

Avnet FXT Evaluation Board SystemACE Reference Design Using SystemACE Module

**Version 1.0
May 2008**

1 Introduction

This document describes a simple PowerPC based design that illustrates the use of Memec SystemACE Module (SAM) on the Avnet FXT evaluation board.

2 Reference Design Requirements

This reference design will require the following software and hardware setups.

2.1 Software

The software requirements for this reference design are:

- WindowsXP
- Xilinx ISE 10.1 with Service Pack 2
- Xilinx EDK 10.1 with Service Pack 2

2.2 Hardware

The hardware setup for this reference design is:

- Computer with 1 GB RAM and 1 GB virtual memory (recommended)
- Avnet FXT evaluation board
- Memec SystemACE Module
- Straight through RS232 cable
- Power supply
- JTAG programming cable (USB or PC4)

3 SystemACE Interface Design Block Diagram

The following figure shows a high-level block diagram of the SystemACE module reference design. The design consists of:

- PowerPC processor
- 32KB of BRAM
- 64MB of DDR SDRAM
- LEDs
- RS232 Port
- SystemACE Controller
- Timer
- Interrupt Controller

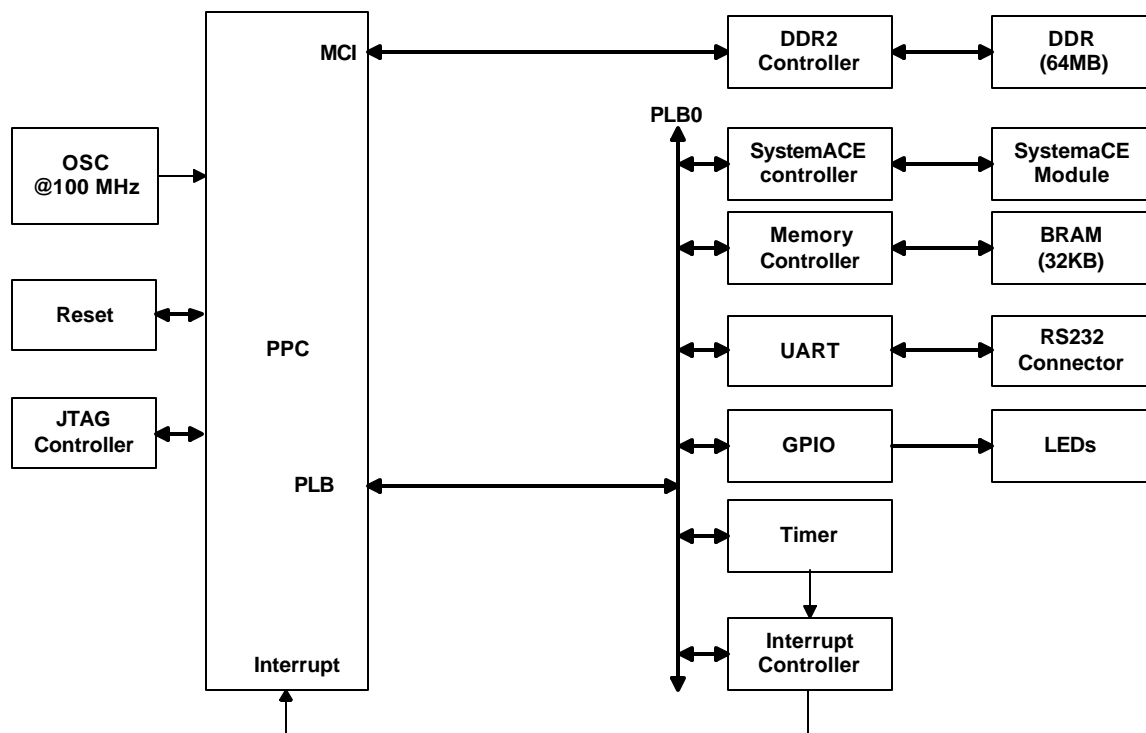


Figure 1 – Reference Design Block Diagram

4 Design Software

The software for this design consists of two test programs and an application program. The **SystemACE_Self_Test** software project simply runs a test on the SystemACE controller device to verify its operation. Once the SystemACE self-test is completed, the **SystemACE_CF_Test** software project is used to verify the CompactFlash card connected to the SystemACE controller. This task is accomplished by writing to a few sectors on the CF as well as reading from it.

