

Corrections to Embedded Microcomputer Systems: Real Time Interfacing, Second Edition, Jonathan W. Valvano, Thomson-Engineering Publishers, ISBN 0534551629

Page 108 change

In this last example, the technique provides a mechanism for allocating large amounts of stack space.

To

In this last example, the technique provides a mechanism for deallocating large amounts of stack space.

Page 225, Program 4.20, change (remove space between & =)

```
DDRJ & = ~0x80;
```

To

```
DDRJ &= ~0x80;
```

Page 363, Program 7.5, add this line in two places.

```
while((SPISR&0x20)==0); // wait for SPTEF
```

Program should be

```
// MC9S12C32
```

```
#define SPIF 0x80
```

```
void DAC_out(unsigned short code){
```

```
unsigned char dummy;
```

```
while((SPISR&0x20)==0); // wait for SPTEF
```

```
SPIDR = (code>>8); // msbyte
```

```
while((SPISR&SPIF)==0); // gadfly wait
```

```
dummy = SPIDR; // clear SPIF
```

```
while((SPISR&0x20)==0); // wait for SPTEF
```

```
SPIDR = code; // lsbyte
```

```
while((SPISR&SPIF)==0); // gadfly wait
```

```
dummy = SPIDR; // clear SPIF
```

```
PTM &= ~0x08; // PM3=LD=0
```

```
PTM |= 0x08; } // PM3=LD=1
```

Page 365, Program 7.7, add this line in three places.

```
while((SPISR&0x20)==0); // wait for SPTEF
```

Program should be

```
// MC9S12C32
```

```
unsigned short ADC_in(unsigned char code){
```

```
unsigned short data;
```

```
PTM &= ~0x08; // PM3=CS=0
```

```
while((SPISR&0x20)==0); // wait for SPTEF
```

```
SPIDR = code; // set channel,mode
```

```
while((SPISR&0x80)==0); // gadfly wait
```

```
data = SPIDR; // clear SPIF
```

```
while((SPISR&0x20)==0); // wait for SPTEF
```

```
SPIDR = 0; // start SPI
```

```
while((SPISR&0x80)==0); // gadfly wait
```

```
data = SPIDR<<8; // msbyte of ADC
```

```
while((SPISR&0x20)==0); // wait for SPTEF
```

```
SPIDR = 0; // start SPI
```

```
while((SPISR&0x80)==0); // gadfly wait
```

```
data += SPIDR; // lsbyte of ADC
```

```
PTM |= 0x08; // PM3=CS=1
```

```
return data>>3;} // right justify
```

Page 366-7, Program 7.9, add this line in six places.

```
while((SPISR&0x20)==0); // wait for SPTEF
```

Program should be

```

#define SPIF 0x80
void out8(char code){ unsigned char dummy;
// assumes DDRM bit 4 is 1, output
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = code;
while((SPISR&SPIF)==0); // gadfly wait for SPIF
dummy = SPIDR;} // clear SPIF
void out9(short code){ unsigned char dummy;
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = code; // lsbyte
while((SPISR&SPIF)==0); // gadfly wait for SPIF
dummy = SPIDR; // clear SPIF
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = (code>>8); // msbyte
while((SPISR&SPIF)==0); // gadfly wait for SPIF
dummy = SPIDR;} // clear SPIF
unsigned char in8(void){ short n; unsigned char result;
DDRM &=~0x10; // PM4=DQ input
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = 0; // start shift register
while((SPISR&SPIF)==0); // gadfly wait for SPIF
result = SPIDR; // get data, clear SPIF
DDRM |= 0x10; // PM4=DQ output
return result;}
short in9(void){ short result;
DDRM &=~0x10; // PM4=DQ input
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = 0; // start shift register
while((SPISR&SPIF)==0); // gadfly wait for SPIF
result = SPIDR; // get LS data, clear SPIF
while((SPISR&0x20)==0); // wait for SPTEF
SPIDR = 0; // start shift register
while((SPISR&SPIF)==0); // gadfly wait for SPIF
if(SPIDR&0x01) // get MS data, clear SPIF
result |= 0xFF00; // negative
else
result &=~0xFF00; // positive
DDRM |= 0x10; // PM4=DQ output
return result;}

```

Page 395, Observation at bottom of the page. Change 5 ms to 1  $\mu$ s.

