DC Power Supply IVI-C Programming Guide

Models Supported

The series of Programmable DC Power Supply this IVI-C driver is shown below:

Series	Release Version Supporting IVI-C Driver
UDP2000S	1.02 and higher

Software Requirement

If you want to use the IVI Driver, you must install NI-VISA,the IVI Compliance Package, and a C language development system that supports the IVI driver library.They are available for free download at https://www.ni.com/en/support/downloads/drivers/download.ni-visa.html and https://www.ni.com/ivi/ivi_prod.htm.

Accessing the Driver

To acceTo access the driver, all IVI-C driver programs (and your Visual Studio project) must:

- Reference the proper import library: UDP2000S.lib
- Include the proper header file: UDP2000S.h
- Use the function calls that begin with UDP2000S_

To use the UDP2000S driver, perform the following steps in Visual Studio.

1. In Solution Explorer, right-click on the project and select Properties.
2. In the property pages dialog, expand the Linker node and select Input.
3. In the Additional Dependencies field, enter:
    - UDP2000S.lib(Win32) or UDP2000S_64.lib(x64)
    - visa32.lib(Win32) or visa64.lib(x64)
4. Specify the path to your driver's library (.lib) and header (.h) files:
    1. In the property pages dialog,select VC++ Directories.
    2. Enter the following paths. For:
    - Include files, add
        - C:\Program Files\IVI Foundation\IVI\Include (Win32 & x64)
        - C:\Program Files\IVI Foundation\VISA\Win64\include (Win32 & x64)
    - Library files, add:
        - C:\Program Files\IVI Foundation\IVI\Lib_\msc (Win32)
        - C:\Program Files\IVI Foundation\IVI\Lib_x64\msc (x64)
        - C:\Program Files\IVI Foundation\VISA\Win64\Lib_x64\msc (Win32 & x64)
          Note that these paths:

Start with C:\Program Files (x86) for 32-bit operating systems
Also can be entered in the Project Properties dialog in the Configuration Properties, C++ and Linker panes
In the main application source file (.cpp), add the following line to the top of the file:
  #include "UDP2000S.h"

IVI-C Driver Package List

The IVI-C driver package provides threekinds of files: UDP2000S file, UDP2000S.h file and UDP2000S_64 file.

File	Description
UDP2000S.dll/UDP2000S_64.dll	A dynamic link library file,including variables, functions and data interfaces for various attributes.
UDP2000S.lib/UDP2000S_64.lib	An import library file, including the symbolic name and optional identification number of each exported function in the dynamic file.
UDP2000S.h	A header file,including declarations of variables, functions, and data interfaces.

> You include the UDP2000S.h when programming the DC Power Supply with the IVI driver, and load the UDP2000S.dll/UDP2000S_64.dll dynamic file and UDP2000S.lib/UDP2000S_64.lib import library file into your own project.
> You will findan example that show you how to use these files at the end of this document.

Introduction to IVI

IVI (Interchangeable Virtual Instruments) is a new generation of instrument driver technology specifications introduced by the IVI Foundation. IVI can realize the interchangeability with the instrument, the instrument simulation, and the instrument state tracking and buffer function. All references to IVI drivers in this document refer to IVI-C drivers that are created using NI tools and that rely on the IVI Engine.

Attribute

This chapter describes the attributes of the IVI driver.

