Peer Review #3

# This review is for Lab 6: Pulse-width modulation using the programmable timer in C. The goal for this lab is to use what we have previously learned about timer interrupts and apply it to make a pulse width change depending on the value pressed by a keypad. This applies knowledge of normal GPIO interrupts, the using of a keypad, and an interrupt timer, all mixed into one lab to create this function. Used in this lab includes: A breadboard (with studio), a TM32 NUCLEO-L432KC Board, wires, and a .c file. This lab is very similar to last weeks Lab 5, where we first learned to use the timer to count in different increments, however, this time we use that timer not to count, but to instead change the time variation between pulses that can be seen on an oscilloscope. Duty cycle = T1/T. We then use the scope to view and verify our output for each keypad input. We can also view the keypad value pressed on the DIO pins in Static IO of Waveforms. With these two things we can verify the results of our code.

**The program written obtains the following steps:**

1. Establish variables used, pins set up, and interrupts setup (including normal interrupts and timer)
2. Enter an infinite loop where we do nothing
3. If an interrupt is pressed, we find the value pressed and change the duty cycle output to the corresponding keypad value pressed.
4. Then go back to the loop and wait.

My code for this worked very well and did not have too many problems. The connections I made matched the pinout that I drew for Lab 5, and added the PA0 output pin for testing the output duty cycle on the scope. The code for this lab is shown on the next page.

After setting up the lab and testing it out I had a couple small problems where I had forgot to change TIM7 to TIM2 and this caused a lot of errors in the code. After going through and changing all those, I could not get the ARR to upload right, because I had OR’d the wrong bits into the mode register. However, after getting these figured out the rest of the lab went simple. I learned to also view the input of the keypad after the AND gate to view if the keypad is sending the signal low or not. The scope was easily set up by connecting to any ground on the board, and then probing the output pin set PA0. Screenshots for the Duty cycle of 30% and 50% are shown in the Figures below.

