Configuring Code Composer Studio for OMAP Debugging

Harry Thompson *Software Development Systems/Customer Support*

ABSTRACT

The OMAP™ Code Composer Studio™ (CCStudio) Integrated Development Environment (IDE) provides debug support for the OMAP platform via heterogeneous debugging of the TMS470™ ARM and TMS320C5000™-based DSP subsystem cores, which are connected on the same JTAG scan path within the device. Simultaneous debug of two or more CPUs, sometimes referred to as co-emulation, allows the user to coordinate debugging between both of the processors. The Parallel Debug Manager (PDM) control within Code Composer Studio IDE can be used to execute the targets in parallel, i.e., stepping or running. It can also be used to configure global breakpoints. This feature allows you to designate both processors to halt when either processor halts. Global breakpoints and co-emulation are very useful when trying to debug code that provides communication or signaling between processors.

This application report describes how to setup Code Composer Studio using the heterogeneous device driver, and how to start debugging an OMAP platform using the Parallel Debug Manager. Although some of the values will be different for other OMAP platforms, the process contained in this application report can be used as a framework for those devices.

Contents

1 Installing Code Composer Studio IDE v2.0

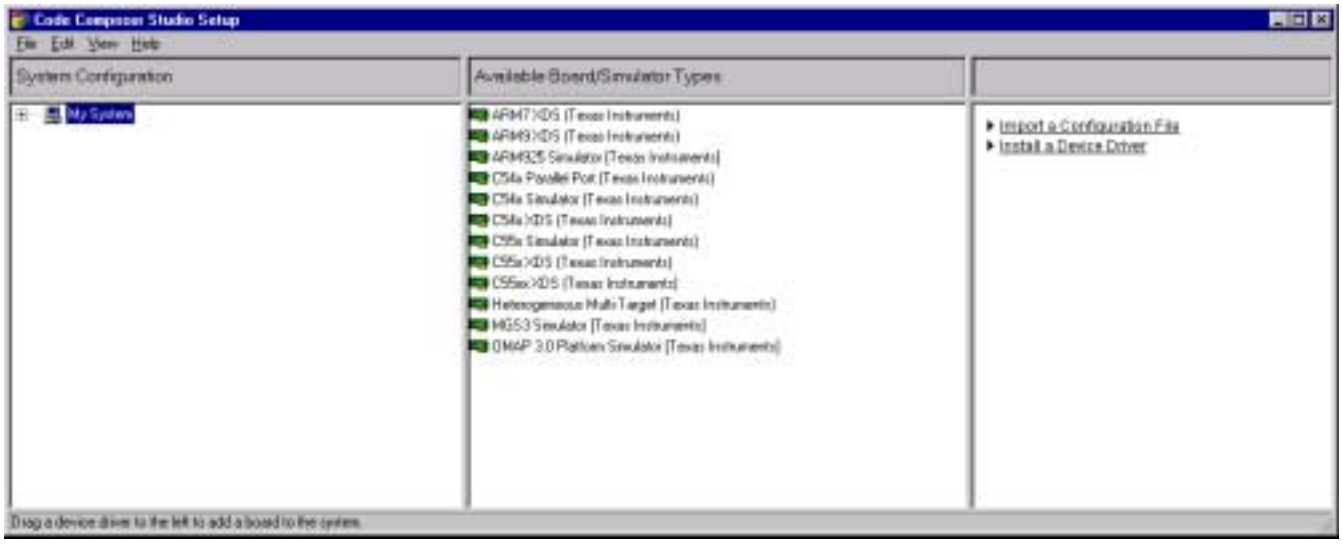
You must install OMAP Code Composer Studio IDE v2.0, which contains support for both the TMS470 ARM and C5000™ processors, before debugging the OMAP platform.

OMAP CCStudio may not contain a predefined system configuration file for the OMAP platform being used, so the configuration file can be created from an existing configuration file or created from scratch. It can then be exported (saved) and then imported each time Code Composer Studio Setup is invoked to create an OMAP system configuration.

1.1 Creating a Configuration File for an OMAP via an Existing System Configuration File

An existing system configuration file may be modified for the OMAP platform you are using. To do this, you should select the existing system configuration file that most closely matches your OMAP platform. The following example uses an existing configuration, HelenDC EVM, which is compatible with an OMAP1510 EVM. For other OMAP platforms, if a system configuration file is not included the process described in this section, it may be modified to your configuration. The steps to do this are:

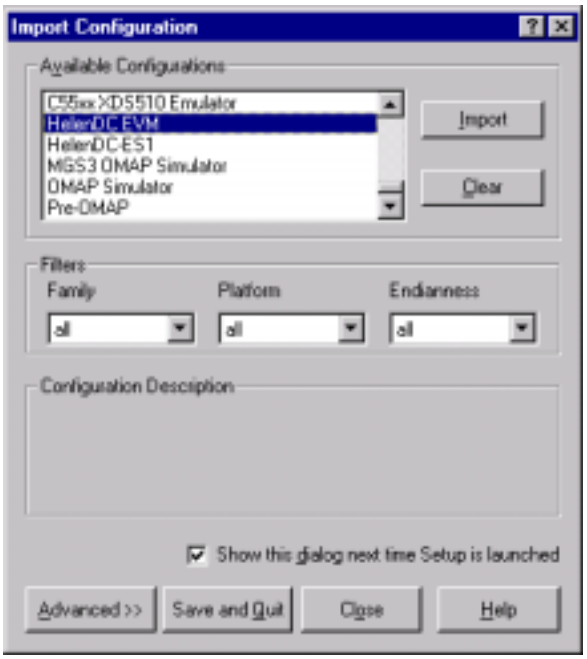
1. Install the XDS emulator and connect the host JTAG cable to the heterogeneous target.
2. Start Code Composer Studio Setup by double–clicking the Setup CCStudio desktop icon.
3. If the **Import Configuration** dialog box does not appear, click on **Import a Configuration File** in the rightmost pane of CCStudio Setup to open it.



