System

COLLABORATORS			
	TITLE : System		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

System

1.1 System-related Classes for AmigaTalk© 1998-2000:

Described herein are the classes & their methods for manipulating

Amiga-System objects with AmigaTalk.

Libraries

Devices

Serial

Audio -- Not Implemented yet!

Narrator

Clipboard

IFFClipBoard

Console -- Not Implemented yet!

Keyboard -- Not Implemented yet!

GamePort

Mouse

AbsJoyStick

RelJoyStick

Input -- Not Implemented yet!

Parallel

Printer -- Not Implemented yet!

SCSI -- Not Implemented yet!

Timer

TrackDisk

Ports

Tasks -- Not Implemented yet!

Processes -- Not Implemented yet!

Memory -- Not Implemented yet!

Lists -- Not Implemented yet!
Interrupt -- Not Implemented yet!
Semaphore -- Not Implemented yet!
Signal -- Not Implemented yet!
Exception -- Not Implemented yet!
ARexx -- Not Implemented yet!

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'
NOTE: This method does NOT tie the opened Library into AmigaTalk.
Its only practical purpose is to allow the User to see if
a specific version of a Library is present in their system.
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
accessability 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!
Valid methods are:
new: newPortName
Initialize a new instance of Port with the name newPortName.
This has to be done before any other method.
makePort messageSize: msgSize priority: pri
Return true if the Port was made & registered, nil otherwise.
This has to be done AFTER new: newPortName
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.
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.
checkForPort
Return true if the Port exists, or false if not.
linkToOutsidePort: sysPortName name: aTalkName size: msgSize
Return true if aTalkName & sysPortName have been linked,
false if not.

1.4 Processes Class (Parent Class = Task):

Not implemented yet!
Class Processes allows the user of the AmigaTalk system to create
& destroy processes that are running in their Amiga.
NOTE: This is NOT the same as the Process Class, which is from the
General/Process.st file!
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
SubClasses:

Proposed methods are:

new: newProcessName

addProcess

setProcessInput: inputName

setProcessOutput: outputName

setProcessState: newProcessState

setProcessPriority: newProcessPriority

freezeProcess

removeProcess

displayProcesses

1.5 Task Class:

Not implemented yet!

Class Task allows the user of the AmigaTalk system to create

& delete tasks that are running in their Amiga.

SubClass: Processes

WARNING: You should know what you're doing to the Amiga OS before

messing with this Class, or any other System Class!

Proposed methods are:

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!

Proposed methods are:

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!

Proposed methods are:

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!
Proposed methods are:

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!
Proposed methods are:

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!
Proposed methods are:

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!
Proposed methods are:

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!
Proposed methods are:

1.13 Device Class:

Class Device is an abstract class.
Methods defined by the Class are:
getDeviceAddressList
Return an Array of Device Addresses.
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
IFFClipBoard
Console
Keyboard
GamePort
Mouse
AbsJoyStick
RelJoyStick
Input

Parallel

Printer

SCSI

Timer

TrackDisk

1.14 SerialDevice Class (Parent Class = Device):

Class SerialDevice 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.
new: newSerialName
Initialize an Instance of SerialDevice & name it newSerialName.
open
Open the serial device & assign it the given name with the
buffer size given (one for reads & one for writes).
setBufferSize: newSize
Set the size of the (initial or opening) buffers.
close
Close the serial device & deallocate the buffers.
initializeWithTerm: charVals
Initialize the serial device & utilize the charVals as EOF indicators.
readSync: syncValue
Read the serial device. If syncValue == 0, then perform:
BeginIO();
WaitIO();
else perform:
DoIO();
writeThis: writeString
Write the given string to the serial device.
reset
Issue a CMD_RESET to the serial device.
pause
Issue a CMD_STOP to the serial device.
restart

Issue a CMD_START to the serial device.

sendBreakOfDuration: usecs

Issue a CMD_BREAK to the serial device.

getStatus

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

flush

Issue a CMD_FLUSH to the serial device.

clearReadBuffer

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

setSyncType: newSync

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

setBaud: newBaud

Set the BAUD rate for the serial device.

setParity: 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: newSize

Set the read buffer length to newSize.

