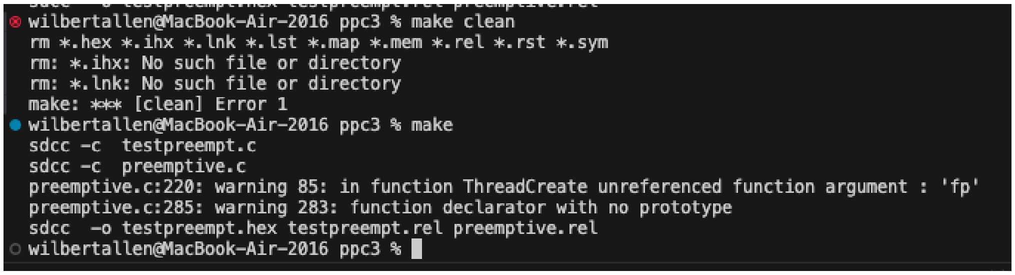
鍾維聖 110062261

Programming Project Checkpoint 3 Report

Typescript for Compiling

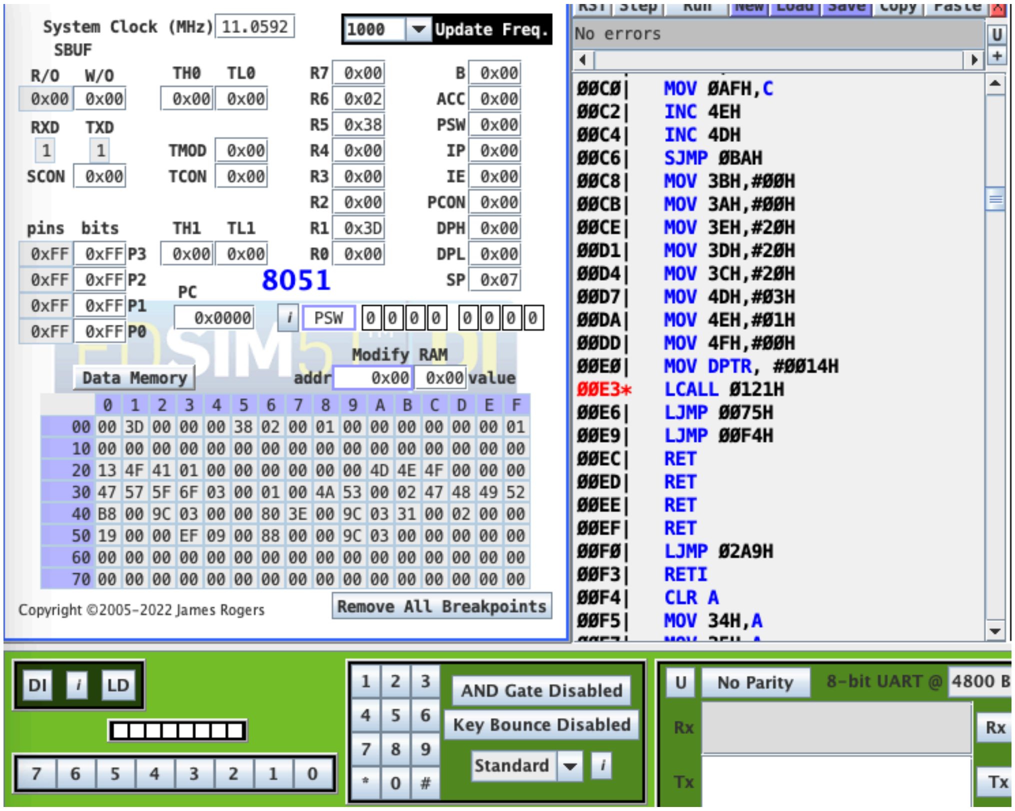


```
wilbertallen@MacBook-Air-2016 ppc3 % make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: *.ihx: No such file or directory
rm: *.lnk: No such file or directory
make: *** [clean] Error 1
wilbertallen@MacBook-Air-2016 ppc3 % make
sdcc -c  testpreempt.c
sdcc -c  preemptive.c
preemptive.c:220: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
preemptive.c:285: warning 283: function declarator with no prototype
sdcc  -o testpreempt.hex testpreempt.rel preemptive.rel
wilbertallen@MacBook-Air-2016 ppc3 %
```