A picture containing graphical user interface

Description automatically generated

Figure 1

A picture containing timeline

Description automatically generated

Figure 2

1 ÏÏÏ/\*====================================================\*/  
 2 ÏÏÏ  
 3 ÏÏÏ/\* ELEC 3040 - Lab 6  
 4 ÏÏÏ  
 5 ÏÏÏ/\*====================================================\*/  
 6 ÏÏÏ  
 7 ÏÏÏ  
 8 ÏÏÏ  
 9 ÏÏÏ#include "stm32l4xx.h" /\* microcontroller information \*/  
 10 ÏÏÏ  
 11 ÏÏÏ  
 12 ÏÏÏ  
 13 ÏÏÏ/\* Define global variables \*/  
 14 ÏÏÏ  
 15 ÏíÏstatic unsigned int counter; //value of count (0-9)  
 16 ÏÏÏ  
 17 ÏíÏstatic unsigned int button; //value of button press  
 18 ÏÏÏ  
 19 ÏíÏunsigned int col; //what column has been pressed  
 20 ÏÏÏ  
 21 ÏíÏunsigned int row; //what row has been pressed  
 22 ÏÏÏ  
 23 ÏíÏstatic unsigned int colNum; //what column # has been pressed  
 24 ÏÏÏ  
 25 ÏíÏstatic unsigned int rowNum; //what row # has been pressed  
 26 ÏÏÏ  
 27 ÏíÏunsigned int ccrNum; //value for duty cycle  
 28 ÏÏÏ  
 29 ÏíÏstatic unsigned int go; //handler variable  
 30 ÏÏÏ  
 31 ÏíÏunsigned int i,j,n,k; //delay variables  
 32 ÏÏÏ  
 33 ÏíÏstatic unsigned int keypad\_map [4][4] = { //keypad matrix, no press = 0xFF  
 34 ÏÏÏ  
 35 ÏÏÏ {0x01,0x02,0x03,0x0A}, //0,0;1st row  
 36 ÏÏÏ  
 37 ÏÏÏ {0x04,0x05,0x06,0x0B}, //1,0;2nd row  
 38 ÏÏÏ  
 39 ÏÏÏ {0x07,0x08,0x09,0x0C}, //2,0;3rd row  
 40 ÏÏÏ  
 41 ÏÏÏ {0x0E,0x00,0x0F,0x0D} //3,0;4th row  
 42 ÏÏÏ  
 43 ÏÏÏ};//0,0; 0,1; 0,2; 0,3;  
 44 ÏÏÏ  
 45 ÏíÏstatic unsigned int ccr\_value [11] = { //CCRy values according to button press  
 46 ÏÏÏ  
 47 ÏÏÏ 0, 399, 799, 1199, 1599, 1999,  
 48 ÏÏÏ  
 49 ÏÏÏ 2399, 2799, 3199, 3599, 4001  
 50 ÏÏÏ  
 51 ÏÏÏ};  
 52 ÏÏÏ  
 53 ÏÏÏ  
 54 ÏÏÏ  
 55 ÏÏÏ/\*---------------------------------------------------\*/  
 56 ÏÏÏ  
 57 ÏÏÏ/\* initialize clocks used in the program \*/  
 58 ÏÏÏ  
 59 ÏÏÏ/\* initialize GPIOB pins used in the program \*/  
 60 ÏÏÏ  
 61 ÏÏÏ/\* PB[0] = interrupt trigger, output of AND gate (row signals) \*/  
 62 ÏÏÏ  
 63 ÏÏÏ/\* PB[6:3] = displayed value, counter or button \*/  
 64 ÏÏÏ  
 65 ÏÏÏ/\*---------------------------------------------------\*/  
