Alarmist

Mathew Wilson

COLLABORATORS

	TITLE : Alarmist		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Mathew Wilson	August 24, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Alarmist

1.1 Alarmist

Welcome to the Alarmist v1.10 User Guide.

~~~~~~

1.2 About Alarmist.

About Alarmist.

Alarmist is a multiple event alarm clock. Most alarm clocks only handle one entry. This isn't too usefull, especially if your'e a busy person. Alarmist can handle any number of alarms and warn you appropriately when they occur with an optional snooze
~~~~~~

and pre-warn function. On top of this it keeps a log of all past and occuring events. It also has the added bonus of a flexible Workbench clock and the ability to display any number of clocks from around the world.

Make sure you take the time to read the entire Alarmist documentation as there are little things here and there that make life easier 8).

Alarmist can be used alone but it was designed to function along side the On-Time© schedule manager program by the same author. Alarmist will load On-Time schedule files and alert the user of any future, past or reminder events. Alarmist is only concerned with events for the current day whereas On-Time deals with events on a year-round basis. So they are best used as a team.

Alarmist is a MUI(Magic User Interface) application. This means you need the MUI files/library to use Alarmist. Version 3.3 of MUI is required. I chose to develop using MUI because of its flexibility and easy of use, both for the user and programmer. MUI is hopefully the direction of the future. MUI is available as a reduced product on the Public Domain, but if you wish to have full control, and support a great product that supports the continuing life of the Amiga, then register for the full version NOW!.

Alarmist© is classified FreeWare. See Disclaimer .

Mathew Wilson

Contact me at:

Box 6022, Halifax Street Post Office

Adelaide SA, 5000

AUSTRALIA

At present I can also be e-mailed via David Cottrell

dcottrel@adam.com.au

1.3 Using the Program.

Using the Program.

Alarmist is a standard Amiga Commodity (although most MUI programs are commodities by default). This means that it can load and hide itself from the user and be called up instantly at the click of a key. If you require further information on Commodity applications then please consult your Amiga Workbench manual.

The program uses standard tooltypes for settings. These are parameters set in the programs icon, from Workbench, that it looks for at run-time. These settings are generally global and all Alarm event information is saved to a separate file.

The program's interface consists of a main window with a list-view showing the current Alarms and 6 buttons. The button labelled 'Config' will open another window with a page-group allowing you to edit the program's configuration. A window called the Event Log Window that holds a list of messages that the program gives you. Another Window called the World Clock Window (optional) shows clocks from around the world, updating every minute.

If you are going to have Alarmist run programs as if they were run from Workbench, you will need Stephan Becker's 'WBStart-Handler.' The full archive can be obtained from Ami-net - 'WBSTART1_4.LHA'

If you are using ToolManager, also by Stephan Becker, then you will already have the required handler installed.

Also see Special Note About Iconification .

1.4 The Alarm List.

The Alarm List.

Alarmist handles multiple alarm events which appear summarised in the Event list which appears in the main window. You can see at a glance what events are lined up for the day.

To create a new event just click on the "New" gadget at the bottom of the list. The Edit Window will open and you can create an event.

You may alter the settings of each event by activating the entry in the list and clicking on "Edit." This will bring up the same edit window. Alternatively you could just just double-click on the event to edit it.

You can easily remove events from the list by selecting all the events you want to delete and clicking the "Remove" button.

The button marked "Toggle" allows you to change the state of events in the list. When an event is marked "off", it means that it is suspended and will NOT cause an alarm at its given time!

In the list you will see each event displayed with its Type, Time, Active State and its Name.

The 'Type' is indicated by either an 'a' or 'o', indicating where the event was created. Alarmist events ('a') are ones that have been created within Alarmist and On-Time events ('o') are ones imported automatically from within On-Time .

At the top of the window you will see the text "Alarmist" with either "no entries" or 3 numbers in the format '#:# of #.' The first number is the currently active (not necessarily the next Alarm to occur) entry in the event list. The second number after the colon is the number of selected entries in the list. This refers to the multi-selection capability of the MUI-List (refer to MUI documentation). The last number after the word "of" is the total number of Alarm events in the list.

A star (asterix) will appear after the window title text to indicate some changes have been made to the Alarm list.

If at any time you wish to make the next Alarm to occur the active entry in the list just press the DELETE key.

You may drag-and-drop icons onto any of the Alarmist windows or its app-icon (when iconified). This is performed via a User-Configurable ARexx Script .

1.5 Drag and Drop Script

Drag and Drop Script.

When an Icon is dropped onto any of the Main Alarmist windows an ARexx script named 'Alarmist_DropObj.rexx' is called with its first argument being the name of the File that was dropped. You must indicate where this script is by manually setting the tooltype 'DROP_SCRIPT' in the Alarmist program icon.

This method enables the user to specify exactly what is to be done with files that are dropped onto Alarmist. The script that is installed with Alarmist takes this filename and creates a new Alarm with the Execute string set to the file name of the item dropped. It then proceeds to open the Edit Window allowing you to make a few adjustments. This is useful if you need to run a program to remind yourself of something coming up, but of course you may tailor the script for your own specific use.

Like all scripts that are run from Alarmist, the invoked script has its host address set to the Alarmist port automatically. This way the script will know exactly who to send its messages to (see example).

(You may only run one version of Alarmist at one time, so the port will always be 'ALARMIST'.)

1.6 The Alarm Edit Window.

The Alarm Event Edit Window.

This window consists of a number of gadgets for editing Alarmist events. At the bottom of the window there are two buttons 'Accept' and 'Cancel'. The first and last button allow you to accept or cancel changes and revert back to the original settings.

If you have just created an Alarm using the "New" function then this Alarm will take on the global defaults set in "ENV:Alarmist.default."

*NOTE: This window contains 'Context Menus.' One menu (right mouse button over the window) allows you to perform operations on all of the settings gadgets in the window. The other menu is called over any of the settings gadgets and only effects them individually. They enable you load to and save all or individual default settings for the gadgets.

There are three main areas for each Alarmist event.