Page 410, Program 8.17, change  
// MC68HC812A4  
to  
// MC9S12C32

Page 410, Program 8.17, add this line in three places.

```
while((SPISR&0x20)==0); // wait for SPTEF
```

add this line in three places.

```
dummy = SPIDR;
```

add this line

```
unsigned char dummy;
```

Program should be

```
void LED_out(unsigned char data[3]){ unsigned char dummy;
PTM &= ~0x08; // ENABLE=0
```

```

while((SPISR&0x20)==0);
SPIDR = data[2]; // send MSbyte
while(SPISR&0x80)==0){};
dummy = SPIDR;
while((SPISR&0x20)==0);
SPIDR = data[1]; // send middle byte
while(SPISR&0x80)==0){};
dummy = SPIDR;
while((SPISR&0x20)==0);
SPIDR = data[0]; // send LSbyte
while(SPOSR&0x80)==0){};
dummy = SPIDR;
PTM |= 0x08;} // ENABLE=1

```

Typesetter: if you need space, it can be written as

```

void LED_out(unsigned char data[3]){ char dummy;
PTM &= ~0x08; // ENABLE=0
while((SPISR&0x20)==0);
SPIDR = data[2]; // send MSbyte
while(SPISR&0x80)==0){};
dummy = SPIDR; while((SPISR&0x20)==0);
SPIDR = data[1]; // send middle byte
while(SPISR&0x80)==0){};
dummy = SPIDR; while((SPISR&0x20)==0);
SPIDR = data[0]; // send LSbyte
while(SPOSR&0x80)==0){};
dummy = SPIDR; PTM |= 0x08;} // ENABLE=1

```

Page 420. Add “design” to performance tip. It should read

**Performance Tip:** *A good transistor design is one that the input/output ...*

Page 426. Figure 8.68. Change 1N4003 to 1N914

Page 428. Figure 8.73. Change 1N4003 to 1N914

Page 440. Figure 8.88. Change 1N4003 to 1N914

Page 570. Program 11.10 caption, remove dash in *software*.

*Program 11.10. Software implementation of a sigma-delta ADC.*

Page 683, change (fast to slow, slow to fast, slowing down to speeding up, speeding up to slowing down)

<i>ENL</i>	True if the motor is spinning much too slow
<i>ENS</i>	True if the motor is spinning a little bit too slow
<i>EZE</i>	True if the motor is spinning at the proper speed
<i>EPS</i>	True if the motor is spinning a little bit too fast
<i>EPL</i>	True if the motor is spinning much too fast
<i>DNL</i>	True if the motor speed is slowing down a lot
<i>DNS</i>	True if the motor speed is slowing down a little
<i>DZE</i>	True if the motor speed is remaining the same
<i>DPS</i>	True if the motor speed is speeding up a little
<i>DPL</i>	True if the motor speed is speeding up a lot

To

<i>ENL</i>	True if the motor is spinning much too fast
<i>ENS</i>	True if the motor is spinning a little bit too fast
<i>EZE</i>	True if the motor is spinning at the proper speed
<i>EPS</i>	True if the motor is spinning a little bit too slow
<i>EPL</i>	True if the motor is spinning much too slow

*DNL* True if the motor speed is speeding up a lot  
*DNS* True if the motor speed is speeding up a little  
*DZE* True if the motor speed is remaining the same  
*DPS* True if the motor speed is slowing down a little  
*DPL* True if the motor speed is slowing down a lot

Page 683, Program 13.17

change (fast to slow, slow to fast, increasing to decreasing, decreasing to increasing)

EPL:	ds	1	; speed way too fast
EPS:	ds	1	; speed too fast
EZE:	ds	1	; speed OK
ENS:	ds	1	; speed too slow
ENL:	ds	1	; speed way too slow
DPL:	ds	1	; speed decreasing a lot
DPS:	ds	1	; speed decreasing
DZE:	ds	1	; speed constant
DNS:	ds	1	; speed increasing
DNL:	ds	1	; speed increasing a lot

to

EPL:	ds	1	; speed way too slow
EPS:	ds	1	; speed too slow
EZE:	ds	1	; speed OK
ENS:	ds	1	; speed too fast
ENL:	ds	1	; speed way too fast
DPL:	ds	1	; speed increasing a lot
DPS:	ds	1	; speed increasing
DZE:	ds	1	; speed constant
DNS:	ds	1	; speed decreasing
DNL:	ds	1	; speed decreasing a lot

