

*Lab 3: LTM Integration

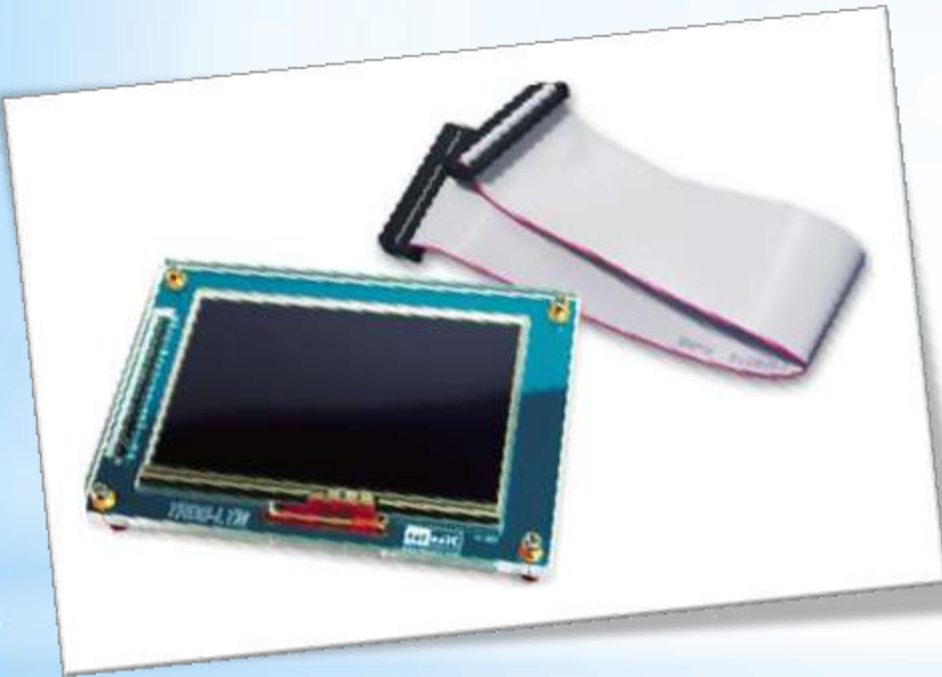
How to create a SOPC system integrated with LTM?

Written by Geng You Chen
(gengyouchen@gmail.com)

*What you've learned...

- * Introduction to SOPC Lab (Total 17 pages written by Geng You Chen)
 - * The motives of dividing system functions into hardware and software
 - * The difference of hardware structures between a system written by yourself and a system generated by Altera SOPC builder
 - * The relationship between IORD/IOWR and Interface Registers of IPs
 - * The responsibility of Drivers: Register Maps, Hardware Abstractions
- * Lab 1: Hello μ C/OS-II (Total 109 pages written by Geng You Chen)
 - * The role of μ C/OS-II in SOPC and its mechanism: Context Switching
 - * The functions of on-chip components (LE, PLL, M4K, Multiplier) and off-chip components (SSRAM, SDRAM, Flash, JTAG, Character LCD)
 - * The meanings of Reset Vector and Exception Vector in Nios II CPU
- * Lab 2: Hello Avalon-MM (Total 27 pages written by Geng You Chen)
 - * The scheduling behavior of μ C/OS-II for different priorities: RTOS
 - * The Memory Mapped viewpoint of CPU running C/C++ programs
 - * The behavior of CPU Cache and its drawback to Memory Mapped I/O
 - * The purposes of each hardware interfaces: Clock, Interrupt, Conduit, Avalon-MM, Avalon-MM Tristate, Avalon-ST

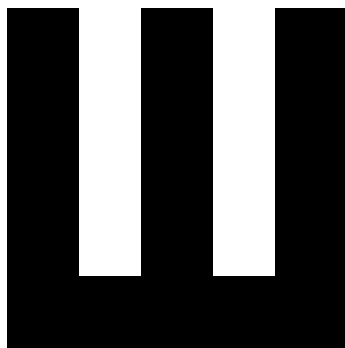
*What you're going to learn...



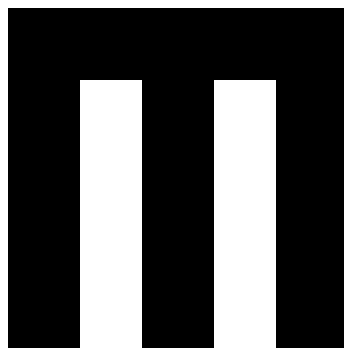
- * LTM (Touch Panel Module) is one of the most conspicuous device on Terasic DE2-70 Board.
- * However, LTM is not come from Altera Corporation, so you need to write a controller by yourself.
- * For simplicity of your controller, in Lab 3, **you will use only the display function** without using the touch function.

* A simple LTM controller for eye vision charts

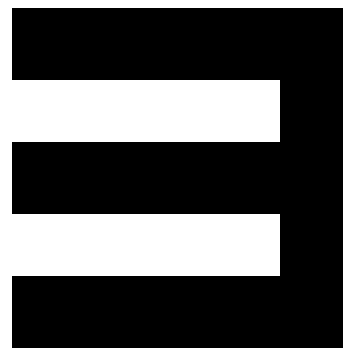
- * In Lab 3, because learning how to control LTM is much more important than trying to draw a very complicated picture, **you only need to implement an eye vision charts controller rather than a fully function controller.**
- * Your eye vision charts controller must provide **an interface register “symbol” which allows the C/C++ program to change the current symbol** among the following four ones:



Symbol(0)



Symbol(1)

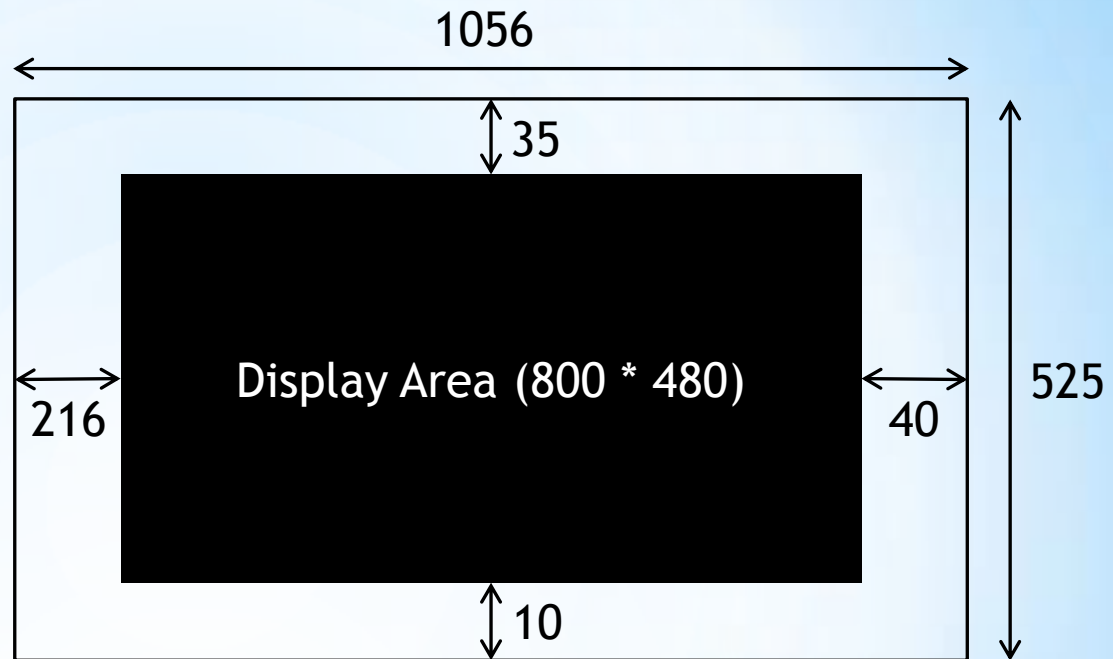


Symbol(2)



Symbol(3)

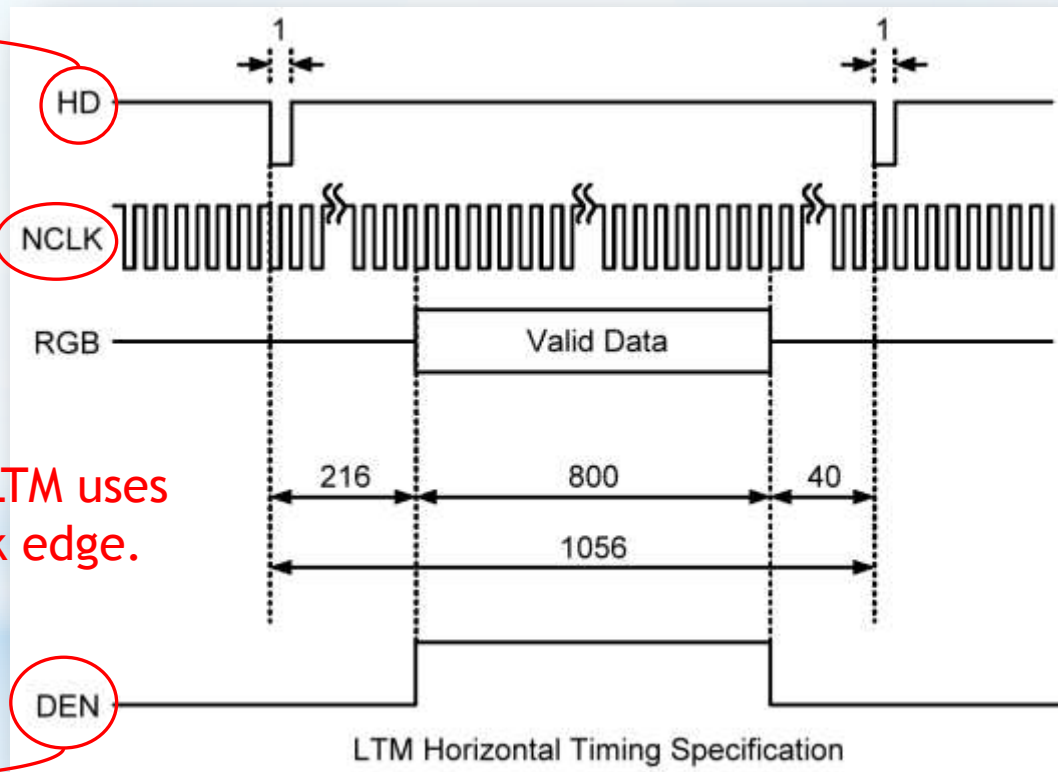
* Scanning RGB plane



- * Unfortunately, your IP cannot simply tell LTM that “I want to change the color of (X,Y) to (R,G,B)”. Instead, you need to sequentially tell LTM that the RGB color of (0,0), (1,0), (2,0), (3,0)...(1055,0), (0,1), (1,1), (2,1), (3,1)...(1055,1), (0,2), (1,2), (2,2), (3,2)...(1055,2).....(1055,524). And then, you need to back to (0,0) and repeat this routine. In other words, **your LTM controller is scanning RGB plane repeatedly.**
- * Noticed that your RGB outputs will be displayed only from (216,35) to (1015,514). Although other RGB outputs will be ignored by LTM, your LTM controller still need to **treat 1056 * 525 as your plane size, not 800 * 480. Otherwise, you will violate the timing specification.**

