Neousys Technology Inc.

Nuvis-5306RT

User Manual

Rev. 1.0

Table of Contents

Appendix B: PoE On/ Off Control

Appendix C: NuMCU Programming Guide

Appendix D: DTIOv2 Programming Guide

Programming with DTIOv2 Library

Quadrature Encoder Interface (QEI)

Deterministic Trigger I/O version 2 (DTIOv2)

DTIOv2 Programming Tips

Appendix E: Advanced DTIOv2

Pulse Width Modulator (PWM)

Advanced Examples

Legal Information

All Neousys Technology Inc. products shall be subject to the latest Standard Warranty Policy

Neousys Technology Inc. may modify, update or upgrade the software, firmware or any accompanying user documentation without any prior notice. Neousys Technology Inc. will provide access to these new software, firmware or documentation releases from download sections of our website or through our service partners.

Before installing any software, applications or components provided by a third party, customer should ensure that they are compatible and interoperable with Neousys Technology Inc. product by checking in advance with Neousys Technology Inc.. Customer is solely responsible for ensuring the compatibility and interoperability of the third party's products. Customer is further solely responsible for ensuring its systems, software, and data are adequately backed up as a precaution against possible failures, alternation, or loss.

For questions in regards to hardware/ software compatibility, customers should contact Neousys Technology Inc. sales representative or technical support.

To the extent permitted by applicable laws, Neousys Technology Inc. shall NOT be responsible for any interoperability or compatibility issues that may arise when (1) products, software, or options not certified and supported; (2) configurations not certified and supported are used; (3) parts intended for one system is installed in another system of different make or model.

Contact Information

Headquarters (Taipei, Taiwan)	**Neousys Technology Inc.** 15F, No.868-3, Zhongzheng Rd., Zhonghe Dist., New Taipei City, 23586, Taiwan Tel: +886-2-2223-6182 Fax: +886-2-2223-6183 Email, Website
Americas (Illinois, USA)	**Neousys Technology America Inc.** 3384 Commercial Avenue, Northbrook, IL 60062, USA Tel: +1-847-656-3298 Email, Website
China	**Neousys Technology (China) Ltd.** Room 612, Building 32, Guiping Road 680, Shanghai Tel: +86-2161155366 Email, Website

Declaration of Conformity

FCC	This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at own expense.
CE	The product(s) described in this manual complies with all applicable European Union (CE) directives if it has a CE marking. For computer systems to remain CE compliant, only CE-compliant parts may be used. Maintaining CE compliance also requires proper cable and cabling techniques.

Copyright Notice

Disclaimer

This manual is intended to be used as an informative guide only and is subject to change without prior notice. It does not represent commitment from Neousys Technology Inc. Neousys Technology Inc. shall not be liable for any direct, indirect, special, incidental, or consequential damages arising from the use of the product or documentation, nor for any infringement on third party rights.

Patents and Trademarks

Neousys, the Neousys logo, Expansion Cassette, MezIO™ are registered patents and trademarks of Neousys Technology, Inc.

Windows is a registered trademark of Microsoft Corporation.

Intel®, Core™ is a registered trademark of Intel Corporation

NVIDIA®, GeForce® is a registered trademark of NVIDIA Corporation

Texas Instruments (TI) and Sitara are registered trademarks of Texas Instruments Incorporated.

All other names, brands, products or services are trademarks or registered trademarks of their respective owners.

Safety Precautions

- Read these instructions carefully before you install, operate, or transport the system.
- Install the system or DIN rail associated with, at a sturdy location
- Install the power socket outlet near the system where it is easily accessible
- Secure each system module(s) using its retaining screws
- Place power cords and other connection cables away from foot traffic. Do not place items over power cords and make sure they do not rest against data cables
- Shutdown, disconnect all cables from the system and ground yourself before touching internal modules
- Ensure that the correct power range is being used before powering the device
- Should a module fail, arrange for a replacement as soon as possible to minimize down-time
- If the system is not going to be used for a long time, disconnect it from mains (power socket) to avoid transient over-voltage

Service and Maintenance

- ONLY qualified personnel should service the system
- Shutdown the system, disconnect the power cord and all other connections before servicing the system
- When replacing/ installing additional components (expansion card, memory module, etc.), insert them as gently as possible while assuring connectors are properly engaged

ESD Precautions

- Handle add-on module, motherboard by their retention screws or the module's frame/ heat sink. Avoid touching the PCB circuit board or add-on module connector pins
- Use a grounded wrist strap and an anti-static work pad to discharge static electricity when installing or maintaining the system
- Avoid dust, debris, carpets, plastic, vinyl and styrofoam in your work area.
- Do not remove any module or component from its anti-static bag before installation

About This Manual

This manual introduces Neousys Technology Nuvis-5306RT, a fully featured machine vision controller that supports 6th generation Intel Core i7/ i5 processors with vision specific I/O, real-time control and GPU-computing.

Revision History

Version	Date	Description
1.0	Mar. 2018	Initial release

1 Introduction

Neousys' Nuvis-5306RT is a fully featured machine vision controller that supports 6th Gen Intel® Core™ i7/ i5 processors with vision specific I/Os, real-time control and GPU-computing.

1.1 Nuvis-5306RT Overview

Nuvis-5306RT features Intel® 6th Gen Core™ i7/i5 processor paired with Intel® Q170 Platform Controller Hub. It is the world's first fully featured machine vision controller in a compact footprint that integrates exceptional computing power, built-in camera interfaces and real-time vision-specific I/O controls.

Nuvis-5306RT provides a powerful machine vision platform with integrated LED lighting controller, camera trigger, encoder input, pulse width modulation (PWM) output and digital I/O, to simultaneously connect and control all vision devices. With Neousys' patented technologies, Deterministic Trigger I/O (DTIO) and NuMCU (based on MCU-based architecture), they manage all vision-specific I/Os and allow users to program a deterministic timing correlation between input and output signals in microsecond scale. In addition, the innovative NuMCU technology grants users full control of MCU by integrating programming environment, run-time download/debug capability, to achieve comprehensive I/O control.

Nuvis-5306RT features rich I/Os that include four IEEE 802.3at PoE, four USB 3.0, four USB 2.0, one VGA, two DisplayPorts, three serial COM, one mic-in and one speaker-out port. In addition, Nuvis-5306RT can also accommodate an NVIDIA® GeForce® GTX 950/ 1050 to leverage CPU-accelerated vision library or deep-learning vision applications. Combining built-in PoE+, USB 3.0 interfaces and the expandability for Camera Link and CoaXPress, Nuvis-5306RT is the ideal platform for demanding machine vision controller applications.

1.2 Nuvis-5306RT Specifications

System Platform	
Processor	Supports 6th-Gen Intel® Core™ LGA1151 CPU ● I ntel® Core™ i7-6700 (8M Cache,3.4/4.0 GHz, 65W TDP) ● Intel® Core™ i5-6500 (6M Cache, 3.2/3.6 GHz, 65W TDP) ● Intel® Core™ i7-6700TE (8M Cache, 2.4/3.4 GHz, 35W TDP) ● Intel® Core™ i5-6500TE (6M Cache, 2.3/3.3 GHz, 35W TDP)
Chipset	Intel® Q170 Platform
Graphics	Integrated Intel® HD Graphics 530
Memory	2x 260-pin SO-DIMM sockets, up to 32 GB DDR4 2133 MHz SDRAM
Vision Specific I/O Interface	
LED Lighting Controller	4-CH LED lighting controller output , supporting ● Constant current mode (up to 1 A per channel, 100 kHz dimming control) ● Constant voltage mode (24 VDC, 100 kHz dimming control)
Camera Trigger	4-CH camera trigger output (12 VDC output)
Encoder Input	1-CH quadrature encoder input (A/B/Z)
Isolated Digital Output	4-CH isolated high-speed digital output (<2 us transient time, for strobe/PWM) 4-CH isolated high-current digital output (up to 500 mA rated current)
Isolated Digital Input	8-CH isolated high-speed digital input (<2 us transient time)
Real-time I/O Control	Patented MCU-based real-time I/O control with DTIO V2 or OpenMCU firmware
I/O Interface	
Ethernet	1x Ethernet port by Intel® I219 5x Ethernet ports by Intel® I210
PoE+	3x IEEE 802.3at (80W total) Gigabit PoE+ (port 3 ~6)
Video Port	1x VGA supporting maximum 1920 x 1200 resolution 1x DVI-D supporting maximum 1920 x 1200 resolution 2x DisplayPort outputs, supporting maximum 3840 x 2160 resolution
USB	4x USB 3.0 ports via native xHCI controller 4x USB 2.0 ports
Serial Port	2x software-programmable RS-232/422/485 (COM1 & COM2) 1x RS-232 port (COM3)
Audio	1x mic-in and 1x speaker-out
Storage Interface	

SATA HDD	2x internal SATA port for 2.5” HDD/SSD (support RAID 0/ 1)
mSATA	1x full-size mSATA (mux with mini-PCIe)
Expansion Bus	
PCI/ PCI Express	1x PCIe x16 slot @ Gen3, 8-lanes PCIe signals in Cassette, supporting ● 75W NVIDIA® GeForce® GTX 1050 GPU card ● COTS CameraLink and CoaXPress camera interface card
Mini PCI-E	1x internal mini PCI Express socket with front-accessible SIM socket 1x internal mini PCI Express socket with internal SIM socket (mux with mSATA)
Power Supply & Ignition Control	
DC Input	1x 3-pin pluggable terminal block for 8~35VDC DC input
Remote Ctrl. & Status Output	1x 10-pin (2x5) wafer connector for remote on/off control and status LED output
Max. Power Consumption	
Mechanical	
Dimension	240 mm (W) x 225 mm (D) x 111 mm (H)
Weight	4.9 kg (including CPU, GPU, memory and HDD)
Mounting	Wall mount bracket Rack mount kit (optional)
Environmental	
Operating Temperature	Using i7-6700TE or i5-6500TE, configured @ 35W TDP -25°C ~ 60°C ** Using i7-6700 or i5-6500, configured @ 65W/ 51W TDP -25°C ~ 60°C */** (configured as 35W CPU mode) -25°C ~ 50°C */** (configured as 65W CPU mode)
Storage Temperature	-40°C ~85°C
Humidity	10%~90% , non-condensing
Vibration	Operating, 5 Grms, 5-500 Hz, 3 Axes (w/ SSD, according to IEC60068-2-64)
Shock	Operating, 50 Grms, Half-sine 11 ms Duration (w/ SSD, according to IEC60068-2-27)
EMC	CE/FCC Class A, according to EN 55022 & EN 55024

* *For i7-6700 running at 65W mode, the high operating temperature shall be limited to 50°C and thermal throttling may occur when sustained full-loading applied. Users can configure CPU power in BIOS to obtain higher operating temperature.*

** *For sub-zero operating temperature, a wide temperature HDD drive or Solid State Disk (SSD) is required.*

NOTE

When running CPUs with greater than 35W TDP, the maximum operating temperature shall be limited to 50°C and thermal throttling may occur when sustained full-load is applied. Users can configure CPU power in BIOS to obtain higher operating temperature.

1.3 Isolated DIO Specifications

Isolated Digital Input	
No. of Channel	8-CH Isolated Digital Input Channels
Logic Level	Logic High: 5 to 24V Logic Low: 0 to 1.5V
Isolated Voltage	2500 Vrms
Input Resistance	4.7kΩ
Operation Mode	Polling I/O with Change-of-State Interrupt, DTIO
Isolated Digital Output	
No. of Channel	8-CH Isolated Digital Output Channels
Sink Current (per channel)	100 mA (sustained loading) 250 mA (peak loading)
Isolated Voltage	2500 Vrms
Operation Mode	Polling, Change-of-State Interrupt
Output Type	Power MOSFET + Analog Device iCoupler®
Operation Mode	Polling I/O, DTIO

1.4 Supported CPUs

The system supports Intel® 6th Gen. i7/ i5/ i3 processor via the LGA 1151 CPU socket.

- Intel® Core™ i7-6700 (8M Cache,3.4/ 4.0 GHz, 65W TDP)
- Intel® Core™ i5-6500 (6M Cache, 3.2/ 3.6 GHz, 65W TDP)
- Intel® Core™ i7-6700TE (8M Cache, 2.4/ 3.4 GHz, 35W TDP)
- Intel® Core™ i5-6500TE (6M Cache, 2.3/ 3.3 GHz, 35W TDP)

Alternatively, you may also select a processor from Intel's embedded solution "Products formerly Skylake" that utilizes LGA1151 CPU socket.

NOTE

Other processors may result in different overall system power consumption or generate excess heat.

Avoid using CPUs not tested by Neousys Technology, as Neousys Technology may not be able guarantee the system's stability and functionality under its designated working environment.

If in doubt, please contact Neousys technical support!

1.5 Supported Graphics Card

Nuvis-5306RT series supports NVIDIA GTX 1050 GPU cards with maximum 75W TDP. Additionally, the expansion Cassette and heat dissipation mechanism shall work with selected graphics cards to obtain best thermal stability of GPU. For users who do not purchase graphics card directly from Neousys, please refer to the following list of recommended consumer-grade graphics cards.

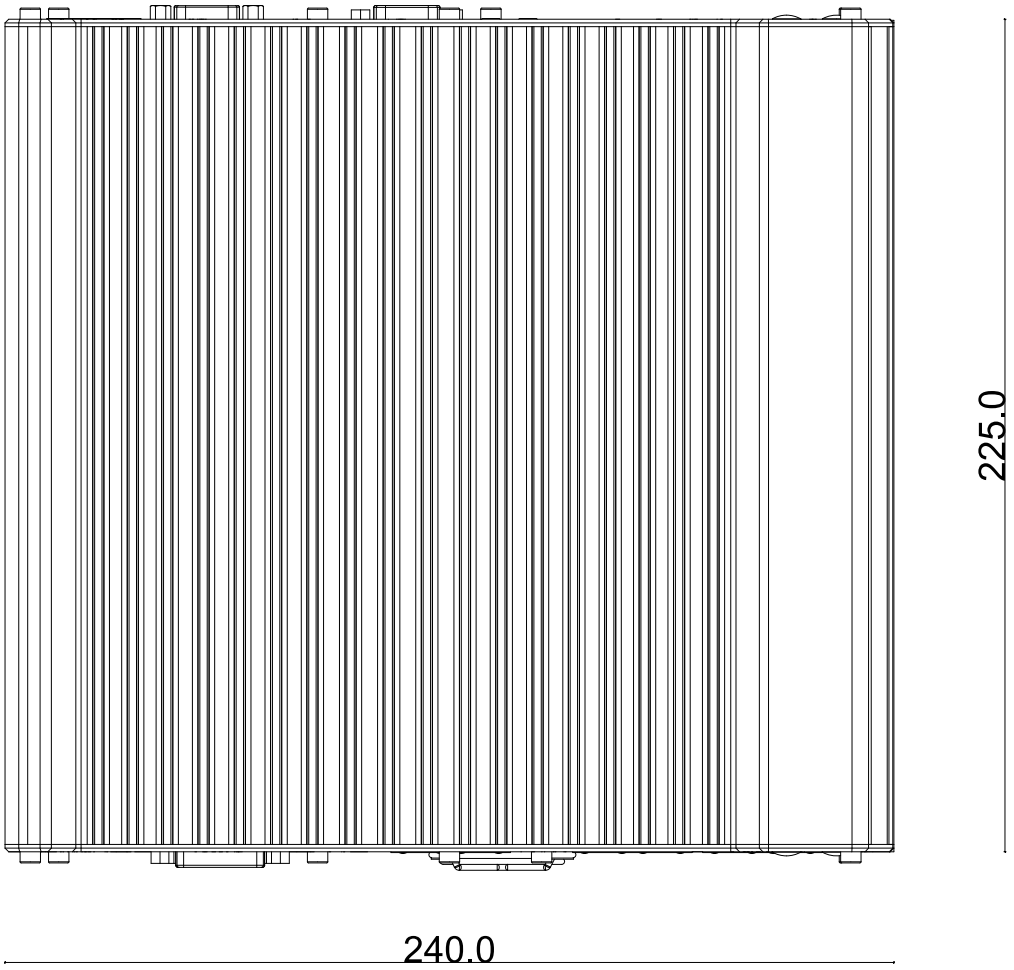
GTX 1050

- GIGABYTE GeForce® GTX 1050 D5 2G

Neousys will keep updating this list as new graphics cards available in the market. Please contact us for the latest GPU card support list.

1.6 Dimension

1.6.1 Superior View

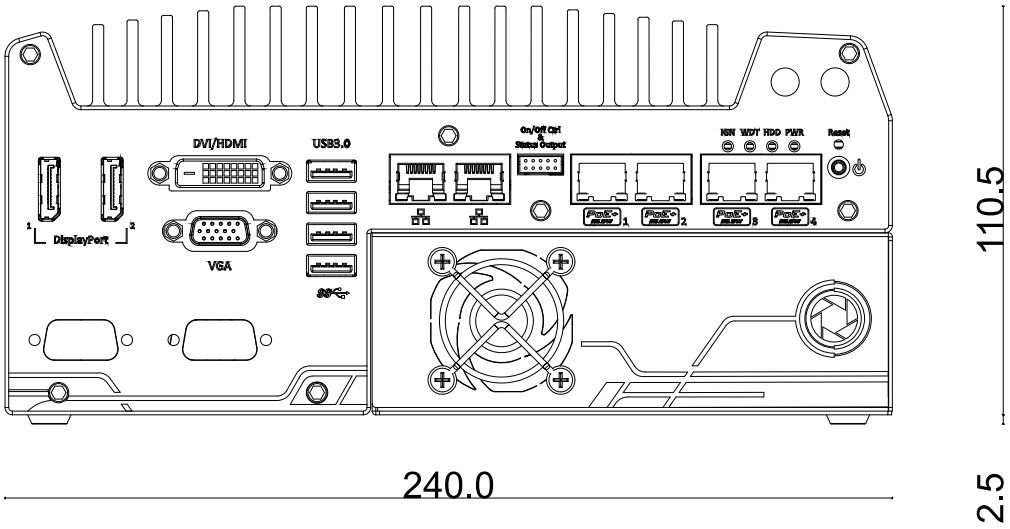


NOTE

All measurements are in millimeters (mm).

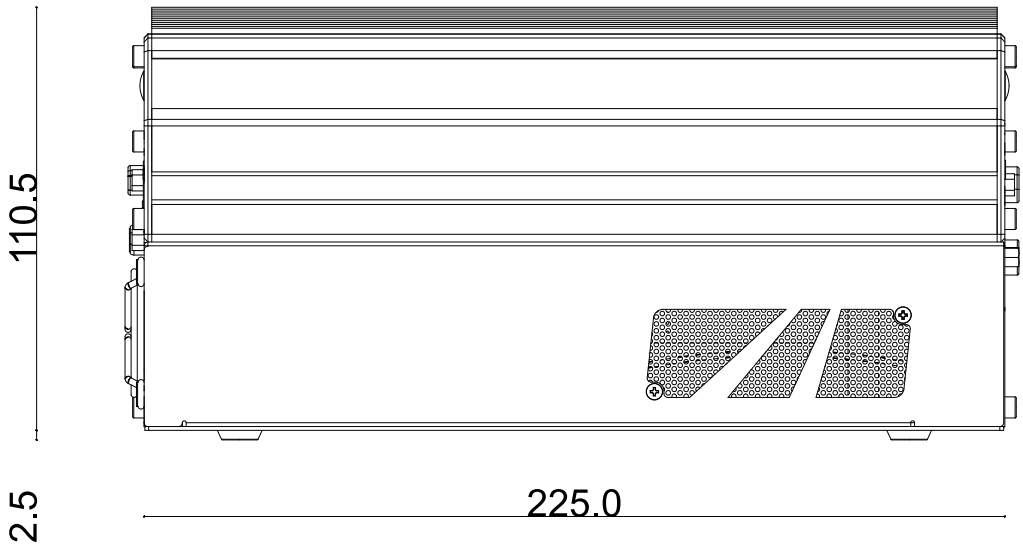
1.6.2 Front Panel View

The numbers “2.5” represents the height of the rubber stands at 2.5mm.



1.6.3 Side View

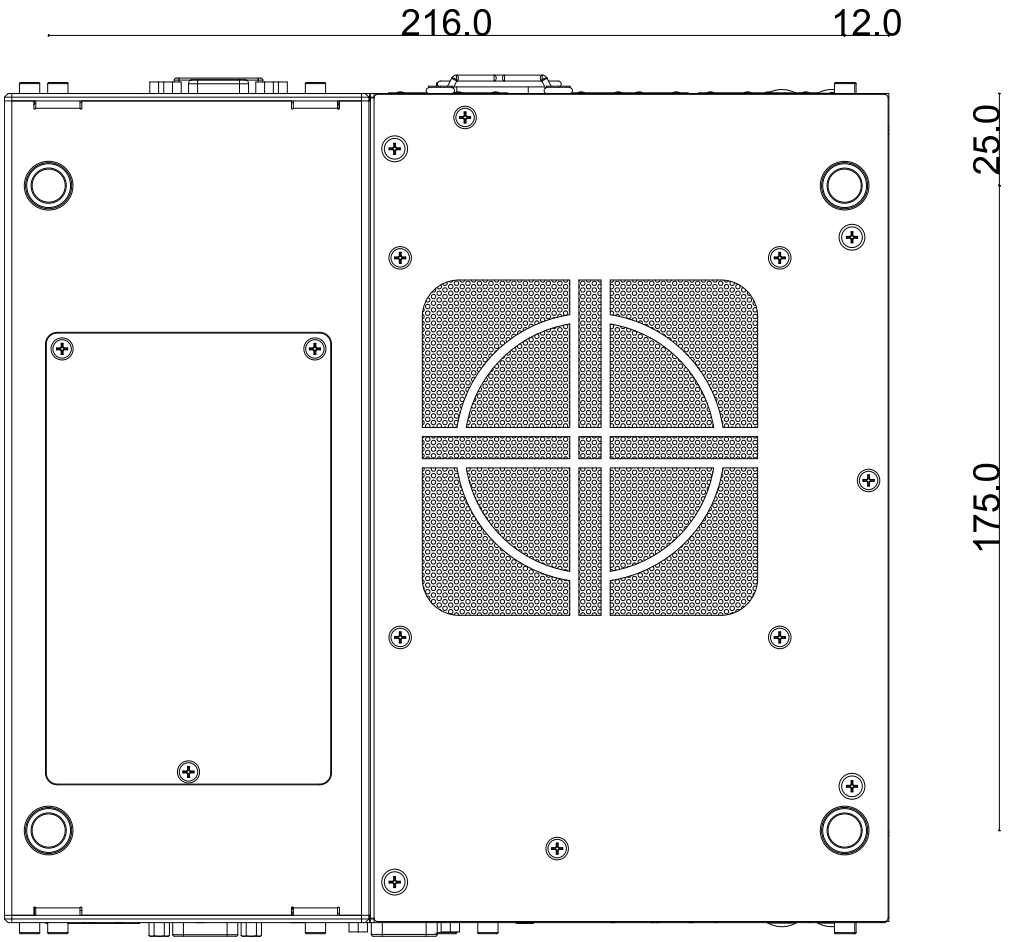
The numbers “2.5” represents the height of the rubber stands at 2.5mm.



NOTE

All measurements are in millimeters (mm).

