ET7H16 series

Standard API User Manual

Version 1.0.3, October 2019

Service and usage information for

PET-7H16M

Written by Sean

Edited by Anna Huang

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Contact US

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

TABLE OF CONTENTS

About this Guide

This manual is intended for software developers who want to integrate the functionality of standard PET-7H16M family into their applications.

What Models are covered in this Manual?

The following models are covered in this manual:

ET7H16 series

ET7H16 series	
PET-7H16M	8 channels AI, 4 channels DO, 4 channels DI
	1 channel 32-bit High-speed counter 4 channels 32-bit Low-speed counter

Related Information

For additional information about your module that can be obtained from downloading the latest version from ICP DAS web site.

ET-7H16 series

http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/sdk/

How to contact us?

For support for this or any ICP DAS product, you can contact ICP DAS Customer Support in one of the following ways:

- Visit the ICP DAS Storage Manager technical support Web site at:
  http://www.icpdas.com/faq/faq.htm
- Submit a problem management record (PMR) electronically from our Web site at:
  http://www.icpdas.com/sevices/contact_customerservice.htm
- Send e-mail to:
  service@icpdas.com

Revision History

The table below lists the revision history.

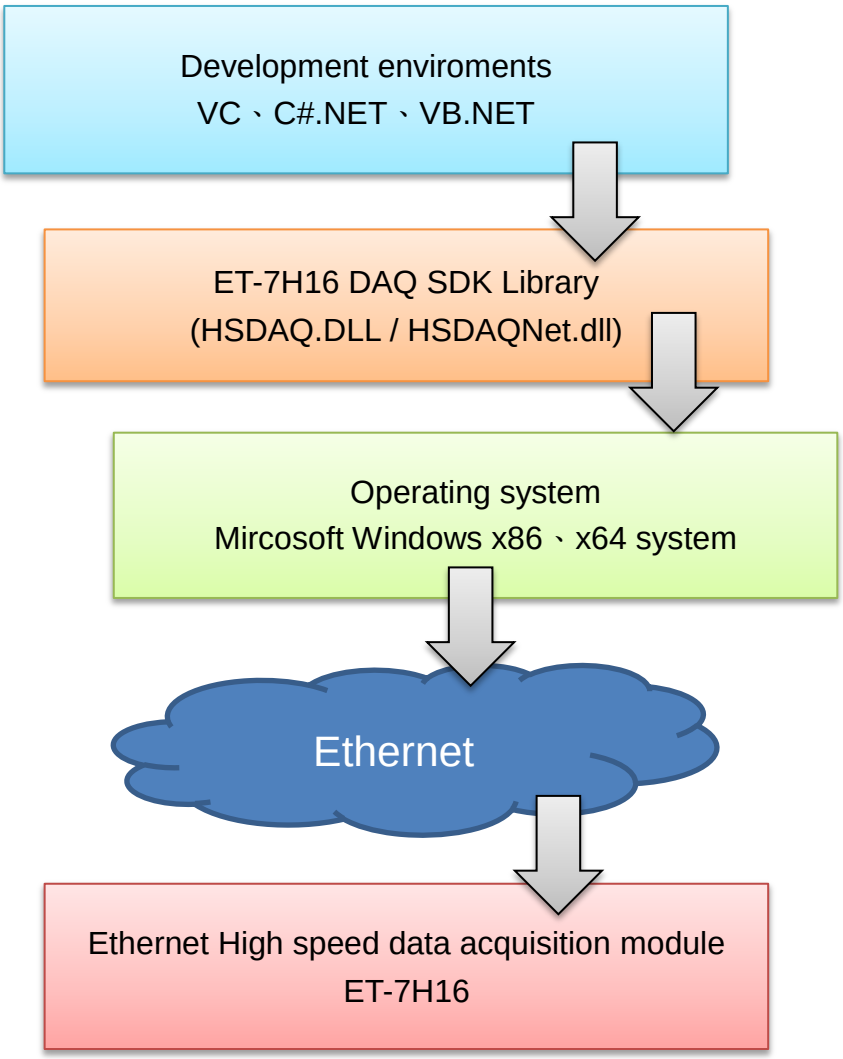
Revision	Date	Description
1.0.1	January 2019	Initial issue
1.0.2	Feburay 2019	Add API function call process chart for data logger
1.0.3	September 2019	Add API fucntions for counters and synchronous Input data acquisition

1. Getting Started

This chapter provides a guided tour that describes the steps needed to know, download, install and configure of the basic procedures for user working with the ET-7H16 for the first time.

1.1. Introducing the ET-7H16 DAQ SDK library

ET-7H16 DAQ SDK library are software development kits that contain header files, libraries, documentation and tools required to develop applications for ET-7H16 series.



The files in ET-7H16 DAQ SDK Library

File	Description
HSDAQ.dll HSDAQ.lib HSDAQ.h	Used for VC programs
HSDAQNet.dll	Used for .Net programs

Notes:

1. The **HSDAQ.dll** does not need to be registered in the system but must be placed in the VC application folder, in the PATH environmental variable, or in the Windows system directory.

The **HSDAQNet.dll** does not need to be registered in the system but must be placed in the C#/VB.Net application folder.

1.2. Getting the ET-7H16 DAQ SDK

User can get the SDK library files from FTP site, the file path is as below:

- http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/sdk/

1.3. System requirement

Minimum system requirements for ET-7H16 DAQ SDK library are given below:

- Processor - 1.4GHz or higher processor recommended
- RAM - 2GB or more of RAM is recommended
- Support for Super VGA graphics
- Hard Driver - At least 50 MB of available space (If data logger requirement, increase the disk storage space according to the size of the stored data.)
- Ethernet port/NIC (Network Interface Card)-100Mbps or 1Gbps
- Microsoft Windows 2000 or later(32-bit or 64-bit Windows Operating System)

Operating system of Microsoft Windows requirement

32-bit(x86)	64-bit(x64)
Windows 2000	Windows XP 64-bit
Windows XP 32- bit	Windows 2003 64-bit
Windows 2003 32 bit	Windows Vista 64-bit
Windows Vista 32 bit	Windows 7 64-bit
Windows 7 32 bit	Windows 2008 64-bit
Windows 2008 32 bit	Windows 8 64-bit
Windows 8 32 bit	Windows 2012 64-bit
	Windows 10 64-bit

PS： Microsoft Windows 3.1/95/98/ME/NT not supported

1.4. Application development

Provides a introduction help the user to step-by-step process to create a simple application in many development environments.

Prepare development tools

Install Microsoft Visual Studio 2005/2008 or later on PC

1.4.1. C#

Applied platforms

- PC series

Required header and library files

The following DLL files is needed to include to develop a PC application or plug-in

- HSDAQ.dll

How to create a program with HSDAQNet.dll using Visual Studio

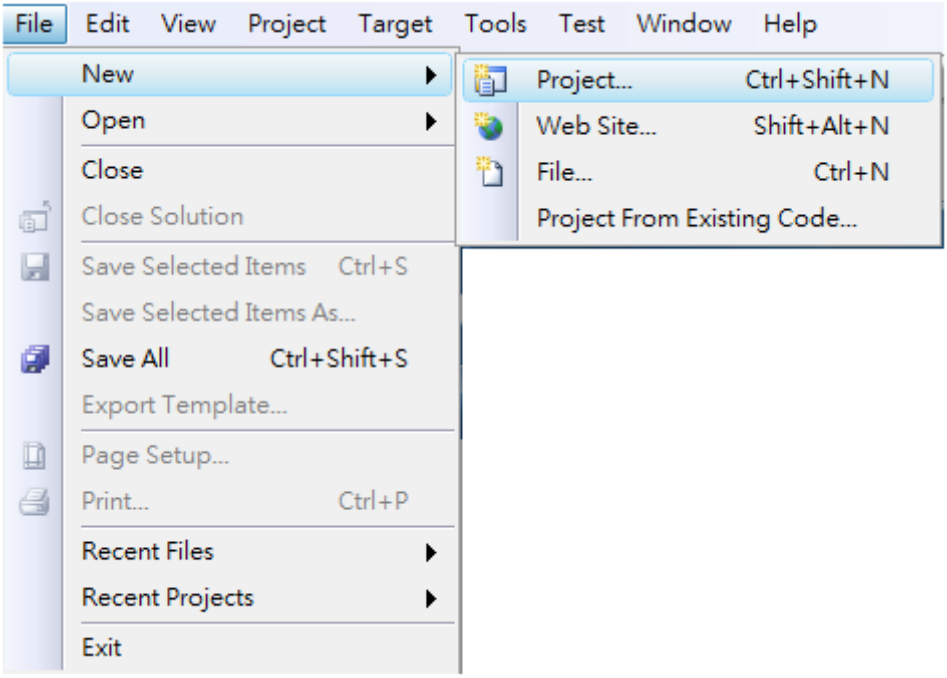
1. Using HSDAQNet.dll

HSDAQNet.dll is a .net Compact framework .

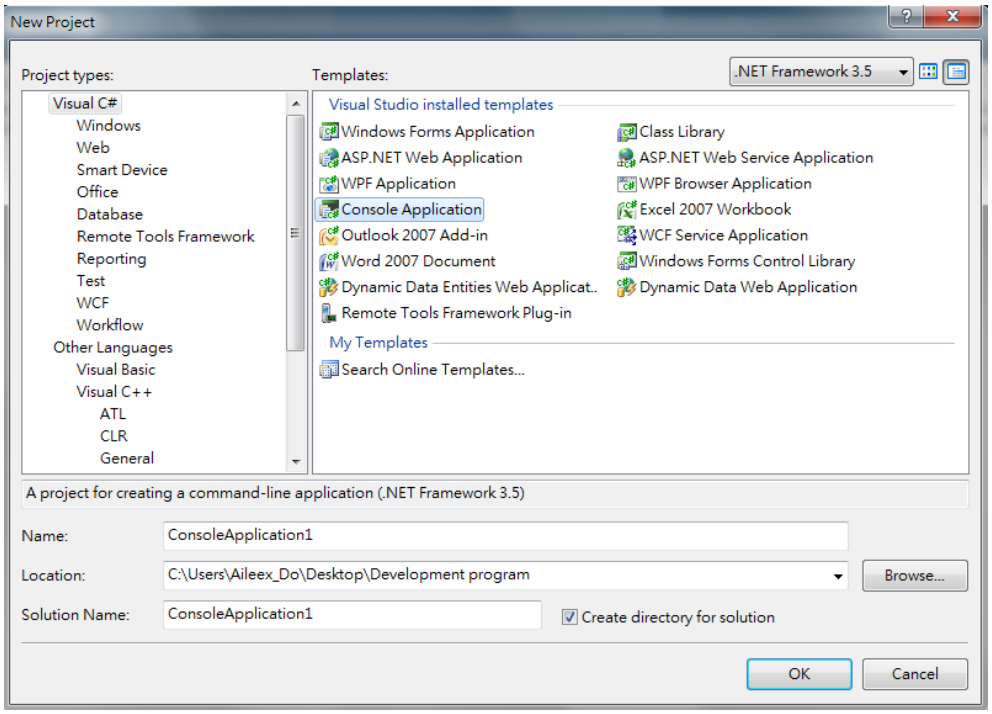
- HSDAQNet.dll (the execution file should be put in the same directory of the HSDAQNet.dll)

The latest version of this library is located at:
ftp://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/sdk/.net/

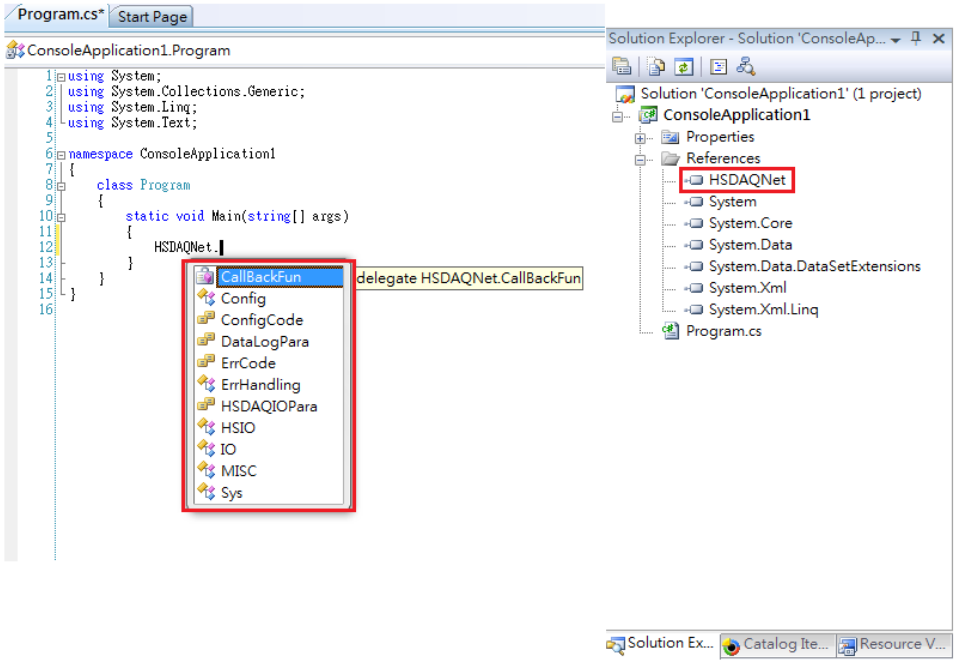
Step 1: Create a new project by using Visual Studio 2008

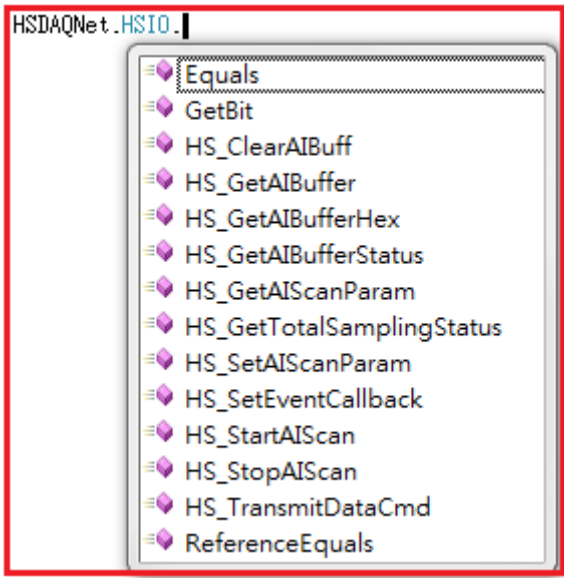


Step 2: Select Console Application



Step 3: Add the HSDAQNet.dll into the references of the project, and using HSDAQ





1.4.2. VB.NET

Applied platforms

- PC series

Required header and library files

The following DLL files is needed to include to develop a PC application or plug-in

- HSDAQ.dll

How to create a program with HSDAQNet.dll using Visual Studio

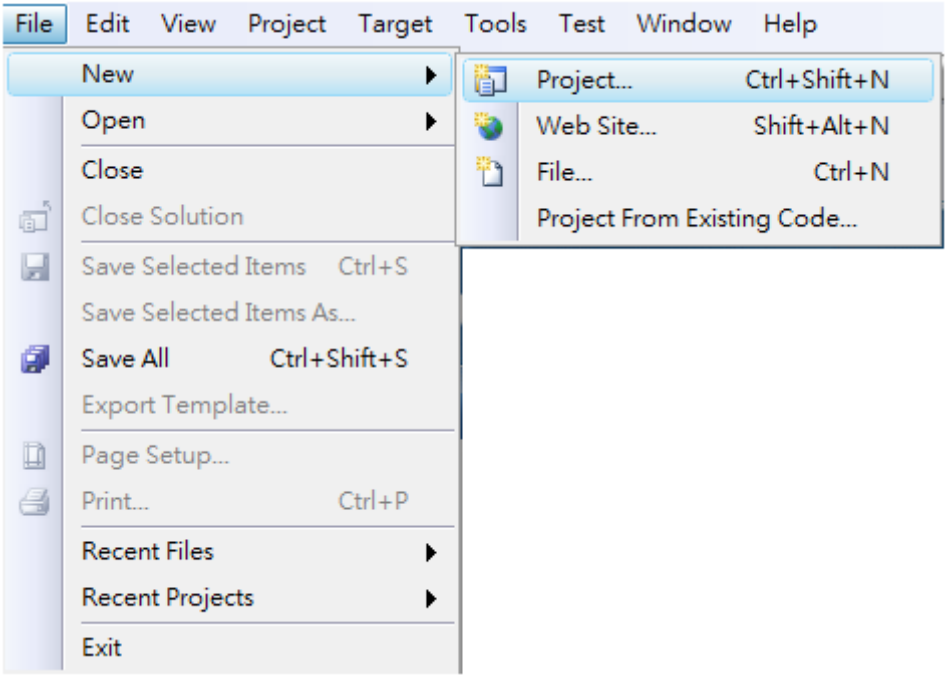
1. **Using HSDAQNet.dll**

HSDAQNet.dll is a .net Compact framework and HSDAQNet.dll isn't used for C# program but also used for VB.net program.

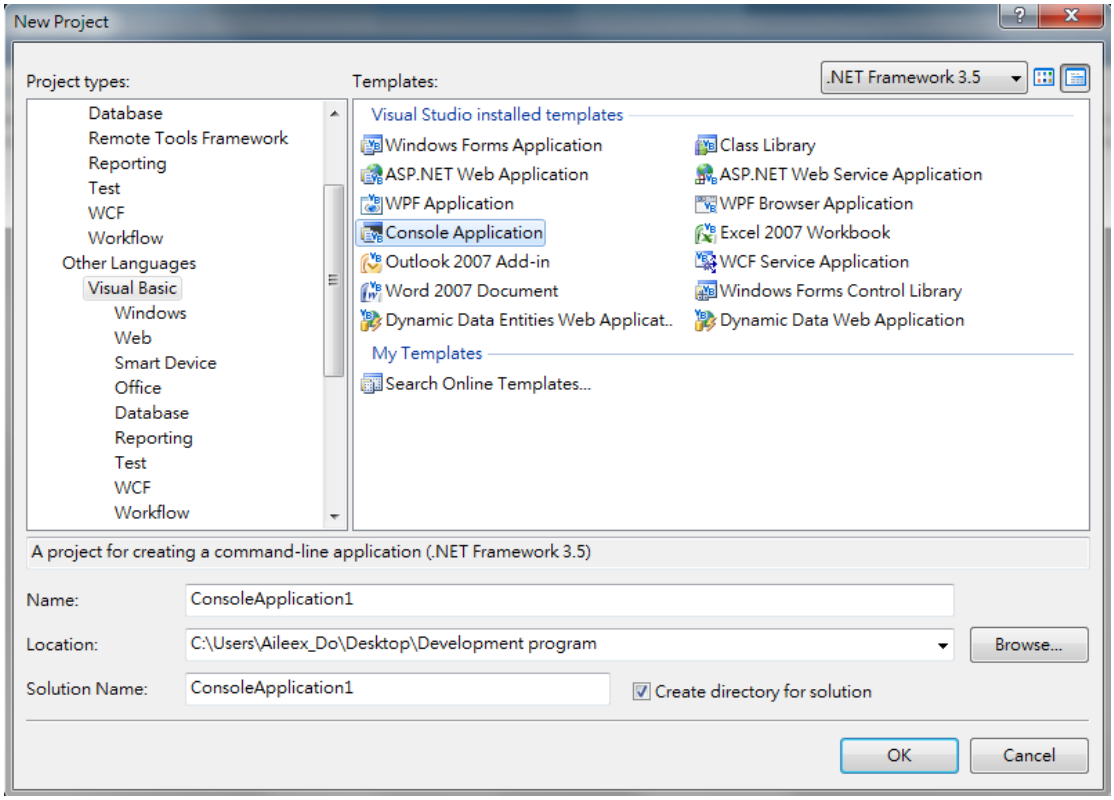
- HSDAQNet.dll (the execution file should be put in the same directory of the HSDAQNet.dll)

The latest version of this library is located at:
ftp://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/sdk/.net/

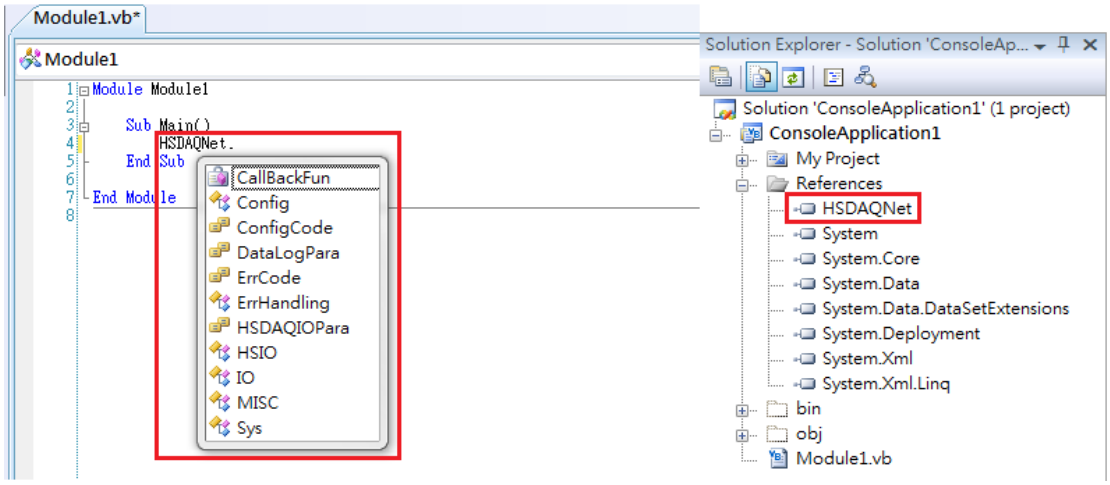
Step 1: Create a new project by using Visual Studio 2008

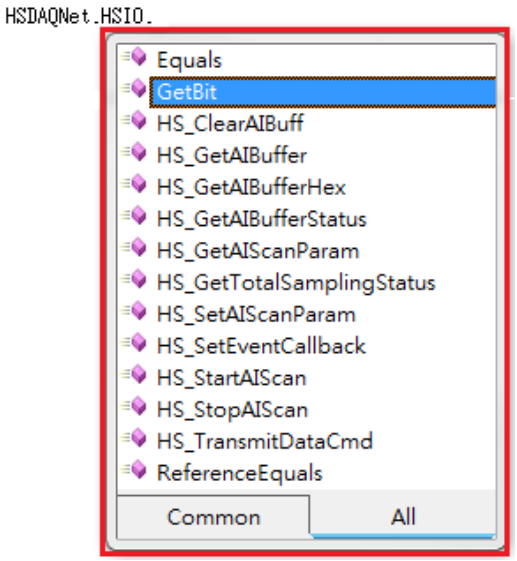


Step 2: Select VB Console Application



Step 3: Add the HSDAQNet.dll into the references of the project, and using HSDAQ





1.4.3. C/C++/MFC based on Visual Studio

Applied platforms

- PC series

Required header and library files

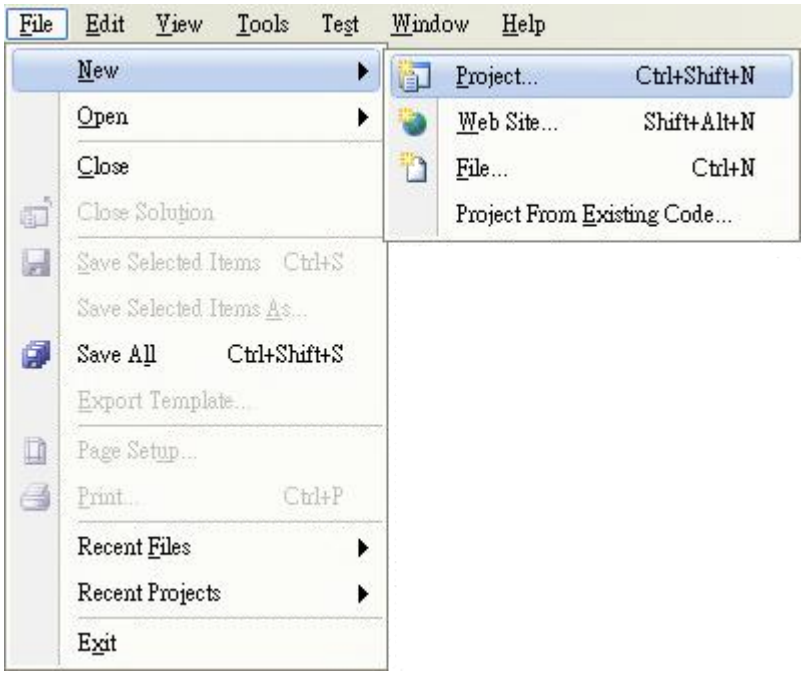
The following list lists the libraries, header files or DLL files you will need to include developing a ET-7H16 application or plug-in