*LTM timing specification (zoomed-in waveform)

You need to output a signal “HD” to do horizontal synchronization (low-active) at the beginning (1 clock cycle) of every horizontal lines (1056 clock cycles).

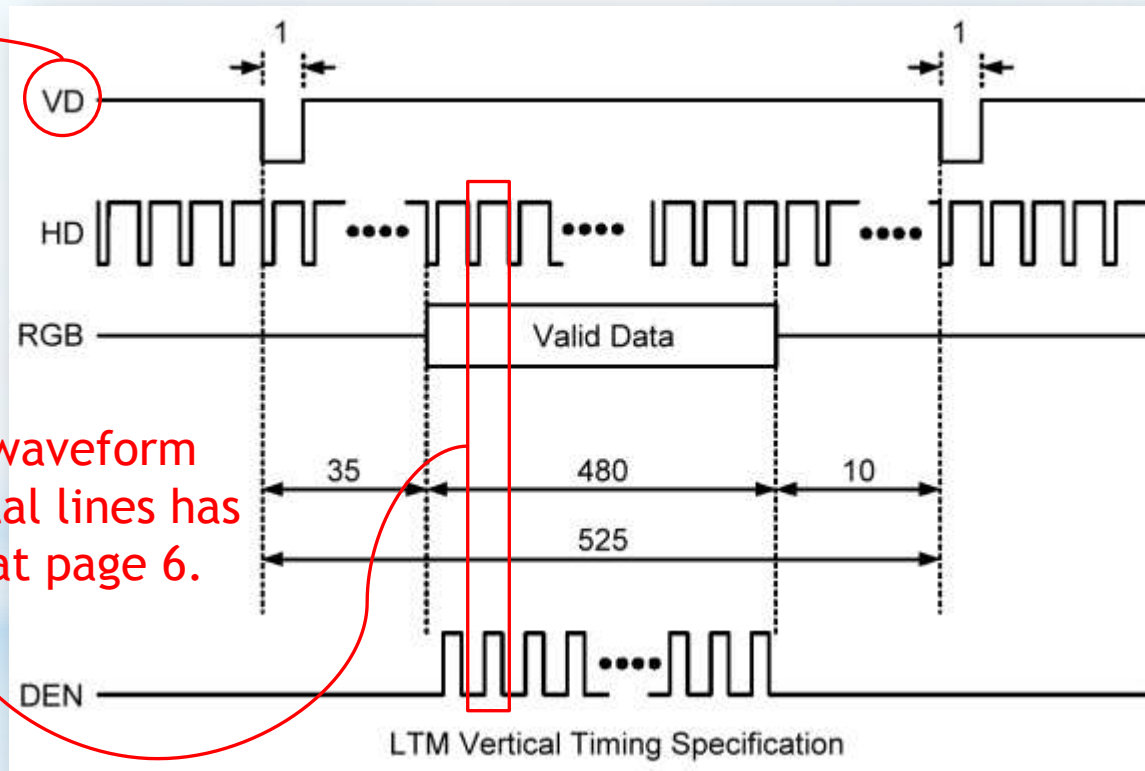


Noticed that LTM uses negative clock edge.

You need to output a signal named “DEN” to enable RGB data (high-active) inside the display area (800 * 480).

*LTM timing specification (zoomed-out waveform)

You need to output a signal “VD” to do vertical synchronization (low-active) at the beginning (1 horizontal line) of every vertical lines (525 horizontal lines).



The zoomed-in waveform of each horizontal lines has been discussed at page 6.

*Pseudocode (C/C++) for LTM timing controller

```
void timing_controller ( ) {
```

```
    while ( 1 ) {
```

```
        for ( j = 0 ; j < 525 ; j++ ) {
```

```
            for ( i = 0 ; i < 1056 ; i++ ) {
```

```
                bool HD = ( i != 0 ) ;
```

```
                bool VD = ( j != 0 ) ;
```

```
                bool DEN = ( i >= 216 ) && ( i < 1016 ) && ( j >= 35 ) && ( j < 515 ) ;
```

```
                int RGB = pattern_generator ( i - 216 , j - 35 ) ;
```

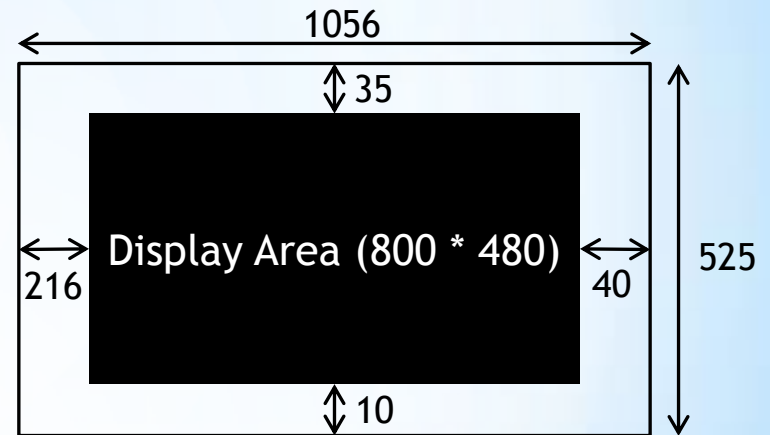
```
                output_to_LTM ( HD , VD , DEN , RGB ) ;
```

```
            }
```

```
        }
```

```
    }
```

```
}
```



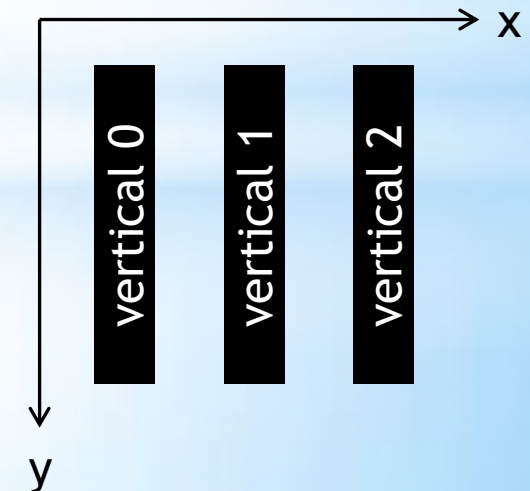
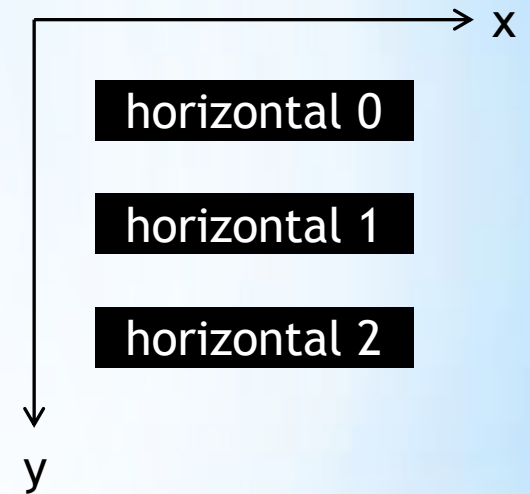
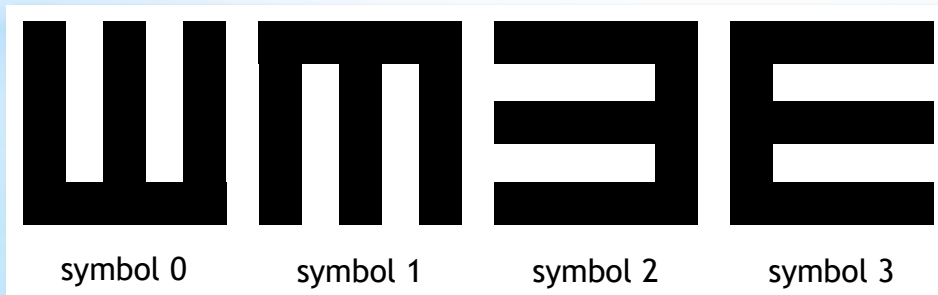
The pseudofunction `output_to_LTM` will lock the output data for a clock cycle.

If (i , j) is inside the display area, the pattern generator will convert (x , y) at ($i - 216$, $j - 35$) to a RGB value. Otherwise, the output data of the pattern generator will be ignored by LTM because “DEN” is equal to zero.

