Design Assignment 5: Timer



Alexis Adie and Madison Mastroberte

ELC 411-01: Embedded Systems

Submitted:

November 16th 2017

Design Assignment 5: Timer

Alexis Adie and Madison Mastroberte

Department of Electrical and Computer Engineering

The College of New Jersey

2000 Pennington Road, Ewing, NJ 08618, USA

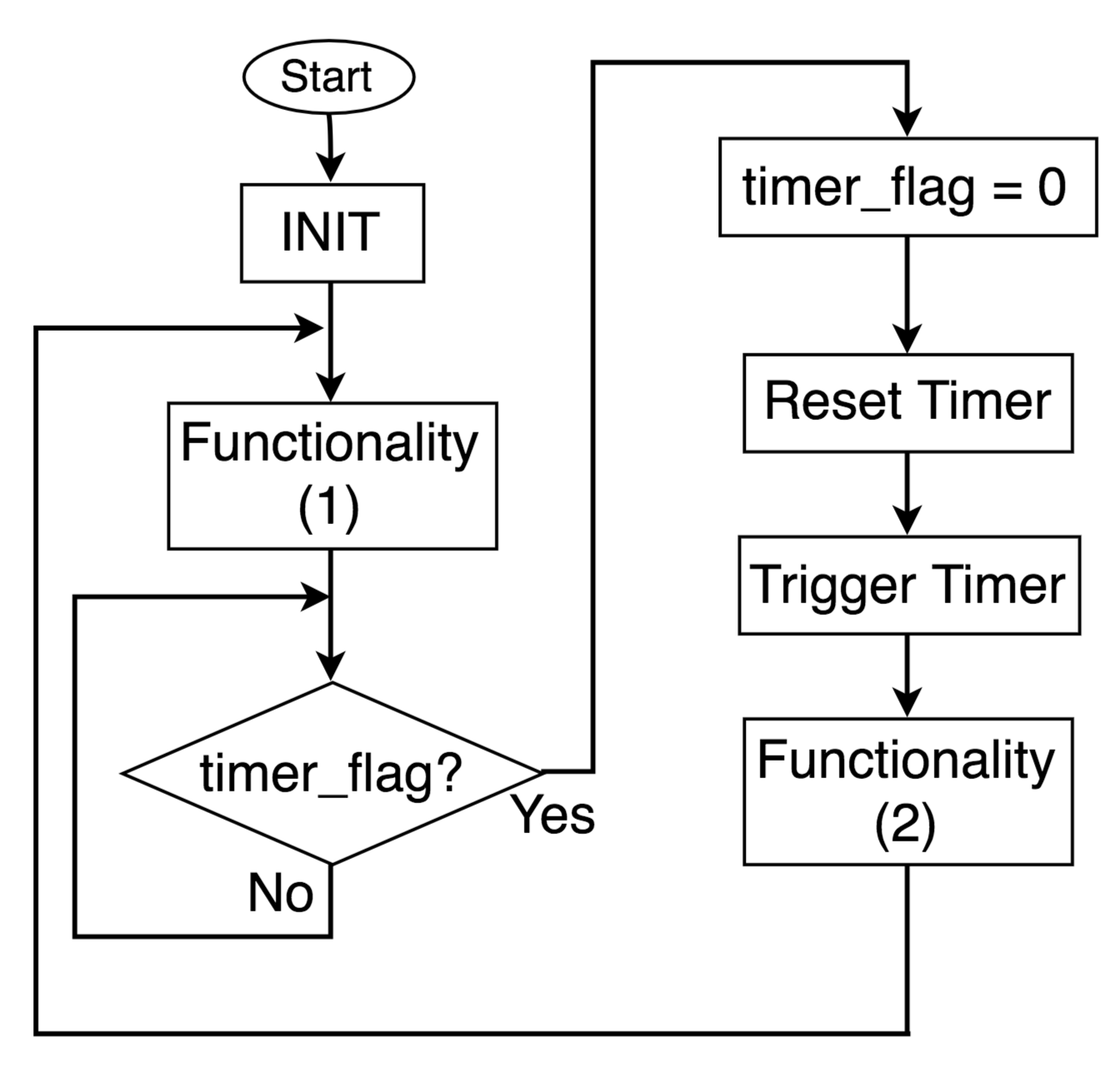
(adiea1, mastrom7) @tcnj.edu

# Introduction

The purpose of this experiment was to incorporate a Timer into the project in Design Assignment 4 rather than the CyDelay function. By utilizing this methodology, the process is more streamlined due to latencies in the CyDelay function. As a result, the final output contains more consistency and does not featured skipping of waveforms.

