

Developer Instructions for 4.21 SDK

Introduction

This document describes the interface to be used by the application software to initialize and retrieve data from the InterSense devices using the InterSense library (isense.dll / libisense.so / libisense.dylib). This library and API is provided to simplify communications with all models of InterSense tracking devices. It can detect, configure, and get data from up to 8 trackers, which may have multiple stations in some cases such as the IS-900 processor. The library maintains compatibility with existing devices, and also makes the applications forward compatible with all future InterSense products. The library is intended to be backwards compatible, in the sense that

Sample Program

The library is distributed with sample programs written in C (all platforms), C# (Windows only), and Visual Basic (Windows only) to demonstrate usage. It includes a header file (isense.h) with data structure definitions and function prototypes. Most of the API description below can also be obtained from the header file. The header file is heavily commented and contains detailed information about the structures and function calls.

<code>main.c</code>	Main loop of the program. All API calls are made from here.
<code>isense.c</code>	DLL import procedures. This file is included instead of an import library to provide compatibility with all compilers, not just the VC++ 6.0.
<code>isense.h</code>	Header file containing function prototypes and definitions, some of which are only applicable to InterSense Professional Series devices and are not used with InterTrax. This file should not be modified.
<code>types.h</code>	Header file containing data type definitions.
<code>isense.dll</code>	The InterSense DLL. This file should be placed in the Windows system directory, or in the working directory of the application. The CD installation program will automatically perform this step if it is run. This is contained in subfolders based on the platform (x86_32, x86_64, or UniversalLib for Mac OS X), and has slightly different names on other platforms; libisense.so for Linux and libisense.dylib for Mac OS X.
<code>dlcompat.c</code>	Mac OS X library support code; not needed on other platforms.
<code>dlcompat.h</code>	Mac OS X library support header; not needed on other platforms.

Usage

The API provides an extensive set of functions that can read and set tracker configuration, but in its simplest form can be limited to just 4 calls, as shown below:

```
void main()
{
    ISD_TRACKER_HANDLE    handle;
    ISD_TRACKER_INFO_TYPE  tracker;
    ISD_TRACKING_DATA_TYPE data;

    handle = ISD_OpenTracker( NULL, 0, FALSE, FALSE );

    if(handle > 0)
        printf( "\n      Az      El      Rl      X      Y      Z \n" );
    else
        printf( "Tracker not found. Press any key to exit" );

    while( !kbhit() )
    {
        if(handle > 0)
        {
            ISD_GetTrackingData( handle, &data );

            printf( "%7.2f %7.2f %7.2f %7.3f %7.3f %7.3f  ",
                    data.Station[0].Euler[0],
                    data.Station[0].Euler [1],
                    data.Station[0].Euler [2],
                    data.Station[0].Position[0],
                    data.Station[0].Position[1],
                    data.Station[0].Position[2] );

            ISD_GetCommInfo( handle, &tracker );

            printf( "%5.2fKbps %d Records/s \r",
                    tracker.KBitsPerSec, tracker.RecordsPerSec );
        }
        Sleep( 6 );
    }

    ISD_CloseTracker( handle );
}
```

API

ISD_TRACKER_HANDLE

```
ISD_OpenTracker(  HWND hParent,
                  DWORD commPort,
                  Bool infoScreen,
                  Bool verbose )
```

- | | |
|------------|---|
| hParent | Handle to the parent window. This parameter is optional and should only be used if information screen or tracker configuration tools are to be used when available in the future releases. All included sample programs pass NULL. |
| commPort | If this parameter is a number other than 0, program will try to locate an InterSense tracker on the specified RS232 port. Otherwise it looks for USB device, then for serial port device on all ports at all baud rates. Most applications should pass 0 for maximum flexibility. If you have more than one InterSense device and would like to have a specific tracker, connected to a known port, initialized first, then enter the port number instead of 0. |
| infoScreen | This feature has not been implemented. Its purpose is to display an information window to show the tracker detection progress and results. Currently DLL writes only to Windows console. Most applications should pass False. |
| verbose | Pass True if you would like a more detailed report of the DLL activity. Messages are printed to Windows console. |

Bool

```
ISD_CloseTracker( ISD_TRACKER_HANDLE handle )
```

This function call de-initializes the tracker, closes communications port and frees the resources associated with this tracker. If 0 is passed, all currently open trackers are closed. When last tracker is closed, program frees the DLL. Returns FALSE if failed for any reason.