1. The Event Name

This is simply a line of text that identifies your Alarm. It may contain up to 128 characters. Try and use simple but easily identifiable names. This enables you to see at a glance what the event is without having to read a short story. eg. If you are reminding yourself to watch a certain television program then you could perhaps use an event name of "TV: <program name>." instead of "Time to watch <program name> on TV."

Pre-Warn

Every Alarm event can be given a pre-warning. This means that you will receive a warning about the event before its official time. See Warn Events for information on global settings. For each Alarm you may set a Warn type of 'Global', 'Local' or turn it off via the Warn cycle gadget. The Globol setting causes the Warn event to use the event details set in the configuration. Using the Local settings means that the Warn event will use the settings of the current Alarm.

To clarify:

If you don't want a pre-warn reminder for an Alarm then set 'Warn' to 'OFF'.

If you would like to be reminded of an event using the Warn Event settings in the configuration window then set 'Warn' to 'Global'.

If you want the Warn event to consist of the actual Alarm settings then set 'Warn to 'Local'.

2. The Alarm

This is the core element of the event. The Alarm time is represented by a slider which contains the actual time in its knob-gadget. It should be pointed out here that Alarmist events are minute accurate only. It seemed pointless and trivial to have Alarm times accurate to the second.

To the right of the Alarm tiem slider is a checkmark labelled 'Log Popup.'

This causes the Event Log Window to popup when the Alarm occurs. If you have an Alarm that just simply sends out an ARexx command then you may not want the Log Window to open. Just leave the checkmark deselected and the window won't open. Alarms are always entered into the log regardless of this switch.

Underneath is the actual Alarm event settings. There is an 'Exec' string gadget which contains the event to be executed. To the left of it is a cycle gadget which controls how the Exec string is executed:

CLI - The string will be executed just as though you had typed it into CLI/Shell (ie. ignores any associated icons.)

WB - Stephan Becker's WBStart-Handler will be used to execute the string as though it were being run from Workbench (ie. associated icon is taken into account.)

ARexx - The string will be be treated as an ARexx command - it will be sent to AREXX:. Therefore if you had typed in the name of an ARexx script, it would be executed. If you wanted to send a message to Alarmist then you must tell ARexx to Address Alarmist first. eg. "Address Alarmist appear" (Remember to place the string in double quotes if it contains more than one command!)

You might use your favourite sample player to play an appropriate sample or you could use anything. Included with Alarmist is a program called Talk by the same author. This program acts very much like the old Commodore "Say" program but with a load of extra functions like the ability to speak the current time. This way you could have Alarmist tell you about an alarm by speaking the time and a description of the event. A novel way to remind you of something.

A program called 'Sam' is provided with Alarmist that lets you play audio samples from Workbench.

Using the "Test" button will enable you to check your settings to see if they work ok. This is advisable because incorrect settings could cause you to miss an event although Alarmist will generally alert you in other ways as well.

3. The Snooze Function

This is one of the major advantages of using Alarmist. Setting a Snooze on an event means that you can be reminded after an event in much the same way as the original alarm. This can happen a certain number of times after the original Alarm has occurred.

The settings look much the same as the Alarm Time settings but instead of using Hours and Minutes, it uses Minutes and Seconds. This refers to the time between Snooze events. For example; you may have the Alarm set to remind you to watch a TV program but you are in the habit of ignoring or forgetting the Alarm after it has gone off. So you set a snooze time for 15 seconds. Now after the initial Alarm event, the Snooze event will occur every 15 seconds for a set number of times.

If a Snooze event occurs and there is no Snooze Exec string, then the original Alarm Event is used.

The repeat times for a Snooze event can be set with the numeric button labelled 'Snoozes.' The range is from 1 to 99 and INFINITE. The infinite setting is handy for those times when you never want to forget something. This means that you must have some way of turning off the Snooze events or they will go on forever and become extremely hazardous to your mental health 8). Well Alarmist uses a standard Commodity Hot-Key for this. Look under Config Page for more information on how to set this up.

Just simply hit the Hot-Key and the Snooze event ceases. It will cease no matter what the snooze repeat value is. When a Snooze event is manually stopped, it will appear in the Event Log.

I have a friend who uses Alarmist to wake up in the morning. One Alarm event is set that has a Snooze time of 30 seconds on an Infinite setting. After 5 minutes another Alarm occurs that has a Snooze time of 5 seconds set for infinity. This means you must get out of bed and hit the correct Hot-Key sequence before the Alarm/Snooze stops.

* This brings us to an important point. If an Alarm's Snooze is still active and another Alarm event occurs then the first snooze will *stop* automatically. It would be too cumbersome to manage multiple Snooze events. So in my previous example of two events set with Snoozes, the second event that occurs 5 minutes after the first will stop the original Snooze event that was set for Infinity.

You may have any number of Alarm Edit Windows open at once. Try to remember though, each alarm that you are editing is still active. This means that if you are editing the next occuring alarm then it can still occur while the Alarm Edit Window is open. This practice may change in later versions. One possible answer could be to 'toggle' the active state of the Alarm before the Edit Window opens, and then re-activate it after it closes.

1.7 Alarmist Configuration.

Alarmist Configuration

The Configuration window may be accessed by selecting 'Config' from the main window or from its menus.

Clock Settings.

Event Settings.

Options.

Keys.

The user-definable ARexx Scripts are saved as ToolTypes in the Alarmist icon along with the rest of the global configuration (not Alarm events).

1.8 The Workbench Clock

The Workbench Clock.

An alarm clock program cannot be complete without a visual clock of some sort. Alarmist provides you with a Locale based clock string that appears on your Workbench screen.

Whenever there is a "Next Event" to occur, a star (Asterix) will appear after the clock text. This is a quick indication that at sometime or other an Alarm event will occur. If you double-click on the clock text it will be replaced with the time of the next Alarm along with the hours and minutes until the event. You will not be able to perform this function if there are no more events for the current day. A current List of Events could be made up of a mixture of On-Time and Alarmist events so we would have to recheck the OnTime event file for events tomorrow to accurately report information for that day. This would be time consuming and therefore this is left out.

On the Clock Page you will see a variety of options that allow you to customize your clock settings.

