FAQ: What is a Board Support Package?

By Ron Fredericks, Wind River, August 20, 2001, updated November 20, 2002

Introduction

There are three separate views to what defines a Board Support Package (BSP). I believe each of these views are correct as independent descriptions of a BSP. Together each of these views provide insight into the embedded target from three different perspectives or engineering disciplines. Read this FAQ to learn more about the essence of embedded systems and how Wind River can offer its customers a common set of tools and industry's leading operating systems on the world's largest set of diverse architectures because of the BSP. See Wind River's BSP web site (to be launched soon) to find hardware support for your next design from Wind River and its WindLink partners.

(view #1) A BSP is the kernel's interface to device drivers

A Board Support Package (BSP) provides a standardized interface between hardware and the operating system. A BSP does not directly access hardware. Although a BSP does provide an interface to device drivers which in turn allows the kernel to communicate with the hardware's assets such as device controllers, the microprocessor (CPU), memory, internal and external busses.

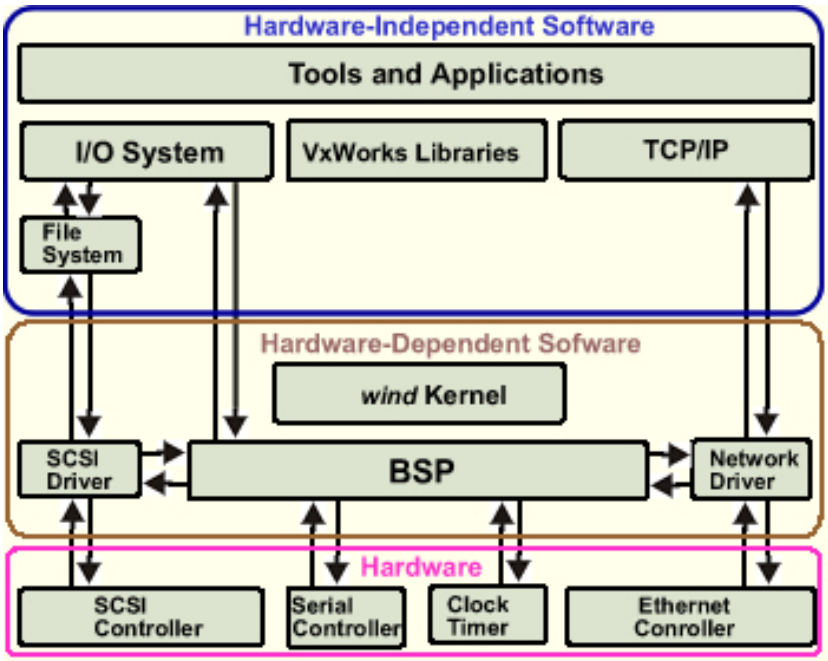

Figure 1 - BSP is an interface between kernel and hardware

Figure 1 demonstrates how VxWorks and the developer's application software remain hardware independent while the BSP provides an interface to the embedded computer system's architecture and hardware assets.

Wind River BSPs conform to a standard, introduced with BSP Version 1.1. The standard is fully described in the

Tornado BSP Developer's Kit for VxWorks. The BSP provides a common application programming interface (API) and a stable environment for the real-time operating system. The standard interface defines both BSP conventions and validation procedures.

Wind River provides a standardized BSP validation procedure for engineers who are developing their own BSP's. Validation is very efficient with a BSP designed for VxWorks because the BSP testing can be performed without having to test the operating system at the same time. Unlike Wind River's real-time operating systems, most operating systems can not separate the testing of the BSP from the testing of the operating system itself making code coverage during testing a much larger and more inaccurate process. The BSP validation test suite provided by Wind River is repeatable and as such can be performed by the company developing a BSP as well as by others because of this standard. Wind River's partners can request that the WindLink partner program grant "Tornado Certified" status to a new BSP after passing the validation procedure as part of an overall certification process.

The validation procedures include tests for:

- Package validation
- Installation tests
- Functional tests
- Code review process including certification from the WindLink partner program

The Tornado BSP standard includes conventions to be used by developers that fall into four main categories:

- Coding Conventions
- Documentation guidelines
- BSP Packaging
- Driver guidelines

One strength of VxWorks is that it provides a high degree of architectural and hardware independence for application code because of this API. Application code and VxWorks can both migrate from one architecture to an other because the BSP and related device drivers offer a consistent interface and modular design. To further illustrate the BSP's ability to interface hardware to software: Those construction workers that lay carpet for our floors use another example of this type of API in the home and office. The carpet layer uses two-sided sticky tape to quickly attach just about any style carpet to any architectural floor plan beneath it. The BSP acts like this two-sided sticky tape attaching the embedded hardware to the operating system that uses it.

