PIC18F452 Timer adjustment

Terry Sturtevant

Wilfrid Laurier University

February 10, 2012

Here's a code fragment for the timer:

Here's a code fragment for the timer:

```
timeadj  equ  d'65536'-d'25000'+d'12'+2


    movff   TMR0L,templ         ;read 16 bit counter
    movff   TMR0H,temph         ;get buffered high half
    movlw   low time_adj        ;add time adjustment
    addwf   templ,F
    movlw   high timeadj
    addwfc  temph,F
    movff   temph,TMR0H         ;pre-load high half
    movff   templ,TMR0L         ;write 16 bit counter
    bcf     INTCON,TMR0IF       ;clear flag
```

Whether using polling or interrupts, this section of code is used to reload the timer after an overflow.

Whether using polling or interrupts, this section of code is used to reload the timer after an overflow.

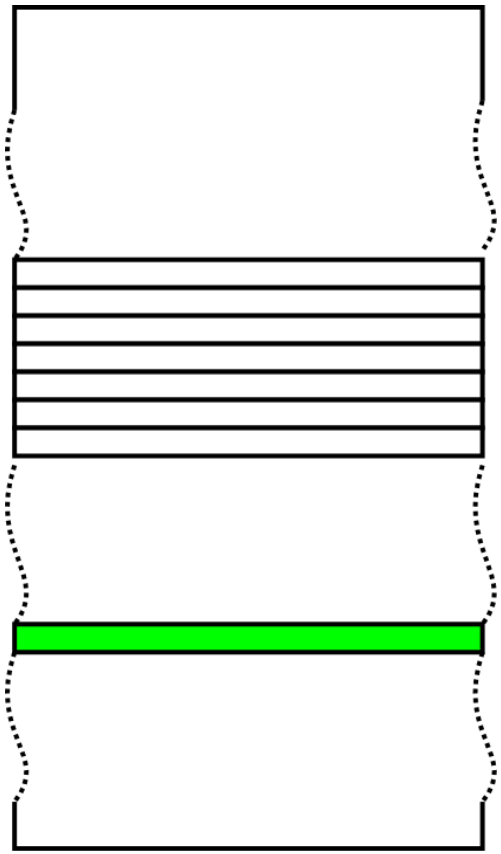
Following is an explanation of what it does, and why it's important.

Whether using polling or interrupts, this section of code is used to reload the timer after an overflow.

Following is an explanation of what it does, and why it's important.

For instance, the line
timeadj equ d'65536'-d'25000'+d'12'+2
needs some explanation.

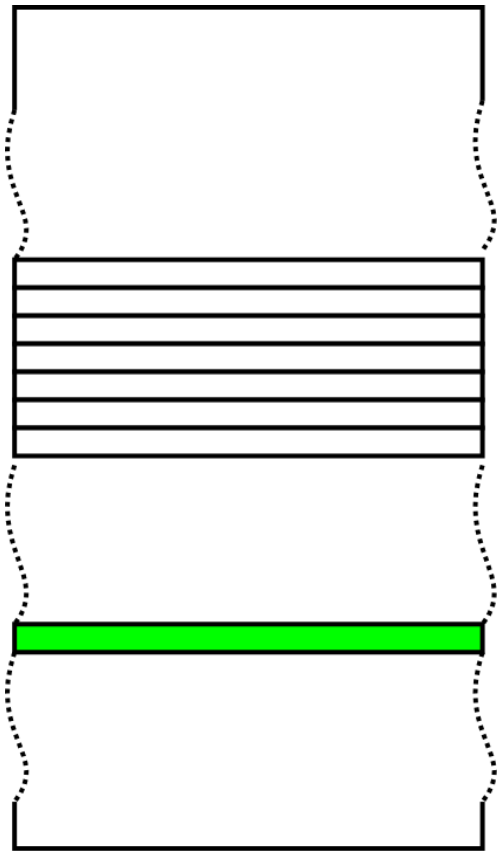
Timer adjustment



← timer value
at interrupt service

From the interrupt, (or after the flag is polled), until it gets processed, some time passes.

