101a1c80-0

Hans Olsen

Copyright © 1 May 1997

COLLABORATORS

	TITLE : 101a1c80-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Hans Olsen	July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

101a1c80-0

1.1 timer.device for Blitz Basic 2

INC_Timer.bb2 Version 0.1 Beta

Written by Hans Olsen

Copyright © 1997

All Rights Reserved

1.2 Why did he do this?

Why?

Well, in the blitz mailinglist there was a debate about how
to decrease CPU load at the same time as you handle all
window events and execute subroutines at a regular basis.
The best way is to use the timer device (so I have been told),
so I decided to implement a set of functions for Blitz that
makes it easy to handle the device.

1.3 The Legal stuff

I take no resposibility for any damage these functions
may cause, etc etc... Don't blame or flame me if this
doesn't work as it's supposed to.
This code is freeware, do whatever you wan't as long
as you don't claim it as your own code.
This is the first beta release, and there probebly is
a few bugs somewhere.
I don't have any docs on how to use the timer.device,
and there is a few things that I might have done in
the wrong way. If you have any docs on this topic,
please email me.
This guide was written quite hastely so there is lots of
misspelling, but I thought it was more important to get
this out as fast as possible.

1.4 Requirements

Requirements
Blitz Basic 2.1 (Older may work)
Acidlibs from Red When Excited's website. (Older may work)
System used for developement
Amiga 1200/030EC/882/40MHZ/16MB/240MB
Kickstart 39.106, Workbench 40.42

1.5 How to install

Installation
This archive should include the following files:
demo.bb2 - Example code in tokenized format.
demo.asc - Same example but in plain ascii.
INC_Timer.bb2 - The actual timer functions.
INC_Times.asc - The timer functions in ascii.
You shouldn't have to use the ascii versions, but if
there is any "??????" in the tokenized version try the
ascii instead.
The INC_Timer.bb2 contains the actual functions, place
this file where you usualy keep your Blitz include files.

(For an explenation of this file take a look at the end
of this guide.)
The demo.bb2 assumes the INC_Timer.bb2 is in your current
directory, so you may need to alter the XINCLUDE line at
the beginning.

1.6 Contacting the Author

If you have found any bugs, have suggestions for improvements
or any other comments please email me.
eMail: pt96hol@student.hk-r.se
WWW: http://www.student.hk-r.se/~pt96hol (Personal homepage)
http://oden.rsn.hk-r.se/~ols (Amiga/Blitz pages)
If you have any comments that might be interesting for all
blitzers mail the blitz mailinglist instead.
Newer versions if this timer package will be available at my
Amiga/Blitz pages.

1.7 The commands

How to use the include file:
In the program that you include the INC_Timer.bb2 file you
must have the blitzlibs:amigalibs.res loaded, otherwise you
will get an error like 'unknown newtype' or something like
that.
Then at the begining of your program you should use the
XINCLUDE command to include the file.
What's in the include file?
When you have done this you will have a set of new timer
functions.
CreateTimer{} - This function opens the device and
some other things.
DeleteTimer{} - This statement removes the timer device
and nicely closes everything.
You MUST call this before you end your
program.
SetTimer{sec,micro} - This statement sets the timer to call you
after a sertain time.
CheckIfTimer{} - This function returns true if it was the

timer that called ous after a WAIT.
(This is the routine that you most likely
will be useing.)
WaitForTimer{} - This statement will wait for the timer
to call you, it will ignore everything
else.
ReplyTimer{} - You shouldn't use this statement.
How do i use this?
First you call CreateTimer, you should only create one, check if
you actualy got a timer or if there was an error.
When you have done this you are free to start the timer useing
the SetTimer statement.
For a look at how you should design your main loop look at the
demo program. You should use the blitz WAIT command, but remember
to check for ALL events that might have appeared before going
back to the WAIT command.
When you have checked if it was the timer that called you should
(if it was the timer) set the timer again to make it call you
again.
If your timer delay is short you should try to make your main
loop as fast as possible.
When you are finished you MUST remove the timer with DeleteTimer
if you don't do this there will be 'junk' left in the memory and
a system crash isn't too far away.

1.8 CreateTimer

result = CreateTimer{}
Purpose:
This function tries to setup a new timer, you should only call
this function once.
The result can be:
0 - Everything is fine.
1 - Can't open port.
2 - Can't create request.
3 - Can't open timer device.
If you get anything above 0 the function faild.

1.9 DeleteTimer

DeleteTimer{ }
Purpose:
This statement frees everything the CreateTimer command allocated,
and if the timer is active it will be deactivated before removal.
You should allways call this statement before ending your program,
otherwise the timer will queue lots and lots of request to the
timer port. And if the timer port is removed before the timer is
deactivated you most certanly will get a guru.
Result:
none

1.10 SetTimer

SetTimer{sec, micro}
Purpose:
This command activates the timer. The sec and micro is how long the
timer will wait before calling you. You should only call this once
and only call it again when you have recived the call.
(It may be possible to call it several times before recieving the
call, but it wasn't design with that in mind.)
Result:
none

1.11 WaitForTimer

WaitForTimer{ }
Purpose:
This statement will wait until the timer calls you, any other
calls is ignored. If the timer isn't set to call you the program
will lock.
Result:
none

1.12 CheckIfTimer

result = CheckIfTimer{ }

Purpose:

After calling a WAIT you must check what was calling you, this function checks if it was the timer.

If it was the timer, remember to set the timer again if you would like it to call again.

Result:

True - It was the timer.

False - It was something else.

1.13 ReplyTimer

ReplyTimer{ }

Purpose:

If you mess around with the timer port and get the timer message your self you should use this command to reply to the timer.

It seems like the device system doesn't send regual messages but rather requests so the ordinary replymsg command won't work.

Result:

none

1.14 The details

The theory

To gain access to the timer.device you must create a message port that will allow the timer to call you.

To access the device system you must create an IO request that will talk to the timer.device.

When you have the above you can open the timer device.

The timer device can be opened in several modes, I have choosen the first mode. (I tried some of the others but they didn't work.)

Now you have a fully operational timer device connected to your port. To activate the timer you have to send the previously prepared IO request containing the delay.

When the timer device have waited it calls your port, now you have to reply to the timer that you have got the call, this part I'm not so sure about since it doesn't work as ordinary messages. Currently

I deactivate the request.
When the program ends (or you don’t need the timer any more) you must
deactivate any active timer request, otherwise the timer device might
call a port that you/Blitz have removed. If the port still exists
but the request isn’t deactivated the timer device will call the port
and wait for an answer forever.
When there is no more active requests the device must be closed, the
request must be returned to the system and finaly the port can safely
be closed.
For handling of the port and request there is two variables,
hc_timer_wp.l - the port pointer,
*hc_timer_ioreq.timerequest - the IO request pointer.
It shouldn’t be too difficult to avoid useing these names by
accident.