《VXWORKS 内核、设备驱动与 BSP 开发详解》

作者：华清远见

第 6 章 交叉调试——Debugger

本章简介

嵌入式系统调试多采用交叉调试方式。所谓交叉调试是指需要调试的程序在目标机上运行，而调试则通过主机的控制完成，也就是说，整个过程需要两台计算机。采用这种调试方式主要的原因在于，所开发的程序不能在主机系统下运行。

Tornado 为开发者提供了简单易用的可视化调试界面 Debugger，使 VxWorks 程序的调试工作变得方便简捷。

6.1 基本调试过程

与通常使用的调试器一样，Debugger 支持设置断点、单步运行、暂停运行等多种调试手段。使用过其他调试器的读者不会对这些功能感到陌生，这里我们主要介绍一些嵌入式系统 VxWorks 调试所特有的功能。

6.1.1 运行 Debugger

由于采用交叉调试方式，Debugger 必须在 Target Server 已经与目标机连接上的情况下才可以执行。于是，在连接好目标机后才可以通过工具栏上的图标启动 Debugger，调试器图标如图 6.1 所示。
Debugger 在启动时会试图连接 Target Server，如果可以正确连接，Debugger 会将其他相关的按钮置为使能状态，如图 6.2 所示。

图 6.1 工具栏中的调试器图标

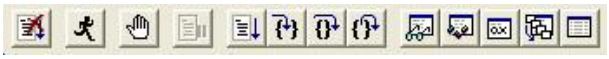

图 6.2 调试工具栏

从左至右的图标分别是停止运行 Debugger、生成任务、设置（或取消）断点、暂停任务的执行、继续执行（直到下一个断点）、单步执行（进入函数体）、单步执行（跳过函数体）、执行到函数体结束、查看指定的变量、查看局部变量、查看目标机寄存器、查看运行轨迹和查看内存。除了这些可以在工具栏找到的功能外，在 Tornado 开发环境的 Debug 菜单中还可以找到其他功能，如图 6.3 所示。
这些没有在工具栏中列出的功能包括连接任务、释放任务、释放并恢复、断点管理、添加临时断点、打开调试命令行。没有在工具栏中列出并不代表它们不重要，而仅仅表示它们不是“那么”常用，而这些功能才是我们所关心的、针对于嵌入式系统的功能。

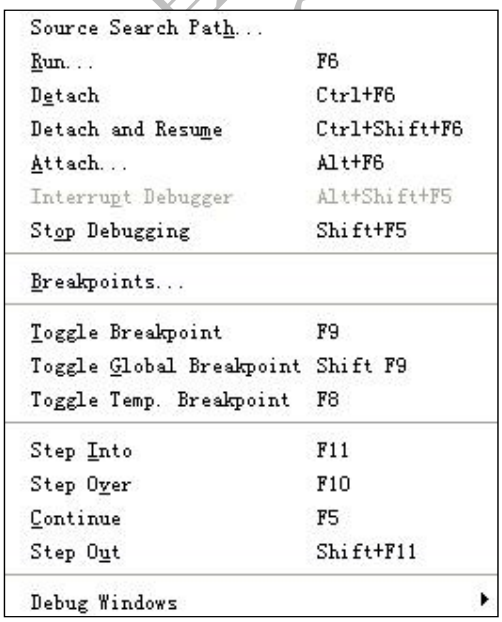

图 6.3 Debugger 菜单

6.1.2 发起任务

在 VxWorks 系统中，任务是最常见的概念，所有程序都以任务的形式被执行，所以基本的调试过程也是针对任务的。调试一个任务首先需要发起这个任务。运行任务功能（Run）即可发起一个任务。单击运行任务按钮，开发环境将弹出图 6.4 所示的窗口，用户可以在其中输入需要运行的函数及函数所需要的参数。

