

CPE 325: Embedded Systems Laboratory

Laboratory Tutorial #6:

MSP430 Interrupts and Universal Clock Subsystem

Aleksandar Milenković

Email: milenka@uah.edu

Web: <http://www.ece.uah.edu/~milenka>

Objective:

This tutorial will teach you how to write interrupt service routines in assembly and C/C++ programming languages. In addition, it will describe the universal clock subsystem of the MSP430F5529 device that is responsible for generating internal clocks. You will learn the following topics:

Using interrupts in C/assembly (specifically working with port interrupts)

The clock subsystem and clock configuration

Working with the TI experimenter's board

Notes:

All previous tutorials are required for successful completion of this lab, especially, the tutorials introducing the MSP-EXP430F5529LP board and the Code Composer Studio software development environment.

Contents

Objective:	1
Notes:	1
Contents	1
1 Interfacing Switches and LEDs in Assembly (Polling and Interrupts)	2
1.1 Toggling LEDs in Assembly Language	2
1.2 Interfacing Switches in Assembly Language (Polling)	4
1.3 Interfacing Switches in Assembly Language (Interrupt Service Routine)	6
2 Interfacing Switches and LEDs Using Interrupts in C	8
3 Clock Module	11
3.1 Unified Clock System (UCS)	11
3.2 Changing Processor Clocks: Examples	14
4 References	20

1 Interfacing Switches and LEDs in Assembly (Polling and Interrupts)

In the handout for Laboratory #3 we learned how to interface with the MSP-EX430F5529LP hardware, specifically LEDs and switches, using C language. We will redo the same examples using the MSP430 assembly language.

1.1 Toggling LEDs in Assembly Language

Figure 1 shows the assembly code of the blink application (Lab6_D1.asm). Here is a brief description of the assembly code for this application. In addition to the portions of the code that were discussed in the previous labs we can discuss some new additions. The .text is a segment control assembler directive that controls how code and data are located in memory. .text is used to mark the beginning of a relocatable code. The linker can recognize any other type of segment (e.g., __STACK_END for code stack). Our main loop that flashes the LEDs starts at the InfLoop label. The code starting at the label SWDelay1 implements the software delay to make sure the LEDs blink at the appropriate interval. To exactly calculate the software delay we need to know the instruction execution time and the clock cycle time. The register R15 is loaded with 65,535 (the maximum unsigned integer that can fit in a 16-bit register). The dec.w instruction takes 1 clock cycle to execute, and jnz L1 takes 2 clock cycles to execute (note: this can be determined by enabling and reading the value of the clock in CCS). The nop instruction takes 1 clock cycle. The number of nop instructions in the loop is determined so that the total number of clocks in the SWDelay1 loop is 16. Determining clock cycle time requires in-depth understanding of the FLL-Clock module of the MSP430 which is discussed later in this tutorial. We note that the processor clock frequency is 1,048,576 Hz (2^{20} Hz) for the default configuration. The total delay is thus $65,535 \cdot 16 / 2^{20} \sim 1s$. Note: nop instructions are often used in creating software delays because they do not affect the state of the registers and take exactly one clock cycle to execute.

```
1 ;-----
2 ; File:      Lab6_D1.asm
3 ; Description: The program toggles LEDs periodically.
4 ;             LED1 is initialized off, LED2 is initialized on.
5 ;             Main program loop:
6 ;                 the SWDelay1 loop creates 1s delay before
7 ;                 toggling the LEDs (ON/OFF).
8 ;
9 ; Clocks:     ACLK = 32.768kHz, MCLK = SMCLK = default DCO = 2^20=1,048,576 Hz
10 ; Platform:   TI EXP430F5529LP Launchpad
11 ;
12 ;             MSP430xF5529
13 ;
14 ;             /|\|
15 ;             | |
16 ;             --| RST
17 ;             |
18 ;             | P1.0|-->LED1(RED)
19 ;             | P4.7|-->LED2(GREEN)
20 ;
21 ; Author:     Aleksandar Milenkovic, milenkovic@computer.org
```

```

21 ; Date:      September 14, 2018
22 ; Modified:   Prawar Poudel, August 08, 2019
23 ;-----
24 ; MSP430 Assembler Code Template for use with TI Code Composer Studio
25 ;
26 ;
27 ;-----
28         .cdecls C,LIST,"msp430.h"          ; Include device header file
29
30 ;-----
31         .def      RESET                    ; Export program entry-point to
32                                         ; make it known to linker.
33 ;-----
34         .text                             ; Assemble into program memory.
35         .retain                             ; Override ELF conditional linking
36                                         ; and retain current section.
37         .retainrefs                         ; And retain any sections that have
38                                         ; references to current section.
39
40 ;-----
41 RESET    mov.w    #__STACK_END,SP          ; Initialize stackpointer
42 StopWDT   mov.w    #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer
43
44 SETUP:
45         bis.b     #0x01,&P1DIR              ; set P1.0 as output, 0'b0000 0001
46         bis.b     #0x80,&P4DIR              ; set P4.7 as output, 0'b1000 0000
47
48         bic.b     #0x01,&P1OUT              ; turn P1.0 OFF
49         bis.b     #0x80,&P4OUT              ; turn P4.7 ON
50 ;-----
51 ; Main loop here
52 ;-----
53 InfLoop:
54         mov.w     #0xFFFF,R5               ; move 0xFFFF to R5 which will be out counter
55 SWDelay1:
56         nop
57         nop
58         nop
59         nop
60         nop
61         nop
62         nop
63         nop
64         nop                                ; 13 NOPs + extra 3cc is a delay of 16cc
65         nop                                ; so the total delay is 65535*16cc/2^20 ~ 1s
66         nop
67         nop
68         nop
69         dec.w     R5                        ; 1cc
70         jnz       SWDelay1                  ; 2cc
71         xor.b     #0x01,&P1OUT              ; toggle 1.0
72         xor.b     #0x80,&P4OUT              ; toggle 4.7
73         jmp       InfLoop                   ; go to InfLoop
74         nop
75

```

```

76
77 ;-----
78 ; Stack Pointer definition
79 ;-----
80     .global __STACK_END
81     .sect   .stack
82
83 ;-----
84 ; Interrupt Vectors
85 ;-----
86     .sect   ".reset"                ; MSP430 RESET Vector
87     .short  RESET
88