NOTE: This method does NOT reallocate the buffer.

setStops: newStops

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

setBreakLen: duration

Set the duration of break signals to duration.

setRBufSize: size

Reset the size of the read buffer to size.

setFlags: newFlags

Set the serial device Flags.

setTerminators: 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!
<primitive 220 0 cmd# channel# channelname>
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!
Proposed methods are:

1.16 Narrator Class (Parent Class = Device):

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!
Class Methods are:
new
Initialize the instance of Class Narrator. NOTE: Class Narrator
is a Singleton class, which means there can be only one narrator
while Amigatalk is running.
close
De-allocate the resources for the narrator.
setVolume: newSpeakingVolume
Change the volume of the narrator. Range: 0 to 64 (max volume).
setSex: newSpeakerSex
Only Male = 1 or Female = 0 (at this time) are available.
setPitch: newSpeakingPitch
Set the basic frequency of the narrator's voice.
RANGE 65 to 320 Hz (max).
setMode: newModeString
Robotic, Natural or Manual are the only recognized values for
newModeString.
setRate: newSpeakingRate
Set the speaking rate in words-per-minute (40 to 400 max).
setFormant1: percentDeviation
Change the tuning of the lowest formant frequency. Positive values
raise the formant frequency (Default = 0).
RANGE: -100 to 100 by 5% steps.

setFormant2: percentDeviation

Change the tuning of the middle formant frequency. Positive values

raise the formant frequency (Default = 0).

RANGE: -100 to 100 by 5% steps.

setFormant3: percentDeviation

Change the tuning of the highest formant frequency. Positive values

raise the formant frequency (Default = 0).

RANGE: -100 to 100 by 5% steps.

setFormant1Amplitude: newAmplitude

Change the amplitude of the lowest formant frequency.

Positive values raise the formant Amplitude.

RANGE: 31 to -32 dB (= OFF)

setFormant2Amplitude: newAmplitude

Change the amplitude of the middle formant frequency.

Positive values raise the formant Amplitude.

RANGE: 31 to -32 dB (= OFF)

setFormant3Amplitude: newAmplitude

Change the amplitude of the highest formant frequency.

Positive values raise the formant Amplitude.

RANGE: 31 to -32 dB (= OFF)

setFlags:

NDB_NEWIORB = 1, NDB_WORDSYNC = 2, NDB_SYLSYNC = 4.

Default: NDB_NEWIORB only.

setVoicingAmplitude: newAVBias

Play with it & see what it does. RANGE: 31 to -32 dB.

setFricationAmplitude: newAFBias

Play with it & see what it does. RANGE: 31 to -32 dB.

speak: normalString

Translate the normalString into a phonetic string & have the

Narrator speak it.

speakPhonetics: phoneticString

Have the Narrator speak the phoneticString.

setPriority: newSpeakingPriority

Default: 100. Range: -128 to 127 (max).

setEnthusiasm: aFloat

set Enthusiasm of the narrator from 1/32 to 32/32.

(32/32 is the default).

setPitchModulation: voiceQuiver

Add pitch modulation to the narrator. RANGE: 0 to 255. (0 = default).

Non-zero values make the narrator sound older.

setArticulation: newPercentArticulation

Set amount of slurring of words.

RANGE: 0 to 255% (max). (Default = 100 = 100%)

setPhonemes: phonemeString

NOTE: Used in conjunction with setCentralizeValue: method to alter the Narrator's acccent.

Valid strings are:

IY long e as in beet, eat.

IH short i as in bit, in.

EH short e as in bet, end.

AE short a as in bat, ad.

AA short o as in bottle, on.

AH short u as in but, up.

AO short a as in ball, awl.

OW long o as in boat, own. (diphthong)

UH short u as in book, soot.

ER short i as in bird, early.

UW long u as in brew, boolean. (diphthong)

No checking is performed on your string, so get it right!

setCentralizeValue: newCentralizePercent

RANGE: 0 to 100% (0 = default).

NOTE: Used in conjunction with setPhonemes: method to alter the Narrator's acccent.

1.17 Clipboard Class (Parent Class = Device):

Class Clipboard allows the user of AmigaTalk to utilize the Clipboard Device that the Amiga PC uses to temporarily store text & images. You may open up to 256 different clipboards, but there is a unique instance of this Class for each Clipboard you use.

Valid unit Numbers are from 0 to 255.

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

The methods are:

new

Return Clipboard unit 0.

new: unitNum

Return Clipboard unitNum.

setClipUnit: unitNum

Set the unit number of the clipboard to use with other methods.

As a user, you should never have to use this method.

writeFTXTClipToFTXTFile: fileName

Write the Clipboard to the fileName in FTXT format.

writeFTXTClipToASCIIFile: fileName

Write the Clipboard to the fileName in plain ASCII format.

postFTXTToClip: ftxtString

Place the FTXT string contents into the Clipboard.

Experimental, do NOT use.

sendAsciiFileToClip: fileName

Place the contents of an ASCII file to the Clipboard.

update

Send CMD_UPDATE to the Clipboard device.

sendFTXTFileToClip: fileName

Place the contents of an FTXT file to the Clipboard.

sendILBMFileToClip: fileName

Place the contents of an ILBM file to the Clipboard.