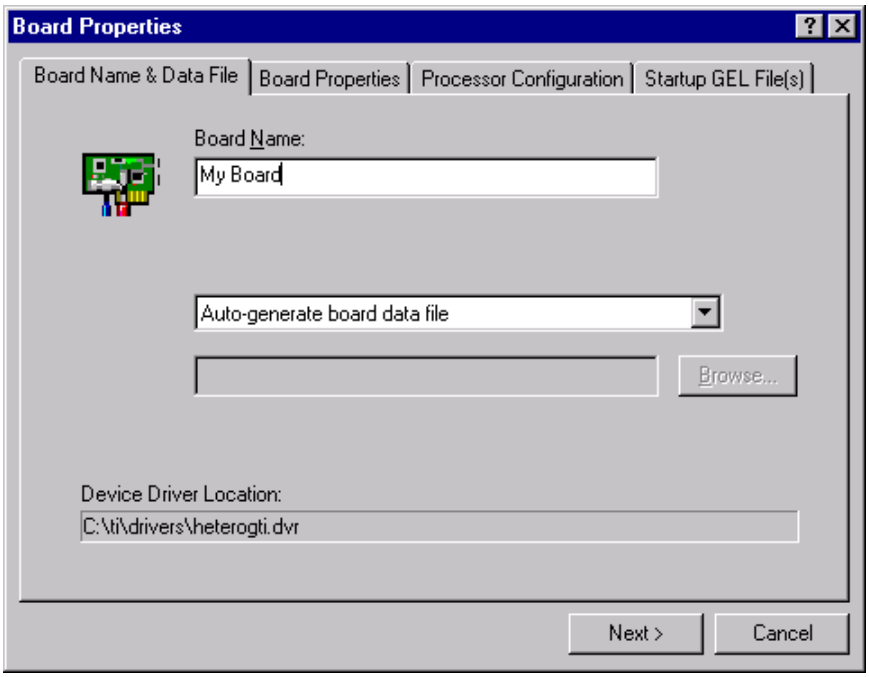
4. In the **Import Configuration** dialog box, click **Clear** to clear the System Configuration pane. Click **yes** when prompted, if you want to clear your system configuration, and then click on **Close** to close the dialog box.

C5000 is a trademark of Texas Instruments.

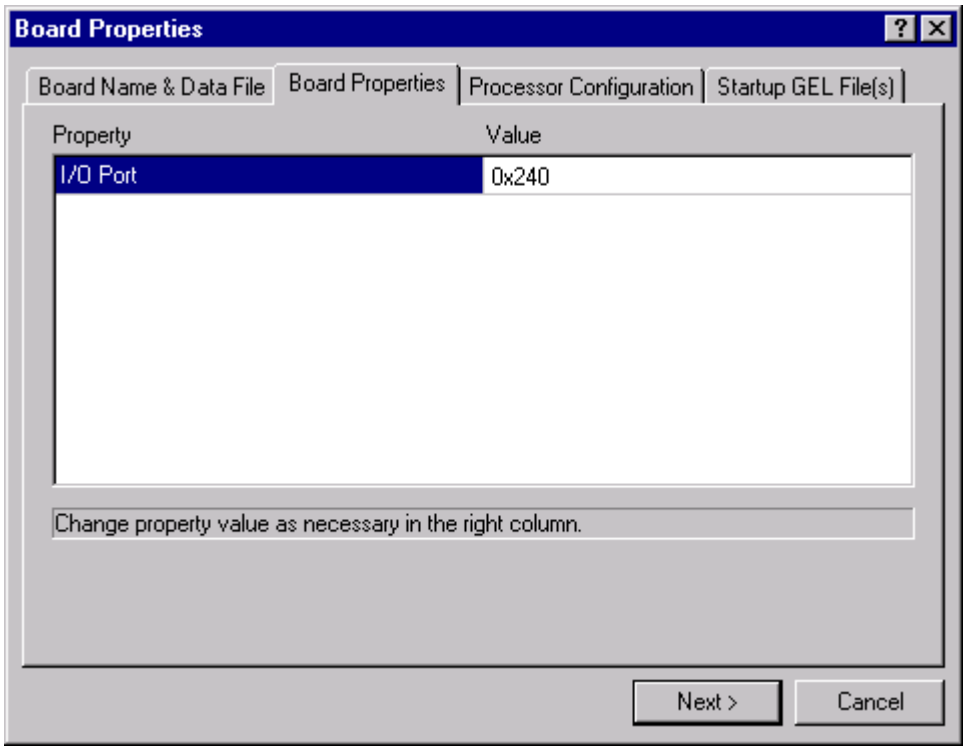
5. In the **Available Configurations** box, scroll down and select the **HelenDC EVM** configuration, and then click **Import**. Click **Close** to close the **Import Configuration** dialog box.



