Tornado 2.2 使用指南

BNC Technologies Co, Ltd.

目 录

1 概述

VxWorks 是美国 Wind River System 公司（ 以下简称风河公司，即 Wind River 公司）推出的一个实时操作系统。Wind River 公司组建于 1981 年，是一个专门从事实时操作系统开发与生产的软件公司，该公司在实时操作系统领域被世界公认为是最具有领导作用的公司。

VxWorks 是一个运行在目标机上的高性能、可裁减的嵌入式实时操作系统。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通讯、军事演习、弹道制导、飞机导航等。在美国的 F-16、FA-18 战斗机、B-2 隐形轰炸机和爱国者导弹上，甚至连 1997 年 4 月在火星表面登陆的火星探测器上也使用到了 VxWorks。

1984 年 Wind River 公司推出它的第一个版本--VxWorks 1.0.1，在 1999 年推出了它的最新版本 VxWorks 5.4。从 1995 年以后，Wind River 公司推出了一套实时操作系统开发环境 Tornado。

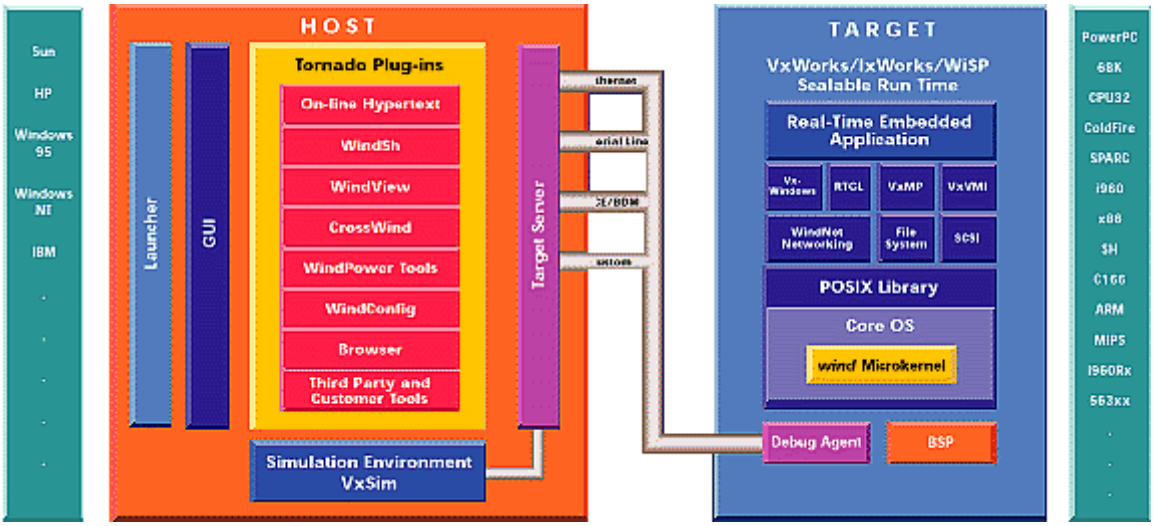
1.1 Tornado 2.2 组件

Tornado 是嵌入式实时领域里最新一代的开发调试环境。Tornado 给嵌入式系统开发人员提供了一个不受目标机资源限制的超级开发和调试环境。Tornado 包含三个高度集成的部分：

. 运行在宿主机和目标机上的强有力的交叉开发工具和实用程序；

. 运行在目标机上的高性能、可裁剪的实时操作系统 VxWorks；

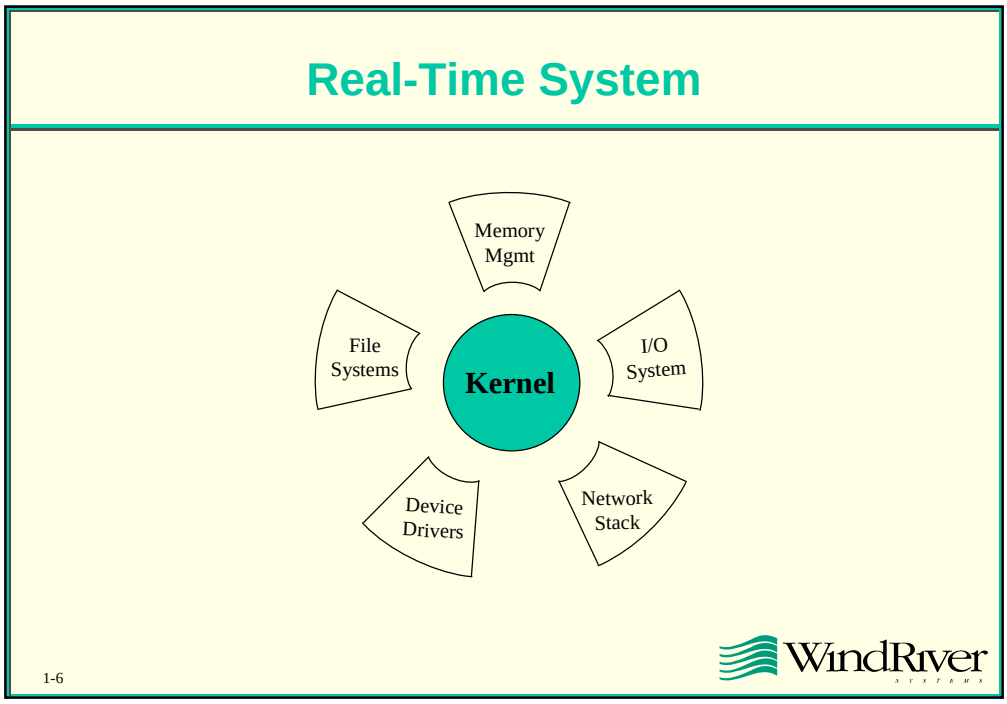
. 连接宿主机和目标机的多种通讯方式，如：以太网，串口线，ICE 或 ROM 仿真器等。



对于不同的目标机，Tornado 给开发者提供一个一致的图形接口和人机界面。当使用 Tornado 的开发人员转向新的目标机时，不必再花费时间学习或适应新的工具；对深嵌入式应用开发者来说更重要的是，Tornado 所有的工具都是驻留在开发平台上的。在嵌入式系统工具发展历史上，Tornado 是第一个实现了当目标机资源有限时开发工具仍可使用而且功能齐全的开发环境。另外，所有工具都通过一个中央服务器（Target Server）处理与目标机的通讯。所以无论连结方式是 Ethernet，还是串口线、ICE 仿真器、ROM 仿真器或客户设计的调试通道，所有工具均可使用。

1.1.1 实时系统

VxWorks 的组成如下图所示：



特点：

- 实时，能满足严格的时间要求；
- 多任务内核：
  实时调度（基于优先级或抢占式的）；
  任务间通讯；
  互斥；
- 其他标准设备作为库的形式支持内核；
- 系统可重新配置，根据需要裁减组件，空间小；
- 所有任务驻留在同一地址空间（任务间通讯快而容易，上下文切换快，但一个任务崩溃会影响别的任务）；
- 所有任务运行在超级（supervisor）模式（没有哪个系统调用具有压倒一切的优势，所有的设备都作为普通子程序调用）；
- 在主机上编写代码和编译；在目标机上进行调试和执行；

1.1.2 Tornado 开发工具

- CrossWind:

  源程序（C 或 C++ 以及汇编程序等）的调试工具。 CrossWind 结合了图形方式和命令行方式的最大特点。最普通的调试方式，例如断点设置和程序执行控制，可 以通过便捷的点击方式实现。同样，程序显示框和数 据侦察窗也提供了一个直接的可视窗口来观察应用程 序中最关键的一部分。如果需要更复杂的调试，CrossWind 也提供了命令行的方式来调用它提供的各种命令。

- Browser:

  可对系统对象（任务、消息队列、信号量等）和存储 器使用情况进行观察的浏览器。可以方便地监视用户的 目标系统。Browser 汇总了应用进程，内存消耗和一个目 标内存的映射。通过 Browser，用户可以观察信号量、 消息队列、内存分配、看门狗计时器、堆栈使用情况、 目标 CPU 使用率、对象模块结构和符号表以及每个任务的详细信息。

- WindSh:

  提供从宿主机到目标机之间的一个命令 shell。WindSh 是一种非常受欢迎的开发工具，它具有很强的交互性和可操作性，允许用户调用内存中的应用程序模块或是 VxWorks 模块中的任何例程。它不但具有一般命令语言的功能，而且也具有 C 语言的设计特点，能够解释几乎任何 C 语言表达式， 执行大多数 C 语言算子，解析符号表数据。对初用者来说，WindSh 学习起来比较简单，使用比较方便，对熟练用户而言，则有较为高级的手段可以应用。

- WindView:

  非常出色的系统可视诊断和分析工具。可非常容易地观察各任务，中断程序之间的相互作用。它是在嵌入式系统应用开发期间的可视工具。

- VxSim:

  快速原型仿真器。可在硬件设备未完成之前，在宿主机上对应用程序进行仿真分析。

- Project Facility:

  工程管理工具，配制应用程序或 vxworks 本身。

注：大部分的工具都能通过使用 Tcl 进行定制，Tcl（Tool Command Language），一种类似于 Bourne Shell 和 C 语言的脚本语言。

1.1.3 Tornado文件目录

VxWorks 的大部分是与机器无关的，相关的部分如下(需要有相应的安装文件)：

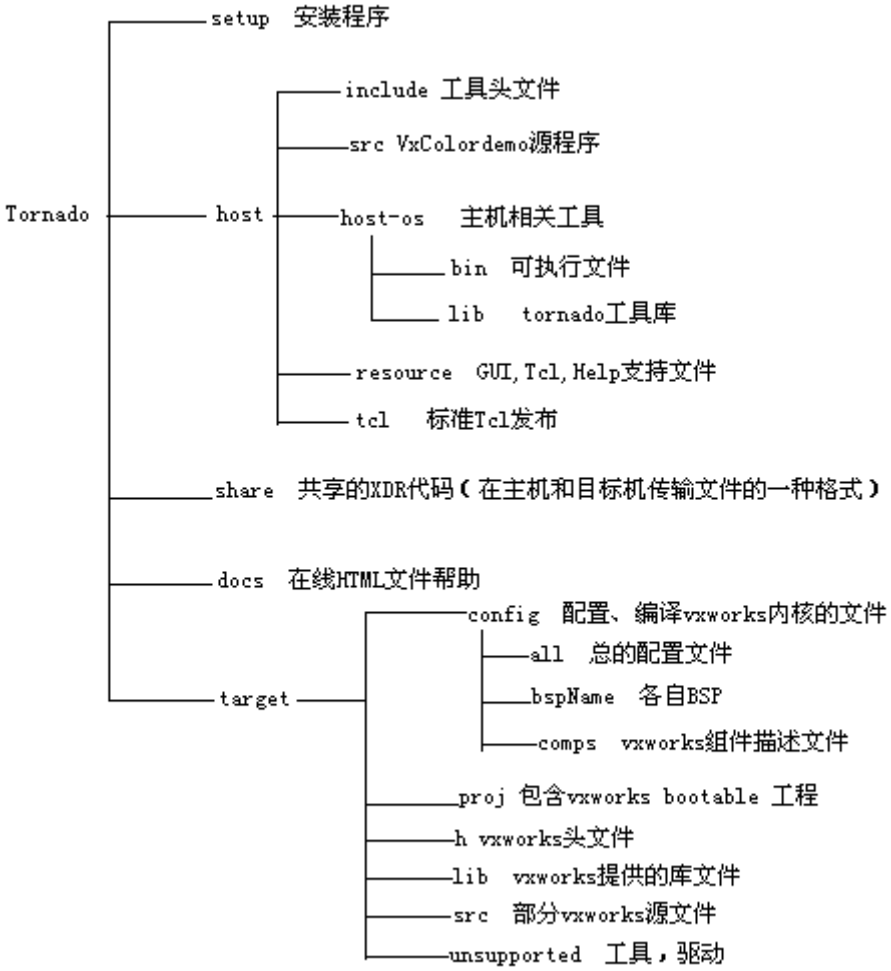
1) 体系结构不同（如 MC680x0, PPC, i960, x86, ARM），所以有相应的 Tornado for PPC，
   Tornado for x86，etc；
2) BSP 不同（表现在硬件初始化，时钟/定时器，VMEbus 接口等）,所以在 Tornado for x86 里有 386,486,pentium 等各种 BSP；
3) 编译器不同（可选），target server 不同。

