;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;; FILENAME: DUALADC\_1INT.asm

;; Version: 2.2, Updated on 2010/12/27 at 15:27:25

;; Generated by PSoC Designer 5.4.3191

;;

;; DESCRIPTION: DualADC Interrupt Service Routines.

;;

;;-----------------------------------------------------------------------------

;; Copyright (c) Cypress Semiconductor 2010. All Rights Reserved.

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include "m8c.inc"

include "memory.inc"

include "DUALADC\_1.inc"

;-----------------------------------------------

; Global Symbols

;-----------------------------------------------

export \_DUALADC\_1\_CNT1\_ISR

export \_DUALADC\_1\_CNT2\_ISR

export \_DUALADC\_1\_PWM16\_ISR

export DUALADC\_1\_cCounter1U

export DUALADC\_1\_cCounter2U

export \_DUALADC\_1\_iResult1

export DUALADC\_1\_iResult1

export \_DUALADC\_1\_iResult2

export DUALADC\_1\_iResult2

export \_DUALADC\_1\_bfStatus

export DUALADC\_1\_bfStatus

export DUALADC\_1\_bSampC

;-----------------------------------------------

; Variable Allocation

;-----------------------------------------------

AREA InterruptRAM(RAM, REL, CON)

DUALADC\_1\_cCounter1U: BLK 1 ;The Upper byte of the Counter

DUALADC\_1\_cCounter2U: BLK 1 ;The Upper byte of the Counter

\_DUALADC\_1\_iResult1:

DUALADC\_1\_iResult1: BLK 2 ;A/D value

\_DUALADC\_1\_iResult2:

DUALADC\_1\_iResult2: BLK 2 ;A/D value

\_DUALADC\_1\_bfStatus:

DUALADC\_1\_bfStatus: BLK 1 ;Data Valid Flag

DUALADC\_1\_bSampC: BLK 1 ;# of times to run A/D

;@PSoC\_UserCode\_INIT@ (Do not change this line.)

;---------------------------------------------------

; Insert your custom declarations below this banner

;---------------------------------------------------

;------------------------

; Includes

;------------------------

;------------------------

; Constant Definitions

;------------------------

;------------------------

; Variable Allocation

;------------------------

;---------------------------------------------------

; Insert your custom declarations above this banner

;---------------------------------------------------

;@PSoC\_UserCode\_END@ (Do not change this line.)

;-----------------------------------------------

; EQUATES and TABLES

;-----------------------------------------------

; Constant Definitions

LowByte: equ 1

HighByte: equ 0

AREA UserModules (ROM, REL)

.LITERAL

DUALADC\_1MaxNegX4Table:

; Bits 7 8 9 10 11 12 13

DB FFh, FEh, FCh, F8h, F0h, E0h, C0h

DUALADC\_1MaxPosX4Table:

IF (DUALADC\_1\_DATA\_FORMAT)

; Bits (signed) 7 8 9 10 11 12 13

DB 01h, 02h, 04h, 08h, 10h, 20h, 40h

ELSE

; Bits (unsigned) 7 8 9 10 11 12 13

DB 02h, 04h, 08h, 10h, 20h, 40h, 80h

ENDIF

.ENDLITERAL

;-----------------------------------------------------------------------------

; FUNCTION NAME: \_DUALADC\_1\_CNT1\_ISR (Counter8 Interrupt)

;

;

; DESCRIPTION:

; Increment the upper (software) half on the counter whenever the

; lower (hardware) half of the counter underflows. This counter

; should start out at the most negative value (0xFF).

;

;-----------------------------------------------------------------------------

;

\_DUALADC\_1\_CNT1\_ISR:

inc [DUALADC\_1\_cCounter1U]

;@PSoC\_UserCode\_BODY\_1@ (Do not change this line.)

;---------------------------------------------------

; Insert your custom code below this banner

;---------------------------------------------------

; NOTE: interrupt service routines must preserve

; the values of the A and X CPU registers.

;---------------------------------------------------

; Insert your custom code above this banner

;---------------------------------------------------

;@PSoC\_UserCode\_END@ (Do not change this line.)

reti

;-----------------------------------------------------------------------------

; FUNCTION NAME: \_DUALADC\_1\_CNT2\_ISR (Counter8 Interrupt)

;

;

; DESCRIPTION:

; Increment the upper (software) half on the counter whenever the

; lower (hardware) half of the counter underflows. This counter

; should start out at the most negative value (0xFF).

