NS9750 Release Notes: NET+Works with Green Hills

Operating system: NET+OS 6.1
Part number/version: 93000519_C
Release date: June 2004
www.netsilicon.com

Contents

Using This Document

Review this section for basic information about this document, as well as for general support contact information.

About this document

This document describes how to build and debug a sample application using NET+OS 6.1 with Green Hills.

Software release

The content of this document supports NET+OS 6.1. By default, this software is installed in the `C:/NETOS61_GH361/` directory.

Who should read this document

This document is for software engineers and others who use NET+Works for NET+OS. To complete the tasks described in this document, you must:

- Be familiar with installing and configuring software.
- Have sufficient user privileges to do these tasks.
- Be familiar with network software and development board systems.

Conventions used in this document

This table describes the typographic conventions used in this document:

This convention	Is used for
italic type	Emphasis, new terms, variables, and document titles.
bold, sans serif type	Menu commands, dialog box components, and other items that appear on-screen.
Select **menu → option**	Menu commands. The first word is the menu name; the words that follow are menu selections.
`monospaced type`	File names, pathnames, and code examples.

Customer support

To get help with a question or technical problem with this product, or to make comments and recommendations about our products or documentation, use the contact information listed in the next tables.

NetSilicon support

For	Contact information
Technical support	Telephone: 1 800 243-2333 / 1 781 647-1234 Fax: 1 781 893-1388 E-mail: tech_support@netsilicon.com
Documentation	techpubs@netsilicon.com
NetSilicon home page	www.netsilicon.com
Online problem reporting	www.netsilicon.com/problemreporting.jsp

Digi support

For	Contact information
Technical support	Telephone: 1 877 912-3444 / 1 952 912-3456 Fax: 1 952 912-4960
Documentation	techpubs@digi.com
Digi home page	www.digi.com/support/eservice/eservicelogin.jsp
Online problem reporting	www.digi.com/problemreporting.jsp

Documentation updates

NetSilicon occasionally provides documentation updates on the Web site.

Be aware that if you see differences between the documentation you received in your NET+Works package and the documentation on the Web site, the Web site content is the latest version.

Building and Debugging a Sample Application

CHAPTER 1

This chapter provides a brief hands-on tutorial for NET+OS with Green Hills.

Overview

In this tutorial, you will:

- Set up the MAJIC debugger.
- Build and download a sample application.
- Run and debug the sample application.
- Specify configuration options for the development board.

The tutorial is brief — it will take approximately 15-20 minutes — so you can go through it in one sitting.

Task 1: Preparing to do the tutorial

This section describes what you need to do before you get started.

Make sure the hardware and software are installed

Verify that the hardware and software are installed. The hardware installation instructions are in the *Quick Install Guide*, and the software installation uses a wizard to guide you through the process.

The instructions in this tutorial are based on the assumption that you installed NET+Works in `C:/NETOS61_GH`, the default installation directory.

Gather information

See your network administrator for information you'll be prompted for when you configure the development board.

Here's what you need to find out; you may find it helpful to write down the information in the space provided:

- IP address for the board_____
- IP address for the MAJIC_____
- Subnet mask _____

- Default gateway _____
- MAC address _____

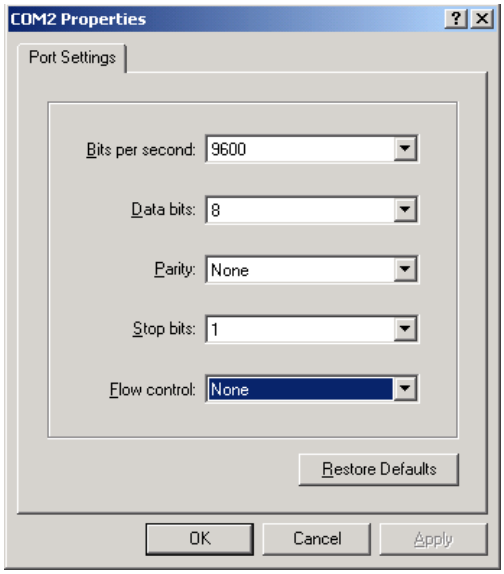
Start a terminal session

From your PC, open a (serial) terminal connection to the development board so you can view outputs from the board. Although several terminal connection applications are available, the one shown in this tutorial is HyperTerminal.

To use HyperTerminal:

1 Select **Start → Programs → Accessories → Communications → HyperTerminal**.

2 Make sure the selections are set up as shown here:



As you go through this tutorial, keep the HyperTerminal window open for quick access.

Task 2: Setting up the MAJIC debugger

In this section,you will configure your debug environment, using a wizard.

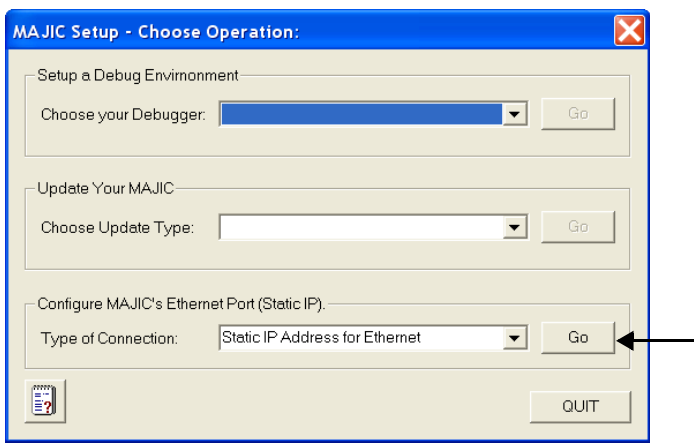
▶ **To set up the debugger:**

1 Select **Start → Programs → EPI Tools EDTA → MAJIC Setup Wizard**.

The **EPI MAJIC Setup Wizard Introduction** window opens:

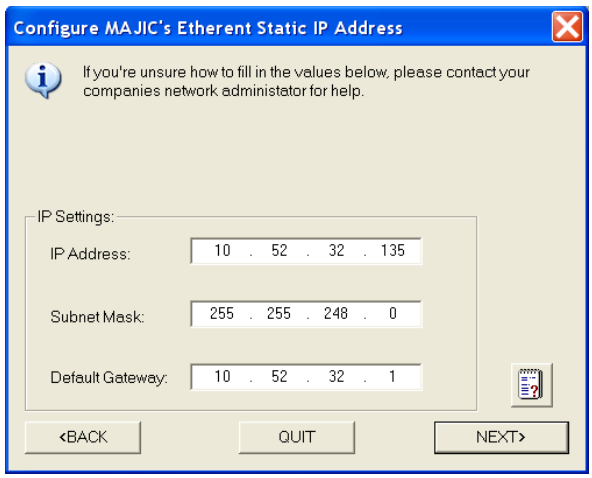
2 Click **NEXT**.

