12Ghosts Timer

Start applications on time and schedule reminders

Timer Q&A Options

12Ghosts Timer is an easy to use and powerful scheduler for starting applications, opening documents or Web pages, setting up reminders or wake up with music, or previewing the exact moon phase. Executable only 190 KB. Start once, countdown, or recurring per second; or start after logon. Set process priority and window size. Batch capable. Runs on Windows 95/98 as well as on NT/2000. Optimized timer control for easy usage.

- Start **once** or **recurring** (between the years 1900 and 2100), or **after logon**
- Programmable, batch capable
- Low memory footprint
- Start applications, documents, Internet pages, or reminders with snooze function
- Specify the **process priority** (low, normal, high, real-time) and window state (minimized, maximized, no activate, hidden)
- Start on working days only, and set a don't-start-after time
- Customize defaults for new timers
- Exact display of the age of the moon in minutes, phase of the moon in percent. Solved

the equation of Kepler, calculating the sun's geocentric ecliptic longitude, and everything.

- Full online documentation, **context help** on every control
- Optional tray icon or free movable icon, user defined hotkey
- Optimized timer control, all settings in one view

Easier Usage

- Shuffle through all timers with Up/Down keys (if description field is selected)
- Holidays now displayed in the settings window
- To execute a scheduled program, you can now click on the little application icon left of the Start Path
- More/Less button to simplify the settings window
- Be productive with 12Ghosts Timer: Select one of ten ready-to-use examples!

For more details on options and registry values please click here.

Some ideas on what you might start with 12Ghosts Timer

- To display a **reminder** just keep the Start Path empty. You can still add a description that will appear in the description window. You can send a reminder to snooze. It will then appear as a new timer with the description "SNOOZE: old description". A "snoozed" timer will in fact appear even after you stopped 12Ghosts Timer or restarted Windows!

- Scan all local, non-removable disk drives in Windows 95/98:

  Start Path: **scandskw.exe**
  Parameters: /all /NonInteractive

- Defragment all local, non-removable disk drives and free space in Windows 95/98:

  Start Path: **defrag**
  Parameters: /all /f /detailed /noprompt

- CD Player for a wake up instead of a reminder! Replace D: with your CD-ROM's drive letter. You can edit the play list to set the track to start with.

  Start Path: **cdplayer**
  Parameters: /play D:

- Start MP3 music title with Winamp:

  Start Path: C:\Program Files\Winamp**Winamp.exe**
  Parameters: "C:\MyMP3s\Title.mp3"

- Close a window at a certain time with 12Ghosts WinControl:

  Start Path: "C:\Program Files\12Ghosts**12wincontrol.exe**"
  Parameters: `/N'<Text in title bar>' /close`

- Start 12Ghosts SetTextColor every couple of minutes to ensure that the colors will always be set as you like them. Copy your favorite color values from the automatically created shortcut in the Startup folder. (t=text, b=background, r=red, g=green, b=blue, values between 0-255, or /trans).

  Start Path: **12setcolor.exe**
  Parameters: /tr:0 /tg:0 /tb:255 /trans

- Backup "My Documents" folder with XCOPY:

  Start Path: **XCOPY**
  Parameters: "C:\My Documents" "D:\Backup\Docs\" /s /e /h /r /d /i /f /c

- Backup customized Windows settings with 12Ghosts ProfileCopy:

  Start Path: C:\Program Files\12Ghosts**12profilecopy.exe**
  Parameters: /s

- Save desktop icon layout with 12Ghosts SaveLayout:

  Start Path: C:\Program Files\12Ghosts**12savelayout.exe**
  Parameters: /s "C:\Windows\Desktop\layout1.sl"

- Check out the 12Ghosts Timer homepage for the latest updates every 60 days:

  Start Path: **http://12Ghosts.com/ghosts/timer.htm**
  Parameters:

- Change screen savers from time to time: To switch screen savers while one is running call the 12Ghosts ScrSavMan:

  Start Path: **12scrsav.exe**
  Parameters: /stop /change:scrnsave.scr /start

- How do you play a *.WAV file? It's not that complicated, but it depends on which version of media player you have:

  Start Path: D:\Windows\System32\sndrec32.exe
  Parameters: /play /close c:\media\sound.wav

  - or -

  Start Path: D:\Windows\System32\rundll32.exe
  Parameters: D:\Windows\System32\amovie.ocx,RunDll /play /close c:\media**sound.wav**

- or -

Start Path: D:\Program Files\Windows Media Player\mplayer2.exe
Parameters: /play /close c:\media\sound.wav

Every program can be scheduled. Please look in the respective shortcut to find out what parameters may be necessary. If you can start it manually, you can also start it with 12Ghosts Timer!

