BIRD.DLL

This driver applies to the:

RS232: Extended Range Controller, Flock of Birds, laserBIRD, Hy-BIRD, miniBIRD, Nest of Birds RS232.

ISA: miniBIRD II ISA, MotionStar Wired. pcBIRD

Ethernet: 3D Navigator, MotionStar Wired and Wireless.

and BIRD.DLL Version 3.1.2

Using BIRD.DLL

This section describes how to use the BIRD.DLL API to perform the following operations:

Sample Code Index

System Initialization

The first operation that must be performed before the system can be used is initialization. This is performed by calling one of the following Wake Up functions appropriate to your tracker's interface.

BirdTCPIPWakeUp	MotionStar Wired and Wireless, 3D Navigator.
birdRS232WakeUp	Extended Range Controller, Flock of Birds, laserBIRD, Hy-BIRD, miniBIRD
birdISAWakeUp	pcBIRD, miniBIRD II ISA , RS232 MotionStar Wired.

System Setup

Before any system parameters can be changed, the system configuration must be read into a data structure type **BIRDSYSTEMCONFIG**. This is done using the **birdGetSystemConfig** command. Changes to the system are made to this structure and then sent back to the hardware using **birdSetSystemConfig**.

Device Setup

Before any device parameters can be changed, the device configuration must be read into a data structure type **BIRDDEVICECONFIG**. This is done using the **birdGetDeviceConfig** or **birdGetFastDeviceConfig** command. See the definitions for the differences between the two functions. The device setup involves selecting a data format, setting the filter and quality parameters, determining the sensor angle alignment and hemisphere of operation. All of these parameters have an associated default value. The parameter only needs to be changed if the default is inappropriate. In most cases the default filter parameters will be found to provide adequate performance for most applications. Unless the sensor is going to be attached to something that would cause it to be tilted while in its reference position, then the angle align parameters will not need to be changed. The hemisphere will need to be changed if the sensor is going to operate anywhere other than the forward hemisphere, which is the default. Typically the user will only have to set up the data format if something other than position/angles is required. At a minimum, nothing need be changed and the system will still operate successfully. Note: It is necessary to set or change the parameters for each of the sensors individually. This allows each sensor to have its parameters set to different values.

Standalone vs. Master/Slave

Some Ascension products can run in either standalone or master/slave mode. In standalone mode, there is only one device and it is located at bird address 0. The data for standalone mode devices is thus in the `birdframe[0]`. By default a standalone device initializes to a running state. In master/slave mode, there are one or more devices are present. It should be noted that the first sensor may or may not have a bird address of 1. In many configurations, the transmitter controller (ERC) is at address 1 with sensors starting at address 2. Devices not in standalone mode need to be set to the running state by command. Refer to the information returned from `birdGetSystemConfig` to determine the configuration of your bird devices.

BIRD.DLL API Reference

The following elements are used with the PCIBird system.

BIRD.DLL API Functions
BIRD.DLL API Structures

BIRD.DLL API Functions

The following functions are used with the BIRD.DLL driver

birdDisableMeasurementCycleReporting	**birdReadingReady**
birdDisplayErrorDialogs	**birdRS232ClearBuffer**
birdEnableMeasurementCycleReporting	**BirdRS232GetResponse**
birdFrameReady	**birdRS232GroupModeEnabled**
birdGetDeviceConfig	**birdRS232Reset**
birdGetDLLVersion	**birdRS232Resynch**
birdGetErrorCode	**birdRS232SendCommand**
birdGetErrorMessage	**birdRS232SetGroupMode**
birdGetFastDeviceConfig	**birdRS232WakeUp**
birdGetFrame	**birdSetDeviceConfig**
birdGetMostRecentFrame	**birdSetFastDeviceConfig**
birdGetReading	**birdSetSystemConfig**
birdGetSystemConfig	**birdStartFrameStream**
birdISAClearBuffer	**birdStartReading**
birdISAGetResponse	**birdStartSingleFrame**
birdISAReset	**birdStopFrameStream**
BirdISAResynch	**birdTCPIPClearBuffer**
birdISASendCommand	**birdTCPIPGetResponse**
birdISAWakeUp	**birdTCPIPSendCommand**
birdNewMeasurementCycle	**BirdTCPIPWakeUp**

birdDisableMeasurementCycleReporting

Disable measurement cycle reporting

```
BOOL birdDisableMeasurementCycleReporting();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Stops the birds from signaling at the start of each measurement cycle. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

TOP

birdDisplayErrorDialogs

Enabled or Disables the display of error dialogs

```
BOOL birdDisplayErrorDialogs(BOOL bEnable);
```

Parameters

BOOL	bEnable

Return Values

Void

Remarks

Enabled or Disables the display of error dialogs

TOP

birdEnableMeasurementCycleReporting

Enable measurement cycle reporting

```
BOOL birdEnableMeasurementCycleReporting();
```

Parameters

Int nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use `birdGetErrorMessage()` to view error.

Remarks

Causes the birds to signal every time they begin a new measurement cycle. The signal is detected using the routine `birdNewMeasurementCycle()`. "nGroupID" is the group identifier that was passed to `birdWakeUp()`.

NOTE: This routine cannot be called while streaming data. To check for new measurement cycles in stream mode, simply call `birdFrameReady()`.

NOTE: This routine cannot be called in RS232 mode if group mode is enabled. This is so because the master bird can only send the data-ready character if the birds are in non-group mode. To ensure that this is the case, first call `birdRS232SetGroupMode()` with an argument of GMS_GROUP_MODE_NEVER

TOP

birdFrameReady

Frame Ready?

```
BOOL birdFrameReady();
```

Parameters

Int nGroupID

Return Values

Value	Meaning
TRUE	Reading is Ready
FALSE	Reading is not ready

Remarks

Determines if a new frame of bird data is ready, during either single or streamed frame acquisition. "nGroupID" is the group identifier that was passed to `birdWakeUp()`.

SAMPLE CODE
TOP

birdGetDeviceConfig

Gets device configuration.

```
BOOL birdGetDeviceConfig();
```

Parameters

Int	nGroupID
Int	nDeviceNum
BIRDDEVICECONFIG	*pdevcfg
BOOL	bGetDriverCopy

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Gets the configuration of a single device. "nGroupID" is the group identifier that was passed to **birdWakeUp()**. "nDeviceNum" specifies the bird number (ignored for stand-alone mode). The device configuration is returned in "pdevcfg". See the definition of **BIRDDEVICECONFIG** for an explanation of this structure.

Related

For a quicker method of getting a device configuration, see **birdGetFastDeviceConfig**.

SAMPLE CODE
TOP

birdGetDLLVersion

Gets the current BIRD.DLL version.

```
BOOL GetDLLVersion();
```

Parameters

Return Values

Value	Meaning

Remarks

Gets the current BIRD.DLL version.

TOP

birdGetErrorCode

Get error code.

```
BOOL birdGetErrorCode();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
BE_NOERROR	No errors occurred.
<bird error>	See **BIRD ERROR TABLE**

Remarks

Gets the current error code, and resets it to BE_NOERROR.

