NEON-2000-JNX Series

DI/O Function Library Reference



Manual Rev.: 1.1

Revision Date: July 15, 2021

Part No: 50M-00006-1010

LEADING EDGE COMPUTING

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Revision History

Revision	Release Date	Description of Change(s)
1.0	2021-04-07	Initial release
1.1	2021-07-15	Remove Neon_GetDOState function; update Neon SDK installation info.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Preface

Copyright © 2021 ADLINK Technology, Inc.

This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Environmental Responsibility

ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

Trademarks

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.



Additional information, aids, and tips that help users perform tasks.



Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.



Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Table of Contents

Revision History ..... **ii**

Preface ..... **iii**

1 Introduction **1**

1.1 Setting Up the Neon SDK 1

2 Function Library **3**

2.1 List of Functions 3

2.2 Function Library 4

2.2.1 Device Control Function 4

2.2.2 DIO Functions 5

Important Safety Instructions **19**

Getting Service **23**

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

This page intentionally left blank.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1 Introduction

The NEON-2000-JNX Function Library Reference provides necessary API details for functions that can used to develop applications in C/C++. When the NEON-2000-JNX driver is installed, functions can be directly called with the Neon SDK (the driver is pre-installed at the factory).

Example:

```
#include "Neon.h"
using namespace std;
char buffer[256];
if (Neon_GetDeviceInfo(buffer, sizeof
  (buffer)) != COMMON_ERROR) cout << buffer
  << endl;
...
```

For more examples, refer to the sample code.

1.1 Setting Up the Neon SDK

If users need to update or re-install the Neon SDK, please go to the NEON-2000-JNX product page at www.adlinktech.com and download the driver package.

Use the following steps to set up the Neon SDK.

1. Install the Neon SDK:

sudo dpkg -i NeonSDK.deb

2. Set the file paths:

- libraries at **/usr/lib/Neon**
- headers at **/usr/include/Neon/Neon.h**
- samples at **/usr/src/Neon/Sample**

3. Build Neon_Information Sample:

#cd /usr/src/Neon/Sample/Neon_Information

#make

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4. Build Neon_Setting Sample:

#cd /usr/src/Neon/Sample/Neon_Setting

#make

5. Build Neon_ListenTrigger Sample:

#cd /usr/src/Neon/Sample/Neon_ListenTrigger

#make

6. Run Neon_Information Sample:

#cd /usr/src/Neon/Sample/Neon_Information

#sudo ./NeonInformation

7. Neon_Setting Help:

Execute the application with no parameters.

#cd /usr/src/Neon/Sample/Neon_Setting

#sudo ./NeonSet

8. Neon_ListenTrigger Sample:

The program will receive a trigger event via a callback function. Use <Ctrl>+<c> to terminate this program.

#cd /usr/src/Neon/Sample/Neon_ListenTrigger

#sudo ./NeonListenTrigger

9. Run Neon_Python Sample:

#cd /usr/src/Neon/Sample/Neon_Python

#sudo python3 Neon.py

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2 Function Library

2.1 List of Functions

Category	Function
Device Control	Neon_GetDeviceInfo
DIO	Neon_GetDINums
	Neon_GetDONums
	Neon_SetDOState
	Neon_GetDIState
	Neon_SetDIO0Config
	Neon_GetDIO0Config
	Neon_SetDITriggerPolarity
	Neon_GetDITriggerPolarity
	Neon_SetDITriggerCallback
	Neon_SetStrobeOutEnable
	Neon_GetStrobeOutEnable
	Neon_SetStrobeOutDelay
	Neon_GetStrobeOutDelay
	Neon_SetStrobeOutPulseWidth
	Neon_GetStrobeOutPulseWidth
	Neon_SetExposureDelay
	Neon_GetExposureDelay
	Neon_SetDebounceTime
	Neon_GetDebounceTime
	Neon_SetStrobeOutInvertedPolarity
	Neon_GetStrobeOutInvertedPolarity

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.2 Function Library

2.2.1 Device Control Function

Neon_GetDeviceInfo

Retrieves device information.

