Building Board Support Packages

Bradford Harrison
Senior Technical Writer

Microtec provides customers with two types of Board Support Packages (BSPs): XRAY Debugger Monitor and Spectra BSP.

The XRAY Debugger Monitor (XDM) debugs single-threaded applications that do not require an operating system. A Spectra BSP implements the Xtrace Monitor and Xtrace Protocol, and supports single-threaded applications (non-OS mode debugging) or multi-threaded applications running on VRTX/OS (OS mode debugging). This article describes both types of BSPs. However, the Spectra BSP is more sophisticated, so it is described in greater depth.

BSP Development Tools

Embedded systems development requires BSPs, since commercial microprocessor target boards typically only ship with a basic debug monitor, which is used for boot and initialization of the board and to establish a serial (RS-232) connection with a host system. The developer must supply his own board support services for embedded software development.

BSPs are purchased prebuilt or are built by the developer or by third-party consultants. Because pre-built BSPs, especially for custom hardware configurations, are unavailable for every board, embedded systems software suppliers provide BSP development tools to facilitate the development of custom BSPs.

BSP development tools must support a wide range of board and processor configurations, and be easy to use. The developer must have the ability to customize his BSP to whatever degree is required. These BSPs provide services that support the overall embedded systems software development environment sold by the supplier.

XRAY Debugger Monitor (XDM)

To create an XDM, Microtec provides the Monitor Configuration Tool (MCT). This menu-driven host utility creates a highly functional debug monitor, which is downloaded to the target system via the target board's own built-in debug monitor. The developer then issues a **go** statement to the starting address of the XDM, and the XDM takes control of the board, supporting all XRAY commands and services for the development of a single-tasking application.

MCT provides direct support for a variety of target boards. BSPs generated by MCT immediately can be downloaded and executed on the target. If the target is not directly supported by MCT, the developer must use MCT to generate the "next closest" BSP (configured for, at a minimum, a processor from the same family as the developer's board), and alter the source code before compiling and downloading to the target. Microtec provides complete documentation detailing how to alter this code, but the developer brings knowledge of the processor and peripheral devices to the task.

The XDM executable can be linked with boot and initialization code to provide a complete BSP package. This BSP is programmed into PROM, eliminating the need for the manufacturer's built-in debug monitor.

The boot and initialization code can be written using documentation or software supplied by the board manufacturer. The boot and initialization code is often taken by the programmer from the supplied debug monitor using XRAY's disassembling capabilities. It is then assembled and linked with XDM.

Spectra BSP

A Spectra BSP provides more services than does an XDM. It supports and debugs single-threaded applications or multi-threaded applications running on an OS. The OS often is, but need not necessarily be, VRTX/OS.

Spectra BSP development uses Microtec BSPBuilder development tools and methodology, including forms (templates), procedures, makefiles, development tests, header files, and libraries.

The forms, procedures, and tests are relatively generic. Those more target-specific features consist of the libraries supplied with BSPBuilder and the prebuilt (binary) drivers that BSPBuilder provides for the support of a variety of processors and I/O devices. If the developer builds a BSP for a board that contains supported processors and I/O devices, then the use of prebuilt drivers dramatically reduce BSP development time.

BSPBuilder forms are used to create device drivers in cases where developers do not use prebuilt drivers. Serial, Ethernet, timer, and shared memory drivers for a variety of processors and I/O devices are generated using forms, and can be combined with the binary drivers into a single BSP.

Iterative BSPBuilder Development Process

BSPBuilder BSP development is iterative, as shown in Figure 1. Developers begin usually with a serial driver, though Ethernet drivers are often developed first when there is only a single serial port on the board. If there are two serial ports, the developer can retain the connection to the board's debug monitor while developing the serial driver for the other port. If there is only a single serial port, developers can continue to use the serial port for the debug monitor while developing an Ethernet device driver.

In either case, the goal is to develop a working driver in order to build a working Spectra BSP and establish a working Spectra bridge: the connection between the Xtrace Daemon running on the target, and Target Manager, running on the host. Once a Spectra bridge is operational, the serial, Ethernet, or shared memory device can be shared between Xtrace and an application, and high-level debugging can begin.

If they exist for the board, binary drivers are used; otherwise, forms are used to develop the drivers. In either case, the driver is then linked with the appropriate BSPBuilder test code, files, and libraries using BSPBuilder-supplied makefiles. These files and libraries include the following:

- logio.lib — Contains modules that implement the Logical I/O (*logio*) layer of Spectra.
- board.c — Contains board initialization code. **board.c** contains code that handles the non-configurable portion of board set-up. These functions need to be performed whether or not any of the board's devices are installed.
- devcnfg.c — Defines configuration data structures containing information necessary to program devices on a board. For example, a data structure specifies the type of device (packet or tty), clock speed, baud rate, parity, stop bits, and receive, transmit, and error vectors for any given serial device. It specifies port addresses and functions for enabling and disabling interrupts. For each device, **devcnfg.c** declares a device descriptor and a *logio* method. It also defines a function for installing all of the devices on a board as *logio* devices. The developer is required to modify this file for her board.
- boot.lib — Contains code for initializing the target board and bringing it to a known state after power-up or reset. This library contains modules that initialize *logio*, create and initialize devices as *logio* devices, set up a console, copy the memory map from ROM to RAM, check for overlaps in the memory map, and construct the Boot Item List.
- cpu.lib — Supports processor-specific features. It defines cache and MMU functions, register set (including floating-point), and interrupt control functions.
- packt.lib — Contains code to support packetized communication on a serial line used by the Xtrace protocol. The library includes functions to encapsulate buffered data in a packet, compute a checksum, compare checksums, and extract data from a packet.

