Real-Time Systems Using Cyclic Executive Approach

Objective: Cyclic execution is a commonly used approach in embedded systems, which can also be used for implantation of basic real-time systems. The purpose of this lab is to practice the cyclic execution approach. Through the lab exercises, you will become more familiar with the interval timer in DE2-series computers and learn how the timer in a DE2-Series Computer System is used to trigger a periodic interrupt, where the period is set to a value equal to the frame size. Upon the arrival of an interrupt, a predetermined schedule is used to determine the task(s) to be executed in each frame.
You will also learn how the module's counter can be used to estimate the performance of an arbitrary application program, in terms of the total number of clock cycles needed to execute the program.

Resources: In order to carry out this lab, you will require the following resources. More details of the software resources to be used in the lab can be found on Altera website and the class webpage associated with this lab.

1. A computer running Linux, preferably Redhat Enterprise Linux (RHEL) 6.8 or 7.3. Any of the computers provided in the Computer Engineering Labs can be used for carrying out the work associated with this lab.
2. Intel FPGA Development and evaluation board: The process of downloading and debugging a Nios II program requires the presence of an FPGA board to implement the Nios II hardware system. In this lab, Altera DE2-115 Development and Education board has been used to run this lab. The board can be obtained from the Support Engineers. Other DE-series boards (DE1-SoC, DE2, etc.) can also be used.
3. Intel Quartus: The subscription editions of Intel Quartus has been used to run this lab.
4. Intel Nios Embedded Design Suite (EDS): Nios II EDS is released as part of Quartus software.
5. Intel FPGA Monitor Program: Intel FPGA Monitor Program is released as part of University Program Design Suite (UPDS).

Preparation:

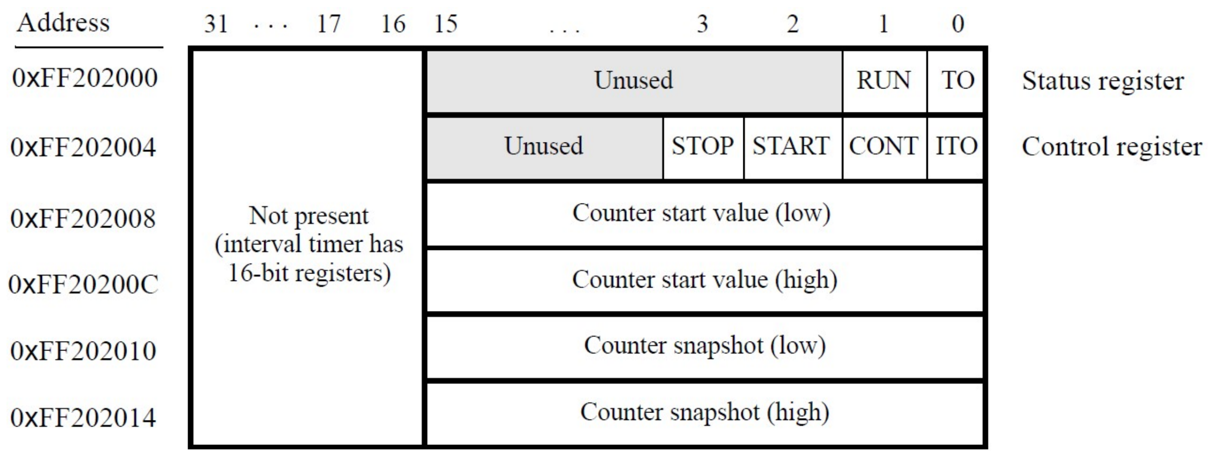
1. Review DE2-115 Computer system
2. Review Intel FPGA Monitor Program
3. Review C language

Procedure:

This section describes the specific steps you are to take in carrying out this lab. Your mark will be based on the contents of your lab report and the evaluations of your in-lab work by the person(s) running the lab session. Be complete in your lab report: include the output produced by the run steps (either by hand-written entries or by printing) and clearly write all required analyses and modifications. Submit your lab report to the "cyclic_lab" hand-in folder through Black Board course tools.

Part I: Measurement of Execution Time

In real-time embedded systems it is often necessary to know execution time of an application program. The Interval Timer can be used for this purpose by measuring the elapsed clock cycles for the application program. The registers for one of the Interval Timers in DE2-115 computer are shown below.