(Once I figure out a clean way to do it, I'll write the code

to retrieve an ILBM clip).

sendAsciiStringToClip: clipString

Place the plain ASCII string contents into the Clipboard.

clipTypeIs

Return 0 if the Clipboard data is FTXT or 1 if the data is ILBM.

Once the Hook class is working the following methods will be

made available:

WARNING!!!! There is only space for ONE change hook for the

Clipboard, so be extremely careful how you use it!

openHookedClipboard: clipNumber withHook: aHook

Open a Clipboard with a changeHook function.

closeHookedClipboard

Close a Clipboard that has a changeHook function.

SEE ALSO, IFFClipBoard Class

1.18 IFFClipBoard Class (Parent Class = Device):

Class IFFClipboard allows the user of AmigaTalk to utilize the

Clipboard Device that the Amiga PC uses to temporarily store text only.

You may open up to 256 different clipboards, but there

is a unique instance of this Class for each Clipboard you use.
Valid unit Numbers are from 0 to 255.
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
The methods are:
writeToClipUnit: unit fromFTXTString: ftxtString
Place the contents of the FTXT-format string into the Clipboard.
sendFTXTClip: unit toFTXTString: ftxtString size: numBytes
Place the contents of the Clipboard into a string as FTXT.
SEE ALSO, ClipBoard Class

1.19 ConsoleClass (Parent Class = Device):

Not implemented yet!
<primitive 222 0 cmd# consolename>
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!
Proposed methods are:
new: newConsoleName
open
openWithWindow: consoleString
close
attachToWindow: windowTitle
detachFromWindow
readChar
readString
readSpecialKey
writeChar: consChar
writeString: consString
writeCSIString: consCSIString

1.20 Keyboard Class (Parent Class = Device):

Not implemented yet!
<primitive 222 1 cmd# keyboardname>
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!
Proposed methods are:

1.21 Mouse Class (Parent Class = GamePort):

Class Mouse allows the user of AmigaTalk to utilize the
GamePort Device that the Amiga PC uses to detect input events from a
mouse, such as movement or button clicks.
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
The Class implements the following methods:
openMousePort: 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.
closeMousePort
Close the GamePort & remove it from the AmigaTalk system.
clearMousePortBuffer
Flush the gameport events out of the device.
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/button 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.
setKeyTransition: transType
Tell AmigaTalk which type of key/button 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.
waitForButton: kvalue
Tell AmigaTalk to wait for a mouse button being pressed. See the
TestFiles/TestMousePort file or Amiga OS 3.0+ include file
Devices/InputEvent.h for additional information.
waitForQualifier: qvalue
Tell AmigaTalk to wait for a mouse button being pressed. See the
TestFiles/TestMousePort 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.
If xvalue is less than zero, the User has to move the mouse left.
waitForYPos: yvalue
Tell AmigaTalk to wait for the given y-position value to occur.
If yvalue is less than zero, the User has to move the mouse ??.

1.22 AbsJoyStick Class (Parent Class = GamePort):

Class AbsJoyStick allows the user of AmigaTalk to utilize the
GamePort Device that the Amiga PC uses to detect input events from an
Absolute-type joystick (normally, a User should be using the
RelJoyStick Class).
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
The Class implements the following methods:
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.

clearGamePortBuffer

Flush the gameport events out of the device.

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 (normally, xvalue = 1).

setYDeltaTransition: yvalue

Tell AmigaTalk how far in the vertical direction the gameport

device has to move to generate an event (normally yvalue = 1).

waitForButton: kvalue

Tell AmigaTalk to wait for the Fire 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 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.

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.

1.23 RelJoyStick Class (Parent Class = GamePort):

Class RelJoyStick allows the user of AmigaTalk to utilize the

GamePort Device that the Amiga PC uses to detect input events from a

Relative-type joystick.

WARNING: You should know what you're doing to the Amiga OS before

messing with this Class, or any other System Class!

The Class implements the following methods:

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.

clearGamePortBuffer

Flush the gameport events out of the device.

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.

waitForButton: kvalue

Tell AmigaTalk to wait for the Fire 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 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.

waitForYPos: yvalue

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

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.