图 6.4　发起任务

选择好是否在函数入口点处暂停，就可以通过单击“OK”按钮执行程序了。如果函数需要参数，可以将其添加到“Arguments”一栏中，多个参数间可以使用空格隔开。利用该功能所发起的任务名称叫做 tDbgTask，优先级为 100，如下所示。

```
-> i
  NAME  ENTRY  TID   PRI  STATUS     PC      SP     ERRNO  DELAY
---------- --------- -------- --- ---------- -------- -------- ------- -----
tExcTask   _excTask   ef8de0   0 PEND        40984c   ef8ce4      0     0
tLogTask   _logTask   ef32b0   0 PEND        40984c   ef31b4      0     0
tWdbTask   _wdbTask  eee668   3 READY       40984c   eee51c      0     0
tDbgTask   _FunctionA ee92b8 100 SUSPEND     d3532e   ee9234      0     0
value = 0 = 0x0
```

6.1.3　连接任务

对于那些已经在运行的任务，Tornado 提供了另一种功能可使 Debugger 取得这些任务的控制权。
连接任务、释放任务、释放并恢复是一组相关的功能。连接任务的作用是取得某一个运行中任务的控制权，这个任务可以处于运行状态、挂起状态或休眠状态。释放任务的作用是令 Debugger 放弃对任务的控制，使其保持释放前的状态；如果释放时任务停留在一个断点处，那么它将永远停留在断点处。释放并恢复的作用在释放一个任务的同时保证任务继续执行，而不是使其处于 Debugger 释放它时所处的状态。
单击连接任务按钮，开发环境将弹出图 6.5 所示的窗口。

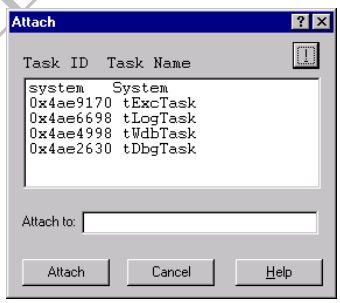

图 6.5　连接已有任务

用户可以从列表中选择需要调试的任务。图 6.5 所示窗口的第一个选项“system System”表示在系统调试模式下调试。
任务调试模式只关心本身任务的情况，即使使用断点停止了当前任务的执行，系统中的其他任务依然在运行中。而系统调试模式不同，使用系统调试模式时，一旦程序在断点处停止，整个系统的所有任务都会停止其执行。系统级调试模式的详细描述和适用场合参见 6.4 节的介绍。
连接任务命令并不像表面上一样简单，通过设置，连接任务命令可以自动连接那些出现异常的任务。
通过菜单“Tools→Options”打开开发环境选项窗口，选择“Debugger”选项卡，在“自动连接任务”栏有如下 3 种选项。

- Never——从不连接任何任务。
- Only if not already attached to any task——只有在没有连接任务的情况下才执行自动连接功能。

- Always——不论何时总是连接那些出现了异常的任务。

事实上，对于调试最有利的选项就是在没有连接任务的情况下执行自动连接功能。自动连接功能可以第一时间令用户发现程序的异常之处，从而为开发带来便捷。

6.2 断点

为了能够更好地调试程序，Tornado 2.2 所带有的 Debugger 损失了一些性能。这导致单步调试的过程中，每一步的运行速度都非常慢，因而迫使开发者在调试的过程使用大量的断点和其他辅助工具。值得庆幸的是，Debugger 提供了强大的断点管理工具。

6.2.1 设置断点

在 Debugger 下，断点包括永久性和临时性两种。永久性断点是指那些在调试过程中一直存在的断点，这些断点不会因为程序已经执行过而消失，去除断点的办法只有用户手动取消。而临时性断点只会中断程序一次，程序中断后，断点自动消失，释放对应的资源。这两种断点的图标分别是实心和空心的，如图 6.6 所示。

