Code to increment the registers 1, 2, and 3. Registers 1, and 2 will increment up too 60, then roll back to 0, register 3 will increment up to 24.

INCT: MOV A,R1 //mover register 1 (seconds) to the ACC

ADD A, #1 //increment acc

DA A //convert to bcd

MOV R1, A // move acc back to register 1

CJNE A, #60H, INCE //if not 60, return, else continue

MOV R1,#0 //reset R1 to 0

MOV A,R2 // mover register 2 (minutes) to the ACC

ADD A, #1 //increment acc

DA A //convert to bcd

MOV R2, A //move acc back to register 2

CJNE A, #60H, INCE //if not 60, return, else continue

MOV R2,#0 //reset R2 to 0

MOV A,R3 // mover register 3 (hours) to the ACC

ADD A, #1 //increment acc

DA A //convert to bcd

MOV R3, A //move acc back to register 3

CJNE A, #24H, INCE //if not 24, return

MOV R3,#0

Code to display each of the registers to the display.

DIST: MOV A,#01 //Update display routine

ACALL COMNWRT //Reset display

ACALL DELAY2

MOV A,R3 //move register 3 (hours)

ACALL SHIFT //jump to shift(rotates right 4 times)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A,R3 //move register 3 (hours)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A, #3AH //move the ASCI value of ‘:’ to the acc

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A,R2 //move register 2 (min)

ACALL SHIFT //jump to shift(rotates right 4 times)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A,R2 //move register 3 (hours)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A, #3AH //move the ASCI value of ‘:’ to the acc

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A,R1 //move register 1 (seconds)

ACALL SHIFT //jump to shift(rotates right 4 times)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

MOV A,R1 //move register 3 (hours)

ANL A, #0FH //logical AND to mask upper nibble

ORL A, #30H //logical OR to place ‘0011’ in upper nibble

ACALL DATAWRT //write to port

ACALL DELAY1 //delay

RET

Code to rotate the bits 4 times

SHIFT:

RR A

RR A

RR A

RR A

RET

**Additional tasks:**

1. To speed up the verification process that the clock is counting correctly, you can decrease the delays within the code, thereby making the timer count up faster. You could also manually set the initial values within the registers, thereby getting the timer to start at a set time.
2. the output of Port2-Bit-3 is not square because we are using a polling approach to check if the overflow on our timer has occurred. Because of this the processor is tied up both executing our code, and checking the state of the counter. This can be solved by using on of the onboard timers and an interrupt that when triggered increments the registers accordingly.
3. To get the clock to start at a different time, you can pre-set the values of R1, R2, and R3 to be whatever numbers you want. When this code is uploaded to the microprocessor the timer will begin at the time within these registers.