Student ID : 111062307,　　Name : 陳大佑

```
__data __at (0x30) char saved_SP[MAXTHREADS];
__data __at (0x34) ThreadID cur_ID;
__data __at (0x35) char bitmap_ID;
__data __at (0x36) char temp_SP;
__data __at (0x37) ThreadID new_ID;
__data __at (0x38) char producer;
```

```
__data __at (0x39) char head;
__data __at (0x3A) char tail;
__data __at (0x3B) char nextChar;
__data __at (0x3C) char nextNum;
__data __at (0x3D) char buffer[3];

__data __at (0x20) Semaphore mutex;
__data __at (0x21) Semaphore full;
__data __at (0x22) Semaphore empty;
```

Set some parameters on manually allocated memory.

```
void myTimer0Handler(void) {
    EA = 0;
        SAVESTATE;
        //P1 = cur_ID;
        if(&bitmap_ID) {
            P1 = cur_ID;
            if (cur_ID == 0) {
                if(producer == 1) {
                    cur_ID = 1;
                    producer = 0;
                }
                else if(producer == 0){
                    cur_ID = 2;
                    producer = 1;
                }
            }
            else{
                cur_ID = 0;
            }
        }
        RESTORESTATE;
    EA = 1;
    __asm
        RETI
    __endasm;
}
```

Use P1 for debugging.

EA = 0; marks the start of a **critical section**.

If the current thread is the Consumer (cur_ID == 0):

Then choose the next thread according to the producer

Else if current thread is not consumer then switch to consumer

EA = 1; End of critical section

Screenshots for compliation :

```
dylan@LAPTOP-S5FOLV5V ~/os/ppc4
$ 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/ppc4
$ make
sdcc -c  test3threads.c
sdcc -c  preemptive.c
preemptive.c:164: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc  -o test3threads.hex test3threads.rel preemptive.rel
```
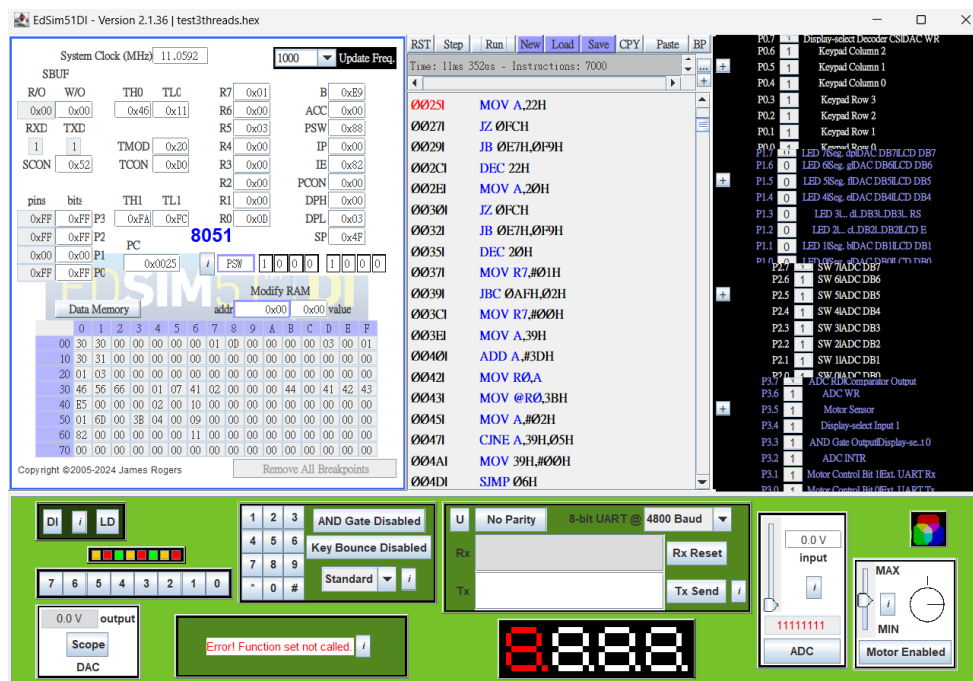
Screenshots and explanation:

Cur_ID (34H) is 1, which means producer1 is running.

