Important changes to *Embedded Microcomputer Systems*, 3rd printing by Jonathan W. Valvano

Page 8 change
(assuming Port B is located at $0001)
to
(assuming Port B is located at $0001 on the 6812)

Page 39, figure 1.42 change
4K bytes EEPROM
to
768 bytes EEPROM

Page 40, Table 1.21 change
<table>
<thead>
<tr>
<th>Size</th>
<th>512</th>
<th>1024</th>
<th>4096</th>
</tr>
</thead>
</table>
to
| Size  | 512 | 1024 | 4096 |

Page 49 last line change
active
to
activate

Page 53 (4th line from the bottom) change
Power supply current
to
Power supply voltage

Page 58 line 5 change
From Table 1.28, we see that $I_{HH}$ is 40 µA, so $I_{out}$ should be larger than 5•40 µA or 0.2mA.
to
From Table 1.26, we see that $I_{HH}$ is 40 µA, so $I_{out}$ should be larger than 5•40 µA or 0.2mA.

Page 66 change
In C (Program 1.19), an I/O port name (defined in a manner similar to Program 1.12) on the left-hand side of an assignment operator is implemented with an input port read access.
to
In C (Program 1.19), an I/O port name (defined in a manner similar to Program 1.12) on the right-hand side of an assignment operator is implemented with an input port read access.

Page 69 Program 1.27 6812 version change
```
main lds #$0C00 ;SP=$00FF
```
to
```
main lds #$0C00 ;SP=$0C00
```

Page 73 RAM definition change
memory where is the information
to
memory where the information
Page 101 Program 2.7 change (see third and fourth to the last lines)

```c
unsigned int Median(unsigned int u1, unsigned int u2, unsigned int u3){
    unsigned int result;
    printf("The inputs are %d, %d, %d.\n",u1,u2,u3);
    if(u1>u2)  
        if(u2>u3)   result=u2;  // u1>u2,u2>u3       u1>u2>u3
        else       result=u1;   // u1>u2,u3>u2,u1>u3 u3>u1>u2
    else
        if(u3>u2)   result=u2;  // u2>u1,u3>u2       u3>u2>u1
        else       result=u3;   // u2>u1,u2>u3,u1>u3 u2>u3>u1
    printf("The median is %d.\n",result);
    return(result);
}
```

to

```c
unsigned int Median(unsigned int u1, unsigned int u2, unsigned int u3){
    unsigned int result;
    printf("The inputs are %d, %d, %d.\n",u1,u2,u3);
    if(u1>u2)  
        if(u2>u3)   result=u2;  // u1>u2,u2>u3       u1>u2>u3
        else       result=u1;   // u1>u2,u3>u2,u1>u3 u3>u1>u2
    else
        if(u3>u2)   result=u2;  // u2>u1,u3>u2       u3>u2>u1
        else       result=u3;   // u2>u1,u2>u3,u1>u3 u2>u3>u1
    printf("The median is %d.\n",result);
    return(result);
}
```

Page 118, last line change

```
port<unsigned char> InPort(0x0003,0x0007);  // bidirectional port
```

to

```
port<unsigned char> InPort(0x1003,0x1007);  // bidirectional port
```

Page 140 change

At the next periodic interrupt, the software will read the data and save in global structure.

to

At the next periodic interrupt, the software will read the data and save them in a global structure.