The **Choose Operation** window appears:



3 From the **Type of Connection** pulldown menu, select **Static IP Address for Internet**, and then click **Go** (as indicated by the arrow in the previous illustration).

The **Configure MAJIC's Ethernet Static IP Address** window opens:

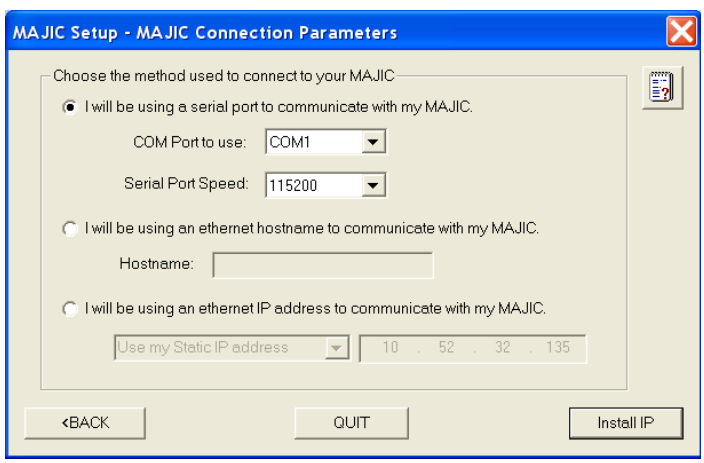


4 Enter the information that you previously gathered:

- IP address
- Subnet mask
- Default gateway

and click **NEXT**.

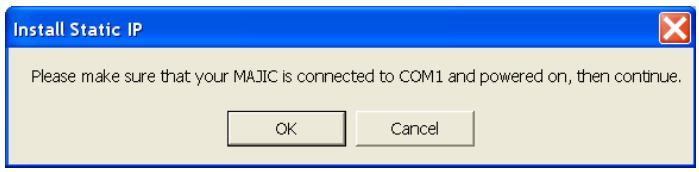
The **MAJIC Connection Parameters** window appears:



5 Do these steps:

- Click **I will be using a serial port to communicate with my MAJIC**.
- Connect the MAJIC serial crossover cable between the MAJIC serial port and an available COM port on your PC, and from the COM port to use pulldown menu, select the serial port number.
- Click **Install IP**.

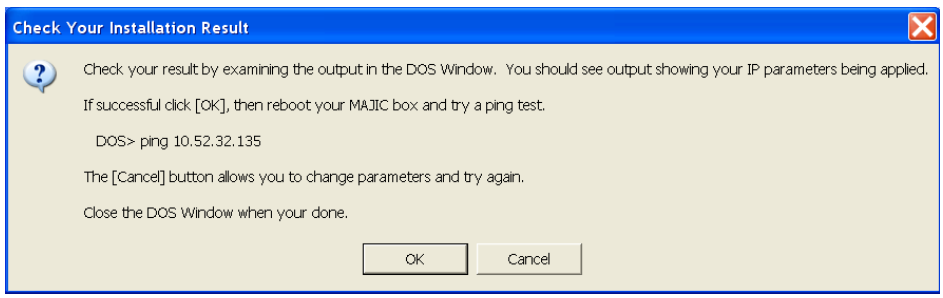
You see this dialog box:



6 Verify that the serial cable is attached from your PC to the MAJIC, and then click **OK**.

A dialog box and a DOS window open.

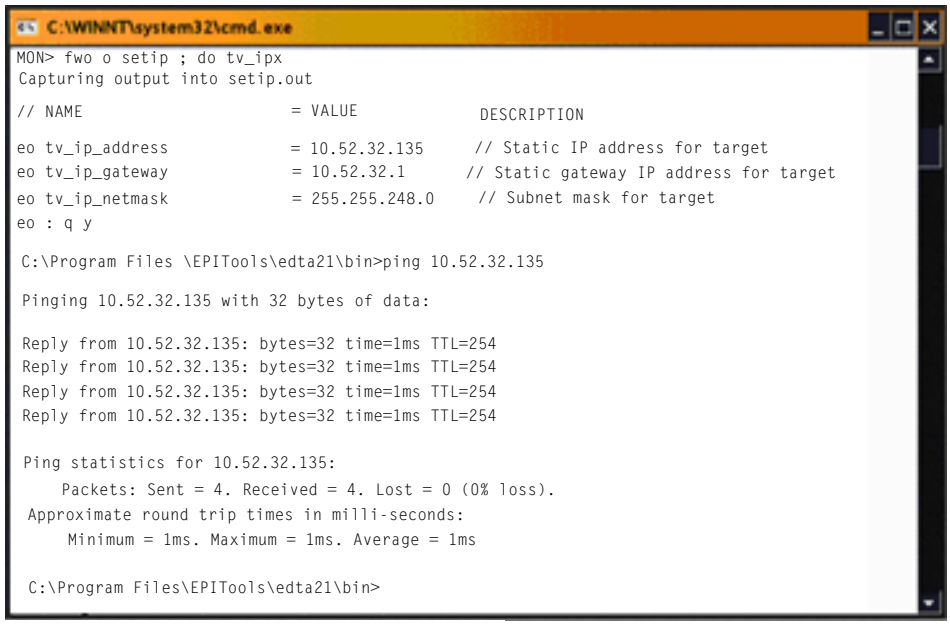
7 The **Check Your Installation Result** dialog box prompts you to verify that the IP address information in the DOS window is correct:



8 Power-cycle the MAJIC.

9 In the DOS window, `ping` the IP address.

This is what you see in the DOS window:



Note that the colors in this illustration have been changed for easier reading.