The executable is downloaded to the target using the target board's debug monitor. The device driver is tested using a combination of test code on

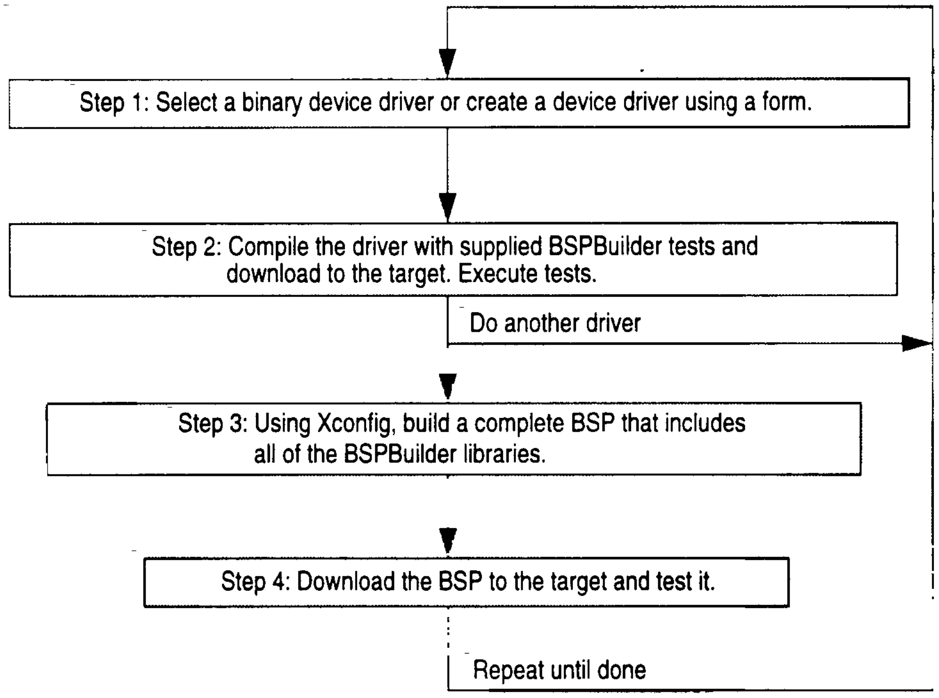
Step 1: Select a binary device driver or create a device driver using a form.
Step 2: Compile the driver with supplied BSPBuilder tests and download to the target. Execute tests.
Do another driver
Step 3: Using Xconfig, build a complete BSP that includes all of the BSPBuilder libraries.
Step 4: Download the BSP to the target and test it.
Repeat until done

Figure 1. Spectra BSP Development Process

the target and tools on the host.

If the developer has a problem with getting the first Ethernet or serial driver to work (there is no response from the target at all), then the developer can use the BSPBuilder memory console debugging feature. This allows the developer to insert **printf** statements into the driver code in order to write diagnostics to memory. When the driver fails, memory can be examined using the debug monitor to determine the cause of the failure.

Once the driver checks out, the developer then either goes onto the next driver, or using the Spectra Xconfig configuration utility, creates a BSP containing the driver and the remaining BSPBuilder files and libraries necessary to build a complete BSP. These files and libraries are the following:

- crt0.s — Includes all necessary code for starting up the target board. It defines a stack for use at start-up/reset; initializes the program counter and stack pointer; disables all interrupts; flushes and invalidates the cache (if present and enabled); and clears RAM. If the board comes packaged with debug PROMs, the code in **crt0.s** may closely resemble the code present in the debug PROMs.
- bootcnfg.c — Generated by Xconfig from the template **bootcnfg.tpl**, this file declares functions, constant data, tables, boot stack, Xtrace workspace, and other boot-related items with configurable values.
- devices.c — Generated by Xconfig from the template **devices.tpl**, this file declares a table of the board's devices. For each device, the table establishes a correspondence between the device name (a character string used to identify the device) and the *logio* method used to access it. The *logio* method is a data structure containing pointers to two other data structures. The first of these defines a set of operations valid for members of some family of devices. The second contains all information pertinent to a specific instance of a device. The device belongs to the class of devices (timer, serial, Ethernet, or shared memory) characterized by the first data structure.
- vtdm.lib — Encapsulates Xtrace. This library contains the functions that execute as the Xtrace Daemon (also known as the debug server) on the target. The debug server communicates with Target Manager (running on the host) over the Spectra bridge device. These three components — debug server, Spectra bridge, and Target Manager — make up the Spectra debug connection.
- router.lib — Contains functions that route messages passed over the Spectra bridge. This library is used in both single-board and multi-board configurations, and enables an application to share the Spectra bridge with the debug server (Xtrace) running on the target.