注： BSP 是 Board Support Package 的缩写，该术语通常用于嵌入式领域，主要指在开发嵌入式应用时系统开发商提供的各种驱动支持库。不过该术语即使在嵌入式领域人们对它的理解也有一些不同，有的认为它就是驱动程序，有的认为它是 OS 的驱动程序，也有认为它就是 HAL（HardWare Abstract Layer ）。实际上这几种理解都只是侧重于某个部分，再由于每个嵌入式系统提供商都根据自己的系统而提出对 BSP 的不同理解，因此

在涉及到 BSP 的具体涵义时人们往往有一种似是而非的感觉。嵌入式系统提供商的龙头老大 WindRiver 公司对 BSP 的理解偏向于是 OS 的驱动程序（注：从其 BSP 的文档中可以看出）因为嵌入式系统中的各种设备的确名目繁多，因此将 BSP 定位于 OS 的驱动的确有一定的道理。对于认为 BSP 就是驱动程序的人来讲，估计他们通常是接触的嵌入式系统提供商提供的某种应用解决方案的应用系统（Total Solution）。在这种开发系统中 BSP 完全有理由被认为是所有驱动程序，因为开发人员没有必要自己去开发驱动程序，而只是验证驱动程序在自己的系统中是否正确了事。对于开发嵌入式 OS 的人来讲，似乎将 BSP 看成是对硬件平台的抽象层(HAL)和 CPU 的驱动程序更恰当。因此各种理解都有一定的道理，但由于出发点不同，对 BSP 的理解都有失全面甚至有错误的地方.

所有的人肯定对搭积木都有一定的了解，可以用各种简单的图形积木搭建成各种物体。在程序设计的世界中人们一直希望能够利用一些可重复使用的基本程序单元来构建自己的程序或者系统。在这方面已经有了一些比较成功的案例：各种标准共享库、标准程序组件等的广泛使用。但是这些成功的案例都有一个共同的特点：都是不基于任何硬件平台的程序。当开发某个平台的、与硬件相关的程序时，往往不得不从设置某个寄存器的某个位开始编程。在嵌入式领域，这种情况更为明显。在嵌入式领域中，几乎所有的设备控制和各种协议控制都在同一个嵌入式 CPU 当中，非常有利于对 CPU Core 和设备进行抽象。如果能对 CPU Core 和设备的各种控制进行抽象，人们在移植 OS 或者开发驱动程序时就没有必要对 CPU 进行非常深入的了解，不必要了解某个寄存器的某个位是控制什么的，也没有必要了解怎样初始化某个控制寄存器等等。因此 BSP 是一种能为程序开发人员提供对硬件进行描述性操作的开发支撑库。描述性操作是指在控制硬件时只需知道要完成什么，而不需要知道如何去完成，每个操作都是一些单一的动作。例如：对于设置一个串口的波特率，只需要知道是那个串口，波特率是多少，而不需要知道要写那一个寄存器以及如何写等。在利用 BSP 编写 Driver 时，编程人员只需要了解该 Driver 的初始化顺序以及初始化的内容而不需要了解初始化的具体细节就能完成驱动程序。显然可以大大的提高工作效率，并且对于硬件的具体细节设置是在驱动程序中最容易出错的地方，而利用 BSP 支撑库则可以大大的减少出错的可能性。在 BSP 支撑库中除了对硬件的描述性操作部分的代码外，还包含了对目标板的初始化部分、中断管理部分以及一些简单的驱动程序程序单元。这样的 BSP 可以不用依赖于任何的操作系统和驱动程序，但是可以作为操作系统和驱动程序的开发支撑库，可以非常方便的移植或者开发 OS 与驱动程序。在最好的情况下，OS 与驱动程序的移植只需要更换相应平台下的 BSP 支撑库就完成了移植。

下面看 Tornado 的文件树：



上图给出了安装后的 Tornado 文件夹的内容和意义，需要解释的是：

1） host-os 目录根据你的主机硬件和操作系统而自动命名，如

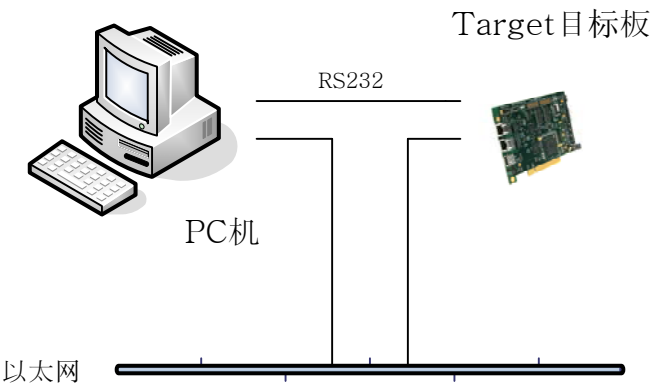
x86-win32	所有的 windows 系统主机
sun4-solaris2	Solaris 操作系统主机
parisc-hpux10	HP-UX 主机

2） ./target/config/all 包含了 boot 程序和 vxWorks 开始的源代码，是非常重要的文件，后面会有专门介绍，本人建议仔细浏览一下程序；

3） 对于每个库文件（.lib），都有相应的包含头文件，编程时需要用上，对于 vxworks 系统有哪些库文件，需要有个印象，如：

Library	Routine	Include files
---	---	---
taskLib	taskSpawn	taskLib.h
memPartLib	malloc	stdLib.h
semLib	semTake	semLib.h
lstLib	lstGet	lstLib.h
sockLib	send	sockLib.h

1.2 软/硬件配置



上图是一个典型的开发环境。

Tornado 下，一个开发环境的建立，需要以下的步骤：

- 配置好目标机硬件；
- 定义主机环境变量，配置好网络软件；
  在\x86-win32\bin 下，有 torVars.bat 脚本程序，用来设置环境变量，网络要配置好 TCP/IP 协议，修改主机 hosts 文件（见 booting 部分）；
- 开始 Tornado 注册器，wtxregd (管理目标服务器的工具)；
- 指明目标机启动参数（boot parameters）并启动目标机（后面介绍）；
- 配置并启动目标服务器（target server）,目标服务器是管理主机开发工具和目标机通讯的；

1.3 booting过程介绍

对于目标机来说，在启动的时候需要有一个 boot 程序，boot 程序的主要功能是引导 vxworks 内核,所以 boot 程序需要知道 vxworks 的内核存放在何处,通过什么手段去获取。在 vxworks 缺省的 boot 程序里有一条内建的 default boot line,它指明了获得 vxworks 内核的途径，在 boot 程序启动时，它先寻找 NVRAM 里面有无 boot line，如没有，则等里的内容添入 BOOT_PARAMS 结构里，然后，启动程序和 vxworks 内核利用此结构寻找启动参数。

```
typedef struct   /* BOOT_PARAMS */
{
    char bootDev [BOOT_DEV_LEN];    /* boot device code */
    char hostName [BOOT_HOST_LEN];  /* name of host */
    char targetName [BOOT_HOST_LEN];    /* name of target */
    char ead [BOOT_TARGET_ADDR_LEN];    /* ethernet internet addr */
    char bad [BOOT_TARGET_ADDR_LEN];    /* backplane internet addr */
    char had [BOOT_ADDR_LEN];       /* host internet addr */
    char gad [BOOT_ADDR_LEN];       /* gateway internet addr */
    char bootFile [BOOT_FILE_LEN];  /* name of boot file */
    char startupScript [BOOT_FILE_LEN]; /* name of startup script file */
```

```
    char usr [BOOT_USR_LEN];        /* user name */
    char passwd [BOOT_PASSWORD_LEN];    /* password */
    char other [BOOT_OTHER_LEN];    /* available for applications */
    int  procNum;           /* processor number */
    int  flags;             /* configuration flags */
    int  unitNum;                       /* network device unit number */
} BOOT_PARAMS;
```

下面看**boot line**结构：

bootDev / 设备名，以太网为motfcc，flash为tffs0
unitNum / 设备单元号，一般指为0
procnum / cpu的处理器号，一般为0
flags / 标识，十六进制数，意义如下：
0x01: 关闭对处理器0的系统控制
0x02: 将局部symbols和全局symbols装入目标机symbols表
0x04: 禁止自动启动（即由用户输入boot line）
0x08: 快速boot(不计数等待用户输入)
0x40: 使用BOOTP or DHCP client
0x80: 使用TFTP获取image,否则使用RSH或FTP，用FTP时pw不为空
0x100: 使目标机登记为一个代理ARP client
ead / 目标机ip地址，此值如为空，网络接口不被帮定
bad / 背板接口
had / 主机ip地址
gad / 网关地址，如果主机和目标机不在一个局网里，需要
bootFile: / 存放vxworks image的路径
usr: / 使用FTP或RSH时的用户名
passwd: / ftp password
other: / 从网络启动时此值可为空，当从软盘或硬盘启动时，如果此值为你的网络设备，boot会为你绑定网络设备
hostname: / 主机名,任意
targetName: / 目标机名
startupScript: / 脚本名，在boot以后的target shell里执行

在boot line中，e,b,h等等参数都不要求次序，你也可以让它为空值，如”pw= ”就是指口令为空参数，看一个例子：

```
    boot device           : motfcc
    unit number           : 0
    processor number      : 0
    host name             : host
    file name             : vxWorks
    inet on ethernet (e)  : 192.168.2.97     (目标板的 IP 地址)
    host inet (h)         : 192.168.2.42     (FTP Server 的主机 Host 的 IP 地址)
    user (u)              : vxworks          (FTP Server 用户名)
    ftp password (pw)     : vxworks          (FTP Server 用户密码)
    flags (f)             : 0x0
```

这个例子表示，从motfcc0网口启动。登录到指定Host主机上的FTP Server，使用vxworks:vxwokrs用户/密码登录并下载vxWorks。

在vxworks里面，针对每一种的BSP（什么叫BSP？参看本文上），都有各自的配置文件，在C:\Tornado\target\config*bspname*\config.h里， 打开config文件，就会看到上面所说的default_boot_line，接下来要做的就是根据你的情况修改此行参数，使之符合自己的要求。在实时应用系统的开发调测阶段，往往采用以PC机作为目标机来调测程序。主机PC和目标机PC之间可采取串口或是网口进行联结。由于大多数目标已配有网卡，网络联结成为最简单快速的连接方式。下面是它的详细步骤：

1) 修改通用配置文件\\Tornado\target\config\bspname\config.h.针对不同的网卡，其名称不同，如NE2000及其兼容网卡为ENE，3COM以太网卡为ELT，Intel网卡为EEX，Intel82559网卡为fei ，3C905B PCI网卡为elPci。（以3COM以太网卡为例）
2) 针对目标机的网卡，#define INCLUDE_ELT，同时 #undef 其它网卡
   在 config.h 文件中修改相应网卡类型（如网卡为 3COM 网卡）的定义部分：
   #define IO_ADRS_ELT 网卡 I/O 地址
   #define INT_LVL_ELT 网卡中断号
3) 修改#define DEFAULT_BOOT_LINE 的定义：
   #elif (CPU_VARIANT == PENTIUM) （修改此行后的 DEFAULT_BOOT_LINE）
   #define DEFAULT_BOOT_LINE \
   "elt(0,0)主机标识名:vxWorks h=主机 IP e=目标机 IP u=登录用户名 pw=口令 tn=目标机名"
   例如：#define DEFAULT_BOOT_LINE \
   "elt(0,0)comps:VxWorks h=10.132.101.88 e=10.132.101.82 u=x86 pw=xxx tn=x86"