Syntax

C/C++

```
int Neon_GetDeviceInfo (char* sVersion, int
iSize)
```

Parameter(s)

sVersion:

An allocated buffer must be assigned to read the device version. The device information format is:

[Device1 Name]:[Version]

[Device2 Name]:[Version]

…

Each device description is separated by a break line character '\n'

iSize:

The number of "sVersion" bytes. The suggested size is 256.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.2.2 DIO Functions

Neon_GetDINums

Retrieves the number of DIs.

Syntax

C/C++

```
int Neon_GetDINums (unsigned int* iCounts)
```

Parameter(s)

iCounts:

Returns the number of DIs.

Return Code

Returns true if successful, otherwise returns COMMON_ERROR.

Neon_GetDONums

Retrieves the number of DOs.

Syntax

C/C++

```
int Neon_GetDONums (unsigned int* iCounts)
```

Parameter(s)

iCounts:

Returns the number of DOs.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_SetDOState

Turns DO on or off.

Syntax

C/C++

```
int Neon_SetDOState (unsigned int iIndex, int
iState)
```

Parameter(s)

iIndex:

Sets the Index of the DO; zero-based.

iState:

Sets the state of the DO: 1 is on, 0 is off.

e.g. Neon_SetDOState (1,1) -> Sets DO channel 1 to on

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetDIState

Gets the DI's state.

Syntax

C/C++

```
int Neon_GetDIState (unsigned int iIndex, int*
iState)
```

Parameter(s)

iIndex:

Sets the Index of the DI; zero-based.

iState:

Gets the state of the DI: 1 is on, 0 is off.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Neon_SetDIO0Config

Sets the DIO0 config type. DIO_CONFIGTYPE_DIGITAL means the DI0 and DO0 are the same as the general DI and DO. DIO_CONFIGTYPE_SNAPSHOT means the DI0 and DO0 are for triggering the camera and strobe out.

Syntax

C/C++

```
int Neon_SetDIO0Config (int iCfg_type)
```

Parameter(s)

iCfg_type:

The config type value can be 0 or 1. Use the following enumeration to define the value.

```
enum DIO_CONFIGTYPE
{
    DIO_CONFIGTYPE_SNAPSHOT=0,
    DIO_CONFIGTYPE_DIGITAL,
    DIO_CONFIGTYPE_MAXIMUM
};
```

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetDIO0Config

Gets the DIO0 config type. DIO_CONFIGTYPE_DIGITAL means the DI0 and DO0 are the same as the general DI and DO. DIO_CONFIGTYPE_SNAPSHOT means the DI0 and DO0 are for triggering the camera and strobe out.

Syntax

C/C++

```
int Neon_GetDIO0Config (int* iCfg_type)
```

Parameter(s)

iCfg_type:

The config type value can be 0 or 1. Use the following enumeration to define the value.

```
enum DIO_CONFIGTYPE
{
    DIO_CONFIGTYPE_SNAPSHOT=0,
    DIO_CONFIGTYPE_DIGITAL,
    DIO_CONFIGTYPE_MAXIMUM
};
```

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Neon_SetDITriggerPolarity

Sets the DI trigger polarity. (Default is 0)

Syntax

C/C++

```
int Neon_SetDITriggerPolarity (unsigned int
iIndex, int iPositiveTriggerPolarity)
```

Parameter(s)

iIndex:

Sets the DI Index; zero-based.

iPositiveTriggerPolarity:

The trigger polarity. Refer to DI_TRIGGERPOLARITY.

```
enum DI_TRIGGERPOLARITY
{
    DI_TRIGGERPOLARITY_FALLING=0,
    DI_TRIGGERPOLARITY_RISING,
    DI_TRIGGERPOLARITY_BOTH,
    DI_TRIGGERPOLARITY_MAXIMUM
};
```

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetDITriggerPolarity

Gets the DI trigger polarity. (Default is 0)

Syntax

C/C++

```
int Neon_GetDITriggerPolarity (unsigned int
iIndex, int* iPositiveTriggerPolarity)
```