```

Figure 1. Blinking the LEDs in Assembly Language

1.2 Interfacing Switches in Assembly Language (Polling)

Figure 2 shows assembly program that interfaces S1 and LED1. S1 is connected to P1.BIT0 (ports are configured by default as input) and LED1 is connected to P2.BIT2 (should be configured as a digital output). BIT0 of P1 is checked. If pressed a logic 0 should be detected in P1IN.BIT0; otherwise it should read as a logic 1. When a press is detected, a software delay of 20 ms is implemented to support de-bouncing of the switch. If the switch is still pressed, the program turns on LED1. The program continually checks whether the switch is still pressed. If a release (depress) is detected, LED1 is turned off.

```

1  ;-----
2  ; File:      Lab6_D2.asm
3  ; Description: The program demonstrates Press/Release using S1 and LED1.
4  ; LED1 is initialized off.
5  ; When an S1 press is detected, a software delay of 20 ms
6  ; is used to implement debouncing. The switch is checked
7  ; again, and if it's on, LED1 is turned on until S1 is released.
8  ;
9  ; Clocks:    ACLK = 32.768kHz, MCLK = SMCLK = default DCO = 2^20=1,048,576 Hz
10 ; Platform:  TI EXP430F5529LP Launchpad
11 ;
12 ;           MSP430F5529
13 ;
14 ;           /|\
15 ;           |
16 ;           --RST
17 ;
18 ;           |
19 ;           |
20 ;           |
21 ;           |
22 ;           |
23 ;           |
24 ;           |
25 ;           |
26 ;           |
27 ;           |
28 ;           |
29 ;           |
30 ;           |
31 ;           |
32 ;           |
33 ;           |
34 ;           |
35 ;           |
36 ;           |
37 ;           |
38 ;           |
39 ;           |
40 ;           |
41 ;           |
42 ;           |
43 ;           |
44 ;           |
45 ;           |
46 ;           |
47 ;           |
48 ;           |
49 ;           |
50 ;           |
51 ;           |
52 ;           |
53 ;           |
54 ;           |
55 ;           |
56 ;           |
57 ;           |
58 ;           |
59 ;           |
60 ;           |
61 ;           |
62 ;           |
63 ;           |
64 ;           |
65 ;           |
66 ;           |
67 ;           |
68 ;           |
69 ;           |
70 ;           |
71 ;           |
72 ;           |
73 ;           |
74 ;           |
75 ;           |
76 ;           |
77 ;           |
78 ;           |
79 ;           |
80 ;           |
81 ;           |
82 ;           |
83 ;           |
84 ;           |
85 ;           |
86 ;           |
87 ;           |
88 ;           |
89 ;           |
90 ;           |
91 ;           |
92 ;           |
93 ;           |
94 ;           |
95 ;           |
96 ;           |
97 ;           |
98 ;           |
99 ;           |
100 ;           |
101 ;           |
102 ;           |
103 ;           |
104 ;           |
105 ;           |
106 ;           |
107 ;           |
108 ;           |
109 ;           |
110 ;           |
111 ;           |
112 ;           |
113 ;           |
114 ;           |
115 ;           |
116 ;           |
117 ;           |
118 ;           |
119 ;           |
120 ;           |
121 ;           |
122 ;           |
123 ;           |
124 ;           |
125 ;           |
126 ;           |
127 ;           |
128 ;           |
129 ;           |
130 ;           |
131 ;           |
132 ;           |
133 ;           |
134 ;           |
135 ;           |
136 ;           |
137 ;           |
138 ;           |
139 ;           |
140 ;           |
141 ;           |
142 ;           |
143 ;           |
144 ;           |
145 ;           |
146 ;           |
147 ;           |
148 ;           |
149 ;           |
150 ;           |
151 ;           |
152 ;           |
153 ;           |
154 ;           |
155 ;           |
156 ;           |
157 ;           |
158 ;           |
159 ;           |
160 ;           |
161 ;           |
162 ;           |
163 ;           |
164 ;           |
165 ;           |
166 ;           |
167 ;           |
168 ;           |
169 ;           |
170 ;           |
171 ;           |
172 ;           |
173 ;           |
174 ;           |
175 ;           |
176 ;           |
177 ;           |
178 ;           |
179 ;           |
180 ;           |
181 ;           |
182 ;           |
183 ;           |
184 ;           |
185 ;           |
186 ;           |
187 ;           |
188 ;           |
189 ;           |
190 ;           |
191 ;           |
192 ;           |
193 ;           |
194 ;           |
195 ;           |
196 ;           |
197 ;           |
198 ;           |
199 ;           |
200 ;           |
201 ;           |
202 ;           |
203 ;           |
204 ;           |
205 ;           |
206 ;           |
207 ;           |
208 ;           |
209 ;           |
210 ;           |
211 ;           |
212 ;           |
213 ;           |
214 ;           |
215 ;           |
216 ;           |
217 ;           |
218 ;           |
219 ;           |
220 ;           |
221 ;           |
222 ;           |
223 ;           |
224 ;           |
225 ;           |
226 ;           |
227 ;           |
228 ;           |
229 ;           |
230 ;           |
231 ;           |
232 ;           |
233 ;           |
234 ;           |
235 ;           |
236 ;           |
237 ;           |
238 ;           |
239 ;           |
240 ;           |
241 ;           |
242 ;           |
243 ;           |
244 ;           |
245 ;           |
246 ;           |
247 ;           |
248 ;           |
249 ;           |
250 ;           |
251 ;           |
252 ;           |
253 ;           |
254 ;           |
255 ;           |
256 ;           |
257 ;           |
258 ;           |
259 ;           |
260 ;           |
261 ;           |
262 ;           |
263 ;           |
264 ;           |
265 ;           |
266 ;           |
267 ;           |
268 ;           |
269 ;           |
270 ;           |
271 ;           |
272 ;           |
273 ;           |
274 ;           |
275 ;           |
276 ;           |
277 ;           |
278 ;           |
279 ;           |
280 ;           |
281 ;           |
282 ;           |
283 ;           |
284 ;           |
285 ;           |
286 ;           |
287 ;           |
288 ;           |
289 ;           |
290 ;           |
291 ;           |
292 ;           |
293 ;           |
294 ;           |
295 ;           |
296 ;           |
297 ;           |
298 ;           |
299 ;           |
300 ;           |
301 ;           |
302 ;           |
303 ;           |
304 ;           |
305 ;           |
306 ;           |
307 ;           |
308 ;           |
309 ;           |
310 ;           |
311 ;           |
312 ;           |
313 ;           |
314 ;           |
315 ;           |
316 ;           |
317 ;           |
318 ;           |
319 ;           |
320 ;           |
321 ;           |
322 ;           |
323 ;           |
324 ;           |
325 ;           |
326 ;           |
327 ;           |
328 ;           |
329 ;           |
330 ;           |
331 ;           |
332 ;           |
333 ;           |
334 ;           |
335 ;           |
336 ;           |
337 ;           |
338 ;           |
339 ;           |
340 ;           |
341 ;           |
342 ;           |
343 ;           |
344 ;           |
345 ;           |
346 ;           |
347 ;           |
348 ;           |
349 ;           |
350 ;           |
351 ;           |
352 ;           |
353 ;           |
354 ;           |
355 ;           |
356 ;           |
357 ;           |
358 ;           |
359 ;           |
360 ;           |
361 ;           |
362 ;           |
363 ;           |
364 ;           |
365 ;           |
366 ;           |
367 ;           |
368 ;           |
369 ;           |
370 ;           |
371 ;           |
372 ;           |
373 ;           |
374 ;           |
375 ;           |
376 ;           |
377 ;           |
378 ;           |
379 ;           |
380 ;           |
381 ;           |
382 ;           |
383 ;           |
384 ;           |
385 ;           |
386 ;           |
387 ;           |
388 ;           |
389 ;           |
390 ;           |
391 ;           |
392 ;           |
393 ;           |
394 ;           |
395 ;           |
396 ;           |
397 ;           |
398 ;           |
399 ;           |
400 ;           |
401 ;           |
402 ;           |
403 ;           |
404 ;           |
405 ;           |
406 ;           |
407 ;           |
408 ;           |
409 ;           |
410 ;           |
411 ;           |
412 ;           |
413 ;           |
414 ;           |
415 ;           |
416 ;           |
417 ;           |
418 ;           |
419 ;           |
420 ;           |
421 ;           |
422 ;           |
423 ;           |
424 ;           |
425 ;           |
426 ;           |
427 ;           |
428 ;           |
429 ;           |
430 ;           |
431 ;           |
432 ;           |
433 ;           |
434 ;           |
435 ;           |
436 ;           |
437 ;           |
438 ;           |
439 ;           |
440 ;           |
441 ;           |
442 ;           |
443 ;           |
444 ;           |
445 ;           |
446 ;           |
447 ;           |
448 ;           |
449 ;           |
450 ;           |
451 ;           |
452 ;           |
453 ;           |
454 ;           |
455 ;           |
456 ;           |
457 ;           |
458 ;           |
459 ;           |
460 ;           |
461 ;           |
462 ;           |
463 ;           |
464 ;           |
465 ;           |
466 ;           |
467 ;           |
468 ;           |
469 ;           |
470 ;           |
471 ;           |
472 ;           |
473 ;           |
474 ;           |
475 ;           |