- HSDAQ.h
- HSDAQ.lib

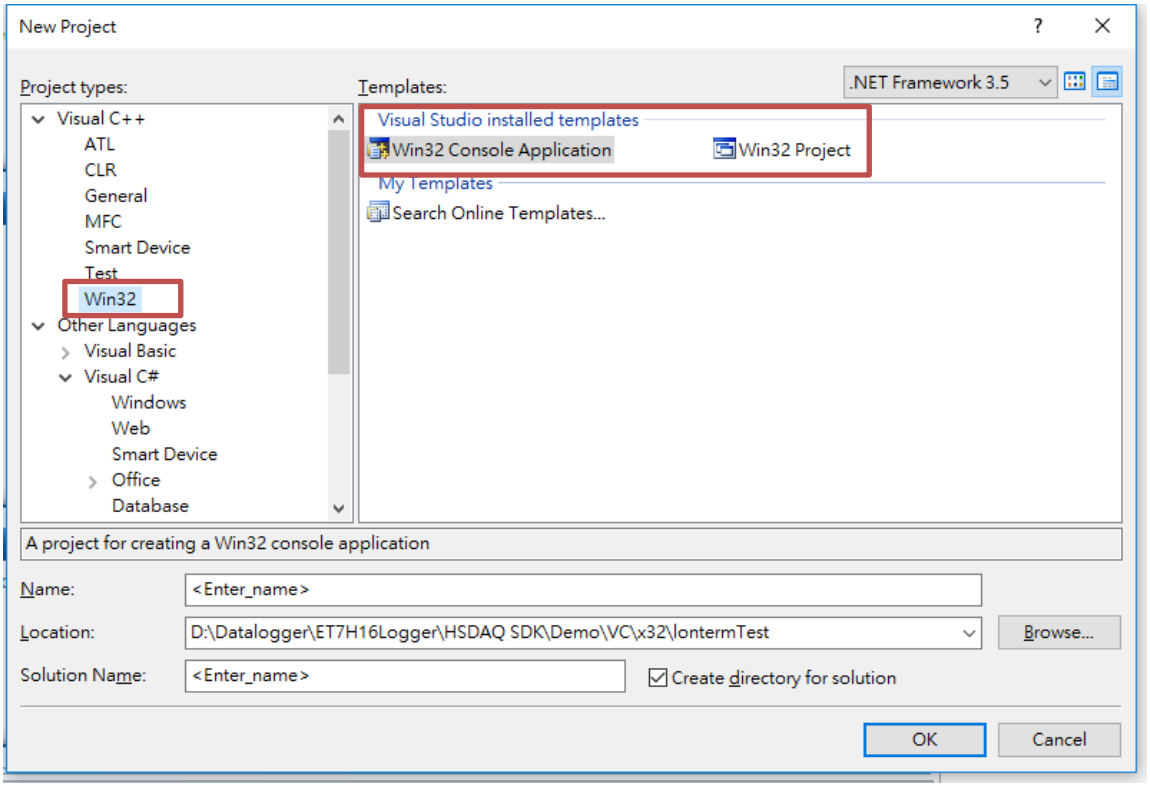
The latest version of this library is located at:
ftp://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/sdk/vc/

How to create a program with HSDAQ.dll using Visual Studio

Step 1: Create a new project by using Visual Studio 2008



Step 2: Select Win32 and Win32 Console Application or Win32 Project



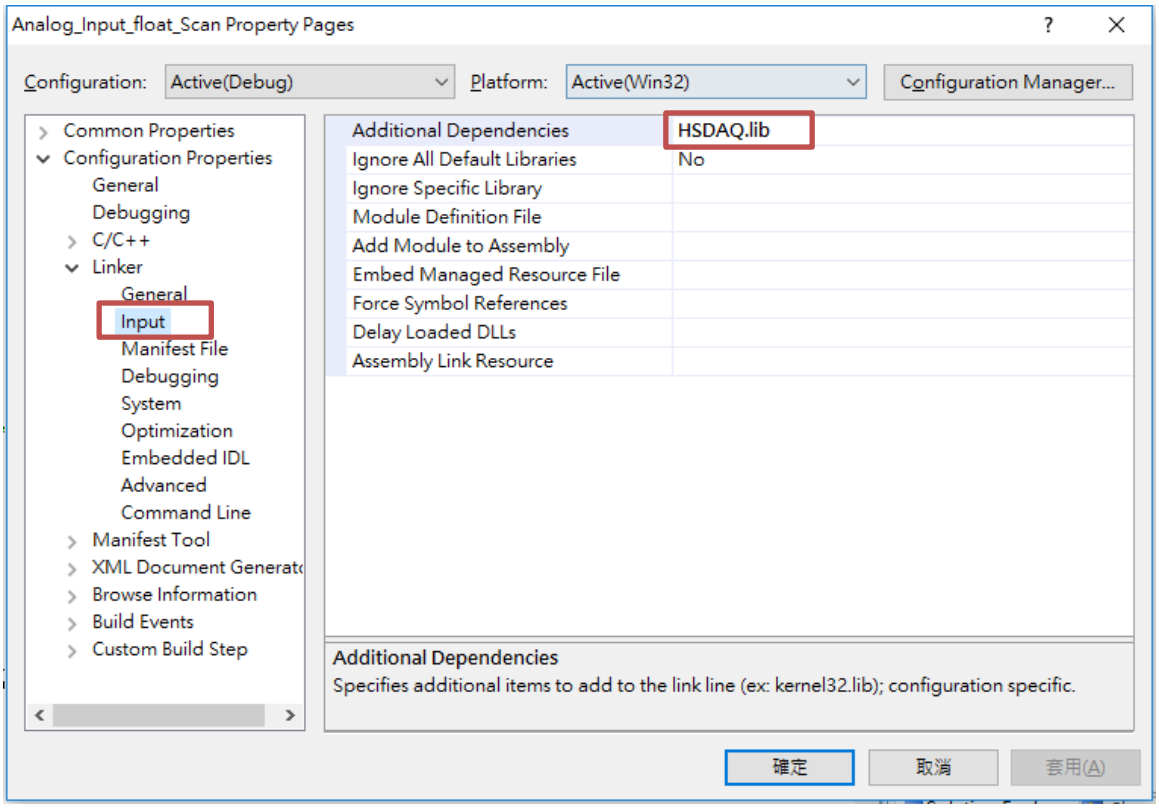
Step 3: Copy HSDAQ.h to the application folder and include it

1. Copy HSDAQ.h to the application folder.
2. Add #include “HSDAQ.h”.

```
#include "stdafx.h"
#include "HSDAQ.h"
#include <stdio.h>
#include <conio.h>
```

Step 4: Copy HSDAQ.lib to the application folder and Include it

1. Copy HSDAQ.lib to the application folder.
2. Open the project’s Property Page dialog box.
3. Click the Linker folder.
4. Click the Input property page.
5. In the right pane, type the PACSDK.lib in the Additional Dependencies item.



1.5. Demo program

➢ Visual C++ Samples

The ET-7H16 VC demo includes the following samples that demonstrate the use of the ET-7H16 Standard APIs in a Visual C++ language environment. The following samples can be found from ICP DAS web site.

For Visual C++ applications, these demo programs can be obtained from:
http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/demo/vc

➢ C# Samples

The ET-7H16 C# demo includes the following samples that demonstrate the use of the ET-7H16 Standard APIs in a C# language environment. The following samples can be found from ICP DAS web site.

For C# applications, these demo programs can be obtained from:
http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/demo/c#

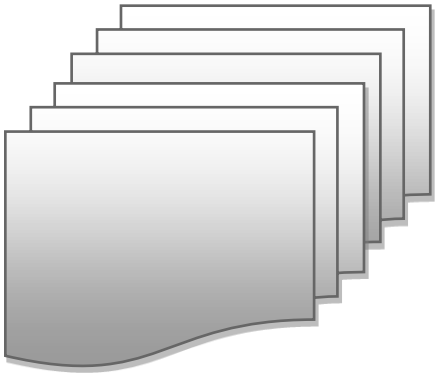
➢ VB.net Samples

The ET-7H16 VB.net demo includes the following samples that demonstrate the use of the ET-7H16 Standard APIs in a VB.net language environment. The following samples can be found from ICP DAS web site.

For C# applications, these demo programs can be obtained from:
http://ftp.icpdas.com/pub/cd/6000cd/napdos/et7h16/tools/dll/demo/vb.net

2. API Functions

The diagram below shows the set of each API provided in the ET7H16 DAQ SDK (HSDAQ.dll/ HSDAQNet.dll)



- System API
- Configration API
- IO API
- High speed IO API
- Data logger API
- Error Handling API

Function Description

Please attend the following keyword before you reading this chapter.

Function Paramter Keyword	Set a value from Parameter	Returns a value in the Parameter
[in]	Yes	No
[out]	No	Yes

2.1. System API

The system API functions and messages describe how to create/release the connection and read the information from ET-7H16 module.

Supported PACs

The following API functions are used to retrieve or set ET-7H16 module.

HSDAQ.dll Functions	HSDAQNet.dll Functions	Description
HS_Device_Create	Sys.HS_Device_Create	Create a connection to the device and initialize the deivce. This function is the driver entry.
HS_Device_Release	Sys.HS_Device_Release	Release the device from system.
HS_Reboot	Sys.Reboot	This function reboots the ET-7H16.
HS_GetFirmwareVersion	Sys.GetFirmwareVersion	Read the firmware version of ET-7H16
HS_GetSDKVersion	Sys.GetSDKVersion	This function retrieves the version number of the current HSDAQ.dll
	Sys.GeHSDAQNetVersion	This function retrieves the version number of the current HSDAQNet.dll
HS_GetHWFirmwareVersion	Sys. HS_GetHWFirmwareVersion	Read the Hardware firmware version of ET-7H16

2.1.1. HS_Device_Create

Create a connection to the device and initialize the deivce. This function must be called before calling any functions.

Syntax

C/C++

```
HANDLE HS_Devcie_Create (
   LPCSTR ConnectionString
);
```

.Net

```
IntPtr HS_Device_Create(
   string ConnectionString
}
```

Parameter

ConnectionString

[[in] Specifies the IP addresss, command port, data transmit port of ET-7H16 series

The default of command port is 9999

The default of data transmit port is 10010.

Return Values

If the function succeeds, the return value is an handle for a specified devcie.

If the function fails, the return value is NULL.To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create(“192.168.1.1”);
```

[C#]

```
IntPtr hDEv;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
```

Remark

The The following inpt formats are acceptable

HS_Device_Create(“192.168.1.10”). It indicates that a connection with IP address 192.168.1.10 and default command port 9999, data transmit port 10010 will be created.

HS_Device_Create(“192.168.1.10,9000,10011”). It indicates that a connection with IP address 192.168.1.10 and command port 9000, data transmit port 10011 will be created. (The command port and data transmit port must be changed according to the ET-7H16 setting mode).

2.1.2. HS_Device_Release

Release the device from system.It must be called after calling any functions.

Syntax

C/C++

```
bool HS_Devcie_Release (
   HANDLE hobj
);
```

.Net

```
bool HS_Device_Release (
  IntPtr hobj
)
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
if(hHS!=NULL)
{
    … // Create connection success!!
}
Else
{
    … //Create Connection fail!!
}
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
if (hHS != IntPtr.Zero)
{
    … // Create connection success!!
}
Else
{
    … //Create Connection fail!!
}
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.1.3. HS_Reboot

This function reboots the ET-7H16..

Syntax

C/C++

```
bool HS_Reboot (
   HANDLE hobj
);
```

.Net

```
bool Reboot (
  IntPtr hobj
)
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create(“192.168.1.1”);
HS_Reboot(hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.Sys.Reboot(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.1.4. HS_GetFirmwareVersion

Read the firmware version of ET-7H16.

Syntax

C/C++

```
bool HS_GetFirmwareVersion (
  HANDLE hobj,
  LPSTR version
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

version

[out] The version number of ET-7H16 firmware.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Syntax

.Net

```
string GetFirmwareVersion(IntPtr hobj);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create.

Return Values

If the function succeeds, the return string is te version number of ET-7H16 firmware.If the function fails, the return value is NULL. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
char fw_version[32]={0};
hHS = HS_Device_Create("192.168.1.1");
HS_GetFirmwareVersion (hHS, fw_version);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
string fwVersion=HSDAQNet.Sys.GetFirmwareVersion(hHS).ToString();
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.1.5. HS_GetSDKVersion

This function retrieves the version number of the current HSDAQ.dll.

Syntax

C/C++

```
bool HS_GetSDKVersion (
   LPSTR sdk_version
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

sdk_version

[out] The version number of HSDAQ.dll

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Syntax

.Net

```
string GetSDKVersion ();
```

Parameter

Nono.

Return Values

If the function succeeds, the return string is the version number of HSDAQ.dll

If the function fails, the return value is NULL. To get extended error information, call HS_GetLastError.

Examples

[C]

```
char sdk_version [32]={0};
HS_HS_GetSDKVersion(fw_version);
```

[C#]

```
string version = HSDAQNet.Sys.GetSDKVersion().ToString();
```

Remark

None.

2.1.6. GetHSDAQNetVersion

This function retrieves the version number of the current HSDAQNet.dll. (**Only used for .Net program**)

Syntax

.Net

```
string GetHSDAQNetVersion();
```

Parameter

None.

Return Values

If the function succeeds, the return string is the version number of HSDAQNet.dll

If the function fails, the return value is NULL. To get extended error information, call HS_GetLastError.

Examples

[C#]

```
string version = HSDAQNet.Sys.GetHSDAQNetVersion ();
```

Remark

None.

2.1.7. HS_GetHWFirmwareVersion

Read the hardware firmware version of ET-7H16.

Syntax

C/C++

```
bool HS_GetHWFirmwareVersion (
   HANDLE hobj,
   LPSTR fpga_version
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

version

[out] The version number of ET-7H16 Hardware firmware.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Syntax

.Net

```
string GetHWFirmwareVersion(IntPtr hobj);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create.

Return Values

If the function succeeds, the return string is te version number of ET-7H16 hardware firmware.If the function fails, the return value is NULL. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
char fw_version[32]={0};
hHS = HS_Device_Create("192.168.1.1");
HS_GetHWFirmwareVersion (hHS, fw_version);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
string fwVersion=HSDAQNet.Sys.GetHWFirmwareVersion(hHS).ToString();
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.2. Configuration API

The configuration API functions describe or change the configuration, settings, and attributes of ET-7H16 module.

Configuration Functions

The following functions are used to retrieve or set the IO modules.

HSDAQ.dll Functions	HSDAQNet.dll Functions	Description
HS_ReadGainOffset	Config.HS_ReadGainOffset	Read the gain/offset values for application to calibrate each channel's analog data
HS_SetConfig	Config.HS_SetConfig	Set the configuration option for a device.
HS_SetConfigString	Config.HS_SetConfigString	Set the configuration option for a device with the string.
HS_GetConfig	Config.HS_GetConfig	Read the configuration option for a devcice.
HS_GetConfigString	Config.HS_GetConfigString	Read the configuration option for a device with the string.

2.2.1. HS_ReadGainOffset

Read the gain/offset values for application to calibrate each channel's analog raw data.

Syntax

C/C++

```
bool HS_ReadGainOffset (
    HANDLE hobj,
    int ch,
    unsigned short *gainVal,
    short *offsetVal
);
```

.Net

```
bool HS_ ReadGainOffset (
    IntPtr hobj,
    int ch,
    ref UInt16 gainVal,
    ref Int16 offsetVal
}
```

Parameter

obj

[in] A handle to the specified device opened by HS_Device_Create

ch

[in] The channel that reads the gain/offset values value from the module.

gainVal

[out] Get the gain value of the specified AI channel.

offsetVal

[out] Get the offset value of the specified AI channel.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
unsigned short gainVal=0;
short offsetVal=0;

hHS = HS_Device_Create("192.168.1.1");
for (int ch = 0; ch < 8; ch++)
{
   HS_ReadGainOffset(hHS,ch,&gainVal,&offsetVal);
   …//user-define code
}
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt16 gVal = 0;
Int16 oVal = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
for (int ch = 0; ch < 8; ch++)
{
   HSDAQNet.Config.HS_ReadGainOffset(hHS,ch,ref gVal,ref oVal);
   …//user-define code
}
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

The data received from the buffer through calling the HS_GetAIBufferHex function is the raw data without calibration. The user needs to call HS_ReadGainOffset to obtain the gain and offset value to calibrate the raw data.

2.2.2. HS_SetConfig

Set the configuration option for a device.

Syntax

C/C++

```
bool HS_SetConfig (
   HANDLE hobj,
   int configtype,
   int param,
   long settingval
);
```

.Net

```
bool HS_SetConfig (
   IntPtr hobj,
   int configtype,
   int param,
   int settingval
}
```

Parameter

obj

[in] A handle to the specified device opened by HS_Device_Create

configtype

[in] Specifies which configuration type set for the specified module.

param

[in] Specifies which parameter of the configuration type set for the specified module.

settingval

[in] The setting value to set the specified configuration and parameter.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;

hHS = HS_Device_Create("192.168.1.1");
HS_SetConfig (hHS, HSDAQ_CONFIG, HSDAQ_CONNECT_TIMEOUT,1);
//set tcp connection timeout as 1 seconds
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.Config.HS_SetConfig(hHS, (int)ConfigCode.HSDAQ_CONFIG,
(int)HSDAQIOPara.HSDAQ_CONNECT_TIMEOUT, 1);
  //set tcp connection timeout as 1 seconds
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.2.3. HS_GetConfig

Read the configuration option for a device.

Syntax

C/C++

```
bool HS_GetConfig (
   HANDLE hobj,
   int configtype,
   int param,
   long *settingval
);
```

.Net

```
bool HS_GetConfig (
   IntPtr hobj,
   int configtype,
   int param,
   ref int settingval
}
```

Parameter

obj

[in] A handle to the specified device opened by HS_Device_Create

configtype

[in] Specifies which configuration type set for the specified module.

param

[in] Specifies which parameter of the configuration type set for the specified module.

settingval

[out] Get the setting value of the specified configuration and parameter.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
long lvalue=0;
hHS = HS_Device_Create("192.168.1.1");
HS_GetConfig(hHS, HSDAQ_CONFIG, HSDAQ_CONNECT_TIMEOUT,& lvalue);
//Get the tcp connection timeout value
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
int lvalue=0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.Config.HS_GetConfig(hHS, (int)ConfigCode.HSDAQ_CONFIG,
(int)HSDAQIOPara.HSDAQ_CONNECT_TIMEOUT, ref lvalue);
  //Get tcp connection timeout value
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

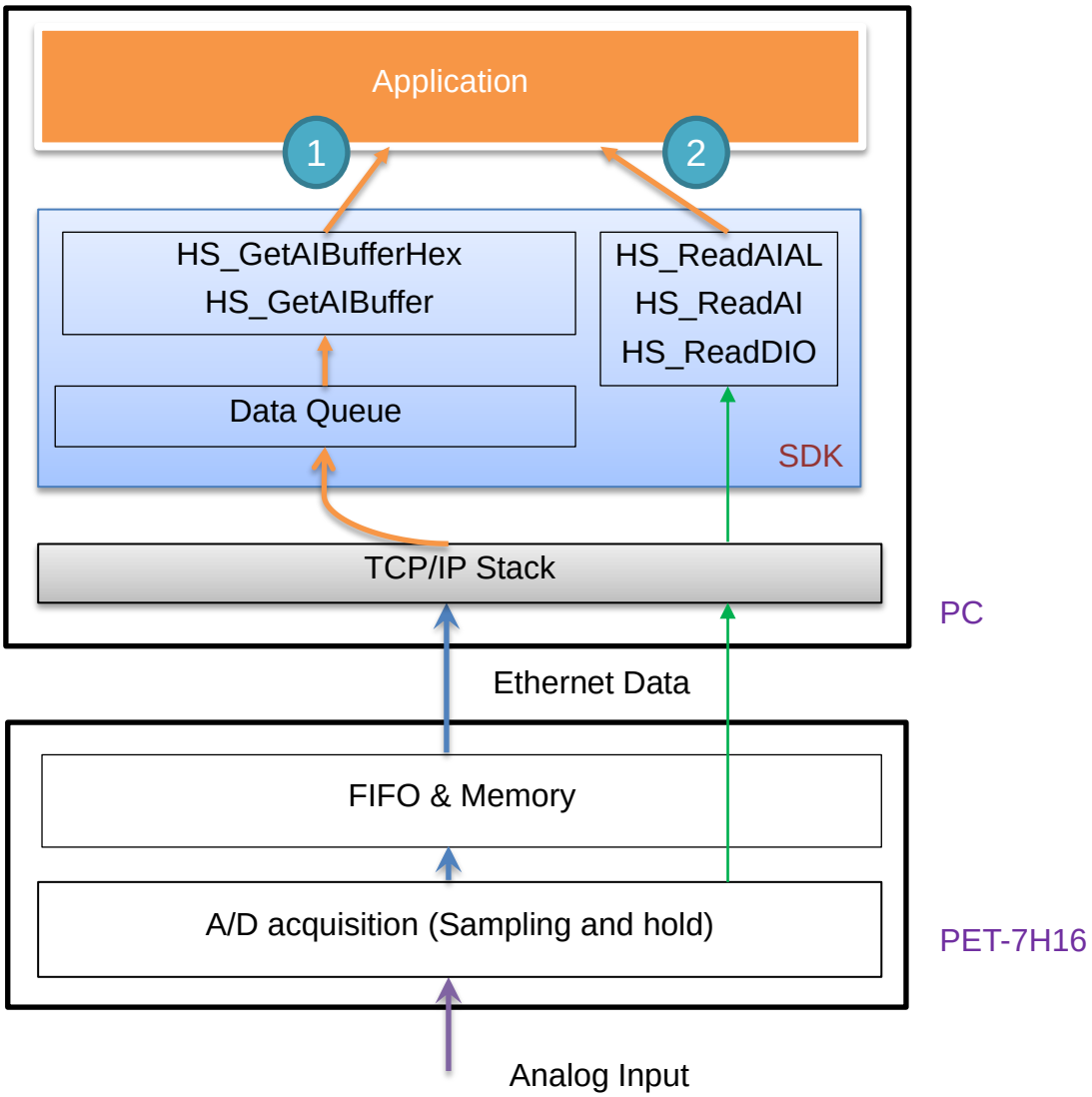
2.3. IO API

IO API supports to operate IO modules.

Data flow chart

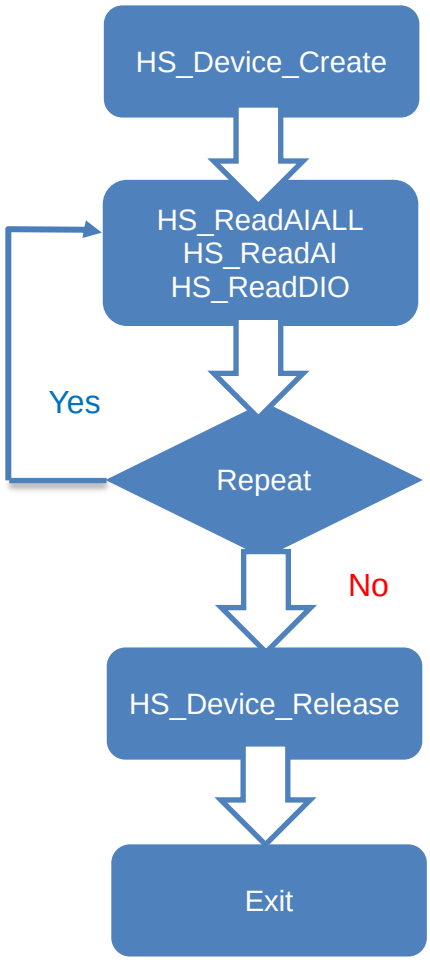
Flow 1: It means the module is in high speed data acquisition (Refer to the **Chapter 2.4**), The acquiring data is sent continuously from PET-7H16M to PC over the Ethernet. The huge queue provided by the ET-7H16 DAQ SDK is used to receive the data, The queue buffer provides the data first-in and first-out. And the queues can save data to avoid the data loss even if the PC is busy or Ethernet isn't stable.

Flow 2: It means that the application simply polls the AI/DIO value from PET-7H16M.



Polling the AI/DI value

API function call process chart



Create a connection to the device and initialize the deivce

Reads the AI/DIO values

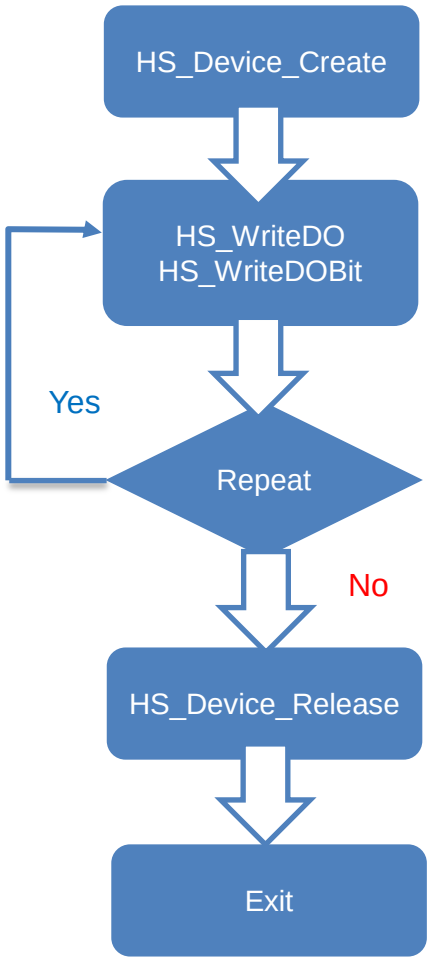
Determine total number of samples reaches the target count

Release the device from system

Exit

Write the DO value

API function call process chart



Create a connection to the device and initialize the deivce

Writes the DO values

Determine total number of samples reaches the target count

Release the device from system

Exit

I/O Functions

The following functions are used to retrieve or set the IO value asynchronously.