Fig.1 Typescript for compiling using the given makefile

Producer is Running and Changing Semaphore

At a glance, we can see that the Producer's value is always changing, incrementing the value by 3. We can see that by the value from *shared_buff[3]* and *buffer*



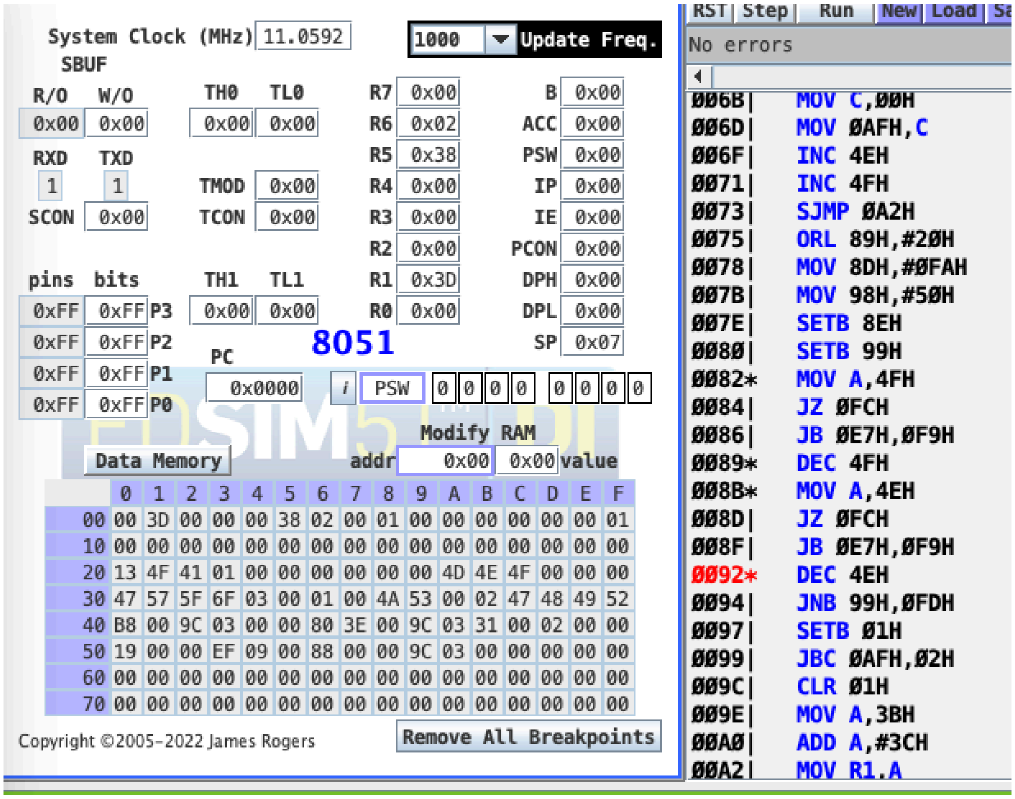
```
System Clock (MHz) 11.0592        1000  Update Freq.
SBUF
R/O   W/O     TH0   TL0     R7 0x00      B 0x00
0x00  0x00    0x00  0x00    R6 0x02    ACC 0x00
RXD   TXD                   R5 0x38    PSW 0x00
1     1       TMOD  0x00    R4 0x00     IP 0x00
SCON  0x00    TCON  0x00    R3 0x00     IE 0x00
                            R2 0x00   PCON 0x00
pins  bits    TH1   TL1     R1 0x3D    DPH 0x00
0xFF  0xFF P3 0x00  0x00    R0 0x00    DPL 0x00
0xFF  0xFF P2         8051              SP 0x07
0xFF  0xFF P1   PC
0xFF  0xFF P0   0x0000     PSW 0 0 0 0  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 00 3D 00 00 00 38 02 00 01 00 00 00 00 00 00 01
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 13 4F 41 01 00 00 00 00 00 00 4D 4E 4F 00 00 00
30 47 57 5F 6F 03 00 01 00 4A 53 00 02 47 48 49 52
40 B8 00 9C 03 00 00 80 3E 00 9C 03 31 00 02 00 00
50 19 00 00 EF 09 00 88 00 00 9C 03 00 00 00 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 ©2005-2022 James Rogers
```

```
No errors
00C0|   MOV 0AFH,C
00C2|   INC 4EH
00C4|   INC 4DH
00C6|   SJMP 0BAH
00C8|   MOV 3BH,#00H
00CB|   MOV 3AH,#00H
00CE|   MOV 3EH,#20H
00D1|   MOV 3DH,#20H
00D4|   MOV 3CH,#20H
00D7|   MOV 4DH,#03H
00DA|   MOV 4EH,#01H
00DD|   MOV 4FH,#00H
00E0|   MOV DPTR, #0014H
00E3*   LCALL 0121H
00E6|   LJMP 0075H
00E9|   LJMP 00F4H
00EC|   RET
00ED|   RET
00EE|   RET
00EF|   RET
00F0|   LJMP 02A9H
00F3|   RETI
00F4|   CLR A
00F5|   MOV 34H,A
```

```
U  No Parity  8-bit UART @ 4800 B
AND Gate Disabled
Key Bounce Disabled
Standard
```