476 ;           |
477 ;           |
478 ;           |
479 ;           |
480 ;           |
481 ;           |
482 ;           |
483 ;           |
484 ;           |
485 ;           |
486 ;           |
487 ;           |
488 ;           |
489 ;           |
490 ;           |
491 ;           |
492 ;           |
493 ;           |
494 ;           |
495 ;           |
496 ;           |
497 ;           |
498 ;           |
499 ;           |
500 ;           |
501 ;           |
502 ;           |
503 ;           |
504 ;           |
505 ;           |
506 ;           |
507 ;           |
508 ;           |
509 ;           |
510 ;           |
511 ;           |
512 ;           |
513 ;           |
514 ;           |
515 ;           |
516 ;           |
517 ;           |
518 ;           |
519 ;           |
520 ;           |
521 ;           |
522 ;           |
523 ;           |
524 ;           |
525 ;           |
526 ;           |
527 ;           |
528 ;           |
529 ;           |
530 ;           |
531 ;           |
532 ;           |
533 ;           |
534 ;           |
535 ;           |
536 ;           |
537 ;           |
538 ;           |
539 ;           |
540 ;           |
541 ;           |
542 ;           |
543 ;           |
544 ;           |
545 ;           |
546 ;           |
547 ;           |
548 ;           |
549 ;           |
550 ;           |
551 ;           |
552 ;           |
553 ;           |
554 ;           |
555 ;           |
556 ;           |
557 ;           |
558 ;           |
559 ;           |
560 ;           |
561 ;           |
562 ;           |
563 ;           |
564 ;           |
565 ;           |
566 ;           |
567 ;           |
568 ;           |
569 ;           |
570 ;           |
571 ;           |
572 ;           |
573 ;           |
574 ;           |
575 ;           |
576 ;           |
577 ;           |
578 ;           |
579 ;           |
580 ;           |
581 ;           |
582 ;           |
583 ;           |
584 ;           |
585 ;           |
586 ;           |
587 ;           |
588 ;           |
589 ;           |
590 ;           |
591 ;           |
592 ;           |
593 ;           |
594 ;           |
595 ;           |
596 ;           |
597 ;           |
598 ;           |
599 ;           |
600 ;           |
601 ;           |
602 ;           |
603 ;           |
604 ;           |
605 ;           |
606 ;           |
607 ;           |
608 ;           |
609 ;           |
610 ;           |
611 ;           |
612 ;           |
613 ;           |
614 ;           |
615 ;           |
616 ;           |
617 ;           |
618 ;           |
619 ;           |
620 ;           |
621 ;           |
622 ;           |
623 ;           |
624 ;           |
625 ;           |
626 ;           |
627 ;           |
628 ;           |
629 ;           |
630 ;           |
631 ;           |
632 ;           |
633 ;           |
634 ;           |
635 ;           |
636 ;           |
637 ;           |
638 ;           |
639 ;           |
640 ;           |
641 ;           |
642 ;           |
643 ;           |
644 ;           |
645 ;           |
646 ;           |
647 ;           |
648 ;           |
649 ;           |
650 ;           |
651 ;           |
652 ;           |
653 ;           |
654 ;           |
655 ;           |
656 ;           |
657 ;           |
658 ;           |
659 ;           |
660 ;           |
661 ;           |
662 ;           |
663 ;           |
664 ;           |
665 ;           |
666 ;           |
667 ;           |
668 ;           |
669 ;           |
670 ;           |
671 ;           |
672 ;           |
673 ;           |
674 ;           |
675 ;           |
676 ;           |
677 ;           |
678 ;           |
679 ;           |
680 ;           |
681 ;           |
682 ;           |
683 ;           |
684 ;           |
685 ;           |
686 ;           |
687 ;           |
688 ;           |
689 ;           |
690 ;           |
691 ;           |
692 ;           |
693 ;           |
694 ;           |
695 ;           |
696 ;           |
697 ;           |
698 ;           |
699 ;           |
700 ;           |
701 ;           |
702 ;           |
703 ;           |
704 ;           |
705 ;           |
706 ;           |
707 ;           |
708 ;           |
709 ;           |
710 ;           |
711 ;           |
712 ;           |
713 ;           |
714 ;           |
715 ;           |
716 ;           |
717 ;           |
718 ;           |
719 ;           |
720 ;           |
721 ;           |
722 ;           |
723 ;           |
724 ;           |
725 ;           |
726 ;           |
727 ;           |
728 ;           |
729 ;           |
730 ;           |
731 ;           |
732 ;           |
733 ;           |
734 ;           |
735 ;           |
736 ;           |
737 ;           |
738 ;           |
739 ;           |
740 ;           |
741 ;           |
742 ;           |
743 ;           |
744 ;           |
745 ;           |
746 ;           |
747 ;           |
748 ;           |
749 ;           |
750 ;           |
751 ;           |
752 ;           |
753 ;           |
754 ;           |
755 ;           |
756 ;           |
757 ;           |
758 ;           |
759 ;           |
760 ;           |
761 ;           |
762 ;           |
763 ;           |
764 ;           |
765 ;           |
766 ;           |
767 ;           |
768 ;           |
769 ;           |
770 ;           |
771 ;           |
772 ;           |
773 ;           |
774 ;           |
775 ;           |
776 ;           |
777 ;           |
778 ;           |
779 ;           |
780 ;           |
781 ;           |
782 ;           |
783 ;           |
784 ;           |
785 ;           |
786 ;           |
787 ;           |
788 ;           |
789 ;           |
790 ;           |
791 ;           |
792 ;           |
793 ;           |
794 ;           |
795 ;           |
796 ;           |
797 ;           |
798 ;           |
799 ;           |
800 ;           |
801 ;           |
802 ;           |
803 ;           |
804 ;           |
805 ;           |
806 ;           |
807 ;           |
808 ;           |
809 ;           |
810 ;           |
811 ;           |
812 ;           |
813 ;           |
814 ;           |
815 ;           |
816 ;           |
817 ;           |
818 ;           |
819 ;           |
820 ;           |
821 ;           |
822 ;           |
823 ;           |
824 ;           |
825 ;           |
826 ;           |
827 ;           |
828 ;           |
829 ;           |
830 ;           |
831 ;           |
832 ;           |
833 ;           |
834 ;           |
835 ;           |
836 ;           |
837 ;           |
838 ;           |
839 ;           |
840 ;           |
841 ;           |
842 ;           |
843 ;           |
844 ;           |
845 ;           |
846 ;           |
847 ;           |
848 ;           |
849 ;           |
850 ;           |
851 ;           |
852 ;           |
853 ;           |
854 ;           |
855 ;           |
856 ;           |
857 ;           |
858 ;           |
859 ;           |
860 ;           |
861 ;           |
862 ;           |
863 ;           |
864 ;           |
865 ;           |
866 ;           |
867 ;           |
868 ;           |
869 ;           |
870 ;           |
871 ;           |
872 ;           |
873 ;           |
874 ;           |
875 ;           |
876 ;           |
877 ;           |
878 ;           |
879 ;           |
880 ;           |
881 ;           |
882 ;           |
883 ;           |
884 ;           |
885 ;           |
886 ;           |
887 ;           |
888 ;           |
889 ;           |
890 ;           |
891 ;           |
892 ;           |
893 ;           |
894 ;           |
895 ;           |
896 ;           |
897 ;           |
898 ;           |
899 ;           |
900 ;           |
901 ;           |
902 ;           |
903 ;           |
904 ;           |
905 ;           |
906 ;           |
907 ;           |
908 ;           |
909 ;           |
910 ;           |
911 ;           |
912 ;           |
913 ;           |
914 ;           |
915 ;           |
916 ;           |
917 ;           |
918 ;           |
919 ;           |
920 ;           |
921 ;           |
922 ;           |
923 ;           |
924 ;           |
925 ;           |
926 ;           |
927 ;           |
928 ;           |
929 ;           |
930 ;           |
931 ;           |
932 ;           |
933 ;           |
934 ;           |
935 ;           |
936 ;           |
937 ;           |
938 ;           |
939 ;           |
940 ;           |
941 ;           |
942 ;           |
943 ;           |
944 ;           |
945 ;           |
946 ;           |
947 ;           |
948 ;           |
949 ;           |
950 ;           |
951 ;           |
952 ;           |
953 ;           |
954 ;           |
955 ;           |
956 ;           |
957 ;           |
958 ;           |
959 ;           |
960 ;           |
961 ;           |
962 ;           |
963 ;           |
964 ;           |
965 ;           |
966 ;           |
967 ;           |
968 ;           |
969 ;           |
970 ;           |
971 ;           |
972 ;           |
973 ;           |
974 ;           |
975 ;           |
976 ;           |
977 ;           |
978 ;           |
979 ;           |
980 ;           |
981 ;           |
982 ;           |
983 ;           |
984 ;           |
985 ;           |
986 ;           |
987 ;           |
988 ;           |
989 ;           |
990 ;           |
991 ;           |
992 ;           |
993 ;           |
994 ;           |
995 ;           |
996 ;           |
997 ;           |
998 ;           |
999 ;           |
1000 ;           |

