# 9 INTERRUPTS, TIMERS and TASKS – Code Snips

**Note: These Code Snips are taken straight from the book chapter; i.e. the “Program Examples”. In some cases therefore they are not complete programs.**

/\* Program Example 9.1: Simple interrupt example. External input causes interrupt, while led flashes

\*/

#include "mbed.h"

InterruptIn button(p5); //define and name the interrupt input

DigitalOut led(LED1);

DigitalOut flash(LED4);

void ISR1() { //this is the response to interrupt, i.e. the ISR

led = !led;

}

int main() {

button.rise(&ISR1); // attach the address of the ISR function to the

// interrupt rising edge

while(1) { // continuous loop, ready to be interrupted

flash = !flash;

wait(0.25);

}

}

Program Example 9.1: Introductory use of an interrupt

/\* Program Example 9.2: Tests interrupt latency. External input causes interrupt, which pulses external LED while LED4 flashes continuously.

\*/

#include "mbed.h"

InterruptIn squarewave(p5); //Connect input square wave here

DigitalOut led(p6);

DigitalOut flash(LED4);

void pulse() { //ISR sets external led high for fixed duration

led = 1;

wait(0.01);

led = 0;

}

int main() {

squarewave.rise(&pulse); // attach the address of the pulse function to

// the rising edge

while(1) { // interrupt will occur within this endless loop

flash = !flash;

wait(0.25);

}

}

**Program Example 9.2: Testing interrupt latency**

/\* Program Example 9.3: A simple Timer example, from mbed web site.

Activate host terminal to test.

\*/

#include "mbed.h"

Timer t; // define Timer with name “t”

Serial pc(USBTX, USBRX);

int main() {

t.start(); //start the timer

pc.printf("Hello World!\n");

t.stop(); //stop the timer

pc.printf("The time taken was %f seconds\n", t.read()); //print to pc

}

Program Example 9.3: A simple Timer application

/\*Program Example 9.4: Program which runs two time-based tasks

\*/

#include "mbed.h"

Timer timer\_fast; // define Timer with name "timer\_fast"

Timer timer\_slow; // define Timer with name "timer\_slow"

DigitalOut ledA(LED1);

DigitalOut ledB(LED4);

void task\_fast(void); //function prototypes

void task\_slow(void);

int main() {

timer\_fast.start(); //start the Timers

timer\_slow.start();

while (1){

if (timer\_fast.read()>0.2){ //test Timer value

task\_fast(); //call the task if trigger time is reached

timer\_fast.reset(); //and reset the Timer

}

if (timer\_slow.read()>1){ //test Timer value

task\_slow();

timer\_slow.reset();

}

}

}

void task\_fast(void){ //”Fast” Task

ledA = !ledA;

}

void task\_slow(void){ //”Slow” Task

ledB = !ledB;

}

Program Example 9.4: Running two timed tasks

/\* Program Example 9.5: Tests Timer duration, displaying current time values to terminal

\*/

#include "mbed.h"

Timer t;

float s=0; //seconds cumulative count

float m=0; //minutes cumulative count

DigitalOut diag (LED1);

Serial pc(USBTX, USBRX);

int main() {

pc.printf("\r\nTimer Duration Test\n\r");

pc.printf("-------------------\n\n\r");

t.reset(); //reset Timer

t.start(); // start Timer

while(1){

if (t.read()>=(s+1)){ //has Timer passed next whole second?

diag = 1; //If yes, flash LED and print a message

wait (0.05);

diag = 0;

s++ ;

//print the number of seconds exceeding whole minutes

pc.printf("%1.0f seconds\r\n",(s-60\*(m-1)));

}

if (t.read()>=60\*m){

printf("%1.0f minutes \n\r",m);

m++ ;

}

if (t.read()<s){ //test for overflow

pc.printf("\r\nTimer has overflowed!\n\r");

for(;;){} //lock into an endless loop doing nothing

}

} //end of while

}

Program Example 9.5: Testing Timer duration

/\*Program Example 9.6: Demonstrates Timeout, by triggering an event a fixed duration after a button press. \*/

#include "mbed.h"

Timeout Response; //create a Timeout, and name it "Response"

DigitalIn button (p5);

DigitalOut led1(LED1);

DigitalOut led2(LED2);

DigitalOut led3(LED3);

void blink() { //this function is called at the end of the Timeout

led2 = 1;

wait(0.5);

led2=0;

}

int main() {

while(1) {

if(button==1){

Response.attach(&blink,2.0); // attach blink function to Response Timeout,

//to occur after 2 seconds

led3=1; //shows button has been pressed

}

else {

led3=0;

}

led1=!led1;

wait(0.2);

}

}

Program Example 9.6: Simple Timeout application

/\*Program Example 9.7: Demonstrates the use of Timeout and interrupts, to allow response to an event-driven task while a time-driven task continues.