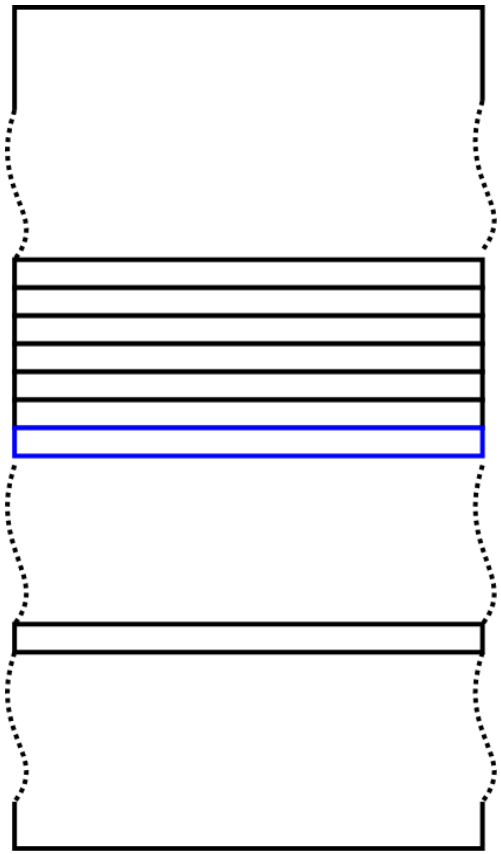
Timer adjustment



← timer value
at interrupt service

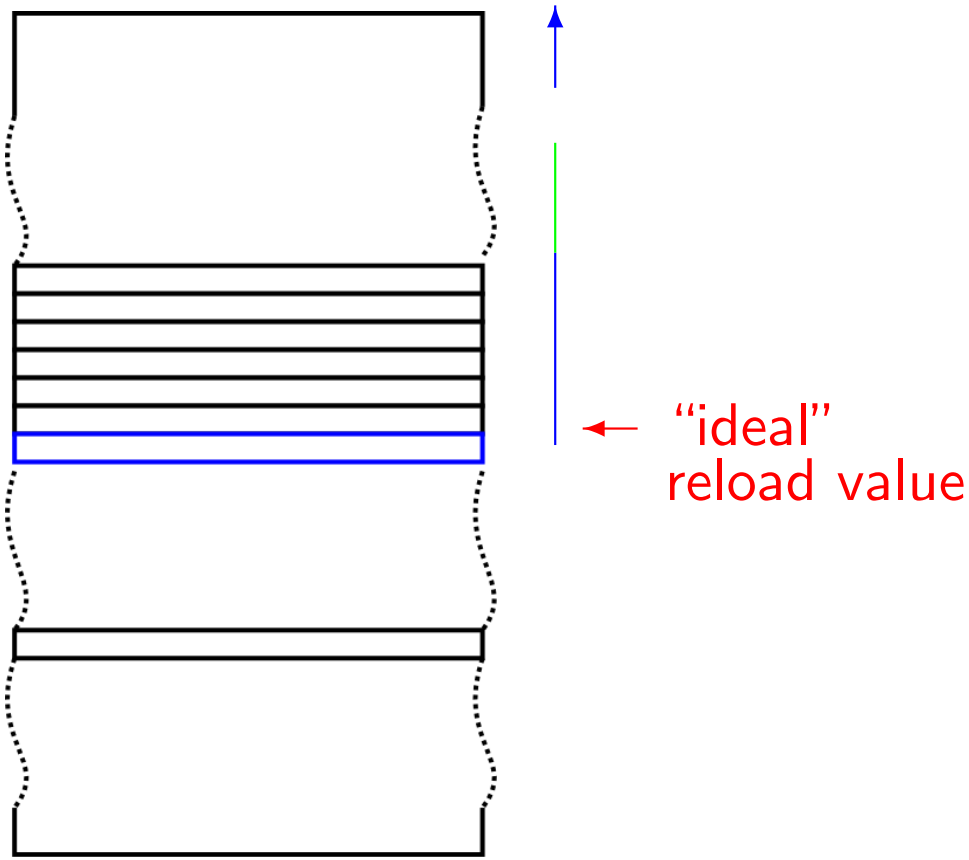
Probably only a few cycles, but important for precise timing.

Timer adjustment



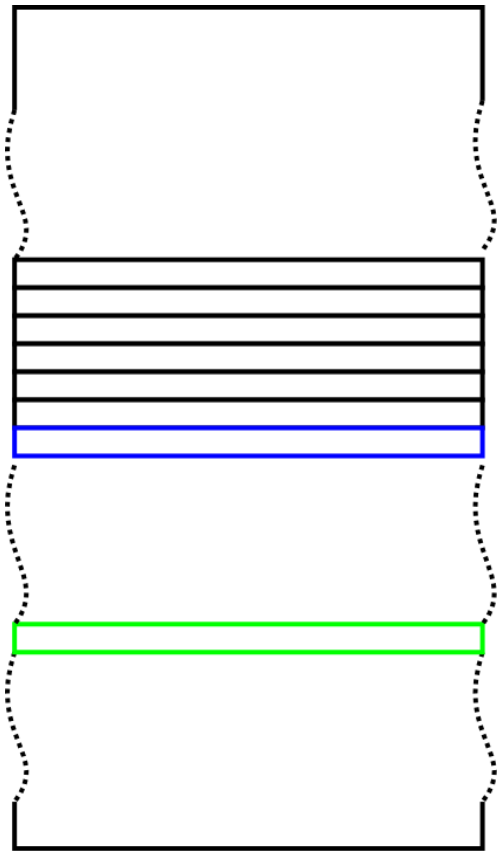
← "ideal" reload value

Timer adjustment



Interval desired is from "ideal" value to overflow (eg. FFFF)

