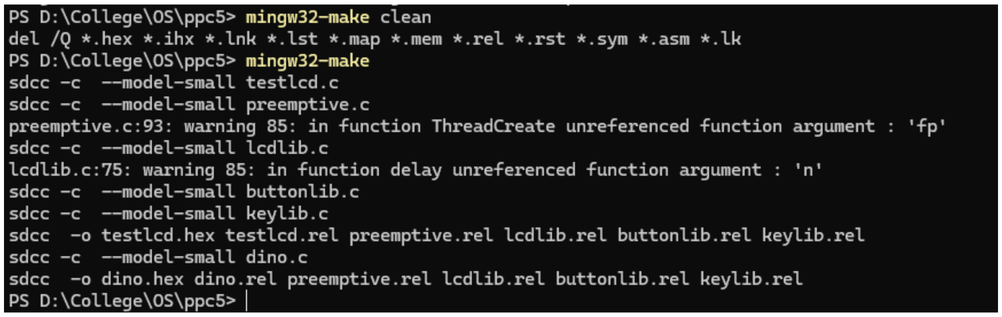
Operating System CP5

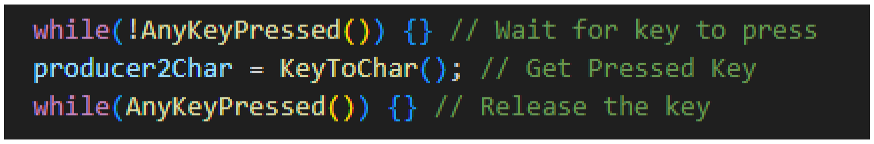
許木羽 / 111000177

1. Peripheral devices



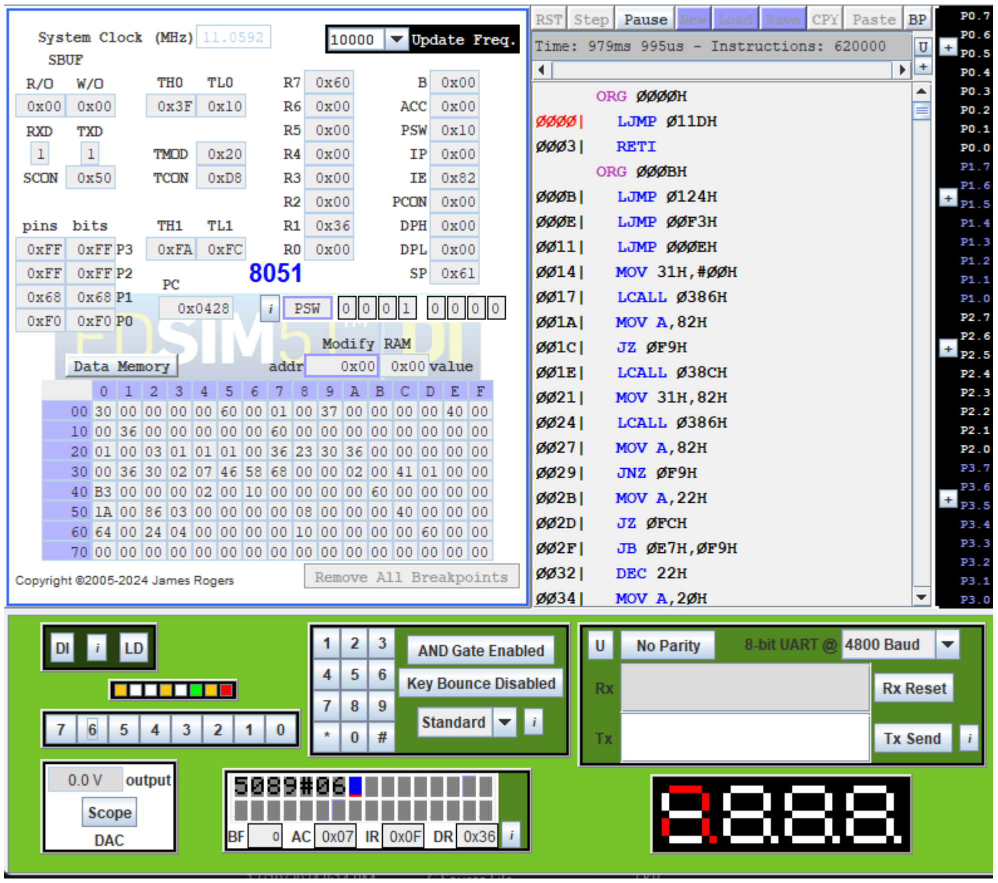
```
PS D:\College\OS\ppc5> mingw32-make clean
del /Q *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
PS D:\College\OS\ppc5> mingw32-make
sdcc -c  --model-small testlcd.c
sdcc -c  --model-small preemptive.c
preemptive.c:93: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc -c  --model-small lcdlib.c
lcdlib.c:75: warning 85: in function delay unreferenced function argument : 'n'
sdcc -c  --model-small buttonlib.c
sdcc -c  --model-small keylib.c
sdcc  -o testlcd.hex testlcd.rel preemptive.rel lcdlib.rel buttonlib.rel keylib.rel
sdcc -c  --model-small dino.c
sdcc  -o dino.hex dino.rel preemptive.rel lcdlib.rel buttonlib.rel keylib.rel
PS D:\College\OS\ppc5> |
```