1.24 GamePort Class (Parent Class = Device):

Class GamePort is an abstract Parent Class for the following
classes:
Mouse
AbjJoyStick
RelJoyStick
The subClasses implement more functionality for the GamePort Device.
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
The Class implements the following methods:
openGamePort: whichUnit named: portname
For this Class, this method returns an error String that tells the
User that a subClass has to implement this method.
getControllerType: portname
This method returns an Integer that corresponds to:
-1 = GPCT_ALLOCATED -- indicating port is already being used.
0 = GPCT_NOCONTROLLER -- indicating port is NOT being used.
1 = GPCT_MOUSE -- indicating port is a mouse.
2 = GPCT_RELJOYSTICK -- indicating port is a Relative JoyStick.
3 = GPCT_ABSJOYSTICK -- indicating port is an Absolute JoyStick.
new: dummy
For this Class, this method returns an error String that tells the
User that new: is NOT understood by this Class.

1.25 Input Class (Parent Class = Device):

Not implemented yet!
<primitive 223 1 cmd# inputname>
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!
Proposed methods are:

1.26 ParallelClass (Parent Class = Device):

Class ParallelDevice allows the user of AmigaTalk to
utilize the Parallel Device that the Amiga PC uses to control the
parallel port. NOTE: The parallel.device has to be opened with the
PARF_SHARED flag set, so AmigaTalk makes sure to provide it before
opening the device is attempted.
This class is a Singleton Class. In the future, this class will be
modified to allow more than one Parallel port to be open at a time
(for those of us fortunate enough to have more than one Parallel Port.
WARNING: You should know what you're doing to the Amiga OS before
messing with this Class, or any other System Class!
status
Return the status (PDCMD_QUERY) of the Parallel Port.
The returned status has the following meaning:
BIT: ACTIVE: FUNCTION:
0 HIGH Printer Busy toggle (offline).
1 HIGH Paper out.
2 HIGH Printer Select.
3 ---- Read = 0, Write = 1
4-7 ---- Reserved.
resetPort
send a CMD_RESET to the ParallelPort.
flushPort
send a CMD_FLUSH to the ParallelPort.
stopPort
send a CMD_STOP to the ParallelPort.
startPort
send a CMD_START to the ParallelPort.
readThisMany: numChars
Read the desired amount of characters (bytes) from the Parallel Port.
The return value should equal the amount of characters requested.
writeToPort: aString thisLong: numChars
Write the given string to the Parallel Port. numChars should be less
than or equal to the length of aString.
Your device must handshake with the Amiga. Most printers do, other-
wise, the method will hang up at a WaitIO() forever!
setTerminatorsTo: aString
Set the termination array of characters to the given string of 4

characters. These characters have to be in descending ASCII order.
AmigaTalk does NOT currently check that this is true.
setPortDirectionAtomic: rwFlag
This method actually writes to a hardware register (0xBFD200) that
controls the direction of the Parallel Port Data bits. It's NOT
needed for reading & writing to the Parallel Port.
sendPortControlBits: newBits
This method actually writes to a hardware register (0xBFD101) that
holds the Parallel Port control bits.
NOTE: Only the 3 least-significant bits will be written to the
hardware. This is to prevent your code from interfering
with the Serial device.
readControlBitsMaskedBy: ctrlMask
This method actually reads a hardware register (0xBFD101) that
holds the Parallel Port control bits.
NOTE: Only the 3 least-significant bits have any meaning for the
Parallel Port. Use ctrlMask of seven (7).
privateOpen: parallelFlags
This is the method that actually opens the Parallel Port. Use
new: instead.
close
This is the method that closes the Parallel Port.
new
This method overrides the new method in Class & simply
prints an error string.
privateSetup: parallelFlags
This is a private method, use new instead.
new: parallelFlags
This method overrides the new: method in Class & makes this class
a Singleton Class. Use this method instead of open:
testToggleCtrlBits: loopCount
Use this method only for verifying operation of control bits
using test equipment of some kind on your hardware!
loopCount of 1 is around 60 milli-Seconds so do NOT use values
greater than 60,000 (over 1 hour).
testToggleDataBits: loopCount
Use this method only for verifying operation of data bits
using test equipment of some kind on your hardware!
loopCount of 1 is around 60 milli-Seconds so do NOT use values
greater than 60,000 (over 1 hour).

1.27 Printer Class (Parent Class = Device):

Not implemented yet!
<primitive 225 cmd# printername>
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!
Proposed methods are:

1.28 SCSI Class (Parent Class = Device):

Not implemented yet!
<primitive 226 cmd# SCSIname>
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!
Proposed methods are:

1.29 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
& are totally different from the methods in this class.
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.

1.30 TrackDisk Class (Parent Class = Device):

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!

(See the contents of AmigaTalk:TestFiles/TestTrackDisk or type:

)r AmigaTalk:TestFiles/TestTrackDisk into the Command Line string Gadget

when AmigaTalk is running to see how this Class is used).