Timer adjustment

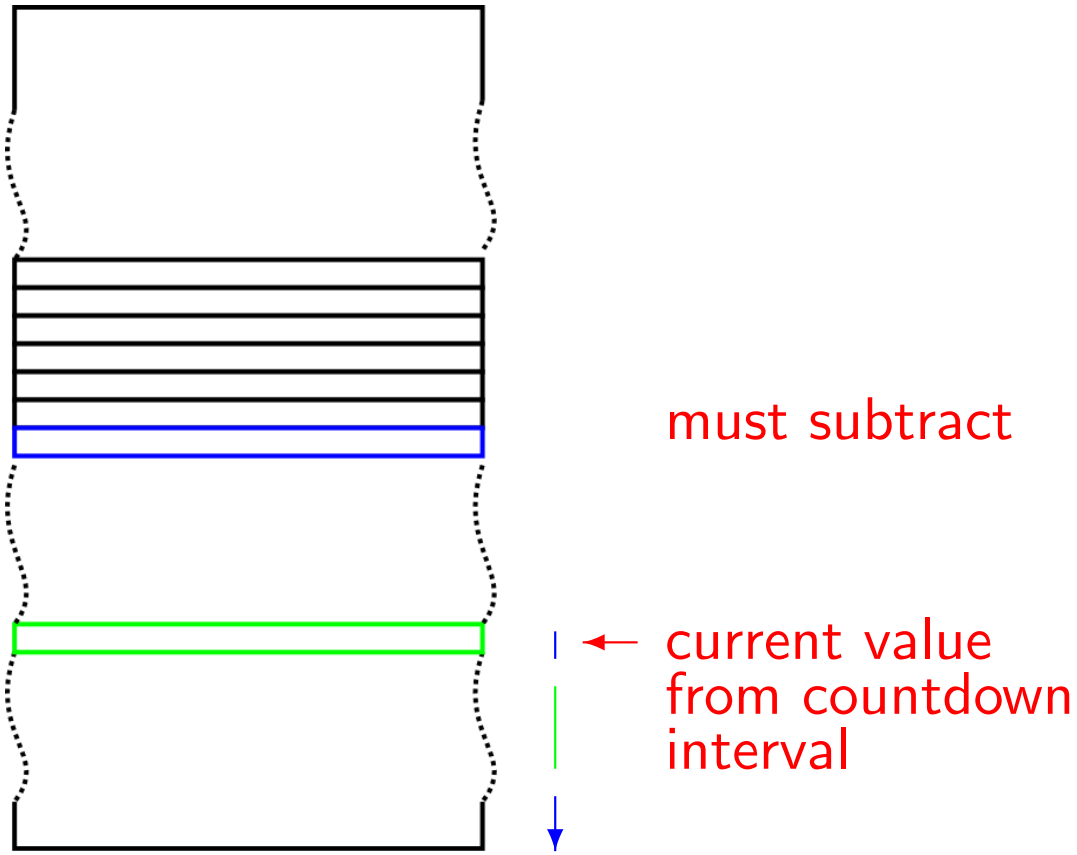


must subtract

← current value from countdown interval

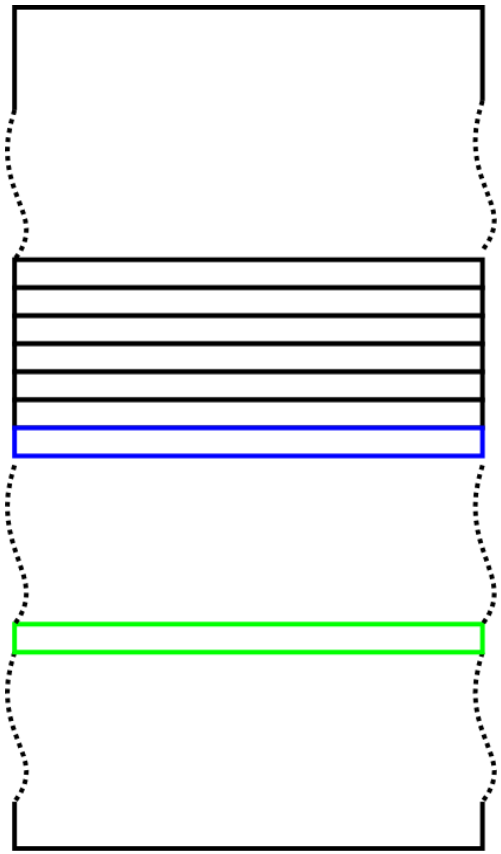
For instance, for an interval of 25000 cycles, we would nominally want to reload with d'65536'-d'25000'

Timer adjustment



We must subtract the time that happened before the interrupt service (or poll response).

