



Irisys .NET API for IRC3xxx Series (Blackfin-based) Devices

API Version 3.0.40504.01

Contents

1	Irisys .NET API for IRC3xxx Series (Blackfin-based) Devices	5
1.1	Introduction	5
1.2	Release Notes	5
1.2.1	Version 3.0.40504.01 (May 2012)	5
1.2.2	Version 2.0.30712.01 (August 2011)	6
2	Basic Usage	7
2.1	Linking to the API	7
2.2	Initializing the API	7
2.3	Connecting to a Device	7
2.3.1	Serial Connection	7
2.3.2	IP Connection	7
2.4	Working with device configuration	8
2.5	Disconnecting from a device	8
2.6	Setting IP board configuration	8
3	Application Notes	10
3.1	Using Irisys Timezones	10
3.1.1	Missing Timezones	10
3.2	Using Device Status for Troubleshooting	10
3.3	Packet Timeouts	11
4	Sample Applications	12
4.1	BlackfinAPIConsole	12
4.2	Server Samples	12
4.2.1	ServerSample1	12
4.2.2	ServerSample2	12
4.2.3	ServerSample3	13
4.2.4	ServerSample4	13
4.3	BlackfinAPIWPFSample	13
5	Count Logs	14
5.1	Introduction	14
5.2	Logging Period	14
5.3	Accessing Count Logs	14
5.4	Storage	14
5.5	Resetting Count Logs	15
6	Frequently Asked Questions	16
7	Class Documentation	17
7.1	Irisys.BlackfinAPI.Blackfin Class Reference	17
7.1.1	Detailed Description	19
7.1.2	Constructor & Destructor Documentation	19
7.1.2.1	Blackfin	19
7.1.3	Member Function Documentation	19
7.1.3.1	GetCurrentCount	19
7.1.3.2	GetLastNCounts	20
7.1.3.3	GetCounts	20

7.1.3.4	GetDeviceStatus	20
7.1.3.5	GetTime	21
7.1.3.6	GetUnitTime	21
7.1.3.7	GetUptime	21
7.1.3.8	GetUpTime	21
7.1.3.9	GetUnitTimeZone	21
7.1.3.10	GetMACAddress	21
7.1.3.11	GetClientConfigHostname	22
7.1.3.12	GetClientConfigIP	22
7.1.3.13	GetDeviceName	22
7.1.3.14	GetDeviceID	23
7.1.3.15	GetSiteName	23
7.1.3.16	GetSiteID	23
7.1.3.17	GetLocaleString	23
7.1.3.18	GetUserString	24
7.1.3.19	GetRegisterLabels	24
7.1.3.20	GetSerialNumber	24
7.1.3.21	GetIPFirmwareVersion	25
7.1.3.22	GetMonitorFirmwareVersion	25
7.1.3.23	GetDNS1	25
7.1.3.24	GetDNS2	25
7.1.3.25	GetDNS3	26
7.1.3.26	GetClientConfigPort	26
7.1.3.27	GetClientConfigPortNumber	26
7.1.3.28	GetClientConfigTimeout	26
7.1.3.29	GetClientConfigEnable	26
7.1.3.30	GetDHCPEnabled	27
7.1.3.31	GetIPAddress	27
7.1.3.32	GetSubnetMask	27
7.1.3.33	GetGateway	28
7.1.3.34	GetCountLogPeriod	28
7.1.3.35	GetNetworkChecksum	28
7.1.3.36	ResetCountLogs	29
7.1.3.37	ResetCurrentCount	29
7.1.3.38	ResetDeviceStatus	29
7.1.3.39	SetPacketTimeout	29
7.1.3.40	SetCommsTimeout	29
7.1.3.41	SetUnitTimeZone	29
7.1.3.42	SetUnitTime	30
7.1.3.43	SetTime	30
7.1.3.44	SetDeviceName	30
7.1.3.45	SetDeviceID	30
7.1.3.46	SetSiteName	31
7.1.3.47	SetSiteID	31
7.1.3.48	SetLocaleString	31
7.1.3.49	SetUserString	32
7.1.3.50	SetDNS1	32
7.1.3.51	SetDNS2	32
7.1.3.52	SetDNS3	32
7.1.3.53	SetClientConfigIP	33
7.1.3.54	SetClientConfigHostname	33
7.1.3.55	SetClientConfigPort	33

7.1.3.56	SetClientConfigPortNumber	34
7.1.3.57	SetClientConfigTimeout	34
7.1.3.58	SetClientConfigEnable	34
7.1.3.59	SetCountLogPeriod	34
7.1.4	Member Data Documentation	35
7.1.4.1	MAX_PROPERTY_STRING	35
7.1.4.2	MAX_DEVICEID_STRING	35
7.1.5	Property Documentation	35
7.1.5.1	SerialNumber	35
7.1.5.2	CountLogPeriod	35
7.1.5.3	MACAddress	35
7.1.5.4	ClientConfigIP	35
7.1.5.5	ClientConfigHostname	35
7.1.5.6	ClientConfigPort	35
7.1.5.7	ClientConfigPortNumber	36
7.1.5.8	ClientConfigTimeout	36
7.1.5.9	ClientConfigEnable	36
7.1.5.10	Counts	36
7.1.5.11	DeviceStatus	36
7.1.5.12	UnitTime	36
7.1.5.13	LocaleString	36
7.1.5.14	UserString	36
7.1.5.15	RegisterLabels	36
7.1.5.16	DeviceName	37
7.1.5.17	DeviceID	37
7.1.5.18	SiteName	37
7.1.5.19	SiteID	37
7.1.5.20	DHCPEnabled	37
7.1.5.21	IPAddress	37
7.1.5.22	SubnetMask	37
7.1.5.23	Gateway	37
7.1.5.24	IPFirmwareVersion	37
7.1.5.25	MonitorFirmwareVersion	38
7.1.5.26	DNS1	38
7.1.5.27	DNS2	38
7.1.5.28	DNS3	38
7.1.5.29	UpTime	38
7.1.5.30	UnitTimeZone	38
7.1.5.31	NetworkChecksum	38
7.1.5.32	PacketTimeout	38
7.2	Irisys.BlackfinAPI.BlackfinEngine Class Reference	38
7.2.1	Detailed Description	39
7.2.2	Member Enumeration Documentation	39
7.2.2.1	ConnectionErrorType	39
7.2.3	Constructor & Destructor Documentation	39
7.2.3.1	BlackfinEngine	39
7.2.4	Member Function Documentation	40
7.2.4.1	ConnectionErrorHandler	40
7.2.4.2	GetAPIVersion	40
7.2.4.3	StartEngine	40
7.2.4.4	ShutdownEngine	40
7.2.4.5	AddNewCounterEndPoint	40

7.2.4.6	AddNewCounterEndPoint	40
7.2.4.7	AddNewCounterEndPoint	41
7.2.4.8	AddNewCounterEndPoint	41
7.2.4.9	RemoveCounterEndPoint	41
7.2.4.10	ClockCounters	42
7.3	Irisys.BlackfinAPI.Interfaces.Count Class Reference	42
7.3.1	Detailed Description	42
7.3.2	Member Data Documentation	42
7.3.2.1	countLines	42
7.3.2.2	countTime	42
7.4	Irisys.BlackfinAPI.Interfaces.DeviceLogEntry Class Reference	42
7.4.1	Detailed Description	42
7.4.2	Member Data Documentation	43
7.4.2.1	errorDescription	43
7.4.2.2	timestamp	43
7.5	Irisys.BlackfinAPI.Interfaces.DeviceStatus Class Reference	43
7.5.1	Detailed Description	43
7.5.2	Member Data Documentation	43
7.5.2.1	m_warnList	43
7.5.2.2	m_errorList	43
7.5.2.3	m_infoList	43
7.6	IrisysTime.IrisysTimeZone Class Reference	43
7.6.1	Detailed Description	44
7.6.2	Constructor & Destructor Documentation	44
7.6.2.1	IrisysTimeZone	44
7.6.3	Property Documentation	44
7.6.3.1	ApplyDST	44
7.6.4	Event Documentation	45
7.6.4.1	PropertyChanged	45
7.7	IrisysTime.IrisysTimeZoneInfo Class Reference	45
7.7.1	Detailed Description	45
7.7.2	Constructor & Destructor Documentation	45
7.7.2.1	IrisysTimeZoneInfo	45
7.7.3	Member Function Documentation	46
7.7.3.1	Equals	46
7.7.3.2	GetHashCode	46
7.7.3.3	ToString	46
7.7.4	Member Data Documentation	46
7.7.4.1	UtcTimeZone	46
7.7.5	Property Documentation	46
7.7.5.1	BaseUtcOffset	46
7.7.5.2	SupportsDST	47
7.7.5.3	TimeZoneID	47
7.7.5.4	DisplayName	47
7.7.5.5	TimeZones	47
7.8	Irisys.BlackfinAPI.SerialNumberConverter Class Reference	47
7.8.1	Detailed Description	47
7.8.2	Member Function Documentation	47
7.8.2.1	ConvertSerialNumber	47

1 Irisys .NET API for IRC3xxx Series (Blackfin-based) Devices

1.1 Introduction

The Irisys IRC3xxx series of People Counters provide TCP/IP connectivity via an IP master device. Irisys provides four Application Programming Interfaces (APIs) that developers can use to connect to and download count data from IRC3xxx series devices, in addition to setting and reading a number of user configurable identification strings on the device. The available APIs are;

- .NET (3.5)
- Java (1.6)
- Win32 (C++)
- Linux (C++)

Third parties can use these APIs to integrate count data from IRC3xxx series devices into their own custom applications. It is not intended to be used for the setup of counter networks or the retrieval of advanced data such as path maps or video, thus no functionality is provided for these tasks.

This documentation is intended for use by developers with a working knowledge of application programming and it is recommended to be read alongside the accompanying [Sample Applications](#). Note that this documentation assumes devices have been installed in accordance with our recommended practices, as outlined in the accompanying Installation Guide IPU40182 and Applications Guide IPU40184. Failure to follow these practices may affect the accuracy of the count data provided by IRC3xxx series devices.

The IRC3xxx series of devices use a Blackfin processing chip. Blackfin is a Registered Trademark of Analog Devices Inc.

1.2 Release Notes

1.2.1 Version 3.0.40504.01 (May 2012)

This release of the API adds support for new counter features in the latest firmware for IRC3xxx series devices, such as the ability to support up to 32 count registers.

Changelog

- Added new `Irisys.BlackfinAPI.Blackfin.GetRegisterLabels()` function and the associated accessor function `Irisys.BlackfinAPI.Blackfin.RegisterLabels()` to download the new register labels
- Functions to set property strings will now throw an `ArgumentException` if the length of the string to be set exceeds the maximum length of that field. Affected functions include;

- `Irisys.BlackfinAPI.Blackfin.SetDeviceName()`
- `Irisys.BlackfinAPI.Blackfin.SetDeviceID()`
- `Irisys.BlackfinAPI.Blackfin.SetSiteName()`
- `Irisys.BlackfinAPI.Blackfin.SetSiteID()`

- `Irisys.BlackfinAPI.Blackfin.SetLocaleString()`
- `Irisys.BlackfinAPI.Blackfin.SetUserString()`
- Added new `Irisys.BlackfinAPI.Blackfin.GetNetworkChecksum()` function and the associated accessor function `Irisys.BlackfinAPI.Blackfin.NetworkChecksum()` to create a checksum of the settings for all devices on the connected CAN network

1.2.2 Version 2.0.30712.01 (August 2011)

In this version of the API some functions have been renamed to provide consistency across all Irisys APIs for Blackfin-based devices. This is a breaking change from previous versions of the API and therefore any code based on an older version of the API may need to be updated to compile with this version. Other significant changes in this version include support for multiple (>2) count lines, Irisys time zones and improved documentation.

Changelog

- Merged `BlackfinUser.dll`, `BlackfinCore.dll` and `BaseDevice.dll` libraries into a single library named `BlackfinUser.dll`
- Moved `Blackfin` and `BlackfinEngine` classes into the `Irisys.BlackfinAPI` namespace, from the `Blackfin_API` and `IrisysDevices.BlackfinCore` namespaces respectively.
- Added support for Irisys time zones
- Added support for getting & setting a new 'user string' setting
- Deprecated the following functions and accessors, each of which has been replaced with an equivalent function which is consistent with the other Blackfin APIs;
 - `Irisys.BlackfinAPI.Blackfin.GetTime()`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.GetUnitTime()`
 - `Irisys.BlackfinAPI.Blackfin.SetTime()`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.SetUnitTime()`
 - `Irisys.BlackfinAPI.Blackfin.GetClientConfigPortNumber()`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.GetClientConfigPort()`
 - `Irisys.BlackfinAPI.Blackfin.SetClientConfigPortNumber()`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.SetClientConfigPort()`
 - `Irisys.BlackfinAPI.Blackfin.GetUptime()`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.GetUpTime()`
 - `Irisys.BlackfinAPI.Blackfin.ClientConfigPortNumber`, which has been replaced with `Irisys.BlackfinAPI.Blackfin.ClientConfigPort`
- Made a lot of classes internal rather than public. This should not affect user code
- Consolidated and improved `SampleServer` sample application to aid understanding
- Added `Irisys.BlackfinAPI.SerialNumberConverter` class to convert between numeric and alpha-numeric serial number representations

2 Basic Usage

This section provides an introduction to the basic usage of the Irisys API (.NET) to connect to a device and perform some simple tasks with it. For more detailed information about the functions named in this section and other functions available in the API refer to the included class documentation.

2.1 Linking to the API

You need to include `BlackfinUser.dll` as a reference in your .NET project. This will make all of the user classes available from within the `Irisys.BlackfinAPI` and `IrisysTime` namespaces. Visual Studio automatically copies the DLL into the executable path.

2.2 Initializing the API

Before attempting to use any of the functions or classes provided with this API you must first initialise it by creating an object of type `Irisys.BlackfinAPI.BlackfinEngine` and calling the `Irisys.BlackfinAPI.BlackfinEngine.StartEngine()` member function. This object will be used by your application to create and destroy connections to devices.

Upon shutdown your application should clean up your engine by calling the `Irisys.BlackfinAPI.BlackfinEngine.ShutdownEngine()` member function. Note that any connected `Irisys.BlackfinAPI.Blackfin` objects should first be disconnected by passing them to the `Irisys.BlackfinAPI.BlackfinEngine.RemoveCounterEndPoint()` member function. After shutting down the engine your application must not use any other functions or classes provided with this API.

2.3 Connecting to a Device

The API supports connecting to devices by either serial or TCP/IP, where supported by the device. For either type of connection you must first create an object of type `Irisys.BlackfinAPI.Blackfin` using the default constructor.

2.3.1 Serial Connection

To connect via a serial COM port you should create an object of type `System.IO.Ports.SerialPort` on the desired port. You should then call the `Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint()` function on your initialized `Irisys.BlackfinAPI.BlackfinEngine` object, passing as the parameters the `Irisys.BlackfinAPI.Blackfin` object you created earlier and your new serial port object.

2.3.2 IP Connection

There are two mechanisms for creating connections to devices using TCP/IP, which are only possible on devices which feature an IP board. Your application can connect to the device directly by creating an object of type `System.Net.Sockets.Socket` and connecting it to the devices IP address.

Alternatively your application can receive an incoming connection from a device with client connection mode enabled, which can be configured via the People Counter Setup Tool or by using the `SetClientConfig*` family

of commands in the API. An object of type `System.Net.Sockets.TcpListener` can be used to listen for these connections, as demonstrated in the sample server applications provided.

Regardless of how you created your IP connection you should call the `Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint()` function of your initialized `Irisys.BlackfinAPI.BlackfinEngine` object, passing in the `Irisys.BlackfinAPI.Blackfin` object created earlier and your connected socket.

2.4 Working with device configuration

Once an `Irisys.BlackfinAPI.Blackfin` object is connected to a device there are many API functions that can be used to interact with the device. Many of these functions relate to getting and setting device configuration items such as the device id and device name. The usage of these functions are all broadly similar to the example below demonstrating how to get and set the site name configuration item on a connected device.

```
// In this code bfin is a connected instance of the \bfclass class

// Getting the site name from the device
if ( bfin.GetSiteName() )
    Console.WriteLine( "Device site name is " + bfin.SiteName );
else
    Console.WriteLine( "Failed to get site name" );

// Setting the site name on the device
if ( bfin.SetSiteName("My Site 1") )
    Console.WriteLine( "Successfully set new site name on the device" );
else
    Console.WriteLine( "Failed to set site name" );
```

The `Irisys.BlackfinAPI.Blackfin.GetSiteName()` function initiates a blocking communications call to the connected device to query its configured site name string, returning `true` upon success or `false` after a timeout or any other failure. If successful the site name string configured on the device is available via the `Irisys.BlackfinAPI.Blackfin.SiteName` property.

Similarly the `Irisys.BlackfinAPI.Blackfin.SetSiteName()` initiates a blocking communications call to the connected device to write the new site name string to its configuration, returning `true` upon success or `false` after a timeout or any other failure.

2.5 Disconnecting from a device

To disconnect a `Irisys.BlackfinAPI.Blackfin` object from a serial port or IP socket you should pass it to the `Irisys.BlackfinAPI.BlackfinEngine.RemoveCounterEndPoint()` function of your initialized `Irisys.BlackfinAPI.BlackfinEngine` object. Your application must disconnect all devices in this way before shutting down the engine as part of its general cleanup.

2.6 Setting IP board configuration

Whilst it is possible to call the `SetClientConfig*` and `SetDNS*` families of functions whilst connected to a device by TCP/IP it is **NOT** recommended as this will cause the IP board to be reset and drop the connection. If your application does this your `Irisys.BlackfinAPI.Blackfin` will be unable to communicate with the device and you should disconnect it and destroy the object before, if necessary, creating a new connection as outlined above. If a connection to a device connected in client connection mode is dropped in this way you may be unable to connect

to that device again until the TCP/IP timeout occurs, which may take up to 10 minutes. It is recommended that you connect to a device via serial if you wish to call any of these functions. Note that the `GetClientConfig*` and `GetDNS*` families of functions are not affected by this.

3 Application Notes

3.1 Using Irisys Timezones

As of API version 2.0.30712.01, it is now possible to get and set the time zone in which a device resides - the same setting which is available in the People Counter Setup Tool and Validation Tool. This setting does not affect the `Irisys.BlackfinAPI.Blackfin.GetUnitTime()` function which always returns a value in UTC time.

The time zone setting is restricted to a (large) range of time zones which correspond to Windows time zones ID's. The function `IrisysTime.IrisysTimeZoneInfo.TimeZones` enumerates the set of time zones available. Each `IrisysTime.IrisysTimeZone` object has a base UTC offset and a Boolean value representing whether or not the time zone supports Daylight Savings Time (DST). When setting the time zone, the appropriate `IrisysTime.IrisysTimeZoneInfo` object must be passed along with a boolean representing whether or not DST is to be applied or not. Note the distinction between the user information returned by `IrisysTime.IrisysTimeZone.ApplyDST` and the API information returned by `IrisysTime.IrisysTimeZoneInfo.SupportsDST`.

If DST is to be taken into account when converting a UTC time to a local time then it is necessary to have more information than the API provides - the base UTC offset must be adjusted by a set of rules dependant on the particular time zone. These rules are non-trivial for many zones and are typically represented in a database. This information is available in the Windows registry, or by using the `System.TimeZoneInfo` class.

General procedure for calculating local time:

- Retrieve UTC time from device using `Irisys.BlackfinAPI.Blackfin.GetUnitTime()`
- Retrieve time zone from device using `Irisys.BlackfinAPI.Blackfin.GetUnitTimeZone()`
- If not applying daylight savings rules, add the time zone base UTC offset to the UTC time.
- If applying daylight savings rules:
 - Query database (e.g. the Windows Registry) for daylight savings rules
 - Apply rules to the UTC time to generate appropriate UTC offset for that point in time
 - Add UTC offset to the UTC time to get adjusted local time

3.1.1 Missing Timezones

Some of the time zone IDs contained within the Irisys API do not exist on some versions of Microsoft Windows when the latest time zone updates have not been installed, therefore you are advised not to assume the registry keys containing the time zone information exist on a given target machine.

You can add the missing time zones on Microsoft Windows NT 5.0 (XP, Server 2000) and newer by installing the latest time zone update via Windows Update or by downloading the latest time zone update package from the Microsoft Support website.

3.2 Using Device Status for Troubleshooting

The `Irisys.BlackfinAPI.Blackfin.GetDeviceStatus()` function allows you to query the diagnostic log of the connected device, the results of which will be available via the `Irisys.BlackfinAPI.Blackfin.DeviceStatus` property. This can be used to determine if the device is in an error state or is issuing any warning messages, as well as informative messages such as the last reset time.

Note that IRC3xxx series devices will reset a device every minute in the event that a fatal error condition (such as array failure) has occurred. This ensures the device never becomes unresponsive, however it will cause any communications calls to be more likely to fail, especially those which take a long time to complete such as downloading the count log.

3.3 Packet Timeouts

Communications calls made by an `Irisys.BlackfinAPI.Blackfin` instance are marked as having failed if no response is received within the configured timeout period, which defaults to 5 seconds. You can query the current timeout period using the `Irisys.BlackfinAPI.Blackfin.PacketTimeout` property. For the majority of API functions this timeout is the maximum time the function will block for before returning a result, with the exceptions to this rule being:

- `Irisys.BlackfinAPI.Blackfin.GetCounts()`
- `Irisys.BlackfinAPI.Blackfin.GetLastNCounts()`
- `Irisys.BlackfinAPI.Blackfin.GetDeviceStatus()`

If you are connecting to a device on a high latency network it may be useful to increase the timeout period using the `Irisys.BlackfinAPI.Blackfin.SetPacketTimeout()` function to allow the API more time to wait for responses from that device, however this has the side effect of increasing the delay before a blocking API function will return a failure when the connection has been lost. One potential strategy is to count the number of failed API calls and, upon reaching a set threshold, raise the timeout value to try and work around network latency issues.

4 Sample Applications

This section provides a brief description of the sample applications included with the .NET version of the API. If you have received this document without the sample applications listed below please contact your supplier.

4.1 BlackfinAPIConsole

A simple console application which demonstrates the correct procedure for connecting directly to a device using a TCP/IP socket and allows the user to test the various functions available through the API.

This application is a useful way of exploring the functionality available through the API and performing simple tasks on a device and could easily be modified to work with batch or powershell scripts if desired, allowing the API to be used indirectly with almost any programming language.

The entry point for the application creates a BlackfinEngine object and initializes it so it is ready to create connections. It then prompts the user for an IP address of a Blackfin device and creates a TCP/IP socket to connect to that address on port 4505. A Blackfin object is created and connected to the socket by passing both to the AddNewCounterEndPoint function of the BlackfinEngine. If the connection is successful it drops into a command processing loop until the device is disconnected, at which point the engines RemoveCounterEndPoint is called to clean up the connection and the application will prompt the user for a new IP address to connect to.

Whilst in the command processing loop any of the commands available in the API can be tested by typing in their name when prompted and providing appropriate parameters on request. The help command can be used to output a list of all available commands and a short description of that command. Note that using the SetClientConfig* or SetDNS* commands will disconnect the application from a device and you must reconnect manually if desired.

4.2 Server Samples

A selection of sample server applications which listen for and accept client connections from Blackfin devices, upon which they execute a series of API calls. Each of the samples represents a different usage scenario.

4.2.1 ServerSample1

This server sample adds all incoming connections to a queue for processing, ensuring no incoming connections are missed whilst the previous one is being processed. The data thread checks to see if there are any Blackfin objects in its queue and, if so, removes the first item. It then executes a series of API calls on that object before disconnecting it from the server. This usage scenario ensures only one device is actively communicating with the application at any given time, but allows for other devices to connect and wait to be processed.

4.2.2 ServerSample2

This server sample spawns a thread for each connection it receives, which then creates a Blackfin object for the connection and associates them via the BlackfinEngine. This allows the main thread to continue accepting new connections whilst existing ones are being processed. Once all the API functions have been executed the Blackfin object is removed from the engine, the connection closed and the thread stops. This usage scenario allows multiple devices to communicate with the application at the same time, each in their own thread.

4.2.3 ServerSample3

This server sample spawns a thread for each connection it receives, which then creates a BlackfinWrapper object for the connection. This wrapper around the Blackfin class provides functions for connecting the underlying Blackfin object to a socket via the BlackfinEngine and executing a variety of API calls. All of the functions in the wrapper handle comms errors internally, allowing the main application code to be written more cleanly. Once all the API functions have been executed the BlackfinWrapper is deleted, resulting in the underlying Blackfin object being removed from the engine and the connection closed, at which point the thread stops. This usage scenario allows multiple devices to communicate with the application at the same time, each in their own thread.

4.2.4 ServerSample4

This server sample is almost identical to [ServerSample3](#), except that it does not close the connection once all of the API functions have been executed. Instead the thread calls the same set of API functions every 30 seconds, maintaining a persistent connection to the Blackfin device. This usage scenario allows multiple devices to communicate with the application at the same time, each in their own thread and maintaining persistent connections.

4.3 BlackfinAPIWPFsample

This is a simple GUI application built with the WPF toolkit which allows a user to connect to an IRC3xxx series device via IP or serial and execute a variety of API commands upon it.

Like the BlackfinAPIConsole sample application this provides another way of exploring the functionality available via the API with the added ease of a GUI in place of the command line interface. This application can easily be used to configure some basic settings on a device if desired.

To connect to a device either type its IP address into the text box below the Connect button and click Connect or select a serial COM# port from the list and click the Serial Connect button. If successful the buttons background will turn green and the rest of the GUI will be enabled, otherwise the buttons background will turn red and an error message will be shown. To connect to another device click on the Disconnect or Serial Disconnect button (depending on how you connected to the device) and either enter a new IP address or select a new serial COM# number to connect to and click the appropriate button.

Once connected the other buttons in the application can be used to get and set all of the configuration items which are available via the API and perform other functions available via the API, such as downloading count log entries from the device. The success or failure of each function call is indicated by the background of the button turning green or red and, if getting information, the GUI will be updated with the information retrieved.

5 Count Logs

5.1 Introduction

This section describes how Irisys IRC3xxx series devices handle count logs internally and how you can interact with them using the API. A count log is a record of the count values for each configured register at the time the log was written to the device's non-volatile memory.

5.2 Logging Period

The frequency with which count logs are created is determined by the `CountLogPeriod` setting which specifies, in seconds, the interval between each count log. The interval is based on the number of seconds since the start of the day to provide predictable logging times, hence the default logging interval of 900 seconds (15 minutes) will create count logs at 0000, 0015, 0030, etc.

You can use the `Irisys.BlackfinAPI.Blackfin.GetCountLogPeriod()` API function and the associated accessor function `Irisys.BlackfinAPI.Blackfin.CountLogPeriod()` to read the current value of this setting from the device. The `Irisys.BlackfinAPI.Blackfin.SetCountLogPeriod()` API function or the People Counter Setup Tool can be used to change the value of this setting. Any changes to this setting will take effect immediately without any further action required.

5.3 Accessing Count Logs

There are two functions in the API for retrieving count logs from a device, `Irisys.BlackfinAPI.Blackfin.GetCounts()` and `Irisys.BlackfinAPI.Blackfin.GetLastNCounts()`. The first of these allows you to retrieve all count logs which fall into a specified time period, whilst the second retrieves a specified number of the most recent log entries.

Regardless of which function you call, the retrieved count log entries are available via the `Irisys.BlackfinAPI.Blackfin.Counts` property, which returns a `List` containing an `Irisys.BlackfinAPI.Interfaces.Count` instance for each count log entry downloaded from the device. This class in turn contains a UTC timestamp representing the time the log entry was recorded and a `List` of unsigned integers (32bits) which contain the count value for each configured counter. The size of this list depends on the number of count registers configured at the time the log entry was recorded and may not be consistent between all of the log entries returned if the device configuration was changed during the log period retrieved.

Note that subsequent calls to any of the `Irisys.BlackfinAPI.Blackfin.GetCounts()`, `Irisys.BlackfinAPI.Blackfin.GetLastNCounts()` or `Irisys.BlackfinAPI.Blackfin.GetCurrentCount()` functions will overwrite the downloaded count data with the result of that function call.

5.4 Storage

Count logs are written to the device's non-volatile flash memory and therefore are not lost even if the power to the device is removed. The number of count log entries which can be stored on the device depends on the number of count registers which have been configured.

Due to the nature of the flash memory used on the device the only way to erase data is to erase the entire sector. Therefore the count logs are split between two sectors, with the oldest sector being erased when the newer sector

is full. Assuming you do not reset the count logs at any point this means that the minimum number of stored count logs (after both flash sectors have been filled at least once) is half of the maximum theoretical capacity of the device and the number of counts available via the API will be any amount between the minimum and maximum.

The maximum and minimum storage capacities for a range of configured registers are given in the table below, along with the minimum number of days worth of logging this represents at 5, 15 and 60 minute logging intervals.

Registers	Minimum Capacity	Maximum Capacity	5 Minute Interval (Min)	15 Minute Interval (Min)	60 Minute Interval (Min)
2	4680	9361	16 Days	48 Days	195 Days
4	2978	5957	10 Days	31 Days	124 Days
8	1724	3448	5 Days	17 Days	71 Days
16	936	1872	3 Days	9 Days	39 Days
32	489	978	1 Day	5 Days	20 Days

The default logging period of 15 minutes with 2 enabled count registers will have a minimum capacity of 48 days worth of count log entries.

5.5 Resetting Count Logs

You can use the API to reset the count log on an IRC3xxx series device by calling the `Irisys.BlackfinAPI.Blackfin.ResetCountLogs()` function, which will permanently erase all count log entries from the device. Note that flash memory has a finite lifespan and excessive use of this function could reduce the lifespan of your devices. Rather than resetting the count logs your application should keep track of the most recent count log entry retrieved from the device and use this as the starting point when requesting new data.

6 Frequently Asked Questions

What is the baud rate (speed) of an IP connection?

An Irisys IRC3xxx device can have a link speed of 10Mbps or 100Mbps, however the maximum data baud rate is limited to 1Mbps

What is the baud rate (speed) of a serial connection?

The baud rate of a serial connection to an Irisys IRC3xxx device is 115200bps

Which TCP/IP ports are used to communicate with an Irisys IRC3xxx device?

Port 4505 is used to establish direct IP connections to the device.

Port 80 is used to connect to the web interface on the device and configure it using the built in People Counter Setup Tool (PCST) software.

By default port 5000 is used by devices for outgoing connections when client connection mode is enabled, however this can be configured via the web interface.

What is the default IP address of a device?

The factory default IP address is 192.168.0.10, using a subnet mask of 255.255.255.0

What is the maximum value of a count line?

The count registers on the device are 32 bits, giving a maximum value of over 4.2 billion, which is reset to 0 each time the device is powered off and on again.

What is the lifespan of the flash memory storage on the device?

The flash memory used on the device has a rated minimum lifespan of 100,000 erase / write cycles, which equates to roughly 26,000 years of logging at a 15 minute interval. Resetting the count log files will cause additional erase / write cycles to occur and will reduce the useful life of the device, therefore you should avoid doing this unless absolutely necessary. For more information about the count log files on the device see the [Count Logs](#) section.

7 Class Documentation

7.1 Irisys.BlackfinAPI.Blackfin Class Reference

Public Member Functions

- [Blackfin](#) ()
- bool [ResetCountLogs](#) ()
- bool [ResetCurrentCount](#) ()
- bool [ResetDeviceStatus](#) ()

Setters

- bool [SetPacketTimeout](#) (int milliseconds)
- bool [SetCommsTimeout](#) (int milliseconds)
- bool [SetUnitTimeZone](#) ([IrisysTimeZone](#) timezone)
- bool [SetUnitTime](#) (DateTime tm)
- bool [SetTime](#) (DateTime t)
- bool [SetDeviceName](#) (string deviceName)
- bool [SetDeviceID](#) (string deviceID)
- bool [SetSiteName](#) (string sitename)
- bool [SetSiteID](#) (string siteid)
- bool [SetLocaleString](#) (string locale)
- bool [SetUserString](#) (string user)
- bool [SetDNS1](#) (IPAddress dns1)
- bool [SetDNS2](#) (IPAddress dns2)
- bool [SetDNS3](#) (IPAddress dns3)
- bool [SetClientConfigIP](#) (IPAddress ipAddress)
- bool [SetClientConfigHostname](#) (string hostname)
- bool [SetClientConfigPort](#) (ushort portNumber)
- bool [SetClientConfigPortNumber](#) (ushort portNumber)
- bool [SetClientConfigTimeout](#) (uint timeout)
- bool [SetClientConfigEnable](#) (bool enable)
- bool [SetCountLogPeriod](#) (int countLogPeriod)

Public Attributes

- const int [MAX_PROPERTY_STRING](#) = 64
- const int [MAX_DEVICEID_STRING](#) = 160

Properties

- int [SerialNumber](#) [get, set]
- int [CountLogPeriod](#) [get, set]
- string [MACAddress](#) [get, set]
- IPAddress [ClientConfigIP](#) [get, set]
- string [ClientConfigHostname](#) [get, set]
- ushort [ClientConfigPort](#) [get, set]
- ushort [ClientConfigPortNumber](#) [get]
- uint [ClientConfigTimeout](#) [get, set]

- bool [ClientConfigEnable](#) [get, set]
- List< [Count](#) > [Counts](#) [get, set]
- [DeviceStatus](#) [DeviceStatus](#) [get, set]
- DateTime [UnitTime](#) [get, set]
- string [LocaleString](#) [get, set]
- string [UserString](#) [get, set]
- List< string > [RegisterLabels](#) [get, set]
- string [DeviceName](#) [get, set]
- string [DeviceID](#) [get, set]
- string [SiteName](#) [get, set]
- string [SiteID](#) [get, set]
- bool [DHCPEnabled](#) [get, set]
- IPAddress [IPAddress](#) [get, set]
- IPAddress [SubnetMask](#) [get, set]
- IPAddress [Gateway](#) [get, set]
- string [IPFirmwareVersion](#) [get, set]
- string [MonitorFirmwareVersion](#) [get, set]
- IPAddress [DNS1](#) [get, set]
- IPAddress [DNS2](#) [get, set]
- IPAddress [DNS3](#) [get, set]
- UInt64 [UpTime](#) [get, set]
- [IrisysTimeZone](#) [UnitTimeZone](#) [get, set]
- String [NetworkChecksum](#) [get, set]
- int [PacketTimeout](#) [get]

Getters

- bool [GetCurrentCount](#) ()
- bool [GetLastNCounts](#) (uint n)
- bool [GetCounts](#) (DateTime start, DateTime end)
- bool [GetDeviceStatus](#) ()
- bool [GetTime](#) ()
- bool [GetUnitTime](#) ()
- bool [GetUptime](#) ()
- bool [GetUpTime](#) ()
- bool [GetUnitTimeZone](#) ()
- bool [GetMACAddress](#) ()
- bool [GetClientConfigHostname](#) ()
- bool [GetClientConfigIP](#) ()
- bool [GetDeviceName](#) ()
- bool [GetDeviceID](#) ()
- bool [GetSiteName](#) ()
- bool [GetSiteID](#) ()
- bool [GetLocaleString](#) ()
- bool [GetUserString](#) ()
- bool [GetRegisterLabels](#) ()
- bool [GetSerialNumber](#) ()
- bool [GetIPFirmwareVersion](#) ()

- bool [GetMonitorFirmwareVersion](#) ()
- bool [GetDNS1](#) ()
- bool [GetDNS2](#) ()
- bool [GetDNS3](#) ()
- bool [GetClientConfigPort](#) ()
- bool [GetClientConfigPortNumber](#) ()
- bool [GetClientConfigTimeout](#) ()
- bool [GetClientConfigEnable](#) ()
- bool [GetDHCPEnabled](#) ()
- bool [GetIPAddress](#) ()
- bool [GetSubnetMask](#) ()
- bool [GetGateway](#) ()
- bool [GetCountLogPeriod](#) ()
- bool [GetNetworkChecksum](#) ()

7.1.1 Detailed Description

Class used for accessing data and settings from an Irisys IP device

A [Blackfin](#) instance must be added to a [BlackfinEngine](#) along with a socket or serial port in order to retrieve or set data on a remote device.

Note that when connected to a non-master device via serial many of the comms function calls will fail. This is because some of them require the require the IP board (master only) to be present. This includes all functions with the ClientConfig prefix.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `Irisys.BlackfinAPI.Blackfin.Blackfin ()`

Creates a new blank [Blackfin](#) object, not associated with any connection.

7.1.3 Member Function Documentation

7.1.3.1 `bool Irisys.BlackfinAPI.Blackfin.GetCurrentCount ()`

Get the live count line values from the device. These values are not written to the count log and therefore may not be the same as the most recent entry returned by the `GetCounts()` function. Upon successful completion the count line values will be available via the `Counts` variable.

See also

[Counts](#)

Returns

True on success, false on comms error or timeout

7.1.3.2 bool Irisys.BlackfinAPI.Blackfin.GetLastNCounts (uint *n*)

Get the latest N count log entries from the device. Upon successful completion the count log entries will be available via the Counts variable.

Note

The number of count log entries retrieved may be less than N if there are less than N entries in the device count log

Parameters

<i>n</i>	Number of count log entries to retrieve
----------	---

See also

[Counts](#)

Returns

True on success, false on comms error or timeout

7.1.3.3 bool Irisys.BlackfinAPI.Blackfin.GetCounts (DateTime *start*, DateTime *end*)

Gets the count log entries from the device which fall within the specified time range. Upon successful completion the count log entries will be available via the Counts variable.

Parameters

<i>start</i>	UTC time of the earliest count log entry to retrieve
<i>end</i>	UTC time of the latest count log entry to retrieve

See also

[Counts](#)

Returns

True on success, False on comms error or timeout

7.1.3.4 bool Irisys.BlackfinAPI.Blackfin.GetDeviceStatus ()

Get the device status logs since the last boot up or device log reset. Upon successful completion the device status logs will be available via the DeviceStatus variable.

Note

This call may take longer than the configured packet timeout

See also

[DeviceStatus](#)

Returns

True on success, False on comms error or timeout

7.1.3.5 bool Irisys.BlackfinAPI.Blackfin.GetTime ()**7.1.3.6 bool Irisys.BlackfinAPI.Blackfin.GetUnitTime ()**

Get the current UTC time of the devices internal clock, which is not affected by the configured timezone. Upon successful completion the devices UTC time will be available via the UnitTime variable.

See also

[UnitTime](#)

Returns

True on success, False on comms error or timeout

7.1.3.7 bool Irisys.BlackfinAPI.Blackfin.GetUptime ()**7.1.3.8 bool Irisys.BlackfinAPI.Blackfin.GetUpTime ()**

Get the elapsed time, in seconds, since the device was booted. Upon successful completion the uptime will be available via the UpTime variable.

See also

[UpTime](#)

Returns

True on success, False on comms error or timeout

7.1.3.9 bool Irisys.BlackfinAPI.Blackfin.GetUnitTimeZone ()

Get the timezone configured on the device. Note that this has no functional impact on the device itself, which will always use UTC time. Upon successful completion the timezone configured on the device will be available via the UnitTimeZone variable.

Note

The timezone can be configured using the People Counter Setup Tool

[UnitTimeZone](#)

Returns

True on success, false if the devices firmware is too old or a timeout or other comms error occurred

7.1.3.10 bool Irisys.BlackfinAPI.Blackfin.GetMACAddress ()

Get the MAC address of the device, which must feature an IP board. Upon successful completion the devices MAC address will be available via the MACAddress variable.

See also

[MACAddress](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.11 bool Irisys.BlackfinAPI.Blackfin.GetClientConfigHostname ()

Get the client connection hostname configured on the device, which must feature an IP board. This setting specifies the hostname the device will attempt to connect to when client connection mode is enabled. Upon successful completion the client connection hostname configured on the device will be available via the ClientConfigHostname variable.

Note

If both an IP address and hostname are specified for client connection mode the IP address is used

See also

[ClientConfigHostname](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.12 bool Irisys.BlackfinAPI.Blackfin.GetClientConfigIP ()

Get the client connection IP address configured on the device, which must feature an IP board. This setting specifies the IP address the device will attempt to connect to when client connection mode is enabled. Upon successful completion the client connection IP address configured on the device will be available via the ClientConfigIP variable.

Note

If both an IP address and hostname are specified for client connection mode the IP address is used

See also

[ClientConfigIP](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.13 bool Irisys.BlackfinAPI.Blackfin.GetDeviceName ()

Get the device name string from the device. Upon successful completion the device name will be available via the DeviceName variable.

See also

[DeviceName](#)

Returns

True on success, False on comms error or timeout

7.1.3.14 bool Irisys.BlackfinAPI.Blackfin.GetDeviceID ()

Get the device ID string from the device. Upon successful completion the device ID will be available via the DeviceID variable.

See also

[DeviceID](#)

Returns

True on success, False on comms error or timeout

7.1.3.15 bool Irisys.BlackfinAPI.Blackfin.GetSiteName ()

Get the site name string from the device. Upon successful completion the site name will be available via the SiteName variable.

See also

[SiteName](#)

Returns

True on success, False on comms error or timeout

7.1.3.16 bool Irisys.BlackfinAPI.Blackfin.GetSiteID ()

Get the site ID string from the device. Upon successful completion the site ID name will be available via the SiteID variable.

See also

[SiteID](#)

Returns

True on success, False on comms error or timeout

7.1.3.17 bool Irisys.BlackfinAPI.Blackfin.GetLocaleString ()

Get the locale string from the device. Upon successful completion the locale string will be available via the LocaleString variable.

Note

This value is not linked to the unit timezone and has no functional impact on the device

See also

[LocaleString](#)

Returns

True on success, False on comms error or timeout

7.1.3.18 bool Irisys.BlackfinAPI.Blackfin.GetUserString ()

Get the user string from the device. Upon successful completion the user string will be available via the UserString variable.

See also

[UserString](#)

Returns

True on success, false if the devices firmware is too old or a timeout or other comms error occurred

7.1.3.19 bool Irisys.BlackfinAPI.Blackfin.GetRegisterLabels ()

Retrieve the labels for the enabled count registers from the device. Upon successful completion the retrieved labels will be available via the RegisterLabels variable This command was introduced in the 3.0.x series of APIs

Minimum DSP Firmware Version: 454

See also

[RegisterLabels](#)

Returns

True on success, false if the devices firmware is too old or a comms failure occurs

7.1.3.20 bool Irisys.BlackfinAPI.Blackfin.GetSerialNumber ()

Get the unique serial number from the device. Upon successful completion the serial number will be available via the SerialNumber variable.

See also

[SerialNumber](#)

Returns

True on success, False on comms error or timeout

7.1.3.21 bool Irisys.BlackfinAPI.Blackfin.GetIPFirmwareVersion ()

Get the version of IP firmware installed on the device, which must feature an IP board. Upon successful completion the IP firmware version of the device will be available via the IPFirmwareVersion variable.

See also

[IPFirmwareVersion](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.22 bool Irisys.BlackfinAPI.Blackfin.GetMonitorFirmwareVersion ()

Get the version of firmware installed on the device. Upon successful completion the firmware version of the device will be available via the MonitorFirmwareVersion variable.

See also

[MonitorFirmwareVersion](#)

Returns

True on success, False on comms error or timeout

7.1.3.23 bool Irisys.BlackfinAPI.Blackfin.GetDNS1 ()

Get the primary DNS server for the device, which must feature an IP board. Upon successful completion the primary DNS server for the device will be available via the DNS1 variable.

See also

[DNS1](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.24 bool Irisys.BlackfinAPI.Blackfin.GetDNS2 ()

Get the secondary DNS server for the device, which must feature an IP board. Upon successful completion the secondary DNS server for the device will be available via the DNS2 variable.

See also

[DNS2](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.25 bool `Irisys.BlackfinAPI.Blackfin.GetDNS3` ()

Get the tertiary DNS server for the device, which must feature an IP board. Upon successful completion the tertiary DNS server for the device will be available via the DNS3 variable.

See also

[DNS3](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.26 bool `Irisys.BlackfinAPI.Blackfin.GetClientConfigPort` ()

Get the client connection port configured on the device, which must feature an IP board. This setting specifies the port the device will attempt to connect to when client connection mode is enabled. Upon successful completion the client connection port configured on the device will be available via the ClientConfigPort variable.

See also

[ClientConfigPort](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.27 bool `Irisys.BlackfinAPI.Blackfin.GetClientConfigPortNumber` ()

7.1.3.28 bool `Irisys.BlackfinAPI.Blackfin.GetClientConfigTimeout` ()

Get the client connection timeout configured on the device, which must feature an IP board. This setting specifies the timeout, in seconds, between client connection attempts by the device when client connection mode is enabled. Upon successful completion the client connection timeout configured on the device will be available via the ClientConfigTimeout variable.

See also

[ClientConfigTimeout](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.29 bool `Irisys.BlackfinAPI.Blackfin.GetClientConfigEnable` ()

Get the client connection state configured on the device, which must feature an IP board. This setting specifies whether the device will attempt to establish an outgoing connection to the IP address or hostname configured on the device. Upon successful completion the client connection state configured on the device will be available via the ClientConfigEnable variable.

See also

[ClientConfigEnable](#)

Returns

True on success, False on comms error or timeout

7.1.3.30 bool Irisys.BlackfinAPI.Blackfin.GetDHCPEnabled ()

Get the DHCP state of the device, which must feature an IP board. This setting determines if the device obtains an IP address from a DHCP server or uses a statically configured address. Upon successful completion the DHCP state of the device will be available via the DHCPEnabled variable.

See also

[DHCPEnabled](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.31 bool Irisys.BlackfinAPI.Blackfin.GetIPAddress ()

Get the statically configured IP address of the device, which must feature an IP board. This setting determines the IP address of a device when it is not using DHCP. Upon successful completion the IP address of the device will be available via the IPAddress variable.

