

Attached is the link for the [Simulation Files](#) with Proteus 8 Professional.

Code

```
ORG 000H //initial starting address

MOV P0, #07H    //Initializing the buttons DEC, INC, SUBMIT
MOV R1, #03H    // Setting the initial period with 30 seconds
MOV P1, #3FH    // Setting the LSB of display as 0
MOV P3, #4FH    // Setting the MSB of display as 3
MOV DPTR, #NUM  // Locate the first address of the 7-Segment LOOK-UP Table in
DPTR
MOV R5, #01H    // Set the initial step time to 1 second

READSW: MOV A, P0    // Get the P0 data to the accumulator
        RRC A        // Rotate the accumulator to the right to get the LSB to
the Carry
        JNC DECREMENT // If Carry is LOW (Button-DEC Pressed), Jump to
DECREMENT
        RRC A        // Rotate one more time to check the second bit (second
button - INC)
        JNC INCREMENT // If the second bit is LOW (Button - INC Pressed), jump
to INCREMENT
        RRC A        // Rotate one more time to check the third button - OK
        JNC OK        // If the third bit translated to the Carry is LOW, jump
to OK Subroutine
        SJMP READSW  // Read switch status again.

// DECREMENT decrements the period by a 10: 40 --> 30 --> 20
DECREMENT:
        MOV R4, A
        MOV A, R1
        SUBB A, #01H // Check that A (R1) isn't 10 so that no decrement is
required if so
        JZ RETURN1   // If no decrement will occur, return to checking
buttons readings
        DEC R1
        ACALL SHOWN  // Show the selected period on the 7-Segments
        ACALL DELAY1S // Delay for 1 second to avoid any bouncing
        SJMP READSW  // Return after decrementing to checking buttons
readings

// Return to checking in case of no decrement occurs
RETURN1: MOV A, R4
        SJMP READSW

// INCREMENT increments the period by 10: 30 -> 40 -> 50
INCREMENT:
        MOV R4, A
        MOV A, R1
        CLR C
        SUBB A, #09  // Checking that R1 isn't 90. if so, no increment is
required
        JZ RETURN2   // If so, return to checking the buttons without
incrementing
        INC R1
        ACALL SHOWP  // Show the incremented and new period on the 7-segments
```

```

        ACALL DELAY1S    // Delay 1 second to avoid any bouncing
        SJMP READSW

// In case of no increment occurs, return to checking buttons readings
RETURN2:
        MOV A, R4
        SJMP READSW

// OK gets the application to the "Step Time Select" stage
OK:
        ACALL DELAY1S
        SJMP SETDELAY

// Showing the new values on the segments in case of decrement SHOWN or
increment SHOWP
SHOWN:
        MOV A, #09H
        SUBB A, R1
        MOVC A, @A+DPTR
        MOV P3, A
        RET

SHOWP:
        MOV A, #0AH
        SUBB A, R1
        MOVC A, @A+DPTR
        MOV P3, A
        RET

// "Step Time Select"
// In this block, the user determine the time between each 2 counts, may be set
to 2 or 3 seconds between each 2 values
// Default value is #01H in R5

SETDELAY:
        // Set the segment to show the initial value (1 second)
        MOV P1, #06H
        MOV P3, #3FH
READSWDELAY: MOV A,P0    // Moving the port value to Accumulator.
        RRC A          // Rotating the accumulator to check the value of the
button bits (PULL-DOWN)
        JNC DECREMENTDELAY
        RRC A
        JNC INCREMENTDELAY
        RRC A
        JNC OKDELAY
        SJMP READSWDELAY    // Read switch status again.

// Decrement the step time by 1 ( 3 -> 2 -> 1)
DECREMENTDELAY:
        MOV R4, A
        MOV A, R5
        SUBB A, #01H

```

```

        JZ RETURN1DELAY
        DEC R5
        ACALL SHOWNDELAY
        ACALL DELAY1S
        SJMP READSWDELAY

// Return to checking in case of no decrement occurs
RETURN1DELAY: MOV A, R4
              SJMP READSWDELAY

// Increment the step time by 1 ( 3 -> 4 -> 5)
INCREMENTDELAY:
            MOV R4, A
            MOV A, R5
            CLR C
            SUBB A, #09
            JZ RETURN2DELAY
            INC R5
            ACALL SHOWPDELAY
            ACALL DELAY1S
            SJMP READSWDELAY

// If case of no increment, return to checking buttons readings directly
RETURN2DELAY:
            MOV A, R4
            SJMP READSWDELAY

// The submit that gets the application to the MAIN loop
OKDELAY:

        SJMP MAIN

// Show the new step time in the segments after decrement (SHOWNDELAY) and
// increment (SHOWPDEALY)
SHOWNDELAY:
        MOV A, #09H
        SUBB A, R5
        MOVC A, @A+DPTR
        MOV P1, A
        RET

SHOWPDELAY:
        MOV A, #0AH
        SUBB A, R5
        MOVC A, @A+DPTR
        MOV P1, A
        RET

// MAIN FUNCTION
// Here is our main function where the countdown takes place with repition and
// toggling LEDs after each time
MAIN:    SETB P2.0      // RED LED ON
        CLR P2.1      // GREEN LED OFF

