

System

COLLABORATORS

	<i>TITLE :</i> System		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		August 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	System	1
1.1	System-related Classes for AmigaTalk© 1998:	1
1.2	Library Class (Parent Class = Object):	2
1.3	Port Class:	2
1.4	Processes Class:	3
1.5	Task Class (Parent Class = Processes):	3
1.6	Memory Class:	3
1.7	Lists Class:	4
1.8	Interrupt Class:	4
1.9	Semaphore Class:	4
1.10	Signal Class:	4
1.11	Exception Class:	4
1.12	ARexx Class:	5
1.13	Device Class:	5
1.14	Serial Class (Parent Class = Device):	6
1.15	Audio Class (Parent Class = Device):	7
1.16	Narrator Class (Parent Class = Device):	7
1.17	Clipboard Class (Parent Class = Device):	8
1.18	ConsoleClass (Parent Class = Device):	8
1.19	Keyboard Class (Parent Class = Device):	8
1.20	GamePort Class (Parent Class = Device):	8
1.21	Input Class (Parent Class = Device):	11
1.22	ParallelClass (Parent Class = Device):	11
1.23	Printer Class (Parent Class = Device):	11
1.24	SCSI Class (Parent Class = Device):	11
1.25	Timer Class (Parent Class = Device):	12
1.26	TrackDisk Class (Parent Class = Device):	13

Chapter 1

System

1.1 System-related Classes for AmigaTalk© 1998:

Described herein are the classes & their methods for manipulating Amiga-System objects with AmigaTalk.

Libraries

Devices

Serial

Audio

Narrator

Clipboard

Console

Keyboard

GamePort

Input

Parallel

Printer

SCSI

Timer

TrackDisk

Ports

Processes

Tasks

Memory

Lists

Interrupt

Semaphore

Signal

Exception

ARexx

1.2 Library Class (Parent Class = Object):

Class Library allows the user of the AmigaTalk system to retrieve values for any Library known to AmigaTalk & to open or close them.

Valid methods are:

new: libname

Initialize the Library class instance variable to libname.

open: libraryName version: ver

Open the library libraryName at the version supplied.

Example libraryName: 'intuition.library'

close: libraryName

Close the given library.

getIDString: libraryName

Return the library's Internal identifier string.

getVersion: libraryName

Return the library's Version number.

getRevision: libraryName

Return the library's Revision number.

getChecksum: libraryName

Return the library's CheckSum.

getOpenCount: libraryName

Return a count of the number of times the library has been opened by the Amiga OS.

NOTE: The following methods are only supplied to complete the accessibility to the Library class & are probably not of much use to the casual user:

getNegSize: libraryName

Return the amount of bytes in front of the library ROMTAG.

getPosSize: libraryName

Return the amount of bytes after the library ROMTAG.

getFlags: libraryName

Return the library's Flags value.

1.3 Port Class:

Class Port allows the user of the AmigaTalk system to send & retrieve messages from the MessagePort named.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

NOTE: Once a port has been made, all messages to & from the port are supposed to be the given size (MessageSize). Any code that you write to use Ports should therefore verify the size of the bytearrays passed in & out or clip them to the known size.

Valid methods are:

makePort: portName messageSize: msgSize priority: pri

Return o_true if the Port was made & registered, o_nil otherwise.

killPort

Close & delete the Port from the AmigaTalk system.

getMessage

Return the ByteArray that was present on the Port.

sendMessage: byteArray

Place the given byteArray (Message) on the Port.

1.4 Processes Class:

Not implemented yet!

Class Processes allows the user of the AmigaTalk system to create & destroy processes that are running in their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

SubClasses:

Tasks

1.5 Task Class (Parent Class = Processes):

Not implemented yet!

Class Task allows the user of the AmigaTalk system to create & delete tasks that are running in their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.6 Memory Class:

Not implemented yet!

Class Memory allows the user of the AmigaTalk system to manipulate memory blocks inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.7 Lists Class:

Not implemented yet!

Class Lists allows the user of the AmigaTalk system to manipulate Lists inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.8 Interrupt Class:

Not implemented yet!

Class Interrupt allows the user of the AmigaTalk system to manipulate software Interrupt handlers inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.9 Semaphore Class:

Not implemented yet!

Class Semaphorte allows the user of the AmigaTalk system to manipulate Semaphores inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.10 Signal Class:

Not implemented yet!

Class Signal allows the user of the AmigaTalk system to manipulate signals inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.11 Exception Class:

Not implemented yet!

Class Exception allows the user of the AmigaTalk system to manipulate system Exception handlers inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.12 ARexx Class:

Not implemented yet!

