SignalHawk™ Application Programming Interface

Programming Manual

This is a preliminary manual. Specifications, limits, and text are subject to change without notice. The information within this manual was as complete as possible at the time of printing. Bird Electronic Corporation is not liable for errors.

Safety Precautions

The following are general safety precautions that are not necessarily related to any specific part or procedure, and do not necessarily appear elsewhere in this publication. These precautions must be thoroughly understood and apply to all phases of operation and maintenance.

WARNING
Keep Away From Live Circuits
Operating Personnel must at all times observe general safety precautions. Do not replace components or make adjustments to the inside of the test equipment with the high voltage supply turned on. To avoid casualties, always remove power.

WARNING
Shock Hazard
Do not attempt to remove the RF transmission line while RF power is present.

WARNING
Do Not Service Or Adjust Alone
Under no circumstances should any person reach into an enclosure for the purpose of service or adjustment of equipment except in the presence of someone who is capable of rendering aid.

WARNING
Safety Earth Ground
An uninterruptible earth safety ground must be supplied from the main power source to test instruments. Grounding one conductor of a two conductor power cable is not sufficient protection. Serious injury or death can occur if this grounding is not properly supplied.

WARNING
Resuscitation
Personnel working with or near high voltages should be familiar with modern methods of resuscitation.

WARNING
Remove Power
Observe general safety precautions. Do not open the instrument with the power on.

Safety Symbols

WARNING
Warning notes call attention to a procedure, which if not correctly performed, could result in personal injury.

CAUTION
Caution notes call attention to a procedure, which if not correctly performed, could result in damage to the instrument.

Note: *Calls attention to supplemental information.*

Safety Statements

USAGE

ANY USE OF THIS INSTRUMENT IN A MANNER NOT SPECIFIED BY THE MANUFACTURER MAY IMPAIR THE INSTRUMENT'S SAFETY PROTECTION.

USO

EL USO DE ESTE INSTRUMENTO DE MANERA NO ESPECIFICADA POR EL FABRICANTE, PUEDE ANULAR LA PROTECCIÓN DE SEGURIDAD DEL INSTRUMENTO.

BENUTZUNG

WIRD DAS GERÄT AUF ANDERE WEISE VERWENDET ALS VOM HERSTELLER BESCHRIEBEN, KANN DIE GERÄTESICHERHEIT BEEINTRÄCHTIGT WERDEN.

UTILISATION

TOUTE UTILISATION DE CET INSTRUMENT QUI N'EST PAS EXPLICITEMENT PRÉVUE PAR LE FABRICANT PEUT ENDOMMAGER LE DISPOSITIF DE PROTECTION DE L'INSTRUMENT.

IMPIEGO

QUALORA QUESTO STRUMENTO VENISSE UTILIZZATO IN MODO DIVERSO DA COME SPECIFICATO DAL PRODUTTORE LA PROZIONE DI SICUREZZA POTREBBE VENIRNE COMPROMESSA.

SERVICE

SERVICING INSTRUCTIONS ARE FOR USE BY SERVICE - TRAINED PERSONNEL ONLY. TO AVOID DANGEROUS ELECTRIC SHOCK, DO NOT PERFORM ANY SERVICING UNLESS QUALIFIED TO DO SO.

SERVICIO

LAS INSTRUCCIONES DE SERVICIO SON PARA USO EXCLUSIVO DEL PERSONAL DE SERVICIO CAPACITADO. PARA EVITAR EL PELIGRO DE DESCARGAS ELÉCTRICAS, NO REALICE NINGÚN SERVICIO A MENOS QUE ESTÉ CAPACITADO PARA HACERIO.

WARTUNG

ANWEISUNGEN FÜR DIE WARTUNG DES GERÄTES GELTEN NUR FÜR GESCHULTES FACHPERSONAL.

ZUR VERMEIDUNG GEFÄHRLICHE, ELEKTRISCHE SCHOCKS, SIND WARTUNGSARBEITEN AUSSCHLIEßLICH VON QUALIFIZIERTEM SERVICEPERSONAL DURCHZUFÜHREN.

ENTRENTIEN

L'EMPLOI DES INSTRUCTIONS D'ENTRETIEN DOIT ÊTRE RÉSERVÉ AU PERSONNEL FORMÉ AUX OPÉRATIONS D'ENTRETIEN. POUR PRÉVENIR UN CHOC ÉLECTRIQUE DANGEREUX, NE PAS EFFECTUER D'ENTRETIEN SI L'ON N'A PAS ÉTÉ QUALIFIÉ POUR CE FAIRE.