- | | |
|--------|--|
| handle | Handle to the tracking device. This is the handle returned by ISD_OpenTracker. |
|--------|--|

```
Bool  
ISD_GetTrackerConfig( ISD_TRACKER_HANDLE handle,  
                     ISD_TRACKER_INFO_TYPE *tracker,  
                     Bool verbose )
```

Get general tracker information, such as type, model, port, etc. Also retrieves genlock synchronization configuration, if available. See `ISD_TRACKER_INFO_TYPE` structure definition for complete list of items.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

tracker Pointer to a structure of type `ISD_TRACKER_INFO_TYPE`. See `isense.h` for structure definition.

```
Bool  
ISD_SetTrackerConfig( ISD_TRACKER_HANDLE handle,  
                     ISD_TRACKER_INFO_TYPE *tracker,  
                     Bool verbose )
```

When used with IS Precision Series (IS-300, IS-600, IS-900) tracking devices this function call will set genlock synchronization parameters, all other fields in the `ISD_TRACKER_INFO_TYPE` structure are for information purposes only.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

tracker Pointer to a structure of type `ISD_TRACKER_INFO_TYPE`. See `isense.h` for structure definition.

```
Bool  
ISD_GetCommInfo( ISD_TRACKER_HANDLE handle,  
                ISD_TRACKER_INFO_TYPE *tracker)
```

Get `RecordsPerSec` and `KBitsPerSec` without requesting genlock settings from the tracker. Use this instead of `ISD_GetTrackerConfig` to prevent your program from stalling while waiting for the tracker response. This call is used to obtain data rate information.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

tracker Pointer to a structure of type `ISD_TRACKER_INFO_TYPE`. See `isense.h` for structure definition.

```
Bool
ISD_SetStationConfig( ISD_TRACKER_HANDLE handle,
                     ISD_STATION_INFO_TYPE *station,
                     WORD stationID,
                     Bool verbose )
```

Configure station as specified in the `ISD_STATION_INFO_TYPE` structure. Before this function is called, all elements of the structure must be assigned valid values. General procedure for changing any setting is to first retrieve current configuration, make the change, and then apply them. Calling `ISD_GetStationConfig` is important because you only want to change some of the settings, leaving the rest unchanged.

This function is ignored if used with InterTrax30 and InterTrax2 products. InterTraxLC and InertiaCube2 only allow the `Compass` field to be changed.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

station Pointer to a structure of type `ISD_STATION_INFO_TYPE`. See `isense.h` for structure definition.

StationID Number from 1 to `ISD_MAX_STATIONS`.

```
Bool
ISD_GetStationConfig( ISD_TRACKER_HANDLE handle,
                     ISD_STATION_INFO_TYPE *station,
                     WORD stationID,
                     Bool verbose )
```

Fills the `ISD_STATION_INFO_TYPE` structure with current settings. Function requests configuration records from the tracker and waits for the response. If communications are interrupted, it will stall for several seconds while attempting to recover the settings.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

station Pointer to a structure of type `ISD_STATION_INFO_TYPE`. See `isense.h` for structure definition.

StationID Number from 1 to `ISD_MAX_STATIONS`.

Bool

```
ISD_GetTrackingData( ISD_TRACKER_HANDLE handle,  
                    ISD_TRACKING_DATA_TYPE *data )
```

Get data from all configured stations. Data is places in the `ISD_TRACKER_DATA_TYPE` structure. Orientation array may contain Euler angles or Quaternions, depending on the settings of the `AngleFormat` field of the `ISD_STATION_INFO_TYPE` structure. `TimeStamp` is only available if requested by setting `TimeStamped` field to `TRUE`. Returns `FALSE` if failed for any reason.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

data Pointer to a structure of type `ISD_TRACKING_DATA_TYPE`. See `isense.h` for structure definition. The structure is designed to accommodate InterSense Professional Series devices that support multiple sensors.

Bool

```
ISD_GetCameraData( ISD_TRACKER_HANDLE handle,  
                  ISD_CAMERA_DATA_TYPE *Data )
```

Get camera encode and other data for all configured stations. Data is places in the `ISD_CAMERA_DATA_TYPE` structure. This function does not service serial port, so `ISD_GetTrackerData` must be called prior to this.

Should only be used with IS Precision Series tracking devices, not valid and will be ignored if used with InterTrax.

handle Handle to the tracking device. This is the handle returned by `ISD_OpenTracker`.

data Pointer to a structure of `ISD_CAMERA_DATA_TYPE`. See `isense.h` for structure definition.

```
Bool  
ISD_SendScript( ISD_TRACKER_HANDLE handle,  
                char *command )
```

Send a configuration script to the tracker. Script must consist of valid commands as described in the interface protocol. Commands in the script should be terminated by the New Line character '\n'. Line Feed character '\r' is added by the function and is not required.

Should only be used with IS Precision Series tracking devices, except InertiaCube2, not valid and will be ignored if used with InterTrax.

handle Handle to the tracking device. This is the handle returned by ISD_OpenTracker.

command Pointer to a string containing the command script.

```
Bool  
ISD_NumOpenTrackers( WORD *count )
```

Number of currently opened trackers is stored in the parameter passed to this function.

```
Bool  
ISD_BoresightReferenced( ISD_TRACKER_HANDLE handle,  
                          WORD stationID,  
                          float yaw,  
                          float pitch,  
                          float roll )
```

Boresight station using specific reference angles. This is useful when you need to apply a specific offset to system output. For example, if a sensor is mounted at 40 degrees relative to the HMD, you can enter 0, 40, 0 to get the system to output zero when HMD is horizontal.

handle Handle to the tracking device. This is the handle returned by ISD_OpenTracker.

command Pointer to a string containing the command script.

stationID Number from 1 to ISD_MAX_STATIONS.

yaw, pitch, roll
 Boresight reference angles.

```
Bool  
ISD_Boresight ( ISD_TRACKER_HANDLE handle,  
                WORD stationID,  
                Bool set )
```

Boresight, or unboresight a station. If 'set' is TRUE, all angles are reset to zero. Otherwise, all boresight settings are cleared, including those set by ISD_ResetHeading and ISD_BoresightReferenced

handle Handle to the tracking device. This is the handle returned by ISD_OpenTracker.

stationID Number from 1 to ISD_MAX_STATIONS.

set TRUE or FALSE, to set to clear boresight.

```
Bool  
ISD_ResetHeading( ISD_TRACKER_HANDLE handle,  
                  WORD stationID )
```

Reset heading to zero.

handle Handle to the tracking device. This is the handle returned by ISD_OpenTracker.

stationID Number from 1 to ISD_MAX_STATIONS.

DATA STRUCTURES

ISD_TRACKER_INFO_TYPE

```
typedef struct
{
    float    LibVersion;
    DWORD    TrackerType;
    DWORD    TrackerModel;
    DWORD    Port;
    DWORD    RecordsPerSec;
    float    KBitsPerSec;
    DWORD    SyncState;
    float    SyncRate;
    DWORD    SyncPhase;
    DWORD    Interface;
    DWORD    UltTimeout;
    DWORD    UltVolume;
    DWORD    dwReserved4;
    float    FirmwareRev;
    float    fReserved2;
    float    fReserved3;
    float    fReserved4;
    Bool     LedEnable;
    Bool     bReserved2;
    Bool     bReserved3;
    Bool     bReserved4;
}
ISD_TRACKER_INFO_TYPE;
```

LibVersion

InterSense Library version.

