Stopwatch Functions with the timer.device

by Mike Sinz

To measure time on the Amiga the system software provides the *timer.device*. The *timer.device* is very handy for many purposes, however, it does not have a built-in stopwatch. It does have some time manipulation functions that make this a simple thing to do though. This article shows how to do stopwatch functions using the *timer.device*.

The example code contains four routines needed to set up and use the stopwatch. The routines are:

- ❑ `struct timerequest  *Init_Timer  (VOID)`
- ❑ `VOID  Timer_Start (struct timerequest *)`
- ❑ `VOID  Timer_Stop  (struct timerequest *)`
- ❑ `VOID  Free_Timer  (struct timerequest *)`

The stopwatch functions all use a struct timerequest as their main parameter. This is just the usual Amiga IORequest structure with two extra fields for seconds and microseconds. Init_Timer() sets up the *timer.device* and returns a pointer to a struct timerequest. This pointer is used in Timer_Start() and Timer_Stop() which measure the elapsed time. Finally, Free_Timer() is used to close the *timer.device* and free the memory used for the timerequest.

An example main() function is listed below to show how to use the stopwatch functions to measure elapsed time. The example just does a Delay(73L) and then displays the amount of time that took. Some uses for the program might include measuring how long a student takes to answer a question, the amount of time a player has been playing, or how fast the system is. For more information on the *timer.device,* refer to the *ROM Kernel Manual: Libraries and Devices* (p. 871, ISBN 0-201-18187-8).

```
/*
 * Example stopwatch functions using the Amiga timer.device...
 */

/*
 * Makefile used to compile with Lattice C 5.04 or 5.05
 *

#
# MakeFile for StopWatch
#

CFLAGS= -b1 -cfirstq -ms0 -rr1 -v -w

OBJS=   StopWatch.o

LIBS=   LIB:lcsr.lib

.c.o:
@LC $(CFLAGS) $*

StopWatch: $(OBJS)
@BLink FROM LIB:c.o $(OBJS) TO StopWatch LIB $(LIBS) SMALLDATA SMALLCODE
 *
 *
 */

#include    <exec/types.h>
#include    <exec/memory.h>
#include    <devices/timer.h>

#include    <proto/exec.h>
#include    <proto/timer.h>
#include    <proto/dos.h>

#include    <stdio.h>

/*
 * The library base for the timer...
 */
struct  Library *TimerBase=NULL;

/*
 * This gets the starting time...
 */
VOID Timer_Start(struct timerequest *Time_Req)
{
    Time_Req->tr_node.io_Command=TR_GETSYSTIME;
    Time_Req->tr_node.io_Flags=IOF_QUICK;
    DoIO((struct IORequest *)Time_Req);
}
```

```
/*
 * This gets the ending time and computes the time difference
 * in the timerequest->tr_time.
 */
VOID Timer_Stop(struct timerequest *Time_Req)
{
struct  timeval StartTime;

    StartTime=Time_Req->tr_time;

    Time_Req->tr_node.io_Command=TR_GETSYSTIME;
    Time_Req->tr_node.io_Flags=IOF_QUICK;
    DoIO((struct IORequest *)Time_Req);

    SubTime(&(Time_Req->tr_time),&StartTime);
}

/*
 * Initialize the stopwatch...
 */
struct timerequest *Init_Timer(VOID)
{
register    struct  timerequest *Time_Req=NULL;
register    struct  MsgPort     *port=NULL;


    if (port=CreatePort(NULL,NULL))
    {
        if (Time_Req=(struct timerequest *)CreateExtIO(port,
                                            sizeof(struct timerequest)))
        {
            if (!OpenDevice(TIMERNAME,UNIT_VBLANK,
                            (struct IORequest *)Time_Req,NULL))
            {
                TimerBase=(struct Library *)Time_Req->tr_node.io_Device;
            }
            else
            {
                DeleteExtIO((struct IORequest *)Time_Req);
                Time_Req=NULL;
            }
        }
        if (!Time_Req)
        {
            DeletePort(port);
            port=NULL;
        }
    }

    return(Time_Req);
}
```

```
/*
 * Free up the timer...
 */
VOID Free_Timer(struct timerequest *Time_Req)
{
    if (Time_Req)
    {
        CloseDevice((struct IORequest *)Time_Req);
        DeletePort(Time_Req->tr_node.io_Message.mn_ReplyPort);
        DeleteExtIO((struct IORequest *)Time_Req);
    }
}


/*
 * A simple main() to use these features...
 */
VOID main(int argc, char *argv[])
{
register    struct  timerequest *StopWatch;

    if (argc)   /* Check if CLI... */
    {
        if (StopWatch=Init_Timer())
        {
            Timer_Start(StopWatch);

            /* Now we do something... */
            Delay(73L);

            Timer_Stop(StopWatch);

            /* Now we display the results... */

            printf("Time taken: %ld.%06ld seconds\n",
                        StopWatch->tr_time.tv_secs,
                        StopWatch->tr_time.tv_micro);

            /* Free the structure again... */
            Free_Timer(StopWatch);
        }
    }
}
```