;

;-----------------------------------------------------------------------------

;

\_DUALADC\_1\_CNT2\_ISR:

inc [DUALADC\_1\_cCounter2U]

;@PSoC\_UserCode\_BODY\_2@ (Do not change this line.)

;---------------------------------------------------

; Insert your custom code below this banner

;---------------------------------------------------

; NOTE: interrupt service routines must preserve

; the values of the A and X CPU registers.

;---------------------------------------------------

; Insert your custom code above this banner

;---------------------------------------------------

;@PSoC\_UserCode\_END@ (Do not change this line.)

reti

;-----------------------------------------------------------------------------

; FUNCTION NAME: \_DUALADC\_1\_PWM16\_ISR (PWM16 Interrupt)

;

; DESCRIPTION:

; This ISR is called when the ADC has completed and integrate cycle.

; The ADC value is calculated and stored in a global location before

; the end of the ISR.

;

;-----------------------------------------------------------------------------

;

\_DUALADC\_1\_PWM16\_ISR:

and reg[DUALADC\_1\_bCounter1\_CR0], ~DUALADC\_1\_fDBLK\_ENABLE ; Disable Counter

and reg[DUALADC\_1\_bCounter2\_CR0], ~DUALADC\_1\_fDBLK\_ENABLE ; Disable Counter

IF DUALADC\_1\_NoAZ

or reg[DUALADC\_1\_bfADC1cr2], DUALADC\_1\_fAutoZero ; Put Integrator in AutoZero

or reg[DUALADC\_1\_bfADC2cr2], DUALADC\_1\_fAutoZero ; Put Integrator in AutoZero

ENDIF

or reg[DUALADC\_1\_bfADC1cr3],DUALADC\_1\_fFSW0 ; Put Integrator in reset

or reg[DUALADC\_1\_bfADC2cr3],DUALADC\_1\_fFSW0 ; Put Integrator in reset

; Enable interrupts for a short period of time just in case.

; Make sure we didn't have a counter interrupt ready to fire

M8C\_EnableGInt

nop ; Wait a couple cycles

M8C\_DisableGInt ; Disable interrupt, read to complete processing

push A ; Save the Accumulator

; Get ADC1 result

mov A,reg[DUALADC\_1\_bCount1] ; Read counter value (Bogus read puts value in Period register)

mov A,reg[DUALADC\_1\_bCompare1] ; Read counter value

dec A ; Decrement by one to make sure we didn't miss a count

cpl A ; Invert the value

jnc DUALADC\_1\_INT\_CALCV1 ; if carry, then inc MSB as well

inc [DUALADC\_1\_cCounter1U]

DUALADC\_1\_INT\_CALCV1:

mov [(DUALADC\_1\_iResult1 + LowByte)],A ; Store LSB value

mov A, [DUALADC\_1\_cCounter1U] ; Store MSB from temp counter

mov [(DUALADC\_1\_iResult1 + HighByte)],A

; Get ADC2 result

mov A,reg[DUALADC\_1\_bCount2] ; Read counter value (Bogus read puts value in Period register)

mov A,reg[DUALADC\_1\_bCompare2] ; Read counter value

dec A ; Decrement by one to make sure we didn't miss a count

cpl A ; Invert the value

jnc DUALADC\_1\_INT\_CALCV2 ; if carry, then inc MSB as well

inc [DUALADC\_1\_cCounter2U]

DUALADC\_1\_INT\_CALCV2:

mov [(DUALADC\_1\_iResult2 + LowByte)],A ; Store LSB value

mov A, [DUALADC\_1\_cCounter2U] ; Store MSB from temp counter

mov [(DUALADC\_1\_iResult2 + HighByte)],A

; The new value has been stored,

; so get counters ready for next reading first.

