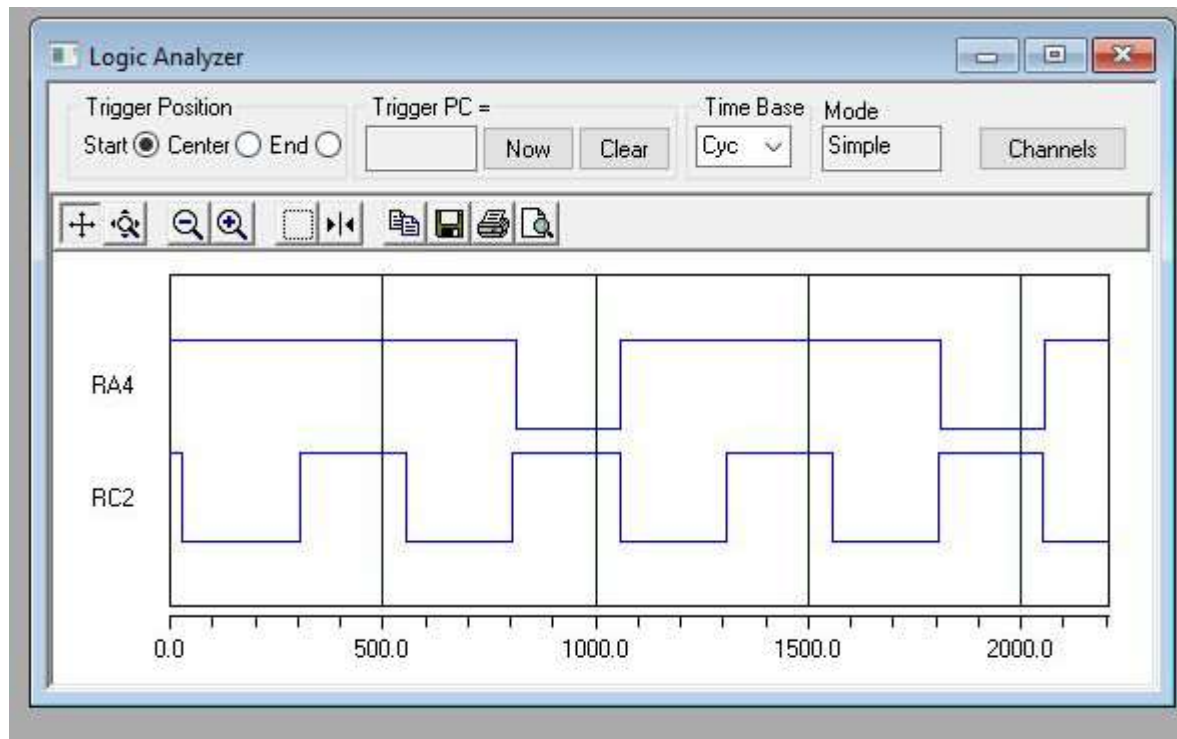


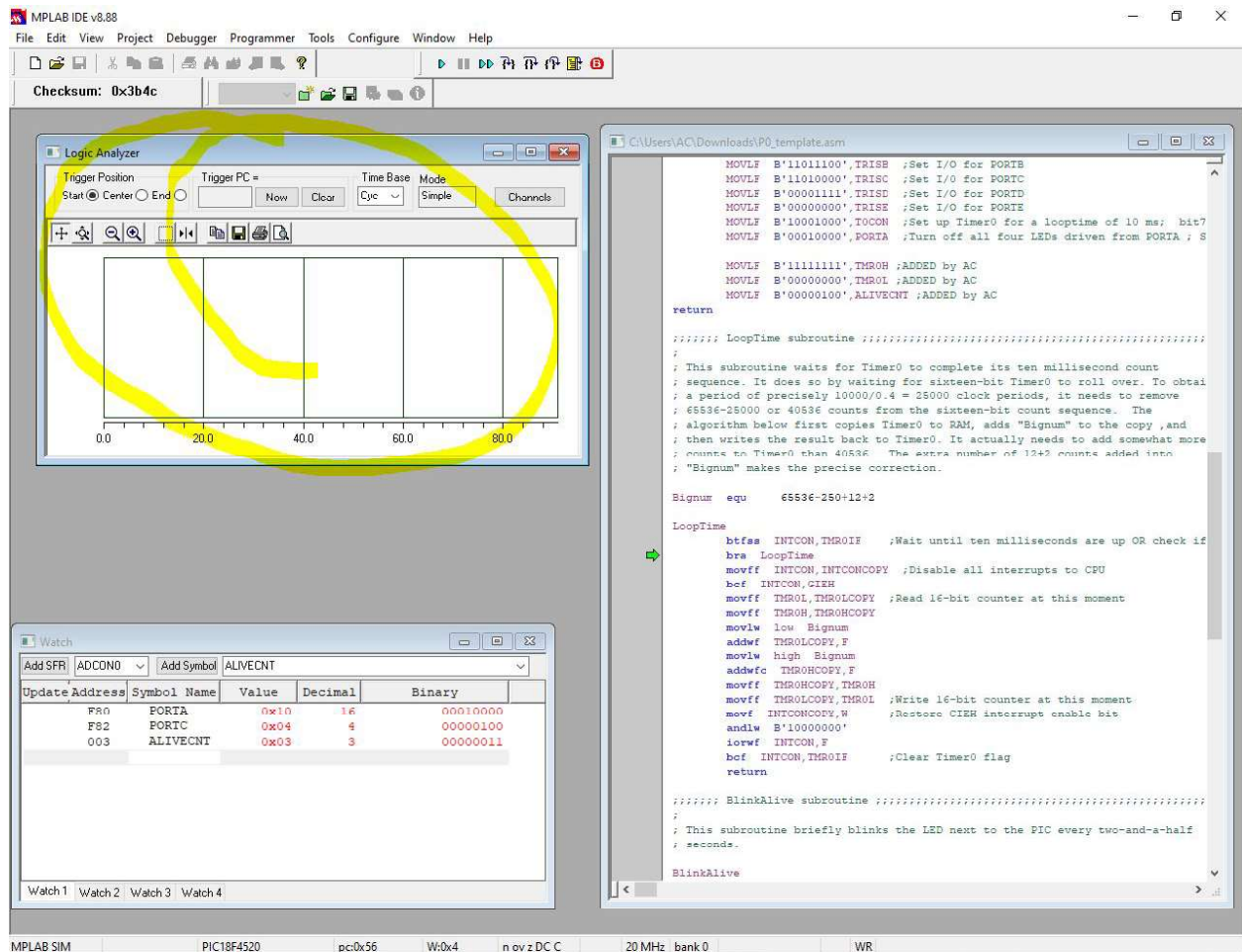
## MPLAB IDE Logic Analyzer

Once you have set up the IDE simulator, as shown above, now you can use the embedded logic analyzer to monitor how the states of particular bits in certain registers change with time. For example, with respect to “P0\_template.asm,” we can analyze how the states of bits **RA4** and **RC2** change as the program is simulated as shown below