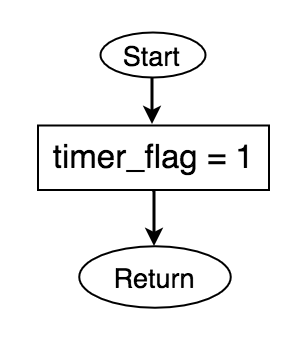
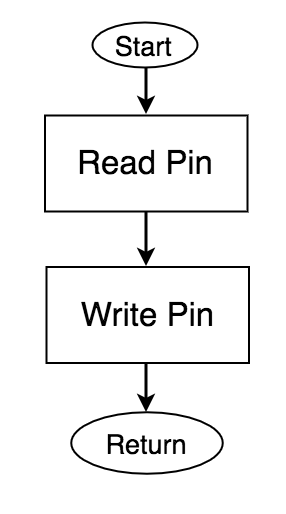
II. Results

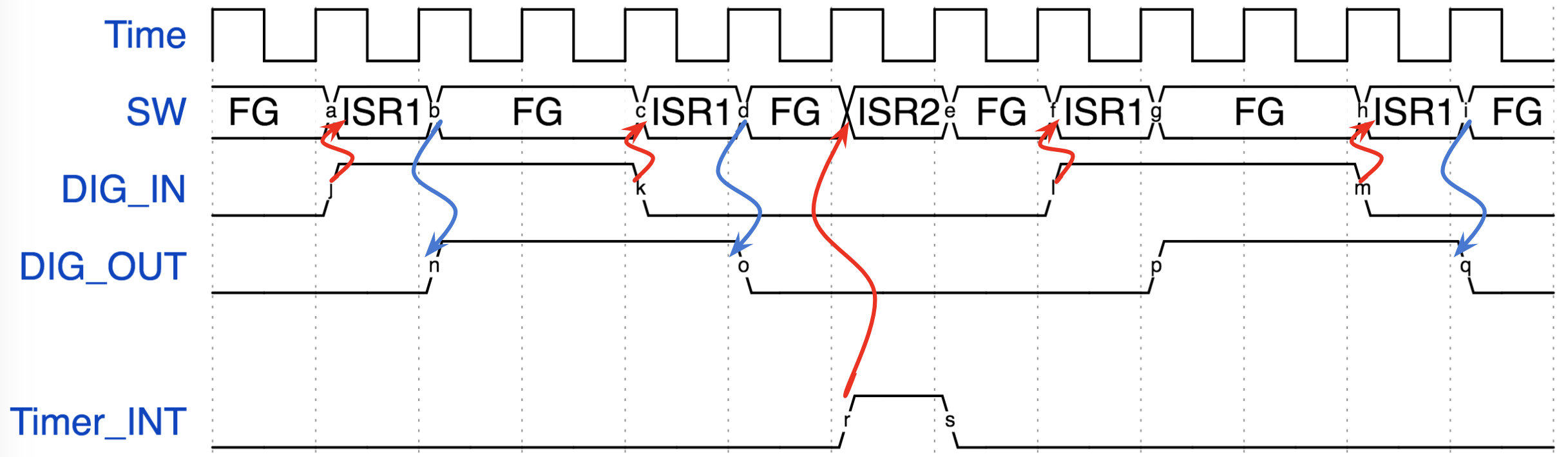
*A. Software Block Diagrams and Timing Diagram*

**Figure 1:** Flowchart of the functional thread of the project including the foreground thread, initialization and loop. 

Functionality 1 consists of starting the components and converting values. These values consists of converting integers and fixed point.

Functionality 2 consists of writing to the LCD display and reading the switch states to determine the delay. The display consists of a horizontal bar that animates the processes. For example, the bar should move at the same speed until a specific point in the frequency, which stops functionality. Specifically, functionality stopped at 150 kHz, which caused the bar to stop moving. Anything about this frequency cause the bar to no longer display to the LCD.