Basic

Attributes that control the basic features of the DC Power Supply.The basic group has the following attributes:

- UDP2000S_ATTR_CHANNEL_COUNT
- UDP2000S_ATTR_VOLTAGE_LEVEL
- UDP2000S_ATTR_OVP_ENABLED
- UDP2000S_ATTR_OVP_LIMIT
- UDP2000S_ATTR_CURRENT_LIMIT_BEHAVIOR
- UDP2000S_ATTR_CURRENT_LIMIT
- UDP2000S_ATTR_OUTPUT_ENABLED

UDP2000S_ATTR_CHANNEL_COUNT

Description: Returns the number of currently available outputs
Data Type: ViBoolean
Access: R
Get Function: ViStatus UDP2000S_GetAttributeViBoolean (ViSession vi,ViConstString channelName,ViAttr attributeId,ViBoolean *value);
Set Function: N/A

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_CHANNEL_COUNT
- *value*: VI_FALSE | VI_TRUE

UDP2000S_ATTR_VOLTAGE_LEVEL

Description: Get or set output voltage
Data Type: ViReal64
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 *value)
Set Function: ViStatus UDP2000S_SetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 value)

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_VOLTAGE_LEVEL
- *value*: The settable range is 0 to rated voltage

UDP2000S_ATTR_OVP_ENABLED

Description: Get or set the overvoltage protection enable value
Data Type: ViBoolean
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViBoolean (ViSession vi,ViConstString channelName,ViAttr attributeId,ViBoolean *value);
Set Function: ViStatus UDP2000S_SetAttributeViBoolean (ViSession vi,ViConstString channelName,ViAttr attributeId,ViBoolean value);

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_OVP_ENABLED
- *value*: VI_FALSE | VI_TRUE

UDP2000S_ATTR_OVP_LIMIT

Description: Get or set overvoltage protection value.
Data Type: ViReal64
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 *value)
Set Function: ViStatus UDP2000S_SetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 value)

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_OVP_LIMIT
- *value*: The settable range is 0 to rated voltage

UDP2000S_ATTR_CURRENT_LIMIT_BEHAVIOR

Description: Get or set the current behavior mode during overcurrent
Data Type: ViInt32
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViInt32 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViInt32 *value)
Set Function: ViStatus UDP2000S_SetAttributeViInt32 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViInt32 value)

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_CURRENT_LIMIT_BEHAVIOR
- *value*: UDP2000S_VAL_CURRENT_REGULATE | UDP2000S_VAL_CURRENT_TRIP

UDP2000S_ATTR_CURRENT_LIMIT

Description: Get or set overcurrent protection value.
Data Type: ViReal64
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 *value)
Set Function: ViStatus UDP2000S_SetAttributeViReal64 (ViSession vi,ViConstString channelName,ViAttr attributeId,ViReal64 value)

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_CURRENT_LIMIT
- *value*: The settable range is 0 to rated current

UDP2000S_ATTR_OUTPUT_ENABLED

Description: Get or set the output status of channel
Data Type: ViBoolean
Access: R/W
Get Function: ViStatus UDP2000S_GetAttributeViBoolean (ViSession vi,ViConstString channelName,ViAttr attributeId,ViBoolean *value);
Set Function: ViStatus UDP2000S_SetAttributeViBoolean (ViSession vi,ViConstString channelName,ViAttr attributeId,ViBoolean value);

- *vi*: Instrument handle
- *channelName*: "OutputChannel1"~"OutputChannel2"
- *attributeId*: UDP2000S_ATTR_OUTPUT_ENABLED
- *value*: VI_FALSE | VI_TRUE

Function

Basic

ViStatus UDP2000S_InitWithOptions (ViRsrc resourceName,ViBoolean idQuery,ViBoolean resetDevice,ViConstString optionString, ViSession *vi)

This function creates a new IVI session.

Parameter	Description
resourceName	Specifies the resource name of the instrument
idQuery	To perform ID query or not
resetDevice	To reset the device or not
optionString	This parameter is the option string setsto the InitWithOptions,function of the instrument driver.It includes settings for Simulate,RangeCheck, QueryInstrStatus and Cache
vi	Instrument handle

Example:

```
UDP2000S_InitWithOptions ("USB0::0x0483::0x5740::SN12345678901::INSTR ",
VI_TRUE,
VI_FALSE,"Simulate=false,RangeCheck=true,QueryInstrStatus=false,Cache=false",
&vi);
```

ViStatus UDP2000S_close (ViSession vi)

This function closes the instrument.