ASSISTENZA TECNICA

LE ISTRUZIONI RELATIVE ALL'ASSISTENZA SONO PREVISTE ESCLUSIVAMENTE PER IL PERSONALE OPPORTUNAMENTE ADDESTRATO. PER EVITARE PERICOLOSE SCOSSE ELETTRICHE NON EFFETTUARRE ALCUNA RIPARAZIONE A MENO CHE QUALIFICATI A FARLA.

About This Manual

This manual covers the Application Programming Interface (API) for the following models:

SH-36S-PC

Changes to this Manual

We have made every effort to ensure this manual is accurate. If you discover any errors, or if you have suggestions for improving this manual, please send your comments to our Solon, Ohio factory. This manual may be periodically updated. When inquiring about updates to this manual refer to the part number and revision on the title page.

Chapter Layout

Introduction — Describes the features of the SignalHawk Application Programming Interface (API).

Reference — Describes and gives an overview of the programming protocols and language.

Table of Contents

Chapter 1 Introduction

This document describes an Application Programming Interface (API) for the Bird PC Hawk spectrum analyzer.

The API is written in C and provides a C calling interface to a Windows DLL. The API for Windows has groups of functions distinguished by their prefixes as follows:

- sh... for general management and control functions.
- shSa... for functions specific to Spectrum Analyzer measurements.

The general outline of the steps required to perform a sweep are as follows:

1. Initialize the API at application startup.
2. Establish a connection to the PC Hawk.
3. Set the parameters for a measurement sweep.
4. Start the sweep.
5. Get the data from the sweep.
6. Stop the sweep.
7. Disconnect from the PC Hawk.
8. Finalize the API on application exit.

There are several C and Visual Basic sample projects provided that demonstrate these operations. Each sample projects sets up a measurement using the PC Hawk and the API, acquires data and writes the data to the console.

Chapter 2 Reference

General Types

shConnection_t Structure

This structure is used to represent a connection that has been established with a particular Signal Hawk RF Module. The API assigns the handle. User applications must use a valid shConnection_t object to communicate with specific PC Hawk devices.

shConnection_t Structure Members

Device	A handle used by the API to reference a specific PC Hawk inside the API.

shDevices_t Structure

This structure contains information about connected Signal Hawk devices. The API assigns the fields of the structure. The API will populate information for up to MAX_DEVICES PC Hawks. MAX_DEVICES is currently 256. The structure is not needed after a connection is established.

shDevices_t Structure Members

NumDevices	The number of PC Hawks found connected to the USB.
SerialNumber[MAX_DEVICES][]	Array of character strings containing the serial numbers for the detected PC Hawk devices.

shDllProperties_t Structure

This structure contains information about the PC Hawk API revision.

shDllProperties_t Structure Members

BirdShVersion[256]	A character string indicating the revision level of the API.

shErrorData Structure

This structure contains information about the most recent error encountered by the API.

shErrorData Structure Members

ErrorCode	Numeric code for the error.
ErrorMessage[MAX_ERROR_MES_SIZE]	Short message of type TCHAR, describing the error.

General Functions

shInitialize

This function initializes the API. This must be called before any other API function.

DWORD shInitialize()

Arguments

None.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INITIALIZE_USB_LIBRARY_FAILED	The USB driver library failed to load.
Others	Call shGetLastError() for information on the specific error.

shFinalize

This function releases resources used by the API.

DWORD shFinalize()

Arguments

None.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Finalize was successful.
SH_ERROR_FINALIZE_FAILED	There was an error while cleaning up the USB library.
Other	Call shGetLastError() for information on the specific error.

shConnectDefault

This function establishes a connection with the first Signal Hawk RF Module found.

int BIRD_SH_API shConnectDefault(shConnection_t* Connection);

Arguments

Connection	Pointer to a structure of type shConnection_t to hold the context for the connection.