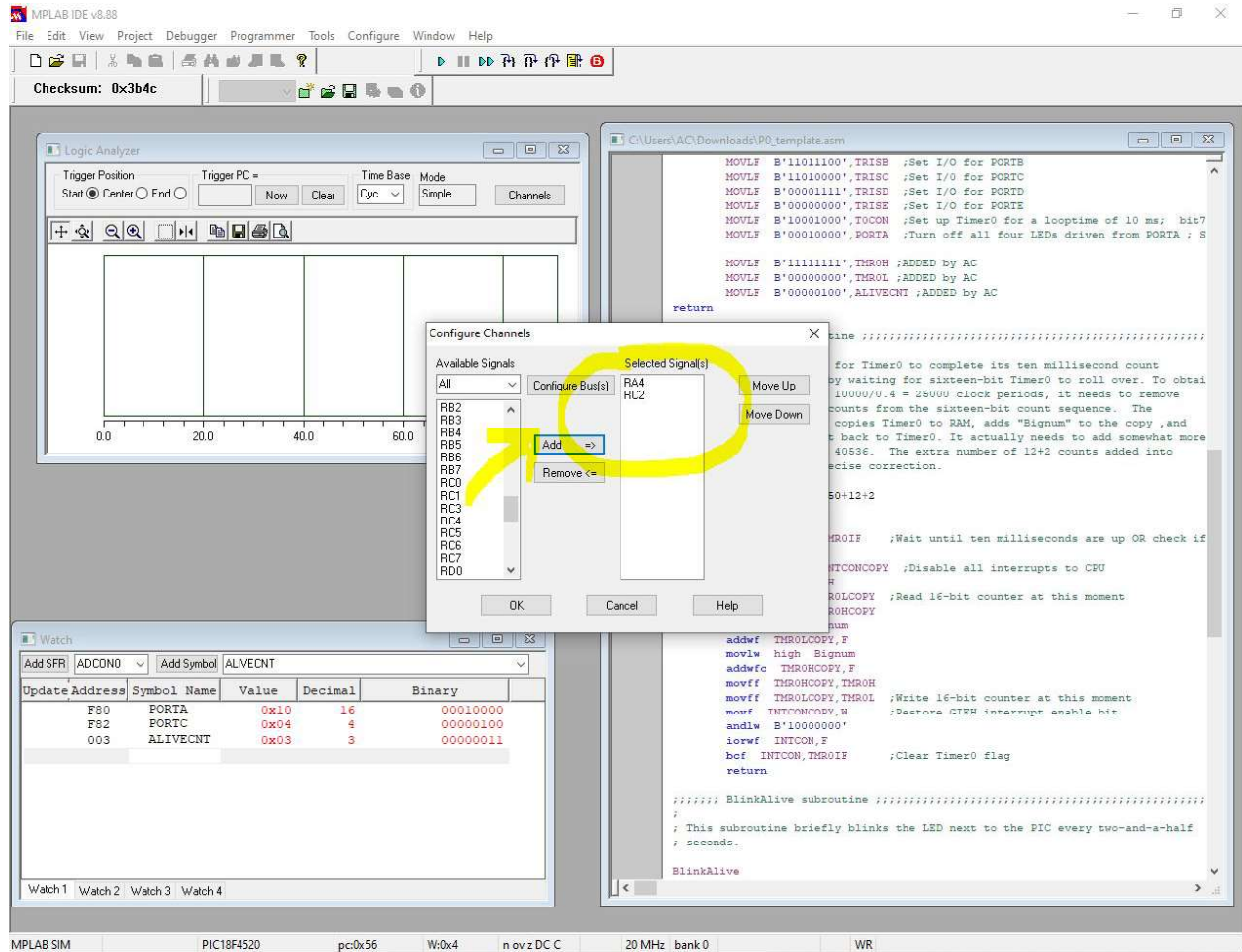


Please note that the following instructions assume that you have already set up the MPLAB SIM tool (as shown at the beginning of this tutorial) and that your code has been compiled.

1. Click on **View >> Simulator Logic Analyzer**. The *Logic Analyzer* window will launch and your IDE should look as follows:



- Click on the **Channels** button and the *Configure Channels* window will launch. From the *Available Signals* menu, first select the bit **RA4** and then click on the **Add** button. This will move the bit **RA4** to the *Selected Signal(s)* menu on the right. Repeat these last steps in order to add and select the bit RC2. After this, your window should look as follows.



3. Once you have added all the desired signals, click on the **OK** button. The selected signals should appear in the *Logic Analyzer* window as follows:

The screenshot displays the MPLAB IDE v6.88 interface. The **Logic Analyzer** window is open, showing a timing diagram for signals RA4 and RC2. The RA4 signal is high from approximately 10 to 30 units of time, and the RC2 signal is high from approximately 20 to 40 units. The Watch window shows the following data:

Update Address	Symbol Name	Value	Decimal	Binary
F80	PORTA	0x10	16	00010000
F82	PORTC	0x04	4	00000100
003	ALIVECNT	0x03	3	00000011

The main assembly code editor shows the following code:

```
MOVLF B'11011100',TRISB ;Set I/O for PORTB
MOVLF B'11010000',TRISC ;Set I/O for PORTC
MOVLF B'00001111',TRISE ;Set I/O for PORTD
MOVLF B'00000000',TRISE ;Set I/O for PORTE
MOVLF B'10001000',TOCON ;Set up Timer0 for a looptime of 10 ms; bit7
MOVLF B'00010000',PORTA ;Turn off all four LEDs driven from PORTA ; S

return

; LoopTime subroutine ;
; This subroutine waits for Timer0 to complete its ten millisecond count