Valid methods are:

new: newDiskName

Initialize a new instance of TrackDisk Class & call it newDiskName.

NOTE: Diskname is in the form: DFx:, where x is the drive

number (0 to 3).

openDisk: unitNumber

Open a TrackDisk Device & add it to the AmigaTalk system.

NOTE: unitNumber should be a number from 0 to 3 & has to match the

drive number in newDiskName (see new: newDiskName).

closeDisk

Close the TrackDisk device & remove it from the AmigaTalk system.

readTrack: trackNumber

Read the given trackNumber from a TrackDisk & return it as a

ByteArray.

writeTrack: outputBytes track: trackNumber

Write a ByteArray to the given trackNumber for a TrackDisk. Returns

an Integer, with zero meaning success.

getErrorString

Translate the current TrackDisk error number to an informative string.

getSectorSize

Return the TrackDisk SectorSize (normally 512 bytes).

getTrackSize

Return the TrackDisk (SectorSize * number of TrackSectors).

Normally 11 TrackSectors * 512 bytes.

getDeviceType

Return a string indicating the TrackDisk type. Known values are:

"Type = Direct Access, Removable | Non-removable media."

"Type = Sequential Access, Removable | Non-removable media."

"Type = Printer, Removable | Non-removable media."

"Type = Processor, Removable | Non-removable media."

"Type = W.O.R.M., Removable | Non-removable media."

"Type = CD-ROM, Removable | Non-removable media."

"Type = Scanner, Removable | Non-removable media."

"Type = Optical Disk, Removable | Non-removable media."

"Type = Medium Changer, Removable | Non-removable media."

"Type = Communication, Removable | Non-removable media."

"Type = Unknown, Removable | Non-removable media."

getDriveType

Return an Integer Object indicating the type of TrackDisk device.

1 = 3-1/2" Drive.

2 = 5-1/4" Drive.

3 = 3-1/2" Drive spinning at 150 RPM.

displayDriveType

Display in the status window a String Object indicating the type
of TrackDisk device. The Strings will be:

"3-1/2" Floppy Disk."

"5-1/4" Floppy Disk."

"3-1/2" Floppy spinning at 150 RPM."

getNumberOfTracks

Return the number of tracks that the TrackDisk recognizes.
Normally 160, or 80 Cylinders.

getTotalSectors

Return the total number of sectors present in the TrackDisk.
Normally 1760 for a 3-1/2" Amiga disk.

getTotalSize

Return the total number of bytes present in the TrackDisk.
Normally 1760 * 512 for a 3-1/2" Amiga disk.

seekTrack: trackNumber

Move the head of the drive to a specific track.
NOTE: Seeking is NOT reading! The heads only move, nothing else.

clearReadBuffer

Tell the TrackDisk Device to mark the track buffer as invalid, forcing
a re-read of the disk on the next operation.

isDiskPresent

Return true if a disk is in the TrackDisk, false otherwise.

isWriteProtected

Return true if a disk is write-protected, false otherwise.

displayBytes: byteArray title: reqTitle

Display an array of bytecodes (such as those returned from readTrack:
for instance) in a Requester. This Method is also used in ByteArray .

WARNING: The following methods are considered dangerous (to the health
of your floppy disks), and should NOT be used by amateurs.

turnMotorOn

Start the TrackDisk motor. This method is used by all the other
methods as required, so it doesn’t really have to be called by the
User.

turnMotorOff

Turn off the TrackDisk motor. This method is used by all the other

methods as required, so it doesn't really have to be called by the
User.
ejectDisk
Tell the TrackDisk to send an eject disk command to the disk drive.
Return an Integer.
WARNING: Most Amiga disk drives won't know what to do with this
method, except return an error. Read your disk drive
instruction manuals to determine whether ejection is
supported.
formatTrack: trackNumber data: theDataBytes
Format the given trackNumber.
WARNING: This is a sure, quick way of trashing a disk. Make sure
you really know what you're doing!
readRawData: trackNumber
Read the given trackNumber from a TrackDisk & return it as a
ByteArray of MFM data.
NOTE: This method is only really useful for reading & decoding other
disk formats, such as MS-DOS (yuck!).
writeRawData: rawDataBytes track: trackNumber
Write a ByteArray of (MFM) data to the given trackNumber for a
TrackDisk.
NOTE: This method is only really useful for encoding & writing other
disk formats, such as MS-DOS (yuck!).
setSyncType: newSyncType
Change the sync-word that TrackDisk will use during reads & writes.
If newSyncType is zero, the drive will synchronize to the Index
hole in the floppy. All other values will tell the drive to
synchronize to the Sync Pattern (16r4489) in Track Gaps.