Namespace HackRFDotnet.Api

Namespaces

HackRFDotnet.Api.Extensions

HackRFDotnet.Api.Interfaces

HackRFDotnet.Api.Services

HackRFDotnet.Api.Streams

HackRFDotnet.Api.Utilities

Classes

DigitalRadioDevice

Radio Device to receive IQ Samples with.

Structs

Bandwidth

Bandwidth of a signal sample.

Frequency

Frequency of a signal.

Hertz

Number of oscillations per second.

SampleRate

Class DigitalRadioDevice

Namespace: HackRFDotnet.Api

Assembly: HackRFDotnet.dll

Radio Device to receive IQ Samples with.

```
public class DigitalRadioDevice : IDigitalRadioDevice, IDisposable
```

Inheritance

object ← DigitalRadioDevice

Implements

IDigitalRadioDevice , IDisposable

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Extension Methods

RfDeviceExtensions.AttenuateAmplification(DigitalRadioDevice)

Fields

DevicePtr

```
public readonly HackRFDevice* DevicePtr
```

Field Value

HackRFDevice*

Properties

Bandwidth

Current capture bandwidth.

```
public Bandwidth Bandwidth { get; set; }
```

Property Value

Bandwidth

DeviceSamplingRate

Current capture sample rate.

```
public SampleRate DeviceSamplingRate { get; set; }
```

Property Value

SampleRate

DeviceStream

The immutable stream running on this device.

```
public IIQStream? DeviceStream { get; }
```

Property Value

IIQStream?

Frequency

Current frequency tuned to.

```
public Frequency Frequency { get; set; }
```

Property Value

Frequency

IsConnected

Is the device connected to the usb host in the native library?

```
public bool IsConnected { get; }
```

Property Value

bool

Methods

Dispose()

Dispose the Rf Device from the library.

```
public void Dispose()
```

SetAmplifications(uint, uint, bool)

Set the Lna, Vga, and Internal amp settings for the Rf Device.

```
public void SetAmplifications(uint lna, uint vga, bool internalAmp)
```

Parameters

`lna` uint

`vga` uint

`internalAmp` bool

SetFrequency(Frequency, Bandwidth)

Set the tuning frequency and bandwidth for the Rf Device.

```
public bool SetFrequency(Frequency radioFrequency, Bandwidth bandwidth)
```

Parameters

`radioFrequency` Frequency

`bandwidth` Bandwidth

Returns

bool

SetSampleRate(SampleRate)

Set the sample rate for the radio device to capture data at. This will also set the baseband filter the smallest filter that fits the sample rate's Nyquist frequency cutoff.

```
public void SetSampleRate(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

StartRx(HackRFSampleBlockCallback)

Start receiving data from the RfDevice.

```
public bool StartRx(HackRFSampleBlockCallback rxCallback)
```

Parameters

`rxCallback` HackRFSampleBlockCallback

Returns

bool

Exceptions

NullCallbackException

StopRx()

Stop receiving data from the Rf Device.

```
public bool StopRx()
```

Returns

bool

Struct Bandwidth

Namespace: HackRFDotnet.Api

Assembly: HackRFDotnet.dll

Bandwidth of a signal sample.

```
public readonly record struct Bandwidth : IEquatable<Bandwidth>
```

Implements

IEquatable<Bandwidth>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

Bandwidth(long)

```
public Bandwidth(long hz)
```

Parameters

`hz` long

Bandwidth(Hertz)

```
public Bandwidth(Hertz hertz)
```

Parameters

`hertz` Hertz

Properties

Hz

```
public long Hz { get; }
```

Property Value

long

NyquistSampleRate

https://en.wikipedia.org/wiki/Nyquist_rate The smallest sample rate that can be used to represent the bandwidth.

```
public SampleRate NyquistSampleRate { get; }
```

Property Value

SampleRate

Methods

FromGHz(double)

```
public static Bandwidth FromGHz(double ghz)
```

Parameters

ghz double

Returns

Bandwidth

FromHz(Hertz)

```
public static Bandwidth FromHz(Hertz hz)
```

Parameters

`hz` Hertz

Returns

Bandwidth

FromHz(long)

```
public static Bandwidth FromHz(long hz)
```

Parameters

`hz` long

Returns

Bandwidth

FromKHz(double)

```
public static Bandwidth FromKHz(double khz)
```

Parameters

`khz` double

Returns

Bandwidth

FromMHz(double)

```
public static Bandwidth FromMHz(double mhz)
```

Parameters

mhz double

Returns

Bandwidth

Operators

operator +(Bandwidth, Bandwidth)

```
public static Bandwidth operator +(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

Bandwidth

operator /(Bandwidth, int)

```
public static Bandwidth operator /(Bandwidth a, int b)
```

Parameters

a Bandwidth

b int

Returns

Bandwidth

operator /(Bandwidth, double)

```
public static Bandwidth operator /(Bandwidth a, double b)
```

Parameters

a Bandwidth

b double

Returns

Bandwidth

operator >(Bandwidth, Bandwidth)

```
public static bool operator >(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

bool

operator >=(Bandwidth, Bandwidth)

```
public static bool operator >=(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

bool

implicit operator Frequency(Bandwidth)

```
public static implicit operator Frequency(Bandwidth b)
```

Parameters

b Bandwidth

Returns

Frequency

implicit operator Hertz(Bandwidth)

```
public static implicit operator Hertz(Bandwidth b)
```

Parameters

b Bandwidth

Returns

Hertz

operator <(Bandwidth, Bandwidth)

```
public static bool operator <(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

bool

operator <=(Bandwidth, Bandwidth)

```
public static bool operator <=(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

bool

operator %(Bandwidth, Bandwidth)

```
public static Bandwidth operator %(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

Bandwidth

operator *(Bandwidth, int)

```
public static Bandwidth operator *(Bandwidth a, int b)
```

Parameters

a Bandwidth

b int

Returns

Bandwidth

operator *(Bandwidth, double)

```
public static Bandwidth operator *(Bandwidth a, double b)
```

Parameters

a Bandwidth

b double

Returns

Bandwidth

operator -(Bandwidth, Bandwidth)

```
public static Bandwidth operator -(Bandwidth a, Bandwidth b)
```

Parameters

a Bandwidth

b Bandwidth

Returns

Bandwidth

operator -(Bandwidth)

```
public static Bandwidth operator -(Bandwidth a)
```

Parameters

a Bandwidth

Returns

Bandwidth

Struct Frequency

Namespace: HackRFDotnet.Api

Assembly: HackRFDotnet.dll

Frequency of a signal.

```
public readonly record struct Frequency : IEquatable<Frequency>
```

Implements

IEquatable<Frequency>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

Frequency(long)

```
public Frequency(long hz)
```

Parameters

`hz` long

Frequency(Hertz)

```
public Frequency(Hertz hertz)
```

Parameters

`hertz` Hertz

Properties

Hz

```
public long Hz { get; }
```

Property Value

long

Methods

FromGHz(double)

```
public static Frequency FromGHz(double ghz)
```

Parameters

ghz double

Returns

Frequency

FromHz(Hertz)

```
public static Frequency FromHz(Hertz hz)
```

Parameters

hz Hertz

Returns

Frequency

FromHz(long)

```
public static Frequency FromHz(long hz)
```

Parameters

`hz` long

Returns

Frequency

FromKHz(double)

```
public static Frequency FromKHz(double khz)
```

Parameters

`khz` double

Returns

Frequency

FromMHz(double)

```
public static Frequency FromMHz(double mhz)
```

Parameters

`mhz` double

Returns

Frequency

Operators

operator +(Frequency, Frequency)

```
public static Frequency operator +(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

Frequency

operator /(Frequency, int)

```
public static Frequency operator /(Frequency a, int b)
```

Parameters

a Frequency

b int

Returns

Frequency

operator /(Frequency, double)

```
public static Frequency operator /(Frequency a, double b)
```

Parameters

a Frequency

b double

Returns

Frequency

operator >(Frequency, Frequency)

```
public static bool operator >(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

bool

operator >=(Frequency, Frequency)

```
public static bool operator >=(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

bool

implicit operator Bandwidth(Frequency)

```
public static implicit operator Bandwidth(Frequency f)
```

Parameters

f Frequency

Returns

Bandwidth

implicit operator Hertz(Frequency)

```
public static implicit operator Hertz(Frequency f)
```

Parameters

f Frequency

Returns

Hertz

operator <(Frequency, Frequency)

```
public static bool operator <(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

bool

operator <=(Frequency, Frequency)

```
public static bool operator <=(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

bool

operator %(Frequency, Frequency)

```
public static Frequency operator %(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

Frequency

operator *(Frequency, int)

```
public static Frequency operator *(Frequency a, int b)
```

Parameters

a Frequency

b int

Returns

Frequency

operator *(Frequency, double)

```
public static Frequency operator *(Frequency a, double b)
```

Parameters

a Frequency

b double

Returns

Frequency

operator -(Frequency, Frequency)

```
public static Frequency operator -(Frequency a, Frequency b)
```

Parameters

a Frequency

b Frequency

Returns

Frequency

operator -(Frequency)

```
public static Frequency operator -(Frequency a)
```

Parameters

a [Frequency]

Returns

[Frequency]

Struct Hertz

Namespace: HackRFDotnet.Api

Assembly: HackRFDotnet.dll

Number of oscillations per second.

```
public readonly record struct Hertz : IEquatable<Hertz>
```

Implements

IEquatable<Hertz>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

Hertz(long)

```
public Hertz(long hz)
```

Parameters

hz long

Properties

Hz

Number of oscillations per second.

```
public long Hz { get; }
```

Property Value

long

Khz

Number of oscillations per second divided by 1,000

```
public double Khz { get; }
```

Property Value

double

Mhz

Number of oscillations per second divided by 1,000,000

```
public double Mhz { get; }
```

Property Value

double

Methods

FromGHz(double)

```
public static Hertz FromGHz(double ghz)
```

Parameters

`ghz` double

Returns

Hertz

FromHz(long)

```
public static Hertz FromHz(long hz)
```

Parameters

`hz` long

Returns

Hertz

FromKHz(double)

```
public static Hertz FromKHz(double khz)
```

Parameters

`khz` double

Returns

Hertz

FromMHz(double)

```
public static Hertz FromMHz(double mhz)
```

Parameters

`mhz` double

Returns

Hertz

Operators

operator +(Hertz, Hertz)

```
public static Hertz operator +(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

Hertz

operator /(Hertz, int)

```
public static Hertz operator /(Hertz a, int b)
```

Parameters

a Hertz

b int

Returns

Hertz

operator /(Hertz, double)

```
public static Hertz operator /(Hertz a, double b)
```

Parameters

a Hertz

b double

Returns

Hertz

operator >(Hertz, Hertz)

```
public static bool operator >(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

bool

operator >=(Hertz, Hertz)

```
public static bool operator >=(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

bool

operator <(Hertz, Hertz)

```
public static bool operator <(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

bool

operator <=(Hertz, Hertz)

```
public static bool operator <=(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

bool

operator %(Hertz, Hertz)

```
public static Hertz operator %(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

Hertz

operator *(Hertz, int)

```
public static Hertz operator *(Hertz a, int b)
```

Parameters

a Hertz

b int

Returns

Hertz

operator *(Hertz, double)

```
public static Hertz operator *(Hertz a, double b)
```

Parameters

a Hertz

b double

Returns

Hertz

operator -(Hertz, Hertz)

```
public static Hertz operator -(Hertz a, Hertz b)
```

Parameters

a Hertz

b Hertz

Returns

Hertz

operator -(Hertz)

```
public static Hertz operator -(Hertz a)
```

Parameters

a Hertz

Returns

Hertz

Struct SampleRate

Namespace: HackRFDotnet.Api

Assembly: HackRFDotnet.dll

```
public record struct SampleRate : IEquatable<SampleRate>
```

Implements

IEquatable<SampleRate>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

SampleRate(long)

```
public SampleRate(long sps)
```

Parameters

`sps` long

SampleRate(Hertz)

```
public SampleRate(Hertz hertz)
```

Parameters

`hertz` Hertz

Properties

Ksps

```
public double Ksps { get; }
```

Property Value

double

Msps

```
public double Msps { get; }
```

Property Value

double

NyquistFrequencyBandwidth

https://en.wikipedia.org/wiki/Nyquist_frequency The largest bandwidth this sample rate can represent.

```
public Bandwidth NyquistFrequencyBandwidth { get; }
```

Property Value

Bandwidth

Sps

```
public long Sps { get; }
```

Property Value

long

Methods

FromGsps(double)

```
public static SampleRate FromGsps(double ghz)
```

Parameters

`ghz` double

Returns

SampleRate

FromKsps(double)

```
public static SampleRate FromKsps(double khz)
```

Parameters

`khz` double

Returns

SampleRate

FromMsps(double)

```
public static SampleRate FromMsps(double mhz)
```

Parameters

`mhz` double

Returns

SampleRate

FromSps(long)

```
public static SampleRate FromSps(long hz)
```

Parameters

hz long

Returns

SampleRate

ToString()

Returns the fully qualified type name of this instance.

```
public override string ToString()
```

Returns

string

The fully qualified type name.

Operators

operator +(SampleRate, SampleRate)

```
public static SampleRate operator +(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

SampleRate

operator /(SampleRate, int)

```
public static SampleRate operator /(SampleRate a, int b)
```

Parameters

a SampleRate

b int

Returns

SampleRate

operator /(SampleRate, double)

```
public static SampleRate operator /(SampleRate a, double b)
```

Parameters

a SampleRate

b double

Returns

SampleRate

operator >(SampleRate, SampleRate)

```
public static bool operator >(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

bool

operator >=(SampleRate, SampleRate)

```
public static bool operator >=(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

bool

implicit operator Hertz(SampleRate)

```
public static implicit operator Hertz(SampleRate s)
```

Parameters

s SampleRate

Returns

Hertz

operator <(SampleRate, SampleRate)

```
public static bool operator <(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

bool

operator <=(SampleRate, SampleRate)

```
public static bool operator <=(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

bool

operator %(SampleRate, SampleRate)

```
public static SampleRate operator %(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

SampleRate

operator *(SampleRate, int)

```
public static SampleRate operator *(SampleRate a, int b)
```

Parameters

a SampleRate

b int

Returns

SampleRate

operator *(SampleRate, double)

```
public static SampleRate operator *(SampleRate a, double b)
```

Parameters

a SampleRate

b double

Returns

SampleRate

operator -(SampleRate, SampleRate)

```
public static SampleRate operator -(SampleRate a, SampleRate b)
```

Parameters

a SampleRate

b SampleRate

Returns

SampleRate

operator -(SampleRate)

```
public static SampleRate operator -(SampleRate a)
```

Parameters

a SampleRate

Returns

SampleRate

Namespace HackRFDotnet.Api.Extensions

Classes

HostBuilderExtensions

RfDeviceExtensions

Class HostBuilderExtensions

Namespace: HackRFDotnet.Api.Extensions

Assembly: HackRFDotnet.dll

```
public static class HostBuilderExtensions
```

Inheritance

object ← HostBuilderExtensions

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

UseFirstRadioDevice(HostBuilder, SampleRate)

Will automatically open the first device, create a stream, and add the instances as singletons to the DI host.

```
public static HostBuilder UseFirstRadioDevice(this HostBuilder hostBuilder,
SampleRate sampleRate)
```

Parameters

`hostBuilder` HostBuilder

`sampleRate` SampleRate

Returns

HostBuilder

Exceptions

Exception

UseRfDeviceController(HostBuilder)

You must open the IQDeviceStream manually for each device managed via the HackRFDotnet.Api.Services.RfDeviceService

```
public static HostBuilder UseRfDeviceController(this HostBuilder hostBuilder)
```

Parameters

`hostBuilder` HostBuilder

Returns

HostBuilder

Class RfDeviceExtensions

Namespace: HackRFDotnet.Api.Extensions

Assembly: HackRFDotnet.dll

```
public static class RfDeviceExtensions
```

Inheritance

object ← RfDeviceExtensions

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

AttenuateAmplification(DigitalRadioDevice)

```
public static void AttenuateAmplification(this DigitalRadioDevice rfDevice)
```

Parameters

`rfDevice` DigitalRadioDevice

Namespace HackRFDotnet.Api.Interfaces

Interfaces

IDigitalRadioDevice

IRfDeviceService

Interface IDigitalRadioDevice

Namespace: HackRFDotnet.Api.Interfaces

Assembly: HackRFDotnet.dll

```
public interface IDigitalRadioDevice
```

Properties

Bandwidth

```
Bandwidth Bandwidth { get; set; }
```

Property Value

Bandwidth

DeviceSamplingRate

```
SampleRate DeviceSamplingRate { get; set; }
```

Property Value

SampleRate

DeviceStream

```
IIQStream? DeviceStream { get; }
```

Property Value

IIQStream?

Frequency

```
Frequency Frequency { get; set; }
```

Property Value

Frequency

IsConnected

```
bool IsConnected { get; }
```

Property Value

bool

Methods

Dispose()

```
void Dispose()
```

SetAmplifications(uint, uint, bool)

```
void SetAmplifications(uint lna, uint vga, bool internalAmp)
```

Parameters

`lna` uint

`vga` uint

`internalAmp` bool

SetFrequency(Frequency, Bandwidth)

```
bool SetFrequency(Frequency radioFrequency, Bandwidth bandwidth)
```

Parameters

`radioFrequency` Frequency

`bandwidth` Bandwidth

Returns

bool

SetSampleRate(SampleRate)

```
void SetSampleRate(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

StartRx(HackRFSampleBlockCallback)

```
bool StartRx(HackRFSampleBlockCallback rxCallback)
```

Parameters

`rxCallback` HackRFSampleBlockCallback

Returns

bool

StopRx()

```
bool StopRx()
```

Returns

bool

Interface IRfDeviceService

Namespace: HackRFDotnet.Api.Interfaces

Assembly: HackRFDotnet.dll

```
public interface IRfDeviceService
```

Methods

ConnectToFirstDevice()

```
DigitalRadioDevice? ConnectToFirstDevice()
```

Returns

DigitalRadioDevice?

FindDevices()

```
HackRFDeviceList FindDevices()
```

Returns

HackRFDeviceList

Namespace HackRFDotnet.Api.Services

Classes

AnaloguePlayer

RfDeviceService

Class AnaloguePlayer

Namespace: HackRFDotnet.Api.Services

Assembly: HackRFDotnet.dll

```
public class AnaloguePlayer : IDisposable
```

Inheritance

object ← AnaloguePlayer

Implements

IDisposable

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

AnaloguePlayer(WaveSignalStream)

```
public AnaloguePlayer(WaveSignalStream signalStream)
```

Parameters

`signalStream` WaveSignalStream

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

PlayStreamAsync(Frequency, Bandwidth, SampleRate)

```
public virtual void PlayStreamAsync(Frequency centerOffset, Bandwidth bandwidth,
SampleRate audioRate)
```

Parameters

`centerOffset` Frequency

`bandwidth` Bandwidth

`audioRate` SampleRate

Class RfDeviceService

Namespace: HackRFDotnet.Api.Services

Assembly: HackRFDotnet.dll

```
public class RfDeviceService : IRfDeviceService
```

Inheritance

object ← RfDeviceService

Implements

IRfDeviceService

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

RfDeviceService()

```
public RfDeviceService()
```

Fields

RfDevices

```
public readonly List<DigitalRadioDevice> RfDevices
```

Field Value

List<DigitalRadioDevice>

Methods

ConnectToFirstDevice()

```
public DigitalRadioDevice? ConnectToFirstDevice()
```

Returns

DigitalRadioDevice?

FindDevices()

```
public HackRFDeviceList FindDevices()
```

Returns

HackRFDeviceList

Namespace HackRFDotnet.Api.Streams

Namespaces

HackRFDotnet.Api.Streams.Device

HackRFDotnet.Api.Streams.Exceptions

HackRFDotnet.Api.Streams.Interfaces

HackRFDotnet.Api.Streams.SignalProcessing

HackRFDotnet.Api.Streams.SignalStreams

Classes

SweepingIQStream

Structs

IQ

This represents a 32bit complex number. The real represents the InPhase Sin of real voltage measurement in time. The imaginary represents the Quadrature of the real voltage measurement in time. The relationship between the I and Q allow us to represent the signal in lower sample rate than it was captured.

InterleavedSample

HackRFDotnet.Api.Streams.InterleavedSample comes directly from the HackRF device in transfer chunks. Memory alignment allows us to ready and copy them into objects very quickly.

Class SweepingIQStream

Namespace: HackRFDotnet.Api.Streams

Assembly: HackRFDotnet.dll

```
public class SweepingIQStream
```

Inheritance

object ← SweepingIQStream

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

SweepingIQStream()

```
public SweepingIQStream()
```

Struct IQ

Namespace: HackRFDotnet.Api.Streams

Assembly: HackRFDotnet.dll

This represents a 32bit complex number. The real represents the InPhase Sin of real voltage measurement in time. The imaginary represents the Quadrature of the real voltage measurement in time. The relationship between the I and Q allow us to represent the signal in lower sample rate than it was captured.

```
public struct IQ
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

IQ(float, float)

```
public IQ(float real, float imaginary)
```

Parameters

`real` float

`imaginary` float

IQ(InterleavedSample)

```
public IQ(InterleavedSample interleavedSample)
```

Parameters

`interleavedSample` InterleavedSample

Fields

ImaginaryOne

```
public static readonly IQ ImaginaryOne
```

Field Value

IQ

Infinity

```
public static readonly IQ Infinity
```

Field Value

IQ

NaN

```
public static readonly IQ NaN
```

Field Value

IQ

One

```
public static readonly IQ One
```

Field Value

IQ

Zero

```
public static readonly IQ Zero
```

Field Value

IQ

Properties

I

Real

```
public float I { get; set; }
```

Property Value

float

Magnitude

```
public float Magnitude { get; }
```

Property Value

float

Phase

```
public float Phase { get; }
```

Property Value

float

Q

Imaginary

```
public float Q { get; set; }
```

Property Value

float

Methods

Abs(IQ)

```
public static float Abs(IQ value)
```

Parameters

`value` IQ

Returns

float

Add(IQ, IQ)

```
public static IQ Add(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

Add(IQ, float)

```
public static IQ Add(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

Add(float, IQ)

```
public static IQ Add(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

Conjugate(IQ)

```
public static IQ Conjugate(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Cos(IQ)

```
public static IQ Cos(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Cosh(IQ)

```
public static IQ Cosh(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Divide(IQ, IQ)

```
public static IQ Divide(IQ dividend, IQ divisor)
```

Parameters

`dividend` IQ

`divisor` IQ

Returns

IQ

Divide(IQ, float)

```
public static IQ Divide(IQ dividend, float divisor)
```

Parameters

`dividend` IQ

`divisor` float

Returns

IQ

Divide(float, IQ)

```
public static IQ Divide(float dividend, IQ divisor)
```

Parameters

`dividend` float

`divisor` IQ

Returns

IQ

Equals(object?)

Indicates whether this instance and a specified object are equal.

```
public override bool Equals(object? obj)
```

Parameters

`obj` object?

The object to compare with the current instance.

Returns

bool

true if `obj` and this instance are the same type and represent the same value; otherwise, false.

Equals(IQ)

```
public bool Equals(IQ value)
```

Parameters

`value` IQ

Returns

bool

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

A 32-bit signed integer that is the hash code for this instance.

Multiply(IQ, IQ)

```
public static IQ Multiply(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

Multiply(IQ, float)

```
public static IQ Multiply(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

Multiply(float, IQ)

```
public static IQ Multiply(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

Negate(IQ)

```
public static IQ Negate(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Reciprocal(IQ)

```
public static IQ Reciprocal(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Sin(IQ)

```
public static IQ Sin(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Sinh(IQ)

```
public static IQ Sinh(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Subtract(IQ, IQ)

```
public static IQ Subtract(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

Subtract(IQ, float)

```
public static IQ Subtract(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

Subtract(float, IQ)

```
public static IQ Subtract(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

Tan(IQ)

```
public static IQ Tan(IQ value)
```

Parameters

`value` IQ

Returns

IQ

ToString()

Returns the fully qualified type name of this instance.

```
public override string ToString()
```

Returns

string

The fully qualified type name.

ToString(string?)

```
public string ToString(string? format)
```

Parameters

`format` string?

Returns

string

ToString(IFormatProvider?)

```
public string ToString(IFormatProvider? provider)
```

Parameters

`provider` IFormatProvider?

Returns

string

ToString(string?, IFormatProvider?)

```
public string ToString(string? format, IFormatProvider? provider)
```

Parameters

`format` string?

`provider` IFormatProvider?

Returns

string

Operators

operator +(IQ, IQ)

```
public static IQ operator +(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

operator +(IQ, float)

```
public static IQ operator +(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

operator +(float, IQ)

```
public static IQ operator +(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

operator /(IQ, IQ)

```
public static IQ operator /(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

operator /(IQ, float)

```
public static IQ operator /(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

operator /(float, IQ)

```
public static IQ operator /(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

operator ==(IQ, IQ)

```
public static bool operator ==(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

bool

explicit operator IQ(decimal)

```
public static explicit operator IQ(decimal value)
```

Parameters

`value` decimal

Returns

IQ

implicit operator IQ(byte)

```
public static implicit operator IQ(byte value)
```

Parameters

`value` byte

Returns

IQ

implicit operator IQ(char)

```
public static implicit operator IQ(char value)
```

Parameters

`value` char

Returns

IQ

implicit operator IQ(float)

```
public static implicit operator IQ(float value)
```

Parameters

`value` float

Returns

IQ

implicit operator IQ(Half)

```
public static implicit operator IQ(Half value)
```

Parameters

`value` Half

Returns

IQ

implicit operator IQ(short)

```
public static implicit operator IQ(short value)
```

Parameters

`value` short

Returns

IQ

implicit operator IQ(int)

```
public static implicit operator IQ(int value)
```

Parameters

`value` int

Returns

IQ

implicit operator IQ(long)

```
public static implicit operator IQ(long value)
```

Parameters

`value` long

Returns

IQ

implicit operator IQ(nint)

```
public static implicit operator IQ(nint value)
```

Parameters

`value` nint

Returns

IQ

implicit operator IQ(sbyte)

```
public static implicit operator IQ(sbyte value)
```

Parameters

`value` sbyte

Returns

IQ

implicit operator IQ(ushort)

```
public static implicit operator IQ(ushort value)
```

Parameters

`value` ushort

Returns

IQ

implicit operator IQ(uint)

```
public static implicit operator IQ(uint value)
```

Parameters

`value` uint

Returns

IQ

implicit operator IQ(ulong)

```
public static implicit operator IQ(ulong value)
```

Parameters

`value` ulong

Returns

IQ

implicit operator IQ(nuint)

```
public static implicit operator IQ(nuint value)
```

Parameters

`value` nuint

Returns

IQ

implicit operator IQ(Complex)

```
public static implicit operator IQ(Complex value)
```

Parameters

`value` Complex

Returns

IQ

implicit operator Complex(IQ)

```
public static implicit operator Complex(IQ value)
```

Parameters

`value` IQ

Returns

Complex

operator !=(IQ, IQ)

```
public static bool operator !=(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

bool

operator *(IQ, IQ)

```
public static IQ operator *(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

operator *(IQ, float)

```
public static IQ operator *(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

operator *(float, IQ)

```
public static IQ operator *(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

operator -(IQ, IQ)

```
public static IQ operator -(IQ left, IQ right)
```

Parameters

`left` IQ

`right` IQ

Returns

IQ

operator -(IQ, float)

```
public static IQ operator -(IQ left, float right)
```

Parameters

`left` IQ

`right` float

Returns

IQ

operator -(float, IQ)

```
public static IQ operator -(float left, IQ right)
```

Parameters

`left` float

`right` IQ

Returns

IQ

operator -(IQ)

```
public static IQ operator -(IQ value)
```

Parameters

`value` IQ

Returns

IQ

Struct InterleavedSample

Namespace: HackRFDotnet.Api.Streams

Assembly: HackRFDotnet.dll

HackRFDotnet.Api.Streams.InterleavedSample comes directly from the HackRF device in transfer chunks. Memory alignment allows us to ready and copy them into objects very quickly.

```
public struct InterleavedSample
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

I

```
public sbyte I
```

Field Value

sbyte

Q

```
public sbyte Q
```

Field Value

sbyte

Methods

Clone()

```
public InterleavedSample Clone()
```

Returns

InterleavedSample

Namespace HackRFDotnet.Api.Streams.Device

Classes

IQDeviceStream

IQ Stream from the Rf Device. This stream is the root of all data processed with the library. This stream must remain immutable from all other HackRFDotnet.Api.Streams.SignalStreams.SignalStream`1.

IQFileStream

Class IQDeviceStream

Namespace: HackRFDotnet.Api.Streams.Device

Assembly: HackRFDotnet.dll

IQ Stream from the Rf Device. This stream is the root of all data processed with the library. This stream must remain immutable from all other HackRFDotnet.Api.Streams.SignalStreams.SignalStream`1.

```
public class IQDeviceStream : IIQStream, IDisposable
```

Inheritance

object ← IQDeviceStream

Implements

IIQStream , IDisposable

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

IQDeviceStream(DigitalRadioDevice)

```
public IQDeviceStream(DigitalRadioDevice rfDevice)
```

Parameters

`rfDevice` DigitalRadioDevice

Fields

RfDevice

```
public readonly DigitalRadioDevice RfDevice
```

Field Value

DigitalRadioDevice

Properties

BufferLength

The number of bytes available to read in the buffer.

```
public int BufferLength { get; }
```

Property Value

int

SampleRate

The capture sample rate from the device.

```
public SampleRate SampleRate { get; }
```

Property Value

SampleRate

Methods

Close()

Close the stream on the device.

```
public void Close()
```

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

OpenRx(SampleRate?)

Open an Rx stream to read IQ samples.

```
public void OpenRx(SampleRate? sampleRate = null)
```

Parameters

`sampleRate` SampleRate?

ReadBuffer(Span<IQ>)

Fill a span with data from the ring buffer.

```
public int ReadBuffer(Span<IQ> iqBuffer)
```

Parameters

`iqBuffer` Span<IQ>

Returns

int

SetSampleRate(SampleRate)

Set the capture sample rate of the stream and device.

```
public void SetSampleRate(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

TxBuffer(Span<IQ>)

Open a Tx stream to write IQ samples.

```
public int TxBuffer(Span<IQ> iqFrame)
```

Parameters

`iqFrame` Span<IQ>

Returns

int

Class IQFileStream

Namespace: HackRFDotnet.Api.Streams.Device

Assembly: HackRFDotnet.dll

```
public class IQFileStream : IIQStream, IDisposable
```

Inheritance

object ← IQFileStream

Implements

IIQStream , IDisposable

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

IQFileStream(string)

```
public IQFileStream(string fileName)
```

Parameters

`fileName` string

Properties

BufferLength

The number of bytes available to read in the buffer.

```
public int BufferLength { get; }
```

Property Value

int

Frequency

```
public Frequency Frequency { get; set; }
```

Property Value

Frequency

SampleRate

The capture sample rate from the device.

```
public SampleRate SampleRate { get; set; }
```

Property Value

SampleRate

Methods

Close()

Close the stream on the device.

```
public void Close()
```

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

OpenRx(SampleRate?)

Open an Rx stream to read IQ samples.

```
public void OpenRx(SampleRate? sampleRate = null)
```

Parameters

`sampleRate` SampleRate?

ReadBuffer(Span<IQ>)

Fill a span with data from the ring buffer.

```
public int ReadBuffer(Span<IQ> iqFrame)
```

Parameters

`iqFrame` Span<IQ>

Returns

int

SetSampleRate(SampleRate)

Set the capture sample rate of the stream and device.

```
public void SetSampleRate(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

TxBuffer(Span<IQ>)

Open a Tx stream to write IQ samples.

```
public int TxBuffer(Span<IQ> iqFrame)
```

Parameters

`iqFrame` Span<IQ>

Returns

int

WriteBuffer(Span<byte>)

```
public int WriteBuffer(Span<byte> iqFrame)
```

Parameters

`iqFrame` Span<byte>

Returns

int

Namespace HackRFDotnet.Api.Streams.Exceptions

Classes

NullCallbackException

Class NullCallbackException

Namespace: HackRFDotnet.Api.Streams.Exceptions

Assembly: HackRFDotnet.dll

```
public class NullCallbackException : Exception, ISerializable
```

Inheritance

object ← Exception ← NullCallbackException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

NullCallbackException(string?)

```
public NullCallbackException(string? message)
```

Parameters

`message` string?

Namespace HackRFDotnet.Api.Streams.Interfaces

Interfaces

IIQStream

HackRFDotnet.Api.Streams.Interfaces.IIQStream is an immutable stream that buffers the data directly from the IQ device.

Interface IIQStream

Namespace: HackRFDotnet.Api.Streams.Interfaces

Assembly: HackRFDotnet.dll

HackRFDotnet.Api.Streams.Interfaces.IIQStream is an immutable stream that buffers the data directly from the IQ device.

```
public interface IIQStream : IDisposable
```

Implements

IDisposable

Properties

BufferLength

The number of bytes available to read in the buffer.

```
int BufferLength { get; }
```

Property Value

int

SampleRate

The capture sample rate from the device.

```
SampleRate SampleRate { get; }
```

Property Value

SampleRate

Methods

Close()

Close the stream on the device.

```
void Close()
```

OpenRx(SampleRate?)

Open an Rx stream to read IQ samples.

```
void OpenRx(SampleRate? sampleRate = null)
```

Parameters

`sampleRate` SampleRate?

ReadBuffer(Span<IQ>)

Fill a span with data from the ring buffer.

```
int ReadBuffer(Span<IQ> iqBuffer)
```

Parameters

`iqBuffer` Span<IQ>

Returns

int

SetSampleRate(SampleRate)

Set the capture sample rate of the stream and device.

```
void SetSampleRate(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

TxBuffer(Span<IQ>)

Open a Tx stream to write IQ samples.

```
int TxBuffer(Span<IQ> iqFrame)
```

Parameters

`iqFrame` Span<IQ>

Returns

int

Namespace HackRFDotnet.Api.Streams.SignalProcessing

Namespaces

HackRFDotnet.Api.Streams.SignalProcessing.Effects

HackRFDotnet.Api.Streams.SignalProcessing.ErrorRate

HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters

HackRFDotnet.Api.Streams.SignalProcessing.Interfaces

Classes

SignalProcessingPipeline<TInput>

Effects chain processor.

Class SignalProcessingPipeline<TInput>

Namespace: HackRFDotnet.Api.Streams.SignalProcessing

Assembly: HackRFDotnet.dll

Effects chain processor.

```
public class SignalProcessingPipeline<TInput> where TInput : struct
```

Type Parameters

`TInput`

Inheritance

object ← SignalProcessingPipeline<TInput>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

ApplyPipeline(Span<TInput>)

```
public int ApplyPipeline(Span<TInput> signalTheta)
```

Parameters

`signalTheta` Span<TInput>

Returns

int

WithRootEffect<TOutput>(SignalEffect<TInput, TOutput>)

```
public SignalEffect<TInput, TOutput> WithRootEffect<TOutput>(SignalEffect<TInput, TOutput>
rootEffect) where TOutput : struct
```

Parameters

`rootEffect` SignalEffect<TInput, TOutput>

Returns

SignalEffect<TInput, TOutput>

Type Parameters

`TOutput`

Namespace HackRFDotnet.Api.Streams.SignalProcessing.Effects

Classes

FftEffect

Fast Fourier Transform Effect. Can be used for forward and inverse transforms. Must be given a chunk with a size that is a multiple of 2 [Length % 2 == 0] Must be configured with a chunk size for caching a convert buffer.

FrequencyCenteringEffect

Shift the frequency by a HackRFDotnet.Api.Frequency offset. This only works for IQ samples, we can shift frequency without losing information.

IQDownSampleEffect

HackRFDotnet.Api.Streams.SignalProcessing.Effects.IQDownSampleEffect removes extraneous information from your signal using your desired bandwidth. Example: an FM radio's band is around 200 kHz; the minimum sample rate required to represent this is 400 kS/s (400,000 samples per second). It is recommended that you reduce the sample rate of your audio signal this way before further signal processing to save CPU.

LowPassFilterEffect

Low Pass Filter Effect to remove unwanted signals from the input signal. Configured with a bandwidth to limit via the filter.

SignalEffect<TInput, TOutput>

Signal effect base class.

SquelchEffect

Squelch Effect to remove noise when there is no detected signal present.

Class FftEffect

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

Fast Fourier Transform Effect. Can be used for forward and inverse transforms. Must be given a chunk with a size that is a multiple of 2 [Length % 2 == 0] Must be configured with a chunk size for caching a convert buffer.

```
public class FftEffect : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>, IDisposable
```

Inheritance

object ← SignalEffect<IQ, IQ> ← FftEffect

Implements

ISignalEffectInput<IQ> , IDisposable

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

FftEffect(bool, int)

```
public FftEffect(bool forward, int chunkSize)
```

Parameters

`forward` bool

`chunkSize` int

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class FrequencyCenteringEffect

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

Shift the frequency by a HackRFDotnet.Api.Frequency offset. This only works for IQ samples, we can shift frequency without losing information.

```
public class FrequencyCenteringEffect : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, IQ> ← FrequencyCenteringEffect

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

FrequencyCenteringEffect(Frequency, SampleRate)

```
public FrequencyCenteringEffect(Frequency frequencyOffset, SampleRate sampleRate)
```

Parameters

`frequencyOffset` Frequency

`sampleRate` SampleRate

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class IQDownSampleEffect

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

HackRFDotnet.Api.Streams.SignalProcessing.Effects.IQDownSampleEffect removes extraneous information from your signal using your desired bandwidth. Example: an FM radio's band is around 200 kHz; the minimum sample rate required to represent this is 400 kS/s (400,000 samples per second). It is recommended that you reduce the sample rate of your audio signal this way before further signal processing to save CPU.

```
public class IQDownSampleEffect : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>, IDisposable
```

Inheritance

object ← SignalEffect<IQ, IQ> ← IQDownSampleEffect

Implements

ISignalEffectInput<IQ> , IDisposable

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

IQDownSampleEffect(SampleRate, SampleRate, out SampleRate, out int)

Configure a signal down sampler. You should do this to reduce cpu time when processing your signal.

```
public IQDownSampleEffect(SampleRate sampleRate, SampleRate reducedSampleRate, out
SampleRate newSampleRate, out int producedChunkSize)
```

Parameters

`sampleRate` SampleRate

Sample rate of the incoming signal.

`reducedSampleRate` SampleRate

Desired reduced sample rate.

`newSampleRate` SampleRate

The closest possible sample rate achievable.

`producedChunkSize` int

The chunk size after down sampling.

IQDownSampleEffect(SampleRate, SampleRate, int, out SampleRate, out int)

Configure a signal down sampler. You should do this to reduce cpu time when processing your signal.

```
public IQDownSampleEffect(SampleRate sampleRate, SampleRate reducedSampleRate, int
processingSize, out SampleRate newSampleRate, out int producedChunkSize)
```

Parameters

`sampleRate` SampleRate

Sample rate of the incoming signal.

`reducedSampleRate` SampleRate

Desired reduced sample rate.

`processingSize` int

The input chunk size. Used to calculate the nearest achievable sample rate.

`newSampleRate` SampleRate

The closest possible sample rate achievable.

`producedChunkSize` int

The chunk size after down sampling.

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class LowPassFilterEffect

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

Low Pass Filter Effect to remove unwanted signals from the input signal. Configured with a bandwidth to limit via the filter.

```
public class LowPassFilterEffect : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, IQ> ← LowPassFilterEffect

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

LowPassFilterEffect(SampleRate, Bandwidth)

Apply a low pass filter on the signal. Expects Frequency Domain input.

```
public LowPassFilterEffect(SampleRate sampleRate, Bandwidth bandwith)
```

Parameters

`sampleRate` SampleRate

`bandwith` Bandwidth

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class SignalEffect<TInput, TOutput>

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

Signal effect base class.

```
public abstract class SignalEffect<TInput, TOutput> : ISignalEffectInput<TInput> where
TInput : struct where TOutput : struct
```

Type Parameters

`TInput`

`TOutput`

Inheritance

object ← SignalEffect<TInput, TOutput>

Implements

ISignalEffectInput<TInput>

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

AddChildEffect<TFormat>(SignalEffect<TOutput, TFormat>)

```
public SignalEffect<TOutput, TFormat> AddChildEffect<TFormat>(SignalEffect<TOutput, TFormat>
childEffect) where TFormat : struct
```

Parameters

`childEffect` SignalEffect<TOutput, TFormat>

Returns

SignalEffect<TOutput, TFormat>

Type Parameters

`TFormat`

TransformSignal(Span<TInput>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public virtual int TransformSignal(Span<TInput> signalTheta, int length)
```

Parameters

`signalTheta` Span<TInput>

`length` int

Returns

int

Class SquelchEffect

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Effects

Assembly: HackRFDotnet.dll

Squelch Effect to remove noise when there is no detected signal present.

```
public class SquelchEffect : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, IQ> ← SquelchEffect

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

SquelchEffect(SampleRate)

```
public SquelchEffect(SampleRate sampleRate)
```

Parameters

`sampleRate` SampleRate

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Namespace HackRFDotnet.Api.Streams.SignalProcessing.ErrorRate

Classes

BitErrorRate

Class BitErrorRate

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.ErrorRate

Assembly: HackRFDotnet.dll

```
public class BitErrorRate
```

Inheritance

object ← BitErrorRate

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Namespace HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters

Namespaces

HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Classes

OfdmChannelizer

This is discover the channel division inside an OFDM stream. It will then divide the samples into each channel via FFT. Each bin will be a channel with a complex number. Use this complex number at each bin to demodulate information from the complex plane.

Class OfdmChannelizer

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters

Assembly: HackRFDotnet.dll

This is discover the channel division inside an OFDM stream. It will then divide the samples into each channel via FFT. Each bin will be a channel with a complex number. Use this complex number at each bin to demodulate information from the complex plane.

```
public class OfdmChannelizer : SignalEffect<IQ, IQ>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, IQ> ← OfdmChannelizer

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, IQ>.TransformSignal(Span<IQ>, int) , SignalEffect<IQ, IQ>.AddChildEffect<TFormat>(SignalEffect<IQ, TFormat>) , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

OfdmChannelizer()

```
public OfdmChannelizer()
```

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

signalTheta Span<IQ>

length int

Returns

int

Namespace HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Classes

AmDemodulator

BpskDemodulator

FmDemodulator

QpskDemodulator

Class AmDemodulator

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Assembly: HackRFDotnet.dll

```
public class AmDemodulator : SignalEffect<IQ, float>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, float> ← AmDemodulator

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, float>.TransformSignal(Span<IQ>, int) ,
SignalEffect<IQ, float>.AddChildEffect<TFormat>(SignalEffect<float, TFormat>) ,
object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

AmDemodulator()

```
public AmDemodulator()
```

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class BpskDemodulator

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Assembly: HackRFDotnet.dll

```
public class BpskDemodulator : SignalEffect<IQ, byte>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, byte> ← BpskDemodulator

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, byte>.TransformSignal(Span<IQ>, int) ,
SignalEffect<IQ, byte>.AddChildEffect<TFormat>(SignalEffect<byte, TFormat>) ,
object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

BpskDemodulator()

```
public BpskDemodulator()
```

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class FmDemodulator

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Assembly: HackRFDotnet.dll

```
public class FmDemodulator : SignalEffect<IQ, float>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, float> ← FmDemodulator

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, float>.TransformSignal(Span<IQ>, int) ,
SignalEffect<IQ, float>.AddChildEffect<TFormat>(SignalEffect<float, TFormat>) ,
object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

FmDemodulator()

```
public FmDemodulator()
```

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Class QpskDemodulator

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.FormatConverters.Demodulators

Assembly: HackRFDotnet.dll

```
public class QpskDemodulator : SignalEffect<IQ, byte>, ISignalEffectInput<IQ>
```

Inheritance

object ← SignalEffect<IQ, byte> ← QpskDemodulator

Implements

ISignalEffectInput<IQ>

Inherited Members

SignalEffect<IQ, byte>.TransformSignal(Span<IQ>, int) ,
SignalEffect<IQ, byte>.AddChildEffect<TFormat>(SignalEffect<byte, TFormat>) ,
object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

QpskDemodulator()

```
public QpskDemodulator()
```

Methods

TransformSignal(Span<IQ>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
public override int TransformSignal(Span<IQ> signalTheta, int length)
```

Parameters

`signalTheta` Span<IQ>

`length` int

Returns

int

Namespace HackRFDotnet.Api.Streams.Signal Processing.Interfaces

Interfaces

ISignalEffectInput<TInput>

Interface ISignalEffectInput<TInput>

Namespace: HackRFDotnet.Api.Streams.SignalProcessing.Interfaces

Assembly: HackRFDotnet.dll

```
public interface ISignalEffectInput<TInput>
```

Type Parameters

`TInput`

Methods

TransformSignal(Span<TInput>, int)

Manipulate in-place, the signal provided and return a new length is samples were reduced.

```
int TransformSignal(Span<TInput> signalTheta, int length)
```

Parameters

`signalTheta` Span <TInput>

`length` int

Returns

int

Namespace HackRFDotnet.Api.Streams.SignalStreams

Namespaces

HackRFDotnet.Api.Streams.SignalStreams.Analogue

HackRFDotnet.Api.Streams.SignalStreams.Digital

Classes

SignalStream<TOutput>

A HackRFDotnet.Api.Streams.SignalStreams.SignalStream`1 allows you to process effects from a pipeline, and read the result like a stream reader. Stream must be created from a HackRFDotnet.Api.Streams.Interfaces.IIQStream.

Class SignalStream<TOutput>

Namespace: HackRFDotnet.Api.Streams.SignalStreams

Assembly: HackRFDotnet.dll

A HackRFDotnet.Api.Streams.SignalStreams.SignalStream`1 allows you to process effects from a pipeline, and read the result like a stream reader. Stream must be created from a HackRFDotnet.Api.Streams.Interfaces.IIQStream.

```
public class SignalStream<TOutput> : IDisposable where TOutput : struct
```

Type Parameters

`TOutput`

Inheritance

object ← SignalStream<TOutput>

Implements

IDisposable

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

SignalStream(IIQStream, SignalProcessingPipeline<IQ>?, bool)

```
public SignalStream(IIQStream iQStream, SignalProcessingPipeline<IQ>? processingPipeline =
null, bool keepOpen = true)
```

Parameters

`iQStream` IIQStream

`processingPipeline` SignalProcessingPipeline<IQ>?

`keepOpen` bool

Fields

_iQStream

```
protected readonly IIQStream _iQStream
```

Field Value

IIQStream

_keepOpen

```
protected readonly bool _keepOpen
```

Field Value

bool

_processingPipeline

```
protected SignalProcessingPipeline<IQ>? _processingPipeline
```

Field Value

SignalProcessingPipeline<IQ>?

Properties

Bandwidth

```
public Bandwidth Bandwidth { get; protected set; }
```

Property Value

Bandwidth

Center

```
public Frequency Center { get; protected set; }
```

Property Value

Frequency

SampleRate

```
public SampleRate SampleRate { get; }
```

Property Value

SampleRate

Methods

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

ReadSpan(Span<TOutput>)

```
public void ReadSpan(Span<TOutput> dataBuffer)
```

Parameters

`dataBuffer` Span<TOutput>

SetBand(Frequency, Bandwidth)

Set the band and bandwidth the filtering engine will use.

```
public void SetBand(Frequency center, Bandwidth bandwidth)
```

Parameters

`center` Frequency

`bandwidth` Bandwidth

Namespace HackRFDotnet.Api.Streams.SignalStreams.Analogue

Classes

AmSignalStream

Demodulate AM audio from the HackRFDotnet.Api.Streams.Interfaces.IIQStream.

FmSignalStream

Demodulate FM audio from HackRFDotnet.Api.Streams.Interfaces.IIQStream.

WaveSignalStream

NAudio NAudio.Wave.ISampleProvider base stream implementation, HackRFDotnet.Api.Streams.SignalStreams.Analogue.AmSignalStream and HackRFDotnet.Api.Streams.SignalStreams.Analogue.FmSignalStream stream inherit this.

Class AmSignalStream

Namespace: HackRFDotnet.Api.Streams.SignalStreams.Analogue

Assembly: HackRFDotnet.dll

Demodulate AM audio from the HackRFDotnet.Api.Streams.Interfaces.IIQStream.

```
public class AmSignalStream : WaveSignalStream, ISampleProvider, IDisposable
```

Inheritance

object ← SignalStream<float> ← WaveSignalStream ← AmSignalStream

Implements

ISampleProvider , IDisposable

Inherited Members

WaveSignalStream.WaveFormat , WaveSignalStream.Read(float[], int, int) , SignalStream<float>.Center , SignalStream<float>.Bandwidth , SignalStream<float>.SampleRate , SignalStream<float>._processingPipeline , SignalStream<float>._iQStream , SignalStream<float>._keepOpen , SignalStream<float>.ReadSpan(Span<float>) , SignalStream<float>.SetBand(Frequency, Bandwidth) , SignalStream<float>.Dispose() , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

AmSignalStream(IIQStream, Bandwidth, bool)

```
public AmSignalStream(IIQStream deviceStream, Bandwidth stationBandwidth, bool keepOpen
= true)
```

Parameters

`deviceStream` IIQStream

`stationBandwidth` Bandwidth

`keepOpen` bool

Class FmSignalStream

Namespace: HackRFDotnet.Api.Streams.SignalStreams.Analogue

Assembly: HackRFDotnet.dll

Demodulate FM audio from HackRFDotnet.Api.Streams.Interfaces.IIQStream.

```
public class FmSignalStream : WaveSignalStream, ISampleProvider, IDisposable
```

Inheritance

object ← SignalStream<float> ← WaveSignalStream ← FmSignalStream

Implements

ISampleProvider , IDisposable

Inherited Members

WaveSignalStream.WaveFormat , WaveSignalStream.Read(float[], int, int) , SignalStream<float>.Center , SignalStream<float>.Bandwidth , SignalStream<float>.SampleRate , SignalStream<float>._processingPipeline , SignalStream<float>._iQStream , SignalStream<float>._keepOpen , SignalStream<float>.ReadSpan(Span<float>) , SignalStream<float>.SetBand(Frequency, Bandwidth) , SignalStream<float>.Dispose() , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

FmSignalStream(IIQStream, Bandwidth, bool)

```
public FmSignalStream(IIQStream deviceStream, Bandwidth stationBandwidth, bool stereo
= true)
```

Parameters

`deviceStream` IIQStream

`stationBandwidth` Bandwidth

`stereo` bool

Class WaveSignalStream

Namespace: HackRFDotnet.Api.Streams.SignalStreams.Analogue

Assembly: HackRFDotnet.dll

NAudio NAudio.Wave.ISampleProvider base stream implementation, HackRFDotnet.Api.Streams.SignalStreams.Analogue.AmSignalStream and HackRFDotnet.Api.Streams.SignalStreams.Analogue.FmSignalStream stream inherit this.

```
public class WaveSignalStream : SignalStream<float>, ISampleProvider, IDisposable
```

Inheritance

object ← SignalStream<float> ← WaveSignalStream

Derived

AmSignalStream , FmSignalStream

Implements

ISampleProvider , IDisposable

Inherited Members

SignalStream<float>.Center , SignalStream<float>.Bandwidth , SignalStream<float>.SampleRate , SignalStream<float>._processingPipeline , SignalStream<float>._iQStream , SignalStream<float>._keepOpen , SignalStream<float>.ReadSpan(Span<float>) , SignalStream<float>.SetBand(Frequency, Bandwidth) , SignalStream<float>.Dispose() , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

WaveSignalStream(IIQStream, SignalProcessingPipeline<IQ>, SampleRate, bool, bool)

```
public WaveSignalStream(IIQStream deviceStream, SignalProcessingPipeline<IQ>
```

```
processingPipeline, SampleRate sampleRate, bool stero = true, bool keepOpen = true)
```

Parameters

`deviceStream` IIQStream

`processingPipeline` SignalProcessingPipeline<IQ>

`sampleRate` SampleRate

`stero` bool

`keepOpen` bool

Properties

WaveFormat

Gets the WaveFormat of this Sample Provider.

```
public WaveFormat? WaveFormat { get; protected set; }
```

Property Value

WaveFormat?

Methods

Read(float[], int, int)

Fill the specified buffer with 32 bit floating point samples

```
public virtual int Read(float[] buffer, int offset, int count)
```

Parameters

`buffer` float[]

The buffer to fill with samples.

`offset` int

Offset into buffer

`count` int

The number of samples to read

Returns

int

the number of samples written to the buffer.

Namespace HackRFDotnet.Api.Streams.SignalStreams.Digital

Classes

OfdmSignalStream

QpskSignalStream

Class OfdmSignalStream

Namespace: HackRFDotnet.Api.Streams.SignalStreams.Digital

Assembly: HackRFDotnet.dll

```
public class OfdmSignalStream : SignalStream<byte>, IDisposable
```

Inheritance

object ← SignalStream<byte> ← OfdmSignalStream

Implements

IDisposable

Inherited Members

SignalStream<byte>.Center , SignalStream<byte>.Bandwidth , SignalStream<byte>.SampleRate , SignalStream<byte>._processingPipeline , SignalStream<byte>._iQStream , SignalStream<byte>._keepOpen , SignalStream<byte>.ReadSpan(Span<byte>) , SignalStream<byte>.SetBand(Frequency, Bandwidth) , SignalStream<byte>.Dispose() , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

OfdmSignalStream(IIQStream, Bandwidth)

```
public OfdmSignalStream(IIQStream iQStream, Bandwidth signalBandwidth)
```

Parameters

`iQStream` IIQStream

`signalBandwidth` Bandwidth

Methods

Read(Span<byte>, int)

```
public int Read(Span<byte> buffer, int count)
```

Parameters

`buffer` Span<byte>

`count` int

Returns

int

Class QpskSignalStream

Namespace: HackRFDotnet.Api.Streams.SignalStreams.Digital

Assembly: HackRFDotnet.dll

```
public class QpskSignalStream : SignalStream<byte>, IDisposable
```

Inheritance

object ← SignalStream<byte> ← QpskSignalStream

Implements

IDisposable

Inherited Members

SignalStream<byte>.Center , SignalStream<byte>.Bandwidth , SignalStream<byte>.SampleRate , SignalStream<byte>._processingPipeline , SignalStream<byte>._iQStream , SignalStream<byte>._keepOpen , SignalStream<byte>.ReadSpan(Span<byte>) , SignalStream<byte>.SetBand(Frequency, Bandwidth) , SignalStream<byte>.Dispose() , object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Constructors

QpskSignalStream(IIQStream, Bandwidth)

```
public QpskSignalStream(IIQStream iQStream, Bandwidth signalBandwidth)
```

Parameters

`iQStream` IIQStream

`signalBandwidth` Bandwidth

Methods

Read(Span<byte>, int)

```
public int Read(Span<byte> buffer, int count)
```

Parameters

`buffer` Span<byte>

`count` int

Returns

int

Namespace HackRFDotnet.Api.Utilities

Classes

BinaryUtilities

SignalUtilities

Class BinaryUtilities

Namespace: HackRFDotnet.Api.Utilities

Assembly: HackRFDotnet.dll

```
public static class BinaryUtilities
```

Inheritance

object ← BinaryUtilities

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

NextPowerOfTwo(int)

```
public static int NextPowerOfTwo(int n)
```

Parameters

n int

Returns

int

Class SignalUtilities

Namespace: HackRFDotnet.Api.Utilities

Assembly: HackRFDotnet.dll

```
public class SignalUtilities
```

Inheritance

object ← SignalUtilities

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

ApplyFrequencyOffset(Span<IQ>, Frequency, SampleRate)

```
public static void ApplyFrequencyOffset(Span<IQ> iqFrame, Frequency freqOffset,
SampleRate sampleRate)
```

Parameters

`iqFrame` Span<IQ>

`freqOffset` Frequency

`sampleRate` SampleRate

CalculateRmsDb(ReadOnlySpan<IQ>)

```
public static float CalculateRmsDb(ReadOnlySpan<IQ> iqFrame)
```

Parameters

`iqFrame` ReadOnlySpan<IQ>

Returns

float

CalculateSignalDb(ReadOnlySpan<IQ>)

```
public static float CalculateSignalDb(ReadOnlySpan<IQ> iqFrame)
```

Parameters

`iqFrame` ReadOnlySpan<IQ>

Returns

float

FrequencyResolution(int, SampleRate, bool)

```
public static long FrequencyResolution(int length, SampleRate sampleRate, bool positiveOnly
= true)
```

Parameters

`length` int

`sampleRate` SampleRate

`positiveOnly` bool

Returns

long

IQCorrection(Span<IQ>)

```
public static void IQCorrection(Span<IQ> iqFrame)
```

Parameters

iqFrame Span<IQ>

NormalizeRms(Span<float>, float)

```
public static void NormalizeRms(Span<float> buffer, float targetRms = 0.04)
```

Parameters

buffer Span<float>

targetRms float

Namespace HackRFDotnet.NativeApi

Namespaces

HackRFDotnet.NativeApi.Enums

HackRFDotnet.NativeApi.Lib

HackRFDotnet.NativeApi.Structs

Classes

NativeConstants

Class NativeConstants

Namespace: HackRFDotnet.NativeApi

Assembly: HackRFDotnet.dll

```
public class NativeConstants
```

Inheritance

object ← NativeConstants

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

BOARD_ID_HACKRF_ONE

These deprecated board ID names are provided for API compatibility.

```
public const HackrfBoardId BOARD_ID_HACKRF_ONE = BOARD_ID_HACKRF1_OG
```

Field Value

HackrfBoardId

BOARD_ID_INVALID

These deprecated board ID names are provided for API compatibility.

```
public const HackrfBoardId BOARD_ID_INVALID = BOARD_ID_UNDETECTED
```

Field Value

HackrfBoardId

BYTES_PER_BLOCK

Number of bytes per tuning for sweeping.

```
public const uint BYTES_PER_BLOCK = 16384
```

Field Value

uint

HACKRF_BOARD_REV_GSG

Made by GSG bit in @ref hackrf_board_rev enum and in platform ID.

```
public const uint HACKRF_BOARD_REV_GSG = 128
```

Field Value

uint

HACKRF_OPERACAKE_ADDRESS_INVALID

Invalid Opera Cake add-on board address, placeholder in HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.GetOperacakeBoards(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte*).

```
public const uint HACKRF_OPERACAKE_ADDRESS_INVALID = 255
```

Field Value

uint

HACKRF_OPERACAKE_MAX_BOARDS

Maximum number of connected Opera Cake add-on boards.

```
public const uint HACKRF_OPERACAKE_MAX_BOARDS = 8
```

Field Value

uint

HACKRF_OPERACAKE_MAX_DWELL_TIMES

Maximum number of specifiable dwell times for Opera Cake add-on boards.

```
public const uint HACKRF_OPERACAKE_MAX_DWELL_TIMES = 16
```

Field Value

uint

HACKRF_OPERACAKE_MAX_FREQ_RANGES

Maximum number of specifiable frequency ranges for Opera Cake add-on boards.

```
public const uint HACKRF_OPERACAKE_MAX_FREQ_RANGES = 8
```

Field Value

uint

HACKRF_PLATFORM_HACKRF1_OG

HACKRF ONE (pre r9) platform bit in result of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadSupportedPlatform(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32*).

```
public const uint HACKRF_PLATFORM_HACKRF1_OG = 2
```

Field Value

uint

HACKRF_PLATFORM_HACKRF1_R9

HACKRF ONE (r9 or later) platform bit in result of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadSupportedPlatform(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32*).

```
public const uint HACKRF_PLATFORM_HACKRF1_R9 = 8
```

Field Value

uint

HACKRF_PLATFORM_JAWBREAKER

JAWBREAKER platform bit in result of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadSupportedPlatform(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32*).

```
public const uint HACKRF_PLATFORM_JAWBREAKER = 1
```

Field Value

uint

HACKRF_PLATFORM_RAD1O

RAD1O platform bit in result of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadSupportedPlatform(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32*).

```
public const uint HACKRF_PLATFORM_RAD1O = 4
```

Field Value

uint

MAX_SWEEP_RANGES

Maximum number of sweep ranges to be specified for HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.InitSweep(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt16*,System.Int32,System.UInt32,System.UInt32,System.UInt32,HackRFDotnet.NativeApi.Enums.SweepStyle).

```
public const uint MAX_SWEEP_RANGES = 10
```

Field Value

uint

SAMPLES_PER_BLOCK

Number of samples per tuning when sweeping.

```
public const uint SAMPLES_PER_BLOCK = 8192
```

Field Value

uint

Namespace HackRFDotnet.NativeApi.Enums

Namespaces

HackRFDotnet.NativeApi.Enums.Peripherals

HackRFDotnet.NativeApi.Enums.System

Enums

RfPathFilter

RF filter path setting enum.

Used only when performing explicit tuning using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.SetFrequency(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt64,System.UInt64,HackRFDotnet.NativeApi.Enums.RfPathFilter), or can be converted into a human-readable string using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.FilterPathName(HackRFDotnet.NativeApi.Enums.RfPathFilter).

This can select the image rejection filter (U3, U8 or none) to use - using switches U5, U6, U9 and U11. When no filter is selected, the mixer itself is bypassed.

SweepStyle

Enum RfPathFilter

Namespace: HackRFDotnet.NativeApi.Enums

Assembly: HackRFDotnet.dll

RF filter path setting enum.

Used only when performing explicit tuning using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.SetFrequency(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt64,System.UInt64,HackRFDotnet.NativeApi.Enums.RfPathFilter), or can be converted into a human-readable string using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.FilterPathName(HackRFDotnet.NativeApi.Enums.RfPathFilter).

This can select the image rejection filter (U3, U8 or none) to use - using switches U5, U6, U9 and U11. When no filter is selected, the mixer itself is bypassed.

```
public enum RfPathFilter
```

Fields

`RF_PATH_FILTER_BYPASS = 0`

No filter is selected, **the mixer is bypassed**, \f$f_{center} = f_{IF}\f$

`RF_PATH_FILTER_HIGH_PASS = 2`

HPF is selected, \f$f_{center} = f_{IF} + f_{LO}\f$

`RF_PATH_FILTER_LOW_PASS = 1`

LPF is selected, \f$f_{center} = f_{IF} - f_{LO}\f$

Enum SweepStyle

Namespace: HackRFDotnet.NativeApi.Enums

Assembly: HackRFDotnet.dll

```
public enum SweepStyle
```

Fields

`INTERLEAVED = 1`

Each step is divided into two interleaved sub-steps, allowing the host to select the best portions of the FFT of each sub-step and discard the rest.

`LINEAR = 0`

step_width is added to the current frequency at each step.

Namespace HackRFDotnet.NativeApi.Enums.Peripherals

Enums

LedState

OperacakePorts

OperacakeSwitchingMode

Opera Cake port switching mode. Set via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.SetOperacakeMode(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte,HackRFDotnet.NativeApi.Enums.Peripherals.OperacakeSwitchingMode) and queried via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.GetOperacakeMode(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte,HackRFDotnet.NativeApi.Enums.Peripherals.OperacakeSwitchingMode*).

Enum LedState

Namespace: HackRFDotnet.NativeApi.Enums.Peripherals

Assembly: HackRFDotnet.dll

```
public enum LedState : byte
```

Fields

RxLight = 2

TxLight = 4

UsbLight = 0

Enum OperacakePorts

Namespace: HackRFDotnet.NativeApi.Enums.Peripherals

Assembly: HackRFDotnet.dll

```
public enum OperacakePorts
```

Fields

OPERACAKE_PA1 = 0

OPERACAKE_PA2 = 1

OPERACAKE_PA3 = 2

OPERACAKE_PA4 = 3

OPERACAKE_PB1 = 4

OPERACAKE_PB2 = 5

OPERACAKE_PB3 = 6

OPERACAKE_PB4 = 7

Enum OperacakeSwitchingMode

Namespace: HackRFDotnet.NativeApi.Enums.Peripherals

Assembly: HackRFDotnet.dll

Opera Cake port switching mode. Set via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.SetOperacakeMode(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte,HackRFDotnet.NativeApi.Enums.Peripherals.OperacakeSwitchingMode) and queried via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.GetOperacakeMode(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte,HackRFDotnet.NativeApi.Enums.Peripherals.OperacakeSwitchingMode*).

```
public enum OperacakeSwitchingMode
```

Fields

`OPERACAKE_MODE_FREQUENCY = 1`

Port connections are switched automatically when the frequency is changed. Frequency ranges can be set using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.SetOperacakeFrequencyRanges(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Structs.Devices.HackRFOperacakeFreqRange*,System.Byte). In this mode, B0 mirrors A0.

`OPERACAKE_MODE_MANUAL = 0`

Port connections are set manually using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.SetOperacakePorts(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte,System.Byte,System.Byte). Both ports can be specified, but not on the same side.

`OPERACAKE_MODE_TIME = 2`

Port connections are switched automatically over time. dwell times can be set with HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Operacake.SetOperacakeDwellTimes(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Structs.Devices.HackRFOperacakeDwellTime*,System.Byte). In this mode, B0 mirrors A0.

Namespace HackRFDotnet.NativeApi.Enums.System

Enums

HackrfBoardId

HACKRF board id enum.

Returned by HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadBoardId(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte*) and can be converted to a human-readable string using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.BoardIdName(HackRFDotnet.NativeApi.Enums.System.HackrfBoardId).

HackrfBoardRev

HackrfError

Error enum, returned by many libhackrf functions.

HackrfUsbBoardId

USB board ID (product ID) enum

Contains USB-IF product id (field `idProduct` in `libusb_device_descriptor`). Can be used to identify general type of hardware. Only used in HackRFDotnet.NativeApi.Structs.Devices.HackRFDeviceList.usb_board_ids field of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Devices.QueryDeviceList, and can be converted into human-readable string via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.UsbBoardIdName(HackRFDotnet.NativeApi.Enums.System.HackrfUsbBoardId).

Enum HackrfBoardId

Namespace: HackRFDotnet.NativeApi.Enums.System

Assembly: HackRFDotnet.dll

HACKRF board id enum.

Returned by HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadBoardId(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.Byte*) and can be converted to a human-readable string using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.BoardIdName(HackRFDotnet.NativeApi.Enums.System.HackrfBoardId).

```
public enum HackrfBoardId
```

Fields

`BOARD_ID_HACKRF1_OG = 2`

HackRF One (prior to rev 9, same limits: 1-6000MHz, 20MSPS, bias-tee).

`BOARD_ID_HACKRF1_R9 = 4`

`BOARD_ID_JAWBREAKER = 1`

Jawbreaker (beta platform, 10-6000MHz, no bias-tee).

`BOARD_ID_JELLYBEAN = 0`

Jellybean (pre-production revision, not supported).

`BOARD_ID_RAD1O = 3`

`BOARD_ID_UNDETECTED = 255`

Unknown board (detection not yet attempted, should be default value).

`BOARD_ID_UNRECOGNIZED = 254`

Unknown board (failed detection).

Enum HackrfBoardRev

Namespace: HackRFDotnet.NativeApi.Enums.System

Assembly: HackRFDotnet.dll

```
public enum HackrfBoardRev : byte
```

Fields

`BOARD_REV_GSG_HACKRF1_R10 = 133`

Board revision 10, made by GSG

`BOARD_REV_GSG_HACKRF1_R6 = 129`

Board revision 6, made by GSG

`BOARD_REV_GSG_HACKRF1_R7 = 130`

Board revision 7, made by GSG

`BOARD_REV_GSG_HACKRF1_R8 = 131`

Board revision 8, made by GSG

`BOARD_REV_GSG_HACKRF1_R9 = 132`

Board revision 9, made by GSG

`BOARD_REV_HACKRF1_OLD = 0`

Older than rev6

`BOARD_REV_HACKRF1_R10 = 5`

Board revision 10, generic

`BOARD_REV_HACKRF1_R6 = 1`

Board revision 6, generic

`BOARD_REV_HACKRF1_R7 = 2`

Board revision 7, generic

```
BOARD_REV_HACKRF1_R8 = 3
```

Board revision 8, generic

```
BOARD_REV_HACKRF1_R9 = 4
```

Board revision 9, generic

```
BOARD_REV_UNDETECTED = 255
```

Unknown board revision (detection not yet attempted)

```
BOARD_REV_UNRECOGNIZED = 254
```

Unknown board revision (detection failed)

Enum HackrfError

Namespace: HackRFDotnet.NativeApi.Enums.System

Assembly: HackRFDotnet.dll

Error enum, returned by many libhackrf functions.

```
public enum HackrfError
```

Fields

`HACKRF_ERROR_BUSY = -6`

Resource is busy, possibly the device is already opened.

`HACKRF_ERROR_INVALID_PARAM = -2`

The function was called with invalid parameters.

`HACKRF_ERROR_LIBUSB = -1000`

LibUSB error, use HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Debug.GetErrorName(HackRFDotnet.NativeApi.Enums.System.HackrfError) to get a human-readable error string (using `libusb_strerror`).

`HACKRF_ERROR_NOT_FOUND = -5`

USB device not found, returned at opening.

`HACKRF_ERROR_NOT_LAST_DEVICE = -2000`

Can not exit library as one or more HackRFs still in use.

`HACKRF_ERROR_NO_MEM = -11`

Memory allocation (on host side) failed.

`HACKRF_ERROR_OTHER = -9999`

Unspecified error.

`HACKRF_ERROR_STREAMING_EXIT_CALLED = -1004`

Streaming thread exited (normally).

`HACKRF_ERROR_STREAMING_STOPPED = -1003`

Streaming thread stopped due to an error.

`HACKRF_ERROR_STREAMING_THREAD_ERR = -1002`

Streaming thread could not start due to an error.

`HACKRF_ERROR_THREAD = -1001`

Error setting up transfer thread (pthread-related error).

`HACKRF_ERROR_USB_API_VERSION = -1005`

The installed firmware does not support this function.

`HACKRF_SUCCESS = 0`

No error happened.

`HACKRF_TRUE = 1`

TRUE value, returned by some functions that return boolean value. Only a few functions can return this variant, and this fact should be explicitly noted at those functions.

Enum HackrfUsbBoardId

Namespace: HackRFDotnet.NativeApi.Enums.System

Assembly: HackRFDotnet.dll

USB board ID (product ID) enum

Contains USB-IF product id (field `idProduct` in `libusb_device_descriptor`). Can be used to identify general type of hardware. Only used in HackRFDotnet.NativeApi.Structs.Devices.HackRFDeviceList.usb_board_ids field of HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Devices.QueryDeviceList, and can be converted into human-readable string via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.UsbBoardIdName(HackRFDotnet.NativeApi.Enums.System.HackrfUsbBoardId).