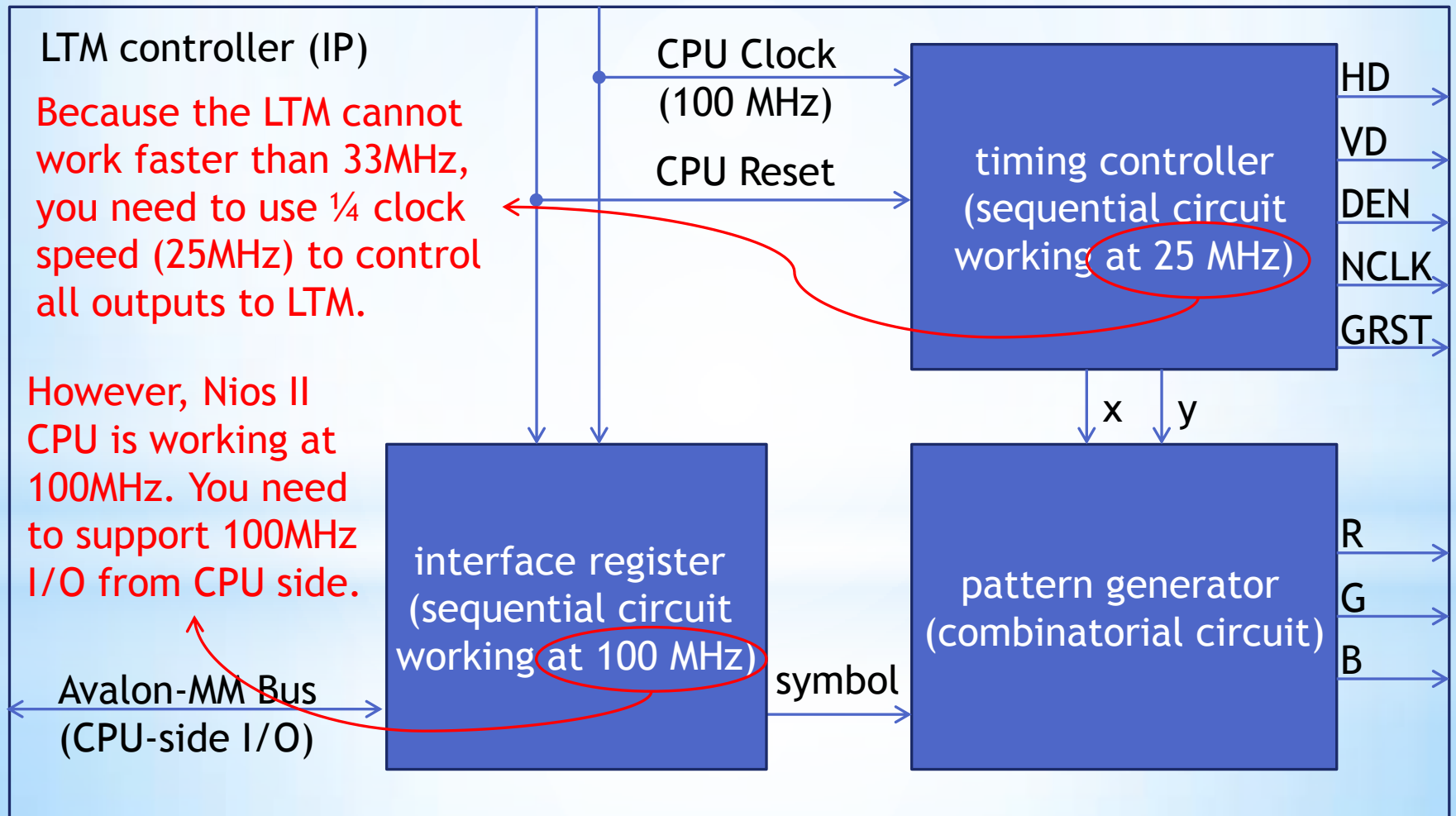
*Pseudocode (C/C++) for LTM pattern generator

```
int pattern_generator ( int x , int y ) {  
    bool horizontal0 = ( x >= 200 ) && ( x < 600 ) && ( y >= 40 ) && ( y < 120 ) ;  
    bool horizontal1 = ( x >= 200 ) && ( x < 600 ) && ( y >= 200 ) && ( y < 280 ) ;  
    bool horizontal2 = ( x >= 200 ) && ( x < 600 ) && ( y >= 360 ) && ( y < 440 ) ;  
    bool vertical0 = ( x >= 200 ) && ( x < 280 ) && ( y >= 40 ) && ( y < 440 ) ;  
    bool vertical1 = ( x >= 360 ) && ( x < 440 ) && ( y >= 40 ) && ( y < 440 ) ;  
    bool vertical2 = ( x >= 520 ) && ( x < 600 ) && ( y >= 40 ) && ( y < 440 ) ;  
    bool symbol0 = horizontal2 || vertical0 || vertical1 || vertical2 ;  
    bool symbol1 = horizontal0 || vertical0 || vertical1 || vertical2 ;  
    bool symbol2 = horizontal0 || horizontal1 || horizontal2 || vertical2 ;  
    bool symbol3 = horizontal0 || horizontal1 || horizontal2 || vertical0 ;  
    if ( ( symbol == 0 ) ? symbol0 : ( ( symbol == 1 ) ? symbol1 : ( ( symbol == 2 ) ? symbol2 : symbol3 ) ) ) {  
        return 0x000000 ;  
    } else {  
        return 0xFFFF ;  
    }  
}
```

Combine 3 horizontals and 3 verticals into 4 symbols.



*Implement pseudocode as a real hardware IP



*Timing Controller (1/3)

```
module timing_controller (  
  
    // Define output signals to LTM  
    output HD,  
    output VD,  
    output DEN,  
    output NCLK,  
    output reg GRST,  
  
    // Define output signals to pattern generator  
    output [31:0] x,  
    output [31:0] y,  
  
    // Define input signals from LTM controller  
    input clock,  
    input reset  
  
);
```

*Timing Controller (2/3)

```
// Generate 25 MHz clock signal for LTM
reg [2:0] counter1;
assign NCLK = ~counter1[2];
always @ (posedge clock or negedge reset) begin
    if (~reset) begin
        counter1 <= 0;
    end else begin
        counter1 <= counter1 + 1;
    end
end

// Generate delayed reset signal for LTM
reg [23:0] counter2;
always @ (posedge clock or negedge reset) begin
    if (~reset) begin
        GRST <= 0;
        counter2 <= 0;
    end else if (counter2 != 24'hFFFFFF) begin
        GRST <= 0;
        counter2 <= counter2 + 1;
    end else begin
        GRST <= 1;
        counter2 <= counter2;
    end
end
```

*Timing Controller (3/3)

```
// Generate timing signal using LTM's clock and reset
reg [31:0] i;
reg [31:0] j;
assign x = i - 216;
assign y = j - 35;
assign HD = i != 0;
assign VD = j != 0;
assign DEN = i >= 216 & i < 1016 & j >= 35 & j < 515;
always @ (negedge NCLK or negedge GRST) begin
    if (~GRST) begin
        i <= 0;
        j <= 0;
    end else if (i < 1055) begin
        i <= i + 1;
        j <= j;
    end else if (j < 524) begin
        i <= 0;
        j <= j + 1;
    end else begin
        i <= 0;
        j <= 0;
    end
end
end

endmodule
```


*Pattern Generator (1/2)

```
module pattern_generator (  
  
    // Define output signals to LTM  
    output [7:0] R,  
    output [7:0] G,  
    output [7:0] B,  
  
    // Define input signals from timing controller  
    input [31:0] x,  
    input [31:0] y,  
  
    // Define input signals from LTM controller  
    input [1:0] symbol  
  
);
```


*Pattern Generator (2/2)

```
// Define six rectangles regions
wire horizontal0, horizontal1, horizontal2, vertical0, vertical1, vertical2;
assign horizontal0 = x >= 200 & x < 600 & y >= 40 & y < 120;
assign horizontal1 = x >= 200 & x < 600 & y >= 200 & y < 280;
assign horizontal2 = x >= 200 & x < 600 & y >= 360 & y < 440;
assign vertical0 = x >= 200 & x < 280 & y >= 40 & y < 440;
assign vertical1 = x >= 360 & x < 440 & y >= 40 & y < 440;
assign vertical2 = x >= 520 & x < 600 & y >= 40 & y < 440;

// Combine the appropriate regions into four symbols
wire symbol0, symbol1, symbol2, symbol3;
assign symbol0 = horizontal2 | vertical0 | vertical1 | vertical2;
assign symbol1 = horizontal0 | vertical0 | vertical1 | vertical2;
assign symbol2 = horizontal0 | horizontal1 | horizontal2 | vertical2;
assign symbol3 = horizontal0 | horizontal1 | horizontal2 | vertical0;

// Check if the (x, y) point is covered by the selected symbol
wire covered;
assign covered = symbol == 0 ? symbol0 : symbol == 1 ? symbol1 : symbol == 2 ? symbol2 : symbol3;

// Output the appropriate (r, g, b) color
assign R = covered ? 0 : 255;
assign G = covered ? 0 : 255;
assign B = covered ? 0 : 255;

endmodule
```

*LTM Controller (1/3)

```
module ltm (  
  
    // Define "s0" as "Clock Input"  
    input csi_s0_clk,  
    input csi_s0_reset_n,  
  
    // Define "s1" as "Avalon Memory Mapped Slave"  
    input avs_s1_write_n,  
    input avs_s1_read_n,  
    input [1:0] avs_s1_writedata,  
    output reg [1:0] avs_s1_readdata,  
  
    // Define "s2" as "Conduit"  
    output coe_s2_export_NCLK,  
    output coe_s2_export_GRST,  
    output coe_s2_export_HD,  
    output coe_s2_export_VD,  
    output coe_s2_export_DEN,  
    output [7:0] coe_s2_export_R,  
    output [7:0] coe_s2_export_G,  
    output [7:0] coe_s2_export_B  
);
```

*LTM Controller (2/3)

```
// Implement only one interface register
always @ (posedge csi_s0_clk or negedge csi_s0_reset_n) begin
    if (~csi_s0_reset_n) begin
        avs_s1_readdata <= 0;
    end else if (~avs_s1_write_n) begin
        avs_s1_readdata <= avs_s1_writedata;
    end else begin
        avs_s1_readdata <= avs_s1_readdata;
    end
end
end
```