注意：对于PCI网卡，无需步骤2，即不用修改网卡的I/O地址和中断号。

4) 制作启动软盘：
   - ✧ 准备一张已格式化的空盘插入软驱；
   - ✧ 在Tornado集成环境中点取Build菜单，选取Build Boot Rom，选择对应的BSP，选择Image为bootrom_uncmp，OK。
   - ✧ 进入DOS命令提示符，执行命令\tornado\host\x86-win32\bin\torvars（建立命令行环境）；改变目录到\tornado\target\config\pcpentium; 执行命令 mkboot a: bootrom_uncmp

注：image文件的种类
vxWorks_rom ：可以写到 ROM 的、没有带符号表和 Shell 的、没有压缩的 vxWorks。
vxWorks.st ：带有符号表的 vxWorks。
vxWorks.st_rom：可以写到 ROM 的、带有符号表和 Shell 的、压缩的 vxWorks。
vxWorks.res_rom ：可以写到 ROM 的、带有符号表和 Shell 的、只有数据段拷贝到内存的没有压缩的 vxWorks。
vxWorks.res_rom_nosym：可以写到 ROM 的、只有数据段拷贝到内存的、没有压缩 vxWorks。
bootrom：压缩的 bootrom 。
bootrom_uncmp：没有压缩的bootrom 。

附：硬盘启动方法：

1.在config.h中将INCLUDE_ATA包括进去；

2 .修改config.h里的

#define DEFAULT_BOOT_LINE "ata=0,0(0,0)host:/ata0/vxWorks "

3 .将目标硬盘格式化，用vxsys.exe（在host/x86_win32/bin/下）做盘；

4. 把bootrom.sys做在软盘上，拷到目标硬盘上；

5. 把你生成的vxworks文件拷到目标硬盘上；重新启动就OK。

1） 新建Bootbal工程（主要目的是提供新的定制过的内核）

在 Tornado 环境中新建 Bootbal 工程：

在第一步中设定“Location”为 c:\myprojects\BootPen\Project0；

在第二步中选择“A BSP”为你的 BSP, 如 pcPentium；

Build新建的工程，生成VxWorks。

2） 启动Tornado组件FTP Server (Host)[注：如果想从软盘加载内核，不需此步，只要将上步生成的vxworks文件拷入软盘即可]

启动 Tornado 组件 FTP Server，在 WFTPD 窗口中选择菜单 Security 中的 User/right...,在其弹出窗口中选择 New User...,根据提示信息输入登录用户名和口令，用户名为 x86，密码为 xxx；

指定下载文件 vxWorks 所在根目录，在此为 c:\myprojects\BootPen\Project0

还必选取主菜单 Logging 中 Log options,使 Enable Logging、Gets 、Logins 、Commands 、Warnings。

最后，将系统制作盘插入目标机软驱，加电启动目标机即可通过FTP方式从主机下载VxWorks系统。

建议：如果采用网络从主机FTP下载vxworks image，需用交叉网线，我所大网不行。

下面是booting过程中的一些细节：

1. 正如上面所说的，开始时要等待用户输入boot 参数，缺省时间是7秒，这可通过修改bootconfig.h文件来修改缺省时间，或给flag附值来改变（如快速启动），在7秒内如没有输入，就读取default-boot-line的启动参数。建议手工输入boot 参数，方法为：输入：$dev (0,0) host:/file h=# e=# g=# u=# ……，这样就不需要一次次的做启动软盘，做一次就够了。同boot-line格式一样。通过输入p,可列出启动参数，如：

```
[VxWorks Boot]: p

boot device          : motfcc
unit number          : 0
processor number     : 0
host name            : host
file name            : vxWorks
inet on ethernet (e) : 192.168.2.97
host inet (h)        : 192.168.2.47
user (u)             : vxworks
ftp password (pw)    : vxworks
flags (f)            : 0x0
```

2．整个启动界面大致如下图：

如果你在自己的启动过程中和上面的不一样，可能的原因有几种情况：

1）在第一个 Attaching network interface….没有 done 结尾：可能网卡类型不对；

2）Loading….没有出现 vxworks image 的大小：可能网线或连接有问题；

3）Loading…54677…没有第二和第三部分：要用交叉网线；

4）Starting at…后没有进一步的指示：可能 image 无效；

3．重起的方法：

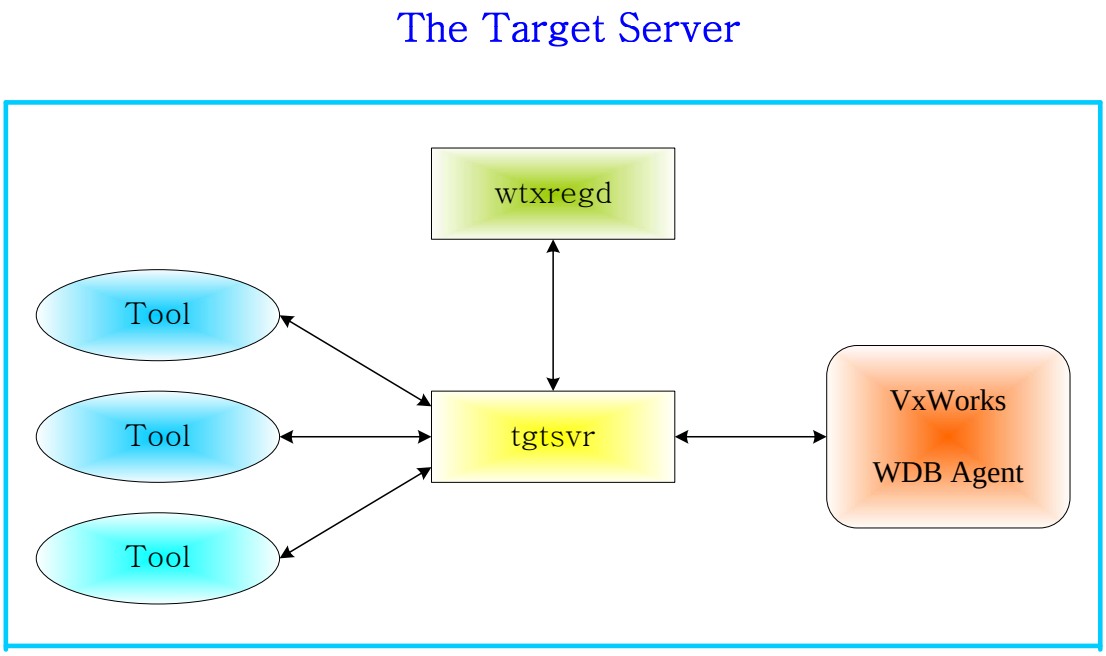
在 WindSh 里则输入 reboot;

在目标机上按 CTRL-X

1.4 建立tornado开发环境

在目标机起来以后，只要再将主机上的 target server 配置并启动起来后，主机同目标机的通讯就建立起来了，整个开发环境就已搭好了，接下来就是生成程序并下载到目标机运行了。

我们先看一下 taraget server 的作用：

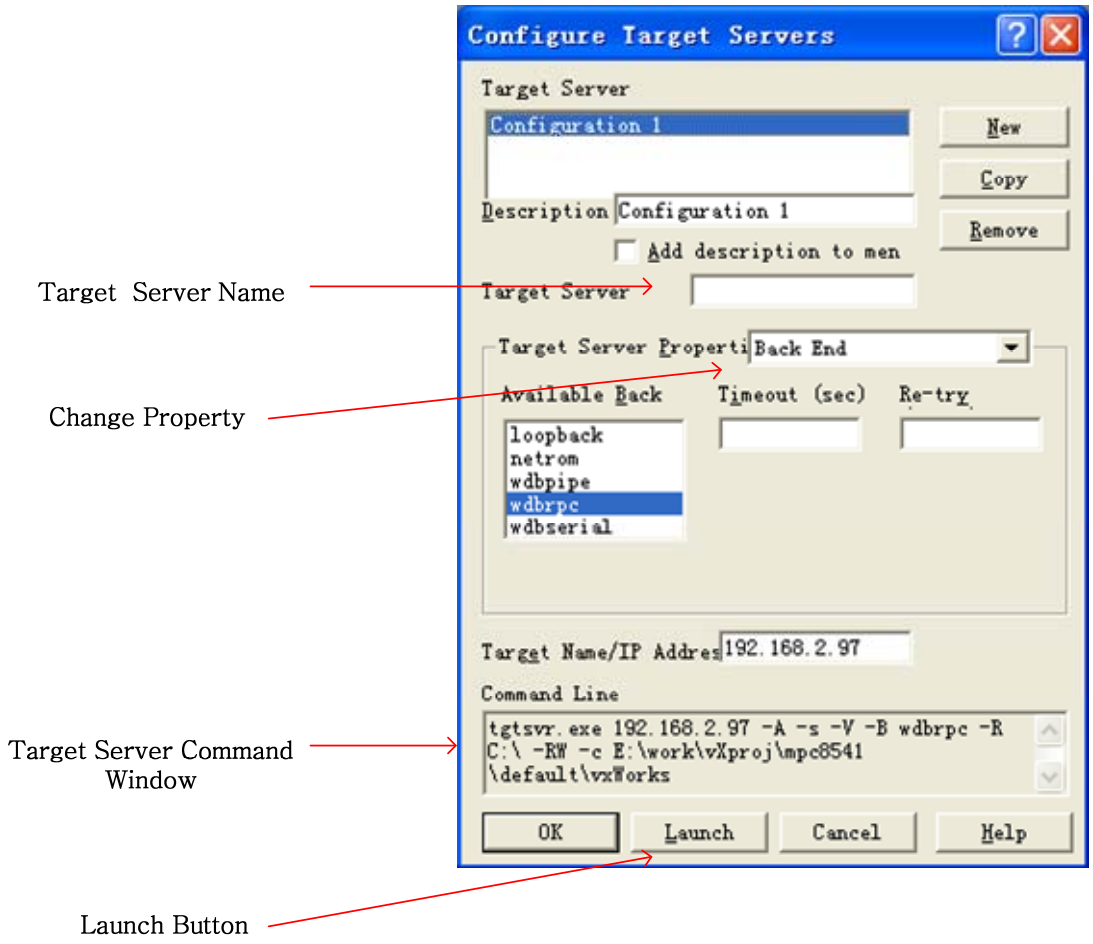


1)、与目标机上的 debug agent 通讯；
2)、模块的动态装载和卸载；
3)、主机驻留的符号表；
4)、为主机工具在目标机上安排内存；
5)、缓存目标机 RAM 里的程序代码段；
6)、虚拟 I/O 设备管理；

所有的 Tornado 工具使用 WTX (Wind River Tool Exchange Protocol)同 target server 通讯，target server 再将请求消息送往目标机上的 WDB 代理，这就避免了主机工具频繁地访问 target.

