PrimeTime Set interval for timer and start it ticking

#include <Timer.h> **Time Manager**

void	**PrimeTime(***tmTaskPtr*, *msCount* **);**	
QElemPtr	*tmTaskPtr* ;	address of a 12-byte TMTask structure
long	*msCount* ;	interval before alarm, in milliseconds

Trap macro	_PrimeTime	
On entry	A0:	address of TMTask record
	D0:	specified delay time (long)
On exit	D0:	result code

PrimeTime starts the clock ticking on a timer-alarm triggered task previously prepared by means of a call to **InsTime**.

If the *count* parameter is a positive value, it is interpreted in milliseconds. If *count* is a negative value, it is interpreted in negated microseconds. (Microsecond delays are allowable only in the revised and extended Time Managers.) The task record specified by tmTaskPtr must already be inserted into the queue (by a previous call to **InsTime** or **InsXTime**) before your application calls the **PrimeTime** procedure. The **PrimeTime** procedure returns immediately, and the specified routine is executed after the specified delay has elapsed. If you call **PrimeTime** with a time delay of 0, the procedure runs as soon as interrupts are enabled.

In the revised and extended Time Managers, **PrimeTime** sets the high-order bit of the qType field to 1. In addition, any value of the *count* parameter that exceeds the maximum millisecond delay is reduced to the maximum. If you pause an unexpired task (with **RmvTime**) and then reinstall it (with **InsXTime**), you can continue the previous delay by calling **PrimeTime** with the count parameter set to 0.

tmTaskPtr is the address of a 12-byte TMTask structure previously used in a call to **InsTime**.

msCount specifies how long, in milliseconds, to wait before calling your wakeup routine.

Returns: none

Notes: **PrimeTime** may be called more than once for any previously installed **Time Manager** task (see **InsTime**). Secondary calls override the previously set interval. Thus, you can use this as a "watchdog" timer to be called before you start some event that is prone to getting locked into a loop.

PrimeTime does not make any demands on the **Memory Manager**, so it can be called from inside your alarm handling routine in order to set a new interval for that routine.

See **InsTime** for an example of usage.