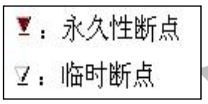

图 6.6 不同的断点标记

临时断点有助于节省资源，同时简化调试过程。调试程序时，可以在那些不重要的程序段设置临时断点。这样，在二次执行时，程序不会中断在那些已经确定正确的位置。

6.2.2 管理断点

Tornado 为开发调试提供了功能强大的断点管理器。通过断点管理器，用户可以设置断点的行为，包括中断后取消或保留、中断程序需要符合的条件、跳过断点的次数。选择菜单“Debug→Breakpoints”，弹出断点管理器窗口，如图 6.7 所示。

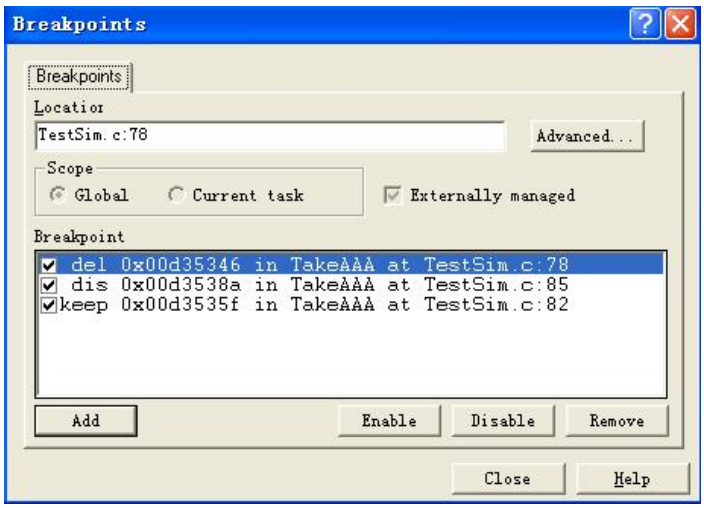

图 6.7 断点管理窗口

该窗口中有个断点列表，列表中标明了断点的属性、地址、所在函数、所在程序及程序行。

通过按钮可以执行“添加断点”、“使能断点”、“禁用断点”、“去除断点”等功能，使能和禁用断点的功能也可以通过每一个断点前面的复选框实现。实际上，添加断点属于没有什么特别用处的功能，断点管理器可以自动管理程序中添加的断点。利用这个断点列表，可以对程序中所设置的断点一目了然，使能那些有用的，禁用那些暂时不会用到的断点。如果在添加断点时，突然发现无法添加新的断点，可能是因为断点的数量到达了断点数量极限，此时禁用几个断点即可。

断点管理器还提供了断点的高级设置，通过这些设置可以实现功能复杂的断点。单击断点管理器上的“Advanced”按钮，弹出图 6.8 所示窗口。

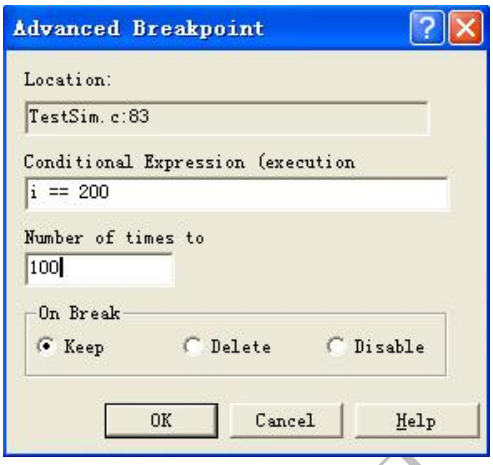

图 6.8　为断点设置激活条件

在该窗口中可以设置断点中断程序时需要符合的条件、断点中断前需要忽略的次数、断点中断后的状态等。通过这些属性的设置，可以实现功能强大的断点。窗口中当前的设置是 i 为 200 时断点才有效，然而需要跳过断点 100 次后才中断程序。