TOP

birdGetErrorMessage

Get text of error message.

```
BOOL birdGetErrorMessage();
```

Parameters

None

Return Values

LPSTR to message text

Remarks

Gets the text of the error message

TOP

birdGetFastDeviceConfig

Gets device configuration.

```
BOOL birdGetFastDeviceConfig();
```

Parameters

Int	nGroupID
Int	nDeviceNum
BIRDDEVICECONFIG	*pdevcfg

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Faster (and less error-prone) version of **`birdGetDeviceConfig()`**. In RS232 and ISA modes, the following fields of BIRDDEVICECONFIG are not used by Setup

byReportRate
byHemisphere
wAlphaMin[]
wAlphaMax[]
wVM[]
anglesReferenceFrame
anglesAngleAlign

If you need to use one of the above parameters, use **`birdGetDeviceConfig`** instead.

SAMPLE CODE
TOP

birdGetFrame

Get frame of data

```
BOOL birdGetFrame();
```

Parameters

Int	nGroupID
BIRDFRAME	*pframe

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Gets the next frame of bird data, during either single or streamed frame acquisition.

"nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. Note that **`birdStartFrameStream`** or **`birdStartSingleFrame`** must be called first. The data is returned in "pframe". Seethe definition of BIRDFRAME for an explanation of this structure.

SAMPLE CODE
TOP

birdGetMostRecentFrame

Get most recent frame of data

```
BOOL birdGetMostRecentFrame();
```

Parameters

Int	nGroupID
BIRDFRAME	*pframe

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Gets the most recent frame of bird data during streamed frame acquisition. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. The data is returned in "pframe". See the definition of BIRDFRAME for an explanation of this structure.

SAMPLE CODE
TOP

birdGetReading

Sets device configuration.

```
BOOL birdGetReading();
```

Parameters

Int	nGroupID
Int	nDeviceNum
BIRDREADING	*preading

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Gets the bird reading that was started by **birdStartReading()**. "nGroupID" is the group identifier that was passed to birdWakeUp(). "nDeviceNum" specifies the bird number (ignored for stand-alone mode). The data is returned in "preading". See the definition of **BIRDREADING** for an explanation of this structure.

SAMPLE CODE
TOP

birdGetSystemConfig

Gets System configuration.

```
BOOL birdGetSystemConfig();
```

Parameters

Int	nGroupID
BIRDSYSTEMCONFIG	*psyscfg
BOOL	bGetDriverCopy

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Gets the system configuration of a group of birds. "nGroupID" is the group identifier that was passed to birdWakeUp(). The system configuration is returned in "psyscfg". See the definition of **BIRDSYSTEMCONFIG** for an explanation of this structure. When “bGetDriverCopy” is set to FALSE, this data is read from the hardware. For a faster response from this command, set “bGetDriverCopy” to TRUE - a copy of psyscfg that was put in memory when the BirdWakeUp command was called will be used.

The sysconfig structure takes the following format:

typedef struct tagBIRDSYSTEMCONFIG

{

BYTE	bySystemStatus	current system status (see **BIRD SYSTEM STATUS** bits)
BYTE	byError	error code flagged by server or master bird
BYTE	byNumDevices	number of devices in system
BYTE	byNumServers	number of servers in system
BYTE	byXmtrNum	transmitter number (see **BIRD TRANSMITTER NUMBER** bits)
WORD	wXtalSpeed	crystal speed in MHz
double	dMeasurementRate	measurement rate in frames per second
BYTE	byChassisNum	chassis number
BYTE	byNumChassisDevices	number of devices within this chassis
BYTE	byFirstDeviceNum	number of first device in this chassis
WORD	wSoftwareRev	software revision of server application or master bird
BYTE	byFlockStatus[ ]	status of all devices in flock, indexed by bird number

} **(see note in** BIRDFRAME **definition) – also see** BIRD FLOCK STATUS **bits**

SAMPLE CODE
TOP

BirdISAClearBuffer

Clear Buffer. ISA mode

```
BOOL birdISAClearBuffer();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Clears the ISA receiver buffer. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

TOP

BirdISAGetResponse

Get a response. ISA mode

```
BOOL birdISAGetResponse();
```

Parameters

Int	nGroupID
Int	nDeviceNum
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Gets a generic response in ISA mode. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. The response bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

BirdISAReset

Reset. ISA mode

```
BOOL birdISAReset();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Resets a group of birds. "nGroupID" is the group identifier that was passed to **birdWakeUp()**.

TOP

BirdISAResynch

Resynchronize. ISA mode

```
BOOL birdISAResynch();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Resynchronizes the ISA data stream during streamed frame acquisition. "nGroupID" is the group identifier that was passed to **birdWakeUp()**.

TOP

BirdISASendCommand

Send a command in ISA mode

```
BOOL birdISASendCommand();
```

Parameters

Int	nGroupID
Int	nDeviceNum
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Sends a generic command in RS232 mode. "nGroupID" is the group identifier that was passed to `birdWakeUp()`. The command bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

birdISAWakeUp

Wakes up a group of birds using the ISA interface.

```
BOOL birdISAWakeUp();
```

Parameters

BOOL	bStandAlone
Int	nNumDevices
WORD	*pwAddress
DWORD	dwReadTimeout
DWORD	dwWriteTimeout

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use `birdGetErrorMessage()` to view error.

Remarks

Wakes up a group of birds in ISA mode. "nGroupID" can be any number between 0 and BIRD_MAX_GROUP_ID that the user wants to associate with this group of birds. "bStandAlone" indicates if the group consists of a single bird operating in stand-alone mode. "nNumDevices" is the number of devices in the group, including any ERC's, and should be equal to the bird number of the last device. (This field is only relevant when bStandAlone = FALSE.) "pwAddress" points to an array of words, each of which is the memory-mapped address of one of the birds. If bStandAlone = TRUE, then the bird's address is passed in pwAddress[0]. Otherwise, the array is indexed by bird number - for example, pwAddress[2] would be the address of bird #2. Any birds which are external to the PC must be given an address of zero. "dwReadTimeout" is the maximum time, in msecs, that the application will take when trying to receive a character. "dwWriteTimeout" is the maximum time, in msecs, that the application will take when trying to transmit a character.

NOTE: The first bird in this group must be internal to the PC (not an ERC). Otherwise, use `birdRS232WakeUp()` instead of this routine.

NOTE: The memory pointed to by "pwAddress" may be released after the function returns.

SAMPLE CODE
TOP

birdNewMeasurementCycle

New Measurement Cycle

```
BOOL NewMeasurementCycle();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	If a new measurement cycle has begun

FALSE	If a new measurement cycle has not begun

Remarks

Determines if a new measurement cycle has been reported by the birds. Measurement cycle reporting must first be enabled by calling the routine `birdEnableMeasurementCycleReporting()`. "nGroupID" is the group identifier that was passed to `birdWakeUp()`.

TOP

birdReadingReady

Is reading ready?

```
BOOL birdReadingReady();
```

Parameters

Int nGroupID
Int nDeviceNum

Return Values

Value	Meaning
TRUE	Reading is Ready
FALSE	Reading is not Ready

Remarks

Determines if the bird reading that was started by `birdStartReading()` is ready. "nGroupID" is the group identifier that was passed to `birdWakeUp()`. "nDeviceNum" specifies the bird number (ignored for stand-alone mode).

TOP

BirdRS232ClearBuffer

Clear Buffer. RS232 mode

```
BOOL birdRS232ClearBuffer();
```

Parameters

Int nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use `birdGetErrorMessage()`to view error.

Remarks

Clears the RS232 receiver buffer. "nGroupID" is the group identifier that was passed to `birdWakeUp()`.

TOP

BirdRS232GetResponse

Get a response. RS232 mode

```
BOOL birdRS232GetResponse();
```

Parameters

Int	nGroupID
Int	nDeviceNum
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **birdGetErrorMessage()** to view error.

Remarks

Gets a generic response in RS232 mode. "nGroupID" is the group identifier that was passed to **birdWakeUp()**. The response bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

BirdRS232GroupModeEnabled

Returns current status of group mode

```
BOOL GroupModeEnabled();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	Group Mode is Enabled
FALSE	Group Mode is not Enabled

Remarks

Reports the current status of group mode. Group mode affects the speed at which data can be collected, and whether or not measurement cycle reporting can be enabled. "nGroupID" is the group identifier that was passed to **birdWakeUp()**.

TOP

BirdRS232Reset

Reset. RS232 mode

```
BOOL birdRS232Reset();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Resets a group of birds by toggling the RS232 RTS line. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

TOP

BirdRS232Resynch

Resynchronize. RS232 mode

```
BOOL birdRS232Resynch();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Resynchronizes the RS232 data stream during streamed frame acquisition. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

TOP

BirdRS232SendCommand

Send a command in RS232 mode

```
BOOL birdRS232SendCommand();
```

Parameters

Int	nGroupID
Int	nDeviceNum
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Sends a generic command in RS232 mode. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. The command bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

BirdRS232SetGroupMode

Set RS232 group mode

```
BOOL birdRS232Reset();
```

Parameters

Int	nGroupID
Int	nGroupMode setting

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Allows the user to specify whether or not the birds will be put into group mode during data collection. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. "nGroupModeSetting" may take one of three values: `GMS_DEFAULT`, `GMS_GROUP_MODE_NEVER`, or `GMS_GROUP_MODE_ALWAYS`. The default setting is for the birds to be put into group mode if and only if there are one or more birds lacking a direct RS232.

Note

The latest implementation of **`birdRS232WakeUp`** allows you to specify the group mode setting when you call the function, making a call to birdRS232GroupMode unnecessary unless you need to change modes after initializing the flock.

TOP

birdRS232WakeUp

Wakes up a group of birds using the RS232 interface.

```
BOOL birdRS232WakeUp();
```

Parameters

Int	nGroupID
BOOL	bStandAlone
Int	nNumDevices
WORD	*pwAddress
DWORD	dwReadTimeout
DWORD	dwWriteTimeout

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Wakes up a group of birds in RS232 mode. "nGroupID" can be any number between 0 and BIRD_MAX_GROUP_ID that the user wants to associate with this group of birds. "bStandAlone" indicates if the group consists of a single bird operating in stand-alone mode. "nNumDevices" is the number of devices in the group, including any ERC's, and should be equal to the bird number of the last device. (This field is only relevant when bStandAlone = FALSE.) "pwComport" points to an array of words, each of which is the number of the comport attached to one of the birds (e.g., COM1 = 1, COM2 = 2, etc.) If bStandAlone = TRUE, then the bird's comport number is passed in pwComport[0]. Otherwise, the array is indexed by bird number - for example, pwComport[2] would be the comport attached to bird #2. Any birds which do not have a direct connection to a comport must be given a comport number of zero. "dwBaudRate" is the baud rate to use. "dwReadTimeout" is the maximum time, in msecs, that the application will take when trying to receive a character. "dwWriteTimeout" is the maximum time, in msecs, that the application will take when trying to transmit a character. "nGroupMode" defines whether or not the data sampling will happen in group mode. In group mode all the data is passed back via the Master com port (1). In non-group mode all the data is passed back via dedicated com ports. Non-group mode is the only time that the pwComport array values with indices of greater than 1 are used.

NOTE: The first bird in this group must have a direct connection to a comport.

NOTE: The memory pointed to by "pwComport" may be released after the function returns.

SAMPLE CODE
TOP

birdSetDeviceConfig

Sets device configuration.

```
BOOL birdSetDeviceConfig();
```

Parameters

Int	nGroupID
Int	nDeviceNum
BIRDDEVICECONFIG	*pdevcfg

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Sets the configuration of a single device. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. "nDeviceNum" specifies the bird number (ignored for stand-alone mode). The device configuration is returned in "pdevcfg". See the definition of the **BIRDDEVICECONFIG** for an explanation of this structure.

SAMPLE CODE

TOP

birdSetFastDeviceConfig

Sets device configuration.

```
BOOL birdSetFastDeviceConfig();
```

Parameters

Int	nGroupID
Int	nDeviceNum
BIRDDEVICECONFIG	*pdevcfg

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Faster (and less error-prone) version of birdSetDeviceConfig(). In RS232 and ISA modes, the following fields of **BIRDDEVICECONFIG** are not used by Setup

byReportRate
byHemisphere
wAlphaMin[]
wAlphaMax[]
wVM[]
anglesReferenceFrame
anglesAngleAlign

If you need to use one of the above parameters, use **`birdSetDeviceConfig`** instead.

SAMPLE CODE

TOP

birdSetSystemConfig

Sets the System Configuration.

```
BOOL birdSetSystemConfig();
```

Parameters

Int	nGroupID
BIRDSYSTEMCONFIG	*psyscfg

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Sets the system configuration of a group of birds. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. The system configuration is passed in "psyscfg". See the definition of BIRDSYSTEMCONFIG for an explanation of this structure.

NOTE: The memory pointed to by "psyscfg" may be released after the function returns.

SAMPLE CODE

TOP

birdStartFrameStream

Start frame stream

```
BOOL birdStartFrameStream();
```

Parameters

Int nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Starts streaming of bird data frames in TCP/IP or RS232 mode. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**. The frames are retrieved by calling **`birdGetFrame()`**.

NOTE: Group-mode data collection under ISA has been disabled.

NOTE: This routine cannot be called while measurement cycle reporting is enabled, i.e. between calls to **`birdEnableMeasurementCycleReporting()`** and **`birdDisableMeasurementCycleReporting()`**.

NOTE: In master/slave mode, the first call to **`birdStartFrameStream()`** may involve a substantial delay, because the routine must take the birds into group mode. You should therefore make at least one call to **`birdStartFrameStream()`** before doing so from a time-critical part of the code.

SAMPLE CODE

TOP

birdStartReading

Starts acquisition of a single bird reading in RS232 or ISA mode

```
BOOL birdStartReading();
```

Parameters

Int	nGroupID
Int	nDeviceNum

Return Values

Value	Meaning

TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Starts acquisition of a single bird reading in RS232 or ISA mode. "nGroupID" is the group identifier that was passed to birdWakeUp(). "nDeviceNum" specifies the bird number (ignored for stand-alone mode). The reading is retrieved by calling `birdGetReading()`.

NOTE: If the bird is accessible only via the FBB, you should call **`birdGetReading()`** directly after **`birdStartReading()`** - i.e., before issuing any other bird commands.

NOTE: In master/slave mode, the first call to **`birdStartReading()`** may involve a substantial delay because the routine must take the birds out of group mode. You should therefore make at least one call to **`birdStartReading()`** before doing so from a time-critical part of the code.

TOP

birdStartSingleFrame

Start single frame

```
BOOL birdStartSingleFrame();
```

Parameters

Int nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Starts acquisition of a single frame of bird data. "nGroupID" is the group identifier that was passed to `birdWakeUp()`. The frame is retrieved by calling `birdGetFrame()`.

NOTE: In TCP/IP mode, this function will not work if the MotionStar is a tethered system using software prior to version 14.29. In such instances, you must instead call `birdStartFrameStream()`, `birdGetFrame()`, and `birdStopFrameStream()` to get a single frame of bird data.

NOTE: In ISA mode, data is collected from internal birds only. If you need to collect data from external birds, use `birdStartReading()` instead of this routine to manually get readings from each bird.

NOTE: In RS232 mode, if all the birds are connected to RS232 ports, then data will be collected from each bird directly. Otherwise, the birds are put into group mode, and all of the data is collected via the master bird.

NOTE: In master/slave mode, the first call to `birdStartSingleFrame()` may involve a substantial delay because the routine must take the birds into or out of group mode. You should therefore make at least one call to `birdStartSingleFrame()` before doing so from a time-critical part of the code.

TOP

birdStopFrameStream

Stop Frame Stream

```
BOOL birdStopFrameStream();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	Reading is Ready
FALSE	Reading is not ready

Remarks

Stops streaming of bird data frames. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

SAMPLE CODE
TOP

BirdTCPIPClearBuffer

Clear the buffer. TCP/IP mode

```
BOOL birdTCPIPClearBuffer();
```

Parameters

Int	nGroupID

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use **`birdGetErrorMessage()`** to view error.

Remarks

Clears the TCP/IP receiver buffer. "nGroupID" is the group identifier that was passed to **`birdWakeUp()`**.

TOP

BirdTCPIPGetResponse

Get a response. TCP/IP mode

```
BOOL birdTCPIPGetResponse();
```

Parameters

Int	nGroupID
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully

FALSE	An error occurred while processing command, use `birdGetErrorMessage()` to view error.

Remarks

Gets a generic response in TCP/IP mode. "nGroupID" is the group identifier that was passed to `birdWakeUp()`. The response bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

birdTCPIPSendCommand

Send a command in TCP/IP mode

```
BOOL birdTCPIPSendCommand();
```

Parameters

Int	nGroupID
Void	*pbuffer
WORD	wNumBytes

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use `birdGetErrorMessage()` to view error.

Remarks

Sends a generic command in TCP/IP mode. "nGroupID" is the group identifier that was passed to `birdWakeUp()`. The command bytes are passed in "pbuffer", and the number of bytes is passed in "wNumBytes".

TOP

BirdTCPIPWakeUp

Wake Up a Flock. TCP/IP mode

```
BOOL birdTCPIPWakeUp;
```

Parameters

Int	nGroupID
LPCTSTR	lpszServerIPAddress
WORD	wServerIPPort
Int	nNumDevices

Return Values

Value	Meaning
TRUE	No errors occurred. Call completed successfully
FALSE	An error occurred while processing command, use `birdGetErrorMessage()` to view error.

Remarks

Wakes up a group of birds in TCP/IP mode. "nGroupID" can be any number between 0 and BIRD_MAX_GROUP_ID that the user wants to associate with this group of birds. "lpszServerIPAddress" is a string denoting the IP address of the bird server. "wServerIPPort" is the server's port number. "nNumDevices" is the number of devices in the group, including any ERC's, and should be equal to the bird number of the last device. If set to 0, the driver will attempt to query the value from the birds.

SAMPLE CODE

TOP

BIRD.DLL Data Structures

typedef struct tagBIRDSYSTEMCONFIG

{

BYTE	bySystemStatus	current system status (see BIRD SYSTEM STATUS bits)
BYTE	byError	error code flagged by server or master bird
BYTE	byNumDevices	number of devices in system
BYTE	byNumServers	number of servers in system
BYTE	byXmtrNum	transmitter number (see TRANSMITTER NUMBER bits)
WORD	wXtalSpeed	crystal speed in MHz
double	dMeasurementRate	measurement rate in frames per second
BYTE	byChassisNum	chassis number
BYTE	byNumChassisDevices	number of devices within this chassis
BYTE	byFirstDeviceNum	number of first device in this chassis
WORD	wSoftwareRev	software revision of server application or master bird
BYTE	byFlockStatus[ ]	status of all devices in flock, indexed by bird number (see note in BIRDFRAME definition and BIRD FLOCK STATUS bits

} BIRDSYSTEMCONFIG

typedef struct tagBIRDDEVICECONFIG

{

BYTE	byStatus	device status (see BIRD DEVICE STATUS bits)
BYTE	byID	device ID code (see BIRD DEVICE ID bits)
WORD	wSoftwareRev	software revision of device
BYTE	byError	error code flagged by device
BYTE	bySetupsetup	information (see BIRD DEVICE SETUP bits)
BYTE	byDataFormat	data format (see BIRD DATA FORMAT bits)
BYTE	byReportRate	rate of data reporting, in units of frames
WORD	wScaling	full scale measurement, in inches
BYTE	byHemisphere	hemisphere of operation (see BIRD HEMISPHERE CODES)
BYTE	byDeviceNum	bird number
BYTE	byXmtrType	transmitter type (see BIRD TRANSMITTER TYPE bits)
WORD	wAlphaMin[7]	filter constants (see ALPHA_MIN FILTER_TABLE for values)
WORD	wAlphaMax[7]	filter constants (see ALPHA_MAX FILTER_TABLE for values)
WORD	wVM[7]	filter constants (see Vm FILTER_TABLE for values)
BIRDANGLES	anglesReferenceFrame	reference frame of bird readings
BIRDANGLES	anglesAngleAlign	alignment of bird readings

} BIRDDEVICECONFIG

typedef struct tagBIRDFRAME

{

DWORD	dwTime;	time at which readings were taken, in msecs
BIRDREADING	reading[ ]	reading from each bird

}BIRDFRAME

NOTE: In stand-alone mode, the bird reading is stored in reading[0], and all other array elements are unused. In master/slave mode, the "reading" array is indexed by bird number - for example, bird #1 is at reading[1], bird #2 is at reading[2], etc., and reading[0] is unused.

typedef struct tagBIRDREADING

```
{
        BIRDPOSITION        position  position of receiver
        BIRDANGLES          angles          orientation of receiver, as angles
        BIRDMATRIX          matrix          orientation of receiver, as matrix
        BIRDQUATERNION      quaternion      orientation of receiver, as quaternion
        WORD                wButtons        button states
} BIRDREADING
```

NOTE: In stand-alone mode, the bird reading is stored in reading[0], and all other array elements are unused. In master/slave mode, the "reading" array is indexed by bird number - for example, bird #1 is at reading[1], bird #2 is at reading[2], etc., and reading[0] is unused.

Bird flock status bits

#define BFS_FBBACCESSIBLE	0x80	device is accessible on the FBB
#define BFS_RUNNING	0x40	device is initialized and running
#define BFS_RECEIVERPRESENT	0x20	device has a receiver
#define BFS_ERC	0x10	device is an ERC
#define BFS_ERT3	0x08	if an ERC, ERT#3 is present
#define BFS_ERT2	0x04	if an ERC, ERT#2 is present
#define BFS_ERT1	0x02	if an ERC, ERT#1 is present
#define BFS_ERT0	0x01	if an ERC, ERT#0 is present, else SRT is present

Bird system status bits

#define BSS_RUNNING	0x80	system is initialized and running
#define BSS_ERROR	0x40	error in system occurred
#define BSS_FBB_ERROR	0x20	error on FBB bus occurred
#define BSS_LOCAL_ERROR	0x10	error in local chassis occurred
#define BSS_LOCAL_POWER	0x08	error in local power status occurred
#define BSS_MASTER	0x04	local chassis is a master
#define BSS_CRTSYNC_TYPE	0x02	type of CRT sync being used
#define BSS_CRTSYNC	0x01	CRT sync mode is enabled

Bird device status bits

#define BDS_ERROR	0x80	error in device occurred
#define BDS_RUNNING	0x40	device is initialized and running
#define BDS_BUTTONSPRESENT	0x08	device has buttons
#define BDS_RECEIVERPRESENT	0x04	device has a receiver
#define BDS_TRANSMITTERPRESENT	0x02	device has a transmitter
#define BDS_TRANSMITTERRUNNING	0x01	device has an active transmitter

Bird device ID's

#define BDI_6DFOB	1	Standalone (SRT)
#define BDI_6DERC	2	Extended Range Controller
#define BDI_6DBOF	3	Motionstar (old ID)
#define BDI_PCBIRD	4	PC Bird
#define BDI_SPACEPAD	5	Spacepad
#define BDI_MOTIONSTAR	6	Motionstar (new ID)
#define BDI_UNRECOGNIZED	255	unrecognized device

Bird device setup bits

#define BDS_SUDDENOUTPUTCHANGE	0x20	sudden large data change will not update output data
#define BDS_XYZREFERENCE	0x10	position is derived from XYZ reference frame angle table
#define BDS_APPENDBUTTONDATA	0x08	button data is appended
#define BDS_ACNARROWNOTCHFILTER	0x04	AC narrow notch filter is in use
#define BDS_ACWIDENOTCHFILTER	0x02	AC wide notch filter is in use
#define BDS_DCFILTER	0x01	DC filter is in use

Bird data formats

#define BDF_NOBIRDDATA	0	no data - RS232 and ISA modes have no way of specifying this
#define BDF_POSITION	1	position only
#define BDF_ANGLES	2	angles only
#define BDF_MATRIX	3	matrix only
#define BDF_POSITIONANGLES	4	position and angles
#define BDF_POSITIONMATRIX	5	position and matrix
#define BDF_QUATERNION	7	quaternion only
#define BDF_POSITIONQUATERNION	8	position and quaternion

Bird hemisphere codes

#define BHC_FRONT	0	front hemisphere
#define BHC_REAR	1	rear hemisphere
#define BHC_UPPER	2	upper hemisphere
#define BHC_LOWER	3	lower hemisphere
#define BHC_LEFT	4	left hemisphere
#define BHC_RIGHT	5	right hemisphere

Bird transmitter type bits

#define BTT_ERT	0x80	ERT is present
#define BTT_SRT	0x40	SRT is present
#define BTT_PCBIRD	0x20	PCBIRD is present
#define BTT_ACTIVE	0x10	transmitter is active
#define BTT_SELECTED1	0x08	index of selected ERT
#define BTT_SELECTED0	0x04	N/A
#define BTT_NUMBER1	0x02	number of ERT's present
#define BTT_NUMBER0	0x01	N/A

Bird transmitter number bits

#define BTN_FBBADDRESS	0xF0	FBB address of transmitter
#define BTN_FBBADDRESS3	0x80	
#define BTN_FBBADDRESS2	0x40	
#define BTN_FBBADDRESS1	0x20	
#define BTN_FBBADDRESS0	0x10	
#define BTN_TRANSMITTERNUMBER	0x03	index of transmitter
#define BTN_TRANSMITTERNUMBER1	0x02	
#define BTN_TRANSMITTERNUMBER0	0x01	

BIRD.DLL Error table

#define BE_NOERROR	0x00	no error
#define BE_GENERICERROR	0x01	generic error
#define BE_TCPIPERROR	0x10	TCP/IP communications error
#define BE_BIRDERROR	0x11	error signaled by birds
#define BE_INVALIDPACKETTYPEERROR	0x12	invalid packet type received
#define BE_INVALIDBIRDNUMBERERROR	0x13	invalid bird number received
#define BE_INVALIDDATAFORMATERROR	0x14	invalid data format received
#define BE_RS232ERROR	0x20	RS232 communications error
#define BE_PHASEERROR	0x21	error in phasing bits
#define BE_RESYNCHERROR	0x22	unable to resynch after phase error
#define BE_OVERRUNERROR	0x23	measurement cycle was overrun
#define BE_ISAERROR	0x30	ISA communications error

Alpha Min and Alpha Max Data Structures

ALPHA_MIN FILTER_TABLE structure

SRT (Standard Range Transmitter) or pcBIRD

Word	Range of values	Default	Range (Inches)
1	0 – 0.99996	.02	0 to 17
2	“ “	“ “	17 to 22
3	“ “	“ “	22 to 27
4	“ “	“ “	27 to 34
5	“ “	“ “	34 to 42
6	“ “	“ “	42 to 54
7	“ “	“ “	54+

ERC / ERT (Extended Range Controller / Extended Range Transmitter

Word	Range of values	Default	Range (Inches)
1	0 – 0.99996	.02	0 to 55
2	“ “	“ “	55 to 70
3	“ “	“ “	70 to 90
4	“ “	“ “	90 to 110
5	“ “	“ “	110 to 138
6	“ “	“ “	138 to 170
7	“ “	“ “	170+

Remarks

When ALPHA_MIN = 0 Hex, the filter will provide an infinite amount of filtering (the outputs will never change even if you move the sensor). When ALPHA_MIN = 0.99996 = 7FFF Hex, the DC filter will provide no filtering of the data.

At the shorter ranges you may want to increase ALPHA_MIN to obtain less lag while at longer ranges you may want to decrease ALPHA_MIN to provide more filtering (less noise/more lag). If you decrease the value below 0.008, the output noise will actually increase due to loss of mathematical precision. ALPHA_MIN must always be less than ALPHA_MAX.

ALPHA_MAX FILTER_TABLE structure

SRT (Standard Range Transmitter) or pcBIRD

Word	Range of values	Default	Range (Inches)
1	0-0.99996	0.9	0 to 17
2	“ “	“ “	17 to 22
3	“ “	“ “	22 to 27
4	“ “	“ “	27 to 34
5	“ “	“ “	34 to 42
6	“ “	“ “	42 to 54
7	“ “	“ “	54+

ERC / ERT (Extended Range Controller / Extended Range Transmitter

Word	Range of values	Default	Range (Inches)
1	0-0.99996	0.9	0 to 55
2	“ “	“	55 to 70
3	“ “	“	70 to 90
4	“ “	“	90 to 110
5	“ “	“	110 to 138
6	“ “	“	138 to 170
7	“ “	“	170+

Remarks

When there is a fast motion of the sensor, the adaptive filter reduces the amount of filtering by increasing the ALPHA used in the filter. It will increase ALPHA only up to the limiting ALPHA_MAX value. By doing this, the lag in the filter is reduced during fast movements. When ALPHA_MAX = .99996 = 0x7FFF, the filter will provide no filtering of the data during fast movements.

During fast motion, you may want to decrease ALPHA_MAX to increase the amount of filtering if The Bird's outputs are too noisy during rapid sensor movement. ALPHA_MAX must always be greater than ALPHA_MIN.

See the product manual for more information on the DC filter.

Vm FILTER_TABLE structure

SRT (Standard Range Transmitter) or pcBIRD

Word	Range of values	Default	Range (Inches)
1	1 – 32767	2	0 to 17
2	“ “	4	17 to 22
3	“ “	8	22 to 27
4	“ “	32	27 to 34
5	“ “	64	34 to 42
6	“ “	256	42 to 54
7	“ “	512	54+

ERC / ERT (Extended Range Controller / Extended Range Transmitter

Word	Range of values	Default	Range (Inches)
1	1 – 32767	2	0 to 55
2	“ “	4	55 to 70
3	“ “	8	70 to 90
4	“ “	32	90 to 110
5	“ “	64	110 to 138
6	“ “	256	138 to 170
7	“ “	512	170+

Remarks

The DC filter is adaptive in that it tries to reduce the amount of low pass filtering in The Bird as it detects translation or rotation rates in The Bird's sensor. Reducing the amount of filtering results in less filter lag. Unfortunately electrical noise in the environment, when measured by the sensor, also makes it look like the sensor is undergoing a translation and rotation. As the sensor moves farther and farther away from the transmitter, the amount of noise measured by the sensor appears to increase because the measured transmitted signal level is decreasing and the sensor amplifier gain is increasing. In order to decide if the amount of filtering should be reduced, The Bird has to know if the measured rate is a real sensor rate due to movement or a false rate due to noise. The Bird gets this knowledge by the user specifying what the expected noise levels are in the operating environment as a function of distance from the transmitter. These noise levels are the 7 words that form the Vm table. The Vm values can range from 1 for almost no noise to 32767 for a lot of noise.

As Vm increases with range so does the amount of filter lag. To reduce the amount of lag, reduce the larger Vm values until the noise in The Bird's output is too large for your application.

Example Code #1   RS232 in Stand-Alone Mode Wake Up

```
/*
RS232 Simple Stand Alone Mode
Wakes up a single tracker set in Stand-Alone Mode
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"

// Constants
#define GROUP_ID        1           // arbitrary designation for group
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)
#define BAUD_RATE       115200      // 115.2K baud

void main ()
{
      // local variables
      BOOL status = 0;               // return status of bird calls
      WORD COM_port=3;               // BIRD COM port

      printf("Ascension Technology Corporation - Simple Stand Alone
      Mode(RS232) Wake Up 05/16/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdRS232WakeUp(GROUP_ID,
            TRUE,                         // stand-alone
            1,                            // Number of Devices
            &COM_port,                    // Com Port
            BAUD_RATE,                    // BAUD
            READ_TIMEOUT,WRITE_TIMEOUT,   // Response timeouts
            GMS_GROUP_MODE_NEVER)))       // Group mode doesn’t apply
                                          // when using Stand-Alone mode
      {
            printf("%s\n",birdGetErrorMessage());
            exit(-1);
      }
      printf(“Bird is now awake\n”);
      return;
}
```

Example Code #2   RS232 Master Wake Up

```
/*
RS232 Simple Master Mode
Wakes up a single tracker set in Master Mode
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"


// Constants
#define GROUP_ID        1           // arbitrary designation for group
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)
#define BAUD_RATE       115200      // 115.2K baud

void main ()
{
      // local variables
      WORD COM_port[2]={0,3}; // BIRD COM port. When not using Stand-Alone
                              // Mode, first element must be 0

      printf("Ascension Technology Corporation - Simple Master Mode(RS232)
      Wake Up 05/16/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdRS232WakeUp(GROUP_ID,
            FALSE,                        // Don’t use stand-alone mode
            1,                            // Number of Devices
            COM_port,                     // Com Port
            BAUD_RATE,                    // BAUD
            READ_TIMEOUT,WRITE_TIMEOUT,   // Response timeouts
            GMS_GROUP_MODE_NEVER)))       // Don’t use Group mode
      {
            printf("%s\n",birdGetErrorMessage());
            exit(-1);
      }
      printf(“Bird is now awake\n”);
      return;

}
```

Example Code #3   ISA in Stand-Alone Mode Wake Up

```
/*
ISA Wake Up
Wakes up a single ISA tracker
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"

// Constants
#define GROUP_ID        1           // arbitrary designation for group
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)

 #define GROUP_ID       1           // arbitrary designation for group
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)

void main ()
{

// local variables
WORD  ISA_port[2] = {0,512};        // The first element must be 0 and the
                                    // second elements is the BASE address
                                    // of the first device.

      if    ((!birdISAWakeUp(
            GROUP_ID,                     // arbitrary designation
            TRUE,                         // Use Stand-Alone Mode
            1,                            // 1 Device
            ISA_port,                     // ISA Port
            READ_TIMEOUT,WRITE_TIMEOUT))) // Reponse timeouts
            {
            printf("%s\n",birdGetErrorMessage());
            exit(-1);
      }
      printf(“Bird is now awake\n”);
      return;
}
```

Example Code #4 TCPIP Mode Wake Up

```
/*
TCPIP Simple Wake Up
Wakes up a tracker using the TCPIP interface
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"


// Constants
#define GROUP_ID        1           // arbitrary designation for group
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)
#define BAUD_RATE       115200      // 115.2K baud
#define UDP             5000        // Port
#define IP_Address "192.168.1.60"   // IP Address

void main ()
{
      printf("Ascension Technology Corporation – Simple TCPIP Wake Up
      05/16/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdTCPIPWakeUp(
            GROUP_ID,   // arbitrary designation for group
            IP_Address, // IP Address
            UDP,        // TCP protocol
            2)))        // Number of Devices
      {
            printf("%s\n",birdGetErrorMessage());
            exit(-1);
      }
      printf(“Bird is now awake\n”);
      return;

}
```

Example Code #5   Get and Set System Config

```
/*
Get and Set system config demo
Makes and Apply changes to system parameters
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"


#define GROUP_ID      1       // arbitrary designation for group
#define DEVICE_COUNT  1       // number of birds in Flock
#define READ_TIMEOUT  2000    // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT 2000    // 2000 mSec (2 seconds)
#define BAUD_RATE     115200  // 115.2K baud

void main ()
{     // local variables
      WORD  COM_port=3;       // see birdRS232WakeUp description for details
      BIRDSYSTEMCONFIG sysconfig;

      // Wake up the flock
      birdRS232WakeUp(GROUP_ID,
            TRUE,                   // Use stand-alone mode
            1,                      // Number of Devices
            &COM_port,              // Com Port
            BAUD_RATE,              // Baud Rate
            READ_TIMEOUT,           // Read and Write Timeouts
            WRITE_TIMEOUT,
            GMS_GROUP_MODE_NEVER);  // Don’t use group mode

// Get the system configuration
birdGetSystemConfig(GROUP_ID,
&sysconfig,                         // Structure to put sysconfig into
TRUE);                              // Use copy of sysconfig read at Wake Up

      // Change the measurement rate in the sysconfig structure
      sysconfig.dMeasurementRate = 101.3;

      // send the sysconfig structure to flock
      birdSetSystemConfig(
            GROUP_ID,
            &sysconfig);

return;
}
```

Example Code #6   Get and Set Device Config

```
/*
Get and Set device config demo
Makes and Apply changes to device parameters
*/


#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"


#define GROUP_ID        1           // arbitrary designation for group
#define DEVICE_COUNT    1           // number of birds in Flock
#define READ_TIMEOUT    2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT   2000        // 2000 mSec (2 seconds)
#define BAUD_RATE       115200      // 115.2K baud

void main ()
{     // local variables
      WORD  COM_port=3;             // See birdRS232WakeUp for details
      BIRDSYSTEMCONFIG sysconfig;   // Structure for system config
      BIRDDEVICECONFIG devconfig[1];// Structure for device config
      double measurement_rate;      // Variable for measurement rate


      // Wake up the flock
      birdRS232WakeUp(GROUP_ID,
            FALSE,                  // Use stand-alone mode
            1,                      // Number of Devices
            &COM_port,              // Com Port
            BAUD_RATE,              // Baud rate
            READ_TIMEOUT,           // Read and Write timeout
            WRITE_TIMEOUT,
            GMS_GROUP_MODE_NEVER)   // Don't use group mode

// Get the system configuration
birdGetSystemConfig(GROUP_ID,
&sysconfig,                         // Structure to put sysconfig into
TRUE);                              // Use copy of sysconfig read at Wake Up


// Get the device configuration
birdGetFastDeviceConfig(
GROUP_ID,
1,
&devconfig[0]);

// Change the sensor data format rate in the devconfig structure
devconfig[0].byDataFormat = 5;      // 5 corresponds to Position Matrix
                                    // in the Bird Data Formats table.

// send the devconfig structure to flock
birdSetFastDeviceConfig(
      GROUP_ID,
      0,
      &devconfig[0]);
return;
}
```

Example Code #7   Single Bird Frame Stream

```
/*
RS232 Simple Stand Alone Mode
05/16/2006
Gets Position / Angles from RS232 Tracker and streams to screen
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"

// Constants
#define GROUP_ID              1           // arbitrary designation for group
#define READ_TIMEOUT          2000        // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT         2000        // 2000 mSec (2 seconds)
#define BAUD_RATE             115200      // 115.2K baud

void main ()
{
// local variables

      WORD  COM_port = 3;     // When using stand-alone mode, first
                              // element must be COM port of first tracker
      BIRDSYSTEMCONFIG sysconfig;         // Holds System configuration
      BIRDDEVICECONFIG devconfig[1];      // Holds Bird configuration
      BIRDFRAME frame;                    // Holds new frames of data
      double pos[3],ang[3];               // Array for P+O

      printf("Ascension Technology Corporation - Simple Stand Alone
Mode(RS232)    05/16/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdRS232WakeUp(
            GROUP_ID,                     // arbitrary designation for group
            TRUE,                         // Stand_alone mode
            1,                            // Number of Devices
            &COM_port,                    // Com Port
            BAUD_RATE,                    // BAUD
            READ_TIMEOUT,WRITE_TIMEOUT,   // Reponse timeouts
            GMS_GROUP_MODE_NEVER)))       // Don’t use group mode in Stand-
                                          // alone mode
      {
            printf("Can't Wake Up Flock!\n");
            Sleep(1000);
            exit(-1);
      }

// Read system configuration data from bird into sysconfig structure
      if (!birdGetSystemConfig(
            GROUP_ID,   // arbitrary designation for group
            &sysconfig, // Structure for System configuration
            TRUE))      // Yes, use copy of System Config from Wake Up command
      {
            printf("%s\n",birdGetErrorMessage());
            Sleep(1000);
            exit(-1);
      }
      // Read the device configuration into the devconfig structure
      if (!birdGetFastDeviceConfig(
            GROUP_ID,         // arbitrary designation for group
                  1,          // Device address, ignored in Stand-Alone mode
                              // but a value must be here.
            &devconfig[0]))   // Holds device configuration for first device,
                              // in stand-alone mode the index of the first
                              // device is 0.
      {
            printf("%s\n",birdGetErrorMessage());
            printf("Couldn't get device configuration for bird!");
            Sleep(1000);
            exit(-1);
      }

      // Start getting data
      birdStartFrameStream(GROUP_ID);
      do    // Until Keypress
      {
            // Check if there's data available
            if(birdFrameReady(GROUP_ID))
            {
            // Reads data from bird
            birdGetMostRecentFrame(GROUP_ID,&frame);
                  BIRDREADING *bird_data;       // Moves data into structure
                  bird_data = &frame.reading[0]; // Sets pointer to bird 0

                  // Convert data into inches and degrees and scale
                  pos[0] = bird_data->position.nX * 36 / 32767.;
                  pos[1] = bird_data->position.nY * 36 / 32767.;
                  pos[2] = bird_data->position.nZ * 36 / 32767.;
                  ang[0] = bird_data->angles.nAzimuth * 180. / 32767.;
                  ang[1] = bird_data->angles.nElevation * 180. / 32767.;
                  ang[2] = bird_data->angles.nRoll * 180. / 32767.;

                  // print data
                  printf("%+6.1f  %+6.1f  %+6.1f  ",pos[0], pos[1], pos[2]);
                  printf("%+6.1f  %+6.1f  %+6.1f  \n",ang[0], ang[1], ang[2]);

            } // end if frame ready routine
      }while(!kbhit()); // loop until any key is pressed

      printf("EXITING...                             \n");

      birdStopFrameStream(GROUP_ID);
      birdShutDown(GROUP_ID);

      return;
}
```