See also

[IPAddress](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.32 bool Irisys.BlackfinAPI.Blackfin.GetSubnetMask ()

Get the statically configured subnet mask of the device, which must feature an IP board. This setting determines the subnet mask of a device when it is not using DHCP. Upon successful completion the subnet mask of the device will be available via the SubnetMask variable.

See also

[SubnetMask](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.33 bool `Irisys.BlackfinAPI.Blackfin.GetGateway ()`

Get the statically configured gateway of the device, which must feature an IP board. This setting determines the gateway of a device when it is not using DHCP. Upon successful completion the gateway of the device will be available via the Gateway variable.

See also

[Gateway](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.34 bool `Irisys.BlackfinAPI.Blackfin.GetCountLogPeriod ()`

Get the count log period currently configured on the device, which specifies the interval, in seconds, between count log entries being written to the devices memory. Upon successful completion the configured count log period will be available via the CountLogPeriod variable.

See also

[CountLogPeriod](#)

Returns

True on success, False on comms error or timeout

7.1.3.35 bool `Irisys.BlackfinAPI.Blackfin.GetNetworkChecksum ()`

Generates a checksum of the current configuration settings of all devices on the CAN network of the connected device, which can be used to detect whether any configuration changes have taken place on the network between two subsequent calls to this function. Upon successful completion the network checksum will be available via the [NetworkChecksum\(\)](#) accessor function.

Note

This checksum may also be affected by firmware upgrades to any device on the CAN network

This command was introduced in the 3.0.x series of APIs

See also

[NetworkChecksum\(\)](#)

Returns

True on success, false if a timeout or other comms error occurred

7.1.3.36 bool Irisys.BlackfinAPI.Blackfin.ResetCountLogs ()

Empty the count log stored on the device. This action cannot be undone and will not affect the live count line values.

Returns

True on success, False on comms error or timeout

7.1.3.37 bool Irisys.BlackfinAPI.Blackfin.ResetCurrentCount ()

Reset the live count line values on the device. This action cannot be undone and does not affect existing count log entries stored on the device.

Returns

True on success, False on comms error or timeout

7.1.3.38 bool Irisys.BlackfinAPI.Blackfin.ResetDeviceStatus ()

Clear all entries in the device status logs. This action cannot be undone.

Returns

True on success, False on comms error or timeout

7.1.3.39 bool Irisys.BlackfinAPI.Blackfin.SetPacketTimeout (int *milliseconds*)

Set the packet timeout value for the API to use when communicating with the connected device, which specifies how long it will wait for a response before returning a failure. Longer values can help reduce failures on high latency connections but will also cause Get* and Set* function calls to block for longer whilst they wait for a response when the connection has been lost.

See also

[PacketTimeout](#)

Parameters

<i>milliseconds</i>	Timeout value, in milliseconds, to use during communications with the device
---------------------	--

7.1.3.40 bool Irisys.BlackfinAPI.Blackfin.SetCommsTimeout (int *milliseconds*)**7.1.3.41 bool Irisys.BlackfinAPI.Blackfin.SetUnitTimeZone (IrisysTimeZone *timezone*)**

Set the time zone on the device. Note that this has no functional impact on the device itself, which will always use UTC time internally.

Note

The time zone can also be configured using the People Counter Setup Tool

Parameters

<i>timezone</i>	Irisys time zone to set on the device
-----------------	---------------------------------------

Returns

True on success, False on comms error or timeout

7.1.3.42 bool Irisys.BlackfinAPI.Blackfin.SetUnitTime (DateTime tm)

Sets the internal clock on the device to the specified time, which should be in UTC

Parameters

<i>tm</i>	The UTC time to set on the device
-----------	-----------------------------------

Returns

True on success, False on comms error or timeout

7.1.3.43 bool Irisys.BlackfinAPI.Blackfin.SetTime (DateTime t)

7.1.3.44 bool Irisys.BlackfinAPI.Blackfin.SetDeviceName (string deviceName)

Set the device name string on the device to the specified value

Parameters

<i>deviceName</i>	Device name string to set on the device, up to 63 characters long
-------------------	---

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.45 bool Irisys.BlackfinAPI.Blackfin.SetDeviceID (string deviceID)

Set the device ID string on the device to the specified value

Parameters

<i>deviceID</i>	Device ID string to set on the device, up to 159 characters long
-----------------	--

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.46 bool Irisys.BlackfinAPI.Blackfin.SetSiteName (string sitename)

Set the site name string on the device to the specified value

Parameters

<i>sitename</i>	Site name string to set on the device, up to 63 characters long
-----------------	---

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.47 bool Irisys.BlackfinAPI.Blackfin.SetSiteID (string siteid)

Set the site ID string on the device to the specified value

Parameters

<i>siteid</i>	Site ID string to set on the device, up to 63 characters long
---------------	---

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.48 bool Irisys.BlackfinAPI.Blackfin.SetLocaleString (string locale)

Set the locale string on the device to the specified value

Parameters

<i>locale</i>	Locale string to set on the device, up to 63 characters long
---------------	--

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.49 bool Irisys.BlackfinAPI.Blackfin.SetUserString (string user)

Set the user string on the device to the specified value

Parameters

<i>user</i>	User string to set on the device, up to 63 characters long
-------------	--

Returns

True on success, False on comms error or timeout

Exceptions

<i>ArgumentException</i>	Thrown if the specified string exceeds the maximum length
--------------------------	---

7.1.3.50 bool Irisys.BlackfinAPI.Blackfin.SetDNS1 (IPAddress dns1)

Set the primary DNS server for the device to use when resolving hostnames, which must feature an IP board. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Parameters

<i>dns1</i>	Primary DNS server IP address to set on this device
-------------	---

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.51 bool Irisys.BlackfinAPI.Blackfin.SetDNS2 (IPAddress dns2)

Set the secondary DNS server for the device to use when resolving hostnames, which must feature an IP board. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Parameters

<i>dns2</i>	Secondary DNS server IP address to set on this device
-------------	---

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.52 bool Irisys.BlackfinAPI.Blackfin.SetDNS3 (IPAddress dns3)

Set the tertiary DNS server for the device to use when resolving hostnames, which must feature an IP board. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Parameters

<i>dns3</i>	Tertiary DNS server IP address to set on this device
-------------	--

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.53 bool Irisys.BlackfinAPI.Blackfin.SetClientConfigIP (IPAddress *ipAddress*)

Set the client connection IP address on the device, which must feature an IP board. This setting specifies the IP address the device will attempt to connect to when client connection mode is enabled. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Note

Set this value to 0.0.0.0 if you want to specify a hostname to connect to instead

Parameters

<i>ipAddress</i>	The client connection IP address to set on the device
------------------	---

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.54 bool Irisys.BlackfinAPI.Blackfin.SetClientConfigHostname (string *hostname*)

Set the client connection hostname on the device, which must feature an IP board. This setting specifies the hostname the device will attempt to connect to when client connection mode is enabled. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Note

If a client connection IP address other than 0.0.0.0 is specified this setting has no effect

Parameters

<i>hostname</i>	The client connection hostname to set on the device
-----------------	---

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.55 bool Irisys.BlackfinAPI.Blackfin.SetClientConfigPort (ushort *portNumber*)

Set the client connection port number on the device, which must feature an IP board. This setting specifies the port the device will attempt to connect to when client connection mode is enabled. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Parameters

<i>portNumber</i>	The client connection port number to set on the device
-------------------	--

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.56 bool `Irisys.BlackfinAPI.Blackfin.SetClientConfigPortNumber (ushort portNumber)`

7.1.3.57 bool `Irisys.BlackfinAPI.Blackfin.SetClientConfigTimeout (uint timeout)`

Set the client connection timeout on the device, which must feature an IP board. This setting specifies the timeout, in seconds, between client connection attempts by the device when client connection mode is enabled. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

[Setting IP board configuration](#)

Parameters

<i>timeout</i>	The client connection timeout to set on the device, in seconds
----------------	--

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.58 bool `Irisys.BlackfinAPI.Blackfin.SetClientConfigEnable (bool enable)`

Enable or disable client connection mode on the device, which must feature an IP board. If this setting is enabled the device will regularly attempt to establish an outgoing client connection to the IP address or hostname specified in the client connection settings. Please refer to the notes on [Setting IP board configuration](#) before using this function with a TCP/IP connection.

Parameters

<i>enable</i>	Client connection enabled state to set on the device
---------------	--

See also

[SetClientConfigIP](#), [SetClientConfigHostname](#), [SetClientConfigPort](#), [SetClientConfigTimeout](#)

Returns

True on success, false if the device does not have an IP board or a timeout or other comms error occurred

7.1.3.59 bool `Irisys.BlackfinAPI.Blackfin.SetCountLogPeriod (int countLogPeriod)`

Set the count log period on the device which specifies the interval, in seconds, between count log entries being written to the devices memory.

Parameters

<i>countLogPeriod</i>	The count log period to set on the device, in seconds
-----------------------	---

Returns

True on success, False on comms error or timeout

7.1.4 Member Data Documentation

7.1.4.1 `const int Irisys.BlackfinAPI.Blackfin.MAX_PROPERTY_STRING = 64`

7.1.4.2 `const int Irisys.BlackfinAPI.Blackfin.MAX_DEVICEID_STRING = 160`

7.1.5 Property Documentation

7.1.5.1 `int Irisys.BlackfinAPI.Blackfin.SerialNumber [get, set]`

The unique serial number of the Irisys IP device.

