PCI Timer Card Test Procedure
Tom Brimeyer 5/17/05

- Currently the board is programmed to operate on byte wise memory allocation as opposed to the previous word wise memory allocation.

- The Timer Board can be tested on the NCAR RADAR DRX Computer.

- The following files are needed for testing.

bytewise_eeprom.hex → This file contains the HEX Code to program the EEPROM to assure that the data be read/written byte wise. The older version read/wrote word wise.

* When running timertest with the older EEPROM Code version, the data would only write to every 4th byte.

byteaddr_PIC.asm → This file contains the Assembly level code for the PIC. This code has been updated to work bytewise.

byteaddr_PIC.hex → This file contains the HEX code that was compiled directly from byteaddr_PIC.asm.

*The following folders and files can be found on the NCAR RADAR DRX Computer.

C:\Develop\radar\timertest\timertest.cpp

timertest.cpp → This file contains the C++ code to test the read/write of memory to the PCI RAM. It attempts to write 100 bytes to memory locations 0x40 and up. After the writing process, it then attemps to read the same memory locations and verify that the correct data is in each memory location.

C:\Develop\radar\timerset\timerset.cpp

timerset.cpp → This file contains the C++ code to set the first sequence of the PCI Timer card so that the board outputs the needed pulses.

C:\Develop\radar\timer\timerlib.cpp

timerlib_byte.cpp → This file contains the library definitions/subroutines used for Timertest and Timerset.

* This file must be renamed "timerlib.cpp" in order to be used correctly by Timertest and Timerset.

C:\Develop\radar\timerset\debug\config.tmr

config.tmr → This file contains the configuration for the single sequence to be used for setting/testing the PCI Timer Card. When Timerset is executed, the values from this file are used and correspond to the configuration of the output pulses.

- Software & Hardware needed for testing/changing/updating

Hardware:

EPIC Programmer (MicroEngineering Labs, Inc.) → PIC Programmer Hardware.

EEPROM Prgrammer → ALLPRO-LC (Logical Devices, Inc.)

Software:

Gateway G6-200 Computer

EPIC Programmer (MicroEngineering Labs, Inc.) → EPICWIN.exe is the windows based PIC Programmer.

* Must power PCI Timer Board with PC Power Supply in order to program the PIC.

EEPROM Programmer → ALLPRO_LC for Windows

RTF Processing Developement Computer (Drive F:\mplab tools\MPLAB v6.40)

MPLAB (Microchip) → MPASMWIN.exe is the assembler program. Converts the PIC Assembly Code into HEX Code.

* No known disk, but Can be found on Aditi's old hard drive.

- Testing the PCI Timer Board

1) Test RAM memory allocation.

Execute Timertest.exe and observe memory dump. Make sure that each memory Location passes.

Troubleshooting: If every fourth byte is being written to and read, then update EEPROM with newer bytewise_eeprom.hex.

2) Test PIC Operations.

Execute Timerset.exe and observe memory dump. Make sure that each Interrupt passes without timing out.

Troubleshooting: Upon boot up, make sure that the PIC is loading the boot up sequences into RAM. Obersve memory dump using timertest. should observe the following:

0x00 40 1F CF 85 88 3F 1F FF 85 40 3F 1F FF 03 20 3F
0x10 1F FF 02 14 3F 1F FF 05 00 3F 1F FF 04 F0 3F 1F
0x20 FF 07 A2 3F 1F FF 06 50 3F 1F FF 03 C0 3F 1F FF
0x30 02 31 3F 1F FF 05 70 3F 1F FF 04 60 00 00 00 00

If this is not the case, check PIC clock frequency.

If there is no PIC clock, then PLL or VCO may not be operating. Try adjusting the variable inductor of the VCO until the PLL LED beings to blink and the PIC LED stays lit.

Reboot and begin step 2 again.

If a PCI timeout occurs, check PIC clock as well to see if PIC is operational.

3) Test Pulse Outputs from DB25.

Pulse Outputs 0-5 can be found from the corresponding DB25 pins.
Bpulse0 → Pin 9
Bpulse1 → Pin 22
Bpulse2 → Pin 10
Bpulse3 → Pin 23
Bpulse4 → Pin 11
Bpulse5 → Pin 24

Each test pulse must be measured in reference to a single test pulse.
i.e. When testing pulses1-5, use pulse0 as a reference pulse to compare.

All pulses/pulse values are normalized to 8 MHz.

All test pulse values can be found/changed via the config.tmr file (see files above)

Test Pulse variables:
delay → delay time before pulse begins

Time Delay = delay/8MHz i.e. delay=16 → Time Delay = 16/8MHz = 2us

width → width of pulse

Pulse Width = width/8MHz i.e. width=8 → Pulse Width = 8/8MHz = 1us

period → length of period for all pulses

Period = period/8MHz i.e. period=4000 => Period = 4000/8MHz = 500us