*LTM Controller (3/3)

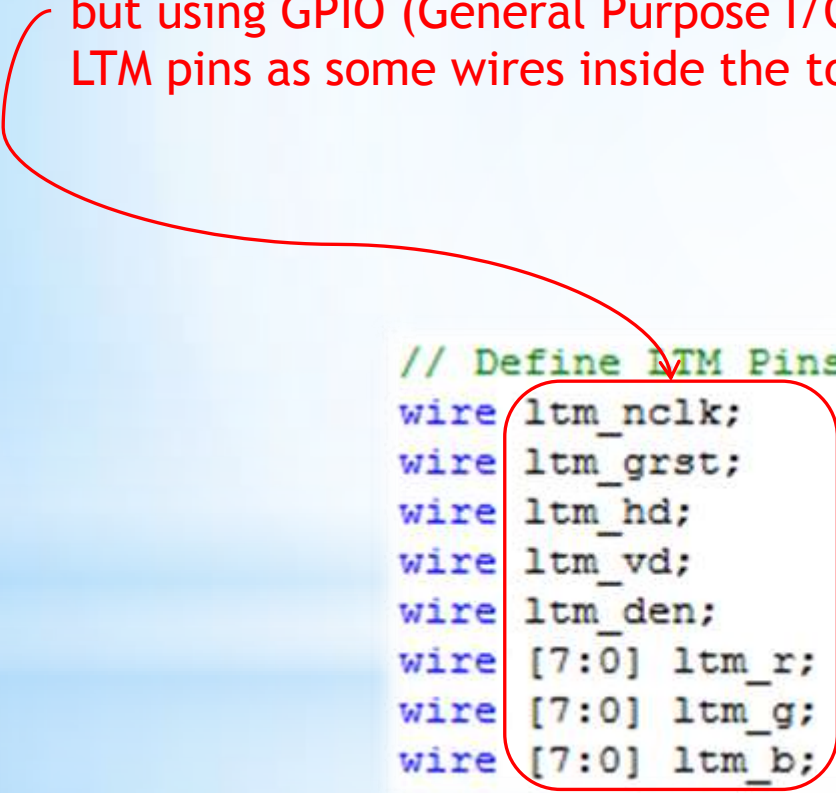
```
// Get the current (x, y) point from timing controller
wire [31:0] x;
wire [31:0] y;
timing_controller u0(
    .HD(coe_s2_export_HD),
    .VD(coe_s2_export_VD),
    .DEN(coe_s2_export_DEN),
    .NCLK(coe_s2_export_NCLK),
    .GRST(coe_s2_export_GRST),
    .x(x),
    .y(y),
    .clock(csi_s0_clk),
    .reset(csi_s0_reset_n)
);

// Use the current (x, y) point to generate (r, g, b) for LTM
pattern_generator u1(
    .R(coe_s2_export_R),
    .G(coe_s2_export_G),
    .B(coe_s2_export_B),
    .x(x),
    .y(y),
    .symbol(avs_s1_readdata)
);

endmodule
```


*Top Module Connection (1/3)

Because in top module, LTM pins are not defined in the I/O pins of top module, but using GPIO (General Purpose I/O) of top module, for convenience, I define LTM pins as some wires inside the top module.



```
// Define LTM Pins for using LTM_controller
wire ltm_nclk;
wire ltm_grst;
wire ltm_hd;
wire ltm_vd;
wire ltm_den;
wire [7:0] ltm_r;
wire [7:0] ltm_g;
wire [7:0] ltm_b;
```

*Top Module Connection (2/3)

And then, I convert my LTM pins to GPIO pins of top module.



```
// Assign LTM pins for real LTM I/O
assign GPIO_0[0] = 1'bz;
assign GPIO_0[1] = 1'bz;
assign GPIO_0[2] = 1'bz;
assign GPIO_0[3] = ltm_b[3];
assign GPIO_0[4] = ltm_b[2];
assign GPIO_0[5] = ltm_b[1];
assign GPIO_0[6] = ltm_b[0];
assign GPIO_0[7] = ltm_nclk;
assign GPIO_0[8] = ltm_den;
assign GPIO_0[9] = ltm_hd;
assign GPIO_0[10] = ltm_vd;
assign GPIO_0[11] = ltm_b[4];
assign GPIO_0[12] = ltm_b[5];
assign GPIO_0[13] = ltm_b[6];
assign GPIO_0[14] = ltm_g[0];
assign GPIO_0[15] = ltm_g[2];
assign GPIO_0[16] = ltm_g[3];
assign GPIO_0[17] = ltm_g[4];
assign GPIO_0[18] = ltm_g[5];
assign GPIO_0[19] = ltm_g[6];
assign GPIO_0[20] = ltm_g[7];
assign GPIO_0[21] = ltm_r[0];
assign GPIO_0[22] = ltm_r[1];
assign GPIO_0[23] = ltm_r[2];
assign GPIO_0[24] = ltm_r[3];
assign GPIO_0[25] = ltm_r[4];
assign GPIO_0[26] = ltm_r[5];
assign GPIO_0[27] = ltm_r[6];
assign GPIO_0[28] = ltm_r[7];
assign GPIO_0[29] = ltm_grst;
assign GPIO_0[30] = 1'bz;
assign GPIO_0[31] = 1'bz;
assign GPIO_CLKOUT_NO = ltm_b[7];
assign GPIO_CLKOUT_PO = ltm_g[1];
```


*Top Module Connection (3/3)

```
// For seven_segment_displays_controller:
.coe_s2_export_oHEX0_DP_from_the_seven_segment_displays_controller(oHEX0_DP),
.coe_s2_export_oHEX0_D_from_the_seven_segment_displays_controller(oHEX0_D),
.coe_s2_export_oHEX1_DP_from_the_seven_segment_displays_controller(oHEX1_DP),
.coe_s2_export_oHEX1_D_from_the_seven_segment_displays_controller(oHEX1_D),
.coe_s2_export_oHEX2_DP_from_the_seven_segment_displays_controller(oHEX2_DP),
.coe_s2_export_oHEX2_D_from_the_seven_segment_displays_controller(oHEX2_D),
.coe_s2_export_oHEX3_DP_from_the_seven_segment_displays_controller(oHEX3_DP),
.coe_s2_export_oHEX3_D_from_the_seven_segment_displays_controller(oHEX3_D),
.coe_s2_export_oHEX4_DP_from_the_seven_segment_displays_controller(oHEX4_DP),
.coe_s2_export_oHEX4_D_from_the_seven_segment_displays_controller(oHEX4_D),
.coe_s2_export_oHEX5_DP_from_the_seven_segment_displays_controller(oHEX5_DP),
.coe_s2_export_oHEX5_D_from_the_seven_segment_displays_controller(oHEX5_D),
.coe_s2_export_oHEX6_DP_from_the_seven_segment_displays_controller(oHEX6_DP),
.coe_s2_export_oHEX6_D_from_the_seven_segment_displays_controller(oHEX6_D),
.coe_s2_export_oHEX7_DP_from_the_seven_segment_displays_controller(oHEX7_DP),
.coe_s2_export_oHEX7_D_from_the_seven_segment_displays_controller(oHEX7_D),
```

```
// For LTM controller:
.coe_s2_export_B_from_the_LTM_controller(ltm_b),
.coe_s2_export_DEN_from_the_LTM_controller(ltm_den),
.coe_s2_export_GRST_from_the_LTM_controller(ltm_grst),
.coe_s2_export_G_from_the_LTM_controller(ltm_g),
.coe_s2_export_HD_from_the_LTM_controller(ltm_hd),
.coe_s2_export_NCLK_from_the_LTM_controller(ltm_nclk),
.coe_s2_export_R_from_the_LTM_controller(ltm_r),
.coe_s2_export_VD_from_the_LTM_controller(ltm_vd),
```

```
// System reset:
.reset_n(cpu_reset_n)
```

```
);
```

```
endmodule
```

I need to connect my IP's exported signals (Conduit) to LTM pins. These wires are not auto-generated by SOPC Builder.

* A system integrated with LTM

- * In Lab3, you need to design a SOPC system running four functions at the same time (using $\mu\text{C}/\text{OS-II}$). A global string `professor_name` stores a certain professor's name (e.g. Huang Shi Yu).
- * Function 1: Every 1 second, read `professor_name` and print its value to Character LCD. Also:
 - * If `professor_name` starts with "H", turn image $+90^\circ$ (clockwise).
 - * If `professor_name` starts with "C", turn image -90° (counterclockwise).
 - * If `professor_name` starts with "L", turn image 180° (mirror).
- * Function 2: Every 100 milliseconds, read `professor_name` and print its length to Seven Segment Display.
- * Function 3: Every 10 milliseconds, read `professor_name` and let only i -th red LED blink where i is equal to the length of `professor_name`.
- * Function 4: Open a text file `"/mnt/rozipfs/professor.txt"`. Read file line by line and store each line to `professor_name` with replacement. If the position is EOF (End Of File), set it back to the beginning of the file and repeat this reading routine. Noticed that the Function 4 need to perform this routine anytime. In other words, no `OSTimeDlyHMSM`!