mov reg[DUALADC\_1\_bPeriod1], ffh ; Initialize counter1 to FF - Set to overflow after 256 counts

mov reg[DUALADC\_1\_bPeriod2], ffh ; Initialize counter2 to FF - Set to overflow after 256 counts

or reg[DUALADC\_1\_bCounter1\_CR0],DUALADC\_1\_fDBLK\_ENABLE ; Enable Counter1

or reg[DUALADC\_1\_bCounter2\_CR0],DUALADC\_1\_fDBLK\_ENABLE ; Enable Counter2

IF (DUALADC\_1\_DATA\_FORMAT) ; Only check for Negative numbers if SIGNED result

mov A, [DUALADC\_1\_bfStatus] ; Get Status with Resolution

and A, DUALADC\_1\_bRES\_MASK ; Mask of resolution

index DUALADC\_1MaxNegX4Table ; Get Maximum negative value from table

mov [DUALADC\_1\_cCounter1U], A ; Place result back into MSB of counter

mov [DUALADC\_1\_cCounter2U], A ; Place result back into MSB of counter

ELSE

mov [DUALADC\_1\_cCounter1U], 00h ; Place result back into MSB of counter

mov [DUALADC\_1\_cCounter2U], 00h ; Place result back into MSB of counter

ENDIF

;@PSoC\_UserCode\_BODY\_3@ (Do not change this line.)

;---------------------------------------------------

; If the input is muxed with multiple inputs

; this is a good place to change inputs.

; Insert your custom code below this banner

;---------------------------------------------------

; NOTE: interrupt service routines must preserve

; the values of the A and X CPU registers. At this

; point A has been preserved on the stack and will

; be restored before the return from interrupt.

; However, if you use the X register, you must

; preserve its value and restore it here.

;---------------------------------------------------

; Insert your custom code above this banner

;---------------------------------------------------

;@PSoC\_UserCode\_END@ (Do not change this line.)

and reg[DUALADC\_1\_bfADC1cr3],~DUALADC\_1\_fFSW0 ; Take Integrator out of reset

and reg[DUALADC\_1\_bfADC2cr3],~DUALADC\_1\_fFSW0 ; Take Integrator out of reset

IF DUALADC\_1\_NoAZ

and reg[DUALADC\_1\_bfADC1cr2],~DUALADC\_1\_fAutoZero ; Take Integrator out of AutoZero

and reg[DUALADC\_1\_bfADC2cr2],~DUALADC\_1\_fAutoZero ; Take Integrator out of AutoZero

ENDIF

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;M8C\_EnableGInt ; May want to re-enable interrupts at this point,

; ; if stack space isn't at a premium.

; NOTE: this will make system more responsive but, will increase the

; overall processing time of the A/D calctime. If an interrupt is

; taken, it must return within the specified CalcTime to guarantee

; successful acquisition of the next byte.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; ADC1

IF (DUALADC\_1\_DATA\_FORMAT) ; Only check for Negative numbers if SIGNED result

; Negative Overflow Check

tst [(DUALADC\_1\_iResult1 + HighByte)],80h

jnz DUALADC\_1\_NOT\_2POVFL1

ENDIF

; Postive Overflow Check

; Get MSB of Max Positive value x4 + 1

mov A,[DUALADC\_1\_bfStatus] ; Get Status with Resolution

and A,DUALADC\_1\_bRES\_MASK ; Mask of resolution normalized to 0

index DUALADC\_1MaxPosX4Table ; Get Maximum positive value x4 + 1 from table

push A

and A, [(DUALADC\_1\_iResult1 + HighByte)]

jz DUALADC\_1\_NOT\_POVFL1

; Positive overflow, fix it - set to Max Positive + 1

pop A

sub A, 01h

; Force most positive \* 4 into result

mov [(DUALADC\_1\_iResult1 + HighByte)], A

mov [(DUALADC\_1\_iResult1 + LowByte)], ffh

jmp DUALADC\_1\_NOT\_2POVFL1

DUALADC\_1\_NOT\_POVFL1:

pop A

DUALADC\_1\_NOT\_2POVFL1:

asr [(DUALADC\_1\_iResult1 + HighByte)] ; Shift MSB and LSB right twice to divide by four

rrc [(DUALADC\_1\_iResult1 + LowByte)] ; Remember digital clock 4 times analog clock

asr [(DUALADC\_1\_iResult1 + HighByte)]

rrc [(DUALADC\_1\_iResult1 + LowByte)]

