

```

1  #include "stdint.h"
2  #include "stdio.h"
3  #include "stdlib.h"
4  #include "tm4c123gh6pm.h"
5  #include "UART.h"
6  #include "PLL.h"
7
8  //pwm def
9  #define PWM_1_GENB_ACTCMPBD_ONE 0x00000C00 // Set the output signal to 1
10 #define PWM_1_GENB_ACTLOAD_ZERO 0x00000008 // Set the output signal to 0
11
12 #define SYSCTL_RCC_USEPWMDIV      0x00100000 // Enable PWM Clock Divisor
13 #define SYSCTL_RCC_PWMDIV_M       0x000E0000 // PWM Unit Clock Divisor
14 #define SYSCTL_RCC_PWMDIV_2       0x00000000 // /2
15
16 #define UART_FR_TXFF               0x00000020 // UART Transmit FIFO Full
17 #define UART_FR_RXFE               0x00000010 // UART Receive FIFO Empty
18 #define UART_LCRH_WLEN_8           0x00000060 // 8 bit word length
19 #define UART_LCRH_FEN              0x00000010 // UART Enable FIFOs
20 #define UART_CTL_UARTEN            0x00000001 // UART Enable
21
22 #define IR (*(volatile unsigned long *)0x40006040) //PC4
23 #define blue 0x04
24 #define red 0x02
25 #define green 0x08
26 #define white 0x0E
27
28
29
30 // basic functions defined at end of startup.s
31 void DisableInterrupts(void); // Disable interrupts
32 void EnableInterrupts(void); // Enable interrupts
33 void WaitForInterrupt(void); // low power mode
34 void OutCRLF(void);
35
36 char value[5];
37 char string[4];
38 char buffer[4];
39 char freq;
40 unsigned int a,b,c, bright;
41
42 char temp;
43 unsigned short i;
44 unsigned long dutyCycle;
45 unsigned long press;
46 char msg;
47 char *test;
48 char in;
49
50 unsigned int device;
51 unsigned char cmd;
52 unsigned long IR_current_time=0; // counter tracking IR time
53 unsigned char IR_busy=0; // IR busy flag
54 unsigned char Got_IR=0; // IR transmission request
55
56
57
58 int val;
59 void slice_str( char * str, char * buffer, unsigned int start, unsigned int end);
60
61 /*****
62     PORT F INIT
63     *****/
64 void PortF_Init(void){
65     volatile unsigned long delay;
66     SYSCTL_RCGC2_R |= 0x00000020; // 1) F clock
67     delay = SYSCTL_RCGC2_R; // delay

```

```

68
69     GPIO_PORTF_LOCK_R   = 0x4C4F434B; // 2) unlock PortF PF0
70     GPIO_PORTF_CR_R    |= 0x1F;      // allow changes to PF4-0
71     GPIO_PORTF_DEN_R    |= 0x1F;
72     GPIO_PORTF_AMSEL_R  &= ~0x1F;    // 3) disable analog function
73     GPIO_PORTF_PCTL_R   &= ~0x000F0F0F; // 4) GPIO clear bit PCTL
74     //GPIO_PORTF_PCTL_R |= 0x00055555;
75
76     GPIO_PORTF_DIR_R    &= ~0x10;    // PF2 output
77
78     GPIO_PORTF_DIR_R    |= 0x0F;      // PF2 output
79     GPIO_PORTF_AFSEL_R  &= ~0x1F;    // 6) no alternate function
80     GPIO_PORTF_PUR_R    |= 0x1F;    // enable weak pull-up on PF4,0
81
82     GPIO_PORTF_IS_R     &= ~0x10;
83     GPIO_PORTF_IBE_R    &= ~0x10;
84     GPIO_PORTF_IEV_R    &= ~0x10;
85
86     GPIO_PORTF_ICR_R    = 0x10;
87     GPIO_PORTF_IM_R     |= 0x10;
88
89     NVIC_PRI7_R = (NVIC_PRI7_R & 0xFFFFFFF) | 0x00400000; // (g) priority 2
90     NVIC_EN0_R = 0x40000000; // (h) enable interrupt 30 in NVIC
91 }
92
93 /*****
94     PORT B INIT
95 *****/
96 /*
97 void PortB_Init(void){
98     unsigned volatile delay;
99     SYSCTL_RCGCGPIO_R |= 0x00000002; // activate clock for Port B
100     delay = SYSCTL_RCGCGPIO_R;
101     GPIO_PORTB_AMSEL_R &= 0x00;      // disable analog on PB
102     GPIO_PORTB_PCTL_R &= ~0xF0000000; // PCTL GPIO on PB7
103     GPIO_PORTB_PCTL_R |= GPIO_PCTL_PB7_M0PWM1;
104     GPIO_PORTB_DIR_R  |= 0x80;      // PB0-7 are outputs
105     GPIO_PORTB_AFSEL_R |= 0x80;    // enable alt. function on PB7
106     GPIO_PORTB_DEN_R  |= 0x80;    // enable digital I/O on PB7
107
108     // PWM FOR PB7
109     SYSCTL_RCGCPWM_R |= 1;          // enable PWM module 0
110     delay = SYSCTL_RCGCPWM_R;
111     SYSCTL_RCC_R |= 0x00100000;    // start the clock divider
112     SYSCTL_RCC_R &= ~0x000C0000;  // 0x1, 80Mhz/4, 20Mhz clock
113     PWM0_0_CTL_R &= ~0x01;        // disable PWM for setup
114     PWM0_0_GENB_R = 0x0000080C;    // M0PWM1 output set when reload
115     PWM0_0_LOAD_R = 526 - 1;      // setting the reload to be 40KHz
116     PWM0_0_CMPB_R = 263 - 1;      // compare the values for low signal
117 }
118 */
119
120 /*****
121     PORTC Init for IR LED
122 *****/
123 void PortC_Init(void){ volatile unsigned long delay;
124     SYSCTL_RCGCGPIO_R |= 0x04;    // 2) activate port C
125     GPIO_PORTC_LOCK_R = GPIO_LOCK_KEY;
126     delay = SYSCTL_RCGCGPIO_R;    // allow time to finish activating delay here
127     GPIO_PORTC_AFSEL_R &= ~0x10;  // enable alt funct on PC4
128     GPIO_PORTC_PCTL_R &= ~0x000F0000; // configure PC4 as GPIO
129     GPIO_PORTC_AMSEL_R &= ~0x10;  // disable analog functionality on PC4
130     GPIO_PORTC_DIR_R  |= 0x10;    // Output PC4
131     GPIO_PORTC_DEN_R  |= 0x10;    // enable digital I/O on PC4
132     IR = 0;                      // Turn off IR LED

```

```

133 }
134
135
136 /*****
137     IR LED PWM INIT
138     *****/
139 void PWM1_A6_Init(uint32_t period, uint32_t duty){ unsigned long volatile delay;
140
141     SYSCTL_RCGCPWM_R |= 0x02;          // 1) activate PWM1
142     //SYSCTL_RCGCGPIO_R |=0x01; // PortA
143     delay = SYSCTL_RCGCGPIO_R;
144     SYSCTL_RCC_R = 0x00100000 |          // 3) use PWM divider
145         (SYSCTL_RCC_R & (~0x000E0000)); // ~ might mess up
146     PWM1_1_CTL_R = 0x00000000;          // 4) re-loading down-counting mode
147     PWM1_1_GENA_R = 0x000000C8;          // Generator B set-up
148     PWM1_1_LOAD_R = period - 1;          // 5) cycles needed to count down to 0
149     PWM1_1_CMPA_R = duty - 1;            // 6) count value when output rises
150     PWM1_1_CTL_R |= 0x00000002;          // 7) start PWM1
151     PWM1_ENABLE_R |= 0x00000002;          // enable PF2 M1PWM6
152 }
153 void PWM1_A6_Duty(uint16_t duty){
154     PWM1_1_CMPA_R = duty -1;
155 }
156
157 /*****
158     LED DIM PWM
159     *****/
160
161 void PWM1A_Init( uint32_t period, uint32_t duty){ unsigned long volatile delay;
162     //dutyCycle = 1255;          // duty cycle for red LED
163
164     SYSCTL_RCGCPWM_R |= 0x02;          // 1) activate PWM1
165     delay = SYSCTL_RCGCGPIO_R;
166     SYSCTL_RCGCGPIO_R |= 0x01;          // 2) activate port A
167     while((SYSCTL_PRGPIO_R&0x01) == 0){};
168     GPIO_PORTA_AFSEL_R |= 0x40;          // enable alt funct on PA6
169     GPIO_PORTA_PCTL_R  &= ~0x0F000000;  // configure PA6 as PWM0
170     GPIO_PORTA_PCTL_R  |= 0x04000000;
171     GPIO_PORTA_AMSEL_R &= ~0x40;          // disable analog functionality on PA6
172     GPIO_PORTA_DEN_R   |= 0x40;          // enable digital I/O on PA6
173
174     SYSCTL_RCC_R = 0x00100000 |          // 3) use PWM divider
175         (SYSCTL_RCC_R & (~0x000E0000)); // ~ might mess up
176     PWM1_3_CTL_R = 0x00000000;          // 4) re-loading down-counting mode
177     PWM1_3_GENA_R = 0x000000C8;          // Generator B set-up
178     PWM1_3_LOAD_R = period - 1;          // 5) cycles needed to count down to 0
179     PWM1_3_CMPA_R = duty - 1;            // 6) count value when output rises
180     PWM1_3_CTL_R |= 1;                    // 7) start PWM1
181     PWM1_ENABLE_R |= 0x00000040;          // enable PF2 M1PWM6
182
183 }
184
185 void PWM1A_Duty( uint16_t duty){
186     PWM1_3_CMPA_R = duty -1;
187 }
188 /*****
189     SysTckInit *note* might need (unsigned long period) if nothing
190     can be passed in the SysTick Init()in main()
191     *****/
192 void SysTick_Init(unsigned long period) {
193     NVIC_ST_CTRL_R = 0;          // disable SysTick during setup
194     NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M; // reload value
195     NVIC_ST_CURRENT_R = 0;        // any write to current clears it

```

```

196     //NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0xC0000000; // priority 6
197     // enable SysTick with core clock and interrupts
198     NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC;
199 }
200
201 /*****
202     System Wait for different delays
203 *****/
204 void SysTick_Wait(unsigned long delay){
205     volatile unsigned long elapsedTime;
206     unsigned long startTime = NVIC_ST_CURRENT_R;
207     do{
208         elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
209     }
210     while(elapsedTime <= delay);
211 } // end of systick_wait
212
213 void SysTick_Wait1ms(uint32_t delay){
214     uint32_t i;
215     for( i = 0; i<delay; i++)
216     {
217         SysTick_Wait(80000);
218     }
219     }// systick_wait1ms
220
221 void SysTick_Wait800us( uint32_t delay){
222     uint32_t i;
223     for( i = 0; i<delay; i++)
224     {
225         SysTick_Wait(64000);
226     }
227     }// systick_wait800us
228
229 void SysTick_Wait450us( uint32_t delay){
230     uint32_t i;
231     for( i = 0; i<delay; i++)
232     {
233         SysTick_Wait(36000);
234     }
235     }// systick_wait450us
236
237 void SysTick_Wait900us( uint32_t delay){
238     uint32_t i;
239     for( i = 0; i<delay; i++)
240     {
241         SysTick_Wait(72000);
242     }
243     }// systick_wait800us
244
245
246 ///////////////////////////////////////////////////
247 /*****
248     Data frame function
249 *****/
250 /*
251 //device is 2bits, cmd is 3bits
252 void data_frame(uint32_t device, uint32_t cmd){
253     UART_OutString("switch here");
254     UART_OutUDec(cmd);
255     switch(cmd){
256         case 0 : UART_OutString("found it");
257
258         case 1 : GPIO_PORTA_DATA_R = device | 0x001;
259

```

```

260         case 2 : GPIO_PORTA_DATA_R = device | 0x010;
261
262         case 3 : GPIO_PORTA_DATA_R = device | 0x011;
263
264         case 4 : GPIO_PORTA_DATA_R = device | 0x100;
265
266         case 5 : GPIO_PORTA_DATA_R = device | 0x101;
267
268         case 6 : GPIO_PORTA_DATA_R = device | 0x110;
269
270         case 7 : GPIO_PORTA_DATA_R = device | 0x111;
271
272         default: GPIO_PORTA_DATA_R = 0x00000;
273     }
274     } // end
275
276     /*
277     /*****
278     Start Function
279     *****/
280     void start_Pulse(void) {
281         PWM1_ENABLE_R |= 0x00000002;
282         SysTick_Wait1ms(1);
283         PWM1_ENABLE_R &= ~0x00000002;
284         SysTick_Wait800us(1);
285
286     }
287     /*****
288     Logic 0 function
289     *****/
290     void logic0() {
291         //PWM1_1_CTL_R |= # ;
292         PWM1_ENABLE_R |= 0x00000002;
293         SysTick_Wait450us(1);
294         //PWM1_1_CTL_R &= ~# ;
295         PWM1_ENABLE_R &= ~0x00000002;
296         SysTick_Wait450us(1);
297     }
298
299     /*****
300     Logic 1 function
301     *****/
302     void logic1() {
303         //PWM1_1_CTL_R |= #;
304         PWM1_ENABLE_R |= 0x00000002;
305         SysTick_Wait900us(1);
306         //PWM1_1_CTL_R &= ~# ;
307         PWM1_ENABLE_R &= ~0x00000002;
308         SysTick_Wait450us(1);
309     }
310
311
312     void address_0(void) {
313         logic0();
314         logic0();
315
316     }
317
318     void address_1(void) {
319         logic0();
320         logic1();
321
322     }

```

```
323
324 void address_2(void){
325     logic1();
326     logic0();
327
328 }
329
330 void address_3(void){
331     logic1();
332     logic1();
333
334 }
335
336 void Command_0(void){
337     logic0();
338     logic0();
339     logic0();
340
341 }
342
343 void Command_1(void){
344     logic0();
345     logic0();
346     logic1();
347 }
348
349 void Command_2(void){
350     logic0();
351     logic1();
352     logic0();
353
354 }
355 void Command_3(void){
356     logic0();
357     logic1();
358     logic1();
359
360 }
361
362 void Command_4(void){
363     logic1();
364     logic0();
365     logic0();
366
367 }
368
369 void Command_5(void){
370     logic1();
371     logic0();
372     logic1();
373
374 }
375
376 void Command_6(void){
377     logic1();
378     logic1();
379     logic0();
380
381 }
382
383 void Command_7(void){
384     logic1();
385     logic1();
386     logic1();
387
388 }
389
390 void device_select(uint16_t device_sel){
391     if( device_sel == 0 ){ address_0(); }
```

```

392     else if( device_sel == 1 ){ address_1(); }
393     else if( device_sel == 2 ){ address_2(); }
394     else if( device_sel == 3 ){ address_3(); }
395     else{address_0();}
396 }
397 /*****
398     Port F handler for button push
399 *****/
400 void GPIOPortF_Handler(void){ // called on touch of either SW1 or SW2
401
402     if(GPIO_PORTF_RIS_R&0x10){ // SW2 touch
403         GPIO_PORTF_ICR_R = 0x10; // acknowledge flag0
404         press++;
405     }
406
407     if(press == 1){
408
409         GPIO_PORTF_DATA_R &= ~0x0E;
410         GPIO_PORTF_DATA_R |= red;
411         device = 0x00;
412         UART_OutString("device has been changed to 0 \r\n");
413         UART1_OutString("device has been changed to 0 \r\n");
414     }
415     else if(press==2){
416
417         GPIO_PORTF_DATA_R &= ~0x0E;
418         GPIO_PORTF_DATA_R |= green;
419         device = 0x01;
420         UART_OutString("device has been changed to 1 \r\n");
421         UART1_OutString("device has been changed to 1 \r\n");
422     }
423     else if(press==3){
424
425         GPIO_PORTF_DATA_R &= ~0x0E;
426         GPIO_PORTF_DATA_R |= blue;
427         device = 0x02;
428         UART_OutString("device has been changed to 2 \r\n");
429         UART1_OutString("device has been changed to 2 \r\n");
430     }
431     else if(press==4){
432
433         GPIO_PORTF_DATA_R &= ~0x0E;
434         GPIO_PORTF_DATA_R |= white;
435         device = 0x03;
436         UART_OutString("device has been changed to 3 \r\n");
437         UART1_OutString("device has been changed to 3 \r\n");
438         press = 0;
439     }
440     //else{
441         //GPIO_PORTF_DATA_R &= ~0x0E;
442         //GPIO_PORTF_DATA_R |= red;
443         //device = 0x00;
444     //}
445
446 } // end of handler
447
448
449
450
451
452 int main(void){
453     PLL_Init();
454     PortF_Init();
455     PortC_Init();
456     UART_Init();
457     PortB_UART1_Init();
458     PWM1A_Init(256, 80);

```

```

459     PWM1_A6_Init(1053,526); // change value
460     EnableInterrupts();
461     GPIO_PORTF_DATA_R = 0x02;
462
463     //SysTick_Init();
464     //device = 0;
465     SysTick_Init(625); // 80KHz interrupt
466     while(1){
467         cmd = UART1_InChar();
468         //UART_OutChar(cmd);
469         OutCRLF();
470
471         if(cmd != 0){
472
473             switch(cmd){
474
475                 case '0' : start_Pulse();
476
477                             UART1_OutString("Start\r\n");
478
479                             device_select(device);
480
481                             UART1_OutString("device# ");
482                             UART1_OutUDec(device);
483                             UART1_OutString("\r\n");
484
485                             Command_0();
486
487                             UART1_OutString("command# ");
488                             UART1_OutChar(cmd);
489                             UART1_OutString("\r\n");
490
491                             break;
492
493                 case '1' : start_Pulse();
494                             device_select(device);
495                             Command_1();
496
497                             break;
498
499                 case '2': start_Pulse();
500                             device_select(device);
501                             Command_2();
502                             break;
503
504                 case '3' : start_Pulse();
505                             device_select(device);
506                             Command_3();
507                             break;
508
509                 case '4' : start_Pulse();
510                             device_select(device);
511                             Command_4();
512                             break;
513
514                 case '5' : start_Pulse();
515                             device_select(device);
516                             Command_5();
517                             break;
518
519                 case '6' : start_Pulse();
520                             device_select(device);
521                             Command_6();
522                             break;
523
524                 case '7' : start_Pulse();
525                             device_select(device);
526                             Command_7();
527                             break;

```



```

528         default: break;
529
530     } // end of switch case
531
532
533     } // end of if cmd is valid
534
535
536
537
538
539     } // end of while(1)
540
541 } //end of main
542
543 /*****
544     UART Clear Line Feed
545 *****/
546 void OutCRLF(void){
547     UART_OutChar(CR);
548     UART_OutChar(LF);
549 }
550
551 /*****
552     String Slice for UART7
553 *****/
554 void slice_str(char *value, char *buffer, unsigned int start, unsigned int end)
555 {
556     unsigned int j = 0;
557     unsigned int i = 0;
558     for (i = start; i <= end; i++) {
559         buffer[j++] = value[i];
560     }
561     buffer[j] = 0;
562 }
563
564 //////////// END OF CODE
565
```