注：wtxregd(注册器程序)，必须在 target server 之前启动，主要管理工具需要同 target server 连接的信息。

下面是具体的一个配置过程：



运行 Tornado(tornado.exe)

执行菜单命令 Tools | Target Server | Configure，弹出目标服务器设置对话框，点击 New 产生一个新的配置。

设置 Description 域（可以任意设置）；

设置 Target Server 域（可以任意设置）；

在 Change Property 域选取 **Back End 项**（该项设置主机与目标机如何连接，缺省为网口连接，如果使用串口连接，需要修改 configall.h 文件，重新编译链接 VxWorks 映象），如果使用网口调试，选择 wdbrpc，在目标 IP 名或地址域中给出目标机的 IP 名或地址（建议给出 IP 名，因为这样会快得多），如果给出的是 IP 名，需要在 HOSTS 文件中给出 IP 名与 IP 地址的对应关系，如果用串口调试，选择 wdbserial，选择相应的串口和波特率（注意：此处的串口是指主机的串口不是目标机的串口）；

在 Change Property 域选取 **Core File and Symbols 项**，选中 File 项输入相应的文件（同目标机上运行的内核要一致），点击 Launch，运行目标服务器。

执行菜单命令 File | New 创建一个新的文件，并打开编辑器 Editor（该编辑器功能不是很强大，可以使用其它编辑器如 Source Insight）。

单独编译生成的源文件，生成目标文件（.o），编译连接过程的详细介绍请见后面。

选取相应的目标服务器。

执行菜单命令 Tools | Debugger 运行调试器。

执行菜单命令 Debug | Download 下载要调试的目标文件（.o）

在 Editor 窗口设置断点。

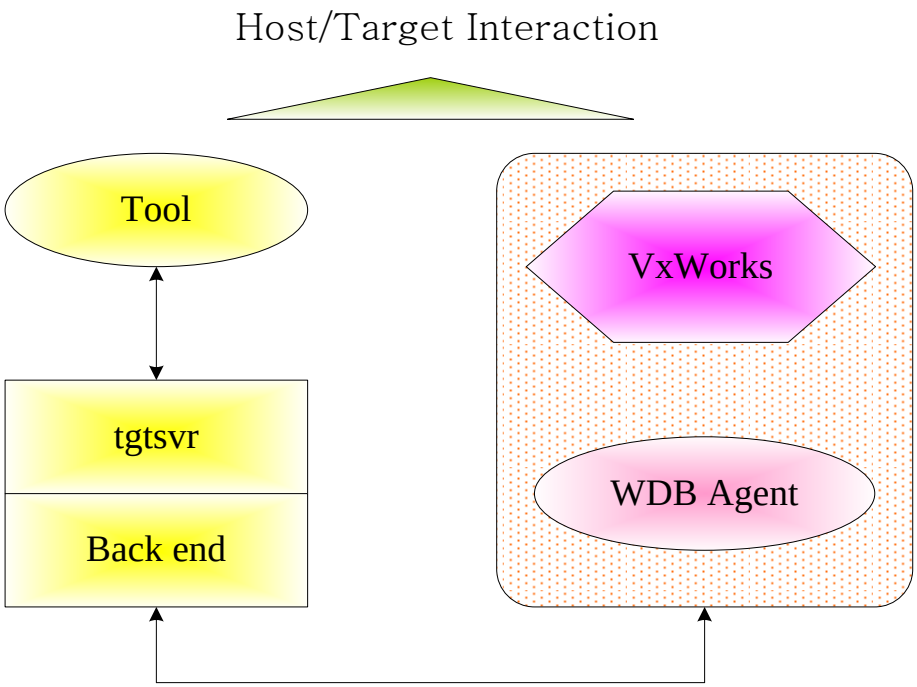
执行菜单命令 Debug | Run 弹出对话框，要求输入调试入口函数，输入要调试的函数。

进行源码级调试

执行菜单命令 Tools | Shell 运行 Shell。可以在 Shell 窗口查看/修改全局变量、内存，查看任务列表、各模块使用内存的情况、对象（如任务、队列、信号量、定时器、内存分区）等信息。

执行菜单命令 Tools | Browser 运行 Browser。在 Browser 中可以查看任务列表、各模块使用内存的情况、对象（如任务、队列、信号量、定时器、内存分区）等信息。

WDB 代理



WDB 代理（WDB Agent）运行在目标机上，代表 target server 和 Tornado tools:

1)、读或修改内存；
2)、设置或清除断点；
3)、创建、开始、停止、删除任务；
4)、调用函数；
5)、收集系统对象信息；

WDB Agent 可配置性：

1)、可在任务级、系统级模式下 debug
2)、可选择同 target server 的通讯方式

注：target server 和 WDB agent 通过 WDB (Wind Debug) protocal 进行通讯。

2 Project

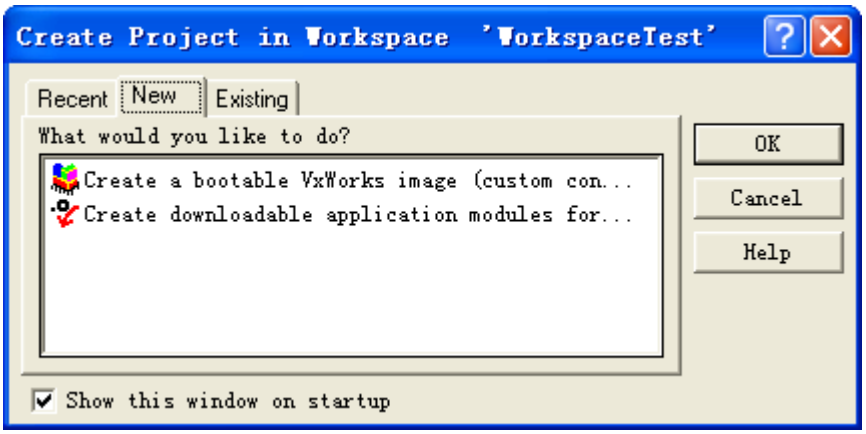
2.1 Downloadable project

Tornado 使用工程 (project) 和工程区 (workspace) 来管理用户代码、配置 vxWorks、定制编译环境。

Project 是一个源文件和二进制文件的集合，workspace 是一组相关 projects 的集合。

在 tornado 下，选择 **File->New Project…**开始生成新工程：

step1:

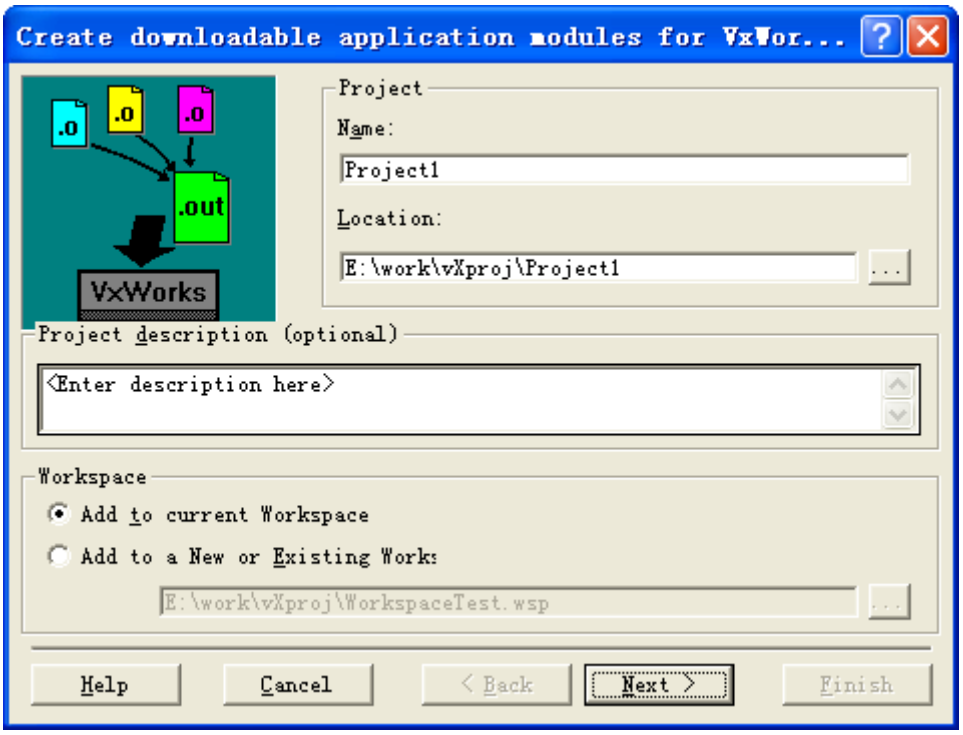


Bootable & downloadable:

- 一个完整的应用分：应用+Vxworks+Bsp
- 为适合不同的环境，Vxworks 将应用和环境分开：
  应用+环境（Vxworks+Bsp），环境诸如：ppc, x86, 模拟；
- 所以： bootable --- 完整应用
  downloadable --- 仅应用，在调试的时候选择环境
- 模拟是一特殊的环境，将宿主机也模拟成目标机；

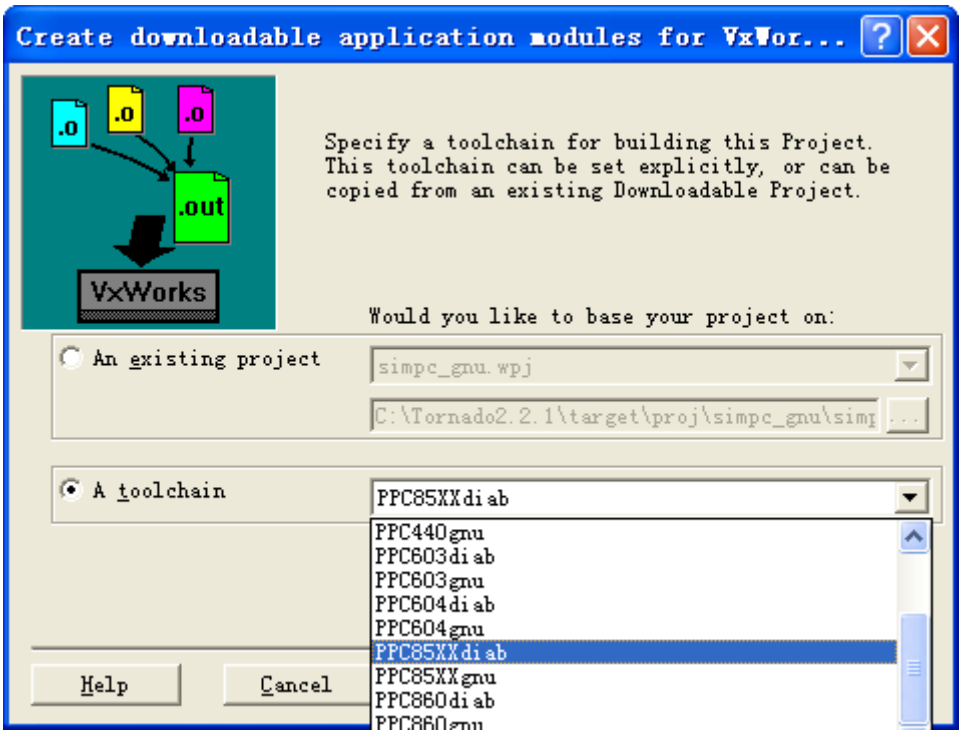
所以一般应用选择 downloadable，完整应用（如烧制到 falsh）选择 bootable;

Step 2: 输入新工程设置



Name: 工程名； Location: 工程位置 WorkSpace: 开发环境（可以几个工程合用一个）

Step 3: 选择工程环境



可以选取以前的工程环境，也可以自己选择；

一般选择后者，可以选择目标环境，

- 不在单板上运行时：选择模拟；

  在单板上运行时：选择相应的选项

  到此已经生成了基本程序框架，可以加入相应的应用；

Step 4: 创建自己的应用：
可以直接在 Tornado 环境内进行开发，**File…->New**， 为工程填加一个 c/c++文件；
如建立了文件 main.c:

```
#include <vxworks.h>
#include <stdlib.h>
#include <stdio.h>

int Add(int a, int b)
{
	return a+b;
}

void main()
{
	int a,b,num;
	printf("hello world!\n");
	a= 1;
	b= 2;
	printf("%d+%d=%d\n", a,b,Add(a,b));
}
```

Step 5: 编译联接

第一个按钮： build
第二个按钮： build all
第三个按钮： Compile
第四个按钮： 建立整个工程的调用关系，如果没有执行，会自动调用；
结果是自动将被调用的文件加入到工程；并自动分类；