```

```

28         .def      RESET                ; Export program entry-point to
29                                         ; make it known to linker.
30 ;-----
31         .text                          ; Assemble into program memory.
32         .retain                        ; Override ELF conditional linking
33                                         ; and retain current section.
34         .retainrefs                   ; And retain any sections that have
35                                         ; references to current section.
36
37 ;-----
38 RESET:    mov.w    #__STACK_END,SP      ; Initialize stack pointer
39 StopWDT:  mov.w    #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer
40 ;-----
41 SetupP2:
42         bis.b     #001h, &P1DIR          ; Set P1.0 to output
43                                         ; direction (0000_0001)
44         bic.b     #001h, &P1OUT          ; Set P1OUT to 0x0000_0001 (ensure
45                                         ; LED1 is off)
46
47         bic.b     #002h, &P2DIR          ; SET P2.1 as input for SW1
48         bis.b     #002h, &P2REN          ; Enable Pull-Up resistor at P2.1
49         bis.b     #002h, &P2OUT          ; required for proper IO set up
50
51 ChkSW1:   bic.b     #001h, &P1OUT          ;
52         bit.b     #002h, &P2IN           ; Check if SW1 is pressed
53                                         ; (0000_0010 on P1IN)
54         jnz       ChkSW1                 ; If not zero, SW1 is not pressed
55                                         ; loop and check again
56
57 Debounce:
58 SWD20ms:  mov.w     #2000, R15            ; Set to (2000 * 10 cc = 20,000 cc)
59                                         ; Decrement R15
60         nop
61         nop
62         nop
63         nop
64         nop
65         nop
66         jnz       SWD20ms                ; Delay over?
67         bit.b     #00000010b, &P2IN      ; Verify SW1 is still pressed
68         jnz       ChkSW1                 ; If not, wait for SW1 press
69
70 LEDon:    bis.b     #001h, &P1OUT          ; Turn on LED1
71 SW1wait:  bit.b     #002h, &P2IN          ; Test SW1
72         jz        SW1wait                ; Wait until SW1 is released
73         bic.b     #001h, &P1OUT          ; Turn off LED1
74         jmp       ChkSW1                 ; Loop to beginning
75         nop
76
77 ;-----
78 ; Stack Pointer definition
79 ;-----
80         .global   __STACK_END
81         .sect     .stack
82

```

```

83 ;-----
84 ; Interrupt Vectors
85 ;-----
86 .sect ".reset" ; MSP430 RESET Vector
87 .short RESET
88 .end
89
90

```

Figure 2. Turn on LED1 when S1 is Pressed (Lab6_D2.asm)

1.3 Interfacing Switches in Assembly Language (Interrupt Service Routine)

With microcontrollers, it is often useful to be able to use interrupts in our programs. An interrupt allows an automatic break from the current program flow based on a set of conditions. Some of the I/O ports on the MSP430 have an interrupt capability that you can configure. When the interrupt conditions are met, the program execution departs into a service routine that handles the interrupt event. Once service routine is completed the control transfers back to the main program where it left off using a RETI (return from interrupt) instruction. We will learn more about interrupts in a subsequent lab, but you should understand how interrupt vectors are used and what interrupts do. To set up an interrupt for an I/O port, we have to perform a few tasks:

- Enable global interrupts in the status register
- Enable interrupts to occur for the particular events by setting control bits in corresponding registers associated with ports P1 or P2
- Specify whether the interrupt is triggered on a falling edge or rising edge
- Initialize the interrupt flag by clearing it