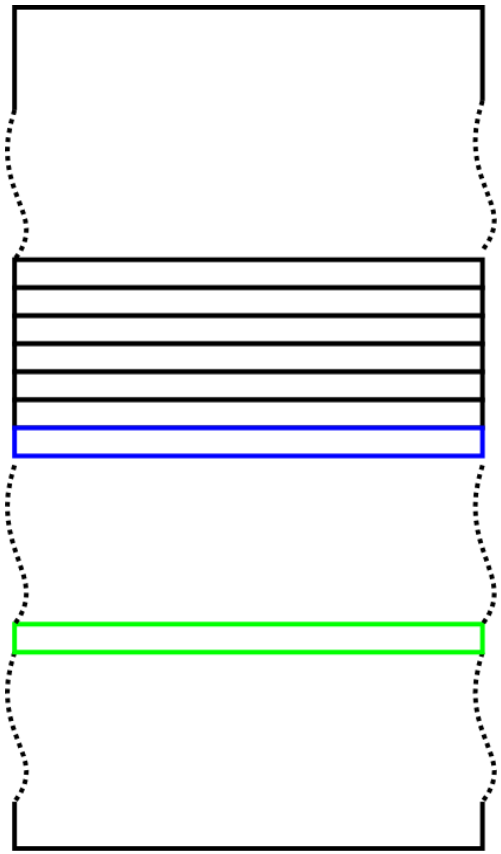
Timer adjustment



movff TMR0L,templ

First we must load the timer value, low byte first. This latches the high byte.

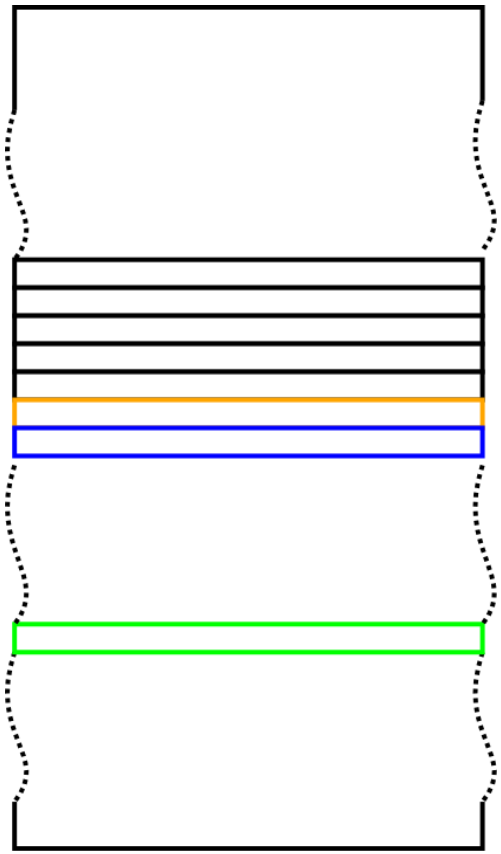
Timer adjustment



movff TMR0H,temph

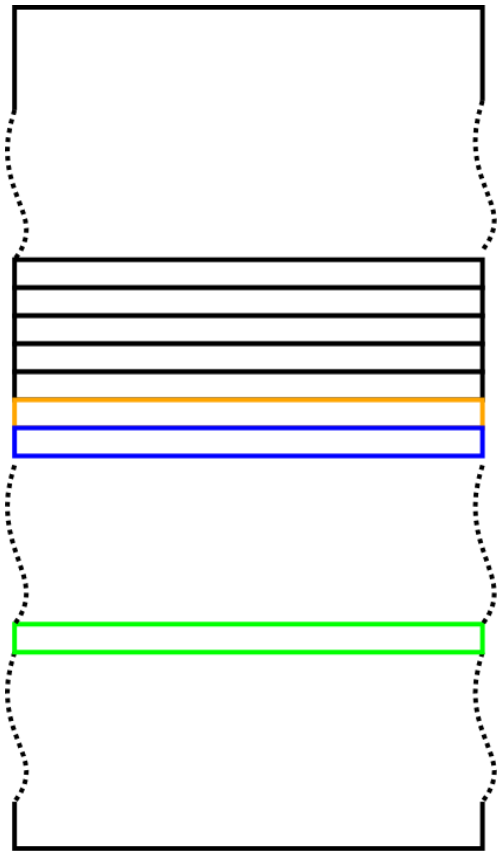
As we read the (latched) high timer byte, time is passing (2 cycles for this instruction).