(view #2) A BSP is a set of libraries offering a hardware abstraction layer to kernel

The BSP enforces a modular design by isolating hardware-specific functionally into a set of libraries that provide an identical software interface to the hardware functions available on an embedded system. See detailed description of some of the common libraries and related files later in this FAQ.

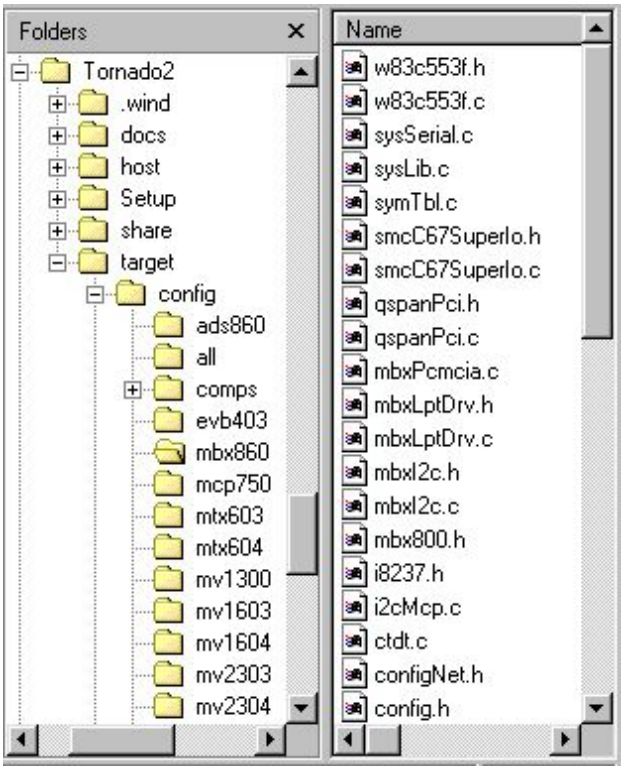

Figure 2 - Location of BSP files

Figure 2 shows an example of the typical files supporting a PowerPC target's BSP and their location in a Tornado installation on a host computer such as Windows NT.

The BSP includes facilities for hardware initialization when power is first applied or when a hardware reset has been initiated. The BSP also includes support for interrupt handling and generation, hardware clock and timer management, and mapping of available local and bus memory. These basic services provide the real-time kernel with multi-tasking services and memory space for the designer's application code.

Other device drivers and related support services can be included in this set of libraries extending the hardware abstraction to support the embedded system's custom hardware services. These services would include networking, security, storage, graphics, and input/output to the outside world for example. The devices referenced by the BSP can be located on a single board computer, a system on a chip (SoC), or on peripheral devices located across a wireless boundary or on a backplane using a hardware bus.

(view #3) A BSP supports a host/target cross development environment

The BSP provides support for developers using Tornado tools on a host computer for engineering development such as an editor, compiler, linker, and debugger through a client/server communications protocol between the developer's host computer and the embedded target's CPU, hardware devices and on-board memory. For VxWorks developers, the BSP offers an interface to the WDB agent that provides an open standard for cross-hosted programming on a specific target's architecture. The host computer supports these services using the Tornado tool suite through a target server acting as the communications agent. The target BSP can support a networked development environment using one or more of the following devices: Ethernet, JTAG, serial UART, ROM socket, or custom driver, to communicate with the host's target server. When an application is complete and the embedded device is ready for production, the WDB agent can be removed from the target's BSP.

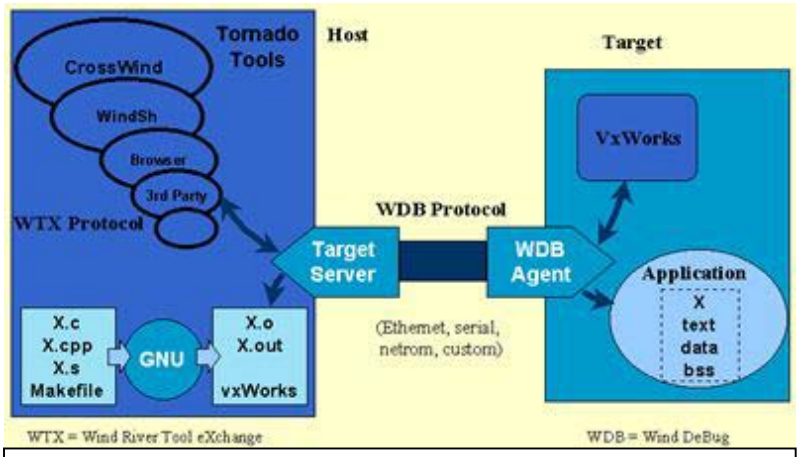

Figure 3 - BSP supports host/target cross development

Figure 3 demonstrates a typical cross development embedded programming model where the WDB client supported by the target BSP communicates with the host's target server supporting the engineering cycle: design, edit, compile, link, download, debug, test, and deploy.

