Application Report
SPRA978 − November 2003

TMS320C6000 Boot Mode and Emulation Reset

Dipa Rao *DSP Applications*

ABSTRACT

Emulation-generated target reset, issued via a pull down menu command (Debug → Reset CPU) in Code Composer Studio, causes an internal chip reset. Since this reset is internal, any devices external to the DSP will not be aware of it. The $\overline{\text{RESET}}$ signal, which is an input to the DSP, is not affected by this command. This may cause problems in a system environment where an external host or board-level system logic drives the DSP device configuration and boot pins, and if some type of host boot mode (HPI, PCI, or XBUS) is selected. Texas Instruments C6xx DSP devices have configuration pins that select the boot process after DSP reset. Depending on the boot process selected, asserting target reset through emulation can have undesired results. This document explains these conditions and the precautions that must be taken to avoid problems.

Contents

1 Hardware Reset

$\overline{\text{RESET}}$ is an input pin on the TMS320C6xx devices. This signal forces a DSP system reset that sets internal control registers and output control signals to their default values. Refer to the *TMS320C6000 CPU and Instruction Set Reference Guide* (SPRU189) and TMS320C6000 Peripherals Guides for register default values.

When $\overline{\text{RESET}}$ is deasserted, its rising edge device configuration and/or boot mode pins are read and latched. These pins continue to be read for at least a few cycles after $\overline{\text{RESET}}$ is raised high for synchronization. Please refer to the datasheet of the specific device for detailed timing information. After the boot configuration pins are latched, the appropriate boot process takes place as described in the next section. In host port boot mode, the rising edge of the $\overline{\text{RESET}}$ signal can be used to trigger the host to start the boot process.

The target will not react to the $\overline{\text{RESET}}$ being asserted from the external system, if the emulator is plugged in and Code Composer Studio debugger has been invoked, EXCEPT for the boot logic. The following steps will illustrate this behavior.

1. Bring up Code Composer Studio on a 6711 DSK, perform an emulation generated target reset (Debug → Reset CPU), and single step a few addresses to get the PC past 0.
2. Fill memory with a pattern beginning at address 0, press the rest button, and observe that nothing apparently happens.
3. Single step once and observe that lower memory is overwritten by the contents of Flash. The boot process was initiated but delayed because the emulation logic had control. During the single step, the boot logic was able to gain control and overwrite memory.

2 Boot Mode

After device reset, the DSP may boot code from external memory or an external device before beginning execution. The configuration pins that are read at reset determine the DSP boot process and other device configurations.

The C6000 devices are capable of three types of boot modes:

- **No boot:** If this mode is selected, upon completing reset, the CPU directly begins execution from address 0 of memory. Please ensure that the CPU does not execute invalid instructions, as execution of invalid opcodes will result in undefined operations. During development stages, while using the Code Composer Studio debugger, it may be beneficial to hold down the DSP reset button (if available) until the Code Composer Studio splash screen vanishes. This minimizes the amount of time the processor runs unknown and perhaps invalid program code, thus increasing the chances of success in gaining control of the DSP for emulation.
- **Non volatile memory (ROM/Flash) boot:** When the DSP is brought out of reset in this mo de, the CPU core is internally held in reset while the DMA/EDMA peripheral transfers code from external non-volatile memory to address 0 of DSP memory. The amount of data transferred varies depending on the DSP in question. After completing the transfer, the CPU begins execution from address 0.

  Typically, an unprogrammed flash device would result in the DSP boot loading 0xFFFFFFFF, which is decoded as a NOP. This by itself is not a problem; however, when the processor starts executing the instructions it could potentially continue running beyond defined memory, which could lead to problems. This might even put the processor in a state where the emulator cannot be invoked. In order to avoid this condition, it is useful, during development stage, to program the flash with at least a minimal snippet of code to ensure that the DSP emulator can be brought up reliably.

  For example, a simple code snippet that can be put into the flash (at the starting address of flash) is:

```
Infinite_loop:   B    .S1  Infinite_Loop
                 NOP
                 NOP
                 NOP
                 NOP
                 NOP
                 NOP
                 NOP
```

The sample code shows a full- packet of code that would reliably help bring up the debugger.

