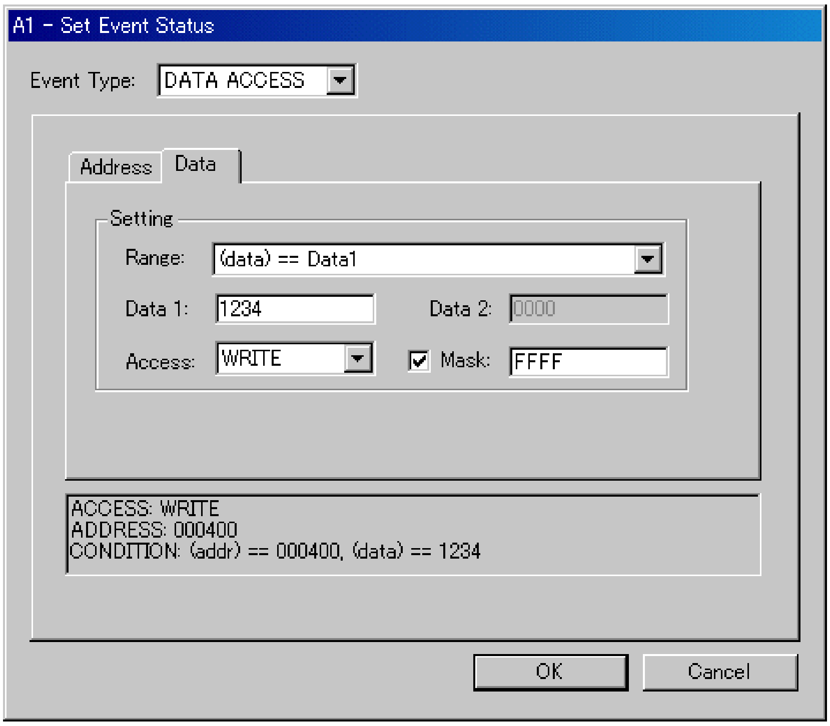
第二个事件点


A2 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (addr) == Address1
Address1: 400
Address2: 000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
OK
Cancel


A2 - Set Event Status
Event Type: DATA ACCESS
Address
Data
Setting
Range: (data) == Data1
Data 1: 0012
Data 2: 0000
Access: WRITE
Mask: 00FF
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data&00FF) == 0012
OK
Cancel

（16 位总线宽度）

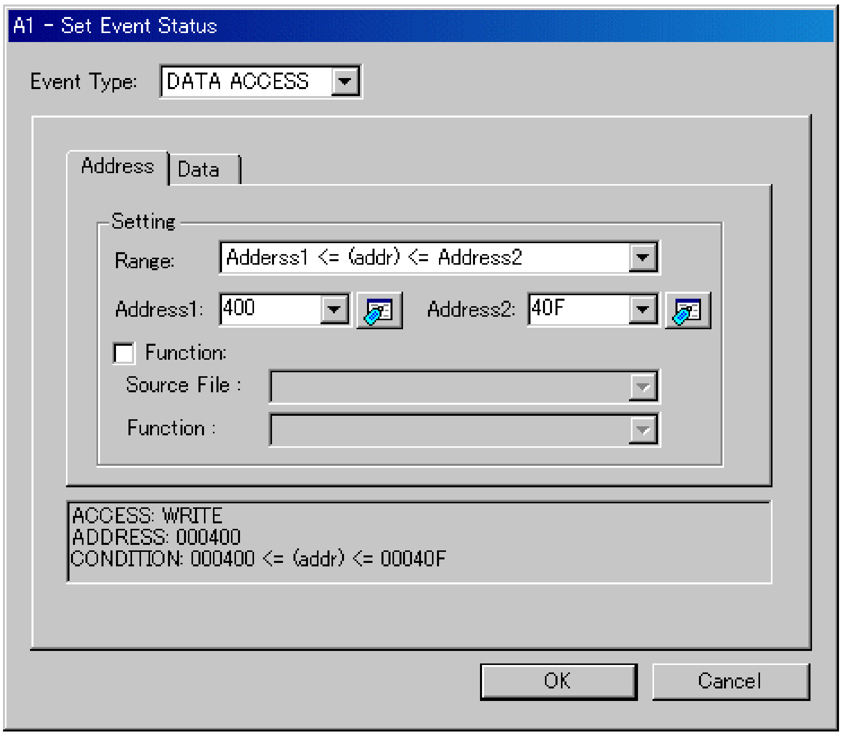

A1 - Set Event Status
Event Type:
DATA ACCESS
Address
Data
Setting
Range:
(addr) == Address1
Address1:
400
Address2:
000000
Function:
Source File :
Function :
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
OK
Cancel

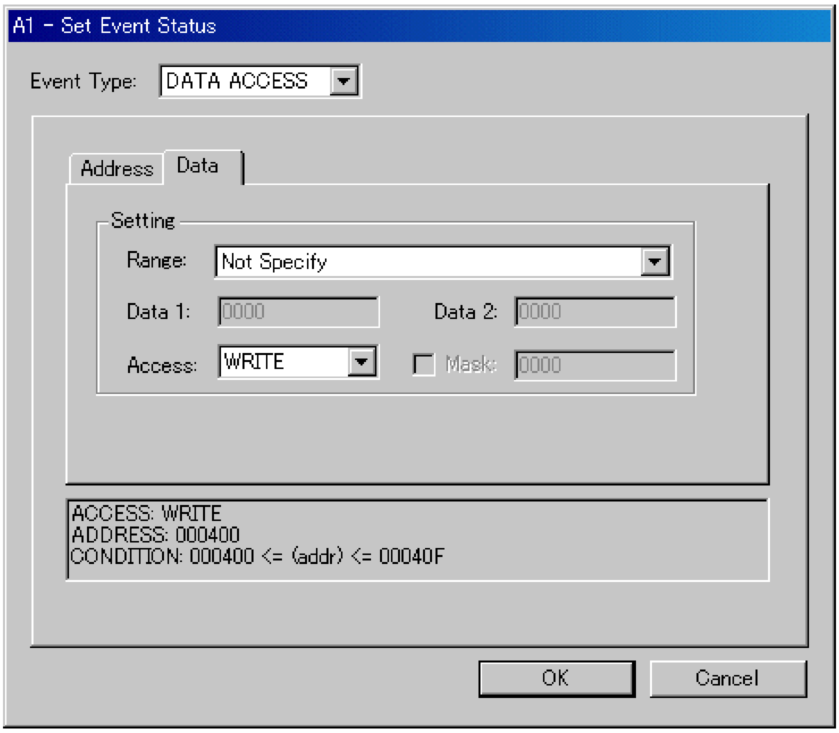

A1 - Set Event Status
Event Type:
DATA ACCESS
Address
Data
Setting
Range:
(data) == Data1
Data 1:
1234
Data 2:
0000
Access:
WRITE
Mask:
FFFF
ACCESS: WRITE
ADDRESS: 000400
CONDITION: (addr) == 000400, (data) == 1234
OK
Cancel

7.6.5.2.2 从指定地址范围读取 / 写入数据

设置如下所示。

示例）将数据写入 400h 至 40Fh 的地址范围

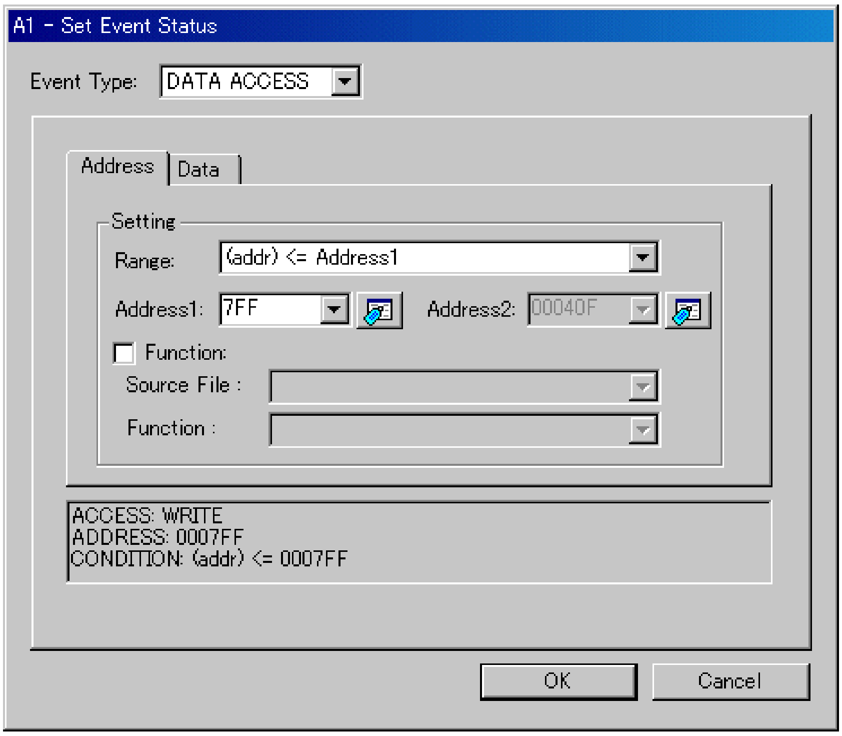


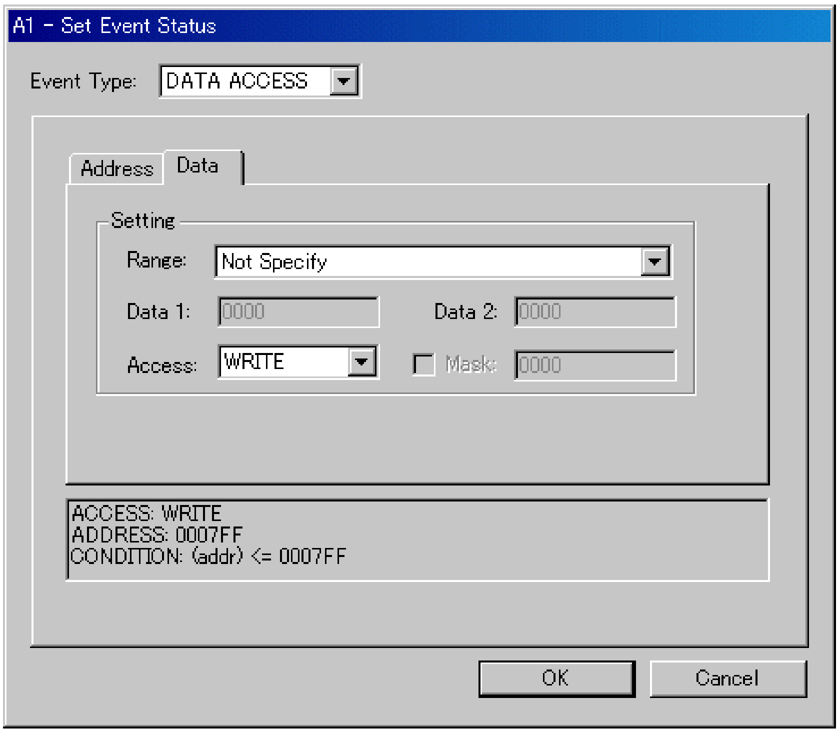


7.6.5.2.3 从指定地址范围以外读取 / 写入数据

设置如下所示。

示例）将数据写入小于 7FFh 的地址





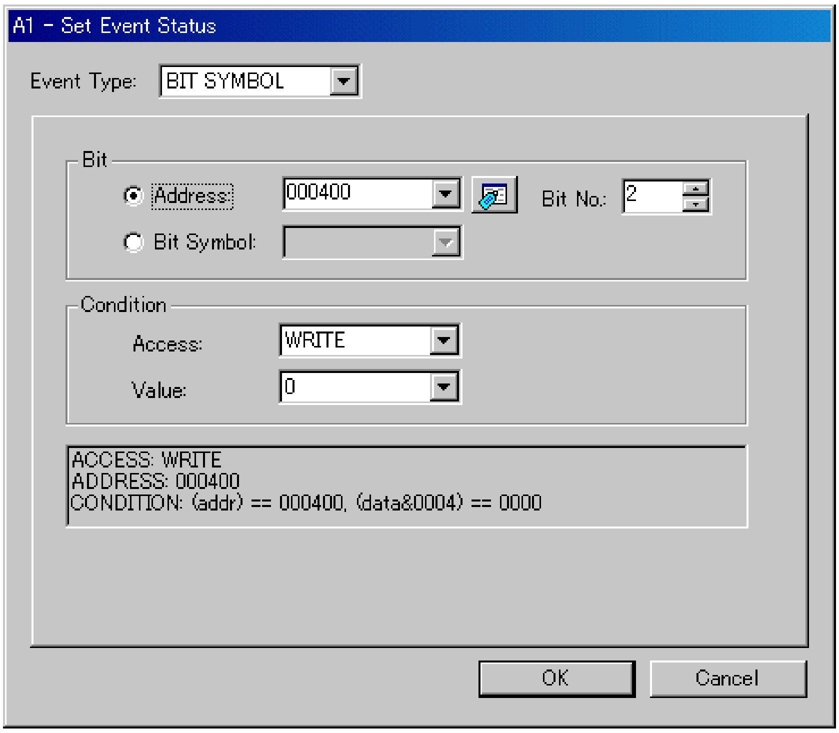
7.6.6　　指定事件 （位存取）

要指定位存取事件，请将事件选择对话框的 [Event Type] （事件类型）更改为 “BIT SYMBOL”（位符号）。在指定条件下对指定地址的指定位或指定位符号进行存取时建立事件。

7.6.6.1　　写入 / 读取指定位

设置如下所示。

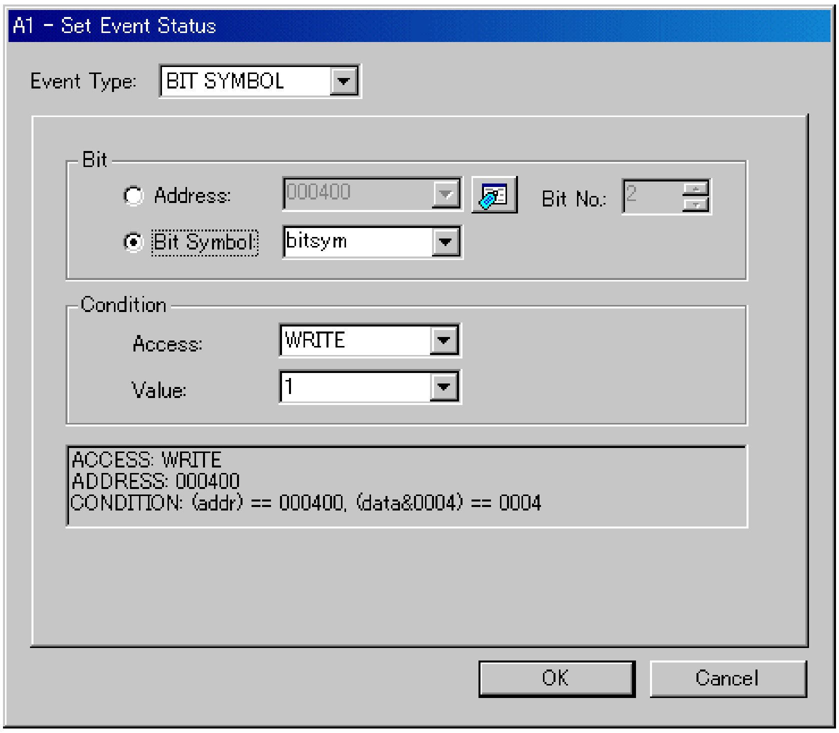
示例）将 “0” 写入地址 400h 的第 2 位



7.6.6.2 写入 / 读取指定位符号

设置如下所示。

示例）将 “1” 写入位符号 “bitsym”



7.6.7　　指定事件组合条件

使用事件设置窗口的 [Combination] （组合）框指定事件的组合条件。

可使用对两个或两个以上的事件的组合。

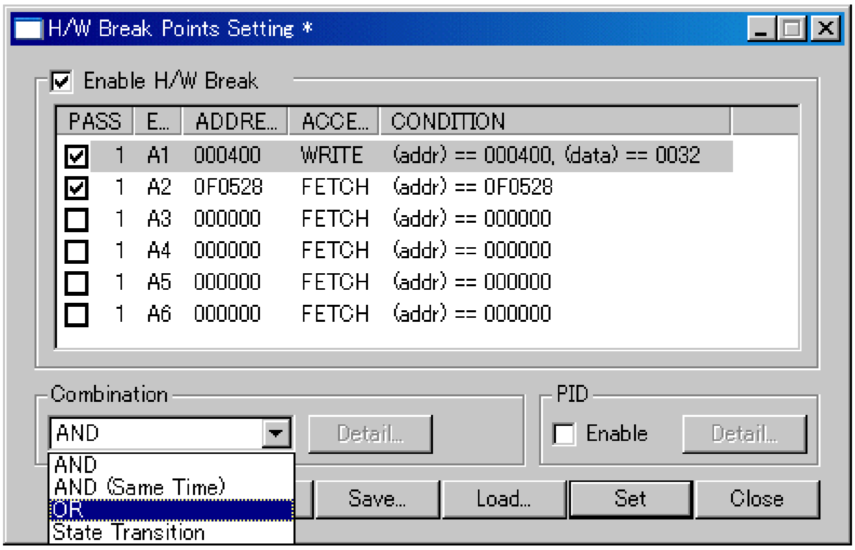
可选择以下组合条件之一。

AND （与）	建立所有指定事件。
AND (Same Time) （与 （同时））	指定的多个事件同时建立。
OR （或）	建立指定事件之一。

可为每一事件指定通过计数 （已通过次数），指定范围为 1-255。如果指定的组合条件为 [AND (Same Time)] （与 （同时）），则无法设置通过计数 （值固定为 1）。