Class ARexx allows the user of the AmigaTalk system to manipulate ARexx ports & messaging inside their Amiga.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.13 Device Class:

Class Device is an abstract class. Derived classes use its methods to talk to the Amiga OS. All of the following methods only return an error message, they should be re-defined by sub-classes:

clear

flush: devName

invalid

read: devName

reset: devName

stop

start

update

write: devName this: string

SubClasses:

Serial

Audio

Narrator

Clipboard

Console

Keyboard

GamePort

Input

Parallel

Printer

SCSI

Timer

TrackDisk

1.14 Serial Class (Parent Class = Device):

Class Serial allows the user of AmigaTalk to utilize the Serial

Device that the Amiga PC uses to communicate to the outside world.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

NOTE: All commands (except Read & Write) use BeginIO with IOF_QUICK set, regardless of the state of the SyncType.

open: serialName withBufSize: size

Open the serial device & assign it the given name with the buffer size given (one for reads & one for writes).

close: serialName withBufSize: size

Close the serial device & deallocate the buffers.

initialize: serialName withTerm: charVals

Initialize the serial device & utilize the charVals as EOF indicators.

read: serialName sync: onOrOff

Read the serial device. If onOrOff == 0, then perform:

BeginIO();

WaitIO();

else perform:

DoIO();

write: serialName this: writeString

Write the given string to the serial device.

reset: serialName

Issue a CMD_RESET to the serial device.

pause: serialName

Issue a CMD_STOP to the serial device.

restart: serialName

Issue a CMD_START to the serial device.

sendBreak: serialName ofDuration: usecs

Issue a CMD_BREAK to the serial device.

getStatus: serialName

Return the status bits (SDCMD_QUERY) of the serial device.

flush: serialName

Issue a CMD_FLUSH to the serial device.

clearReadBuffer: serialName

Place nils ('\\0') in all of the read buffer locations.

setSyncType: serialName to: newSync

Set the type of synching to use, synchronous > 0 or asynchronous = 0.

setBaud: serialName to: newBaud

Set the BAUD rate for the serial device.

setParity: serialName to: newParity status: onOrOff

Enable or disable Parity, where newParity has the following values:

0 = Space onOrOff: 0 = disable, 1 = enable.

1 = Mark

2 = Even

3 = Odd

setDataSize: serialName to: newSize

Set the read buffer length to newSize.

NOTE: This method does NOT reallocate the buffer.

setStops: serialName to: newStops

Set the number (0, 1, or 2) of Stop bits to use for communication.

setBreakLen: serialName to: duration

Set the duration of break signals to duration.

setRBufSize: serialName to: size

Reset the size of the read buffer to size.

setFlags: serialName to: newFlags

Set the serial device Flags.

setTerminators: serialName to: termChars

Set the termination characters array that will signal the serial device to break.

termchars = 8 bytes in descending order representing the characters that the serial channel should recognize as EOF characters.

1.15 Audio Class (Parent Class = Device):

Not implemented yet!

220 0 cmd channel# channelname -- ActOnAudioChannel

Class Audio allows the user of AmigaTalk to utilize the Audio Device that the Amiga PC uses to generate sounds & speech.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.16 Narrator Class (Parent Class = Device):

Not implemented yet!

220 1 cmd narratorname -- ActOnNarrator

Class Narrator allows the user of AmigaTalk to utilize the Narrator Device that the Amiga PC uses to talk with.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.17 Clipboard Class (Parent Class = Device):

Not implemented yet!

221 cmd clipboardname -- ActOnClipboard

Class Clipboard allows the user of AmigaTalk to utilize the Clipboard Device that the Amiga PC uses to temporarily store text & images.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.18 ConsoleClass (Parent Class = Device):

Not implemented yet!

222 0 cmd consolename -- ActOnConsole

Class Console allows the user of AmigaTalk to utilize the Console Device that the Amiga PC uses to send & receive keyboard input from the User.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.19 Keyboard Class (Parent Class = Device):

Not implemented yet!

222 1 cmd keyboardname -- ActOnKeyboard

Class Keyboard allows the user of AmigaTalk to utilize the Keyboard Device that the Amiga PC uses to control the keyboard.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.20 GamePort Class (Parent Class = Device):

Class GamePort allows the user of AmigaTalk to utilize the GamePort Device that the Amiga PC uses to detect input events, such as mouse movement or button clicks or joystick movement.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

openGamePort: unit named: portname

Open the given GamePort unit & assign it the given portname. If a unit is already in use, AmigaTalk will NOT open the given unit.

closeGamePort

Close the GamePort & remove it from the Amigatalk system.

setKeyTransition: transType

Tell AmigaTalk which type of key press to react to.

transType is either GPTF_UPKEYS = 2 or GPTF_DOWNKEYS = 1 or both in value.

setTimeTransition: timeOutValue

Tell Amigatalk when to let time expire on GamePort events.

setXDeltaTransition: xvalue

Tell AmigaTalk how far in the horizontal direction the gameport device has to move to generate an event.

setYDeltaTransition: yvalue

Tell AmigaTalk how far in the vertical direction the gameport device has to move to generate an event.

clearGamePortBuffer

Flush the gameport events out of the device.

waitForButton: kvalue

Tell AmigaTalk to wait for the Fire button, or a mouse button being pressed. See the TestFiles/TestGamePort file or Amiga OS 3.0+ include file Devices/InputEvent.h for additional information.

waitForQualifier: qvalue

Tell AmigaTalk to wait for the Fire button, or a mouse button being pressed. See the TestFiles/TestGamePort file or Amiga OS 3.0+ include file Devices/InputEvent.h for additional information.

waitForXPos: xvalue

Tell AmigaTalk to wait for the given x-position value to occur.

