Tasker

COLLABORATORS			
	TITLE : Tasker		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 14, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Tasker

1.1 AFC Module: tasker/Main

** Tasker V1.10 - Original By Andrea Galimberti **

Part of Amiga Foundation Classes

Introduction

Author(s) Info

History
Requires: porttools

Base: $8004

COMMANDS	BRIEF DESCRIPTION

tasker()
Initialises the object

end()
Object destructor

code()
Selects the task code

stack()
Sets the task stack dimension

start()
Starts a TASK

stop()
Stops a task or a process

```
                process()
                Starts a PROCESS

                buildport()
                Creates a port for the TASK

                endport()
                Kills the task message port

                port()
                Returns the task port

                dosport()
                Returns the process DOS port

                send()
                Sends a message to the task

                dossend()
                Sends a message to the process' DOS port

                clone()
                Creates a copy of the tasker object

                version()
                Returns tasker version

Procedures:


                storea4()
                Stores the E global data pointer

                geta4()
                Gets the E global data pointer



                ERROR TABLE
```

1.2 Author(s) Info

```
 Original By:  Andrea Galimberti

 E Version By: Andrea Galimberti
```

1.3 Tasker / Introduction

```
Tasker.
```

This module allows you to transform a procedure in an independent task that runs asynchronously to the main program. You can choose if you want to start a task or a process, and how big the task's stack must be, and so on... A task can also be terminated by the main program. Instructions are also supplied to create message ports to communicate between the main program and the tasks.

Remember: a TASK cannot access disks and, in general, all resources managed by the dos.library, while a PROCESS can. Tasks and processes share with the main program only variables that have been defined as global.

1.4 Tasker / Error Table

Val (Hex)	Name	Description
$0000	TASK_ERR_MEM	No Memory
$0001	TASK_ERR_NOCODE	Cannot get E code for the task
$0002	TASK_ERR_NOTASK	Cannot start the task
$0003	TASK_ERR_NOPROC	Cannot start the process
$0004	TASK_ERR_NOPORT	Cannot create the message port

1.5 AFC module: tasker / history

V1.10 - Added clone() method.
- Added resourceTracker support.

V1.02 - added kill flag in tasker() and modified the end() method to use this flag.
- now stop() and end() check if the task is still present before deleting it.

V1.01 - added methods dosport() and dossend() to exploit the DOS message port of a process.
- now stop() and end terminate also a process (not only a task).
- added procedures: setupMsg() AND sendMsg() TO make life easier with the message mechanism.

V1.00 - First Release

1.6 procedure / storea4()

NAME: storea4()

DESCRIPTION: memorizes the E global data pointer. You MUST call this procedure (only once) at the beginning of your main program before you start any process or task. Then, in the task body, you MUST call the
geta4()
procedure before accessing any

```
              global variable or calling a procedure.

       INPUTS: NONE

      RESULTS: NONE

    SEE ALSO:
               geta4()
```

1.7 procedure / geta4()

```
                        NAME: geta4()

  DESCRIPTION: gets the E global data pointer. You MUST call this
               procedure in the body of your process or task before you
               access any global variable or call a procedure.

       INPUTS: NONE

      RESULTS: NONE

    SEE ALSO:
               storea4()
```

1.8 AFC module: tasker / tasker()

```
                        NAME: tasker(name, kill=TRUE, resTracker=NIL)

  DESCRIPTION: this method creates a task object: name is the name
               (possibly unique) the task will have when started. Some
               values are set to their defaults:  the stack allocated for
               the task will be of 4000 bytes if not specified with the

               stack()
                method.
                  If the kill flag is TRUE then the task (or process) will
               be stopped and its port (if it has been created with

               buildport()
               ) will be closed when the tasker object
               is disposed; if the kill flag is FALSE then it's your duty
               to close the port (with
               endport()
               ) and
               stop()
                the task
               before disposing of the object.
               The optional resTracker parameter points to an instance of
               the resourceTracker class.
```

```
       INPUTS: name: name of the task
               kill: if TRUE the task and its port are killed before
                     disposing the object
               resTracker: resourceTracker object

      RESULTS: NONE

     SEE ALSO:
                end()
```

1.9 AFC module: tasker / end()

```
                        NAME: end()

   DESCRIPTION: object destructor. The task (or process) will be stopped,
                and its port (if any) will be deleted, before disposing the
                object only if you specified TRUE for the 'kill' flag when
                you allocated the object.

       INPUTS: NONE

      RESULTS: NONE

     SEE ALSO:
                tasker()
```

1.10 AFC module: tasker / code()

```
                        NAME: code(procedure)

   DESCRIPTION: chooses the code that will be executed when the task is
                started.

       INPUTS: procedure is the ADDRESS of the procedure you want to start
                as a separate task.

      RESULTS: NONE

     SEE ALSO:
                start()
```