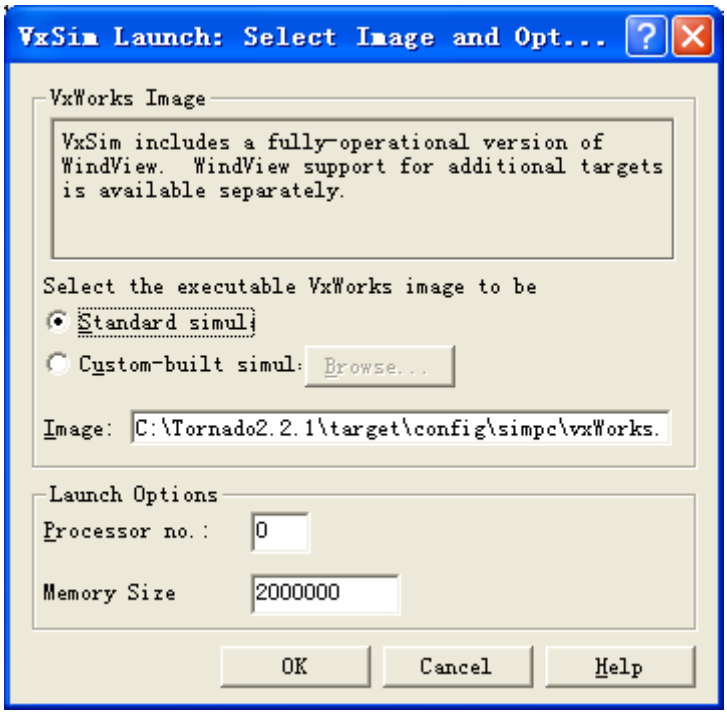
这基本同其他调试工具； 用它进行编译联接；
如果出错，则修改相应的原码

Step 6: 下载文件到目标机
基本过程如下：

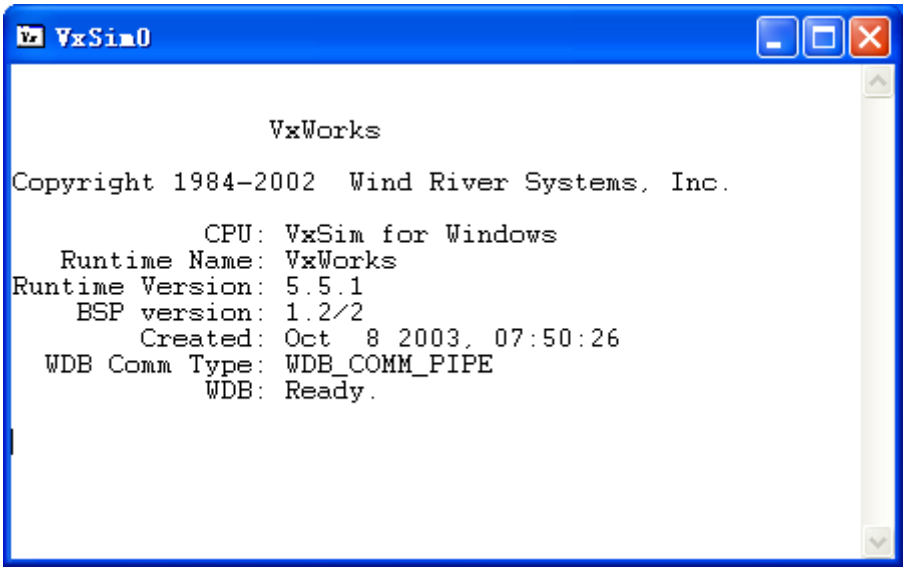
- 如果使用模拟，就启动模拟

   : 会出现下面的对话框

  有标准的和自定义的 2 种，一般选择标准的；

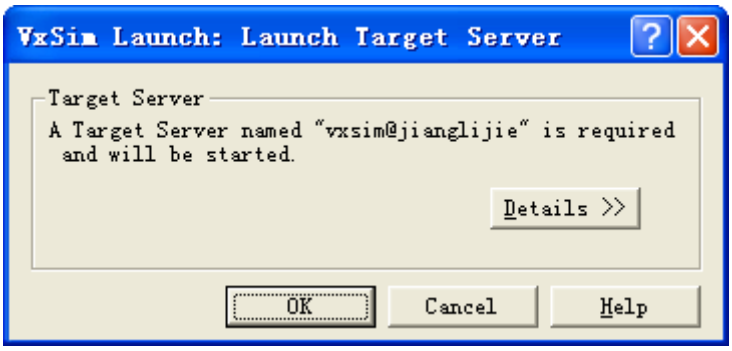


ok 后就启动了模拟：



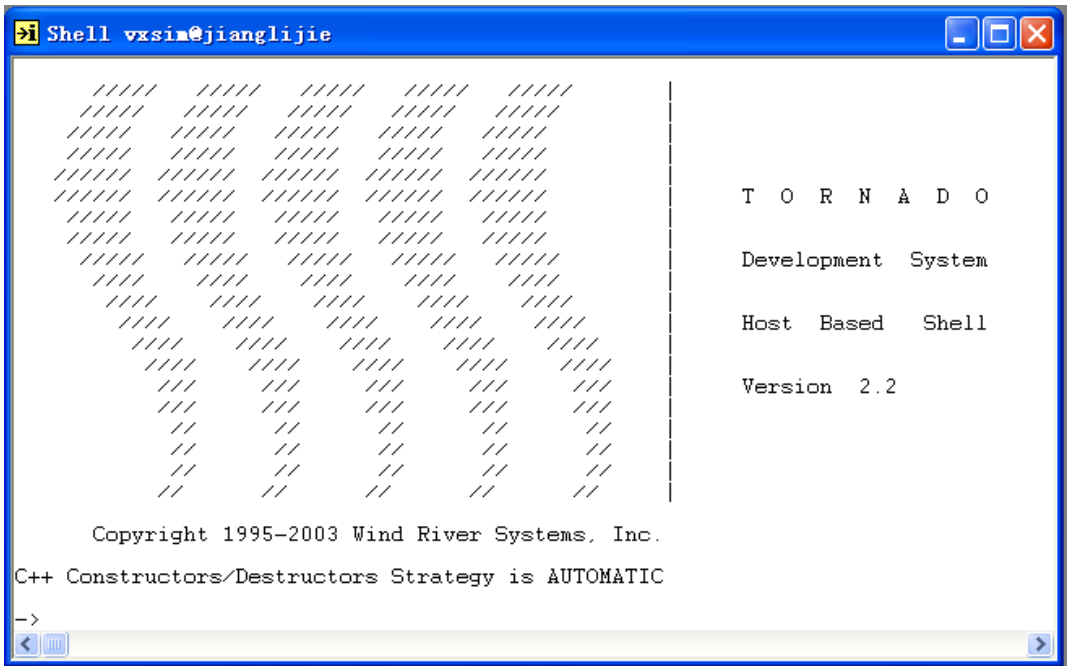
它除了担负做模拟机的任务外（加载了 Vxworks 和响应的驱动）；
还可做显示输出（printf）

- 启动 Target Server:
  在上面一步完成后，会出现下面的对话框；



确定后会调起 Target Server;

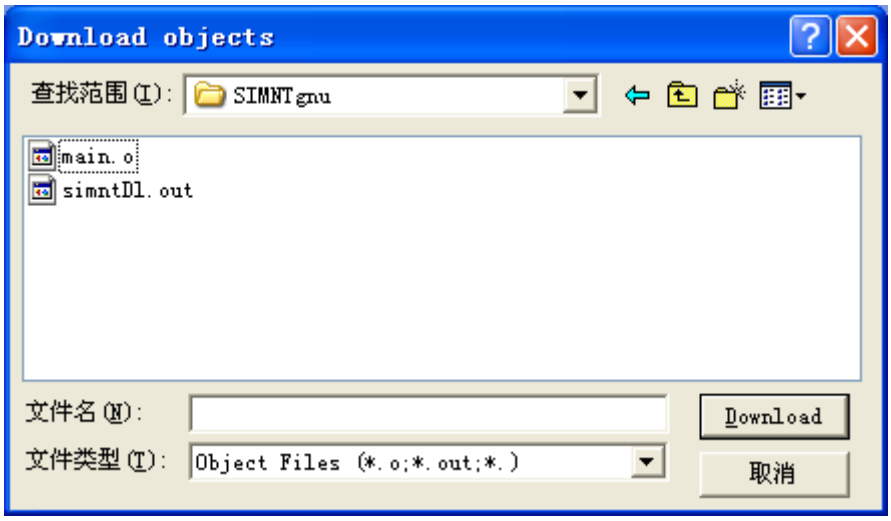
启动 SHELL  , 具体工具的使用参见另外说明；



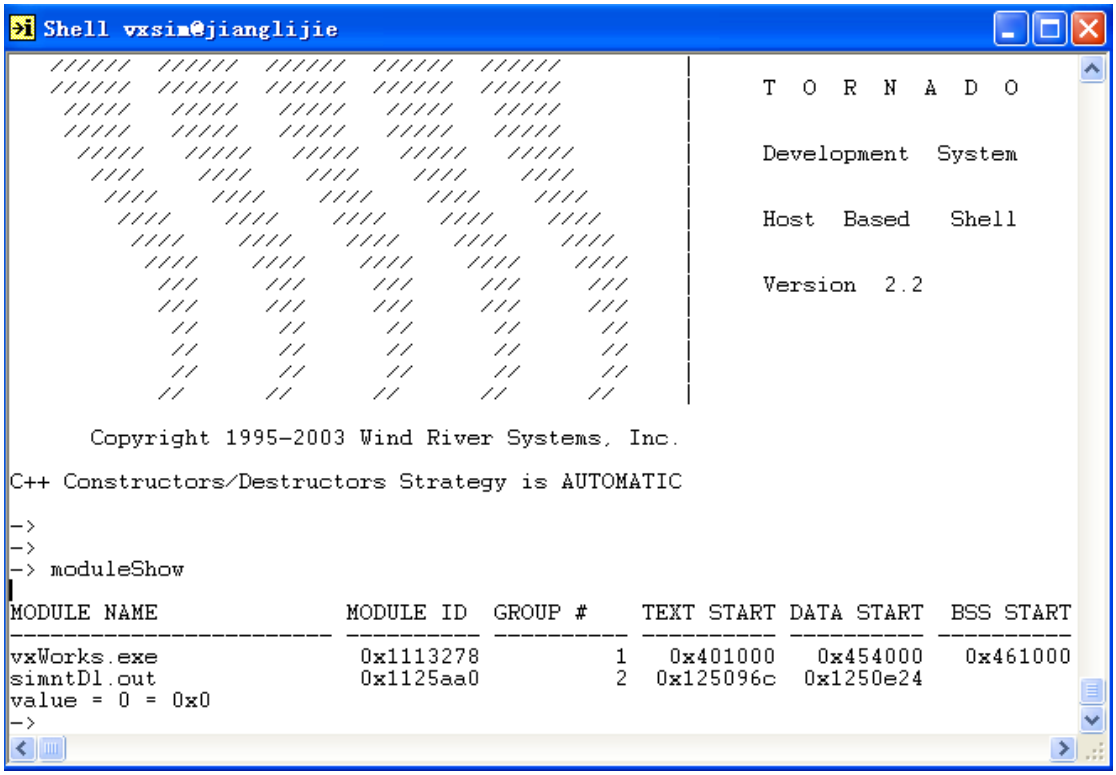
键入：moduleShow，察看启动的模块，可以看到 Vxworks 已经启动， 即环境已经建立

- 下载应用： 

  选择相应的应用下载到目标机



在 Shell 中键入 moduleShow:

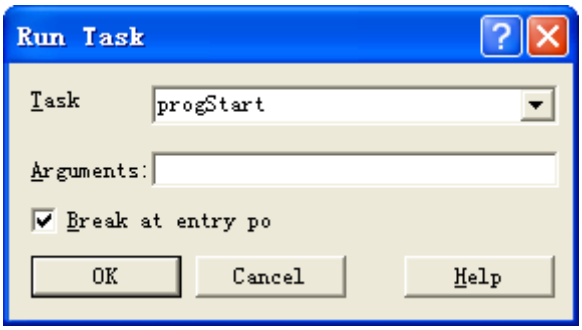


可看到应用也运行了。

至此，应用已经下载完毕；

Step 7：调试程序：

- 启动调试器： 
- 运行： 



如果 Task 内没有所要的 task,直接键入;

```
/*************************************************************
*
* progStart - start the sample program.
*
* Create various semaphores and spawn various tasks, while
* incredibly little error checking.
*
* RETURNS: OK
*/

STATUS progStart (void)
    {
    syncSemId = semBCreate (SEM_Q_FIFO, SEM_EMPTY);
    dataSemId = semBCreate (SEM_Q_FIFO, SEM_EMPTY);

    currNodeSemId = semMCreate (  SEM_Q_PRIORITY
                                | SEM_INVERSION_SAFE
                                | SEM_DELETE_SAFE);

    pCurrNode = NULL;   /* just in case */
```