Fig2.1 Calling ThreadCreate(Producer)

After the producer is called, we can see that the assembly jumps to 0x75H (Consumer). There we are greeted by some variable initialization.



```
System Clock (MHz) 11.0592          1000
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B 0x00
0x00  0x00     0x00  0x00    R6 0x02    ACC 0x00
RXD   TXD                    R5 0x38    PSW 0x00
1     1        TMOD  0x00    R4 0x00     IP 0x00
SCON  0x00     TCON  0x00    R3 0x00     IE 0x00
                             R2 0x00   PCON 0x00
pins  bits     TH1   TL1     R1 0x3D    DPH 0x00
0xFF  0xFF P3  0x00  0x00    R0 0x00    DPL 0x00
0xFF  0xFF P2                            SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0000   PSW 0 0 0 0  0 0 0 0
addr 0x00  0x00 value
```

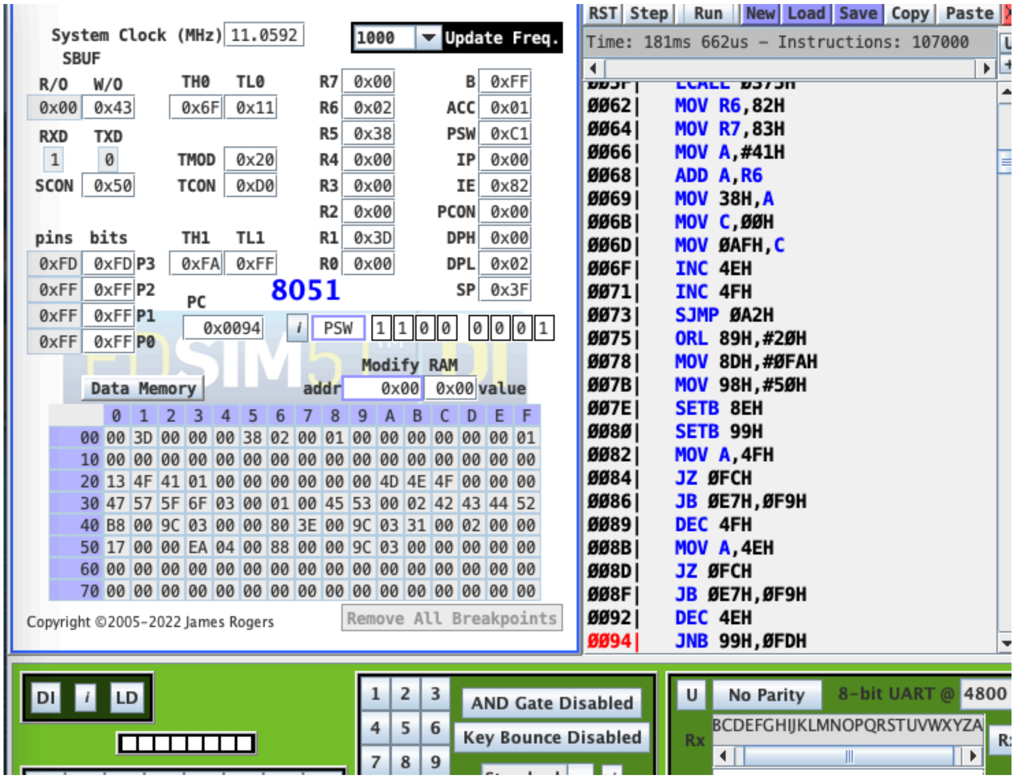
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	3D	00	00	00	38	02	00	01	00	00	00	00	00	00	01
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	13	4F	41	01	00	00	00	00	00	00	4D	4E	4F	00	00	00
30	47	57	5F	6F	03	00	01	00	4A	53	00	02	47	48	49	52
40	B8	00	9C	03	00	00	80	3E	00	9C	03	31	00	02	00	00
50	19	00	00	EF	09	00	88	00	00	9C	03	00	00	00	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 ©2005-2022 James Rogers

```
No errors
ØØ6B|   MOV C,ØØH
ØØ6D|   MOV ØAFH,C
ØØ6F|   INC 4EH
ØØ71|   INC 4FH
ØØ73|   SJMP ØA2H
ØØ75|   ORL 89H,#2ØH
ØØ78|   MOV 8DH,#ØFAH
ØØ7B|   MOV 98H,#5ØH
ØØ7E|   SETB 8EH
ØØ8Ø|   SETB 99H
ØØ82*   MOV A,4FH
ØØ84|   JZ ØFCH
ØØ86|   JB ØE7H,ØF9H
ØØ89*   DEC 4FH
ØØ8B*   MOV A,4EH
ØØ8D|   JZ ØFCH
ØØ8F|   JB ØE7H,ØF9H
ØØ92*   DEC 4EH
ØØ94|   JNB 99H,ØFDH
ØØ97|   SETB Ø1H
ØØ99|   JBC ØAFH,Ø2H
ØØ9C|   CLR Ø1H
ØØ9E|   MOV A,3BH
ØØAØ|   ADD A,#3CH
ØØA2|   MOV R1,A
```

