ADC EXAMPLE 1

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral using polling and

; single conversion mode.

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #INCH\_0, &ADC10CTL1 ; Let's read the temperature

BIS.B #0x01, &ADC10AE0 ; Enable analog input A0

BIS.W #ENC, &ADC10CTL0 ; Enable conversion

main: NOP ; main program

BIT.W #ADC10BUSY, &ADC10CTL1 ; Test the BUSY flag

JNZ main ; If conversion going on jump back

BIS.W #ADC10SC, &ADC10CTL0 ; Start a conversion

JMP main ; jump to current location '$'

; (endless loop)

END

ADC EXAMPLE2

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral using continuous

; conversion mode

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #MSC, &ADC10CTL0 ; Enable multiple conversions

BIS.W #CONSEQ\_2, &ADC10CTL1 ; Single channel continuous

BIS.W #INCH\_0, &ADC10CTL1 ; Let's read the temperature

BIS.B #0x01, &ADC10AE0 ; Enable analog input A0

BIS.W #ENC, &ADC10CTL0 ; Enable conversion

BIS.W #ADC10SC, &ADC10CTL0 ; Start conversion

main: NOP ; main program

JMP main ; jump to main

; (endless loop)

END

ADC EXAMPLE 3

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral using continuous

; conversion mode with the internal voltage reference at 1.5V.

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #MSC, &ADC10CTL0 ; Enable multiple conversions

BIS.W #REFON, &ADC10CTL0 ; Turn on the voltage reference

BIS.W #SREF\_1, &ADC10CTL0 ; Set the reference voltage as ref. input

BIS.W #CONSEQ\_2, &ADC10CTL1 ; Single channel continuous

BIS.W #INCH\_0, &ADC10CTL1 ; Let's read the temperature

BIS.B #0x01, &ADC10AE0 ; Enable analog input A0

BIS.W #ENC, &ADC10CTL0 ; Enable conversion

BIS.W #ADC10SC, &ADC10CTL0 ; Start conversion

main: NOP ; main program

JMP main ; jump to main

; (endless loop)

END

ADC EXAMPLE 4

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral by reading A0 and A1

; using sequence of channels mode with the internal voltage reference at 1.5V.

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

#define BASE\_MEM 0x200

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #MSC, &ADC10CTL0 ; Self trigger to obtain next sample

; from the following channel.

BIS.W #REFON, &ADC10CTL0 ; Turn on the voltage reference

BIS.W #SREF\_1, &ADC10CTL0 ; Set the reference voltage as ref. input

BIS.W #CONSEQ\_1, &ADC10CTL1 ; Single Sequence of channels

BIS.W #INCH\_1, &ADC10CTL1 ; Let's set the start channel

BIS.B #0x03, &ADC10AE0 ; Enable analog input A0 and A1

MOV.B #0x02, &ADC10DTC1 ; Make 2 conversions

MOV.W #0x00, &BASE\_MEM

MOV.W #0x00, &BASE\_MEM+2

main: NOP ; main program

MOV.W #BASE\_MEM, &ADC10SA ; Get DTC ready to start transfer

BIS.W #ENC+ADC10SC, &ADC10CTL0; Start sampling

CALL #waitForADC ; Sit here while busy

JMP main ; jump to main

; (endless loop)

waitForADC:

loop:

BIT.W #ADC10BUSY, &ADC10CTL1

JNZ loop

RET

END

ADC EXAMPLE 5

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral using interrupts and

; single conversion mode.

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

#define VARIABLE 0x200

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

ORG 0FFE4h ; Set the address of the P1 ISR in the

; interrupt vector table.

DC16 P1ISR ; Address is 16-bit.

ORG 0FFEAh ; ADC10 Vector

DC16 ADC10ISR

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #INCH\_0, &ADC10CTL1 ; Let's read the temperature

BIS.B #0x01, &ADC10AE0 ; Enable analog input A0

BIS.W #ENC, &ADC10CTL0 ; Enable conversion

BIS.W #ADC10IE, &ADC10CTL0 ; Enable ADC10 Interrupt

; Configure the entire port as input

MOV.B #0x00, &P1DIR

; Make sure the pins are set to GPIO mode.

MOV.B #0x00, &P1SEL

MOV.B #0x00, &P1SEL2

; Clear the interrupt flag before enabling interrupts.

BIC.B #0x40, &P1IFG

; Enable interrupts on P1.6 whenever falling edge occurs.

BIS.B #0x40, &P1IE

BIS.B #0x40, &P1IES

MOV.W #0x0000, &VARIABLE ; Set variable to 0

; Enable global interrupts

EINT

main: NOP ; main program

INC.W R4

JMP main ; jump to main

; (endless loop)

ADC10ISR:

MOV.W &ADC10MEM, &VARIABLE

RETI

P1ISR:

BIS.W #ADC10SC, &ADC10CTL0; Start ADC conversion

; Clear the interrupt flag before returning from interrupt.

BIC.B #0x40, &P1IFG

RETI

END

ADC EXAMPLE 6

#include "msp430.h" ; #define controlled include file

; This example illustrates the ADC10 hardware peripheral using interrupts and

; single conversion mode.

NAME main ; module name

PUBLIC main ; make the main label vissible

; outside this module

; Define labels to the memory locations that hold the values to display.