There are advantages to having a BSP that is separated from the kernel….

A BSP allows target applications to be ported to many architectures

The three views of a BSP described above are all consistent with today's engineering needs for a common interface to development tools residing on a robust host computer in support of an embedded application requiring operating system support and services that is modular and portable across many different architectures.

Wind River and its partners offer a wide variety of BSP's for custom off the shelf (COTS) single board computers, and time saving reference designs on more than 12 different architectures. See our list of BSP's on our external web site sorted by architecture families here:

http://www.windriver.com/products/bsp_web/bsp_architecture.html

Description of BSP technical features

There is a file named target.nr defined in the BSP documentation standards containing a list of technical features supported and unsupported for a particular COTS or reference board. The list of features describes what a BSP is expected to do during hardware initialization when the CPU executes its first instructions as well as what hardware assets should be presented to the operating system. This information is summarized here.

Boot Monitor

- Describe the methods and devices available for booting RTOS on this board. (sm, SCSI, Ethernet, vendor-bug ROMs, open firmware, visionWARE, etc).
- Describe the network support available for booting; bootp, proxy-arp, etc…

Default Ram size

Additional supported memory options

Cache support

Flash and ROM memory

- How are ROMs made and installed? (How do they split?) Can FLASH ROMs be burned on board? If so, how?

Timer support in additional to system clock

- Describe routines that are associated with speed of CPU and Timers

Devices supported:

- The chip drivers included are:

  templateSio.c - template serial driver

  templateTimer.c - template timer driver

  templateVme.c - template VMEbus driver

  templateNvram.c - template non-volatile RAM driver

- Describe the device drivers for this board. Are there special driver ptions that must be used with this board such as CompactPCI hot swap support? Read

the specific driver documentation and describe any special options used with this board.

Memory Map

- Provide a detailed memory map of the local bus, and all additional busses from a bus master point of view. Identify only slave devices actually on each bus, not their apparent address on some other bus. (i.e. if a serial device is a PCI device, do not list it on the local bus memory map, just the PCI memory map). Identify all addresses that the user can adjust by changing config.h, as being software controllable. Identify all bus master devices (DMA) on each bus.
- If VME is present, describe the default parameters for the master windows and how to change them, if possible. (master access means accessing some other board from this board). Describe the default parameters for the slave access windows, and how to change them, if possible. (Slave access means an access from another bus master). The normal VxWorks default is to enable the slave access window(s) only on CPU 0, as part of the routine sysProcNumSet(). Otherwise, slave accesses are not normally permitted.
- If PCI is present, describe the configuration access methods used to access PCI configuration space. Describe any address mapping from local addresses to PCI addresses and vice versa. Describe how the user can alter this, if appropriate.

Shared Memory

- If appropriate, identify the main bus and the type of shared memory support provided, if any. Identify the type of mailbox support and Test-And-Set support used. Identify any special support, or consideration, necessary from other boards in a multi-board system.

Interrupts

- Provide a list of interrupts/exceptions by priority and/or vector number. Describe any special interrupt connection routines or considerations used with this BSP.

Serial Ports

- Identify the default configuration for all serial ports. Document any special considerations: i.e. hardware flow control cannot be disabled thus requiring a jumper to be installed when no terminal device is connected.

SCSI Support

- Are there any known SCSI configuration limitations? Does the BSP support SCSI-1 or SCSI-2? Describe any special connectors or adapters. Does the board include terminators on-board? Is there a jumper to activate the termination network?

Ethernet Support

- Describe the Ethernet ports on the board. Identify how the MAC address is obtained or specified for each port.

Delivered Objects

- Which pre-compiled objects are delivered with the BSP?

Make Targets

- If there is any special information about a specific make target, then it should be explained. Identify any standard targets that do not build or run correctly. (bootrom, bootrom_uncmp, bootrom.res_rom, examples on VxWorks also include: vxWorks, vxWorks.st, vxWorks.res_rom, vxWorks.res_rom_nosym, vxWorks_rom).

Special routines

- Are there any unique BSP routines available to the user?

Divide by zero exception

- Describe exception handling of divide by zero event.

Host-Target Communications Support

- Name of client server agent such as WDB for VxWorks
- Describe additional WDB agents supported for debug support with host if any.