Fig.2.2 Calling SemaphoreWait

We can see the semaphore changes by observing the steps below that. We can see on 0x82H and 0x8BH, that it is calling SemaphoreWait(full) and SemaphoreWait(mutex) based on the assembly we write on preemptive.h. Here we can see that Producer and Consumer are communicating via the semaphores ensuring mutual exclusion, and preventing simultaneous access to critical sections.



```
System Clock (MHz) 11.0592          1000
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B 0xFF
0x00  0x43     0x6F  0x11    R6 0x02    ACC 0x01
RXD   TXD                    R5 0x38    PSW 0xC1
1     0        TMOD  0x20    R4 0x00     IP 0x00
SCON  0x50     TCON  0xD0    R3 0x00     IE 0x82
                             R2 0x00   PCON 0x00
pins  bits     TH1   TL1     R1 0x3D    DPH 0x00
0xFD  0xFD P3  0xFA  0xFF    R0 0x00    DPL 0x02
0xFF  0xFF P2                            SP 0x3F
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0094   PSW 1 1 0 0  0 0 0 1
addr 0x00  0x00 value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	3D	00	00	00	38	02	00	01	00	00	00	00	00	00	01
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	13	4F	41	01	00	00	00	00	00	00	4D	4E	4F	00	00	00
30	47	57	5F	6F	03	00	01	00	45	53	00	02	42	43	44	52
40	B8	00	9C	03	00	00	80	3E	00	9C	03	31	00	02	00	00
50	17	00	00	EA	04	00	88	00	00	9C	03	00	00	00	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

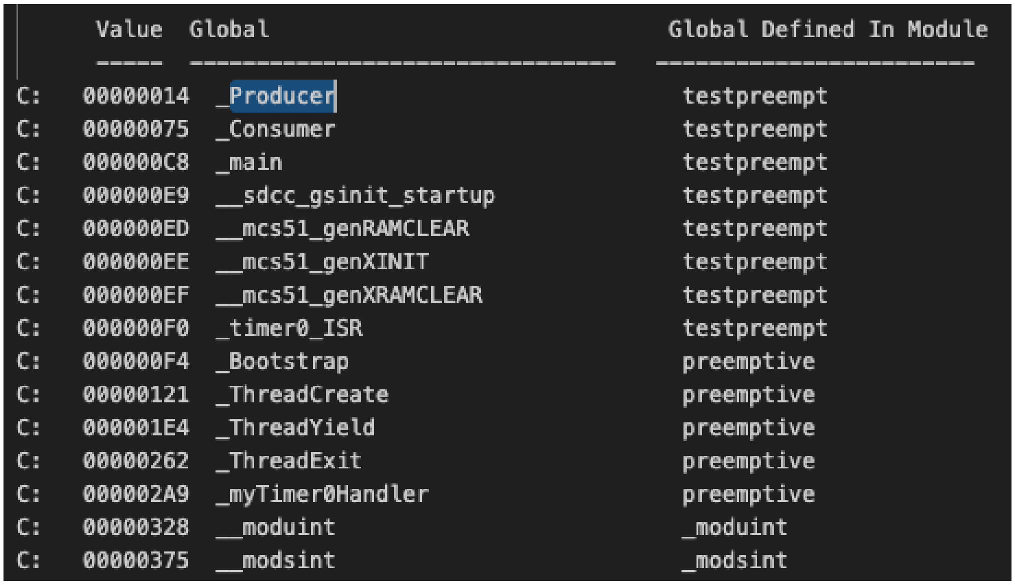
Copyright ©2005-2022 James Rogers