An example of using interrupts to interface the switches of the MSP430 experimenter board is shown in Figure 3. The main program configures ports, enables the global interrupts (GIE bit is set), enables interrupt from BIT1 of Port1 (P1IE=0x0000_0010b). As pressing a switch corresponds to having input signal from a logic '1' to a logic '0', the interrupt arises when a falling edge is detected at P1IN.BIT1. The interrupt service routine starts at label SW2_ISR. The state of the input is checked; if P1IN.BIT1 is not a logic 0 we exit the ISR; otherwise, debouncing is performed. If SW2 is still pressed after 20 ms, LED1 is turned on. The program then waits for SW2 to be released. Note lines 94 and 95 that initialize the IVT entry 47 reserved for Port 1.

```

1 ;-----
2 ; File: Lab6_D3.asm
3 ; Description: The program demonstrates Press/Release using S2 and LED1.
4 ; LED1 is initialized off. The main program enables interrupts
5 ; from P1.BIT1 (S2) and remains in an infinite loop doing nothing.
6 ; P1_ISR implements debouncing and waits for a S2 to be released.
7 ;
8 ; Clocks: ACLK = 32.768kHz, MCLK = SMCLK = default DCO = 2^20=1,048,576 Hz
9 ; Platform: TI EXP430F5529LP Launchpad
10 ;
11 ; MSP430F5529

```



```

12 ;
13 ;           /\|
14 ;           |  |
15 ;           --RST
16 ;           |          P1.0|-->LED1(RED)
17 ;           |          P1.1|<--S2
18 ;
19 ;   Author:   Aleksandar Milenkovic, milenkovic@computer.org
20 ;   Date:     September 14, 2018
21 ;   Modified: Prawar Poudel, August 8, 2019
22 ;-----
23         .cdecls C,LIST,"msp430.h"          ; Include device header file
24
25 ;-----
26         .def      RESET                    ; Export program entry-point to
27                                         ; make it known to linker.
28         .def      S2_ISR
29 ;-----
30         .text                               ; Assemble into program memory.
31         .retain                             ; Override ELF conditional linking
32                                         ; and retain current section.
33         .retainrefs                         ; And retain any sections that have
34                                         ; references to current section.
35
36 ;-----
37 RESET:   mov.w    #__STACK_END, SP          ; Initialize stack pointer
38 StopWDT: mov.w    #WDTPW|WDTHOLD, &WDTCTL ; Stop watchdog timer
39
40 Setup:
41         bis.b     #001h, &P1DIR              ; Set P1.0 to output
42                                         ; direction (0000_0001)
43         bic.b     #001h, &P1OUT              ; Set P1OUT to 0x0000_0001
44
45         bic.b     #002h, &P1DIR              ; SET P1.1 as input for S2
46         bis.b     #002h, &P1REN              ; Enable Pull-Up resistor at P1.1
47         bis.b     #002h, &P1OUT              ; required for proper IO set up
48
49
50         bis.w     #GIE, SR                  ; Enable Global Interrupts
51         bis.b     #002h, &P1IE              ; Enable Port 1 interrupt from bit 1
52         bis.b     #002h, &P1IES              ; Set interrupt to call from hi to low
53         bic.b     #002h, &P1IFG              ; Clear interrupt flag
54 InfLoop:
55         jmp       $                          ; Loop here until interrupt
56
57 ;-----
58 ; P1_0 (S2) interrupt service routine (ISR)
59 ;-----
60 S2_ISR:
61         bic.b     #002h, &P1IFG              ; Clear interrupt flag
62 ChkSW2:   bit.b   #02h, &P1IN                ; Check if S2 is pressed
63                                         ; (0000_0010 on P1IN)
64         jnz       LExit                      ; If not zero, SW is not pressed
65                                         ; loop and check again
66 Debounce: mov.w   #2000, R15                  ; Set to (2000 * 10 cc )

```

```

67 SWD20ms:    dec.w    R15                                ; Decrement R15
68            nop
69            nop
70            nop
71            nop
72            nop
73            nop
74            nop
75            jnz      SWD20ms                            ; Delay over?
76            bit.b    #00000010b,&P1IN                  ; Verify S2 is still pressed
77            jnz      LExit                              ; If not, wait for S2 press
78 LEDon:      bis.b    #001h,&P1OUT                      ; Turn on LED1
79 SW2wait:    bit.b    #002h,&P1IN                      ; Test S2
80            jz       SW2wait                            ; Wait until S2 is released
81            bic.b    #001,&P1OUT                        ; Turn off LED1
82 LExit:      reti                                       ; Return from interrupt
83 ;-----
84 ; Stack Pointer definition
85 ;-----
86            .global  __STACK_END
87            .sect    .stack
88
89 ;-----
90 ; Interrupt Vectors
91 ;-----
92            .sect    ".reset"                          ; MSP430 RESET Vector
93            .short    RESET
94            .sect    ".int47"                          ; PORT1_VECTOR,
95            .short    S2_ISR                            ; please check the MSP430F5529.h header file
96            .end
97

```

Figure 3. Press/Release Using Port 1 ISR (Lab6_D3.asm)

2 Interfacing Switches and LEDs Using Interrupts in C

Figure 4 shows a C program that turns LED1 on when S2 is pressed and turns LED1 off when S2 is released. The main program configures and initializes ports, configures interrupts, and enters an infinite loop where the program waits for S2 to be released to turn off LED1. P1_ISR is entered upon detection of the switch press; the code inside clears P1.IFG1 and turns on LED1. Please note C convention to indicate that Port1_ISR corresponds to PORT1_VECTOR in the interrupt vector table.

```

1  /*****
2  *   File:      Lab6_D4.c
3  *   Description: The program detects when S2 is pressed and turns on LED1.
4  *               LED1 is kept on as long as S2 is pressed.
5  *               P1_ISR is used to detect when S2 is pressed.
6  *               Main program polls S2 and turns off when a release is detected.
7  *   Board:     MSP-EXP430F5529LP Launchpad
8  *   Clocks:    ACLK = 32.768kHz, MCLK = SMCLK = default DCO
9  *
10 *               MSP430F5529

```



```

11  *          +-----+
12  *          |               |
13  *          |               |
14  *          |               |
15  *          |               |
16  *          |               | P1.0|--> LED1
17  *          |               | P1.1|<-- S2
18  *
19  *   Author:   Aleksandar Milenkovic, milenkovic@computer.org
20  *   Date:     September 2010
21  *   Modified: Prawar Poudel, August 08, 2019
22  *****/
23  #include <msp430.h>
24  #define S2 BIT1&P1IN           // S2 is P1IN&BIT1
25
26  void main(void) {
27      WDTCTL = WDTPW+WDTHOLD;    // Stop WDT
28
29      P1DIR |= BIT0;             // Set LED1 as output
30      P1OUT = 0x00;             // Clear LED1
31
32      P1DIR &= ~BIT1;           // Set the direction at S2 as input
33      P1REN |= BIT1;            // Enable Pull-up resistor
34      P1OUT |= BIT1;            // Required for proper IO
35
36      _EINT();                  // Enable interrupts
37
38      P1IE |= BIT1;             // P1.1 interrupt enabled
39      P1IES |= BIT1;            // P1.1 hi/low edge
40      P1IFG &= ~BIT1;          // P1.1 IFG cleared
41
42      for(;;) {
43          while((S2) == 0);      // Wait until S2 is released
44          P1OUT &= ~BIT0;        // LED1 is turned off
45      }
46  }
47
48  // Port 1 interrupt service routine
49  #pragma vector = PORT1_VECTOR
50  __interrupt void Port1_ISR (void) {
51      P1OUT |= BIT0;             // LED1 is turned ON
52      P1IFG &= ~BIT1;          // P1.0 IFG cleared
53  }
54