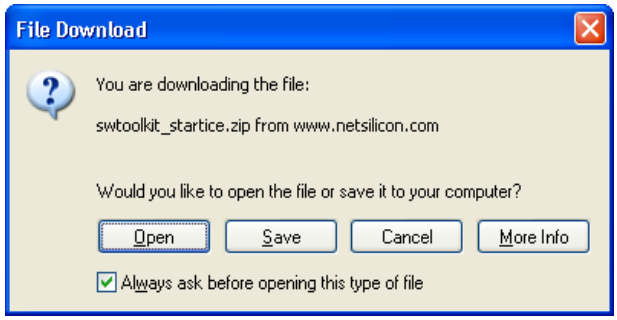
If the `ping` reply is successful, the IP address is installed.

10 Close the DOS window.

11 In the **Check Your Install Results** dialog box, click **OK**.

12 Using a browser, go to `www.netsilicon.com/support/softwaretoolkit.jsp`, and click **NET+OS 6.1 MAJIC debugger update (June 2004).**

The **File Download** dialog box opens:



13 Do either of these steps:

- Open the `startice.cmd` file and extract it to the `c:/ghs/arm361` directory, overwriting the existing `startice.cmd` file.
- Save the `swtoolkit_startice` WinZip file and unzip it to the `c:/ghs/arm361`directory, , overwriting the existing `startice.cmd` file.

Task 3: Building and downloading the template application

In this section, you will build `netCentral` and then build `Hello World`, one of the template applications provided with NET+Works.

About building netCentral

In typical use, you don't need to build `netCentral` every time you build an application. The first time you use the software, however, you must first build `netCentral`, and then build your application.

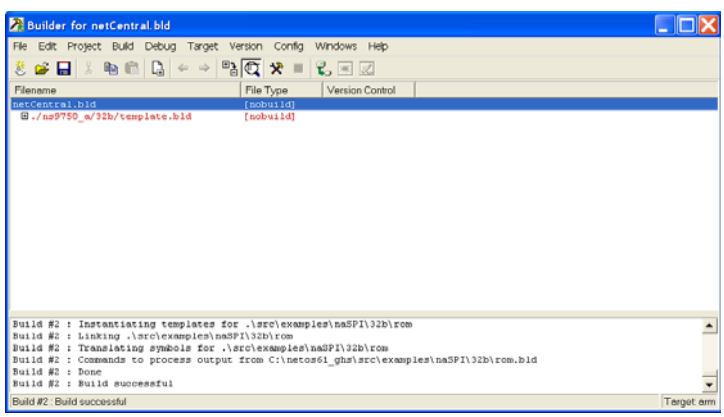
Building netCentral

▶ **To build netCentral:**

1 In Explorer, go to the release directory:

`C` → `NET+OS61_ghs` → `netCentral.bld`

The **Builder for netCentral.bld** window opens:



2 Select **Build → Rebuild All**.

This build may take a few minutes to complete.

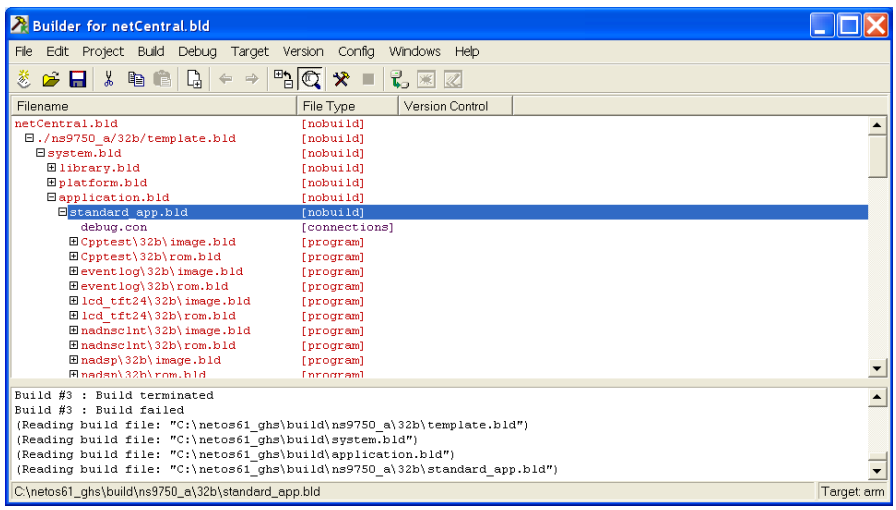
When the process finishes, you see this message in the bottom part of the window:

```
Build successful
```

3 Close any child windows that open during the build.

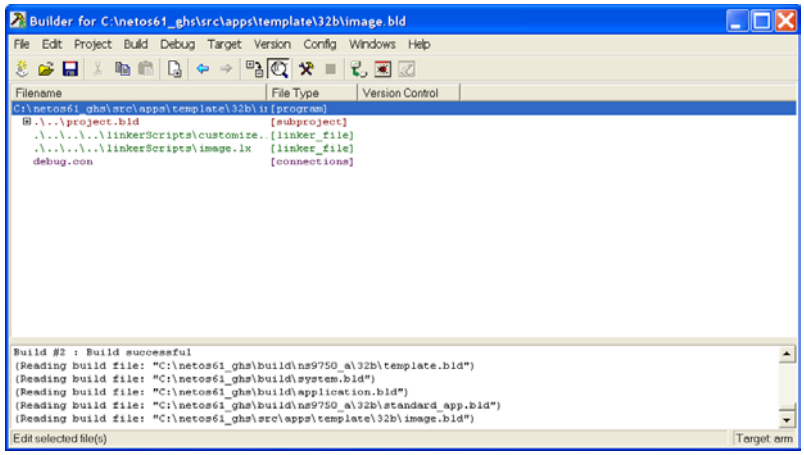
Building the template application

1 In the **Build for netCentral.bld** window, navigate to `standardapp.bld`, as shown here:



2 Under `standardapp.bld`, double-click `template /32b/image.bld`.

The window changes to the **Builder for …/template/32b/image.bld** window:



3 Select **Build → Build image.bld**

When the process finishes, in the bottom part of the window, you see this message:

```
Build successful
```

Preparing the board

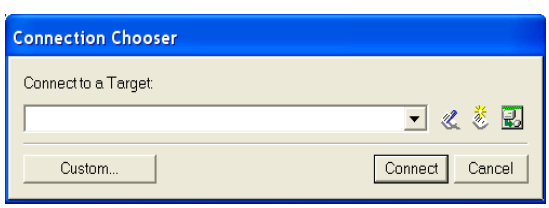
To prepare the board for the download, power cycle the development board and the MAJIC.