Timer Q&A Options

Contents Order FAQ HowTo Reviews 12Ghosts.com E-mail Contact

Our computers run on renewable energy only.

12Ghosts - The "Seven Dwarves" for Windows. ®

12Ghosts Timer

Timer Questions & Answers

Timer Q&A Options

Questions

How can I close a DOS box after it has finished?

There's an easier way than to change the default PIF ofcommand com: start command.com with parameters = '/c batchfile.bat'. This closes a DOS box when it’s finished!

See also the description of WinControl on how to close windows and terminate applications automatically.

Is 12-Timer more exact than the Windows clock?

Your confidence in Microsoft software may be unlimited, but... then don't look at the sluggish display of seconds in the Date/Time Properties or Clock.exe:

- Compare both times with a metronome or a analog wrist watch.
- Open a command prompt and enter the command "time" shortly before the next wrap of a second, compare the milliseconds.

After doing a comparison, you will notice a slightly earlier wrap to the next second in 12-ShowTime. Technically, for ShowTime, this wrap simply can not occur before the next second because we set the timer exactly to the next full second. In other words, the time in the settings window title can only wrap to the next second a couple of milliseconds later and never earlier. On the other hand, the seconds in the Date/Time Properties or Clock.exe are sometimes half a second too late.

I have set a recurring timer to 1 second. How can I stop it?

If the Timer settings window is open, you should be able to press Alt+A and Alt+S within one second to de-activate the timer and save the changes.

Or right click on the tray icon and select Dismiss All Timers and Quit. You can also select "End Task", after pressing Ctrl+Alt+Del (NT: start the TaskManager). Then open Start - Run - type "regedit" - OK, move down to the appropriate Timer subkey: HKEY_CURRENT_USER\Software\PACT Software\Timer\Txxx and change the value StartActivated to 0.

Is the low memory footprint important for me?

As long as you have enough main memory, it's not. However, most people don't and running a program around the clock is an additional load for the memory, which should be kept as small as possible.

What most likely is slowing down your system is the additional swapping of memory to and from the hard disk, if memory becomes short. Windows swaps automatically to have always enough main memory. You can load hundreds of megabytes of documents, although you might only have 16 MB RAM. The problem is, that reading from the hard disk is much slower than accessing the RAM.

Usually you will not notice 12-Timer at all because of the small memory footprint.

You say the "memory footprint" is 350 KB. Is that the same as "memory usage"?

The size of the executable is about 90 KB, the necessary 12Ghosts.dll has 150 KB, and the heap usage should be at about 100 KB. So the memory footprint, that is, the memory 12-Timer really allocates for the program code and internal variables, is about 350 KB.

What is rather confusing, if you have ever looked at the "Memory Usage" counter, for example, in the System Monitor or in the NT Task Manager, is that you actually find what usually is called the "Working Set". The size of the Working Set will change over time and is related to overall system activity.

(Quotation from Windows Help: "Working Set is the current number of bytes in the Working Set of this process. The Working Set is the set of memory pages touched recently by the threads in the process. If free memory in the computer is above a threshold, pages are left in the Working Set of a process even if they are not in use.

When free memory falls below a threshold, pages are trimmed from Working Sets. If they are needed they will then be soft-faulted back into the Working Set before they leave main memory.")

In other words, 12-Timer may, for example, be assigned 3 MB of memory, but it doesn't use all of it. You have 128 MB main memory and only a few programs running. You still have free main memory left, that's why Windows decideds to let 12-Timer claim the unused 2.7 MB memory for now. But if main memory becomes shorter, the 2.7 MB will be freed up and can be used by other programs. Then, and only then, the "Memory Usage" will indeed display 350 KB for 12-Timer, the memory that really is in use.

The Working Set is not the memory that the application really needs! You can verify this: in Performance Monitor add counters for Private Bytes, Virtual Bytes, and Page File Bytes for the process object, instance "12timer". These values should not change, even if you set up several timers, recurring once per second.

Then add a counter for the Working Set and observe it while there is a great deal of timer activity. The Working Set should grow a little. Now start all the programs you can find to use up memory, your office suite, browsers, editors, compilers - really big programs. Then the Working Set for 12-Timer should shrink to 350 KB because the pages not in use are suddenly needed and are therefore freed up.