; ADC2

IF (DUALADC\_1\_DATA\_FORMAT) ; Only check for Negative numbers if SIGNED result

; Negative Overflow Check

tst [(DUALADC\_1\_iResult2 + HighByte)],80h

jnz DUALADC\_1\_NOT\_2POVFL2

ENDIF

; Postive Overflow Check

; Get MSB of Max Positive value x4 + 1

mov A,[DUALADC\_1\_bfStatus] ; Get Status with Resolution

and A,DUALADC\_1\_bRES\_MASK ; Mask of resolution normalized to 0

index DUALADC\_1MaxPosX4Table ; Get Maximum positive value x4 + 1 from table

push A

and A, [(DUALADC\_1\_iResult2 + HighByte)]

jz DUALADC\_1\_NOT\_POVFL2

; Positive overflow, fix it - set to Max Positive + 1

pop A

sub A, 01h

; Force most positive \* 4 into result

mov [(DUALADC\_1\_iResult2 + HighByte)], A

mov [(DUALADC\_1\_iResult2 + LowByte)], ffh

jmp DUALADC\_1\_NOT\_2POVFL2

DUALADC\_1\_NOT\_POVFL2:

pop A

DUALADC\_1\_NOT\_2POVFL2:

asr [(DUALADC\_1\_iResult2 + HighByte)] ; Shift MSB and LSB right twice to divide by four

rrc [(DUALADC\_1\_iResult2 + LowByte)] ; Remember digital clock 4 times analog clock

asr [(DUALADC\_1\_iResult2 + HighByte)]

rrc [(DUALADC\_1\_iResult2 + LowByte)]

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Data is ready at this point.

; If processing Data at Interrupt level - add

; User Code to handle the data

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;@PSoC\_UserCode\_BODY\_4@ (Do not change this line.)

;---------------------------------------------------

; Insert your custom code below this banner

;---------------------------------------------------

; NOTE: interrupt service routines must preserve

; the values of the A and X CPU registers. At this

; point A has been preserved on the stack and wil

; be restored later. However, if you use X, be

; sure to take care of it yourself!

;---------------------------------------------------

; Insert your custom code above this banner

;---------------------------------------------------

;@PSoC\_UserCode\_END@ (Do not change this line.)

pop A ; Restore A, not used any more

or [DUALADC\_1\_bfStatus],DUALADC\_1\_fDATA\_READY ; Set Data ready bit

tst [DUALADC\_1\_bSampC], ffh ; If sample\_counter == 0 -->> continuous data collection

jz DUALADC\_1\_END\_PWM16\_ISR

dec [DUALADC\_1\_bSampC] ; Dec sample counter and check for zero

jnz DUALADC\_1\_END\_PWM16\_ISR

and reg[DUALADC\_1\_fPWM\_LSB\_CR0], ~DUALADC\_1\_fDBLK\_ENABLE ; Disable the PWM

and reg[DUALADC\_1\_bCounter1\_CR0], ~DUALADC\_1\_fDBLK\_ENABLE ; Disable the Counter

and reg[DUALADC\_1\_bCounter2\_CR0], ~DUALADC\_1\_fDBLK\_ENABLE ; Disable the Counter

IF DUALADC\_1\_NoAZ

or reg[DUALADC\_1\_bfADC1cr2], DUALADC\_1\_fAutoZero ; Put the Integrator into Autozero mode

or reg[DUALADC\_1\_bfADC2cr2], DUALADC\_1\_fAutoZero ; Put the Integrator into Autozero mode

ENDIF

or reg[DUALADC\_1\_bfADC1cr3], DUALADC\_1\_fFSW0 ; Put Integrator into reset

or reg[DUALADC\_1\_bfADC2cr3], DUALADC\_1\_fFSW0 ; Put Integrator into reset

and reg[DUALADC\_1\_bfPWM16\_INT\_REG], ~DUALADC\_1\_bfPWM16\_Mask ; Disable interrupts

and reg[DUALADC\_1\_bfCounter1\_INT\_REG], ~DUALADC\_1\_bfCounter1\_Mask

and reg[DUALADC\_1\_bfCounter2\_INT\_REG], ~DUALADC\_1\_bfCounter2\_Mask

DUALADC\_1\_END\_PWM16\_ISR:

reti

DUALADC\_1\_APIINT\_END: