Nautiz X7

Software Development Kit

handheld

Revision History

Title		Nautiz X7 Software Development Kit Specification		
Rev	Date	Page	Sec	Description
A	9/15/2009	All	All	Initial release.

Table of Contents

1. Overview

This document describes the architecture and functionalities of the Nautiz X7 SDK (Software Development Kit). The SDK will provide a set of APIs to query E-compass, G-sensor, Pressure sensor, and GPS data.

2. System Support

Hardware

The SDK will support the following Windows Mobile device:

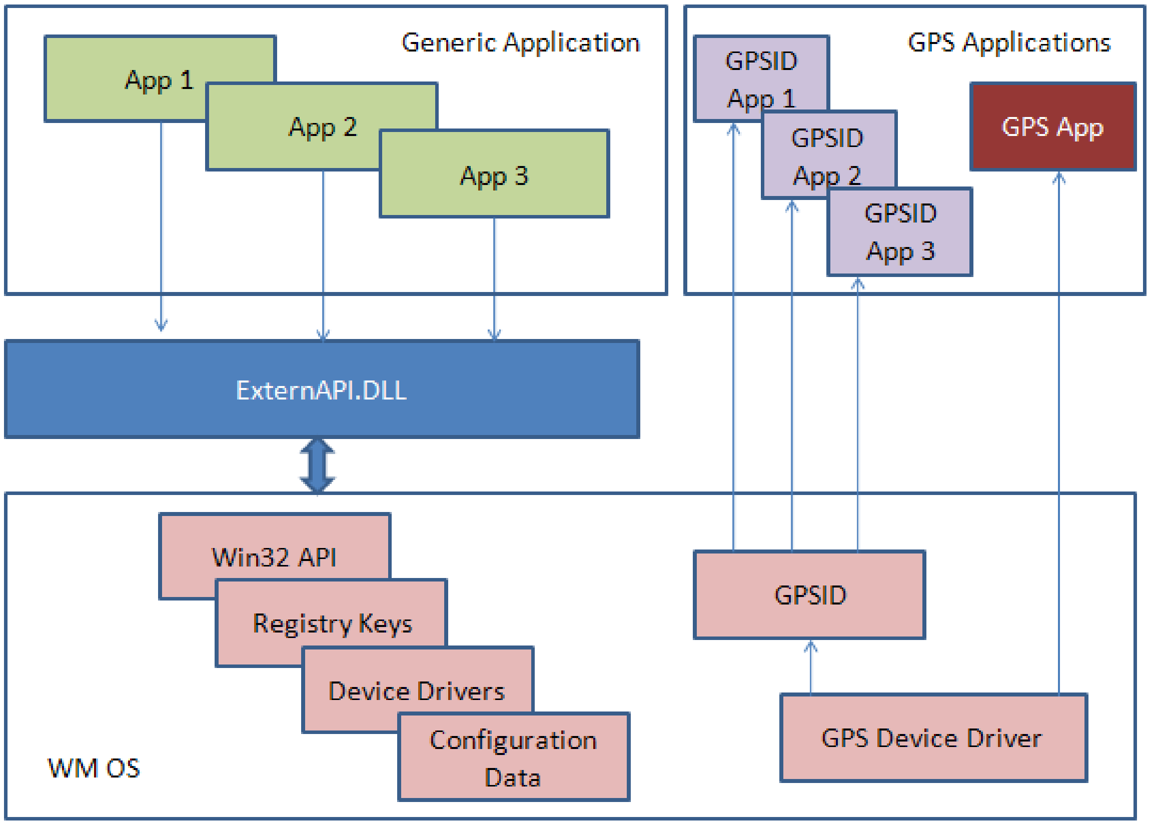
- Nautiz X7

OS

The SDK will support the following Operating Systems:

- Microsoft Windows Mobile 6.1 Professional Edition
- Microsoft Windows Mobile 6.1 Classic Edition

3. Architecture



ExternAPI.DLL

This DLL will be shipped as part of the ROM image on Nautiz X7 and it will have to support all functions referenced by this document. The header file will be published to expose all APIs defined within the DLL. The intention of this DLL is to allow other applications to query and set characteristics for theNautiz X7 hardware and software.

GPS and GPSID

The GPS Intermediate Driver that proposed by Microsoft is useful to developers writing applications that use GPS (Global Positioning System) devices as well as to GPS hardware manufacturers. It is useful because it provides an intermediate layer that abstracts the actual GPS device from developers and manufacturers.

- GPS Without the GPS Intermediate Driver

Without the GPS Intermediate Driver, applications generally access GPS hardware directly through a

COM port.

Applications interact directly with the GPS hardware by calling CreateFile to obtain a handle to the device. With most GPS devices, this connection is exclusive, so only one application at a time can interact with the GPS hardware.

Applications then call ReadFile repeatedly to retrieve GPS location data encoded using the National Marine Electronics Association (NMEA) standard or GPS raw data.

- Benefits of the GPS Intermediate Driver

With the GPS Intermediate Driver, applications use the GPS Intermediate Driver instead of interacting with the GPS hardware directly. The GPS Intermediate Driver is the only code that interacts directly with GPS hardware.

The GPS Intermediate Driver provides two main advantages:

- Enable multiple applications to use GPS hardware at the same time. The GPS Intermediate Driver makes it appear that each application has its own dedicated GPS hardware.

- Remove the need for applications to parse NMEA strings to obtain meaningful data. The GPS Intermediate Driver internally parses the NMEA strings obtained from the GPS hardware and makes the parsed information available through a friendly API that contains structures like GPS_POSITION and calls like GPSGetLocation. Applications can also use a backward-compatible stream/**ReadFile** interface. This ability provides an easy way for existing applications to use the GPS Intermediate Driver without requiring modification.