Parameter(s)

iIndex:

Gets the DI Index; zero-based.

iPositiveTriggerPolarity:

The trigger polarity. Refer to DI_TRIGGERPOLARITY.

```
enum DI_TRIGGERPOLARITY
{
    DI_TRIGGERPOLARITY_FALLING=0,
    DI_TRIGGERPOLARITY_RISING,
    DI_TRIGGERPOLARITY_BOTH,
    DI_TRIGGERPOLARITY_MAXIMUM
};
```

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_SetDITriggerCallback

Sets the DI trigger callback function. When the DI is triggered, the callback function will be executed.

Syntax

C/C++

```
int Neon_SetDITriggerCallback
(NEON_DITRIGGER_CALLBACK callback)
```

Parameter(s)

callback:

The NEON_DITRIGGER_CALLBACK function pointer.

```
typedef void (*NEON_DITRIGGER_CALLBACK)
(unsigned int index, int currentState);
```

index:

The trigger index of the DI. DI0 will be triggered only when the config type of DIO0 is DIGITAL.

currentState:

The current state of the DI.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_SetStrobeOutEnable

Enables/disables the strobe out trigger when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_SetStrobeOutEnable(int iEnable)
```

Parameter(s)

iEnable:

1 = enable, 0 = disable.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetStrobeOutEnable

Gets the enabled state of the strobe out trigger when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_GetStrobeOutEnable(int* iEnable)
```

Parameter(s)

iEnable:

Returns the enabled state.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Neon_SetStrobeOutDelay

Sets the delay time of the strobe out in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_SetStrobeOutDelay (unsigned int
iMicrosecond)
```

Parameter(s)

iMicrosecond:

The number of microseconds for the strobe out delay.

(0 - 4294967295, map to 1 - 4294967296 µs)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetStrobeOutDelay

Gets the delay time of the strobe out in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_GetStrobeOutDelay (unsigned int*
iMicrosecond)
```

Parameter(s)

iMicrosecond:

Returns the number of microseconds of the strobe out delay.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_SetStrobeOutPulseWidth

Sets the pulse width of the strobe out in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_SetStrobeOutPulseWidth (unsigned int
iMicrosecond)
```

Parameter(s)

iMicrosecond:

The number of microseconds for the strobe out pulse width.

(0 - 4294967295, map to 1 - 4294967296 µs)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetStrobeOutPulseWidth

Gets the pulse width of the strobe out in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_GetStrobeOutPulseWidth (unsigned int*
iMicrosecond)
```

Parameter(s)

iMicrosecond:

Returns the number of microseconds of the strobe out pulse width.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Neon_SetExposureDelay

Sets the delay time of the exposure in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_SetExposureDelay (unsigned int
iMicrosecond)
```

Parameter(s)

iMicrosecond:

The number of microseconds for the exposure delay.

(0 - 4294967295 µs)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetExposureDelay

Gets the delay time of the exposure in microseconds when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_GetExposureDelay (unsigned int*
iMicrosecond)
```

Parameter(s)

iMicrosecond:

Returns the number of microseconds of the exposure delay.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_SetDebounceTime

Sets the debounce time when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_SetDebounceTime(unsigned int
iNtimes40ns)
```

Parameter(s)

iNtimes40ns:

The number of microseconds for setting debounce time.

(0 - 16777215, 0 is bypass; others (1 - 16777215) * 40ns)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetDebounceTime

Gets the debounce time when DIO0 is set as DIO_CONFIGTYPE_SNAPSHOT.

Syntax

C/C++

```
int Neon_GetDebounceTime(unsigned int*
iNtimes40ns)
```

Parameter(s)

iNtimes40ns:

Returns the number of microseconds of debounce time.

Return Code

Returns true if successful, otherwise COMMON_ERROR.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Neon_SetStrobeOutInvertedPolarity

Sets the inverted polarity of Strobe Out. If set to 1, the output polarity will be inverted. (Default is 0)

Syntax

C/C++