```

```

MOV DPTR, #NUM // First address of 7-segment LOOK-UP Table is written
in DPTR
MOV A, #09H
SUBB A, R1 // Getting the steps at which the countdown will start
MOV R1, A
MOV A, R1
MOV R2, A
MOVC A, @A+DPTR // Puts the code of starting point in accumulator
MOV P3, A // Puts the code in accumulator in the P3 for the MSB 7-
Segment
MOV A, R1
ACALL DISPLAYMAX // Displays the max number (starting count)
ACALL START // Starts the decrement and the countdown

START:
MOV A, #00H // initial value of accumulator
MOV B, A
MOV A, B
MOV R0, #0AH //Register R0 initialized as counter which counts from 10 to
0

LABEL: ACALL DISPLAYLEAST // Controls the countdown of the LSB 7-Segment
ACALL DELAYTIME // calls the delay of the timer (edited by the step
time selected by user)
DEC R0 //Counter R0 decremented by 1
MOV A, R0 // R0 moved to accumulator to check if it is zero in
next instruction.
JZ DISPLAYMOST //Checks accumulator for zero and jumps to START.
Done to check if counting has been finished.
MOV A, B
SJMP LABEL //LOOPING

NUM: DB 6FH // digit drive pattern for 9
DB 7FH // digit drive pattern for 8
DB 07H // digit drive pattern for 7
DB 7DH // digit drive pattern for 6
DB 6DH // digit drive pattern for 5
DB 66H // digit drive pattern for 4
DB 4FH // digit drive pattern for 3
DB 5BH // digit drive pattern for 2
DB 06H // digit drive pattern for 1
DB 3FH // digit drive pattern for 0

DELAY1S: // Dealy for 1 second ( 250 ms * 4 times)
MOV R7, #250D

LOOP1:
ACALL DELAY1M
DJNZ R7, LOOP1
MOV R7, #250D

```

```

LOOP2:
    ACALL DELAY1M
    DJNZ R7, LOOP2
    MOV R7, #250D

LOOP3:
    ACALL DELAY1M
    DJNZ R7, LOOP3
    MOV R7, #250D

LOOP4:
    ACALL DELAY1M
    DJNZ R7, LOOP4
    RET

// Delay 1 ms using Timer/Counter 0
DELAY1M:    MOV TMOD, #01H
            MOV TH0, #00FCH
            MOV TL0, #0018H
            SETB TR0
WAIT:       JNB TF0, WAIT
            CLR TR0
            CLR TF0
            RET

// Complement the O/P of the LEDS (RED: ON -> OFF -> ON), Same for GREEN
CMPLEDS:    CPL P2.0
            CPL P2.1
            RET

// DISPLAYLEAST gets the code of the digit in turn and display it on the LSB 7-
Segment
DISPLAYLEAST:  MOVC A,@A+DPTR // adds the byte in A to the program counters
address
            MOV P1,A
            MOV A,B
            INC A
            MOV B,A
            RET

// DISPLAYMOST gets the code of the digit in turn and display it on the MSB 7-
Segment
DISPLAYMOST:   MOV B, A
            INC R1
            MOV A, R1
            MOVC A, @A+DPTR
            MOV P3, A
            MOV A, R1
            SUBB A, #09H
            JZ RESET
            MOV A, B
            JMP START

// When reaching 00 Reset the Countdown, apply the CMPLEDS and DISPLAYMAX
RESET:  MOV A, R2
        MOV R1, A
        ACALL CMPLEDS

```

```

        ACALL DISPLAYMAX
        RET

DISPLAYMAX: MOV R3, A
            MOVC A, @A+DPTR
            MOV P3, A
            MOV A, #09H
            MOVC A,@A+DPTR // adds the byte in A to the program counters address
            MOV P1, A
            ACALL DELAYTIME
            MOV A, R3
            SUBB A, #08H
            JNZ DISPLAYMOST
            JZ DISPLAYZ
            RET

// SPECIAL CASE TO SHOW ZERO AT THE BEGINNING OF COUNTDOWN (USED AT DISPLAYMAX)

DISPLAYZ:  MOV P3, #3FH
            RET

// Depending on the value of R5 set by the use at "Step Time Select" stage,
// The DELAYTIME loops around the value of R5, delauing 1 seconde each iteraion
// The loop will stop when R5 becomes 0
DELAYTIME:
            MOV A, R5
DELAYDEC:
            ACALL DELAY1S
            DEC A
            JNZ DELAYDEC
            RET

END

```