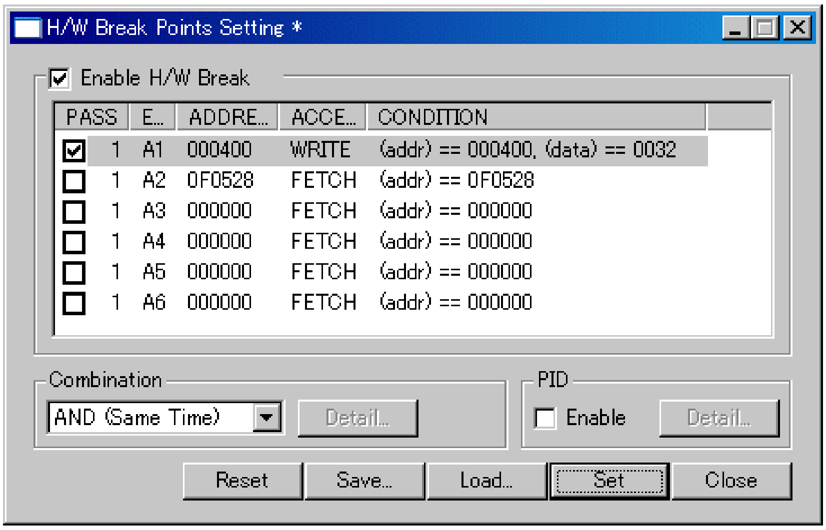
7.6.7.1　　选择 [AND] （与）、 [OR] （或）

将 [Combination] （组合）框改为 “AND”（与）以指定 [AND] （与）作为组合条件，或改为 “OR”（或）以指定 [OR] （或）作为组合条件。接下来，从事件指定区域选取 （打开）要使用的事件，并为该事件指定通过计数。要更改通过计数，请在要更改的事件处于选中状态时，单击该事件的通过计数值。



7.6.7.2 选择 [AND (Same Time)]（与 （同时））

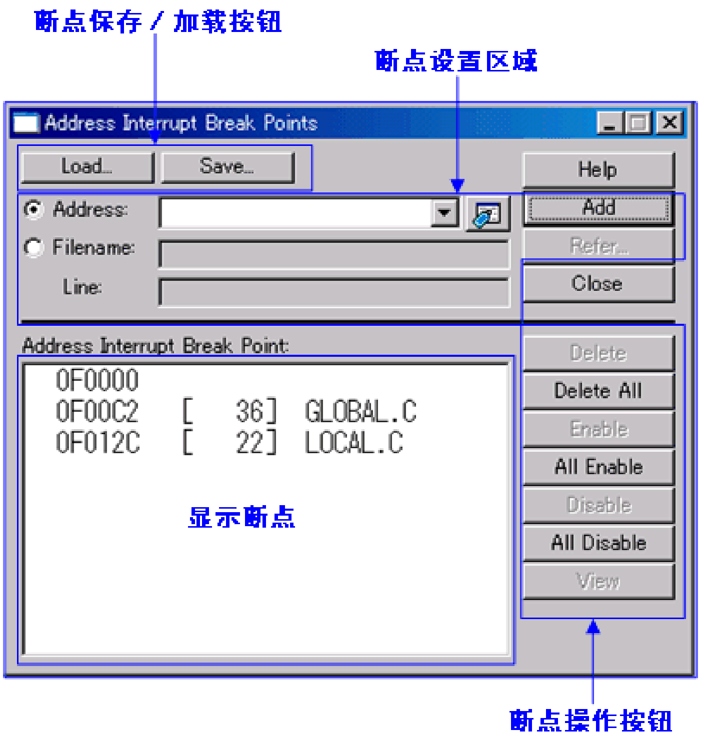
将 [Combination] （组合）框改为 “AND (Same Time)”（与 （同时））。接下来，从事件指定区域选取（打开）要使用的事件。无法指定通过计数 （值固定为 1）。



7.7 [Address Interrupt Break Point Setting] （地址中断断点设置）窗口

通过 [Address Interrupt Break Point Setting] （地址中断断点设置）窗口可以设置地址中断断点。

在指定地址执行指令之前，此功能会立即停止目标程序。此功能是通过使用 MCU 的地址匹配中断实现的。因此只有在用户应用程序中未使用地址匹配中断时，才能使用地址中断断点功能。



- 只有在使用地址中断断点功能时，此窗口才可用。
  使用 [Init] （初始化）对话框中的 [MCU] 选项卡可以指定是否要使用地址中断断点功能。在此选项卡上，选中 [Enable the Address Match Interrupt Break Function] （允许地址匹配中断断点功能）复选框。（详细信息）。
- 可以设置的地址中断断点数因产品而异。
- 可以通过 “Address”（地址）或 “Filename + Line No.”（文件名 + 行号）来指定断点。
- 如果设置了多个断点，则遇到任一断点地址时，程序停止执行 （OR 条件）。
- 可以清除、允许或禁止断点显示区中通过单击选中的断点。也可以通过双击断点来允许和禁止断点。
- 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。
- 如果从文件中加载断点，这些断点会添加到任何现有的断点中。

7.7.1　　命令按钮

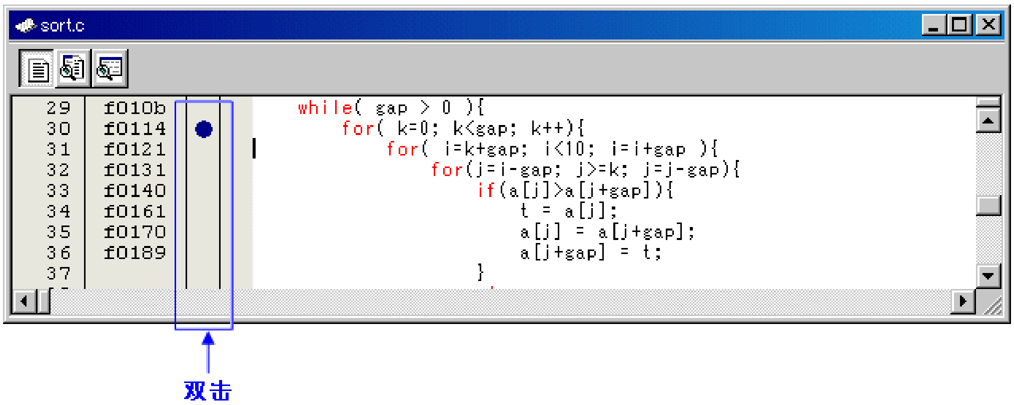
此窗口中的按钮具有以下含义。

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。

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

只有在用户应用程序中未使用地址匹配中断时，才能使用地址中断断点功能。
有关详细信息，请参阅 “1.2.3 地址中断断点功能”。

可以在 [Editor (Source)] （编辑器 （源））窗口中设置断点。若要进行设置，请双击要设置断点的行的断点设置区 （“Address Match Interrupt Break”（地址匹配中断断点）列）。（设置了断点的行上会显示一个蓝色标记。）

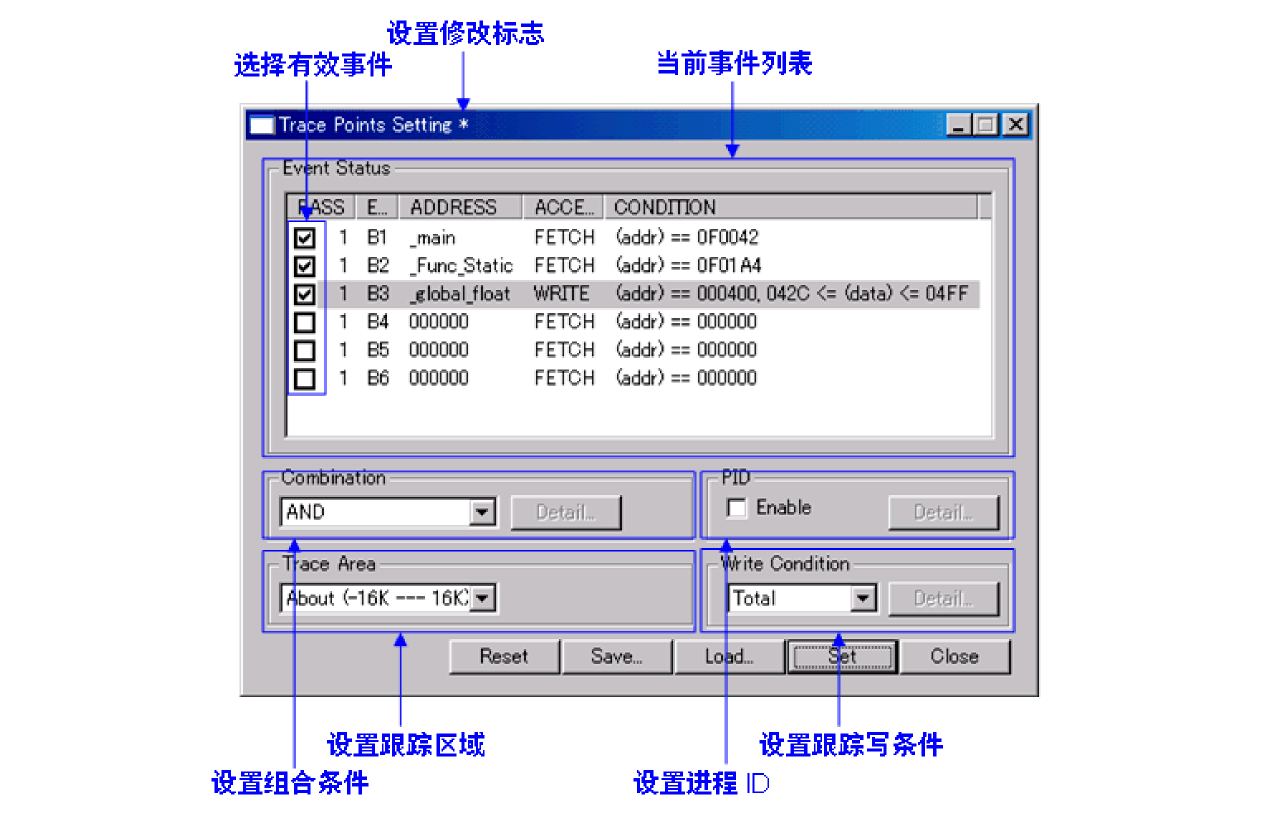


再次双击断点设置区 （“Address Match Interrupt Break”（地址匹配中断断点）列）即可删除断点。

在 [Editor (Source)] （编辑器 （源））窗口中，“Address Match Interrupt Break”（地址匹配中断断点）列默认设置为 “Enable”（允许）显示。若要删除此列，请选择主菜单 - [Edit] （编辑）→ [Define Column Format] （定义列格式）打开对话框，在此对话框中取消选中 [Address Match Interrupt Break] （地址匹配中断断点）复选框。“Address Match Interrupt Break”（地址匹配中断断点）列会从所有的 [Editor (Source)] （编辑器 （源））窗口中删除。在 [Editor (Source)] （编辑器 （源））窗口中选择弹出菜单 - [Columns] （列）→ [Address Match Interrupt Break] （地址匹配中断断点），可以为每个 [Editor (Source)] （编辑器 （源））窗口添加这一列。

7.8 [Trace Point Setting] （跟踪点设置）窗口

[Trace Point Setting] （跟踪点设置）窗口用于设置跟踪点。



- 下列事件可以指定为跟踪事件。如果事件内容发生变化，则事件的标题栏上会标上一个星号 (*)。设置仿真器后不再显示星号 (*)。

  [Fetch] （取）、 [Memory Access] （存储器存取）、 [Bit Access] （位存取）

- 最多可使用两个点处的事件。

  [H/W Break Point Setting] （硬件断点设置）窗口和 [Trace Point Setting] （跟踪点设置）窗口使用相同的仿真器资源。使用 [Init] （初始化）对话框中的 [MCU] 选项卡可以指定资源用于哪个功能。在此选项卡上，选中 [Enable the Trace Point Function] （允许跟踪点功能）复选框。

- 可以通过下列方式之一组合这些事件：

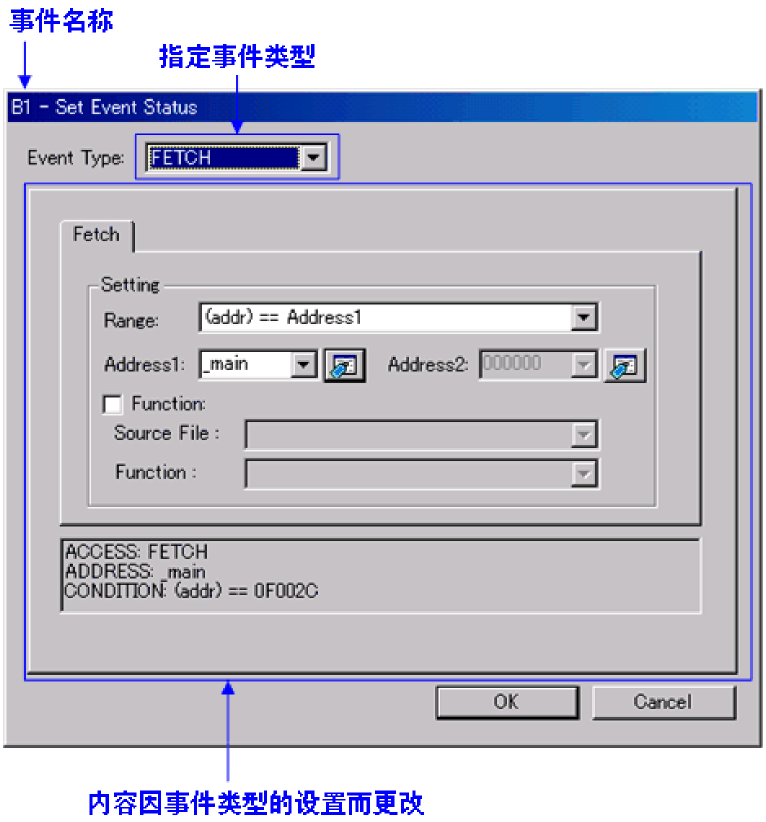
  — 当所有的有效事件都已建立时进行跟踪 （[AND] （与）条件）

  — 当所有有效事件同时建立时进行跟踪 （同时的 [AND] （与）条件）

  — 当其中一个有效事件建立时进行跟踪 （[OR] （或）条件）

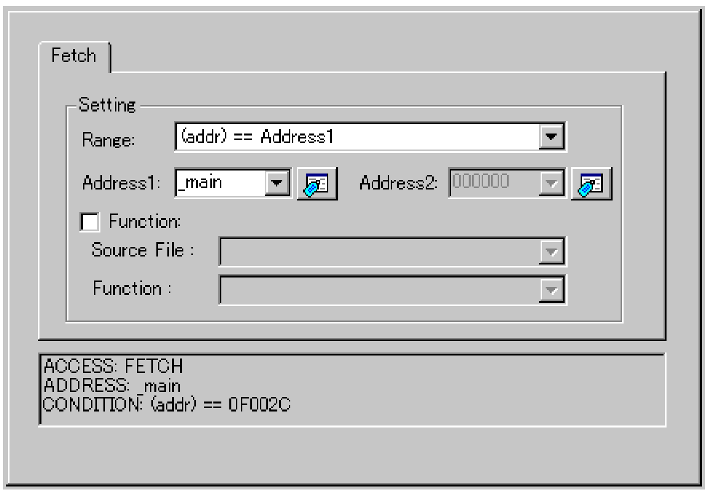
7.8.1 指定跟踪事件

若要设置事件，请从 [Trace Point Setting] （跟踪点设置）窗口的事件设置区双击以选择要设置的事件。此时会打开如下对话框。

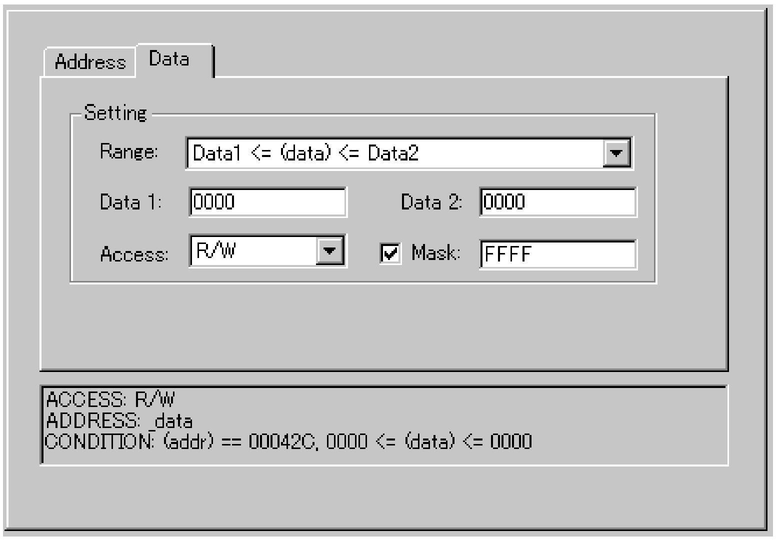


通过在此对话框中指定 [Event Type] （事件类型），可以设置以下事件。

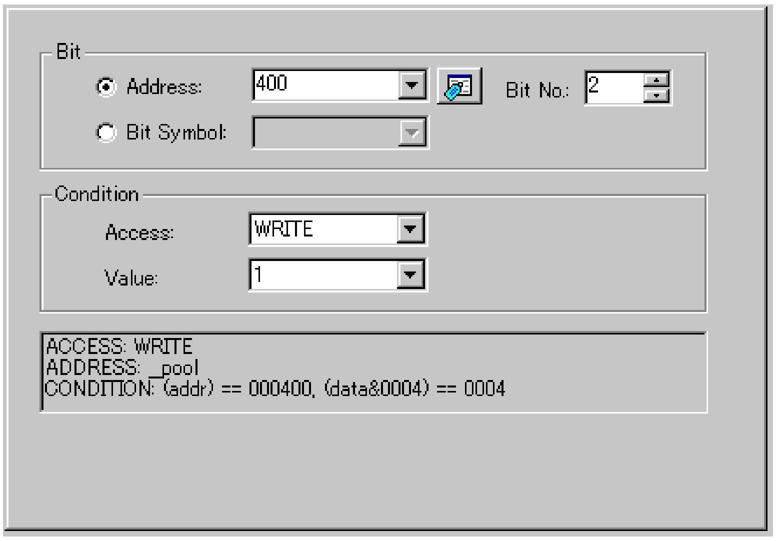
- 选择 [FETCH] （取）时
  跟踪取指令。



- 选择 [DATA ACCESS] （数据存取）时
  跟踪存储器存取。



- 选择 [BIT SYMBOL] （位符号）时
  跟踪位存取。

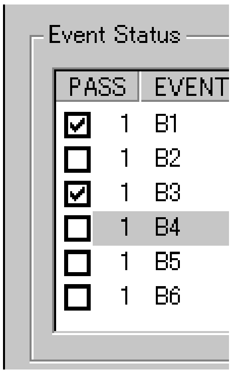


7.8.2　　指定组合条件

要指定组合条件，请从组合条件指定区域选择所需条件。

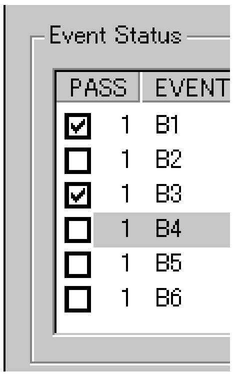
- 选择 [AND] （与）或 [OR] （或）时

  在事件指定区域，可以指定所用事件和该事件的通过计数。要更改通过计数，请在要更改的事件处于选中状态时，单击该事件的通过计数值。



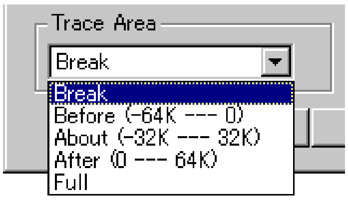
- 选择 [AND (Same Time)] （与 （同时））时

  在事件指定区域，可以指定所用事件。不能指定通过计数。



7.8.3 指定跟踪范围

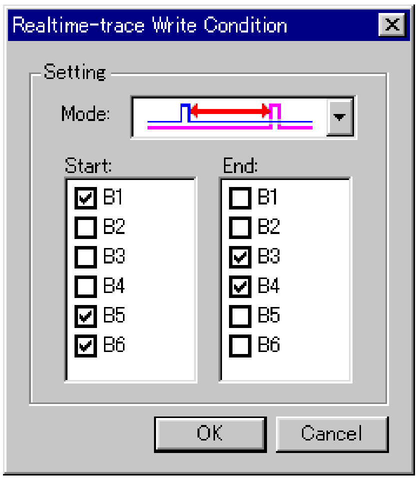
对于小型仿真调试器，可以记录相当于 64K 周期的数据。



[Break] （暂停）	存储目标程序停止点之前的 64K 个周期 （-64K 到 0 周期）。
[Before] （之前）	存储跟踪点之前的 64K 个周期及当前跟踪点 （-64K 到 1 周期）。
[About] （周围）	存储跟踪点两侧的 64K 个周期 （-32K 到 32K 周期）。
[After] （之后）	存储跟踪点之后的 64K 个周期 （0 到 64K 周期）的跟踪数据。
[Full] （完整）	存储跟踪开始之后的 64K 个周期 （-64K 到 0 周期）的跟踪数据。