Downloading the template application

▶ **To download the template application:**

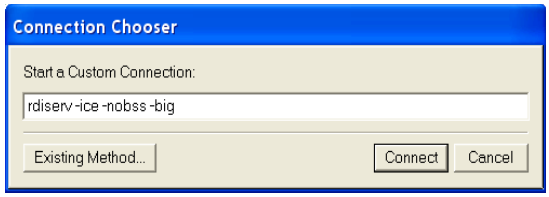
1 In the **…/template/32b/image.bld** window, select **Target → Connect to Target**.

The Connection Chooser opens:



2 Because no targets exist at this point, click **Custom**.

The Connection Chooser opens again so you can specify a custom target:

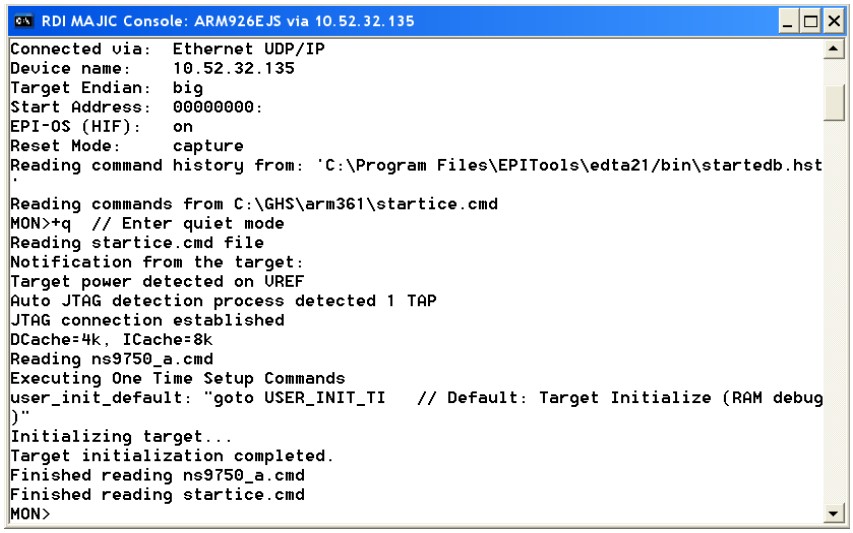


3 From the **Start a Custom Connection** text box, enter this command:

rdiserv -ice -nobss -big

and then click **Connect.**

The **RDI MAGIC Console** window opens:



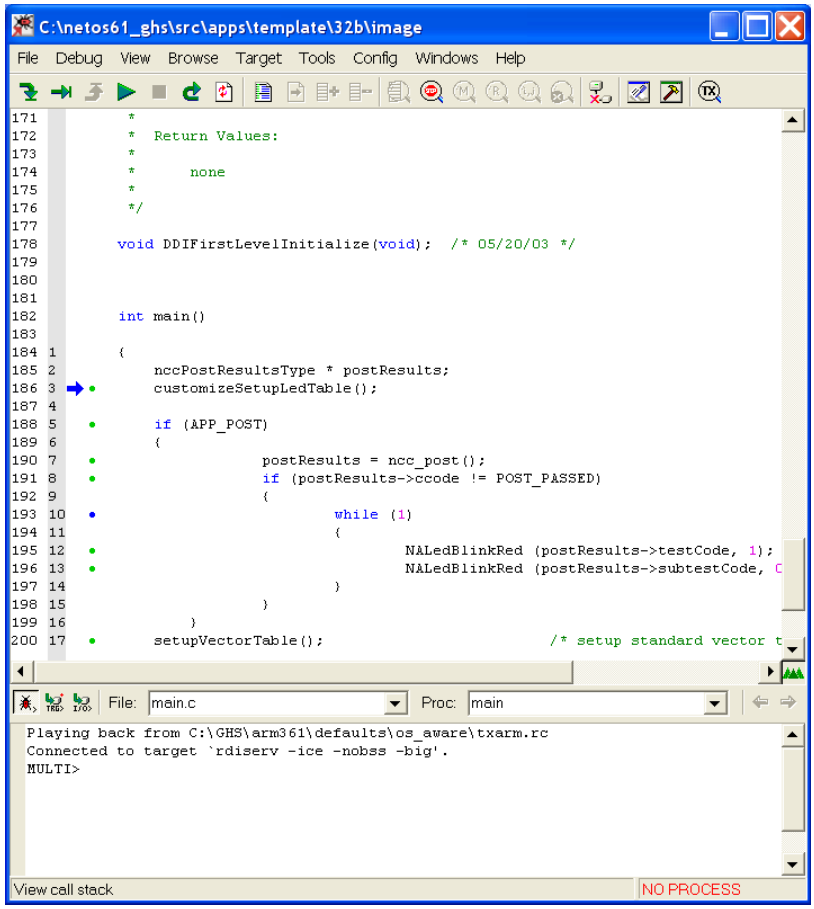
You see messages about connecting to the target, and the window is minimized.

When the communication finishes, you see this message near the bottom of the window:

```
Target initialization completed
```

4 Select **Debug → Debug image.**

The **…/src/apps/template/32b/image** debugger window opens:



5 If you want to set a breakpoint, at the MULTI prompt in the bottom of the window, enter this command:

```
b applicationStart
```

and then press Return.

6 To run the template application, enter this command at the MULTI prompt:

```
c
```

and press Return.

You see the download box as the template application is being downloaded. This process can take up to a few minutes, depending on your system.

The application does one of these steps:

- If you did not set a break point, the application runs to completion.
- If you set a break point in step 6, the application runs until it encounters the break point.

  When the application stops, select **Debug → Go** to run the application to completion.

  Note that `Hello World` appears on the bottom line of the terminal widow.