Simply change the producer value instead of generating ABCD to KeyToChar(), and it needs to wait for AnyKeyPressed();, and for better responsiveness, after getting the key wait until key is unpressed.



```
while(!AnyKeyPressed()) {} // Wait for key to press
producer2Char = KeyToChar(); // Get Pressed Key
while(AnyKeyPressed()) {} // Release the key
```

Same thing goes to ButtonPressed();



```
System Clock (MHz) 11.0592        10000 Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x60     B    0x00
0x00  0x00     0x3F  0x10    R6 0x00     ACC  0x00
RXD   TXD                    R5 0x00     PSW  0x10
1     1        TMOD  0x20    R4 0x00     IP   0x00
SCON  0x50     TCON  0xD8    R3 0x00     IE   0x82
                             R2 0x00     PCON 0x00
pins  bits     TH1   TL1     R1 0x36     DPH  0x00
0xFF  0xFF P3  0xFA  0xFC    R0 0x00     DPL  0x00
0xFF  0xFF P2                8051        SP   0x61
0x68  0x68 P1  PC
0xF0  0xF0 P0  0x0428     PSW 0 0 0 1 0 0 0 0
                          Modify RAM
Data Memory               addr 0x00  0x00 value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	00	00	00	00	60	00	01	00	37	00	00	00	00	40	00
10	00	36	00	00	00	00	00	60	00	00	00	00	00	00	00	00
20	01	00	03	01	01	01	00	36	23	30	36	00	00	00	00	00
30	00	36	30	02	07	46	58	68	00	00	02	00	41	01	00	00
40	B3	00	00	00	02	00	10	00	00	00	00	60	00	00	00	00
50	1A	00	86	03	00	00	00	00	08	00	00	00	40	00	00	00
60	64	00	24	04	00	00	00	00	10	00	00	00	00	60	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2024 James Rogers