; sequence. It does so by waiting for sixteen-bit Timer0 to roll over. To obtain
; a period of precisely 10000/0.4 = 25000 clock periods, it needs to remove
; 65536-25000 or 40536 counts from the sixteen-bit count sequence. The
; algorithm below first copies Timer0 to RAM, adds "Bignum" to the copy, and
; then writes the result back to Timer0. It actually needs to add somewhat more
; counts to Timer0 than 40536. The extra number of 12+2 counts added into
; "Bignum" makes the precise correction.

Bignum equ 65536-250+12+2

LoopTime
    btfsc INTCON,THROIF ;Wait until ten milliseconds are up OR check if
    bra LoopTime
    movff INTCON,INTCONCOPY ;Disable all interrupts to CPU
    hrf INTCON,CIR
    movff THRO,THROLCOPY ;Read 16-bit counter at this moment
    movff THRO,THROHCOPY
    movlw low Bignum
    addwf THROLCOPY,F
    movlw high Bignum
    addwfc THROHCOPY,F
    movff THROLCOPY,THROH ;Write 16-bit counter at this moment
    movff THROHCOPY,THROL
    movff INTCONCOPY,W ;Restore CIR interrupt enable bit
    andlw B'10000000'
    iorwf INTCON,F
    bcf INTCON,THROIF ;Clear Timer0 flag
    return

; BlinkAlive subroutine ;
; This subroutine briefly blinks the LED next to the PIC every two-and-a-half
; seconds.

BlinkAlive
```

4. Now you can *animate* the code and see the behaviour of the two bits **RA4** and **RC2** with respect to simulation time. Let the animation run for a while and see what happens on the *Logic Analyzer* window.
5. For your convenience, you may terminate the animation after a certain number of simulation cycles and you should see the periodic behaviors of these two bits. After a certain time, you Logic Analyzer window should look as follows.
  - a. Output a 75% duty cycle pulse train at bit **RA4**.
  - b. Output a 50% duty cycle pulse train at bit **RC2**.