Parameter	Description
vi	Instrument handle

Example:

```
UDP2000S_close (vi);
```

ViStatus UDP2000S_ConfigureCurrentLimit (ViSession vi,ViConstString channelName,ViInt32 behavior,ViReal64 limit)

This function configures the current limit. It specifies the output current limit value and the behavior of the
power supply when the output current is greater than or equal to that value.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. "OutputChannel1"~"OutputChannel2"
behavior	Specifies the behavior of the power supply when the output current is greater than or equal to the value of the limit parameter. (UDP2000S_VAL_CURRENT_REGULATE|UDP2000S_VAL_CURRENT_TRIP)
limit	Specifies the power supply's output current limit

Example:

```
UDP2000S_ConfigureCurrentLimit (vi,  "OutputChannel1",
UDP2000S_VAL_CURRENT_REGULATE , 1.0);
```

ViStatus UDP2000S_ConfigureOutputEnabled (ViSession vi,ViConstString channelName,ViBoolean enabled)

Configures whether the signal that the power supply produces appears at the output connector..

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
enabled	Specifies the power supply’s output current limit.

Example:

```
UDP2000S_ConfigureOutputEnabled (vi, "OutputChannel1" ,VI_TRUE);
```

ViStatus UDP2000S_ConfigureOVP (ViSession vi,ViConstString channelName,ViBoolean enabled,ViReal64 limit)

Configures the over-voltage protection.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
enabled	Specifies the behavior of the power supply when the ouptut voltage is greater than or equal to the value of the Limit parameter.
limit	Specifies the power supply’s over-voltage protection limit

Example:

```
UDP2000S_ConfigureOVP (vi, "OutputChannel1" ,VI_TRUE,1.0);
```

ViStatus UDP2000S_ConfigureVoltageLevel (ViSession vi,ViConstString channelName,ViReal64 level)

Configures the voltage level the power supply attempts to generate.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
level	Specifies the voltage level the power supply attempts to generate

Example:

```
UDP2000S_ConfigureVoltageLevel (vi, "OutputChannel1" ,2.0);
```

ViStatus UDP2000S_GetChannelName (ViSession vi,ViInt32 index,ViInt32 bufferSize,ViChar name[])

This function returns the specific driver defined output name that corresponds to the one-based index that
the user specifies.

Parameter	Description
vi	Instrument handle
index	A one-based index that defines which name to return. (1~6)
bufferSize	The number of bytes in the ViChar array that the user specifies for the Name parameter
name	A user-allocated buffer into which the driver stores the output name

Example:

```
UDP2000S_GetChannelName (vi, 1, 256, buff);
```

ViStatus UDP2000S_QueryMaxCurrentLimit (ViSession vi,ViConstString channelName,ViReal64 voltageLevel,ViReal64* maxCurrentLimit)

This function returns the maximum programmable current limit that the power supply accepts for a
particular voltage level on an output.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
voltageLevel	The voltage level for which to determine the maximum programmable current limit
maxCurrentLimit	Returns the maximum programmable current limit for the specified voltage level

Example:

```
UDP2000S_QueryMaxCurrentLimit (vi, "OutputChannel1", 1.0, buff);
```

ViStatus UDP2000S_QueryMaxVoltageLevel (ViSession vi,ViConstString channelName,ViReal64 currentLimit,ViReal64* maxVoltageLevel)

This function returns the maximum programmable voltage level that the power supply accepts for a
particular current limit on an output.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
currentLimit	The current limit for which to determine the maximum programmable voltage level
maxVoltageLevel	Returns the maximum programmable voltage level for the specified current limit

Example:

```
UDP2000S_QueryMaxVoltageLevel (vi, "OutputChannel1", 1.0, buff);
```

