# Circle

## Primary Classes

### Kernel Members

#### CMemorySystem

#### CActLED

#### CKernelOptions

#### CDeviceNameService

#### CScreenDevice

#### CMiniUartDevice

#### CExceptionHandler

#### CInterruptSystem

#### CTimer

#### CSerialDevice

#### CLogger

#### CScheduler

#### CDWHCIDevice

#### CTouchScreenDevice

#### CMultiCoreSupport

### Primary Utility Classes

#### CString

#### CGPIOPin

#### CGPIOClock

#### CGPIOManager

#### CI2CMaster

#### CMachineInfo

#### CSpinLock

### CDevice

#### CUSBMIDIDevice

#### CUSBBluetoothDevice

#### CUSBBulkOnlyMassStorageDevice

#### CUSBCDCEthernetDevice

#### CUSBHIDDevice

#### CUSBMouseDevice

#### CUSBKeyboardDevice

#### CMouseDevice

### Addons

#### SDCard

Driver for SD card access using the internal EMMC controller (by John Cronin)

#### fatfs

Generic FAT file system module with LFN support (by ChaN)

#### linux (parts may be used by other addons)

Linux kernel device driver emulation (used by HDMI sound and accelerated graphics)

#### UGUI

#### vc4 (may be used by other addons?)

HDMI sound and accelerated graphics (EGL, OpenGL ES, OpenVG, Dispmanx) support

### Primary Available

#### CTime

#### CCPUThrottle

#### CI2CSlave

#### CSPIMaster

#### CSPIMasterAUX

#### CSPIMasterDMA

## Low Level

### Low Level Likely/Possibly Used

#### CMouseBehaviour

#### CClassAllocator

#### CDMAChannel

#### CGPIOPinFIQ

#### CPageTable

#### CPtrArray

#### CPtrList

#### CSynchronizationEvent

#### CTask

#### CCharGenerator

#### CTranslationTable

#### CBcmFrameBuffer

#### CBcmMailBox

#### CbcmPropertyTags

### Not Used to my knowledge

#### CBcmRandomNumberGenerator

#### CNullDevice

#### CUserTimer

#### CVirtualGPIOPin

#### CTracer

#### Unused Audio System

##### CSoundBaseDevice

##### CI2SSoundBaseDevice

##### CPWMOutput

##### CPWMSoundDevice

##### CPWMSoundBaseDevice

### USB library Hidden

#### CDWHCIDevice

#### CDWHCIFrameScheduler

#### CDWHCIFrameSchedulerNonPeriodic

#### CDWHCIFrameSchedulerNoSplit

#### CDWHCIFrameSchedulerPeriodic

#### CDWHCIRegister

#### CDWHCIRootPort

#### CDWHCITransactionQueue

#### CDWHCITransferStageData

#### CLAN7800Device

#### CSMSC951xDevice

#### CMACAddress

#### CNetDevice

#### CUSBConfigurationParser

#### CUSBDevice

#### CUSBDeviceFactory

#### CUSBEndpoint

#### CUSBFunction

#### CUSBHostController

#### CUSBRequest

#### CUSBStandardHub

#### CUSBString

### USB Library not used

#### CUSBGamePadDevice

#### CUSBGamePadPS3Device

#### CUSBGamePadPS4Device

#### CUSBGamePadStandardDevice

#### CUSBGamePadSwitchProDevice

#### CUSBGamePadXbox360Device

#### CUSBGamePadXboxOneDevice

#### CUSBPrinterDevice

### Input library (probably hidden)

#### CConsole

#### CKeyboardBehaviour

#### CKeyboardBuffer

#### CKeyMap

#### CLineDiscipline

### FS library (possibly used / usable)

#### CPartition

#### CPartitionManager

### Net library

#### CARPHandler

#### CChecksumCalculator

#### CDHCPClient

#### CDNSClient

#### CHTTPClient

#### CHTTPDaemon

#### CICMPHandler

#### CIPAddress

#### CLinkLayer

#### CMQTTClient

#### CMQTTReceivePacket

#### CMQTTSendPacket

#### CNetConfig

#### CNetConnection

#### CNetDeviceLayer

#### CNetQueue

#### CNetSocket

#### CNetSubSystem

#### CNetTask

#### CNetworkLayer

#### CNTPClient

#### CNTPDaemon

#### CRetransmissionQueue

#### CRetransmissionTimeoutCalculator

#### CRouteCache

#### CSocket

#### CSysLogDaemon

#### CTCPConnection

#### CTCPRejector

#### CTFTPDaemon

#### CTransportLayer

#### CUDPConnection

## Not Used

### FAT FS library (unused)

#### CFAT

#### CFATInfo

#### CFATDirectory

#### CFATFileSystem

#### CFATCache

### Bluetooth library (unused)

#### CBTDeviceManager

#### CBTHCILayer

#### CBTInquiryResults

#### CBTLogicalLayer

#### CBTQueue

#### CBTSubSystem

#### CBTTask

#### CBTUARTTransport

## Currently Unused Addons

### dio

Library providing access to the spi\_dio board by BitWizard.nl

### display

Library providing drivers for displays (e.g. LCD dot-matrix)

### OneWire

Support library for 1-wire devices (by Paul Stoffregen)

### Properties

Library providing access to configuration properties saved in a file

### qemu

Demos for using Circle with QEMU

### rtc

Library providing drivers for real-time clocks (RTC)

### sensor

Drivers for I2C and other sensor devices

### Spectrum

ZX Spectrum screen emulator class (by Jose Luis Sanchez)

### tftpfileserver

TFTP file server supporting kernel image and firmware updates

### webconsole

Library providing remote access to the system log using a web browser

### WS28XX

Driver for WS28XX controlled LED stripes (by Arjan van Vught)

# Initial Demonstration 4-track recorder

## UGUI Addon

### CUIFramerwork (currently CUGUI c++ class)

### CWindow

### CButton

### CCheckbox

### CTextbox

## App Framework

### CApplication

### CMenu

### CTitleBar

## Widgits

### CVuMeter

### CDialogSelectDevice

### CButtonDeviceSelect

### CTrackDisplay

### CRecordTrack

## Test Application

### CWindowStatus

### CWindowRecord

### CWindowMain

# UISystem

## wsEvent

has an object type

has an object id within the type

has an event type

has an event id

has a 32 bit event data

- u32 usually

- can be float

## wsObject

has type

has id

has parent

has doubly linked list of children

### wsEventHandler …

## wsEventHandler : CObject

### wsWindow ….

### wsAudioControl …

### AudioObject ...

## wsWindowBase

has relative coordinates

has fill, outline, 3d, and other rendering characteriscs

has separate virtual and visible areas, start offsets, etc

has onDraw(), onFocus(), etc, methods for event handling

has window management sub objects inluding scrollbars (onScroll, etc)

has drawing management members like z\_order, clip\_regions, etc.

### wsWindow

#### wsPopupWindow

#### wsTopLevelWindow

has title bar sub-window

has status bar sub-window

##### wsFrame

##### wsDialog

###### wsAlertDialog

###### wsFileDialog

###### wsProgressDialog

#### wsSplitterWindow

#### wsTipWindow

### wsControl

getValue(), setValue()

can be bound to an Audio object and control id

#### wsStaticFill

#### wsStaticLine

#### wsStaticText

#### wsButton

inlcudes text and/or bitmap

includes toggle states

##### wsMenuButton

has a command id

can be a separator

can be part of a wsMenu

#### wsCheckBox (square or round)

includes optional text

includes toggle states

can be part of a wsCheckBoxList

#### wsNumericControl

##### wsProgressBar (display)

##### wsVUMeter (display)

##### wsRotaryControl

##### wsSlider

###### wsScrollbar

a slider specifically attached to a window

#### wsTextControl

##### wsNumericTextControl

an editable integer or floating point number

#### wsBitmap

##### wsImage

###### wsImagePNG

###### wsImageJPG

###### wsImageGIF

###### wsImageBMP

#### Combo Objects (wsCombObject)

##### wsCheckBoxList

A list of related checkboxes (i.e. mutually exclusive)

##### wsComboControl

A control that also includes a button ans a popup wsListControl

###### wsComboBox

A wsCombControl with API optimized for strings

##### wsListControl

A list of atomic objects, probably static text, menu buttons, etc

May be horizontal or vertical

a group of objects organized into a visual group

may have an outline, all be enabled at once, etc

###### wsTextListControl

Vertical wsListControl optimized for strings

##### wsSpinner

A control that contains buttons to increment and decrement a value

###### wsNumericSpinner

one that works with numbers

###### wsListSpinner

one that works with a list of controls (i.e. text or other objects)

##### wsTreeControl

A list of objects organized and manipulated hiearchially

##### wsMenu

A wsListControl of wsMenuButtons under a button

can be vertical or horizontal

##### wsMenuTree

A hiearchial menu tree under a single button

can be vertical or horizontal

##### wsMenuBar

A list of wsMenuTrees forming the mnu system of a toolbar

the system menu

can be vertical or horizontal

#### High Level Objects

##### wsDirCtrl

a control that allows for the selection of, and/or creation and deletion of, directories in the file system

may allow for selection of multiple devices

may be implemented as a tree and/or general file manager

##### wsFileCtrl

a control that allows for the selection, and/or naming of new and deletion of old, files in a given directory

##### wsRichTextCtrl

#### App Specific

##### wsTitleBar

Combines a possible menu bar, title, and list of app defined controls

Can be tied to specific windows and understands window activation context

# AudioSystem

## BCMPCM

## Wire/Arduino support/etc

## AudioObject

Have a standardized control (parameter) interface

Can transmit change notification events

Can be bound to one ore more UI objects for direct notification, access to specific members, etc

### AudioCodec

i2sDevices typically inititalize the BCMPCM and present a control interface

I2S streams are related to a codec. It is up to the client to ensure that the correct objects are instantiated.

I2S streams inherit their half of the controls from the codec. i.e. An AudioInputI2S object will reduntantly present the input controls from the related AudioCodec, like input selection and input volumes. The AudioOutputI2S device will inherit and present the main Volume control and any other controls for the codec.

By default Codec controls tend to be grouped as “output” controls, as they ultimately control the output sound of the whole device.

It may be possible for a Codec to present controls that are neither input or output. Perhaps it has a set of controls for monitoring the temperature of the device.

#### AudioControlCS42448 (slave, though it generates MCLK)

requires AudioInputTDM and AudioOutputTDM

#### AudioControlSGTL5000

requires AudioInputI2S and AudioOutputI2S

#### AudioControlWM8731

requires AudioInputI2S and AudioOutputI2S

#### AudioControlWM8731Slave (rpi is not recommended as I2S master)

requires AudioInputI2S and AudioOutputI2S

### AudioStream

AudioStreams can be grouped for the UI as hardware inputs and outputs, effects, synths, and so on.

#### Inputs and Outputs

Hardware input and output objects are special.

They are typically the objects that drive the update() loop.

Inputs particularly are sorted first topologically.

Outputs are generally sorted last.

It may be desireable to have low priority objects that are updated AFTER the actual audio output has taken place, that just “reset” themselves if the audio system performance overflows.

\*\* It could be possible to have the update() loop monitor the time() and cutoff any optional devices, clearing their input queues and not calling update() on them.

##### AudioInputI2S

##### AudioOutputI2S

##### AudioInputTDM

##### AudioOutputTDM

##### AudioInputTeensyQuad

makes direct use of BCM\_PCM

##### AudioOutputTeensyQuad

makes direct use of BCM\_PCM

#### Effects

##### AudioEffectFreeverb

##### AudioEffectReverb

#### Synths

##### AudioSynthWaveformSine

##### AudioSynthWaveformSineHires

##### AudioSynthWaveformSineModulated

#### Mixers

##### AudioMixer4

#### Tools

##### AudioAnalyzePeak

##### AudioAnalyzeRMS

#### Other

##### AudioRecorder

# MidiSystem

# System

The system exists to combine audio, midi, and ui devices and objects into an application.

The generic audio, midi, and user interface systems are always linked into the kernel.

Each subsystem may be explicitlly disabled in the kernel to ensure there is (vitually) no impact from not using a given subsystem.

The objects available within each subsystem at runtime are determined by compile time linking references in the top most (application) cpp file.

This is facilitated by the following methods:

- uiSystem : the client supplies a wsApplication::Create() method which creates the ui object tree.

-audioSystem: the objects may be linked statically (danger!) and/or created dynamically in the setup() and loop() methods

-midiSystem: USB midi devices are added automatically if USB is enabled.

# Circle Devices

## Named Devices (registered with CDeviceNameService)

Only certain USB “functions” are registered with the DeviceNameService.

Classes derived from CDevice are added to the CDeviceNameService with the following names

### null = CNullDevice

### tty1 = CScreenDevice

### ttyS1 = CSerialDevice

### ttyS2 = CMiniUartDevice

### console = input/CConsole

### touch1 = input/CTouchScreenDevice (ft5406 based touch screen device)

### mouseN = input/CMouseDevice

### sndi2s = CI2SSoundBaseDevice

### ethN = net/CNetDevice

### btusbN = usb/CUSBBluetoothDevice

### upadN = usb/CUSBGamePadDevice

### ukbdN = usb/CUSBKeyboardDevice

### umidiN = usb/CUSBMIDIDevice

### uprnN = usb/CUSBPrinterDevice

### uhubN = usb/CUSBStandardHub

### umsdN = usb/CUSBBulkOnlyMassStorageDevice

### umsdN-1 = fs/CPartition of usmdN (via CUSBBulkOnlyMassStorageDevice::CPartitionManager)

### ttyBT1 = usb/CBTUARTTransport

### emmc1 = addon/SDCard/CEMMCDevice

### sndvchiq = addon/vc4/sound/CVCHIQSoundBaseDevice

## Devices Used by Name (not including sample programs)

These are known cases where the system itself uses DeviceNameService::GetDevice() of devices by name

### in bt/CBTHCILayer: ubt1, ttyBT1

### in bt/ CBTSubSystem: ubt1, ttyS1 - to see if should create “ttyBT1=usb/CBTUARTTransport”

### in input/CConsole: ukbd1, tty1 - requires both to become “console = input/CConsole”

### in net/CNetDeviceLayer: eth0 - to fail initialization if not exists

### addon/fatfs/diskio.cpp:: disk\_initialize(): emmc1, usmd1, usmd2, or usmd3

calls GetDevice(s\_pVolumeName[pdrv])

which may be one of emmc1, usmd1, usmd2, or usmd3

### addon/ugui/CUGUI: mouse1, touch1 - for initialization

# USBDevice::GetNames()

returns a possibly comma delimited string containing the next two names

# USBDevice::GetName(DeviceNameVendor)

I am used to a 4 digit hex pair of VID and PID (vendor\_id and product\_id) from Windows Device Manager

USBDevice::GetName(DeviceNameVendor) combines these into one string, dropping any leading zeros

ven2011-715 = VID(0x2011) PID(0x0715) = MPKmini2

ven9e8-34 = VID(0x09E8) PID(0x0034) = MPD218

# USBDevice::GetName(DeviceNameDevice)

Returns a string for the outer level device descriptor

where X=class, Y=subclass, and Z=protocol:

dev9-0-2 = class(USB\_HUB) subclass(0) protocol(02= Hi-speed hub with multiple TTs)

This is usually “unknown” for HID and MIDI devices which specify the classes via the interfaces

# USBFunction::GetInterfaceName()

This naming convention is followed into interfaces

with X-Y-Z for class, subclass, and protocol

int1-1-0 = class(AUDIO\_DEVICE) subclass(AUDIOCONTROL) = not supported

int1-3-0 =class(AUDIO\_DEVICE) subclass(MIDISTREAMING)

int3-0-0=class(HID\_DEVICE) subclass(none) protcol(none)

## USBDevice

### GetHubAddress()

### GetHubPortNumber()

### CString \*GetNames (void) const

### CString \*GetName (TDeviceNameSelector Selector) const

DeviceNameVendor

DeviceNameDevice

DeviceNameUnknown

## USBFunction:CDevice

## Other Device Names

Other names are harder to figure out and the devices can be recursive.

## NET Device Names

## USB Device Names

## Other names are harder to figure out.

These are mostly NET and USB devices and subdevices.

USBFunction : public USBDevice

CUSBBulkOnlyMassStorageDevice : public USBFunction

- has a CPartitionManager

- where each partition is given a different device name