Page 164, Program 3.19 change

```
; MC68HC812A4/MC68HC912B32
; PS7=RST PS6=CLK PS5=DQ
Init ldaa #$E0  ;PD5-3 output
    staa DDRS
    ldaa #$60  ;RST=0,CLK=1
    staaa PORTS ;DQ=1
    rts
```

to

```
; MC68HC812A4/MC68HC912B32
; PS7=RST PS6=CLK PS5=DQ
Init ldaa #$E0  ;PS7-5 output
    staa DDRS
    ldaa #$60  ;RST=0,CLK=1
    staaa PORTS ;DQ=1
    rts
```
Page 170, Program 3.25 (6812 version) change

```
bra  next
```
to

```
bra  onext
```

Page 175, Program 3.30 (6812 version) change

```
bne  ilop
```
to

```
bne  jlop
```

Page 195, Figure 4.7 should look like

```
a) main program waits because FIFO is empty
b) ISR reads data and puts into FIFO
c) main program get data from FIFO
d) main program processes data
```

Page 199, last line change

```
(Resulttx)>>1
```
to

```
(Result+x)>>1
```

Page 208, program 4.16 change

```
 tpa  restore CCR to previous value
tba
```
to

```
 tap  restore CCR to previous value
tba
```

Page 209, program 4.17 change

```
 cpx  PutPt  Empty if initially the same
tba
```
to

```
 cpy  PutPt  Empty if initially the same
tba
```

Page 209, program 4.17 change

```
 tpa  restore CCR to previous value
tba
```
to

```
 tap  restore CCR to previous value
tba
```

Page 211, program 4.20 change

```
 tpa  restore CCR to previous value
tba
```
to

```
 tap  restore CCR to previous value
tba
```
Page 212, program 4.21 change

<table>
<thead>
<tr>
<th>tpa</th>
<th>restore CCR to previous value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tba</td>
<td></td>
</tr>
</tbody>
</table>

Page 214, program 4.24 change

<table>
<thead>
<tr>
<th>cmpb #FifoSize Full if Size==FifoSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq PutNotFull</td>
</tr>
</tbody>
</table>

Page 214, program 4.24 change

<table>
<thead>
<tr>
<th>tpa</th>
<th>restore CCR to previous value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tba</td>
<td></td>
</tr>
</tbody>
</table>

Page 215, program 4.25 change

<table>
<thead>
<tr>
<th>tpa</th>
<th>restore CCR to previous value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tba</td>
<td></td>
</tr>
</tbody>
</table>

Page 215, program 4.25 change

| stab 0,x      Return by reference |
|---------------|-----------------------------------|
| ldab GetI     |                                  |

Page 279, change the sequence of Program 5.13 so it reads as follows

* To block a thread on semaphore S, execute SWI

SWIHan

<table>
<thead>
<tr>
<th>ldx RunPt running process “to be blocked”</th>
</tr>
</thead>
<tbody>
<tr>
<td>sts SP,x save Stack Pointer in its TCB</td>
</tr>
</tbody>
</table>

* Unlink “to be blocked” thread from RunPt list

<table>
<thead>
<tr>
<th>ldy Next,x find previous thread (Figure 5.10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sty RunPt next one to run</td>
</tr>
</tbody>
</table>

look

<table>
<thead>
<tr>
<th>cpx Next,y search to find previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq found</td>
</tr>
<tr>
<td>ldy Next,y</td>
</tr>
<tr>
<td>bra look</td>
</tr>
</tbody>
</table>

found

<table>
<thead>
<tr>
<th>ldd RunPt one after blocked (Figure 5.11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>std Next,y link previous to next to run</td>
</tr>
</tbody>
</table>

* Put “to be blocked” thread on block list

<table>
<thead>
<tr>
<th>ldy BlockPt (Figure 5.12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sty Next,x link “to be blocked”</td>
</tr>
<tr>
<td>stx BlockPt</td>
</tr>
</tbody>
</table>

* Launch next thread

<table>
<thead>
<tr>
<th>ldx RunPt</th>
</tr>
</thead>
<tbody>
<tr>
<td>lds SP,x set SP for this new thread</td>
</tr>
<tr>
<td>ldd TCNT Next thread gets a full 10ms time slice</td>
</tr>
<tr>
<td>addd #20000 interrupt after 10 ms</td>
</tr>
<tr>
<td>std TOC5</td>
</tr>
<tr>
<td>ldaa #$08 ($20 on the 6812)</td>
</tr>
<tr>
<td>staa TFLG1 clear OC5F</td>
</tr>
<tr>
<td>rti</td>
</tr>
</tbody>
</table>
Page 286 change
1) the current TCNT value to be copied into the input capture register
to
1) the current TCNT value is copied into the input capture register