Update Time:

This refers to the time intervals between clock updates. This allows you to minimize CPU loading with larger Update Times. Alarmist will not load up the CPU a great deal but you can experiment with this setting if you like. It is important to note that this setting also affects when Alarmist checks for Alarm and Snooze Events. If you want your Alarm events to occur exactly on the minute then I suggest you use an Update Time of 1 second. An Update Time of 15 seconds and a Snooze time of 5 seconds doesn't make much sense either. It is recommended to leave the Update Time at around 1to 5 seconds. But you could change this, like most commands, via ARexx .

Clock Format:

This string gadget allows you to change what the Clock Display shows. The clock uses standard Locale date/time formatting. See Clock Formatting .

Just next to the Clock Format gadget is an unlabelled checkmark gadget. This is just an on/off switch for the clock. It doesn't deactivate Alarmist, it just stops the clock from being displayed.

Pen: Paper:

These two cycle gadgets let you change the colours of your clock.

Clock XY:

This checkmark allows you to lock the position of the clock display on the screen. If it is selected then the clock is locked to a certain position on the screen. If it is unselected then you may click on the clock text and move it round like you would a normal window. In fact the clock is a window; it just has all the normal window parts hidden. So you can move the clock around to any position on the screen and then lock it in position. The default position is on the Workbench menu bar just left of the screen cycle gadget. You may return the clock to this position by clicking on the Reset XY button.

Font:

This popup gadget allows you to specify what font the Clock Display will use. You may use the pop-up asl font requester or you can specify the font by typing in the name followed by a '/' and the size of the font. When you press return, Alarmist will attempt to load the font. If the font cannot be loaded or you have entered a 'blank' entry then it will revert to the default screen font.

NOTE: The fact that the Clock is actually a window means that Intuition will not be allowed to fully reset itself when you change a Workbench preference setting like ScreenMode unless the Clock Window closes. Some clock programs just simply blit (copy) the clock directly onto the Workbench screen. This means that it can easily disappear. For example; if it was on the screen menu bar and the a menu is used then it would disappear until next clock update. Since Alarmist uses a window for its clock then it will always be there unless another window covers it. So if you ever see the message "Intuition is attempting to reset the Workbench screen. Please close all windows, except drawers." and all windows appear closed then it is probably the Alarmist clock because it is still open. This can be easily resolved in a number of ways!

1 - Click on the Clock as you would a window to activate it and then press the ESCAPE key on the keyboard. The Clock will now disappear and the Workbench will be allowed to be reset.

2 - Pop-up the Alarmist interface and turn the clock off as shown above.

3 - Disable Alarmist via the standard Workbench Commodity Exchange program and the clock will also disappear.

4 - Send an ARexx Command to turn the clock display off.

Its a small price to pay for the clock to be visible all the time.

NOTE: If you have Stefan Becker's ScreenNotify Library installed then Alarmist will attempt to close its window when Intuition resets. This doesn't seem to be too efficient at the moment and you may get the standard requester asking you to close all windows. In this case just hit retry. It may happen more than once. This library does not work on all computer setups but it is very handy indeed!

Public Screen:

You may set the Public Screen which the Clock appears. This isn't too user friendly at the moment but will probably improve in later versions. You could write a script to use one of the freely available CLI programs that get the name of the frontmost public screen and send an ARexx command to Alarmist to change the clock screen.

1.9 Extra Event Settings

Extra Event Settings.

This page of settings deals with two extra type of events; Chime and Warn.

Chime Events.

Selecting the check mark enables the Chime. At the beginning of every hour the Chime function will execute the "Exec:" string. You may set the actual hours that the chime will function with the "From:" and "To:" slider gadgets underneath. These are set in 24 hour mode from 0 to 23 hours (12:00 AM - 11:00 PM).

There is a standard dragable Execution group here that contains a pop-file string gadget, an execution-type cycle gadget and a test button.

The supplied ARexx Script 'Alarmist_Chime.rexx' gives an example of using ARexx for a Chime Event.

Warn Events

Usually when a Time type event occurs it will give you no warning unless you change the Alarm time to just before the actual time of the event. This is one option but it may get confusing if you can't remember whether you set the Alarm for the actual time or just before it. So the Warn option gives you this facility.

It has the usual checkmark gadget to Activate it and the Exec string. Below this is the Pre-Warn time. This is the time (in minutes) that you would like to be reminded before each Alarm Event occurs. At the moment you will only be reminded *once* at the exact Pre-Warn time before the event. This may change in later versions but it has not been decided yet.

You may do tricky things by executing an ARexx Script on Warn Events that could do a variety of things like finding out what event will occur next.

There is a standard dragable Execution group here that contains a pop-file string gadget, an execution-type cycle gadget and a test button.

There are two types of Warn Events that can be set when editing or creating an Alarm event .

1.10 The Commodity Keys.

The Commodity Keys Page.

This page contains all the Commodity key strings for operation of Alarmist.

o "Pop Key" Hot-Key that brings up the Alarmist interface.

o "Stop Key" Hot-Key that allows you to stop a Snooze event.

o "Log Key" Hot-Key that opens and closes the Log Window .

o "World Clock Key" Hot-Key that opens and closes the World Clock Window .

They all take standard Commodity key description strings.

1.11 The Options Page.

The Options Page.

This is the final page that allows you set some preference options in Alarmist.

CX PopUp:

This switch causes the main window to 'pop up' when the program loads. This could be useful if you wish to scan the list for events upon loading.

Main Gadgets:

Some people may wish to leave the Main Window open. This way you will always see the current list of events by glancing at the window. These sort of people will most probably have a large monitor (eg. 21") and Workbench space "for lease" . The Main Gadgets switch lets you decide whether the buttons on the Main Window will be dispayed or not. Leaving them out will mean that the window won't require as much space, or you'll have more space for the Event List as it will be the only object.

Confirm Remove:

Activates user confirmation requests when Removing alarms from the list.

Prior alarm popup:

There are various occasions when the Log Window will open and become visible. If Alarmist loads and finds out that there have been some previous Alarms during the day then you may want it to popup the the Log Window to indicate this. This is optional an can be turned off with this checkmark. On other occasions such as Alarm events occurring, the Log Window will always open and become visible.