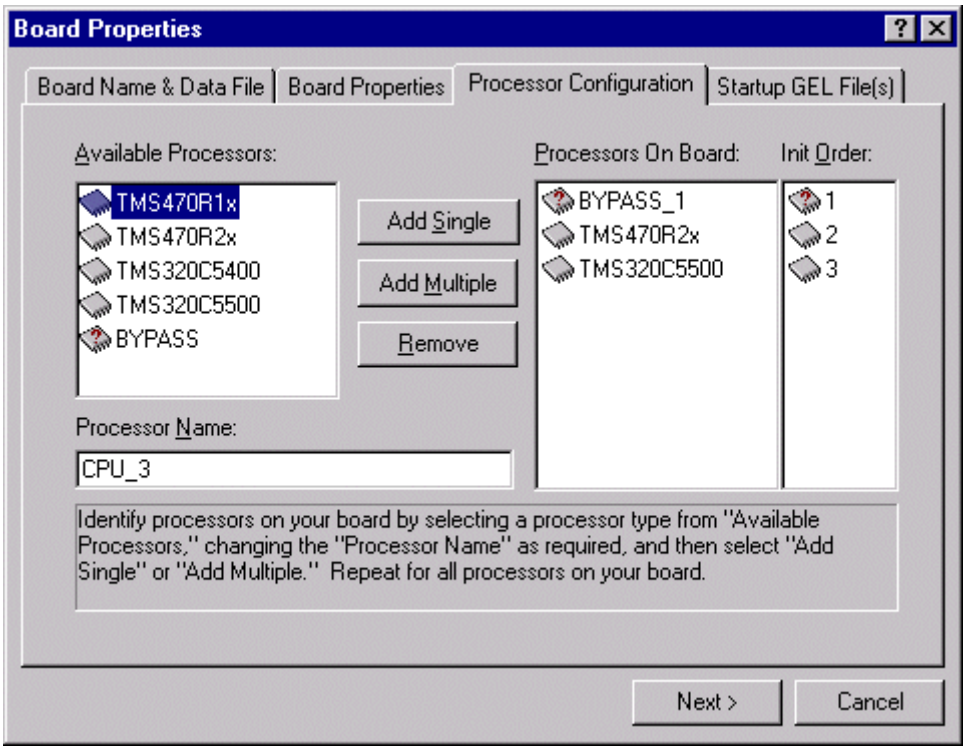
6. In the **System Configuration** pane, right-click on **HelenDC EVM** and select **Properties**.
7. The **Board Properties** dialog box, which contains 4 tabs, should appear with the first tab, **Board Name & Data File** active. Modify **Board Name** in the first tab to a name you wish to use for your board, and click on **Next >**.



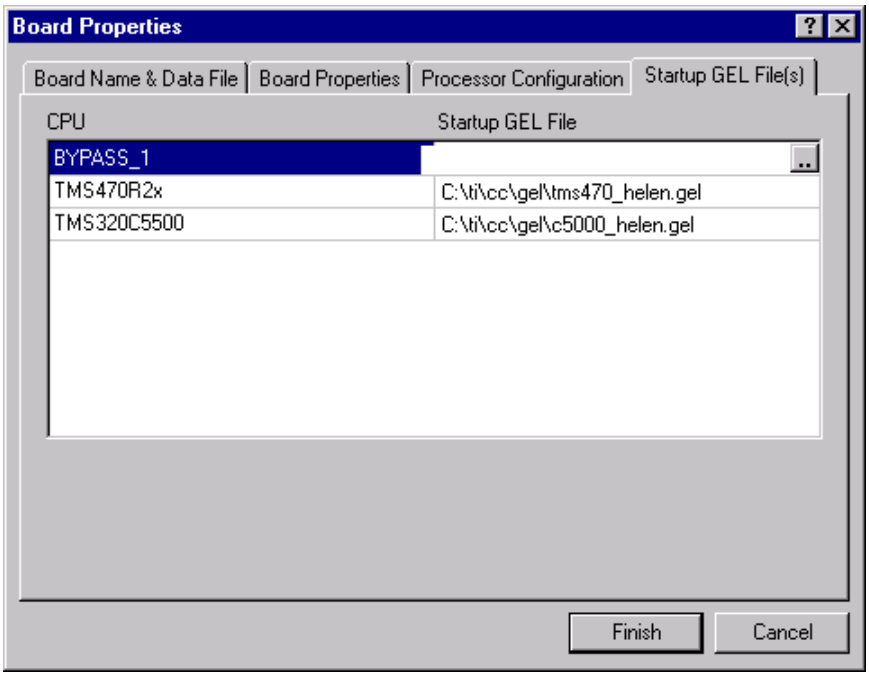
8. Modify I/O Port to the address of the I/O port being used by the XDS510 (in most cases this value is 0x240), if appropriate, and click on **Next>**.



9. The **Processor Configuration** tab should now be active and display the drivers needed for your OMAP platform. The configuration selected contains the JTAG Test Access Ports (TAPs), identified as the “Processors On Board:”, that must be identified for the OMAP. Note that the TAPs must be selected in the order they appear on the scan path (from TDI to TDO). All OMAP processors contain an Enhanced TAP Linking Module, ETLM, which appears first on the scan chain and is bypassed. The second and third TAPs on the OMAP correspond to the TMS470 ARM and the C5000-based DSP subsystem core, respectively.