TrackerType

One of the values defined in ISD_SYSTEM_TYPE

TrackerModel

One of the values defined in ISD_SYSTEM_MODEL

Port

Number of the hardware port the tracker is connected to. Starts with 1.

RecordsPerSec

Communications statistics.

KBitsPerSec

Communications statistics.

SyncState

Applies to IS-X Series devices only. Can be one of 4 values:

- 0 - OFF, system is in free run
- 1 - ON, hardware genlock frequency is automatically determined
- 2 - ON, hardware genlock frequency is specified by the user
- 3 - ON, no hardware signal, lock to the user specified frequency

SyncRate

Sync frequency - number of hardware sync signals per second, or, if SyncState is 3 - data record output frequency.

SyncPhase

The time within the sync period at which a data record is transmitted. The phase point is specified as a percentage of the sync period. 0% (the default) instructs the tracker to output a data record as soon as possible after the sync period begins. 100% delays the output of a record as much as possible before the next sync period begins.

Interface

Hardware interface type, as defined in `ISD_INTERFACE_TYPE`.

UltTimeout

IS-900 only, ultrasonic timeout (sampling rate).

UltVolume

IS-900 only, ultrasonic speaker volume.

FirmwareRev

Firmware revision.

LedEnable

IS-900 only, blue led on the SoniDiscs enable flag.

ISD_STATION_INFO_TYPE

This data structure is used to get and set station configuration.

```
typedef struct
{
    DWORD    ID;
    Bool     State;
    Bool     Compass;
    LONG     InertiaCube;
    DWORD    Enhancement;
    DWORD    Sensitivity;
    DWORD    Prediction;
    DWORD    AngleFormat;
    Bool     TimeStamped;
    Bool     GetInputs;
    Bool     GetEncoderData;
    Byte     CompassCompensation;
    Byte     ImuShockSuppression;
    Byte     UrmRejectionFactor;
    Byte     bReserved2;
    DWORD    CoordFrame;
    DWORD    AccelSensitivity;
    DWORD    dwReserved3;
    DWORD    dwReserved4;
    float    TipOffset[3];
    float    fReserved4;
    Bool     GetCameraData;
    Bool     GetAuxInputs;
    Bool     bReserved3;
    Bool     bReserved4;
}
ISD_STATION_INFO_TYPE;
```

ID

A unique number identifying a station. It is the same as that passed to the `ISD_SetStationState` and `ISD_GetStationState` functions and can be 1 to `ISD_MAX_STATIONS`.

State

TRUE if on, FALSE if off

Compass

0 or 2 for OFF or ON. Setting 1 is not in use and will have the same result as ON. Only available for IS-X Series devices and InertiaCube2 Pro. For all others this setting is always 2. This controls the state of the compass component of the InertiaCube. Compass is only used when station is configured for GEOS or Dual modes, in Fusion mode compass readings are not used, regardless of this setting. When station is configured for FULL compass mode, the readings produced by the magnetometers inside the InertiaCube are used as absolute reference orientation for yaw. Compass can be affected by metallic objects and electronic equipment in close proximity to the InertiaCube. If compass is OFF, no heading compensation is applied.

InertiaCube

InertiaCube associated with this station. If no InertiaCube is assigned, this number is -1. Otherwise, it is a positive number 1 to 4. Only relevant for IS-300 and IS-600 Series devices. For IS-900 it is always the same as the station number, for InterTrax and InertiaCube2 it's always 1.

Enhancement

In order to provide the best performance for a large range of various applications, three levels of perceptual enhancement are available. None of the modes introduces any additional latency. InterTrax and InertiaCube2 (not the Pro) are restricted to Mode 2.