Configuring the development board

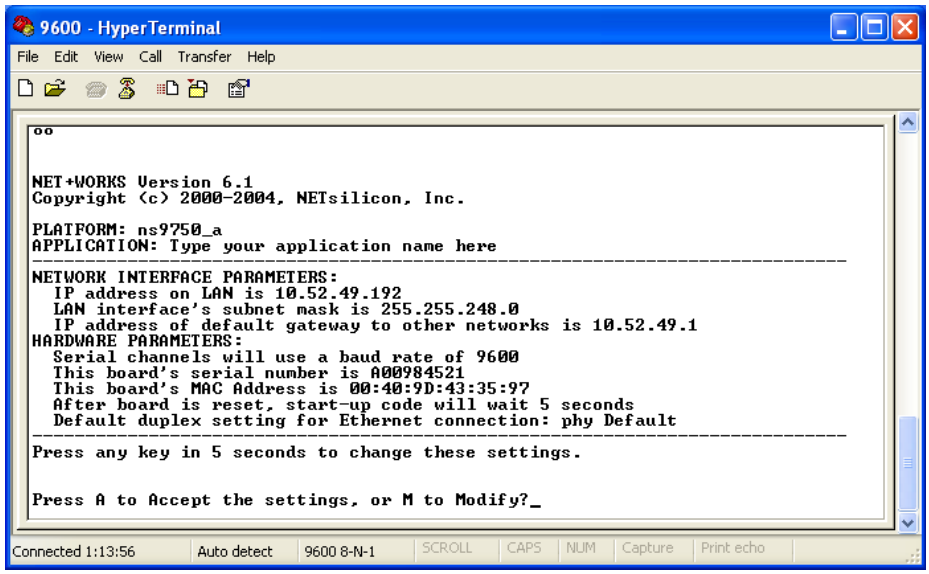
In this procedure, you need to be prepared to go quickly to your HyperTerminal after you do step 5, because you have only a few seconds to respond to the prompt.

▶ **To configure the development board:**

1. Power-cycle the development board and the MAJIC.
2. In the `appconf.h` file, verify that the USE_NVRAM define is set to 1.

   If you need to change this setting, you must rebuild your application.
3. Select **Target → rdiserv -ice -nobss -big.**
4. Select **Debug → Debug image.**
5. Get ready to go to your HyperTerminal window, and begin the download by selecting **Debug → Go**.

In the HyperTerminal window, you see this information:



6 To change the configuration, press M and then Enter.

A password prompt appears.

7 Enter `Netsilicon`, the default password.

The first of a series of configuration prompts appears.

8 At each prompt, do one of these steps:

- To accept the current value, press Enter.
- To change a setting, enter a value and press Enter.

After you scroll through the settings, the display shows your updated configuration, and the prompt appears to accept the changes. When the prompt times out, execution returns to the `Hello World` application.

Using Central Build

CHAPTER 2

This chapter describes the central build system and how to use it.

Overview

The central build system is a set of build files that operate under the Green Hills MULTI 2000 environment. This system uses one build file as the main access point for building all the libraries, the BSP, and the applications you need for a NET+OS project.

Design

The design of the central build system gives you access to the NET+OS under one central location in the Green Hills environment, allowing you to navigate the build environment more easily. In addition, with this design, your application can inherit build defines and compiler options from `netCentral.bld`, the master build file.

Each supported platform contains a template build file that controls the options that are used during the build process. At the tip of the hierarchy is the build file – `template.bld`. The parent build file – `netCentral.bld` – contains templates for all supported platforms. Each platform is structured by the definition of a system, which consists of a platform, library, and application template build file that define the options for the entire platform.

Using a standalone build environment (without central build)

Also provided with NET+OS are examples of standalone build files that allow you to build your application without using the central build environment. These build files are:

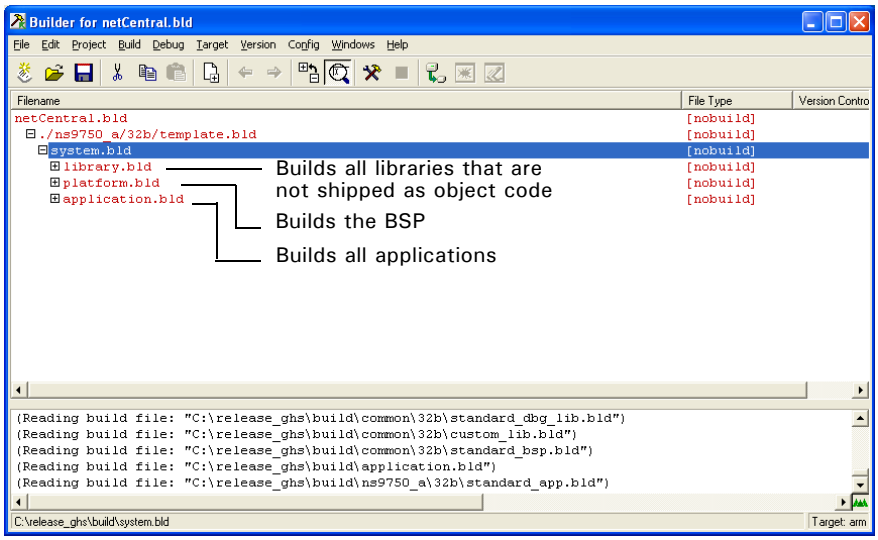
- `src/apps/template/32b/imageArm9.bld` – Builds a debug image
- `src/apps/template/32b/romArm9.bld` – Builds a ROM image

If you prefer not to use the central build, use these files as templates for build files for your application. To use the standalone build, copy these build files into your `applications/32b` directory. Then open either the `imageArm9.bld` or the `romArm9.bld` project in MULTI 2000 instead of `netCentral.bld`.

Structure

When you build the central build system, always open the `netCentral.bld` build. From here, you can either build the entire system or navigate to your application's build file.

The central build file includes `template.bld`, which is a platform-specific template build file. This file contains the master build options that the lower-level build files inherit. The lower-level build files are divided into three component types – library, platform and application – as shown here:

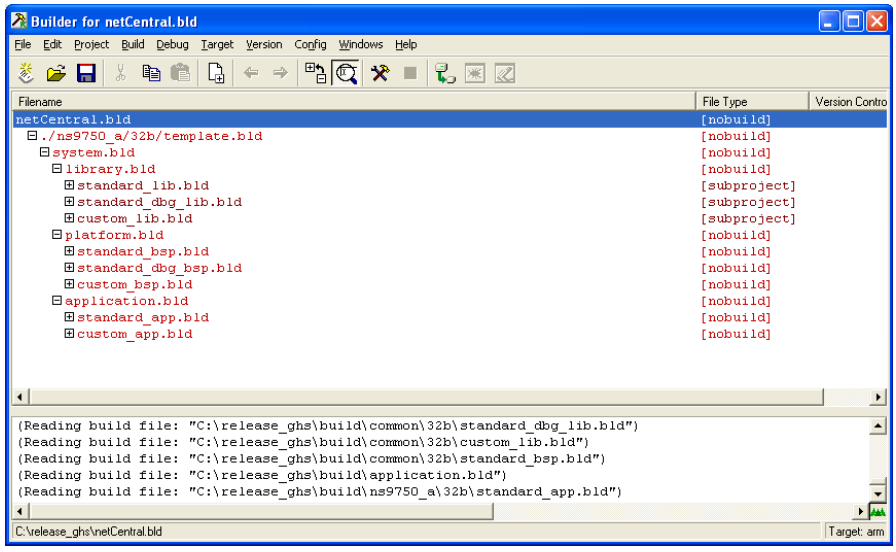


The system recognizes the files used by the BSP, libraries, and applications by accessing these predefined sub-build files:

- Libraries:
  - `standard_lib_bld` – Contains the posix, flash, and SNMPD library build files
  - `standard_dbg_lib_bld` – Contains the posix, flash, and SNMPD library build files with debug turned on
  - `custom_lib.bld` – Customer-added libraries

- Platform BSP:
  - `standard_bsp.bld` – Builds the BSP and the bootloader
  - `standard_dbg_bsp.bld` – Builds a debug version of the BSP and the bootloader
  - `custom_bsp.bld` – Contains customer BSPs
- Applications:
  - `standard_app.bld` – All the standard applications that ship with NET+OS
  - `custom_app.bld` – Customer applications

These are the build files:



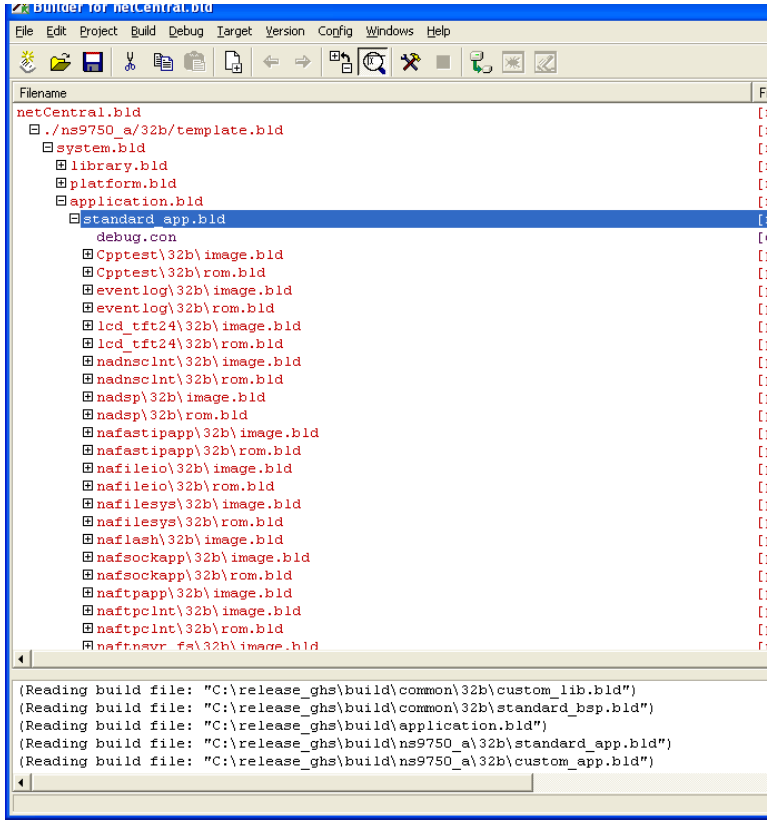
This table provides an overview of the build files:

File name	Description	Location	Content
netCentral.bld	Main access to all platforms in the system. Open this file when you want to build.	./	template.bld
template.bld	Defines a specific platform with options such as endian, CPU type, optimization, and debug level	./netos/build/32b	system.bld
system.bld	Defines the system's platform BSP, library, and applications	./build	■ platform.bld ■ library.bld ■ application.bld
platform.bld	Standard and custom BSPs	./build	■ standard_bsp.bld ■ standard_db_bsp.bld ■ custom_bsp.bld
library.bld	Standard, custom, and in-house libraries	./build	■ standard_lib.bld ■ standard_dbg_lib.bld ■ custom_lib.bld

Applications

The *standard applications* are those that ship with NET+OS, such as the examples for the Web server and the FTP server.

These applications, which are listed under `netCentral.bld` → `template.bld` → `system.bld` → `application.bld` → `standard_app.bld`, are shown next:



Building with netCentral.bld

This section describes how to build using the `netCentral` build environment.

Be aware that if you are *not* using `netCentral.bld`, you can no longer execute `bsp.bld`, `image.bld`, or `rom.bld` directly from MULTI 2000.

▶ **To build the library, BSP, and examples for the ns9750_a platform:**

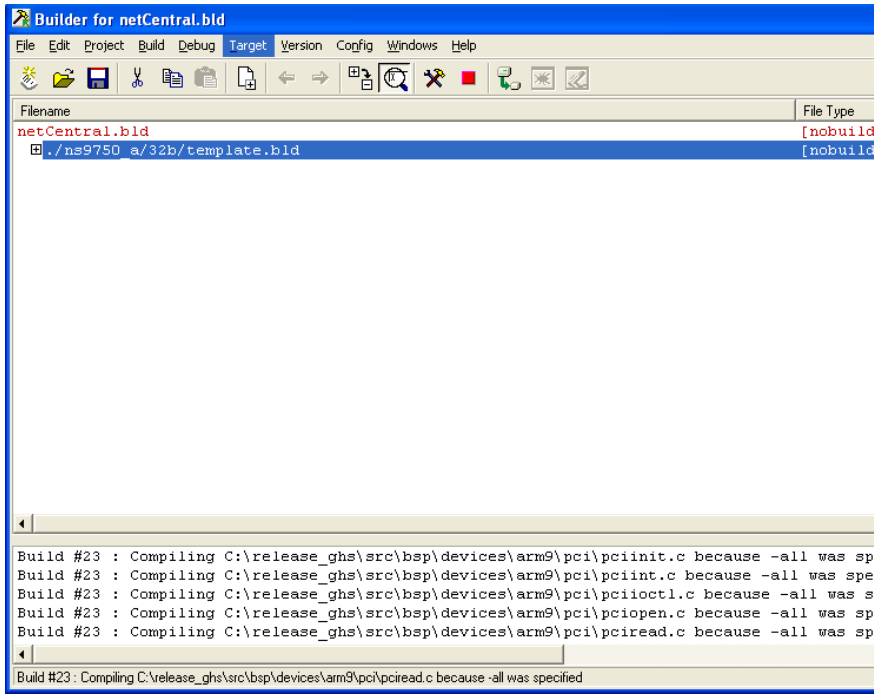
1. Open Green Hills MULTI 2000 v3.6.1.
2. Open `netCentral.bld`.