*LTM Controller Driver (1/2)

```
#ifndef __LTM_H__  
#define __LTM_H__
```

```
#include <io.h>
```

```
// =====  
// Register Map  
// =====
```

```
#define SYMBOL 0
```

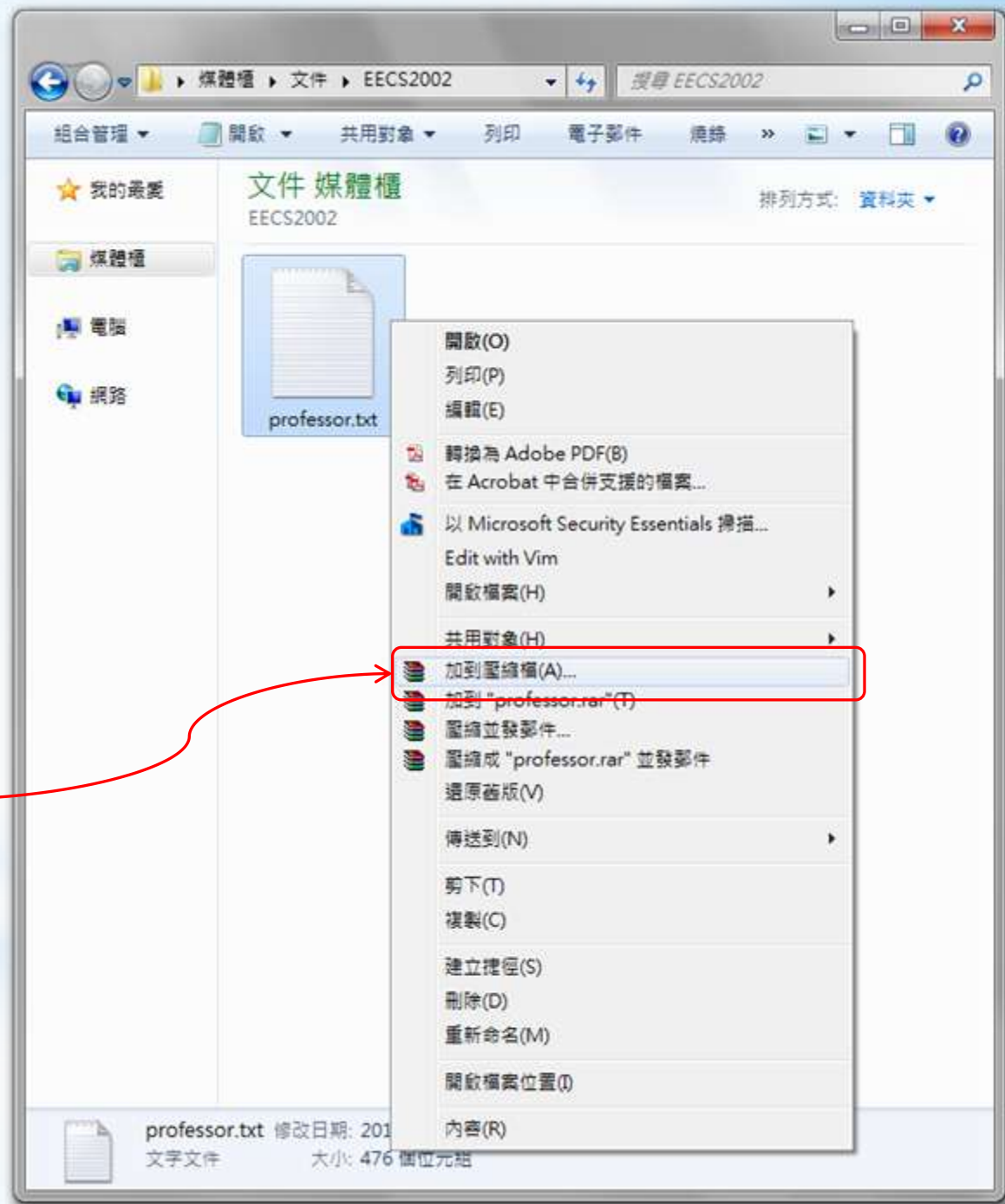

*LTM Controller Driver (2/2)

```
// =====  
// Hardware Abstraction  
// =====  
  
void turn_right_ltm(void *base)  
{  
    int value = IORD(base, SYMBOL);  
    if (value == 0) IOWR(base, SYMBOL, 3);  
    else if (value == 1) IOWR(base, SYMBOL, 2);  
    else if (value == 2) IOWR(base, SYMBOL, 0);  
    else IOWR(base, SYMBOL, 1);  
}  
  
void turn_back_ltm(void *base)  
{  
    int value = IORD(base, SYMBOL);  
    if (value == 0) IOWR(base, SYMBOL, 1);  
    else if (value == 1) IOWR(base, SYMBOL, 0);  
    else if (value == 2) IOWR(base, SYMBOL, 3);  
    else IOWR(base, SYMBOL, 2);  
}  
  
void turn_left_ltm(void *base)  
{  
    int value = IORD(base, SYMBOL);  
    if (value == 0) IOWR(base, SYMBOL, 2);  
    else if (value == 1) IOWR(base, SYMBOL, 3);  
    else if (value == 2) IOWR(base, SYMBOL, 1);  
    else IOWR(base, SYMBOL, 0);  
}  
  
#endif
```

*Read-Only Zip File System

- * You can see that in Lab 3, you may need a method to store some files or directories into DE2-70 Board just like a Hard Disk in PC. The only NVM (Non-Volatile Memory) on DE2-70 Board is Flash.
- * However, both Hard Disk and Flash are just a lot of non-volatile bits that can hold values when the power is off. To treat these bits as some files and directories, it needs some complicated data structures and algorithms to make sure that you can freely create or delete or move or edit any files or directories without damaging other files and directories. This mechanism is called File System. In your PC, the File System of Hard Disk is NTFS by Microsoft Corporation. In DE2-70 Board, the File System of Flash is ROZIPFS (Read-Only Zip File System) by Altera Corporation.
- * ROZIPFS is very simple. It only supports reading from Flash. It cannot write anything to Flash. To use ROZIPFS, you need to burn an uncompressed zip file into Flash. And then, ROZIPFS will treat all the files and directories inside that zip file as a read-only file system. Now you can use C/C++ program to freely read any files and directories inside that zip file.

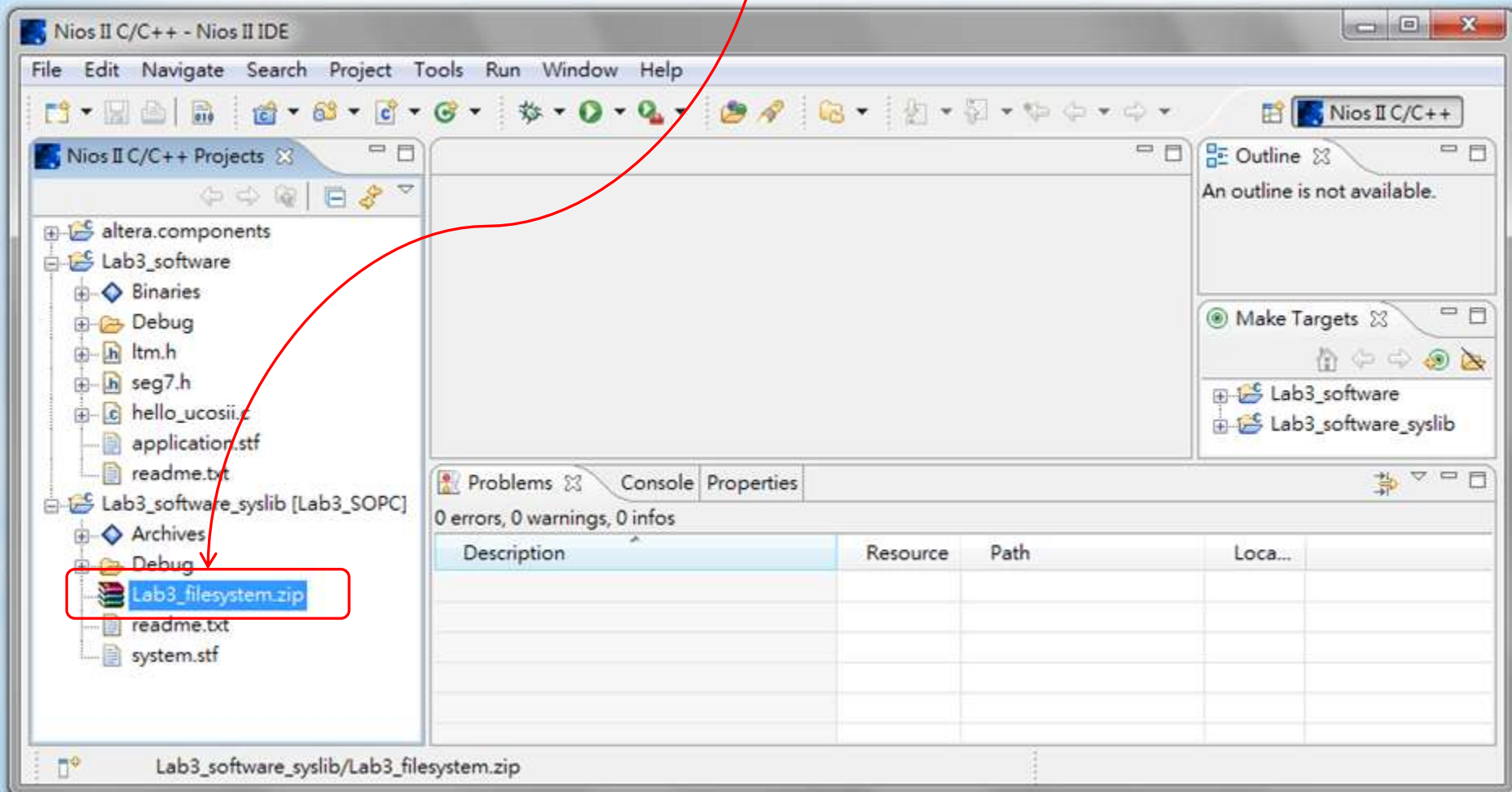
Take WinRAR for an example,
I right-click “professor.txt”
and click “Add to archive...”.