The **Running_From_DDR** application software project executes code from the external DDR SDRAM and runs a program that illustrates features of the Xilinx Micro Kernel (XMK) Operating System.

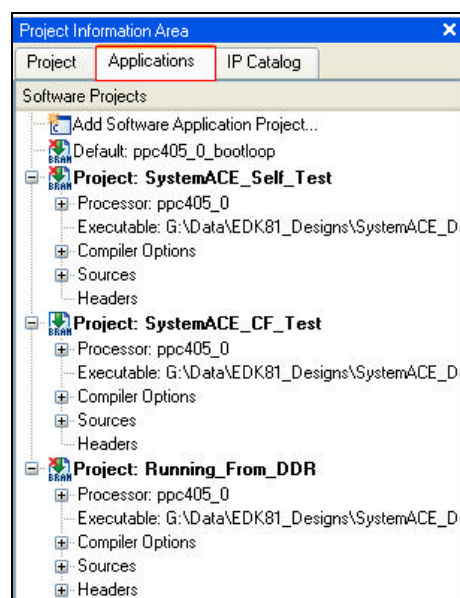
5 Setting Up the Board

Perform the following steps to setup the board for running the SystemACE module reference design programs.

1. Verify the Power switch, **SW7**, is in the **OFF** position.
2. Install a jumper on JP3 pins 2-3
3. Install a jumper on JP2 pins 2-3
4. Install a jumper on JP1 pins 1-2
5. Install a jumper on JP5 pins 2-3 (FPGA JTAG mode)
6. Connect the power supply to the J11 connector on the FXT evaluation board and also plug it into the AC outlet.
7. Plug the SystemACE module onto the FXT evaluation board using the 50-pin header (JP9) located on the FXT evaluation board.
8. Connect the JTAG cable to JP4 located on the **SystemACE Module** and the parallel/USB port of the PC.
9. Connect a straight through RS232 cable to the board DB-9 connector (P1) and the serial port of the PC. Alternatively, you can use an RS232-USB adapter and connect this adapter to the DB-9 connector and the USB port of the PC. In this case, you must install the RS232-USB driver for the adapter.
10. Slide the power switch to the **ON** position

6 Implementing the Design

- Open the SystemACE module reference design in XPS and click on the **Applications** tab. You should see three software projects associated with this design as shown in the following figure.



- Select **Software > Build All User Applications** to compile the software source codes for the three software projects associated with this reference design.
- Select **Device Configuration > Update Bitstream** from the XPS GUI to build the design and generate a bit file.
- Open an EDK shell from the XPS GUI by selecting **Project > Launch EDK Shell**.
- At the EDK shell prompt, enter the following to generate an ACE file for the SystemACE self test program:

```
cd SystemACE_Self_Test/Generate_ACE_File
./genace_systemace_self_test.sh
```

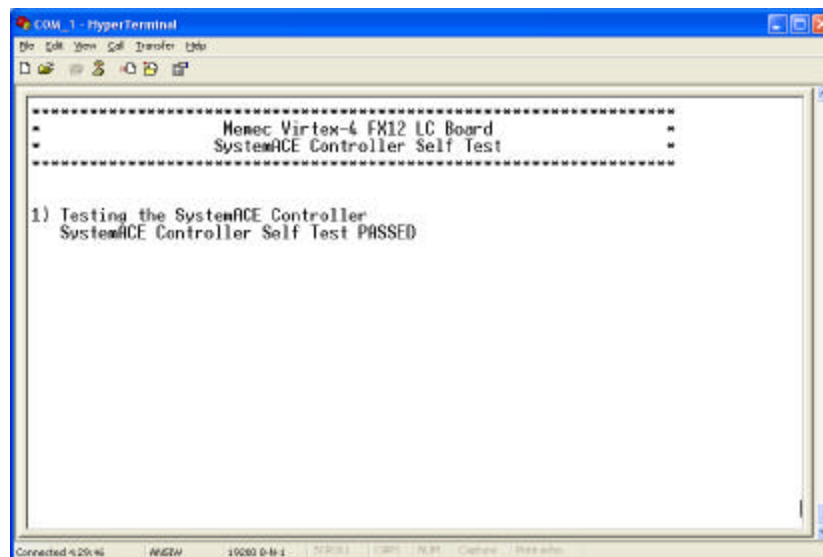
The shell file stores the generated ACE file in the **ACE_Files\ace_file\rev0** folder of the project directory.