#define THOU 0x200

#define HUND 0x201

#define TENS 0x202

#define ONES 0x203

#define DIS2 0x20

#define DIS3 0x10

#define DIS4 0x08

#define DIS5 0x04

#define ALLD 0x3C

#define SEG\_DELAY 0x03F0

ORG 0FFFEh

DC16 init ; set reset vector to 'init' label

ORG 0FFE4h ; Set the address of the P1 ISR in the

; interrupt vector table.

DC16 P1ISR ; Address is 16-bit.

ORG 0FFEAh ; ADC10 Vector

DC16 ADC10ISR

RSEG CSTACK ; pre-declaration of segment

RSEG CODE ; place program in 'CODE' segment

init: MOV #SFE(CSTACK), SP ; set up stack

MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer

BIS.W #ADC10SHT\_2, &ADC10CTL0 ; Sample and hold for 16 clock cycles

BIS.W #ADC10ON, &ADC10CTL0 ; ADC10 on

BIS.W #INCH\_0, &ADC10CTL1 ; Let's read the temperature

BIS.B #0x01, &ADC10AE0 ; Enable analog input A0

BIS.W #ENC, &ADC10CTL0 ; Enable conversion

BIS.W #ADC10IE, &ADC10CTL0 ; Enable ADC10 Interrupt

MOV.B #0xFF, &P2DIR ; Configure the entirety of P2 as output

; Make sure the pins are set to GPIO mode.

MOV.B #0x00, &P2SEL

MOV.B #0x00, &P2SEL2

; Configure P1.2,3,4, and 5 as outputs

MOV.B #00111100b, &P1DIR

; Make sure the pins are set to GPIO mode.

MOV.B #0x00, &P1SEL

MOV.B #0x00, &P1SEL2

; Clear the interrupt flag before enabling interrupts.

BIC.B #0x40, &P1IFG

; Enable interrupts on P1.6 whenever falling edge occurs.

BIS.B #0x40, &P1IE

BIS.B #0x40, &P1IES

; Initialize our idexes to 0x00

MOV.B #0x00, &ONES

MOV.B #0x00, &TENS

MOV.B #0x00, &HUND

MOV.B #0x00, &THOU

; Enable global interrupts

EINT

main: NOP ; main program

; Ones place.

BIC.B #ALLD, &P1OUT ; Turn off all displays

BIS.B #DIS5, &P1OUT ; Turn on DIS5

MOV.B &ONES, R4 ; Move the number to display to R4

MOV.B ZERO(R4), &P2OUT ; Get the pattern from the lookup table

MOV.W #SEG\_DELAY, R4 ; Pass the delay amount to DelayMS

; though R4

CALL #DelayMS

BIC.B #ALLD, &P1OUT ; Turn off all displays

BIS.B #DIS4, &P1OUT ; Turn on DIS4

MOV.B &TENS, R4 ; Move the number to display to R4

MOV.B ZERO(R4), &P2OUT ; Get the pattern from the lookup table

MOV.W #SEG\_DELAY, R4 ; Pass the delay amount to DelayMS

; though R4

CALL #DelayMS

BIC.B #ALLD, &P1OUT ; Turn off all displays

BIS.B #DIS3, &P1OUT ; Turn on DIS3

MOV.B &HUND, R4 ; Move the number to display to R4

MOV.B ZERO(R4), &P2OUT ; Get the pattern from the lookup table

MOV.W #SEG\_DELAY, R4 ; Pass the delay amount to DelayMS

; though R4

CALL #DelayMS

BIC.B #ALLD, &P1OUT ; Turn off all displays

BIS.B #DIS2, &P1OUT ; Turn on DIS3

MOV.B &THOU, R4 ; Move the number to display to R4

MOV.B ZERO(R4), &P2OUT ; Get the pattern from the lookup table

MOV.W #SEG\_DELAY, R4 ; Pass the delay amount to DelayMS

; though R4.

CALL #DelayMS

JMP main ; jump to main

; (endless loop)

ADC10ISR:

MOV.W &ADC10MEM, R6

MOV.W R6, R7

AND.W #0x000F, R7 ; LSN

MOV.B R7, &ONES

MOV.W R6, R7

RRC.W R7 ; Shift right four times to get the

RRC.W R7 ; next nibble.

RRC.W R7

RRC.W R7

AND.W #0x000F, R7 ; Mask off unwanted data

MOV.B R7, &TENS

MOV.W R6, R7

RRC.W R7 ; Shift right eight times to get the

RRC.W R7 ; next nibble.

RRC.W R7

RRC.W R7

RRC.W R7

RRC.W R7

RRC.W R7

RRC.W R7

AND.W #0x000F, R7 ; Mask off unwanted data

MOV.B R7, &HUND

RETI

P1ISR:

BIS.W #ADC10SC, &ADC10CTL0; Start ADC conversion

; Clear the interrupt flag before returning from interrupt.

BIC.B #0x40, &P1IFG

RETI

; Lookup table

ZERO: DC8 0x3F

ONE: DC8 0x06

TWO: DC8 0x5B

THREE: DC8 0x4F

FOUR: DC8 0x66

FIVE: DC8 0x6D

SIX: DC8 0x7D

SEVEN: DC8 0x07

EIGHT: DC8 0x7F

NINE: DC8 0x67

LA: DC8 0x77

LB: DC8 0x7C

LC: DC8 0x39

LD: DC8 0x5E

LE: DC8 0x79

LF: DC8 0x71

COLON: DC8 0x80

PAD: DC8 0x00

DelayMS:

DEC R4

JNZ DelayMS

RET

END