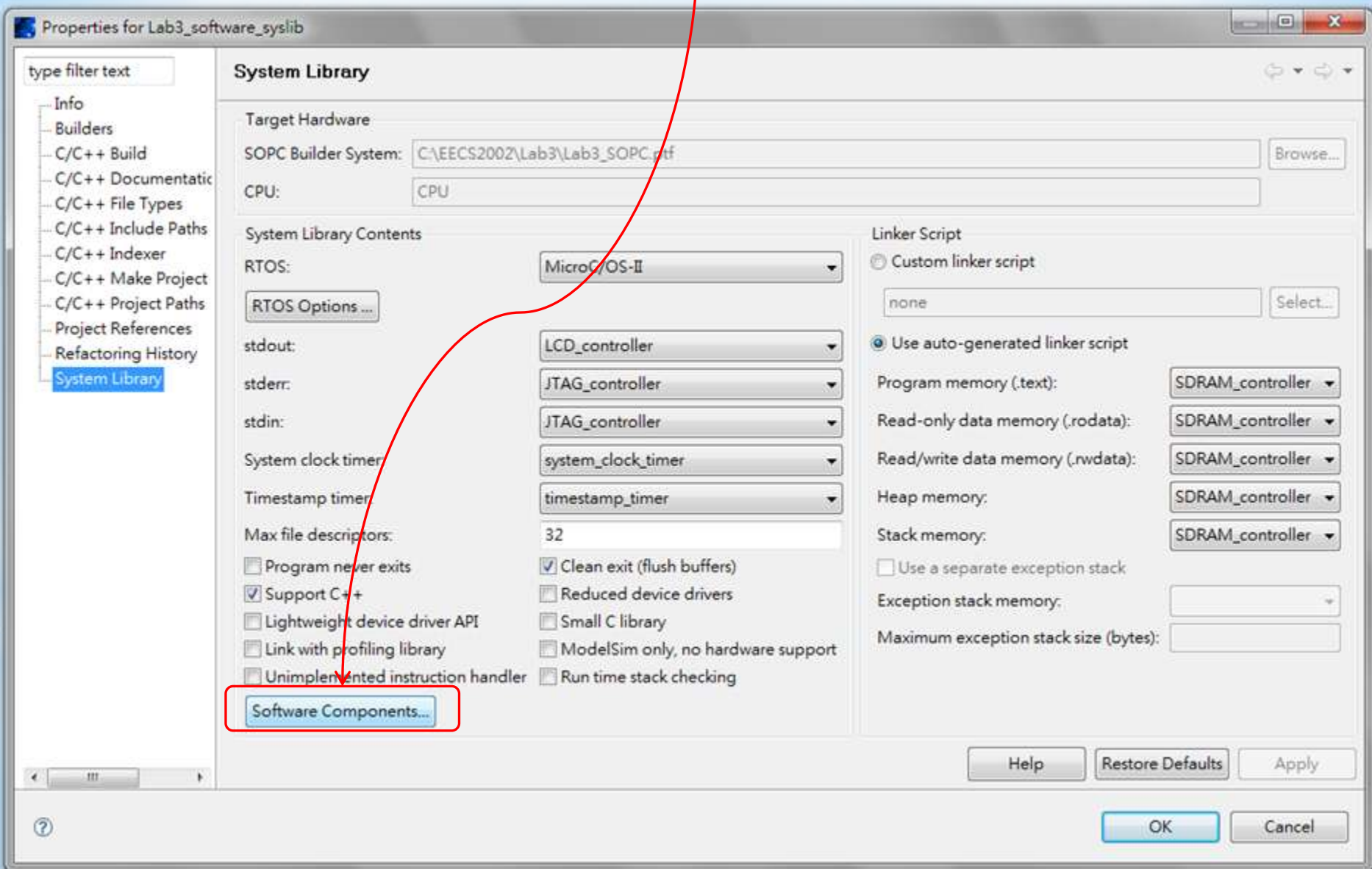
I select “Archive format” as “ZIP” and
“Compression method” as “Store” to
create an uncompressed zip file.



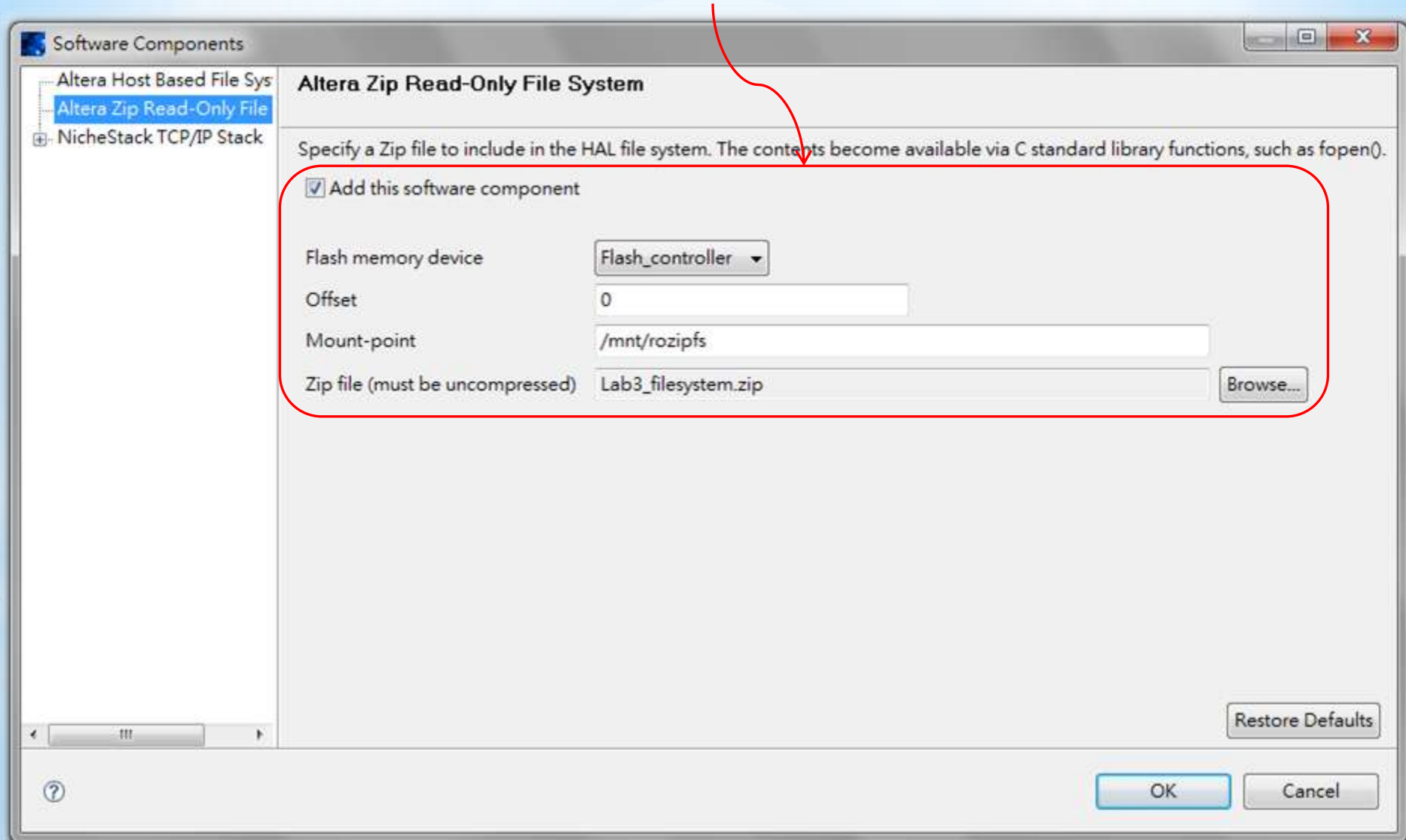
I copy this uncompressed zip file to software core (Lab3_software_syslib).