 66 ÏÏÏ  
 67 ÏÞßàstatic void Setup() {  
 68 ÏÏ§  
 69 ÏÏ§  
 70 ÏÏ§  
 71 ÏÏ§ /\* enable clocks \*/  
 72 ÏÏ§  
 73 ÏÏ¨¹¹ÏRCC->AHB2ENR |= 0x03; //enable GPIOA clock (bit 0) and GPIOB clock (bit 1)  
 74 ÏÏ§  
 75 ÏÏ§  
 76 ÏÏ§  
 77 ÏÏ§ /\* configure GPIO pins \*/  
 78 ÏÏ§  
 79 ÏÏ¨¹¹ÏGPIOA->MODER &= (0xFFFFFFF0); //PA0 = 00 and PA1 = 00  
 80 ÏÏ§  
 81 ÏÏ¨¹¹ÏGPIOA->MODER |= (0x00000006); //PA0 = 10 and PA1 = 01  
 82 ÏÏ§  
 83 ÏÏ§  
 84 ÏÏ§  
 85 ÏÏ¨¹¹ÏGPIOB->MODER &= (0xFFFFC03C); //inputs// display and AND, PB[6:3,0] = 00  
 86 ÏÏ§  
 87 ÏÏ¨¹¹ÏGPIOB->MODER |= (0x00001540); //outputs// display, PB[6:3] = 01  
 88 ÏÏ§  
 89 ÏÏ§  
 90 ÏÏ§  
 91 ÏÏ©}  
 92 ÏÏÏ  
 93 ÏÏÏ  
 94 ÏÏÏ  
 95 ÏÏÏ/\*---------------------------------------------------\*/  
 96 ÏÏÏ  
 97 ÏÏÏ/\* initialize GPIO pins used in the program \*/  
 98 ÏÏÏ  
 99 ÏÏÏ/\*---------------------------------------------------\*/  
100 ÏÏÏ  
101 ÏÞßàstatic void PinSetup1() {  
102 ÏÏ§  
103 ÏÏ§  
104 ÏÏ§  
105 ÏÏ¨¹¹ÏSetup();  
106 ÏÏ§  
107 ÏÏ§ /\* configure GPIOA pins \*/  
108 ÏÏ§  
109 ÏÏ¨¹¹ÏGPIOA->MODER &= (0xFF00F00F); //inputs// column and row, PA[11:8,5:2] = 00  
110 ÏÏ§  
111 ÏÏ¨¹¹ÏGPIOA->MODER |= (0x00550000); //outputs// column, PA[11:8] = 01  
112 ÏÏ§  
113 ÏÏ§  
114 ÏÏ§  
115 ÏÏ§ /\* configure push-pull pins \*/  
116 ÏÏ§  
117 ÏÏ¨¹¹ÏGPIOA->PUPDR &= (0xFFFFF00F); //pull-reset// row, PA[5:2] = 00  
118 ÏÏ§  
119 ÏÏ¨¹¹ÏGPIOA->PUPDR |= (0x00000550); //pull-up// row, PA[5:2] = 01  
120 ÏÏ§  
121 ÏÏ§  
122 ÏÏ§  
123 ÏÏ§  
124 ÏÏ§  
125 ÏÏ©}  
126 ÏÏÏ  
127 ÏÏÏ  
128 ÏÏÏ  
129 ÏÏÏ/\*---------------------------------------------------\*/  
130 ÏÏÏ  
131 ÏÏÏ/\* initialize GPIO pins used in the program \*/  
132 ÏÏÏ  
133 ÏÏÏ/\*---------------------------------------------------\*/  
134 ÏÏÏ  
135 ÏÞßàstatic void PinSetup2() {  
136 ÏÏ§  
137 ÏÏ§  
138 ÏÏ§  
139 ÏÏ¨¹¹ÏSetup();  
140 ÏÏ§  
141 ÏÏ§ /\* configure GPIOA pins \*/  
142 ÏÏ§  
143 ÏÏ¨¹¹ÏGPIOA->MODER &= (0xFF00F00F); //inputs// column and row, PA[11:8,5:2] = 00  
144 ÏÏ§  
145 ÏÏ¨¹¹ÏGPIOA->MODER |= (0x0550); //outputs// row, PA[5:2] = 01  
146 ÏÏ§  
147 ÏÏ§  
148 ÏÏ§  
149 ÏÏ§ /\* configure push-pull pins \*/  
150 ÏÏ§  
151 ÏÏ¨¹¹ÏGPIOA->PUPDR &= (0xFF00FFFF); //pull-reset// row, PA[11:8] = 00  