Page 299, Table 6.8 change
50=5µsec
to
40=4µsec

Page 301, Program 6.5 change
; MC68HC812A4
IC1Han movb #$08,TFLG1 ;clear C3F [4]
to
; MC68HC812A4
IC1Han movb #$02,TFLG1 ;clear C1F [4]

Page 307, Program 6.10, 6811 C code, change "}" to "}"
First = TIC1; Count=0; Mode=1;
if(((TIC1&0x8000)==0) 
& &(TFLG2&0x80)) Count--;

to
First = TIC1; Count=0; Mode=1;
if(((TIC1&0x8000)==0) 
& &(TFLG2&0x80)) Count--;

Page 307, program 6.10, change
TCTL4 = (TCTL4&0xFC)|0x04; // rising

to
TCTL4 = (TCTL4&0xF3)|0x04; // rising

Page 309, Program 6.11 change
; MC68HC812A4
; B=PB7, Q=PT2/IC2
Init  bclr TIOS,$04 ;PT2=input capt.
movb #$90,TSCR ;enable, fast clr
movb #$32,TMSK2 ;500 ns clk
clr TMSK1 ;gadfly, C2I=0
bclr DDRB,$80
rts

to
; MC68HC812A4
; B=PB7, Q=PT2/IC2
Init  bclr TIOS,$04 ;PT2=input capt.
movb #$90,TSCR ;enable, fast clr
movb #$32,TMSK2 ;500 ns clk
clr TMSK1 ;gadfly, C2I=0
bset DDRB,$80
rts
Page 310, Program 6.12 change

; MC68HC812A4
; return Reg D as R in Kohm
Meas movb #$10, TCTL4 ; Rising edge
   movb #$04, TFLG1 ; C2F=0
   bclr PORTB,$80 ; PB7=0
   bset PORTB,$80 ; PB7=1
First brclr TFLG1,$04,First
; Wait for first rising edge
   ldy TC2 ; TCNT at rising, C2F=0
   movb #$20, TCTL4 ; Falling edge
   pshy ; Save on stack
Second brclr TFLG1,$04,Second
; Wait for next falling edge
   ldd TC2 ; TCNT at falling
   subd 2, SP+
; RegD=pulse width 1000 to 2000 cyc
   subd #1000 ; 0=R<=1000Kohm
   rts

to
; MC68HC812A4
; return Reg D as R in Kohm
Meas movb #$10, TCTL4 ; Rising edge
   movb #$04, TFLG1 ; C2F=0
   bclr PORTB,$80 ; PB7=0
   bset PORTB,$80 ; PB7=1
First brclr TFLG1,$04,First
; Wait for first rising edge
   ldy TC2 ; TCNT at rising, C2F=0
   movb #$20, TCTL4 ; Falling edge
   pshy ; Save on stack
Second brclr TFLG1,$04,Second
; Wait for next falling edge
   ldd TC2 ; TCNT at falling
   subd 2, SP+
; RegD=pulse width 1000 to 2000 cyc
   subd #1000 ; 0=R<=1000Kohm
   rts

Page 327, Table 6.12 change

<table>
<thead>
<tr>
<th></th>
<th>6808</th>
<th>9=1.125µs</th>
<th>29=9.625µs</th>
<th>4.75µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>6808</td>
<td>9=1.125µs</td>
<td>29=9.625µs</td>
<td>10.75µs</td>
</tr>
</tbody>
</table>

Page 382, Change SCxDR to SCxDRL (in RDRF description)