HSDAQ.dll Functions	HSDAQNet.dll Functions	Description
HS_ReadAIALL	IO. ReadAIALL	Reads all the AI values of all channels in engineering-mode.
HS_ReadAI	IO. ReadAI	Reads the AI value of a channel in engineering-mode
HS_ReadDIO	IO. ReadDIO	Reads the DI and DO values
HS_WriteDO	IO. WriteDO	Writes the DO value
HS_WriteDOBit	IO. WriteDOBit	Writes a DO value to a channel
HS_SetDICNTConfig	IO. SetDICNTConfig	Set the configuration of DI Counter
HS_SetCounterConfig	IO. SetCounterConfig	Set the configuration of Counter
HS_GetDICNTConfig	IO. GetDICNTConfig	Reads the configuration of DI Counter
HS_GetCounterConfig	IO. GetCounterConfig	Reads the configuration of Counter
HS_GetDICNT	IO. GetDICNT	Reads the DI Counter values
HS_GetCounter	IO. GetCounter	Reads the Counter value
HS_GetDICNTAll	IO. GetDICNTAll	Reads all channel of DI Counter
HS_GetCounterAll	IO. GetCounterAll	Reads all channel of Counter
HS_ClearCounter	IO. ClearCounter	Clear the Counter value
HS_ClearDICNT	IO. ClearDICNT	Clear all channel of Counter
HS_ClearCounterALL	IO. ClearCounter	Clear all channel of Counter value
HS_ClearDICNTALL	IO. ClearDICNTALL	Clear all channel of DI Counter value

2.3.1. HS_ReadAIALL

Reads all the AI values of all channels in engineering-mode.

Syntax

C/C++

```
bool HS_ReadAIALL (
    HANDLE hobj,
    float ai[],
    int totalchannel
);
```

.Net

```
bool ReadAIALL (
    IntPtr hobj,
    float ai[],
    int totalchannel
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

ai

[out] The array contains the AI values that read back from the module. The AI value getting from the HS_ReadAIALL function is calibrated.

totalchannel

[in] A pointer to a variable that specifies the size of the buffer pointed to by the ai.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
float fAI[8];
hHS = HS_Device_Create(“192.168.1.1”);
HS_ReadAIALL(hHS,fAI,8);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
float[] fAI = new float[8];
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.ReadAIALL(hHS, fAI,8);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.2. HS_ReadAI

Reads the AI value of a channel in engineering-mode.

Syntax

C/C++

```
bool HS_ReadAI (
   HANDLE hobj,
   int ChannelIndex,
   float ai
);
```

.Net

```
bool ReadAI (
   IntPtr hobj,
   int ChannelIndex,
   ref float ai
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

ChannelIndex

[in] The channel that reads the AI value back from the module.

ai

[in] The pointer to an AI value that is read back from the module.The AI value getting from the HS_ReadAI function is calibrated.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information,

call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
float fAI=0;
hHS = HS_Device_Create(“192.168.1.1”);
HS_ReadAI(hHS,0,&fAI);
//Read a AI value of the channel 0
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
float fAI=0;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.ReadAI(hHS,0,ref fAI);
//Read a AI value of the channel 0
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.3. HS_ReadDIO

Reads the DI and DO values.

Syntax

C/C++

```
bool HS_ReadDIO (
   HANDLE hobj,
   unsigned long *diVal,
   unsigned long * doVal
);
```

.Net

```
bool ReadDIO (
   IntPtr hobj,
   ref ulong diVal,
   ref ulong doVal
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

diVal

[out] The pointer to the value of DI read back.

doVal

[out] The pointers to the value of DO read back.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
unsigned long ulDI=0,ulDO=0;
hHS = HS_Device_Create(“192.168.1.1”);
HS_ReadDIO(hHS,&ulDI,&ulDO);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
uint uiDI = 0;
uint uiDO = 0;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.ReadDIO(hHS,ref uiDI,ref uiDO);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.4. HS_WriteDO

Writes the DO value.

Syntax

C/C++

```
bool HS_WriteDO (
  HANDLE hobj,
  unsigned long val
);
```

.Net

```
bool WriteDIO (
  IntPtr hobj,
  ulong val
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

Val

[in] A hexadecimal value, where bit 0 corresponds to DO0, bit 1 corresponds to DO1, etc. When the bit is 1, it denotes that the digital output channel is on, and 0 denotes that the digital output channel is off.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
unsigned long ulDO=0x3;
hHS = HS_Device_Create("192.168.1.1");
HS_WriteDO(hHS, ulDO);
//Write 0x3 the digital output
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
ulong uiDO =0x3;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.WriteDO(hHS, uiDO);
//Write 0x3 the digital output
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.5. HS_WriteDOBit

Writes a DO value to a channel.

Syntax

C/C++

```
bool HS_WriteDOBit (
    HANDLE hobj,
    int ChannelIndex,
    bool val
);
```

.Net

```
bool WriteDOBit(
    IntPtr hobj,
    int ChannelIndex,
    bool val
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

ChannelIndex

[in] Write a DO value to the specified channel

Val

[in] 1 is to turn on the DO channel; 0 is off.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create(“192.168.1.1”);
HS_WriteDOBit(hHS, 1, 1);
//Set DO1 (channel 1) to ON
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
ulong uiDO =0x33;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.WriteDOBit(hHS, 1,true);
//Set DO1 (channel 1) to ON
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.6. HS_SetDICNTConfig

Set the configuration of DI Counter

Syntax

C/C++

```
bool HS_SetDICNTConfig (
   HANDLE hobj,
   DWORD wChannel,
   DWORD wMode,
   DWORD dwValue,
   DWORD reserved
);
```

.Net

```
bool SetDICNTConfig(
   IntPtr hobj,
   UInt32 wChannel,
   UInt32 wMode,
   UInt32 dwValue,
   UInt32 reserved
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Set the configuration setting of the specified DI counter channel

wMode

[in] Set counter mode

Mode	Description
CNT_ENABLE /	Enable/Disable the counter / DI counter

CNT_DISABLE	
CNT_SYNC	Set Counter/DI counter as synchronous Input data acquisition

dwValue

[in] DI counter initialize value.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
HS_SetDICNTConfig(hHS, 4, CNT_ENABLE, 0, 0);
//Enable 4-channel DI counter with initialization value set to 0
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
ulong uiDO =0x33;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.SetDICNTConfig(hHS, 4, CNTCong.CNT_ENABLE, 0, 0);
//Enable 4-channel DI counter with initialization value set to 0
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.7. HS_SetCounterConfig

Set the configuration of Counter

Syntax

C/C++

```
bool HS_SetCounterConfig (
   HANDLE hobj,
   DWORD wChannel,
   DWORD wMode,
   DWORD dwValue,
   DWORD reserved
);
```

.Net

```
bool SetDICNTConfig(
   IntPtr hobj,
   UInt32 wChannel,
   UInt32 wMode,
   UInt32 dwValue,
   UInt32 reserved
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Set the configuration setting of the specified Counter channel

wMode

[in] Set counter mode

Mode	Description
CNT_ENABLE /	Enable/Disable the counter / DI counter

CNT_DISABLE	
CNT_SYNC	Set Counter/DI counter as synchronous Input data acquisition

dwValue

[in] Counter initialize value.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
HS_SetCounterConfig(hHS, 1, CNT_ENABLE, 0, 0);
//Enable 1-channel counter with initialization value set to 0
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
ulong uiDO =0x33;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.SetCounterConfig(hHS, 1, CNTCong.CNT_ENABLE, 0, 0);
//Enable 1-channel counter with initialization value set to 0
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.8. HS_GetDICNTConfig

Reads the configuration of DI Counter

Syntax

C/C++

```
bool HS_GetDICNTConfig (
   HANDLE hobj,
   DWORD wChannel,
   DWORD* wMode,
   DWORD* dwValue,
   DWORD* reserved
);
```

.Net

```
bool GetDICNTConfig(
   IntPtr hobj,
   UInt32 wChannel,
   ref UInt32 wMode,
   ref UInt32 dwValue,
   ref UInt32 reserved
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Get the configuration setting of the specified DI counter channel

wMode

[out] Get counter mode

Mode	Description
CNT_ENABLE / CNT_DISABLE	Enable/Disable the counter / DI counter

CNT_SYNC	Set Counter/DI counter as synchronous Input data acquisition

dwValue

[out] Get the DI counter initialize value.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD wmode=0;
DWORD diCntInit=0;
hHS = HS_Device_Create("192.168.1.1");
HS_GetDICNTConfig(hHS, 4, & wmode, &diCntInit , 0);
//Get the configuration of 4-channel DI counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt32 wmode=0;
UInt32 diCntInit=0;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.GetDICNTConfig(hHS, 4, ref wmode,ref diCntInit, 0);
//Get the configuration of 4-channel DI counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.9. HS_GetCounterConfig

Reads the configuration of Counter

Syntax

C/C++

```
bool HS_GetCounterConfig (
    HANDLE hobj,
    DWORD wChannel,
    DWORD* wMode,
    DWORD* dwValue,
    DWORD* reserved
);
```

.Net

```
bool GetCounterConfig(
    IntPtr hobj,
    UInt32 wChannel,
    ref UInt32 wMode,
    ref UInt32 dwValue,
    ref UInt32 reserved
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Get the configuration setting of the specified Counter channel

wMode

[out] Get counter mode

Mode	Description
CNT_ENABLE / CNT_DISABLE	Enable/Disable the counter / DI counter

CNT_SYNC	Set Counter/DI counter as synchronous Input data acquisition

dwValue

[out] Get the counter initialize value.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD wmode=0;
DWORD CntInit=0;

hHS = HS_Device_Create("192.168.1.1");
HS_GetCounterConfig(hHS, 1, &wmode, &CntInit, 0);
//Get the configuration of 1-channel DI counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
ulong uiDO =0x33;
UInt32 wmode=0;
UInt32 CntInit=0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.GetCounterConfig(hHS, 1, ref wmode, ref CntInit, 0);
//Get the configuration of 1-channel DI counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.10. HS_GetDICNT

Reads the DI Counter values.

Syntax

C/C++

```
bool HS_GetDICNT (
   HANDLE hobj,
   DWORD wChannel,
   DWORD *dwValue
);
```

.Net

```
bool GetDICNT (
   IntPtr hobj,
   UInt32 wChannel,
   ref UInt32 dwValue
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Get the DI counter value of the specified channel

dwValue

[out] The pointers to the value of DI counter

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD dwValue = 0;
hHS = HS_Device_Create("192.168.1.1");
HS_GetDICNT(hHS, 2, &dwValue);
//Get the value of 2-channel DI counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt32 dwValue=0;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.GetDICNT(hHS, 2,ref dwValue);
//Get the value of 2-channel DI counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.11. HS_GetCounter

Reads the Counter values.

Syntax

C/C++

```
bool HS_GetCounter (
   HANDLE hobj,
   DWORD wChannel,
   DWORD *dwValue
);
```

.Net

```
bool GetCounter (
   IntPtr hobj,
   UInt32 wChannel,
   ref UInt32 dwValue
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] Get the counter value of the specified channel

dwValue

[out] The pointers to the value of counter

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD dwValue = 0;
hHS = HS_Device_Create("192.168.1.1");
HS_GetCounter(hHS, 2, &dwValue);
//Get the value of 2-channel counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt32 dwValue=0;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.GetCounter(hHS, 2,ref dwValue);
//Get the value of 2-channel counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.12. HS_GetDICNTALL

Reads all channel of DI Counter value

Syntax

C/C++

```
bool HS_GetDICNTALL (
   HANDLE hobj,
   DWORD dwValue[],
   int totalchannel);
```

.Net

```
bool GetDICNTALL (
   IntPtr hobj,
   UInt32[] dwValue,
   UInt32 totalchannel
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

dwValue

[out] The array contains the DI counter values that read back from the module.

wChannel

[in] specifies the size of the buffer pointed to by the dwValue array.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD dwValue[8] = 0;
hHS = HS_Device_Create(“192.168.1.1”);
HS_GetDICNTALL(hHS, dwValue, 8);
//Get the 8-channel value of DI counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt32[] dwValue=new Uint32[8]
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.GetDICNTALL(hHS, dwValue, 8);
//Get the 8-channel value of DI counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.13. HS_GetCounterALL

Reads all channel of Counter value

Syntax

C/C++

```
bool HS_GetCounterALL (
   HANDLE hobj,
   DWORD dwValue[],
   int totalchannel);
```

.Net

```
bool GetCounterALL (
   IntPtr hobj,
   UInt32[] dwValue,
   UInt32 totalchannel
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

dwValue

[out] The array contains the counter values that read back from the module.

wChannel

[in] specifies the size of the buffer pointed to by the dwValue array.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
DWORD dwValue[8] = 0;
hHS = HS_Device_Create("192.168.1.1");
HS_GetCounterALL(hHS, dwValue, 8);
//Get the 8-channel value of counter
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDEv;
UInt32[] dwValue=new Uint32[8]
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.GetCounterALL(hHS, dwValue, 8);
//Get the 8-channel value of counter
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.3.14. HS_ClearDICNT

Clear the DI Counter value

Syntax

C/C++

```
bool HS_ClearDICNT (
   HANDLE hobj,
   DWORD wChannel,
);
```

.Net

```
bool ClearDICNT(
   IntPtr hobj,
   UInt32 wChannel,
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] the specified DI counter channel

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create(“192.168.1.1”);
HS_ClearDICNT (hHS, 1);
//Clear DI counter value of channel 1
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHs;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.IO.ClearDICNT(hHS, 1);
//Clear DI counter value of channel 1
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.15. HS_ClearCounter

Clear the Counter value

Syntax

C/C++

```
bool HS_ClearCounter (
   HANDLE hobj,
   DWORD wChannel,
);
```

.Net

```
bool ClearCounter(
   IntPtr hobj,
   UInt32 wChannel,
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wChannel

[in] the specified Counter channel

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
HS_ClearCounter (hHS, 0);
//Clear Counter value of channel 0
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHs;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.ClearCounter(hHS, 0);
//Clear Counter value of channel 0
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.16. HS_ClearDICNTALL

Clear all channel of DI Counter value

Syntax

C/C++

```
bool HS_ClearDICNTALL (
   HANDLE hobj,
);
```

.Net

```
bool ClearDICNTALL(
   IntPtr hobj,
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
HS_ClearDICNTALL (hHS);
```

```
//Clear all channel of DI counter value
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHs;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.ClearDICNTALL(hHS);
//Clear all channel of DI counter value
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.3.17. HS_ClearCounterALL

