Student ID : 111062307,  Name : 陳大佑

```
__data __at (0x39) char buffer;

__data __at (0x3A) char mutex;
__data __at (0x3B) char full;
__data __at (0x3C) char iter;

__data __at (0x3E) char prev_input;
__data __at (0x3F) char cur_input;
```

```
__data __at (0x20) char dino_pos;
__data __at (0x2E) char score;
__data __at (0x23) char game_over;
__data __at (0x24) char i;
__data __at (0x25) char j;
__data __at (0x26) char k;
__data __at (0x29) char cactus_pos[5];
__data __at (0x27) char pre_score;
__data __at (0x28) char flag;
__data __at (0x2F) char diff;
```

```
__data __at (0x3D) unsigned char lcd_ready;
__data __at (0x30) char saved_SP[MAXTHREADS];
__data __at (0x34) ThreadID cur_ID;
__data __at (0x35) char bitmap_ID;
__data __at (0x36) char
__data __at (0x37) Threa  char producer
__data __at (0x38) char producer;
```

Set some parameters on manually allocated memory.

In fact, I use all of memory from 20-2F and 30-3F

```
void keypad_ctrl(void) {
    while (1) {
    if (game_over) {ThreadExit(); return;}
    if (AnyButtonPressed()) {
        cur_input = ButtonToChar();
        if (cur_input != prev_input && prev_input == 0x00) {
            EA = 0;
            buffer = cur_input;
            prev_input = cur_input;
            EA = 1;
        }
    }
    else if (AnyKeyPressed()) {
        cur_input = KeyToChar();
        if (cur_input != prev_input && prev_input == 0x00) {
            EA = 0;
            buffer = cur_input;
            if (buffer == '2' ) { // Move up
                LCD_cursorGoTo(dino_pos, 0);
                LCD_write_char(' ');
                dino_pos = 0;
                LCD_cursorGoTo(dino_pos, 0); // 固定在第 0 列
                LCD_write_char(0x00);
            } else if (buffer == '8') { // Move down
                LCD_cursorGoTo(dino_pos, 0);
                LCD_write_char(' ');
                dino_pos = 1;
                LCD_cursorGoTo(dino_pos, 0); // 固定在第 0 列
                LCD_write_char(0x00);
            }
            prev_input = cur_input;
            EA = 1;
```

Using keypad_ctrl as a thread to control the button and keypad.

When no pressing any button, prev_input = 0 so we can know we don’t press any button in the moment.

And set cur_inpur != prev_input to prevent multiple output.

```
void render_task(void) {
    //init(); // Initialize game state
    while (1) {
        if (game_over) {ThreadExit(); return;}
        //SemaphoreWait(mutex);
        SemaphoreWait(mutex);
        //LCD_clearScreen();
        //SemaphoreWait(full);
       // SemaphoreWait(mutex);
        // Draw cacti
        EA = 0;
        if(!k){;
                k = 1;
                LCD_cursorGoTo(0, cactus_pos[0]);
                LCD_write_char(' ');
                LCD_cursorGoTo(1, cactus_pos[1]);
                LCD_write_char(
                LCD_cursorGoTo(  (int)32  pos[2]);
                LCD_write_char(' ');
                LCD_cursorGoTo(1, cactus_pos[3]);
                LCD_write_char(' ');
                LCD_cursorGoTo(0, cactus_pos[4]);
                LCD_write_char(' ');

                if(game_over) {return;}
                //LCD_cursorGoTo(0, 0); // 固定在第 0 列
                if (cactus_pos[0] == 0) cactus_pos[0] = 15;
                else cactus_pos[0] = cactus_pos[0] - 1;
                if (cactus_pos[1] == 0) cactus_pos[1] = 15;
                else cactus_pos[1] = cactus_pos[1] - 1;
                if (cactus_pos[2] == 0) cactus_pos[2] = 15;
                else cactus_pos[2] = cactus_pos[2] - 1;
                if (cactus_pos[3] == 0) cactus_pos[3] = 15;
                else cactus_pos[3] = cactus_pos[3] - 1;
                if (cactus_pos[4] == 0) cactus_pos[4] = 15;
                else cactus_pos[4] = cactus_pos[4] - 1;
```

For render_task, it share a flag k with game_ctrl, and every time when flag is off, I left_shitf all cactus_pos[0-4] a unit, and also clear the position it stayed previously.

Using LCD_cursorGoTo(), I can go to the place I want to move or clear.