```
public enum HackrfUsbBoardId
```

Fields

`USB_BOARD_ID_HACKRF_ONE = 24713`

HackRF One USB product id

`USB_BOARD_ID_INVALID = 65535`

Invalid / unknown USB product id

`USB_BOARD_ID_JAWBREAKER = 24651`

Jawbreaker (beta platform) USB product id

`USB_BOARD_ID_RAD1O = 52245`

RAD1O (custom version) USB product id

Namespace HackRFDotnet.NativeApi.Lib

Classes

HackRfNativeLib.Debug

HackRfNativeLib.DeviceStreaming

HackRfNativeLib.Devices

HackRfNativeLib.Firmware

HackRfNativeLib

HackRfNativeLib.Operacake

Class HackRfNativeLib

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib
```

Inheritance

object ← HackRfNativeLib

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

Exit()

Exit libhackrf. Should be called before application exit.

```
public static extern int Exit()
```

Returns

int

Init()

Initialize libhackrf. Should be called before any other function.

```
public static extern int Init()
```

Returns

int

LibraryRelease()

Get library release string.

```
public static extern sbyte* LibraryRelease()
```

Returns

sbyte*

LibraryVersion()

Get library version string.

```
public static extern sbyte* LibraryVersion()
```

Returns

sbyte*

Class HackRfNativeLib.Debug

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib.Debug
```

Inheritance

object ← HackRfNativeLib.Debug

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

GetErrorName(HackrfError)

Convert HackRFDotnet.NativeApi.Enums.System.HackrfError into human-readable string.

```
public static extern sbyte* GetErrorName(HackrfError errcode)
```

Parameters

`errcode` HackrfError

Enum to convert.

Returns

sbyte*

Human-readable name of error.

Class HackRfNativeLib.DeviceStreaming

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib.DeviceStreaming
```

Inheritance

object ← HackRfNativeLib.DeviceStreaming

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

ComputeBasebandFilterBandWidth(uint)

Compute nearest valid baseband filter bandwidth to specified value.

The result can be used via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.SetBaseband FilterBandwidth(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32).

```
public static extern uint ComputeBasebandFilterBandWidth(uint bandwidth_hz)
```

Parameters

`bandwidth_hz` uint

Desired filter bandwidth in Hz.

Returns

uint

Nearest valid filter bandwidth in Hz.

ComputeBasebandFilterBandWidth_round_down_lt(uint)

Compute nearest valid baseband filter bandwidth lower than a specified value.

The result can be used via HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.SetBaseband FilterBandwidth(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt32).

```
public static extern uint ComputeBasebandFilterBandWidth_round_down_lt(uint bandwidth_hz)
```

Parameters

`bandwidth_hz` uint

Desired filter bandwidth in Hz.

Returns

uint

The highest valid filter bandwidth lower than `bandwidth_hz` in Hz.

EnableAmp(HackRFDevice*, byte)

Enable / disable 14dB RF amplifier.

Enable / disable the ~11dB RF RX/TX amplifiers U13/U25 via controlling switches U9 and U14.

```
public static extern HackrfError EnableAmp(HackRFDevice* device, byte value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` byte

Enable (1) or disable (0) amplifier.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

EnableAntenna(HackRFDevice*, byte)

Enable / disable bias-tee (antenna port power).

Enable or disable the **3.3V (max 50mA)** bias-tee (antenna port power). Defaults to disabled.

NOTE: the firmware auto-disables this after returning to IDLE mode, so a perma-set is not possible, which means all software supporting HackRF devices must support enabling bias-tee, as setting it externally is not possible like it is with RTL-SDR for example.

```
public static extern HackrfError EnableAntenna(HackRFDevice* device, byte value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` byte

Enable (1) or disable (0) bias-tee.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

EnableTxFlush(HackRFDevice*, HackRFFlushCallback, void*)

Setup flush (end-of-transmission) callback.

This callback will be called when all the data was transmitted and all data transfers were completed. First parameter is supplied context, second parameter is success flag.

```
public static HackrfError EnableTxFlush(HackRFDevice* device, HackRFFlushCallback callback,
void* flush_ctx)
```

Parameters

`device` HackRFDevice*

Device to configure.

`callback` HackRFFlushCallback

callback to call when all transfers were completed.

`flush_ctx` void*

context (1st parameter of callback).

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

FilterPathName(RfPathFilter)

Convert HackRFDotnet.NativeApi.Enums.RfPathFilter into human-readable string.

```
public static extern sbyte* FilterPathName(RfPathFilter path)
```

Parameters

`path` RfPathFilter

Enum to convert.

Returns

sbyte*

Human-readable name of filter path.

GetTransferBufferSize(HackRFDevice*)

Get USB transfer buffer size.

```
public static extern nuint GetTransferBufferSize(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Unused.

Returns

nuint

Size in bytes.

GetTransferQueueDepth(HackRFDevice*)

Get the total number of USB transfer buffers.

```
public static extern uint GetTransferQueueDepth(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Unused.

Returns

uint

Number of buffers.

InitSweep(HackRFDevice*, ushort*, int, uint, uint, uint, SweepStyle)

Initialize sweep mode.

In this mode, in a single data transfer(single call to the RX transfer callback), multiple blocks of size `num_bytes` bytes are received with different center frequencies.At the beginning of each block, a 10-byte frequency header is present in `0x7F - 0x7F - uint64_t frequency(LSBFIRST, in Hz)` format, followed by the actual samples.

Requires USB API version 0x0102 or above!

```
public static extern HackrfError InitSweep(HackRFDevice* device, ushort* frequency_list, int
num_ranges, uint num_bytes, uint step_width, uint offset, SweepStyle style)
```

Parameters

`device` HackRFDevice*

Device to configure.

`frequency_list` ushort*

List of start-stop frequency pairs in MHz.

`num_ranges` int

Length of array `frequency_list` (in pairs, so total array length / 2!). Must be less than .

`num_bytes` uint

Number of bytes to capture per tuning, must be a multiple of HackRFDotnet.NativeApi.Native Constants.BYTES_PER_BLOCK.

`step_width` uint

Width of each tuning step in Hz.

`offset` uint

Frequency offset added to tuned frequencies.sample_rate / 2 is a good value.

`style` SweepStyle

Sweep style.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

IsStreaming(HackRFDevice*)

Query device streaming status.

```
public static extern HackrfError IsStreaming(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to query.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_TRUE If the device is streaming, else one of HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_STREAMING_THREAD_ERR, HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_STREAMING_STOPPED or HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_STREAMING_EXIT_CALLED.

SetBasebandFilterBandwidth(HackRFDevice*, uint)

Set baseband filter bandwidth.

Possible values: 1.75, 2.5, 3.5, 5, 5.5, 6, 7, 8, 9, 10, 12, 14, 15, 20, 24, 28MHz, default \f$ \le 0.75 \cdot F_s \f$ The functions HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.ComputeBasebandFilterBandWidth(System.UInt32) and HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.ComputeBasebandFilterBandWidth_round_down_lt(System.UInt32) can be used to get a valid value nearest to a given value.

Setting the sample rate causes the filter bandwidth to be (re)set to its default \f$ \le 0.75 \cdot F_s \f$ value, so setting sample rate should be done before setting filter bandwidth.

```
public static extern HackrfError SetBasebandFilterBandwidth(HackRFDevice* device,
uint bandwidth_hz)
```

Parameters

`device` HackRFDevice*

device to configure.

`bandwidth_hz` uint

baseband filter bandwidth in Hz.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetBiasTOptions(HackRFDevice*, HackRFBiasTUserSettingReq*)

Configure bias tee behavior of the HackRF device when changing RF states.

This function allows the user to configure bias tee behavior so that it can be turned on or off automatically by the HackRF when entering the RX, TX, or OFF state. By default, the HackRF switches off the bias tee when the RF path switches to OFF mode.

The bias tee configuration is specified via a bitfield: 0000000TmmRmmOmm

Where setting T/R/O bits indicates that the TX/RX/Off behavior should be set to mode 'mm', 0 = don't modify

mm specifies the bias tee mode:

00 - do nothing. 01 - reserved, do not use. 10 - disable bias tee. 11 - enable bias tee.

```
public static extern HackrfError SetBiasTOptions(HackRFDevice* device,
HackRFBiasTUserSettingReq* req)
```

Parameters

`device` HackRFDevice*

Device to configure.

`req` HackRFBiasTUserSettingReq*

Bias tee states, as a bitfield.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetClockSampleRate(HackRFDevice*, uint, uint)

```
public static extern HackrfError SetClockSampleRate(HackRFDevice* device, uint freq_hz,
uint divider)
```

Parameters

`device` HackRFDevice*

`freq_hz` uint

`divider` uint

Returns

HackrfError

SetFrequency(HackRFDevice*, ulong)

Set the center frequency.

Simple(auto) tuning via specifying a center frequency in Hz.

This setting is not exact and depends on the PLL settings.Exact resolution is not determined, but the actual tuned frequency will be queryable in the future.

```
public static extern HackrfError SetFrequency(HackRFDevice* device, ulong freq_hz)
```

Parameters

`device` HackRFDevice*

Device to tune.

`freq_hz` ulong

freq_hz center frequency in Hz.Defaults to 900MHz. Should be in range 1-6000MHz, but 0-7250MHz is possible. The resolution is ~50Hz, I could not find the exact number.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetFrequency(HackRFDevice*, ulong, ulong, RfPathFilter)

Set the center frequency via explicit tuning.

Center frequency is set to \f$f_{center} = f_{IF k\cdot f_ { LO }\f$ where \f$k\in\left{-1; 0; 1\right}\f$, depending on the value of `path`. See the documentation of HackRFDotnet.NativeApi.Enums.RfPathFilter for details.

```
public static extern HackrfError SetFrequency(HackRFDevice* device, ulong if_freq_hz, ulong
lo_freq_hz, RfPathFilter path)
```

Parameters

`device` HackRFDevice*

Device to tune.

`if_freq_hz` ulong

Tuning frequency of the MAX2837 transceiver IC in Hz. Must be in the range of 2150-2750MHz.

`lo_freq_hz` ulong

Tuning frequency of the RFFC5072 mixer/synthesizer IC in Hz. Must be in the range 84.375-5400MHz, defaults to 1000MHz. No effect if `path` is set to HackRFDotnet.NativeApi.Enums.RfPathFilter.RF_PATH_FILTER_BYPASS.

`path` RfPathFilter

Filter path for mixer. See the documentation for HackRFDotnet.NativeApi.Enums.RfPathFilter for details.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetLnaGain(HackRFDevice*, uint)

Set LNA gain.

Set the RF RX gain of the MAX2837 transceiver IC ("IF" gain setting) in decibels. Must be in range 0-40dB, with 8dB steps.

```
public static extern HackrfError SetLnaGain(HackRFDevice* device, uint value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` uint

RX IF gain value in dB.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetRxOverrunLimit(HackRFDevice*, uint)

Set receive overrun limit.

When this limit is set, after the specified number of samples (bytes, not whole IQ pairs) missing the device will automatically return to IDLE mode, thus stopping operation. Useful for handling cases like program/computer crashes or other problems. The default value 0 means no limit.

Requires USB API version 0x0106 or above!

```
public static extern HackrfError SetRxOverrunLimit(HackRFDevice* device, uint value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` uint

Number of samples to wait before auto-stopping.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetSampleRate(HackRFDevice*, double)

```
public static extern HackrfError SetSampleRate(HackRFDevice* device, double freq_hz)
```

Parameters

`device` HackRFDevice*

`freq_hz` double

Returns

HackrfError

SetTxBlockCompleteCallback(HackRFDevice*, HackRFTxBlockCompleteCallback)

Setup callback to be called when an USB transfer is completed.

This callback will be called whenever an USB transfer to the device is completed, regardless if it was successful or not (indicated by the second parameter).

```
public static HackrfError SetTxBlockCompleteCallback(HackRFDevice* device,
HackRFTxBlockCompleteCallback callback)
```

Parameters

`device` HackRFDevice*

Device to configure.

`callback` HackRFTxBlockCompleteCallback

Callback to call when a transfer is completed.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetTxUnderrunLimit(HackRFDevice*, uint)

Set transmit underrun limit.

When this limit is set, after the specified number of samples (bytes, not whole IQ pairs) missing the device will automatically return to IDLE mode, thus stopping operation. Useful for handling cases like

program/computer crashes or other problems. The default value 0 means no limit.

Requires USB API version 0x0106 or above!

```
public static extern HackrfError SetTxUnderrunLimit(HackRFDevice* device, uint value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` uint

Number of samples to wait before auto-stopping.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetTxVgaGain(HackRFDevice*, uint)

Set RF TX gain of the MAX2837 transceiver IC ("IF" or "VGA" gain setting) in decibels. Must be in range 0-47dB in 1dB steps.

```
public static extern HackrfError SetTxVgaGain(HackRFDevice* device, uint value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` uint

TX IF gain value in dB.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetVgaGain(HackRFDevice*, uint)

Set baseband RX gain of the MAX2837 transceiver IC ("BB" or "VGA" gain setting) in decibels. Must be in range 0-62dB with 2dB steps.

```
public static extern HackrfError SetVgaGain(HackRFDevice* device, uint value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` uint

RX BB gain value in dB.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

StartRx(HackRFDevice*, HackRFSampleBlockCallback, void*)

Start receiving.

Should be called after setting gains, frequency and sampling rate, as these values won't get reset but instead keep their last value, thus their state is unknown.

The callback is called with a HackRFDotnet.NativeApi.Structs.HackrfTransfer object whenever the buffer is full. The callback is called in an async context so no libhackrf functions should be called from it. The callback should treat its argument as read-only.

```
public static HackrfError StartRx(HackRFDevice* device, HackRFSampleBlockCallback callback,
void* rx_ctx)
```

Parameters

`device` HackRFDevice*

Device to configure.

`callback` HackRFSampleBlockCallback

Rx_callback.

`rx_ctx` void*

User provided RX context. Not used by the library, but available to `callback` as HackRFDotnet.NativeApi.Structs.HackrfTransfer.rx_ctx.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

StartRxSweep(HackRFDevice*, HackRFSampleBlockCallback, void*)

Start RX sweep.

See HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.InitSweep(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,System.UInt16*,System.Int32,System.UInt32,System.UInt32,System.UInt32,HackRFDotnet.NativeApi.Enums.SweepStyle) for more info.

Requires USB API version 0x0104 or above!

```
public static HackrfError StartRxSweep(HackRFDevice* device, HackRFSampleBlockCallback
callback, void* rx_ctx)
```

Parameters

`device` HackRFDevice*

Device to start sweeping.

`callback` HackRFSampleBlockCallback

Rx callback processing the received data.

`rx_ctx` void*

User provided RX context.Not used by the library, but available to `callback` as HackRFDotnet.NativeApi.Structs.HackrfTransfer.rx_ctx.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

StartTx(HackRFDevice*, HackRFSampleBlockCallback, void*)

Start transmitting (TX).

⚠ Warning: Transmitting radio signals may be subject to national and international regulations. Use of this function without the appropriate license or authorization may violate FCC regulations (or equivalent regulatory authorities in your region) and could result in legal penalties.

```
public static HackrfError StartTx(HackRFDevice* device, HackRFSampleBlockCallback callback,
void* tx_ctx)
```

Parameters

`device` HackRFDevice*

Device to configure.