Clear all channel of Counter value

Syntax

C/C++

```
bool HS_ClearCounterALL (
   HANDLE hobj,
);
```

.Net

```
bool ClearCounterALL(
   IntPtr hobj,
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
HS_ClearCounterALL (hHS);
```

```
//Clear all channel of counter value
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHs;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.IO.ClearCounterALL(hHS);
//Clear all channel of counter value
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

None.

2.4. High Speed IO API

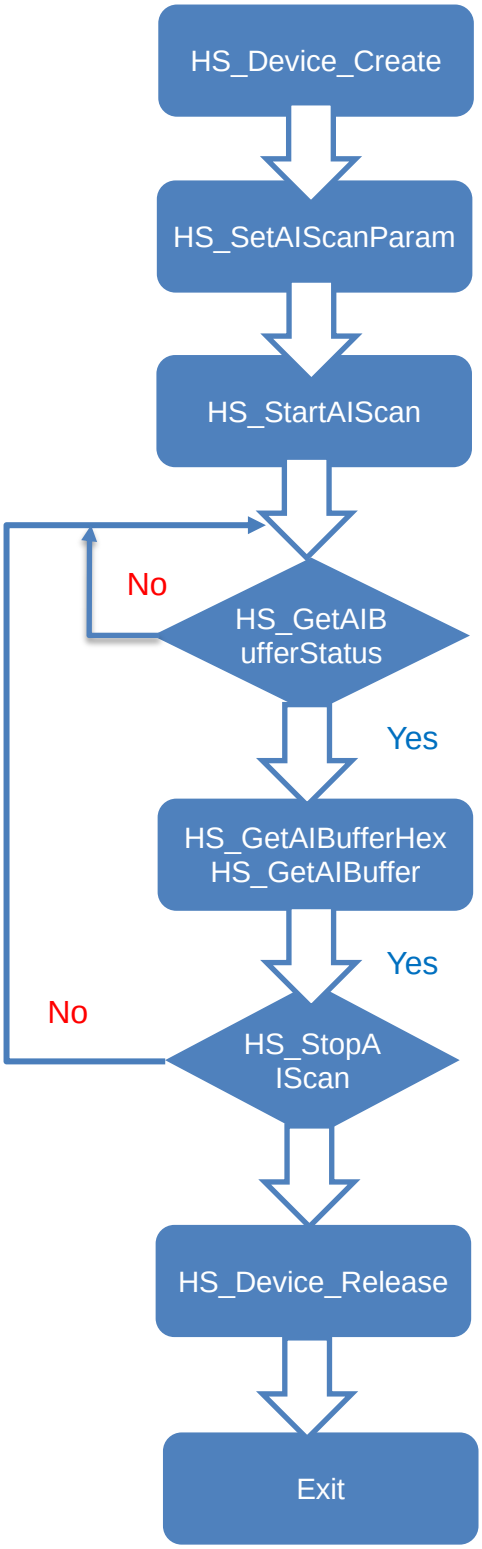
High Speed IO API supports to high speed data acquisition for AI channel.

There are two data acquisition modes and several trigger modes of analog input operation for high speed data acquisition.The following chart shows the acquisition and trigger modes and their operation frequency of each combination.

Acquisition / Trigger	Continuous	N Sample
Software AD	1~30KHz	1 Hz ~ 200 KHz
External CLK AD	1~30KHz	-
Analog Input Trigger	-	1 Hz ~ 200 KHz
Post-Trigger	-	1 Hz ~ 200 KHz
Pre-Trigger	-	1 Hz ~ 200 KHz
Delay Trigger	-	1 Hz ~ 200 KHz

Continuous mode acquisition and software AD trigger

API function call process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate <=30KHz
- Trigger mode = 0 (Software trigger)
- TargetCnt =0 (0: Continuous mode)

Start data acquisition of scanning AI channels

Determine how much data is in the buffer

Read the data from the buffer

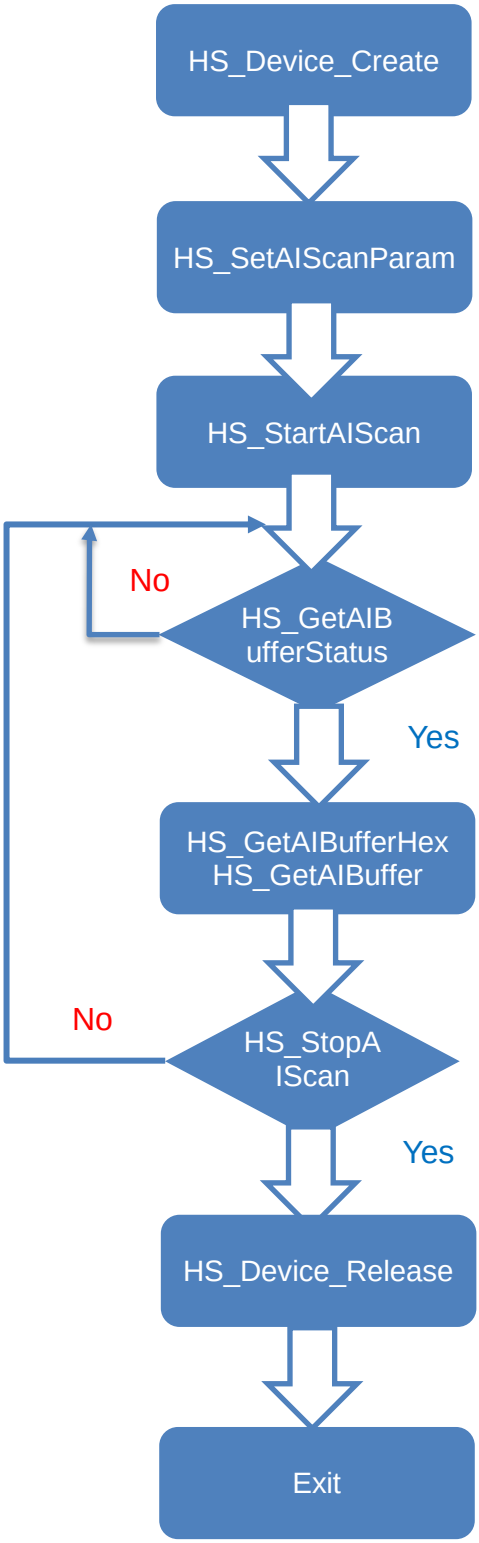
Whether stop data acquisition manually

Release the device from system

Exit

Continuous mode acquisition and External CLK AD

API function call process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate <=30KHz
- Trigger mode = 1 (Externl CLK AD)
- TargetCnt =0 (0: Continuous mode)

Start data acquisition of scanning AI channels

Determine how much data is in the buffer

Read the data from the buffer

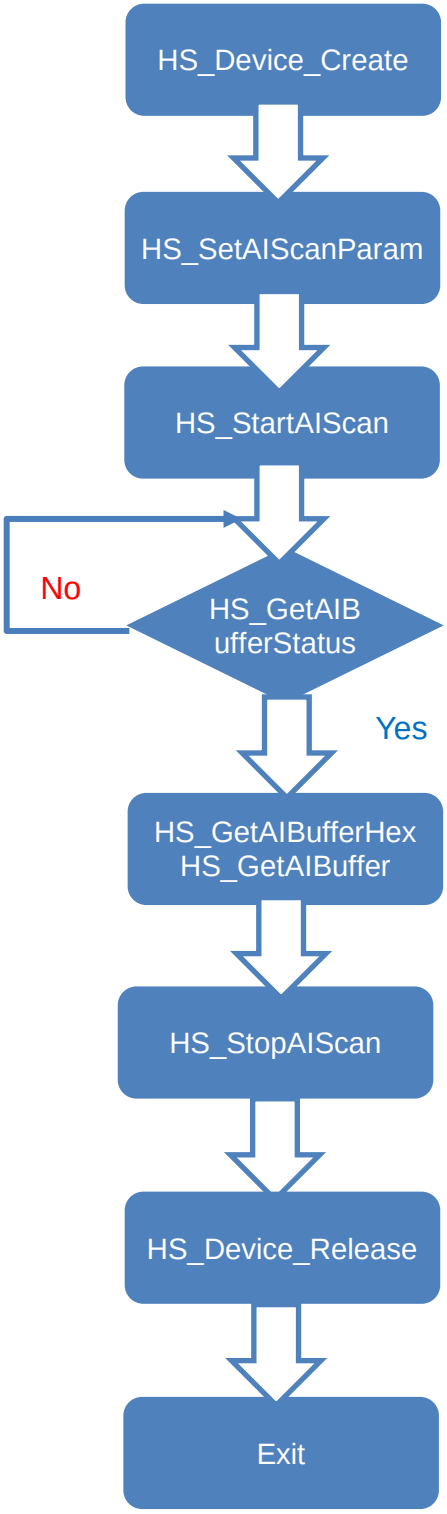
Whether stop data acquisition manually

Release the device from system

Exit

N sample mode acquisition and software AD trigger

API function process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate : 1 ~ 200KHz
- Trigger mode = 0 (Software trigger)
- TargetCnt >0 (>0: N sample mode)

Start data acquisition of scanning AI channels

Determine total number of samples reaches the target count

Read the data from the buffer

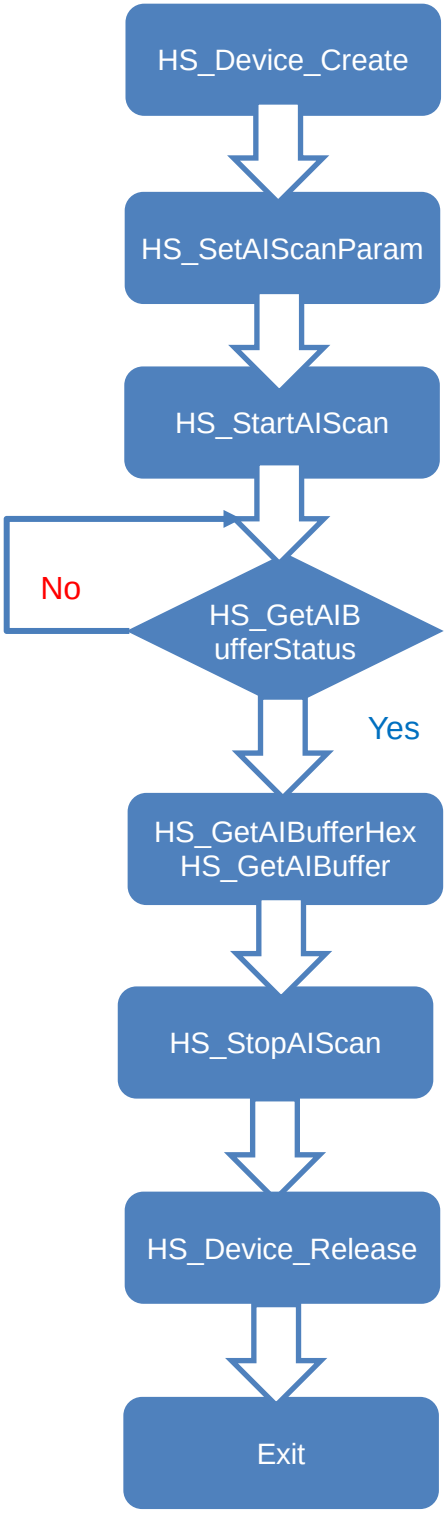
Stop data acquisition

Release the device from system

Exit

N sample mode acquisition and Post-Trigger

API function process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate : 1 ~ 200KHz
- Trigger mode = 2 (Post-trigger)
- TargetCnt >0 (>0: N sample mode)

Start data acquisition of scanning AI channels

Determine total number of samples reaches the target count

Read the data from the buffer

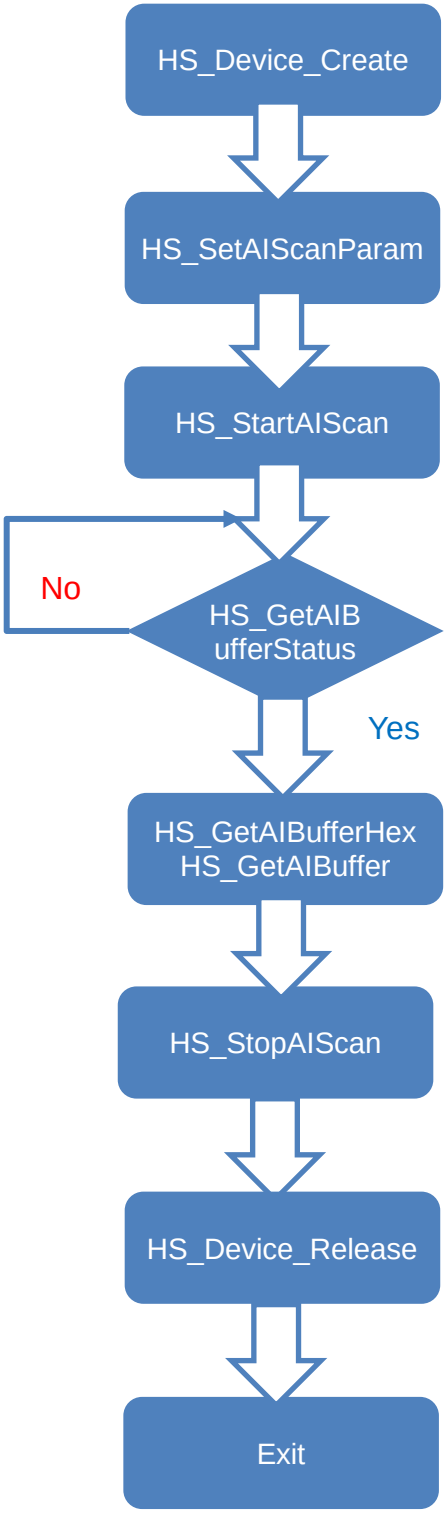
Stop data acquisition

Release the device from system

Exit

N sample mode acquisition and Pre-Trigger

API function process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate : 1 ~ 200KHz
- Trigger mode = 3 (Pre-trigger)
- TargetCnt >0 (>0: N sample mode)

Start data acquisition of scanning AI channels

Determine total number of samples reaches the target count

Read the data from the buffer

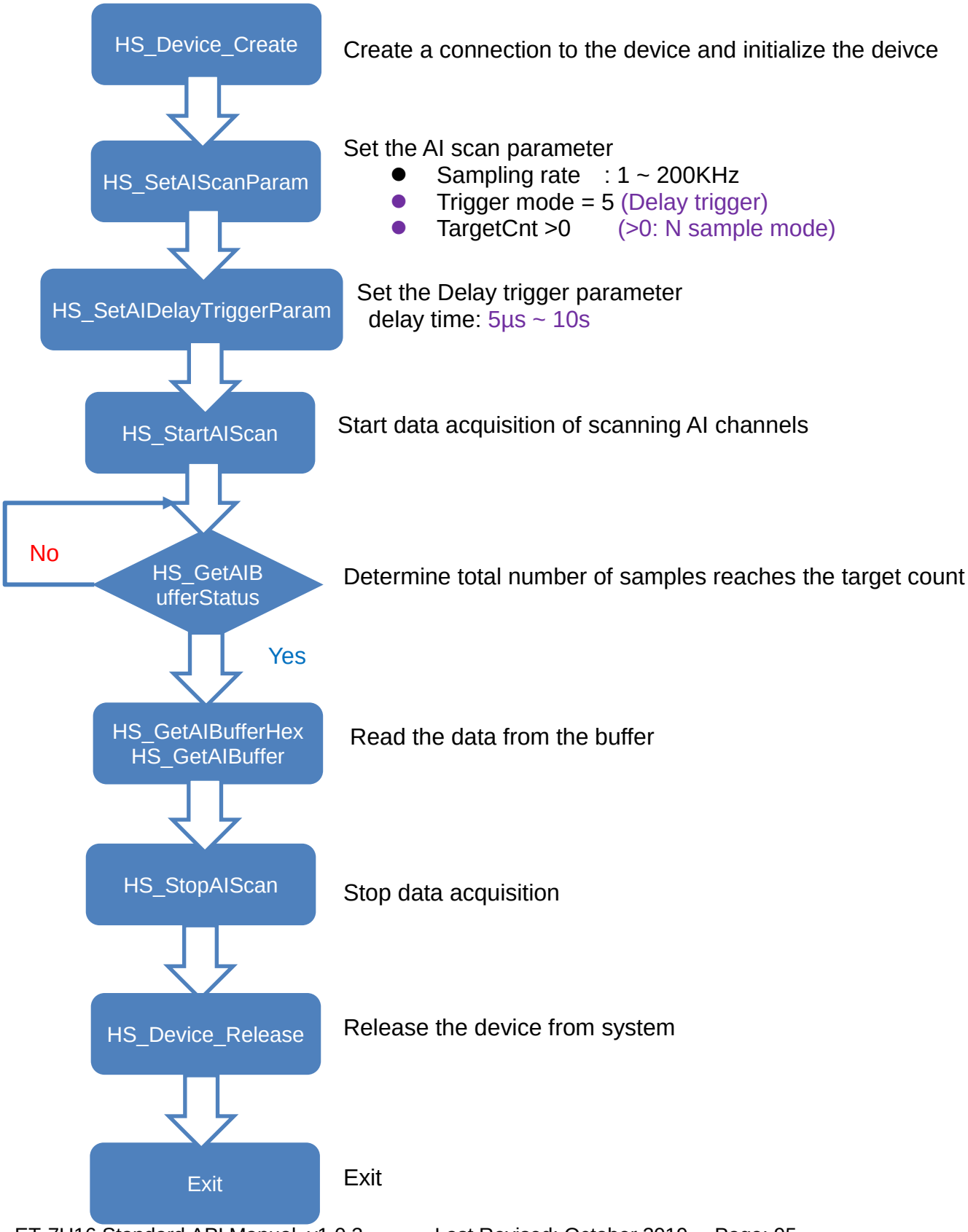
Stop data acquisition

Release the device from system

Exit

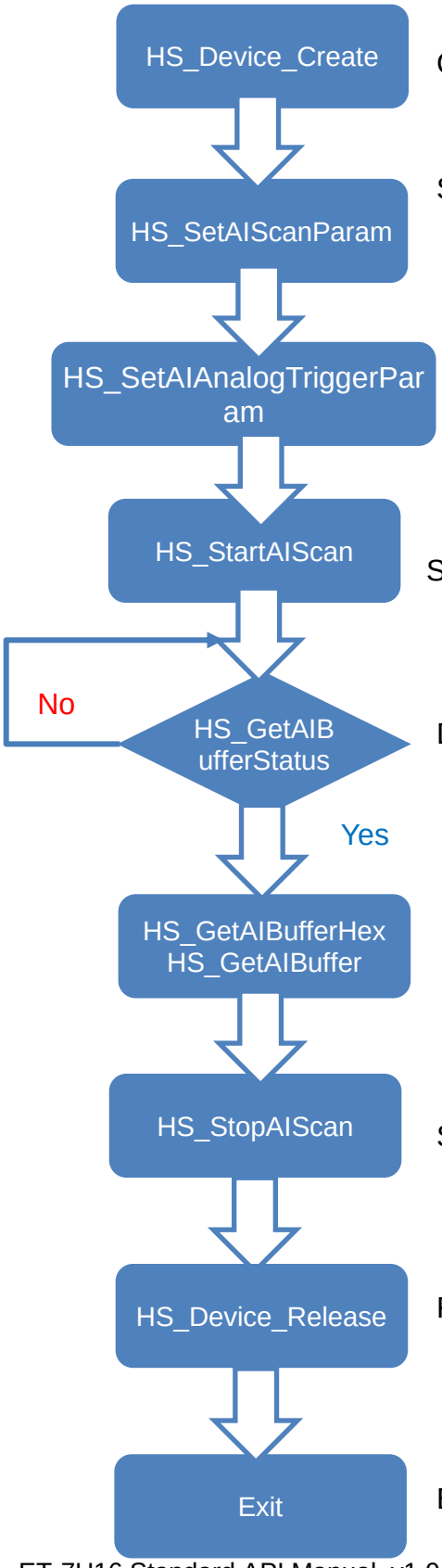
N sample mode acquisition and Delay Trigger

API function process chart



N sample mode acquisition and Analog Input Trigger

API function process chart



Create a connection to the device and initialize the deivce

Set the AI scan parameter

- Sampling rate : 1 ~ 200KHz
- Trigger mode = 6 (Analog Input trigger)
- TargetCnt >0 (>0: N sample mode)

Set the Analog Input trigger parameter

- Analog mode : (4 modes)
- Set high level or low level value for each channel
- Set AI scan target count before triggered or tarrget count after triggered

Start data acquisition of scanning AI channels

Determine total number of samples reaches the target count

Read the data from the buffer

Stop data acquisition

Release the device from system

Exit

High Speed I/O Functions

HSDAQ.dll Functions	HSDAQNet.dll Functions	Description
HS_GetAIScanParam	HSIO.HS_GetAIScanParam	Get the AI scan parameter from PET-7H16M regarding of the sampling rate, scan channels, pacer gain, trigger mode.
HS_SetAIScanParam	HSIO.HS_SetAIScanParam	Set the AI scan parameter for PET-7H16M regarding of the sampling rate, scan channels, pacer gain, trigger mode.
HS_StartAIScan	HSIO.HS_StartAIScan	Start data acquisition. The data is stored in memory and transfer to the data buffer on PC.
HS_StopAIScan	HSIO.HS_StopAIScan	Stop data acquisition.
HS_GetAIBuffer	HSIO.HS_GetAIBuffer	Get the floating-point value from data buffer on PC
HS_GetAIBufferHex	HSIO.HS_GetAIBufferHex	Get the binary data from data buffer on PC
HS_GetAIBufferStatus	HSIO.HS_GetAIBufferStatus	Get the status and data number from data buffer on PC.
HS_ClearAIBuffer	HSIO.HS_ClearAIBuffer	Clear the data buffer on PC
HS_GetTotalSamplingStatus	HSIO.HS_GetTotalSamplingStatus	Read the module status of ET-7H16 during data sampling
HS_TransmitDataCmd	HSIO.HS_TransmitDataCmd	Notify ET-7H16 module to send data to PC through TCP data port
HS_SetEventCallback	HSIO.HS_SetEventCallback	Bind the event condition to a user-defined callback function
HS_RemoveEventCallback	HSIO.HS_RemoveEventCallback	Disable the event condition and callback function.

HS_GetSyncInScanParam	HSIO.HS_GetSyncInScanParam	Get the parameter of the synchronous input data acquisition from PET-7H16M
HS_SetSyncInScanParam	HSIO.HS_SetSyncInScanParam	Set the parameter of the synchronous input data acquisition from PET-7H16M
HS_GetSyncInBuffer HS_GetSyncInBufferLV	HSIO.HS_GetSyncInBuffer	Get the synchronous input data acquisition parameter
HS_GetSyncInBufferStatus	HSIO.HS_GetSyncInBufferStatus	Get the status of the synchronous input data acquisition and data number from data buffer on PC.
HS_ClearSyncInBuffer	HSIO.HS_ClearSyncInBuffer	Clear the buffer for the synchronous input data acquisition on PC
HS_GetSyncInTotalSamplingStatus	HSIO.HS_GetSyncInTotalSamplingStatus	Read the status of ET-7H16 during synchronous input data acquisition
HS_SetAIAnalogTriggerParam	HSIO. HS_SetAIAnalogTriggerParam	Set the Analog Input trigger parameter for high speed data acquisition for AI channel.
HS_GetAIAnalogTriggerParam	HSIO. HS_GetAIAnalogTriggerParam	Get the Analog Input trigger parameter for high speed data acquisition for AI channel.
HS_SetAIDelayTriggerParam	HSIO. HS_SetAIDelayTriggerParam	Set the delay trigger parameter for high speed data acquisition for AI channel.
HS_GetAIDelayTriggerParam	HSIO. HS_GetAIDelayTriggerParam	Get the delay trigger parameter for high speed data acquisition for AI channel.

2.4.1. HS_GetAIScanParam

Get the AI scan parameter from PET-7H16M regarding of the sampling rate, scan channels, pacer gain, trigger mode.

Syntax

C/C++

```
bool HS_GetAIScanParam (
   HANDLE hobj,
   short *pacerChCnt,
   short *pacerGain,
   short *triggerMode,
   long *sampleRate,
   unsigned long *targetCnt,
   short *DataTransMethod,
   short * AutoRun
);
```

.Net

```
bool HS_GetAIScanParam (
   IntPtr hobj,
   ref short pacerChCnt,
   ref short pacerGain,
   ref short triggerMode,
   ref int sampleRate,
   ref UInt32 targetCnt,
   ref short DataTransMethod,
   ref short AutoRun
   );
```

Parameter

hobj
[in] A handle to the specified device opened by HS_Device_Create

pacerchCnt
[out] Get the AI scan channels from PET-7H16M.

The AI scan channel range for PET-7H16M is 1~8.

pacerGain

[out] Get the AI input type from PET-7H16M.
The gain range for PET-7H16M is

0: +/- 5V
1: +/- 10V

triggermode

[out] Get the AI scan trigger mode from PET-7H16M.
0: disable external trigger (software AD) ,
1: external clock trigger
2: external post-trigger,
3: external pre-trigger
5: delay trigger
6: analog input trigger

sampleRate

[out] Get the AI scan sampling rate from PET-7H16M.
The range of the AI scan sampling rate for PET-7H16M is 1~200KHz.

targetCnt

[out] Get the number of AI scan target count from PET-7H16M.
0 means the data acquisition mode is continuous mode.
>0 means N sample acquisition mode.
(Maximum number is 30,000,000)

DataTransMethod

[out] Get the data transmission method from PET-7H16M.
0: TCP socket

AutoRun

[out] Get the auto run status from PET-7H16M.
0, Auto run disabled
1, Auto run enabled

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
```

```
short chCnt=0;
short useGain=0;
short extriggMode=0;
long sampleRate=0;
unsigned long targetCnt=0;
short DataTransMethod=0;
short AutoRun=0;

hHS = HS_Device_Create("192.168.1.1");
ret=HS_GetAIScanParam(hHS, &chCnt, &useGain, &extriggMode, &sampleRate,
&targetCnt, &DataTransMethod, &AutoRun);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short rChCnt = 0;
short rGain=0;
short rTrigMode = 0;
int rsampleRate = 0;
UInt32 rtargetCnt = 0;
short rDataTransMethod = 0;
short rAutoRun = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_GetAIScanParam(hHS, ref rChCnt, ref rGain, ref rTrigMode,
ref rsampleRate, ref rtargetCnt, ref rDataTransMethod, ref rAutoRun);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.2. HS_SetAIScanParam

Set the AI scan parameter for PET-7H16M regarding of the sampling rate, scan channels, pacer gain, trigger mode.

Syntax

C/C++

```
bool HS_SetAIScanParam (
    HANDLE hobj,
    short pacerChCnt,
    short pacerGain,
    short triggerMode,
    long sampleRate,
    unsigned long targetCnt,
    short DataTransMethod,
    short AutoRun
);
```

.Net

```
bool HS_SetAIScanParam (
    IntPtr hobj,
    short pacerChCnt,
    short pacerGain,
    short triggerMode,
    int sampleRate,
    UInt32 targetCnt,
    short DataTransMethod,
    short AutoRun
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

pacerchCnt

[in] Set the AI scan channels for PET-7H16M.

The AI scan channel range for PET-7H16M is 1~8.

pacerGain

[in] Set the AI input type for PET-7H16M.

The gain range for PET-7H16M is

0: +/- 5V

1: +/- 10V

triggermode

[in] Set the AI scan trigger mode for PET-7H16M.

0: disable external trigger (software AD) ,

1: external clock trigger

2: external post-trigger,

3: external pre-trigger

5: delay trigger

6: analog input trigger

sampleRate

[in] Set the AI scan sampling rate for PET-7H16M.

The range of the AI scan sampling rate for PET-7H16M is 1~200KHz.

targetCnt

[in] Set the number of AI scan target count for PET-7H16M.

0 means the data acquisition mode is continuous mode.

>0 means N sample acquisition mode.

(Maximum number is 30,000,000)

DataTransMethod

[in] Set the data transmission method for PET-7H16M.

0: TCP socket

AutoRun

[in] Set the auto run status for PET-7H16M.

0, Auto run disabled

1, Auto run enabled

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=0;                  //scan channel is 8
short useGain=0;                //AI input type is 0 (+/- 5V)
short extriggMode=0;            // Software trigger
long sampleRate=1000;           // sampling rate is 1KHz
unsigned long targetCnt=1000; // target count is 1000
short DataTransMethod=0;
short AutoRun=0;

hHS = HS_Device_Create("192.168.1.1");
ret=HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate,
targetCnt, DataTransMethod,AutoRun);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short ChCnt = 0;                //scan channel is 8
short Gain=0;                   //AI input type is 0 (+/- 5V)
short TrigMode = 0;             // Software trigger
int sampleRate = 1000; // sampling rate is 1KHz
Uint32 targetCnt = 1000;        // target count is 1000
short DataTransMethod = 0;
short AutoRun = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_SetAIScanParam(hHS, ChCnt, Gain, TrigMode,
sampleRate, targetCnt, DataTransMethod, AutoRun);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.3. HS_StartAIScan

This function is used to start data acquisition of scanning AI channels. The data reading at the specified sampling rate from the specified range of AI channel is stored in memory and transfer to the data buffer on PC.

Syntax

C/C++

```
bool HS_StartAIScan (
   HANDLE obj,
);
```

.Net

```
bool HS_StartAIScan (
   IntPtr obj,
);
```

Parameter

obj

[in] A handle to the specified device opened by HS_Device_Create

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;

hHS = HS_Device_Create(“192.168.1.1”);
ret=HS_StartAIScan (hHS);
…//user-define code
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.HSIO.HS_StartAIScan(hHS);
… //user-define code
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

In high-speed data acquisition. Use HS_StartAIScan to start data acquisition and use HS_StopAIScan to stop data acquisition. The data reading at the specified sampling rate from the specified range of AI channel is stored in memory of ET-7H16 and transfer to the data buffer on PC via Ethernet.It must use HS_GetAIBuffer and HS_GetAIBufferHex function to get data from the data buffer.

2.4.4. HS_StopAIScan

This function is used to stop data acquisition.

Syntax

C/C++

```
bool HS_StopLogger (
   HANDLE obj,
);
```

.Net

```
bool HS_ StopLogger (
   IntPtr obj,
);
```

Parameter

obj

[in] A handle to the specified device opened by HS_Device_Create

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
```

```
hHS = HS_Device_Create(“192.168.1.1”);
```

```
ret=HS_StartAIScan (hHS);
…//user-define code
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.HSIO.HS_StartAIScan(hHS);
… //user-define code
HSDAQNet.HSIO.HS_StopAIScan(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

In high-speed data acquisition. Use HS_StartAIScan to start data acquisition and use HS_StopAIScan to stop data acquisition. The data reading at the specified sampling rate from the specified range of AI channel is stored in memory of ET-7H16 and transfer to the data buffer on PC via Ethernet.It must use HS_GetAIBuffer and HS_GetAIBufferHex function to get data from the data buffer.

2.4.5. HS_GetAIBufferHex

This function is used to get the binary data from data buffer on PC

Syntax

C/C++

```
DWORD HS_GetAIBufferHex (
   HANDLE hobj,
   WORD *wBuffer,
   DWORD dwBufferSize
);
```

.Net

```
UInt32 HS_GetAIBufferHex (
   IntPtr hobj,
   UInt16 [] wBuffer,
   UInt32 dwBufferSize
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wBuffer

[out] A pointer to a WORD buffer reading from the data buffer. Declare the WORD array and the array size is dwBufferSize.

dwBufferSize

[in] The number of data in the data buffer on PC

Return Values

If the function succeeds, returns the number of data actually received from the buffer. If the function fails or no data, the return value is 0. To get extended

error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
WORD BufferStatus=0;
short *dataBuffer;
unsigned long ulleng=0;

hHS = HS_Device_Create(“192.168.1.1”);
HS_StartAIScan (hHS);
dataBuffer=(short *)malloc(sizeof(short)*targetCnt);
ret=HS_GetAIBufferStatus(hHS,&BufferStatus,&ulleng);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng==targetCnt)
   {
      readsize=HS_GetAIBufferHex(hHS,(WORD *)dataBuffer, ulleng);
      …
   }
}
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
UInt16[] HdataBuffer = new UInt16[targetCnt];
uint ulleng = 0;
ushort BufferStatus = 0;

hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.HSIO.HS_StartAIScan(hHS);
```

```
ret = HSDAQNet.HSIO.HS_GetAIBufferStatus(hHS, ref BufferStatus, ref ulleng);
if (ret == false)
{
   Console.WriteLine(IP.ToString() + " Error code 0x" +
HSDAQNet.ErrHandling.GetLastError().ToString("x8"));
}
else
{
   if (ulleng == targetCnt)
   {
      readsize = HSDAQNet.HSIO.HS_GetAIBufferHex(hHS, HdataBuffer, ulleng);
   }
}
HSDAQNet.HSIO.HS_StopAIScan(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

The data received from the buffer through calling the HS_GetAIBufferHex function is the raw data without calibration. The user needs to call HS_ReadGainOffset to obtain the gain and offset value to calibrate the raw data.

2.4.6. HS_GetAIBuffer

This function is used to get the floating-point value from data buffer on PC

Syntax

C/C++

```
DWORD HS_GetAIBuffer (
   HANDLE hobj,
   float *fBuffer,
   DWORD dwBufferSize
);
```

.Net

```
UInt32 HS_GetAIBuffer (
   IntPtr hobj,
   float [] fBuffer,
   UInt32 dwBufferSize
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

fBuffer

[out] A pointer to a float-point buffer reading from the data buffer. Declare the float-pont array and the array size is dwBufferSize.

dwBufferSize

[in] The number of data in the data buffer on PC

Return Values

If the function succeeds, returns the number of data actually received from the buffer. If the function fails or no data, the return value is 0. To get extended

error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
WORD BufferStatus=0;
float fdataBuffer[10000];
unsigned long ulleng=0;

hHS = HS_Device_Create(“192.168.1.1”);
HS_StartAIScan (hHS);

ret=HS_GetAIBufferStatus(hHS,&BufferStatus,&ulleng);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng)
   {
      readsize=HS_GetAIBuffer(hHS,fdataBuffer, ulleng);
      …
   }
}
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
float[] fdataBuffer = new float[10000];
uint ulleng = 0;
ushort BufferStatus = 0;

hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSIO.HS_StartAIScan(hHS);
```

```
ret = HSIO.HS_GetAIBufferStatus(hHS, ref BufferStatus, ref ulleng);
if (ret == false)
{
   Console.WriteLine(IP.ToString() + " Error code 0x" +
HSDAQNet.ErrHandling.GetLastError().ToString("x8"));
}
else
{
   if (ulleng)
   {
      readsize = HSDAQNet.HSIO.HS_GetAIBuffer(hHS, fdataBuffer, ulleng);
      ….
    }
}
HSIO.HS_StopAIScan(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

HS_GetAIBufferHex and HS_GetAIBuffer functions are used to read the data from data buffer on PC.The difference between these two functions is not only the type of the read data, but the floating point value getting from the HS_GetAIBuffer function is calibrated.

2.4.7. HS_GetAIBufferStatus

This function is used to get the status and data number from data buffer on PC.