Returns

Note: *Returns an error code indicating the status of the operation:*

ERROR_SUCCESS	Connection was successful.
SH_ERROR_INVALID_ARGUMENT	One or more of the arguments was not valid.
SH_ERROR_DEVICE_ALREADY_CONNECTED	The device is already connected.
SH_ERROR_NO_DEVICE_ATTACHED	The driver did not find any devices.
SH_ERROR_USB_LIBRARY_NOT_INITIALIZED	The USB driver library has not been initialized. Call shInitialize() to load and initialize the USB libraries before using other functions of the API.
SH_ERROR_USB_CONNECTION_FAILED	There was a problem with the USB driver.
Other	Call shGetLastError() for information on the specific error.

Remarks

Call shDisconnect(…) to release the connection.

shGetDeviceList

This function returns a list of PC Hawk devices physically attached to the PC.

int BIRD_SH_API 3.2.4 shGetDeviceList(shDevices_t* Devices);

Arguments

Devices	Pointer to a structure of type shDevices_t to hold the list of serial numbers and the number of devices attached to the PC. The API populates the structure.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Operation was successful.
SH_ERROR_DEVICE_ ALREADY_CONNECTED	The device is already connected.
SH_ERROR_NO_DEVICE_ ATTACHED	The driver did not find any devices.
SH_ERROR_USB_LIBRARY_ NOT_INITIALIZED	The USB driver library has not been initialized. Call shInitialize() to load and initialize the USB libraries before using other functions of the API.
Other	Call shGetLastError() for information on the specific error.

Remarks

Call shConnect(…) to connect to one of the devices. Only one device may be connected at a time.

shConnect

This function establishes a connection with the specified Signal Hawk RF Module.

```
int BIRD_SH_API shConnect(
        shConnection_t* Connection,
        char* DeviceName
        );
```

Arguments

Connection	Pointer to a structure of type shConnection_t to hold the context for the connection.
DeviceName	A character string containing the serial number of the specifed PC Hawk.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Connection was successful.
SH_ERROR_INVALID_ARGUMENT	One or more of the arguments was not valid.
SH_ERROR_DEVICE_ALREADY_CONNECTED	The device is already connected.
SH_ERROR_NO_DEVICE_ATTACHED	The driver did not find any devices.
SH_ERROR_USB_SELECTED_DEVICE_NOT_FOUND	The driver did not find the requested device.
SH_ERROR_USB_LIBRARY_NOT_INITIALIZED	The USB driver library has not been initialized. Call shInitialize() to load and initialize the USB libraries before using other functions of the API.
SH_ERROR_USB_CONNECTION_FAILED	There was a problem with the USB driver.
Other	Call shGetLastError() for information on the specific error.

Remarks

Call shGetDeviceList(…) to get a list of available PC Hawk devices. The shDevices_t structure returned by shGetDeviceList() contains a list of serial numbers.

Call shDisconnect(…) to release the connection.

Note: *Only one device can be connected at a time.*

shDisconnect

This function disconnects from the Signal Hawk RF Module.

int BIRD_SH_API shDisconnect(shConnection_t* Connection);

Arguments

Connection	Pointer to a structure of type shConnection_t to hold the context for the connection.

Returns

Note: *Returns an error code indicating the status of the operation:*

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INVALID_ ARGUMENT	One or more of the arguments was not valid.
SH_ERROR_DEVICE_NOT_ CONNECTED	No device is connected.
SH_ERROR_SELECTED_ DEVICE_NOT_CONNECTED	The specified device is not connected.
Others	Call shGetLastError() for information on the specific error.

shGetLastError

This function returns information about the most recent error encountered by the API. Use this function to get a text string description of the error.

int BIRD_SH_API shGetLastError(shErrorData* Error);

Arguments

Error Pointer to a data structure of type shErrorData.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INVALID_ ARGUMENT	One or more of the arguments was not valid.

shGetDllProperties

This function returns information about the PC Hawk API.

```
int BIRD_SH_API shGetDllProperties(
        shDllProperties_t* DllProperties
        );
```

Arguments

DllProperties	Pointer to a structure of type shDllProperties_t that will be used to return information about the DLL version.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INVALID_ ARGUMENT	One or more of the arguments was not valid.

Spectrum Analyzer Types

shSaParameters_t Structure

Contains structures used to configure the various measurements of the PC Hawk. MeasureId, SaConfigParameters, SaTriggerParameters are required for all measurements. The remaining structures are measurement specific and can be ignored if you are not using the measurements they apply to.