```
Time: 181ms 662us - Instructions: 107000
ØØ62|   MOV R6,82H
ØØ64|   MOV R7,83H
ØØ66|   MOV A,#41H
ØØ68|   ADD A,R6
ØØ69|   MOV 38H,A
ØØ6B|   MOV C,ØØH
ØØ6D|   MOV ØAFH,C
ØØ6F|   INC 4EH
ØØ71|   INC 4FH
ØØ73|   SJMP ØA2H
ØØ75|   ORL 89H,#2ØH
ØØ78|   MOV 8DH,#ØFAH
ØØ7B|   MOV 98H,#5ØH
ØØ7E|   SETB 8EH
ØØ8Ø|   SETB 99H
ØØ82|   MOV A,4FH
ØØ84|   JZ ØFCH
ØØ86|   JB ØE7H,ØF9H
ØØ89|   DEC 4FH
ØØ8B|   MOV A,4EH
ØØ8D|   JZ ØFCH
ØØ8F|   JB ØE7H,ØF9H
ØØ92|   DEC 4EH
ØØ94|   JNB 99H,ØFDH
```

```
No Parity   8-bit UART @ 4800
Rx BCDEFGHIJKLMNOPQRSTUVWXYZA
```

Fig.2.3 Running Producer

We can also see that the Producer is running by observing the value of shared_buff and buffer changing, in time with the semaphore changes. Screen recording can be seen here

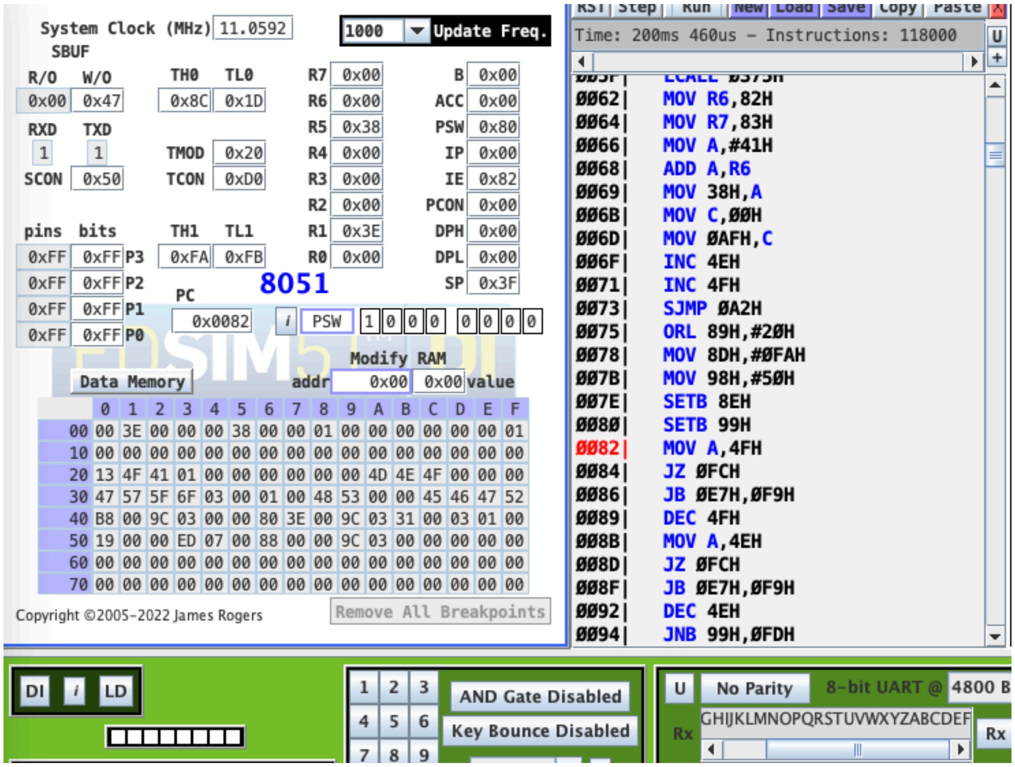