假设有如下的一个循环体，当执行到 50 次时会出现错误。

```
for(i=0; i<100; i++)
{
   /*执行某些功能*/
}
```

如果通过单步执行完成 50 次的循环将会浪费大量的调试时间，这时，只需将断点设置为忽略 49 次。如果断点需要设置在一个条件不是很确定的情况下，可以使用表达式来确定断点是否应该中断程序。例如这个例子中，完全可以在表达式中写入“i==50”，与跳过断点 49 次的效果一样。

尽管 Tornado 中断点的管理功能如此强大，但还是有其不足之处。在使用的过程中会发现，一旦目标机或 Target Server 重新启动，设置好的断点就会消失，给调试带来不便。

这种问题可以通过联合使用脚本和 Shell 来解决。设置好程序断点后，不要急于执行程序，可以首先在 Shell 中执行 b 命令将断点列表打印出来，如下所示。

```
-> b
0x00d35358: _TakeAAA + 0x30    Task:   0xee92b8  Count:  0
0x00d35374: _TakeAAA + 0x4c    Task:   0xee92b8  Count:  0
value = 0 = 0x0
```

然后，将这些断点保存为文本文件，并修改成如下的形式。

```
b 0x00d35358
b 0x00d35374
```

这样，就形成了一个脚本文件。此后，每次调试程序时，首先执行该脚本文件即可将断点设置好。

这样的办法能够方便地设置断点，缺点是其指定的地址是不可变化的。一旦程序发生了改动，这些断点的位置将发生变化。

6.3 独特的查看功能

针对不同的体系架构，Debugger 的一些功能是不同的，这些功能提供的是架构相关的内容。在调试 BSP 或驱动程序时，这些功能尤为重要。

6.3.1 源代码与汇编混合查看

Debugger 可以在调试的过程中提供汇编程序与源代码的对照。这项功能在调节程序的性能、查找硬件错误方面具有不可比拟的优越性。

通过 Tornado 开发环境的菜单“View→Mixed Source and Disassembly”，可以选择查看汇编程序与源代码的对照表，如图 6.9 所示。