Mode 0 provides the best accuracy. The inertial tracker uses gyros to measure angular rotation rates for computing the sensor's orientation. To compensate for the gyroscopic drift, depending on the configuration, the tracker may use accelerometers, magnetometers or SoniDiscs to measure the actual physical orientation of the sensor. That data is then used to compute the necessary correction. In Mode 0 correction adjustments are made immediately, no jitter reduction algorithms are used. This results in somewhat jumpy output (not recommended for head tracking) but with lower RMS error. Use this mode for accuracy testing or for any application that requires best accuracy.

Mode 1 provides accuracy similar to that of mode 0, with an addition of a jitter reduction algorithm. This algorithm reduces the accuracy by only a small amount and does not add any latency to the measurements.

Mode 2 is recommended for use with HMD or other immersive applications. The drift correction adjustments are made smoothly and only while the sensor is moving, so as to be transparent to the user.

Sensitivity

This setting is only used when Perceptual Enhancement Level is set to 2. It controls the minimum angular rotation rate picked up by the InertiaCube. Default is level 3. Increasing sensitivity does not increase latency during normal movements. It may, however, result in some small residual movements for a couple of seconds after the sensor has stopped. If your application requires sensitivity greater than maximum provided by this control, you must use Perceptual Enhancement level 0 or 1 instead. For InterTrax and InertiaCube2 devices this value is fixed to default and can't be changed.

Prediction

InterSense tracker models IS-300 Pro and higher can predict motion up to 50 ms into the future, which compensates for graphics rendering delays and further contributes to eliminating simulator lag. This feature is only available for stations configured with an InertiaCube. Not available on InterTrax and InertiaCube2.

AngleFormat

`ISD_EULER` or `ISD_QUATERNION`. The Euler angles are defined as rotations about Z, then Y, then X in body frame. Angles are returned in degrees. This setting effects how angles are returned from the tracker, they are converted in the DLL and both formats are always available. Default is `ISD_EULER`.

TimeStamped

`TRUE` if time stamp is requested, default is `FALSE`. Time stamps are always available with InertiaCube and PC-Tracker models, regardless of this setting.

GetInputs

`TRUE` if button and analog joystick data is requested, default is `FALSE`.

GetEncoderData

TRUE if raw encoder data is requested, default is FALSE.

CompassCompensation

This setting controls how Magnetic Environment Calibration is applied. This Calibration calculates nominal field strength and dip angle for the environment in which sensor is used. Based on these values system can assign weight to compass measurements, allowing for bad measurements to be rejected. Values from 0 to 3 are accepted. If CompassCompensation is set to 0, the calibration is ignored and all compass data is used. Higher values result in tighter rejection threshold, resulting in more measurements being rejected. If sensor is used in an environment with a lot of magnetic interference this can result in drift due to insufficient compensation from the compass data. Default setting is 2.

ImuShockSuppression

This setting controls the rejection threshold for ultrasonic measurements. Currently implemented only for PC Tracker. Default setting is 4, which results in measurements with range error greater than 4 times the average to be rejected. Please do not change this setting without consulting with InterSense technical support.

UrmRejectionFactor

This setting controls the rejection threshold for ultrasonic measurements. Currently implemented only for PC Tracker. Default setting is 4, which results in measurements with range error greater than 4 times the average to be rejected. Please do not change this setting without consulting with InterSense technical support.

AccelSensitivity

AccelSensitivity is used for 3-DOF tracking with InertiaCube products only. It controls how fast tilt correction, using accelerometers, is applied. Valid values are 1 to 4, with 2 as default. Default is best for head tracking in static environment, with user seated. Level 1 reduces the amount of tilt correction during movement. While it will prevent any effect linear accelerations may have on pitch and roll, it will also reduce stability and dynamic accuracy. It should only be used in situations when sensor is not expected to experience a lot of movement. Level 3 allows for more aggressive tilt compensation, appropriate when sensor is moved a lot, for example, when user is walking for long durations of time. Level 4 allows for even greater tilt corrections. It will reduce orientation accuracy by allowing linear accelerations to effect orientation, but increase stability. This level is appropriate for when user is running, or in other situations when sensor experiences a great deal of movement.