With no timer activity for a long period, the Memory Usage (= *Current* Working Set) should gradually go down to 350 KB (= *Minimal* Working Set). That is what we call the memory footprint.

Where can I find out more about the moon?

Well, how much more? You'll find anything for example at www.yahoo.com. Try searching for "astronomy"! You should find several observatories with rich explanations, the NASA site, the Java UTC atomic clock, astronomic software, NTP connectors, and everything else.

The displayed counters are the age of moon in days and phase of moon in percent. 0d 0h 0m 0s equals New Moon. The average moon cycle varies between 29.2 and 29.9 days, so Full Moon is at about 14d 18h.

Timer Q&A Options

Contents Order FAQ HowTo Reviews 12Ghosts.com E-mail Contact

12Ghosts - The "Seven Dwarves" for Windows.®

12Ghosts Timer

Timer Registry Options

Timer Command Line Parameters

There is just one command line option, /s for silent mode. This keeps the Timer from showing the Timer settings window after start. /s is used in the startup shortcut automatically.

If you don't specify any command line parameter, and Timer is already running, this command line option will cause the running instance to re-check all timer sets. This may be useful, for example, after changing registry values. Timer will re-read all registry values by just starting it again.

12-Timer supports the registry value **StopTimerAfterNextStart**. You can launch a program at a certain time and then stop 12-Timer.

For example, you may want to execute a program from boot-up and set 12-Timer to run this program 30 seconds after startup. Then 12-Timer could itself stop so that no other program were active except for the one started just started.

To activate this stopping feature in 12-Timer, you need to add the DWORD value "StopTimerAfterNextStart" at "HKEY_CURRENT_USER\Software\PACT Software\Timer" and set it to 1. Just copy the following .reg snippet into a new text file (without the ---

lines) and rename it to **StopTimer.reg**. To disable the option use the second snippet. Don't forget an additional carriage-return (empty line) at the end of the .reg file, otherwise RegEdit doesn't accept it. You can use /s for a silent import.

```
--- StopTimerNext.reg ---------------------------------
REGEDIT4

[HKEY_CURRENT_USER\Software\PACT Software\Timer]
"StopTimerAfterNextStart"=dword:00000001

-------------------------------------------------------
```

```
--- KeepTimerRunning.reg ------------------------------
REGEDIT4

[HKEY_CURRENT_USER\Software\PACT Software\Timer]
"StopTimerAfterNextStart"=dword:00000000

-------------------------------------------------------
```

Timer Registry Values

ExeDescription	Up to 100 characters.
ExePath	Up to 255 characters.
ExeParameters	Up to 255 characters.
ExeWorkDir	Up to 255 characters.
StartNextDate	'yyyy/MM/dd HH:mm:ss', 24-hour format, leading zero.
StartEverySec	recurring time in seconds, 0 = not recurring (if StartNextDate empty, then after logon takes precedence).
StartSecAfterBoot	If not 0, this value will be processed, and this action takes precedence. (boot = launch of Timer = after logon if in Startup).
StartActivated	1=activated, anything else deactivated.
StartDeleteAfter	1=delete after starting once, else only deactivate.
StartWorkingOnly	1=recurring on working days only, that is, Monday through Friday, otherwise

	runs every day.
StartBell	1=play sound on action, otherwise no sound.
StartNotAfter	'yyyy/MM/dd HH:mm:ss', 24-hour format, leading zero.
WindowState	0=Normal, 1=Min, 2=Max, 3=No Activate, 4=Min No Activate, 5=Hidden.
PriorityLevel	0=Realtime, 1=High, 2=Normal, 3=Idle.

Example Registry File

Here's an example registry file that you can simply double-click in order to import the values into the registry. Copy the following lines and paste them into a new text document. Save it with the extension *.reg. After changing values, just execute 12timer.exe again (you don't need to stop it) in order to import the new settings.