Page 684, Program 13.17

change (fast to slow, slow to fast, increasing to decreasing, decreasing to increasing)

ep1:	equ	0	; speed way too fast
eps:	equ	1	; speed too fast
eze:	equ	2	; speed ok
ens:	equ	3	; speed too slow
enl:	equ	4	; speed way too slow
dpl:	equ	5	; speed decreasing a lot
dps:	equ	6	; speed decreasing
dze:	equ	7	; speed constant
dns:	equ	8	; speed increasing
dnl:	equ	9	; speed increasing a lot

to

ep1:	equ	0	; speed way too slow
eps:	equ	1	; speed too slow
eze:	equ	2	; speed ok
ens:	equ	3	; speed too fast
enl:	equ	4	; speed way too fast
dpl:	equ	5	; speed increasing a lot
dps:	equ	6	; speed increasing
dze:	equ	7	; speed constant
dns:	equ	8	; speed decreasing
dnl:	equ	9	; speed decreasing a lot

Page 685, Figure 13.27

change (P to N) 8 places, change (N to P) 8 places. New figure is

		$D=X(n)-X(n-1)$				
		<i>DNL</i>	<i>DNS</i>	<i>DZE</i>	<i>DPS</i>	<i>DPL</i>
$E=X^*-X$	<i>ENL</i>	<i>ONL</i>	<i>ONL</i>	<i>ONL</i>		
	<i>ENS</i>	<i>ONL</i>	<i>ONS</i>	<i>ONS</i>		
	<i>EZE</i>	<i>ONL</i>	<i>ONS</i>	<i>OZE</i>	<i>OPS</i>	<i>OPL</i>
	<i>EPS</i>			<i>OPS</i>	<i>OPS</i>	<i>OPL</i>
	<i>EPL</i>			<i>OPL</i>	<i>OPL</i>	<i>OPL</i>

Page 685, Program 13.20, change (p to n) 8 places, change (n to p) 8 places, change (P to N) 8 places, change (N to P) 8 places. New Program is

```

rules:  dc.b  enl, dnl, $FE, onl, $FE          ; if ENL and DNL then ONL
        dc.b  ens, dnl, $FE, onl, $FE          ; if ENS and DNL then ONL
        dc.b  eze, dnl, $FE, onl, $FE          ; if EZE and DNL then ONL
        dc.b  enl, dns, $FE, onl, $FE          ; if ENL and DNS then ONL
        dc.b  ens, dns, $FE, ons, $FE          ; if ENS and DNS then ONS
        dc.b  eze, dns, $FE, ons, $FE          ; if EZE and DNS then ONS
        dc.b  enl, dze, $FE, onl, $FE          ; if ENL and DZE then ONL
        dc.b  ens, dze, $FE, ons, $FE          ; if ENS and DZE then ONS
        dc.b  eze, dze, $FE, oze, $FE          ; if EZE and DZE then OZE
        dc.b  eps, dze, $FE, ops, $FE          ; if EPS and DZE then OPS
        dc.b  epl, dze, $FE, opl, $FE          ; if EPL and DZE then OPL
        dc.b  eze, dps, $FE, ops, $FE          ; if EZE and DPS then OPS
        dc.b  eps, dps, $FE, ops, $FE          ; if EPS and DPS then OPS
        dc.b  eps, dps, $FE, opl, $FE          ; if EPL and DPS then OPL
        dc.b  eze, dpl, $FE, opl, $FE          ; if EZE and DPL then OPL
        dc.b  eps, dpl, $FE, opl, $FE          ; if EPS and DPL then OPL
        dc.b  epl, dpl, $FE, opl, $FE          ; if EPL and DPL then OPL
        dc.b  $FF

```

Page 687, change

```

E:  ds  1  ;temperature error (128 means no error) (units 0.125 F)
To
E:  ds  1  ;error=Measured-Desired (128 means no error) (0.125 F)

```

Page 766 Problem 15.6 change  $\times(n-i)$  to  $x(n-i)$