Clean log on reload:

Lets say you are running a bulletin board and therefore you leave your computer turned on all the time. If you are running Alarmist with On-Time© (eg. running on time 8) then you may end up with a huge list of events in the Log after a while and this can look messy. Well, with this option selected you can have the Log List cleared every time the event list(s) are loaded. See Clearing Event Log .

Daylight Savings:

This switch tells Alarmist that the time it reads from the computer is in Daylight Savings time. This means that the clock is one hour ahead of 'normal' local time. This setting is used when Alarmist calculates Greenwich Mean Time from the internal clock. See World Clock Window .

The above settings are saved as icon tooltypes via the "Save Config" function.

1.12 Running On-Time©

Using Alarmist© with On-Time©

As stated in the introduction, Alarmist was originally designed to operate along side the program On-Time. This way Alarmist can alert you of events that occur from day-to-day. It does this by reloading the standard On-Time list file "ENV:OnTime.list" at 12:00 AM (midnight) every morning (of course you must have your computer switched on 8-) and looks for any events for that new day and inserts them into its list. Alarmist will also alert you of Day events (refer to ON-Time documentation) and let you know of events that have reminders.

Together, Alarmist and On-Time make a great team. You can use Alarmist by itself or you could use it together with On-Time for maximum advantage. Lets say you had just switched on your computer and needed to do something but you had an engagement later on that you didn't want to miss. If it was a one-off event and didn't re-occur then you could just simply pop-up the Alarmist interface via the Hot-Key and enter the event. This way you wouldn't have to load On-Time and enter it indirectly. You could even make up an ARexx script to do most of the work for you (see example script provided).

It's all up to you how you manage the programs but they have been designed to be flexible.

Now you may ask the question "What happens if Alarmist loads the OnTime.list file and inserts an event and then I select 'Save' from Alarmist? Doesn't Alarmist save this new event along with the others in its own file 'ENV:Alarmist.list' ?" Well the answer is no! Alarmist is intelligent enough to make a note of On-Time events that it has inserted into its list and leaves them out of its own event file.

Everytime you load an Alarmist list it searches for the "ENV:OnTime.list" as well and if it is found then it is examined as well. This way you will always have an up-to-date list in Alarmist.

Alarmist uses one of the really powerful features of AmigaDOS - file notification. Whenever the "ENV:OnTime.list" gets changed Alarmist will get a message saying that it's time to reload the OnTime.list. This displays the true power of using these two programs together (and the Amiga).

Just a quick note. Make sure you read the On-Time documentation if you are going to use it. You don't need to put On-Time in your WBStartup Drawer or your Startup-Sequence!! Alarmist goes in the WBStartup drawer because it is a dedicated commodity and needs to remain resident in memory. On-Time should be loaded only when needed. Although if you've used MUI applications before then you'll know that they are usually all Commodities by default. It might sound confusing but just remember to keep Alarmist in your WBStartup drawer and On-Time somewhere else that is easy to get to.

1.13 Loading and Saving Lists

Loading and Saving Lists

It is important to realise where and how the default Alarmist list is saved and how On-Time events are treated.

By default Alarmist acts like a standard preferences program, in that it saves its list file to the ENV: and ENVARC: drawers. When you select "USE" from the main window, the current list will be saved to the ENV: drawer (in RAM:Env). Selecting "SAVE" will save the list to the ENV: and ENVARC: (SYS:Prefs/Env-Archive) drawers. For more information on this - see your Amiga Workbench manuals.

You may choose to load an Alarmist list from a different location either from the 'Project' - 'Open' option or via an ARexx command. NOTE: When you perform a standard 'Load', the global OnTime list will be examined.

From the 'Project' menu and from ARexx there is another way of loading Alarmist lists. This is using the 'Merge' function/option. This will open a specific Alarmist file and load its contents into the current list without clearing the previous list. So all the merging takes place in the list and nothing is saved until you issue that command. NOTE: When you are merge-loading an Alarmist file, the global On-Time list will *NOT* be examined. To do this would cause problems with the previous list in memory.

NOTE: When saving a list you must remember that *ONLY* Alarmist events will be saved (those events created with Alarmist and not imported from On-Time). So if you alter an On-Time event from within Alarmist then select 'Save', the changes are not kept. To save the On-Time events would be meaningless as they are created and changed from their parent program. Also if you were to reload the event list from within Alarmist or simply perform a 'Last Saved' command then the On-Time event(s) will be restored to its original state as saved in the On-Time event file. Altering On-Time events from within Alarmist should be only treated as a temporary measure. Remember that the event may not be present in the Alarmist list the next day as Alarmist will only import On-Time events that occur on the present day. And the list gets reloaded at midnight.

Now just a small mention of something. Don't confuse the 'SAVE' and 'USE' buttons on the main window with the 'Load Config' and 'Save Config' options from the 'Settings' menu. The latter refers only to the Clock, Chime and general settings which get saved as Icon tooltypes. The buttons on the main window are used only in relation to the current Alarm List in memory. So don't go hitting the 'SAVE' button when you want to save the clock settings!

1.14 The Event Log Window

The Event Log Window.

For further enhancement in keeping track of Alarms, a log window was added. This is a separate window that contains a read-only list of messages. Everytime an event of some kind occurs within Alarmist a message will be added into the Event Log List.

The first message in the list will always indicate when Alarmist was activated. This is the exact time that Alarmist was loaded.

Various messages such as Alarm events and Snooze stops are recorded in the list. If you are using On-Time© with Alarmist© then various other messages will be displayed.

Whenever Alarmist is activated or deactivated via the Workbench Commodity Exchange program then this will also we record in the log.

The Log List can be cleared by using the 'Clear Log' function from the menu.

The Log Window can be made to appear and disappear via a Hot-Key. See Config Options .

1.15 Clearing the Event Log List

Clearing the Event Log List.

This is a handy way to clear all the messages in the Log List. Sometimes they can build up and become messy. If you wish you can have the list cleared everytime an event file is loaded. See Config Page.

All the lines of messages will be deleted except for one. The first entry will remain because it indicates when Alarmist was first activated (when it loaded).