1.6.4 Bottom View

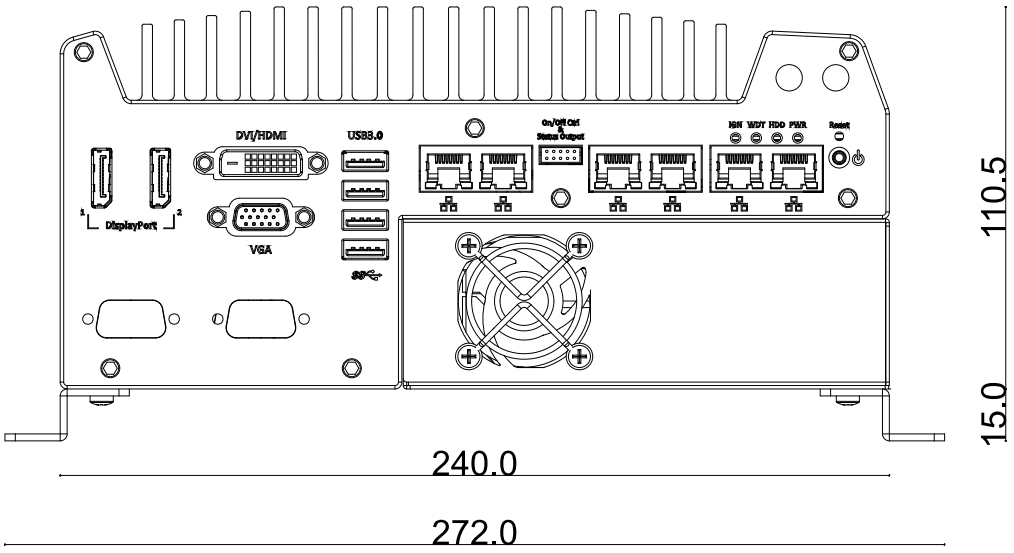


NOTE

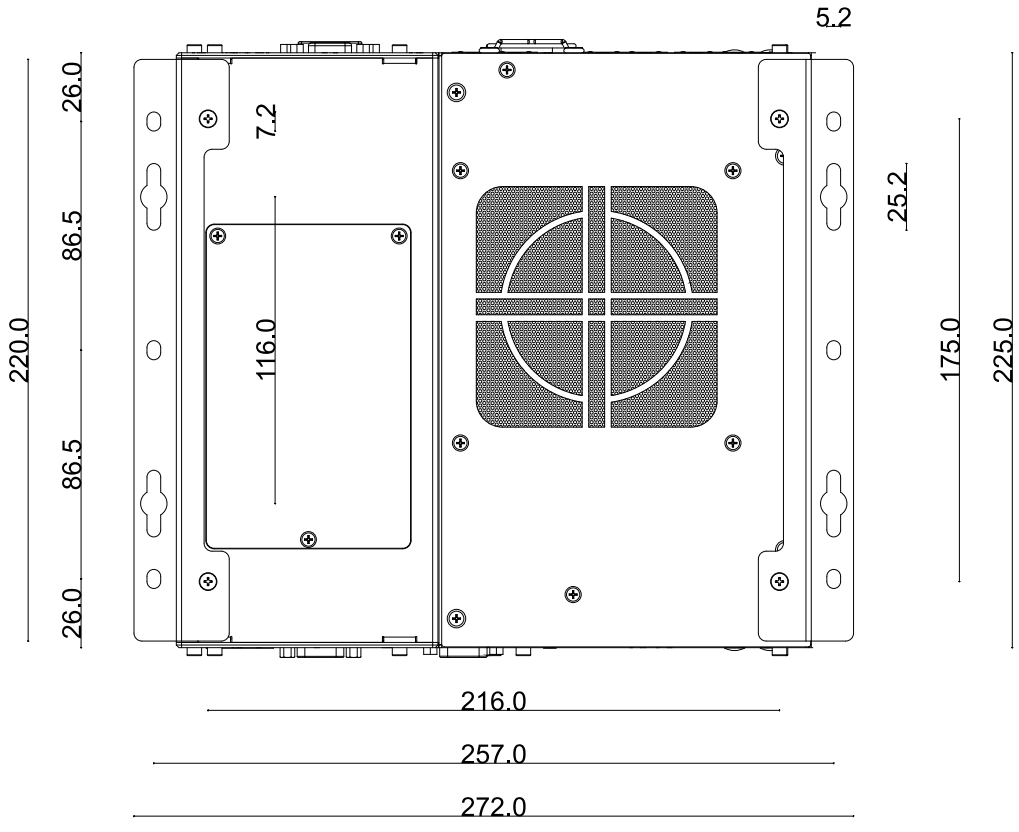
All measurements are in millimeters (mm).

1.7 Dimensions with Mount Bracket

1.7.1 Front View



1.7.2 Bottom View



NOTE

All measurements are in millimeters (mm).

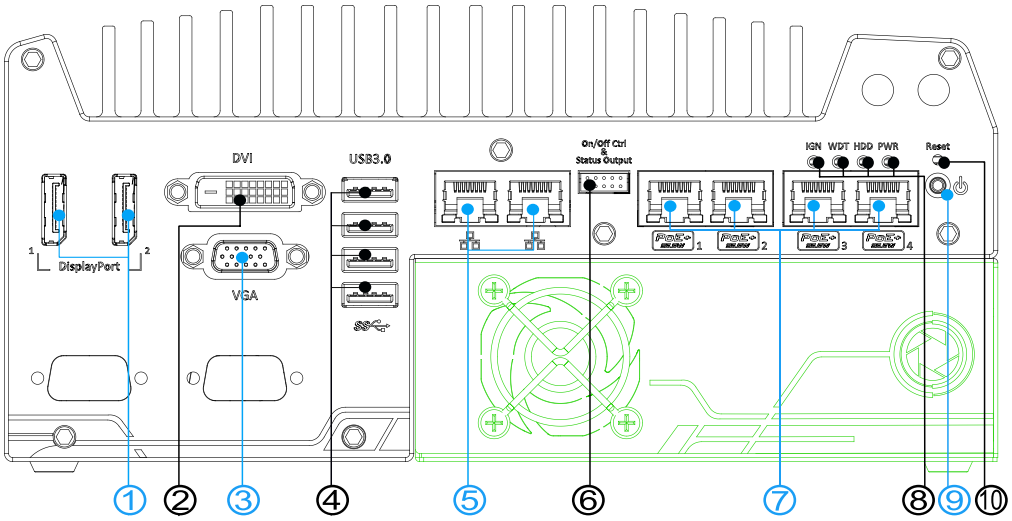
2 System Overview

Upon receiving and unpacking your system, please check immediately if the package contains all the items listed in the following table. If any item(s) are missing or damaged, please contact your local dealer or Neousys Technology.

2.1 Nuvis-5306RT Packing List

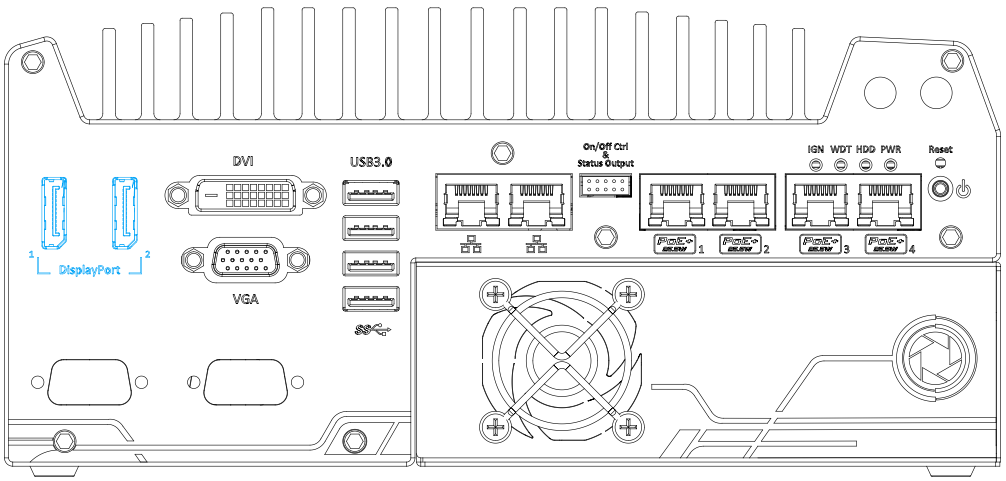
System / Pack	Nuvis-5306RT	Qty
1	Nuvis-5306RT system (If you ordered graphics card/ CPU/ RAM/ HDD, please verify these items)	1
2	Accessory box, which contains	
	● Neousys drivers & utilities DVD	1
	● Dampening bracket for in-vehicle deployment	2
	● Shock-absorbing grommet	8
	● DVI-to-VGA adapter	1
	● 3-pin power terminal block	1
	● HDD thermal pad for 2.5” HDD/SSD (if HDD is not installed)	1
	● Screw pack	1

2.2 Nuvis-5306RT Front Panel



No.	Item	Description
1	DisplayPort 1 & 2	Support display resolutions up to 4096 x 2304. Compatible with HDMI/ DVI via respective adapter/ cable (support resolution may vary).
2	DVI port	DVI-D output supports resolution up to 1920x1200@60Hz and is compatible with other digital connections via an adapter
3	VGA port	VGA output supports resolution up to 1920x1200@60Hz
4	USB 3.0 ports	USB 3.0 port, up to 5Gbit/s data transfer bandwidth.
5	GbE ports	Gigabit Ethernet ports offer fast network access.
6	On/ off control & status output	Allows for external switch extension when the system is placed inside a cabinet.
7	PoE+ GbE ports	Power over Ethernet (PoE) port can provide both data connection and electric power to devices (eg. IP camera).
8	System status LEDs	Four system LEDs, Ignition control (IGN), Watchdog Timer (WDT), Hard Disk Drive (HDD) and Power (PWR).
9	Power button	Use this button to turn on or force shutdown the system.
10	Reset button	Use this button to manual restart the system.
Area in Green	Cassette Enclosure	The cassette enclosure offers a separate compartment to manage thermal conditions and reduce installation complications of an add-on card.

2.2.1 DisplayPort



The system has dual DisplayPort (DP) outputs which are digital display interfaces that mainly connect video source and carry audio to a display device. When connecting a single DP, it can deliver up to 4096 x 2304 resolution and each port can deliver up to 2880 x 1800 resolution when both DPs are connected in conjunction. The system is designed to support passive DP adapter/ cable. You can connect to display devices using DP-to-HDMI cable or DP-to-DVI cable.

DP-to-HDMI

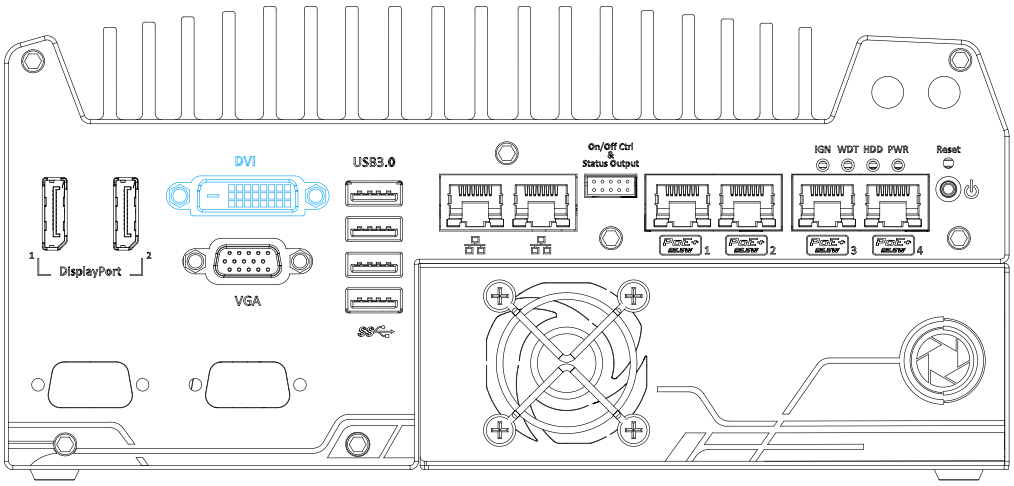
DP-to-DVI

The system supports triple independent display outputs in the following combination of VGA, DVI/ HDMI and DisplayPort. To support multiple display outputs and achieve best DVI/ HDMI output resolution in Windows, you need to install corresponding graphics driver. Please refer to section OS Support and Driver Installation for details.

Triple Independent Display Configuration (resolution may be limited)

Active display 1	Active display 2	Active display 3
DisplayPort	DisplayPort	DVI or VGA
DisplayPort	DVI	VGA

2.2.2 DVI Port



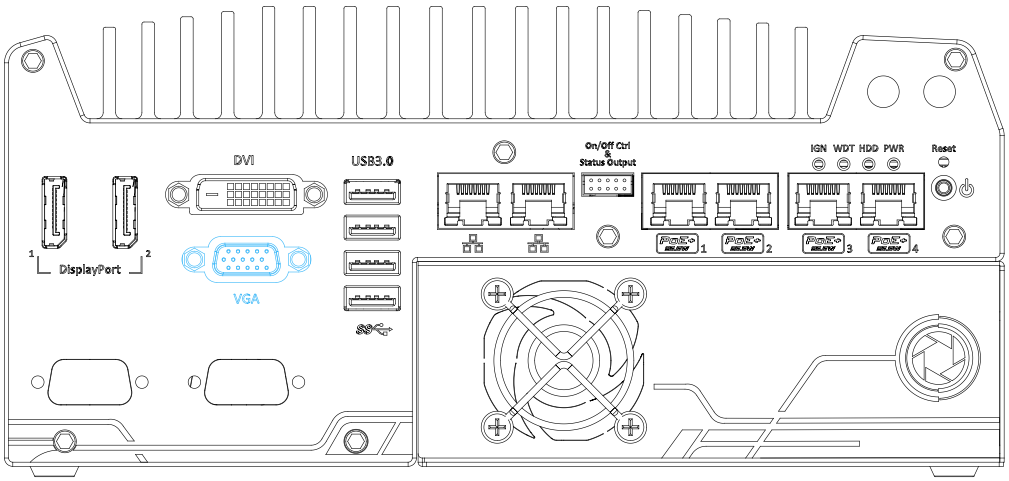
DVI-D transmits graphics data in digital format and therefore can deliver better image quality at high resolution. The DVI connector on the front panel can either output DVI signals or other digital signals (via an adapter/ cable) depending on the display device connected. It supports resolutions up to 1920x1200@60Hz.

The system supports triple independent display outputs in the following combination of VGA, DVI and DisplayPort. To support multiple display outputs and achieve best DVI output resolution in Windows, you need to install corresponding graphics driver. Please refer to section OS Support and Driver Installation for details.

Triple Independent Display Configuration (resolution may be limited)

Active display 1	Active display 2	Active display 3
DisplayPort	DisplayPort	DVI or VGA
DisplayPort	DVI	VGA

2.2.3 VGA Port



VGA connector is the most common video display connection. The VGA output supports up to 1920x1200@60Hz resolution. By default, the VGA output is set to "always-on". For users who want to use only digital display interface (eg. DVI or DP), the VGA Output setting can be disabled. To disable, press F2 upon system startup, go to "**Advanced** > **System Agent (SA) Configuration** > **Graphics Configuration** > **VGA Output** > [**Disable**].

The system supports triple independent display outputs in the following combination of VGA, DVI/ HDMI and DisplayPort. To support multiple display outputs and achieve best DVI/ HDMI output resolution in Windows, you need to install corresponding graphics driver. Please refer to section OS Support and Driver Installation for details.

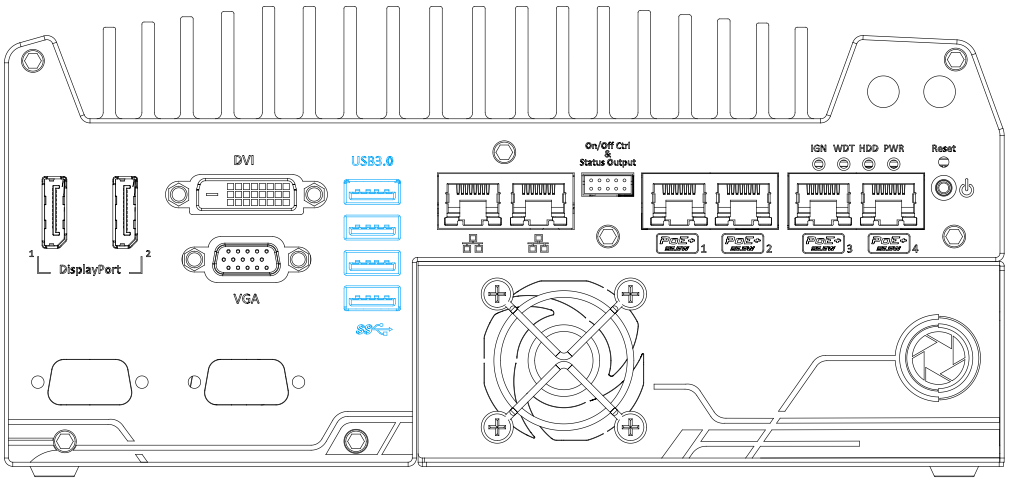
Triple Independent Display Configuration (resolution may be limited)

Active display 1	Active display 2	Active display 3
DisplayPort	DisplayPort	DVI or VGA
DisplayPort	DVI	VGA

NOTE

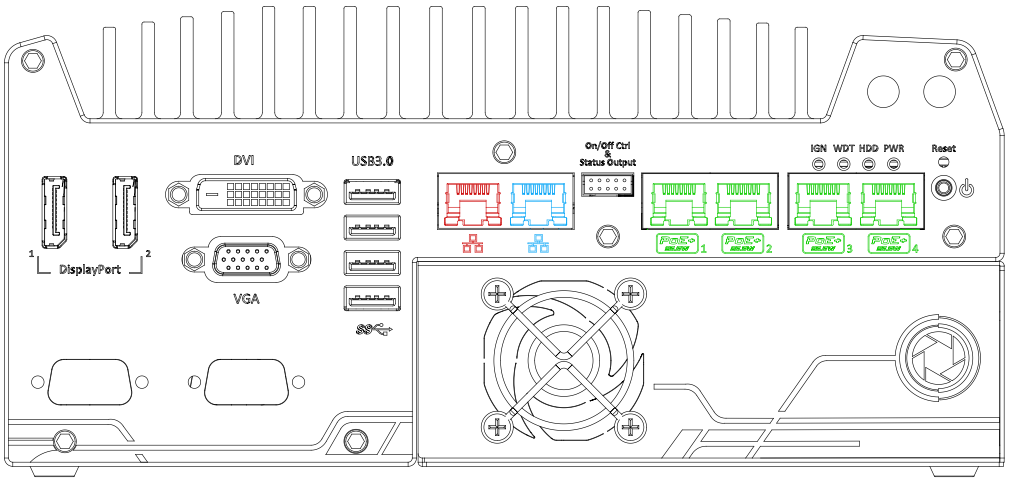
Please make sure your VGA cable includes SDA and SCL (DDC clock and data) signals for correct communication with monitor to get resolution/timing information. A cable without SDA/ SCL can cause blank screen on your VGA monitor due to incorrect resolution/timing output.

2.2.4 USB3.0 Port



The system offers four USB 3.0 (SuperSpeed USB) ports on its front panel. They are implemented via native xHCI (eXtensible Host Controller Interface) controller in Q170 chipset and are backward compatible with USB 2.0, USB 1.1 and USB 1.0 devices. Legacy USB support is also provided so you can use USB keyboard/mouse in DOS environment.

2.2.5 Ethernet Port / PoE



The system offer two GbE ports (in **red** and **blue**) and four additional PoE (Power over Ethernet) ports marked in **green**. The port marked in **blue** is implemented using Intel® I219-LM controller that supports Wake-on-LAN and is also compatible with Intel® AMT (Active Management Technology) to support advanced features such as remote SOL desktop and on/off.

Power over Ethernet (PoE) supplies electrical power and data on a standard CAT-5/CAT-6 Ethernet cable. Acting as a PoE PSE (Power Sourcing Equipment), compliant with IEEE 802.3at, each PoE port delivers up to 25W to a Powered Device (PD). PoE can automatically detect and determine if the connected device requires power or not, so it is compatible with traditional Ethernet devices as well.

Each port has one dedicated PCI Express link for maximum network performance. Please refer to the table below for LED connection statuses.

Active/Link LED (Right)

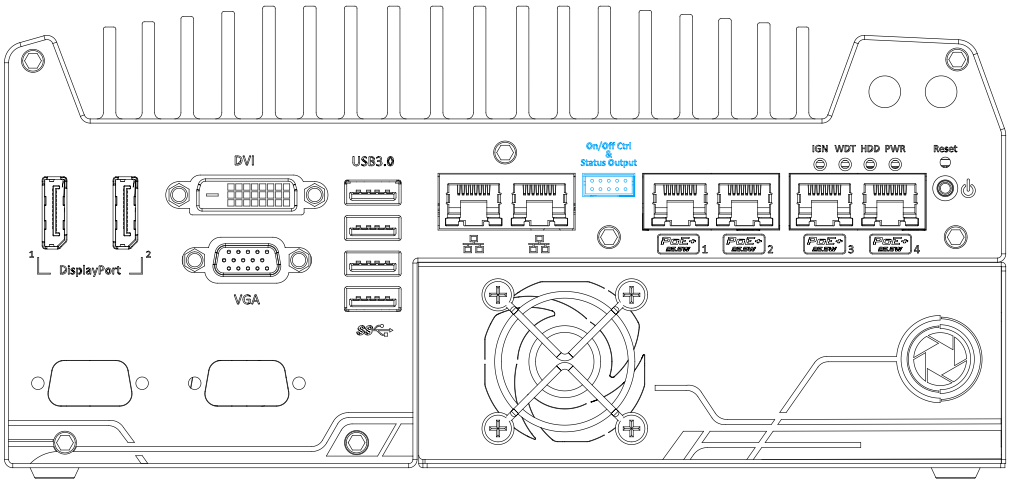
LED Color	Status	Description
Yellow	Off	Ethernet port is disconnected
	On	Ethernet port is connected and no data transmission
	Flashing	Ethernet port is connected and data is transmitting/receiving

Speed LED (Left)

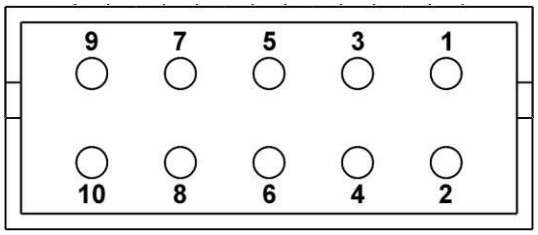
LED Color	Status	Description
Green or Orange	Off	10 Mbps
	Green	100 Mbps
	Orange	1000 Mbps

To utilize the GbE port in Windows, you need to install corresponding driver for Intel® I210-IT/ I219-LM GbE controller.

2.2.6 On/ Off Ctrl & Status Output



The “On/ Off Control Ctrl & Status Output” connection allows for external switch and LED indicator extension. It is useful when the system is placed in a cabinet or a not easily accessed location. This function is provided via a 2x5 2.0mm pitch wafer connector.

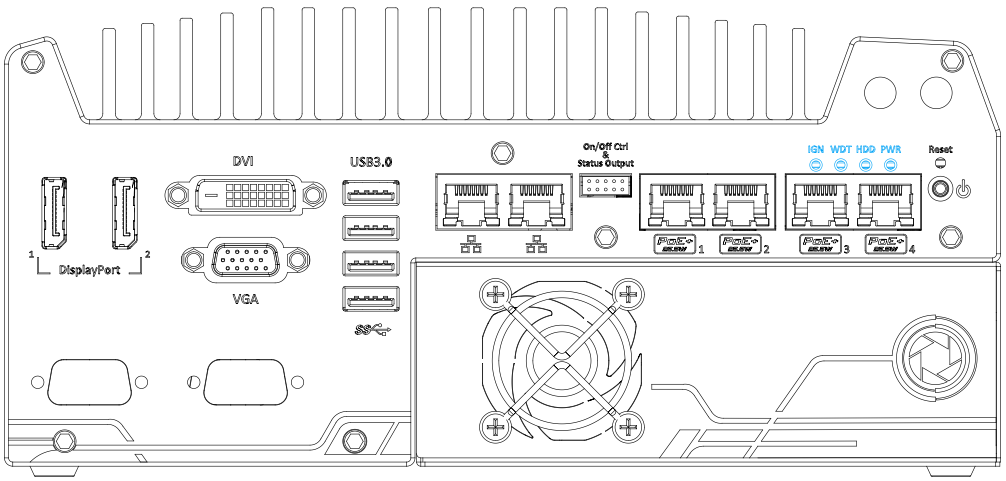


Pin#	Definition	Description
1	Ctrl+	**[Input]** Remote on/off control, connecting to an external switch to turn on/off the system (polarity is negligible).
2	Ctrl-	
3	Power+	**[Output]** System power indicator, on if system is turned on, off if system is turned off.
4	Power-	
5	HDD+	**[Output]** Hard drive indicator, flashing when SATA hard drive is active.
6	HDD-	
7	Standby Power+	**[Output]** Standby power indicator, lighting up when DC power is applied and system is in S5 (standby) mode.
8	Standby Power-	
9	WDT+	**[Output]** Watchdog timer indicator, flashing when watchdog timer is started.
10	WDT-	

NOTE

Please make sure the polarity is correct when you connect the external LED indicator to the Status LED Output.

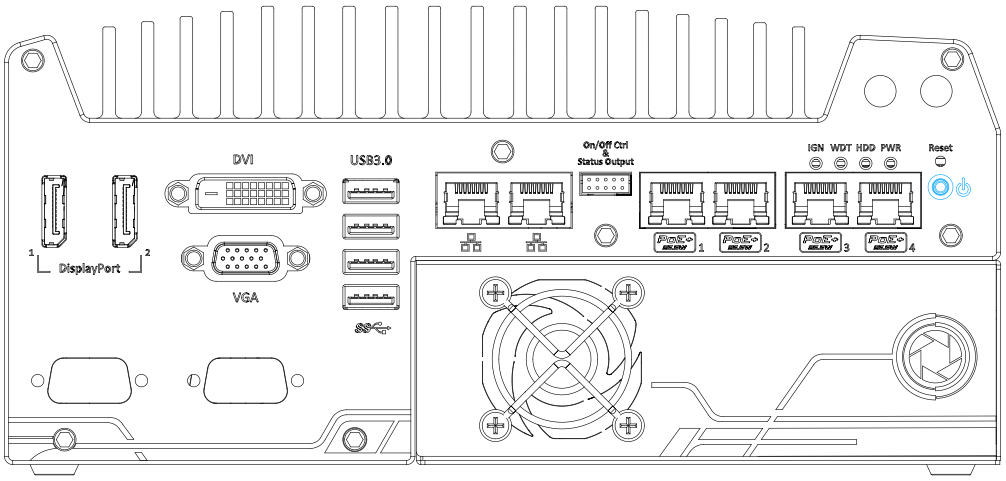
2.2.7 System Status LED



There are four LED indicators on the front panel: PWR, HDD, WDT and IGN. The descriptions of these four LEDs are listed in the following table.

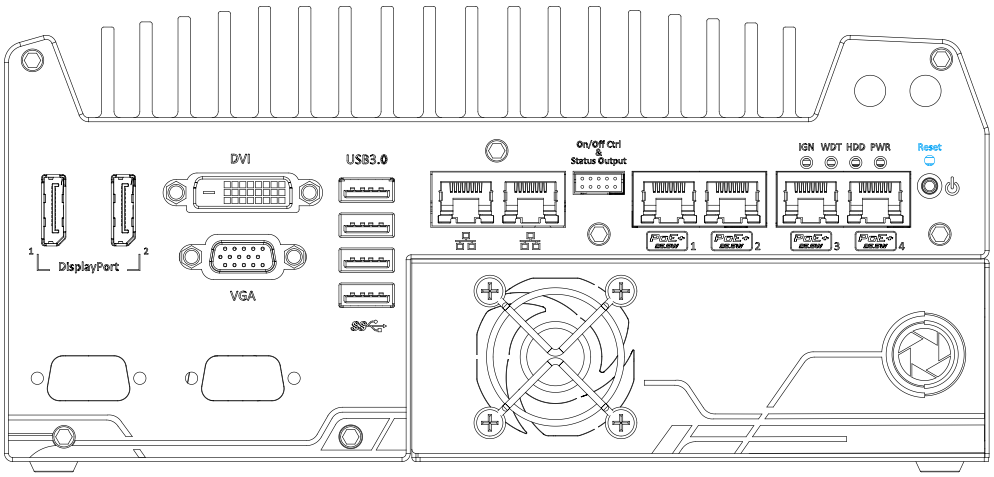
Indicator	Color	Description
IGN	Green	Ignition signal indicator, lid when IGN is high (12V/ 24V).
WDT	Yellow	Watchdog timer indicator, flashing when WDT is active
HDD	Red	Hard drive indicator, flashing when SATA drive is active
PWR	Green	Power indictor, lid when system is on

2.2.8 Power Button



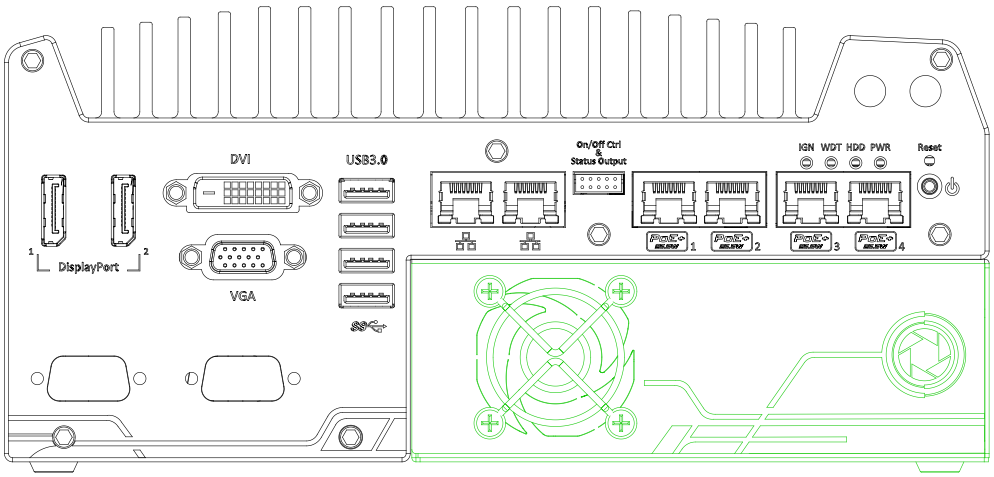
The power button is a non-latched switch for ATX mode on/off operation. To turn on the system, press the power button and the PWR LED should light-up green. To turn off the system, issuing a shutdown command in OS is preferred, or you can simply press the power button. To force shutdown when the system freezes, press and hold the power button for 5 seconds. Please note that there is a 5-second interval between on/off operations (i.e. once turning off the system, there is a 5-second wait before you can power-on the system).

2.2.9 Reset Button



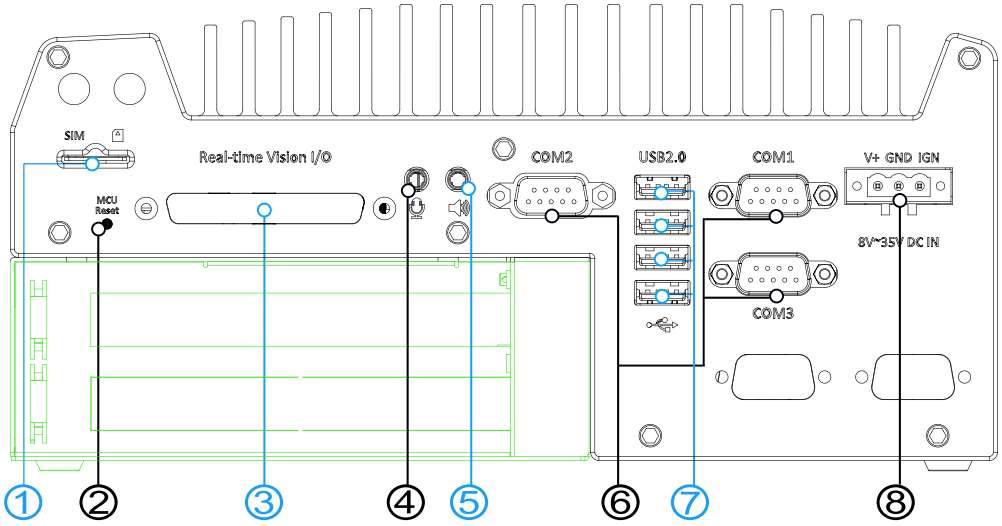
The reset button is used to manually reset the system in case of system halt or malfunction. To avoid unexpected resets, the button is purposely placed behind the panel. To reset, please use a pin-like object (eg. tip of a pen) to access the reset button.

2.2.10 Cassette Enclosure



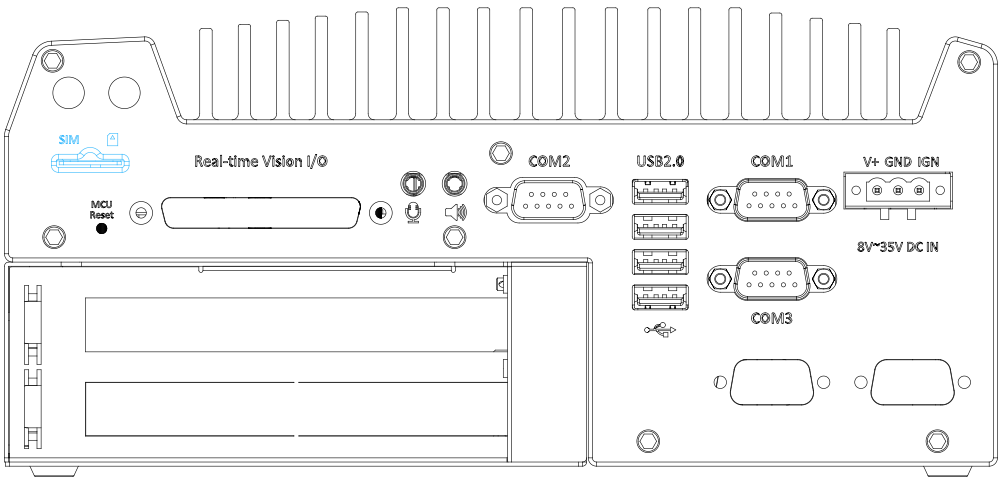
Neousys' patented expansion Cassette (R.O.C. Patent No. M456527) is an innovation design for fanless controller. It provides a separated compartment to accommodate an add-on card. It effectively manages thermal conditions of both the system and the add-on card. The modular concept brought by Cassette also reduces the complexity of installing and replacing an add-on card in the fanless controller. The Cassette enclosure itself incorporates an innovative mechanical design to effectively deal with the heat generated by GPU. This patented architecture (R.O.C. Patent No. M534371) creates a sealed wind tunnel to bring in cold air to the GPU and expels hot air via a system fan. The design offers the system extreme stability and reliability when operating at 60°C with the GPU under 100% load. The expansion Cassette enclosure accepts dual-slot graphics cards with up to 75W TDP.

2.3 Nuvis-5306RT Rear Panel



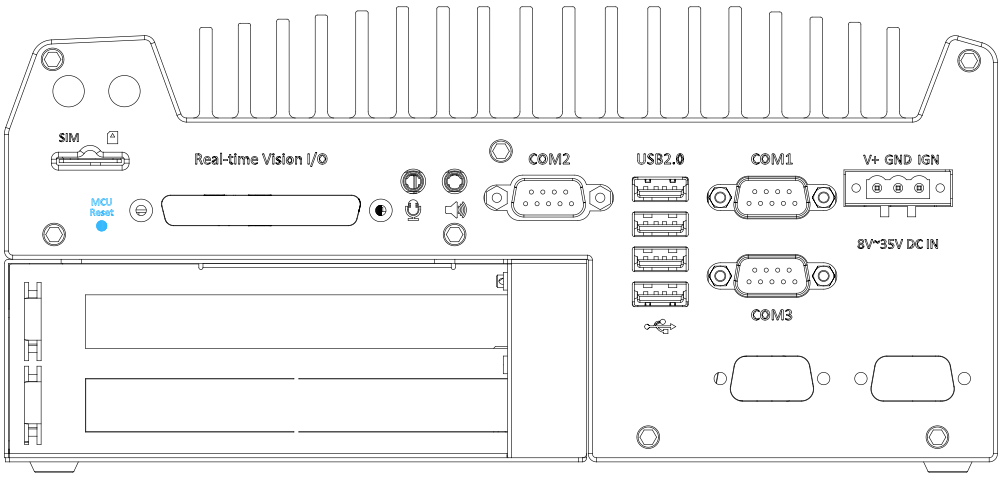
No.	Item	Description
1	SIM card slot	With a 3G/ 4G module installed, insert a SIM card to access the operator's network.
2	MCU reset button	Press the button to reset the MCU.
3	Real-time Vision I/O	Vision specific trigger/ strobe control input/ output for vision/ imagery purposes.
4	Microphone-in jack	Microphone-in jack for voice (microphone) input.
5	Speaker-out jack	Speaker-out jack for sound output.
6	COM ports	There are 3 COM ports for communicating with external devices.
7	USB 2.0	The USB 2.0 ports are compatible with USB 1.1 / 1.0.
8	3-pin terminal block (DC/ ignition input)	Compatible with DC power input from 8~35V, the terminal block is also used for ignition signal input.
9	**Area in green**	The panel opening of the cassette enclosure. When an expansion card is installed, connectors are accessible on this panel.

2.3.1 SIM Card Slot



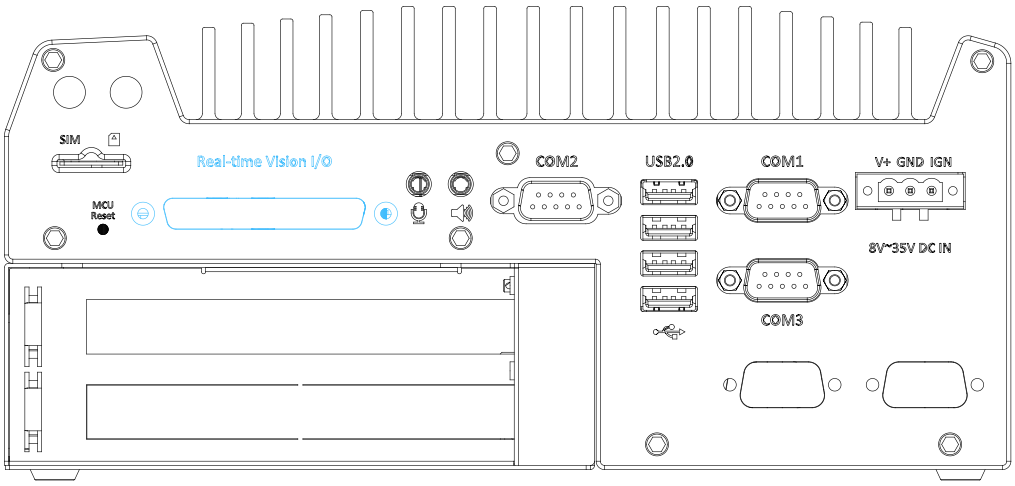
On the rear panel, there is a panel-accessible SIM socket. By installing a 3G/4G module onto the internal mini-PCIe port, you can have Internet access via telecom operator's network. The SIM socket is a push-push type. The push-push mechanism means the SIM card is push-to-install and push-to-retrieve. Please note that the SIM card must be inserted upside down (gold fingers facing upward).

2.3.2 MCU Reset



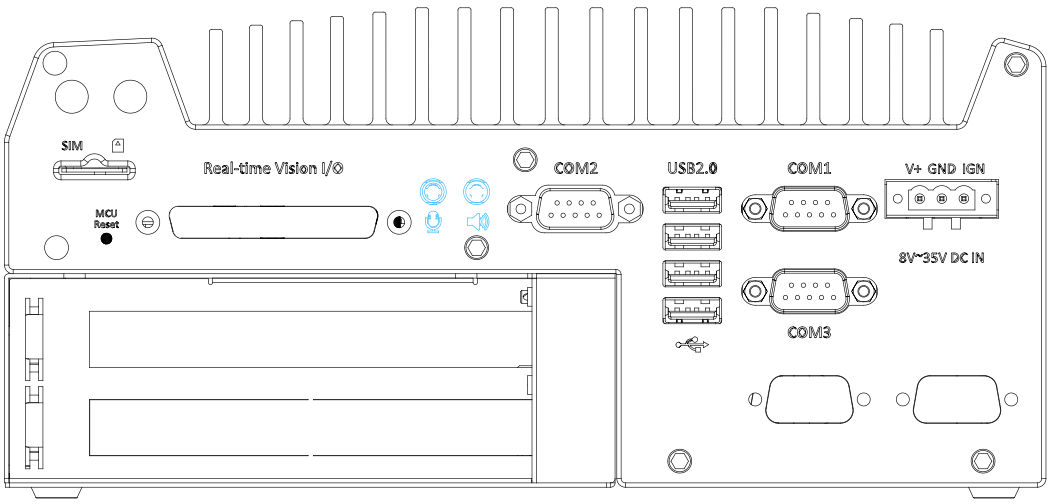
You may use the MCU reset button to manually reset the MCU without resetting the whole system. To avoid unexpected resets, the button is purposely placed behind the panel. To reset, please use a pin-like object (eg. tip of a pen) to access the reset button.

2.3.3 Real-time Vision I/O



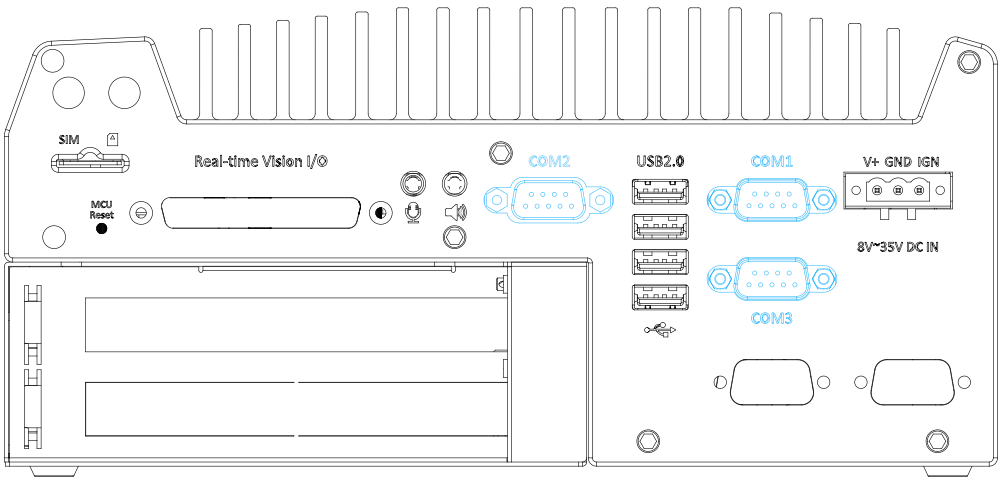
Real-time vision I/O is managed by Neousys' patented MCU-based architecture and DTIO/ NuMCU firmware for microsecond-scale real-time I/O control. It also supports various machine vision peripherals such as CC/ CV lighting controller, quadrature encoder input, PWM output, isolated DI/ DO, 12V camera trigger output etc. Please refer to the section Nuvis-5306RT Vision-Specific I/O.

2.3.4 3.5mm Speaker-out/ Microphone-in Jack



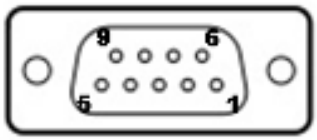
The audio function on Nuvis-5306RT uses Intel® High Definition Audio in Q170 chipset and Realtek ALC262 codec. There are two audio function jacks, the port is used for microphone input, and the port is used for speaker / headphone output. To utilize the audio function in Windows, you need to install corresponding drivers for both Intel® Q170 chipset and Realtek ALC262 codec.

2.3.5 COM Ports



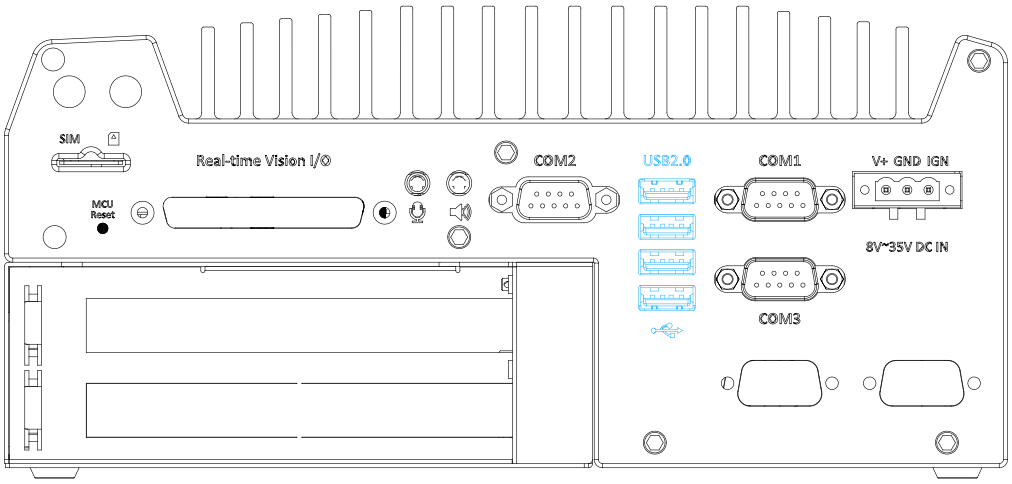
The system has three COM ports for communicating with external devices. COM1, COM2 and COM3 ports are located on the rear panel via 9-pin D-Sub male connectors. They are implemented using industrial-grade ITE8786 Super IO chip (-40 to 85°C) and provide up to 115200 bps baud rate.

COM1 and COM3 are software-configurable RS-232/422/485 ports and COM2 is a standard 9-wire RS-232 port. The operation mode, slew rate and termination of COM1 and COM3 can be set in BIOS setup utility. The following table describes the pin definition of COM ports.



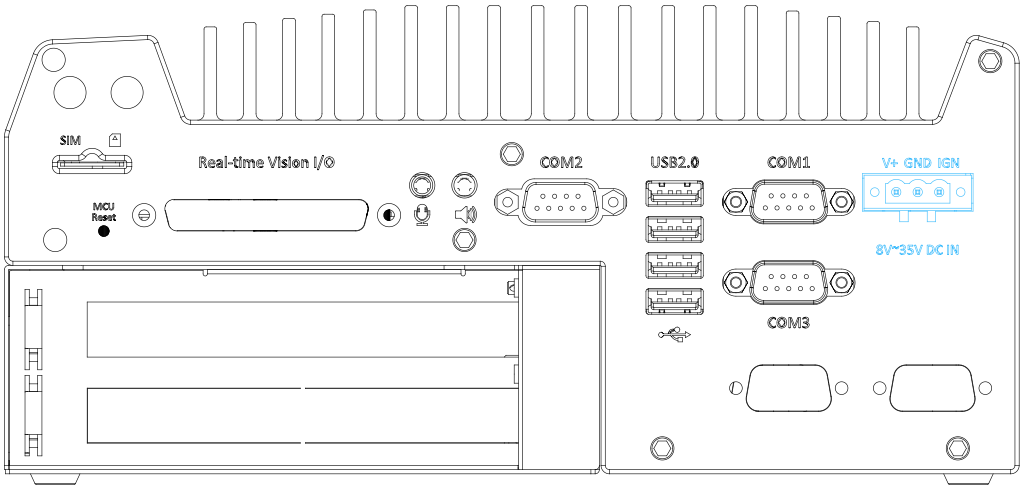
	COM1 / COM3			COM2
Pin#	**RS-232 Mode**	**RS-422 Mode**	**RS-485 Mode (Two-wire 485)**	**RS-232 Mode**
1	DCD	-	-	DCD
2	RX	422 TXD+	485 TXD+/RXD+	RX
3	TX	422 RXD+	-	TX
4	DTR	422 RXD-	-	DTR
5	GND	GND	GND	GND
6	DSR	-	-	DSR
7	RTS	-	-	RTS
8	CTS	422 TXD-	485 TXD-/RXD-	CTS
9	RI	-	-	RI

2.3.6 USB2.0 Ports



The USB2.0 ports are implemented via native xHCI (eXtensible Host Controller Interface) controller in Q170 chipset and are backward compatible with USB 1.1 and USB 1.0 devices. Legacy USB support is also provided so you can use USB keyboard/mouse in DOS environment.

2.3.7 3-Pin Terminal Block for DC and Ignition Input



The system allows an 8 to 35V DC power input from via a 3-pin pluggable terminal block. The screw clamping mechanism is a reliable way to wire DC power. In addition to DC power, this terminal block also accepts ignition signal input (IGN).

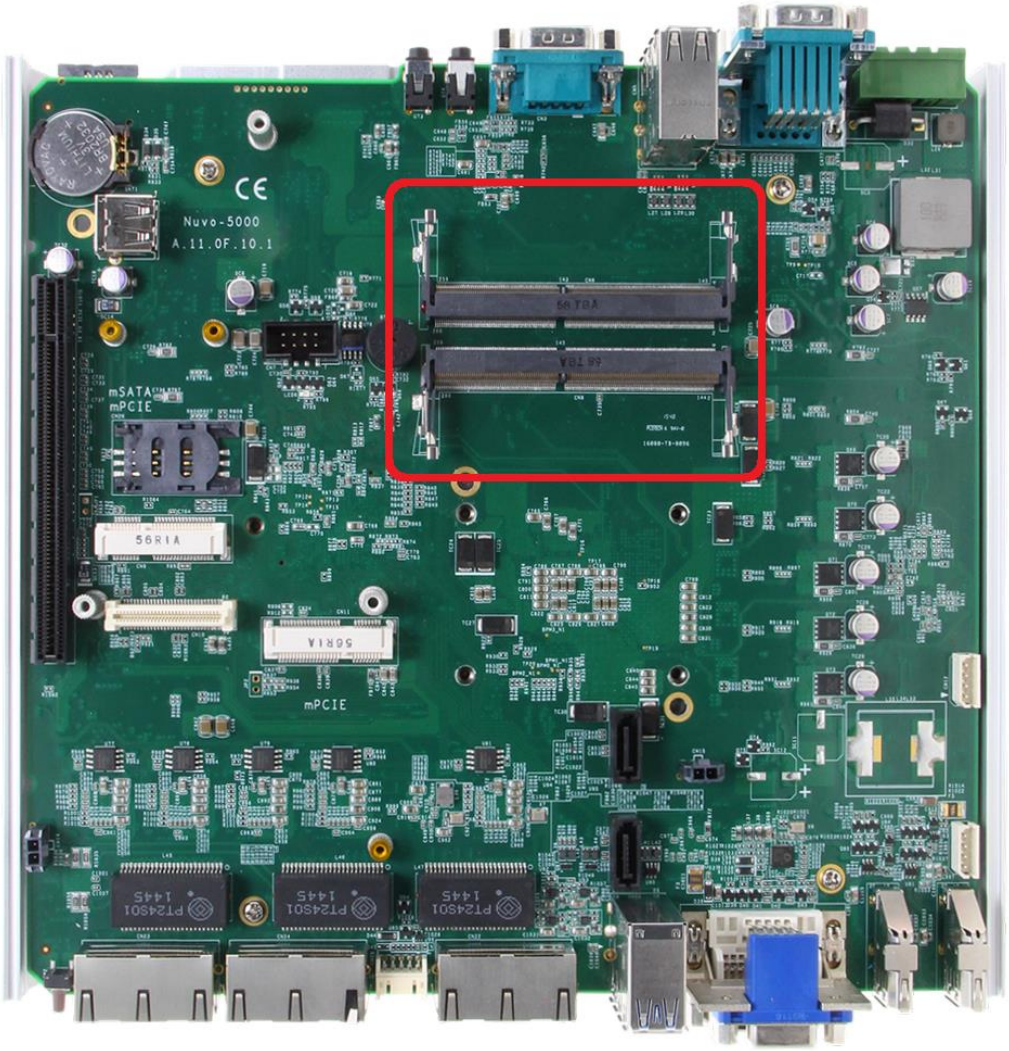
WARNING

Please make sure the voltage of DC power is correct before you connect it to the system. Supplying a voltage over 35V will damage the system.

2.4 Internal I/O Components

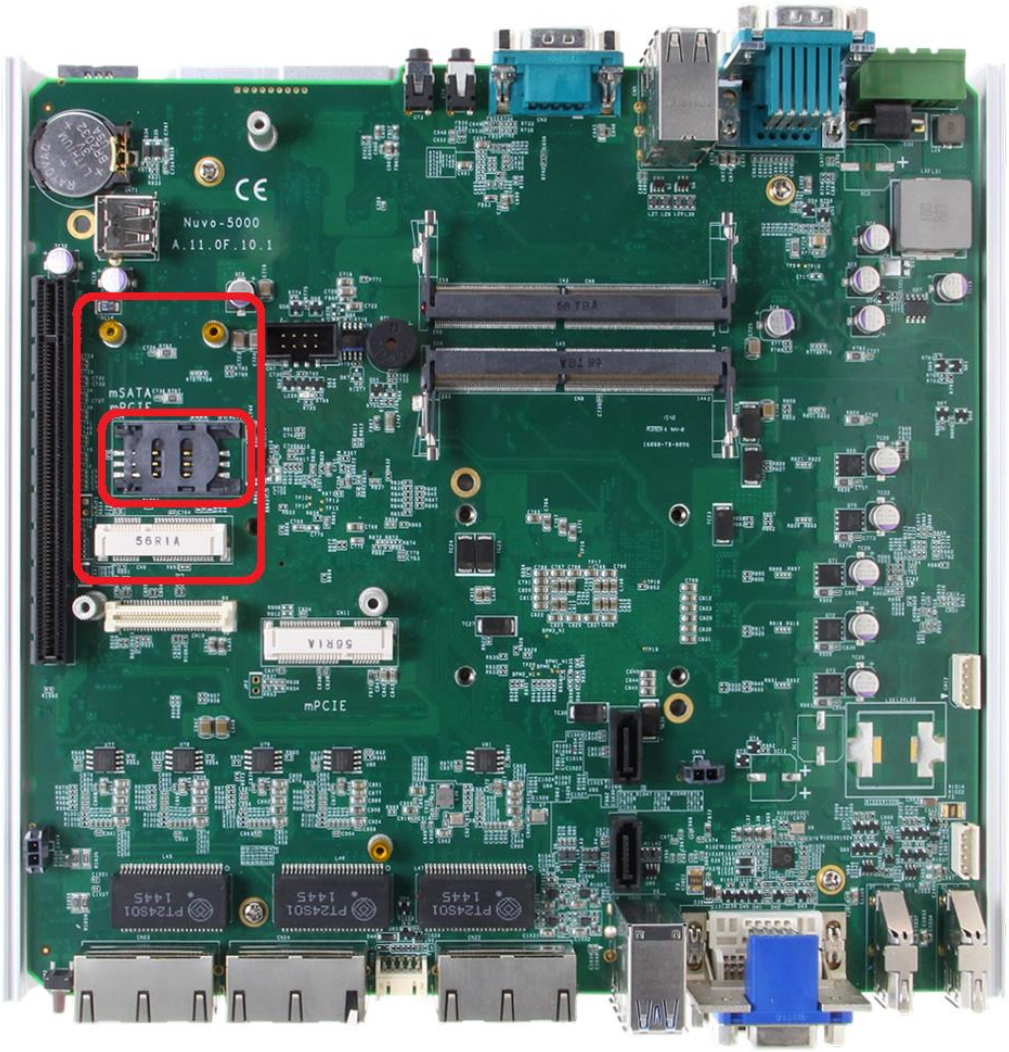
The internal components of Nuvis-5306RT series include two SODIMM sockets, SATA ports, mSATA, mini-PCIe sockets and an internal USB port.

2.4.1 DDR4 SO-DIMM Slots



The system provides two 260-pin DDR4 memory SO-DIMM sockets. It can support up to 32GB maximum capacity by installing two 16GB DDR4 2133 MHz SODIMM modules.

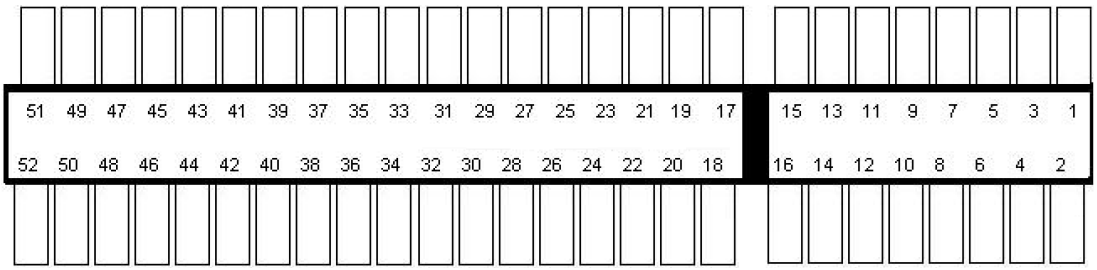
2.4.2 Dual Mode mSATA/ mini-PCIe socket



The system provides a dual mode mSATA/ mini-PCIe socket that is in compliance with mini-PCIe specification rev. 1.2. You can install either an mSATA SSD or mini-PCIe module into this socket and the system will automatically detect and configure it to run PCIe or SATA signals. This mini-PCIe socket is designed with SIM card support. With a SIM card installed, your system can access the internet via your network provider's 3G/ 4G network.

For wireless (WIFI/ 3G/ 4G) communication, multiple SMA antenna apertures can be located on the front and rear panel.

Dual mode mSATA/ mini-PCIe socket definition

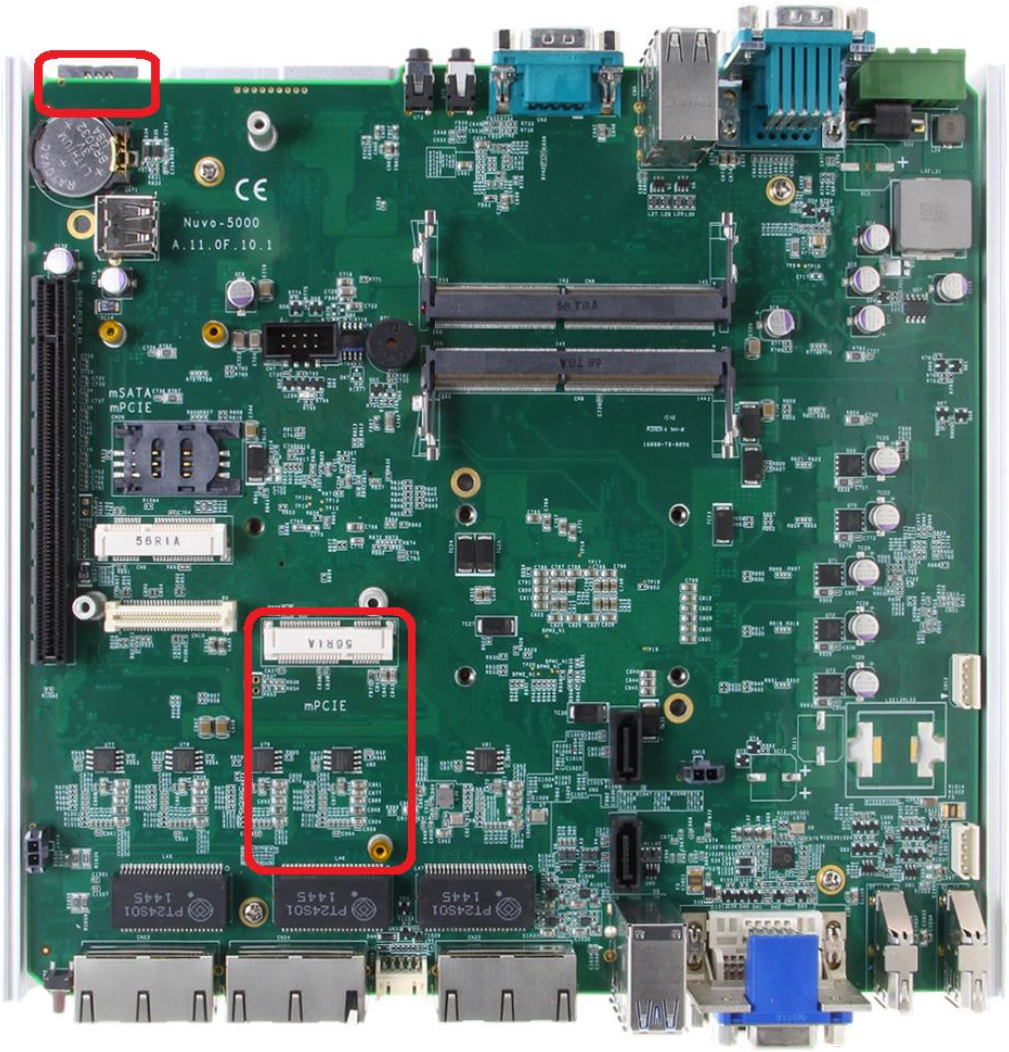


Pin	Signal (mPCIe)	Signal (mSATA)	Pin #	Signal (mPCIe)	Signal (mSATA)
1	WAKE#	-	2	+3.3Vaux	+3.3Vaux
3	COEX1	-	4	GND	GND
5	COEX2	-	6	+1.5V	+1.5V
7	CLKREQ#	-	8	UIM_PWR	-
9	GND	GND	10	UIM_DATA	-
11	REFCLK-	-	12	UIM_CLK	-
13	REFCLK+	-	14	UIM_RESET	-
15	GND	GND	16	UIM_VPP	-
Mechanical Key					
17	Reserved*	-	18	GND	GND
19	Reserved*	-	20	W_DISABLE#	-
21	GND	GND	22	PERST#	-
23	PERn0	SATA_Rxp	24	+3.3Vaux	+3.3Vaux
25	PERp0	SATA_Rxn	26	GND	GND
27	GND	GND	28	+1.5V	+1.5V
29	GND	GND	30	SMB_CLK	SMB_CLK
31	PETn0	SATA_Txn	32	SMB_DATA	SMB_DATA
33	PETp0	SATA_Txp	34	GND	GND
35	GND	GND	36	USB_D-	-
37	GND	GND	38	USB_D+	-
39	+3.3Vaux	+3.3Vaux	40	GND	GND
41	+3.3Vaux	+3.3Vaux	42	LED_WWAN#	-
43	GND	-	44	LED_WLAN#	-
45	Reserved	-	46	LED_WPAN#	-
47	Reserved	-	48	+1.5V	+1.5V
49	Reserved	-	50	GND	GND
51	Reserved	-	52	+3.3Vaux	+3.3Vaux

WARNING

Some off-the-shelf mini-PCIe 4G modules are not compliant to standard mini-PCIe interface. They use 1.8V I/O signals instead of standard 3.3V I/O and may have signal conflict. Please consult with Neousys for compatibility when in doubt! Installing an incompatible 4G module may damage the system or the module itself may be damaged.

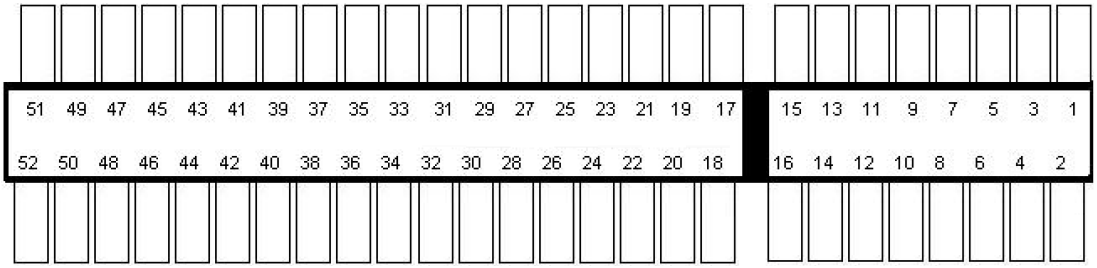
2.4.3 mini-PCIe Socket



This mini-PCIe socket works in cooperation with the panel-accessible SIM slot. By installing a mini-PCIe module, you can add additional features to your system such as WIFI, GPS, CAN bus, analog frame grabber, etc. You can also install a 3G/4G module and SIM card for internet via your service provider's 3G/ 4G network.

For wireless (WIFI/ 3G/ 4G) communication, multiple SMA antenna apertures can be located on the front and rear panel.

mini-PCIe Pin Definition

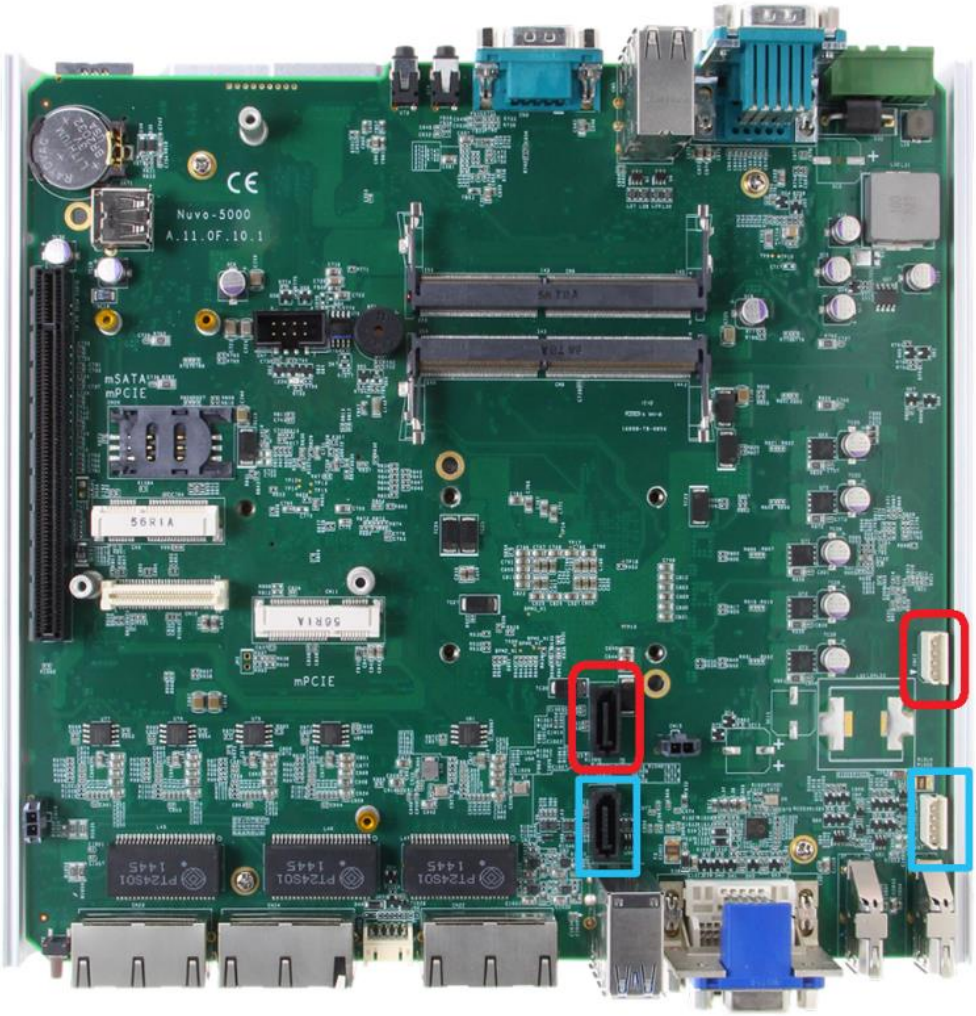


Pin #	Signal	Pin #	Signal
1	WAKE#	2	+3.3Vaux
3	COEX1	4	GND
5	COEX2	6	+1.5V
7	CLKREQ#	8	UIM_PWR
9	GND	10	UIM_DATA
11	REFCLK-	12	UIM_CLK
13	REFCLK+	14	UIM_RESET
15	GND	16	UIM_VPP
Mechanical Key			
17	Reserved* (UIM_C8)	18	GND
19	Reserved* (UIM_C4)	20	W_DISABLE#
21	GND	22	PERST#
23	PERn0	24	+3.3Vaux
25	PERp0	26	GND
27	GND	28	+1.5V
29	GND	30	SMB_CLK
31	PETn0	32	SMB_DATA
33	PETp0	34	GND
35	GND	36	USB_D-
37	GND	38	USB_D+
39	+3.3Vaux	40	GND
41	+3.3Vaux	42	LED_WWAN#
43	GND	44	LED_WLAN#
45	Reserved	46	LED_WPAN#
47	Reserved	48	+1.5V
49	Reserved	50	GND
51	Reserved	52	+3.3Vaux

WARNING

Some off-the-shelf mini-PCIe 4G modules are not compliant to standard mini-PCIe interface. They use 1.8V I/O signals instead of standard 3.3V I/O and may have signal conflict. Please consult with Neousys for compatibility when in doubt! Installing an incompatible 4G module may damage the system or the module itself may be damaged.

2.4.4 SATA Ports

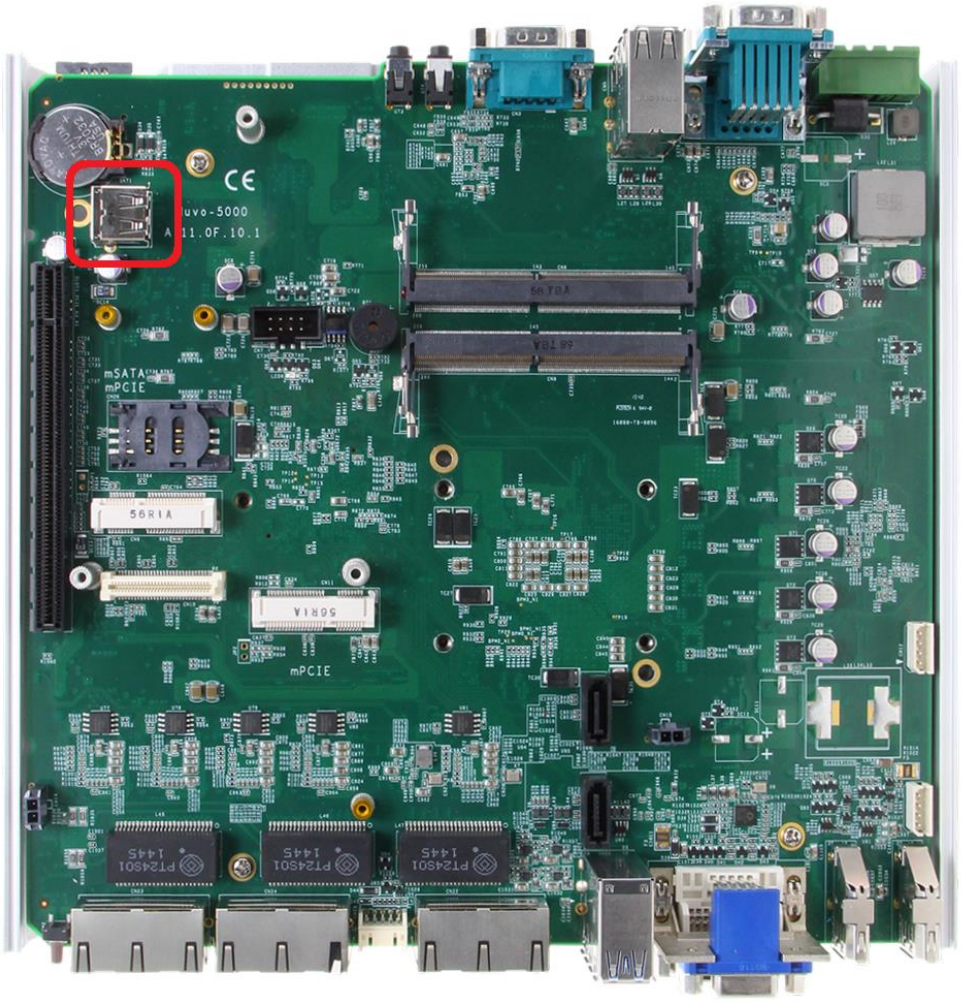


The system provides two SATA ports which support Gen3, 6 Gb/s SATA signals. Each SATA port features a 7-pin SATA connector and a 4-pin power connector.

The SATA port and power plug indicated in **red** is used in conjunction with the hot plug HDD tray to accommodate a 2.5” HDD/SSD. Standard 22-pin SATA connectors are provided with the system. You may enable the Hot Plug function in the SATA Configuration section.

The SATA port and power plug indicated in **blue** accommodates a 2.5” HDD/ SSD in internal HDD bracket. Standard 22-pin SATA connectors are provided with the system. You may refer to the SATA Configuration section for SATA settings.

2.4.5 Internal USB Port



The system has an internal USB2.0 port on the PCBA. You can utilize this USB port to connect a USB protection dongle inside the chassis of the system.

3 System Installation

Before disassembling the system enclosure and installing components and modules, please make sure you have done the following:

- It is recommended that only qualified service personnel should install and service this product to avoid injury or damage to the system.
- Please observe all ESD procedures at all times to avoid damaging the equipment.
- Before disassembling your system, please make sure the system has powered off, all cables and antennae (power, video, data, etc.) are disconnected.
- Place the system on a flat and sturdy surface (remove from mounts or out of server cabinets) before proceeding with the installation/ replacement procedure.

3.1 Disassembling the System Enclosure

1. Turn the system upside-down.

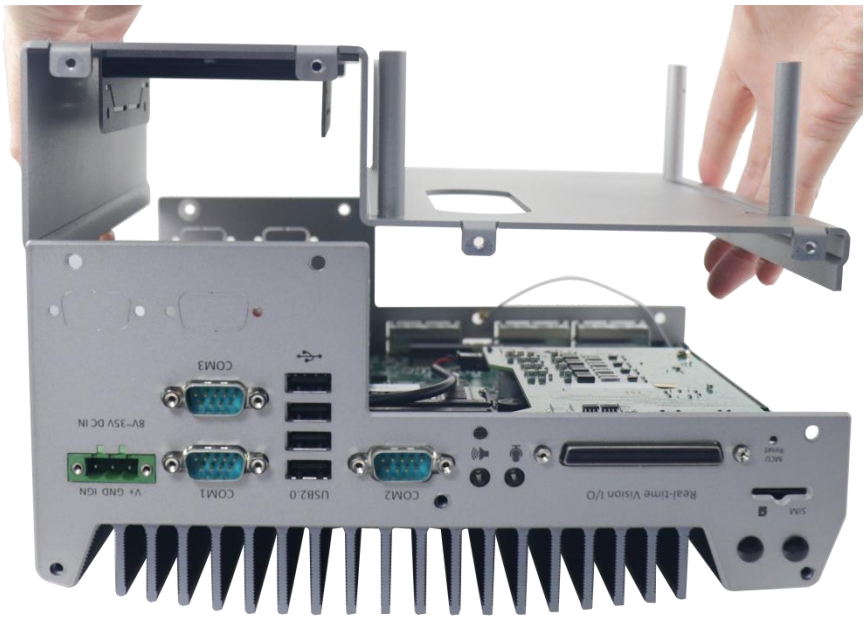
2. Unscrew the four (4) screws indicated at the bottom of the Cassette enclosure.

3. Gently lift and separate the Cassette enclosure from the system enclosure.

4. Remove the seven screws on the front panel shown in the illustration below.

5. Remove the seven screws on the rear panel shown in the illustration below.

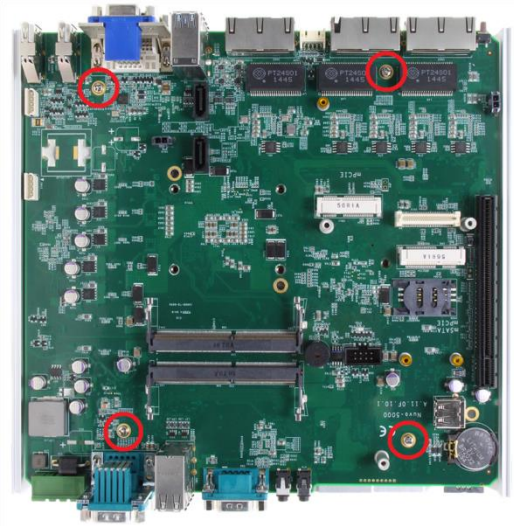
6. Gently lift the base panel and remove front and rear panels



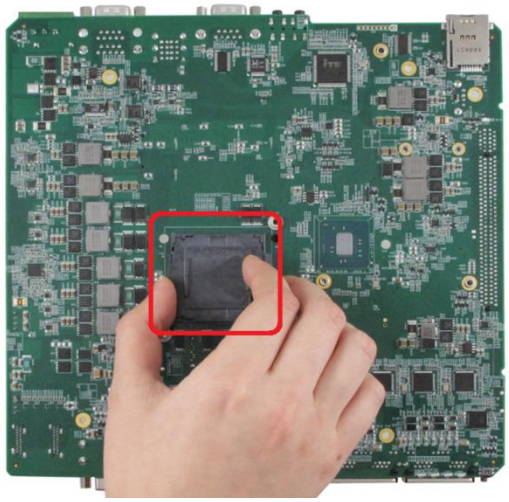
3.2 CPU Installation Procedure

1. DO NOT remove the CPU from its container / tray before it is ready to be installed.

2. Once the enclosure panels have been removed and to install a CPU into the system, remove the four (4) M3 P-head screws in the illustration below.



3. Gently lift the motherboard off the heatsink and turn the motherboard upside-down. You'll see the CPU socket protective cover, place finger tips under the sign "REMOVE" for leverage. Gently lift the cover.

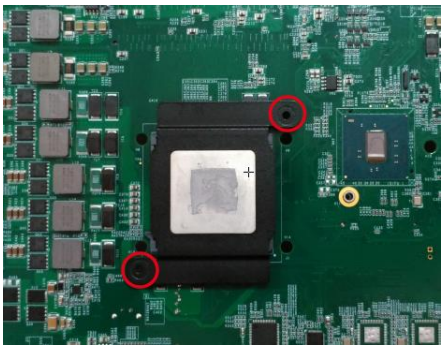


WARNING

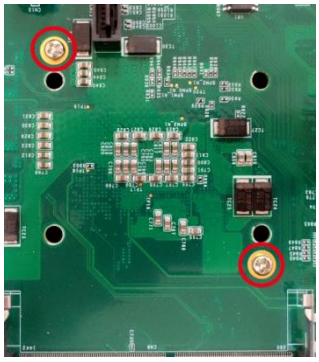
With the protective cover removed, please be careful when handling the motherboard. DO NOT touch the pins in the LGA socket!

4. Remove the CPU from its container/ tray. Match the two notches on the side to the protrusions in the socket, gently lower the CPU into the socket.

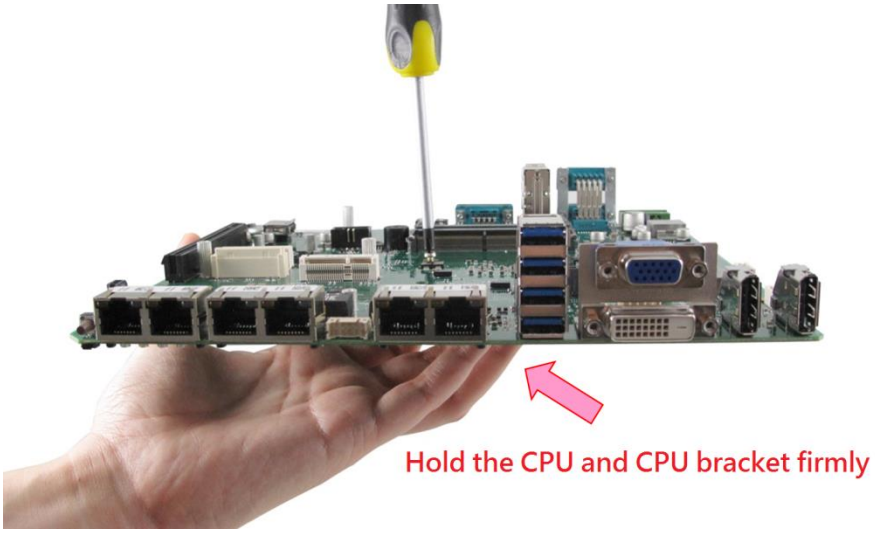
5. Locate the CPU retention bracket in the accessory box. Place the retention bracket on the CPU and hold it in place, turn the motherboard around and secure the bracket by tightening two (2) M3 P-head screws.



Retention bracket screw location

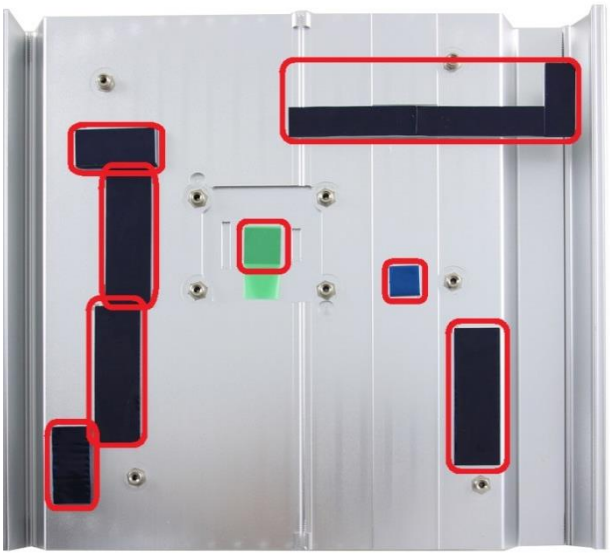


Secure two M3 P-head screws on the opposite side



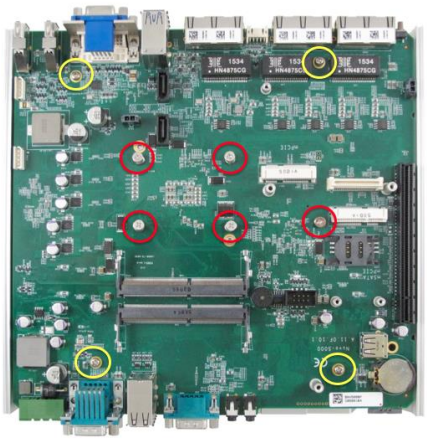
Hold bracket/ CPU in place along with the motherboard

6. Remove all thermal pad protective films pre-placed on the heatsink.

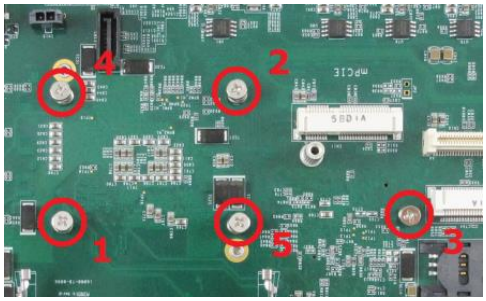


7. With the four (4) motherboard standoffs aligned (please refer to step 2), gently lower the motherboard onto the heatsink

8. Secure the four (4) M3 P-head motherboard screws (indicated in **yellow**) and from the accessory box, five (5) M3 spring screws (indicated in **red**). Gradually tighten the five screws in the following order for even pressure.



Securing the motherboard



Secure five CPU/ heatsink spring screws in order

9. Reinstall the system enclosure and panel when done.

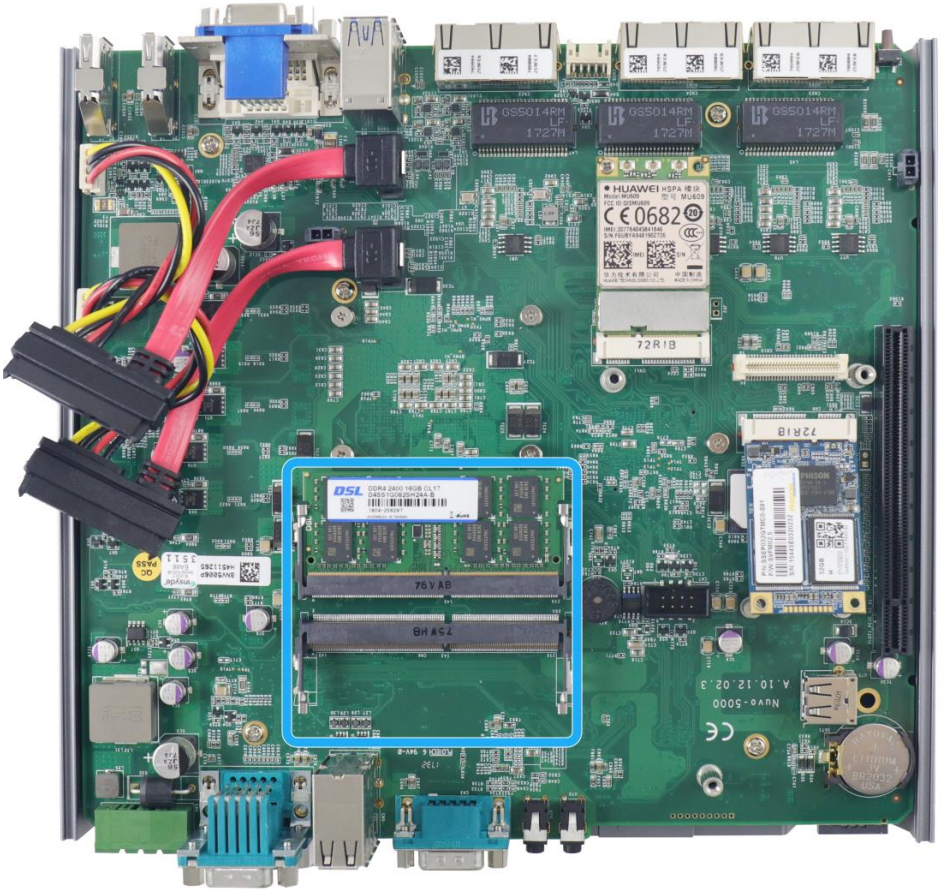
10. If you need to install other components, please refer to respective sections.

3.3 Memory Module Installation

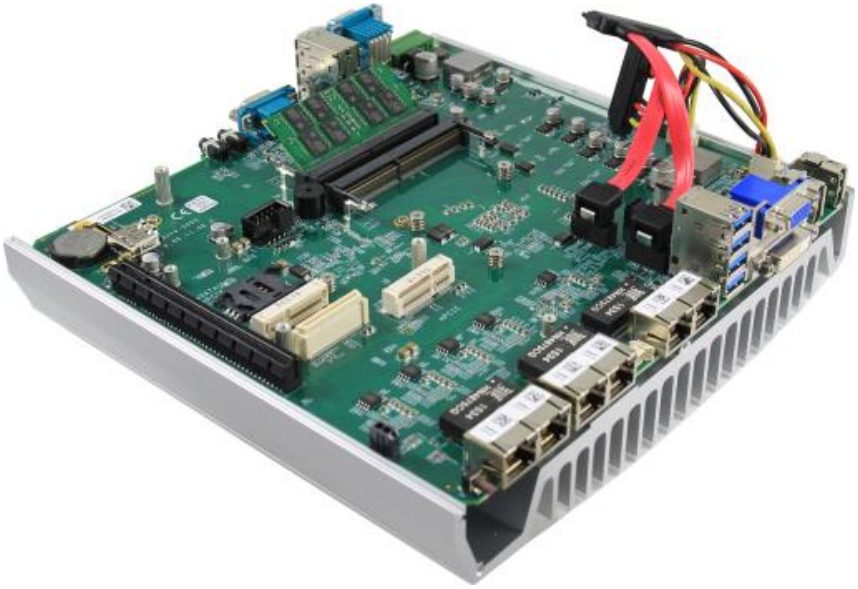
There are two memory SO-DIMM slots on the motherboard that support a total maximum of 32GB DDR4-2133. Please follow the procedures below to replace or install the memory modules.

1. To disassemble the enclosure, please refer to the section "Disassembling the System Enclosure".

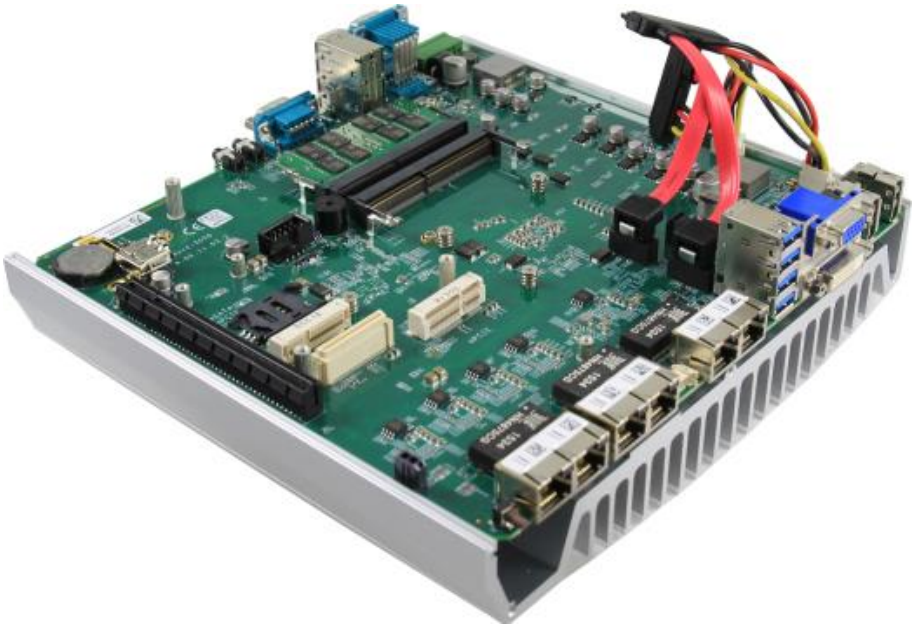
2. The SO-DIMM slots can be located once the bottom cover of the enclosure has been removed.



3. To install the memory module, insert gold fingers of the module into the slot at 45 degree angle.



4. Push down on the edge of the module and the clips on the side should clip the module into place.



5. Repeat steps above to install the other module.

6. Reinstall the system enclosure and panel when done.

7. If you need to install other components, please refer to respective sections.

3.4 Internal 2.5” SATA HDD/ SSD Installation

The system has two SATA ports for connecting SATA HDD/ SSD, one internal and one external. Please follow the procedures below to install or replace the hard drives.

1. Place the system upside down on a flat surface and loosen the three (3) screws show below.

2. Remove the protective films on the thermal pad situated in the center of the bracket.

3. Place the HDD/ SSD on the bracket, gently press it down against the thermal pad. From the accessory box, use M3 flat-head screws to secure the HDD/ SSD. Make sure the HDD/ SSD is secured in the same orientation as shown in the illustration below.

4. Connect the SATA cable inside the enclosure to the HDD/ SSD

5. Gently wiggle the bracket back into the enclosure and secure it with three (3) M3 flat-head screws.

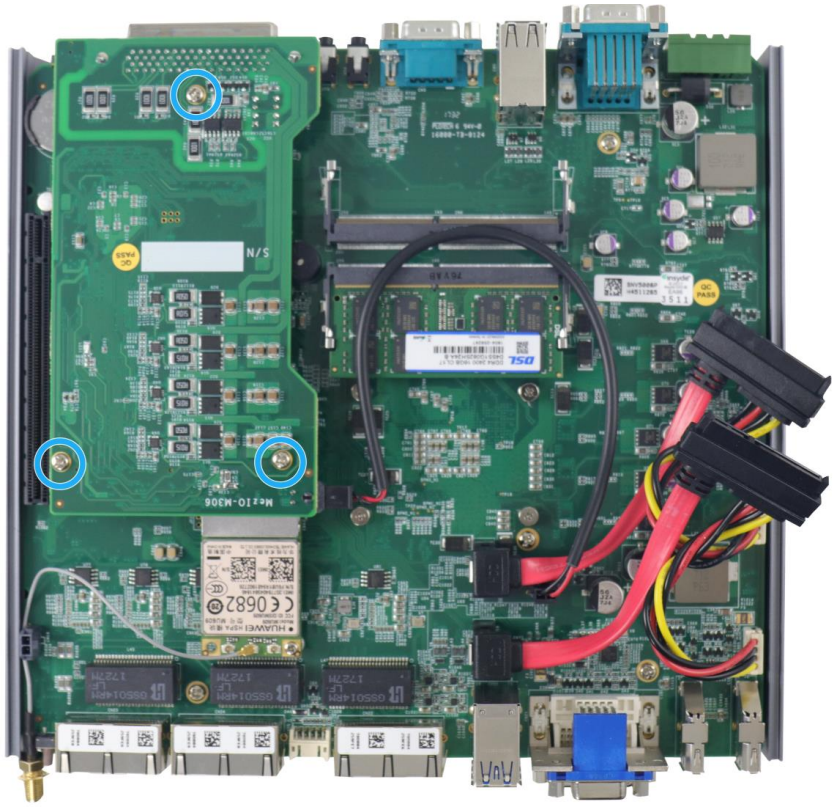
3.5 mini-PCIe Module Installation

There are two full size mini-PCIe sockets with SIM card support on the PCBA and another two on the MezIO™ module. It supports off-the-shelf mini-PCIe modules. Please refer to the following procedures on how to install a mini-PCIe module.

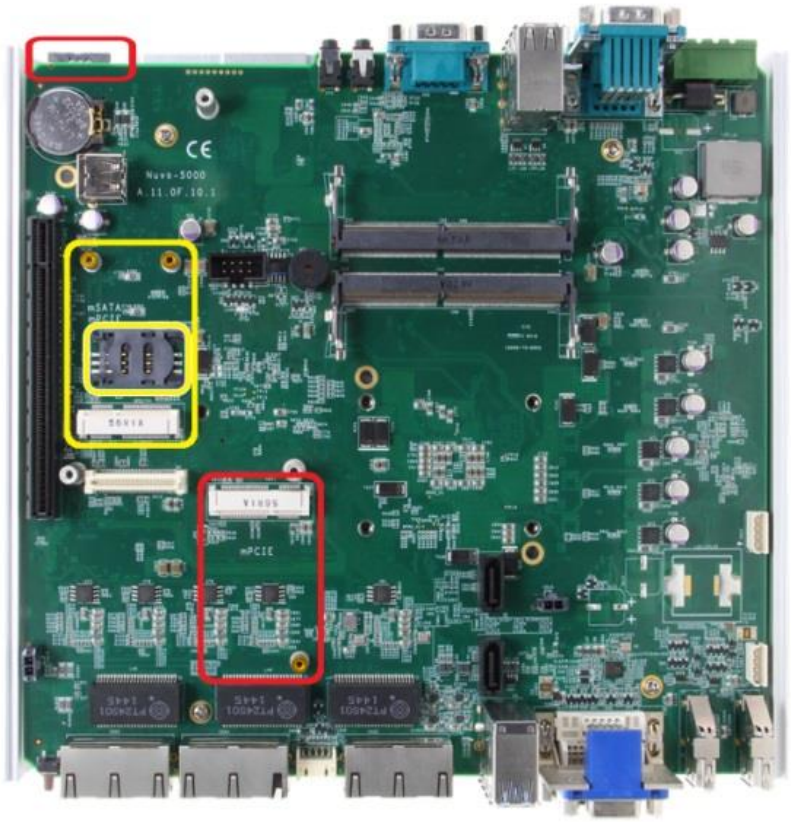
> **WARNING**
>
> *It is recommended to install 4G SIM mini-PCIe modules onto the MezIO™ module's mini-PCIe socket to avoid signal conflict issues.*

1. To disassemble the enclosure, please refer to the section "Disassembling the System Enclosure".

2. Remove the MezIO module by unfastening the three (3) screws shown below.



3. Location of the mini-PCIe sockets on the motherboard shown below. The SIM slot of mini-PCIe socket (in **red**) is situated on the rear panel.

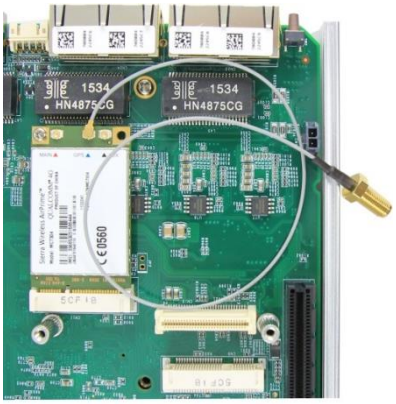


4. Inset the mini-PCIe module's gold finger on a 45 degree angle into the socket, gently press the module down and secure it with an M2.5 P-head screw.

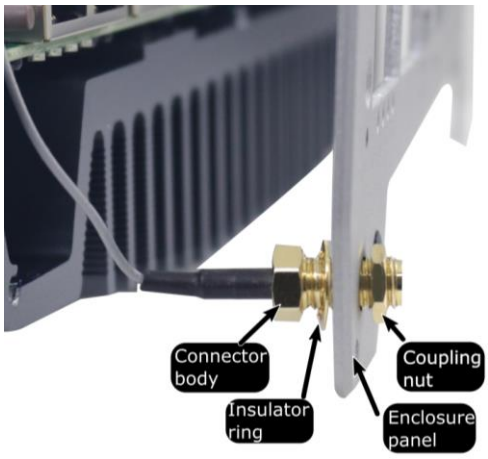
45 degree insertion angle

Secured with M2.5 P-head screw

5. Clip on the IPEX-to-SMA cable to the module and attach the antenna onto the front or rear panel.

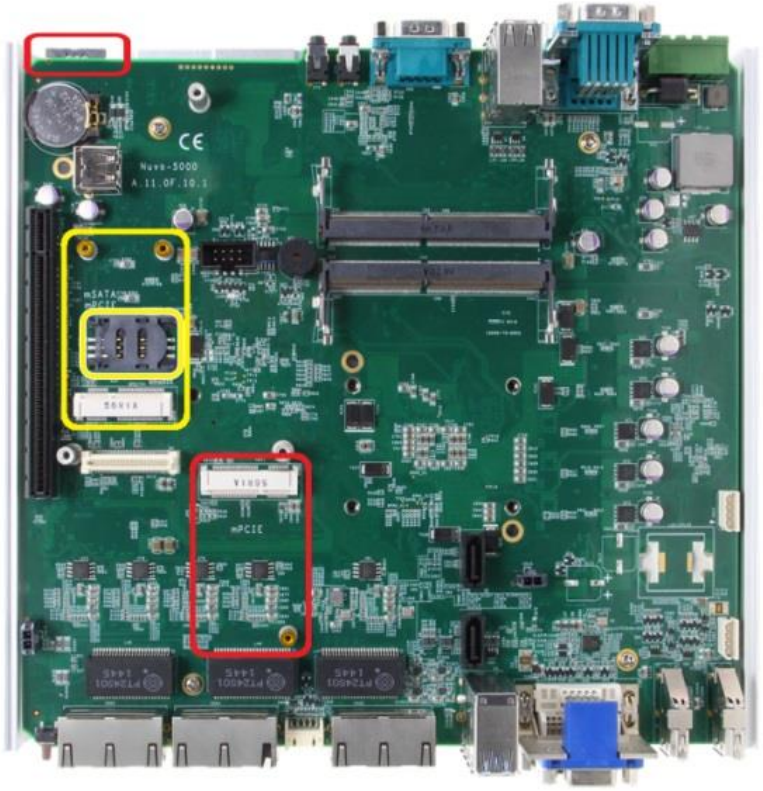


Clip on IPEX-to-SMA cable



Attach antenna to panel

6. Insert the SIM card (if necessary) for your mini-PCIe module.

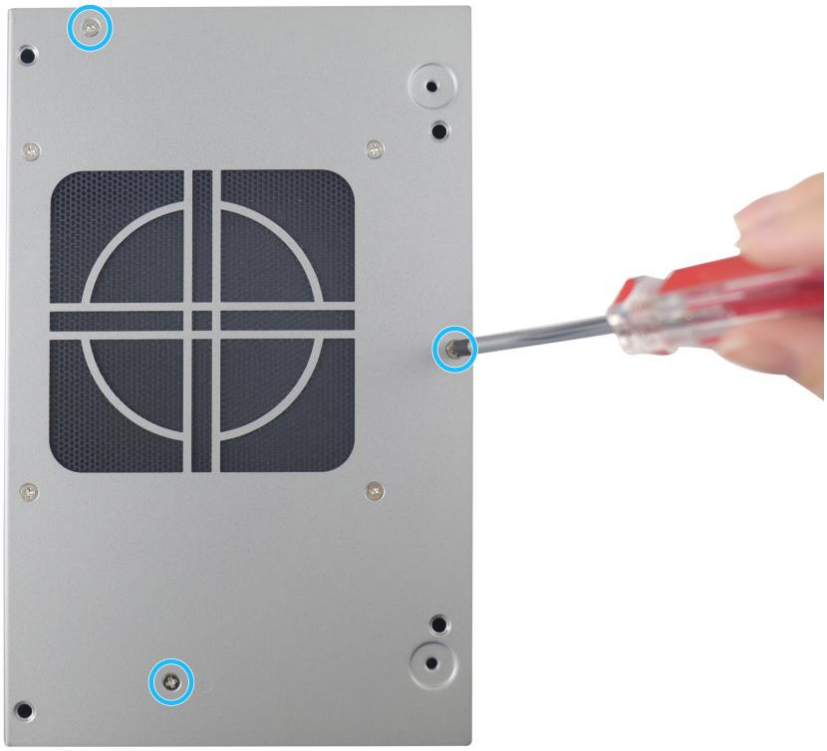


7. The mini-PCIe socket (in **red**) works in conjunction with the panel-accessible SIM slot on the rear panel, while other mini-PCIe sockets work with internal SIM slots. Insert the SIM upside down for 3G/ 4G access via your provider's network. Push the SIM card into the panel, make sure it clicks into the slot and is seated firmly (not protruding) in the slot.
8. Reinstall the system enclosure and panel when done.
9. If you need to install other components, please refer to respective sections.

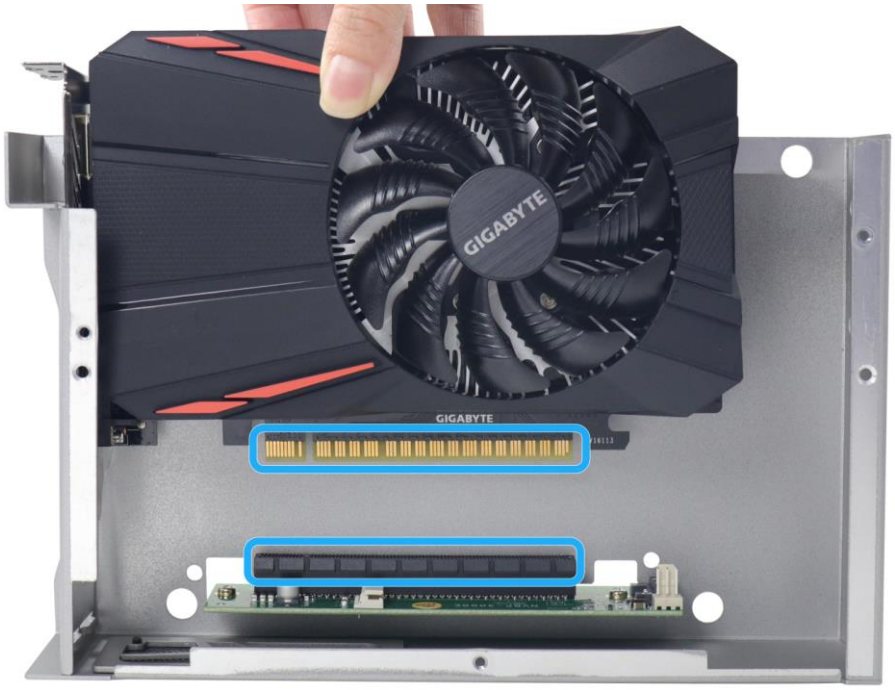
3.6 PCIe Card Installation in Cassette Module

For demonstration, we will be using a NVIDIA Geforce Ti1050 as an example.

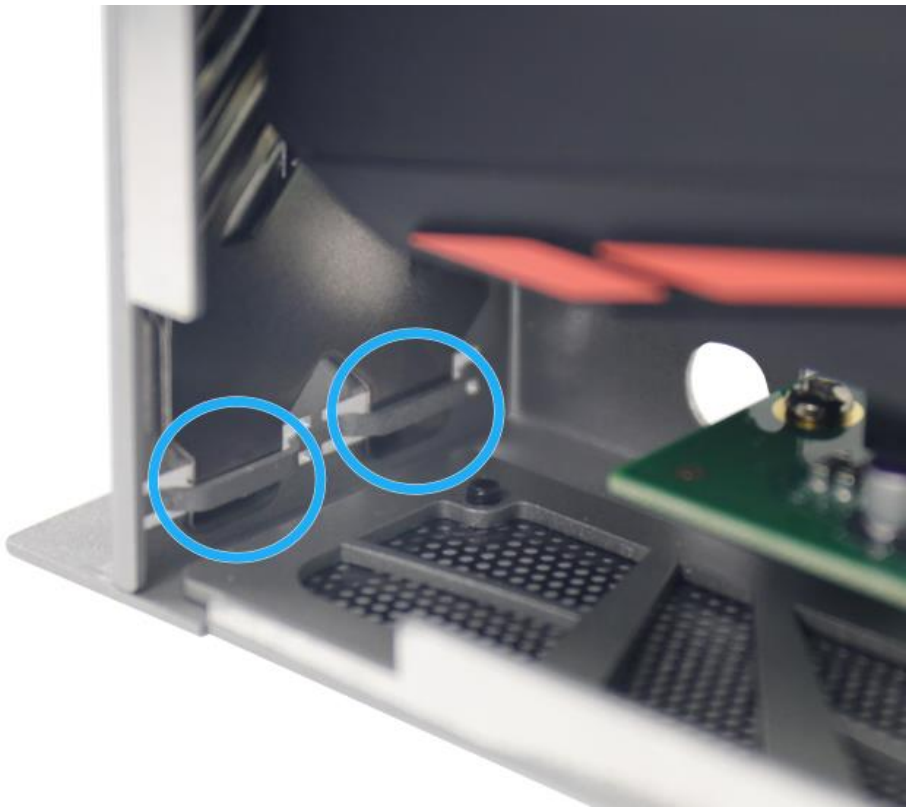
1. Unfasten the three screws at the bottom of the Cassette module (please refer to Disassembling the System Enclosure on Cassette module removal) and remove Cassette module's bottom panel.



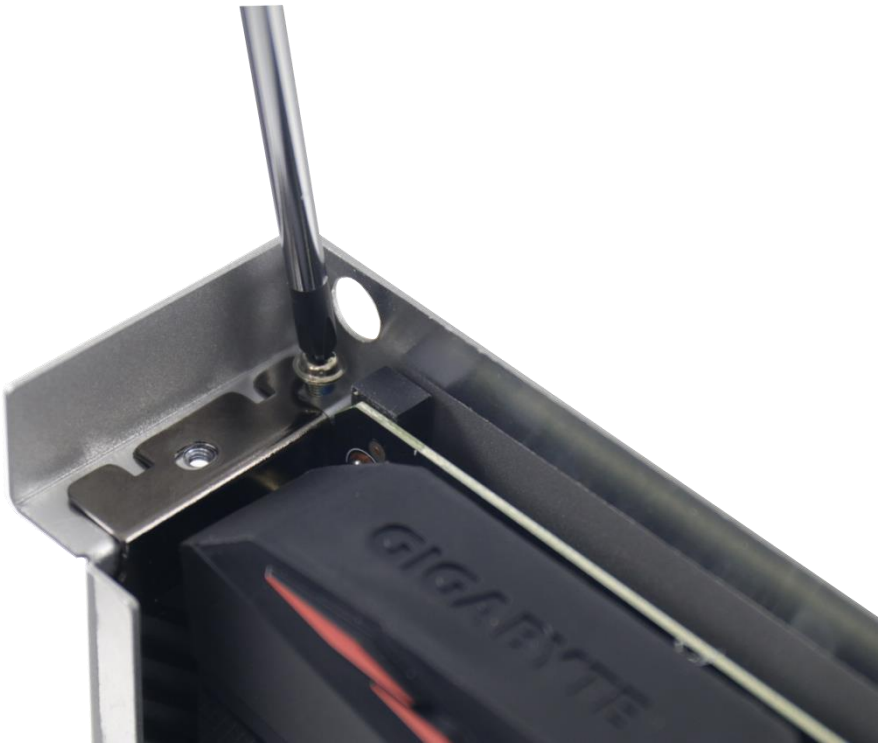
2. Remove the PCIe slot cover and gently lower the PCIe graphics card by matching the PCIe gold fingers into the slot.



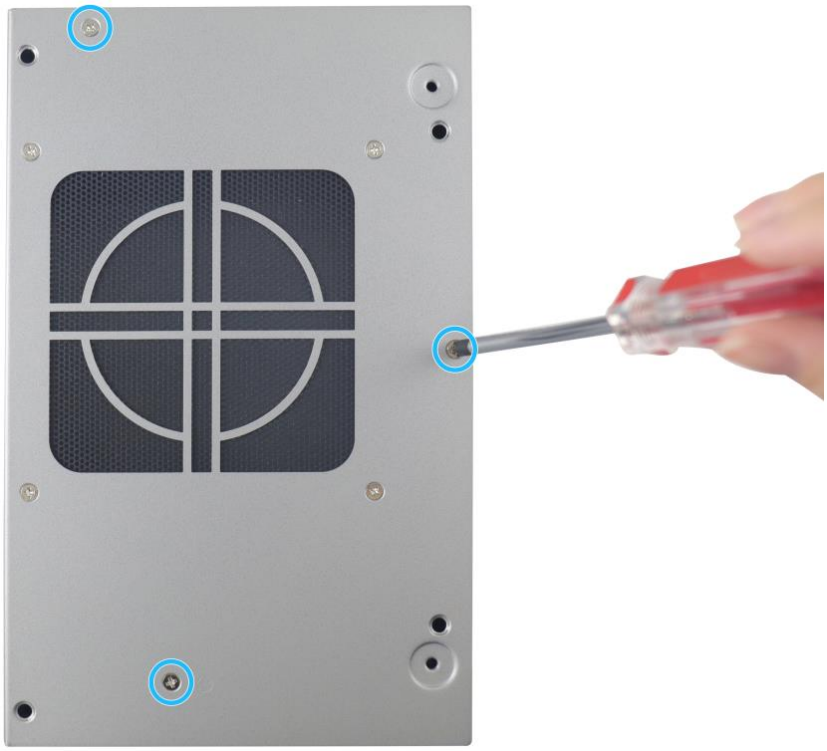
3. Make sure the bottom of the graphics card slot cover is properly inserted into the hinge on the enclosure.



4. Secure the graphics card’s slot cover with two (2) screws.

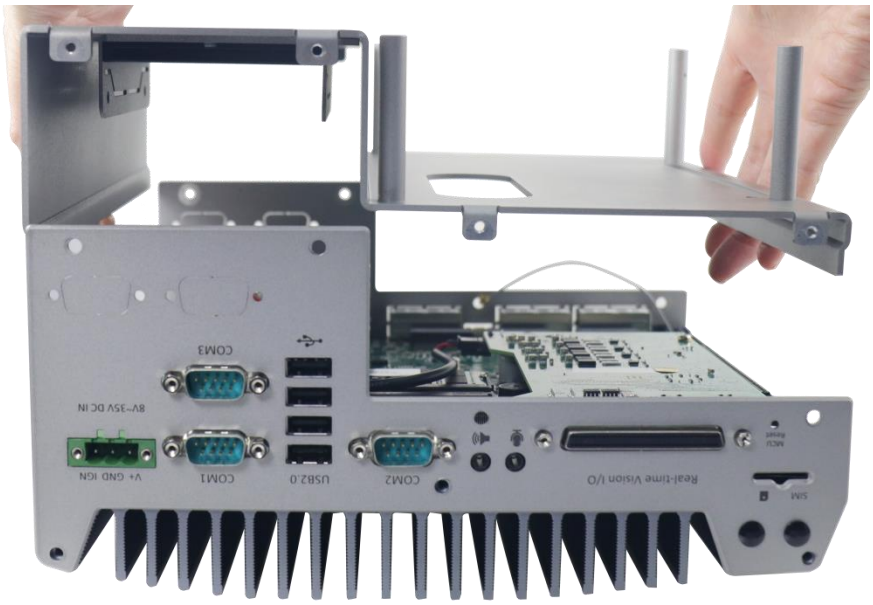


5. Reinstall the Cassette enclosure and secure it with the three (3) screws shown below.



3.7 Installing the System Enclosure

1. To reinstall the system enclosure, fit the front and rear panels and gently lower the base panel.



2. Secure the seven (7) screws on the front panel.

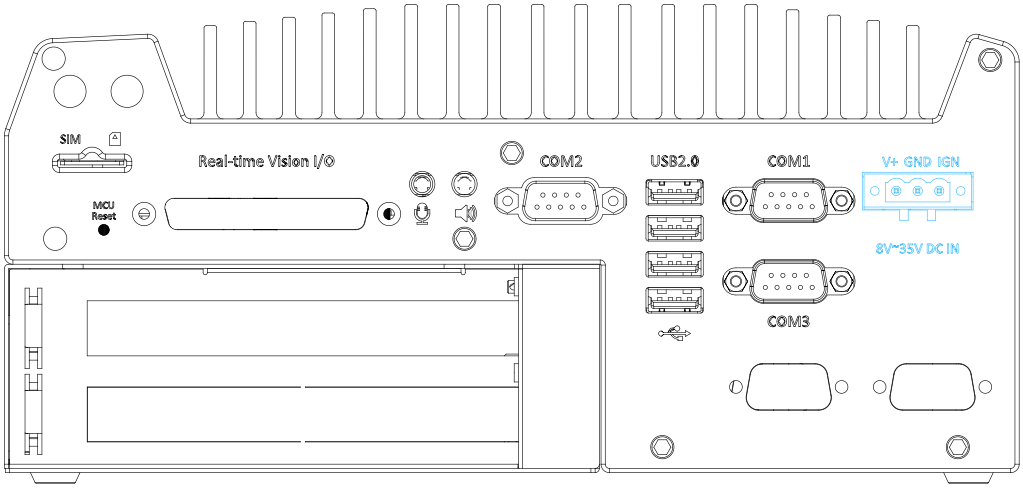
3. Secure the seven (7) screws on the rear panel.

4. Gently lower the Cassette module while matching the four position poles.

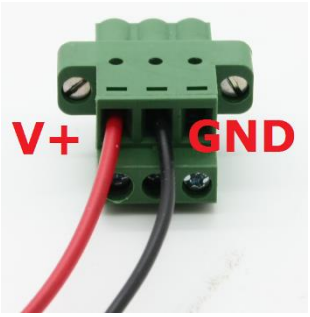
5. Secure the Cassette module by securing the four (4) screws.

3.8 DC Power Connection

The system uses a 3-pin pluggable terminal block to accept 8~35V DC power input. It is a reliable, convenient and easy method to directly wire cables to the DC power connector. The pluggable terminal block is also used to accept ignition signal To connect DC power via the 3-pin pluggable terminal block, please refer to the procedures described below.



1. Before connecting the cables, please make sure the DC power supply is unplugged!
2. Take the 3-pin pluggable terminal block out of the accessory box. The terminal block fits the wires with a gauge of 12~24 AWG.



3. Carefully identify the positive and negative contacts of your DC power supply and the pluggable terminal block. The polarities between DC power supply and terminal block must be positive (+) to positive (+) and ground (GND) to ground (GND).
4. Insert the wires to the matching pluggable terminal block contacts and tighten clamping screws using a Philips screwdriver.
5. Plug in the terminal plug into the 3-pin pluggable terminal block on the system enclosure and secure the plug using a flat-head screwdriver.

WARNING

The system accepts 8~35 VDC when using terminal block for DC input. Please make sure the voltage and polarity of DC power is correct before you connect and power on the system. Supplying a voltage over 35V or incorrect polarity will damage the system!

3.9 Wall Mount/ Anti-Vibration Bracket Installation

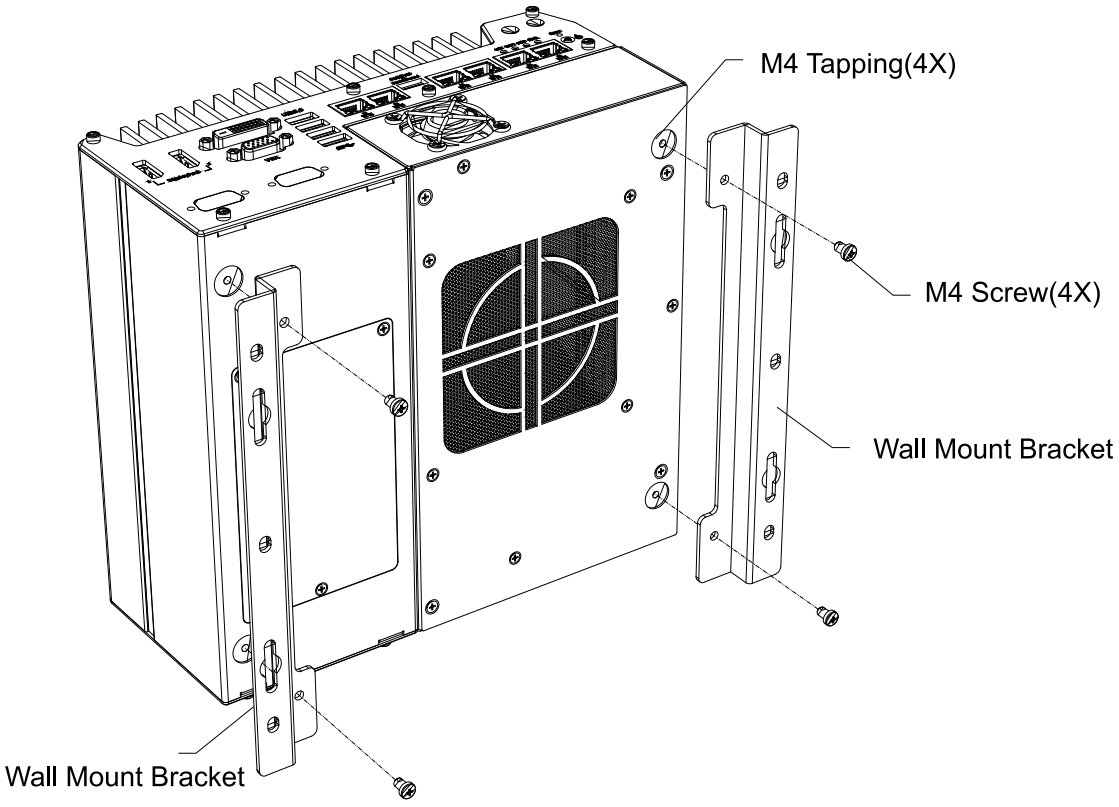
Nuvis-5306RT features a patented mechanical design that creates a sealed tunnel for air flow. To obtain best efficiency for heat dissipation, it is recommended that a minimum clearance of 20mm is reserved at the bottom side of Nuvis-5306RT controller. Nuvis-5306RT is shipped with stand-off brackets designed to create a 20mm clearance. To mount your Nuviso-5306RT controller on the wall or flat surface, please refer to the instructions listed below.

NOTE

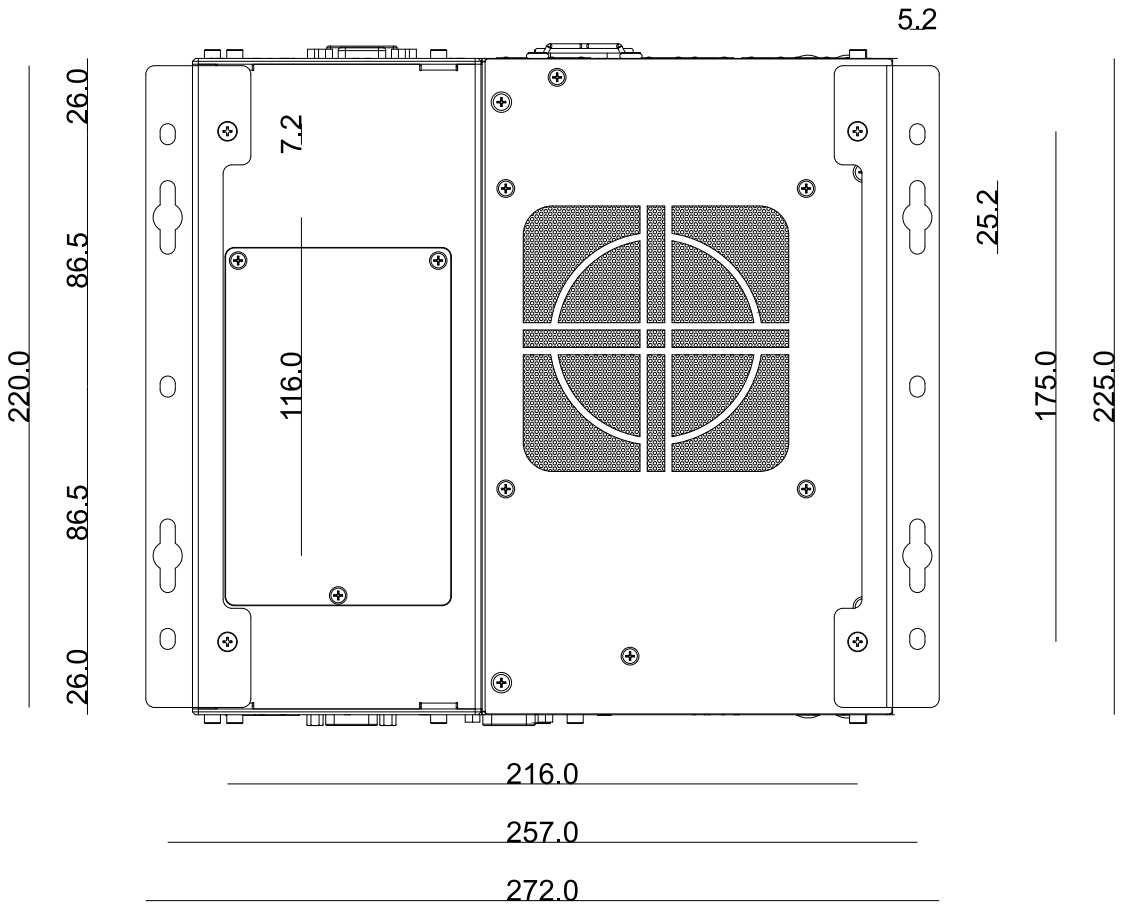
1. *You will need to remove the four (4) rubber stands at the bottom of the enclosure if they have been attached.*
2. *For customers using customized mounting design, please make sure you have at least 20mm clearance underneath the system. Mounting the system without the recommended minimum clearance may significantly reduce GPU performance at high ambient temperature.*

3.9.1 Wall Mount Bracket Installation

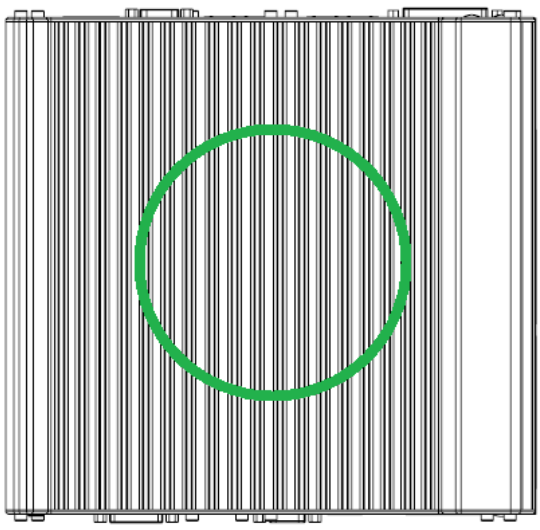
1. Get two wall-mounting brackets and four M4 screws from the accessory box. Fix the mounting brackets onto the bottom of the system using M4 screws.