152 ÏÏ§  
153 ÏÏ¨¹¹ÏGPIOA->PUPDR |= (0x00550000); //pull-up// row, PA[11:8] = 01  
154 ÏÏ§  
155 ÏÏ§  
156 ÏÏ§  
157 ÏÏ©}  
158 ÏÏÏ  
159 ÏÏÏ  
160 ÏÏÏ  
161 ÏÏÏ/\*---------------------------------------------------\*/  
162 ÏÏÏ  
163 ÏÏÏ/\* enable PWM used in the program\*/  
164 ÏÏÏ  
165 ÏÏÏ/\*---------------------------------------------------\*/  
166 ÏÏÏ  
167 ÏÞßàstatic void PulseSetup() {  
168 ÏÏ§  
169 ÏÏ§  
170 ÏÏ§  
171 ÏÏ§ /\* enable clock \*/  
172 ÏÏ§  
173 ÏÏ¨¹¹ÏRCC->AHB2ENR |= 0x01; // Enable GPIOA clock (bit 0)  
174 ÏÏ§  
175 ÏÏ§  
176 ÏÏ§  
177 ÏÏ§ /\* configure pins \*/  
178 ÏÏ§  
179 ÏÏ¨¹¹ÏGPIOA->MODER &= 0xFFFFFFFC; // PA0 = 00, clear  
180 ÏÏ§  
181 ÏÏ¨¹¹ÏGPIOA->MODER |= 0x0002; // PA0 = 01, alternative function mode  
182 ÏÏ§  
183 ÏÏ§  
184 ÏÏ§  
185 ÏÏ§ /\* select desired AF (timer) \*/  
186 ÏÏ§  
187 ÏÏ¨¹¹ÏGPIOA->AFR[0] &= (0xFFFFFFF0); //mask bit[3:0]=00  
188 ÏÏ§  
189 ÏÏ¨¹¹ÏGPIOA->AFR[0] |= (0x00000001); //(0x0002);//configure bit[3:0]=0010, AF1 selected  
190 ÏÏ§  
191 ÏÏ§  
192 ÏÏ§  
193 ÏÏ§ /\* configure timer \*/  
194 ÏÏ§  
195 ÏÏ¨¹¹ÏRCC->APB1ENR1 |= RCC\_APB1ENR1\_TIM2EN; //(0x01);//enable timer module  
196 ÏÏ§  
197 ÏÏ¨¹¹ÏTIM2->CR1 |= 0x01; //enable timer counter  
198 ÏÏ§  
199 ÏÏ¨¹¹ÏTIM2->CCMR1 &= (0xFFFFFF8C); //mask channel one (bit[6:4]=00), output mode  
200 ÏÏ§  
201 ÏÏ¨¹¹ÏTIM2->CCMR1 |= (0x00000060); //configure output mode for PWM mode 1  
202 ÏÏ§  
203 ÏÏ¨¹¹ÏTIM2->CCER &= (0xFFFC); //bit[1:0]=00, clear timer channel 1 output  
204 ÏÏ§  
205 ÏÏ¨¹¹ÏTIM2->CCER |= (0x0001); //bit[1:0]=01, enable timer channel 1 output (active high)  
206 ÏÏ§  
207 ÏÏ§  
208 ÏÏ§  
209 ÏÏ§ /\* configure pulse \*/  
210 ÏÏ§  
211 ÏÏ¨¹¹ÏTIM2->PSC = 0;  
212 ÏÏ§  
213 ÏÏ¨¹¹ÏTIM2->ARR = 4000;  
214 ÏÏ§  
215 ÏÏ¨¹¹ÏTIM2->CCR1 = 0;  
216 ÏÏ§  
217 ÏÏ¨¹¹ÏNVIC\_EnableIRQ(EXTI0\_IRQn); /\* Enable IRQ \*/  
218 ÏÏ§  
219 ÏÏ©}  
220 ÏÏÏ  
221 ÏÏÏ  
222 ÏÏÏ  
223 ÏÏÏ/\*---------------------------------------------------\*/  
224 ÏÏÏ  
225 ÏÏÏ/\* initialize interrupts used in the program \*/  
226 ÏÏÏ  
227 ÏÏÏ/\* EXTI1 = external interrupt one \*/  
228 ÏÏÏ  
229 ÏÏÏ/\* EXTI2 = external interrupt two \*/  
230 ÏÏÏ  
231 ÏÏÏ/\*---------------------------------------------------\*/  
232 ÏÏÏ  