```
int Neon_SetStrobeOutInvertedPolarity(int iInvertedPolarity)
```

Parameter(s)

iInvertedPolarity:

Invert the polarity of strobe out. (0 is normal open and 1 is normal close)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

Neon_GetStrobeOutInvertedPolarity

Gets the inverted polarity of Strobe Out. (Default is 0)

Syntax

C/C++

```
int Neon_GetStrobeOutInvertedPolarity(int* iInvertedPolarity)
```

Parameter(s)

iInvertedPolarity:

Invert the polarity of strobe out. (0 is normal open and 1 is normal close)

Return Code

Returns true if successful, otherwise COMMON_ERROR.

This page intentionally left blank.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Important Safety Instructions

For user safety, please read and follow all instructions, Warnings, Cautions, and Notes marked in this manual and on the associated device before handling/operating the device, to avoid injury or damage.

S'il vous plaît prêter attention stricte à tous les avertissements et mises en garde figurant sur l'appareil , pour éviter des blessures ou des dommages.

- Read these safety instructions carefully
- Keep the User's Manual for future reference
- Read the Specifications section of this manual for detailed information on the recommended operating environment
- The device can be operated at an ambient temperature of 55ºC (with DC supply) and 50ºC (with adapter supply);
- When installing/mounting or uninstalling/removing device; or when removal of a chassis cover is required for user servicing:
  - Turn off power and unplug any power cords/cables
  - Reinstall all chassis covers before restoring power
- To avoid electrical shock and/or damage to device:
  - Keep device away from water or liquid sources
  - Keep device away from high heat or humidity
  - Keep device properly ventilated (do not block or cover ventilation openings)
  - Always use recommended voltage and power source settings
  - Always install and operate device near an easily accessible electrical outlet
  - Secure the power cord (do not place any object on/over the power cord)
  - Only install/attach and operate device on stable surfaces and/or recommended mountings
- If the device will not be used for long periods of time, turn off and unplug from its power source

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- Never attempt to repair the device, which should only be serviced by qualified technical personnel using suitable tools
- A Lithium-type battery may be provided for uninterrupted backup or emergency power.



Risk of explosion if battery is replaced with one of an incorrect type; please dispose of used batteries appropriately.
Risque d'explosion si la pile est remplacée par une autre de type incorrect. Veuillez jeter les piles usagées de façon appropriée.

- The device must be serviced by authorized technicians when:
  - The power cord or plug is damaged
  - Liquid has entered the device interior
  - The device has been exposed to high humidity and/or moisture
  - The device is not functioning or does not function according to the User's Manual
  - The device has been dropped and/or damaged and/or shows obvious signs of breakage
- Disconnect the power supply cord before loosening the thumbscrews and always fasten the thumbscrews with a screwdriver before starting the system up
- It is recommended that the device be installed only in a server room or computer room where access is:
  - Restricted to qualified service personnel or users familiar with restrictions applied to the location, reasons therefor, and any precautions required
  - Only afforded by the use of a tool or lock and key, or other means of security, and controlled by the authority responsible for the location
- If PoE (Power over Ethernet) is enabled for the device, the system can ONLY be deployed indoors. Unless otherwise noted, the PoE system is NOT designed to withstand the rigors of outdoor use.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

	**BURN HAZARD** Touching this surface could result in bodily injury. To reduce risk, allow the surface to cool before touching. ***RISQUE DE BRÛLURES*** *Ne touchez pas cette surface, cela pourrait entraîner des blessures.* *Pour éviter tout danger, laissez la surface refroidir avant de la toucher.*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

This page intentionally left blank.

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Getting Service

Ask an Expert: http://askanexpert.adlinktech.com

ADLINK Technology, Inc.
9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
5215 Hellyer Avenue, #110
San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
300 Fang Chun Rd., Zhangjiang Hi-Tech Park
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology GmbH
Hans-Thoma-Strasse 11
D-68163 Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

Please visit the Contact page at www.adlinktech.com for information on how to contact the ADLINK regional office nearest you:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000