timer

COLLABORATORS			
	TITLE : timer		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 18, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

timer

1.1 timer.doc

```
 --background--      CmpTime()          SubTime()          TR_SETSYSTIME
 AbortIO()           GetSysTime()       TR_ADDREQUEST
 AddTime()           ReadEClock()       TR_GETSYSTIME
```

1.2 timer.device/--background--

```
TIMER REQUEST
    A time request is a non standard IO Request.  It has an IORequest
    followed by a timeval structure or an eclockval structure.

TIMEVAL
    A timeval structure consists of two longwords.  The first is
    the number of seconds, the latter is the fractional number
    of microseconds.  The microseconds must always be "normalized"
    e.g. the longword must be between 0 and one million.

ECLOCKVAL
    A eclockval structure consists of two longwords.  The first is
    the high order 32 bits of a 64 bit number and the second is the
    the low order 32 bits.  The 64 bit number is a count of "E" clock
    ticks.  The "E" clock frequency is related to the master clock
    frequency of the machine and can be determined by calling the
    ReadEClock() library like call.

UNITS
    The timer contains five units -- two designed to accuratly measure
    short intervals, one that has little system overhead and is very
    stable over time, and two that work like an alarm clock.

UNIT_MICROHZ
    This unit uses the programmable timers in the 8520s to keep track
    of its time.  It has precision down to about 2 microseconds, but
    will drift as system load increases.  The accuracy of this unit
    is the same as that of the master clock of the machine.  This unit
    uses a timeval in its timerequest.
```

```
UNIT_VBLANK
    This unit uses a strobe from the power supply to keep track of its
    time or the "E" clock on machines without power supply strobes.
    It is very stable over time, but only has a resolution of that of
    the vertical blank interrupt.  This unit is very cheap to use, and
    should be used by those who are waiting for long periods of time
    (typically 1/2 second or more).  This unit uses a timeval in its
    timerequest.

UNIT_ECLOCK
    This unit is exacly the same as UNIT_MICROHZ except that it uses
    an eclockval instead of a timeval in its timerequest.

UNIT_WAITUNTIL
    This unit waits until the systime is greater than or equal to the
    time in the timeval in the timerequest.  This unit has the same
    resolution and accuracy as that of UNIT_VBLANK.

UNIT_WAITECLOCK
    This unit waits until the E-Clock value as returned by ReadEClock()
    is greater than or equal to the eclockval in the timerequest. This
    unit has the same resolution and accuracy as that of UNIT_ECLOCK.

LIBRARY
    In addition to the normal device calls, the timer also supports
    several direct, library like calls.

BUGS
    In the V1.2/V1.3 release, the timer device has problems with
    very short time requests.  When one of these is made, other
    timer requests may be finished inaccurately.  A side effect
    is that AmigaDOS requests such as "Delay(0);" or
    "WaitForChar(x,0);" are unreliable.
```

1.3 timer.device/AbortIO

```
NAME
    AbortIO -- Remove an existing timer request.

SYNOPSIS
    error = AbortIO( timerequest )
    D0               A1

    LONG AbortIO( struct timerequest * );

FUNCTION
    This is an exec.library call.

    This routine removes a timerquest from the timer.  It runs in
    the context of the caller.

INPUTS
    timerequest - the timer request to be aborted
```

```
RETURNS
    0  if the request was aborted, io_Error will also be set to
       IOERR_ABORTED.
    -1 otherwise

NOTES
    This function may be called from interrupts.

SEE ALSO
    exec.library/AbortIO()

BUGS
```

1.4 timer.device/AddTime

```
NAME
    AddTime -- Add one time request to another.

SYNOPSIS
    AddTime( Dest, Source )
             A0    A1

    void AddTime( struct timeval *, struct timeval *);

FUNCTION
    This routine adds one timeval structure to another.  The
    results are stored in the destination (Dest + Source -> Dest)

    A0 and A1 will be left unchanged

INPUTS
    Dest, Source -- pointers to timeval structures.

NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/CmpTime(),
    timer.device/SubTime()

BUGS
```

1.5 timer.device/CmpTime

```
NAME
    CmpTime -- Compare two timeval structures.

SYNOPSIS
    result = CmpTime( Dest, Source )
    D0                A0    A1

    LONG CmpTime( struct timeval *, struct timeval *);
```

```
FUNCTION
    This routine compares timeval structures

    A0 and A1 will be left unchanged

INPUTS
    Dest, Source -- pointers to timeval structures.

RESULTS
    result will be   0 if Dest has same time as source
                    -1 if Dest has more time than source
                    +1 if Dest has less time than source

NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/AddTime(),
    timer.device/SubTime()

BUGS
    Older version of this document had the sense of the return
    codes wrong; the code hasn't changed but the document has.
```