Syntax

C/C++

```
bool HS_GetAIBufferStatus (
   HANDLE hobj,
   WORD *wBufferStatus,
   DWORD *dwDataCountOnBuffer
);
```

.Net

```
bool HS_GetAIBufferStatus (
   IntPtr hobj,
   ref UInt16 wBufferStatus,
   ref UInt32 dwDataCountOnBuffer
);
```

Parameter

hobj
[in] A handle to the specified device opened by HS_Device_Create

wBufferStatus
[out] The buffer status
0: the data buffer is empty
1: the data exists in the buffer
2: the data buffer is overflow
4: the AI scan is stopped

dwDataCountOnBuffer
[out] The data number in the data buffer on PC

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
WORD BufferStatus=0;
float fdataBuffer[10000];
unsigned long ulleng=0;

hHS = HS_Device_Create(“192.168.1.1”);
HS_StartAIScan (hHS);

ret=HS_GetAIBufferStatus(hHS,&BufferStatus,&ulleng);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(BufferStatus>2) //AI buffer overflow
   {
   /*
        2:   AD_BUF_OVERFLOW
        4:   AD_SCAN_STOP
        8:   AD_DATA_SAMPLING_TIMEOUT
   */
      printf("Error<%d>, %lu\r\n",BufferStatus,ulleng);
      break;
   }
   if(ulleng)
   {
      readsize=HS_GetAIBuffer(hHS,fdataBuffer, ulleng);
      …
   }
}
HS_StopAIScan (hHS);
```

```
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
float[] HdataBuffer = new float[10000];
uint ulleng = 0;
ushort BufferStatus = 0;

hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.HSIO.HS_StartAIScan(hHS);
ret = HSDAQNet.HSIO.HS_GetAIBufferStatus(hHS, ref BufferStatus, ref ulleng);
if (ret == false)
{
   Console.WriteLine(IP.ToString() + " Error code 0x" +
HSDAQNet.ErrHandling.GetLastError().ToString("x8"));
}
else
{
   if (BufferStatus > 2) //AI buffer overflow
   {
     /*
     2:   AD_BUF_OVERFLOW
     4:   AD_SCAN_STOP
     8:   AD_DATA_SAMPLING_TIMEOUT
     */
        Console.WriteLine(IP.ToString() + " Error<" + BufferStatus + ">," + ulleng);
        break;
   }
   if (ulleng == targetCnt)
   {
      readsize = HSDAQNet.HSIO.HS_GetAIBufferHex(hHS, HdataBuffer, ulleng);
   }
}
HSDAQNet.HSIO.HS_StopAIScan(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.8. HS_ClearAIBuffer

Clear the data buffer on PC.

Syntax

C/C++

```
bool HS_ClearAIBuffer(
   HANDLE hobj
);
```

.Net

```
bool HS_ClearAIBuffer(
   IntPtr hobj
);
```

Parameter

Hobj
[in] A handle to the specified device opened by HS_Device_Create.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;

hHS = HS_Device_Create("192.168.1.1");
```

```
HS_ClearAIBuffer(hHS);
ret=HS_StartAIScan (hHS);
…//user-define code
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_ClearAIBuffer(hHS);
```

```
HSDAQNet.HSIO.HS_StartAIScan(hHS);
… //user-define code
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.9. HS_GetTotalSamplingStatus

Read the module status of ET-7H16 during data sampling.

Syntax

C/C++

```
bool HS_GetTotalSamplingStatus (
   HANDLE hobj,
   unsigned long * totalReadCnt,
   unsigned int * SamplingStatus
);
```

.Net

```
bool HS_GetTotalSamplingStatus (
   IntPtr hobj,
   ref UInt32 totalReadCnt,
   ref UInt32 triggerStatus
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

totalReadCnt

[out] The count of the sampling data stored in the memory on the PET-7H16M module

SamplingStatus

[out] The trigger status of PET-7H16M module in data sampling
1: The digital signal is triggered in the pre-trigger mode
0: The signal isn't triggered.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
unsigned long ulleng=0;
unsigned long triggerStatus=0;
hHS = HS_Device_Create("192.168.1.1");
HS_StartLogger(hHS,NULL,2,0);
ret=HS_GetTotalSamplingCnt(hHS,&ulleng,&triggerStatus);
if(ret==false)
{
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng>=targetCnt && targetCnt>0) //N sample mode
   {
         ret=HS_TransmitDataCmd(hHS);
         … //user-define code
   }
}
HS_StopLogger (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
float[] HdataBuffer = new float[10000];
uint ulleng = 0;
uint triggerStatus = 0;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.DATALOG.HS_StartLogger(hHS,NULL,2,0);
ret= HSDAQNet.HSIO.HS_GetTotalSamplingCnt(hHS,ref ulleng,ref triggerStatus);
```

```
if(ret==false)
{
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng>=targetCnt && targetCnt>0) //N sample mode
   {
         ret= HSDAQNet.HSIO.HS_TransmitDataCmd(hHS);
         … //user-define code
   }
}
HSDAQNet.DATALOG.HS_StopLogger (hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.10. HS_TransmitDataCmd

To notify ET-7H16 to send data to PC from data port.

Syntax

C/C++

```
bool HS_TransmitDataCmd (
   HANDLE hobj
);
```

.Net

```
bool HS_TransmitDataCmd (
   IntPtr hobj
);
```

Parameter

hobj
[in] A handle to the specified device opened by HS_Device_Create

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
unsigned long ulleng=0;
```

```
hHS = HS_Device_Create(“192.168.1.1”);
HS_StartLogger(hHS,NULL,2,0);
ret=HS_GetTotalSamplingCnt(hHS,&ulleng);
if(ret==false)
{
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng>=targetCnt && targetCnt>0) //N sample mode
   {
         ret=HS_TransmitDataCmd(hHS);
         … //user-define code
   }
}
HS_StopLogger (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
float[] HdataBuffer = new float[10000];
uint ulleng = 0;

hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.DATALOG.HS_StartLogger(hHS,NULL,2,0);
ret= HSDAQNet.HSIO.HS_GetTotalSamplingCnt(hHS,&ulleng);
if(ret==false)
{
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
   if(ulleng>=targetCnt && targetCnt>0) //N sample mode
   {
         ret= HSDAQNet.HSIO.HS_TransmitDataCmd(hHS);
         … //user-define code
   }
```

```
}
HSDAQNet.DATALOG.HS_StopLogger (hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.11. HS_SetEventCallback

Bind the event condition to a user-defined callback function.

Syntax

C/C++

```
WORD HS_SetEventCallback (
   HANDLE hobj,
   WORD wEventType,
   WORD EventParam,
   PVOID CallbackFun,
   void *pdwCallBackParameter
);
```

.NET

```
UInt16 HS_SetEventCallback (
   IntPtr hobj,
   UInt16 wEventType,
   UInt16 EventParam,
   CallBackFun CallbackFun,
   IntPtr pdwCallBackParameter
   );
```

Callback function type

```
void CallBackFun(UInt32 Param, UInt32 Param2);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wEventType

[in] Sets notification event type, each bit can be enable one mode

Event type	code	descriptions
EVENT_ERROR	0x0001	Generates an event upon while a error occurred
EVENT_N_SAMPLE_REACH	0x0002	Generates an event while the total number of samples reached
EVENT_DATA_SAMPLING_TIMEOUT	0x0004	Generates an event upon while a sampling timeout occurred
EVENT_LAN_BUFFER_OVERFLOW	0x0008	Generates an event while the LAN buffer overflow occurred

EventParam

[in] Sets additional data required to specify event conditions.

CallbackFun

[in] Sets Callback Function. The address of pointer to the user-defined callback function to handle the above event type.

pdwCallBackParameter

[in] Sets the Parameters for Callback Function

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is NoneZero. To get extended error information, call HS_GetLastError.

Remark

None.

2.4.12. HS_RemoveEventCallback

Disable the event condition and callback function.

Syntax

C/C++

```
WORD HS_RemoveEventCallback (
   HANDLE hobj,
   WORD wEventType
);
```

.Net

```
UInt16 HS_RemoveEventCallback (
   IntPtr hobj,
   UInt16 wEventType
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

wEventType

[in] Sets notification event type, each bit can be enable one mode

Event type	code	descriptions
EVENT_ERROR	0x0001	Generates an event upon while a error occurred
EVENT_N_SAMPLE_REACH	0x0002	Generates an event while the total number of samples reached
EVENT_DATA_SAMPLING_TIMEOUT	0x0004	Generates an event upon while a sampling timeout occurred
EVENT_LAN_BUFFER_OVERFLOW	0x0008	Generates an event while the LAN buffer overflow occurred

Return Values

If the function succeeds, the return value is 0.

If the function fails, the return value is NoneZero. To get extended error information, call HS_GetLastError.

Remark

None.

2.4.13. HS_GetSyncInScanParam

Get the parameter of the synchronous input data acquisition from PET-7H16M

Syntax

C/C++

```
bool HS_GetSyncInScanParam (
   HANDLE hobj,
   DWORD * SyncInheader,
   WORD *InChNumArray[],
   WORD *InChTypeArray[],
   WORD Arraycount,
   WORD * ActualArrayAmout ,
   DWORD *Options,
   DWORD *Reserved,
);
```

.Net

```
bool HS_GetSyncInScanParam (
   IntPtr hobj,
   ref UInt32 SyncInheader,
   UInt16[] InChNumArray,
   UInt16[] InChTypeArray,
   UInt16 Arraycount,
   ref UInt16 ActualArrayAmout,
   ref UInt32 Options,
   ref UInt32 Reserved
   );
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

SyncInheader

[Out] Get the header format of data frame of the synchronous input data acquisition

Mode	**Description**
0	None, no header for the data frame
1	Package index, Increase 1 for each synchronous input data acquisition. When counting to the maximum value(Max. 65535), the counter will return to 0 automatically.

2	**Timestamp (type 1)**
	typedef struct daqtime { //timestamp type 1
	__int16 day:5;//5bits
	__int16 hour:5;//5bits
	__int16 minute:6;//6bits
	__int16 sec:6;//6bits
	__int16 msec:10; //10bits
	} DAQ_TIME; //total 32bits

3	Timestamp (type 2)
	typedef struct { //timestamp type 2
	__int32 minute:6;//6bits
	__int32 sec:6;//6bits
	__int32 msec:10; //10bits
	__int32 usec:10; //10bits
	} DAQ_TIME2; //total 32bits

InChNumArray

[Out] Get the input channel number of each element of the arrary used for synchronous input data acquisition (The max. length of the array is 7)

InChTypeArray

[Out] Get the input channel type for the corresponding of InChNumArray array

SYNC_IN_TYPE	Descriptions
SYNC_IN_AI=0,	2bytes AI float data(Hex to float)
SYNC_IN_AI_HEX,	2bytes AI Hex data
SYNC_IN_WORD_DI_CNT,	2bytes DI Counter data (16-bit)
SYNC_IN_WORD_CNT,	2bytes Counter data (16-bit)
SYNC_IN_DWORD_DI_CNT,	4bytes DI Counter data (32-bit)
SYNC_IN_DWORD_CNT,	4bytes Counter data (32-bit)
SYNC_IN_DI,	One bit represents a DI channel value
SYNC_IN_DO,	One bit represents a DO channel value
SYNC_IN_UD_BYTE,	1byte user-defined data
SYNC_IN_UD_WORD,	2byte user-defined data
SYNC_IN_UD_DWORD,	4byte user-defined data
SYNC_IN_UD_FLOAT	4byte user-defined float data

Arraycount

[In] Defined the array length for InChNumArray and InChTypeArray

ActualArrayAmout

[Out]Get the actual array length of InChNumArray and InChTypeArray

Options

[Out] Get the options of the synchronous input data acquisition

Mode	Description
SYNC_ENABE/SYNC_DISABLE (Default : disable)	Enable/disable synchronous data acquisition

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
WORD InChNumArray[8]={0};
WORD InChTypeArray [8]={0};
WORD totalInCount=8;
WORD AcutArryAcount=0;
WORD SyncInheader =0;
WORD woption= 0;

hHS = HS_Device_Create("192.168.1.1");
HS_GetSyncInScanParam(hHS,(DWORD *)&SyncInheader, InChNumArray,
InChTypeArray, totalInCount,(WORD *)&AcutArryAcount,(DWORD *)&
woption,NULL);
…
HS_Device_Release (hHS);
```

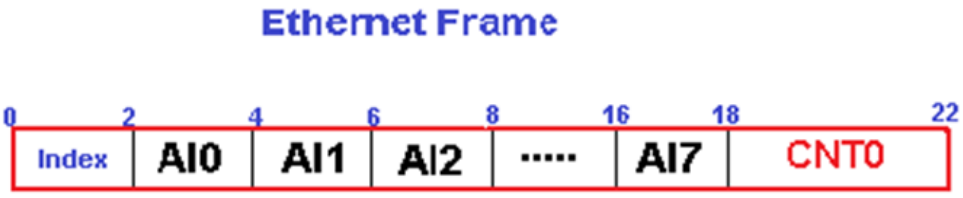
[C#]

```
IntPtr hHS;
UInt32 headertype=0;
UInt16 [] InChNumArray= new Uint16[8];
```

```
UInt16 [] InChTypeArray= new UInt16[8];
UInt16 totalInCount=8;
UInt16 AcutArryAcount=0;
UInt32 woption= 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_GetSyncInScanParam(hHS, ref headertype, InChNumArray,
   InChTypeArray, totalInCount , ref AcutArryAcount, ref woption, 0);
…
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Thd frame data for VC and C# example above is as



Remark

None.

2.4.14. HS_SetSyncInScanParam

Set the parameter of the synchronous input data acquisition for PET-7H16M.

Syntax

C/C++

```
bool HS_SetSyncInScanParam (
   HANDLE hobj,
   DWORD SyncInheader,
   WORD InChNumArray[],
   WORD InChTypeArray[],
   WORD Arraycount,
   DWORD Options,
   DWORD Reserved,
);
```

.Net

```
bool HS_GetSyncInScanParam (
   IntPtr hobj,
   UInt32 SyncInheader,
   UInt16[] InChNumArray,
   UInt16[] InChTypeArray,
   UInt16 Arraycount,
   UInt32 Options,
   UInt32 Reserved
   );
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

SyncInheader

[in] Set the header format of data frame of the synchronous input data acquisition

Mode	**Description**
0	None, no header for the data frame
1	Package index, Increase 1 for each synchronous input data acquisition. When counting to the maximum value(Max. 65535), the counter will return to 0 automatically.

2	**Timestamp (type 1)**
	typedef struct daqtime { //timestamp type 1
	__int16 day:5;//5bits
	__int16 hour:5;//5bits
	__int16 minute:6;//6bits
	__int16 sec:6;//6bits
	__int16 msec:10; //10bits
	} DAQ_TIME; //total 32bits

3	Timestamp (type 2)
	typedef struct { //timestamp type 2
	__int32 minute:6;//6bits
	__int32 sec:6;//6bits
	__int32 msec:10; //10bits
	__int32 usec:10; //10bits
	} DAQ_TIME2; //total 32bits

InChNumArray

[in] Set the input channel number of each element of the arrary used for synchronous input data acquisition (The max. length of the array is 7)

InChTypeArray

[in] Set the input channel type for the corresponding of InChNumArray array

SYNC_IN_TYPE	Descriptions
SYNC_IN_AI=0,	2bytes AI float data(Hex to float)
SYNC_IN_AI_HEX,	2bytes AI Hex data
SYNC_IN_WORD_DI_CNT,	2bytes DI Counter data (16-bit)
SYNC_IN_WORD_CNT,	2bytes Counter data (16-bit)
SYNC_IN_DWORD_DI_CNT,	4bytes DI Counter data (32-bit)
SYNC_IN_DWORD_CNT,	4bytes Counter data (32-bit)
SYNC_IN_DI,	One bit represents a DI channel value
SYNC_IN_DO,	One bit represents a DO channel value
SYNC_IN_UD_BYTE,	1byte user-defined data
SYNC_IN_UD_WORD,	2byte user-defined data
SYNC_IN_UD_DWORD,	4byte user-defined data
SYNC_IN_UD_FLOAT	4byte user-defined float data

Arraycount

[In] Defined the array length for InChNumArray and InChTypeArray

Options

[in] Set the options of the synchronous input data acquisition

Mode	Description

SYNC_ENABE/SYNC_DISABLE (Default : disable)	Enable/disable synchronous data acquisition

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=8;                  //scan channel is 8
short useGain=0;                //AI input type is 0 (+/- 5V)
short extriggMode=0;           // Software trigger
unsigned long targetCnt=0; //Continue mode
long sampleRate=2000;      // sampling rate is 2KHz
short DataTransMethod=0;
short AutoRun=0
WORD InChNumArray[2]={0};
WORD InChTypeArray [2]={0};
WORD totalInCount=2; //scan 2 input type for synchronous input data acquisition.
WORD headertype=1; // Package index for the header of data frame
WORD woption= SYNC_ENABE; //enable synchronous input data acquisition.

//First input type is AI (Hex format) with scanning 8 channels
InChNumArray[0]=8;
InChTypeArray[0]= SYNC_IN_AI_HEX;

//2'nd input type is Counter (32-bit) with scanning 1 channels
InChNumArray[1]=1;
InChTypeArray[1]= SYNC_IN_DWORD_CNT;

hHS = HS_Device_Create("192.168.1.1");
HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate, targetCnt,
DataTransMethod,AutoRun);
```

```
HS_SetSyncInScanParam(hHS, headertype, InChNumArray, InChTypeArray,
totalInCount, woption ,0);
…
HS_Device_Release (hHS);
```

[C#]

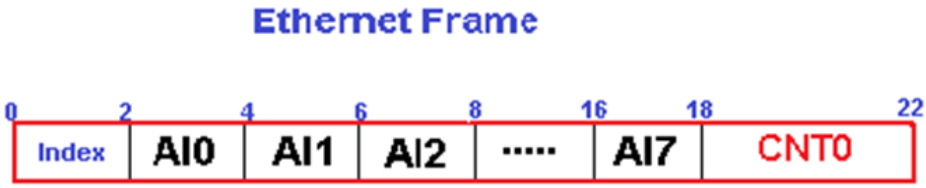
```
IntPtr hHS;
short ChCnt = 8;            //scan channel is 8
short Gain=0;               //AI input type is 0 (+/- 5V)
short TrigMode = 0;         // Software trigger
int sampleRate = 2000; // sampling rate is 2KHz
Uint32 targetCnt = 0;    //Continue mode
short DataTransMethod = 0;
short AutoRun = 0;
UInt32 headertype=1; // Package index for the header of data frame
UInt16 [] InChNumArray= new Uint16[2];
UInt16 [] InChTypeArray= new UInt16[2];
UInt16 totalInCount=2; //scan 2 input type for synchronous input data acquisition.
UInt32 woption= SYNC_STATUS.SYNC_ENABE; //enable synchronous input data
acquisition.

//First input type is AI (Hex format) with scanning 8 channels
InChNumArray[0]=8;
InChTypeArray[0]= SYNC_IN_TYPE.SYNC_IN_AI_HEX;

//2'nd input type is Counter (32-bit) with scanning 1 channels
InChNumArray[1]=1;
InChTypeArray[1]= SYNC_IN_TYPE.SYNC_IN_DWORD_CNT;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_SetAIScanParam(hHS, ChCnt, Gain, TrigMode,
sampleRate, targetCnt, DataTransMethod, AutoRun);
HSDAQNet.HSIO.HS_SetSyncInScanParam (hHS, headertype, InChNumArray,
InChTypeArray, totalInCount, woption, 0);
…
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Thd frame data for VC and C# example above is as



Remark

- In synchronous input data acquisition. Use HS_SetAIScanParam function to set the trigger mode (4th parameter of HS_SetAIScanParam) to 0(software trigger) and set the target count (6th parameter of HS_SetAIScanParam) to 0 (continuous mode).
- In synchronous input data acquisition, the sampling rate (5th parameter of HS_SetAIScanParam) can be 2KHZ Max

2.4.15. HS_GetSyncInBuffer

Get the parameter of the synchronous input data acquisition from PET-7H16M

Syntax

C/C++

```
DWORD HS_GetSyncInBuffer (
    HANDLE hobj,
    void *packetheader,
    void **wfAIBuffer,
    BYTE **bDIBuffer,
    BYTE **bDOBuffer,
    void **pDICNTbuffer,
    void **pCNTbuffer,
    void **pUDbuffer1,
    void **pUDbuffer2,
    DWORD dwFrameDataNumber
);
```

.Net

```
    UInt32 HS_GetSyncInBuffer(
        IntPtr hobj,
        IntPtr packetheader,
        IntPtr[] wfAIBuffer,
        IntPtr bDIBuffer,
        IntPtr bDOBuffer,
        IntPtr[] pDICNTbuffer,
        IntPtr[] pCNTbuffer,
        IntPtr[] pUDbuffer1,
        IntPtr[] pUDbuffer2,
        UInt32 dwFrameDataNumber
    );
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

packetheader

[Out] *the buffer containing the package index or packet timestamp*

wfAIBuffer

[out]the 2-D buffer containing synchronous AI value (Hex, float)

bDIBuffer

[out] the 2-D buffer containing DI value

bDOBuffer

[out] the 2-D buffer containing DO read-back value

pDICNTbuffer

[out] the 2-D buffer containing DI counter value (WORD/DWORD)

pCNTbuffer

[out] the 2-D buffer containing counter value (WORD/DWORD)

pUDbuffer1

[out] the 2-D buffer containing the user-defined array

pUDbuffer2

[out] the 2-D buffer containing the user-defined array

dwFrameDataNumber

the number of data frame in the data buffer

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=8;                  //scan channel is 8
short   useGain=0;              //AI input type is 0 (+/- 5V)
short extriggMode=0;            // Software trigger
```

```
unsigned long targetCnt=0;    //Continue mode
long sampleRate=2000;         // sampling rate is 2KHz
short DataTransMethod=0;
short AutoRun=0;

int SyInTypeNum=2; //scan 2 input types for synchronous input data acquisition
int SyncInheader=1; // Package index for the header of data frame
WORD SyInChTypeArray[2]={SYNC_IN_AI_HEX,SYNC_IN_DWORD_CNT};
WORD SyInChNumArray[2]={chCnt,1};
// 2 input types
//1st input type is AI (Hex format) with scanning 8 channels
//2nd input type is Counter (32-bit) with scanning 1 channels
DWORD dwCNTmode=CNT_ENABLE;
DWORD dwCNTInitValue=0;
WORD BufferStatus=0;
unsigned long ulFramelength=0

hHS = HS_Device_Create("192.168.1.1");
ret=HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate,
targetCnt, DataTransMethod,AutoRun);
HS_SetCounterConfig(hHS,0,dwCNTmode,dwCNTInitValue,0);
HS_SetSyncInScanParam(hHS,SyncInheader,SyInChNumArray,SyInChTypeArray
,SyInTypeNum,SYNC_ENABE,0);

HS_StartAIScan (hHS);
ret=HS_GetSyncInBufferStatus(hHS,&BufferStatus,&ulFramelength);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
      if(ulFramelength)
      {
         DWORD *dwheaderbuf=NULL;
         void *vheaderpointer=NULL;

         WORD **wAIbuffer=NULL;
         DWORD **wCNTbuffer=NULL;
         void **vAIbuffpointer=NULL;
```

```
        void **vCNTbuffpointer=NULL;

        dwheaderbuf=new DWORD[ulFramelength];
        vheaderpointer=dwheaderbuf;
        for(int i=0;i<SyInTypeNum;i++)
        {
           switch(SyInChTypeArray[i])
           {
                 case SYNC_IN_AI_HEX: //2bytes
                    wAIbuffer=NEW2D(ulFramelength, SyInChNumArray[i], WORD);
                    vAIbuffpointer=(void **)wAIbuffer;
                 break;
                 case SYNC_IN_DWORD_CNT:    //4bytes
                    wCNTbuffer=NEW2D(ulFramelength, SyInChNumArray[i],
DWORD);
                    vCNTbuffpointer=(void **)wCNTbuffer;
                 break;
                …
           }
         }
         readsize=HS_GetSyncInBuffer(hHS,vheaderpointer,vAIbuffpointer,NULL,N
ULL,NULL,vCNTbuffpointer,NULL,NULL,ulFramelength);
         if(readsize)
         {
         …
         }
      }
}
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

Remark

It's recommended to use HS_GetSyncInBuffer1D function of HSDAQnet.dll for C# demo.

2.4.16. HS_GetSyncInBufferLV

Get the parameter of the synchronous input data acquisition from PET-7H16M

Syntax

C/C++

```
DWORD HS_GetSyncInBufferLW (
   HANDLE hobj,
   DWORD *packetheader,
   DWORD *wfAIBuffer,
   BYTE *bDIBuffer,
   BYTE *bDOBuffer,
   DWORD *pDICNTbuffer,
   DWORD *pCNTbuffer,
   DWORD *pUDbuffer1,
   DWORD *pUDbuffer2,
   DWORD dwFrameDataNumber
);
```

.Net

```
    UInt32 HS_GetSyncInBuffer1D(
      IntPtr hobj,
      UInt32[] packetheader,
      UInt32[] wfAIBuffer,
      ushort[] bDIBuffer,
      ushort[] bDOBuffer,
      UInt32[] pDICNTbuffer,
      UInt32[] pCNTbuffer,
      UInt32[] pUDbuffer1,
      UInt32[] pUDbuffer2,
      UInt32 dwFrameDataNumber
      );
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

packetheader

[out] the buffer containing the package index or packet timestamp

wfAIBuffer

[out] the 1-D buffer containing synchronous AI value (Hex, float)

bDIBuffer

[out] the 1-D buffer containing DI value

bDOBuffer

[out] the 1-D buffer containing DO read-back value

pDICNTbuffer

[out] the 1-D buffer containing DI counter value (WORD/DWORD)

pCNTbuffer

[out] the 1-D buffer containing counter value (WORD/DWORD)

pUDbuffer1

[out] the 1-D buffer containing the user-defined array

pUDbuffer2

[out] the 1-D buffer containing the user-defined array

dwFrameDataNumber

the number of data frame in the data buffer

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=8;                  //scan channel is 8
short    useGain=0;             //AI input type is 0 (+/- 5V)
short extriggMode=0;            // Software trigger
```

```
unsigned long targetCnt=0;    //Continue mode
long sampleRate=2000;         // sampling rate is 2KHz
short DataTransMethod=0;
short AutoRun=0;

int SyInTypeNum=2; //scan 2 input types for synchronous input data acquisition
int SyncInheader=1; // Package index for the header of data frame
WORD SyInChTypeArray[2]={SYNC_IN_AI_HEX,SYNC_IN_DWORD_CNT};
WORD SyInChNumArray[2]={chCnt,1};
// 2 input types
//1st input type is AI (Hex format) with scanning 8 channels
//2nd input type is Counter (32-bit) with scanning 1 channels
DWORD dwCNTmode=CNT_ENABLE;
DWORD dwCNTInitValue=0;
WORD BufferStatus=0;
unsigned long ulFramelength=0

hHS = HS_Device_Create("192.168.1.1");
ret=HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate,
targetCnt, DataTransMethod,AutoRun);
HS_SetCounterConfig(hHS,0,dwCNTmode,dwCNTInitValue,0);
HS_SetSyncInScanParam(hHS,SyncInheader,SyInChNumArray,SyInChTypeArray
,SyInTypeNum,SYNC_ENABE,0);

HS_StartAIScan (hHS);

ret=HS_GetSyncInBufferStatus(hHS,&BufferStatus,&ulFramelength);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
      if(ulFramelength)
      {
         DWORD *dwheaderbuf=NULL;

         DWORD *wAIbuffer=NULL;
         DWORD *wCNTbuffer=NULL;
```

```
        dwheaderbuf=new DWORD[ulFramelength];
        for(int i=0;i<SyInTypeNum;i++)
        {
           switch(SyInChTypeArray[i])
           {
              case SYNC_IN_AI_HEX: //2bytes
                 temp=ulFramelength*SyInChNumArray[i];
                 wAIbuffer=new DWORD[temp];
                 memset(wAIbuffer,0x0,sizeof(DWORD)*temp);
                 break;
              case SYNC_IN_DWORD_CNT:    //4bytes
                 temp=ulFramelength*SyInChNumArray[i];
                 wCNTbuffer=new DWORD[temp];
                 memset(wCNTbuffer,0x0,sizeof(DWORD)*temp);
                 break;
               …
           }
          }
           readsize=HS_GetSyncInBufferLV(hHS,dwheaderbuf,wAIbuffer,NULL,NU
LL,NULL,wCNTbuffer,NULL,NULL,ulFramelength);
          if(readsize)
          {
          …
          }
      }
}
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short ChCnt = 8;    //scan channel is 8
short Gain = 0;     //AI input type is 0 (+/- 5V)
uint ulFramelength = 0;
ushort BufferStatus = 0;
short TrigMode = 0;     // Software trigger
UInt32 targetCnt = 0;    //Continue mode
```

```
int sampleRate = 2000;    // sampling rate is 2KHz
short DataTransMethod = 0;
short AutoRun = 0;

hHS =Sys.HS_Device_Create("192.168.1.1");
HSIO.HS_SetAIScanParam(hHS, ChCnt, Gain, TrigMode, sampleRate, targetCnt,
DataTransMethod, AutoRun);

UInt32 dwCNTmode = (UInt32)IO.CNTCong.CNT_DISABLE;
UInt32 dwCNTInitValue = 0;
ret=IO.HS_SetCounterConfig(hHS,0, dwCNTmode, dwCNTInitValue, 0);

UInt16 SyInTypeNum = 2;
//scan 2 input types for synchronous input data acquisition
UInt32 SyncInheader = 1; // Package index for the header of data frame
UInt32 options = (UInt32)HSIO.SYNC_STATUS.SYNC_ENABE;

UInt16[] SyInChTypeArray = new
UInt16[]{(UInt16)HSIO.SYNC_IN_TYPE.SYNC_IN_AI_HEX ,(UInt16)HSIO.SYNC_
IN_TYPE.SYNC_IN_DWORD_CNT};
UInt16[] SyInChNumArray = new UInt16[] { (UInt16)ChCnt, 1 };
// 2 input types
//1st input type is AI (Hex format) with scanning 8 channels
//2nd input type is Counter (32-bit) with scanning 1 channels

ret = HSIO.HS_SetSyncInScanParam(hHS,SyncInheader,SyInChNumArray,
SyInChTypeArray,(UInt16) SyInTypeNum,(UInt32)options, 0);

HSIO.HS_StartAIScan(hHS);
ret = HSIO.HS_GetSyncInBufferStatus(hHS, ref BufferStatus, ref ulFramelength);
if (ret == false)
{
Console.WriteLine("Error code3 0x" + ErrHandling.GetLastError().ToString("x8"));
}
else
{
   if (ulFramelength > 0)
   {
      UInt32[] dwheaderbuf = null;
```

```
        UInt32[] wAIbuffer = null;
        UInt32[] wCNTbuffer = null;
        dwheaderbuf = new UInt32[ulFramelength];
        for (i = 0; i < SyInTypeNum; i++)
        {
           switch (SyInChTypeArray[i])
           {
              case 1: //HSIO.SYNC_IN_TYPE.SYNC_IN_AI_HEX: //2bytes
                 temp = ulFramelength * SyInChNumArray[i];
                 wAIbuffer = new UInt32[temp];
                 break;
            case 5: // HSIO.SYNC_IN_TYPE.SYNC_IN_DWORD_CNT:    //4bytes
                 temp = ulFramelength * SyInChNumArray[i];
                 wCNTbuffer = new UInt32[temp];
                 break;
                 …
                 }
                 }
                 readsize = HSIO.HS_GetSyncInBuffer1D(hHS, dwheaderbuf, wAIbuffer,
null, null, NULL, wCNTbuffer, null, null, ulFramelength);
            if(readsize)
            {
            …
            }
        }
     …
}
Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.17. HS_GetSyncInBufferStatus

Get the status of the synchronous input data acquisition and data number from data buffer on PC.

Syntax

C/C++

```
DWORD HS_GetSyncInBufferStatus (
  HANDLE hobj,
  WORD *wBufferStatus,
  DWORD *dwDataCountOnBuffer
);
```

.Net

```
bool HS_GetSyncInBufferStatus (
  IntPtr hobj,
  ref UInt16 wBufferStatus,
  ref UInt32 dwDataCountOnBuffer
);
```

Parameter

hobj
[in] A handle to the specified device opened by HS_Device_Create

wBufferStatus
[out] The buffer status
0: the data buffer is empty
1: the data exists in the buffer
2: the data buffer is overflow
4: the synchronous input data acquisition scan is stopped

dwDataCountOnBuffer
[out] The frame number in the buffer of synchronous input data acquisition on PC

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=8;                  //scan channel is 8
short    useGain=0;             //AI input type is 0 (+/- 5V)
short extriggMode=0;            // Software trigger
unsigned long targetCnt=0;    //Continue mode
long sampleRate=2000;         // sampling rate is 2KHz
short DataTransMethod=0;
short AutoRun=0;

int SyInTypeNum=2; //scan 2 input types for synchronous input data acquisition
int SyncInheader=1; // Package index for the header of data frame
int AcutArryAcount=0;
int options=0;
WORD SyInChTypeArray[2]={SYNC_IN_AI_HEX,SYNC_IN_DWORD_CNT};
WORD SyInChNumArray[2]={chCnt,1};
// 2 input types
//1st input type is AI (Hex format) with scanning 8 channels
//2nd input type is Counter (32-bit) with scanning 1 channels
DWORD dwCNTmode=CNT_ENABLE;
DWORD dwCNTInitValue=0;
WORD BufferStatus=0;
unsigned long ulFramelength=0

hHS = HS_Device_Create("192.168.1.1");
ret=HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate,
targetCnt, DataTransMethod,AutoRun);
HS_SetCounterConfig(hHS,0,dwCNTmode,dwCNTInitValue,0);
HS_SetSyncInScanParam(hHS,SyncInheader,SyInChNumArray,SyInChTypeArray
,SyInTypeNum,SYNC_ENABE,0);
```

```
HS_StartAIScan (hHS);

ret=HS_GetSyncInBufferStatus(hHS,&BufferStatus,&ulFramelength);
if(ret==false){
   printf("Error code 0x%x\r\n",HS_GetLastError());
}
else
{
      if(ulFramelength)
      {
         DWORD *dwheaderbuf=NULL;

         DWORD *wAIbuffer=NULL;
         DWORD *wCNTbuffer=NULL;

         dwheaderbuf=new DWORD[ulFramelength];
         for(int i=0;i<SyInTypeNum;i++)
         {
            switch(SyInChTypeArray[i])
            {
               case SYNC_IN_AI_HEX: //2bytes
                  temp=ulFramelength*SyInChNumArray[i];
                  wAIbuffer=new DWORD[temp];
                  memset(wAIbuffer,0x0,sizeof(DWORD)*temp);
                  break;
               case SYNC_IN_DWORD_CNT:    //4bytes
                  temp=ulFramelength*SyInChNumArray[i];
                  wCNTbuffer=new DWORD[temp];
                  memset(wCNTbuffer,0x0,sizeof(DWORD)*temp);
                  break;
                 …
            }
           }
            readsize=HS_GetSyncInBufferLV(hHS,dwheaderbuf,wAIbuffer,NULL,NU
LL,NULL,wCNTbuffer,NULL,NULL,ulFramelength);
           if(readsize)
           {
           …
```

```
            }
        }
}
HS_StopAIScan (hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
float[] HdataBuffer = new float[10000];
uint ulleng = 0;
ushort BufferStatus = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_StartAIScan(hHS);
ret = HSDAQNet.HSIO.HS_GetAIBufferStatus(hHS, ref BufferStatus, ref ulleng);
if (ret == false)
{
   Console.WriteLine(IP.ToString() + " Error code 0x" +
HSDAQNet.ErrHandling.GetLastError().ToString("x8"));
}
else
{
   if (BufferStatus > 2) //AI buffer overflow
   {
     /*
     2:   AD_BUF_OVERFLOW
     4:   AD_SCAN_STOP
     8:   AD_DATA_SAMPLING_TIMEOUT
     */
        Console.WriteLine(IP.ToString() + " Error<" + BufferStatus + ">," + ulleng);
        break;
   }
   if (ulleng == targetCnt)
   {
      readsize = HSDAQNet.HSIO.HS_GetAIBufferHex(hHS, HdataBuffer, ulleng);
   }
}
```

```
HSDAQNet.HSIO.HS_StopAIScan(hHS);
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.18. HS_ClearSyncInBuffer

Clear the buffer of synchronous input data acquisition on PC.

Syntax

C/C++

```
bool HS_ClearSyncInBuffer(
   HANDLE hobj
);
```

.Net

```
bool HS_ClearSyncInBuffer(
   IntPtr hobj
);
```

Parameter

Hobj
[in] A handle to the specified device opened by HS_Device_Create.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;

hHS = HS_Device_Create("192.168.1.1");
```

```
HS_ClearAIBuffer(hHS);
ret=HS_StartAIScan (hHS);
…//user-define code
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_ClearAIBuffer(hHS);
```

```
HSDAQNet.HSIO.HS_StartAIScan(hHS);
… //user-define code
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

None.

2.4.19. HS_GetSyncInTotalSamplingStatus

Read the status of ET-7H16 during synchronous input data acquisition.

Syntax

C/C++

```
bool HS_GetSyncInTotalSamplingStatus (
   HANDLE hobj,
   unsigned long * totalReadCnt,
   unsigned int * SamplingStatus
);
```

.Net

```
bool HS_GetSyncInTotalSamplingStatus (
   IntPtr hobj,
   ref UInt32 totalReadCnt,
   ref UInt32 triggerStatus
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

totalReadCnt

[out] The count of the sampling data stored in the memory on the PET-7H16M module

SamplingStatus

[out] The trigger status of PET-7H16M module in data sampling
1: The digital signal is triggered in the pre-trigger mode
0: The signal isn't triggered.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

None.

2.4.20. HS_SetAIAnalogTriggerParam

Set the AI trigger parameter for PET-7H16M.

Syntax

C/C++

```
bool    HS_SetAIAnalogTriggerParam(
    HANDLE obj,
    int analogmode,
    bool En_Channel[],
    float hightriglevel[],
    float lowtriglevel[],
    int totalSetchannel,
    unsigned long leftsidecnt,
    unsigned long rightsidecnt,
    unsigned long RESERVED
);
```

.Net

```
bool HS_ SetAIAnalogTriggerParam (
    IntPtr hobj,
    int analogmode,
    char[] En_Channel,
    float[] hightriglevel,
    float[] lowtriglevel,
    UInt32 leftsidecnt,
    UInt32 rightsidecnt,
    UInt32 RESERVED
);
```

Parameter

hobj

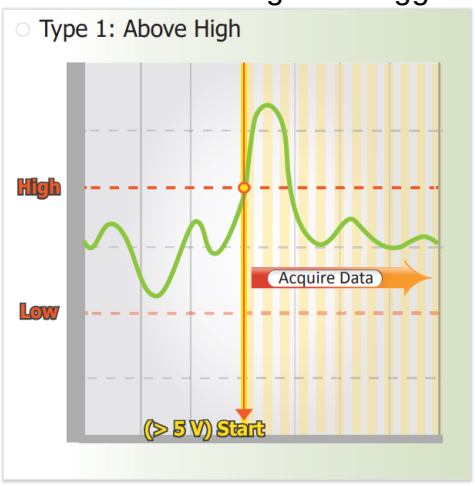
[in] A handle to the specified device opened by HS_Device_Create

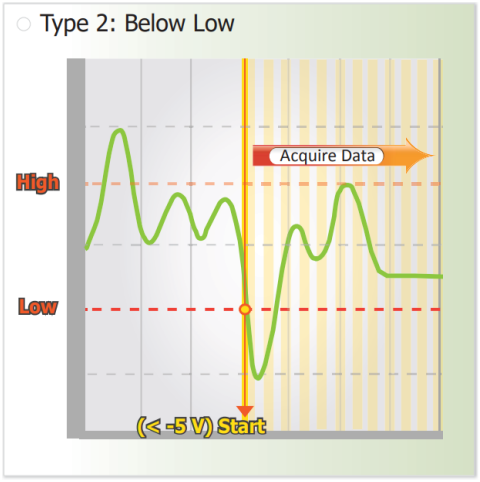
analogmode

[in] : Set Analog input trigger mode(0 : Above high, 1 : Below low, 2 : Leave-region, 3 : Entry-region)

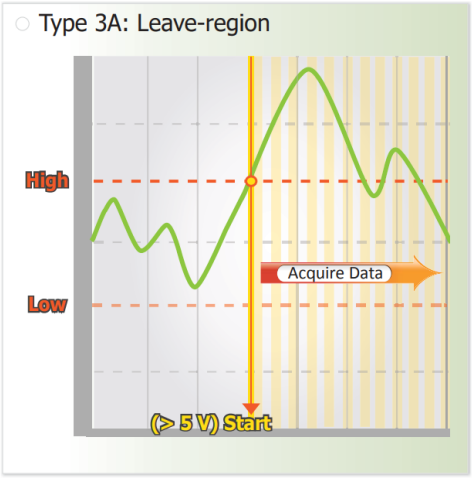
Above High: The signal is triggered above the high level and collects N data.

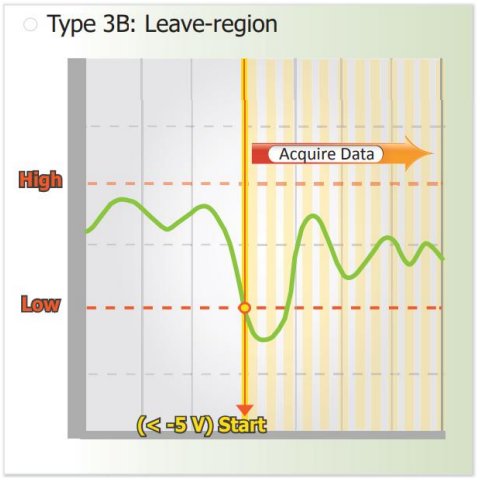
Below Low : The signal is triggered below the low level and collects N data



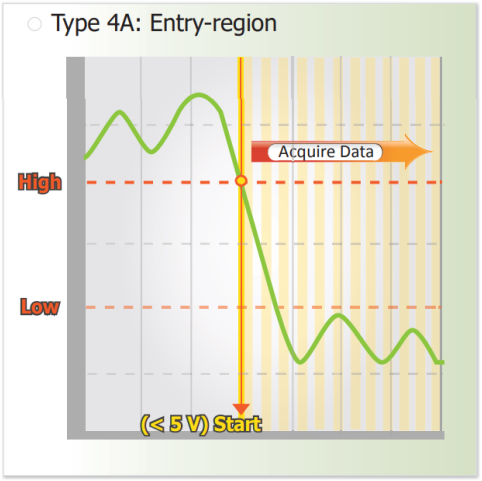


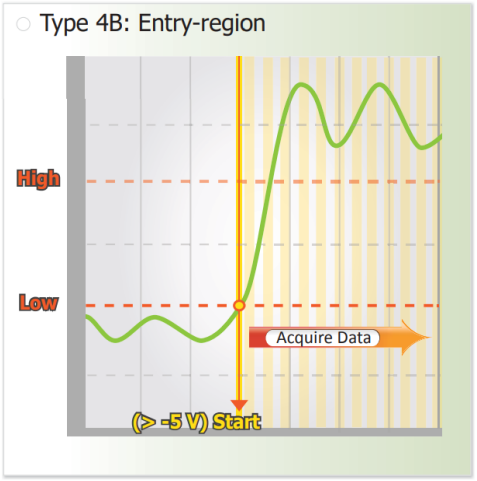
Leave-region : Trigger when the signal leaves the high and low level region, collect N data





Entry-region : Trigger when the signal enters the high and low level region, collect N data





.

En_Channel
[in] Enable/Disable AI trigger function for each channel, and one element of the array represents an AI channel.

hightriglevel
[in] The float arrary is used for set the high level value for each AI channel.

lowtriglevel
[in] The float arrary is used for set the low level value for each AI channel.

totalSetchannel
[in] set the array length for En_Channel/ hightriglevel/ lowtriglevel array

leftsidecnt
[in] Set the number of AI scan target count before triggered.

rightsidecnt
[in] Set the number of AI scan target count after triggered.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=8;                //scan channel is 8
short useGain=0;              //AI input type is 0 (+/- 5V)
short extriggMode= 6;     //6 : Analog Input Mode
long sampleRate=1000;    // sampling rate is 1KHz
unsigned long targetCnt=400; // target count is 400, In Anloag Input Mode, the targetCnt value must equal to leftsidecnt and rightsidecnt value
short DataTransMethod=0;
short AutoRun=0;
int analogmode=2; //AI triger mode sets to Entry-region
bool En_Channel[8]={1,1,1,1,1,1,1,1} //Enable all AI channels for AI trigger
float hightriglevel[8]={4.0,4.0,4.0,4.0,4.0,4.0,4.0,4.0};
//set high level value for all AI channels
float lowtriglevel[8]= {1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0};
```

```
//set Low level value for all AI channels
unsigned long leftsidecnt=200; // target count is 200 before triggered
unsigned long rightsidecnt=200; // target count is 200 after triggered

hHS = HS_Device_Create("192.168.1.1");
ret=HS_SetAIScanParam(hHS, chCnt, useGain, extriggMode, sampleRate,
targetCnt, DataTransMethod,AutoRun);
ret =
HS_SetAIAnalogTriggerParam(hHS,analogmode,En_Channel,hightriglevel,lowtrigl
evel,8,leftsidecnt,rightsidecnt,0);
…
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short ChCnt = 8;            //scan channel is 8
short Gain=0;               //AI input type is 0 (+/- 5V)
short TriggerMode = 6; //Analog Input Trigger
int sampleRate = 1000; // sampling rate is 1KHz
Uint32 targetCnt = 400;    // target count is 400, In Anloag Input Mode, the targetCnt
value must equal to leftsidecnt and rightsidecnt value 0
short DataTransMethod = 0;
short AutoRun = 0;
short AItriggmode=2; //AI triger mode sets to Entry-region
char[] Ch_En_arr = new char[] {1,1,1,1,1,1,1,1};
float[] HighLevel_arr = new float[] {4.0,4.0,4.0,4.0,4.0,4.0,4.0,4.0};
//set high level value for all AI channels
float[] LowLevel_arr = new float[] {1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0};
//set Low level value for all AI channels
UInt32 leftsidecnt=200; // target count is 200 before triggered
UInt32 rightsidecnt=200; // target count is 200 after triggered

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_SetAIScanParam(hHS, ChCnt, Gain, TrigMode,
sampleRate, targetCnt, DataTransMethod, AutoRun);
HSDAQNet.HSIO.HS_SetAIAnalogTriggerParam(hHS, AItriggmode , Ch_En_arr,
HighLevel_arr, LowLevel_arr,8, leftsidecnt, rightsidecnt, (uint)0)
…
```

```
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

- In the Anloag input mode, Use HS_SetAIScanParam function to set the triggerMode (4th parameter of HS_SetAIScanParam) to 6.
- In the Anloag input mode, the targetCnt value (6th parameter of HS_SetAIScanParam) must be equal to the total value of leftsidecnt (AI scan target count before triggered, 6th parameter of HS_SetAIAnalogTriggerParam) and rightsidecnt (AI scan target count after triggered, 7th parameter of HS_SetAIAnalogTriggerParam).

2.4.21. HS_GetAIAnalogTriggerParam

Get the AI trigger parameter for PET-7H16M.

Syntax

C/C++

```
bool    HS_GetAIAnalogTriggerParam(
    HANDLE obj,
    Int* analogmode,
    bool En_Channel[],
    float hightriglevel[],
    float lowtriglevel[],
    int totalSetchannel,
    unsigned long* leftsidecnt,
    unsigned long* rightsidecnt,
    unsigned long* RESERVED
    );
```

.Net

```
bool HS_GetAIAnalogTriggerParam (
    IntPtr hobj,
    Ref int analogmode,
    bool[] En_Channel,
    float[] hightriglevel,
    float[] lowtriglevel,
    ref UInt32 leftsidecnt,
    ref UInt32 rightsidecnt,
    ref UInt32 RESERVED
    );
```

Parameter

hobj

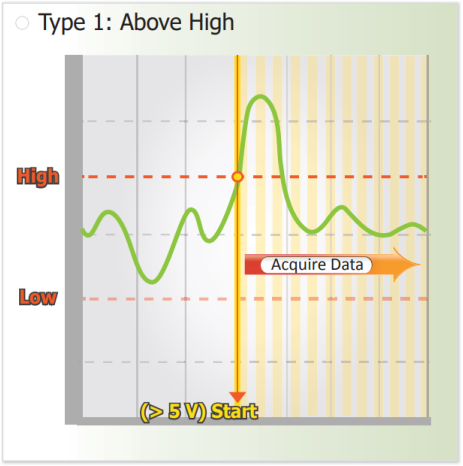
[in] A handle to the specified device opened by HS_Device_Create

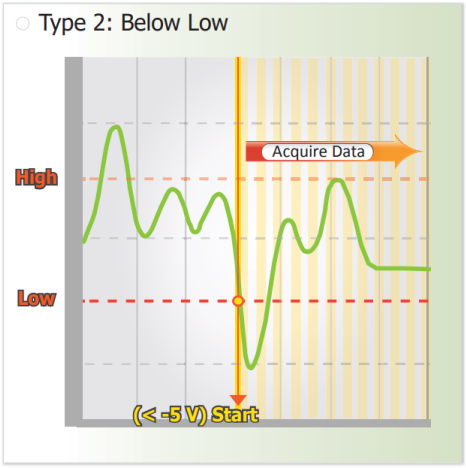
analogmode

[out] : Get Analog input trigger mode(0 : Above high, 1 : Below low, 2 : Leave-region, 3 : Entry-region)

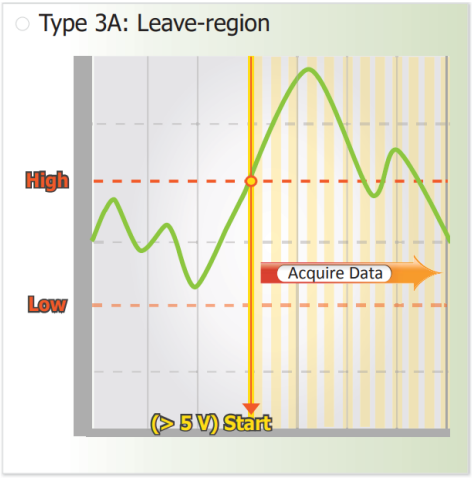
Above High: The signal is triggered above the high level and collects N data.