可以看到任务已经启动， 并停在任务入口处。

- 设置断点
  可以直接在相应的行设置断点—F9；

- 调试工具：

基本同其他的一样。

2.2 bootable project

下列情况时需要创建 bootable project：

- 产品的正式版；
- 定制 vxWorks image;

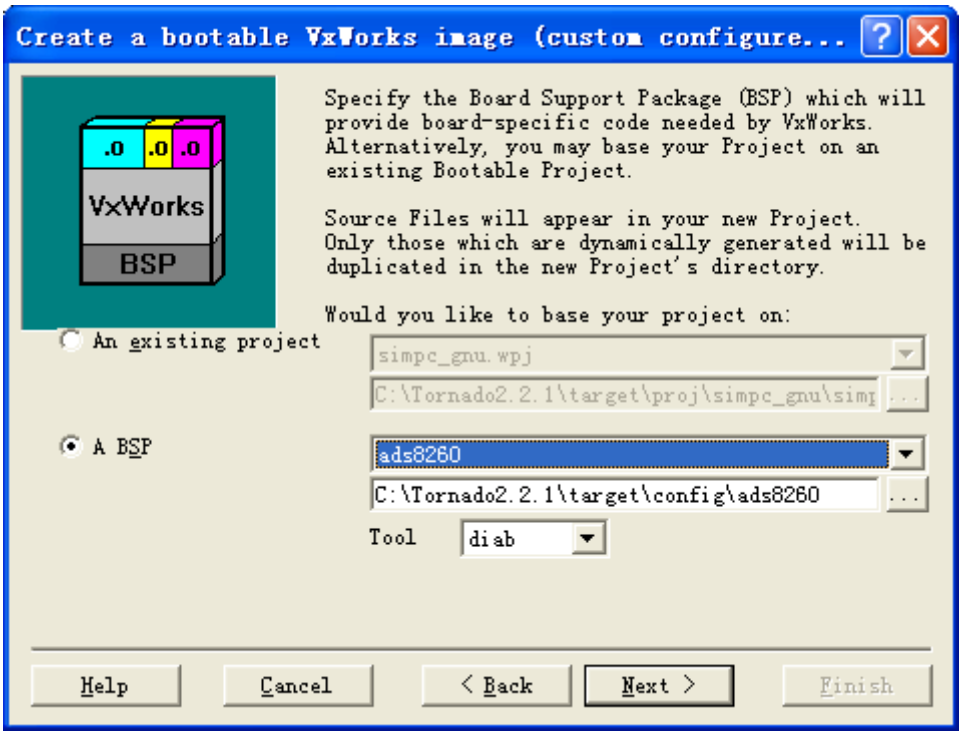
每个 bootable project 都是基于一个特定的 BSP，在 bootable project 里，应用程序代码同 vxworks 内核是静态联接在一起的。

创建 bootable project 步骤：

step1: 同上

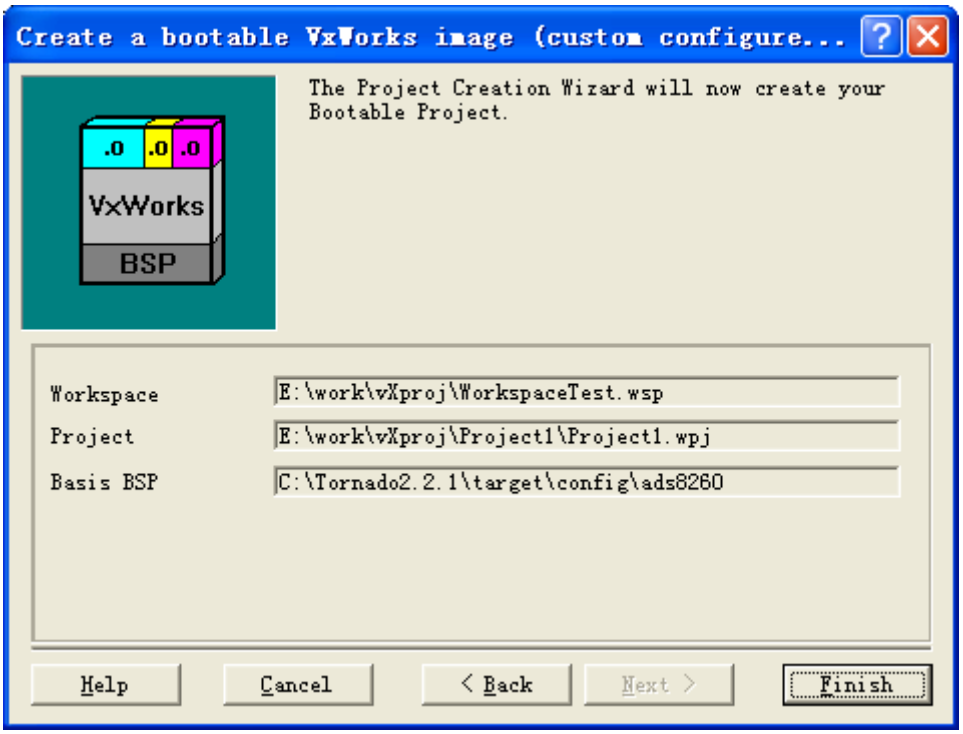
step2: 同上

step3: 选择工程环境

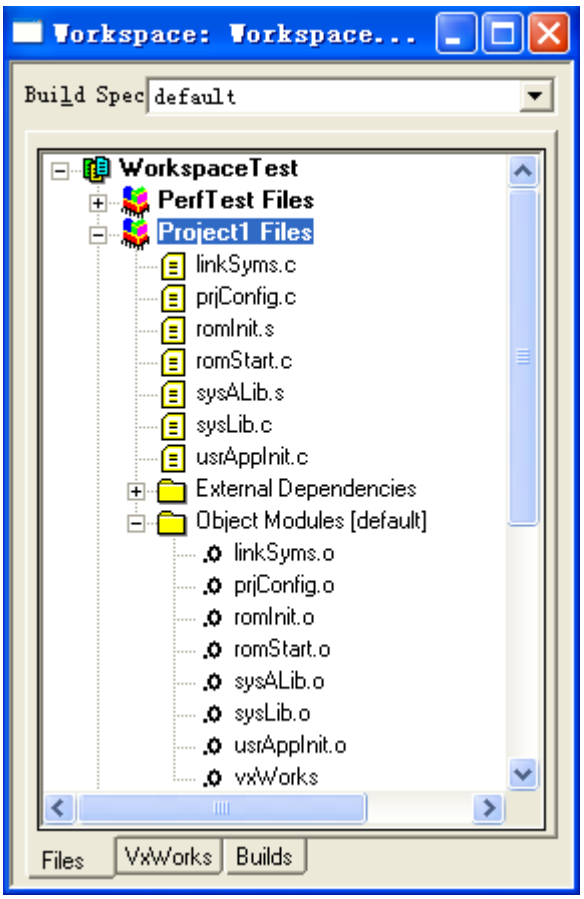


根据自己的 bsp 选择相应的环境配置，有两种情况：选 An existing project 可以继承以前的工程环境配置，无须重新配置，选 A BSP 则自动重新配置。（建议选此项）

step4:



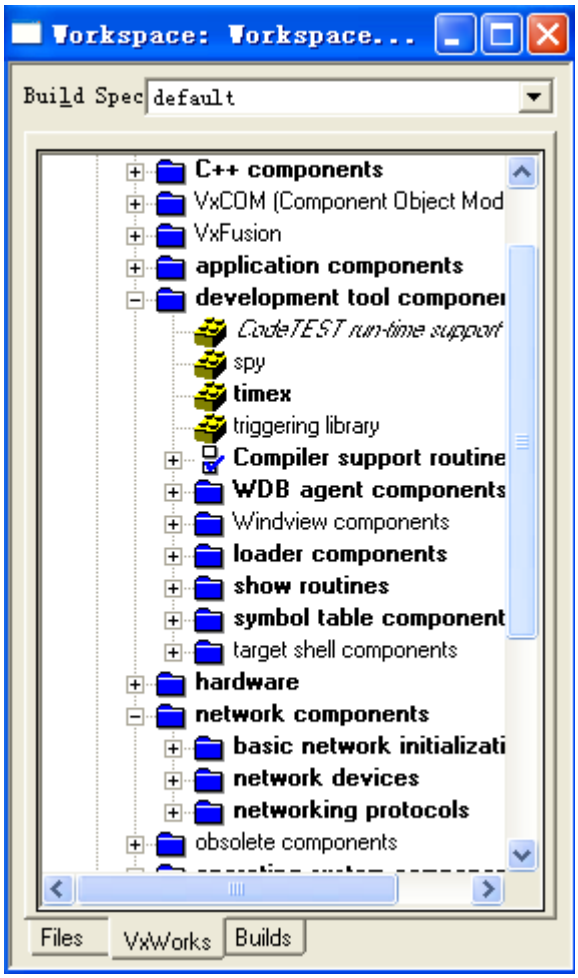
到此已经生成了基本程序框架，可以加入相应的应用；



工程有三个属性表，Files, Vxworks, Builds,下面一一介绍：

Files:

- linkSyms.c and prjConfig.c----工程动态产生的配置文件，包含组件初始化和缩放支持；
- romInit.c, sysAlib.s, sysLib.c, romStart.c----总的开始文件；
- usrAppInit.c----初始化用户程序，用户需要修改此文件来满足自己要求；
- prjComps.h and prjParams.h----工程动态产生的头文件，包含组件选择和参数值。



这个层次面板显示了 vxworks image 的所有可加载的部件和功能，黑体显示的是已包含的部件，浅平显示的是未包含进来的部件，斜体显示的是无效的部件。在选中每个部件时，通过单击鼠标右键，在弹出的快捷菜单上，可 include 或 exclude
此部件，并可修改此部件的属性。**建议大家仔细看一下有哪些可选的部件，部件如没有被选上，则相应的功能就没有，所以当大家调试程序时，如出现一些问题，要仔细的看看对应的部件有没有被 include 进来，如你要在目标机上装 shell 程序你就得 include 上“target shell components”部件。**

Build（见 2.4 节）

2.3 集成模拟器vxsim

VxSim 是 VxWorks 提供的在实验与测试环境下仿真 VxWorks 目标机的程序，而无需硬件的支持。在某些方面，VxSim 和运行在目标机硬件上的真实 VxWorks 环境相同，用户连接应用、rebuild VxWorks 映像，就好象在真实的使用标准 BSP 的 VxWorks 交叉开发环境一样。

VxSim 中，image 作为宿主机的一个过程执行。由于代码在宿主机本地 CPU 结构中，因而没有指令的竞争。 由于不存在与目标机的交互作用，故不适合开发设备驱动程序。

具体区别如下：

1、 Drivers：由于设备驱动程序需要与硬件的交互作用，许多 VxWorks 的设备驱动程序在

VxSim 下不可用。

2、 文件系统：VxSim 缺省使用 pass-through 文将系统（passFs）直接访问工作站上的文件。

3、 Networking ： Tornado 内置的 VxSim 不能使用网络功能；额外购买的增强功能组件可以支持网络功能。

Tornado 包含了一个目标机仿真的有限版本，每用户运行一个实例，无网络支持。需注意的是，一些可选的产品如 STREAMS, SNMP，和 Wind Foundation Classes 在此版本下不可使用。