For more detail information, please refer to the Microsoft Windows Mobile 6.1 documentation.

4. Component API’s

E-compass/G-sensor API’s

EComAcc_InitDevice

Syntax

INT16 EComAcc_InitDevice(void)

Description

This function will initialize E-compass/G-sensor hardware and related software memory handling. It has to be called prior to get the data.

Parameters

None

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

EComAcc_DeinitDevice

Syntax

INT16 EcomAcc_DeinitDevice(void)

Description

This function will de-initialize E-compass/G-sensor hardware and related software memory handling. It has to be called after finishing data query procedure. Otherwise, it might have memory leakage or handle leakage problem.

Parameters

None

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

EComAcc_GetData

Syntax

BOOL EComAcc_GetData(int data[])

Description

This function will used to get E-compass/G-sensor Theta, Pitch, Roll angle data, and HDST status.

Parameters

- ✓ *data*

  [out] Pointer to an array that receives the output data for the operation.

  data[0] ➔ Theta
  data[1] ➔ Pitch
  data[2] ➔ Roll angle
  data[3] ➔ HDST, 0 and 1 means calibration is NOT completed
  2 and 3 means calibration is completed

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

Pressure sensor API's

Pressure_InitDevice

Syntax

BOOL Pressure_InitDevice(void)

Description

This function will initialize Pressure sensor hardware and related software memory handling. It has to be called prior to get the data.

Parameters

None

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

Pressure_DeinitDevice

Syntax

BOOL Pressure_DeinitDevice(void)

Description

This function will de-initialize Pressure sensor hardware and related software memory handling. It has to be called after finishing data query procedure. Otherwise, it might have memory leakage or handle leakage problem.

Parameters

None

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

Pressure_GetPaTemp

Syntax

BOOL Pressure_GetPaTemp(int data[])

Description

This function will used to get Pressure sensor PA and Temperature raw data.

Parameters

- ✓ *data*

  [out] Pointer to an array that receives the output data for the operation.

  data[0] ➔ PA raw data

  data[1] ➔ Temperature raw data

Return value

TRUE indicates success. FALSE indicates failure

Example Sample Code

GPS API's

GPS with GPS Intermediate Driver

CreateFile

Syntax

```
HANDLE CreateFile( LPCTSTR lpFileName,
                   DWORD dwDesiredAccess,
                   DWORD dwShareMode,
                   LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                   DWORD dwCreationDisposition,
                   DWORD dwFlagsAndAttributes,
                   HANDLE hTemplateFile)
```

Description

This function opens a COM port. It returns a handle to access the object.

Parameters

lpFileName

[in] Pointer to a null-terminated string that specifies the name of the object. It has to be set as GPD1 for GPS usage

dwDesiredAccess

[in] Type of access to the object. It has to be (GENERIC_READ | GENERIC_WRITE) for GPS usage.

dwShareMode

[in] Share mode for the object. It has to be (FILE_SHARE_READ | FILE_SHARE_ WRITE) for GPS usage.

lpSecurityAttributes

[in] Not used. It has to be set as NULL for GPS usage.

dwCreationDisposition

[in] Action to take on files that exist, and which action to take when files do not exist. It has to be OPEN_EXISTING for GPS usage.

dwFlagsAndAttributes

[in] File attributes and flags for the file. It has to be FILE_ATTRIBUTE_NORMAL for GPS usage.

hTemplateFile

[in] Ignored. It has to be set as NULL for GPS usage.

Return value

*An open handle to the specified file indicates success. If the specified file exists before the function call and dwCreationDisposition is set to CREATE_ALWAYS or OPEN_ALWAYS, a call to GetLastError returns ERROR_ALREADY_EXISTS, even though the function has succeeded. If the file does not exist before the call, **GetLastError** returns zero. INVALID_HANDLE_VALUE indicates failure. To get extended error information, call **GetLastError**.*

Example Sample Code

GPS without GPS Intermediate Driver

CreateFile

Syntax

```
HANDLE CreateFile( LPCTSTR lpFileName,
                   DWORD dwDesiredAccess,
                   DWORD dwShareMode,
                   LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                   DWORD dwCreationDisposition,
                   DWORD dwFlagsAndAttributes,
                   HANDLE hTemplateFile)
```

Description

This function opens a COM port. It returns a handle to access the object.

Parameters

lpFileName

[in] Pointer to a null-terminated string that specifies the name of the object. It has to be set as COM4 for GPS usage

dwDesiredAccess

[in] Type of access to the object. It has to be (GENERIC_READ | GENERIC_WRITE) for GPS usage.

dwShareMode

[in] Share mode for the object. It has to be (FILE_SHARE_READ | FILE_SHARE_WRITE) for GPS usage.

lpSecurityAttributes

[in] Not used. It has to be set as NULL for GPS usage.

dwCreationDisposition

[in] Action to take on files that exist, and which action to take when files do not exist. It has to be OPEN_EXISTING for GPS usage.

dwFlagsAndAttributes

[in] File attributes and flags for the file. It has to be FILE_ATTRIBUTE_NORMAL for GPS usage.

hTemplateFile

[in] Ignored. It has to be set as NULL for GPS usage.

Return value

An open handle to the specified file indicates success. If the specified file exists before the *function call and dwCreationDisposition is set to CREATE_ALWAYS or OPEN_ALWAYS, a call to GetLastError returns ERROR_ALREADY_EXISTS, even though the function has succeeded. If the file does not exist before the call,* ***GetLastError*** *returns zero. INVALID_HANDLE_VALUE indicates failure. To get extended error information, call* ***GetLastError****.*

Example Sample Code