Page 418 Change PUPEJ to PULEJ three places, two places in the last paragraph, once in program 8.1
Changes to Embedded Microcomputer Systems, 3rd printing by Jonathan W. Valvano

Page 432, Program 8.8 change
clr PUPEJ ; regular input
to
clr PULEJ ; regular input

Page 434, Program 8.9 change
PUPEJ = 0; // regular input
to
PULEJ = 0; // regular input

Page 435, Program 8.10 change
PUPEJ=0; // regular input
to
PULEJ=0; // regular input

Page 437, Program 8.11, MC68HC812A4 version, remove spaces after the two comma, change
Ritual: clr DDRJ ;PJ3-PJ0 inputs
movb #$0F, PUPSJ
movb #$0F, PULEJ
rts ;PJ7-PJ0 oc outputs
to
Ritual: clr DDRJ ;PJ3-PJ0 inputs
movb #$0F,PUPSJ
movb #$0F,PULEJ
rts ;PJ7-PJ0 oc outputs

Page 450, Program 8.15, MC68HC812A4 version, change
TC5=TOC5+10000; // every 5 ms
to
TC5=TC5+10000; // every 5 ms

Page 468, line 2, change made to make

Page 506, Table 9.5, change
PF4 CSD 0xxxxxxxxxxxxxxx $0000 $7FFF 32K (CSDFH=0)
to
PF4 CSD 0xxxxxxxxxxxxxxx $0000 $7FFF 32K (CSDFH=1)

Page 525, Change in the last line from
SMODN, MODB, MODA register.
to
SMODN, MODB, MODA bits in the MODE register.

Page 569, Figure 9.82, wrong figure. Should be
Page 569, Change
"The write timing when controlled by C1 is shown on the left in Figure 9.82; the write timing when controlled by C2 is shown on the right in Figure 9.82."

to
"The write function occurs on either the fall of C1 or the rise of C2, whichever occurs first. Let the setup time be \( t_{\text{ss}} \) and assume the hold time is zero."

Page 635, 7 lines from the bottom change
last time of interface
to
last type of interface

Page 636, 13 lines from the bottom change
output capture interrupt
to
output compare interrupt

Page 809, Program 15.2 (6811 version)
change
```c
    for(i=5;i>0;i++)
```

to
```c
    for(i=5;i>0;i--)
```

Page 814, Program 15.8, change (see second and third to the last lines)
```c
unsigned char Median(unsigned char u1,unsigned char u2,unsigned char u3){
    unsigned char result;
    if(u1>u2)
        if(u2>u3) result=u2;  // u1>u2,u2>u3  u1>u2>u3
            else
                if(u1>u3) result=u3;  // u1>u2,u3>u2,u1>u3  u1>u3>u2
                else result=u1;  // u1>u2,u3>u2,u3>u1  u3>u1>u2
        else
            if(u3>u2) result=u2;  // u2>u1,u3>u2  u3>u2>u1
                else
                    if(u1>u3) result=u1;  // u2>u1,u2>u3,u1>u3  u2>u1>u3
                    else result=u3;  // u2>u1,u2>u3,u3>u1  u2>u3>u1
    return(result);
}
```

to
```c
unsigned char median(unsigned char u1,unsigned char u2,unsigned char u3){
    unsigned char result;
    if(u1>u2)
        if(u2>u3) result=u2;  // u1>u2,u2>u3  u1>u2>u3
            else
                if(u1>u3) result=u3;  // u1>u2,u3>u2,u1>u3  u1>u3>u2
                else result=u1;  // u1>u2,u3>u2,u3>u1  u3>u1>u2
        else
            if(u3>u2) result=u2;  // u2>u1,u3>u2  u3>u2>u1
                else
                    if(u1>u3) result=u1;  // u2>u1,u2>u3,u1>u3  u2>u1>u3
                    else result=u3;  // u2>u1,u2>u3,u3>u1  u2>u3>u1
    return(result);
}
```

Back cover, change 74505 to 74S05