Field	Comments
MeasureId	Enumerated int of type shSaMeasureId indicating the type of measurement.
SaConfigParameters	Structure of type shSaConfig Parameters containing the parameters for a standard spectrum analyzer measurement
SaTriggerParameters	Structure of type shSaTrigger Parameters containing the trigger parameters.
OccupiedBandwidth	Structure of type shSa Occupied BandwidthConfig containing the parameters for a Occupied Bandwidth measurement
ChannelPower	Structure of type shSaChannelConfig containing the parameters for a Channel Power measurement
AdjacentChannel	Structure of type shSa Adjacent Channel Config containing the parameters for an Adjacent Channel Power measurement
ZeroSpan	Structure of type shSa Zero Span Config containing the parameters for a zero span (time domain) measurement
Counters	Structure of type shSa Counters Config containing the Frequency counter parameters
DemodSignal	Structure of type shSa Demod Signal Config containing the parameters for a demodulation measurement

shSaConfigParameters Structure

CenterFrequency	Unsigned 64 bit variable (type Frequency) Specifies the center frequency in Hz
FrequencySpan	Unsigned 64 bit variable (type Frequency) Specifies the frequency span in Hz
VideoBandwidth	Enumerated integer of type shSaBwSettings indicating the video bandwidth setting.
ResolutionBandwidth	Enumerated integer of type shSaBwSettings indicating the resolution bandwidth setting.
DetectorMode	Enumerated integer of type shSaDetectorModes indicating the detector mode.
PreampAttenuation	Enumerated integer of type shSaPreampAttenSettings indicating the preamp/attenuation combination setting. The attenuation and preamp cannot be independently set.

shSaTriggerParameters Structure

TriggerMode	Enumerated integer of type shSaTriggerModes indicating the trigger mode used for the measurement.
TriggerEvent	Enumerated integer of type shSaTriggerEvents indicating the trigger event (edge or level) for the measurement.
PowerLevel	Specifies the power level (of type float) for the video trigger modes
GateDelay	Specifies the gate delay (of type float) following a trigger event.

shSaOccupiedBWType Enumeration

The occupied bandwidth type is an enumerated list of valid settings for the type.

OccBWPercent	The occupied BW level is specified as a percentage of total integrated power.
OccBWdBc	The occupied BW level is specified in decibels relative to the total integrated power.

shSaOccupiedBandwidthConfig Structure

OccupiedBandwidthType	Enumerated integer of type shSaOccupiedBWType indicating the method percent or dBc) used to set the power level for the measurement.
OccupiedBandwidthLevel	Specifies the power level used to calculate the occupied bandwidth.

shSaChannelConfig Structure

ChannelWidth	Unsigned 64 bit variable (type Frequency) Specifies the channel width in Hz.

shSaAdjacentChannelConfig Structure

ChannelWidth	Unsigned 64 bit variable (type Frequency) Specifies the channel width in Hz.
AdjacentChannelWidth	Unsigned 64 bit variable (type Frequency) Specifies the adjacent channel width in Hz.
ChannelSpacing	Unsigned 64 bit variable (type Frequency) Specifies the channel spacing in Hz. Spacing is measured from channel center to adjacent channel center.

shSaDemodSignalConfig Structure

ModulationType	Enumerated integer of type shSaDemod Bands indicating the modulation type for demodulation measurements.

shSaZeroSpanConfig Structure

SweepTime	Unsigned long integer specifying the sweep time for the zero span (time domain) measurement. The sweep time determines the effective sample interval for the measurement. The sample interval is given as SweepTime/(SA_MAX_DATA_SIZE-1)

shSaCountersConfig Structure

CountersEnable[SA_MAX_COUNTERS]	Array of type boolean. The array contains enable state for up to MAX_COUNTERS frequency counter measurements.
CountersFrequency[SA_MAX_COUNTERS]	Array of type Frequency (unsigned 64 bit integer). The array contains the reference frequency (in Hz) for as many as MAX_COUNTERS frequency counter measurements.