1.16 The World Clock Window.

The World Clock Window.

The World Clock Window is a seperate entity. Like the Log Window, it can be opened and closed via a Hot-Key . It contains a list of user-definable clocks from around the world that are updated every minute.

To define a list of clocks, do the following:

o Load your favourite text editor and edit a number of lines, each comprised of a City Name and its Local deviation from Greenwich Mean Time (GMT), in minutes.

eg. Adelaide,-570

Chicago,360

London,0

The City Names can be up to 30 characters long and *must* be seperated from their GMT differences by a comma (',').

o Save this file in the *same* directory as the Alarmist program, which will normally be 'SYS:WBStartup.' The file *must* be called 'Alarmist_World.data' for Alarmist to recognise it!

Now, when Alarmist is first run, it will look for this data file in the directory it started from and load the data from the list. You may have any number of entries in the file but remember that it takes more time to update many entries, but it's not too significant as it only happens once every minute. If the World Clock Window is not open then no update will take place as that would be pointless. The clocks are only updated when they are displayed, and then, once every minute.

Alarmist comes with an example file full of places from around the world. You may use this or tailor it to your own preferences. Each clock entry in the file will not appear in alphabetical order in the World Clock Window. Each entry is inserted in the order that it was loaded from the file. This is so you can customise the ordering to your own tastes. Of course you could sort them alphabetically by using the AmigaDOS 'Sort' command on the 'Alarmist_World.data' text file (See AmigaDOS manual).

In the World Clock Window, each Clock will appear next to its respective name. If the time is prefixed with a '-' this means that it's the day before and a '+' indicates the day after your local day.

Alarmist first calculates your local GMT using the time difference stored by the System Locale settings. Then each Clock is displayed in reference to its own deviation from GMT. So make sure you have set the System Locale Preferences correctly.

It is important to let Alarmist know if you are in Daylight Savings time by setting the switch in the Configuration . If indicated, then Alarmist will subtract one hour from the computer's clock (as well as the local GMT difference) before calculating each city's time. Every country/state has its own Daylight Savings time throughout the year. So keep this in mind because a City's time might be displayed as 10:20AM whereas in reality it is 11:20AM because they are in Daylight Savings time. Nothing is ever simple, is it? 8).

1.17 Setting the Text Format for the Clock

Setting the Text Format for the Clock

The Workbench clock built into Alarmist uses the standard Locale time formatting facilities built into the Amiga's operating system. See your Workbench manual if you wish to know more about Locale.

Alarmist comes with a standard clock format which should suit most people but if you're into changing things then you can use the following parameters to make your clock look different.

%a - abbreviated weekday name

%A - weekday name

%b - abbreviated month name

%B - month name

%c - same as "%a %b %d %H:%M:%S %Y"

%C - same as "%a %b %e %T %Z %Y"

%d - day number with leading 0s

%D - same as "%m/%d/%y"

%e - day number with leading spaces

%h - abbreviated month name

%H - hour using 24-hour style with leading 0s

%I - hour using 12-hour style with leading 0s

%j - julian date

%m - month number with leading 0s

%M - the number of minutes with leading 0s

%n - insert a linefeed

%p - AM or PM strings

%q - hour using 24-hour style

%Q - hour using 12-hour style

%r - same as "%I:%M:%S %p"

%R - same as "%H:%M"

%S - number of seconds with leadings 0s

%t - insert a tab character

%T - same as "%H:%M:%S"

%U - week number, taking Sunday as first day of week

%w - weekday number

%W - week number, taking Monday as first day of week

%x - same as "%m/%d/%y"

%X - same as "%H:%M:%S"

%y - year using two digits with leading 0s

%Y - year using four digits with leading 0s

The default setting is %A %d-%b-%y, %Q:%M:%S %p

1.18 A Special Note About Iconification.

A Special Note About Iconification.

At present Alarmist does not adhere to MUI Iconify directives. Because you need to open one window while another is closed, it is not good enough to just simply Iconify the application as you would any other. If you do, then you can be guaranteed that no windows will open (like the Event Log Window) until the application is uniconified. For this reason Alarmist should never be iconified in the traditional MUI sense. It is therefore advisable to turn off the Iconify Gadget in Alarmist using the MUI preferences program. When you close the Main Window with the close gadget your program will be 'iconified'. This is probably not a good thing to do, but what else can I do? Shoot me down in flames if you like. Burn me to a stake 8).

1.19 Why not register today

This application uses

MUI - MagicUserInterface

(c) Copyright 1993/94 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With

the aid of a preferences program, the user of an application has the

ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing

lots of examples and more information about registration please look for

a file called "muiXXusr.lha" (XX means the latest version number) on

your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US$ 20.-

to

Stefan Stuntz

Eduard-Spranger-Straße 7

80935 München

GERMANY

1.20 Those Lovely Things

Known Bugs in Alarmist

- I originally didn't have a separate class for the guts of the time keeping but I went to the trouble of converting it all over to such a system and then found out that it wouldn't do. That'd be right! So then I went back to plastering signal bits together by hand 8) . Yeah, Alarmist really needs the clock factory to be separate from anything MUI looking because it has to be operational at all times. When the application goes on vacation into the world of sleepless signals the user has to be assured that the job is being done, e.g. the Alarm time-keeping is actually happening. The application should never get iconified anyway (see Special Note). It would have been lovely to have a MUI clock window, even just for the sake of not having to open the ScreenNotify.library myself (slack).

- Alarmist will reset itself at midnight when it loads-up the day's events. This means that any Snooze events will cease...you have to wake up sooner or later 8).

- Snooze events will also be deactivated when accepting changes on any Alarm in the Alarm Edit Window or adding a new event. When something changes in the Event list or something new is added, Alarmist has to recheck everything, thereby clearing the current event status. I might get around this in the future sometime if I can think of a strategy that doesn't have any worse side-effects.

- Veritas est nimis involuta ingeniis hominum.

1.21 In the Future...

Maybe In the Future...

- Possible language catalogs (catalog definitions are a real bore 8). I'm part of the way there. All the Help bubble strings have definitions. Catalogs depend on whether anyone else is interested in my little insignificant programs apart from me 8).