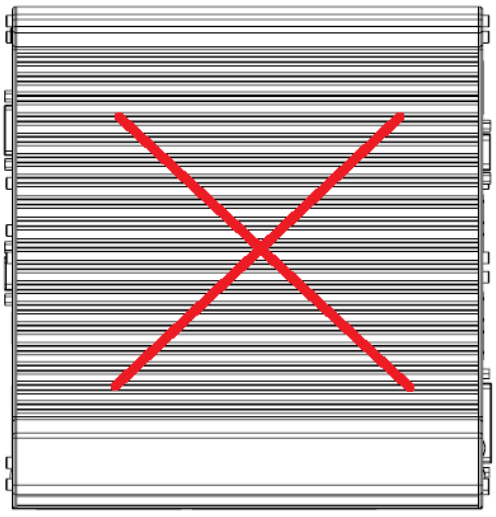


2. Place the system on a flat surface and fix it with screws. You can also take advantage of the keyhole-shaped holes on mounting brackets to suspend the system on the Wall.



3. When wall mounting, please mount the system's in the direction so the heatsink's fins are placed vertically for optimal heat dissipation efficiency.





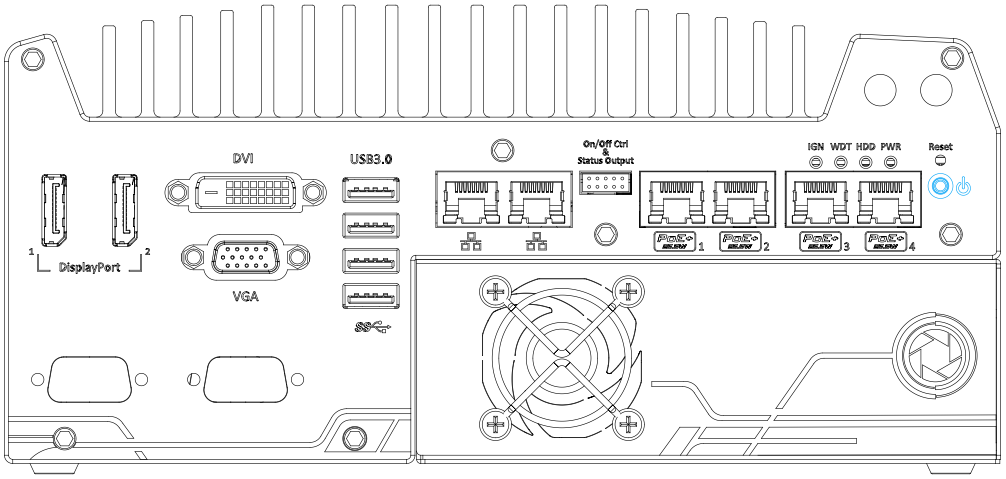
3.10 Powering On the System

There are four methods to power on the system

- Pressing the power button
- Via an external non-latched switch
- Sending a LAN packet via Ethernet (Wake-on-LAN)

3.10.1 Powering On Using the Power Button

This is the simplest way to turn on your system. The power button on the front panel is a non-latched switch and behaves as the ATX-mode on/off control. With DC power connected, pushing the power button will turn on the system and the PWR LED indicator will light up. Pushing the button when system is on will turn off the system. If your operating system supports ATX power mode (i.e. Microsoft Windows or Linux), pushing the power button while the system is in operation will result in a pre-defined system behavior, such as shutdown or hibernation.

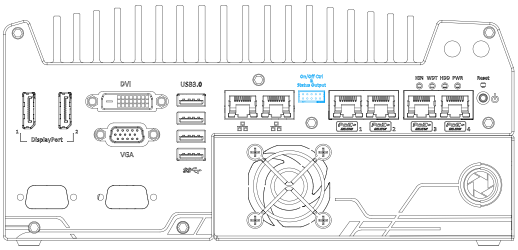


3.10.2 Powering On Using External Non-latched Switch

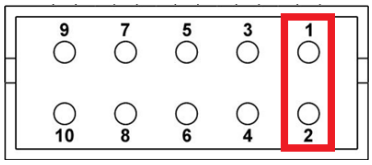
If your application demands the system to be placed inside a cabinet, you may use an external non-latched switch to power on/ off the system. The system provides a "On/ Off Control Ctrl & Status Output" connection (a 2x5, 2.0mm pitch wafer connector) for connecting a non-latched switch and acts as the ATX-mode power on/off control switch. The external non-latched switch acts exactly the same as the power button on the front panel. To setup and power on/ off the system using an external non-latched switch (ATX-mode), please follow the steps described below.

1. Acquire a non-latched switch with a 2x5, 2.0mm pitch wafer terminal and the switch must be connected to pin#1 and pin #2 (polarity is negligible).

2. Connect the wafer terminal to the "On/Off Control Ctrl & Status Output" connector on the system



On/Off Control Ctrl & Status Output

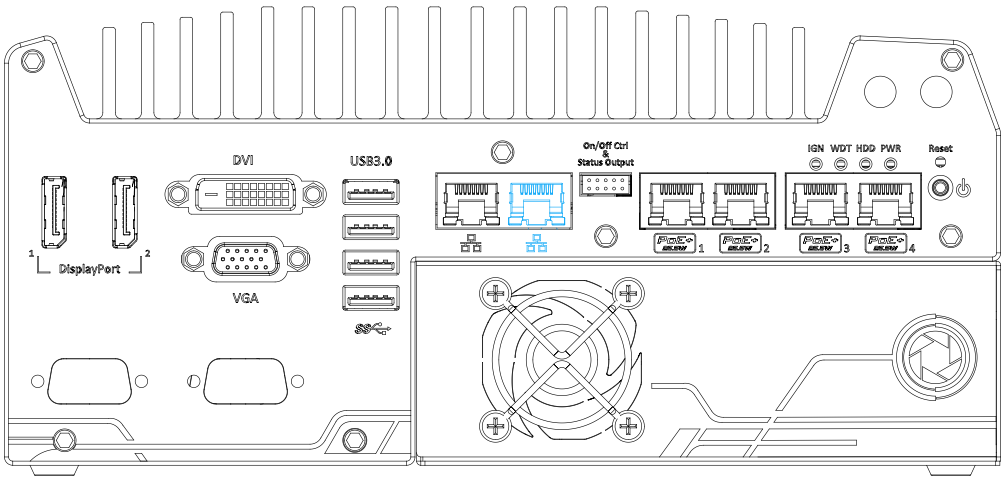


Pin #1 and Pin #2

3. With DC power connected, pushing the power button will turn on the system and the PWR LED indicator will light up. Pushing the button when system is on will turn off the system. If your operating system supports ATX power mode (i.e. Microsoft Windows or Linux), pushing the power button while the system is in operation will result in a pre-defined system behavior, such as shutdown or hibernation.

3.10.3 Powering On Using Wake-on-LAN

Wake-on-LAN (WOL) is a mechanism to wake up a computer system from a S5 (system off with standby power) state via issuing a magic packet. The system's Wake-on-LAN compatible GbE port is shown below.

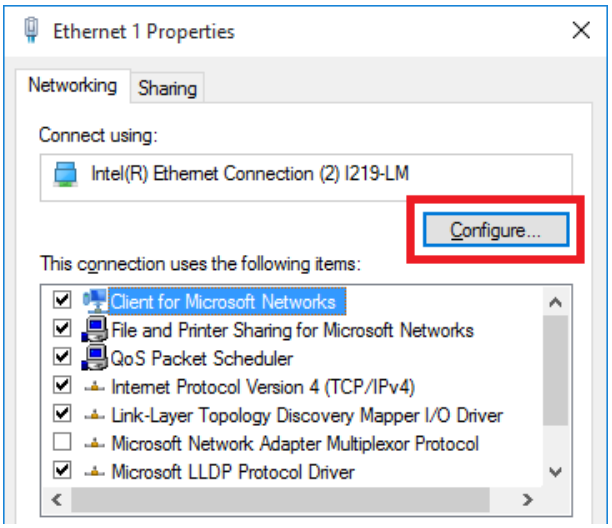


NOTE

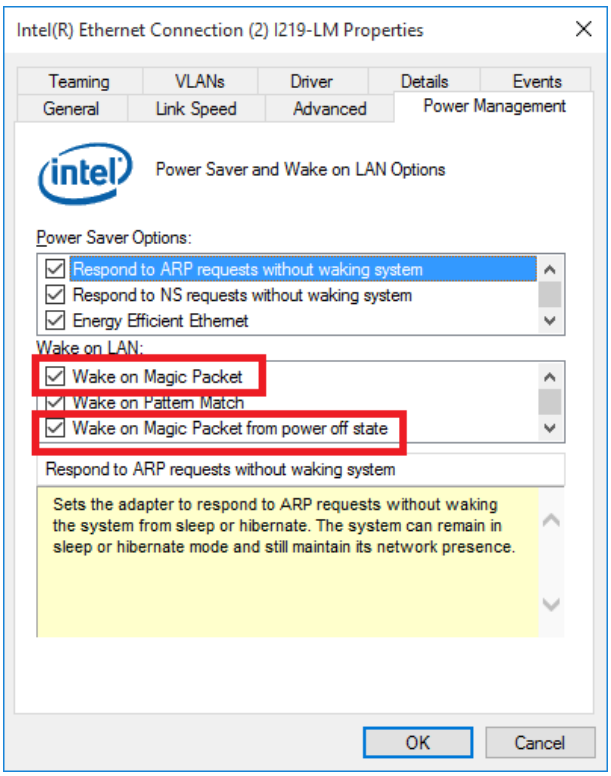
Please make sure the Intel chipset and Ethernet driver has been properly installed prior to setting up WOL function.

To enable WOL function, please set up WOL settings in the BIOS and in the operating system by follow the steps described below.

1. When the system boots up, press F2 to enter BIOS setup utility.
2. Go to the **[Power]** > **[Wake On LAN]** and set it to **[Enabled]**.
3. Press F10 to "**Save changes and exit BIOS**" and allow the system boot into the operating system.
4. Once booted into the Windows system, press "**Windows key + E**", right-click on "**Network** > **Properties** > **Change adapter settings**". Locate and double-click on the adapter Intel® I219 Gigabit Network Connection, click on **Configure...**



5. Click on the **Power Management** tab and check the following options. Click on OK when done.



Magic Packet

The magic packet is a broadcast frame containing anywhere within its payload 6 bytes of all 255 (FF FF FF FF FF FF in hexadecimal), followed by sixteen repetitions of the target computer's 48-bit MAC address.

For example, NIC's 48-bit MAC Address is 78h D0h 04h 0Ah 0Bh 0Ch

DESTINATION SOURCE MISC

FF FF FF FF FF FF

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

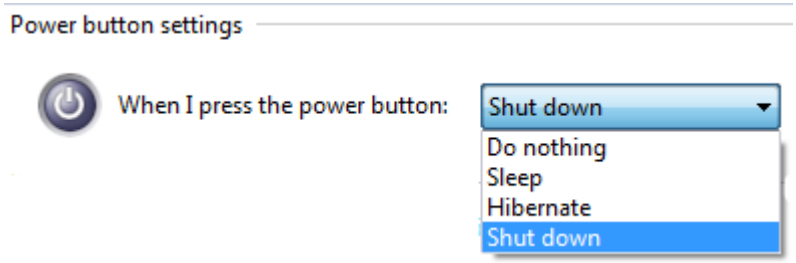
78 D0 04 0A 0B 0C 78 D0 04 0A 0B 0C

MISC CRC

There are some free tools available on Internet that can be used to send a magic packet. Please refer to the following link to understand more about Magic Packet.

3.10.4 Configure your Windows system

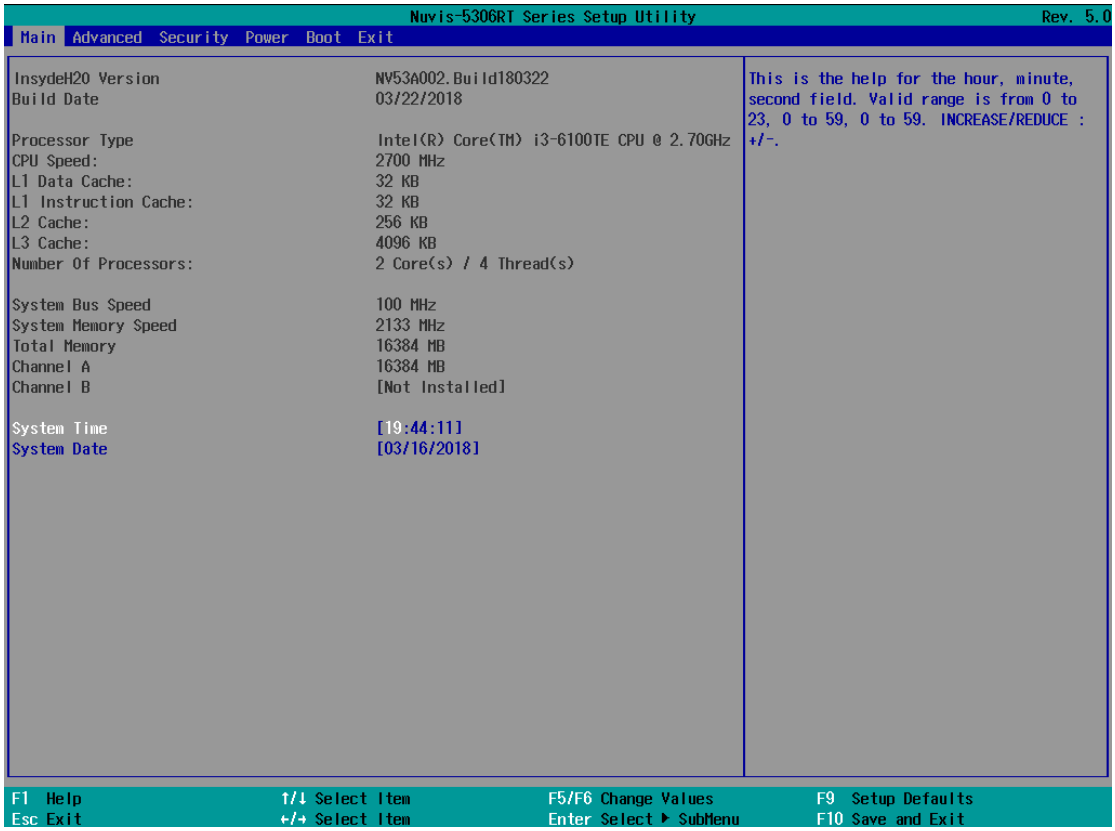
Please make sure you've configured your Windows system to initiate a shutdown process when pressing the power button. By default, Windows 7/ 8/ 10 goes to sleep (S3) mode when power button is pressed. As sleep (S3) is not a complete shutdown behavior, the ignition control function does not recognize the finish of a normal shut down process and thus users will encounter a system hard-off (power cut-off after 10 minutes). Please configure "When I press the power button" to "Shut down" in your Windows system settings.



4 System Configuration

4.1 BIOS Settings

The system is shipped with factory-default BIOS settings meticulously programmed for optimum performance and compatibility. In this section, we'll illustrate some of BIOS settings you may need to modify. Please always make sure you understand the effect of change before you proceed with any modification. If you are unsure of the function you are changing, it is recommended to change one setting at a time to see its effect(s).

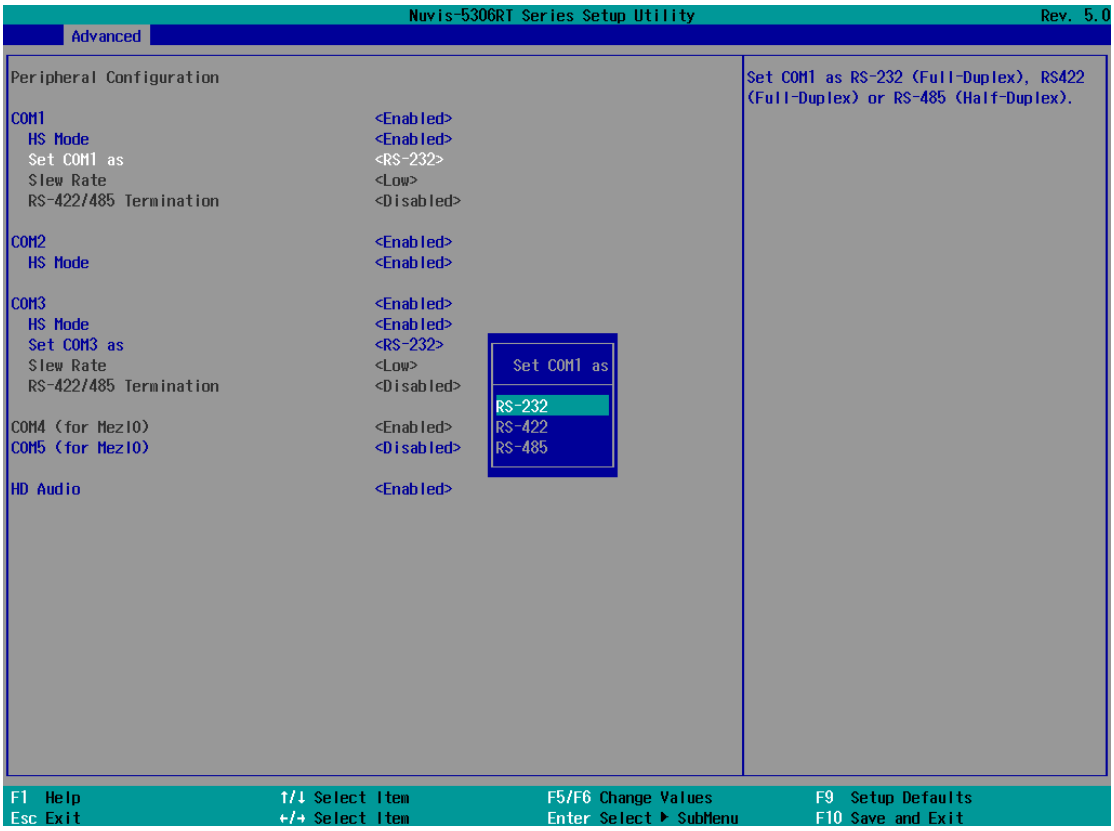


NOTE

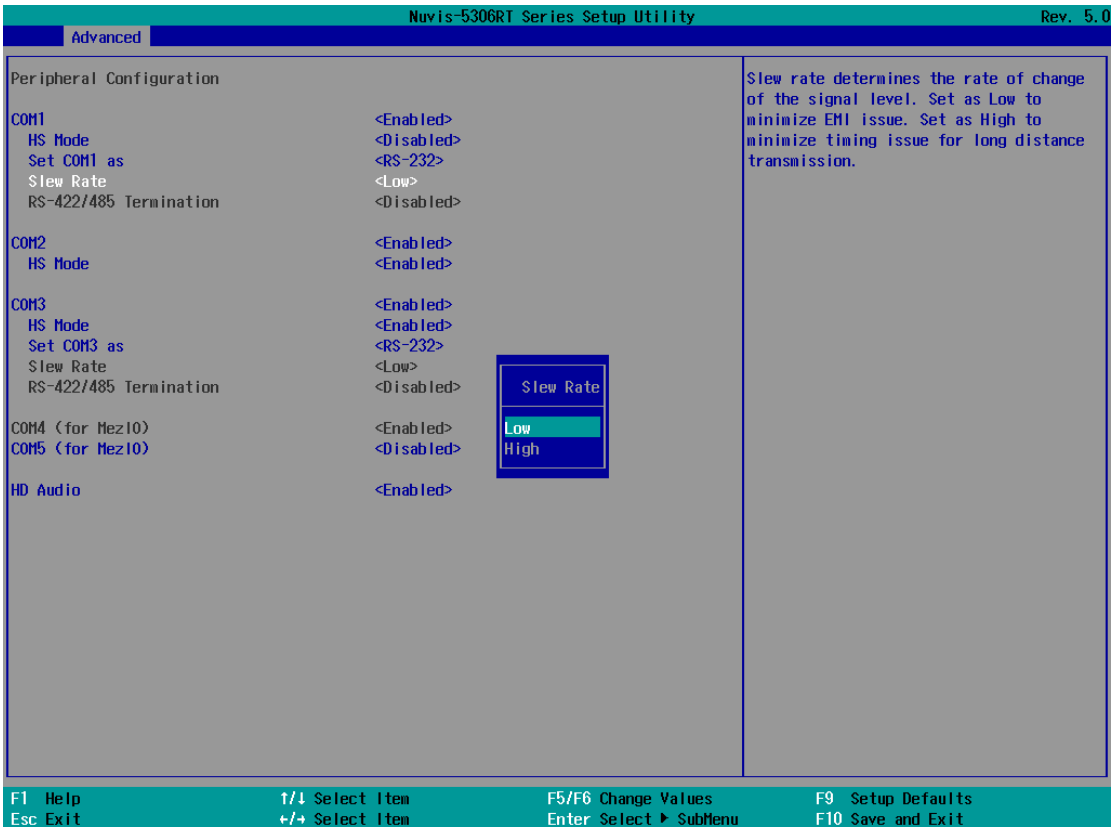
Not all BIOS settings will be discussed in this section. If a particular setting/ function you are after requires specific BIOS settings but is not discussed in this section, please contact Neousys Technical Support staff.

4.1.1 COM1 & COM3 Operating Mode

COM1 and COM3 of System series support RS-232 (full-duplex), RS-422 (full-duplex) and RS-485 (half-duplex) mode. You can set the COM1/ COM3 operating mode via BIOS settings.



Another option in BIOS called "*Slew Rate*" defines how sharp the rising/falling edge is for the output signal of COM1/ COM3. For long-distance RS-422/485 transmission, you may set the "*Slew Rate*" option as "High" to improve signal quality.