- Insert a CompactFlash card into the PC card reader, go to the **/mkdosfs** folder of the project and enter the following command to format the CF card:

```
mkdosfs <CF_Drive_Letter>:
```

For example **mkdosfs J:**, where **J** is the CF card drive letter.

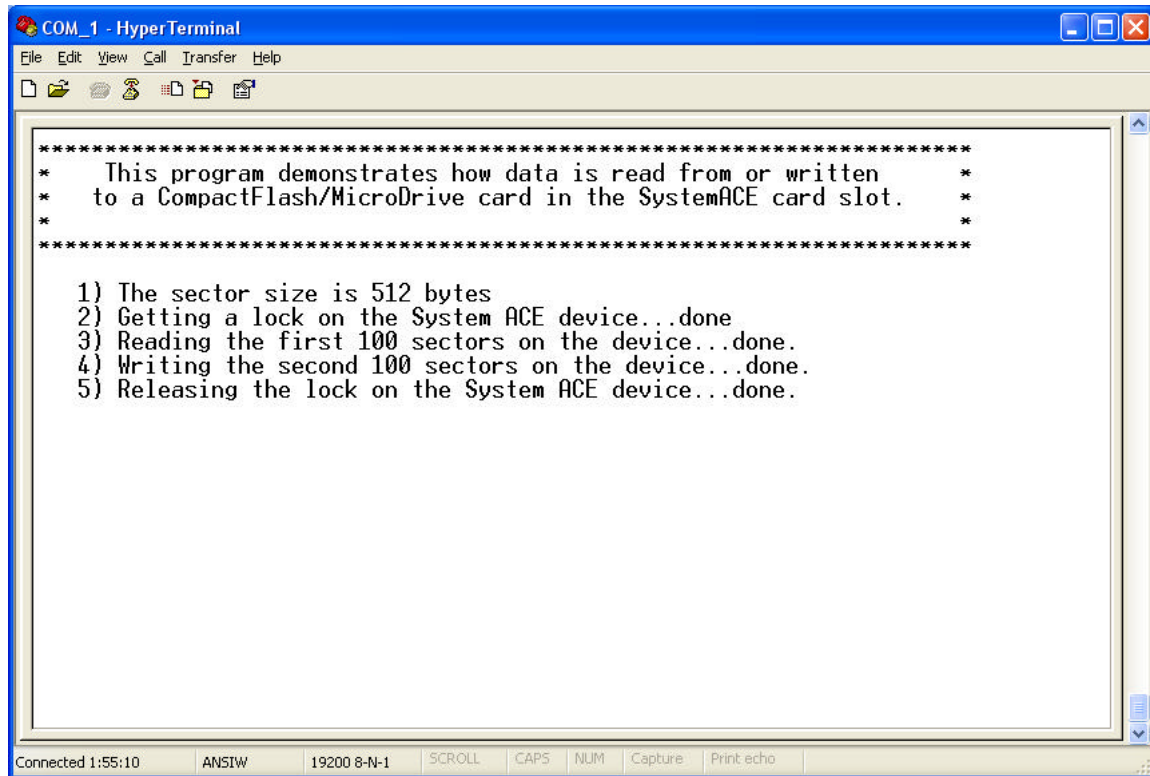
- Go to the **ACE_Files** folder of the project directory and copy the contents of this folder (the **ace_file** folder and the **xilinx.sys** file) to the CF card root directory. When copy is completed, remove the CompactFlash card from the CompactFlash reader.
- Insert the CompactFlash card into the SystemACE Module (SAM).
- Make sure the rotary switch (**SW1**) is set to position '**0**' on the SystemACE module.
- Start a Hyper Terminal session and set the serial port parameters to 19200-baud rate, 8 bits, 1 stop bit, no parity and no flow control.
- Slide the board power switch (**SW7**) to the **ON** position. The SystemACE self-test program will be loaded into the FPGA and you should see the following on the Hyper Terminal.