Timer adjustment



movlw low adjust

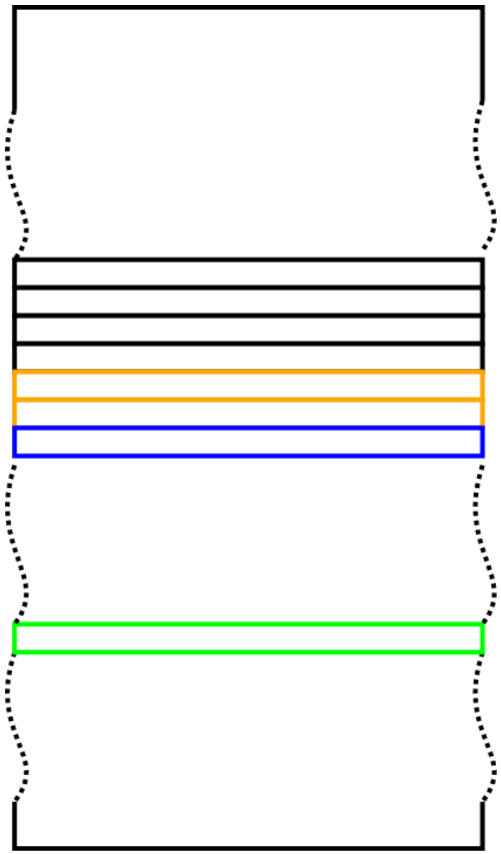
Timer adjustment



movlw low adjust

As we begin the subtraction, time is still passing (2 cycles for this instruction; 4 so far).

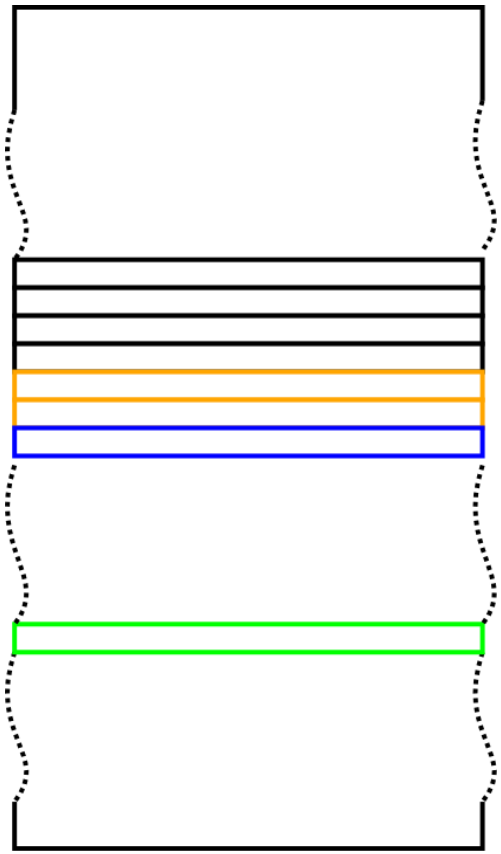
Timer adjustment



addwf templ,F

(Actually, instead of subtracting, we *add* to the reload value.)

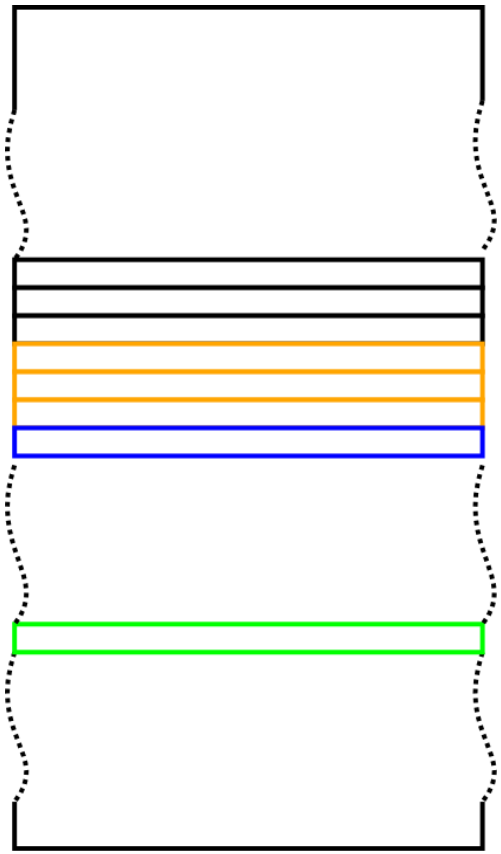
Timer adjustment



addwf templ,F

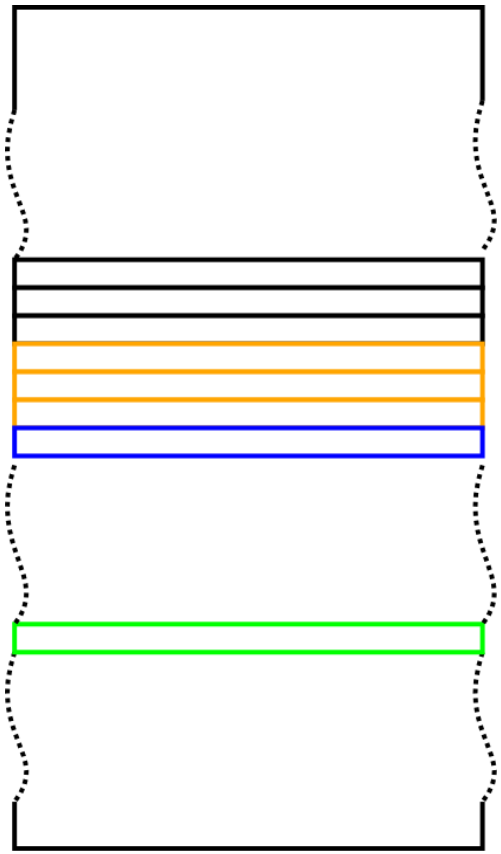
Time is still passing (2 cycles for this instruction; 6 so far).