- I could do a lot more things to the Clock Display to make it more interesting. If anyone has some code that they would like me to use then feel free to send it to me for consideration. Remember, this is Freeware.

One possible solution which could be interesting is to have a set of Public Classes. Each class/library would be a different type of clock and the user could select which one to use. This way you don't take up room in the main program with loads of code that isn't neccessary. An idea that I will look at in the future. Remembering, of course, that all MUI Public Classes have to be registered. Different MUI applications could make use of these classes as well, but not all of them would need a facility to display the next occuring alarm.

- Any suggestions?

Wish List

- Some ACTUAL Beta testers who actually use the program 8) (no joke).

1.22 Disclaimer.

The following notice applies to the program "Alarmist" and everything contained within its original distribution archive.

· The program was written by Mathew Wilson.

· The program is classified FreeWare, which means it is:

- Freely distributable as long as its original archive remains intact.

- Copyrighted by its author and you are NOT permitted to modify the program(s) and documentation in any possible way.

· No warranties are given. No responsibility is taken for any possible use or misuse of the program(s) by its users.

The Author has tried to prevent errors but he can't guarantee that it is 100% bug free. You therefore use this program and any other associated program at your own risk.

· This package must not be sold directly or indirectly without prior written consent given by the Author. Although, it may be placed on any media used in the distribution of free software without consent from the Author.

The MUI system is Copyright Stefan Stuntz.

1.23 History

· v1.10 - Second Release (27th March 1996)

- Fixed numerous Enforcer Hits 8).

- Functions-Log menu wasn't connected (oops).

- Settings-Create Icons menu wasn't updating after a Load Config.

- ARexx Menu update problem fixed.

- Chime and Warn execution groups now use the Execution Group Class that is used in the Alarm Edit Windows. This should have been done in the first release because it makes a load of sense! If your'e going to make classes then you should use them 8).

- Added 'LogMessage' ARexx command.

- Altered template for ARexx command GetField from 'NUMBER/N' to 'ENTRY/N/K,NUMBER/N'

- Altered numeric displays for some slider gadgets in the Config Window.

- Added a quick Startup stack check.

- Added template for CLI loading.

- Fixed CON: de-allocation for ARexx messages.

- Tampered with Exec Group drag & drop so you can't drag them onto themselves or other Exec Groups with the same attribute values.

- Added 'DROP_SCRIPT' icon tooltype.

v1.0 - Original Release (8th February 1996)

1.24 arexxscripts

User-Definable ARexx Scripts

Alarmist allows you to execute external ARexx scripts from within the program. In which case you must have ARexx operational on your system (refer Amiga manuals). Alarmist will not execute the scripts itself but tell ARexx to perform the scripts (standard procedure). The essential difference is that the default Host Port for the script will be Alarmist.

In the ARexx pull-down menu you will find the following items.

Run New Script...

A file requester will open. Select an ARexx script and close the requester. The file will be executed as an ARexx script.

Run Last Script

This calls the script previously invoked with the 'Run New Script' menu command. If no ARexx script exists then a file requester will open.

Install ARexx Script (1 to 4)...

These four menu items are where you can 'install' four different ARexx scripts. They can then be called quickly via the menu or menu shortcut key.

After selecting one of the four menu items, a file requester will open and you can select the script to install. The script will not be performed at this stage. Now, if you look at the menu item again, you will notice that the previous text has been replaced with the name of the file you selected. The file will appear without it's path name.

Clear Script...

This menu command allows you to remove one of the scripts you previously installed with the 'Install ARexx Script' command. A requester will appear, showing all the currently installed scripts. Just select the one to remove and it will dissappear from the ARexx pull-down menu. Of course the script itself will not be deleted, just the reference to it.

The four scripts installed will be saved as Tooltypes in the Alarmist icon when you save the Configuration (not the event list!).

You may invoke the execution of one of the four scripts via ARexx as well. See RX command .

1.25 Controlling Alarmist with ARexx.

Alarmist ARexx Command Set.

Only one version of Alarmist can be running at once and its ARexx port is ALARMIST

Apart from the standard MUI ARexx commands (see MUI Documentation), Alarmist has the following commands. Any errors will be reported in the RC variable (0 = success, any other value indicates an error).

See also: Special Note on Iconification .

You can install four ARexx scripts for easy reference as well.

- A -

About (NO ARGS)

Active ENTRY/N,NOACTIVE/S,ALARM/S

Appear (NO ARGS)

- C -

Chime STATE/N,TOGGLE/S

ChimePeriod MIN/N/K/A,MAX/N/K/A

ClearList (NO ARGS)

ClearLog (NO ARGS)

Clock STATE/N,TOGGLE/S

ClockFont FONTNAME/A,SIZE/N/A

ClockFormat FORMAT

ClockPubScreen NAME/A

ClockXY X/N/K,Y/N/K,RESET/S

- D -

Disappear (NO ARGS)

DoChime (NO ARGS)

DoWarn (NO ARGS)

- E -

EditWindow (NO ARGS)

- G -

GetActive (NO ARGS)

GetEntries (NO ARGS)

GetField ENTRY/N/K,NUMBER/N

GetNextAlarm (NOARGS)

- L -

LastSaved (NO ARGS)

Log STATE/N,TOGGLE/S

LogMessage SPACER/S,POPUP/S,MESSAGE/F

- M -

MainGadgets STATE/N,TOGGLE/S

- N -

New NAME/K/A,AHOUR/N/K/A,AMIN/N/K/A,LOGPOPUP/S,AEXECTYPE/N/K,AEXEC/K,SMIN/N/K,SSEC/N/K,STIMES/N/K,

- O -

Open FILE,MERGE/S

- R -

Remove ACTIVE/S

Restore (NO ARGS)

RX FILE,SCRIPT/N/K

- S -

Save (NO ARGS)

SaveAs FILE,ICON/S

- T -

Toggle STATE/N,TOGGLE/S

- U -

Update (NO ARGS)

UpdateTime SECONDS/N

Use (NO ARGS)

- W -

Warn STATE/N,TOGGLE/S

WarnTime MIN/N

WorldClock STATE/N,TOGGLE/S

1.26 rx_about

About

Purpose: Display the About requester.