3 Double-click your platform:

`./ns9750_a/32b/template.bld`

4 Select **Build → Rebuild All.**

You see the build take place, as shown here:



When the build completes, you have built the BSP, libraries, and all the sample applications. You can then download and run an example as described in Chapter 1 of this document, "Building and Debugging a Sample Application: Green Hills."

Building a single application

After you build the `ns9750_a` platform, you can build an individual application by selecting the application and selecting **Build**, as shown in this example of building the `cpptest` application.

To navigate to the `cpptest` application from the ns9750_a platform:

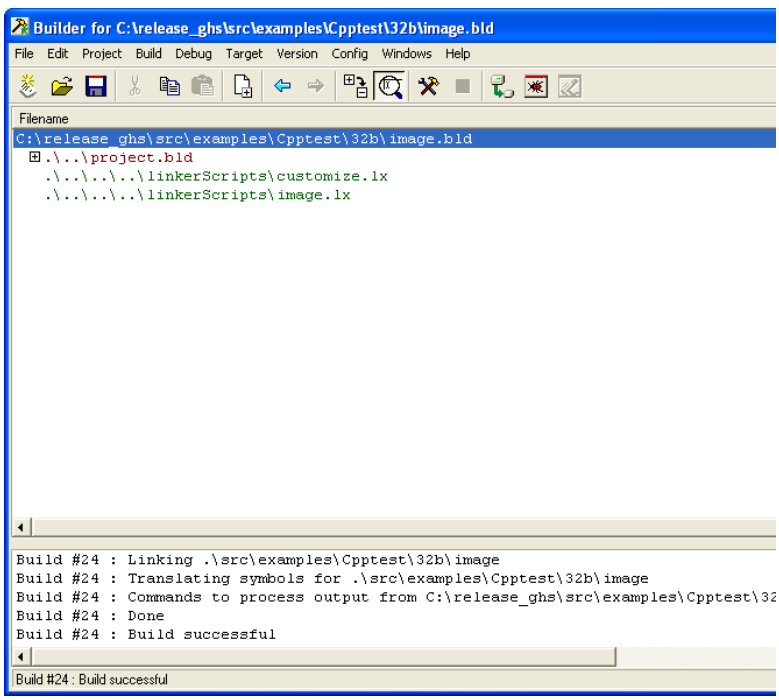
1 Click the icon to the left of `system.bld`.

2 Click `application.bld` → `standard_app.bld`.

3 Double-click `cpptest` → `32b` → `image.bld`.

4 Select **Build** → **Rebuild All**.

You see this in the window:



Adding a new application

▶ **To add a new application:**

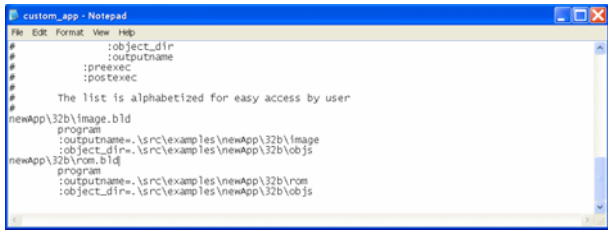
1 Using Windows Explorer, copy an existing application directory and its entire contents to use as a template for your new application.

For example, copy `c:\netos61_ghs\src\examples\Cpptest` to `c:\netos61_gh\src\examples\newApp`.

2 Add your source files to your new applications directory.

3 Using any text editor, open `project.bld`, from your new application directory, `c:\netos61_ghs\src\examples\newApp`. Then add the libraries and source files needed for your application.

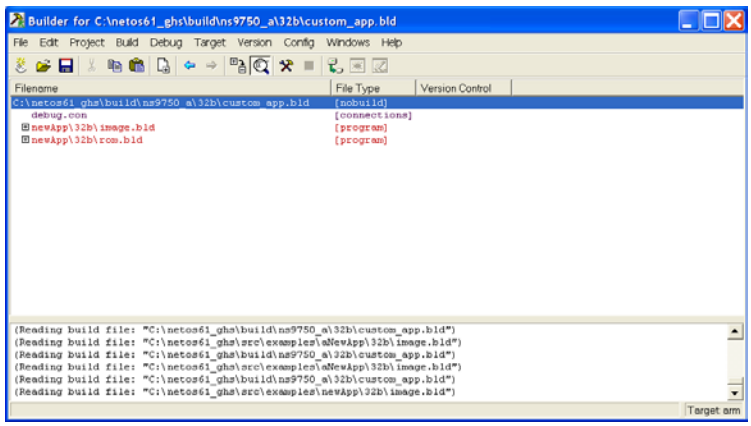
4 Using your text editor, add your new build files to C:\`netos61_ghs\build\ns9750_a\32b\custom_app.bld`:



Your application is now added (that is, linked) into the central build.

5 Reload `netCentral.bld` and navigate to your new application.

6 Double-click either `image.bld` (for a debug image) or `rom.bld` (for a ROM image), as shown here:



7 In the **newApp** window, select **Build → Build image**.

When the build finishes, you can run your application.

Adding a custom BSP

▶ **To insert a new BSP platform, new Platform, into the central build system:**

1. Verify that your new BSP directory is created for the new platform in `netos61_ghs\src\bsp\platform\newPlatform`.

   If the directory does not exist, see the *NET+Works with Green Hills BSP Porting Guide* for information about creating a BSP for a new platform.

2. Create a new directory for the platform's build environment in this location:

   `mkdir \netos61_ghs\build\newPlatform`.

3. Add the template build file (`template.bld`) to the platform's build environment directory by copying the entire contents of an existing platform (ns9750_a) and to the new platform `newPlatform`.