These instructions translate to the following opcodes:

```
0x00000012   Infinite_loop:  B    .S1 Infinite_loop
0x00000000                          NOP
```

- **Host boot:**When $\overline{\text{RESET}}$ is deasserted while in host boot mode, the CPU core continues to be held in reset. While the core is in reset, the external host can exercise the host interface (HPI, PCI or XBUS), access the DSP internal or external memory, and initialize the DSP internal configuration registers. After all the necessary initializations, the host sets DSPINT (host to DSP interrupt signal) to complete the boot process. This causes the CPU to wake up and start executing code at address 0. This interrupt from the host does not cause the processor to execute an interrupt service routine (ISR), since the core was in reset when it was asserted. However, the CPU must clear the DSPINT bit so that future interrupts from the host can be recognized. If the system is designed such that the host processor actually drives the boot mode pins, please ensure that every time $\overline{\text{RESET}}$ is asserted, the host correctly configures the pins so that they can be appropriately latched.

  On the older C6x0x devices, if host boot is selected, the DSP will wait for the host to initialize the DSP and set DSPINT. If the host relies on the $\overline{\text{RESET}}$ signal to start transfer, then the host could stall indefinitely, because it never sees the $\overline{\text{RESET}}$ signal going active during emulation reset.

  The emulation driver for C621x, C671x and C64xx devices features an internal timeout function (the hardware to support this function is not available on C620x and C670x devices). In host boot mode, the emulator waits for a certain amount of time. When a timeout happens, the emulation logic asserts DSPINT to wake up the CPU from reset. Hence, on the C621x, C671x and C64xx devices, the emulator does not stall while waiting for the DSPINT from the external device.

3 Emulation Generated Target Reset

Emulation-generated target reset, as described here, is the reset scanned into the target from the emulator, and is initiated via a pull down menu in Code Composer Studio (Debug → Reset CPU). This reset may also be performed by issuing a command using the GEL script command GEL_Reset(). It should be noted that emulation reset is not performed on starting the Code Composer Studio debugger (unless it is directed by a GEL file). Also, please note that the emulation generated target reset being referred to in this section, is not the same as physically resetting the emulator that is being used in the development system. Unlike a hardware reset, an emulation reset causes a reset that is confined to the internals of the chip. It is not propagated outside the DSP. The system level hardware external to the DSP is not normally aware that an emulation-generated target reset is taking place. When the CPU is reset from Code Composer Studio, the boot mode pins are read, and the processor will attempt to perform the appropriate boot process, depending on the settings on the boot configuration pins. If the DSP is configured to boot from an external host, then the host will not be aware of the reset. Therefore, if the host does not respond correctly to the DSP emulation reset condition; it will not perform the boot process, which could result in emulator lockup.

Special precautions also need to be taken if the device configuration and boot mode pins rely on an external device (such as FPGA or PLD) to drive them. This is to ensure that valid values will be read at the configuration pins during reset and the DSP does not get into an invalid boot mode.

4 Detecting Emulation Reset

Since the $\overline{\text{RESET}}$ signal cannot be used by the host to detect emulation reset, other methods can be implemented. Some output pins do, or can be made to change states during and after reset.

For example, a timer output pin on the C621x or C671x will be in high-Z state during reset, and driven low (default) after reset. If an external pull up is hooked up to this pin, then this pin will go high during reset and low after reset. This can be used to detect an emulation-generated target reset condition. If this pin is unused and an external pull up is connected, then a rising transition of this pin can be taken as $\overline{\text{RESET}}$ going active, and falling transition as $\overline{\text{RESET}}$ going inactive. In general, any unused output pins in Z-group (pins that go high-Z during reset) can be externally pulled up/down opposite to the default after-reset state to detect an emulation reset. Note that the transition of this signal will be delayed with respect to the $\overline{\text{RESET}}$ signal. Refer to device specific datasheet for pins and reset timing information.

Upon detection of this condition, the host can drive the correct values to the configuration pins if necessary, and can perform the boot process that ends with asserting DSPINT.

5 Reset Emulator

The Reset Emulator function (Debug → Reset Emulator) should not be confused with resetting the target. Reset Emulator initializes the emulation hardware (XDS510 or XDS560) and the JTAG state machine within the target by toggling the /TRST signal. Reset Emulator should be used with caution and only in an attempt to recover from errors. If a Reset Emulator command is used, it should be accompanied by an emulation generated target reset to ensure that the target, emulation driver and emulator are in sync.

6 References

1. *TMS320C6000 CPU and Instruction Set Reference Guide* (SPRU189)
2. *TMS320C620x/C670x DSP Boot Modes and Configuration Reference Guide* (SPRU642)

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265