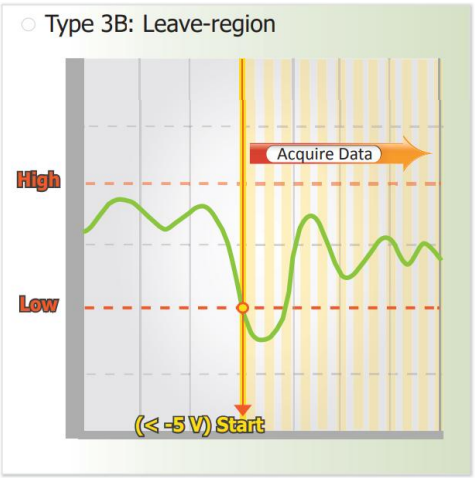
Below Low : The signal is triggered below the low level and collects N data



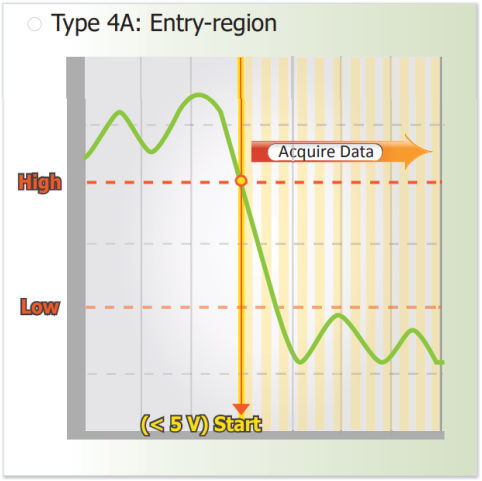


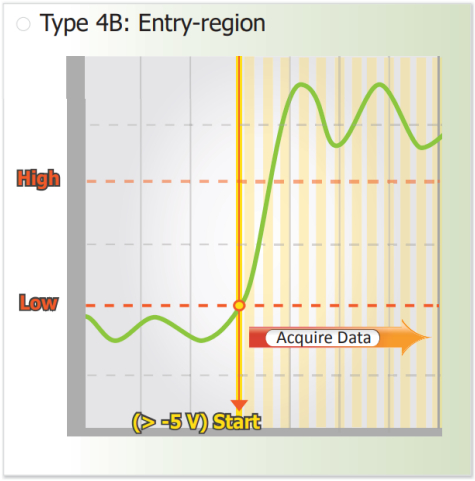
Leave-region : Trigger when the signal leaves the high and low level region, collect N data





Entry-region : Trigger when the signal enters the high and low level region, collect N data





.

En_Channel
[Out] Get theEnable/Disable status of AI trigger function for each channel, and one element of the array represents an AI channel.

hightriglevel
[in] The float arrary is used for get the high level value for each AI channel.

lowtriglevel
[in] The float arrary is used for get the low level value for each AI channel.

totalSetchannel
[in] set the array length for En_Channel/ hightriglevel/ lowtriglevel array

leftsidecnt
[in] Get the number of AI scan target count before triggered.

rightsidecnt
[in] Get the number of AI scan target count after triggered.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
short chCnt=0;
short useGain=0;
short extriggMode=0;
long sampleRate=0;
unsigned long targetCnt=0;
short DataTransMethod=0;
short AutoRun=0;
int analogmode=0;
bool En_Channel[8];
float hightriglevel[8];
float lowtriglevel[8];
unsigned long leftsidecnt=0;
```

```
unsigned long rightsidecnt=0;

hHS = HS_Device_Create(“192.168.1.1”);
HS_GetAIScanParam(hHS, &chCnt, &useGain, &extriggMode, &sampleRate,
&targetCnt, &DataTransMethod, &AutoRun);
```

```
If(extriggMode==6) //Analog Input trigger
   HS_GetAIAnalogTriggerParam(hHS,&analogmode,En_Channel,hightriglevel,low
triglevel,8,&leftsidecnt,&rightsidecnt,0);
…
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short rChCnt = 0;
short rGain=0;
short rTrigMode = 0;
int rsampleRate = 0;
UInt32 rtargetCnt = 0;
short rDataTransMethod = 0;
short rAutoRun = 0;
short AItriggmode=0;
char[] Ch_En_arr;
float[] HighLevel_arr;
float[] LowLevel_arr;
UInt32 leftsidecnt=0;
UInt32 rightsidecnt=0;
uint RESERVED=0;

hHS = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.HSIO.HS_GetAIScanParam(hHS, ref rChCnt, ref rGain, ref rTrigMode,
ref rsampleRate, ref rtargetCnt, ref rDataTransMethod, ref rAutoRun);
If(rTrigMode ==6) //Analog Input trigger
   HSDAQNet.HSIO.HS_GetAIAnalogTriggerParam(hHS, ref analogmode,
En_Channel, hightriglevel, lowtriglevel, 8, ref   leftsidecnt, ref   rightsidecnt, ref
RESERVED)
```

```
...
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remark

In the Anloag input mode, Use HS_SetAIScanParam function to set the triggerMode (4th parameter of HS_SetAIScanParam) to 6.

2.4.22. HS_SetAIDelayTriggerParam

Set the delay trigger parameter for high speed data acquisition

Syntax

C/C++

```
bool HS_SetAIDelayTriggerParam (
   HANDLE hobj,
   unsigned long delaytime,
   unsigned long RESERVED
);
```

.Net

```
bool HS_SetAIDelayTriggerParam (
   IntPtr hobj,
   UInt32 delaytime,
   UInt32 RESERVED
   );
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

delaytime

[in] Set the delay time for delay trigger, 5μs ~ 10s (1 unit: 1μs)

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
int chCnt=8;                //scan channel is 8
int useGain=0;              //AI input type is 0 (+/- 5V)
int TrigMode = 5;           //5 : delay trigger
long sampleRate=1000; // sampling rate is 1KHz
unsigned long targetCnt=1000; // target count is 1000
int DataTransMethod=0;
int AutoRun=0;
unsigned long delaytime = 1000; //Set delay timer to 1000µs
unsigned long Reserv = 0;

hHS = HS_Device_Create("192.168.1.1");
HS_SetAIScanParam(hHS, chCnt, useGain, TrigMode, sampleRate, targetCnt,
DataTransMethod, AutoRun);
ret = HS_SetAIDelayTriggerParam(hHS,delaytime,0); //Set delay time of 1000µs for
delay trigger mode
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short ChCnt = 8;              //scan channel is 8
short Gain=0;                 //AI input type is 0 (+/- 5V)
short TriggerMode =   5;    //5 : delay trigger
int sampleRate = 1000;      // sampling rate is 1KHz
Uint32 targetCnt = 1000;    // target count is 1000
short DataTransMethod = 0;
short AutoRun = 0;
UInt32 delaytime = 1000; //Set delay timer to 1000µs
UInt32 Reserv = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_SetAIScanParam(hHS, ChCnt, Gain, TriggerMode,
sampleRate, targetCnt, DataTransMethod, AutoRun);
HSDAQNet.HSIO.HS_SetAIDelayTriggerParam (hHS, delaytime, Reserv);
//Set delay time of 1000µs for delay trigger mode
```

```
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

In the delay trigger mode, Use HS_SetAIScanParam function to set the triggerMode (4th parameter of HS_SetAIScanParam) to 5.

2.4.23. HS_GetAIDelayTriggerParam

Get the delay trigger parameter for high speed data acquisition

Syntax

C/C++

```
bool HS_GetAIDelayTriggerParam (
   HANDLE hobj,
   unsigned long delaytime,
   unsigned long RESERVED //Reserved
     DWORD* reserved
);
```

.Net

```
bool HS_GetAIDelayTriggerParam (
   IntPtr hobj,
   UInt32 wChannel,
   ref UInt32 wMode,
   ref UInt32 dwValue,
   ref UInt32 reserved
);
```

Parameter

hobj

[in] A handle to the specified device opened by HS_Device_Create

delaytime

[out] Get the delay time of delay trigger, 5µs ~ 10s (1 unit: 1µs)

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
int chCnt=0;
int useGain=0;
int TrigMode = 0;
long sampleRate=0;
unsigned long targetCnt=0;
int DataTransMethod=0;
int AutoRun=0;
unsigned long delaytime = 0;
unsigned long Reserv = 0;

hHS = HS_Device_Create("192.168.1.1");
HS_GetAIScanParam(hHS, &chCnt, &useGain, &TrigMode, &sampleRate,
&targetCnt, &DataTransMethod, &AutoRun);
If(TrigMode==5) / delay trigger
   HS_GetAIDelayTriggerParam(hHS,&delaytime,& Reserv);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hHS;
short rChCnt = 0;
short rGain =0;
short rTrigMode = 0;
int rsampleRate = 0;
Uint32 rtargetCnt = 0;
short rDataTransMethod = 0;
short rAutoRun = 0;
UInt32 delaytime = 0;
UInt32 Reserv = 0;

hHS = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.HSIO.HS_GetAIScanParam(hHS, ref rChCnt, ref rGain, ref rTrigMode,
ref rsampleRate, ref rtargetCnt, ref rDataTransMethod, ref rAutoRun);
If(rTrigMode ==5) / delay trigger
   HSDAQNet.HSIO.HS_GetAIDelayTriggerParam (hHS,ref delaytime, ref Reserv);
```

```
HSDAQNet.Sys.HS_Device_Release(hHS);
```

Remarks

In the delay trigger mode, Use HS_SetAIScanParam function to set the triggerMode (4th parameter of HS_SetAIScanParam) to 5..

2.5. Data Logger API

This chapter describes how to use the data logger API in VC/.NET programs.
The ET-7H16 module is equipped with a data logger function. The data recorded by the ET-7H16 module can be saved as a data logger file with different file types (.bin , .txt, and etcs).

The HS_StartLogger function is used to start the data logging and save data to the local storage disk of host PC with the specified file type (The default file type is binary file).The HS_StopLogger function is used to stop the data logging.

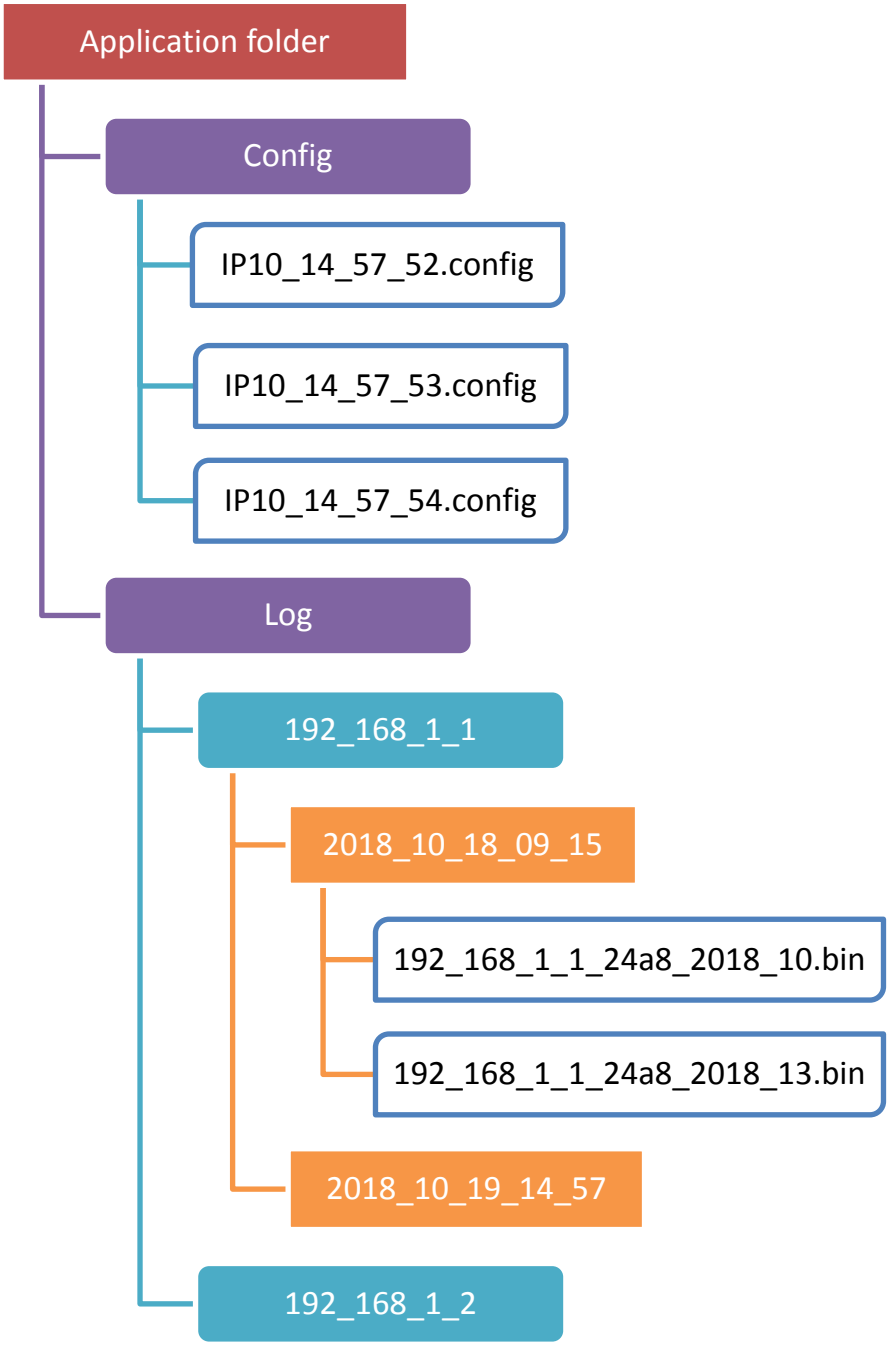
The data log file can be stored in the specified folder or in the application's folder (if no folder is specified), and the data directory structure is

\Config - Configuration folder

This folder stores the configuration files. The configuration file (.config) records the sampling rate of the log file, scanning channel and other information belong to the data log file.

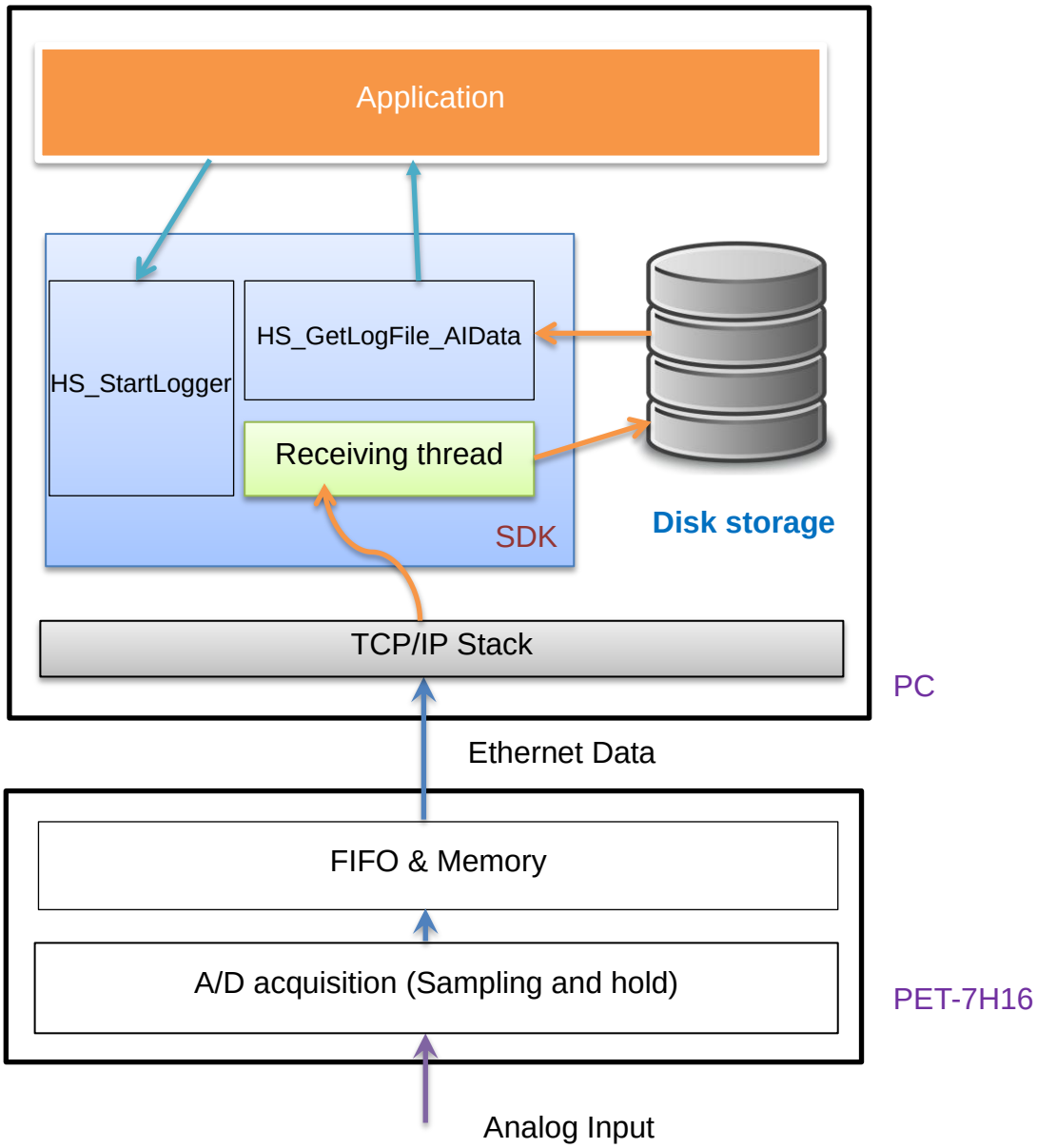
\log - Data logger folder

This folder stores data log files. The directory in the folder is divided into three layers. The directory name of the first layer is the ip address of ET-7H16M module, the directory name of the second layer is the date and time, The default format is yyyy-mm-dd-hh-mm (yyyy: year, mm: month, dd: day, hh: hour, mm: minute). This directory name can be changed by calling HS_SetConfig() function. The third layer is used to place the data log file.



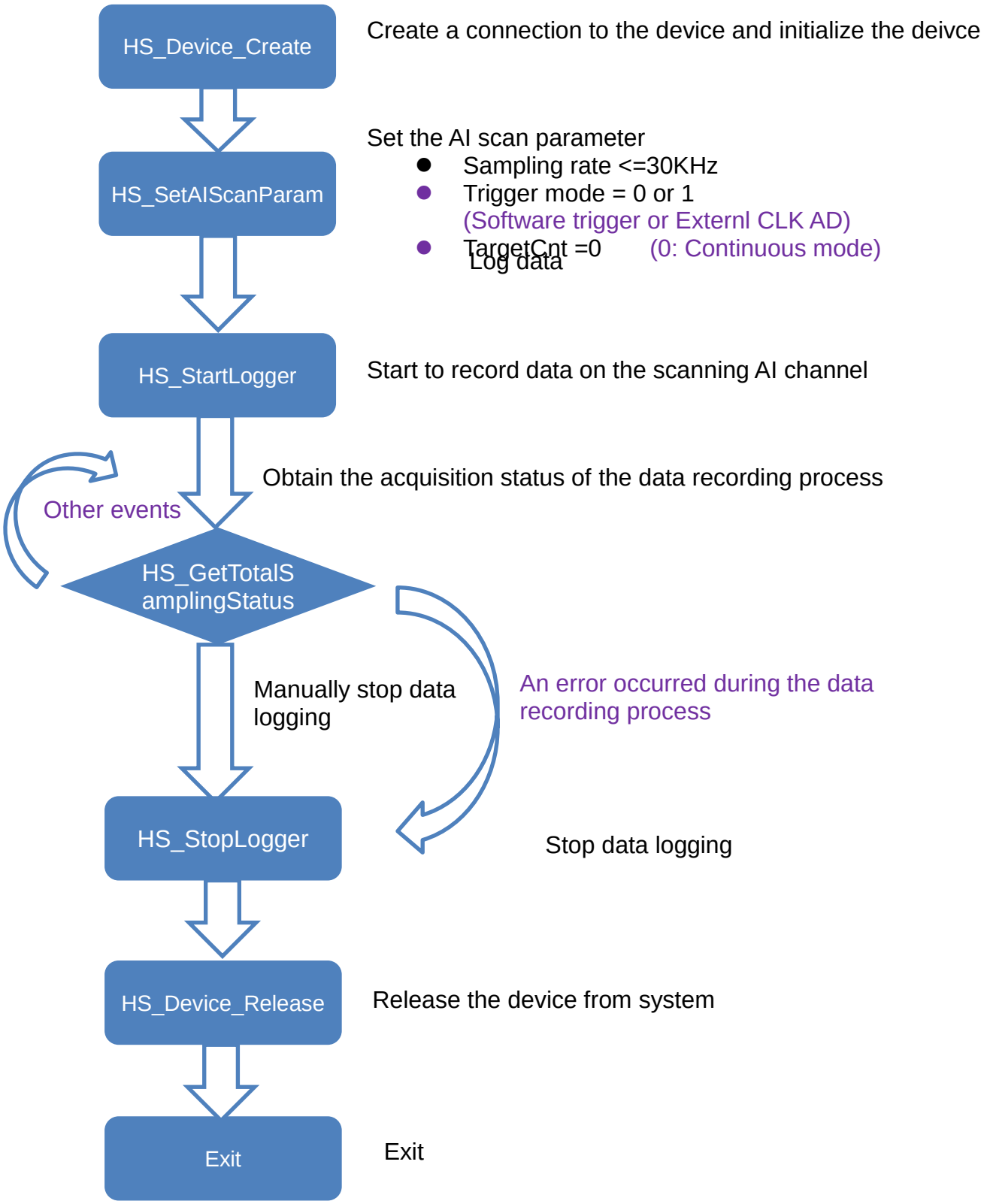
Use HS_StartLogger function to start the data logging and the thread of SDK will handle to receive the data and save them to the specified folder on storage disk of the host PC. Use the HS_GetAllLogFiles/ HS_GetLogFile_AIData/ HS_GetLogFile_AIDataHex functions to quickly find log files and read the log data.

Data flow chart



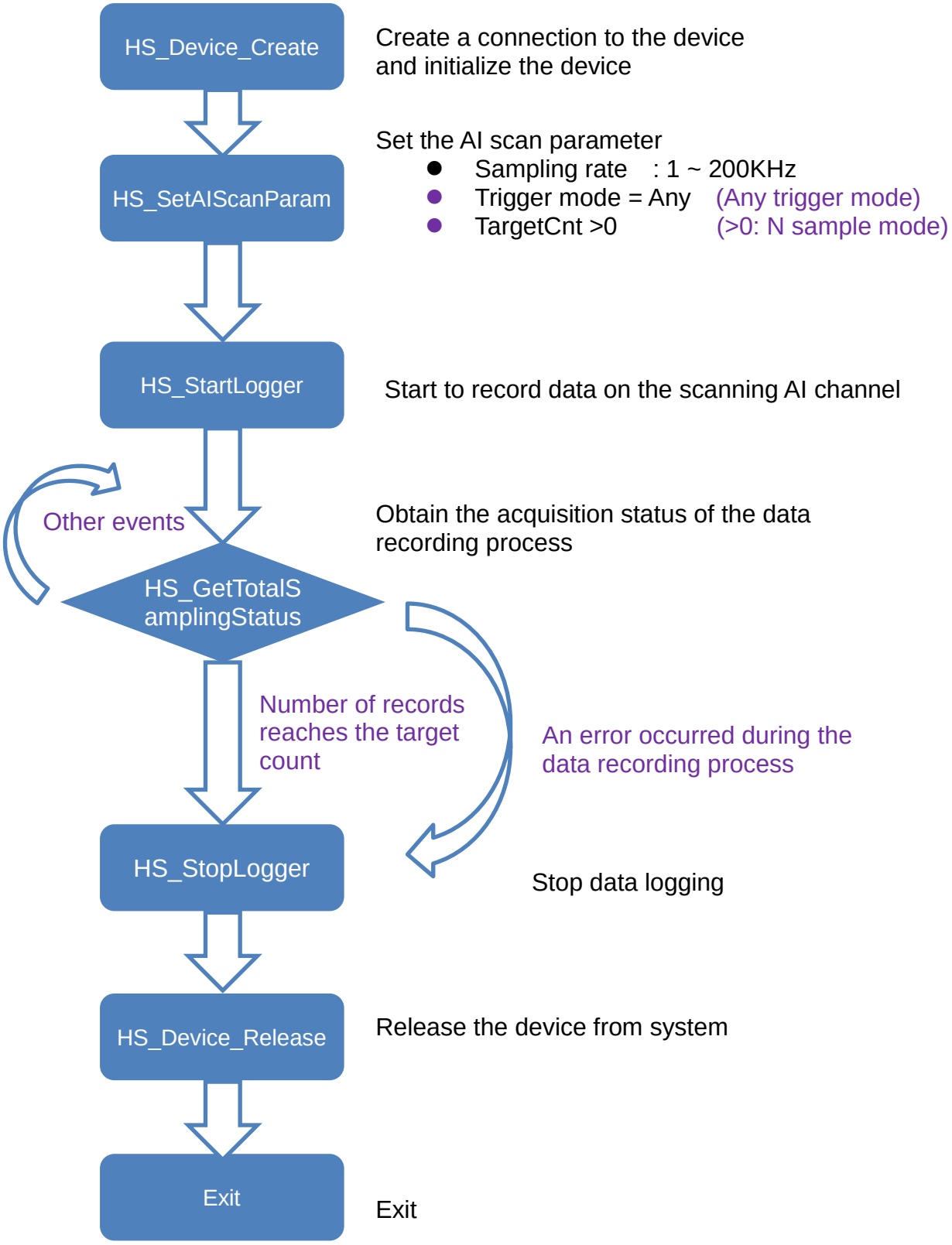
Continuous mode acquisition

API function call process chart



N sample mode acquisition

API function call process chart



Data logger Functions

The following functions are used for data logger.