4. Modify the template build files (`netos61_ghs\build\newPlatform\template.bld`) with your editor:

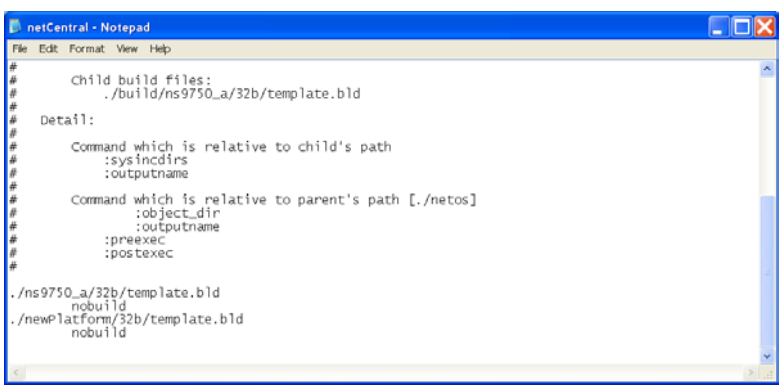
   - Define the name of the new platform:

     ```
     :defines=BSP_PLATFORM=newPlatform
     :sourcedirs=..\..\..\platforms\newPlatform
     :sourcedirs=..\..\..\src\bsp\platforms\newPlatform
     ```

   - Configure the build options, if necessary. These options include cpu, endian, optimization, warning, debug, defines.

5. Link the template build file for the new platform to the central build system:

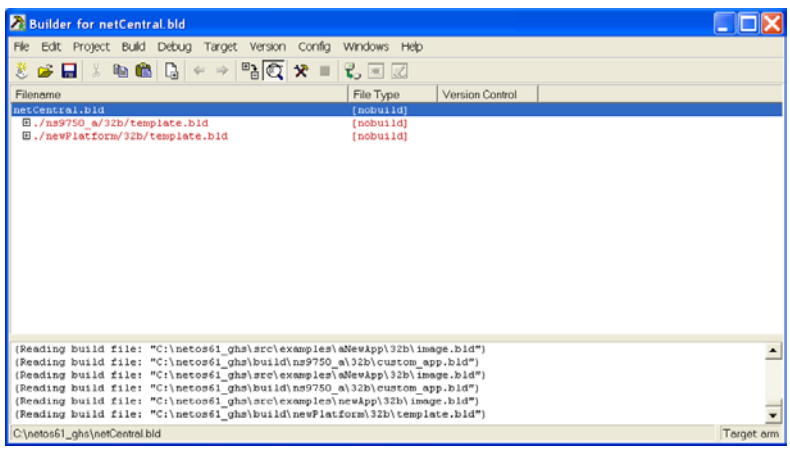
   - Edit `.\netos\netCentral.bld`.
   - Insert the new platform entry, `.\newPlatform\32b\template.bld`, as shown:



6 Modify `\netos61_ghs\src\bsp\platforms\newPlatform\32b\template.bld` in your new platform directory to specify the new platform path.

7 Modify `\netos61_ghs\src\bsp\platforms\newPlatform\32b\template.bld` to specify your new platform directory in the preexec commands.

Now when you reload `netCentral.bld`, this is what you see:



Setting options

In the central build system, all options such as the CPU type, endianness, debug level, and optimization settings are centralized. As a result, each build file no longer needs to define these options.

A child build file inherits the options set of the parent, providingflexibility and easier maintenance for new features in future platforms. All options are centralized in the `template.bld` build file. You can override the options in the other sub-build files such as `platform.bld`, `library.bld`, or `application.bld`.

These options are defined in `template.bld`:

- Platform
- CPU type
- Endian
- Warnings

- Optimizations
- Debug
- Define flags

Platform

This option defines the base options used in `template.bld`. The defined source path directs MULTI 2000 to the correct BSP directory for a specific platform.

```
:defines=BSP_PLATFORM="ns9750_a"
:sourcedirs=../../../platforms/ns9750_a
:sourcedirs=../../../src/bsp/platforms/ns9750_a
```

Warnings

This option controls the assembler code warnings generated by the Green Hills compiler. Currently, all assembler code warnings are disabled.

```
:asm_mode=silent
```

Optimization

This option optimizes the code that the Green Hills compiler generates. You can optimize code for either performance or size. Currently, optimization is disabled at the top level.

Files that require optimization should set this option in the appropriate build file to one of these values:

- `optomizestrategy=none` – no optimization
- `optomizestrategy=speed` – optimized for speed
- `optomizestrategy=space` – optimized for size

Debug

This option defines the debug level. Debug information can be either disabled or generated for MULTI 2000 during compile time. Currently, the system is enabled for debugging.

```
:debuglevel=none - debugging off
:debuglevel=multi - debugging on
```

Define flags

This option defines the variables used in the system.

If you want to set up defines that could be used as compiler directives (for example, to be able to include or exclude debug printing), define them here.

Adding paths

Depending on the command used, path definitions are referenced either from the location of the current build file (child) or the main build file (parent). In the central build system, `netCentral.bld` is the parent build file.

These rules define the use of paths:

- Commands that are relative to the directory of the current build file (./)
  - `:sysincdirs`
  - `:sourcedirs`
  - `:outputname`
- Commands that are relative to the directory of the parent build file (`./ netCentral.bld`):
  - `:object_dir`
  - `:outputname`
  - `:preexec`
  - `:postexec`

Example: relative to the local build file

This include header path is defined for the library build file located in `./netos/build`:

```
:sysincdirs=../h/threadx
```

Note that the path is referenced from the current location of the build file. The path is required to move one level up by using `../` before it can access the `src` directory.

Example: relative to the parent build file

This path is defined for the image build file in the `./netos/src/examples/napara` directory.

Note that the path is referenced at `./src`, which is the location of the `netCentral.bld` parent build file.

Directory path

When you add source files to the system, make sure the build system contains the directory path of the file.

The path searches the source to be built in MULTI 2000. You configure the directory paths with these files:

- `Library.bld`
- `Platform.bld`
- `Application.bld`

File hooks

When you add or remove entries in a specific section of the build system, you need to modify these file hooks:

- library:
  - `standard_lib.bld`
  - `standard_dgb_lib.bld`
  - `custom_bsp_bld`

- platform:
  - `standard_bsp.bld`
  - `standard_dbg_bsp.bld`
  - `custom_bsp.bld`
- application:
  - `standard_app.bld`
  - `custom_app.bld`

PN:(1P) 93000519 C