**Figure 2:** Flowcharts of each ISR thread. On left hand side is ISR1 and the right is ISR2.

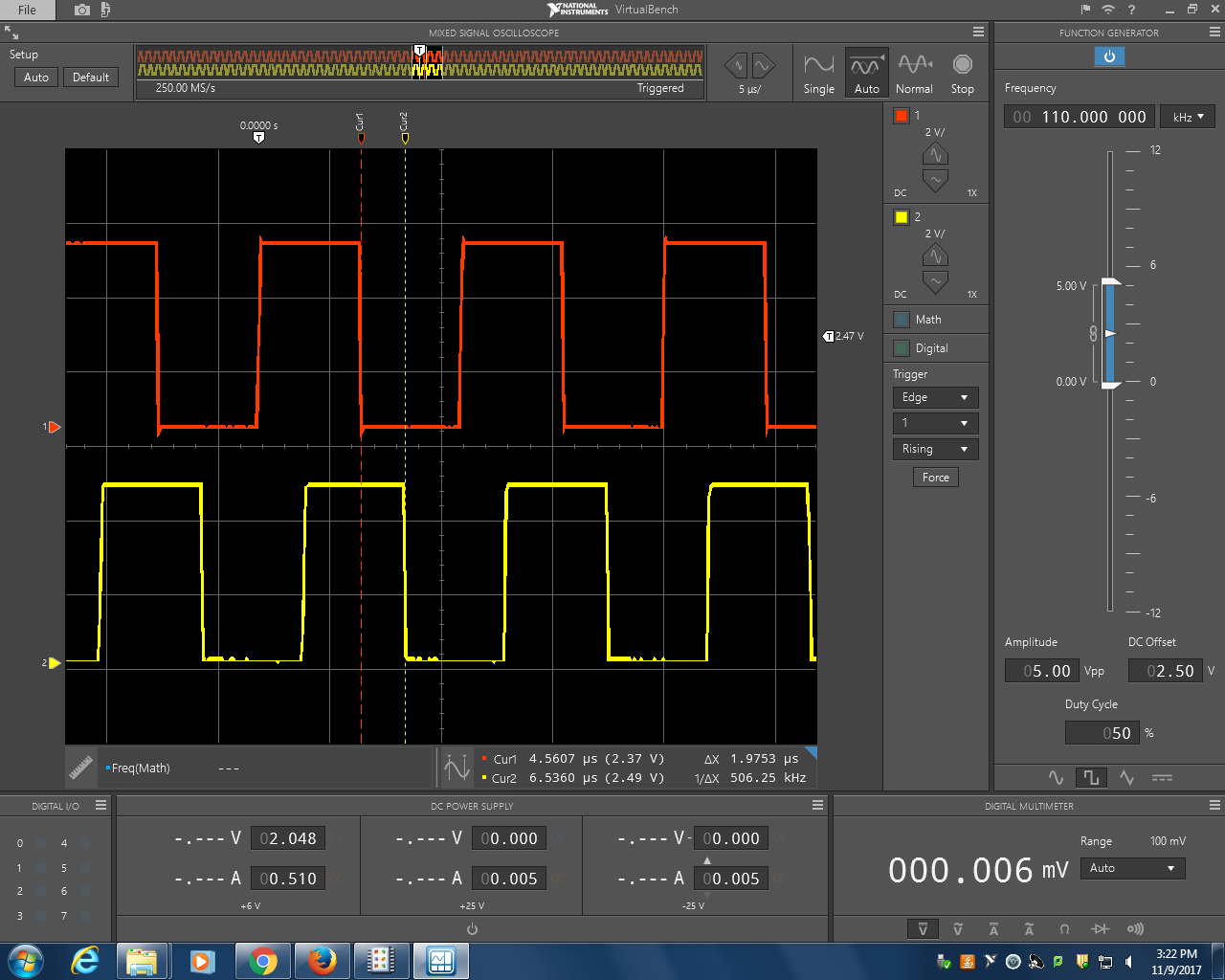


**Figure 3:** Timing Diagram with the switch and the timer with dependencies

*B. Measurements and Observations*

|  |  |
| --- | --- |
| Frequency (kHz) | Latency () |
| 100 | 1.8894 |
| 110 | 1.9753 |
| 130 | 2.1471 |
| 140 | 1.9753 |
| 150 | 2.0612 |
| 160 | 1.9270 |
| 180 | 1.8894 |

**Table 1:** Latency between input and output waveform for each frequency tested



**Figure 4:** Input (red) and output (yellow) square waveforms at 130 kHz [5 /div | 5V/div].

The red vertical line depicts the falling edge of the input signal while the yellow vertical line depicts the falling edge of the output signal. The latency was calculated using the difference between the two lines as seen in the bottom right of Figure 4.