For Absolute-type joysticks (GPCT_ABSJOYSTICK), the valid values are:
-1 = left, 0 = no movement, +1 = right.

waitForYPos: yvalue

Tell AmigaTalk to wait for the given y-position value to occur.

For Absolute-type joysticks (GPCT_ABSJOYSTICK), the valid values are:
-1 = up, 0 = no movement, +1 = down.

getControllerType

The integer returned by this method is one of the following:

-1 -> gameport already allocated by another user.

0 -> gameport Not being used.

1 -> gameport is hooked to a mouse.

2 -> gameport is hooked to a relative value joystick.

3 -> gameport is hooked to an absolute joystick.

setControllerType: newCType

newCType is one of the following:

-1 (custom device, such as a paddle).

1 mouse.

2 relative joystick.

3 absolute joystick.

getPortUnitNumber

Return the port number (0 or 1) that the receiver is attached to.

This is a silly method in that it supplies information that is already known.

getButtonCode

Return the button Code that the gameport received.

getQualifiers

Return the input event Qualifiers that the gameport received.

getXPos

Return the x-position that the gameport received.

getYPos

Return the y-position that the gameport received.

getIEAddress

Return the Event Address that the gameport received.

WARNING: If you don't know what this is, don't use this method!

getTimeStamp

Return the Event seconds value that the gameport received.

getTriggerKeys

Return which type of key presses the gameport is looking for.

Either GPTF_UPKEYS = 2, GPTF_DOWNKEYS = 1 or both = 3 will be valid values.

getTriggerTime

Return the timeout value that the gameport is currently set to.

getTriggerXDelta

Return the horizontal direction trigger value that the gameport device is currently set to.

getTriggerYDelta

Return the vertical direction trigger value that the gameport device is currently set to.

1.21 Input Class (Parent Class = Device):

Not implemented yet!

223 1 cmd inputname -- ActOnInput

Class Input allows the user of AmigaTalk to utilize the Input

Device that the Amiga PC uses to get User input.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.22 ParallelClass (Parent Class = Device):

Not implemented yet!

224 cmd parallelname -- ActOnParallel

Class Parallel allows the user of AmigaTalk to utilize the

Parallel Device that the Amiga PC uses to control the parallel port.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.23 Printer Class (Parent Class = Device):

Not implemented yet!

225 cmd printername -- ActOnPrinter

Class Printer allows the user of AmigaTalk to utilize the

Printer Device that the Amiga PC uses to control printers.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.24 SCSI Class (Parent Class = Device):

Not implemented yet!

226 cmd SCSIname -- ActOnSCSI

Class SCSI allows the user of AmigaTalk to utilize the SCSI

Device that the Amiga PC uses to control SCSI peripherals.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

1.25 Timer Class (Parent Class = Device:)

Class Timer allows the user of AmigaTalk to utilize the Timer

Device that the Amiga PC uses to control timers. NOTE: System Date & Time functions are implemented in class AmigaTalk.

Timer requests fall into two categories:

1. Time delay - wait a specified amount of time.
2. Time measure - Record the time, do other tasks, Record the time again & take the difference between the two times.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!

openTimerType: type name: timerName seconds: s micros: m

Open the given Timer unit & assign it the given timerName. If a unit is already in use, AmigaTalk will NOT open the given unit.

Currently known Timer types recognized by the Amiga are:

1. UNIT_MICROHZ (0)
2. UNIT_VBLANK (1)
3. UNIT_ECLOCK (2)
4. UNIT_WAITUNTIL (3)
5. UNIT_WAITECLOCK (4)

Please read the RKM Devices manual (pg. 288 of 3rd Edition) for a detailed explanation of their differences.

close

Abort the Timer's operation.

stop

Stop the Timer's action.

startWithSecs: s micros: m

Start the Timer with the given parameters.

delaySeconds: s micros: m

Start the Timer & wait for completion of the timing event.

test

Check that the Timer for error conditions.

getSeconds

Return the number of seconds that the Timer is using.

getMicros

Return the number of microseconds that the Timer is using.

setSeconds: s micros: m

Change the timing parameters of the Timer.

compare: s micros: m toSeconds: s2 micros: m

Compare the two given sets of parameters.

if $t1 > t2$, return -1

else if $t1 < t2$, return +1

else if $t1 == t2$, return 0

getEClockHigh

Return the upper long word of the E-Clock time.

getEClockLow

Return the lower long word of the E-Clock time.

1.26 TrackDisk Class (Parent Class = Device):

Not implemented yet!

229 cmd trackdiskname -- ActOnTrackdisk

Class TrackDisk allows the user of AmigaTalk to utilize the TrackDisk Device that the Amiga PC uses to control floppy disks.

WARNING: You should know what you're doing to the Amiga OS before messing with this Class, or any other System Class!