`callback` HackRFSampleBlockCallback

Tx_callback.

`tx_ctx` void*

User provided TX context. Not used by the library, but available to `callback` as HackRFDotnet.NativeApi.Structs.HackrfTransfer.tx_ctx.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

StopRx(HackRFDevice*)

Stop receiving.

```
public static extern HackrfError StopRx(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

device to stop RX on.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

StopTx(HackRFDevice*)

Stop transmission.

```
public static extern HackrfError StopTx(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to stop TX on.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

Class HackRfNativeLib.Devices

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib.Devices
```

Inheritance

object ← HackRfNativeLib.Devices

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

CloseDevice(HackRFDevice*)

Close a previously opened device.

```
public static extern HackrfError CloseDevice(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to close.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or variant of HackRFDotnet.NativeApi.Enums.System.HackrfError.

DeviceListFree(HackRFDeviceList*)

Free a previously allocated HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice list.

```
public static extern void DeviceListFree(HackRFDeviceList* list)
```

Parameters

`list` HackRFDeviceList*

List to free.

DeviceListOpen(HackRFDeviceList*, int, HackRFDevice**)

Open a HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice from a device list.

```
public static extern HackrfError DeviceListOpen(HackRFDeviceList* list, int idx,
HackRFDevice** device)
```

Parameters

`list` HackRFDeviceList*

Device list to open device from.

`idx` int

Index of the device to open.

`device` HackRFDevice**

Device handle to open.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success, HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_INVALID_PARAM on invalid parameters or other HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

OpenDevice(HackRFDevice**)

Open first available HackRF device.

```
public static extern HackrfError OpenDevice(HackRFDevice** device)
```

Parameters

`device` HackRFDevice**

Device handle.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success, HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_INVALID_PARAM if `device` is NULL, HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_NOT_FOUND if no HackRF devices are found or other HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

OpenDeviceBySerial(string, HackRFDevice**)

Open HackRF device by serial number.

```
public static HackrfError OpenDeviceBySerial(string desired_serial_number,
HackRFDevice** device)
```

Parameters

`desired_serial_number` string

Serial number of device to open. If NULL then default to first device found.

`device` HackRFDevice**

Device handle.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success, HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_INVALID_PARAM on invalid parameters or other HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

QueryDeviceList()

List connected HackRF devices.

```
public static extern HackRFDeviceList* QueryDeviceList()
```

Returns

HackRFDeviceList*

List of connected devices. The list should be freed with HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Devices.DeviceListFree(HackRFDotnet.NativeApi.Structs.Devices.HackRFDeviceList*).

ResetDevice(HackRFDevice*)

Reset HackRF device.

Requires USB API version 0x0102 or above!

```
public static extern HackrfError ResetDevice(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to reset.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetDeviceLeds(HackRFDevice*, byte)

Turn on or off (override) the LEDs of the HackRF device. This function can turn on or off the LEDs of the device. There are 3 controllable LEDs on the HackRF one: USB, RX and TX. On the Rad1o, there are 4 LEDs. Each LED can be set individually, but the setting might get overridden by other functions.

The LEDs can be set via specifying them as bits of a 8 bit number `state`, bit 0 representing the first (USB on the HackRF One) and bit 3 or 4 representing the last LED. The upper 4 or 5 bits are unused. For example, binary value 0bxxxxx101 turns on the USB and TX LEDs on the HackRF One.

```
public static extern HackrfError SetDeviceLeds(HackRFDevice* device, byte state)
```

Parameters

`device` HackRFDevice*

Device to query.

`state` byte

LED states as a bitfield.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetDeviceUiEnabled(HackRFDevice*, byte)

Enable / disable UI display (RAD1O, PortaPack, etc.). Enable or disable the display on display-enabled devices (Rad1o, PortaPack).

Requires USB API version 0x0104 or above!

```
public static extern HackrfError SetDeviceUiEnabled(HackRFDevice* device, byte value)
```

Parameters

`device` HackRFDevice*

device to enable/disable UI on.

`value` byte

Enable UI. Must be 1 or 0.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_LIBUSB on usb error.

Class HackRfNativeLib.Firmware

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib.Firmware
```

Inheritance

object ← HackRfNativeLib.Firmware

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

BoardIdName(HackrfBoardId)

Convert HackRFDotnet.NativeApi.Enums.System.HackrfBoardId into human-readable string.

```
public static extern sbyte* BoardIdName(HackrfBoardId board_id)
```

Parameters

`board_id` HackrfBoardId

Enum to convert.

Returns

sbyte*

Human-readable name of board id.

BoardRevName(HackrfBoardRev)

Convert board revision name.

```
public static extern sbyte* BoardRevName(HackrfBoardRev board_rev)
```

Parameters

`board_rev` HackrfBoardRev

Board revision enum from HackRFDotnet.NativeApi.Lib.HackRfNativeLib.Firmware.ReadBoardRev(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Enums.System.HackrfBoardRev*).

Returns

sbyte*

Human-readable name of board revision. Discards GSG bit.

ClearSpiflashStatus(HackRFDevice*)

Clear the status registers of the W25Q80BV SPI flash chip.

See the datasheet for details of the status registers.

Requires USB API version 0x0103 or above!

```
public static extern HackrfError ClearSpiflashStatus(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to clear.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

EraseSpiflash(HackRFDevice*)

Erase firmware image on the SPI flash.

Should be followed by writing a new image, or the HackRF will be soft-bricked (still rescuable in DFU mode).

```
public static extern HackrfError EraseSpiflash(HackRFDevice* device)
```

Parameters

`device` HackRFDevice*

Device to erase.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

GetClkinStatus(HackRFDevice*, byte*)

Get CLKIN status.

Check if an external clock signal is detected on the CLKIN port.

Requires USB API version 0x0106 or above!

```
public static extern HackrfError GetClkinStatus(HackRFDevice* device, byte* status)
```

Parameters

`device` HackRFDevice*

Device to read status from.

`status` byte*

External clock detected (0/1).

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

GetMcuState(HackRFDevice*, HackRFM0State*)

Get the state of the M0 code on the LPC43xx MCU.

Requires USB API version 0x0106 or above!

```
public static extern HackrfError GetMcuState(HackRFDevice* device, HackRFM0State* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`value` HackRFM0State*

MCU code state.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

LookupBoardIdPlatform(HackrfBoardId)

Lookup platform ID (HACKRF_PLATFORM_xxx) from board id (HackRFDotnet.NativeApi.Enums.System.HackrfBoardId).

```
public static extern uint LookupBoardIdPlatform(HackrfBoardId board_id)
```

Parameters

`board_id` HackrfBoardId

HackRFDotnet.NativeApi.Enums.System.HackrfBoardId enum variant to convert.

Returns

uint

HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_JAWBREAKER, HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_HACKRF1_OG, HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_RAD1O, HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_HACKRF1_R9 or 0

ReadBoardId(HackRFDevice*, byte*)

Read HackRFDotnet.NativeApi.Enums.System.HackrfBoardId from a device.

The result can be converted into a human-readable string via HackRFDotnet.NativeApi.Enums.System.HackrfBoardId.

```
public static extern HackrfError ReadBoardId(HackRFDevice* device, byte* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`value` byte*

HackRFDotnet.NativeApi.Enums.System.HackrfBoardId enum value.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadBoardPartIdSerialNo(HackRFDevice*, ReadPartidSerialNo*)

Read board part ID and serial number.

Read MCU part id and serial number. See the documentation of the MCU for details!

```
public static extern HackrfError ReadBoardPartIdSerialNo(HackRFDevice* device,
ReadPartidSerialNo* read_partid_serialno)
```

Parameters

`device` HackRFDevice*

Device to query.

`read_partid_serialno` ReadPartidSerialNo*

Result of query.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadBoardRev(HackRFDevice*, HackrfBoardRev*)

Read board revision of device.

```
public static extern HackrfError ReadBoardRev(HackRFDevice* device, HackrfBoardRev* value)
```

Parameters

`device` HackRFDevice*

Device to read board revision from.

`value` HackrfBoardRev*

Revision enum, will become one of HackRFDotnet.NativeApi.Enums.System.HackrfBoardRev. Should be initialized with HackRFDotnet.NativeApi.Enums.System.HackrfBoardRev.BOARD_REV_UNDETECTED.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_ERROR_LIBUSB.

ReadMax2837(HackRFDevice*, byte, ushort*)

Directly read the registers of the MAX2837 transceiver IC.

Intended for debugging purposes only!

```
public static extern HackrfError ReadMax2837(HackRFDevice* device, byte register_number,
ushort* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`register_number` byte

Register number to read.

`value` ushort*

Value of the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadRffc5071(HackRFDevice*, byte, ushort*)

Directly read the registers of the RFFC5071/5072 mixer-synthesizer IC.

Intended for debugging purposes only!

```
public static extern HackrfError ReadRffc5071(HackRFDevice* device, byte register_number,
ushort* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`register_number` byte

Register number to read.

`value` ushort*

Value of the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadSi5351c(HackRFDevice*, ushort, ushort*)

Directly read the registers of the Si5351C clock generator IC.

Intended for debugging purposes only!

```
public static extern HackrfError ReadSi5351c(HackRFDevice* device, ushort register_number,
ushort* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`register_number` ushort

Register number to read.

`value` ushort*

Value of the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadSpiflash(HackRFDevice*, uint, ushort, byte*)

Read firmware image on the SPI flash.

Should only be used for firmware verification.

```
public static extern HackrfError ReadSpiflash(HackRFDevice* device, uint address, ushort length, byte* data)
```

Parameters

`device` HackRFDevice*

Device to read from.

`address` uint

Address to read from. Firmware should start at 0

`length` ushort

Length of data to read. Must be at most 256.

`data` byte*

Pointer to buffer.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadSupportedPlatform(HackRFDevice*, uint*)

Read supported platform of device.

Returns a combination of HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_JAWBREAKER	**HackRFDotnet.NativeApi.NativeConstants.HACKRF_PLATFORM_HACKRF1_OG**

Requires USB API version 0x0106 or above!

```
public static extern HackrfError ReadSupportedPlatform(HackRFDevice* device, uint* value)
```

Parameters

`device` HackRFDevice*

Device to query.

`value` uint*

Supported platform bitfield.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadUsbApiVersion(HackRFDevice*, ushort*)

Read HackRF USB API version.

Read version as MM.mm 16-bit value, where MM is the major and mm is the minor version, encoded as the hex digits of the 16-bit number.

```
public static extern HackrfError ReadUsbApiVersion(HackRFDevice* device, ushort* version)
```

Parameters

`device` HackRFDevice*

Device to query.

`version` ushort*

USB API version.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

ReadVersion(HackRFDevice*, byte*, byte)

Read HackRF firmware version as a string.

```
public static extern HackrfError ReadVersion(HackRFDevice* device, byte* version,
byte length)
```

Parameters

`device` HackRFDevice*

Device to query.

`version` byte*

Version string.

`length` byte

Length of allocated string **without null byte** (so set it to `length(arr)-1`).

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetClkoutEnable(HackRFDevice*, byte)

Enable / disable CLKOUT.

Requires USB API version 0x0103 or above!

```
public static extern HackrfError SetClkoutEnable(HackRFDevice* device, byte value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` byte

Clock output enabled (0/1).

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SetHardwareSyncMode(HackRFDevice*, byte)

Set hardware sync mode (hardware triggering).

See the documentation on hardware triggering for details.

Requires USB API version 0x0102 or above!

```
public static extern HackrfError SetHardwareSyncMode(HackRFDevice* device, byte value)
```

Parameters

`device` HackRFDevice*

Device to configure.

`value` byte

Enable (1) or disable (0) hardware triggering.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

SpiflashStatus(HackRFDevice*, byte*)

Read the status registers of the W25Q80BV SPI flash chip.

See the datasheet for details of the status registers. The two registers are read in order.

Requires USB API version 0x0103 or above!

```
public static extern HackrfError SpiflashStatus(HackRFDevice* device, byte* data)
```

Parameters

`device` HackRFDevice*

Device to query.

`data` byte*

char[2] array of the status registers.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

UsbBoardIdName(HackrfUsbBoardId)

Convert HackRFDotnet.NativeApi.Enums.System.HackrfBoardId into human-readable string.

```
public static extern sbyte* UsbBoardIdName(HackrfUsbBoardId usb_board_id)
```

Parameters

`usb_board_id` HackrfUsbBoardId

Enum to convert.

Returns

sbyte*

Human-readable name of board id.

~~WriteCpld(HackRFDevice*, byte*, uint)~~ Deprecated

> This function writes the bitstream, but the firmware auto-overrides at each reset, so no changes will take effect.

Write configuration bitstream into the XC2C64A-7VQ100C CPLD.

Device will need to be reset after hackrf_cpld_write.

```
[Obsolete("This function writes the bitstream, but the firmware auto-overrides at each
reset, so no changes will take effect.")]
public static extern HackrfError WriteCpld(HackRFDevice* device, byte* data,
uint total_length)
```

Parameters

`device` HackRFDevice*

device to configure.

`data` byte*

CPLD bitstream data.

`total_length` uint

length of the bitstream to write.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

WriteRffc5071(HackRFDevice*, byte, ushort)

Directly write the registers of the RFFC5071/5072 mixer-synthesizer IC.

Intended for debugging purposes only!

```
public static extern HackrfError WriteRffc5071(HackRFDevice* device, byte register_number,
ushort value)
```

Parameters

`device` HackRFDevice*

Device to write.

`register_number` byte

Register number to write.

`value` ushort

Value to write in the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

WriteSi5351c(HackRFDevice*, ushort, ushort)

Directly write the registers of the Si5351 clock generator IC.

Intended for debugging purposes only!

```
public static extern HackrfError WriteSi5351c(HackRFDevice* device, ushort register_number,
ushort value)
```

Parameters

`device` HackRFDevice*

Device to write.

`register_number` ushort

Register number to write.

`value` ushort

Value to write in the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

WriteSpiflash(HackRFDevice*, uint, ushort, byte*)

Write firmware image on the SPI flash.

Should only be used for firmware updating. Can brick the device, but it's still rescuable in DFU mode.