```

Figure 4. Press/Release Using Port 1 ISR (Lab6_D4.c)

Looking at the program in Figure 4 we can see that release is detected in the main program. A better implementation would delegate both press and release activities into the P1 ISR as shown in Figure 5. To implement this, we need to establish a global variable called `S2pressed` that keeps the current state of the switch (0 – released, 1 – pressed). At the beginning we expect a press event, so Port 1 is configured to wait for a falling edge on `P1IN.BIT1` (SW2 is pressed). In that case, the ISR turns on LED1, sets the `S2pressed` and configures `P1IES` to trigger

an interrupt when a rising edge is detected on P1IN.BIT1. When the switch is pressed and we the ISR is entered, the steps are taken to turn LED1 off and configure P1IES so that a new press event can be detected. This way, all work is done inside the P1 ISR and main program can put the processor into sleep state.

```

1  /*****
2  *   File:      Lab6_D5.c
3  *   Description: The program detects when S2 is pressed and turns on LED1.
4  *               LED1 is kept on as long as S2 is pressed.
5  *               P1_ISR is used to detect both S2 presses and releases.
6  *   Board:     EXP430F5529LP Launchpad
7  *   Clocks:     ACLK = 32.768kHz, MCLK = SMCLK = default DCO
8  *
9  *               MSP430F5529
10 *
11 *               +-----+
12 *               |               |
13 *               |               |
14 *               |               |
15 *               |               | P1.0 | --> LED1
16 *               |               | P1.1 | <-- S2
17 *               |               |
18 *   Author: Aleksandar Milenkovic, milenkovic@computer.org
19 *   Date:  September 2010
20 *****/
21 #include <msp430.h>
22
23 unsigned char S2pressed = 0;    // S2 status (0 not pressed, 1 pressed)
24
25 void main(void) {
26     WDTCTL = WDTPW+WDTHOLD;    // Stop WDT
27     P1DIR |= BIT0;             // Set LED1 as output
28     P1OUT = 0x00;             // Clear LED1 status
29
30     S2pressed = 0;
31
32     P1DIR &= ~BIT1;            // Set the direction at S2 as input
33     P1REN |= BIT1;            // Enable Pull-up resistor
34     P1OUT |= BIT1;            // Required for proper IO
35
36     _EINT();                  // Enable interrupts
37     P1IE |= BIT1;             // P1IE.BIT1 interrupt enabled
38     P1IES |= BIT1;            // P1IES.BIT1 hi/low edge
39     P1IFG &= ~BIT1;           // P1IFG.BIT1 is cleared
40
41     _BIS_SR(LPM0_bits + GIE); // Enter LPM0(CPU is off); Enable interrupts
42 }
43
44 // Port 2 interrupt service routine
45 #pragma vector = PORT1_VECTOR
46 __interrupt void Port1_ISR (void) {
47     if (S2pressed == 0) {
48         S2pressed = 1;
49         P1OUT |= BIT0;        // LED1 is turned ON

```

```

50     P1IFG &= ~BIT1;           // P1IFG.BIT0 is cleared
51     P1IES &= ~BIT1;           // P1IES.BIT0 low/high edge
52 } else if (S2pressed == 1) {
53     S2pressed = 0;
54     P1OUT &= ~BIT0;           // LED1 is turned ON
55     P1IFG &= ~BIT1;           // P1IFG.BIT0 is cleared
56     P1IES |= BIT1;            // P1IES.BIT0 hi/low edge
57 }
58 }
59

```

Figure 5. Press/release Using Port 1 ISR – An Improved Implementation (Lab6_D5.c)

3 Clock Module

In the previous examples we have learned how to write a program that toggles the LEDs connected to the MSP430's output ports. We have also learned how write code to generate software delays. In our example, we assumed that the processor clock is around 1 μ s (i.e., the clock frequency is approximately 1 MHz). The MSP430 family supports several clock modules and a user has a full control over these modules. By changing the content of relevant clock module control registers, one can change the processor clock frequency, as well as the frequency of other clock signals that are used for peripheral devices. In the next section, we will discuss the organization of the Unified Clock System (UCS) used in the MSP430F5529 device.

3.1 Unified Clock System (UCS)

MSP430x5xx family of microcontrollers have Unified Clock System (UCS) that provides various clocks for the MSP430 modules. The UCS module includes up to five clock sources and provide three clock signals.

The five clock sources included in the UCS module are as follows:

- **XT1CLK:** Low-frequency or high-frequency oscillator that can be used either with low-frequency 32768-Hz watch crystals, standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT1CLK can be used as a clock reference into the FLL. Some devices only support the low frequency oscillator for XT1CLK.
- **VLOCLK:** Internal very low power, low-frequency oscillator with 10-kHz typical frequency.
- **REFOCLK:** Internal trimmed low-frequency oscillator with 32768-Hz typical frequency, can be used as a clock reference into the FLL.
- **DCOCLK:** Internal digitally controlled oscillator (DCO) that can be stabilized by the FLL.
- **XT2CLK:** Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT2CLK can be used as a clock reference into the FLL.

- **MCLK:** Master clock. MCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. MCLK can be divided by 1, 2, 4, 8, 16, or 32. MCLK is used by the CPU and system.
- **SMCLK:** Subsystem master clock. SMCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. SMCLK can be divided by 1, 2, 4, 8, 16, or 32. SMCLK is software selectable by individual peripheral modules.

On a PUC, the configuration of UCS is as follows:

- XT1 in LF mode is selected as source for XT1CLK which is selected for ACLK.
- DCOCLKDIV is selected for MCLK and SMCLK.
- FLL operation is enabled and XT1CLK is selected as reference clock for FLL.
- If the 32768 Hz crystal is used for XT1CLK, fault control logic sources ACLK with 32768 Hz REFOCLK because XT1 takes some time to stabilize.
- When the crystal start-up is obtained, FLL stabilizes MCLK and SMCLK at 1048576 Hz (2^{20} Hz). $f_{DCO} = 2097152$ Hz

The operating modes of UCS is controlled using the following registers: **SCG0**, **SCG1**, **OSCOFF** and **CPUOFF**.

The UCS module can be configured using registers from **UCSCTL0** through **UCSCTL8**.