shSaData_t Structure

MeasureId	Enumerated integer of type shSaMeasureId indicating the type of measurement. The measurement type determines which data member structures contain useful information. See the remarks below for more information.
SweepCount	Unsigned 16 bit value (WORD). See below for how to interpret this value.
AdcSaturation	Boolean indicating that the analog to digital converter in the PC Hawk saturated during the measurement. Under these conditions the measurement may be corrupt, and should be discarded.
ExtraMeasurementValid	Boolean indicating that one of the additional measurement structures contains valid data. The additional measurements are OccupiedBandwidthData, ChannelData, AdjacentChannelData, & CountersData.
CountersEnabled	Boolean variable indicating if Frequency counter measurements are enabled. Note that this does not enable any specific FC measurement. See shSaCounter Parameters_t for more detail.
FreqData[SA_MAX_DATA_ SIZE]	Array of type FREQUENCY (unsigned 64 bit) containing the frequencies in Hz corresponding to the spectrum analyzer measurement data.
TimeData[SA_MAX_DATA_SIZE]	Array of type float containing Time Domain measurement data.
SweepData[SA_MAX_DATA_SIZE]	Array of type float containing the Spectrum Analyzer measurement data.
OccupiedBandwidthData	Structure of type shSaOccupied BandwidthData containing the Occupied Bandwidth measurement data.
ChannelData	Structure of type shSaChannelData containing the Channel Power measurement data.

AdjacentChannelData	Structure of type shSaAdjacent ChannelData containing the Adjacent Channel Power measurement data.
CountersData	Structure of type shSaCountersData containing the Frequency Counter measurement data.

Measurement types

The shSaData_t structure contains members for all the possible measurement types in the PC Hawk. Only the members relevant to the specified measurement type will contain valid data. The type and valid data structures is shown here.

Measurement ID	Valid Measurement Data Structures
SaSpecAnId	FreqData[SA_MAX_DATA_SIZE] SweepData[SA_MAX_DATA_SIZE]
SaOccupiedBandwidthId	FreqData[SA_MAX_DATA_SIZE] SweepData[SA_MAX_DATA_SIZE] OccupiedBandwidthData
SaChannelPowerId	FreqData[SA_MAX_DATA_SIZE] SweepData[SA_MAX_DATA_SIZE] ChannelData
SaAdjacentChannelId	FreqData[SA_MAX_DATA_SIZE] SweepData[SA_MAX_DATA_SIZE] AdjacentChannelData
SaZeroSpanId	TimeData[SA_MAX_DATA_SIZE
SaCountersId	FreqData[SA_MAX_DATA_SIZE] SweepData[SA_MAX_DATA_SIZE] CountersData
SaDemodSignalId	None. See shSaDemodSignalData for information on demodulation data structures.

SweepCount

SweepCount is used to provide information about the data set when a timeout occurs while waiting for data. Under normal conditions, the API will always return complete measurements and SweepCount = 705. However, if the requested timeout value passed in shSaGetData() is too short, shSaGetData() will return an ERROR_TIMEOUT error. In this case, it is possible that the PC Hawk has delivered a part of a sweep.

SweepCount will indicate the number of data samples since the start of the sweep.

Subsequent calls to shSaGetData() will accumulate more of the data associated with the sweep until either the sweep is complete or shSaStop() is called to stop the sweep.

It is recommended that shSaGetData be called with a timeout value long enough to accumulate the entire sweep. However, these details allow shSaGetData to be called repeatedly until all the data has been acquired. See the DoSaSweep() function in the DefaultSaMeasureConsole sample project for an example of this use of shSaGetData().

shSaOccupiedBandwidthData Structure

OccupiedBandwidth	Variable of type Frequency (unsigned 64 bit) indicating the Occupied Bandwidth measurement result in Hz.

shSaChannelData Structure

ChannelPower	Variable of type float indicating the Channel Power in dBm.

shSaAdjacentChannelData Structure

ChannelPower	Variable of type float indicating the channel power in dBm.
AdjacentUpperPower	Variable of type float indicating the upper adjacent channel power in dBm.
AdjacentLowerPower	Variable of type float indicating the lower adjacent channel power in dBm.
AdjacentUpperRatio	Variable of type float indicating the upper adjacent channel ratio.
AdjacentLowerRatio	Variable of type float indicating the lower adjacent channel ratio.

shSaCountersData Structure

CountersEnabled [SA_MAX_COUNTERS]	Array of type boolean. The array indicates the enable state for all of the counters.
CountersFrequency [SA_MAX_COUNTERS]	Array of type Frequency (unsigned 64 bit integer). The array contains the measured frequency (in Hz) for all of the enabled counters.
CountersPower [SA_MAX_COUNTERS]	Array of type float. The array contains the power at the measured frequency for all of the enabled counters.

shSaDemodData_t Structure

DemodDataCount	Variable of type WORD indicating the number of samples in the demodulated data array.
DemodData [SA_MAX_DEMOD_SIZE]	Array of type short containing the demodulated sample sequence. This is audio data suitable for playing on a CODEC or otherwise formatted for audio playback. The DemodSignalConsole sample project demonstrates playing the demodulation data.

shSaMeasureId Enumeration

This enumeration specifies the type of measurement as follows:

SaSpecAnId
SaOccupiedBandwidthId
SaChannelPowerId
SaAdjacentChannelId
SaZeroSpanId
SaCountersId
SaDemodSignalId

shSaPreampAttenSettings Enumeration

This enumeration specifies the settings for the attenuator & preamp as follows:

PreampOnAtten00dB
PreampOffAtten00dB
PreampOffAtten10dB
PreampOffAtten20dB
PreampOffAtten30dB

shSaBwSettings Enumeration

This enumeration specifies the settings for Resolution BandWidth or Video BandWidth as follows:

Bw1MHz
Bw300kHz
Bw100kHz
Bw30kHz
Bw10kHz
Bw3kHz
Bw1kHz
Bw300Hz
Bw100Hz
Bw30Hz
Bw10Hz

shSaDetectorModes enumeration

This enumeration specifies the mode for the detector as follows:

DetectorAverage
DetectorSample
DetectorMin
DetectorMax

shSaTriggerEvents Enumeration

This enumeration specifies the settings for the trigger event as follows:

TriggerEventHigh,
TriggerEventLow,
TriggerEventRise,
TriggerEventFall,
TriggerEventAnyEdge

shSaTriggerModes Enumeration

This enumeration specifies the settings for the trigger modes as follows:

TriggerModeFreeSingle,
TriggerModeExtSingle,
TriggerModeVideoSingle,
TriggerModeFreeRep,
TriggerModeExtRep,
TriggerModeVideoRep

Spectrum Analyzer Functions

shSaParametersInit

This function initializes the Spectrum Analyzer parameters structure.

```
    int BIRD_SH_API shSaParametersInit(
            shSaParameters_t* SaParameters
            );
```

Arguments

SaParameters	Pointer to a structure of type shSaParameters_t. The structure is initialized to default sweep parameters for the PC Hawk.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INVALID_ ARGUMENT	The pointer to shSaParameters_t is not valid.
Other	Call shGetLastError() for information on the specific error.

Remarks

The default values are given in the following table.

Field	Comments
MeasureId	shSaSpecAnId

saConfigParameters	
CenterFrequency	1800000000 Hz (1.8 GHz)
FrequencySpan	3600000000 Hz (3.6 GHz)
VideoBandwidth	Bw300KHz
ResolutionBandwidth	Bw300KHz
DetectorMode	DetectorAverage
PreampAttenuation	PreampOffAtten00dB

saTriggerParameters	
TriggerMode	TriggerModeFreeSingle
TriggerEvent	TriggerEventHigh
PowerLevel	0 dB
GateDelay	0 ms

OccupiedBandwidth	
OccupiedBandwidthType	OccBWPercent
OccupiedBandwidthLevel	99.0 %

ChannelPower	
ChannelWidth	360000000 Hz (360 MHz)

AdjacentChannel	
ChannelWidth	360000000 Hz (360 MHz)
AdjacentChannelWidth	360000000 Hz (360 MHz)
ChannelSpacing	360000000 Hz (360 MHz)

ZeroSpan	
SweepTime	5000 ms (5 s)

Counters	
CountersEnable[i]	false
CountersFrequency [i]	1800000000 Hz (1.8 GHz)

DemodSignal	
ModulationType	Wideband_FM_Modulation

shSaGetData

This function returns the results of a sweep.

```
int BIRD_SH_API shSaGetData(
        shConnection_t* Connection,
        shSaData_t* Data,
        WORD Timeout );
```

Arguments

Connection	Pointer to a structure of type shConnection_t which specifies a connection previously established with shConnect(…) or shConnectDefault(…).
Data	Pointer to a structure of type shSaData_t to contain the measurement data.
Timeout	Timeout value of type WORD.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	operation was successful.
SH_ERROR_DEVICE_NOT_ CONNECTED	No device is connected.
SH_ERROR_SELECTED_ DEVICE_NOT_CONNECTED	The specified device is connected.
SH_ERROR_INVALID_ ARGUMENT SH_ERROR_SA_DATA_ MISMATCH	One or more of the arguments is not valid. Requested DemodSignal Data not valid for the current SA measurement.
Others	Call shGetLastError() for information on the specific error.