7 Running the SystemACE CF Test Program

After verifying the operation of the SystemACE controller, the CompactFlash interface can be tested using the **SystemACE_CF_Test** software project. This software performs sector reads and writes of the CF card.

- Select **Device Configuration > Download Bitstream** from the XPS GUI to download the SystemACE CF test design to the board. You should see the following on the Hyper Terminal.



```
*****
*   This program demonstrates how data is read from or written   *
*   to a CompactFlash/MicroDrive card in the SystemACE card slot. *
*                                                                 *
*****

1) The sector size is 512 bytes
2) Getting a lock on the System ACE device...done
3) Reading the first 100 sectors on the device...done.
4) Writing the second 100 sectors on the device...done.
5) Releasing the lock on the System ACE device...done.
```

8 Running Program From External DDR SDRAM

The **Running_From_DDR** software project illustrates the use of Xilinx Micro-Kernel (XMK) operating system. Please refer to the “Using Xilkernel” chapter of the Platform Studio User Guide (**ps_ug.pdf**) located in the /doc folder of the EDK install directory for information on the Xilinx Micro-Kernel operating system.

The software source code for the **Running_From_DDR** software project is located in **Running_From_DDR\src** folder of the project directory. The software consists of the following threads:

Thread	Description
<i>shell</i>	This is the main controlling thread and presents a shell with a few simple commands from which you can launch the other demo threads.

<i>prodcon</i>	Producer consumer example thread(s) using message queues.
<i>llist</i>	Linked list demo using the buffer memory allocation interfaces.
<i>sem</i>	Semaphore example showing multiple competing threads using semaphores to coordinate.
<i>TicTacToe</i>	Simple tic-tac-toe game, which illustrates how to dynamically assign stack memory to a thread when creating it.
<i>TimerTest</i>	Simple time management demo.
<i>prio</i>	Thread illustrating dynamically changing priorities and priority queues in the kernel structures.
<i>mutex</i>	Mutex demo, illustrating pthread mutex locks.
<i>clock</i>	Simple thread, using the second timer device and handling interrupts from it, to keep track of wall-clock time. This illustrates user-level interrupt handling.
<i>standby</i>	Simple standby thread.

- Select **Debug > Launch XMD** from the XPS GUI to download the XMK demo software to the external DDR SDRAM using the GNU debugger and run the program. The XMD command window will appear and it should look similar to the window shown in the following figure.