Timer adjustment



movlw high adjust

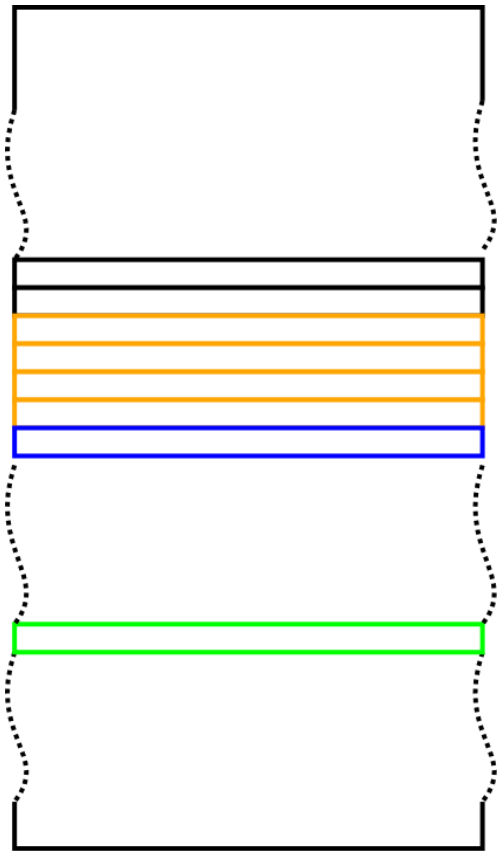
Timer adjustment



movlw high adjust

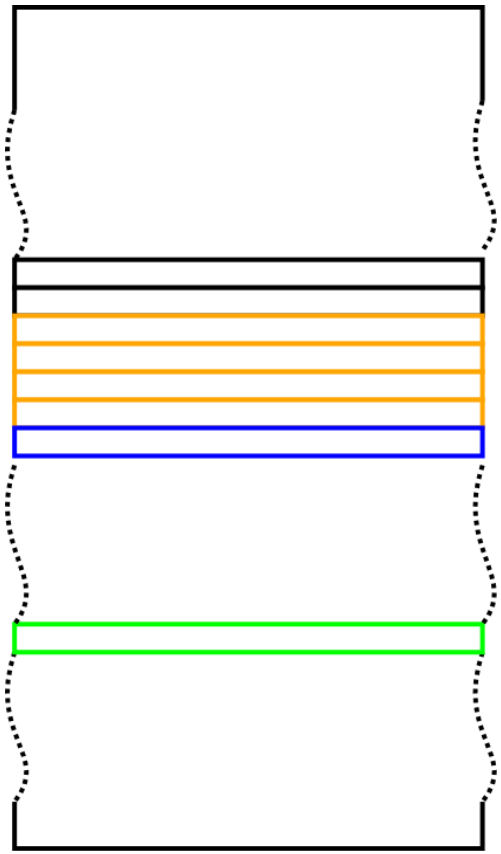
Time is still passing (2 cycles for this instruction; 8 so far).

