

Programming Project Checkpoint 3

IPNS 106010018 Yen-Fong Li

	Value	Global	Global Defined In Module
C:	00000014	_Producer	testpreempt
C:	00000052	_Consumer	testpreempt
C:	0000008E	_main	testpreempt
C:	000000A6	__sdcc_gsinit_startup	testpreempt
C:	000000AA	__mcs51_genRAMCLEAR	testpreempt
C:	000000AB	__mcs51_genXINIT	testpreempt
C:	000000AC	__mcs51_genXRAMCLEAR	testpreempt
C:	000000AD	_timer0_ISR	testpreempt
C:	000000B1	_Bootstrap	preemptive
C:	000000D7	_ThreadCreate	preemptive
C:	00000156	_ThreadYield	preemptive
C:	000001EC	_myTimer0Handler	preemptive
C:	0000027C	_ThreadExit	preemptive

From testpreempt.map, we can find out that the address of ThreadCreate() function is at 000000D7, so set checkpoint at 00D7 and then run Edsim51.

System Clock (MHz) 11.0592 10000 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x03	0x01	0x00	0x00
RKD	TxD	TMOD	0x00	R6	ACC
1	1	TCON	0x10	0x00	0x00
SCON	0x00			R5	PSW
				0x00	0x00
pins	bits	TH1	TL1	R4	IP
0xFF	0xFF	0x00	0x00	0x00	0x82
0xFF	0xFF	PC		R3	IE
0xFF	0xFF			0x00	0x00
0xFF	0xFF			R2	PCON
0xFF	0xFF			0x30	0x00
				R1	DPH
				0x30	0x14
				R0	DPL
					0x41
					SP

8051

PSW 0 0 0 0 0 0 0 0

Data Memory

addr	0x00	0x00	value
0	00	30	00
1	00	00	00
2	00	00	00
3	00	00	00
4	00	00	00
5	00	00	00
6	00	00	00
7	00	00	00
8	00	00	00
9	00	00	00
A	00	00	00
B	00	00	00
C	00	00	00
D	00	00	00
E	00	00	00
F	00	00	00

Remove All Breakpoints

Semaphore:

0x21 for full is 0

0x23 for empty is 1

0x24 for mutex is 1

They are correctly initialized.

Producer

System Clock (MHz) 11.0592 10000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x00 0x04 0x10 R6 0x42 ACC 0x00
RxD TxD R5 0x42 PSW 0x08
1 1 TMOD 0x20 R4 0x00 IP 0x00
SCON 0x50 TCON 0xD0 R3 0x00 IE 0x82
pins bits TH1 TL1 R2 0x80 PCON 0x00
0xFF 0xFF P3 0xFA 0xFF R1 0x00 DPH 0x00
0xFF 0xFF P2 PC 8051 DPL 0x00
0xFF 0xFF P1 0x0014 PSW 0 0 0 0 1 0 0 0
0xFF 0xFF P0

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	31	00	00	00	00	20	00	31	00	80	00	00	42	42	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	41	42	01	00	01	00	00	00	00	00	00	00	00	00	00	00
30	46	56	00	00	01	03	01	42	00	00	00	00	00	00	00	00
40	63	00	00	00	01	00	80	30	30	00	00	00	00	20	00	00
50	14	00	00	00	00	00	08	31	30	80	00	00	4D	4D	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2016 James Rogers Remove All Breakpoints

After Calling Producer()

Since empty and mutex were initialized as 1

It can run through and signal full

So 0x22 become 01

Consumer

System Clock (MHz) 11.0592 10000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x01 B 0x00
0x00 0x52 0x75 0x09 R6 0x20 ACC 0x00
RxD TxD R5 0x00 PSW 0x80
1 1 TMOD 0x20 R4 0x00 IP 0x00
SCON 0x50 TCON 0xD0 R3 0x00 IE 0x82
pins bits TH1 TL1 R2 0x00 PCON 0x00
0xFF 0xFF P3 0xFA 0xFA R1 0x31 DPH 0x00
0xFF 0xFF P2 PC 8051 DPL 0x01
0xFF 0xFF P1 0x0063 PSW 1 0 0 0 0 0 0 0
0xFF 0xFF P0

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	31	00	00	00	00	20	01	31	30	80	00	42	53	53	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	52	53	00	01	01	00	00	00	00	00	00	00	00	00	00	00
30	46	56	00	00	00	03	01	42	00	00	00	00	00	00	00	00
40	63	00	00	00	01	00	80	30	31	00	00	00	00	20	01	00
50	17	00	00	00	00	00	08	31	30	80	00	42	53	53	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2016 James Rogers Remove All Breakpoints

After Calling Consumer ()

Since full became 01

It can run through and signal empty

So 0x23 become 01

Explanations

```
#define SemaphoreWait(s) \
    SemaphoreWaitBody(s, L(__COUNTER__)) \

#define SemaphoreWaitBody(s, label) \
{ __asm \
    label: \
    MOV ACC, CNAME(s) \
    JZ label \
    JB ACC.7, label \
    dec CNAME(s) \
    __endasm; }
```

WAIT()

No need to separate to two functions. Just Call SemaphoreWaitBody and give the right parameters can also do the work.

In SemaphoreWaitBody, We first move s to accumulator Then we decide whether we should jump back to label by checking whether it is zero.

```
#define SemaphoreSignal(s) \
    __asm \
    INC CNAME(s) \
    __endasm;
```

SIGNAL()

In this part, the only thing we need to do is increment the semaphore.