Syntax : About (no parameters)

Example: about

1.27 rx_active

Active

Purpose: Set the active entry in the list.

Syntax : Active ENTRY/N,NOACTIVE/S,ALARM/S

ENTRY/N - Number of the entry in the list (1...n).

NOACTIVE/S - Flag to de-activate the list (eg. no entry active).

ALARM/S - Flag to make the next occuring Alarm active in the list.

Example: active 5

active alarm

1.28 rx_appear

Appear

Purpose: Cause the Main Window to 'show' itself.

(see Special Note on Iconification)

Use this command instead of the standard MUI 'Show' command.

Syntax : Appear (no parameters)

Example: appear

see also Disappear

1.29 rx_chime

Chime

Purpose: Turn the chime function on and off.

Syntax : Chime STATE/N,TOGGLE/S

STATE/N - A value of 0 will turn the chime off, whereas any other value will turn it on.

TOGGLE/S - Toggle the state of the chime on and off.

Example: chime state 0

1.30 rx_chimetime

ChimeTime

Purpose: Set the minimum and maximum times for the chime function.

Syntax : ChimePeriod MIN/N/K/A,MAX/N/K/A

MIN/N/K/A - Set the minimum hour. (0 - Max chime hour)

MAX/N/K/A - Set the maximum hour. (Min chime hour - 23)

Example: chimeperiod min 9 max 22

1.31 rx_clearlist

ClearList

Purpose: Remove all Alarm events from the current list. This includes all On-Time events.

Syntax : ClearList (no parameters)

Example: clearlist

1.32 rx_clearlog

ClearLog

Purpose: Clear the contents of the Event Log Window. All lines will be removed except for the first which always indicates when Alarmist was first activated.

Syntax : ClearLog (no parameters)

Example: clearlog

1.33 rx_clock

Clock

Purpose: Turn the Workbench clock display on and off. This has *no* effect on the internal clock. eg. Alarmist will still watchout for alarm events.

Syntax : Clock STATE/N,TOGGLE/S

STATE/N - A value of 0 will make the clock disappear, whereas any other value will make it appear.

TOGGLE/S - Flag to cause clock display to toggle on and off.

Example: clock state 1

1.34 rx_clockfont

ClockFont

Purpose: Change the font used in the Workbench clock display.

Syntax : ClockFont FONTNAME/A,SIZE/N/A

FONTNAME/A - Any valid font name. eg. ’Helvetica’

SIZE/N/A - Any font size.

Example: clockfont xhelvetica 11

If a font of the same name is found then it will result in a size of font that is closest to that specified.

If there is any problems then the default Screen Font is used.

1.35 rx_clockformat

ClockFormat

Purpose: Specify a new format string for the Workbench clock display.

Syntax : ClockFormat FORMAT

FORMAT Format string.

Example: clockformat "Wow look at the time: %T"

1.36 rx_clockpubscreen

ClockPubScreen

Purpose: Set the Public Screen for the Clock Window.

The Clock Winodw will close then re-open on the given Public Screen.

Syntax : Clock NAME/A

NAME/A - The name of the Public Screen that you wish the Clock Window to appear on.

Example: clockpubscreen mypubscreen

clockpubscreen workbench

1.37 rx_clockxy

ClockXY

Purpose: Change the location of the clock display on the screen.

Syntax : ClockXY X/N/K,Y/N/K,RESET/S

X/N/K - The x co-ordinate (0 - screen width - clock width). (value is validated)

Y/N/K - The y co-ordinate (0 - screen height - clock height). (value is validated)

RESET/S - Flag to cause clock to appear in its default position at the right side of the screen menu bar.

Example: clockxy x 100 y 150

clockxy reset

1.38 rx_disappear

Disappear

Purpose: Cause the Main Window to 'hide' itself. *SEE note on Iconification* refer to a above not about Alarmist iconification.

(see Special Note on Iconification)

Use this command instead of the standard MUI 'Hide' command.

Example: disappear

see also Appear

1.39 rx_dochime

DoChime

Purpose: Perform a Chime Event according to the current configuration settings.

Syntax : DoChime (no parameters)

Example: dochime

1.40 rx_dowarn

DoWarn

Purpose: Perform a Warn Event according to the current configuration settings.

Syntax : DoWarn (no parameters)

Example: dowarn

1.41 rx_editwindow

EditWindow

Purpose: Cause the Event Edit Window to popup.

This will only occur if there is an active entry in the list.

Syntax : EditWindow (no parameters)

Example: editwindow

1.42 rx_getactive

GetActive

Purpose: Return the number of the currently active entry in the list.

Syntax : GetActive (no parameters)

Example: getactive

active = result

1.43 rx_getentries

GetEntries

Purpose: Return the current number of entries in the list.

Syntax : GetEntries (no parameters)

Example: getentries

entries = result

1.44 rx_getfield

GetField

Purpose: Return the contents of a data field from a specific entry or the active entry in the list.

Syntax : GetField ENTRY/N/K,NUMBER/N

ENTRY/N/K - Number of the entry in the list (1 to Entries) to get info on. If you do not specify ENTRY then the current Active entry will be used.

NUMBER/N - The Number of the field to return.

The following are the field numbers relating to the information.

1 - Name of the event.

2 - Type of event (1 = Alarmist, 2 = On-Time)

3 - Alarm Hour (0 to 23 hours - 24 hour clock)

4 - Alarm Minute

5 - Alarm Status (0 = Non-Active, 1 = Active)

Example: getfield 2 /* get field number 2 of the active entry in the list */

Name = result

getfield entry 5 5 /* get field number 5 of list entry 5 */

status = result

1.45 rx_getnextalarm

GetNextAlarm

Purpose: Return the number of the next Alarm Event in the list.

The entry will not be automatically activated in the list.

Syntax : GetNextAlarm (no parameters)

Will return 0 for no event or 1 to n, for the number of the event.

Example: getnextalarm

next = result

1.46 rx_lastsaved

LastSaved

Purpose: Perform the normal LastSaved operation.