```
Time: 979ms 995us - Instructions: 620000

        ORG ØØØØH
ØØØØ|     LJMP ØllDH
ØØØ3|     RETI
        ORG ØØØBH
ØØØB|     LJMP Ø124H
ØØØE|     LJMP ØØF3H
ØØ11|     LJMP ØØØEH
ØØ14|     MOV 31H,#ØØH
ØØ17|     LCALL Ø386H
ØØ1A|     MOV A,82H
ØØ1C|     JZ ØF9H
ØØ1E|     LCALL Ø38CH
ØØ21|     MOV 31H,82H
ØØ24|     LCALL Ø386H
ØØ27|     MOV A,82H
ØØ29|     JNZ ØF9H
ØØ2B|     MOV A,22H
ØØ2D|     JZ ØFCH
ØØ2F|     JB ØE7H,ØF9H
ØØ32|     DEC 22H
ØØ34|     MOV A,2ØH
```

AND Gate Enabled
Key Bounce Disabled
Standard

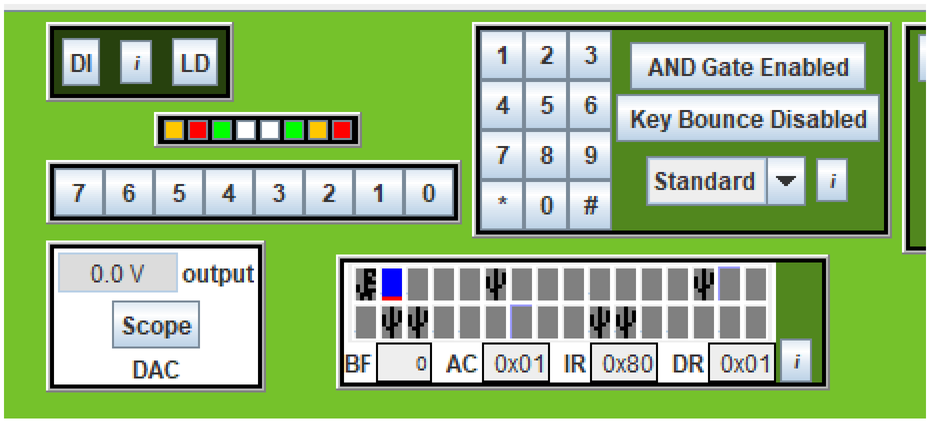
No Parity 8-bit UART @ 4800 Baud

```
5089#06
BF 0  AC 0x07  IR 0x0F  DR 0x36
```

0.0 V output
DAC

RST
Step
Pause
New
Load
Save
CPY
Paste
BP
Remove All Breakpoints
P0.7 P0.6 P0.5 P0.4 P0.3 P0.2 P0.1 P0.0
P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0
P2.7 P2.6 P2.5 P2.4 P2.3 P2.2 P2.1 P2.0
P3.7 P3.6 P3.5 P3.4 P3.3 P3.2 P3.1 P3.0
DI
LD
7 6 5 4 3 2 1 0
1 2 3 4 5 6 7 8 9 * 0 #
Rx
Tx
Rx Reset
Tx Send
Scope

2. Dino Game



Map representation using 4 char variables, as the grid is 16x2. The first row and first 8 columns data inside variable map00, first row and last 7 columns inside variable map01, second row and first 8 columns data inside variable map10, and the second row last 7 columns data inside map 11. However, the render only needs to reed 15x2. Data inside is 0 as air, 1 as cactus.

Here we have 4 thread: Key Control (Producer), Key Response (Consumer), Draw Map, and Update Map.

Updating the map is just using shifting. The map00 shifts to the left and the carry goes to map01, and the carry of map01 after shift goes to map00 end of bit. Same things go to map10 and map 11. Also depending on the state variables, if it's main menu we will ignore this function (selecting difficulty), also if the game start it check if it has passed the cactus to update score

Same idea with the update, draw function just spam API call of LCD write char based on the map and using API LCD cursor go to, to switch the row. After drawing map, we draw the dino by replacing the block, as the column is fix 0, but the column based on variable that will be updated based on the consumer.

Producer similar idea with the test LCD file

Consumer will take the value and check it current state
If it's in the state 1, it will select the difficulty, and check if it's # then we go to the game
If it's in the state 2, it will detect if the dino should go up or down, and also check if it's collide with the cactus, if it so then switch the 4th state

In the state 4, we just ignore the input as the game over

Race condition is between the mutex and mutex1 inside, as the update between draw and update can’t be together, also for fairness simply Thread Yield. There is also race condition inside producer and consumer (get key and response key) solved with similar idea.