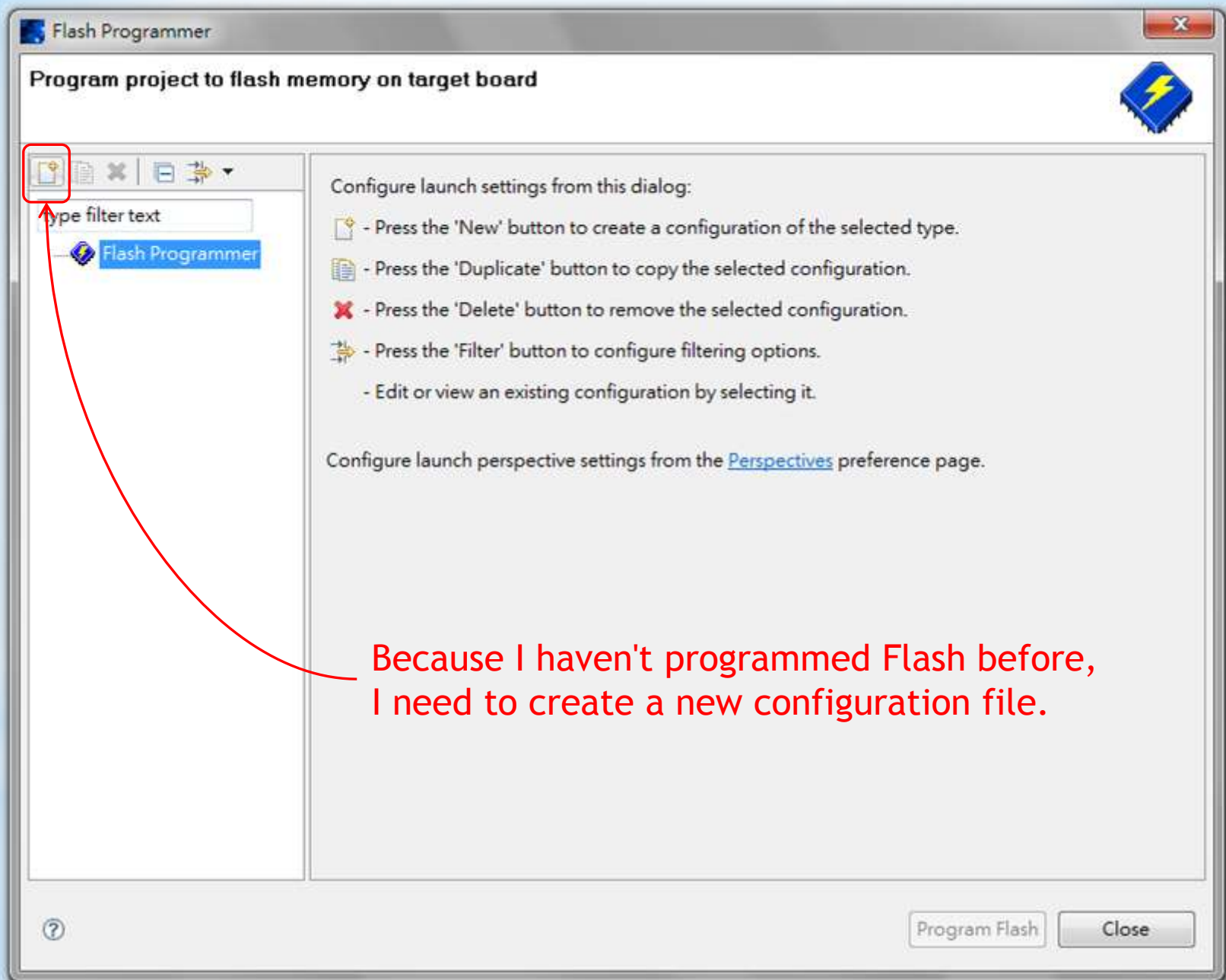


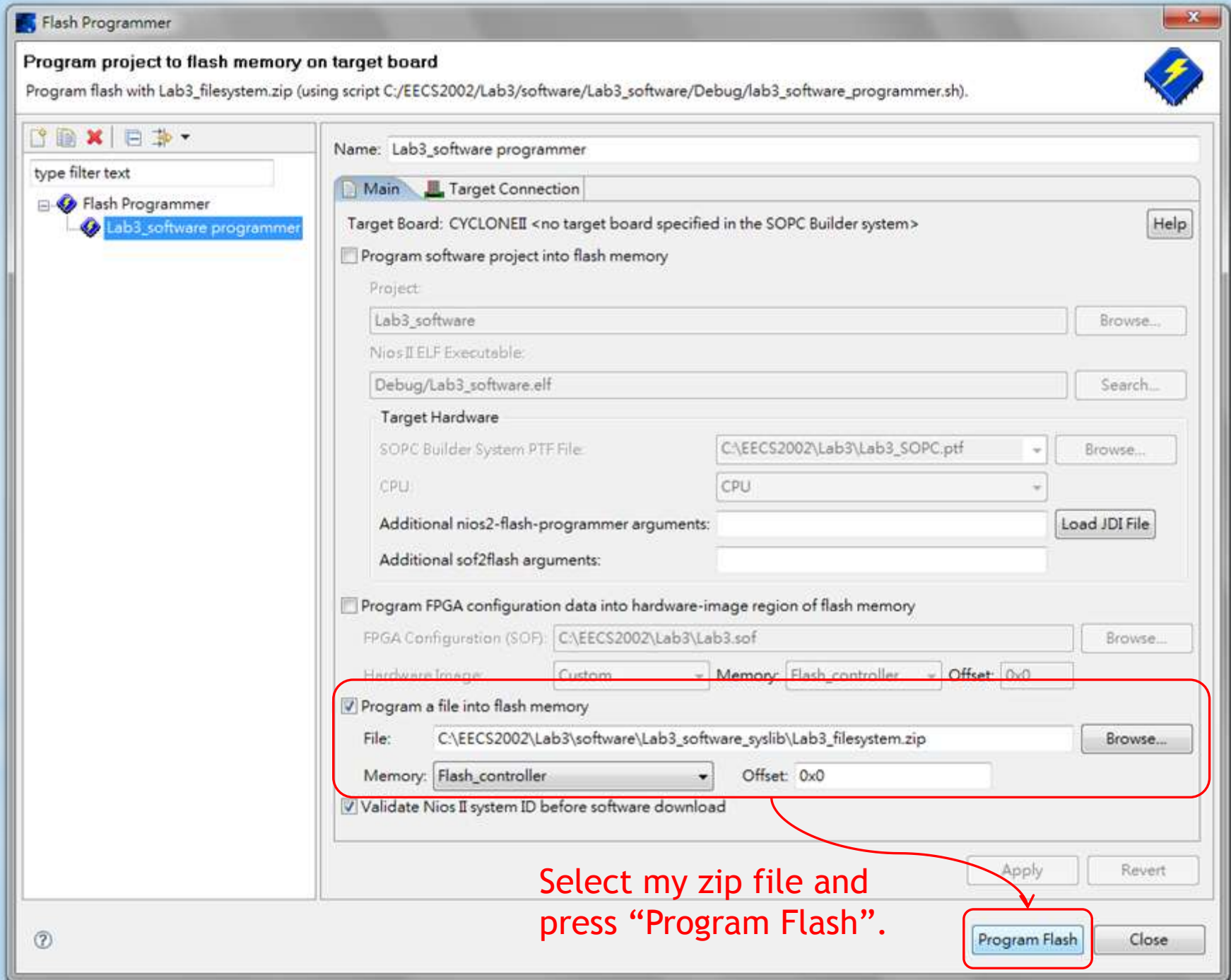
To add ROZIPFS component to software core, simply click “Software Components...”.



I set ROZIPFS to find my Lab3_filesystem.zip in Flash_controller plus zero offset. Now by accessing Mount-point (default setting is “/mnt/rozipfs”), my C/C++ program can access all files and directories in my zip file. For example, if my C/C++ program opens “/mnt/rozipfs/professor.txt”, ROZIPFS will open “professor.txt” inside my zip file.







* Problem when sharing data

- * In Lab 3, you may see the wrong result in Character LCD if you just let all concurrent functions to freely access shared variable professor_name.
- * Assume the original professor_name is “Huang Shi Yu”. Now Function 4 is writing “Ma Hsi Pin” to professor_name but hasn’t finished yet because an interrupt is happened so CPU immediately goes back to μ C/OS-II.
- * Assume a higher-priority Function 1 is not blocked by OSTimeDlyHMSM. μ C/OS-II saves CPU status of Function 4 and loads CPU status of Function 1. Function 1 reads professor_name and prints its value to Character LCD.
- * Because “Huang Shi Yu” are not fully replaced by “Ma Hsi Pin”, you will see strange string on Character LCD such as “Maang Shi Yu”, “Ma Hsi hi Yu” and so on.
- * You may try to fix this problem by setting Function 1 with a lower priority. However, another problem will occur.
- * When Function 1 is not fully reading “Huang Shi Yu”, it may be interrupted by Function 4 and Function 4 writes “Ma Hsi Pin”. When Function 1 is back, it will continue reading “Huang Shi Yu” but get another wrong result such as “Hu Hsi Pin”, “Huangi Pin” and so on.

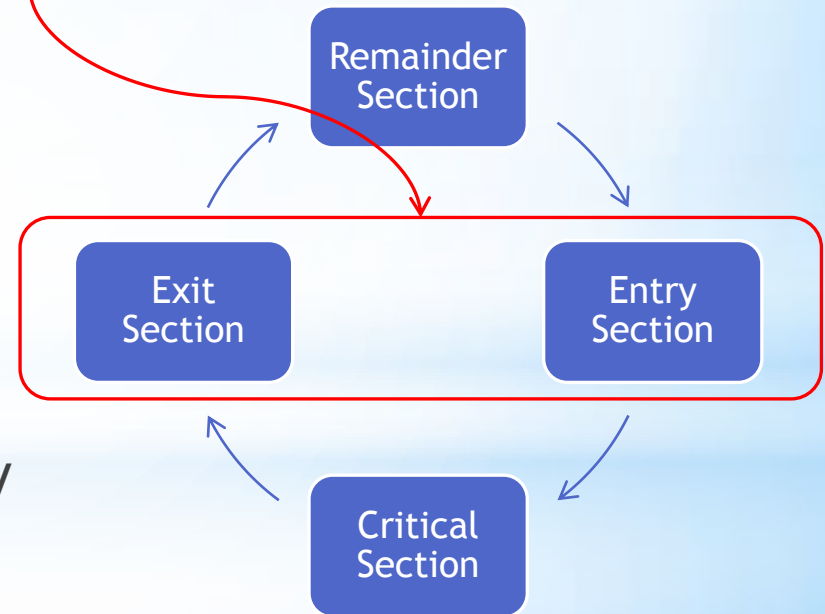
*Critical Section in Multi-task

- * You can see that in multi-tasking program, **when a task want to access some data shared by other tasks, it must guarantee that no other tasks are accessing the same data at the same time.** Otherwise, it may cause unexpected results (when writing and reading are simultaneously happened) and even damage the data (if more than one writing are simultaneously happened).
- * When a task has some codes which access the shared memory, people said that these codes are in Critical Section of this task. If one task are running in its Critical Section, other tasks which want to enter their Critical Section must wait. In other words, you can only allow one task in its Critical Section.
- * With the idea of Critical Section, problem discussed at page 32 will never happens. When Function 4 is writing professor_name but interrupted by Function 1, because Function 4 hasn't leave its Critical Section, Function 1 cannot enter its Critical Section to read professor_name and must wait (OSTimeDlyHMSM) until Function 4 leaves its Critical Section.

*Will this structure always work?

```
bool lock = false ;
void example ( void *p ) {
    while ( true ) {
        /* ===== Remainder Section ===== */
        printf ( “Dear professor ” ) ;
        /* ===== Entry Section ===== */
        while ( lock ) {
            OSTimeDlyHMSM ( 0 , 0 , 0 , 1 ) ;
        }
        lock = true ;
        /* ===== Critical Section ===== */
        printf ( “%s” , professor_name ) ;
        /* ===== Exit Section ===== */
        lock = false ;
        /* ===== Remainder Section ===== */
        printf ( “, nice to see you!\n” ) ;
    }
}
```

For any Critical Section, you can add Entry Section before it, also, add Exit Section after it.