The range of the LCD is (0,1) and (0,15), so when it becomes to 0, I set it to 15.

```c
void game_ctrl(void) {
    //if(game_over) return;
    init(); // Initialize game state
    while (1) {
        //LCD_setDdRamAddress(0x00);
        if (LCD_ready()) {
            //EA = 0;
            update_cactus();
            //EA = 0;
            flag = 1;
            EA = 0;
            i = 0;
                while(1){
                    if (i == 5){ break;}
                    if((cactus_pos[i] == 0)){
                        if((i-dino_pos) % 2 == 0){
                            game_over = 1;
                            EA = 1;
                            return;
                        }else if(flag){
                            flag = 0;
                        }
                    }
                    i = i + 1;
                }
            i = 0;
            j = 20 - 2*diff;
```

For game_ctrl, it uses to initialize all parameters,

and helps update cactus, and check whether the dino runs into the cactus or not.

```c
        }
        delay(255);
        delay(255);
        delay(255);
        delay(255);
        delay(255);
        //ThreadYield();
        delay(255);
        delay(255);
        delay(255);
        delay(255);
        delay(255);
        //ThreadYield();
        i = i + 1;
        if ((j-i) == 0) break;
    }
```

```c
while(1){
    delay(255); // Control game speed
    delay(255);
    delay(255);
    delay(255);
    delay(255);
    //ThreadYield();
    delay(255);
    delay(255);
    delay(255);
    delay(255);
```

And use for delay the game, so that we can control the difficulty of the game.

If there is a collision happen, game_over will become 1, and it will stop all thread and function, and print the result of score.

Question:

1. In fact, I didn't use any bit for type, since I used some bytes like cactus_pos to record each cactus's position and I used dino_pos to record dino position.

   So maybe it means I use bytes for the map.

2. I use the cactus_pos to record all the position the cactus be, and generate them. It will always be the same row which I defined when initialized.

```c
LCD_cursorGoTo(0, cactus_pos[0]);
LCD_write_char(0x01);
LCD_cursorGoTo(1, cactus_pos[1]);
LCD_write_char(0x01);
LCD_cursorGoTo(0, cactus_pos[2]);
LCD_write_char(0x01);
LCD_cursorGoTo(1, cactus_pos[3]);
LCD_write_char(0x01);
LCD_cursorGoTo(0, cactus_pos[4]);
LCD_write_char(0x01);
```

```c
    }
i = 0;
j = 20 - 2*diff;
```

3. As I mentioned in previous, the level of difficulty I set use delay and the loop of while which has a formula of (20 – 2*diff)

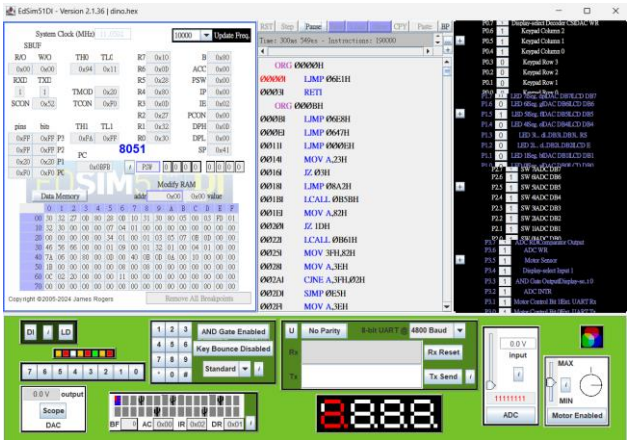
2.4 : In my code, there is a race condition between game_ctrl and render_task in order to race the flag k, so that render_task can update the position and render_task can check whether the dino is with collision of cactus or not.

*Screenshots for compliation :*

```
dylan@LAPTOP-S5FOLV5V ~/os/ppc5/dino
$ make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: cannot remove '*.ihx': No such file or directory
rm: cannot remove '*.lnk': No such file or directory
make: *** [clean] Error 1

dylan@LAPTOP-S5FOLV5V ~/os/ppc5/dino
$ make
sdcc -c  --model-small dino.c
sdcc -c  --model-small preemptive.c
preemptive.c:164: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc -c  --model-small lcdlib.c
lcdlib.c:86: warning 85: in function delay unreferenced function argument : 'n'
sdcc -c  --model-small buttonlib.c
sdcc -c  --model-small keylib.c
sdcc -o dino.hex dino.rel preemptive.rel lcdlib.rel buttonlib.rel keylib.rel
```

screen shot of the beginning



screen shot of the ending.