

```

1:  /*
2:   Initiating ADC Conversion:
3:   1. Writing to ATD0CTL5
4:   2. Edge on external trigger
5:   3. Level on external trigger
6:
7:   Completed Conversion
8:   1. Reading ATD0STAT1
9:   2. Interrupt when complete
10: */
11:
12: #include <hidef.h>          /* common defines and macros */
13: #include <mc9s12dp512.h>    /* derivative information */
14: #pragma LINK_INFO DERIVATIVE "mc9s12dp512"
15:
16: #define PROCEDURE 2
17: #include "PLL.h"
18: #include "ADC.h"
19: #include "OC.h"
20: #include "lcd.h"
21: #include "temperature.h"
22: #include <stdio.h>
23:
24: #if PROCEDURE == 1
25: #include "SCI0.h"
26:
27: unsigned short DataBuffer[100];
28: unsigned short Count=0;
29: void back(void) {
30:     unsigned short data;
31:     if(Count<100) {
32:         data = ADC0_In(0x80); // your program that samples channel 5
33:         DataBuffer[Count++] = data;
34:     }
35: }
36: void main(void){unsigned short i;
37:     PLL_Init(); // 24 MHz
38:     ADC0_Init(); // your module
39:     SCI0_Init(115200); // SCI output to PC
40:     OC0_Init(1000,&back); // your module sampling at 1000 Hz
41:
42:     asm cli;
43:     while(Count<100) {}; // copy ADC to buffer in background
44:     for(i=0; i<100; i++) {
45:         SCI0_OutUDec(DataBuffer[i]); SCI0_OutChar(10);SCI0_OutChar(13);
46:     }
47:     for(;;){};
48: }
49:
50: #else
51: #include "FIFO.h"
52:
53: void getData(void) {
54:     unsigned short data;
55:     data = ADC0_In(0x82);
56:     while(!Fifo_Put(data)) {}
57: }
58:
59: void main(void) {
60:     char buffer[10] = "";
61:     // Initialize needed modules
62:     DDRP |= 0x80;
63:     PLL_Init();
64:     Fifo_Init();
65:     ADC0_Init();
66:     OC0_Init(100, &getData);
67:     LCD_Open();
68:
69:     LCD_Clear();
70:     sprintf(buffer, "      %cC", 223);
71:     LCD_OutString(buffer);
72:
73:     asm cli
74:
75:     for(;;) {
76:         unsigned short data;
77:         unsigned short temperature;
78:         while(!Fifo_Get(&data)) {}

```

```
79:
80:     temperature = Temp_Data(data);
81:     sprintf(buffer, "%2d.%02d", temperature/100, temperature%100);
82:     //sprintf(buffer, "%4d", data);
83:     LCD_GoTo(0,0);
84:     LCD_OutString(buffer);
85:
86: }
87: }
88: #endif
```

```
1: #include <hidef.h>          /* common defines and macros */
2: #include <mc9s12dp512.h>     /* derivative information */
3: #pragma LINK_INFO DERIVATIVE "mc9s12dp512"
4:
5: unsigned short TempTable[];
6:
7: void ADC0_Init(void);
8: unsigned short ADC0_In(unsigned char channel);
```

```
1: #include "ADC.h"
2:
3: void ADC0_Init(void) {
4:     ATD0CTL2 = 0x80;
5:     ATD0CTL3 = 0x08;
6:     ATD0CTL4 = 0x05;
7: }
8:
9: unsigned short ADC0_In(unsigned char channel) {
10:     unsigned short data;
11:     ATD0CTL5 = channel;
12:     while(!(ATD0STAT1&0x01)) {}
13:     data = ATD0DR0;
14:     return data;
15: }
```

```
1: // filename ***** Fifo.h *****
2: // Header file for the receive FIFO (two versions)
3: // Jonathan W. Valvano 10/1/07
4:
5: // This example accompanies the books
6: //   "Embedded Microcomputer Systems: Real Time Interfacing",
7: //   Thompson, copyright (c) 2006,
8: //   "Introduction to Embedded Microcomputer Systems:
9: //   Motorola 6811 and 6812 Simulation", Brooks-Cole, copyright (c) 2002
10:
11: // Copyright 2006 by Jonathan W. Valvano, valvano@mail.utexas.edu
12: //   You may use, edit, run or distribute this file
13: //   as long as the above copyright notice remains
14:
15:
16:
17: #include <hidef.h>          /* common defines and macros */
18: #include <mc9s12dp512.h>    /* derivative information */
19: #pragma LINK_INFO DERIVATIVE "mc9s12dp512"
20:
21: #define FIFOSIZE 24
22: /* Number of characters in the Fifo
23:    the FIFO is full when it has FifoSize-1 characters */
24: void Fifo_Init(void);
25:
26: /*-----Fifo_Put-----
27:    Enter one character into the fifo
28:    Inputs: 8-bit data
29:    Outputs: true if data is properly saved,
30:            false if data not saved because it was previously full*/
31: int Fifo_Put(unsigned short data);
32:
33: /*-----Fifo_Get-----
34:    Remove one character from the fifo
35:    Inputs: pointer to place to return 8-bit data
36:    Outputs: true if data is valid,
37:            false if fifo was empty at the time of the call*/
38: int Fifo_Get(unsigned short *dataptr);
39:
40: /*-----Fifo_Status-----
41:    Check the status of the fifo
42:    Inputs: none
43:    Outputs: true if there is any data in the fifo */
44: int Fifo_Status(void);
45:
```

```

1: // filename *****Fifo.c*****
2: // Two implementations of a FIFO
3: // Jonathan W. Valvano 10/1/07
4:
5: // This example accompanies the books
6: //   "Embedded Microcomputer Systems: Real Time Interfacing",
7: //   Thompson, copyright (c) 2006,
8: //   "Introduction to Embedded Microcomputer Systems:
9: //   Motorola 6811 and 6812 Simulation", Brooks-Cole, copyright (c) 2002
10:
11: // Copyright 2007 by Jonathan W. Valvano, valvano@mail.utexas.edu
12: //   You may use, edit, run or distribute this file
13: //   as long as the above copyright notice remains
14:
15: #include "Fifo.h"
16:
17: /* Number of characters in the Fifo
18:    the FIFO is full when it has FifoSize-1 characters */
19: unsigned short static volatile *PutPt;    /* Pointer of where to put next */
20: unsigned short static volatile *GetPt;    /* Pointer of where to get next */
21:     /* FIFO is empty if PutPt=GetPt */
22:     /* FIFO is full  if PutPt+1=GetPt */
23: unsigned short static Fifo[FIFOSIZE];    /* The statically allocated fifo data */
24:
25: /*-----Fifo_Init-----
26:    Initialize fifo to be empty
27:    Inputs: none
28:    Outputs: none */
29: void Fifo_Init(void){
30:     unsigned char SaveCCR;
31:     asm tpa
32:     asm staa SaveCCR
33:     asm sei          /* make atomic
34:     PutPt=GetPt=&Fifo[0];    /* Empty when PutPt=GetPt
35:     asm ldaa SaveCCR
36:     asm tap          /* end critical section
37: }
38:
39: /*-----Fifo_Put-----
40:    Enter one character into the fifo
41:    Inputs: 8-bit data
42:    Outputs: true if data is properly saved,
43:            false if data not saved because it was previously full*/
44: int Fifo_Put(unsigned short data){
45:     unsigned short volatile *tempPt;
46:     tempPt = PutPt;
47:     *(tempPt) = data;        /* try to Put data into fifo
48:     tempPt++;
49:     if(tempPt == &Fifo[FIFOSIZE]){ // need to wrap?