Restore the Alarm List to previous settings in ENVARC:

Of course the ENV:OnTime event file will be examined.

Syntax : LastSaved (no parameters)

Example: lastsaved

1.47 rx_load

Load

Purpose: Open a valid Alarmist list file.

Syntax : Load FILE,MERGE/S

FILE - Full name and path of the file to be loaded.

MERGE/S - Flag to cause file to be merged with current list in memory. Note: the OnTime list will not be examined when merging as this would disrupt the merging process.

Example: load "ram:testfile.list" merge

1.48 rx_log

Log

Purpose: Open and Close the Event Log Window.

Syntax : Log STATE/N,TOGGLE/S

STATE/N - A value of 0 will close the window, whereas any other value will open it.

TOGGLE/S - Flag to cause window to toggle between open and closed.

Example: log toggle

1.49 rx_logmessage

LogMessage

Purpose: Enter a Log Message into the Log Window.

Syntax : LogMessage SPACER/S,POPUP/S,MESSAGE/F

SPACER/S - Insert a standard spacer.

POPUP/S - Open the Log Window if it is closed.

MESSAGE/F - The message to be appended.

Example: logmessage spacer

logmessage popup This is a message

1.50 rx_maingadgets

MainGadgets

Purpose: Switch the display state of the Gadgets in the Main window on and off.

Syntax : MainGadgets STATE/N,TOGGLE/S

STATE/N - A value of 0 will cause the gadgets to be hidden, whereas any other value will make them appear.

TOGGLE/S - Flag to toggle the Main Gadgets.

Example: maingadgets state 1

1.51 rx_new

New

Purpose: A function to add a complete Alarm into the current list.

Syntax : New NAME/K/A,AHOUR/N/K/A,AMIN/N/K/A,LOGPOPUP/S,AEXECTYPE/N/K,AEXEC/K,SMIN/N/K,SSEC/N/K,STIM

NAME/K/A - Name of the New Event.

AHOUR/N/K/A - Alarm hour (0 to 23, eg. 24 hour mode).

AMIN/N/K/A - Alarm minute (0 to 59).

LOGPOPUP/S - Open Log Window at Alarm time.

AEXECTYPE/N/K - Interpretation for the Alarm Execute string. (0 = CLI, 1 = WB, 2 = ARexx).

AEXEC/K - Alarm event executable string.

SMIN/N/K - Snooze minutes (0 to 59).

SSEC/N/K - Snooze seconds (0 to 59).

STIMES/N/K - Snooze times (0 to 100, 0 = OFF and 100 = INFINITY).

SEXECTYPE/N/K - Interpretation for the Snooze Execute string. (0 = CLI, 1 = WB, 2 = ARexx).

SEXEC/K - Snooze event executable string.

WARN/N/K - Event Warn type (0 = OFF, 1 = GLOBAL, 2 = LOCAL).

Example: 'new name "Quick Alarm" ahour 'hour' amin 'min' aexec "c:Talk -w1 Wake up"'

1.52 rx_remove

Remove

Purpose: Remove all currently selected entries (or single Active) from list.

Syntax : Remove ACTIVE/S

ACTIVE/S - Switch to cause only active entry to be removed. Otherwise ALL the SELECTED entries are removed

Example: remove

returns the number of entries removed.

1.53 rx_restore

Restore

Purpose: Perform the normal Restore operation.

Restore the Alarm list to previous settings in ENV:

Of course the ENV:OnTime event file will be examined.

Syntax : Restore (no parameters)

Example: restore

1.54 rx_rx

RX

Purpose: Invoke an ARexx script from within Alarmist.

Syntax : RX FILE,SCRIPT/N/K

FILE - Full name of the file to be executed.

SCRIPT/N/K - The number of a previously installed ARexx Script to be executed.

Example: rx 'rexx:test_script.rexx'

rx script 2

1.55 rx_save

Save

Purpose: Perform the same function as the 'Save' button in the Main Window.

Syntax : Save (no parameters)

Example: save

1.56 rx_saveas

SaveAs

Purpose: Saves the current Alarm list to a specified file.

No OnTime events in the list will be saved.

Syntax : SaveAs FILE,ICON/S

FILE - Full name of the file to be saved.

ICON/S - Flag to indicate whether a 'presets' icon is to be created for this saved file.

Example: saveas "ram:testfile.list" icon

1.57 rx_toggle

Toggle

Purpose: Toggle the state (on/off) of all the selected entries (or single Active) in the list.

Syntax : Toggle STATE/N,TOGGLE/S

STATE/N - A value of 0 will make toggle selected entries off and and other value will turn them on.

TOGGLE/S - This switch will toggle all selected entries states.

Example: toggle state 1

1.58 rx_update

Update

Purpose: Reset Alarmist's internal settings to current values. This isn't very useful but it is a safe way of making sure Alarmist is up-to-date with its alarm management.

Syntax : Update (no parameters)

Example: update

1.59 rx_updatetime

UpdateTime

Purpose: Change the time between internal clock updates.

Syntax : UpdateTime SECONDS/N

SECONDS - The new time delay between clock updates.

Example: updatetime 5

1.60 rx_use

Use

Purpose: Perform the same function as the 'Use' button in the Main Window.

Syntax : Use (no parameters)

Example: use

1.61 rx_warn

Warn

Purpose: Activate and Deactivate Event Warning.

Syntax : Warn STATE/N,TOGGLE/S

STATE/N - A value of 0 will deactivate Warn, whereas any other value will activate it.

TOGGLE/S - Flag to cause Warn to toggle state.

Example: warn toggle

1.62 rx_warntime

WarnTime

Purpose: Change the Pre-Warn time in minutes.

Syntax : UpdateTime MIN/N

MIN - The new Pre-Warn Time in minutes.

Example: warn 10

1.63 rx_worldclock

WorldClock

Purpose: Open and Close the World Clocks Window.

Syntax : WorldClock STATE/N,TOGGLE/S

STATE/N - A value of 0 will close the window, whereas any other value will open it.

TOGGLE/S - Flag to cause window to toggle between open and closed.

Example: worldclock toggle