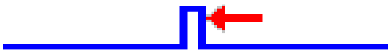
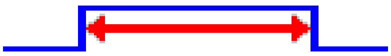
7.8.4 指定跟踪写条件

可以指定要写入跟踪存储器的周期的条件。



[Total]（全部）	写入所有周期。
[Pick up]（提取）	只写入指定条件为真的周期。
[Exclude]（排除）	只写入指定条件不为真的周期。

同时还支持以下三种写模式。

	仅建立指定事件的周期
	从建立指定事件到未建立指定事件的周期
	从建立开始事件到建立结束事件的周期

7.8.5 命令按钮

此窗口中的按钮具有以下含义。

按钮	功能
[Reset]（复位）	删除窗口中显示的内容，并从设置了内容的仿真器加载内容。
[Save...]（保存…）	将窗口中设置的内容保存到文件中。
[Load...]（加载…）	从保存了事件信息的文件加载事件信息。
[Set]（设置）	将窗口中设置的内容发送到仿真器。
[Close]（关闭）	关闭窗口。

7.8.6 指定事件（取指令）

为取指令设置事件的方法与为 [H/W Break Point Setting]（硬件断点设置）窗口设置事件的方法相同。
有关设置的详细信息，请参阅“7.6.4 指定事件（取指令）”。

7.8.7 指定事件（存储器存取）

为存储器存取设置事件的方法与为 [H/W Break Point Setting]（硬件断点设置）窗口设置事件的方法相同。
有关设置的详细信息，请参阅“7.6.5 指定事件（存储器存取）”。

7.8.8 指定事件（位存取）

为位存取设置事件的方法与为 [H/W Break Point Setting]（硬件断点设置）窗口设置事件的方法相同。
有关设置的详细信息，请参阅“7.6.6 指定事件（位存取）”。

7.8.9 指定事件（中断）

设置事件组合的方法与为 [H/W Break Point Setting]（硬件断点设置）窗口设置事件的方法相同。
有关设置的详细信息，请参阅“7.6.7 指定事件组合条件”。

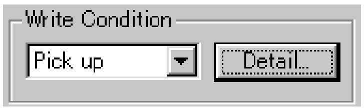
7.8.10 指定写条件

可以指定跟踪数据写条件。
可以指定以下写条件：

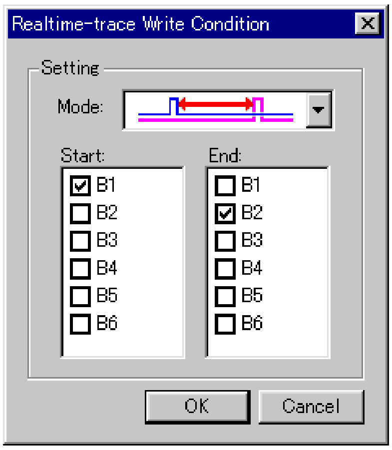
1. 无限制的写条件 （默认设置）
2. 从建立开始事件到建立结束事件的周期
3. 仅建立开始事件的周期
4. 从建立开始事件到取消建立开始事件的周期
5. 从建立开始事件到建立结束事件的周期之外
6. 建立开始事件的周期之外
7. 从建立开始事件到取消建立开始事件的周期之外

若要指定条件 1，请从窗口中 “Write Condition”（写条件）项目的列表框中选择 “Total”（全部）。

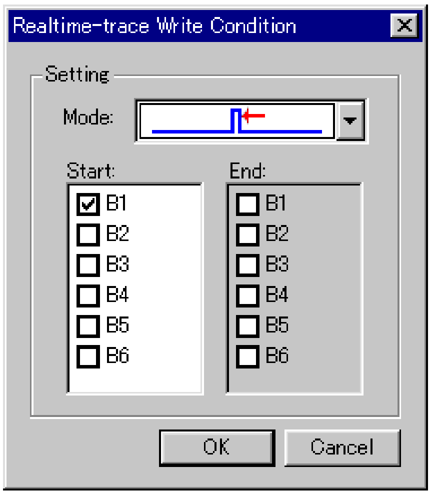
若要指定条件 2 到 4，请选择 “Pick Up”（提取）并单击 “Detail...”（详细信息…）按钮，以打开 “Realtime-trace Write Condition”（实时跟踪写条件）对话框。



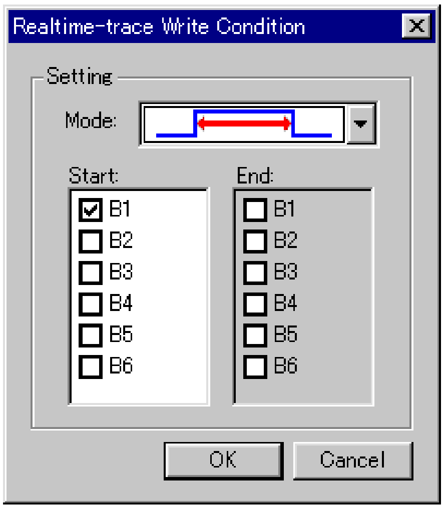
- 对于条件 2，请选择如下模式并设置开始和结束事件。



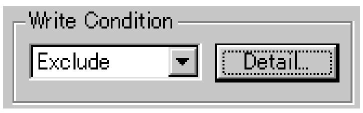
- 对于条件 3，请选择如下模式并设置开始事件。



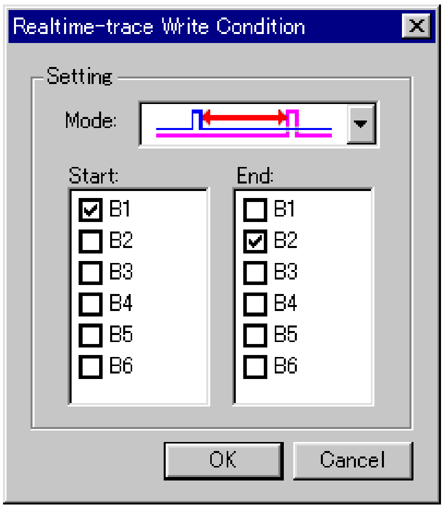
- 对于条件 4，请选择如下模式并设置开始事件。



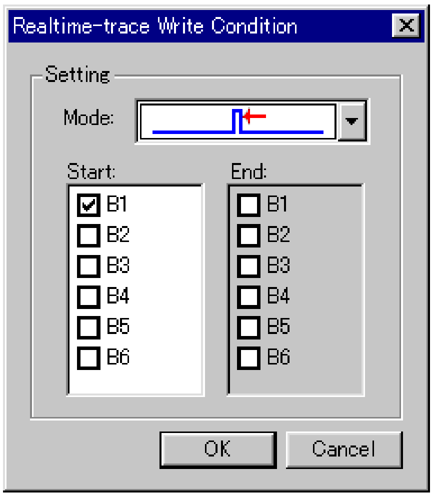
类似地，当指定条件 5 到 7 时，请选择 “Exclude”（排除）并单击 “Detail...”（详细信息…）按钮，以打开 [Realtime-trace Write Condition] （实时跟踪写条件）对话框。



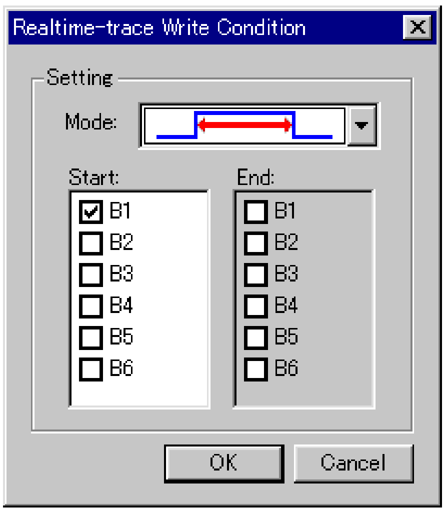
• 对于条件 5，请选择如下模式并设置开始和结束事件。



• 对于条件 6，请选择如下模式并设置开始事件。



• 对于条件 7，请选择如下模式并设置开始事件。



7.9 [Trace] （跟踪）窗口

[Trace] （跟踪）窗口用于显示实时跟踪计量的结果。计量结果可以在以下显示模式下显示。

- “Bus mode”（总线模式）

  该模式允许按周期查看总线信息。显示内容取决于使用的 MCU 和仿真器系统。除总线信息外，此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）

  此模式允许查看执行的指令。除反汇编信息外，此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）

  此模式允许查看数据读/写周期。除数据存取信息外，此模式还允许以组合形式显示源行信息。
- “Source mode”（源模式）

  此模式允许查看源程序中的程序执行路径。

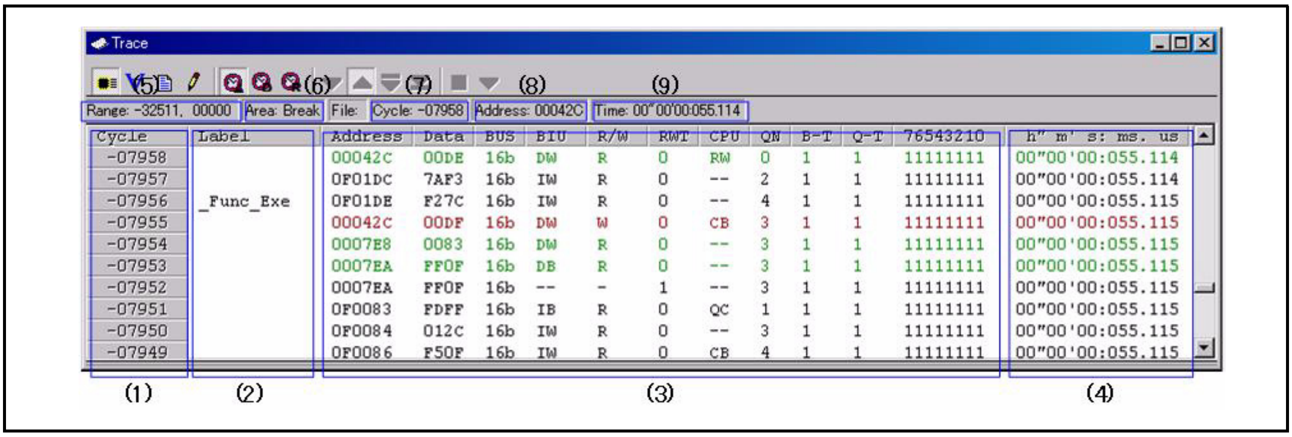
跟踪计量结束时，会显示计量结果。当跟踪计量再启动时，会清除窗口显示内容。

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“7.8 [Trace Point Setting] （跟踪点设置）窗口”。使用默认设置时，会记录程序停止之前的跟踪信息。

7.9.1 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。

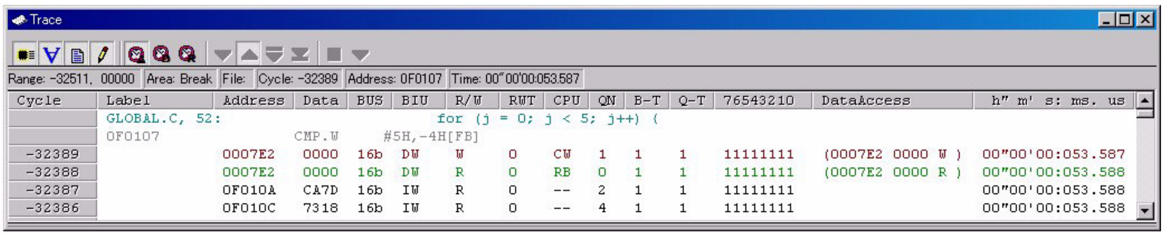


1. 周期显示区：
   显示跟踪周期。双击这里可以打开更改显示周期的对话框。
2. 标签显示区：
   显示与地址总线信息相对应的标签。双击这里可以打开搜索地址的对话框。
3. 总线信息显示区：
   这里显示的内容因所用的 MCU 或仿真器系统而异。

- 请参阅“7.9.6 显示 M32C 调试器的总线信息”。
- 请参阅“7.9.7 显示 M16C/R8C 调试器的总线信息”。

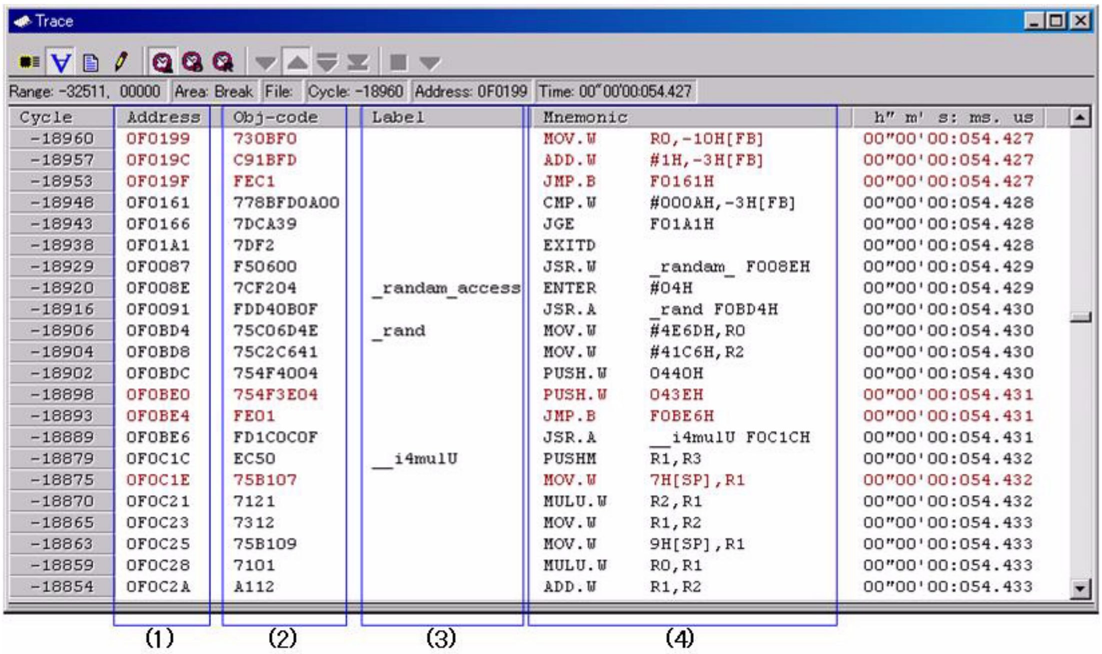
4. 时间信息显示区：
   显示跟踪计量结果的时间信息。可以从菜单中选择以下三种模式之一。（不支持小型仿真调试器。）
   — “Absolute Time”（绝对时间）：按照绝对时间显示从程序开始运行的时间到现在所经过的时间（默认）。
   — “Differences”（差异）：显示与相邻的前一个周期的时间差异。
   — “Relative Time”（相对时间）：显示与选择的周期的相对时间。不过，请注意，更新跟踪计量结果时，此模式会更改为绝对时间显示模式。
5. 获取的跟踪计量结果范围：
   显示当前获取的跟踪计量结果范围。
6. 跟踪计量范围：
   显示当前设置的跟踪计量范围。
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。