TipOffset

Coordinates in station frame of the point being tracked.

GetCameraData

TRUE to get computed FOV, aperture, etc. default is FALSE.

GetAuxInputs

TRUE to get values from auxiliary inputs connected to the I²C port in the MiniTrax device. Applicable to IS-900 only.

ISD_STATION_DATA_TYPE

This data structure is used to return current data for a station, including position, orientation, time stamp, button and analog channel state. It is passed to `ISD_GetTrackingData` as part of `ISD_TRACKING_DATA_TYPE`

```
typedef struct
{
    ISD_STATION_STATE_TYPE Station[ISD_MAX_STATIONS];
}
ISD_TRACKER_DATA_TYPE;
```

```
typedef struct
{
    BYTE    TrackingStatus;
    BYTE    NewData;
    BYTE    CommIntegrity;
    BYTE    BatteryState;
    float   Euler[3];
    float   Quaternion[4];
    float   Position[3];
    float   TimeStamp;
    float   StillTime;
    float   BatteryLevel;
    float   CompassYaw;
    Bool    ButtonState[MAX_NUM_BUTTONS];
    short   AnalogData[ISD_MAX_CHANNELS];
    BYTE    AuxInputs[ISD_MAX_AUX_INPUTS];
    float   AngularVelBodyFrame[3];
    float   AngularVelNavFrame[3];
    float   AccelBodyFrame[3];
    float   AccelNavFrame[3];
    float   VelocityNavFrame[3];
    float   AngularVelRaw[3];
    DWORD   Reserved[64];
}
ISD_STATION_DATA_TYPE;
```

TrackingStatus

Tracking status byte. Available only with IS-900 firmware versions 4.13 and higher, and isense.dll versions 3.54 and higher. It is a value from 0 to 255 that represents tracking quality.

NewData

TRUE if this is new data. Every time `ISD_GetData` is called this flag is reset.

CommIntegrity

Communication integrity of wireless link.

BatteryState

Wireless devices only 0=n/a, 1=low, 2=ok.

Euler

Orientation in Euler, returned in degrees.

Quaternion

Orientation in Quaternion form.

Position

Station position in meters.

TimeStamp

Only if requested, in seconds.

StillTime

InertiaCube and PC-Tracker products only.

BatteryLevel

Battery Voltage, if available.

CompassYaw

Magnetometer heading, computed based on current orientation.

ButtonState

Only if requested.

AnalogData

Only if requested. Current hardware is limited to 10 channels, only 2 are used. The only device using this is the IS-900 wand that has a built-in analog joystick. Channel 1 is x-axis rotation, channel 2 is y-axis rotation. Values are from 0 to 255, with 127 representing the center.

AuxInputs

Only if requested.

AngularVelBodyFrame

Angular rotation speed in sensor body coordinate frame. This is the processed angular rate, with current biases removed, rad/sec. This is the angular rate used to produce orientation updates.

AngularVelNavFrame

Angular rotation speed in world coordinate frame, with boresight and other transformations applied, rad/sec.

AccelBodyFrame

Acceleration in sensor body coordinate frame, meter²/sec. Only factory calibration is applied to this data, gravity component is not removed.

AccelNavFrame

Acceleration in the navigation (earth) coordinate frame, meters/sec². This is the accelerometer measurements with calibration, and current sensor orientation applied, and gravity subtracted. This is the best available estimate of acceleration.

VelocityNavFrame

meters/sec, 6-DOF systems only.

AngularVelRaw

Raw gyro output, only factory calibration is applied. Some errors due to temperature dependant gyro bias drift will remain.