10. Click on **Next >** to proceed to the **Startup GEL File(s)** tab.
11. A GEL file must be specified for the TMS470 ARM and C5000-based DSP subsystem processors. In this example, the default GEL files, tms470_helen.gel and c5000_helen.gel, can be used. The GEL commands in the GEL_Startup function are executed when CCStudio loads the designated GEL startup file. The associated debug window loads the file when the CPU associated with that window is initialized. On multi-core setups, the file will be loaded as each core is initialized. Startup files and the GEL_Startup function can be used to initialize the associated processor and setup the debug window. See the Code Composer Studio online help for more information about startup GEL files. (Select Help→ Contents→ Using CCS IDE→ General Extension Language (GEL)→ Auto–executing GEL Functions at Startup.)



12. Click **Finish** to close the Board Properties dialog and end driver setup.
13. Save the custom configuration by highlighting the board name in the System Configuration pane, and then select **File–>Export**. An export dialog box will appear; enter a filename that the configuration should be saved as using the file type ".ccs" and click **Save**. If you delete your system configuration in the future and then wish to reload the configuration, press Advanced>> in the Import Configuration dialog box, and then select the .ccs file you saved as the file to import.
14. Select **File**→**Exit**, then click Yes to save changes to the System Configuration and exit the setup program. By default, Code Composer Studio is automatically started.

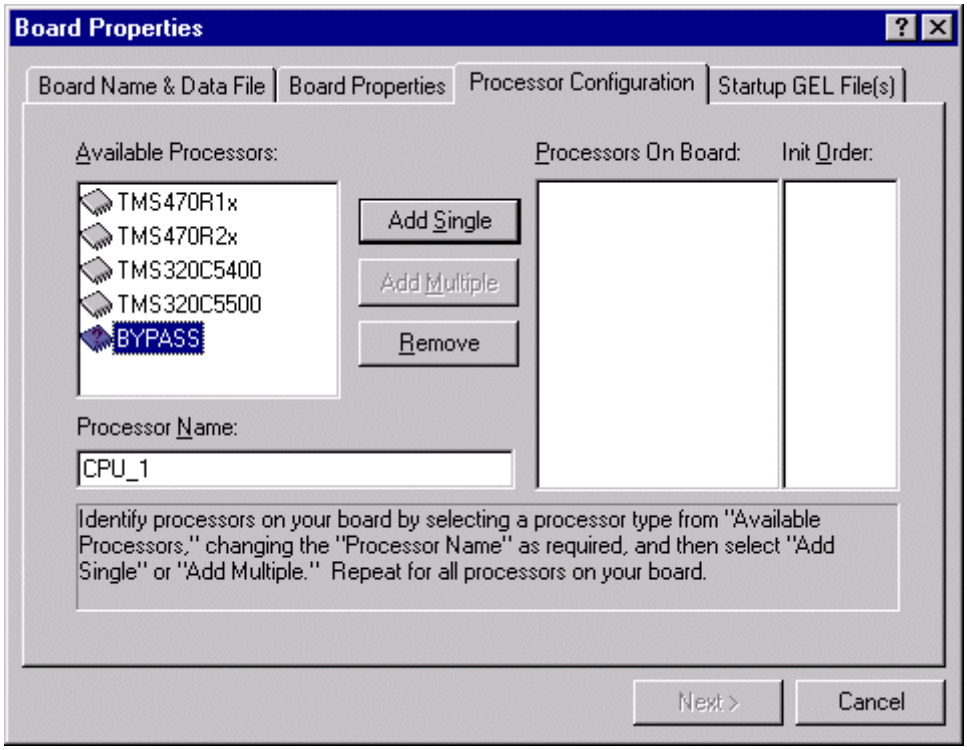
1.2 Creating a Configuration File for an OMAP From Scratch

If a predefined system configuration file does not exist for your OMAP platform, you may create a configuration file manually. After creating the configuration file you can export (save) the configuration file, and then import the configuration file each time Code Composer Studio Setup is invoked, to create an OMAP system configuration for that platform.

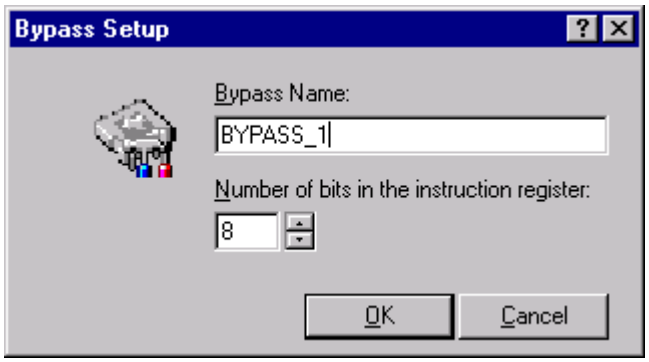
If a third-party emulator is being used, the third-party vendor should provide a driver which is compatible with CCStudio, and provide instructions on creating a configuration file. The following instructions, using the vendor's driver, can be used as a guide if those instructions are not available. Also note that, for parallel port drivers, the vendors instructions regarding installation and parallel port characteristics must be closely followed.

1. Install the XDS emulator and connect the host JTAG cable to the heterogeneous target.
2. Start Code Composer Studio Setup by double–clicking the Setup CCS desktop icon.
3. If the **Import Configuration** dialog box does not appear, click on **Import a Configuration File** in the rightmost pane of CCS Setup to open it.