7.9.2 配置 “Disassemble Mode”（反汇编模式）

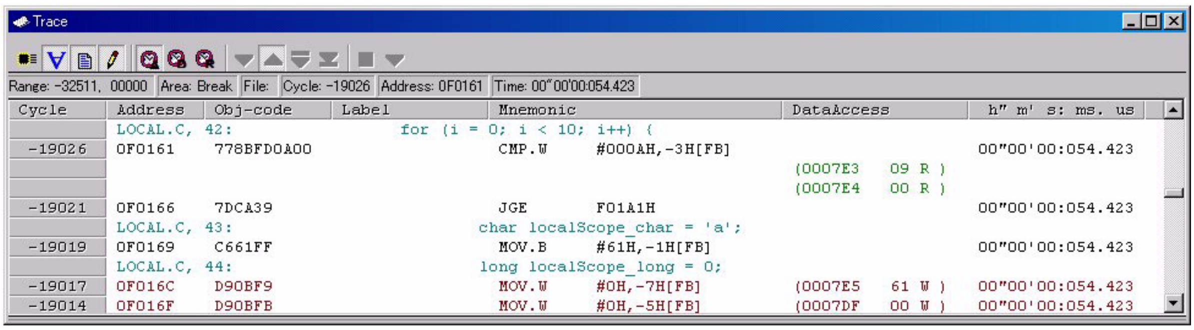
当选择 “disassemble mode”（反汇编模式）而取消选择 “bus mode”（总线模式）时，跟踪信息会以反汇编模式显示。反汇编模式的配置如下所示。



1. 地址显示区：
   显示与指令相对应的地址。双击这里可以打开搜索地址的对话框。
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。双击这里可以打开搜索地址的对话框。
4. 助记符显示区：
   显示指令的助记符。

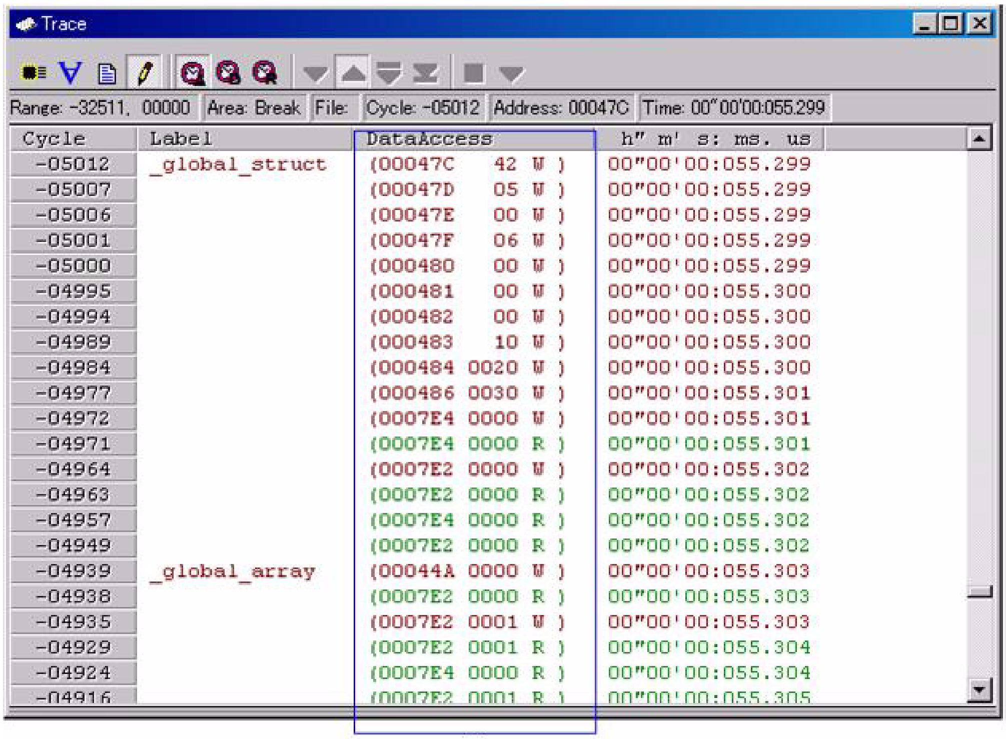
其他显示区与总线模式的显示区相同。

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。



7.9.3　　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。



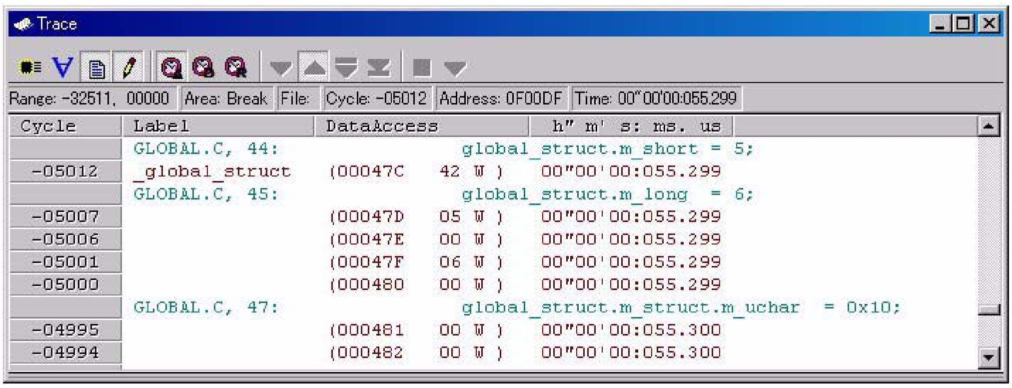
(1)

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。

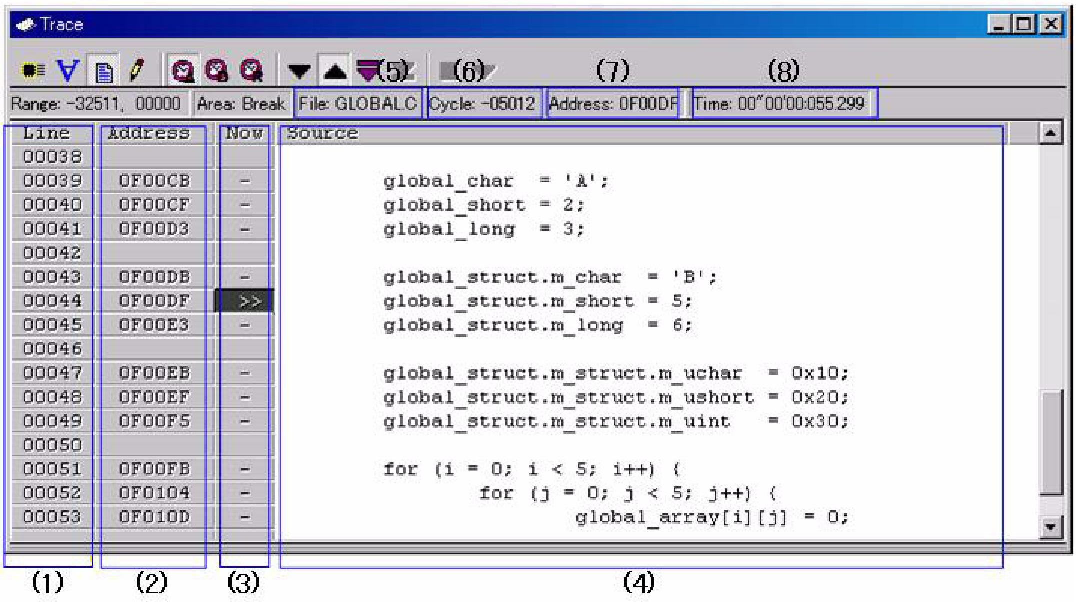
其他显示区与总线模式的显示区相同。

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。



7.9.4 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。双击这里可以打开搜索地址的对话框。

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。

7.9.5　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。

7.9.6　显示 M32C 调试器的总线信息

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
- BUS
  外部数据总线的宽度（“8b”表示 8 位数据总线，“16b”表示 16 位数据总线）
- BHE
  表示 BHE (Byte High Enable) 信号的状态（0 或 1）。
  如果 BHE 为 0，则表示 CPU 在存取奇地址。
- BIU
  此项显示 BIU（总线接口单元）和存储器以及 BIU 和 I/O 之间的状态。

表示法	BIU 状态
-	无存取
WAIT	正在执行 wait 指令
RBML	读存取（字节， ML 开启）
F	取存取
QC	不连续的取存取（队列缓冲器）
RWML	读存取（字， ML 开启）
INT	中断应答
RB	读存取（字节）
WB	写存取（字节）
DRB	DMA 读存取（字节）
DWB	DMA 写存取（字节）
RW	读存取（字）
WW	写存取（字）
DRW	DMA 读存取（字）
DWW	DMA 写存取（字）

- R/W

  显示数据总线的状态 （“R”表示读，“W”表示写，“-”表示无存取）。
- RWT

  此信号显示总线周期中的有效位置 （有效时为 “0”）。当 RWT 为 “0”时，地址、数据和 BIU 信号有效。
- CPU、 OPC、 OPR

  此项显示 CPU 和 BIU 之间的信号。在 “CPU”列中，数据显示 CPU 是否存取 BIU。在 “OPC”列中，数据显示读操作码的字节大小。在 “OPR”列中，数据显示读操作数的字节大小。

表示法			状态	
CPU	OPC	OPR	操作码大小	操作数大小
-	-	-	无存取	
CPU	0	1	0 字节	1 字节
CPU	0	2	0 字节	2 字节
CPU	0	3	0 字节	3 字节
CPU	1	0	1 字节	0 字节
CPU	1	1	1 字节	1 字节
CPU	1	2	1 字节	2 字节
CPU	1	3	1 字节	3 字节
CPU	2	0	2 字节	0 字节
CPU	2	1	2 字节	1 字节
CPU	2	2	2 字节	2 字节
CPU	3	0	3 字节	0 字节
CPU	3	1	3 字节	1 字节
DMA	-	-	DMA 存取	
DMAT	-	-	DMA 存取 （终端计数）	

7.9.7 显示 M16C/R8C 调试器的总线信息

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
- BUS
  外部数据总线的宽度 （“8b” 表示 8 位数据总线，“16b” 表示 16 位数据总线）
- BHE
  表示 BHE (Byte High Enable) 信号的状态 （0 或 1）。
  如果 BHE 为 0，则表示 CPU 在存取奇地址。
- BIU
  此项显示 BIU （总线接口单元）和存储器以及 BIU 和 I/O 之间的状态。

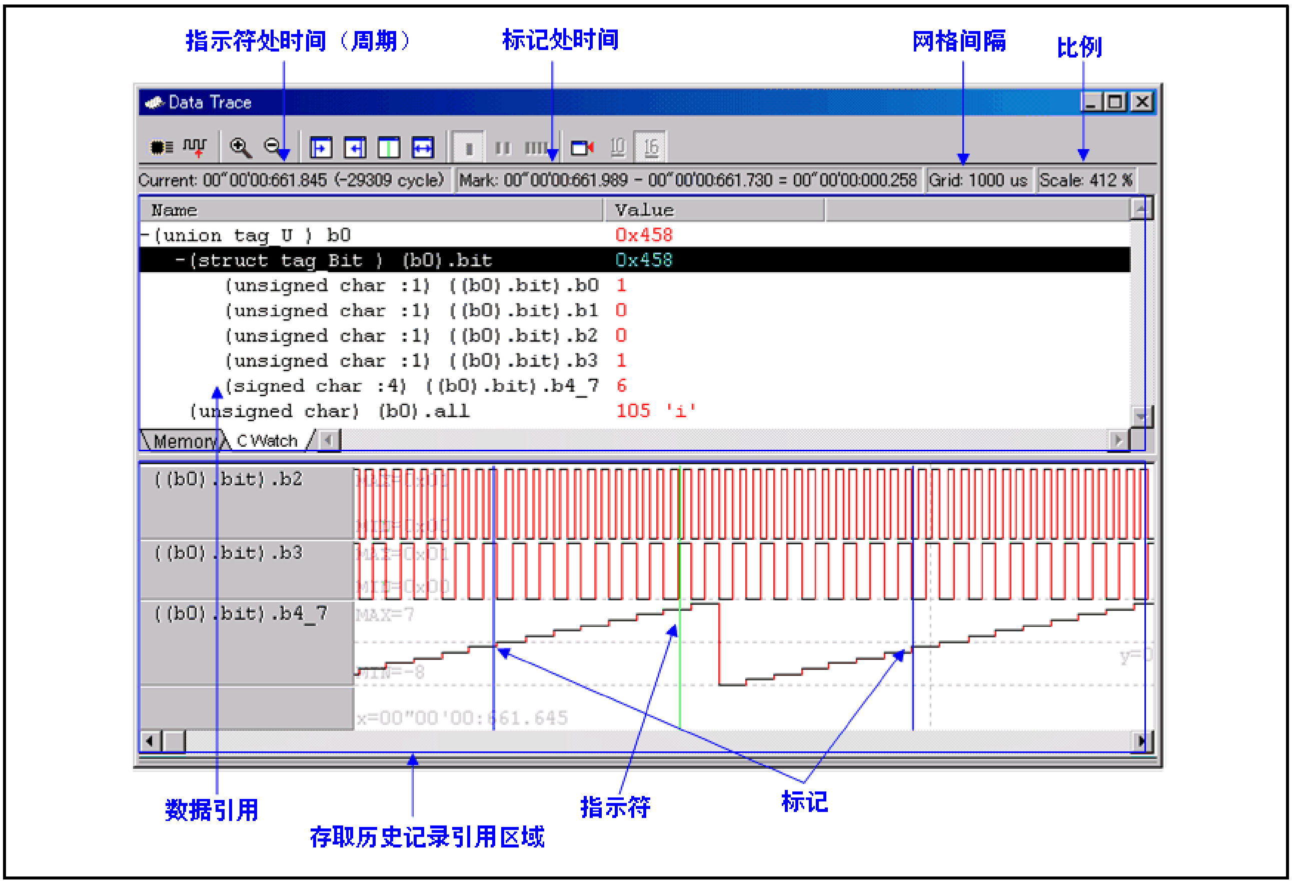
表示法	BIU 状态
-	无存取
DMA	非 CPU 原因导致的数据存取，例如 DMA
INT	启动 INTACK 序列
IB	由于 CPU 原因导致的指令码读操作 （字节）
DB	由于 CPU 原因导致的数据存取 （字节）
IW	由于 CPU 原因导致的指令码读操作 （字）
DW	由于 CPU 原因导致的数据存取 （字）

- R/W
  显示数据总线的状态 （“R” 表示读，“W” 表示写，“-” 表示无存取）。
- RWT
  此信号显示总线周期中的有效位置 （有效时为 “0”）。当 RWT 为 “0” 时，地址、数据和 BIU 信号有效。
- CPU
  此项显示 CPU 与 BIU 之间的信号。

表示法	状态
-	无存取
CB	操作码读 （字节）
RB	操作数读 （字节）
QC	指令队列缓冲器清除
CW	操作码读 （字）
RW	操作数读 （字）

7.10　[Data Trace]（数据跟踪）窗口

[Data Trace]（数据跟踪）窗口用于分析实时跟踪计量结果，并用图形显示数据存取信息。
该窗口与 [Trace]（跟踪）窗口结合使用。
小型仿真器的调试器不支持时间信息。



- 在数据引用区域，可以检查当前涉及的周期点的存储器值，或检查已注册的 C 变量的值。
- 在存取历史记录引用区域，可以查看图表形式的存取注册地址的历史记录。
- 与 [Trace]（跟踪）窗口结合使用时，可以在 [Trace]（跟踪）窗口中查看正在监视的周期点的存储器值。反过来，也可以在 [Data Trace]（数据跟踪）窗口中显示正在监视的 [Trace]（跟踪）窗口中的周期。

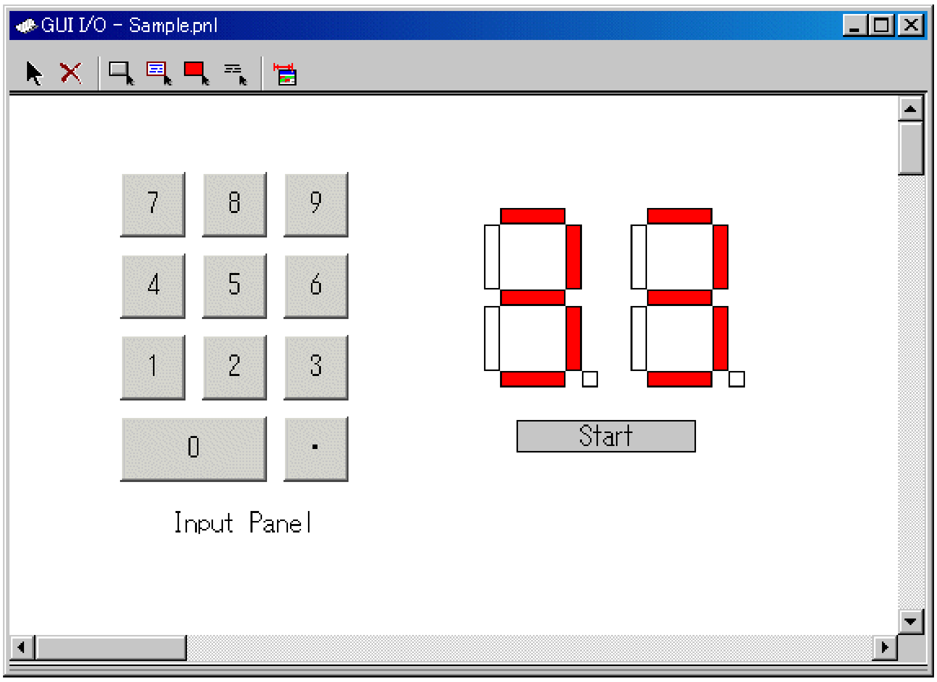
7.10.1 扩展菜单

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
[Analyze Trace Data] （分析跟踪数据）		分析实时跟踪数据。
[Set Cycle...] （设置周期…）		指定显示周期。
[Sync with Trace Window] （与跟踪窗口同步）		与 [Trace] （跟踪）窗口同步。
[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。
[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。
[Address...] （地址…）		从指定地址显示。
[Add C Watch] （添加 C 监视）		添加 C 监视点。
[Remove C Watch] （移除 C 监视）		移除选定的 C 监视点。
[Hide Type Name] （隐藏类型名称）		隐藏变量的类型名称。
[Add...] （添加…）		向存取历史记录引用区域添加新的监视项目。
[Remove] （移除）		移除存取历史记录引用区域中的选定监视项目。
[Zoom] （缩放）	[Zoom In] （放大）	增大显示比例。
	[Zoom Out] （缩小）	缩小显示比例。
	[Zoom...] （缩放…）	指定显示比例。
[Marker] （标志）	[Start Marker] （开始标志）	在显示区中移动开始标志。
	[End Marker] （结束标志）	在显示区中移动结束标志。
	[Indicator] （指示器）	在显示区中移动指示器。
	[Adjust] （调整）	设置标志间的周期范围。
[Change Grid Interval...] （更改网格间隔…）		更改网格间隔。
[Change Row Setting...] （更改行设置…）		更改选定行的设置。
[Color...] （颜色…）		更改显示颜色。
[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。