\*/

#include "mbed.h"

void blink\_end (void);

void blink (void);

void ISR1 (void);

DigitalOut led1(LED1);

DigitalOut led2(LED2);

DigitalOut led3(LED3);

Timeout Response; //create a Timeout, and name it Response

Timeout Response\_duration; //create a Timeout, and name it Response\_duration

InterruptIn button(p5); //create an interrupt input, named button

void blink() { //This function is called when Timeout is complete

led2=1;

// set the duration of the led blink, with another timeout, duration 0.1 s

Response\_duration.attach(&blink\_end, 1);

}

void blink\_end() { //A function called at the end of Timeout Response\_duration

led2=0;

}

void ISR1(){

led3=1; //shows button is pressed; diagnostic and not central to program

//attach blink1 function to Response Timeout, to occur after 2 seconds

Response.attach(&blink, 2.0);

}

int main() {

button.rise(&ISR1); //attach the address of ISR1 function to the rising edge

while(1) {

led3=0; //clear LED3

led1=!led1;

wait(0.2);

}

}

Program Example 9.7: Improved use of Timeout

/\*Program Example 9.8: Tests reaction time, and demos use of Timer and Timeout functions

\*/

#include "mbed.h"

#include <stdio.h>

#include <stdlib.h> //contains rand() function

void measure ();

Serial pc(USBTX, USBRX);

DigitalOut led1(LED1);

DigitalOut led4(LED4);

DigitalIn responseinput(p5); //the player hits the switch connected here to respond

Timer t; //used to measure the response time

Timeout action; //the Timeout used to initiate the response speed test

int main (){

pc.printf("Reaction Time Test\n\r");

pc.printf("------------------\n\r");

while (1) {

int r\_delay; //this will be the “random” delay before the led is blinked

pc.printf("New Test\n\r");

led4=1; //warn that test will start

wait(0.2);

led4=0;

r\_delay = rand() % 10 + 1; // generates a pseudorandom number range 1-10

pc.printf("random number is %i\n\r", r\_delay); // allows test randomness;

//removed for normal play

action.attach(&measure,r\_delay); // set up Timeout to call measure()

// after random time

wait(10); //test will start within this time, and we then return to it

}

}

void measure (){ // called when the led blinks, and measures response time

if (responseinput ==1){ //detect cheating!

pc.printf("Don't hold button down!");

}

else{

t.start(); //start the timer

led1=1; //blink the led

wait(0.05);

led1=0;

while (responseinput==0) {

//wait here for response

}

t.stop(); //stop the timer once response detected

pc.printf("Your reaction time was %f seconds\n\r", t.read());

t.reset();

}

}

Program Example 9.8: Reaction time test: applying Timer and Timeout

/\* Program Example 9.9: Simple demo of "Ticker". Replicates behaviour of first

led flashing program.

\*/

#include "mbed.h"

void led\_switch(void);

Ticker time\_up; //define a Ticker, with name “time\_up”

DigitalOut myled(LED1);

void led\_switch(){ //the function that Ticker will call

myled=!myled;

}

int main(){

time\_up.attach(&led\_switch, 0.2); //initialises the ticker

while(1){ //sit in a loop doing nothing, waiting for Ticker interrupt

}

}

Program Example 9.9: Applying Ticker to our very first program

/\*Program Example 9.10: Metronome. Uses Ticker to set beat rate

\*/

#include "mbed.h"

#include <stdio.h>

Serial pc(USBTX, USBRX);

DigitalIn up\_button(p5);

DigitalIn down\_button(p6);

DigitalOut redled(p19); //displays the metronome beat

Ticker beat\_rate; //define a Ticker, with name “beat\_rate”

void beat(void);

float period (0.5); //metronome period in seconds, inital value 0.5

int rate (120); //metronome rate, initial value 120

int main() {

pc.printf("\r\n");

pc.printf("mbed metronome!\r\n");

pc.printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\r\n");

period = 1;

redled = 1; //diagnostic

wait(.1);

redled = 0;

beat\_rate.attach(&beat, period); //initialises the beat rate

//main loop checks buttons, updates rates and displays

while(1){

if (up\_button ==1) //increase rate by 4

rate = rate + 4;

if (down\_button ==1) //decrease rate by 4

rate = rate - 4;

if (rate > 208) //limit the maximum beat rate to 208

rate = 208;

if (rate < 40) //limit the minimum beat rate to 40

rate = 40;

period = 60/rate; //calculate the beat period

pc.printf("metronome rate is %i\r", rate);

//pc.printf("metronome period is %f\r\n", period); //optional check

wait (0.5);

}

}

void beat() { //this is the metronome beat

beat\_rate.attach(&beat, period); //update beat rate at this moment

redled = 1;

wait(.1);

redled = 0;

}

Program Example 9.10: Metronome, applying Ticker

/\* Program Example 9.11: Toggles LED1 every time p18 goes high. Uses hardware build shown in Figure 9.3.

\*/

#include "mbed.h"

InterruptIn button(p5); // Interrupt on digital pushbutton input p18

DigitalOut led1(LED1); // mbed LED1

void toggle(void); // function prototype

int main() {

button.rise(&toggle); // attach the address of the toggle

} // function to the rising edge

void toggle() {

led1=!led1;

}

Program Example 9.11: Toggles LED1 every time mbed pin 5 goes high

/\* Program Example 9.12: Event driven LED switching with switch debounce

\*/

#include "mbed.h"

InterruptIn button(p18); // Interrupt on digital pushbutton input p18

DigitalOut led1(LED1); // digital out to LED1

Timer debounce; // define debounce timer

void toggle(void); // function prototype

int main() {

debounce.start();

button.rise(&toggle); // attach the address of the toggle

} // function to the rising edge

void toggle() {

if (debounce.read\_ms()>10) // only allow toggle if debounce timer

led1=!led1; // has passed 10 ms

debounce.reset(); // restart timer when the toggle is performed

}

Program Example 9.12: Event driven LED switching with switch debounce