get\_ticks( )

Syntax:

value = get\_ticks([stream]);

Parameters:

stream – optional parameter specifying the stream defined in #USE TIMER

Returns:

value – a 8, 16 or 32 bit integer. (int8, int16 or int32)

Function:

Availability:

All Devices

Returns the current tick value of the tick timer. The size returned depends on the size of the tick timer.

Requires:

#USE TIMER(options)



Examples:

#USE TIMER(TIMER=1,TICK=1ms,BITS=16,NOISR)



void main(void) {



unsigned int16 current\_tick;

current\_tick = get\_ticks();

}

See Also:

#USE TIMER, set\_ticks()

Usaremos:

#USE TIMER(TIMER=1,TICK=1us,BITS=32,ISR)

Pois deseja-se medir:

* o tempo com a resolução de 1us;
* o valor total do contador de ticks com 32 bits para contarmos até 232= 4.294.967.296, ou seja, aproximadamente 4300 segundos, ou 71 minutos.
* O Timer1 só tem 16 bits, os outros 16 bits são reservados pelo compilador e são incrementados

toda vez que Timer1 gera um overflow, que se dá quando a contagem passa de 65.535 para 0.

A opção ISR, garante o incremento do registrador adicional a cada interrupção de overflow de Timer1, caso se escolhesse NOISR, seria necessário chamar a função “get\_ticks()” de forma frequente para se evitar que houvesse um overflow que não fosse computado. Como ISR estará ativa, temos que habilitar as interrupções com o código:

enable\_interrupts(GLOBAL);

Não é necessário se habilitar a interrupção do Timer1, o compilador faz isso quando usamos a opção ISR.