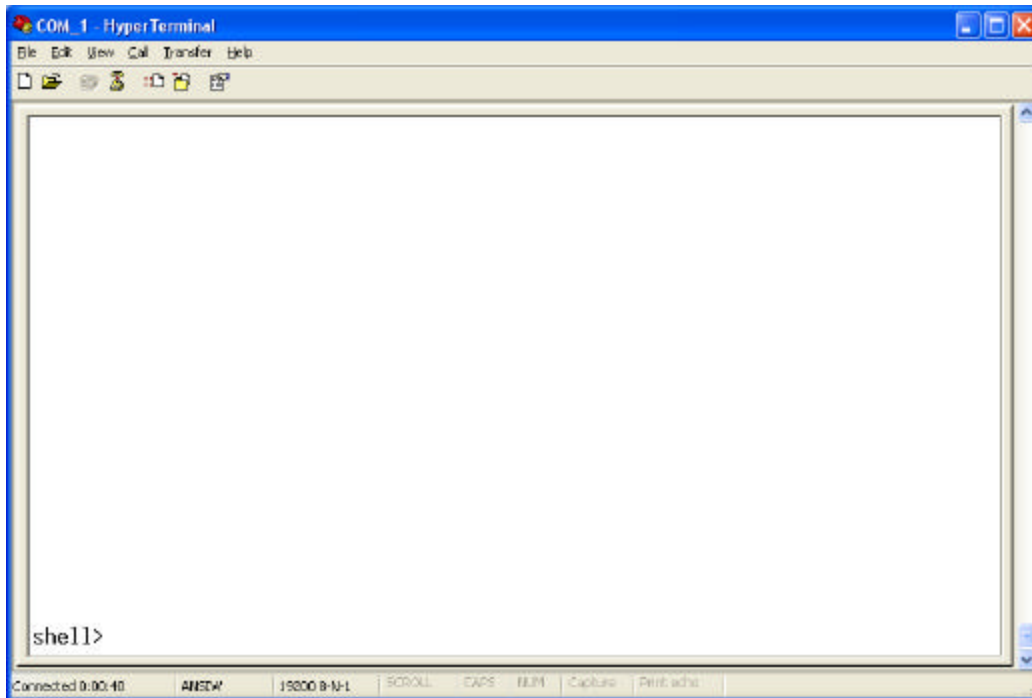
```

C:\> C:\EDK_82\bin\nt\xmd.exe

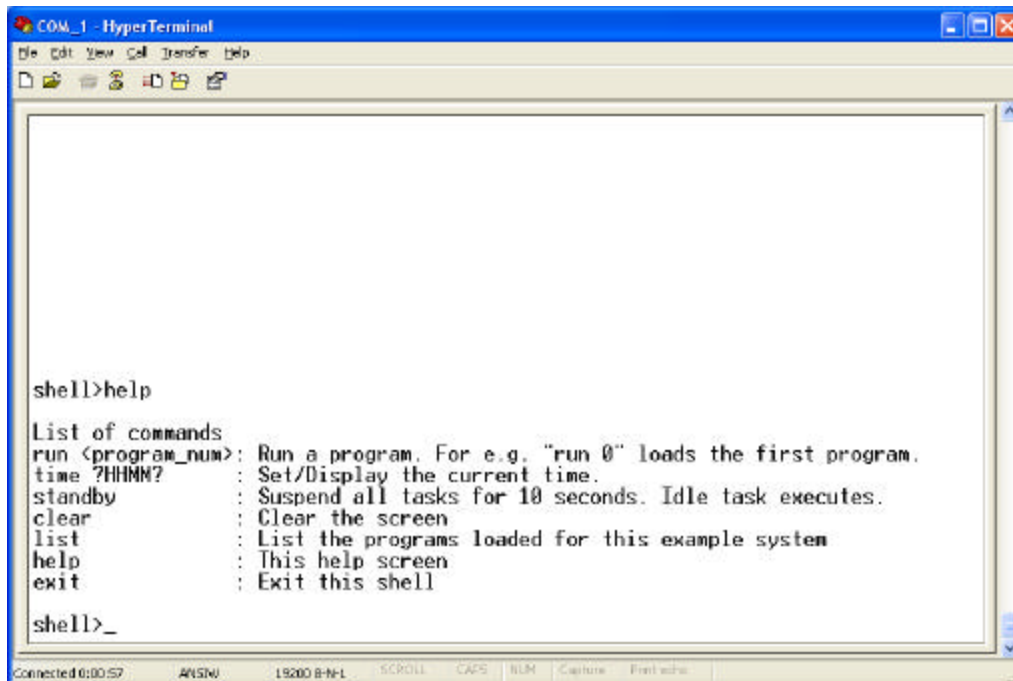
section, .boot: 0x000013158-0x00001315c
section, .rodata: 0x00000b550-0x00000e216
section, .sdata2: 0x00000e218-0x00000e218
section, .sbss2: 0x00000e218-0x00000e218
section, .data: 0x00000e218-0x00000e5c8
section, .got: 0x00000e5c8-0x00000e5c8
section, .got1: 0x00000e5c8-0x00000e5c8
section, .got2: 0x00000e5c8-0x00000e5e4
section, .ctors: 0x00000e5e4-0x00000e5ec
section, .dtors: 0x00000e5ec-0x00000e5f4
section, .fixup: 0x00000e5f4-0x00000e5f4
section, .eh_frame: 0x00000e5f4-0x00000e5fc
section, .jcr: 0x00000e5fc-0x00000e600
section, .gcc_except_table: 0x00000e600-0x00000e600
section, .sdata: 0x00000e600-0x00000e628
section, .sbss: 0x00000e628-0x00000e750
section, .bss: 0x00000e750-0x000013148
section, .stack: 0x00001315c-0x000013560
section, .heap: 0x000013560-0x000013960
Downloaded Program xilkernel_demo/executable.elf
Setting PC with program start addr = 0x000013158
PC reset to 0x000013158. Clearing MSR Register
PC reset to 0x000013158. Clearing MSR Register
Processor started. Type "stop" to stop processor
XMD%