7.11　[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
    如果有任意值写入指定地址 （或指定位），则更改任意区域的显示颜色。（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。

7.11.1　扩展菜单

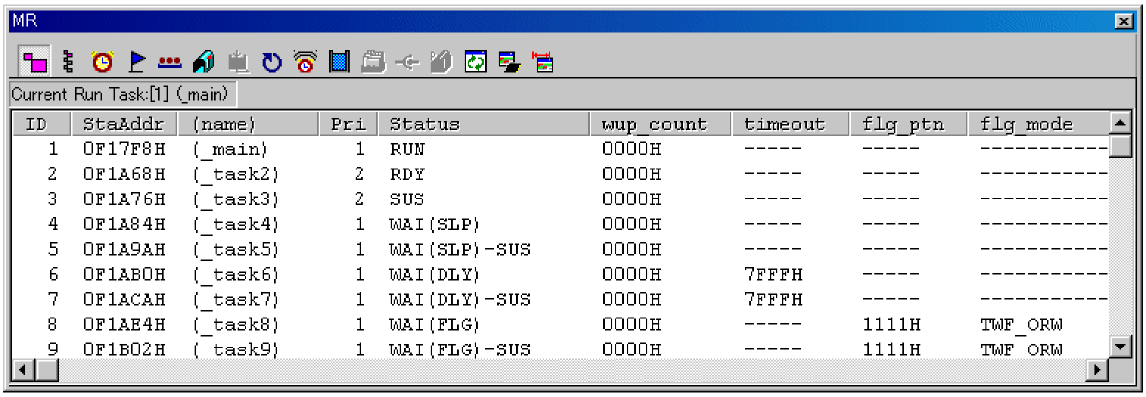
此窗口具有如下弹出菜单，在窗口中右键单击便可显示该菜单。

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。

7.12 [MR] 窗口

使用 [MR] 窗口可显示实时操作系统的状态。

只有下载使用实时操作系统的程序后，才能使用 [MR] 窗口 （如果下载的程序不使用 MR，则打开 [MR] 窗口后，该窗口中不会显示任何内容。）



- 可以打开与显示模式数量同样多的 [MR] 窗口。
- 单击所需按钮， [MR] 窗口显示模式及显示数据会更改。
- 双击所需的任务行，可显示任务的上下文数据。
- 可以拖动光标来更改各个模式中显示区的宽度。
- 如果下载的程序不使用 MR，则无法选择用于选择显示模式的所有菜单。
- 可用的显示模式因 MRxx 而异。

注意事项

制作下载的程序时，请使用内容与 MRxx 版本匹配的启动文件 (crt0mr.axx/start.axx)。如果使用的启动文件与 MRxx 的版本不匹配，则 [MR] 窗口和 MR 命令将无法正常运行。

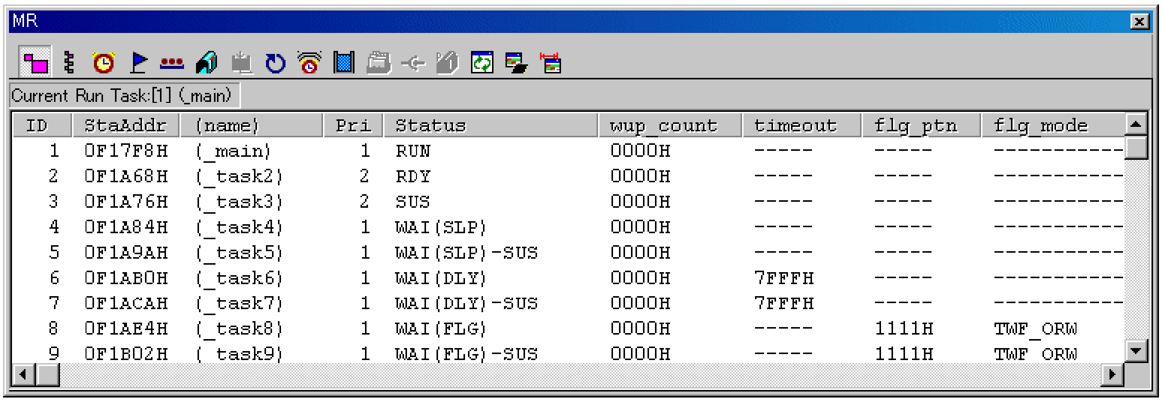
7.12.1　扩展菜单

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
[Mode] （模式）	[Task] （任务）	显示任务状态。
	[Ready Queue] （就绪队列）	显示就绪状态。
	[Timeout Queue] （超时队列）	显示超时状态。
	[Event Flag] （事件标志）	显示事件标志状态。
	[Semaphore] （信标）	显示信标状态。
	[Mailbox] （邮箱）	显示邮箱状态。
	[Data Queue] （数据队列）	显示数据队列状态。
	[Cyclic Handler] （周期处理程序）	显示周期处理程序状态。
	[Alarm Handler] （警报处理程序）	显示警报处理程序状态。
	[Memory Pool] （存储池）	显示存储池状态。
	[Message Buffer] （消息缓冲器）	显示消息缓冲器状态。
	[Port] （端口）	显示端口状态。
	[Mailbox(with Priority)] （邮箱 （带优先级））	显示邮箱 （带优先级）状态。
[Context...] （上下文…）		显示上下文。
[Layout] （布局）	[Status Bar] （状态栏）	在显示或不显示状态栏之间切换。
[Refresh] （刷新）		刷新存储器数据。
[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。

7.12.2 显示任务状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式） → [Task] （任务）。



双击任意行， [Context] （上下文）对话框中将显示任务上下文的相关信息。有关 [Context] （上下文）对话框的详细信息，请参阅 “7.12.12 显示任务上下文” 状态栏中会显示下列数据。

7.12.2.1 显示任务状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有任务会按照 ID 号顺序列出。每个项目的功能描述如下。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	任务 ID
[StaAddr]	任务的起始地址
[(name)] （（名称））	任务名称
[Pri]	优先级
[Status] （状态） *1	任务状态
[wup_count]	唤醒计数
[timeout] （超时）	超时值
[flg_ptn]	事件标志的等待位模式 （Wait bit pattern）
[flg_mode*2]	事件标志的等待取消条件

- *1 任务状态

监视器	状态
RUN	运行状态
RDY	就绪状态
SUS	挂起状态
DMT	休眠状态
WAI(SLP)	睡眠状态
WAI(SLP)-SUS	睡眠状态 （挂起）
WAI(SLP-TMO)	超时睡眠状态
WAI(SLP-TMO)-SUS	超时睡眠状态 （挂起）
WAI(DLY)	dly_tsk 导致的延迟状态
WAI(DLY)-SUS	dly_tsk 导致的延迟状态 （挂起）
WAI(FLG)	事件标志的等待状态
WAI(FLG)-SUS	事件标志的等待状态 （挂起）
WAI(FLG-TMO)	超时事件标志的等待状态
WAI(FLG-TMO)-SUS	超时事件标志的等待状态 （挂起）
WAI(SEM)	信标资源的等待状态
WAI(SEM)-SUS	信标资源的等待状态 （挂起）
WAI(SEM-TMO)	超时信标资源的等待状态
WAI(SEM-TMO)-SUS	超时信标资源的等待状态 （挂起）
WAI(MBX)	邮箱的接收等待状态
WAI(MBX)-SUS	邮箱的接收等待状态 （挂起）
WAI(MBX-TMO)	超时邮箱的接收等待状态
WAI(MBX-TMO)-SUS	超时邮箱的接收等待状态 （挂起）

- *2 显示事件标志的等待取消条件

flg_mode	状态
TWF_ANDW	等待设置等待位模式 (wait bit pattern) 中设置的所有位 （AND 等待）
TWF_ANDW+TWF_CLR	出现 AND 等待且任务等待状态已取消时，将事件标志清零
TWF_ORW	等待设置等待位模式 (Wait bit pattern) 中设置的任一位 （OR 等待）
TWF_ORW+TWF_CLR	出现 OR 等待且任务等待状态已取消时，将事件标志清零

7.12.2.2　显示任务状态（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有任务会按照 ID 号顺序列出。每个项目的功能描述如下。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	任务 ID
[Name]（名称）	任务名称
[Pri]	优先级
[Status]（状态） *1	任务状态
[Wupcnt]	唤醒计数
[Actcnt]	激活计数
[Tmout]	超时值
[Flgptn]	事件标志的等待位模式（Wait bit pattern）
[Wfmode]*2	事件标志的等待取消条件

- *1 任务状态

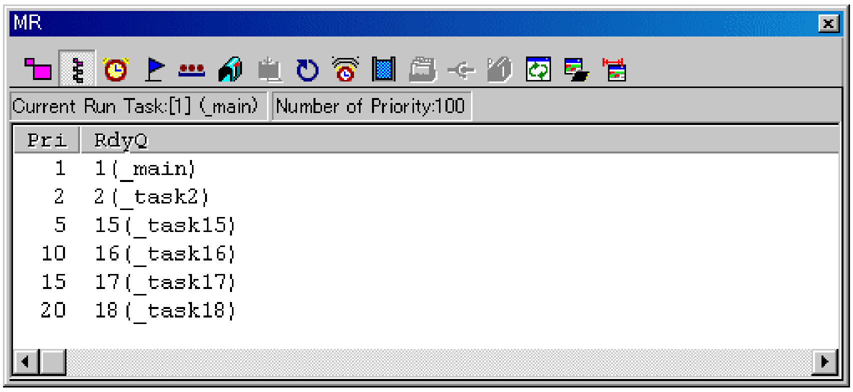
监视器	状态
RUN	运行状态
RDY	就绪状态
SUS	挂起状态
DMT	休眠状态
WAI(SLP)	睡眠状态
WAI(SLP)-SUS	睡眠状态 （挂起）
WAI(SLP-TMO)	超时睡眠状态
WAI(SLP-TMO)-SUS	超时睡眠状态 （挂起）
WAI(DLY)	dly_tsk 导致的延迟状态
WAI(DLY)-SUS	dly_tsk 导致的延迟状态 （挂起）
WAI(FLG)	事件标志的等待状态
WAI(FLG)-SUS	事件标志的等待状态 （挂起）
WAI(FLG-TMO)	事件标志的等待状态
WAI(FLG-TMO)-SUS	事件标志的等待状态 （挂起）
WAI(SEM)	信标资源的等待状态
WAI(SEM)-SUS	信标资源的等待状态 （挂起）
WAI(SEM-TMO)	超时信标资源的等待状态
WAI(SEM-TMO)-SUS	超时信标资源的等待状态 （挂起）
WAI(MBX)	邮箱的接收等待状态
WAI(MBX)-SUS	邮箱的接收等待状态 （挂起）
WAI(MBX-TMO)	超时邮箱的接收等待状态
WAI(MBX-TMO)-SUS	超时邮箱的接收等待状态 （挂起）
WAI(SDTQ)	数据队列的发送等待状态
WAI(SDTQ)-SUS	数据队列的发送等待状态 （挂起）
WAI(SDTQ-TMO)	超时数据队列的发送等待状态
WAI(SDTQ-TMO)-SUS	超时数据队列的发送等待状态 （挂起）
WAI(RDTQ)	数据队列的接收等待状态
WAI(RDTQ)-SUS	数据队列的接收等待状态 （挂起）
WAI(RDTQ-TMO)	超时数据队列的接收等待状态
WAI(RDTQ-TMO)-SUS	超时数据队列的接收等待状态 （挂起）
WAI(VSDTQ)	扩展数据队列的发送等待状态
WAI(VSDTQ)-SUS	扩展数据队列的发送等待状态 （挂起）
WAI(VSDTQ-TMO)	超时扩展数据队列的发送等待状态
WAI(VSDTQ-TMO)-SUS	超时扩展数据队列的发送等待状态 （挂起）
WAI(VRDTQ)	扩展数据队列的接收等待状态
WAI(VRDTQ)-SUS	扩展数据队列的接收等待状态 （挂起）
WAI(VRDTQ-TMO)	超时扩展数据队列的接收等待状态
WAI(VRDTQ-TMO)-SUS	超时扩展数据队列的接收等待状态 （挂起）
WAI(MPF)	固定大小存储块的等待状态
WAI(MPF)-SUS	固定大小存储块的等待状态 （挂起）
WAI(MPF-TMO)	超时固定大小存储块的等待状态
WAI(MPF-TMO)-SUS	超时固定大小存储块的等待状态 （挂起）

- *2 显示事件标志的等待取消条件

Wfmode	状态
TWF_ANDW	等待设置等待位模式 （Wait bit pattern）中设置的所有位 （AND 等待）
TWF_ORW	等待设置等待位模式 （Wait bit pattern）中设置的任一位 （OR 等待）

7.12.3 显示就绪队列状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Ready Queue] （就绪队列）。



状态栏中会显示下列数据。

7.12.3.1 显示就绪队列状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

每个项目的功能描述如下。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[Pri]	显示优先级
[RdyQ]	显示就绪队列中任务的 ID 号和任务名称

- [RdyQ] 字段最多显示任务名称的 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.3.2 显示就绪队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

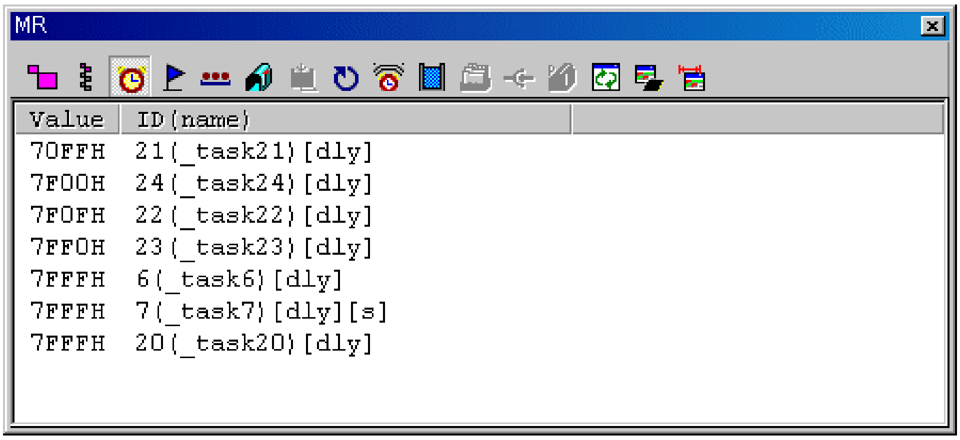
每个项目的功能描述如下。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[Pri]	显示优先级
[Ready Queue] （就绪队列）	显示就绪队列中任务的 ID 号和任务名称

- [Ready Queue] （就绪队列）字段最多显示任务名称的 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.4 显示超时队列状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Timeout Queue] （超时队列）。



7.12.4.1 显示超时队列状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

每个项目的功能描述如下。

当前处于等待状态的任务按照超时值的降序显示。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[Value] （值）	显示每个任务的超时值
[ID(name)] （ID （名称））	显示超时队列中任务的 ID 号和任务名称

- 下列字符串用于指示等待状态的类型。

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

- 当连接到超时队列的任务处于强制等待 （双重等待）状态时， [ID (name)] （ID （名称））字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]

7.12.4.2 显示超时队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

每个项目的功能描述如下。

当前处于等待状态的任务按照超时值的降序显示。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[Tmout]	显示每个任务的超时值 (ms)
[ID(Name)] （ID （名称））	显示超时队列中任务的 ID 号和任务名称

- 下列字符串用于指示等待状态的类型。

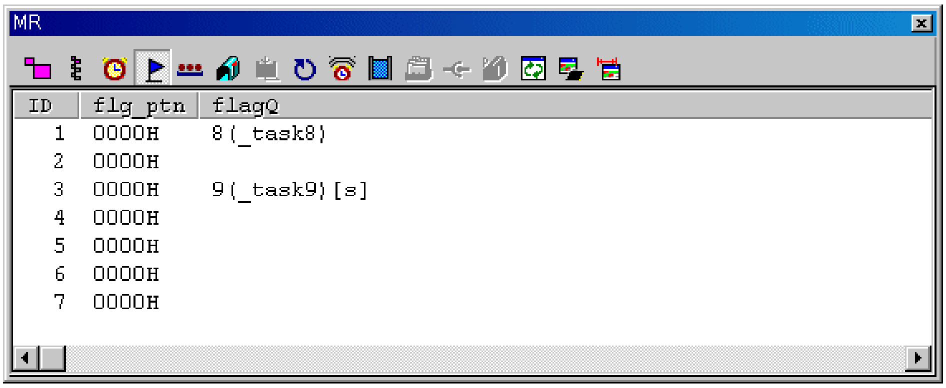
字符串	等待状态
[slp]	tslp_tsk 导致的等待
[dly]	dly_tsk 导致的等待
[flg]	twai_flg 导致的等待
[sem]	twai_sem 导致的等待
[mbx]	trcv_mbx 导致的等待
[mpf]	tget_mpf 导致的等待
[sdtq]	tsnd_dtq 导致的等待
[rdtq]	trcv_dtq 导致的等待
[vsdtq]	vtsnd_dtq 导致的等待
[vrdtq]	vtrcv_dtq 导致的等待

- 当连接到超时队列的任务处于强制等待 （双重等待）状态时， [ID(Name)] （ID （名称））字段显示的字符串后面会追加一个字符串 “[s]”，表明处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]

7.12.5 显示事件标志状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Event Flag] （事件标志）。



7.12.5.1 显示事件标志状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有事件标志会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	事件标志的 ID 号
[flg_ptn]	每个事件标志的位模式 （Bit pattern）
[flagQ]	事件标志队列中的任务 ID 号和任务名称

- 当连接到事件标志队列的任务处于允许超时等待 （twai_flg 等待）状态时， [flag Q] （标志队列）字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时等待状态。