1.11 AFC module: tasker / stack()

```
          NAME: stack(bytes)

   DESCRIPTION: with this method you enter the dimension of the stack (in
                bytes) to be allocated when the task is started.
```

```
      INPUTS: the dimension in bytes of the task's stack: this defaults
              to the value of 4000 bytes.

     RESULTS: NONE

    SEE ALSO:
```

1.12 AFC module: tasker / start()

```
                        NAME: start(priority=0)

 DESCRIPTION: starts a TASK. The task ends if stopped with the
              stop()
                              method, or if the task object is ENDed, or if the  ↩
                                 task code
              terminates its job.

      INPUTS: the priority of the task: default is 0 (normal priority).

     RESULTS: raises an exception if it's not been able to create the
              task. (see
              Error Table
              )

    SEE ALSO:
               stop()
```

1.13 AFC module: tasker / process()

```
                        NAME: process(priority=0)

 DESCRIPTION: starts a PROCESS: remember that a process is a task that
              can access the Dos resources (e.g., drives) because it is
              managed by the dos.library. The process' code and stack are
              entered with the same methods used with tasks.
                The process ends when it has finished its job (just as
              any other program does), or when it is stopped with the

              stop()
               method, or when the tasker associated with it is
              ENDed.

      INPUTS: priority of the process: default is 0.

     RESULTS: raises an exception if it cannot create the process.
              (see
              Error Table
              )

    SEE ALSO:
```

```
                code()

                stack()

                stop()
                                Important: see
                storea4()

                geta4()
```

1.14 AFC module: tasker / stop()

```
                          NAME: stop()

   DESCRIPTION: terminates the TASK: Exec will remove the task from its
                list and will automatically deallocate the task's stack.
                Remember that you have to free yourself all resources you
                have allocated for this task. The task object is not
                modified, so you can start again the same task by calling
                the
                start()
                 method once again.
                  This works also with a process (only you call the

                process()
                 method to restart it).

        INPUTS: NONE

       RESULTS: NONE

      SEE ALSO:
                 start()
```

1.15 AFC module: tasker / buildport()

```
                          NAME: buildport(name=NIL, priority=0)

   DESCRIPTION: this method creates a message port associated with the
                task.  Important:  the port is associated to the task that
                created it, so make sure that this method is called by the
                procedure that constitutes the task, not by the main
                program.
                  If you supply a name the port will be made public, and in
                such a case you can even choose the priority of your public
                port.

        INPUTS: name: <>NIL if you want the port to be public
                priority: only needed if the port is a public port.
```

```
      RESULTS: raises an exception if unable to create the port
               (see
               Error Table
               )

     SEE ALSO:
                endport()
```

1.16 AFC module: tasker / endport()

```
                         NAME: endport()

  DESCRIPTION: removes the task's message port created with
               buildport()
               .
               Remember to get all the messages (and to reply) before
               ending the port (if you are sure that the port is empty you
               needn't do this).  The method cannot do it for you because
               it can't know if the task that sent the message is still
               present or not (replying to an unexisting task will cause a
               guru).

       INPUTS: NONE

      RESULTS: NONE

     SEE ALSO:
                buildport()
```

1.17 AFC module: tasker / port()

```
                         NAME: port()

  DESCRIPTION: returns the address of the port created by
               buildport()
               ,
               or NIL if the port doesn't exist.

       INPUTS: NONE

      RESULTS: the port address.

     SEE ALSO:
                buildport()
```

1.18 AFC module: tasker / dosport()

```
                        NAME: dosport()

   DESCRIPTION: returns the address of the port associated to the process:
                this port is created by AmigaDOS when the process is
                started and is reserved to exclusive use of AmigaDOS, so
                you are invited to create your own port with the

                buildport()
                 method if you need one.

        INPUTS: NONE

       RESULTS: the port address, or NIL if the task is not a process.

      SEE ALSO:
                 buildport()
```

1.19 AFC module: tasker / send()

```
                        NAME: send(message)

   DESCRIPTION: sends an Exec message to the port created with the

                buildport()
                 method.

        INPUTS: the message address.

       RESULTS: TRUE if message sent, otherwise FALSE.

      SEE ALSO:
                 buildport()

                 port()
```

1.20 AFC module: tasker / dossend()

```
                        NAME: dossend(message)

   DESCRIPTION: sends an Exec message to the port associated with the
                process (the one created by and reserved to AmigaDOS).

        INPUTS: the message address.

       RESULTS: TRUE if message sent, otherwise FALSE.

      SEE ALSO:
                 dosport()
```

1.21 AFC MODULE: tasker / clone()

```
                        NAME: clone(name, resTracker=NIL)

 DESCRIPTION: allocates a copy of the current tasker object. You have to
              supply the name of the new task.

      INPUTS: name: name of the new (copy) task.
              resTracker: resourceTracker object

     RESULTS: pointer to the allocated instance of the tasker object, or
              NIL if something went wrong.

    SEE ALSO:
               tasker()
```

1.22 AFC module: tasker / version()

```
         NAME: version()

 DESCRIPTION: returns version and revision of tasker code.

      INPUTS: NONE

     RESULTS: version, revision

    SEE ALSO:
```