```

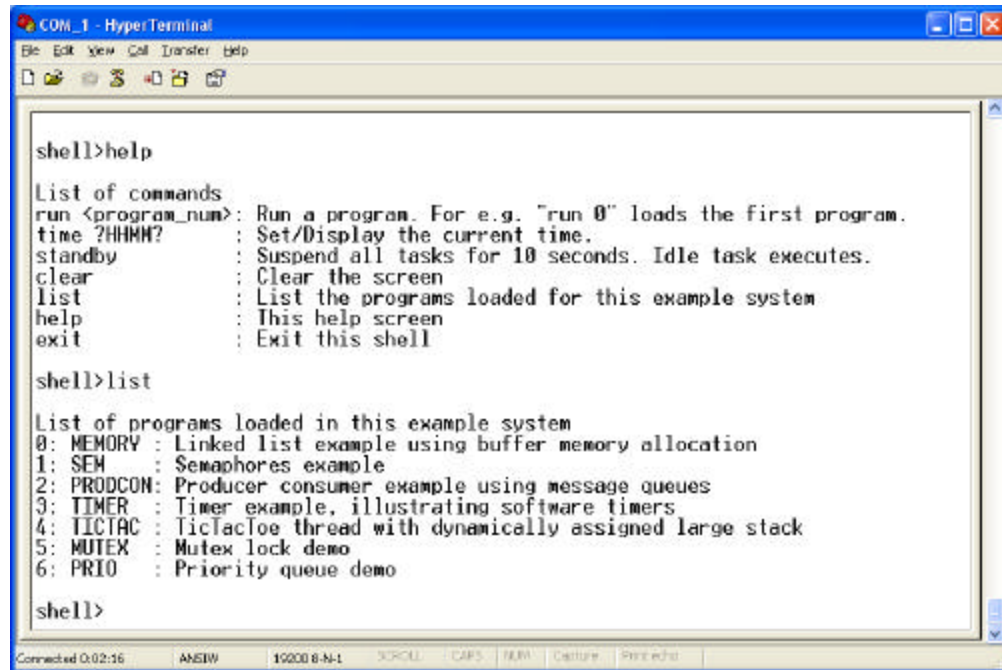
- After downloading the XMK demo program to the external DDR SDRAM, you should see the following on the Hyper Terminal.



- Enter “**help**” to get a list of commands.



- Enter **“list”** to get a list of programs.



```

COM_1 - HyperTerminal
File Edit View Call Transfer Help

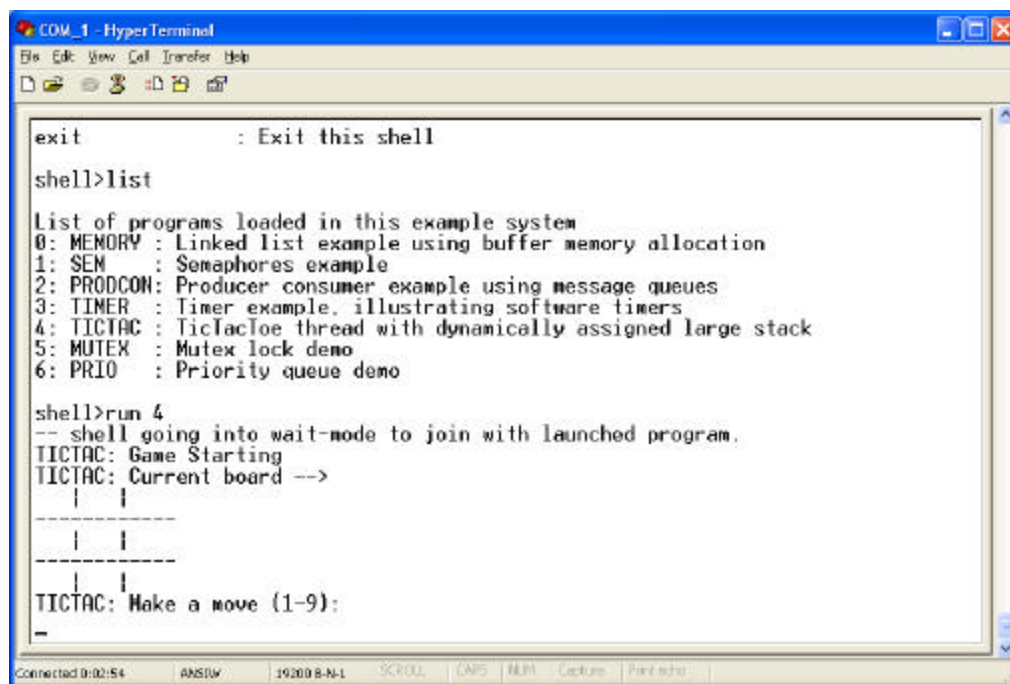
shell>help
List of commands
run <program_num>: Run a program. For e.g. "run 0" loads the first program.
time ?HHMM?      : Set/Display the current time.
standby          : Suspend all tasks for 10 seconds. Idle task executes.
clear            : Clear the screen
list             : List the programs loaded for this example system
help             : This help screen
exit             : Exit this shell

shell>list
List of programs loaded in this example system
0: MEMORY : Linked list example using buffer memory allocation
1: SEM     : Semaphores example
2: PRODCON: Producer consumer example using message queues
3: TIMER   : Timer example, illustrating software timers
4: TICTAC  : TicTacToe thread with dynamically assigned large stack
5: MUTEX   : Mutex lock demo
6: PRIO    : Priority queue demo

shell>

