TMS320 DSP
DESIGNER'S NOTEBOOK

TMS320C40 Emulator Tips

APPLICATION BRIEF: SPRA228

Rosemarie Piedra
Digital Signal Processing Products
Semiconductor Group

Texas Instruments
June 1993

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

TMS320C40 Emulator Tips

Abstract

This document discusses special precautions that need to be taken when working with the TMS320C40 emulator. There are several potential problem situations:

- The debugger will break any pending CPU/DMA access that is not completed within the time-out period.
- Comm port logic stops when the emulator is halting the device (in a breakpoint or between single steps). (However, you can still work with the comm ports under emulator single-stepping.)
- Interrupts are disabled during assembly single-stepping.

Situations where these events could occur are discussed in detail, as well as techniques to avoid problems.

Design Problem

What special precautions need to be taken when working with the TMS320C40 emulator?

Solution

General Issues

- The debugger will break any pending CPU/DMA access that is not completed within a time-out period (1 second) during single-stepping or after an emulator halt. This could happen in the following situations:
  - During DMA/CPU reads/writes from/to comm ports without comm port synchronization. In the case of the DMA, this will even cause the DMA counter to decrement by one.
  - During execution of a large RPTS execution.
  - During interlocked instructions.

    Current debugger versions (2.20 or higher) will send a “processor access time-out adr=xxxx” message if a read or write access doesn’t complete. However, debugger version 2.01 or lower may not send any warning message if a read access is broken. This “incomplete read” can be misinterpreted as an access completion.

    The “adr=xxxx” provided in the message will “approximately” correspond to the address where the time-out occurs when this is the result of a “debugger” access time-out (for example from displaying memory). In the case of a CPU/DMA time-out, what you probably will receive is an “address = 0xaaaababe” that is not meaningful. In this case, the previous three to four instructions to the PC value should give you an indication of where the time-out occurs.
- The ’C40 Debugger (version 2.01 or lower) executes a “double read” when accessing memory locations (including on-chip memory and memory-mapped peripherals). Debugger version 2.20 restricts this problem to memory locations 0x0 to 0x0xfff. If you have external FIFOs mapped into this region, be careful when using any emulator command that displays/reads those locations.
- Remember, when you “quit” a debugger session, the processor will continue halted (by the JTAG circuit) unless a “runf” command has been issued before the “quit” command. Another way to “unhalt” the ’C40 is to issue an “emurst” from the DOS/OS2 command prompt.

- File sharing:

  When using the OS2 emulator with C programs, you cannot edit in another window the C source file that is currently being used for the debugger. The debugger “locks” access to the last five C-source files displayed in the file window during the current debugger session.

  To overcome this limitation without quitting the debugger, you can create an alias “free” command that will release ownership of the previous files:

```
alias free,"file dummy.c;file dummy.c;
   file dummy.c;file dummy.c;file dummy.c"
```

  This is not an issue in the SUN or VAX platforms.

Comm Ports

Comm port logic stops when the emulator is halting the device (in a breakpoint or between single steps). However, you can still work with the comm ports under emulator single-stepping but the following precautions should be taken:

- Single-stepping through comm port transfers:

  If the receiving side is halted by the emulator, the comm port logic will receive one word into the IFIFO and stop after that. Each assembly single-stepping will produce one word being received.

  If the sending side is halted by the emulator, the comm port will send one complete word if you single step one more time after the instruction that writes into the OFIFO. Without this additional single stepping after the “store” to OFIFO instruction, the emulator will break any pending comm port byte transfer and the receiver end could get only one of the four bytes, causing the comm port byte counter to go out of sync.

  Summarizing, for comm port transfer, the following sequence should work:
  step (sender) - step (sender) - step (receiver).

- Don’t display comm port FIFOs on any debugger window. The debugger in fact will issue a read/write from/to the IFIFO/OFIFO and the data will be gone.

- In a multiprocessing distributed-memory system, avoid issuing a “reset” command in an individual debugger window. A ’C40 Debugger “reset” command in fact resets the device and changes the direction of ’C40 comm ports to their status after reset. This could create bus contention with the comm port connected on the other end that could potentially damage ’C40 comm port drivers. The safest way to “reset” the multiprocessing

board without quitting the debugger sessions is to “run free”, do a hardware reset to the entire board, and then “halt”.

Interrupts

- Interrupts are disabled during assembly single-stepping. This feature is used to avoid receiving an interrupt after every single-step with real-time external interrupts. If you wish to take interrupts during single-stepping, use the “run 1" command that is totally equivalent to the single-stepping key except that it doesn’t disable interrupts. Interrupts are always enabled during C single-stepping.
- You can manually reproduce an external interrupt by setting the IIF bits as follows:
  a) select pin as an interrupt pin (FUNCx=1)
  b) select edge trigger interrupt (TYPEx=0):
     Level trigger will not work!
  c) enable external interrupt (EIIOFx=1)
  d) assert external interrupt (FLAGx=1)
  e) enable GIE bit