Timer adjustment



addwfc temph,F

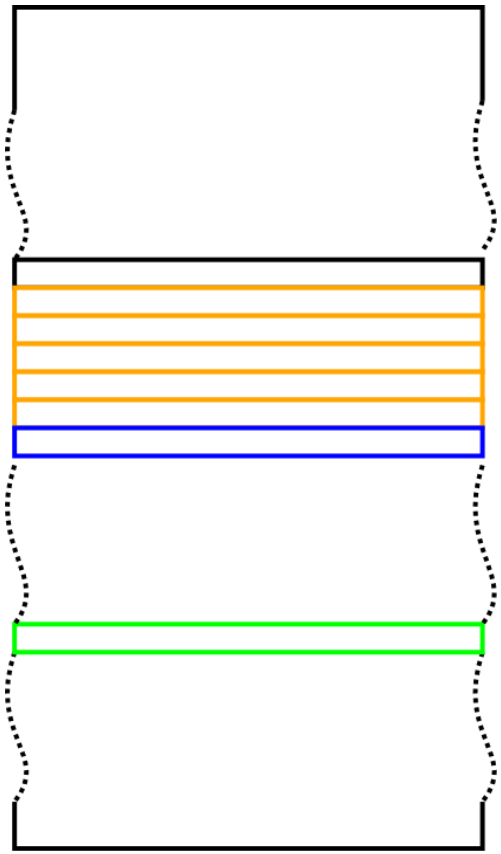
Timer adjustment



addwfc temph,F

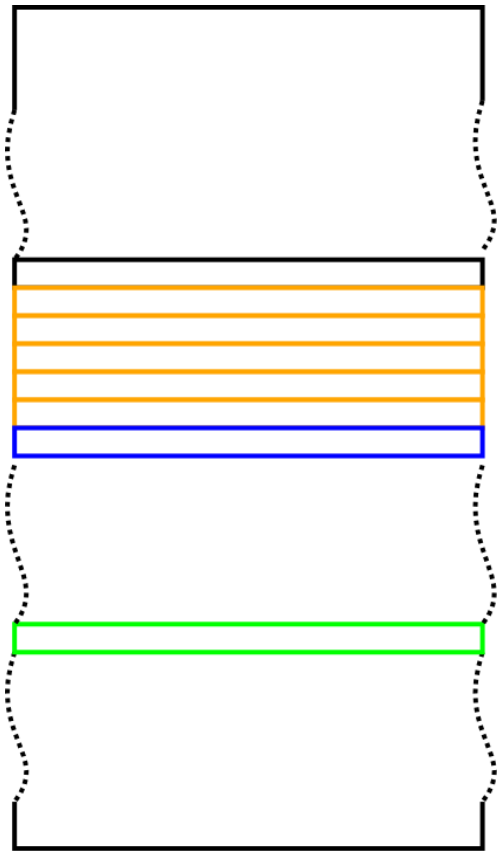
Time is still passing (2 cycles for this instruction; 10 so far).

Timer adjustment



movff temph,TMR0H

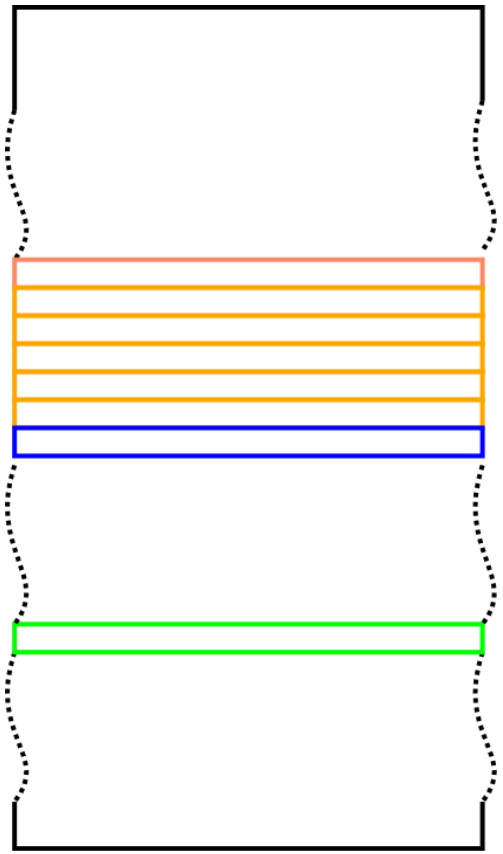
Timer adjustment



movff temph,TMR0H

Time is still passing (2 cycles for this instruction; 12 so far).

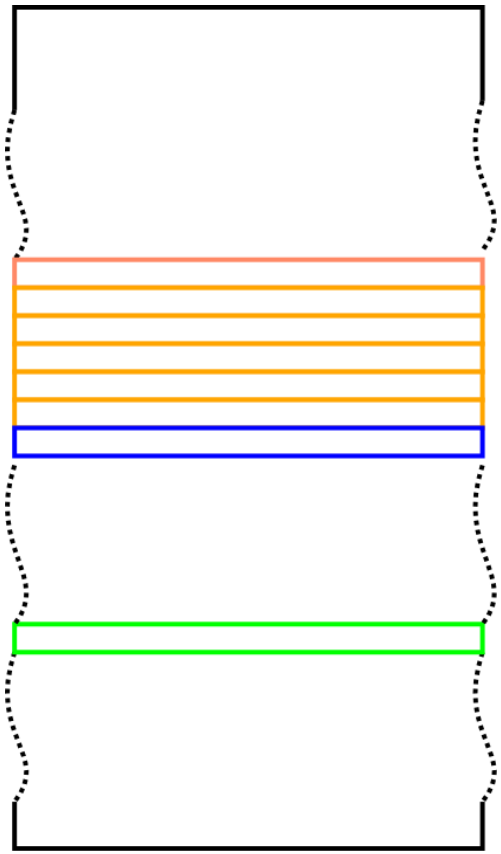
Timer adjustment



resynch delay

When we rewrite the low byte, the timer will be updated....