The time delay caused by the Timer within this experiment can be calculated using the equation:

where t1 is the initial time captured from the FIFO and f is the input frequency. For solving purposes, the frequency assumed is 150 kHz. The timer period was set to 792000. Therefore the latency is:

5.28 ms

# IV. CONCLUSION

In this experiment, the team modified a past assignment in order to understand how a timer implementation affects latency. Since timers provide less latency, the final results were more reliable and consistent for analysis purposes. Therefore utilizing interrupts rather than CyDelay is a better process to decrease execution time of events.

# 

# 

# VI. Appendix

**Commented and Debugged “main.c” Code**

#include <project.h>

#include <stdio.h>

// Unsigned Fixed Point Macros, UQm.n

#define FIX\_n (16) // fixed point 'n' value

#define FIX\_m (16) // fixed point 'm' value

#define FIX\_N (FIX\_n + FIX\_m) // total bits in Qm.n

#define FIX\_FACTOR (1 << FIX\_n ) // fixed point fraction factor (2^n)

#define FIX\_\_0\_5 (1 << (FIX\_n-1)) // 0.5 expressed in UQ16.16

#define FIX\_\_1\_0 (1 << FIX\_n ) // 1.0 expressed in UQ16.16

extern uint32 volatile int\_count;

extern int volatile timer\_flag;

// Procedure:

// fix2int - round a fixed point value to the nearest integer

// Inputs:

// fix - fixed point value to round

// Return value:

// result of rounding the fixed point value to nearst integer, in uint32\_t container

uint32\_t fix2int( uint32\_t fix )

{

//Adds 0.5 in fixed point and discards the 16 fractional bits to zero

return (fix + FIX\_\_0\_5) >> FIX\_n;

}

// Procedure:

// fix2double - Convert a fixed point value to double precision

// Inputs:

// fix - value to convert

// Return value:

// double precision representation of fixed point value

double fix2double( uint32\_t fix )

{

//Adds 0.5 in fixed point and discards the 32 fractional bits to zero

fix = (fix + FIX\_\_0\_5) >> 16;

return fix;

}

// Procedure:

// double2fix - Convert a double precision value to fixed point, with rounding

// Inputs:

// x - value to convert

// Return value:

// fixed point approximation of the input value, in uint32\_t container

uint32\_t double2fix( double x )

{

return (uint32\_t) (x \* FIX\_FACTOR + 0.5);

}

// Procedure:

// fix2decimalstr - convert fixed point value to decimal string

// Inputs:

// x - fixed point value to convert to string

// str - pointer to destination string

// dotn - desired decimal precision

void fix2decimalstr( uint32\_t x, char \*str, int dotn )

{

int i;

uint64\_t lx;

int len;

// Use 64 bit int to avoid overflow

lx = x;

// Multiply by 10^dotn, to shift all fractional decimal into integer part

for (i = 0; i < dotn; ++i)

{

lx \*= 10;

}

// Get the integer part via rounding by adding half, and right shifting n

lx += FIX\_\_0\_5;

lx >>= FIX\_n;

x = (uint32\_t) lx;

// Print the number, but without decimal point

sprintf( str, "%d", (int) x );

len = strlen(str);

// Insert the decimal point in the correct location

// First move all of the last 'dotn' characters to the right to make space

str[len+1] = '\0';

for (i = 0; i < dotn; ++i)

{

str[len-i] = str[len-i-1];

}

str[len-dotn] = '.';

}

int main()