```
     Value  Global                              Global Defined In Module
     -----  --------------------------------   -------------------------
C:   00000014  _Producer                          testpreempt
C:   00000075  _Consumer                          testpreempt
C:   000000C8  _main                              testpreempt
C:   000000E9  __sdcc_gsinit_startup              testpreempt
C:   000000ED  __mcs51_genRAMCLEAR                testpreempt
C:   000000EE  __mcs51_genXINIT                   testpreempt
C:   000000EF  __mcs51_genXRAMCLEAR               testpreempt
C:   000000F0  _timer0_ISR                        testpreempt
C:   000000F4  _Bootstrap                         preemptive
C:   00000121  _ThreadCreate                      preemptive
C:   000001E4  _ThreadYield                       preemptive
C:   00000262  _ThreadExit                        preemptive
C:   000002A9  _myTimer0Handler                   preemptive
C:   00000328  __moduint                          _moduint
C:   00000375  __modsint                          _modsint
```

Fig.2.4 Function addresses value

Running Consumer and Changing Semaphores

By the same logic, we already prove that Producer and Consumer are communicating through semaphore on the point above.We can then observe the running consumer by the value being submitted to SBUF that changes from A to Z and writing it to the received data. On the case below, SBUF is writing 0x47H which is ‘G’ to the received data.



```
System Clock (MHz) 11.0592        1000  Update Freq.
 SBUF
R/O   W/O      TH0   TL0     R7 0x00      B 0x00
0x00  0x47     0x8C  0x1D    R6 0x00    ACC 0x00
RXD   TXD                    R5 0x38    PSW 0x80
 1     1       TMOD 0x20     R4 0x00     IP 0x00
SCON  0x50     TCON 0xD0     R3 0x00     IE 0x82
                             R2 0x00   PCON 0x00
pins  bits     TH1   TL1     R1 0x3E    DPH 0x00
0xFF  0xFF P3  0xFA  0xFB    R0 0x00    DPL 0x00
0xFF  0xFF P2                            SP 0x3F
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x0082     PSW 1 0 0 0  0 0 0 0
                            Modify RAM
                     addr 0x00  0x00 value
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 00 3E 00 00 00 38 00 00 01 00 00 00 00 00 00 01
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 13 4F 41 01 00 00 00 00 00 00 4D 4E 4F 00 00 00
30 47 57 5F 6F 03 00 01 00 48 53 00 00 45 46 47 52
40 B8 00 9C 03 00 00 80 3E 00 9C 03 31 00 03 01 00
50 19 00 00 ED 07 00 88 00 00 9C 03 00 00 00 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 ©2005-2022 James Rogers

```
Time: 200ms 460us - Instructions: 118000
005F|    LCALL 0375H
0062|    MOV R6,82H
0064|    MOV R7,83H
0066|    MOV A,#41H
0068|    ADD A,R6
0069|    MOV 38H,A
006B|    MOV C,00H
006D|    MOV 0AFH,C
006F|    INC 4EH
0071|    INC 4FH
0073|    SJMP 0A2H
0075|    ORL 89H,#20H
0078|    MOV 8DH,#0FAH
007B|    MOV 98H,#50H
007E|    SETB 8EH
0080|    SETB 99H
0082|    MOV A,4FH
0084|    JZ 0FCH
0086|    JB 0E7H,0F9H
0089|    DEC 4FH
008B|    MOV A,4EH
008D|    JZ 0FCH
008F|    JB 0E7H,0F9H
0092|    DEC 4EH
0094|    JNB 99H,0FDH
```

8-bit UART @ 4800 B

GHIJKLMNOPQRSTUVWXYZABCDEF