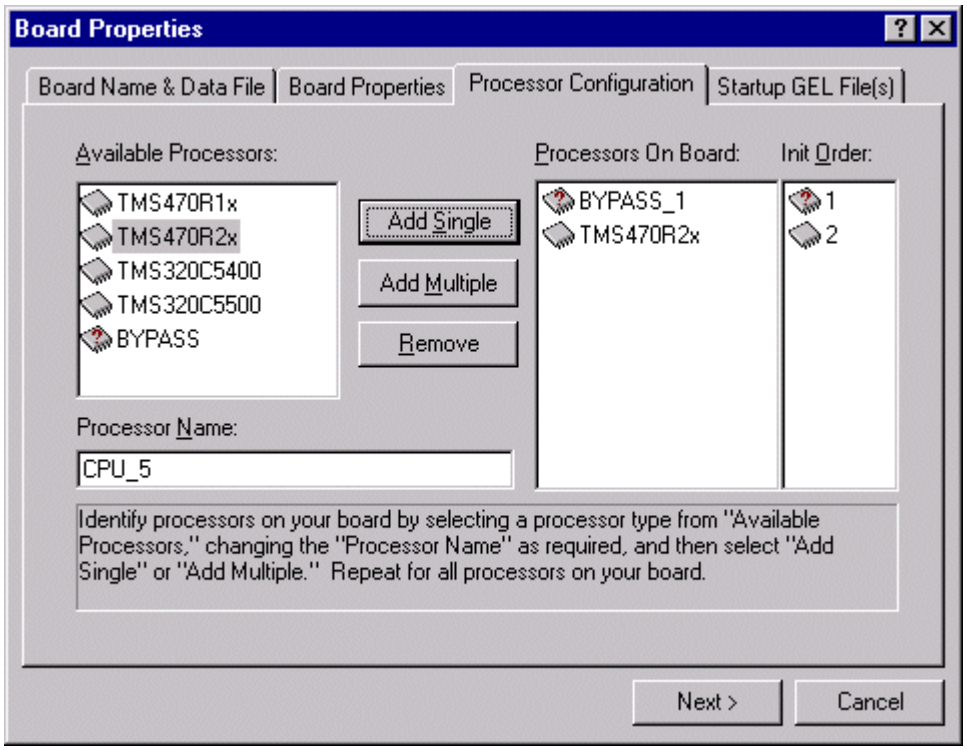
4. In the **Import Configuration** dialog box, click **Clear** to clear the System Configuration pane. Click **yes** when prompted if you want to clear your system configuration, and then click on **Close** to close the dialog box.
5. Drag the **Heterogeneous Multi-Target** driver from the **Available Board/Simulator Types** pane to the **System Configuration** pane.
6. The **Board Properties** dialog box, which contains 4 tabs, should appear with the first tab, **Board Name & Data File** active. Modify **Board Name** in the first tab to a name you wish to use for your board, and press **Next >.**
7. Modify I/O Port to the address of the I/O port being used by the XDS510 (in most cases this value is 0x240), and press **Next>**.
8. The **Processor Configuration** tab should now be active. The heterogeneous device driver automatically queries your installation to detect all of the XDS510 capable drivers you have installed on your system. An aggregate list of all of the CPU types supported by these drivers appears in the **Available Processors** list on the **Processor Configuration** tab.



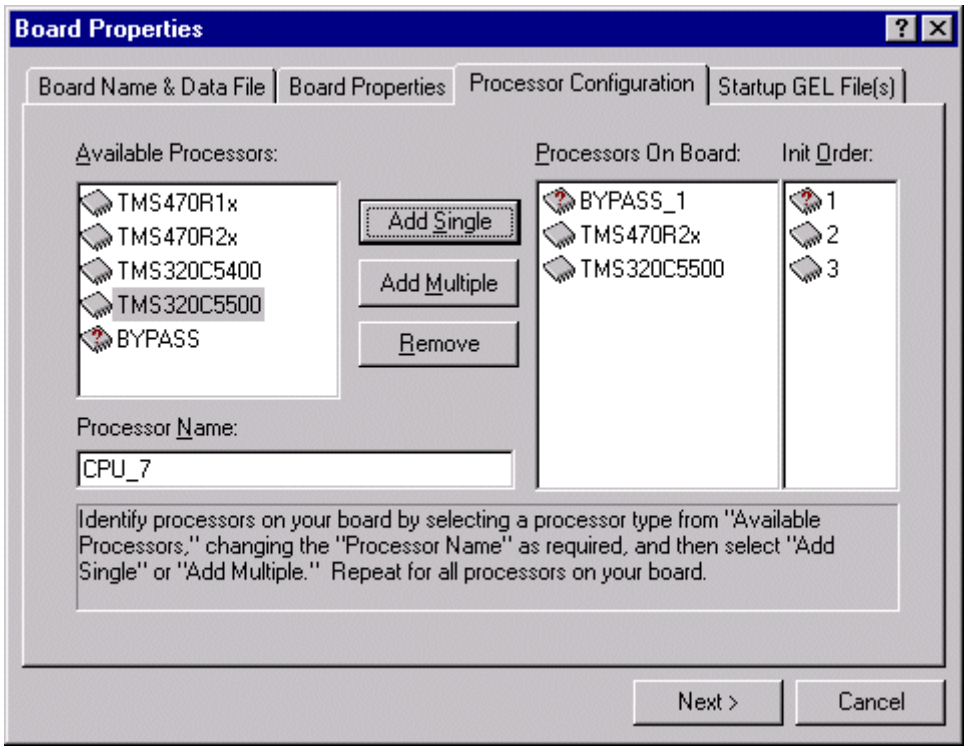
9. All OMAP processors contain an Enhanced TAP Linking Module, ETLM, which appears first on the scan chain. This should be bypassed by selecting **BYPASS** from **Available Processors:** and clicking **Add Single**. Verify that the number of bits in the instruction register is set to 8, and click **OK**.