{

int k; // Current position of bouncing box (relative to LCD)

int direction; // +1 --> move right, -1 --> move left

char num\_str[17]; // Array to render the value of rate as a string

char msg\_str[17]; // Entire message, to write to LCD

int sw2; // Holds current switch state

int sw3; // Holds current switch state

int sw2\_prev; // Holds previous state, for button down detection

int sw3\_prev; // Holds previous state, for button down detection

CyGlobalIntEnable; /\* Enable global interrupts. \*/

uint32\_t delay = 20 \* FIX\_\_1\_0; // UQ16.16

uint32\_t llim = 20 \* FIX\_\_1\_0; // Upper limit of delay expressed in UQ16.16

uint32\_t ulim = 200 \* FIX\_\_1\_0; // Upper limit of delay expressed in UQ16.16

uint32\_t incr = double2fix( 10.0/3.0 ); // Represent 3.33... in fixed point

LCD\_Display\_Start(); // Start the LCD component

//starts the other components

isr\_1\_Start();

isr\_2\_Start();

Timer\_1\_Start();

timer\_flag=0;

//reset

Control\_Reg\_2\_Write(1);

//trigger timer

Control\_Reg\_1\_Write(1);

k = 0; // Initialize position

direction = 1; // and direction

sw2 = sw3 = sw2\_prev = sw3\_prev = 1; // Initialize switch states to open

// Loop forever

for(;;)

{

// Convert current delay to a string, with 3 decimal places precision

fix2decimalstr(delay, num\_str, 3);

// Generate composite message string

sprintf(msg\_str,"Cnt=%12lu",int\_count );

// Render current state onto the display

// Top line is bouncing square

// Bottom line is current delay

LCD\_Display\_ClearDisplay(); // Must clear entire display before new rendering

LCD\_Display\_DrawHorizontalBG(0, k, 1, 5); // Draw the box on top line

LCD\_Display\_Position(1, 0); // Position on bottom line

LCD\_Display\_PrintString(msg\_str); // Print the msg on bottom line

//sets timer flag back to zero once it is 1 and writes to each register

if (timer\_flag==1)

{

timer\_flag=0;

//reset

Control\_Reg\_2\_Write(1);

//trigger timer

Control\_Reg\_1\_Write(1);

}

k += direction; // Compute new position

if (k == 15)

direction = -1;

else if (k == 0)

direction = 1;

sw2 = SW2\_Read(); // Get current switch state

sw3 = SW3\_Read(); // Get current switch state

if (sw2 == 0 && sw2\_prev == 1) // If Switch 2 button down event, decrease delay

delay -= incr;

if (sw3 == 0 && sw3\_prev == 1) // If Switch 3 button down event, increase delay

delay += incr;

sw2\_prev = sw2; // Update previous sw2 state

sw3\_prev = sw3; // Update previous sw3 state

// Saturate delay to upper and lower limits

if (delay > ulim) delay = ulim;

if (delay < llim) delay = llim;

}

}

**Commented and Debugged “isr\_1.c” Code**

//include pins’ .h files so that isr\_1 may access them

#include <DIG\_IN\_PIN.h>

#include <DIG\_OUT\_PIN.h>

#include <cydevice\_trm.h>

#include <CyLib.h>

#include <isr\_1.h>

#include "cyapicallbacks.h"

#if !defined(isr\_1\_\_REMOVED) /\* Check for removal by optimization \*/

// start and definition codes

/\* `#START isr\_1\_intc` \*/

uint32 volatile int\_count =0;

/\* `#END` \*/

#ifndef CYINT\_IRQ\_BASE

#define CYINT\_IRQ\_BASE 16

#endif /\* CYINT\_IRQ\_BASE \*/

#ifndef CYINT\_VECT\_TABLE

#define CYINT\_VECT\_TABLE ((cyisraddress \*\*) CYREG\_NVIC\_VECT\_OFFSET)

#endif /\* CYINT\_VECT\_TABLE \*/

/\* Declared in startup, used to set unused interrupts to. \*/

CY\_ISR\_PROTO(IntDefaultHandler);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_Start (starts isr, unchanged from original

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_Start(void)

{

/\* For all we know the interrupt is active. \*/

isr\_1\_Disable();

/\* Set the ISR to point to the isr\_1 Interrupt. \*/

isr\_1\_SetVector(&isr\_1\_Interrupt);

/\* Set the priority. \*/

isr\_1\_SetPriority((uint8)isr\_1\_INTC\_PRIOR\_NUMBER);