Example Code #8   Multiple Bird Frame Stream

```
/*
RS232 Multiple Bird Frame Stream
05/16/2006
For multiple RS232 Trackers and 1 COM port.
Gets Position / Angles and streams to screen
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"

// Constants
#define GROUP_ID                1     // arbitrary designation for group
#define READ_TIMEOUT            2000  // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT           2000  // 2000 mSec (2 seconds)
#define BAUD_RATE               115200      // 115.2K baud
#define measurement_rate        103.3 // Bird measurement rate

void main ()
{
      // local variables
      BOOL  status = 0;                   // return status of bird calls

      /*
      COM port definition
      When using multiple birds and a single COM port, the first element is
      always 0,   the second element is the COM port of first tracker and
      additional elements for each device are 0. Additional elements for
      trackers not present are acceptable
      */

      WORD  COM_port[5] = {0,1,0,0,0};
      BIRDSYSTEMCONFIG sysconfig;         // Holds System configuration
      BIRDDEVICECONFIG devconfig[5];      // Holds Bird configuration
      BIRDFRAME frame;                    // Holds new frames of data
      double pos[3],ang[3];               // Array for Position + Orientation
                                          // data
      int DEVCOUNT = 3;                   // Number of Trackers

      printf("Ascension Technology Corporation - Multiple Bird Stream
            Mode(RS232) 01/31/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdRS232WakeUp(GROUP_ID,
            FALSE,                                    // Not stand-alone
            DEVCOUNT,                                 // Number of Devices
            COM_port,                                 // COM Port
            BAUD_RATE,                                // BAUD
            READ_TIMEOUT,WRITE_TIMEOUT,               // Reponses timeouts
            GMS_GROUP_MODE_ALWAYS)))            // Use group mode when 1 COM
                                                // port and multiple trackers
      {
            printf("Can't Wake Up Flock!\n");
            Sleep(1000);
            exit(-1);
      }
```

```
// Read system configuration data from bird into sysconfig structure
if (!birdGetSystemConfig(GROUP_ID,&sysconfig,FALSE))
{
      printf("%s\n",birdGetErrorMessage());
      Sleep(1000);
      exit(-1);
}

// Set the measurement rate by changing it in the sysconfig structure
sysconfig.dMeasurementRate = measurement_rate;

// Make changes to configuration by sending sysconfig structure
if (!birdSetSystemConfig(GROUP_ID,&sysconfig))
{
      printf("%s\n",birdGetErrorMessage());
      Sleep(1000);
      exit(-1);
}

// Read the device configuration into the devconfig structure
for(int i=0; i<DEVCOUNT; i++ )
{
      if (!birdGetFastDeviceConfig(GROUP_ID,i+1,&devconfig[i]))
      {
            printf("%s\n",birdGetErrorMessage());
            printf("Couldn't get device configuration for bird %i",i);
            Sleep(1000);
            exit(-1);
      }
}
// Start getting data...
birdStartFrameStream(GROUP_ID);
do    // Until Keypress
{
      if(birdFrameReady(GROUP_ID))  // Check if there's data available
      {
            birdGetMostRecentFrame(GROUP_ID,&frame);//Get data from bird
            BIRDREADING *bird_data; // Transfers data into structure

            for(i=1; i<DEVCOUNT+1; i++ )  // Loop to get data from birds
            {
                  // Change pointer to index of first bird (1)
                  bird_data = &frame.reading[i];
                  // Convert data into inches and degrees and scale
                  pos[0] = bird_data->position.nX * 36 / 32767.;
                  pos[1] = bird_data->position.nY * 36 / 32767.;
                  pos[2] = bird_data->position.nZ * 36 / 32767.;
                  ang[0] = bird_data->angles.nAzimuth * 180. / 32767.;
                  ang[1] = bird_data->angles.nElevation * 180. / 32767.;
                  ang[2] = bird_data->angles.nRoll * 180. / 32767.;

                  // print data
      printf("%i> %+6.1f  %+6.1f  %+6.1f   ”, i,pos[0], pos[1],pos[2]);
      printf(“%+6.1f  %+6.1f  %+6.1f      \n",ang[0], ang[1], ang[2]);
            }     // end move data from structure to screen loop

      } // end if frame ready loop

}while(!kbhit()); // loop until any key is pressed
```

```
        printf("EXITING...                              \n");

        birdStopFrameStream(GROUP_ID);
        birdShutDown(GROUP_ID);

        return;

}
```

Example Code #9   Multiple Bird Frame Stream with Dedicated COM ports

```
/*
RS232 Multiple Bird Frame Stream
05/16/2006
For multiple RS232 Trackers and 1 COM port.
Gets Position / Angles and streams to screen
*/

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <wincon.h>
#include "bird.h"

// Constants
#define GROUP_ID              1           // arbitrary designation for group
#define READ_TIMEOUT          2000  // 2000 mSec (2 seconds)
#define WRITE_TIMEOUT         2000  // 2000 mSec (2 seconds)
#define BAUD_RATE             115200      // 115.2K baud
#define measurement_rate      103.3 // Bird measurement rate

void main ()
{
      // local variables
      BOOL  status = 0;                   // return status of bird calls

      /*
      COM port definition
      When using multiple birds and dedicated COM ports, the first element is
      always 0,   the additional elements are the COM port of the trackers in
      the order they appear on the FBB Bus.
      */

      WORD  COM_port[5] = {0,1,2,3,4};
      BIRDSYSTEMCONFIG sysconfig;         // Holds System configuration
      BIRDDEVICECONFIG devconfig[5];      // Holds Bird configuration
      BIRDFRAME frame;                    // Holds new frames of data
      double pos[3],ang[3];               // Array for Position + Orrientation
                                          // data
      int DEVCOUNT = 3;                   // Number of Trackers

      printf("Ascension Technology Corporation - Multiple Bird Stream
            Mode(RS232) 01/31/2006\n");

      printf("Initializing Flock Of Birds\n\n");
      if    ((!birdRS232WakeUp(GROUP_ID,
            FALSE,                                  // Not stand_alone
            DEVCOUNT,                               // Number of Devices
            COM_port,                               // COM Port
            BAUD_RATE,                              // BAUD
            READ_TIMEOUT,WRITE_TIMEOUT,       // Reponse timeouts
            GMS_GROUP_MODE_NEVER)))       // Don’t use group mode when using
                                          // dedicated COM ports
      {
            printf("Can't Wake Up Flock!\n");
            Sleep(1000);
            exit(-1);
```

```
}

// Read system configuration data from bird into sysconfig structure
if (!birdGetSystemConfig(GROUP_ID,&sysconfig,FALSE))
{
      printf("%s\n",birdGetErrorMessage());
      Sleep(1000);
      exit(-1);
}

// Set the measurement rate by changing it in the sysconfig structure
sysconfig.dMeasurementRate = measurement_rate;

// Make changes to configuration by sending sysconfig structure
if (!birdSetSystemConfig(GROUP_ID,&sysconfig))
{
      printf("%s\n",birdGetErrorMessage());
      Sleep(1000);
      exit(-1);
}

// Read the device configuration into the devconfig structure
for(int i=0; i<DEVCOUNT; i++ )
{
      if (!birdGetFastDeviceConfig(GROUP_ID,i+1,&devconfig[i]))
      {
            printf("%s\n",birdGetErrorMessage());
            printf("Couldn't get device configuration for bird %i",i);
            Sleep(1000);
            exit(-1);
      }
}

// Start getting data...
birdStartFrameStream(GROUP_ID);
do    // Until Keypress
{
      if(birdFrameReady(GROUP_ID))  // Check if there's data available
      {

            birdGetMostRecentFrame(GROUP_ID,&frame);//Get data from bird
            BIRDREADING *bird_data; // Transfers data into structure

            for(i=1; i<DEVCOUNT+1; i++ )  // Loop to get data from birds
            {
                  // Change pointer to index of first bird (1)
                  bird_data = &frame.reading[i];
                  // Convert data into inches and degrees and scale
                  pos[0] = bird_data->position.nX * 36 / 32767.;
                  pos[1] = bird_data->position.nY * 36 / 32767.;
                  pos[2] = bird_data->position.nZ * 36 / 32767.;
                  ang[0] = bird_data->angles.nAzimuth * 180. / 32767.;
                  ang[1] = bird_data->angles.nElevation * 180. / 32767.;
                  ang[2] = bird_data->angles.nRoll * 180. / 32767.;

                  // print data
                  printf("%i> %+6.1f  %+6.1f  %+6.1f  %+6.1f  %+6.1f
                  %+6.1f      \n",
                   i,pos[0], pos[1], pos[2], ang[0], ang[1], ang[2]);
            }     // end move data from structure to screen loop
            } // end if frame ready loop

      }while(!kbhit()); // loop until any key is pressed

      printf("EXITING...                             \n");

      birdStopFrameStream(GROUP_ID);
      birdShutDown(GROUP_ID);

      return;

}
```