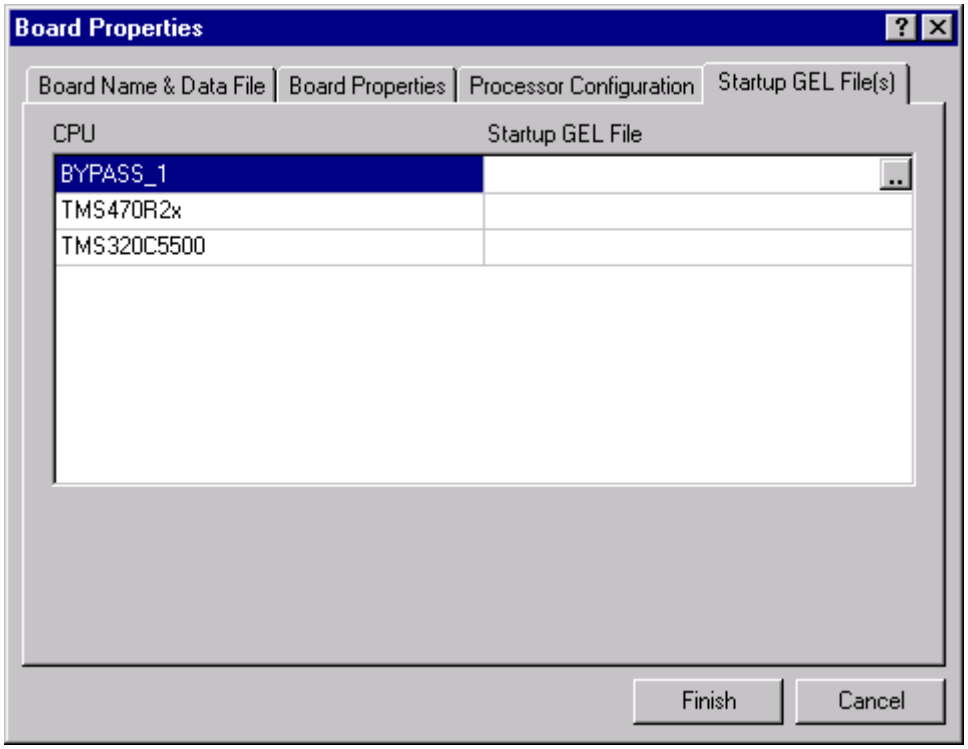
10. The second processor on the scan chain will be a TMS470 ARM. In this example, an TMS470 ARM925 is selected by selecting **TMS470R2x** from **Available Processors**, changing the **Processor Name** to TMS470R2x, and clicking **Add Single**. The **Board Properties** screen should now look like:



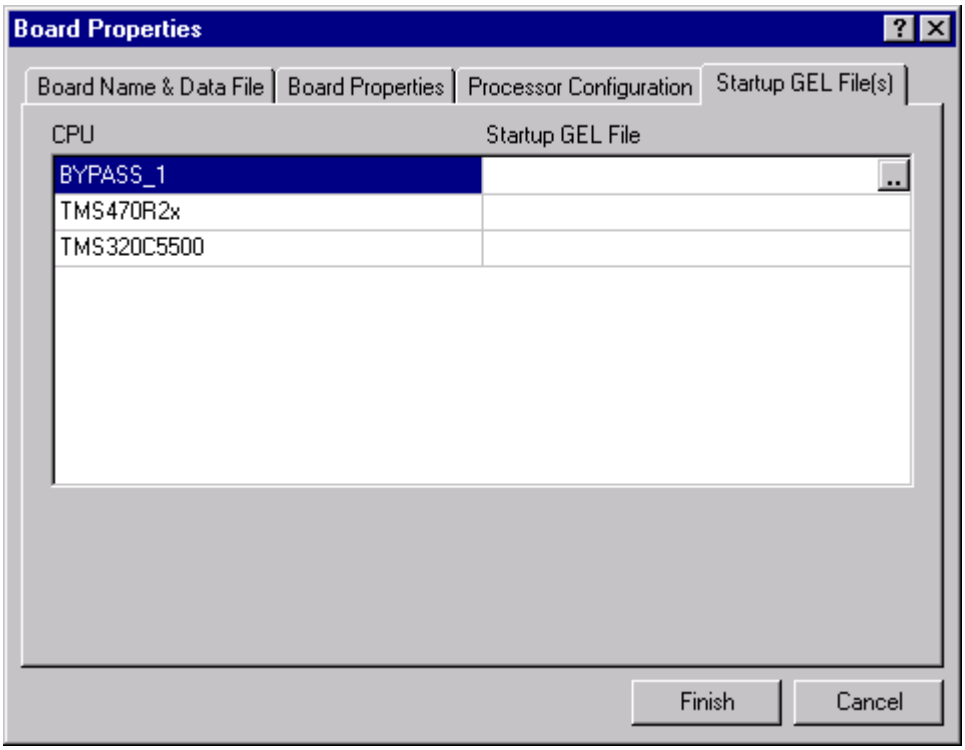
11. The third processor in the scan chain is the C5000-based DSP subsystem. In this example, a TMS320C5500 is selected by selecting **TMS320C5500** from **Available Processors**, changing the **Processor Name** to TMS320C5500, and clicking **Add Single**. The **Board Properties** screen should now look like:



12. Click on **Next>** to proceed to the **Startup GEL File(s)** tab.



13. A GEL file must be specified for the TMS470 ARM and C5000-based DSP subsystem processors. Click on the processor you wish to specify a GEL file for, and then click on the "**..**" box under **Startup GEL File** to display a list of GEL files. You must pick a GEL file which contains the appropriate startup commands for that processor you selected. Section 3 of this application report contains information on requirements regarding heterogeneous debugging, and taking one target out of reset. For this example, the GEL files tms470_helen.gel and c5000_helen.gel were selected. The GEL commands in a startup file are executed whenever a Code Composer Studio debug window is opened. Startup files can be used to initialize the associated processor and to setup the debug window. See the Code Composer Studio online help for more information about startup GEL files. (Select Help→Contents→Using CCS IDE→General Extension Language (GEL)→Auto–executing GEL Functions at Startup.)

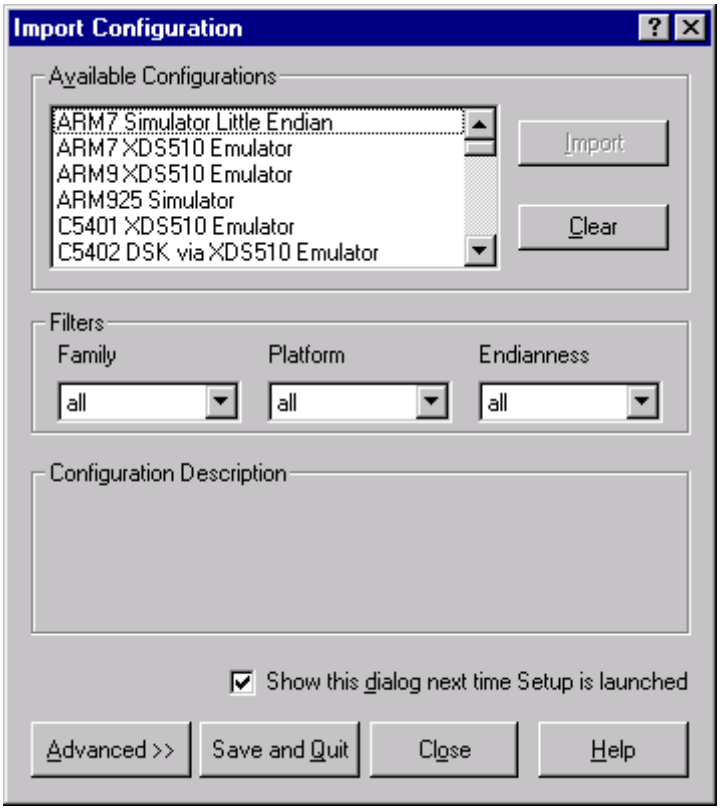


14. Click **Finish** to close the Board Properties dialog and end driver setup.
15. Save the custom configuration by highlighting the board name in the System Configuration pane, and then select **File–>Export**. An export dialog box will appear; enter a filename. The configuration should be saved by using the file type ".ccs" and clicking **Save**.
16. Select **File**→**Exit**, then click **Yes** to save changes to the System Configuration, and exit the setup program. By default, Code Composer Studio is automatically started.

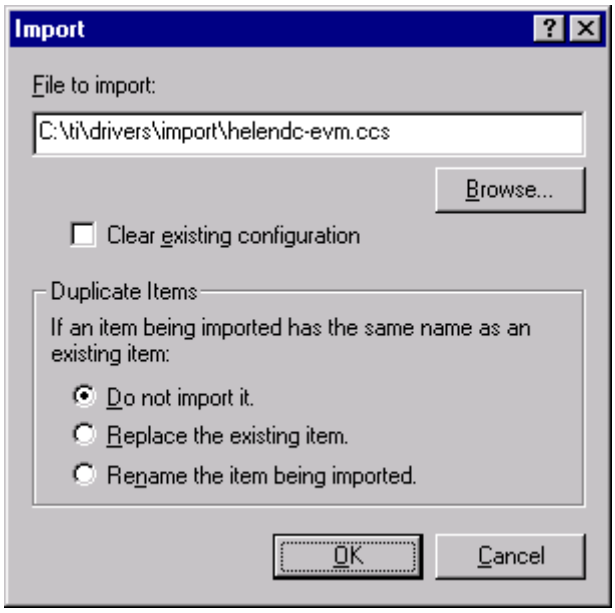
1.3 Restoring a Previously Created Configuration File

If you delete your system configuration in the future which was saved (see item 15 in section 1.2), and then wish to reload the configuration, the configuration can be restored via the following steps. Press Advanced>> in the Import Configuration dialog box, and then select the .ccs file you saved as the file to import.

1. Start Code Composer Studio Setup by double–clicking the Setup CCS desktop icon.
2. If the **Import Configuration** dialog box does not appear, click on **Import a Configuration File** in the rightmost pane of CCS Setup to open it.
3. In the **Import Configuration** dialog box, click **Clear** to clear the System Configuration pane. Click **yes** when prompted if you want to clear your system configuration, and then click on **Close** to close the dialog box.
4. Restore the configuration file by selecting **File–>Import** to open the **Import Configuration** dialog box, and then clicking on **Advanced>>** to open the **Import** dialog box.