233 ÏÞßàstatic void InterruptSetup() { //maybe void in ()  
234 ÏÏ§  
235 ÏÏ§  
236 ÏÏ§  
237 ÏÏ§ /\* enable clocks \*/  
238 ÏÏ§  
239 ÏÏ¨¹¹ÏRCC->APB2ENR |= 0x01; //enable interrupt clock SYSCFG  
240 ÏÏ§  
241 ÏÏ§  
242 ÏÏ§  
243 ÏÏ§ /\* configure port PA0 as input source of EXTI0 \*/  
244 ÏÏ§  
245 ÏÏ¨¹¹ÏSYSCFG->EXTICR[0] &= 0xFFF0; //clear EXTI1 bit in config reg ~(0xF)  
246 ÏÏ§  
247 ÏÏ¨¹¹ÏSYSCFG->EXTICR[0] |= 0x0001; //PB configuration in EXTI0  
248 ÏÏ§  
249 ÏÏ§  
250 ÏÏ§  
251 ÏÏ§  
252 ÏÏ§  
253 ÏÏ§ /\* configure and enable EXTI0 as falling-edge triggered \*/  
254 ÏÏ§  
255 ÏÏ¨¹¹ÏEXTI->FTSR1 |= 0x0001; //falling edge trigger enabled  
256 ÏÏ§  
257 ÏÏ¨¹¹ÏEXTI->IMR1 |= 0x0001; //enable (unmask) EXTI0  
258 ÏÏ§  
259 ÏÏ¨¹¹ÏEXTI->PR1 |= 0x0001; //clear EXTI0 pending bit for line 1  
260 ÏÏ§  
261 ÏÏ¨¹¹ÏNVIC\_ClearPendingIRQ(EXTI0\_IRQn); ////////\* Clear NVIC pending bit \*/  
262 ÏÏ§  
263 ÏÏ¨¹¹ÏNVIC\_EnableIRQ(EXTI0\_IRQn); //enable IRQ with EXTI line 0 interrupt  
264 ÏÏ§  
265 ÏÏ§  
266 ÏÏ§  
267 ÏÏ§  
268 ÏÏ§  
269 ÏÏ©}  
270 ÏÏÏ  
271 ÏÏÏ/\*----------------------------------------------------------\*/  
272 ÏÏÏ  
273 ÏÏÏ/\* debounce delay function - do nothing for about 0.001 second \*/  
274 ÏÏÏ  
275 ÏÏÏ/\*----------------------------------------------------------\*/  
276 ÏÏÏ  
277 ÏÞßàstatic void debounce() { //  
278 ÏÏ§  
279 ÏÏ¨¹¹±for (i=0; i<15; i++) { //outer loop  
280 ÏÏ§ÏÏ5  
281 ÏÏ§ÏÏ7¹¹±for (j=0; j<40; j++) { //inner loop  
282 ÏÏ§ÏÏ5ÏÏ5  
283 ÏÏ§ÏÏ5ÏÏ7¹¹Ïn = j; //dummy operation for single-step test  
284 ÏÏ§ÏÏ5ÏÏ5  
285 ÏÏ§ÏÏ5ÏÏ°} //do nothing  
286 ÏÏ§ÏÏ5  
287 ÏÏ§ÏÏ°}  
288 ÏÏ§  
289 ÏÏ©}  
290 ÏÏÏ  
291 ÏÏÏ  
292 ÏÏÏ  
293 ÏÏÏ/\*----------------------------------------------------------\*/  
294 ÏÏÏ  
295 ÏÏÏ/\* delay function - do nothing for about 1 second \*/  
296 ÏÏÏ  
297 ÏÏÏ/\*----------------------------------------------------------\*/  
298 ÏÏÏ  
299 ÏÞßàstatic void delay() { //  
300 ÏÏ§  
301 ÏÏ¨¹¹±for (i=0; i<15; i++) { //outer loop  
302 ÏÏ§ÏÏ5  
303 ÏÏ§ÏÏ7¹¹±for (j=0; j<10000; j++) { //inner loop \*4000\*  
304 ÏÏ§ÏÏ5ÏÏ5  
305 ÏÏ§ÏÏ5ÏÏ7¹¹Ïn = j; //dummy operation for single-step test  
306 ÏÏ§ÏÏ5ÏÏ5  
307 ÏÏ§ÏÏ5ÏÏ°} //do nothing  
308 ÏÏ§ÏÏ5  
309 ÏÏ§ÏÏ°}  
310 ÏÏ§  
311 ÏÏ©}  
312 ÏÏÏ  
313 ÏÏÏ  
314 ÏÏÏ  
315 ÏÏÏ/\*----------------------------------------------------------\*/  
316 ÏÏÏ  
317 ÏÏÏ/\* keypad function - find which button has been pressed \*/  
318 ÏÏÏ  
319 ÏÏÏ/\*----------------------------------------------------------\*/  
320 ÏÏÏ  
321 ÏÞßàstatic void keypad() {  
322 ÏÏ§  
323 ÏÏ§  
324 ÏÏ§  
325 ÏÏ¨¹¹Ïbutton = 0;  
326 ÏÏ§  
327 ÏÏ¨¹¹ÏrowNum = 0;  
328 ÏÏ§  
329 ÏÏ¨¹¹ÏcolNum = 0;  
330 ÏÏ§  
331 ÏÏ¨¹¹ÏccrNum = 0;  
332 ÏÏ§  
333 ÏÏ¨¹¹Ïcol = 0; //initialize col  
334 ÏÏ§  
335 ÏÏ¨¹¹Ïrow = 0; //initialize row  
336 ÏÏ§  
337 ÏÏ§  
338 ÏÏ§  
339 ÏÏ§ /\* clear unwanted values \*/  
340 ÏÏ§  
341 ÏÏ¨¹¹ÏGPIOB->ODR &= 0xFF87; //mask PB[6:3] to 0  
342 ÏÏ§  
343 ÏÏ§ /\* columns output 0 and find which row is 0\*/  
344 ÏÏ§  
345 ÏÏ§ //PinSetup1();  
346 ÏÏ§  
347 ÏÏ¨¹¹Ïrow=0;  
348 ÏÏ§  
349 ÏÏ¨¹¹ÏGPIOA->ODR &= (0xF0FF); //set column to output 0, PA[11:8] = 0  
350 ÏÏ§  
351 ÏÏ¨¹¹#for(k=0; k<4; k++); //delay for values to load  
352 ÏÏ§  
353 ÏÏ¨¹¹Ïrow = (~GPIOA->IDR & 0x003C); //get row inputs, PA[5:2] = 0\*\*\*check~  
354 ÏÏ§  
355 ÏÏ¨¹¹Ïrow = row >> 2; //shift right by 2  
356 ÏÏ§  
357 ÏÏ¨¹¹®do {  
358 ÏÏ§ÏÏ5  
359 ÏÏ§ÏÏ7¹¹Ïrow = row << 1; //shift left by 1 to find row count  
360 ÏÏ§ÏÏ5  
361 ÏÏ§ÏÏ7¹¹ÏrowNum++; //add to row count  
362 ÏÏ§ÏÏ5  
363 ÏÏ§ÏÏò} while(row < 0x10) ; //can only shift four times  
364 ÏÏ§  
365 ÏÏ§  
366 ÏÏ§  
367 ÏÏ§ /\* rows output 0 and find which column is 0 \*/  
368 ÏÏ§  
369 ÏÏ¨¹¹ÏPinSetup2();  
370 ÏÏ§  
371 ÏÏ¨¹¹Ïdebounce();  
372 ÏÏ§  
373 ÏÏ¨¹¹Ïcol=0;  
374 ÏÏ§  
375 ÏÏ¨¹¹ÏGPIOA->ODR &= (0xFFC3); //set row to output 0, PA[5:2] = 0  
376 ÏÏ§  
377 ÏÏ¨¹¹#for(k=0; k<4; k++); //delay for values to load  
378 ÏÏ§  
379 ÏÏ¨¹¹Ïcol = (~GPIOA->IDR & 0xF00); //get column inputs, PA[11:8] = 0  
380 ÏÏ§  
381 ÏÏ¨¹¹Ïcol = col >> 8; //shift right by 8  
382 ÏÏ§  
383 ÏÏ¨¹¹®do {  
384 ÏÏ§ÏÏ5  
385 ÏÏ§ÏÏ7¹¹Ïcol = col << 1; //shift left by 1 to find column count  
386 ÏÏ§ÏÏ5  
387 ÏÏ§ÏÏ7¹¹ÏcolNum++; //add to column count  
388 ÏÏ§ÏÏ5  
389 ÏÏ§ÏÏò} while(col < 0x10) ; //can only shift four times  
390 ÏÏ§  
391 ÏÏ§  
392 ÏÏ§  
393 ÏÏ¨¹¹Ïbutton = keypad\_map[--rowNum][--colNum]; //test and see if works\*\*\*\*  
394 ÏÏ§  
395 ÏÏ§  
396 ÏÏ§  
397 ÏÏ¨¹¹ÏccrNum = ccr\_value[button];  
398 ÏÏ§  
399 ÏÏ¨¹¹ÏTIM2->CCR1 = ccrNum;  
400 ÏÏ§  
401 ÏÏ§  
402 ÏÏ§  
403 ÏÏ¨¹¹Ïbutton = button << 3;  
404 ÏÏ§  
405 ÏÏ¨¹¹ÏGPIOB->ODR &= 0xFF87; //mask PB[6:3] to 0  
406 ÏÏ§  
407 ÏÏ¨¹¹ÏGPIOB->ODR |= button; //output button value, PB[6:3]  
408 ÏÏ§  
409 ÏÏ§  
410 ÏÏ§  
411 ÏÏ§  
412 ÏÏ§  
413 ÏÏ¨¹¹ÏccrNum = 0;  
414 ÏÏ§  
415 ÏÏ¨¹¹Ïbutton = 0;  
416 ÏÏ§  
417 ÏÏ¨¹¹Ïdelay (); //1 sec delay  
418 ÏÏ§  
419 ÏÏ¨¹¹Ïdelay (); //1 sec delay  
420 ÏÏ§  
421 ÏÏ©}  
422 ÏÏÏ  
423 ÏÏÏ  
424 ÏÏÏ  
425 ÏÏÏ/\*----------------------------------------------------------\*/  
426 ÏÏÏ  
427 ÏÏÏ/\* interrupt handler EXTI0 - keypad has been pressed \*/  
428 ÏÏÏ  
429 ÏÏÏ/\*----------------------------------------------------------\*/  
430 ÏÏÏ  
431 ÏÞßàvoid EXTI0\_IRQHandler() { //maybe put void in ()  
432 ÏÏ§  
433 ÏÏ§  
434 ÏÏ§  
435 ÏÏ§  
436 ÏÏ§  
437 ÏÏ¨¹¹Ïdebounce();  
438 ÏÏ§  
439 ÏÏ¨¹¹Ïgo=~go;  
440 ÏÏ§  
441 ÏÏ¨¹¹Ïkeypad(); //keypad logic  
442 ÏÏ§  
443 ÏÏ¨¹¹ÏPinSetup1();  
444 ÏÏ§  
445 ÏÏ¨¹¹Ïdebounce();  
446 ÏÏ§  
447 ÏÏ§  
448 ÏÏ§  
449 ÏÏ¨¹¹ÏEXTI->PR1 |= 0x0001; //clear EXTI0 pending bit\*  
450 ÏÏ§  
451 ÏÏ¨¹¹ÏNVIC\_ClearPendingIRQ(EXTI0\_IRQn); //clear NVIC pending bit with EXTI line 1 interrupt  
452 ÏÏ§  
453 ÏÏ¨¹¹Ï\_\_enable\_irq(); //enable interupts\* Maybe this has to go before above line  
454 ÏÏ§  
455 ÏÏ©}  
456 ÏÏÏ  
457 ÏÏÏ  
458 ÏÏÏ  
459 ÏÏÏ/\*------------------------------------------------\*/  
460 ÏÏÏ  
461 ÏÏÏ/\* main program \*/  
462 ÏÏÏ  
463 ÏÏÏ/\*------------------------------------------------\*/  
464 ÏÏÏ  
465 ÏÞßàint main(void) {  
466 ÏÏ§  
467 ÏÏ§  
468 ÏÏ§  
469 ÏÏ¨¹¹ÏSetup(); //configure clocks and GPIOB pins  
470 ÏÏ§  
471 ÏÏ¨¹¹ÏPinSetup1();  
472 ÏÏ§  
473 ÏÏ¨¹¹ÏInterruptSetup(); //configure interrupts  
474 ÏÏ§  
475 ÏÏ¨¹¹ÏPulseSetup();  
476 ÏÏ§  
477 ÏÏ¨¹¹Ïgo = 1; //initialize go  
478 ÏÏ§  
479 ÏÏ§  
480 ÏÏ§  
481 ÏÏ§ /\* Endless loop \*/  
482 ÏÏ§  
483 ÏÏ¨¹¹±while(1){ //endless loop  
484 ÏÏ§ÏÏ5  
485 ÏÏ§ÏÏ7¹¹Ïdelay(); //delay for 1 seconds  
486 ÏÏ§ÏÏ5  
487 ÏÏ§ÏÏ7¹³´if(go != 0x01){ //see if button has been pressed  
488 ÏÏ§ÏÏ5Ï6§  
489 ÏÏ§ÏÏ5Ï6¾¹¹Ïgo=~go;  
490 ÏÏ§ÏÏ5Ï6Ï  
491 ÏÏ§ÏÏ5Ï¶Ï}  
492 ÏÏ§ÏÏ5  
493 ÏÏ§ÏÏ°} /\* repeat forever \*/  
494 ÏÏ§  
495 ÏÏ©}  
496 ÏÏÏ