  当连接到事件标志队列的任务处于强制等待 （双重等待）状态时， [flag Q] （标志队列）字段显示的字符串后面会追加一个字符串 “[s]”，表明处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [flag Q] （标志队列）字段的任务名称最多可以显示 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.5.2 显示事件标志状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有事件标志会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	事件标志的 ID 号
[Flgatr]	每个事件标志的属性
[Flgptn]	每个事件标志的位模式 （Bit pattern）
[Flag Queue] （标志队列）	事件标志队列中的任务 ID 号和任务名称

- Flgatr 区域显示内容如下：

TA_TFIFO	任务等待队列处于 FIFO 顺序
TA_TPRI	任务等待队列处于任务优先级顺序
TA_WSGL	只允许一个任务处于事件标志等待状态
TA_WMUL	允许多个任务处于事件标志等待状态
TA_CLR	任务脱离等待状态后，该事件标志的位模式 （Bit pattern）将被清除

- 当连接到事件标志队列的任务处于允许超时等待 （twai_flg 等待）状态时， [Flag Queue] （标志队列）字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时等待状态。

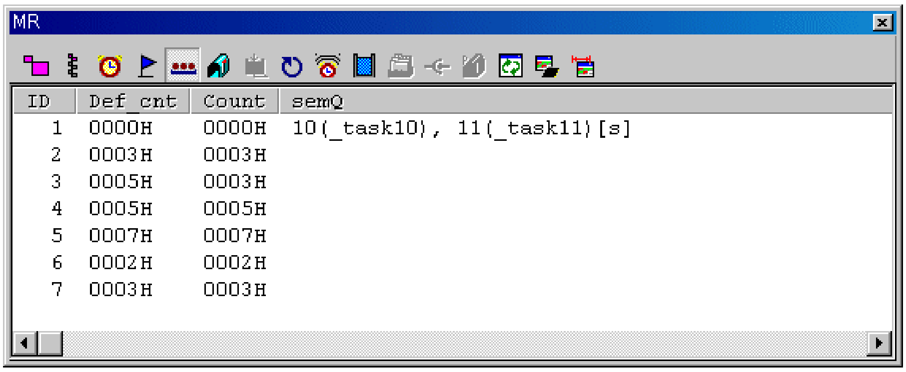
  当连接到事件标志队列的任务处于强制等待 （双重等待）状态时， [Flag Queue] （标志队列）字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [Flag Queue] （标志队列）字段的任务名称最多可以显示 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.6　　显示信标状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Semaphore] （信标）。



7.12.6.1　　显示信标状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有信标会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	信标的 ID 号
[Def_cnt]	信标计数器的默认值
[count]	信标计数
[semQ]	信标队列中的任务 ID 号和任务名称

- 当连接到信标队列的任务处于允许超时等待 （twai_sem 等待）状态时， [semQ] 字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时等待状态。

  当连接到信标队列的任务处于强制等待 （双重等待）状态时， [semQ] 字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [semQ] 字段的任务名称最多可以显示 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.6.2　显示信标状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有信标会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	信标的 ID 号
[Sematr]	每个信标的属性
[Semcnt]	信标计数
[Semaphore Queue] （信标队列）	信标队列中的任务 ID 号和任务名称

- Sematr 区域显示内容如下：

TA_TFIFO	任务等待队列处于 FIFO 顺序
TA_TPRI	任务等待队列处于任务优先级顺序

- 当连接到信标队列的任务处于允许超时等待 （twai_sem 等待）状态时， [Semaphore Queue] （信标队列）字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时等待状态。

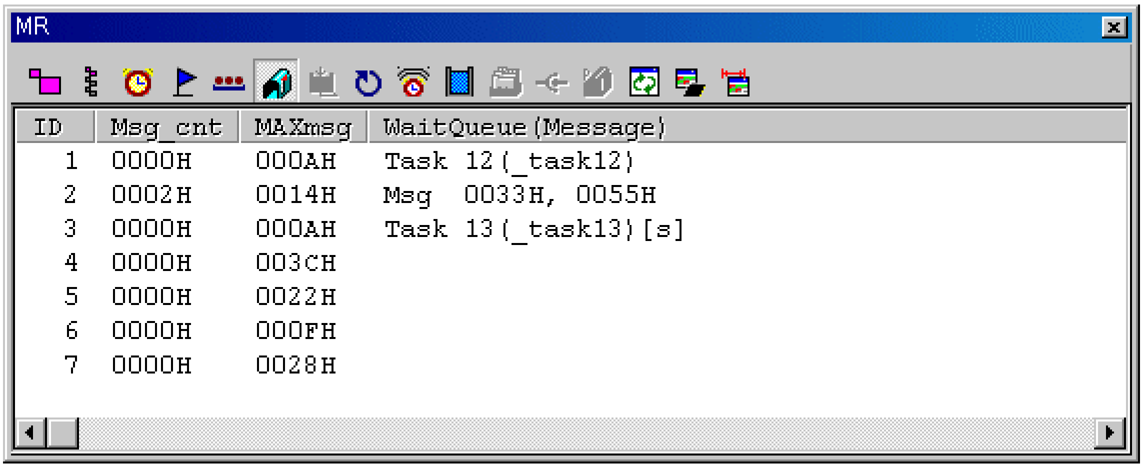
  当连接到信标队列的任务处于强制等待 （双重等待）状态时， [Semaphore Queue] （信标队列）字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [Semaphore Queue] （信标队列）字段的任务名称最多可以显示 8 个字符。如果任务名称超过 8 个字符，则忽略多余字符。

7.12.7 显示邮箱状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Mailbox] （邮箱）。



7.12.7.1 显示邮箱状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有邮箱会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	邮箱的 ID 号
[Msg_cnt]	每个邮箱中的邮件数量
[MAXmsg]	每个邮箱可以容纳邮件的最大数量
[Wait Queue(Message)] （等待队列 （邮件））	存储在邮箱中的邮件或正在等待邮件的任务的 ID 号和任务名称

- [WaitQueue (Message)] （等待队列 （邮件））字段会在存储邮件后 （上述 [Msg_cont] 不为零时）显示字符串 “Msg”，然后显示存储的邮件。

  未存储任何邮件时 （[Msg_cont] 为零时），如果存在等待邮件的任务，则 [WaitQueue] （等待队列）字段显示字符串 “Task”（任务），然后显示正在等待邮件的任务的 ID 号和名称。
- 当连接到邮箱队列的任务处于允许超时等待 （trcv_msg 等待）状态时， [WaitQueue (Message)] （等待队列 （邮件））字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时状态。

  当连接到邮箱队列的任务处于强制等待 （双重等待）状态时， [WaitQueue (Message)] （等待队列 （邮件））字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [WaitQueue (Message)] （等待队列 （邮件））字段的任务名称最多可以显示 8 个字符。

  如果任务名称超过 8 个字符，则忽略多余字符。

7.12.7.2　　显示邮箱状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有邮箱会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	邮箱的 ID 号
[Mbxatr]	每个邮箱的属性
[Mailbox Queue (Wait)] （邮箱队列 （等待））	正在等待邮件的任务的 ID 号和任务名称
[Mailbox Queue (Message)] （邮箱队列 （邮件））	存储在邮箱中的邮件

- Mbxatr 区域显示内容如下：

TA_TFIFO	任务等待队列处于 FIFO 顺序
TA_TPRI	任务等待队列处于任务优先级顺序
TA_MFIFO	邮件队列处于 FIFO 顺序
TA_MPRI	邮件队列处于邮件优先级顺序

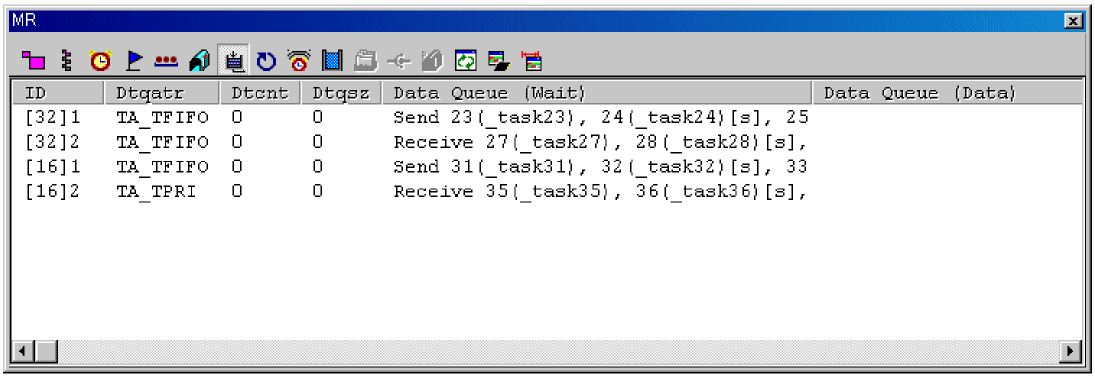
- 当连接到邮箱队列的任务处于允许超时等待 （trcv_mbx 等待）状态时， [Mailbox Queue (Wait)] （邮箱队列 （等待））字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时状态。
  当连接到邮箱队列的任务处于强制等待 （双重等待）状态时， [Mailbox Queue (Wait)] （邮箱队列 （等待））字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

- [Mailbox Queue (Wait)] （邮箱队列 （等待））字段的任务名称最多可以显示 8 个字符。
  如果任务名称超过 8 个字符，则忽略多余字符。

7.12.8 显示数据队列状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式） → [Data Queue] （数据队列）。



7.12.8.1 显示数据队列状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有数据队列会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	数据队列的 ID 号
[Dtqatr]	每个数据队列的属性
[Dtcnt]	每个数据队列中的邮件数量
[Dtqsz]	每个数据队列可以容纳邮件的最大数量
[Data Queue (Wait)] （数据队列 （等待））	正在等待邮件传输等待或邮件接收等待的任务的 ID 号和任务名称
[Data Queue (Data)] （数据队列 （数据））	存储在数据队列中的邮件

- [ID] 字段的显示内容取决于指定的是标准数据 （32 位）还是扩展数据 （16 位）。

  MR308/4

  — 如果指定的是标准数据 （32 位），则 [ID] 字段显示字符串 “[32]” 和数据队列 ID 号。

  — 如果指定的是扩展数据 （16 位），则 [ID] 字段显示字符串 “[16]” 和数据队列 ID 号。

  MR30/4

  — 如果指定的是标准数据 （16 位），则 [ID] 字段显示字符串 “[16]” 和数据队列 ID 号。

  — 如果指定的是扩展数据 （32 位），则 [ID] 字段显示字符串 “[32]” 和数据队列 ID 号。

- Dtqatr 区域显示内容如下：

TA_TFIFO	任务等待队列处于 FIFO 顺序
TA_TPRI	任务等待队列处于任务优先级顺序

- 如果任务正在等待邮件发送，则 [Data Queue (Wait)] （数据队列 （等待））字段显示字符串 “Send” （发送），然后显示正在等待邮件发送的任务的 ID 号和名称。同样，如果任务正在等待邮件接收，则显示字符串 “Receive”（接收），然后显示正在等待邮件接收的任务的 ID 号和名称。
- 当连接到数据队列的任务处于允许超时等待状态时， [Data Queue (Wait)] （数据队列 （等待））字段显示的字符串后面会追加一个字符串 “[tmo]”，表示处于允许超时状态。

  当连接到数据队列的任务处于强制等待 （双重等待）状态时， [Data Queue (Wait)] （数据队列 （等待））字段显示的字符串后面会追加一个字符串 “[s]”，表示处于双重等待状态。

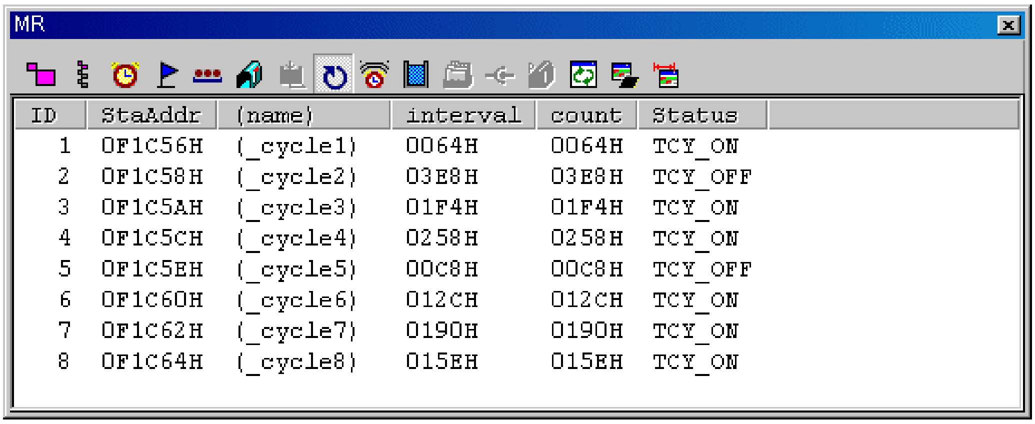
普通显示	26(_task26)
处于 WAIT-SUSPEND 状态时的显示	26(_task26)[s]
处于超时 WAIT-SUSPEND 状态时的显示	26(_task26)[tmo][s]

[Data Queue (Wait)] （数据队列 （等待））字段的任务名称最多可以显示 8 个字符。

如果任务名称超过 8 个字符，则忽略多余字符。

7.12.9　显示周期处理程序状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式） → [Cyclic Handler] （周期处理程序）。



7.12.9.1 显示周期处理程序状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有周期处理程序会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	周期处理程序的 ID 号
[StaAddr]	周期处理程序的起始地址
[(name)] （(名称)）	周期处理程序的名称
[interval] （间隔）	中断间隔
[count] （计数）	中断计数
[Status] （状态）	周期开始处理程序的活动状态

- [Status] （状态）区域显示内容如下：

TCY_ON	允许周期处理程序
TCY_OFF	禁止周期处理程序

7.12.9.2 显示周期处理程序状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

配置中定义的所有周期处理程序会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

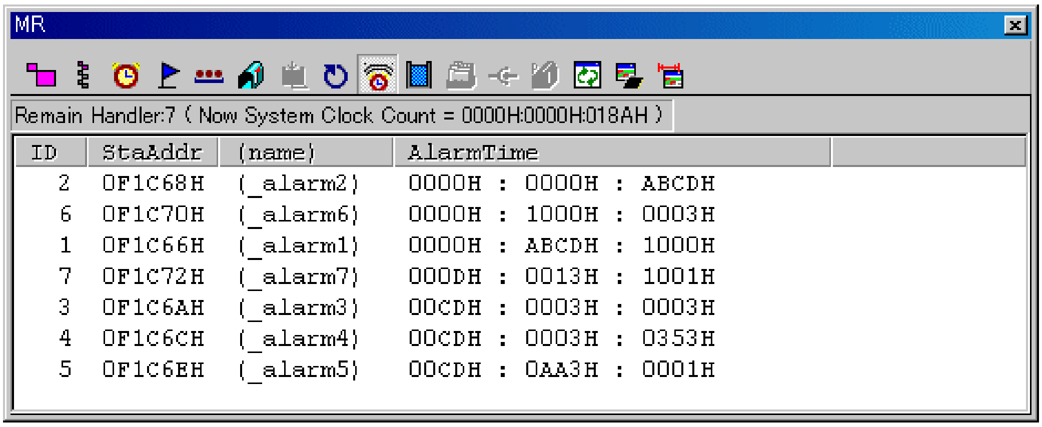
项目	内容
[ID]	周期处理程序的 ID 号
[Name] （名称）	周期处理程序的名称
[Cycphs]	激活阶段 （毫秒）
[Cyctim]	激活周期时间 （毫秒）
[Tmout]	到周期处理程序下次激活时间之前剩余的时间量 （毫秒）
[Status] （状态）	周期开始处理程序的活动状态

- [Status] （状态）区域显示内容如下：

TCYC_STA	周期处理程序处于操作状态
TCYC_STP	周期处理程序处于非操作状态

7.12.10　显示警报处理程序状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Alarm Handler] （警报处理程序）。



当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时，状态栏中会显示下列数据。

7.12.10.1　　显示警报处理程序状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

在配置中定义的所有周期开始处理程序中，只有目前尚未开始的处理程序才会按照开始时间升序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	警报处理程序的 ID 号
[StaAddr]	警报处理程序的起始地址
[(name)] （（名称））	警报处理程序的名称
[AlarmTime] （警报时间）	警报处理程序的开始时间

7.12.10.2　　显示警报处理程序状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

在配置中定义的所有周期开始处理程序中，只有目前尚未开始的处理程序才会按照开始时间升序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

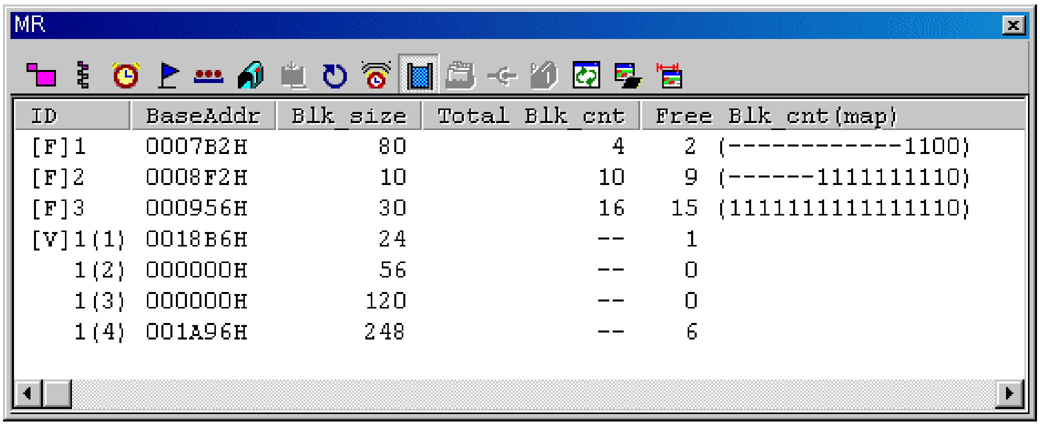
项目	内容
[ID]	警报处理程序的 ID 号
[Name] （名称）	警报处理程序的名称
[Almtim]	到警报处理程序激活时间之前剩余的时间量 （毫秒）
[Status] （状态）	警报处理程序的活动状态