Remarks

shSaGetData() only returns data for the specified measurement. shSaData_t contains structures for all the measurement types, but most of them are ignored. See the remarks on shSaData_t for more detail.

shSaGetData() does not return demodulation data. Use shSaGetDemodData() to get demodulation data.

shSaGetDemodData

This function returns the results of a demodulation measurement.

```
DWORD shSaGetDemodData(
        shConnection_t* Connection,
        shSaDemodData_t* DemodData,
        uint Timeout
        );
```

Arguments

Connection	Pointer to a structure of type shConnection_t which specifies a connection previously established with shConnect(…) or shConnectDefault(…).
DemodData	Pointer to a structure of type shSaDemodData_t to contain the demodulated data.
Timeout	Timeout value of type WORD.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	operation was successful.
SH_ERROR_INVALID_ ARGUMENT SH_ERROR_DEMOD_DATA_ MISMATCH	One or more of the arguments is not valid. Requested SA Data not valid for the current DemodSignal measurement.
Others	Call shGetLastError() for information on the specific error.

Remarks

shSaGetDemodData() only returns data for the demodulation measurement. For the other measurements use shSaGetData().

shSaStart

This function starts a measurement with the specified parameters.

```
int BIRD_SH_API shSaStart(
        shConnection_t* Connection,
        shSaParameters_t* Parameters
        );
```

Arguments

Connection	Pointer to a structure of type shConnection_t which specifies a connection previously established with shConnect(…) or shConnectDefault(…).
Parameters	Pointer to a structure of type shSaParameters_t which specifies the parameters for the measurement to be made. The fields in the structure which must be specified depend on the type of measurement.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	The start operation was successful.
SH_ERROR_INVALID_ ARGUMENT	One or of the arguments was not a valid object.
SH_ERROR_DEVICE_NOT_ CONNECTED	No device is connected.
SH_ERROR_SELECTED_ DEVICE_NOT_CONNECTED	The specified device is not connected.
SH_ERROR_..._INVALID	One of the measurement parameters was out of range. See the Error Reference for more detail on all the parameter range errors.
Others	Call shGetLastError() for information on the specific error.

Remarks

shSaGetData(…)should be called to get the data for all measurements except shSaMeasurementTypes::Demodulation for which shSaGetDemodData(…) should be called.

shSaStop() should be called to terminate the operation even when in single sweep mode.

shSaStop

This function stops a sweep.

```
int BIRD_SH_API shSaStop(
        shConnection_t* Connection,
        WORD Timeout
        );
```

Arguments

Connection	Pointer to a structure of type shConnection_t to hold the context for the connection.
Timeout	Timeout interval in milliseconds.

Returns

Returns an error code indicating the status of the operation:

ERROR_SUCCESS	Initialization was successful.
SH_ERROR_INVALID_ ARGUMENT	One or more of the arguments was not valid.
SH_ERROR_DEVICE_NOT_ CONNECTED	No device is connected.
SH_ERROR_SELECTED_ DEVICE_NOT_CONNECTED	The specified device is not connected.
Others	Call shGetLastError() for information on the specific error.

Remarks

shSaStop(…) should be called to terminate all sweep operations.