ViStatus UDP2000S_QueryOutputState(ViSession vi,ViConstString channelName,ViInt32 outputState,ViBoolean* inState)

This function returns whether the power supply is in a particular output state.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
outputState	The output state for which to query. (UDP2000S_VAL_OUTPUT_CONSTANT_VOLTAGE|UDP2000S_VAL_OUTPUT_CONSTANT_CURRENT)
inState	Returns True if the power supply is currently in the state specified

Example:

```
UDP2000S_QueryOutputState (vi, "OutputChannel1",
UDP2000S_VAL_OUTPUT_CONSTANT_VOLTAGE, buff);
```

Measure

ViStatus UDP2000S_Measure (ViSession vi,ViConstString channelName,ViInt32 measurementType,ViReal64* measurement)

Takes a measurement on the output signal and returns the measured value.

Parameter	Description
vi	Instrument handle
channelName	The name of the output. ("OutputChannel1"~"OutputChannel2")
measurementType	The type of measurement to take. (UDP2000S_VAL_MEASURE_CURRENT|UDP2000S_VAL_MEASURE_VOLTAGE)
measurement	The measured value

Example:

```
UDP2000S_QueryOutputState (vi, "OutputChannel1", UDP2000S_VAL_MEASURE_CURRENT,
buff);
```

Programming Example

```
// CMakeLists.txt

cmake_minimum_required(VERSION 3.26)

project(example LANGUAGES CXX)

set(IVI_DIR "C:/Program Files/IVI Foundation/IVI")
set(VISA_DIR "C:/Program Files/IVI Foundation/VISA/Win64")

add_executable(example example.cpp)

target_include_directories(example PRIVATE
    "${VISA_DIR}/Include"
    "${IVI_DIR}/Include"
)
target_link_directories(example PRIVATE
    "${IVI_DIR}/Lib_x64/msc"
)
target_link_libraries(example PRIVATE UDP2000S_64)
```

```
// main.cpp

#include "UDP2000S.h"
#include <iostream>


const char* ResourceName = "USB0::0x0483::0x5740::SN12345678901::INSTR";
const char* Options =
"QueryInstrStatus=false,RangeCheck=true,Cache=false,Simulate=false";
ViSession Vi;

void checkStatus(ViStatus status, const std::string& tip);

int main()
{
    ViStatus status = UDP2000S_InitWithOptions((ViRsrc)ResourceName, VI_TRUE,
VI_FALSE, (ViConstString)Options, &Vi);
    checkStatus(status, "UDP2000S_InitWithOptions()");

    ViChar ch[256] {};
    status = UDP2000S_GetChannelName(Vi, 1, 256, ch);
    checkStatus(status, "UDP2000S_GetChannelName()");

    status = UDP2000S_SetAttributeViReal64(Vi, ch, UDP2000S_ATTR_VOLTAGE_LEVEL,
1.23);
    checkStatus(status, "UDP2000S_SetAttributeViReal64()");

    ViReal64 voltage_level;
    status = UDP2000S_GetAttributeViReal64(Vi, ch, UDP2000S_ATTR_VOLTAGE_LEVEL,
&voltage_level);
```

```
    checkStatus(status, "UDP2000S_GetAttributeViReal64()");

    status = UDP2000S_ConfigureOutputEnabled(Vi, ch, VI_TRUE);
    checkStatus(status, "UDP2000S_ConfigureOutputEnabled()");

    status = UDP2000S_close(Vi);
    checkStatus(status, "UDP2000S_close()");

    return 0;
}

void checkStatus(ViStatus status, const std::string& tip)
{
    if (status != 0) {
        ViStatus errorCode;
        ViChar errorDesc[256];
        status = UDP2000S_GetError(Vi, &errorCode, 256, errorDesc);
        std::cout << tip << " failure. " << " Code: " << errorCode << "
Description: " << errorDesc << std::endl;
    } else {
        std::cout << tip << " success." << std::endl;
    }
}
```