```

- Enter **“run”** followed by a number (0-6) to run a program. For example, enter **“run 4”** to play the Tic-Tac-Toe game.



```

COM_1 - HyperTerminal
File Edit View Call Transfer Help

exit      : Exit this shell

shell>list
List of programs loaded in this example system
0: MEMORY : Linked list example using buffer memory allocation
1: SEM     : Semaphores example
2: PRODCON: Producer consumer example using message queues
3: TIMER   : Timer example, illustrating software timers
4: TICTAC  : TicTacToe thread with dynamically assigned large stack
5: MUTEX   : Mutex lock demo
6: PRIO    : Priority queue demo

shell>run 4
-- shell going into wait-mode to join with launched program.
TICTAC: Game Starting
TICTAC: Current board -->
|  |
|  |
|  |
-----
|  |
|  |
|  |
TICTAC: Make a move (1-9):
_

```

9 Generating an ACE File to Run the XMK Demo Program From the External DDR SDRAM

In the previous section, the GNU debugger was used to load the XMK demo program to the external DDR SDRAM and run the program. The following section describes how to generate an ACE file for the XMK demo program. When this ACE file is loaded into the FPGA by the SystemACE module, the FPGA is configured, the XMK demo code is loaded into the external DDR SDRAM and the code is executed without any assistance from the GNU debugger.

- Slide the board power switch (**SW7**) to the **OFF** position.
- Open an EDK shell from the XPS GUI by selecting **Project > Launch EDK Shell** (if one is not already open).
- At the EDK shell prompt, enter the following to generate an ACE file for the SystemACE self test program:

```
cd Running_From_DDR/Generate_ACE_File  
./genace_running_from_ddr.sh
```

The shell file stores the generated ACE file in the **ACE_Files\ace_file\rev1** folder of the project directory.

- Remove the CF card from the SystemACE module and insert it into the PC CompactFlash card reader.
- Go to the **ACE_Files** folder of the project directory and copy the contents of this folder (the **ace_file** folder and the **xilinx.sys** file) to the CF card root directory overwriting the existing files on the CF root directory. When copy is completed, remove the CompactFlash from the CompactFlash reader.
- Insert the CompactFlash card into the SystemACE Module (SAM).
- Make sure the rotary switch (**SW1**) is set to position '**1**' on the SystemACE module. The position '**0**' of the rotary switch points to the SystemACE self test program while the position '**1**' points to the XMK demo program.
- Start a Hyper Terminal session (if one is not already started) and set the serial port parameters to 19200-baud rate, 8 bits, 1 stop bit, no parity and no flow control.
- Slide the board power switch (**SW7**) to the **ON** position. The SystemACE module will configure the FPGA, load the XMK demo program from the CF to the on-board SDRAM, and execute it. You should see the following on the Hyper Terminal.