Error Codes

Communication Errors

33554433 SH_ERROR_DEVICE_ALREADY_CONNECTED
SH Connect Error, Device Already Connected

33554434 SH_ERROR_NO_DEVICE_ATTACHED
SH Error, No Device Attached

33554435 SH_ERROR_INITIALIZE_USB_LIBRARY_FAILED
SH Error, Failed to Initialize USB Library

33554436 SH_ERROR_USB_LIBRARY_NOT_INITIALIZED
SH Error, USB Library Not Initialized

33554437 SH_ERROR_USB_CONNECTION_FAILED
SH Error, USB Connection Failed

33554438 SH_ERROR_BULK_READ_FAILED
SH Error, USB Bulk Read Failed

33554439 SH_ERROR_BULK_WRITE_FAILED
SH Error, USB Bulk Write Failed

33554440 SH_ERROR_CONNECT_FAILED
SH Error, Failed to Connect

33554441 SH_ERROR_DISCONNECT_FAILED
SH Error, Failed to Disconnect

33554442 SH_ERROR_DEVICE_NOT_CONNECTED
SH Error, Device Not Connected

33554443 SH_ERROR_SELECTED_DEVICE_NOT_CONNECTED
SH Error, Selected Device Not Connected

33554444 SH_ERROR_USB_SELECTED_DEVICE_NOT_FOUND
SH Error, Selected Device Not Found

Measurement Parameter Errors

33554445 SH_ERROR_MEASURE_ID_INVALID
SH Error, Measure ID Is Not Valid

33554446 SH_ERROR_ZERO_SPAN_SWEEP_TIME_INVALID
SH Error, Zero Span Sweep Time Invalid

33554447 SH_ERROR_DEMODULATED_SIGNAL_TYPE_INVALID
SH Error, DemodSignal Demodulation Type Invalid

33554448 SH_ERROR_ACP_ADJACENT_CHANNEL_WIDTH_INVALID
SH Error, Adjacent Channel Power Adjacent Channel Width Invalid

33554449 SH_ERROR_ACP_CHANNEL_WIDTH_INVALID
SH Error, Adjacent Channel Power Channel Width Invalid

33554450 SH_ERROR_ACP_CHANNEL_SPACING_INVALID
SH Error, Adjacent Channel Power Channel Spacing Invalid

33554451 SH_ERROR_CP_CHANNEL_WIDTH_INVALID
SH Error, Channel Power Channel Width Invalid

33554452 SH_ERROR_OCCUPIED_BANDWIDTH_TYPE_INVALID
SH Error, Occupied Bandwidth Type Invalid

33554453 SH_ERROR_OCCUPIED_BANDWIDTH_DB_LEVEL_INVALID
SH Error, Occupied Bandwidth dB Level Invalid

33554454 SH_ERROR_OCCUPIED_BANDWIDTH_PERCENT_LEVEL_INVALID
SH Error, Occupied Bandwidth Percent Level Invalid

33554455 SH_ERROR_CENTER_FREQUENCY_INVALID
SH Error, Center Frequency Invalid

33554456 SH_ERROR_FREQUENCY_SPAN_INVALID
SH Error, Frequency Span Invalid

33554457 SH_ERROR_VIDEO_BANDWIDTH_INVALID
SH Error, Video Bandwidth Out Invalid

33554458 SH_ERROR_RESOLUTION_BANDWIDTH_INVALID
SH Error, Resolution Bandwidth Invalid

33554459 SH_ERROR_COUNTERS_FREQUENCY_INVALID
SH Error, Counters Frequency Invalid

Control Parameter Errors

33554460 SH_ERROR_TRIGGER_GATE_DELAY_INVALID
SH Error, Trigger Gate Delay Invalid

33554461 SH_ERROR_TRIGGER_LEVEL_INVALID
SH Error, Trigger Level Invalid

33554462 SH_ERROR_TRIGGER_EVENT_INVALID
SH Error, Trigger Event Invalid

33554463 SH_ERROR_TRIGGER_MODE_INVALID
SH Error, Trigger Video Mode Invalid

33554464 SH_ERROR_ATTENUATOR_PREAMP_LEVEL_INVALID
SH Error, Attenuation/Preamp Level Invalid

33554465 SH_ERROR_DETECTOR_MODE_INVALID
SH Error, Detector Mode Level Invalid

33554466 SH_ERROR_FINALIZE_FAILED
SH Error, Failed to Finalize Device

33554467 SH_ERROR_SA_DATA_MISMATCH
SH Error, Requested Demodulation Data not valid for the current SA measurement.

33554468 SH_ERROR_DEMOD_DATA_MISMATCH
SH Error, Requested SA Data not valid for the current DemodSignal measurement.

Other Errors

33554469 SH_ERROR_INVALID_ARGUMENT
SH Error, Invalid Argument (Possibly a NULL Pointer)

33554470 SH_ERROR_UNDEFINED
SH Error, Undefined

Installation and Packaging

Bird API package is installed at C:\Program Files\Bird Technologies Group\PC SignalHawk API.

Compiler and Linker required files are:

- BirdSh.h (Compiler requirement)
- BirdShWinC.lib (Linker requirement)

Copy these files to the appropriate user’s development directory.

The BirdShWinC.dll is the runtime library required file. It can be copied either to the windows/Systems32 or to the appropriate local directory.