The primary files customized for a particular board are **crt0.s**, **board.c**, and **devcnfg.c** for board start-up, initialization, and device configuration.

The complete BSP is downloaded to the target and tested. This BSP, with only a single working serial or Ethernet driver, can now be used to establish a working Spectra bridge. Over a working bridge, the developer can connect with Target Manager and (for serial devices) Connection Server, and use the Spectra host toolset to work with the BSP.

Many developers at this point opt to create a PROM containing their BSP, thereby eliminating the need to work with the board's debug monitor.

The iterative development process continues until all device drivers are working and the BSP is thoroughly tested. A final series of tests check the BSP before the BSP is used to run the OS and develop applications.

BSPBuilder Tests

The BSPBuilder methodology implements several levels of tests, including code to run simple character receive and transmit tests, as well as packet receive and transmit tests. The hostside **rstest** utility is provided in order to receive and transmit packets over a serial link to the serial driver without the use of Connection Server, which requires that a Spectra bridge be in place. The **ethertst** utility, included on most UNIX systems, tests the Ethernet driver.

Once a driver is working properly and a BSP has been built with the Xconfig utility, the BSP is downloaded to the target, and standard Spectra host tools are used to test the BSP. For example, the XSH or XRAY for Spectra debugger can be used to check registers or to investigate memory locations using the services of the BSP. Target Manager must be running on the host, and if over a serial connection, Connection Server must also be running.

BSPBuilder includes as well tests for timer and shared memory drivers, and a serial console test.

logio

Critical to Spectra BSP development is *logio*. Provided as a library of functions, **logio.lib**, *logio* allows devices to be treated as classes of devices that can be operated on by logical function calls rather than device-specific calls. Once created, a *logio* device can be handled logically just like any other similar device.

logio provides three critical services for a BSP:

1. BSP interrupt handling (or the ability to poll for them)
2. BSP initialization
3. Access to BSP services from applications, ISRs, and other software

Xtrace, the Spectra debug monitor, has first claim on all interrupts. This capability is necessary so that a bridge can be shared between Xtrace and an application. *logio* is notified of an interrupt only when Xtrace makes a call to a *logio* interrupt handler. The *logio* default interrupt handler makes calls to handlers for each logical event (interrupt source) that occurs on a vector. These logical event handlers check what caused the interrupt to fire and whether it was a specific event. If the interrupt fired in response to the event the function is handling, the function calls the appropriate ISR and returns **TRUE**. If some other event caused the interrupt, the function returns **FALSE**.

logio initialization routines create and delete logical devices, return device IDs, initialize the devices, and initialize pools of fixed-size buffers used to hold data received and transmitted by a driver.

logio also provides seven standard functions that applications and ISRs use in order to operate on a device. These are:

- initialize
- read
- write
- getmsg
- putmsg
- control
- poll

These standard functions are supplied in four "interface" files — **serial_2.o**, **ether_1.o**, **timer_1.o**, and **shmem_1.o** for serial, Ethernet, timer, and shared memory drivers, respectively. Source code is also provided, for reference only. When the driver is created, the appropriate binary is linked with compiled source code developed in the appropriate driver form. (If the developer is using a supplied binary driver, then the developer does not need to work with this code, already provided in the precompiled driver.)

The driver forms — **sform**, **eform**, **tform**, and **vmeform** for serial, Ethernet, timer, and shared memory drivers, respectively — provide "boilerplate" code for device-specific functions that are coded by the developer. There are numerous functions for each driver developed, and they are accessed by the seven high-level functions using the Function Operations (FOPS) table. The FOPS table contains pointers to those developer-provided device-specific functions just developed in the appropriate form and compiled with the appropriate interface file. Keeping the high-level interface functions separate from the low-level device-specific functions is referred to as **externio**.

The developer often requires the source code provided for the interface files in order to determine exactly how to access the functions under development in the forms. The forms themselves have comments in order to help the developer provide the appropriate code for each device. If a pre-compiled device driver is available for the device, in theory, the process becomes simple, and the developer needs only to link it with the libraries and files described above.

Conclusion

Further modifications are often required once the BSP has been developed, tested, and used as a platform in order to develop an application. Xtrace may be removed from the BSP and the amount of memory required by the BSP reduced by following the procedures described in the *Configuration Tool User's Guide and Reference*.

BSPBuilder currently supports the Motorola 68K and PowerPC processor families with processor libraries and binary device drivers. The methodology, however, is generic, and developers can use the forms and *logio* calls to develop BSPs for other processors and devices. Microtec Consulting Services can be contacted to help with BSP development for custom target boards. Additionally, Microtec provides BSPBuilder training classes.