HSDAQ Functions	HSDAQNET Functions	Description
HS_StartLogger	DATALOG.HS_StartLogger	Start the data logging and save data to the specified folder on storage disk of the host PC
HS_StartLoggerW		HS_StartLoggerW is a wide-character version of HS_StartLogger.The specified folder can included wild-character.
HS_StopLogger	DATALOG.HS_StopLogger	Stop the data logging.
HS_GetAllLogFiles	DATALOG.HS_GetAllLogFiles	Search all log files in the specified folder with the specified file type and return the total number of files
HS_LogFile_Open_byIndex	DATALOG.HS_LogFile_Open_byIndex	Open a data log file by the index number searched by HS_GetAllLogFiles
HS_LogFile_Open	DATALOG.HS_LogFile_Open	Open a data log file by the specified path and file name.
HS_LogFile_OpenW		HS_LogFile_OpenW is a wide-character version of HS_LogFile_Open.The specified folder and file name can included wild-character.
HS_LogFile_Close	DATALOG.HS_LogFile_Close	Closes a data log file opened by HS_LogFile_Open.
HS_GetLogFileInfo	DATALOG.HS_GetLogFileInfo	Get the data log file information including the file version and file size.
HS_GetLogFile_AIScanConfigInfo	DATALOG.HS_GetLogFile_AIScanConfigInfo	Get the data log file information regarding of the sampling rate, scan channels, pacer gain, trigger mode.
HS_GetLogFile_GainOffset	DATALOG.HS_GetLogFile_GainOffset	Get the data log file information regarding of the gain/offset values for each AI channel.
HS_GetLogFile_AIScanSampleInfo	DATALOG.HS_GetLogFile_AIScanSampleInfo	Get the totoal sampling counts and the starting time of first triggered sampling data in the data log file
HS_GetLogFile_AIData	DATALOG.HS_GetLogFile_AIData	Reads AI input data from the text file
HS_GetLogFile_AIDataHex	DATALOG.HS_GetLogFile_AIDataHex	Read AI input data(Hex) from the binary file

2.5.1. HS_StartLogger

This function is used to start the data logging and save data to the specified folder on storage disk of the host PC .

Syntax

C++

```
bool HS_StartLogger (
    HANDLE hobj,
    char *folderpath,
    int interval,
    int filetype
);

bool HS_StartLoggerW (
    HANDLE hobj,
    TCHAR *folderpath,
    int interval,
    int filetype
);
```

.Net

```
bool HS_ StartLogger (
  IntPtr hobj,
  string folderpath,
  int interval,
  int filetype
);
```

Parameter

hobj

[in] A handle to the ET-7H16 creaded by HS_Device_Create.

folderpath

[in] The specified folder for saving the data log file. If NULL, The data log file will

be saved to the current folder where the application is running

interval

[in] Specifies the interval time for changing to next file , in mintues.

filetype

[in] File type

0: binary file (.bin)

1: text file (.txt)

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create(“192.168.1.1”);
HS_StartLogger(hHS,NULL,2,0);
// The data log file (binary type) will be saved to the current folder where the application is running , 2 minutes of interval for changing to next file
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDev;
ulong uiDO =0x33;
hDev = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
HSDAQNet.DATALOG.HS_StartLogger (hDev,NULL,2,0);
// The data log file (binary type) will be saved to the current folder where the application is running , 2 minutes of interval for changing to next file
HSDAQNet.Sys.HS_Device_Release(hDev);
```

Remark

Use HS_StartLogger function to start the data logging. The data will save to the specified folder on storage disk of the host PC. If file type sets to 0 (binary file), the data will be saved the binary file (raw data) to the specified folder. If set to 1 (text file), the data will be calibrated and then saved to the text file (.txt file) to the specified folder.

2.5.2. HS_StopLogger

This function is used to stop the data logging.

Syntax

C++

```
bool HS_StopLogger (
   HANDLE hobj,
);
```

.Net

```
bool HS_ StopLogger (
  IntPtr hobj,
);
```

Parameter

hobj

[in] A handle to the ET-7H16 creaded by HS_Device_Create.

Return Value

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hHS;
hHS = HS_Device_Create("192.168.1.1");
```

```
HS_StartLogger(hHS,NULL,2,0);
// The data log file (binary type) will be saved to the current folder where the application is running , 2 minutes of interval for changing to next file
…
HS_StopLogger(hHS);
HS_Device_Release (hHS);
```

[C#]

```
IntPtr hDev;
ulong uiDO =0x33;
hDev = HSDAQNet.Sys.HS_Device_Create("192.168.1.1");
HSDAQNet.DATALOG.HS_StartLogger (hDev,NULL,2,0);
// The data log file (binary type) will be saved to the current folder where the application is running , 2 minutes of interval for changing to next file
HSDAQNet.DATALOG.HS_StopLogger (hDev);
HSDAQNet.Sys.HS_Device_Release(hDev);
```

Remark

None.

2.5.3. HS_GetAllLogFiles

Search all log files in the specified folder with the specified file type and return the total number of files.

Syntax

C++

```
int HS_GetAllLogFiles (
    TCHAR *folderpath,
    int filetype
);
```

.Net

```
int HS_GetAllLogFiles (
    string folderpath,
    int filetype
);
```

Parameter

folderpath

[in] Specify the folder path where the data log files are saved. If NULL, the current folder where the application is running will be the specified for searching.

filetype

[in] File type

0: binary file (.bin)

1: text file (.txt)

Return Value

If the function succeeds, the return value is total number of the files.

If the function fails, the return value is 0.

Examples

[C]

```
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
```

[C#]

```
Int ind=HSDAQNet.DATALOG.HS_GetAllLogFiles (NULL,0);
//Search log files of binary type in the current folder where the application is running
```

Remark

None.

2.5.4. HS_LogFile_Open_byIndex

Open a data log file and corresponding configuration settings by the index number searched by HS_GetAllLogFiles() function.

Syntax

C++

```
HANDLE HS_LogFile_Open_byIndex (
    int index,
    TCHAR *getfullFilename,
);
```

.Net

```
IntPtr HS_LogFile_Open_byIndex (
    int index,
    string getfullFilename
);
```

Parameter

index

[in] The index number searched by HS_GetAllLogFiles() function.

getfullFilename

[in] Retrieve the full path and name of the log file found by the search index number.

Return Values

If the function succeeds, the return value is an open handle to the specified log file.

If the function fails, the return value is NULL.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     …//user-define code
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path);
     …//user-define code
     HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
   }
}
```

Remark

Before calling this function, It must call HS_GetAllLogFiles() function to search all log files in the specified folder.

2.5.5. HS_LogFile_Open / HS_LogFile_OpenW

Open a data log file and corresponding configuration settings by the specified path and file name.

Syntax

C++

```
HANDLE HS_LogFile_Open (
   char *fullFilename
);

HANDLE HS_LogFile_OpenW (
   TCHAR *fullFilename,
);
```

.Net

```
IntPtr HS_LogFile_Open (
  string fullFilename
);
```

Parameter

fullFilename

[in] Specify the full path and name of the log file.

Return Values

If the function succeeds, the return value is an open handle to the specified log file.

If the function fails, the return value is NULL.

Examples

[C]

```
HANDLE hlf;
hlf=HS_LogFile_Open("D:\\Datalogger\\10_1_107_125\\2018_12_14_13_37\\10_1
_107_125_9d06_2018_12_14_13_37_59.bin")
```

[C#]

```
IntPtr
  hLf=HSDAQNet.DATALOG.HS_LogFile_Open("D:\\Datalogger\\10_1_107_125\\2
  018_12_14_13_37\\10_1_107_125_9d06_2018_12_14_13_37_59.bin");
```

Remark

None.

2.5.6. HS_LogFile_Close

Close a valid handle of a log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

Syntax

C++

```
bool HS_LogFile_Close (
   HANDLE hobj
);
```

.Net

```
bool HS_LogFile_Close (
   IntPtr hobj
);
```

Parameter

hobj

[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     …//user-define code
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path);
     …//user-define code
     HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
   }
}
```

Remark

None.

2.5.7. HS_GetLogFileInfo

Get a data log file information including the file version and file size.

Syntax

C++

```
bool HS_GetLogFileInfo (
   HANDLE hobj,
   DWORD *filesize,
   Int *filetype,
   Int *fileversion
);
```

.Net

```
bool HS_GetLogFileInfo (
   IntPtr hobj,
   ref UInt32 filesize,
   ref int filetype,
   ref int fileversion
);
```

Parameter

hobj
[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

filesize
[out] The size in bytes of the specified log file

filetype
[out] The file type of the specified log file

fileversion
[out] The version of the specified log file

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     DWORD filesize;
     int filetype;
     int fileversion;
     HS_GetLogFileInfo(hlf,&filesize,&filetype,&fileversion);
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
     Uint32 filesize;
```

```
        int filetype;
        int fileversion;
        HSDAQNet.DATALOG.HS_GetLogFileInfo(hlf,ref filesize,ref filetype,ref
    fileversion);
        HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
      }
    }
```

Remark

None.

2.5.8. HS_GetLogFile_AIScanConfigInfo

Get the AI scan information of the data log file regarding of the sampling rate, scan channels, pacer gain, trigger mode.

Syntax

C++

```
bool HS_GetLogFile_AIScanConfigInfo (
   HANDLE hobj,
   short * pacerChCnt,
   short * pacerGain,
   short * triggerMode,
   long * sampleRate,
   short * DataTransMethod,
);
```

.Net

```
bool HS_GetLogFile_AIScanConfigInfo (
     IntPtr hobj,
     ref short pacerChCnt,
     ref short pacerGain,
     ref short triggerMode,
     ref long sampleRate,
     ref short DataTransMethod
   );
```

Parameter

hobj

[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

pacerchCnt

[out] Get the AI scan channels that logged in the specified log file.

The AI scan channel range for PET-7H16M is 1~8.

pacerGain

[out] Get the AI input type that logged in the specified log file
The gain range for PET-7H16M is

0: +/- 5V
1: +/- 10V

triggermode

[out] Get the AI scan trigger mode that logged in the specified log file
0: disable external trigger (software AD) ,
1: external clock trigger
2: external post-trigger,
3: external pre-trigger

sampleRate

[out] Get the AI scan sampling rate that logged in the specified log file
The range of the AI scan sampling rate for PET-7H16M is 1~200KHz.

DataTransMethod

[out] Get the data transmission method that logged in the specified log file
0: TCP socket

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
short gchCnt=0;
short guseGain=0;
short gextriggMode=0;
long gsampleRate=0;
short gDataTransMethod;

int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
```

```
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     HS_GetLogFile_AIScanConfigInfo(hlf,&gchCnt,&guseGain,&gextriggMode,&gs
ampleRate,&gDataTransMethod
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
short gchCnt=0;
short guseGain=0;
short gextriggMode=0;
long gsampleRate=0;
short gDataTransMethod;

String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);

//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
     HSDAQNet.DATALOG.HS_GetLogFile_AIScanConfigInfo(hlf,ref gchCnt,ref
guseGain,ref gextriggMode,ref gsampleRate,ref gDataTransMethod
     HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
   }
}
```

Remark

None.

2.5.9. HS_GetLogFile_GainOffset

Get the information of the data log file regarding of the gain/offset values of the specified AI channel.

Syntax

C++

```
bool HS_GetLogFile_GainOffset(
    HANDLE hobj,
    int ch,
    unsigned short *gainVal,
    short *offsetVal,
);
```

.Net

```
bool HS_GetLogFile_GainOffset (
      IntPtr hobj,
      int ch,
      ref short gainVal,
      ref short offsetVal
    );
```

Parameter

hobj

[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

ch

[in] Specifies the AI channel.

The AI channel range for PET-7H16M is 0~7.

gainVal

[out] Get the gain value of the specified AI channel for calibrating the analog data that logged in the specified log file

offsetVal

[out] Get the offset value of the specified AI channel for calibrating the analog data that logged in the specified log file

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     unsigned short gainVal=0;
     short offsetVal=0;
     for(int j=0;j<gchCnt;j++)
     {
       HS_GetLogFile_GainOffset(hlf,j, &gainVal, &offsetVal);
       …
     }
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
```

```
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     ushort gainVal=0;
     short offsetVal=0;

     hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
     for(int j=0;j<gchCnt;j++)
     {
       HSDAQNet.DATALOG.HS_GetLogFile_GainOffset(hlf,j, ref gainVal, ref
offsetVal);
       …
     }

     HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
   }
}
```

Remark

None.

2.5.10. HS_GetLogFile_AIScanSampleInfo

Get the totoal sampling counts and the starting time of first triggered sampling data in the data log file.

Syntax

C++

```
bool HS_GetLogFile_AIScanSampleInfo(
   HANDLE hobj,
   DWORD *sampleCount,
   char *StartDate,
   char *StartTime,
);
```

.Net

```
bool HS_GetLogFile_AIScanSampleInfo(
     IntPtr hobj,
     ref UInt32 sampleCount,
     ref byte[] StartDate,
     ref byte[] StartTime
     );
```

Parameter

hobj

[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

sampleCount

[out] Get the total sampling count in the specified log file.

StartDate

[out] Get the date of first triggered sampling data in the specified log file.

StartTime

[out] Get the time of first triggered sampling data in the specified log file.

Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE. To get extended error information, call HS_GetLastError.

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     DWORD samplecount;
     char startdate[32],starttime[32];
     HS_GetLogFile_AIScanSampleInfo(hlf,&samplecount,startdate,starttime);
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);

//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     uint samplecount=0;
     string startdate ="";
```

```
    string starttime ="";

    hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
    HSDAQNet.DATALOG.HS_GetLogFile_AIScanSampleInfo(hlf,ref
samplecount,ref startdate,ref starttime);
    HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
   }
}
```

Remark

None.

2.5.11. HS_GetLogFile_AIData

Reads AI input data from the text file.

Syntax

C++

```
DWORD HS_GetLogFile_AIData(
   HANDLE hobj,
   int    StartIndex,
   DWORD count,
   float *fAIData,
);
```

.Net

```
UInt32 HS_GetLogFile_AIData(
     IntPtr hobj,
     int StartIndx,
     UInt32 count,
       float fAIData
     );
```

Parameter

hobj
[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

StartIndex
[out] The first sampling data to read from the log file.

count
[out] The number of sampling data to read from the log file.

fAIData

[out] The array contains the AI values (float value) that read back from the log file.

Return Values

The return value is the total number of sampling data actually read from the log file

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,1);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     DWORD samplecount;
     char startdate[32],starttime[32];
     HS_GetLogFile_AIScanSampleInfo(hlf,&samplecount,startdate,starttime);
     float *fdatabuff=( float *)malloc(sizeof(float)*samplecount);
     HS_GetLogFile_AIData(hlf, 0, samplecount, fdatabuff);
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     uint samplecount=0,targetCnt=1000;
```

```
    string startdate ="";
    string starttime ="";
    hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
    Uint32 samplecount;
    float[] fdatabuff = new float[targetCnt];
    HSDAQNet.DATALOG.HS_GetLogFile_AIScanSampleInfo(hlf,ref
samplecount,ref startdate,ref starttime);

    HSDAQNet.DATALOG.HS_GetLogFile_AIData(hlf, 0, targetCnt, fdatabuff);
    HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
  }
}
```

Remark

None.

2.5.12. HS_GetLogFile_AIDataHex

Read AI input data(Hex) from the binary file.

Syntax

C++

```
DWORD HS_GetLogFile_AIDataHex(
   HANDLE hobj,
   int    StartIndex,
   DWORD count,
   WORD *AIData,
);
```

.Net

```
UInt32 HS_GetLogFile_AIDataHex(
     IntPtr hobj,
     int StartIndx,
     UInt32 count,
       UInt32 AIData
     );
```

Parameter

Hobj

[in] A handle to the specified log file opened by HS_LogFile_Open or HS_LogFile_Open_byIndex

StartIndex

[out] The first sampling data to read from the log file.

count

[out] The number of sampling data to read from the log file.

AIData

[out] The array contains the AI values (Hexadecimal value) that read back from the log file.

Return Values

The return value is the total number of sampling data actually read from the log file

Examples

[C]

```
HANDLE hlf;
TCHAR tcgetfulfilelPath[MAX_PATH]={0};
int ind=HS_GetAllLogFiles(NULL,1);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     hlf=HS_LogFile_Open_byIndex(i,tcgetfulfilelPath);
     DWORD samplecount;
     char startdate[32],starttime[32];
     HS_GetLogFile_AIScanSampleInfo(hlf,&samplecount,startdate,starttime);
     WORD *databuff=(WORD *)malloc(sizeof(WORD)*samplecount);
     HS_GetLogFile_AIDataHEX(hlf, 0, samplecount, databuff);
     HS_LogFile_Close(hlf);
   }
}
```

[C#]

```
IntPtr hlf;
String path = Directory.GetCurrentDirectory();
int ind= HSDAQNet.DATALOG.HS_GetAllLogFiles (path,0);
//Search log files of binary type in the current folder where the application is running
if(ind>0)
{
   for(int i=0;i<ind;i++)
   {
     uint samplecount=0,targetCnt=1000;
```

```
    string startdate ="";
    string starttime ="";
    hlf= HSDAQNet.DATALOG.HS_LogFile_Open_byIndex(i, path)
    Uint32 samplecount;
    UInt32[] hdatabuff = new Uint32[targetCnt];
    HSDAQNet.DATALOG.HS_GetLogFile_AIScanSampleInfo(hlf,ref
samplecount,ref startdate,ref starttime);

    HSDAQNet.DATALOG.HS_GetLogFile_AIData(hlf, 0, targetCnt, hdatabuff);
    HSDAQNet.DATALOG.HS_LogFile_Close(hlf);
  }
}
```

Remark

None.

2.6. Error Handling API

The error handling functions enable you to receive and display error information for your application.

Error Handling Functions

The following functions are used with error handling.

HSDAQ Functions	HSDAQNET Functions	Description
HS_GetLastError	ErrHandling.GetLastError	Retrieves the last-error code value.
HS_SetLastError	ErrHandling.SetLastError	sets the last-error code.
HS_GetLastErrorMessage	ErrHandling.GetErrorMessage	retrieves a message string.
HS_ClearLastError	ErrHandling.ClearLastError	clears the last-error code.

2.6.1. HS_GetLastError

Retrieves the last-error code value.

Syntax

C++

```
DWORD HS_GetLastError();
```

.Net

```
uint GetLastError ();
```

Parameters

This function has no parameters.

Return Value

The Return Value section of each function page notes the conditions under which the function sets the last-error code.

Examples

This function has no examples.

Remarks

You should call the HS_GetLastError function immediately when a function's return value indicates that such a call will return useful data. That is because some functions call HS_SetLastError(0) when they succeed, wiping out the error code set by the most recently failed function.

For an example, please refer to HS_GetErrorMerrage in this chapter.

To obtain an error string for the error codes, use the HS_GetErrorMessage function. For a complete list of error codes, see Appendix A. System Error Code.

The following table lists the system error codes ranges for each function reference.

Error Type	Explanation	Range
HS_ERR_SUCCESS	No error, success	0x00000
HS_ERR_UNKNOWN	A error which is undefined	0x00001
System	System API Error and WSAGetLastError	0x10000 ~
Memory	About memory access	0x14000 ~
DATA log	About Data logger	0x14100 ~
Watchdog	About watchdog	0x15000 ~
Device	About Device open and clos	0x17000 ~
IO	About IO modules	0x18000 ~
Users	For user	0x20000 ~

2.6.2. HS_SetLastError

This function sets the last-error code.

Syntax

C++

```
void HS_SetLastError(
  DWORD errno
);
```

.Net

```
void SetLastError (
  uint errno
);
```

Parameters

errno

[in] Specifies the last-error code.

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

Applications can optionally retrieve the value set by this function by using the HS_GetLastError function.

The error codes are defined as DWORD values. If you are defining an error code, ensure that your error code does not conflict with any defined error codes.

We recommend that your error code should be greater than 0x20000.

For more information about the definition of error codes, please refer to HS_GetLastError in this document.

2.6.3. HS_GetErrorMessage

This function retrieves a message string.

Syntax

C++

```
void HS_GetErrorMessage(
    DWORD dwMessageID,
    LPTSTR lpBuffer
);
```

.Net

```
void GetErrorMessage (
    uint dwMessageID,
    string lpBuffer
    );
```

Parameters

dwMessageID

[in] Specifies the 32-bit message identifier for the requested message.

lpBuffer

[out] A pointer to a buffer that receives the error message.

Return Value

This function has does not return a value.

Examples

[C]

```
HANDLE hHS;
float fAI=0;
BOOL err;
char strErr[32];

hHS = HS_Device_Create(“192.168.1.1”);
err = HS_ReadAI(hHS,0,&fAI);
if(err == FALSE)
{
   HS_GetErrorMessage(HS_GetLastError(), strErr);
   printf("Read SRAM failure!. The error code is %x\n", HS_GetLastError());
}
…
```

[C#]

```
bool err;
IntPtr hDev;
float fAI=0;

hDev = HSDAQNet.Sys.HS_Device_Create(“192.168.1.1”);
err=HSDAQNet.IO.ReadAI(hDev,0,ref fAI);
if (err == false)
{
   Console.WriteLine(HSDAQNet.ErrHandling.GetErrorMessage(HSDAQNet.ErrHa
ndling.GetLastError()) + ". The error code is " +
   HSDAQNet.ErrHandling.GetLastError().ToString() + "\n");
}
…
```

Remarks

The HS_GetErrorMessage function can be used to obtain error message strings for the error codes returned by HS_GetLastError, as shown in the following example.

```
TCHAR Buffer[32];
HS_GetErrorMessage(HS_GetLastError(), Buffer);
MessageBox( NULL, Buffer, L"Error", MB_OK |
MB_ICONINFORMATION );
```

2.6.4. HS_ClearLastError

This function clears the last-error code.

Syntax

C++

```
void HS_ClearLastError();
```

.Net

```
void ClearLastError ();
```

Parameters

This function has no parameters.

Return Value

This function has does not return a value.

Examples

This function has no examples.

Remarks

The HS_ClearLastError function clears the last error, that is, the application is treated as success.

Appendix.

A. System Error Codes

This following table provides a list of system error code that is intended to be used by programmers so that the software they write can better deal with errors.
The error codes/error messages are returned by the HS_GetLastError/HS_GetErrorMessage function when one of the ET-7H16 services functions fail.

Error Code	value	Error Message
HS_ERR_UNKNOWN	0x00001	Unknow Error
HS_ERR_UNKNOWN_MODULE	0x13003	Unknown Module
HS_ERR_FUNCTION_NOT_SUPPORT	0x13006	Function not supported
HS_ERR_MODULE_UNEXISTS	0x13007	Module doesn't exist
HS_ERR_FUNCTION_REPEAT_CALLED	0x13009	Function repeat call error
HS_ERR_INVALID_HANDLE_VALUE	0x13010	Invalid handle
HS_ERR_INVALID_PARAMETER	0x13012	Invalid parameter
HS_ERR_MEMORY_ALLOCATED	0x13014	Memory allocation error
HS_ERR_MEMORY_INVALID_SIZE	0x14008	Invalid memory size
HS_ERR_DATALOG_CONFIGFILE_NOFOUND	0x15002	Invalid configuration file or the configuration not found

Error Code	value	Error Message
HS_ERR_DEVICE_READ_TIMEOUT	0x17002	Read data from the device timeout
HS_ERR_DEVICE_RESPONSE	0x17003	Response from the device error
HS_ERR_DEVICE_UNDER_INPUT_RANGE	0x17004	Under input range
HS_ERR_DEVICE_EXCEED_INPUT_RANGE	0x17005	Exceed input range
HS_ERR_DEVICE_OPEN_FAILED	0x17006	Open the device fail
HS_ERR_DEVICE_INVALID_VALUE	0x17008	The input value is invalid
HS_ERR_DEVICE_SEND	0x17010	Send data to the device error
HS_ERR_DEVICE_DATA_CONNECT	0x17011	Create a connection fail
HS_ERR_IO_NOT_SUPPORT	0x18001	The device does not support this API function
HS_ERR_IO_ID	0x18002	The device does not support this API function
HS_ERR_IO_SLOT	0x18003	Slot's value exceeds its range
HS_ERR_IO_CHANNEL	0x18004	Channel's value exceeds its range
HS_ERR_IO_GAIN	0x18005	Gain's value exceeds its range
HS_ERR_IO_VALUE_OUT_OF_RANGE	0x18007	I/O value is out of the range
HS_ERR_IO_CHANNEL_OUT_OF_RANGE	0x18008	I/O channel is out of the range
HS_ERR_IO_DO_CANNOT_OVERWRITE	0x18010	DO channel can't overwrite
HS_ERR_IO_AO_CANNOT_OVERWRITE	0x18011	AO channel can't overwrite
HS_ERR_IO_OPERATION_MODE	0x18012	Invalid I/O operation mode

B. Configuration type code

This following table provides a list of configuration type code and the corresponding parameter valuses..
The HS_GetConfig /HS_SetConfig functions are use to set the parameter value of configuration type for ET-7H16.

Configuration type	Parameter	Description
BOARD_CONFIG	N/A	Set the device configuration
IO_CONFIG	N/A	Set I/O configuration
HSDAQ_CONFIG	HSDAQ_CONNECT_TIMEOUT	Set TCP connection timeout
DATALOG_CONFIG	LOGFOLDERTYPE	Set folder type for data logger
	LOGFILEMAXSIZE	Set maximum size of log file