Digital Controlled Oscillator (DCO):

- The frequency of DCO can be adjusted by software using DCORSEL (in UCSCTL1 register), DCO and MOD bits (in UCSCTL0).
- DCO can also be stabilized using FLL to a multiple of FLL reference clock (i.e. multiple frequency of $FLLREFCLK/n$). FLL can accept different reference sources selectable by SELREF bits (UCSCTL3). The reference sources can be XT1CLK, REFOCLK or XT2CLK if available.
- The value of n can be defined in FLLREFDIV bits in UCSCTL3 register ($n = 1, 2, 4, 8, 12$ or 16). Default value is $n=1$.
- FLLD bits (in UCSCTL2 register) can be configured for FLL pre-scalar divider value D of 1, 2, 4, 8, 16 or 32. The default of $D = 2$ and MCLK and SMCLK are sourced from DCOCLKDIV thus providing clock frequency of $DCOCLK/2$.
- The divider $(N+1)$ and divider D define DCOCLK and DCOCLKDIV frequencies. Value $N+1$ can be set from FLLN bits (in UCSCTL2 register) where $N > 0$. Setting FLLN to 0 will cause FLLN to be 1 to prevent unintentional write.
- The final frequency of DCOCLK and DCOCLKDIV are as follows

$$f_{DCOCLK} = D \times (N + 1) \times (f_{FLLREFCLK} \div n)$$

$$f_{DCOCLKDIV} = (N + 1) \times (f_{FLLREFCLK} \div n)$$

Figure 7 Formula to compute DCOCLK and DCOCLKDIV frequencies

Frequency Locked Loop (FLL):

- The FLL continuously counts up or down a frequency integrator.
- The count is adjusted +1 with the frequency $f_{FLLREFCLK}/n$ ($n=1,2,4,8,12$ Or 16) or -1 with the frequency $f_{DCOCLK}/[D*(N+1)]$
- Five of the integrator bits (UCSCTL0bits 12 to 8) set the DCO frequency tap. Thirty-two taps are implemented for the DCO, and each is approximately 8% higher than the previous. The modulator mixes two adjacent DCO frequencies to produce fractional taps. For a given DCO bias range setting, time must be allowed for DCO to settle on the proper tap operation. $(n*32)f_{FLLREFCLK}$ cycles are required between taps requiring worst case of $(n*32*32)f_{FLLREFCLK}$ cycles for DCO to settle.

3.2 Changing Processor Clocks: Examples

The following examples illustrate (Figure 8 and Figure 9) how you can change the processor clock frequency by modifying individual bits in the control registers. Please note that these examples only change the clocks and make them visible on external ports (some digital I/O ports have a special function to pass the clocks to the output, so we can observe them from the outside by connecting to oscilloscope). For learning how internal digitally controlled oscillator works read the corresponding user manual.

```
1  /*****
2  *   File:      Lab6_D6.c
3  *   Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 2.45MHz
4  *   This program demonstrates setting the internal DCO to run at
5  *   2.45MHz.
6  *   Clocks:    ACLK = 32768Hz,
7  *              MCLK = SMCLK = DCO = (74+1) x ACLK = 2457600Hz
8  *
9  *              MSP430F5529
10 *
11 *   /|\|      XIN| -
12 *   |  |      |   | 32kHz
13 *   --| RST   XOUT| -
14 *
15 *           |   P7.7 --> MCLK = 2.45MHz
16 *
17 *           |   P2.2 --> SMCLK = 2.45MHz
18 *           |   P1.0 --> ACLK = 32kHz
19 *
20 *
21 *   Author:     Aleksandar Milenkovic, milenkovic@computer.org
22 *   Date:       September 2010
23 *   Modified:    Prawar Poudel, August 2020
24 *****/
25 #include <msp430.h>
26
27 void main(void)
28 {
29     WDTCTL = WDTPW + WDTHOLD;    // Stop watchdog timer
30     P1DIR |= BIT0;               // ACLK set out to pins
```



```

31 P1SEL |= BIT0;
32 P2DIR |= BIT2; // SMCLK set out to pins
33 P2SEL |= BIT2;
34 P7DIR |= BIT7; // MCLK set out to pins
35 P7SEL |= BIT7;
36
37 UCSCTL3 = SELREF_2; // Set DCO FLL reference = REFO
38 UCSCTL4 |= SELA_2; // Set ACLK = REFO
39 UCSCTL0 = 0x0000; // Set lowest possible DCOx, MODx
40
41 // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
42 do
43 {
44     UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
45                                     // Clear XT2,XT1,DCO fault flags
46     SFRIFG1 &= ~OIFG; // Clear fault flags
47 } while (SFRIFG1&OIFG); // Test oscillator fault flag
48
49 __bis_SR_register(SCG0); // Disable the FLL control loop
50 UCSCTL1 = DCORSEL_5; // Select DCO range 16MHz operation
51 UCSCTL2 |= 74; // Set DCO Multiplier for 2.45MHz
52 // (N + 1) * FLLRef = Fdco
53 // (74 + 1) * 32768 = 2.45MHz
54 __bic_SR_register(SCG0); // Enable the FLL control loop
55
56 // Worst-case settling time for the DCO when the DCO range bits have been
57 // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
58 // UG for optimization.
59 // 32 x 32 x 2.45 MHz / 32,768 Hz = 76600 = MCLK cycles for DCO to settle
60 __delay_cycles(76000);
61
62 while(1);
63 }
64

```

Figure 8. Changing DCO to Run at 2.45 MHz using UCS Module (Lab6_D6.c)

```

1  /*****
2  *   File:      Lab6_D7.c
3  *   Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 8MHz
4  *               This program demonstrates setting the internal DCO to run at
5  *               8MHz.
6  *   Clocks:    ACLK = 32768Hz,
7  *               MCLK = SMCLK = DCO = (121+1) x 2 x ACLK = 7995392Hz
8  *
9  *               MSP430F5529
10 *
11 *   /\|----- XIN| -
12 *   |  |         | 32kHz
13 *   --| RST      XOUT| -
14 *
15 *               P7.7| --> MCLK = 8MHz
16 *
17 *               P2.2| --> SMCLK = 8MHz

```

```

18 *           |           P1.0|--> ACLK = 32kHz
19 *           |           |
20 *
21 * Author:   Aleksandar Milenkovic, milenkovic@computer.org
22 * Date:     September 2010
23 * Modified: Prawar Poudel
24 * Date:     August 2020
25 *****/
26
27 #include <msp430.h>
28
29 void main(void)
30 {
31     WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer
32
33     P1DIR |= BIT0;                      // ACLK set out to pins
34     P1SEL |= BIT0;
35     P2DIR |= BIT2;                      // SMCLK set out to pins
36     P2SEL |= BIT2;
37     P7DIR |= BIT7;                      // MCLK set out to pins
38     P7SEL |= BIT7;
39
40     UCSCTL3 = SELREF_2;                 // Set DCO FLL reference = REFO
41     UCSCTL4 |= SELA_2;                 // Set ACLK = REFO
42     UCSCTL0 = 0x0000;                 // Set lowest possible DCOx, MODx
43
44     // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
45     do
46     {
47         UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
48                                     // Clear XT2,XT1,DCO fault flags
49         SFRIFG1 &= ~OIFIFG;         // Clear fault flags
50     } while (SFRIFG1&OIFIFG);       // Test oscillator fault flag
51
52     __bis_SR_register(SCG0);         // Disable the FLL control loop
53     UCSCTL1 = DCORSEL_5;             // Select DCO range 16MHz operation
54     UCSCTL2 |= 249;                 // Set DCO Multiplier for 8MHz
55                                     // (N + 1) * FLLRef = Fdco
56                                     // (249 + 1) * 32768 = 8MHz
57     __bic_SR_register(SCG0);         // Enable the FLL control loop
58
59     // Worst-case settling time for the DCO when the DCO range bits have been
60     // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
61     // UG for optimization.
62     // 32 x 32 x 8 MHz / 32,768 Hz = 250000 = MCLK cycles for DCO to settle
63     __delay_cycles(250000);
64     while(1);                       // Loop in place
65 }
66