And now our mutex = 0, full = 1, empty = 1。
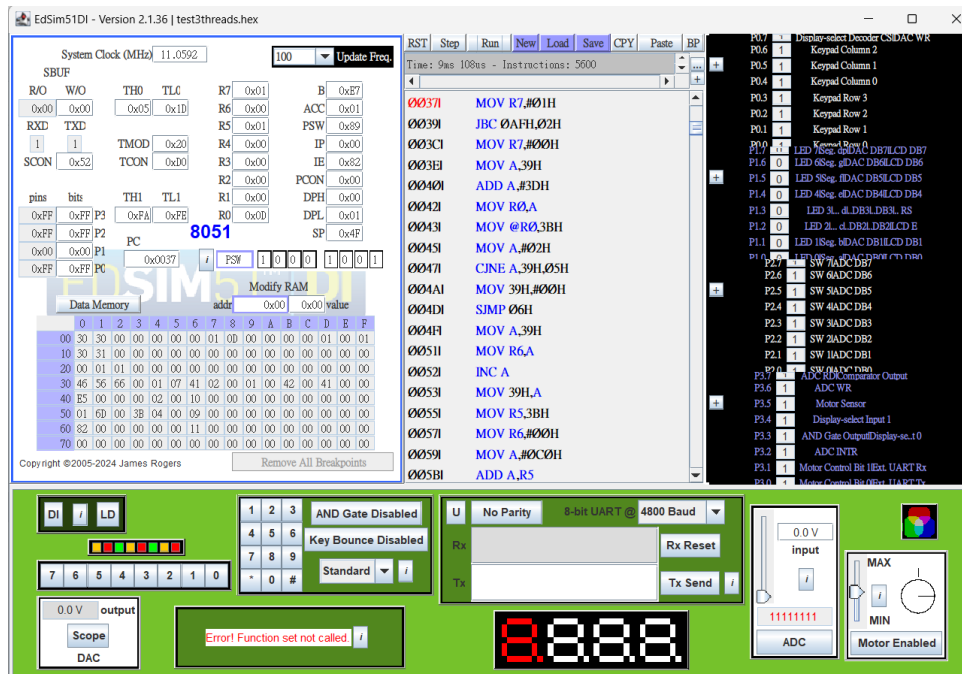
After 1400 iteration, cur_ID(34H) is still 1

But now our mutex = 1, full = 3, empty = 0。 Which means semaphore changes

Cur_ID (34H) is 2, which means producer2 is running.

And now our mutex = 0, full = 0, empty = 2。

After 1000 iteration, cur_ID(34H) is still 2

But now our mutex = 1, full = 3, empty = 0。 Which means semaphore changes

Cur_ID (34H) is 0, which means consumer is running.

And now our mutex = 0, full = 1, empty = 1。

After 1900 iteration, cur_ID(34H) is still 0

But now our mutex = 1, full = 0, empty = 3。 Which means semaphore changes

Fair Version:

At the start, it is producer1's turn. Producer1 produces 3 alphabets, after which the consumer outputs these 3 alphabets. Then, the turn switches to producer2, which produces 3 numbers, and the consumer outputs these 3 numbers. The two producers alternate turns after producing 3 alphabets or numbers each.
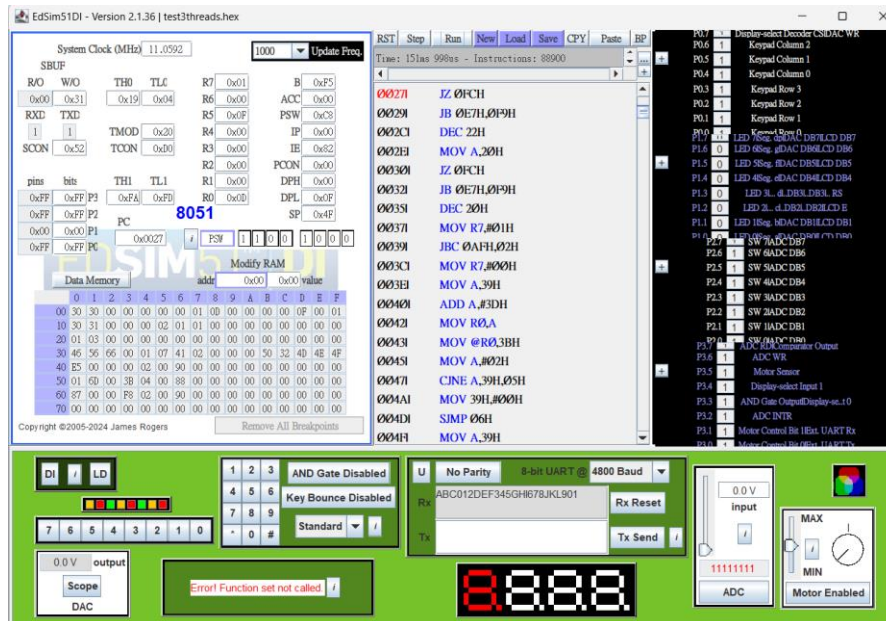
Unfair Version:

At the start, it is producer1's turn. Producer1 produces 3 alphabets, and then the turn switches to producer2. However, since mutex = 1, full = 3, and empty = 0, the buffer is already full. As a result, when the turn switches to the consumer, it can only output 3 alphabets and cannot output 3 numbers.