/\* Enable it. \*/

isr\_1\_Enable();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_StartEx (sets up interrupt, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_StartEx(cyisraddress address)

{

/\* For all we know the interrupt is active. \*/

isr\_1\_Disable();

/\* Set the ISR to point to the isr\_1 Interrupt. \*/

isr\_1\_SetVector(address);

/\* Set the priority. \*/

isr\_1\_SetPriority((uint8)isr\_1\_INTC\_PRIOR\_NUMBER);

/\* Enable it. \*/

isr\_1\_Enable();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_Stop (stops and removes interrupt (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_Stop(void)

{

/\* Disable this interrupt. \*/

isr\_1\_Disable();

/\* Set the ISR to point to the passive one. \*/

isr\_1\_SetVector(&IntDefaultHandler);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_Interrupt (default function, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CY\_ISR(isr\_1\_Interrupt)

{

#ifdef isr\_1\_INTERRUPT\_INTERRUPT\_CALLBACK

isr\_1\_Interrupt\_InterruptCallback();

#endif /\* isr\_1\_INTERRUPT\_INTERRUPT\_CALLBACK \*/

//reading and writing the pins within the isr

/\* Place your Interrupt code here. \*/

/\* `#START isr\_1\_Interrupt` \*/

DIG\_IN\_PIN\_Read() ;

DIG\_OUT\_PIN\_Write(DIG\_IN\_PIN\_Read()) ;

++ int\_count;

DIG\_IN\_PIN\_ClearInterrupt();

/\* `#END` \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_SetVector (changes isr vector, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_SetVector(cyisraddress address)

{

cyisraddress \* ramVectorTable;

ramVectorTable = (cyisraddress \*) \*CYINT\_VECT\_TABLE;

ramVectorTable[CYINT\_IRQ\_BASE + (uint32)isr\_1\_\_INTC\_NUMBER] = address;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_GetVector (gets the address of the vector, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cyisraddress isr\_1\_GetVector(void)

{

cyisraddress \* ramVectorTable;

ramVectorTable = (cyisraddress \*) \*CYINT\_VECT\_TABLE;

return ramVectorTable[CYINT\_IRQ\_BASE + (uint32)isr\_1\_\_INTC\_NUMBER];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_SetPriority (sets interrupt priority, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_SetPriority(uint8 priority)

{

\*isr\_1\_INTC\_PRIOR = priority << 5;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_GetPriority (gets interrupt priority, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 isr\_1\_GetPriority(void)

{

uint8 priority;

priority = \*isr\_1\_INTC\_PRIOR >> 5;

return priority;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_Enable (enables the isr, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_Enable(void)

{

/\* Enable the general interrupt. \*/

\*isr\_1\_INTC\_SET\_EN = isr\_1\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_GetState (gets state of interrupt, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 isr\_1\_GetState(void)

{

/\* Get the state of the general interrupt. \*/

return ((\*isr\_1\_INTC\_SET\_EN & (uint32)isr\_1\_\_INTC\_MASK) != 0u) ? 1u:0u;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_Disable (disables interrupt, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_Disable(void)

{

/\* Disable the general interrupt. \*/

\*isr\_1\_INTC\_CLR\_EN = isr\_1\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_SetPending (puts interrupt into pending state (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_SetPending(void)

{

\*isr\_1\_INTC\_SET\_PD = isr\_1\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_1\_ClearPending (clears pending interrupt (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_1\_ClearPending(void)

{

\*isr\_1\_INTC\_CLR\_PD = isr\_1\_\_INTC\_MASK;

}

#endif /\* End check for removal by optimization \*/

/\* [] END OF FILE \*/

**Commented and Debugged “isr\_2.c” Code**

#include <cydevice\_trm.h>

#include <CyLib.h>

#include <isr\_2.h>

#include "cyapicallbacks.h"

int volatile timer\_flag;

#if !defined(isr\_2\_\_REMOVED) /\* Check for removal by optimization \*/

//include and defines code

/\* `#START isr\_2\_intc` \*/

/\* `#END` \*/

#ifndef CYINT\_IRQ\_BASE

#define CYINT\_IRQ\_BASE 16

#endif /\* CYINT\_IRQ\_BASE \*/

#ifndef CYINT\_VECT\_TABLE

#define CYINT\_VECT\_TABLE ((cyisraddress \*\*) CYREG\_NVIC\_VECT\_OFFSET)