```
public static extern HackrfError WriteSpiflash(HackRFDevice* device, uint address, ushort
length, byte* data)
```

Parameters

`device` HackRFDevice*

Device to write on.

`address` uint

Address to write to. Should start at 0.

`length` ushort

Length of data to write. Must be at most 256.

`data` byte*

Data to write.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

hackrf_max2837_write(HackRFDevice*, byte, ushort)

Directly write the registers of the MAX2837 transceiver IC.

Intended for debugging purposes only!

```
public static extern HackrfError hackrf_max2837_write(HackRFDevice* device, byte
register_number, ushort value)
```

Parameters

`device` HackRFDevice*

Device to query.

`register_number` byte

Register number to read.

`value` ushort

Value of the specified register.

Returns

HackrfError

HackRFDotnet.NativeApi.Enums.System.HackrfError.HACKRF_SUCCESS on success or HackRFDotnet.NativeApi.Enums.System.HackrfError variant.

Class HackRfNativeLib.Operacake

Namespace: HackRFDotnet.NativeApi.Lib

Assembly: HackRFDotnet.dll

```
public static class HackRfNativeLib.Operacake
```

Inheritance

object ← HackRfNativeLib.Operacake

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object?, object?) , object.ToString()

Methods

GetOperacakeBoards(HackRFDevice*, byte*)

Query connected Opera Cake boards Returns a @ref HACKRF_OPERACAKE_MAX_BOARDS size array of addresses, with @ref HACKRF_OPERACAKE_ADDRESS_INVALID as a placeholder

```
public static extern int GetOperacakeBoards(HackRFDevice* device, byte* boards)
```

Parameters

`device` HackRFDevice*

`boards` byte*

Returns

int

GetOperacakeMode(HackRFDevice*, byte, OperacakeSwitching

Mode*)

Query Opera Cake mode

```
public static extern int GetOperacakeMode(HackRFDevice* device, byte address,
OperacakeSwitchingMode* mode)
```

Parameters

`device` HackRFDevice*

`address` byte

`mode` OperacakeSwitchingMode*

Returns

int

OperacakeGpioTest(HackRFDevice*, byte, ushort*)

Perform GPIO test on an Opera Cake addon board Value 0xFFFF means "GPIO mode disabled", and hackrf_operacake advises to remove additional add-on boards and retry. Value 0 means all tests passed. In any other values, a 1 bit signals an error. Bits are grouped in groups of 3. Encoding: 0 - u1ctrl - u3ctrl0 - u3ctrl1 - u2ctrl0 - u2ctrl1

```
public static extern int OperacakeGpioTest(HackRFDevice* device, byte address,
ushort* test_result)
```

Parameters

`device` HackRFDevice*

`address` byte

`test_result` ushort*

Returns

int

SetOperacakeDwellTimes(HackRFDevice*, HackRFOperacakeDwellTime*, byte)

Setup Opera Cake dwell times in @ref OPERACAKE_MODE_TIME mode operation Should be called after @ref hackrf_set_operacake_mode **Note:** this configuration applies to all Opera Cake boards in @ref OPERACAKE_MODE_TIME mode

```
public static extern int SetOperacakeDwellTimes(HackRFDevice* device,
HackRFOperacakeDwellTime* dwell_times, byte count)
```

Parameters

`device` HackRFDevice*

`dwell_times` HackRFOperacakeDwellTime*

`count` byte

Returns

int

SetOperacakeFrequencyRanges(HackRFDevice*, HackRFOperacakeFreqRange*, byte)

Setup Opera Cake frequency ranges in @ref OPERACAKE_MODE_FREQUENCY mode operation Should be called after @ref hackrf_set_operacake_mode **Note:** this configuration applies to all Opera Cake boards in @ref OPERACAKE_MODE_FREQUENCY mode

```
public static extern int SetOperacakeFrequencyRanges(HackRFDevice* device,
HackRFOperacakeFreqRange* freq_ranges, byte count)
```

Parameters

`device` HackRFDevice*

`freq_ranges` HackRFOperacakeFreqRange*

`count` byte

Returns

int

SetOperacakeMode(HackRFDevice*, byte, OperacakeSwitchingMode)

Setup Opera Cake operation mode

```
public static extern int SetOperacakeMode(HackRFDevice* device, byte address,
OperacakeSwitchingMode mode)
```

Parameters

`device` HackRFDevice*

`address` byte

`mode` OperacakeSwitchingMode

Returns

int

SetOperacakePorts(HackRFDevice*, byte, byte, byte)

Setup Opera Cake ports in @ref OPERACAKE_MODE_MANUAL mode operation Should be called after @ref hackrf_set_operacake_mode. A0 and B0 must be connected to opposite sides (A->A and B->B or A->B and B->A but not A->A and B->A or A->B and B->B)

```
public static extern int SetOperacakePorts(HackRFDevice* device, byte address, byte port_a,
byte port_b)
```

Parameters

`device` HackRFDevice*

`address` byte

`port_a` byte

`port_b` byte

Returns

int

~~SetOperacakeRanges(HackRFDevice*, byte*, byte)~~ Deprecated

> Use hackrf_set_operacake_freq_ranges instead.

Setup Opera Cake frequency ranges in @ref OPERACAKE_MODE_FREQUENCY mode operation Old function to set ranges with. Use @ref hackrf_set_operacake_freq_ranges instead! **Note:** this configuration applies to all Opera Cake boards in @ref OPERACAKE_MODE_FREQUENCY mode

```
[Obsolete("Use hackrf_set_operacake_freq_ranges instead.")]
public static extern int SetOperacakeRanges(HackRFDevice* device, byte* ranges,
byte num_ranges)
```

Parameters

`device` HackRFDevice*

`ranges` byte*

`num_ranges` byte

Returns

int

Namespace HackRFDotnet.NativeApi.Structs

Namespaces

HackRFDotnet.NativeApi.Structs.Devices

HackRFDotnet.NativeApi.Structs.System

Structs

HackRFBiasTUserSettingReq

User settings for user-supplied bias tee defaults.

HackRFBoolUserSetting

Helper struct for HackRFDotnet.NativeApi.Structs.HackRFBiasTUserSettingReq. If HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.do_update is true, then the values of HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.change_on_mode_entry and HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.enabled will be used as the new default. If HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.do_update is false, the current default will not change.

HackrfTransfer

USB transfer information passed to RX or TX callback. A callback should treat all these fields as read-only except that a TX callback should write to the data buffer and may write to HackRFDotnet.NativeApi.Structs.HackrfTransfer.valid_length to indicate that a smaller number of bytes is to be transmitted.

Delegates

HackRFFlushCallback

HackRFSampleBlockCallback

HackRFTxBlockCompleteCallback

Struct HackRFBiasTUserSettingReq

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

User settings for user-supplied bias tee defaults.

```
public struct HackRFBiasTUserSettingReq
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

off

```
public HackRFBoolUserSetting off
```

Field Value

HackRFBoolUserSetting

rx

```
public HackRFBoolUserSetting rx
```

Field Value

HackRFBoolUserSetting

tx

```
public HackRFBoolUserSetting tx
```

Field Value

HackRFBoolUserSetting

Struct HackRFBoolUserSetting

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

Helper struct for HackRFDotnet.NativeApi.Structs.HackRFBiasTUserSettingReq. If HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.do_update is true, then the values of HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.change_on_mode_entry and HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.enabled will be used as the new default. If HackRFDotnet.NativeApi.Structs.HackRFBoolUserSetting.do_update is false, the current default will not change.

```
public struct HackRFBoolUserSetting
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

change_on_mode_entry

Change value on mode entry.

```
public bool change_on_mode_entry
```

Field Value

bool

do_update

If true, update default values.

```
public bool do_update
```

Field Value

bool

enabled

Enabled.

```
public bool enabled
```

Field Value

bool

Struct HackrfTransfer

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

USB transfer information passed to RX or TX callback. A callback should treat all these fields as read-only except that a TX callback should write to the data buffer and may write to HackRFDotnet.NativeApi.Structs.HackrfTransfer.valid_length to indicate that a smaller number of bytes is to be transmitted.

```
public struct HackrfTransfer
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

buffer

Transfer data buffer (interleaved 8 bit I/Q samples).

```
public byte* buffer
```

Field Value

byte*

buffer_length

Length of data buffer in bytes.

```
public int buffer_length
```

Field Value

int

device

HackRF USB device for this transfer.

```
public HackRFDevice* device
```

Field Value

HackRFDevice*

rx_ctx

User provided RX context. Not used by the library, but available to transfer callbacks for use. Set along with the transfer callback using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.StartRx(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Structs.HackRFSampleBlockCallback,System.Void*) or HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.StartRxSweep(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Structs.HackRFSampleBlockCallback,System.Void*).

```
public void* rx_ctx
```

Field Value

void*

tx_ctx

User provided TX context. Not used by the library, but available to transfer callbacks for use. Set along with the transfer callback using HackRFDotnet.NativeApi.Lib.HackRfNativeLib.DeviceStreaming.StartRx(HackRFDotnet.NativeApi.Structs.Devices.HackRFDevice*,HackRFDotnet.NativeApi.Structs.HackRFSampleBlockCallback,System.Void*).

```
public void* tx_ctx
```

Field Value

void*

valid_length

Number of buffer bytes that were transferred.

```
public int valid_length
```

Field Value

int

Delegate HackRFFlushCallback

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

```
public delegate void HackRFFlushCallback(nint flush_ctx, int status)
```

Parameters

`flush_ctx` nint

`status` int

Delegate HackRFSampleBlockCallback

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

```
public delegate int HackRFSampleBlockCallback(HackrfTransfer* transfer)
```

Parameters

`transfer` HackrfTransfer*

Returns

int

Delegate HackRFTxBlockCompleteCallback

Namespace: HackRFDotnet.NativeApi.Structs

Assembly: HackRFDotnet.dll

```
public delegate void HackRFTxBlockCompleteCallback(HackrfTransfer* transfer, int status)
```

Parameters

`transfer` HackrfTransfer*

`status` int

Namespace HackRFDotnet.NativeApi.Structs.Devices

Structs

HackRFDevice

HackRFDeviceList

HackRFOperacakeDwellTime

HackRFOperacakeFreqRange

Struct HackRFDevice

Namespace: HackRFDotnet.NativeApi.Structs.Devices

Assembly: HackRFDotnet.dll

```
public struct HackRFDevice
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Struct HackRFDeviceList

Namespace: HackRFDotnet.NativeApi.Structs.Devices

Assembly: HackRFDotnet.dll

```
public struct HackRFDeviceList
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

devicecount

Number of connected HackRF devices.

```
public int devicecount
```

Field Value

int

serial_numbers

Array of human-readable serial numbers. Each entry can be NULL.

```
public char** serial_numbers
```

Field Value

char**

usb_board_ids

ID of each board, based on USB product ID.

```
public HackrfUsbBoardId* usb_board_ids
```

Field Value

HackrfUsbBoardId*

usb_device_index

USB device index for each HW entry.

```
public int* usb_device_index
```

Field Value

int*

usb_devicecount

Number of all queried USB devices.

```
public int usb_devicecount
```

Field Value

int

usb_devices

All USB devices (as libusb_device** array).

```
public void** usb_devices
```

Field Value

void**

Struct HackRFOperacakeDwellTime

Namespace: HackRFDotnet.NativeApi.Structs.Devices

Assembly: HackRFDotnet.dll

```
public struct HackRFOperacakeDwellTime
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

dwell

Dwell time for port (in number of samples)

```
public uint dwell
```

Field Value

uint

port

Port to connect A0 to (B0 mirrors this choice) Must be one of operacake_ports

```
public byte port
```

Field Value

byte

Struct HackRFOperacakeFreqRange

Namespace: HackRFDotnet.NativeApi.Structs.Devices

Assembly: HackRFDotnet.dll

```
public struct HackRFOperacakeFreqRange
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

freq_max

Stop frequency (in MHz)

```
public ushort freq_max
```

Field Value

ushort

freq_min

Start frequency (in MHz)

```
public ushort freq_min
```

Field Value

ushort

port

Port (A0) to use for that frequency range. Port B0 mirrors this. Must be one of operacake_ports

```
public byte port
```

Field Value

byte

Namespace HackRFDotnet.NativeApi.Structs.System

Structs

HackRFM0State

State of the SGPIO loop running on the M0 core.

ReadPartidSerialNo

Struct HackRFM0State

Namespace: HackRFDotnet.NativeApi.Structs.System

Assembly: HackRFDotnet.dll

State of the SGPIO loop running on the M0 core.

```
public struct HackRFM0State
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

active_mode

Active mode. Same values as requested_mode. Possible values are the same as in HackRFDotnet.NativeApi.Structs.System.HackRFM0State.requested_mode.

```
public uint active_mode
```

Field Value

uint

error

Error that caused M0 to revert to IDLE. 0 (NONE), 1 (RX_TIMEOUT), 2 (TX_TIMEOUT), 3 (MISSED_DEADLINE).

```
public uint error
```

Field Value

uint

longest_shortfall

Longest shortfall in bytes.

```
public uint longest_shortfall
```

Field Value

uint

m0_count

Number of bytes transferred by the M0.

```
public uint m0_count
```

Field Value

uint

m4_count

Number of bytes transferred by the M4.

```
public uint m4_count
```

Field Value

uint

next_mode

Mode which will be switched to when threshold is reached. Possible values are the same as in Hack RFDotnet.NativeApi.Structs.System.HackRFM0State.requested_mode.

```
public uint next_mode
```

Field Value

uint

num_shortfalls

Number of shortfalls.

```
public uint num_shortfalls
```

Field Value

uint

request_flag

Request flag, 0 means request is completed, any other value means request is pending.

```
public ushort request_flag
```

Field Value

ushort

requested_mode

Requested mode. Possible values: 0 (IDLE), 1 (WAIT), 2 (RX), 3 (TX_START), 4 (TX_RUN).

```
public ushort requested_mode
```

Field Value

ushort

shortfall_limit

Shortfall limit in bytes.

```
public uint shortfall_limit
```

Field Value

uint

threshold

Threshold HackRFDotnet.NativeApi.Structs.System.HackRFM0State.m0_count value (in bytes) for next mode change.

```
public uint threshold
```

Field Value

uint

Struct ReadPartidSerialNo

Namespace: HackRFDotnet.NativeApi.Structs.System

Assembly: HackRFDotnet.dll

```
public struct ReadPartidSerialNo
```

Inherited Members

object.Equals(object?) , object.Equals(object?, object?) , object.GetHashCode() , object.GetType() , object.ReferenceEquals(object?, object?) , object.ToString()

Fields

part_id

```
public uint* part_id
```

Field Value

uint*

serial_no

```
public uint* serial_no
```

Field Value

uint*