使用 VxSim

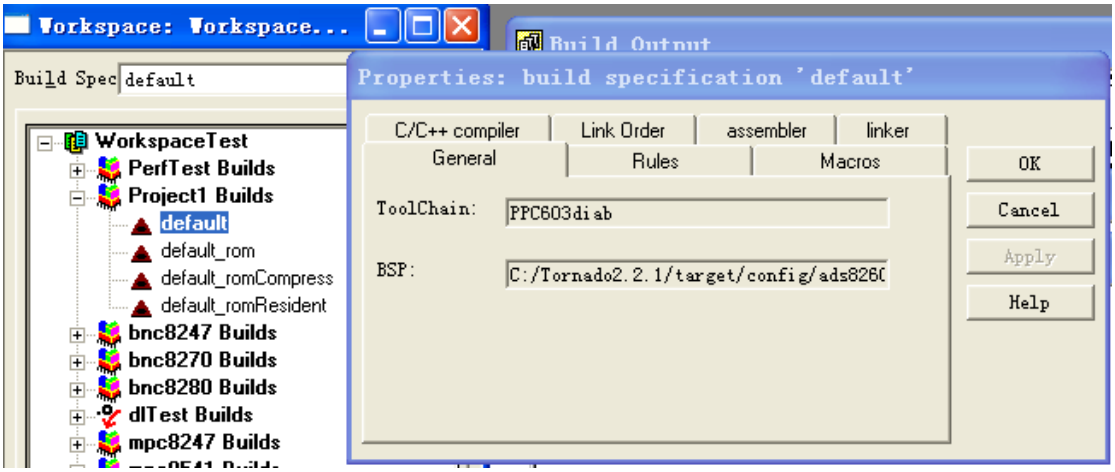
启动：当请求一个需要连接到目标机上的功能时，VxSim 自动启动。例如：当请求下载模块时，如果 VxSim 尚未启动，那末目标机服务器 VxSim 会自动启动。

当然也可以使用命令行、菜单或工具条启动 VxSim。

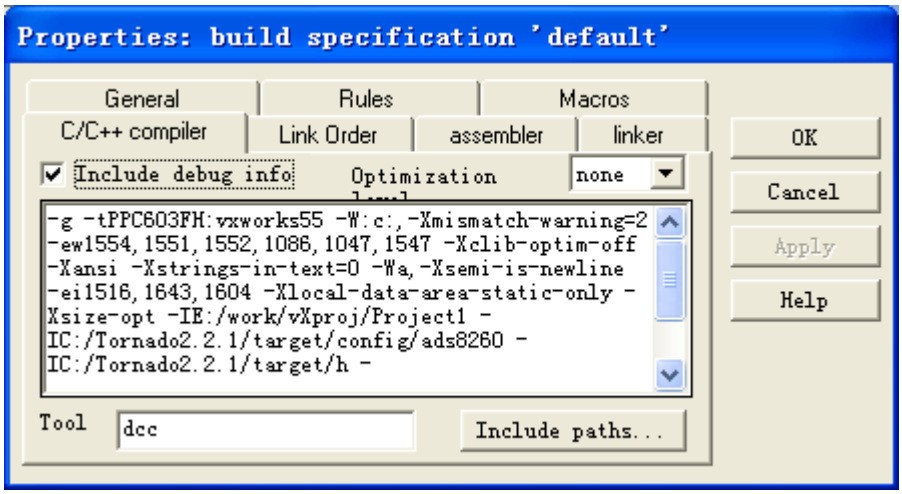
重新启动：在 Shell 中健入 CTRL+X

退出：在 windows 下，关闭窗口。

2.4 Bulid 说明



双击 default 图标，打开编译属性页，如上图，用户可根据自己情况修改参数。在 C/C++ compiler 页上定义了 debug Flags:



各意义如下：

- Include debug info 控件 sets the debug flag(-g);
- Optimization level 设置编译器优化级（O,-O1,-O2）;
- Flags vary with CPU architecture. Some flags shown here are:

  -O0 No optimization
  -g Source debugging support
  -ansi ANSI function declarations and prototypes
  -m68040 Generate output for a MC68040
  -nostdinc Don’t use usual UNIX include directories
  -fvolatile Variables referenced throught pointers assumed volatile
  -fno-builtin Don’t use compiler built-in functions
  -I *dirName* Location of header files
  -DCPU=… Preprocessor definition of CPU type

3 WindSh and Browser

3.1 WindSh(WindShell)

3.1.1 简介

提供从宿主机到目标机之间的一个命令 shell。WindSh 是一种非常受欢迎的开发工具，它具有很强的交互性和可操作性，允许用户调用内存中的应用程序模块或是 VxWorks 模块中的任何例程。它不但具有一般命令语言的功能，而且也具有 C 语言的设计特点，能够解释几乎任何 C 语言表达式， 执行大多数 C 语言算子，解析符号表数据。对初用者来说，WindSh 学习起来比较简单，使用比较方便，对熟练用户而言，则有较为高级的手段可以应用。

WindSh 是一个驻留在主机内的 C 语言解释器，通过它可运行下载到目标机上的所有函数，包括 VxWorks 和应用函数。Tornado 外壳还能解释常规的工具命令语言 TCL。

WindSh 不仅可以解释几乎所有的 C 语言表达式，而且可以实现所有的调试功能。它主要有以下调试功能：下载软件模块；删除软件模块；产生任务；删除任务；设置断点；删除断点；运行、单步、继续执行程序；查看内存、寄存器、变量；修改内存、寄存器、变量；查看任务列表、内存使用情况、CPU 利用率；查看特定的对象（任务、信号量、消息队列、内存分区、类）；复位目标机。

3.1.2 启动和终止

启动：命令行方式为：windSh phobos(目标服务器名称)；菜单 Tool->Shell；工具条。

终止：快捷键[Ctrl+D]；命令 exit() 或 quit()。

3.1.3 Shell特性

I/O 重定向：程序员常常调用例程从标准输入接收数据或在标准输出显示数据，缺省情况下，标准输入输出流被定向到 Tornado Shell。使用 Tcl 过程 shConfig 修改 WindSh 环境变量 SH_GET_TASK_IO，实现 I/O 复位向。

路径与符号补齐：键入任何符号或存在的路径名的开始部分，然后键入[Ctrl+D]，Shell 会自动补齐命令或路径名。 并且可以继续键入若干字符后 加上 [Ctrl+D] 知道命令或路径完整为止。

打印命令摘要（Synopsis）：键入完整的命令后，继续键入空格 +[Ctrl+D] 可以显示命令摘要，同时输入的命令继续作为输入。

显示 HTML 帮助：键入完整的命令后，继续键入空格 +[Ctrl+W]。将会启动浏览器显示该命令的详细帮助信息。

删除一行： [Ctrl+U]

输入类似 Vi 编辑命令：Esc

数据转换与计算： 键入整数或字符后回车可以显示该整数的十进制及十六进制值。也可以键入字符常量或符号地址。

所有的 C 操作符都可以在 Shell 里用于数值计算。也可以在 C 表达式中使用变量。

WindSh 环境变量： 使用 ？ shConfig 命令修改环境变量

SH_GET_TASK_IO： 为调用函数设置 I/O 重定向。

ON：重定向到 WindSh；OFF：I/O 显示到目标机控制台。

LD_CALL_XTORS：

LD_SEND_MODULES: 设置 load 模式。

LD_PATH: 为模块设置搜索路径，用“;”隔开。 例如：ld 命令提交后，Shell 搜寻模块的路径顺序为：首先在当前目录，然后到 LD_PATH 设置的路径。

LD_COMMON_MATCH_ALL:

DSM_HEX_MOD:

注意：因为shConfig是Tcl过程，所以输入命令时使用 ?shConfig。

3.1.4 WindSh内置命令

3.1.4.1 任务管理

sp( )	用缺省参数创建一个任务（priority=100 返回值为任务 ID，或错误）(taskSpawn)
sps( )	创建一个任务，并挂起它
tr( )	恢复一个挂起的任务（与 taskResume 相同）
ts( )	挂起一个任务 （与 taskSuspend 相同）
td( )	删除一个任务 （与 taskDelete 相同）
period( )	创建一个周期调用函数的任务
repeat( )	创建一个重复调用函数的任务
taskIdDefault( )	设置并报告当前缺省的任务 ID。

3.1.4.2 任务状态信息

i( )	显示系统信息，包括当前任务明、状态等（重复查询目标机，有时可能显示不一致）
iStrict( )	类似于 i( )，但只查询目标机一次
ti( )	显示任务的 TCB 信息
w( )	显示所有挂起任务的详细信息。.
tw( )	显示某挂起任务的详细信息。
checkState( )	显示一个任务的使用堆栈的总结，没有定义任务时显示所有
tt( )	显示堆栈记录
taskIdFigure( )	报告任务的 ID,以及名称

3.1.4.3 系统信息

devs( )	列出目标机系统上的所有设备
lkup( )	List symbols from symbol table
lkAddr( )	List symbols whose values are near a specified value
d( )	显示目标机内存
l( )	Disassemble and display a specific number of instructions
printError( )	给出最近的错误值
version()	显示 VxWorks 版本
cd( )	改变宿主机工作目录（不影响目标机）
ls( )	列出宿主机工作目录下的文件

ll( )	列出宿主机工作目录下的文件详细信息
pwd( )	显示宿主机当前工作目录
help( )	显示 Shell 命令的帮助
h( )	显示最近输入的 20 个命令
shellHistory( )	设置或显示 Shell 命令
shellPromptSet( )	改变 C 解释器 Shell 提示
printLogo( )	显示 Tornado Shell 登陆

3.1.4.4 系统修改和调试

ld( )	加载一个对象模块到目标机，并动态连接到 run-time
unld( )	从目标机内存中删除动态连接的对象模块
m( )	Modify memory in Width(byte,short,long)starting at adr.
MRegs( )	为特定的任务改变寄存器的值
b( )	设置或修改断点
bh( )	设置硬件断点
s( )	单步跟踪到下一条指令
so( )	单步跟踪,但跳过子程序
c( )	从断点处继续
cret( )	继续执行，直到当前的子程序返回
bdall( )	删除所有断点
bd( )	删除一个断点
reboot( )	重新启动 targe server
bootChange( )	改变 boot 参数保存值
sysSuspend( )	如果目标机代理支持，进入系统模式
sysResume( )	如果目标机代理支持，从系统模式返回到任务模式
agentModeShow( )	显示代理模式（系统模式 或 任务模式）
sysStatusShow( )	显示系统上下文（suspend 或 running 系统模式下使用）
quit( ) or exit( )	

3.1.4.5 对象命令(WindSh Commands for Object Display)

show( )	在 shell 窗口打印特定对象的信息
browse( )	在 Tornado browser 显示特定的对象
classShow( )	Show information about a class of VxWorks kernel objects.List available classes
taskShow( )	显示任务 TCB 信息
taskCreateHookShow( )	显示任务创建例程列表
taskDeleteHookShow( )	显示任务删除例程列表
taskRegsShow( )	显示任务寄存器的内容
taskSwitchHookShow( )	显示任务切换例程列表
taskWaitShow( )	显示阻塞任务信息
semShow( )	显示信号量的信息
semPxShow( )	显示 POSIX 信号量的信息
wdShow( )	显示看门狗的信息
msgQShow( )	显示消息队列的信息