A good way of using this .reg file is to call up it from your program or a batch file. You can turn timers on and off just by specifying StartActivated=0 or 1, for example. Or you can call up such a .reg file from a batch that was started by the Timer itself and change the start parameters on the fly.

```
---------------------------------------------------------------------------
REGEDIT4

[HKEY_CURRENT_USER\Software\PACT Software\Timer\myNewTimer]
"ExeDescription"="This is an example reminder"
"ExePath"=""
"ExeParameters"=""
"ExeWorkDir"=""
"StartNextDate"="1998/03/12 10:20:00"
"StartEverySec"=dword:00000000
"StartSecAfterBoot"=dword:00000000
"StartActivated"=dword:00000001
"StartDeleteAfter"=dword:00000000
"StartWorkingOnly"=dword:00000000
"StartBell"=dword:00000001
"StartNotAfter"=""
"WindowState"=dword:00000000
"PriorityLevel"=dword:00000002

---------------------------------------------------------------------------
```

Programming to the Registry

If you feel comfortable programming to the registry, you'll certainly want to use the

following functions.

Create a new subkey below the Timer registry key. Do not use a key name starting with *T*, *Example*, or *Default*. For example:

HKEY_CURRENT_USER\Software\PACT Software\Timer\MyTimerSet001\

In this key, set all or just the values you need. **Just execute 12timer.exe again to import the new settings.**

To do just that, here is a ready to use C source code program. 1) you find the type definition of a timer set, 2) the function to create a new timer set, 3) the function to write the timer set to the registry, and 4) a function to find the install path and execute 12timer.exe to get the newly added subkey imported.