Some details on the collection of BSP libraries

BSP Source and Include Files reside in the following two subdirectories of a Tornado installation:

- *TornadoBaseDirectory/target/config/all*
- *TornadoBaseDirectory/target/config/BSPname*

See figure 2 for a view of these subdirectories and fiels. The subdirectory *TornadoBaseDirectory /target/config/bspname* contains the hardware dependent Board Support Package (BSP), which consists of files for the particular hardware used to run VxWorks, such as a CompactPCI board with specific semiconductor silicon assets for serial lines, timers, and other devices. The files include:

- Makefile
- sysLib.c
- sysSerial.c
- sysALib.s
- romInit.s
- bspname.h
- config.h.

Two target-specific libraries, sysLib and sysALib, are included with each port of VxWorks. These libraries are the heart of VxWorks portability; they provide an identical software interface to the hardware functions of all boards. They include facilities for hardware initialization, interrupt handling and generation, hardware clock and timer management, mapping of local and bus memory spaces, memory sizing, and so on.

The System Library

The file sysLib.c provides the board-level interface on which VxWorks and application code can be built in a hardware-independent manner. The functions that should be addressed include:

- Initialization functions
  1. initialize the hardware to a known state
  2. identify the system

- Initialize drivers, such as SCSI or custom drivers

- Memory/address space functions
  1. get the on-board memory size
  2. make on-board memory accessible to external bus (optional)
  3. map local and bus address spaces
  4. enable/disable cache memory
  5. set/get nonvolatile RAM (NVRAM)
  6. define the board's memory map (optional)
  7. virtual-to-physical memory map declarations for processors with MMUs

- Bus interrupt functions
  1. enable/disable bus interrupt levels
  2. generate bus interrupts

- Clock/timer functions
  1. enable/disable timer interrupts
  2. set the periodic rate of the timer

- Mailbox/location monitor functions (optional)
    1. enable mailbox/location monitor interrupts

The sysLib library does not support every feature of every board: some boards may have additional features, others may have fewer, others still may have the same features with a different interface. For example, some boards provide some sysLib functions by means of hardware switches, jumpers, or PALs, instead of by software-controllable registers.

The configuration modules usrConfig.c and bootConfig.c in *TornadoBaseDirectory/target/config/all* subdirectory are responsible for invoking this library's routines at the appropriate time. Device drivers can use some of the memory mapping routines and bus functions.

Each BSP also includes a boot ROM or other boot mechanism. Many of these import the run-time image from the development host. For information on boot ROMs and other booting mechanisms see Tornado Getting Started manual and VxWorks User 's Guide section 8.9 Creating Bootable Applications. For information on target-specific libraries, see 8.2 The Board Support Package (BSP) in the VxWorks User's Guide and the target-specific reference entries for your board type starting with the file target.nr on the BSP distribution.

Virtual Memory Mapping

For boards with MMU support, the data structure sysPhysMemDesc defines the virtual-to-physical memory map. This table is typically defined in sysLib.c, although some BSPs place it in a separate file, memDesc.c. It is declared as an array of the data structure PHYS_MEM_DESC. No two entries in this descriptor can overlap; each entry must be a unique memory space.

The sysPhysMemDesc array should reflect your system configuration, and you may encounter a number of reasons for changing the MMU memory map, for example: the need to change the size of local memory or the size of the VME master access space, or because the address of the VME master access space has been moved. For information on virtual memory mapping, as well as an example of how to modify sysPhysMemDesc, see VxWorks User's Guide section 7.3 Virtual Memory Configuration.

The Serial Driver

The file sysSerial.c provides board-specific initialization for the on-board serial ports. The actual serial I/O driver is located in the *TornadoBaseDirectory /target/src/drv/sio* directory. The library ttyDrv uses the serial I/O driver to provide terminal operations for VxWorks.

BSP Initialization Modules

The following files initialize the BSP:

The file romInit.s contains assembly-level initialization routines.

The file sysALib.s contains initialization and system-specific assembly-level routines.

BSP Documentation

The target.nr file located in the *TornadoBaseDirectory /target/config/bspname* directory is the source of the online reference entries for target-specific information in our new BSP web site. For a description on how to view the target.nr reference files supplied with our product distributions, see the Tornado Getting Started manual. The target.nr file describes the supported board variations, the relevant jumpering, and supported devices. It also includes an ASCII representation of the board layout with an indication of board jumpers (if applicable) and the location of the ROM sockets.

References

- Tornado Getting Started Manual
- VxWorks User 's Guide
- Tornado BSP Developer's Kit for VxWorks: User's Guide (some times referred to as a porting guide)
- Tornado BSP Training Workshop - book 1 (see Wind River training web site for BSP class times)
- Wind River's BSP web site (to be launched soon)