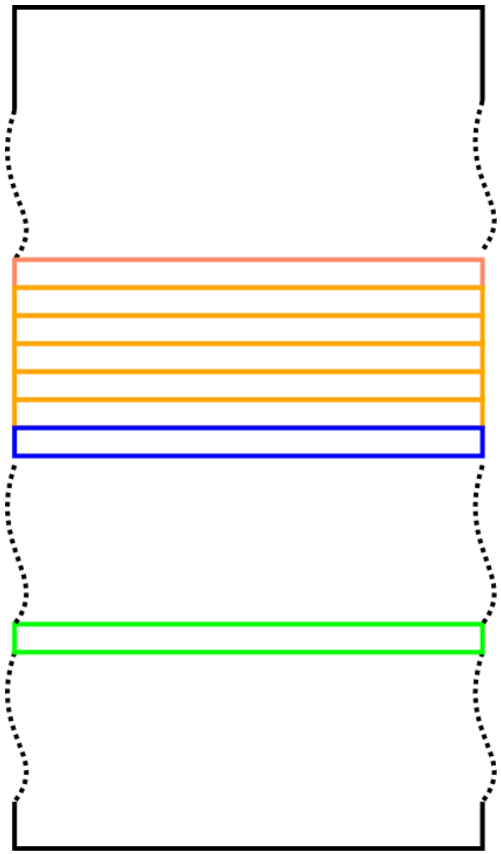
Timer adjustment



resynch delay

.. but there is a 2 cycle delay for the clock to resynch after re-loading.

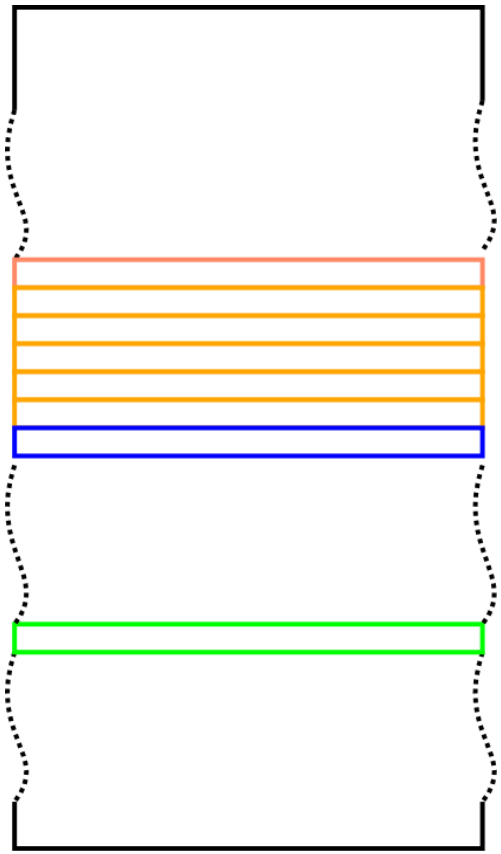
Timer adjustment



resynch delay

In total, there are 14 cycles used by the update, which must be subtracted from the desired interval.

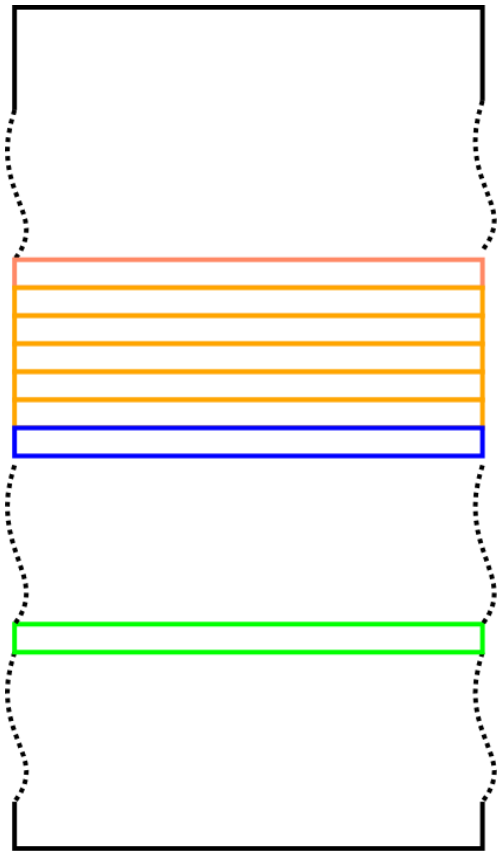
Timer adjustment



resynch delay

As above, these 14 cycles used by the update actually need to be *added* to the reload value.

Timer adjustment



resynch delay

After the adjustment is complete, the timer flag must be cleared.

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

  (eg. d'25000')

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

  (eg. d'25000')
- + d'12'

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

  (eg. d'25000')
- + d'12'

  to allow for the time for the arithmetic

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

  (eg. d'25000')
- + d'12'

  to allow for the time for the arithmetic
- + 2

  to allow for the time for clock resynch

So, whether using polling or interrupts, after an overflow the timer has to be reloaded with a value of

- d'65536'

  the number of counts in a 16 bit sequence, including zero
- – interval length (in clock cycles)

  (eg. d'25000')
- + d'12'

  to allow for the time for the arithmetic
- + 2

  to allow for the time for clock resynch

and then the flag must be cleared for the next interval.