5. Enter the name, including the path of the saved configuration file, and press **OK.**



2 Starting Multiple Debug Sessions

1. Start Code Composer Studio by double-clicking the CCS desktop icon. The Parallel Debug Manager (PDM) control is displayed.

2. From the PDM menu bar, select **Open**. Open a Code Composer Studio debug session for each CPU listed on the Open menu.
3. Each CPU can now be debugged independently through its associated debug session.
4. The Code Composer Studio online help explains the proper use of PDM for coordinating simultaneous debugging between multiple targets. (Select Help→Contents→Using CCS IDE→Parallel Debug Manager (PDM)→Debugging Multiple Processors.)

3 Initializing Devices Where One Target is Held in Reset by Another

The OMAP is configured with the TMS470 ARM core holding the C5000-based DSP subsystem core in reset at power up. The DSP emulator will not initialize while the CPU is in reset. When CCStudio is started, the GEL Startup function in the GEL file for the TMS470 releases the C5000-based DSP from reset by setting bit 1 of address 0xFFFECE10 in the TMS470. This is done before the DSP emulator attempts to initialize an emulation session on the DSP. This allows a heterogeneous emulation session to be started on the device. For the OMAP, no other action is required by the user to release the C5000-based DSP subsystem.

For more general information on heterogeneous debugging, refer to *Configuring Code Composer Studio for Heterogeneous Debugging* (SPRA752).

The TMS470 ARM GEL file contains the command, GEL_MapAddStr, which defines the TMS470 ARM memory map. The parameters for the GEL_MapAddStr command are:

- Starting address
- Page (set to 0 for the TMS470 ARM)
- Length
- Memory attributes: R-read, W-write, AS4-access size is 32 bits, AS2-access size is 16 bits.
- Waitstate (set to 0)

The memory map defined by the TMS470 ARM GEL file is:

```
  /* User Strata Flash on CS3 – 16MB */
  GEL_MapAddStr(0x0C000000, 0, 0x01000000, ”R|AS4”, 0);
  /* Boot Flash or RAM on CS0 – 32MB */
  GEL_MapAddStr(0x00000000, 0, 0x02000000, ”R|W|AS4”, 0);
  /* SRAM on CS0 – 512KB */
  GEL_MapAddStr(0x00400000, 0, 0x00080000, ”R|W|AS4”, 0);
  /* SDRAM on CS4 – 8MB */
  GEL_MapAddStr(0x10000000, 0, 0x00800000, ”R|W|AS4”, 0);
  /* Internal RAM on CS6 – 512KB */
  GEL_MapAddStr(0x20000000, 0, 0x00080000, ”R|W|AS4”, 0);
    /* WDT registers */
  GEL_MapAddStr(0xFFFEC800, 0, 0x00000100, ”R|W|AS2”, 0);
  /* CLKM registers */
  GEL_MapAddStr(0xFFFECE00, 0, 0x00000100, ”R|W|AS2”, 0);
  /* DPLL1 registers */
  GEL_MapAddStr(0xFFFECF00, 0, 0x00000100, ”R|W|AS2”, 0);
  /* EMIF registers */
  GEL_MapAddStr(0xFFFECC00, 0, 0x00000100, ”R|W|AS4”, 0);
  /* Interrupt Handler level 1 registers */
  GEL_MapAddStr(0xFFFECB00, 0, 0x00000100, ”R|W|AS4”, 0);
  /* System DMA registers */
  GEL_MapAddStr(0xFFFED800, 0, 0x00000800, ”R|W|AS2”, 0);
```

For a TMS470 containing a TMS470 ARM9, the GEL file also initializes the memory interface with the commands:

```
/* Configure ARM9 Memory Interface */
  (*(int*)0xFFFECC10) = 0x00203339;  /* EMIFS (nCS0) configuration */
  (*(int*)0xFFFECC14) = 0x00001139;  /* EMIFS (nCS1) configuration */
  (*(int*)0xFFFECC18) = 0x00001139;  /* EMIFS (nCS2) configuration */
  (*(int*)0xFFFECC1C) = 0x00001139;  /* EMIFS (nCS3) configuration */
  (*(int*)0xFFFECC20) = 0x00010074;  /* EMIFF (nCS4) configuration */
  (*(int*)0xFFFECC24) = 0x00000037;  /* MRS (nCS4) initialization  */
```

and disables the TMS470 ARM9 watchdog timer with the commands:

```
/* Disable ARM9 Watchdog Timer */
  (*(short*)0xFFFEC808) = 0x00F5;
  (*(short*)0xFFFEC808) = 0x00A0;
```

The following commands are executed in the GEL file to release the C5000-based DSP subsystem from reset:

```
/* Get the api ram out of host only mode. */
  (*(short*)0xFFFEC91c) = 0x0000;
/* The default value for ARM_IDLECT2 = 0x0100.        */
/* Set bit 6 of ARM_IDLECT2 to turn on the api clock; */
/* the api must be turned on to take api ram out of   */
/* host only mode                                     */
  (*(short*)0xFFFECE08) = 0x0140;
  /* Release C5500 from reset. */
  (*(short*)0xFFFECE10) = 0x0002;
```

4 Reference

1. *Configuring Code Composer Studio for Heterogeneous Debugging* (SPRA752).

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265