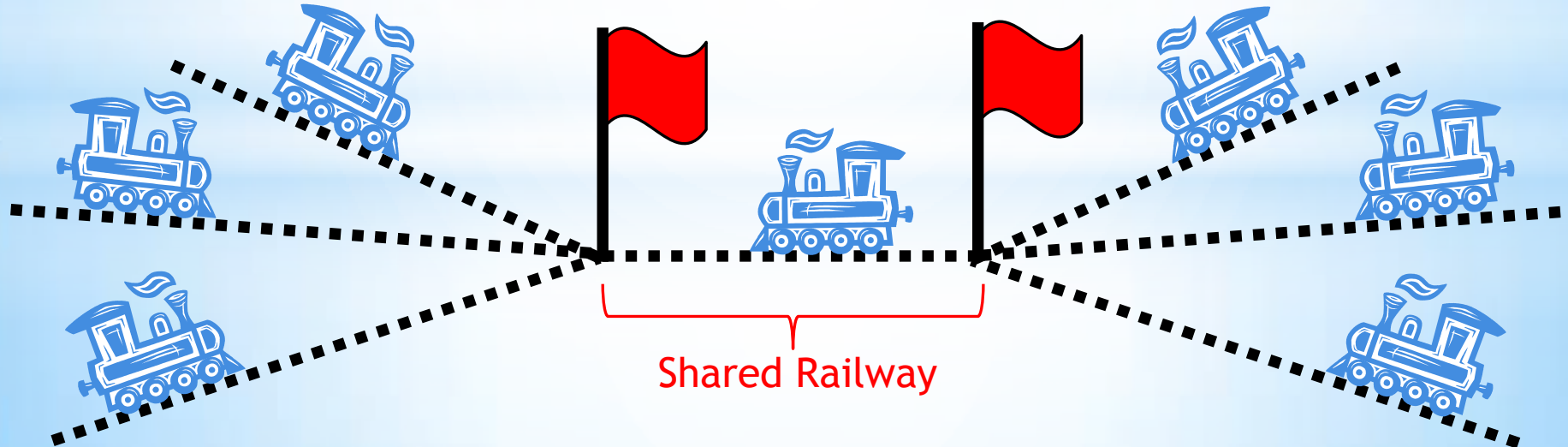


* No! Interrupt can be anywhere!

- * Assume Task A want to enter its Critical Section. It will be blocked by the while loop until the lock value is reset to false by other task. However, there is no any guarantee that after Task A bypasses the while loop, it will immediately set the lock value to true. Why?
- * Assume in this time, a higher priority Task B also want to enter its Critical Section. After Task A bypasses while loop, it's interrupted by Task B unfortunately. Because the lock value is false, Task B can easily enter its Critical Section. Assume now Task B is sleeping by calling OSTimeDlyHMSM. μ C/OS-II will let Task A continue to run. Because Task A has already bypassed its while loop, it will directly set the lock value to true (even if the lock value is already set to true by Task B) and enter its Critical Section. Now, more than one task are in their Critical Section!
- * The problem of the structure in page 34 is: You cannot guarantee that Entry Section and Exit Section won't be interrupted. Noticed that interrupt can be happened anywhere in your codes.

* Semaphore in $\mu\text{C}/\text{OS-II}$

- * In order to solve the programming problem of Critical Section, most of the modern Operating System such as Windows, Solaris and Linux will provide programmers a simple tool: Semaphore.
- * In railway history, Semaphore is used to prevent multiple trains go into shared railway. A train needs to pend a semaphore before it enters the railway and post a semaphore after it exits the railway. Noticed that semaphores in two sides of the railway are connected so they have the same pended/posted status.
- * In $\mu\text{C}/\text{OS-II}$, Semaphore can limit the maximum number of tasks in their Critical Section.



***Semaphore:** **Pended/Posted**

```
OS_EVENT *lock = OSSemCreate ( 1 );  
void example ( void *p ) {  
    while ( true ) {  
        /* ===== Remainder Section ===== */  
        printf ( "Dear professor " );  
        /* ===== Entry Section ===== */  
        unsigned char result ;  
        OSSemPend ( lock , 0 , &result );  
        /* ===== Critical Section ===== */  
        printf ( "%s" , professor_name );  
        /* ===== Exit Section ===== */  
        OSSemPost ( lock );  
        /* ===== Remainder Section ===== */  
        printf ( ", nice to see you!\n" );  
    }  
}
```

At the beginning, you set only 1 posted semaphore. Noticed that the number of pended semaphores may be large but you only care about posted semaphores.

When a task enters its Critical Section, it pends 1 posted semaphore. In other words, the number of posted semaphores is subtracted 1. If there's no posted semaphores, it waits.

When a task leaves its Critical Section, it posts 1 pended semaphore. In other words, the number of posted semaphores is added 1.

*Lab 3 Software (1/6)

I defined a semaphore named `professor_key` guarding a shared string `professor_name`.

```
#include <io.h>
#include <stdio.h>
#include <string.h>
#include "includes.h"
#include "system.h"
#include "seg7.h"
#include "ltn.h"
char professor_name[100];
OS_EVENT *professor_key;
OS_STK s1[2048], s2[2048], s3[2048], s4[2048], s5[2048];
```

*Lab 3 Software (2/6)

In Critical Section, I print professor_name to Character LCD.
Also, I turn the image on LTM according to the first character of professor_name (H = +90°, C = -90°, L = 180°).

```
void f1(void *p) {  
    while (1) {  
        unsigned char e;  
        OSSemPend(professor_key, 0, &e);  
        printf("%s", professor_name);  
        if (professor_name[0] == 'H') {  
            turn_right_ltm((void*) LTM_CONTROLLER_BASE);  
        } else if (professor_name[0] == 'C') {  
            turn_left_ltm((void*) LTM_CONTROLLER_BASE);  
        } else if (professor_name[0] == 'L') {  
            turn_back_ltm((void*) LTM_CONTROLLER_BASE);  
        }  
        OSSemPost(professor_key);  
        OSTimeDlyHMSM(0, 0, 1, 0);  
    }  
}
```

*Lab 3 Software (3/6)

In Critical Section, I print the string length of professor_name to Seven Segment Display.

```
void f2(void *p) {  
    while (1) {  
        unsigned char e;  
        OSSemPend(professor_key, 0, &e);  
        set_seg7((void*) SEVEN_SEGMENT_DISPLAYS_CONTROLLER_BASE, strlen(professor_name));  
        OSSemPost(professor_key);  
        OSTimeDlyHMSM(0, 0, 0, 100);  
    }  
}
```


*Lab 3 Software (4/6)

In Critical Section, I set only i-th red LED blink where i is equal to the string length of professor_name.

```
void f3(void *p) {  
    while (1) {  
        unsigned char e;  
        OSSemPend(professor_key, 0, &e);  
        IOWR(RED_LEDS_CONTROLLER_BASE, 0, 1 << (strlen(professor_name) - 1));  
        OSSemPost(professor_key);  
        OSTimeDlyHMSM(0, 0, 0, 10);  
    }  
}
```

*Lab 3 Software (5/6)

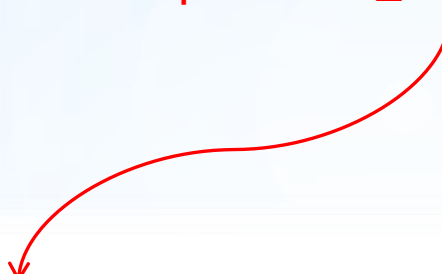
Because in my main function, I initialize professor_key with 0 posted semaphore. Therefore, all the tasks cannot enter their Critical Section. After I open the file, I post a semaphore to allow at most 1 task to enter its Critical Section.

```
void f4(void *p) {  
    printf("EECS2002 Lab3\nName: 9760111\n");  
    OSTimeDlyHMSM(0, 0, 3, 0);  
    FILE *f = fopen("/mnt/rozipfs/professor.txt", "r");  
    if (f == NULL) {  
        printf("No professor.txt\nin /mnt/rozipfs/\n");  
    } else {  
        OSSemPost(professor_key);  
        while (1) {  
            unsigned char e;  
            OSSemPend(professor_key, 0, &e);  
            if (fgets(professor_name, 100, f) == NULL) {  
                fseek(f, 0, SEEK_SET);  
            }  
            OSSemPost(professor_key);  
        }  
    }  
}
```

In Critical Section, I read the file line by line and store each line to professor_name with replacement. If the position is EOF (End Of File), I set it back to the beginning of the file.

*Lab 3 Software (6/6)

Because I don't want any task to enter their Critical Section before I open professor.txt, I initialize professor_key with 0 posted semaphore.



```
int main() {  
    professor_key = OSSemCreate(0);  
    OSTaskCreateExt(f1, 0, (void*) &s1[2047], 1, 1, s1, 2048, 0, 0);  
    OSTaskCreateExt(f2, 0, (void*) &s2[2047], 2, 2, s2, 2048, 0, 0);  
    OSTaskCreateExt(f3, 0, (void*) &s3[2047], 3, 3, s3, 2048, 0, 0);  
    OSTaskCreateExt(f4, 0, (void*) &s4[2047], 4, 4, s4, 2048, 0, 0);  
    OSStart();  
    return 0;  
}
```

*Review Questions

1. In Lab 3, LTM Controller use a Pattern Generator to convert (x, y) into (R, G, B) . Assume you want to show a $800 * 480$ full-color (Each pixel is 24 bits) photo on LTM. Is it possible to design a Pattern Generator that can store entire photo? Why? (Hint: 250 M4Ks)
2. There are 3 tasks. Task 1 has a higher priority than Task 2. Task 3 has a lower priority than Task 2. A data is shared with Task 1 and Task 3. Explain that the semaphores in Task 1 and Task 3 may cause “Priority Inversion” in Task 1 and Task 2. (Hint: Find a situation that Task 1 is waiting for Task 3, but Task 3 is waiting for Task 2)