#endif /\* CYINT\_VECT\_TABLE \*/

/\* Declared in startup, used to set unused interrupts to. \*/

CY\_ISR\_PROTO(IntDefaultHandler);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_Start (starts isr, unchanged from original

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_Start(void)

{

/\* For all we know the interrupt is active. \*/

isr\_2\_Disable();

/\* Set the ISR to point to the isr\_2 Interrupt. \*/

isr\_2\_SetVector(&isr\_2\_Interrupt);

/\* Set the priority. \*/

isr\_2\_SetPriority((uint8)isr\_2\_INTC\_PRIOR\_NUMBER);

/\* Enable it. \*/

isr\_2\_Enable();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_StartEx (sets up interrupt, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_StartEx(cyisraddress address)

{

/\* For all we know the interrupt is active. \*/

isr\_2\_Disable();

/\* Set the ISR to point to the isr\_2 Interrupt. \*/

isr\_2\_SetVector(address);

/\* Set the priority. \*/

isr\_2\_SetPriority((uint8)isr\_2\_INTC\_PRIOR\_NUMBER);

/\* Enable it. \*/

isr\_2\_Enable();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_Stop (stops and removes interrupt (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_Stop(void)

{

/\* Disable this interrupt. \*/

isr\_2\_Disable();

/\* Set the ISR to point to the passive one. \*/

isr\_2\_SetVector(&IntDefaultHandler);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_Interrupt (default function, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CY\_ISR(isr\_2\_Interrupt)

{

#ifdef isr\_2\_INTERRUPT\_INTERRUPT\_CALLBACK

isr\_2\_Interrupt\_InterruptCallback();

#endif /\* isr\_2\_INTERRUPT\_INTERRUPT\_CALLBACK \*/

//sets timer flag to 1

/\* Place your Interrupt code here. \*/

/\* `#START isr\_2\_Interrupt` \*/

timer\_flag=1;

/\* `#END` \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_SetVector (changes isr vector, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_SetVector(cyisraddress address)

{

cyisraddress \* ramVectorTable;

ramVectorTable = (cyisraddress \*) \*CYINT\_VECT\_TABLE;

ramVectorTable[CYINT\_IRQ\_BASE + (uint32)isr\_2\_\_INTC\_NUMBER] = address;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_GetVector (gets the address of the vector, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cyisraddress isr\_2\_GetVector(void)

{

cyisraddress \* ramVectorTable;

ramVectorTable = (cyisraddress \*) \*CYINT\_VECT\_TABLE;

return ramVectorTable[CYINT\_IRQ\_BASE + (uint32)isr\_2\_\_INTC\_NUMBER];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_SetPriority (sets interrupt priority, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_SetPriority(uint8 priority)

{

\*isr\_2\_INTC\_PRIOR = priority << 5;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_GetPriority (gets interrupt priority, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 isr\_2\_GetPriority(void)

{

uint8 priority;

priority = \*isr\_2\_INTC\_PRIOR >> 5;

return priority;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_Enable (enables the isr, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_Enable(void)

{

/\* Enable the general interrupt. \*/

\*isr\_2\_INTC\_SET\_EN = isr\_2\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_GetState (gets state of interrupt, code unchanged)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 isr\_2\_GetState(void)

{

/\* Get the state of the general interrupt. \*/

return ((\*isr\_2\_INTC\_SET\_EN & (uint32)isr\_2\_\_INTC\_MASK) != 0u) ? 1u:0u;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_Disable (disables interrupt, unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_Disable(void)

{

/\* Disable the general interrupt. \*/

\*isr\_2\_INTC\_CLR\_EN = isr\_2\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_SetPending (puts interrupt into pending state (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_SetPending(void)

{

\*isr\_2\_INTC\_SET\_PD = isr\_2\_\_INTC\_MASK;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name: isr\_2\_ClearPending (clears pending interrupt (unchanged code)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void isr\_2\_ClearPending(void)

{

\*isr\_2\_INTC\_CLR\_PD = isr\_2\_\_INTC\_MASK;

}

#endif /\* End check for removal by optimization \*/

/\* [] END OF FILE \*/