```

Figure 9. Changing DCO to Run at 8 MHz using UCS Module (Lab6_D7.c)

```

1  /*****
2  *   File:      Lab6_D8.c
3  *   Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 1MHz
4  *               This program demonstrates setting the internal DCO to run at
5  *               8MHz when SW2 is pressed.
6  *
7  *               A LED will keep blinking at 1Hz (0.5s ON and 0.5s OFF) at
8  *               clock running at default frequency of 1Mhz. When SW2 is pressed,
9  *               the frequency is changed to ~ 8 Mhz. When SW2 is pressed again,
10 *               the frequency is restored to 1Mhz.
11 *
12 *   Clocks:     ACLK = 32768Hz,
13 *               MCLK = SMCLK = DCO = (249+1) x ACLK = 8192000Hz
14 *
15 *   Input:      SW1 (P2.1)
16 *   Output:     LED2 (P4.7)
17 *
18 *               MSP430F5529
19 *
20 *               /|\|
21 *               |   |
22 *               --RST
23 *
24 *   SW2-->|P2.1
25 *   LED2<--|P4.7
26 *
27 *               P7.7 --> MCLK = 1 or 8MHz
28 *               P2.2 --> SMCLK = 1 or 8MHz
29 *               P1.0 --> ACLK = 32kHz
30 *
31 *   Modified:    Prawar Poudel
32 *   Date:        August 2020
33 *****/
34 // mandatory include statement
35 #include <msp430.h>
36
37 // this function configures the clock sources as follows
38 // .. use internal REFOCLK for FLL reference clock (UCSCTL3 = SELREF_2)
39 // .. ACLK is sourced with REFOCLK (UCSCTL4 |= SELA_2)
40 // .. sets DCO tap to 0 (UCSCTL0 = 0)
41 // .. sets the modulation bit counter value to 0 (UCSCTL0 = 0)
42 void configure_clock_sources();
43 // this function changes the frequency of clock to 8 MHZ
44 inline void change_clock_freq_8Mhz();
45 // this function changes the frequency of clock to 1 MHZ
46 inline void change_clock_freq_1Mhz();
47
48 char is8Mhz = 0;
49
50 void main(void)
51 {
52     WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer
53
54     P1DIR |= BIT0;                      // ACLK set out to pins

```

```

56     P1SEL |= BIT0;
57
58     P2DIR |= BIT2;           // SMCLK set out to pins
59     P2SEL |= BIT2;
60
61     P7DIR |= BIT7;           // MCLK set out to pins
62     P7SEL |= BIT7;
63
64     _EINT();                  // enable interrupts
65     P2DIR &= ~BIT1;          // set P2.1 as input (SW2)
66     P2REN |= BIT1;           // enable pull-up resistor
67     P2IE  |= BIT1;           // enable interrupt at P2.1
68     P2IES |= BIT1;           // enable hi->lo edge for interrupt
69     P2IFG &= ~BIT1;          // clear any errornous interrupt flag
70
71     P4DIR |= BIT7;           // set P4.7 as output (LED2)
72
73     configure_clock_sources(); // configure the clock sources
74
75     while(1)                  // Loop in place (infinite)
76     {
77         P4OUT ^= BIT7;        // toggle LED2
78         delay_cycles(500000); // arbitrary delay of 500ms
79     }
80 }
81
82 // this ISR handles the SW2 key press
83 #pragma vector = PORT2_VECTOR
84 __interrupt void PORT2_ISR(void)
85 {
86     // let us clear the flag
87     P2IFG &= ~BIT1;
88
89     //debouncing section
90     delay_cycles(25000);
91
92     // if SW1 is not pressed, return
93     if((P2IN&BIT1)!=0x00)
94         return;
95
96
97     if(is8Mhz==0)
98     {
99         // if not at 8Mhz, let us change to 8Mhz
100         change_clock_freq_8Mhz();
101         is8Mhz = 1;
102     }else
103     {
104         // if already in 8Mhz, let us take back to 1Mhz
105         change_clock_freq_1Mhz();
106         is8Mhz = 0;
107     }
108 }
109
110 // this function changes the frequency of clock to 8 MHZ

```

```

111 void change_clock_freq_8Mhz()
112 {
113     __bis_SR_register(SCG0);           // Disable the FLL control loop
114     UCSCTL1 = DCORSEL_5;               // Select DCO range 16MHz operation
115     UCSCTL2 = 249;                     // Set DCO Multiplier for 8MHz
116                                         // (N + 1) * FLLRef = Fdco
117                                         // (249 + 1) * 32768 = 8MHz
118     __bic_SR_register(SCG0);           // Enable the FLL control loop
119
120     // Worst-case settling time for the DCO when the DCO range bits have been
121     // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
122     // UG for optimization.
123     // 32 x 32 x 8 MHz / 32,768 Hz = 250000 = MCLK cycles for DCO to settle
124     __delay_cycles(250000);
125 }
126
127 // this function changes the frequency of clock to 1 MHz
128 void change_clock_freq_1Mhz()
129 {
130     __bis_SR_register(SCG0);           // Disable the FLL control loop
131     UCSCTL1 = DCORSEL_3;               // Select DCO range 1MHz operation
132     UCSCTL2 = 32;                      // Set DCO Multiplier for 1MHz
133                                         // (N + 1) * FLLRef = Fdco
134                                         // (32 + 1) * 32768 = 1MHz
135     __bic_SR_register(SCG0);           // Enable the FLL control loop
136
137     // Worst-case settling time for the DCO when the DCO range bits have been
138     // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
139     // UG for optimization.
140     // 32 x 32 x 1 MHz / 32,768 Hz = 33792 = MCLK cycles for DCO to settle
141     __delay_cycles(33792);
142 }
143
144 // this function configures the clock sources as follows
145 // .. use internal REFOCLK for FLL reference clock (UCSCTL3 = SELREF_2)
146 // .. ACLK is sourced with REFOCLK (UCSCTL4 |= SELA_2)
147 // .. sets DCO tap to 0 (UCSCTL0 = 0)
148 // .. sets the modulation bit counter value to 0 (UCSCTL0 = 0)
149 void configure_clock_sources()
150 {
151     UCSCTL3 = SELREF_2;                 // Set DCO FLL reference = REFO
152     UCSCTL4 |= SELA_2;                  // Set ACLK = REFO
153     UCSCTL0 = 0x0000;                   // Set lowest possible DCOx, MODx
154
155     // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
156     do
157     {
158         UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG); // Clear XT2,XT1,DCO fault flags
159         SFRIFG1 &= ~OIFIFG;                // Clear fault flags
160     } while (SFRIFG1&OIFIFG);             // Test oscillator fault flag
161 }
162

```

Figure 10 Changing Clock Frequency to Observe Change in LED Blinking

4 References

It is crucial that you become familiar with the basics of how digital ports work – how to set their output direction, read from or write to the ports, set interrupts, and set up their special functions. We will be using these features to control hardware and communication between devices throughout this class. Please reference the following material to gain more insight on the device:

- The MSP-EXP430F5529LP board's user guide
- Chapter 5 in the MSP430F5529 user's guide (pages 158-185)
- Chapter 7 in the John H. Davies' *MSP430 Microcontroller Basics*