1.6 timer.device/GetSysTime

```
NAME
    GetSysTime -- Get the system time. (V36)

SYNOPSIS
    GetSysTime( Dest )
                A0

    void GetSysTime( struct timeval * );

FUNCTION
    Ask the system what time it is.  The system time starts off at
    zero at power on, but may be initialized via the TR_SETSYSTIME
    timer.device command.

    System time is monotonocally increasing and guarenteed to be
    unique (except when the system time is set back).

    A0 will be left unchanged.

    This function is less expensive to use than the TR_GETSYSTIME
    IORequest.

INPUTS
    Dest -- pointer to a timeval structure to hold the system time.

RESULTS
    Dest -- the timeval structure will contain the system time.
```

```
NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/TR_GETSYSTIME,
    timer.device/TR_SETSYSTIME,

BUGS
```

1.7 timer.device/ReadEClock

```
NAME
    ReadEClock -- Get the current value of the E-Clock. (V36)

SYNOPSIS
    E_Freq = ReadEClock( Dest )
    D0                   A0

    ULONG ReadEClock ( struct EClockVal * );

FUNCTION
    This routine calculates the current 64 bit value of the E-Clock
    and stores it in the destination EClockVal structure. The count
    rate of the E-Clock is also returned.

    A0 will be left unchanged

    This is a low overhead function designed so that very short
    intervals may be timed.

INPUTS
    Dest -- pointer to an EClockVal structure.

RETURNS
    Dest -- the EClockVal structure will contain the E-Clock time
    E_Freq -- The count rate of the E-Clock (tics/sec).

NOTES
    This function may be called from interrupts.

SEE ALSO

BUGS
```

1.8 timer.device/SubTime

```
NAME
    SubTime -- Subtract one time request from another.

SYNOPSIS
    SubTime( Dest, Source )
             A0    A1
```

```
    void SubTime( struct timeval *, struct timeval *);

FUNCTION
    This routine subtracts one timeval structure from another.  The
    results are stored in the destination (Dest - Source -> Dest)

    A0 and A1 will be left unchanged

INPUTS
    Dest, Source -- pointers to timeval structures.

NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/AddTime(),
    timer.device/CmpTime()

BUGS
```

1.9 timer.device/TR_ADDREQUEST

```
NAME
    TR_ADDREQUEST -- Submit a request to wait a period of time.

FUNCTION
    Ask the timer to wait a specified amount of time before
    replying the timerequest.

    The message may be forced to finish early with an
    AbortIO()/WaitIO() pair.

TIMER REQUEST
    io_Message      mn_ReplyPort initialized
    io_Device       preset by timer in OpenDevice
    io_Unit         preset by timer in OpenDevice
    io_Command      TR_ADDREQUEST
    io_Flags        IOF_QUICK permitted (but ignored)
    tr_time         a timeval structure specifying how long the
                        device will wait before replying

RESULTS
    tr_time         will be zeroed

NOTES
    This function may be called from interrupts.

    Previous to 2.0, the tr_time field was documented as containing
    junk when the timerequest was returned.

SEE ALSO
    timer.device/AbortIO(),
    timer.device/TimeDelay(),
```

```
BUGS
```

1.10 timer.device/TR_GETSYSTIME

```
NAME
    TR_GETSYSTIME -- get the system time.

FUNCTION
    Ask the system what time it is.  The system time starts off at
    zero at power on, but may be initialized via the TR_SETSYSTIME
    call.

    System time is monotonically increasing, and guaranteed to be
    unique (except when the system time is set backwards).

TIMER REQUEST
    io_Message      mn_ReplyPort initialized
    io_Device       preset by timer in OpenDevice
    io_Unit         preset by timer in OpenDevice
    io_Command      TR_GETSYSTIME
    io_Flags        IOF_QUICK permitted

RESULTS
    tr_time         a timeval structure with the current system
                        time

NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/TR_SETSYSTIME,
    timer.device/GetSysTime(),

BUGS
```

1.11 timer.device/TR_SETSYSTIME

```
NAME
    TR_SETSYSTIME -- Set the system time.

FUNCTION
    Set the system idea of what time it is.  The system starts out
    at time "zero" so it is safe to set it forward to the real
    time.  However, care should be taken when setting the time
    backwards.  System time is generally expected to monotonically
    increasing.

TIMER REQUEST
    io_Message      mn_ReplyPort initialized
    io_Device       preset by timer in OpenDevice
    io_Unit         preset by timer in OpenDevice
    io_Command      TR_GETSYSTIME
```

```
    io_Flags        IOF_QUICK permitted
    tr_time         a timeval structure with the current system
                        time

RESULTS
    tr_time         will contain junk

NOTES
    This function may be called from interrupts.

SEE ALSO
    timer.device/TR_GETSYSTIME,
    timer.device/GetSysTime(),

BUGS
```