

```

1 // C:\CEG411\sample_c.c
2 // A sample C program
3 /*
4     This shows a way to comment out multiple lines
5     */
6 #include <mc9s12c32.h> // include a system header file
7
8 int my_init(int); // a function prototype
9
10 int g1, g2; // global variables
11 char c1, c2; // (char: one byte; short, int: two bytes; long, float: 4 bytes)
12
13 void main()
14 {
15     int i, j, k; // local variables
16     unsigned char buf[6] = {'0', '1', 'z', 4, 0xa3, 0x4D}; // array
17
18     i = my_init(20); // function call
19
20     for(i=0; i<6; i++) buf[i] = 0; // for loop (one statement in the loop body)
21
22     for(i=0; i<6; i+=2) { // another for loop (multiple statements)
23         buf[i] += 5;
24         buf[i] = buf[i] + 5;
25     }
26
27     while(1); // infinite loop;
28     while(1) { } // same as the above
29
30     while(1) {
31         i = 1;
32         j = 2;
33     } // another infinite loop
34
35     if(buf[0] == 5) i = 10;
36     else {
37         i = 4;
38         k = 2;
39     }
40
41     putchar('A'); // print a character 'A'
42     putchar('2');
43     puts("Hello World\r"); // same as printf("Hello World\r\n");
44     printf("i = %d, j= %d\r\n", i, j);
45 }
46
47 int my_init(int in1)
48 {
49     int i; // local variable
50
51     i = in1 + 2;
52     return i;
53 }

```

```
1 // C:\ceg411\switch_LED.c
2 // This program turns on LED1 when SW1 is pressed down and turns off LED1
3 // when SW1 is released.
4 // LED2 is similarly controlled by SW2.
5 //
6 // Note: LEDs and switches are all active low.
7 //     LED1 = PA0 (bi-directional)
8 //     LED2 = PB4 (bi-directional)
9 //     SW1  = PE0 (an input pin)
10 //     SW2  = PP5 (bi-directional)
11 #include <mc9s12c32.h>
12
13 void main()
14 {
15     DDRA |= 0x01; // set PA0 as an output pin (for LED1)
16     DDRB |= 0x10; // set PB4 as an output pin (for LED2)
17     DDRP &= ~0x20; // set PP5 as an input pint (for SW2)
18
19     while(1) {
20         if(!(PORTE & 0x01)) // SW1 (i.e., PE0) is pressed down
21             PORTA &= ~0x01; // turn on LED1 (PA0)
22         else
23             PORTA |= 0x01; // turn off LED1
24
25         if(!(PTP & 0x20)) // SW2 (i.e., PP5) is pressed down
26             PORTB &= ~0x10; // turn on LED2 (PB4)
27         else
28             PORTB |= 0x10; // turn off LED2
29     };
30 }
31
```

```

1 // C:\CEG411\tone_interrupt.c
2 // This program produces a 4 KHz square wave and prints
3 //   a "Hello World" message to the screen every second.
4 // It also flashes LED1 at a frequency of 0.5 Hz (i.e.,
5 //   on for one second and off for one second).
6 // Assume that the E clock frequency is 24 MHz.
7 #include <mc9s12c32.h>
8
9 unsigned int count=0;
10
11 #pragma interrupt_handler tc0_isr
12 void tc0_isr()
13 {
14     TC0=TCNT+3000; // set up TC0 and clear C0F flag
15     count++;
16 }
17
18 void main()
19 {
20     TIOS|=0x01; // use channel 0 for output compare
21     TSCR1=0x90; // enable timer and allow fast flag clear
22     TCTL2=0x01; // select toggle as the action for output compare
23     TC0=TCNT+3000; // set up TC0 and clear C0F flag
24     DDRA = 0xff; // set port A as output pins (for LED1)
25
26     *(void (**)( )) 0x0fee = tc0_isr;
27     // *(int *) 0x0fee = (int) tc0_isr;
28
29     INTR_ON();
30     TIE = 0x01; // locally enable TC0 interrupt
31
32     while(1)
33     {
34         if(count == 8000)
35         {
36             puts("Hello World\r"); // same as printf("Hello World\n\r");
37             PORTA ^= 0x01; // toggle PA0 (for LED1)
38             count = 0;
39         }
40     }
41 }
42

```

```

1 // C:\ceg411\ADC_scan.c
2 // This program reads 80 ADC samples from channel 7 and
3 // prints out their values.
4 // The conversion is 8-bit instead of 10-bit.
5 // WARNING: Use an oscilloscope to make sure the analog signal is
6 //           within the 0 to 5 volt range before connecting it to EVB.
7 //           For best observation results, use a 2 KHz sinusoidal wave,
8 //           0 to 5 volt peak to peak.
9 #include <mc9s12c32.h>
10
11 /* This function prints out a single hexadecimal digit
12    The value of the input, ch, should be from 0 to 15. */
13 void print_hex(unsigned char ch)
14 {
15     if((ch >=0) && (ch <= 9)) putchar('0' + ch);
16     else if ((ch >=10) && (ch <= 15)) putchar('A' + ch - 10);
17     else putchar('?');
18 }
19
20 /* This function prints out an unsigned char as
21    two hexadecimal digits (e.g., 2A) */
22 void print_byte(unsigned char ch)
23 {
24     //putchar('0'); putchar('x');
25     print_hex((ch & 0xf0) >> 4);
26     print_hex(ch & 0x0f);
27 }
28
29 void ATD_init(void)
30 {
31     int i;
32
33     ATDCTL2 = 0xe0; // enable ADT and use fast flag clear
34     for(i=0; i<50; i++) asm("nop"); // a delay of at least 5 micro-sec
35     ATDCTL3 = 0x02; // conversion length : 8 samples
36     ATDCTL4 = 0xA5; // 8-bit resolution;
37                 // 14 (=2+4+8) ATD clock cycles/sample for conversion
38                 // 2 (=24/12) MHz ATD clock
39                 // (Modify this register to get faster conversion.)
40 }
41
42 void main()
43 {
44     int i, j;
45     unsigned char buf[80];
46
47     ATD_init();
48     ATDCTL5 = 0x27; // start an A/D conversion, scan, single channel (7)
49
50     for(i=0; i<80; i+=8) {
51         while(!(ATDSTAT0 & 0x80)); // wait until conversion is complete
52         //ATDCTL5 = 0x07; // start the next A/D conversion (NO NEED in scan mode)
53         buf[i] = ATDDR0H; // ignore ATDDR0L since the conversion is 8-bit only
54         buf[i+1] = ATDDR1H;
55         buf[i+2] = ATDDR2H;
56         buf[i+3] = ATDDR3H;
57         buf[i+4] = ATDDR4H;
58         buf[i+5] = ATDDR5H;
59         buf[i+6] = ATDDR6H;
60         buf[i+7] = ATDDR7H;
61     }
62
63     // Print out the buf[] values
64     for(i=0; i<80; i++) {
65         print_byte(buf[i]);
66         putchar(' ');
67         if((i+1)%10==0) { putchar('\r'); putchar('\n'); } // 10 samples each line
68     }
69
70     asm("JMP $FC00"); // return to Monitor
71 }
72

```