From a Visual Basic program you can also access the registry by declaring the WINAPI functions RegCreateKeyEx, RegSetValueEx, and RegCloseKey.

```
// --- Example on how to init and save a new timer -----

// The examples provided here are not guaranteed to work. No warranties, no liability, no
guaranty for fitness of code.

// Copyright © 1993-2000 12Ghosts, Inc. on all examples and materials. All rights reserved.

// --- Declarations for 12Ghosts Timer -----

#define APPREGKEY    ("Software\\PACT Software\\Timer")

typedef struct _TimerSet {    // ts    set for one timer information
        char SubKeyName[MAX_PATH];
        char ExeDescription[101];         // max 100 (+ \0)
        char ExePath[MAX_PATH];
        char ExeParameters[MAX_PATH];
        char ExeWorkDir[MAX_PATH];
        char StartNextDate[20];         // 1998/03/12 10:20:00    max 20 (= 19 + \0)
        char StartNotAfter[20];
        UINT StartEverySec;
        UINT StartSecAfterBoot;
        DWORD StartActivated;         // default: 1 = yes
        DWORD StartDeleteAfter;         // default: 0 = no, only deactivate
        DWORD StartWorkingOnly;         // default: 0 = run every day
        DWORD StartBell;         // default: 1 = yes, play sound
        DWORD WindowState;         // 0-5         default: 0, normal
        DWORD PriorityLevel;         // 0-3         default: 2, normal
} TIMERSET;

void main();
BOOL SetTimerSet(TIMERSET *pts);
void Start12timer();


// --- main -----------------------------
void main()
{
TIMERSET ts;
SYSTEMTIME st;

    lstrcpy(ts.SubKeyName, "MyTimerSet0001");
    lstrcpy(ts.ExeDescription, "Test Timer");
    lstrcpy(ts.ExePath, "Notepad.exe");
    ts.ExeParameters[0] = 0;
    ts.ExeWorkDir[0] = 0;
    ts.StartEverySec = 0;
    ts.StartSecAfterBoot = 0;
    ts.StartActivated = 1;         // yes
    ts.StartDeleteAfter = 0;         // no
```

```
    ts.StartWorkingOnly = 0;         // no
    ts.StartBell = 1;         // yes
    ts.StartNotAfter[0] = 0;
    ts.WindowState = 0;        // 0-5    default: 0
    ts.PriorityLevel = 2;        // 0-3    default: 2

    GetLocalTime(&st);
    st.wMinute += 5;
    if(st.wMinute > 59)
    {
        st.wMinute -= 60;
        st.wHour++;    // and so on...
    }
    wsprintf(ts.StartNextDate, "%04i/%02i/%02i %02i:%02i:%02i", st.wYear, st.wMonth, st.wDay,
st.wHour, st.wMinute, st.wSecond);

    SetTimerSet(&ts);    // save to reg
    Start12timer();       // restart Timer

    return;
}

// --- save timer to registry --------------------
BOOL SetTimerSet(TIMERSET *pts)
{
HKEY hkNewKey;
char szFullKeyName[MAX_PATH];
DWORD dwDummy;
DWORD dwReturn;

    lstrcpy(szFullKeyName, APPREGKEY);
    lstrcat(szFullKeyName, "\\");
    lstrcat(szFullKeyName, pts->SubKeyName);

    // Open/Create Key for write access
    dwReturn = RegCreateKeyEx(HKEY_CURRENT_USER, szFullKeyName, 0, 0, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &hkNewKey, &dwDummy);
    if(ERROR_SUCCESS != dwReturn)
    {
        SetLastError(dwReturn);
        myErr("Error writing to registry! Can not open application key.");
        return FALSE;
    }

    // Set SZs
    RegSetValueEx(hkNewKey, "ExeDescription", 0, REG_SZ, (CONST BYTE *) pts->ExeDescription,
lstrlen(pts->ExeDescription) + 1);
    RegSetValueEx(hkNewKey, "ExePath", 0, REG_SZ, (CONST BYTE *) pts->ExePath, lstrlen(pts-
>ExePath) + 1);
    RegSetValueEx(hkNewKey, "ExeParameters", 0, REG_SZ, (CONST BYTE *) pts->ExeParameters,
lstrlen(pts->ExeParameters) + 1);
    RegSetValueEx(hkNewKey, "ExeWorkDir", 0, REG_SZ, (CONST BYTE *) pts->ExeWorkDir,
lstrlen(pts->ExeWorkDir) + 1);
    RegSetValueEx(hkNewKey, "StartNextDate", 0, REG_SZ, (CONST BYTE *) pts->StartNextDate,
lstrlen(pts->StartNextDate) + 1);
    RegSetValueEx(hkNewKey, "StartNotAfter", 0, REG_SZ, (CONST BYTE *) pts->StartNotAfter,
lstrlen(pts->StartNotAfter) + 1);

    // Set DWORDs
    RegSetValueEx(hkNewKey, "StartEverySec", 0, REG_DWORD, (CONST BYTE *) &pts->StartEverySec,
sizeof(DWORD));
    RegSetValueEx(hkNewKey, "StartSecAfterBoot", 0, REG_DWORD, (CONST BYTE *) &pts-
>StartSecAfterBoot, sizeof(DWORD));
    RegSetValueEx(hkNewKey, "StartActivated", 0, REG_DWORD, (CONST BYTE *) &pts-
>StartActivated, sizeof(DWORD));
    RegSetValueEx(hkNewKey, "StartDeleteAfter", 0, REG_DWORD, (CONST BYTE *) &pts-
>StartDeleteAfter, sizeof(DWORD));
    RegSetValueEx(hkNewKey, "StartWorkingOnly", 0, REG_DWORD, (CONST BYTE *) &pts-
>StartWorkingOnly, sizeof(DWORD));
    RegSetValueEx(hkNewKey, "StartBell", 0, REG_DWORD, (CONST BYTE *) &pts->StartBell,
sizeof(DWORD));
```

```
    RegSetValueEx(hkNewKey, "WindowState", 0, REG_DWORD, (CONST BYTE *) &pts->WindowState,
sizeof(DWORD));
    RegSetValueEx(hkNewKey, "PriorityLevel", 0, REG_DWORD, (CONST BYTE *) &pts->PriorityLevel,
sizeof(DWORD));

    RegCloseKey(hkNewKey);

    return TRUE;
}


// --- Find InstallPath and start 12timer.exe -----------
void Start12timer()
{
    HKEY hKey;
    DWORD dwDummy = MAX_PATH;
    char pszInstallPath[MAX_PATH] = {0};
    char pszProgPath[MAX_PATH] = {0};
    char pszTemp[MAX_PATH] = {0};
    GetTempPath(MAX_PATH, pszTemp);

    // get install path from Registry
    if(ERROR_SUCCESS != RegOpenKeyEx(HKEY_LOCAL_MACHINE, APPREGKEY, 0, KEY_QUERY_VALUE,
&hKey))
        return;

    if( ERROR_SUCCESS == RegQueryValueEx(hKey, "InstallPath", NULL, NULL, (LPBYTE)
pszInstallPath, &dwDummy) )
    {
        lstrcpy(pszProgPath, pszInstallPath);
        lstrcat(pszProgPath, "\\12timer.exe");
        ShellExecute(NULL, NULL, pszProgPath, NULL, pszTemp, SW_SHOWNORMAL);
    }

    RegCloseKey(hKey);
    return;
}
// -------------------------------------------
```

Timer Q&A Options

Contents Order FAQ HowTo Reviews 12Ghosts.com E-mail Contact

Our computers run on renewable energy only.

12Ghosts - The "Seven Dwarves" for Windows.®