50:         tempPt = &Fifo[0];
51:     }
52:     if(tempPt == GetPt){
53:         return(0);          /* Failed, fifo was previously full
54:     }
55:     else{
56:         PutPt = tempPt;    /* Success, so update pointer
57:         return(1);
58:     }
59: }
60:
61: /*-----Fifo_Get-----
62:    Remove one character from the fifo
63:    Inputs: pointer to place to return 8-bit data
64:    Outputs: true if data is valid,
65:            false if fifo was empty at the time of the call*/
66: int Fifo_Get(unsigned short *dataptr){
67:     if(PutPt == GetPt){
68:         return(0);          /* Empty if PutPt=GetPt
69:     }
70:     else{
71:         *dataptr = *(GetPt);    /* return by reference
72:         GetPt++;                /* removes data from fifo
73:         if(GetPt == &Fifo[FIFOSIZE]){
74:             GetPt = &Fifo[0];    /* wrap
75:         }
76:         return(1);
77:     }
78: }

```

```
79:
80: /*-----Fifo_Status-----
81:    Check the status of the fifo
82:    Inputs: none
83:    Outputs: true if there is any data in the fifo */
84: int Fifo_Status(void){
85:     return (PutPt != GetPt);
86: }
```

```
1: #include <hidef.h>          /* common defines and macros */
2: #include <mc9s12dp512.h>     /* derivative information */
3: #pragma LINK_INFO DERIVATIVE "mc9s12dp512"
4:
5: void OC0_Init(unsigned short freq, void (*func)(void));
```



```
1: #include "OC.h"
2:
3: unsigned short frequency;
4: void (*function)(void);
5:
6: interrupt 8 void TOC0handler(void){ // executes at 100 Hz
7:     TFLG1 = 0x01;           // acknowledge OC0
8:     PTP |= 0x80;
9:     (*function)();
10:    TC0 += 187500/frequency;
11: }
12:
13: //-----OC_Init0-----
14: // arm output compare 0 for 100Hz periodic interrupt
15: // Input: none
16: // Output: none
17: void OC0_Init(unsigned short freq, void (*func)(void)) {
18:     frequency = freq;
19:     function = func;
20:     TIOS |= 0x01; // activate TC0 as output compare
21:     TIE  |= 0x01; // arm OC0
22:     TSCR1 = 0x80; // Enable TCNT, 24MHz boot mode, 8MHz in run mode
23:     TSCR2 = 0x07; // divide by 64 TCNT prescale, TOI disarm
24:     PACTL = 0;    // timer prescale used for TCNT
25:     TC0   = TCNT + 50; // first interrupt right away
26: }
```

```
1: unsigned short Temp_Data(unsigned short adc);
```

```

1: unsigned short const ADCdata[53]={0,60,72,85,97,110,122,135,148,161,174,
2:    187,201,214,227,241,255,268,282,296,310,
3:    324,338,352,366,381,395,409,424,438,453,
4:    468,482,497,512,527,541,556,571,586,601,
5:    616,631,646,662,677,692,707,722,737,752,767,1024};
6:
7:
8: unsigned short const Tdata[53]={4000,4000,3960,3920,3880,3840,3800,3760,3720,3680,3640,
9:    3600,3560,3520,3480,3440,3400,3360,3320,3280,3240,
10:   3200,3160,3120,3080,3040,3000,2960,2920,2880,2840,
11:   2800,2760,2720,2680,2640,2600,2560,2520,2480,2440,
12:   2400,2360,2320,2280,2240,2200,2160,2120,2080,2040,2000,2000};
13:
14:
15: unsigned short Temp_Data(unsigned short adc) {
16:     unsigned short temp;
17:
18:     asm ldd adc
19:     asm Lookup: ldx #ADCdata // first find x1<=xL<x2
20:     asm ldy #Tdata
21:     asm lookx1: cpd 2,x // check xL<x2
22:     asm blo found // stops when X points to x1
23:     asm leax 2,x
24:     asm leay 2,y
25:     asm bra lookx1
26:     asm found: subd 0,x // xL-x1
27:     asm pshd
28:     asm ldd 2,x // x2
29:     asm subd 0,x // D=x2-x1
30:     asm tfr D,X // X=x2-x1
31:     asm puld // D=(xL-x1)
32:     asm fdiv // X=(65536*(xL-x1))/(x2-x1)
33:     asm tfr X,D
34:     asm tfr A,B
35:     // B=(256*(xL-x1))/(x2-x1)
36:     // Y=>y1,y2
37:     asm etbl 0,y
38:     asm std temp
39:
40:     return temp;
41: }

```