Perform the following:

1. Create a new folder, named timer_part1 to hold your Monitor Program project for this part.
2. Create a file called timer_part1.c.
3. Write a C language code that can determine the execution time of a program.
   - The execution time should be measured in terms of the number of clock cycles from start to completion of an application program
   - The result (number of cycles) should be displayed on the seven-segment displays HEX7-0 on the DE-series board in decimal form.
4. Make a new Monitor Program project in the folder where you stored the timer_part1.c file.
5. Compile, download, and test your program.
   - A very simple application program, which determines the largest number in a list of integers, is provided as a test design file with this exercise. Use this application to test your program.
   - Note down the measured execution time in your lab report.

```
/*****************************************************************/
/* This is a simple test program. It finds the largest number in  */
/* a list of integers.                                             */
/*****************************************************************/
#define BIGNUM (int *) 0x20000
int LIST[7] = {4, 5, 3, 6, 1, 8, 2};
void TEST_PROGRAM()
{
   int big, i;
   big = LIST[0];
   for ( i = 1; i <= 6; i++)
   {
         if ( LIST[i] > big )
               big = LIST[i];
   }
   *BIGNUM = big;
}
```

6. Test your program with an algorithm for square root approximation. The pseudocode of an algorithm is provided below.
   - Write a C program to implement the algorithm.

- The input, x, should be set by SW_{17-0}.
- The square root result, y, should be displayed on HEX7-0.
- Press KEY1 to start running the square root approximation program and measuring its execution time.
- Press KEY2 to toggle HEX7-0 display between elapsed clock cycles and square root result, y.
- Measure the execution times using different input values. Note down the measured execution times in your lab report.

```
/**************************************************************************
* This is a pseudocode of a square root approximate algorithm.
* It finds integer square root of a binary number
* Example 1: x = 18₁₀ = 010010₂
* i    x - y²    y
* -------------------------
* 2    > 0      000+100=100
* 1    > 0      100+010=110
* 0    < 0      110-001=101
**************************************************************************/
int x, y;
y = 0;
for i := n-1 downto 0 {
    if (x - y² = 0)
        exit;
    else if (x - y² > 0)
        y = y + 2ⁱ;
    else       /* if (x - y² < 0) */
        y = y - 2ⁱ;
}
```

7. Compress your source code (including test programs) into one zip file named <nsid>_timer_part1.zip. Hand in the zip file to the “cyclic_lab” hand-in folder through Black Board course tools.

Part II: Stopwatch

Part II of this lab is to implement a simple stopwatch that displays elapsed time on HEX7-4 in the format of MM:SS, where MM are minutes and SS are seconds.

Perform the following:

1. Create a new folder, named timer_part2 to hold your Monitor Program project for this part.
2. Create a file called timer_part2.c
3. Write a C program that displays time on the seven segments HEX5-2.
   - The time displayed on HEX7-4 should be in the format of MM:SS, where HEX7-6 display MM and HEX5-4 display SS.
   - Time should be in the accuracy of at least 0.25 seconds
   - When the timer reaches 59:59, it should roll back to 00:00.
   - Toggle between run and stop when any KEY is pressed.
4. Make a new Monitor Program project in the folder where you stored the timer_part2.c file.
5. Compile, download, and test your program.
6. Calculate the accuracy of your real-time clock (deviation per month).

7. Compress your source code into one zip file named <nsid>_timer_part2.zip. Hand in the zip file to the "cyclic_lab" hand-in folder through Black Board course tools.

Part III: Implementation of cyclic executive

We now turn our attention to implementing a simple binary game on DE2-115 board according to the cyclic executive approach. The binary game should meet the following requirements.

- Each round of the game consists of 10 questions.
- For each question, display one randomly generated number in decimal on HEX3-0.
- Use SW to set the equivalent binary number.
- Each question has 60 seconds to answer.
- Display remaining time in seconds on HEX5-4.
- Clear HEX5-4 at the end of a game.
- Display current score on HEX7-6.
- Keep track of total elapsed game time with the accuracy of at least 0.25 seconds. Note: The elapsed time should stop counting at the end of a game.
- Press KEY1 to start a new game or reset the game.
- Press KEY2 to enter an answer.
- Press KEY3 to toggle HEX7-4 display between total elapsed game time and total score after the end of a game. The total elapsed game time should be in MM:SS format

For learning purpose, your implementation must consist of **at least three periodic tasks**; for instance, a game task, and timing task and a display task.

- The system should check **frame-overrun** at the beginning of each frame. Turn on LEDG8 if there is frame overrun. Debug your C program to fix the problem.
- While the system is not running any of the system tasks, LEDG7 should be lit.
- (Optional) Turn on different LEDGs for individual tasks.

Perform the following.

1. In the lab report, describe in detail how you intend to implement the system. For instance,
   - Functional description of each task and task parameters such as phase, deadline, period.
   - Calculation of frame size.
   - A feasible schedule or time table showing the tasks to be run in each frame.
2. Create a new folder, named cyclic_part1 to hold your Monitor Program project for this part.
3. Make a new Monitor Program project in the cyclic_part1 folder. Use the DE1-115 Computer for this project, and select Nios II as the target processor architecture.
4. Write a C program to implement the system.
5. Compile, download, and test your program.
6. Compress your source code into one zip file named <nsid>_cyclic_part1.zip. Hand in the zip file to the "cyclic_lab" hand-in folder through Black Board course tools.
7. In the lab report, describe in detail how you implemented the system in C program for DE2-115 Computer.