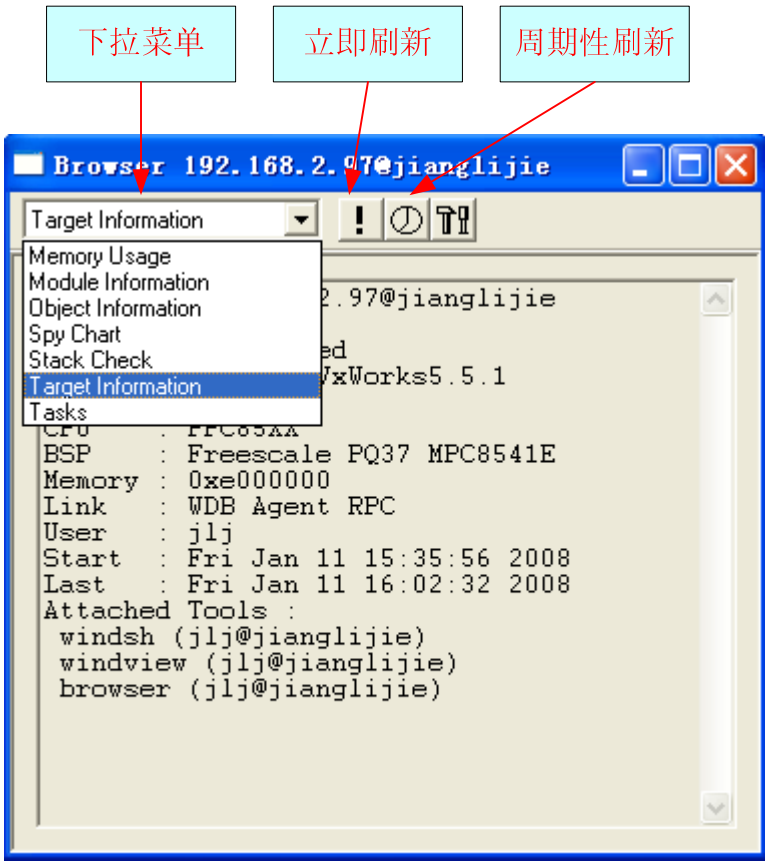
mqPxShow( )	显示 POSIX 消息队列的信息
iosDrvShow( )	显示系统驱动程序的信息
iosDevShow( )	显示系统设备的信息
iosFdShow( )	显示系统命名 descriptor 的信息
memPartShow( )	显示分区块及统计信息
memShow( )	显示系统分区上空闲和已分配空间的总数等等
smMemShow( )	Display the amount of free space and statistics on memory-block allocation for the shared-memory system partition
smMemPartShow( )	Display the amount of free space and statistics on memory-block allocation for a specified shared-memory partition
moduleShow( )	Show the current status for all the loaded modules
moduleIdFigure( )	Report a loaded module's module ID, given its name
intVecShow( )	Display the interrupt vector table. This routine displays(nt 下不可用)

3.2 Browser

可对系统对象（任务、消息队列、信号量等）和存储器使用情况进行观察的浏览器。可以方便地监视用户的 目标系统。Browser 汇总了应用进程，内存消耗和一个目标内存的映像。通过 Browser，用户可以观察信号量、消息队列、内存分配、看门狗计时器、堆栈使用情况、目标 CPU 使用率、对象模块结构和符号表以及每个任务的详细信息。

包括以下几种监测类型：

Memory Usage
Module Information
Object Information
Spy Chart
Stack Check
Target Information
Tasks



3.3 Spy

VxWorks 及其调试环境提供两种工具，可用于对多任务系统的分析: Spy 和 WindView。WindView 具有图形界面，可用于上下文切换、任务状态迁移及其它系统事件的分析和纪录，但要求集成开发环境 Tornado 的支持。Spy 相对而言功能单一且使用简单，主要用于统计各任务和中断的 CPU 占用率。

1. 辅助时钟

Spy 需要一个独立于系统时钟的辅助时钟来为其任务统计提供计时。该辅助时钟的中断频率不能低于 100ticks/sec。对于 PowerPC860 系统而言，VxWorks 本身提供了辅助时钟驱动程序的源代码：[Tornado]\target\src\drv\timer\ppc860Timer.c，该源码使用 CPM Timer2 作为时钟源。如果在系统设计中 CPM Timer2 并没有其它用途，那么对于辅助时钟驱动程序可不加修改而直接引用。

2. 配置 VxWorks

使用 Tornado 的图形化工具配置 VxWorks。将以下两项包含进去：

development tool components --> spy

hardware --> peripherals --> clocks --> AUX clock

如果通过手工修改配置文件，则需增加以下宏定义，并生成新的 project：

#define INCLUDE_SPY

#define INCLUDE_AUX_CLK

在调试阶段，为了使调试工具(如 Shell)能找到 Spy，还需要修改 symbol 表的配置，加入以下项目：

development tool components --> symbol table components --> symbol table initialization components -->select symbol table initialization --> built-in symbol table

3. 开始使用 Spy

可试验在 Shell 中直接调用 Spy 工具。使用方法有两种：

1) spy(freq, ticksPerSec)

该函数会启动一个 task：spyTask()来执行统计和报告，每 freq 秒报告一次。ticksPerSec 是辅助时钟每秒中断次数。

2) spyClkStart(intsPerSec) 和 spyReport()

先调用 spyClkStart 来启动辅助时钟，intsPerSec 是辅助时钟每秒中断次数。然后可不定时地调用 spyReport 观看统计结果，包括自上次调用 spyClkStart 以来的统计数据和自上次调用 spyReport 以来的统计数据。

spy 将统计结果输出到 target system console 或 target shell。也可以在 host 端的调试工具 Shell 和 Virtual Console Window 里观察其输出。统计结果中包含了各 Task、中断服务例程、Kernel 和 Idle 所分别占用的时间百分比和 ticks 数。以上
详见 Tornado Online Manuals。

3.4 WindView

1. 配置 VxWorks

1) include "development tool components -> Windview components"。
2) include "development tool components -> WDB agent components -> WDB agent services -> WDB target server file system"。

若要使 View Graph 的 Timeline 刻度为时间坐标(seconds)，而不是缺省的 sequence numbers 坐标，还需如下配置(需 BSP 和硬件支持)：

3) include "hardware -> peripherals -> clocks -> high resolution timestamping"。
4) change "development tool components -> Windview components –> select timestamping" from "sequential timestamping" to"system-defined timestamping"

2. 配置 Target Server

1) Tools -> Target Server -> Configure...
2) 在对话框 Configure Target Servers 中选择 TSFS:
   Target Server Properties -> Target Server File System
3) 配置如下：
   check "Enable File System";
   enter the root path(用于存放 Log File);
   select "Read / Write" option(若不使用 WindView，TSFS 的访问权限应设为"Read Only");

3. 配置 WindView

1) Tools -> WindView -> Configuration...
2) 在对话框 WindView Collection Configuration 中配置 Logging Level:

a. **Context Switch Event-Logging Level** (CSE)
   记录当前程序的上下文，以及上下文切换的位置，如任务切换，中断/异常处理程序的进入和退出等。

b. **Task State Transition Event-Logging Level** (TST)

纪录 CSE Level 的所有信息(所有的上下文切换都伴随有任务状态的迁移，然而任务状态的迁移却并非总会导致上下文切换)。

记录任务状态的迁移，以及引起任务状态迁移的事件，如 semTake(), semGive, taskDelay()等。

c. **Additional Instrumentation Event-Logging Level** (AIL)

纪录 CSE Level 和 TST Level 的所有信息。

纪录对所选 instrumented library(taskLib, semLib, msgQLib, wdLib, memLib, sigLib)中的对象的所有操作，不论其是否引起上下文切换或任务状态迁移。

3) 在对话框 WindView Collection Configuration 中点击按钮 Properties，弹出对话框 WindView Control Properties:

a. 配置 Ring Buffer: 缺省值。

b. 配置 Upload Path:

(1) Direct to Graph: 将数据直接显示在 host 的 View Graph 中;

(2) File via TSFS: 将数据上传并存放在/tgtsvr/eventLog.wvr;

(3) Socket via TSFS: 将数据上传到/tgtsvr/TCP:hostname:6164;

(4) Socket via TCP/IP:将数据上传到 hostname/6164;

(5) NFS to <I>file:</I> 将数据上传并存放在/eventLog.wvr;

注：

使用方式(4)时，需在配置 VxWorks 时包含"development tool components -> Windview components -> upload path(s) -> TCP/IP socket upload path initialization"，缺省情况下是没有的。

使用方式(5)时，需先在 target 上配置好 NFS。

方式(1)、(3)和(4)的上传目的地是 View Graph，直接显示数据，不涉及文件操作。使用方式(3)和(4)时，需先用 File -> New 创建一个 WindView Log，以便显示上传的数据。使用方式(1)时系统会自动 Launch 一个 View Graph。

方式(2)和(5)的上传目的地是文件，数据不直接显示，需在上传完毕之后用 File -> Open 打开所保存的 WindView Log 文件(*.wvr)进行分析。

方式(1)、(2)和(3)的上传路径是 Target Server，方式(4)和(5)的上传路径是 Network Facilities。

c. 配置 Upload Mode:

(1) Deferred: 先采集，再上传；

特点：数据量受 target 的 memory 限制；对 target 性能影响较小；

(2) Continuous: 边采集，边上传；

特点：数据量不受 target 的 memory 限制；对 target 性能影响较大；

(3) Post-Mortem:用于死机或复位问题的分析；

特点：数据存放在位于 User Reserved Memory 区域的 Ring Buffer 中，留待系统重启之后，将之上传到 host 并进行分析。

注： User Reserved Memory 位于 sysMemTop 和 sysPhysMemTop 之间，在系统 boot 期间不被清空，并且不可 cache。

4. 开始使用 WindView

1) Tools -> WindView -> Launch...
2) 在对话框 Launch WindView 中选择 Target: TargetServerName@Hostname
3) 对话框 WindView Control TargetServerName@Hostname:
   按钮 G 开始采集数据；
   按钮 STOP: 停止采集数据；
   按钮 UPLOAD:上传所采集的数据。
4) 另两种使用 WindView 的方法：

a. 通过 Triggering 控制 WindView 的数据采集；
b. 在 WindSh 或 Source Code 中调用 WindView API: wvOn() and wvOff()。

4 CrossWind

4.1 Debugging简介

VxWorks 具有两种调试模式 开发工具对目标机应用程序的调试方法有两种模式。一种是系统模式。对整个应用系统进行调试，可在系统中设置断点等。调试中应用系统必须停下来；另一种是任务模式（即动态调试）。调试是针对系统中某一任务模块进行的，整个系统仍可保留在工作状态。同样在对整个系统调试时，也可一个模块一个模块进行，调好一个运行一个，这样对加速调试速度，方便系统调试提供了很大方便。

4.2 任务模式调试

在任务调试模式下，在一个集成环境中，在一个任务中调试，在另一个任务中设置断点，设置的断点不起作用。这是因为一个调试器只能处理一个 TCB（任务控制块），每个任务都有一个 TCB，因此一个调试器只能调试一个任务，要调试几个任务就要启动几个调试器。一个集成环境只能启动一个调试器，所以要调试几个任务就要启动几个集成环境。另外，需要在被调试的任务的待调试的第一条语句前加入 taskSuspend(0)语句，挂起该任务，否则任务就可能会在调试前被执行。

在任务调试模式下，在一个任务中调试，当任务运行到此断点时，只有此任务停止，而不是整个系统。

特点：只能调试任务，不能调试 ISR；

缺省情况下，断点只影响 **attached** Task；

当 **attached** 任务停止时，系统中的其它任务及 ISRs 继续运行；

与 WDB 代理的通讯方式是中断驱动方式。

4.3 系统模式调试

系统模式有时也称为外部模式（External Mode），在此模式下，target server 运行在 VxWorks 系统之外。

系统调试模式下，允许开发者挂起整个 VxWorks 操作系统。系统调试模式下一个值得注意的应用是调试 ISRS，因为 ISR 运行在任务上下文之外，并且对缺省任务模式的调试工具不可见。

在系统调试模式下，可以同时调试多个任务、中断服务程序（ISR），调试影响整个系统。

Tornado1.0 集成环境下，在系统模式下进行程序调试，主机与目标机之间必须使用串口通信。Tornado2.2 集成环境提供了通过网口进行系统模式调试的功能(END 功能)。系统缺省使用网口通信，如果需要使用串口通信，需要修改文件 C: \ Tornado \ target\config
系统调试模式下，run 命令不可用，可以使用 WindSh 调试。

特点： 可以调试任务、ISRs 以及核前（pre-kernel）的 VxWorks 执行；

断点使整个系统停止；

当系统停止时，外部 WDB 代理运行在中断锁定方式，在此期间，与 WDB 代理的通讯方式为 **Polled** 模式；

通过以太网调试时，为了支持 Polled 模式通讯 需使用 END 网卡。（注：Tornado 串行驱动也支持 Polled 模式及系统级调试。

注：系统模式和任务模式的互换：

点击 **debug->attach,** 从任务模式到系统模式;
点击 **debug**->**detach,** 从系统模式到任务模式
在 debug-command-line 下，模式转换的命令为（gdb）attach system (gdb)detach