[Status] （状态）区域显示内容如下：

TALM_STA	警报处理程序处于操作状态
TALM_STP	警报处理程序处于非操作状态

7.12.11 显示存储池状态

在 [MR] 窗口中，选择弹出菜单 - [Mode] （模式）→ [Memory Pool] （存储池）。



7.12.11.1 显示存储池状态 （当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

配置中定义的所有存储池会按照 ID 号顺序列出。（固定长度数据在前，任意长度数据在后。）下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.3.0 的 MRxx 时。）

项目	内容
[ID]	存储池的 ID 号
[BaseAddr]	存储池的基址
[Blk_Size]	存储池的块大小
[Total Blk_cnt] （总块数）	存储池的总块数
[Free Blk_cnt(map)] （空闲块数 （映射））	未使用块的数量及未使用存储块的相关信息 （位信息）

- [ID] 字段的显示内容取决于指定的是固定长度还是可选长度。
  - 如果数据为固定长度，则 [ID] 字段显示字符串 “[F]” 和存储池 ID 号。
  - 对于可选长度，第一行显示的内容是字符串 “[V]”、存储池 ID 号和块 ID 号。第二行至第四行显示存储池 ID 和块 ID 号。块 ID 号包含在括号中。
- 指定任意长度存储池时， [Total Mlk_cut] （总 Mlk_cut）字段中显示 “--”。[Free Blk_cnt (map)] （空闲块数 （映射））字段中不显示任何位信息。
- 指定固定长度存储池时， [Free Blk_cnt (map)] （空闲块数 （映射））的存储块信息中每个位的显示格式如下所示：

项目	内容
'0'	正在使用的存储块 （忙）
'1'	未使用的存储块 （就绪）
'-'	无存储块

7.12.11.2 显示存储池状态 （当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

所有存储池会按照 ID 号顺序列出。下面列出了每个项目的功能。（当实时操作系统为符合 uITRON 规范 V.4.0 的 MRxx 时。）

项目	内容
[ID]	存储池的 ID 号
[Mplatr]	每个存储池的属性
[Mpladr]	存储池的基址
[Mplsz]	存储池的大小
[Blkcnt]	固定长度存储池的总块数
[Fblkcnt]	未使用块的数量及未使用存储块的相关信息
[Memory Pool Queue] （存储池队列）	显示存储池中正在等待的任务的 ID 号和名称。

- Mplatr 区域显示内容如下：

TA_TFIFO	任务等待队列处于 FIFO 顺序
TA_TPRI	任务等待队列处于任务优先级顺序

- [ID] 字段的显示内容取决于指定的是固定长度还是可选长度。
  — 如果数据为固定长度，则 [ID] 字段显示字符串 “[F]” 和存储池 ID 号。
  — 对于可选长度，第一行显示的内容是字符串 “[V]”、存储池 ID 号和块 ID 号。第二行至第四行显示存储池 ID 和块 ID 号。块 ID 号包含在括号中。

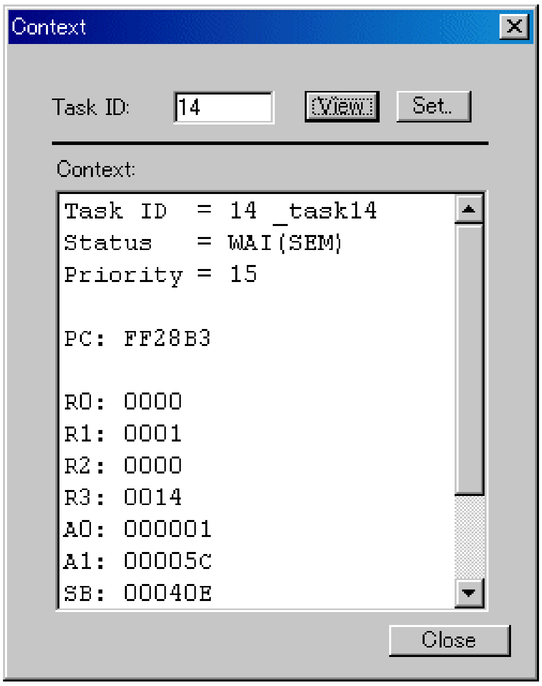
7.12.12 显示任务上下文

7.12.12.1 显示任务上下文

在 [MR] 窗口中，选择弹出菜单 - [Context...] （上下文…）。

此时会打开 [Context] （上下文）对话框。 [Context] （上下文）对话框用于引用 / 指定所指定任务的上下文信息。

也可通过双击任务状态显示模式的数据显示区来打开 [Context] （上下文）对话框。



在 [Task ID] （任务 ID）字段输入任务 ID 号并单击 [View] （视图）按钮 （或按 Enter 键）。

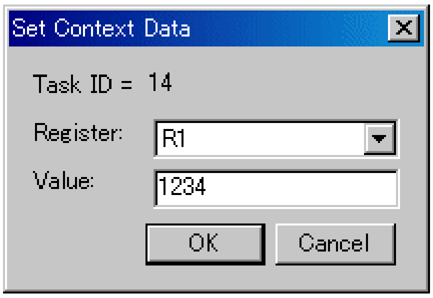
[Context] （上下文）字段中显示指定任务的上下文。

- 如果单击 [View] （视图）按钮时，在 [Task ID] （任务 ID）字段中输入的任务为 “RUN”（运行）或 “DMT”（休眠），则不显示上下文。（[Context] （上下文）字段仅显示任务 ID 和任务状态。）
- 如果单击 [View] （视图）按钮时，在 [Task ID] （任务 ID）字段中输入的任务 ID 号不存在，则会出现错误。

7.12.12.2 更改任务上下文

在 [Context] （上下文）对话框的 [Task ID] （任务 ID）字段中输入任务 ID 号并单击 [Set] （设置）按钮。此时会打开 [Set Context] （设置上下文）对话框。

[Set Context] （设置上下文）对话框用于设置指定任务的指定上下文寄存器值。



在 [Register] （寄存器）字段列表框中指定要更改的寄存器，然后在 “Value:”（值：）字段中输入要设置的值。

如果在 “Value:”（值：）字段中设置的表达式描述有误，或指定的值超出指定寄存器允许设置的范围，则会出现错误。

8 脚本命令表

我们为您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。

8.1 脚本命令表 （按功能分类）

8.1.1 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
OverStepInstruction	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

8.1.2 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
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

8.1.3 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

8.1.4 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）

8.1.5 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

8.1.6　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

8.1.7　地址中断断点设置命令

命令名称	缩写名称	内容
ADdressInterruptBreak	ADIB	设置和检查地址中断断点

8.1.8　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

8.1.9　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

8.1.10 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

8.1.11 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

8.1.12 映像命令

命令名称	缩写名称	内容
Map*	-	检查和设置映像数据

8.1.13 时钟命令

命令名称	缩写名称	内容
Clock	CLK	检查和更改时钟

8.1.14 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

8.1.15 实时操作系统命令

命令名称	缩写名称	内容
MR*	-	显示实时操作系统的状态 (MRxx)

8.1.16 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

8.2 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
ADdressInterruptBreak	ADIB	设置和检查地址中断断点
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

命令名称	缩写名称	内容
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
LoadMot*	LM	下载 Motorola S 类型格式文件
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
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）
MR*	-	显示实时操作系统的状态 (MRxx)
OverStep	O	跳过执行源行
OverStepInstruction	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

命令名称	缩写名称	内容
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
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
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
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

9 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。

9.1 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

- 不能在脚本命令所在的行上包含注释。
- 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。

9.1.1 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）。

9.1.2 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。
- 请始终在宏变量之前添加 “%” 符号。

9.1.3 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）。

9.1.4 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。

9.1.5 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。

9.2 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

9.2.1 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

9.2.2 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 M32R 的调试器、 用于 M32C 的调试器、 用于 M16C/R8C 的调试器	• 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。

9.2.2.1 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

  根据命令正在处理的程序地址，当前作用域会自动转变。

9.2.2.2 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. 作用域之外的局部标签
6. 作用域之外的局部符号

- 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

9.2.3 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 “9.1.2 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 语句的迭代次数。

9.2.4 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。

产品名称	寄存器名称
用于 M32C 的调试器	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 <- 存储体 0 寄存器 1R0、 1R1、 1R2、 1R3、 1A0、 1A1、 1FB、 1SB <- 存储体 1 寄存器
用于 M16C/R8C 的调试器	PC、 USP、 ISP、 SB、 INTB、 FLG 0R0、 0R1、 0R2、 0R3、 0A0、 0A1、 0FB <- 存储体 0 寄存器 1R0、 1R1、 1R2、 1R3、 1A0、 1A1、 1FB <- 存储体 1 寄存器

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。

9.2.5 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 4 个字节长度。）

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。

9.2.6 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。

9.2.7　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。

9.2.8　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

10 C/C++ 表达式

10.1 编写 C/C++ 表达式

使用由以下标记组成的 C/C++ 表达式，可以注册 C 监视点并指定 C 监视点的赋值。

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."...

10.1.1 立即值

可以使用十六进制、十进制和八进制的立即值。以 0x 开头的值被作为十六进制值处理，以 0 开头的值被作为八进制值处理，没有这两种前缀的值被作为十进制值处理。

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。
- 只有在指定 C/C++ 监视点的 C/C++ 语言表达式中使用立即值，或者使用立即值指定这些表达式的赋值时，立即值才有效。当使用浮点数时，不能对类似 1.0+2.0 这样的表达式执行运算。