[GetSerialNumber\(\)](#)

7.1.5.2 `int Irisys.BlackfinAPI.Blackfin.CountLogPeriod [get, set]`

The period at which count values are logged to the flash memory in the device.

[GetCountLogPeriod\(\)](#)

7.1.5.3 `string Irisys.BlackfinAPI.Blackfin.MACAddress [get, set]`

The unique MAC address of the IP device.

[GetMACAddress\(\)](#)

7.1.5.4 `IPAddress Irisys.BlackfinAPI.Blackfin.ClientConfigIP [get, set]`

The IP address the IP counter will attempt to connect to if ClientConfigEnable is set to true.

[GetClientConfigIP\(\)](#)

7.1.5.5 `string Irisys.BlackfinAPI.Blackfin.ClientConfigHostname [get, set]`

The hostname the IP counter will attempt to connect to if ClientConfigEnable is set to true.

[GetClientConfigHostname\(\)](#)

7.1.5.6 `ushort Irisys.BlackfinAPI.Blackfin.ClientConfigPort [get, set]`

The port number IP counter will attempt to connect to if ClientConfigEnable is set to true.

[GetClientConfigPortNumber\(\)](#)

7.1.5.7 ushort Irisys.BlackfinAPI.Blackfin.ClientConfigPortNumber [get]

7.1.5.8 uint Irisys.BlackfinAPI.Blackfin.ClientConfigTimeout [get, set]

The time in seconds between reconnection attempts if ClientConfigEnable is set to true.

[GetClientConfigTimeout\(\)](#)

7.1.5.9 bool Irisys.BlackfinAPI.Blackfin.ClientConfigEnable [get, set]

Whether or not the counter will attempt to connect out to a server.

[GetClientConfigEnable\(\)](#)

7.1.5.10 List<Count> Irisys.BlackfinAPI.Blackfin.Counts [get, set]

The Count Log containing historic count information.

[GetLastNCounts\(\)](#) [GetCounts\(\)](#) [GetCurrentCounts\(\)](#)

7.1.5.11 DeviceStatus Irisys.BlackfinAPI.Blackfin.DeviceStatus [get, set]

The device diagnostic log.

[GetDeviceStatus\(\)](#)

7.1.5.12 DateTime Irisys.BlackfinAPI.Blackfin.UnitTime [get, set]

The time on the device (UTC) as of the time when [GetUnitTime\(\)](#) called.

[GetUnitTime\(\)](#)

7.1.5.13 string Irisys.BlackfinAPI.Blackfin.LocaleString [get, set]

A freeform user string.

[GetLocaleString\(\)](#)

7.1.5.14 string Irisys.BlackfinAPI.Blackfin.UserString [get, set]

A freeform user string.

[GetUserString\(\)](#)

7.1.5.15 List<string> Irisys.BlackfinAPI.Blackfin.RegisterLabels [get, set]

An ordered list of labels assigned to the enabled count registers on the device.

[GetRegisterLabels\(\)](#)

7.1.5.16 string Irisys.BlackfinAPI.Blackfin.DeviceName [get, set]

A freeform user string.

[GetDeviceName\(\)](#)

7.1.5.17 string Irisys.BlackfinAPI.Blackfin.DeviceID [get, set]

A freeform user string.

[GetDeviceID\(\)](#)

7.1.5.18 string Irisys.BlackfinAPI.Blackfin.SiteName [get, set]

A freeform user string.

[GetSiteName\(\)](#)

7.1.5.19 string Irisys.BlackfinAPI.Blackfin.SiteID [get, set]

A freeform user string.

[GetSiteID\(\)](#)

7.1.5.20 bool Irisys.BlackfinAPI.Blackfin.DHCPEnabled [get, set]

Whether or not the device uses DHCP to obtain an IP address.

[GetDHCPEnabled\(\)](#)

7.1.5.21 IPAddress Irisys.BlackfinAPI.Blackfin.IPAddress [get, set]

The static IP address for the counter to use if DHCPEnabled is false.

[GetIPAddress\(\)](#)

7.1.5.22 IPAddress Irisys.BlackfinAPI.Blackfin.SubnetMask [get, set]

[GetSubnetMask\(\)](#)

7.1.5.23 IPAddress Irisys.BlackfinAPI.Blackfin.Gateway [get, set]

[GetGateway\(\)](#)

7.1.5.24 string Irisys.BlackfinAPI.Blackfin.IPFirmwareVersion [get, set]

The firmware version for the IP board.

[GetIPFirmwareVersion\(\)](#)

7.1.5.25 string `Irisys.BlackfinAPI.Blackfin.MonitorFirmwareVersion` [get, set]

The firmware version for the DSP head.

[GetMonitorFirmwareVersion\(\)](#)

7.1.5.26 IPAddress `Irisys.BlackfinAPI.Blackfin.DNS1` [get, set]

The primary DNS server the device will use to resolve host names.

[GetDNS1\(\)](#)

7.1.5.27 IPAddress `Irisys.BlackfinAPI.Blackfin.DNS2` [get, set]

The secondary DNS server the device will use to resolve host names.

[GetDNS2\(\)](#)

7.1.5.28 IPAddress `Irisys.BlackfinAPI.Blackfin.DNS3` [get, set]

The tertiary DNS server the device will use to resolve host names.

[GetDNS3\(\)](#)

7.1.5.29 UInt64 `Irisys.BlackfinAPI.Blackfin.UpTime` [get, set]

The amount of seconds since last reset.

[GetUpTime\(\)](#)

7.1.5.30 IrisysTimeZone `Irisys.BlackfinAPI.Blackfin.UnitTimeZone` [get, set]

The time zone in which the device resides.

[GetUnitTimeZone\(\)](#)

7.1.5.31 String `Irisys.BlackfinAPI.Blackfin.NetworkChecksum` [get, set]

The network settings checksum generated by the last successful call to [GetNetworkChecksum\(\)](#) This command was introduced in the 3.0.x series of APIs

[GetNetworkChecksum\(\)](#)

7.1.5.32 int `Irisys.BlackfinAPI.Blackfin.PacketTimeout` [get]

The maximum time to wait for a response from the device before flagging a comms call as timed out

7.2 Irisys.BlackfinAPI.BlackfinEngine Class Reference

Class for connecting [Irisys.BlackfinAPI.Blackfin](#) objects to a serial port or Socket.

Public Types

- enum [ConnectionErrorType](#) { [FATAL](#), [WARNING](#) }

Public Member Functions

- delegate void [ConnectionErrorHandler](#) ([Blackfin](#) sender, string errorString, Exception exp, [ConnectionErrorType](#) errorType)
- [BlackfinEngine](#) ()
- void [StartEngine](#) ()
- void [ShutdownEngine](#) ()
- bool [AddNewCounterEndPoint](#) ([Blackfin](#) counter, Socket socket)
- bool [AddNewCounterEndPoint](#) ([Blackfin](#) counter, Socket socket, [ConnectionErrorHandler](#) error)
- bool [AddNewCounterEndPoint](#) ([Blackfin](#) counter, System.IO.Ports.SerialPort serialPort, [ConnectionErrorHandler](#) error)
- bool [AddNewCounterEndPoint](#) ([Blackfin](#) counter, System.IO.Ports.SerialPort serialPort)
- object [RemoveCounterEndPoint](#) ([Blackfin](#) counter)
- void [ClockCounters](#) ()

Static Public Member Functions

- static string [GetAPIVersion](#) ()

7.2.1 Detailed Description

Class for connecting [Irisys.BlackfinAPI.Blackfin](#) objects to a serial port or Socket.

7.2.2 Member Enumeration Documentation

7.2.2.1 enum [Irisys::BlackfinAPI::BlackfinEngine::ConnectionErrorType](#)

Types of connection error possible.

Enumerator:

FATAL

WARNING

7.2.3 Constructor & Destructor Documentation

7.2.3.1 [Irisys.BlackfinAPI.BlackfinEngine.BlackfinEngine](#) ()

Instantiates a new [Blackfin](#) Engine ready to be initialized.

7.2.4 Member Function Documentation

7.2.4.1 `delegate void Irisys.BlackfinAPI.BlackfinEngine.ConnectionErrorHandler (Blackfin sender, string errorString, Exception exp, ConnectionErrorType errorType)`

Error handler interface for [Blackfin](#) API This allows a client to receive a call back when an error occurs.

Parameters

<i>sender</i>	The Blackfin connection on which the error occurred
<i>errorString</i>	A description of the error
<i>exp</i>	The exact exception thrown
<i>errorType</i>	Whether the error was fatal or a warning

7.2.4.2 `static string Irisys.BlackfinAPI.BlackfinEngine.GetAPIVersion () [static]`

Returns a string representation of the version of the executing API.

Returns

A string containing a version number

7.2.4.3 `void Irisys.BlackfinAPI.BlackfinEngine.StartEngine ()`

Clocks the [Blackfin](#) object, which also will clock all virtual blackfins.

7.2.4.4 `void Irisys.BlackfinAPI.BlackfinEngine.ShutdownEngine ()`

Shuts down the [Blackfin](#) Engine and releases resources This stops network activity for any connected [Blackfin](#) devices, but does not disconnect them. You should call `RemoveCounterEndPoint` for all [Blackfin](#) objects BEFORE calling this method.

7.2.4.5 `bool Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint (Blackfin counter, Socket socket)`

Connects a [Blackfin](#) object to a connected Socket object. This enabled the [Blackfin](#) to establish a connection and communicate with the remote device.