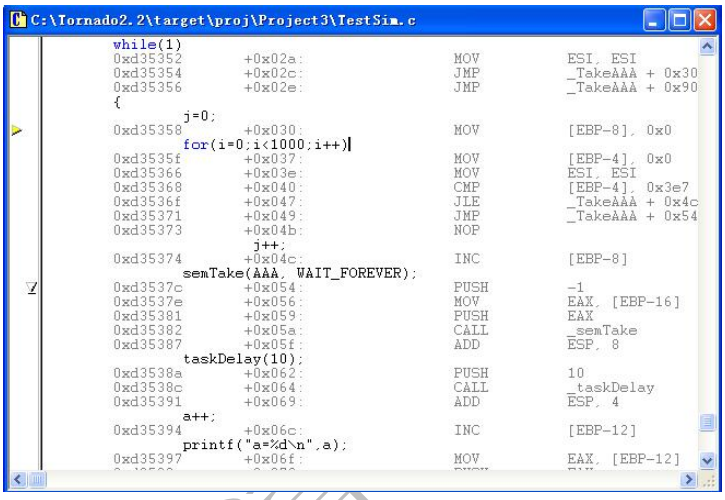

图 6.9 汇编与代码混合显示

可以看到，每一句 C 语言程序都对应着一行或若干行汇编代码。

如果某一段程序性能较差，可以使用该功能查看 C 代码与汇编代码的关系，通过分析汇编代码找到 C 代码冗余之处，从而进行修改。

如果硬件出现问题，程序可能会在某一段程序“跑飞”，这时可以查看这一段程序与汇编代码的对照，找出其中原因。

6.3.2 调试命令行

为了获得更加强大的调试功能，Tornado 还提供了调试命令行窗口，用于手动输入调试命令，如图 6.10 所示。

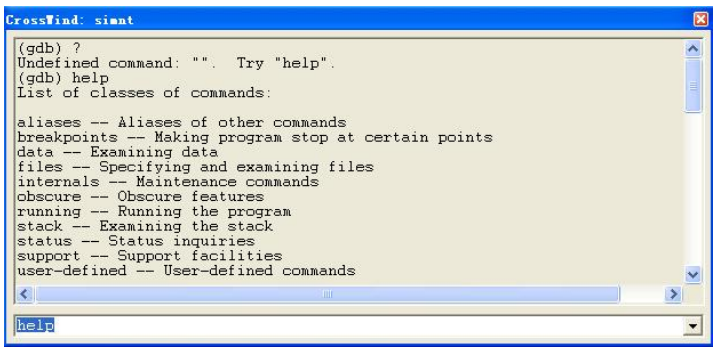

图 6.10 调试命令行

调试命令行提供了多种命令以手动操作调试器。说实话，调试命令行虽然强大但并不方便，大多数功能已经可以通过图形化界面或 Shell 命令完成，调试命令行的存在几乎可以忽略。
调试命令行可以实现自动化调试、执行 tcl 表达式、Shell 命令、自定义调试等功能。
如果需要在调试命令行中执行 Shell 命令，可以在 Shell 命令之前加“wind-”前缀。例如，执行 i 命令显示任务信息，则可以通过执行“wind-i”实现。
调试命令行可以执行 tcl 语言的表达式。例如，执行下述命令可获得 tcl 的版本号。

```
tcl info tclversion
```

6.3.3 其他功能

Tornado 手册中对于 Debugger 有更详细的介绍，包括如何执行自动调试，如何通过创建 tcl 描述文件修改 Debugger 的行为，等等。有兴趣的读者可以自行查阅。

6.4 系统级调试设置

系统调试模式区别于任务调试模式之处在于，系统调试模式中断程序的同时会中断整个系统的执行。这对于中断服务程序等较为特殊的程序调试尤为重要。
Target Server 使用 WDB 的 END 连接方式和串行端口连接方式都可以支持系统调试模式。执行“Attach”命令打开连接任务窗口，选择“system System”即可将系统中断于 wdbSuspendSystemsHere()处，进入系统调试模式。在系统级调试模式状态下，所有的任务都作为线程存在，用户可以单独执行某一个线程或者为某个线程设置断点。
系统级调试模式的调试过程与任务级调试基本相同。主要不同之处如下。

- 系统级调试可以在多个任务的源程序中设置断点，系统执行过程中遇到任何一个断点都会中断整个系统的执行。任务级调试遇到断点时只停止当前任务的执行，其他任务不受影响。
- 系统级调试支持中断服务程序的调试，任务级调试只能调试任务。

系统级调试模式下中断系统时，如果使用 Shell 的 i 命令查看任务，将会发现所有任务都处于悬挂状态，如下所示。

```
-> i
  NAME    ENTRY     TID   PRI  STATUS    PC      SP    ERRNO  DELAY
-------- ------------ -------- --- ---------- -------- -------- ------- -----
tExcTask   _excTask    ef8de0   0 PEND        40984c   ef8ce4      0     0
tLogTask   _logTask    ef32b0   0 PEND        40984c   ef31b4      0     0
tWdbTask   _wdbTask   eee668   3 PEND       40984c   eee51c      0     0

Agent mode    : Extern
System context : Suspended
value = 0 = 0x0
```

这里，系统级调试模式也叫做外部调试模式（Extern）。
系统级调试模式的优势在于其可以同时控制多个任务，可以通过调试确定这些任务之间的关系是否与设计相符。
系统级调试的另一个重要功能就是查看中断服务程序的执行。在系统级调试中，中断服务程序同样被看作一个线程，在中断服务程序中设置断点与在普通任务中设置断点是相同的。

联系方式

集团官网：www.hqyj.com 嵌入式学院：www.embedu.org 移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn 物联网学院：www.topsight.cn 研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218