10.1.2　作用域解析

提供的作用域解析运算符 :: 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
```

10.1.3　数学运算符

可以使用加号 (+)、减号 (-)、乘号 (*) 和除号 (/) 数学运算符。以下显示使用上述数学运算符进行计算时的优先级顺序。

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

注意

- 目前不支持对浮点数使用数学运算符。

10.1.4　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

- 立即值不能当作指针来处理。例如，不能指定 *0xE000。

10.1.5　引用

引用由 “&” 符指示。只能指定 “& 变量名称” 这种形式。

10.1.6　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 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。

10.1.7 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 都可正确地引用成员。

10.1.8 使用箭头运算符的成员引用

使用箭头符时只能通过 “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 都能正确地引用成员。

10.1.9 指向成员的指针

对于使用 “.*” 或 “->*” 运算符指向成员的指针，只能采用 “变量名称 .* 成员名称” 或 “变量名称 ->* 成员名称” 形式来引用。

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 都能正确地引用成员指针类型的变量。

注意

- 请注意，表达式 *mp 不能视为成员指针类型的变量。

10.1.10 括号

使用 “(” 和 “)” 可指定表达式中计算的优先级。

10.1.11 数组

可以使用 “[” 和 “]” 指定数组中的元素。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。

10.1.12 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。执行到指针类型转换时，还可以使用指向指针的指针和指向指针的指针的指针等。

请注意，如果未指定是否有符号，则默认值如下所示：

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

注意

- 在 C++ 基本数据类型中，不能执行到 bool 类型、 wchar_t 类型和 floating-point 类型 （浮点或双精度）的转换。
- 不能执行到寄存器变量的转换。

10.1.13 typedef 类型转换

可以通过执行到 typedef 类型 （除 C 基本数据类型以外的类型）的转换来将指针类型转换为 typedef 类型。执行到指针类型的转换时，还可以使用指向指针的指针和指向指针的指针的指针，等等。

注意

- 不能执行到 struct 类型或 union 类型的转换，也不能将指针类型转换为这些类型。

10.1.14 变量名

可以根据 C/C++ 惯例使用以英文字母开头的变量名。

变量名最长允许有 255 个字符。

可以使用 “this” 指针。

10.1.15 函数名称

可以根据 C 惯例使用以英文字母开头的函数名。

在使用 C++ 时，不能使用任何函数名。

10.1.16 字符常数

可将一些字符放在单引号 (') 中作为字符常数。例如， 'A'、 'b' 等等。这些字符常数会转换为 ASCII 码，可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。
- 只有在指定 C 监视点的 C/C++ 表达式中使用字符常数，或者在指定赋值使用字符常数时，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）。

10.1.17 字符串文字

可将一些括在双引号 (“”) 中的字符串用作字符串文字。例如，“abcde”、“I am a boy.”等等。

注意

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

10.2 C/C++ 表达式的显示格式

在 [C Watch] （C 监视点）窗口的数据显示区域中， C/C++ 表达式的显示格式为：类型名称、 C/C++ 表达式 （变量名）和计算结果 （值），如下所示。

下文分别介绍各种类型的显示格式。

10.2.1 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，则以如下形式显示：
  (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）

10.2.2 基本数据类型

- 如果计算结果为除 char 类型或浮点类型以外的基本数据类型，则以如下形式显示：
  (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）
- 如果计算结果为 char 类型，则以如下形式显示：
  (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）
- 如果计算结果为浮点类型，则以如下形式显示：
  (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）

10.2.3 指针类型

- 如果计算结果为非 char* 类型的指针类型，则以如下十六进制形式显示：

  (unsigned int *) p = 0x1234 （所有基数）

- 如果计算结果为 char* 类型，则可以在 [C Watch] （C 监视点）窗口的 [Display String] （显示字符串）菜单中选择字符串形式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

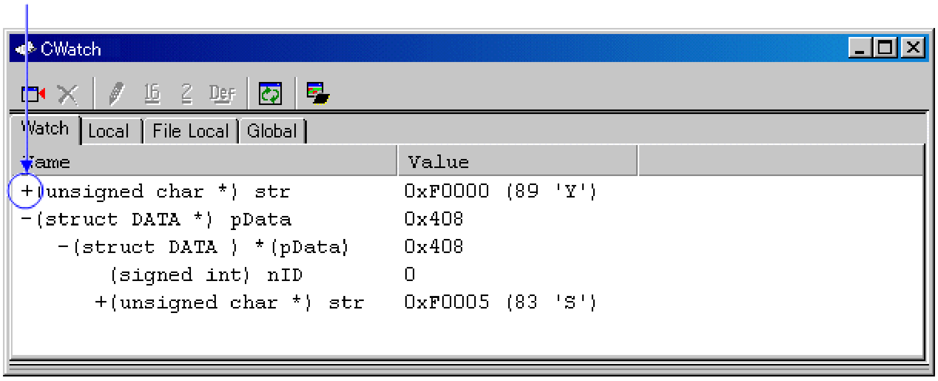
  (unsigned char *) str = 0x1234 (74 'J') （所有基数）

如果计算结果为 char* 类型，则以如下形式显示：

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

如果字符串中某个字符前包含一个非打印字符，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，且不显示右引号。

“+” 表示指针类型



可以双击由 “+” 指示的行来查看该结构或联合的成员。显示成员时 “+” 会变为 “-”。要恢复原来的显示，请双击现在由 “-” 指示的那一行。

10.2.4 数组类型

- 如果计算结果为非 char [ ] 类型的数组类型时，则以如下十六进制数形式显示起始地址：

    (signed int [10]) z = 0x1234 （所有基数）

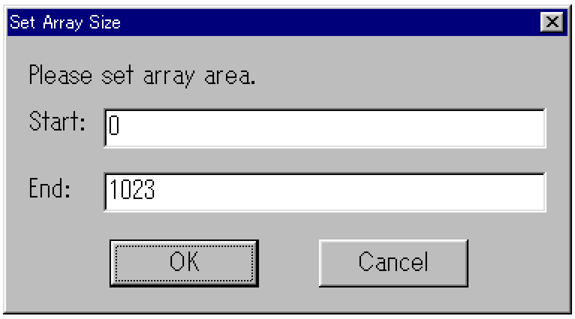
- 如果计算结果为 char [ ] 类型，则以如下形式显示：

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

如果字符串中某个字符前包含一个非打印字符，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，且不显示右引号。

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

此外，如果字符串字符数超过 80，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，在类型名称左侧会显示 “+”。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 “10.2.3 指针类型”）当数组元素个数超过 100 时，会打开下面的对话框。请在该对话框中指定元素个数。



此时会显示从 “Start”（开始）中指定的下标到 “End”（结束）中指定的下标所对应的元素。如果指定的值超出数组的最大下标，则该值被视为数组的最大下标。如果单击 “Cancel”（取消）按钮，则不会显示这些元素。

10.2.5 函数类型

- 如果计算结果为函数类型，则以如下十六进制形式显示起始地址：

    (void()) main = 0xF000 （所有基数）

10.2.6 引用类型

- 如果计算结果为引用类型，则以如下十六进制形式显示引用地址：

    (signed int &) ref = 0xD038 （所有基数）

10.2.7 位域类型

- 如果计算结果为位域类型，则以如下形式显示：

    (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）

10.2.8 未找到任何 C 符号时

如果计算表达式包含找不到的 C 符号，则以如下形式显示：

`() x = <not active>` （所有基数）

10.2.9 语法错误

- 如果计算的表达式包含语法错误时，则以如下形式显示：

  `() str*(p = <syntax error>` （所有基数）

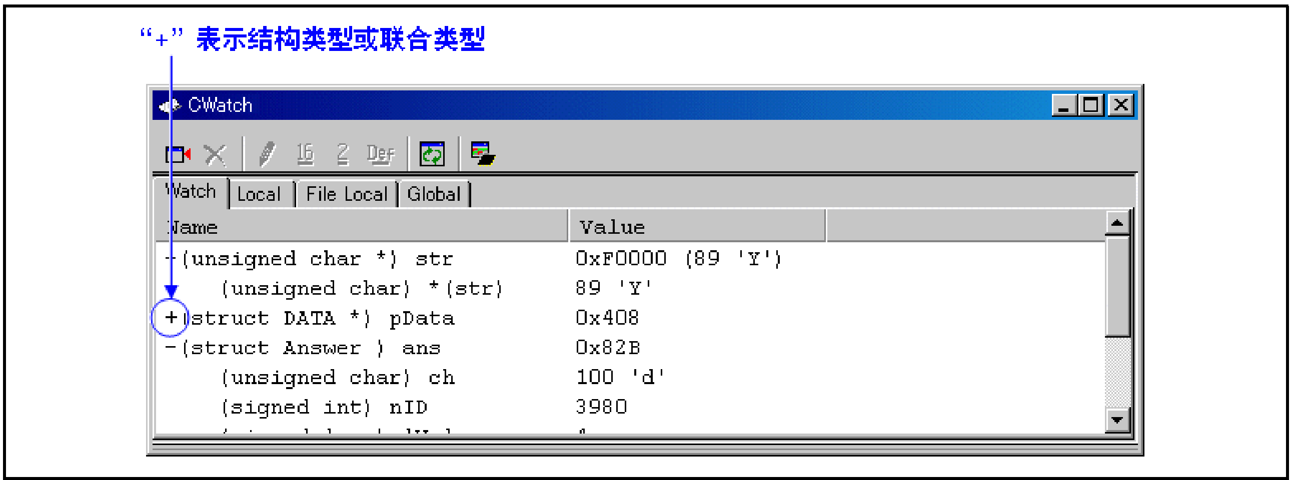
  （其中 `str*(p` 为该语法错误）

10.2.10 结构类型和联合类型

- 如果计算结果为结构类型或联合类型，则以如下十六进制形式显示地址：

  `(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）左侧会显示一个 “+”。



可以双击由 “+” 指示的行来查看该结构或联合的成员。显示成员时 “+” 会变为 “-”。要恢复原来的显示，请双击现在由 “-” 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，则不能引用该变量。

- 寄存器变量

如果计算结果为寄存器变量，则在类型名称的左侧会显示 “register”，如下所示：

`(register signed int) j = 100`

11 显示程序停止的原因

如果程序被调试功能停止，则会在 [Output] （输出）窗口或 [Status] （状态）窗口 （[Platform] （平台）页）中显示停止原因。

显示内容及 “停止原因”如下所示：

显示内容	停止原因
[Halt] （暂停）	由 [Halt Program] （暂停程序）按钮 / 菜单停止
[S/W break] （软件断点）	软件断点
[Address match interrupt break] （地址匹配中断断点）	地址中断断点
[H/W event, Combination] （硬件事件，组合）	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] （硬件事件，组合， 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] （硬件事件，状态转换，从 xx）	硬件断点，符合状态转换条件 （从 xx：上一状态 （start、 state1、 state2））
[H/W event, State transition, Timeout] （硬件事件，状态转换，超时）	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] （硬件事件，存取保护错误）	保护中断

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 “---”。

12　注意事项

12.1　常见注意事项

12.1.1　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
   — 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。

12.1.2　可设置软件断点的区域

可设置软件断点的区域因 MCU 类型而异。

12.1.2.1　适用于 M32C 的调试器

可设置软件断点的区域因处理器模式而异。

处理器模式	可设置区域
单片	内部 RAM、内部 ROM
存储器扩展	内部 RAM、内部 ROM 仿真存储器 （仅限内部区域）
微处理器	内部 RAM 仿真存储器 （仅限内部区域）

要停止仿真存储器区域以外的目标程序，请使用 COME 执行。

注意事项

- 当仿真存储器板连接到仿真器时，便可设置仿真存储器。
- 当目标程序正在运行时，不能在内部 ROM 中设置软件断点。

12.1.2.2　适用于 M16C/R8C 的调试器

可设置软件断点的区域因处理器模式而异。

处理器模式	可设置区域
单片	内部 RAM、内部 ROM
存储器扩展	内部 RAM、内部 ROM 仿真存储器
微处理器	内部 RAM 仿真存储器

注意事项

- 当仿真存储器板连接到仿真器时，便可设置仿真存储器。
- 当目标程序正在运行时，不能在内部 ROM 中设置软件断点。

12.1.3　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，则不能引用该变量。
- 不能更改寄存器变量的值。
- 不能更改 64 位宽变量 （long long、 double 等）的值。
- 不能更改未指定存储地址和大小的 C/C++ 表达式的值。
- 为了优化， C 编译器可能在同一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确显示。
- 文字字符串只能替换为 char 数组和 char 指针类型的变量。
- 不能对浮点类型执行任何数学运算。
- 不能对浮点类型执行任何取反操作。
- 不能对浮点类型执行转换。
- 不能对寄存器变量执行转换。
- 不能对结构类型或联合类型执行转换，或者不能将指针类型转换为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。
- 以下值可替换为位域。
  — 整数常数、字符常数和枚举数
  — Bool 类型、字符类型、整数类型以及枚举类型的变量。
  — 位域

  如果替换值超出位域大小，则会截断该值。
- 在 SFR 区域中分配的位域成员可能无法转换为正确的值。
- 目标程序正在运行时，不能修改局部变量和位域的值。

12.1.4 C++ 中的函数名

- 如果在设置显示地址、设置断点等时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）。
- 在对话框中指定列的地址值中，不能使用函数名指定任何地址。
- 不能在 [C watch] （C 监视）窗口中直接正确地引用成员函数的指针。

12.1.5 下载模块的选项设置

以下选项可在 [Debug Settings] （调试设置）对话框中设置，这些选项对此调试器无效：

- [Offset] （偏移量）：指定值视为 “0”
- [Access size] （存取大小）：指定值视为 “1”
- [Perform memory verify during download] （下载期间执行存储器检验）：不支持。

12.1.6 对多个模块进行调试

如果在一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。
如果在一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。

12.1.7 同步调试

同步调试功能不可用。

12.1.8 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，再次接通仿真器的电源，然后重新启动调试器。
重新下载程序。

12.2 M32C 调试器的注意事项

12.2.1 仿真器使用的堆栈区

仿真器使用中断堆栈区作为工作区 （20 字节）。

进行调试时，请分配足够的中断堆栈区，这包括常规使用大小外加 20 字节。

12.2.2 复位目标程序时的中断堆栈指针

在复位目标程序时仿真器将中断堆栈指针 (ISP) 设置为 0500h。

请记住，在实际仿真器上，中断堆栈指针 (ISP) 会设置为 0000h。

12.2.3 C 编译器 / 汇编器 / 连接器的选项

有时无法正常下载 / 调试信息，这与编译器、汇编器和连接器的选项指定有关。

请参阅以下内容来了解选项的详细说明。

请参阅 “12.4 编译器、汇编器和连接器的选项”。

M32C 调试器可使用的编译器：

- NCxx
- IAR EC++ 编译器
- IAR C 编译器

12.2.4 目标 MCU HOLD 终端

如果目标 MCU HOLD 终端设置为 LOW，则不能停止执行目标程序。请将 HOLD 终端设置为 High 后再次尝试停止目标程序。

即使 HOLD 终端设置为 LOW 的时间很短，也不能在 HOLD 终端设置为 LOW 时停止目标程序。

如果出现此情况，请尝试再次停止目标程序。

12.2.5 硬件事件

如果指定奇地址中的字长（2 字节长度）数据作为执行后续数据存取时的一个事件，则不会检测到该事件。此外，即使在位存取过程中对包含指定位的地址中的其他位进行了存取，该事件还能生效。

— 硬件断点事件

— 实时跟踪事件

- 下面是指定，数据存取事件的示例及其值对比。

地址	存取大小	16 位数据总线	8 位数据总线
偶地址	WORD 示例：mov.w #1234h, 8000h	地址：8000h 数据：1234h 数据掩码：忽略	<-
偶地址	BYTE 示例：mov.b #34h, 8000h	地址：8000h 数据：34h 数据掩码：00FFh	<-
奇地址	WORD 示例：mov.w #1234h, 8001h	不支持	<-
奇地址	BYTE 示例：mov.b #34h, 8001h	地址：8001h 数据：3400h 数据掩码：FF00h	地址：8001h 数据：34h 数据掩码：00FFh

12.2.6 CPU 重写

- 请勿重写块 0 (FFC000h - FFFFFFh)。否则，仿真器会失控。
- 启用 CPU 重写模式后，就不能使用以下调试功能：

  — 设置地址匹配中断断点

  — 在内部闪存 ROM 中设置软件断点

  — 在内部闪存 ROM 中设置 COME 执行
- 不能对 CPU 重写控制程序使用以下功能（从设置 CPU 重写模式中选择位来清除它）。否则，可能无法切换到 CPU 重写模式，或者可能无法正常读取闪存 ROM 中的内容。

  — 单步调试

  — 软件断点

  — 硬件断点

  — COME 执行

要查看由程序重写的数据，请将程序与 CPU 重写控制程序断开，然后在转储、存储器等窗口中查看。

12.3 M16C/R8C 调试器注意事项

12.3.1 小型仿真器使用的堆栈区映射

小型仿真器使用中断堆栈区作为工作区 （20 个字节）。

进行调试时，请分配足够的中断堆栈区，这包括常规使用大小外加 20 字节。

12.3.2 复位目标程序时的中断堆栈指针

在复位目标程序时仿真器将中断堆栈指针 (ISP) 设置为 0500h。

请记住，在实际仿真器上，中断堆栈指针 (ISP) 会设置为 0000h。

12.3.3 编译器、汇编器和连接器的选项

有时无法正常下载 / 调试信息，这与编译器、汇编器和连接器的选项指定有关。

请参阅以下内容来了解选项的详细说明。

请参阅 “12.4 编译器、汇编器和连接器的选项”。

M16C/R8C 调试器可使用的编译器：

- NCxx
- IAR EC++ 编译器
- IAR C 编译器
- TASKING C 编译器

12.3.4 TASKING C 编译器

当对使用 TASKING C 编译器 “CCM16”编译的程序进行调试时，位域的类型固定为 “unsigned short int”，因为 CCM16 以 “unsigned short int”形式输出位域类型的调试信息。

12.3.5 目标 MCU HOLD 终端

如果目标 MCU HOLD 终端设置为 LOW，则不能停止执行目标程序。请将 HOLD 终端设置为 High 后再次尝试停止目标程序。即使 HOLD 终端设置为 LOW 的时间很短，也不能在 HOLD 终端设置为 LOW 时停止目标程序。如果出现此情况，请尝试再次停止目标程序。

12.3.6 硬件断点功能

正在运行程序时，不会执行下列操作：

- 执行 BreakMode 命令
- 打开 [H/W break point setting] (硬件断点设置) 窗口

12.3.7　硬件事件

如果指定奇地址中的字长（2 字节长度）数据作为执行后续数据存取时的一个事件，则不会检测到该事件。此外，即使在位存取过程中对包含指定位的地址中的其他位进行了存取，该事件还能生效。

- 硬件断点事件
- 实时跟踪事件
- 下面是指定数据存取事件的示例及其值对比。

地址	存取大小	16 位数据总线	8 位数据总线
偶地址	WORD 示例：mov.w #1234h, 8000h	地址：8000h 数据：1234h 数据掩码：忽略	<-
偶地址	BYTE 示例：mov.b #34h, 8000h	地址：8000h 数据：34h 数据掩码：00FFh	<-
奇地址	WORD 示例：mov.w #1234h, 8001h	不支持	<-
奇地址	BYTE 示例：mov.b #34h, 8001h	地址：8001h 数据：3400h 数据掩码：FF00h	地址：8001h 数据：34h 数据掩码：00FFh

12.3.8　存储器空间扩展

- 存储器映像
  如果在 [MCU setting]（MCU 设置）对话框中为 [Memory Space Expansion]（存储器空间扩展）选择"4MB Mode"（4MB 模式），则存储器映像取决于 mcu 的其他设置：

处理器模式	PM13	OFS	MCU 的存取区	Bank0 - Bank5	Bank6	Bank7
存储器扩展模式	1	0	40000h - 7FFFFh	EXT	EXT	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
	0	0	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	EXT	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	EXT	---
微处理器模式	---	0	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	EXT	---
			C0000h - FFFFFh	---	---	MAP
		1	40000h - 7FFFFh	EXT	EXT	MAP
			80000h - BFFFFh	EXT	MAP	---

PM13：处理器模式寄存器 1 (00005h) 的位 3

OFS：偏移位 （数据存储体寄存器 (0000Bh) 的位 2）

EXT：存取目标系统的存储器

MAP：存取某些区域的存储器，这些区域取决于 [MCU Settingts] （MCU 设置）对话框中的 MAP 设置（INT 代表小型仿真器中的仿真存储器， EXT 代表目标系统）

存取扩展区域的存储器

在 [C watch] （C 监视）窗口、 [Memory] （存储器）窗口以及其他调试窗口中显示存储器空间扩展功能的扩展区域时，这些窗口中显示的值可能不正确。在调试窗口中不会考虑存储体寄存器的状态。

要查看此区域，请使用下面列出的脚本命令并指定 M16C 的存储体、偏移位状态和地址：

DumpByte2、 DumpWord2 或 DumpLword2

还可以使用以下命令：

DumpByte2、 DumpWord2、 DumpLword2、 SetMemoryByte2、 SetMemoryWord2、 SetMemoryLword2、 FillByte2、 FillWord2、 FillLword2、 Move2、 MoveWord2

注意

下列功能在对扩展区域进行存取时可能无法预期运行。调试器不能区分存储体信息与 BUS 信息。

— RAM 监控功能 （[RAM Monitor] （RAM 监控）窗口、 [C watch] （C 监视）窗口等）

— 覆盖率计量功能 （[Coverage] （覆盖率）窗口、 Coverage 命令等）

— 存储器保护功能 （[Protect] （保护）窗口、 Protect 命令等）

— 硬件事件 （硬件断点事件、实时跟踪事件、时间计量事件等）

关于硬件事件，可以指定所需事件的 AND (same time) 组合及存储体选择寄存器的状态，通过检查存取来收集存储体。

12.3.9 CPU 重写

- 请勿重写最后一个块。否则，仿真器会失控。
- 启用 CPU 重写模式后，就不能使用以下调试功能：
  — 在内部闪存 ROM 中设置软件断点
  — 在内部闪存 ROM 中设置 COME 执行
- 不能对 CPU 重写控制程序使用以下功能 （从设置 CPU 重写模式中选择位来清除它）。否则，可能无法切换到 CPU 重写模式，或者可能无法正常读取闪存 ROM 中的内容。
  — 单步调试
  — 软件断点
  — 地址匹配中断断点
  — 硬件断点
  — COME 执行

要查看由程序重写的数据，请将程序与 CPU 重写控制程序断开，然后在转储、存储器等窗口中查看。

12.4　编译器、汇编器和连接器的选项

我们未对其他设置进行评估，因此无法建议添加其他选项。

12.4.1　使用 NCxx 时

如果在编译时指定了 -O、 -OR 或 -OS 选项，可能因为优化方面的原因而不正常生成源代码行信息，同时导致单步执行出现异常。

为避免出现此问题，请在指定 -O、 -OR 或 -OS 选项的同时指定 -ONBSD （或 -Ono_Break_source_debug）选项。

12.4.2　使用 IAR C 编译器 (EW) 时

请按如下步骤指定工程设置：

1. IAR Embedded Workbench 中的设置

   选择菜单 [Project] （工程） → [Options...] （选项…）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）区域中，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 （示例：lnkm16c.xcl）。

2. 编辑 XCL 文件

   在 XCL 文件中添加命令行选项 “-y”。“-y”选项的指定取决于不同的产品。

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

3. 完成上述设置后建立程序。

采用上述选项以外的选项时，不会完成操作检查。请注意，除上述介绍的选项外，不推荐使用其他选项。

12.4.3 使用 IAR EC++ 编译器 (EW) 时

请按如下步骤指定工程设置：

1. IAR Embedded Workbench 中的设置

   选择菜单 [Project] （工程）→ [Options...] （选项…）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）区域中，选择 “Other”（其他）选项，然后选择 “elf/dwarf” 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 （示例：lnkm32cf.xcl）。

2. 编辑 XCL 文件

   在 XCL 文件中添加命令行选项 “-y”。“-y” 选项的指定取决于不同的产品。

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

3. 完成上述设置后建立程序。

采用上述选项以外的选项时，不会完成操作检查。请注意，除上述介绍的选项外，不推荐使用其他选项。

12.4.4 使用 IAR C 编译器 (ICC) 时

12.4.4.1 指定选项

请根据以下步骤进行编译，然后建立连接。

- 编译时
  指定 “-r”选项
- 连接前
  在进行连接时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。“-y”选项的指定取决于不同的产品。

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

- 连接时
  使用 “-f”选项指定连接器选项定义文件名。

采用上述选项以外的选项时，不会完成操作检查。请注意，除上述介绍的选项外，不推荐使用其他选项。

12.4.4.2 命令执行示例

以下示例显示如何根据不同的产品输入不同的命令

- 适用于 M32C 的调试器

```
>ICCMC80 -r file1.c<Enter>
>ICCMC80 -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm80.xcl file1 file2<Enter>
```

- 适用于 M16C/R8C 的调试器

```
>ICCM16C -r file1.c<Enter>
>ICCM16C -r file2.c<Enter>
>XLINK -o filename.695 -f lnkm16c.xcl file1 file2<Enter>
```

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

12.4.5　使用 TASKING C 编译器 (EDE) 时

请按如下步骤指定工程设置：

1. 选择菜单 - [EDE]→[C Compiler Option] （C 编译器选项）→[Project Options...] （工程选项…）。随后会打开 [M16C C Compiler Options [Project Name]] （M16C C 编译器选项 [工程名称]）对话框。
   请在此对话框中进行如下设置。
   — [Optimeze] （优化）选项卡
   请在 [Optimization level] （优化级别）中指定 “No optimization”（无优化）。
   — [Debug] （调试）选项卡
   请仅选中 “Enable generation of any debug information(including type checkeing)”（允许生成所有调试信息 （包括正检查的类型））和 “Genarate symbolic debug information”（生成符号式调试信息）。
2. 选择菜单 - [EDE]→[Linker/Locator Options...] （连接器 / 定位器选项…）。随后会打开 [M16C Linker/Locator Options [Project Name]] （M16C 连接器 / 定位器选项 [工程名称]）对话框。
   请在此对话框中进行如下设置。
   — [Format] （格式）选项卡
   请在 [Output Format] （输出格式）中指定 “IEEE 695 for debuggers(abs)”。
3. 完成上述设置后建立程序。

采用上述选项以外的选项时，不会完成操作检查。请注意，除上述介绍的选项外，不推荐使用其他选项。

12.4.6　使用 TASKING C 编译器 (CM) 时

12.4.6.1　指定选项

请在编译时指定 “-g”和 “- O0”选项。
采用上述选项以外的选项时，不会完成操作检查。请注意，除上述介绍的选项外，不推荐使用其他选项。

12.4.6.2　命令执行示例

下面显示输入命令的示例。

```
>CM16 -g -O0 file1.c<Enter>
```

[备忘录]

Rev.	发行日	修订内容	
		页	修订处
1.00	2008.01.16	-	初版发行

M16C R8C 小型仿真调试器 V.1.03 用户手册

Publication Date: Rev.1.00, Jan. 16, 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

M16C R8C 小型仿真调试器V.1.03

瑞萨电子株式会社