Returns

True on connection success, False on failure

7.2.4.6 `bool Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint (Blackfin counter, Socket socket, ConnectionErrorHandler error)`

Connects a [Blackfin](#) object to a connected Socket object. This enabled the [Blackfin](#) to establish a connection and communicate with the remote device The error interface will be informed of all connection errors, so you should check that the counter is the relevant object.

Parameters

<i>counter</i>	The Blackfin object to connect
<i>socket</i>	A connected socket to the device
<i>error</i>	An error callback interface.

Returns

True on connection success, False on failure

7.2.4.7 bool Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint ([Blackfin counter](#), System.IO.Ports.SerialPort *serialPort*, ConnectionErrorHandler *error*)

Connects a [Blackfin](#) object to a connected SerialPort object. This enabled the [Blackfin](#) to establish a connection and communicate with the remote device The error interface will be informed of all connection errors, so you should check that the counter is the relevant object.

Parameters

<i>counter</i>	
<i>serialPort</i>	
<i>error</i>	

Returns

True on connection success, False on failure

7.2.4.8 bool Irisys.BlackfinAPI.BlackfinEngine.AddNewCounterEndPoint ([Blackfin counter](#), System.IO.Ports.SerialPort *serialPort*)

Connects a [Blackfin](#) object to a connected Socket object. This enabled the [Blackfin](#) to establish a connection and communicate with the remote device.

Parameters

<i>counter</i>	The Blackfin object to connect
<i>serialPort</i>	The open serial port connected to the remote device

Returns

True on connection success, False on failure

7.2.4.9 object Irisys.BlackfinAPI.BlackfinEngine.RemoveCounterEndPoint ([Blackfin counter](#))

Disconnects the [Blackfin](#) object from the remote device. The underlying port will not be closed, this is the responsibility of the calling code.

Parameters

<i>counter</i>	The Blackfin object to disconnect
----------------	---

Returns

Returns the connection object used (of type Socket or SerialPort) or null if device was not connected

7.2.4.10 void Irisys.BlackfinAPI.BlackfinEngine.ClockCounters ()

This should be used if the user wants to clock the counters themselves without an extra thread.

7.3 Irisys.BlackfinAPI.Interfaces.Count Class Reference

A single count log entry.

Public Attributes

- List< uint > `countLines` = new List<uint>()
- DateTime `countTime`

7.3.1 Detailed Description

A single count log entry.

7.3.2 Member Data Documentation

7.3.2.1 List<uint> Irisys.BlackfinAPI.Interfaces.Count.countLines = new List<uint>()

A list of count line count values.

7.3.2.2 DateTime Irisys.BlackfinAPI.Interfaces.Count.countTime

A timestamp for this count log entry.

7.4 Irisys.BlackfinAPI.Interfaces.DeviceLogEntry Class Reference

A single device log entry.

Public Attributes

- string `errorDescription` = ""
- DateTime `timestamp`

7.4.1 Detailed Description

A single device log entry. This could be an error, warning or message

7.4.2 Member Data Documentation

7.4.2.1 `string Irisys.BlackfinAPI.Interfaces.DeviceLogEntry.errorDescription = ""`

A string describing the condition of the counter.

7.4.2.2 `DateTime Irisys.BlackfinAPI.Interfaces.DeviceLogEntry.timestamp`

A timestamp when the entry was logged.

7.5 `Irisys.BlackfinAPI.Interfaces.DeviceStatus` Class Reference

A class representing the status of the device.

Public Attributes

- `List< DeviceLogEntry > m_warnList = new List<DeviceLogEntry>()`
- `List< DeviceLogEntry > m_errorList = new List<DeviceLogEntry>()`
- `List< DeviceLogEntry > m_infoList = new List<DeviceLogEntry>()`

7.5.1 Detailed Description

A class representing the status of the device.

7.5.2 Member Data Documentation

7.5.2.1 `List<DeviceLogEntry> Irisys.BlackfinAPI.Interfaces.DeviceStatus.m_warnList = new List<DeviceLogEntry>()`

A list of all the warnings in the device diagnostic log.

7.5.2.2 `List<DeviceLogEntry> Irisys.BlackfinAPI.Interfaces.DeviceStatus.m_errorList = new List<DeviceLogEntry>()`

A list of all the errors in the device diagnostic log.

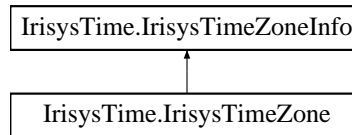
7.5.2.3 `List<DeviceLogEntry> Irisys.BlackfinAPI.Interfaces.DeviceStatus.m_infoList = new List<DeviceLogEntry>()`

A list of all of the information messages in the device diagnostic log.

7.6 `IrisysTime.IrisysTimeZone` Class Reference

A mutable class representing a time zone which can be set in the flash table.

Inheritance diagram for IrisysTime.IrisysTimeZone:



Public Member Functions

- [IrisysTimeZone](#) ([IrisysTimeZoneInfo](#) info)

Properties

- bool [ApplyDST](#) [get, set]

Events

- PropertyChangedEventHandler [PropertyChanged](#)

7.6.1 Detailed Description

A mutable class representing a time zone which can be set in the flash table.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 IrisysTime.IrisysTimeZone.IrisysTimeZone ([IrisysTimeZoneInfo](#) info)

Parameters

<i>info</i>

7.6.3 Property Documentation

7.6.3.1 bool IrisysTime.IrisysTimeZone.ApplyDST [get, set]

Flag showing whether or not Daylight Savings time should be applied. Note that this is a user setting and has no effect on the device.

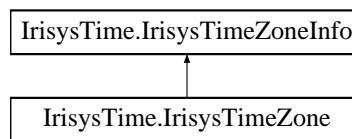
7.6.4 Event Documentation

7.6.4.1 PropertyChangedEventHandler IrisysTime.IrisysTimeZone.PropertyChanged

7.7 IrisysTime.IrisysTimeZoneInfo Class Reference

Immutable class representing an available TimeZone for a Blackfin device.

Inheritance diagram for IrisysTime.IrisysTimeZoneInfo:



Public Member Functions

- override bool [Equals](#) (object obj)
- override int [GetHashCode](#) ()
- override string [ToString](#) ()

Static Public Attributes

- static readonly [IrisysTimeZoneInfo UtcTimeZone](#) = timeZones[0]

Protected Member Functions

- [IrisysTimeZoneInfo](#) (TimeSpan t, bool b, string id)

Properties

- TimeSpan [BaseUtcOffset](#) [get, set]
- bool [SupportsDST](#) [get, set]
- string [TimeZoneID](#) [get, set]
- string [DisplayName](#) [get]
- static IEnumerable< [IrisysTimeZoneInfo](#) > [TimeZones](#) [get]

7.7.1 Detailed Description

Immutable class representing an available TimeZone for a Blackfin device.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 IrisysTime.IrisysTimeZoneInfo.IrisysTimeZoneInfo (TimeSpan t, bool b, string id) [protected]

Creates a new time zone with the specified parameters.

Parameters

<i>t</i>	The base UTC offset
<i>b</i>	Whether or not DST is supported
<i>id</i>	The Windows Time zone ID

7.7.3 Member Function Documentation

7.7.3.1 override bool IrisysTime.IrisysTimeZoneInfo.Equals (object *obj*)

Compares timezones by the TimeZoneID.

Parameters

<i>obj</i>

Returns

7.7.3.2 override int IrisysTime.IrisysTimeZoneInfo.GetHashCode ()

Returns the TimeZoneID hashcode.

Returns

7.7.3.3 override string IrisysTime.IrisysTimeZoneInfo.ToString ()

Returns the three properties, semicolon delimited.

Returns

7.7.4 Member Data Documentation

7.7.4.1 readonly IrisysTimeZoneInfo IrisysTime.IrisysTimeZoneInfo.UtcTimeZone = timeZones[0]
[static]

The UTC Timezone.

7.7.5 Property Documentation

7.7.5.1 TimeSpan IrisysTime.IrisysTimeZoneInfo.BaseUtcOffset [get, set]

The base UTC offset for the timezone.

7.7.5.2 bool IrisysTime.IrisysTimeZoneInfo.SupportsDST [get, set]

Whether or not the timezone can use DST (NOT whether it is using it)

7.7.5.3 string IrisysTime.IrisysTimeZoneInfo.TimeZoneID [get, set]

The windows ID string of the timezone.

7.7.5.4 string IrisysTime.IrisysTimeZoneInfo.DisplayName [get]

A string representation of the timezone.

7.7.5.5 IEnumerable<IrisysTimeZoneInfo> IrisysTime.IrisysTimeZoneInfo.TimeZones [static, get]

All of the Irisys supported time zones.

7.8 Irisys.BlackfinAPI.SerialNumberConverter Class Reference

Utility class for conversions.

Static Public Member Functions

- static string [ConvertSerialNumber](#) (int serialNumber)

7.8.1 Detailed Description

Utility class for conversions.

7.8.2 Member Function Documentation

7.8.2.1 static string Irisys.BlackfinAPI.SerialNumberConverter.ConvertSerialNumber (int *serialNumber*) [static]

Converts a long serial number to an alpha numeric serial number, as displayed on the IRC3000 series hardware.

Parameters

<i>serialNumber</i>	A numeric serial number returned from the Blackfin.SerialNumber property
---------------------	--

Returns

The alpha numeric serial as a string