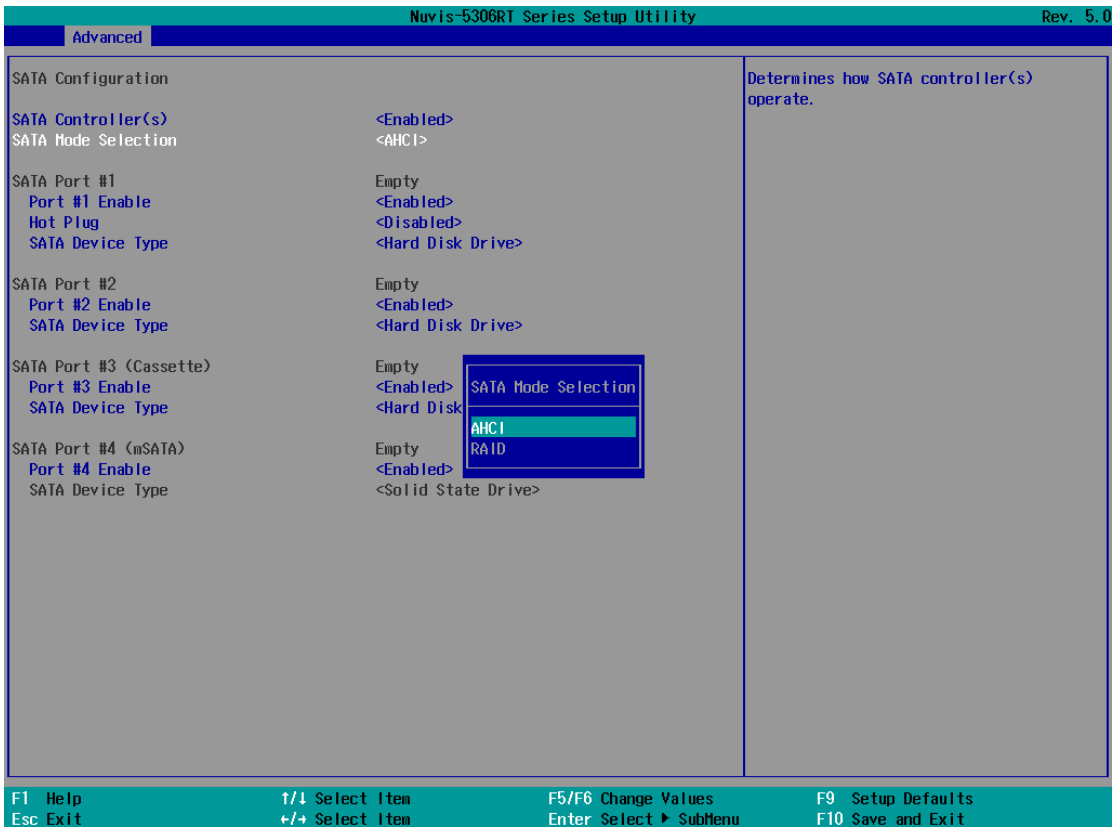


To set COM port operating mode:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Advanced] > [Peripheral Configuration]**.
3. Highlight the COM port you wish to set and press Enter to bring up setting options. Use up/ down arrow to highlight your selection and press Enter.
4. Repeat step 2 to set other COM ports.
5. Press F10 to “Exit Saving Changes”.

4.1.2 SATA Configuration

The system SATA controller supports two (2) operating modes: **AHCI** and **RAID** mode. **AHCI** mode, which exposes SATA's advanced capabilities such as hot swapping and native command queuing, is supported in several later version of operating systems. **RAID** mode provides redundant data storage (RAID 1) or a higher throughput (RAID 0). The system features built-in hardware RAID. No additional H/W or driver is needed to use RAID function.



Recommended SATA controller mode settings:

- If you're using Windows Vista, Windows 7/ 8/ 10, or Linux kernel 2.6.19 or later, you can select **AHCI** mode for better performance.
- If you're installing two 2.5" HDD/ SSD and looking for data striping (RAID 0) or data mirroring (RAID 1), you can select **RAID** mode to utilize built-in RAID.

To set SATA controller mode:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Advanced]** > **[SATA Configuration]**.

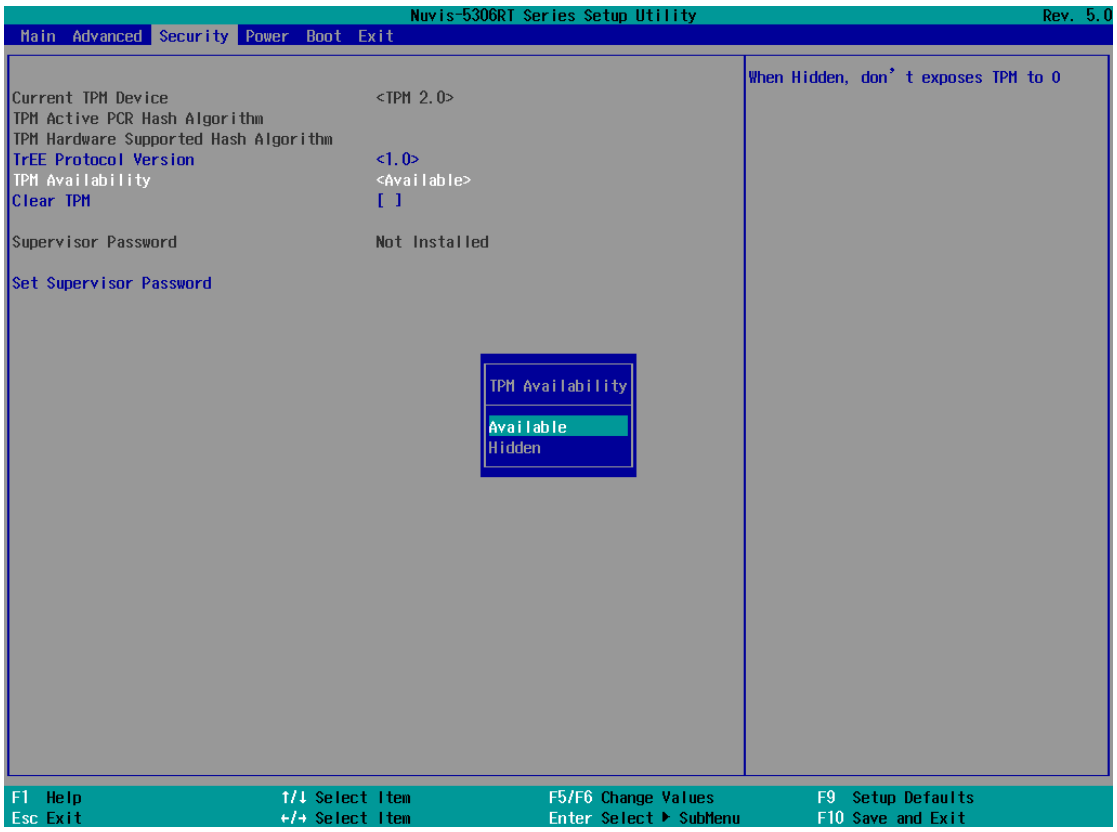
3. Highlight the SATA port you wish to set and press Enter to bring up setting options. Use up/ down arrow to highlight your selection and press Enter.
4. Repeat step 3 to set other SATA ports.
5. Press F10 to “Exit Saving Changes”.

NOTE

By enabling Hot Plug function for SATA Port #1, the front accessible 2.5” HDD/SSD tray may be removed while the system is in operation, please avoid installing operating system onto this drive!

4.1.3 TPM Availability

Trusted Platform Module (TPM) is a hardware-based cryptoprocessor to secure hardware by integrating cryptographic keys into devices. The system is designed with on-board TPM 2.0 module. As TPM 2.0 requires 64-bit Windows 7/8/10 with UEFI boot mode, it is disable in BIOS by default. For customers who want to utilize TPM feature, you will need to enable TPM in BIOS as well as install Windows with UEFI mode.

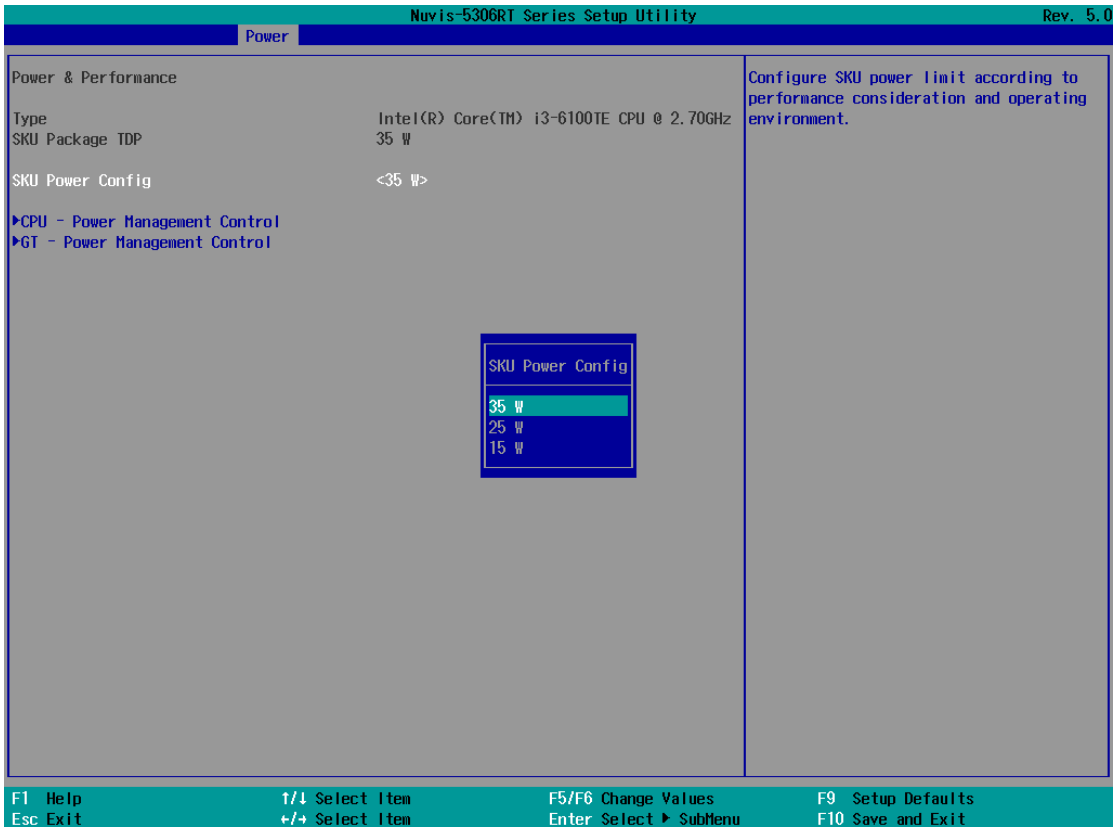


To enable TMP availability:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Security] > [TPM Availability]**, press Enter to bring up Options, Available/ Hidden.
3. Highlight your selection, press Enter and press F10 to “Exit Saving Changes”.

4.1.4 CPU SKU Power Configuration

The system supports various 6th-Gen Skylake LGA1151 CPUs. A unique feature, "**SKU Power Config**" is implemented in BIOS to allow users to specify user-defined SKU power limit. Although the system is designed to have best thermal performance with CPUs of 35W TDP, you can install a 65W CPU and limit its SKU power to obtain more computing power. This feature gives you the flexibility of CPU selection and great balance between computing power and operating temperature range.



Here is our suggestion regarding specifying SKU power for system with 65W/ 54W/ 51W CPUs.

- For system running at up to 70°C ambient, specify SKU power to 35 W.
- For system running at up to 60°C ambient, specify SKU power to 45 W.
- For system running at up to 50°C ambient, specify SKU power to Max. TDP.

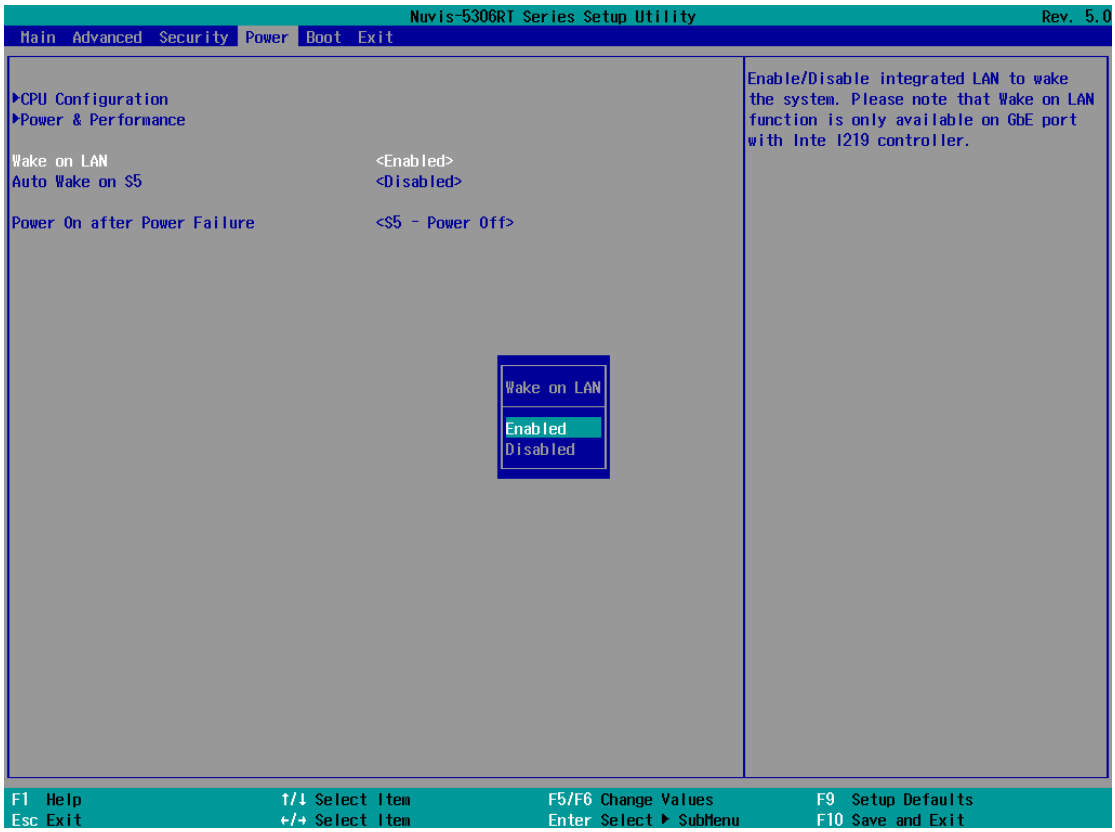
To configure the CPU SKU power limit:

1. When the system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Power]** → **[Power & Performance]**.

Select a proper value of SKU power limit for **[SKU Power Config]** option.

4.1.5 Wake on LAN Option

Wake-on-LAN (WOL) is a mechanism which allows you to turn on your System series via Ethernet connection. To utilize Wake-on-LAN function, you have to enable this option first in BIOS settings. Please refer "Powering On Using Wake-on-LAN" to set up the system.

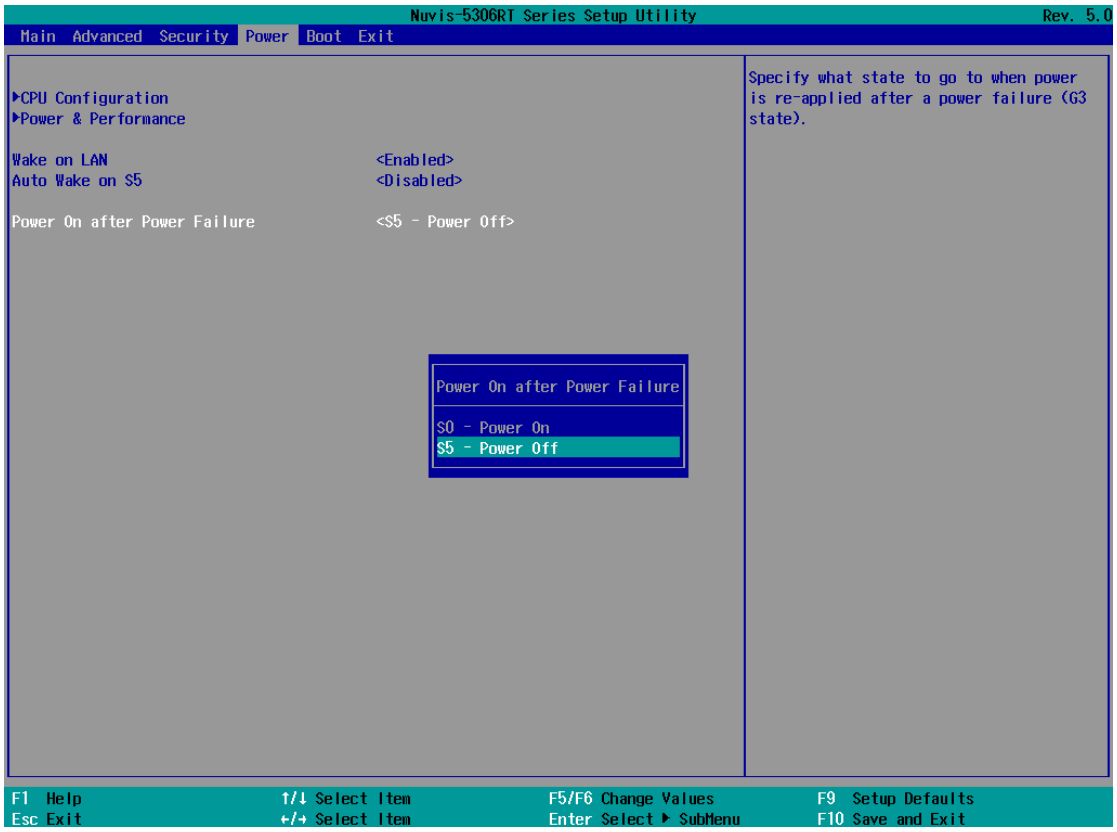


To enable/disable "Wake on LAN" option:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Power] > [Wake on LAN]**.
3. Press Enter to bring up setting options, scroll to the setting you desire and press Enter to set.
4. Press F10 to "Exit Saving Changes.

4.1.6 Power On after Power Failure Option

This option defines the behavior of system when DC power is supplied.



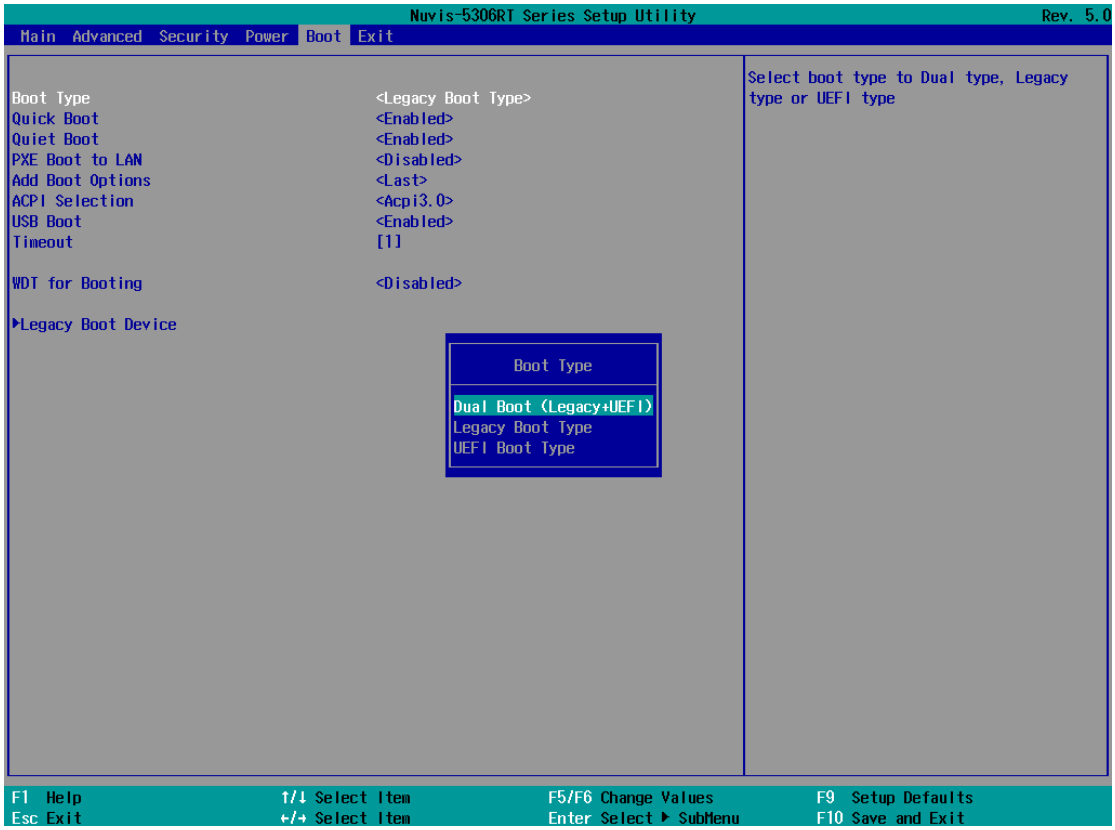
Value	Description
S0 – Power On	System is powered on when DC power is supplied.
S5 – Power Off	System is kept in off state when DC power is supplied.

To set “Power On after Power Failure” option:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Power] > [Power On after Power Failure]**.
3. Scroll down to highlight **[Power On after Power Failure]**, press Enter to bring up setting options, S0 – Power On or S5 – Power Off, and press Enter to select the setting.
4. Press F10 to “Exit Saving Changes”

4.1.7 Boot Type (Legacy/ UEFI)

The system supports both Legacy and Unified Extensible Firmware Interface (UEFI) boot modes. UEFI is a specification proposed by Intel to define a software interface between operating system and platform firmware. Most modern operating systems, such as Windows 7/ 8/ 10 and Linux support both Legacy and UEFI boot modes. The Legacy boot mode uses MBR partition for disk and VBIOS for video initialization, the UEFI boot mode uses GPT partition which supports greater than 2TB partition size and GOP driver for faster video initialization.



It is recommended that:

- If you need greater than 2TB disk partition or want to use TPM 2.0 function, you shall choose UEFI boot mode and install operating system accordingly.
- Otherwise you can choose Legacy boot mode for most cases.

To configure Boot Type:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Boot] > [Boot Type]**, press Enter to bring up options, Dual Boot (Legacy+UEFI), Legacy Boot Type, UEFI Boot Type.
3. Highlight you selection and press Enter.
4. Press F10 to “Exit Saving Changes”.

4.1.8 Boot Option for Newly Added Device

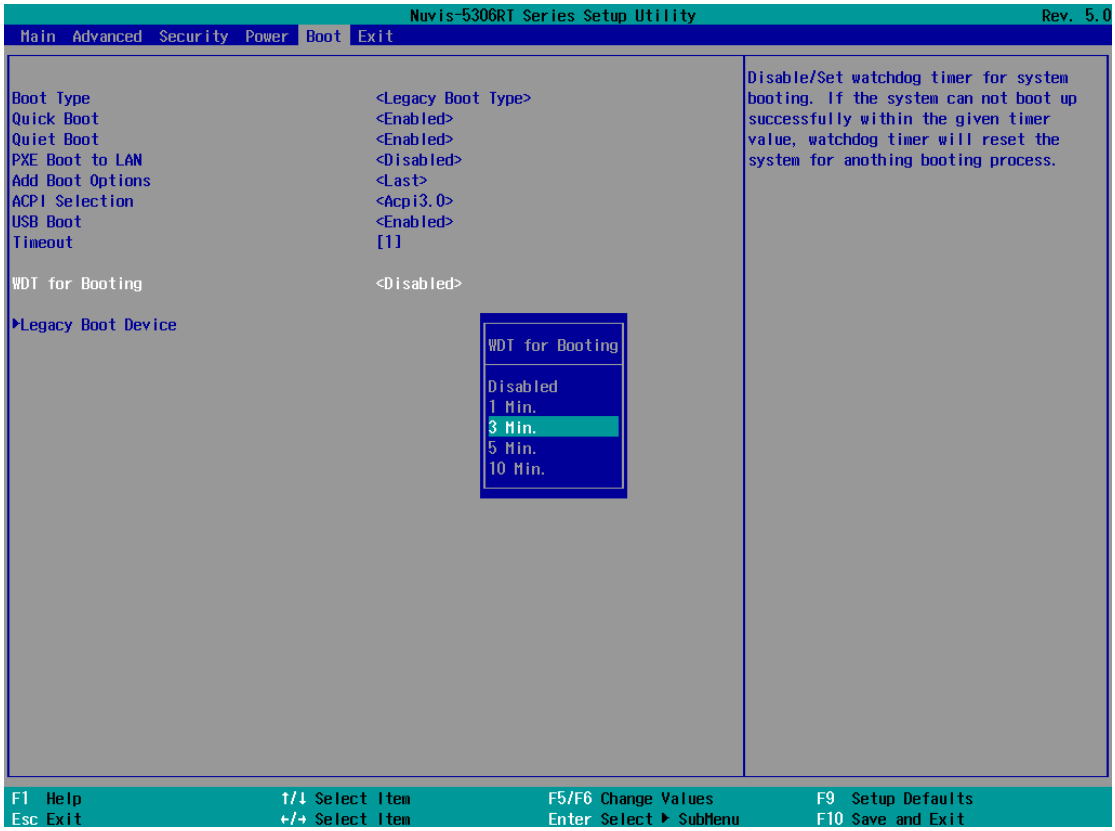
The Add Boot Options dedicates the boot sequence order of a newly added device (eg. USB flash drive). The setting allows you to set the newly added device as the first device or as the last device to boot.

To set Add Boot Options:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to [Boot] > [Add Boot Options], press Enter to bring up options, First or Last.
3. Highlight your selection and press Enter, press F10 to “Exist Saving Changes”.

4.1.9 Watchdog Timer for Booting

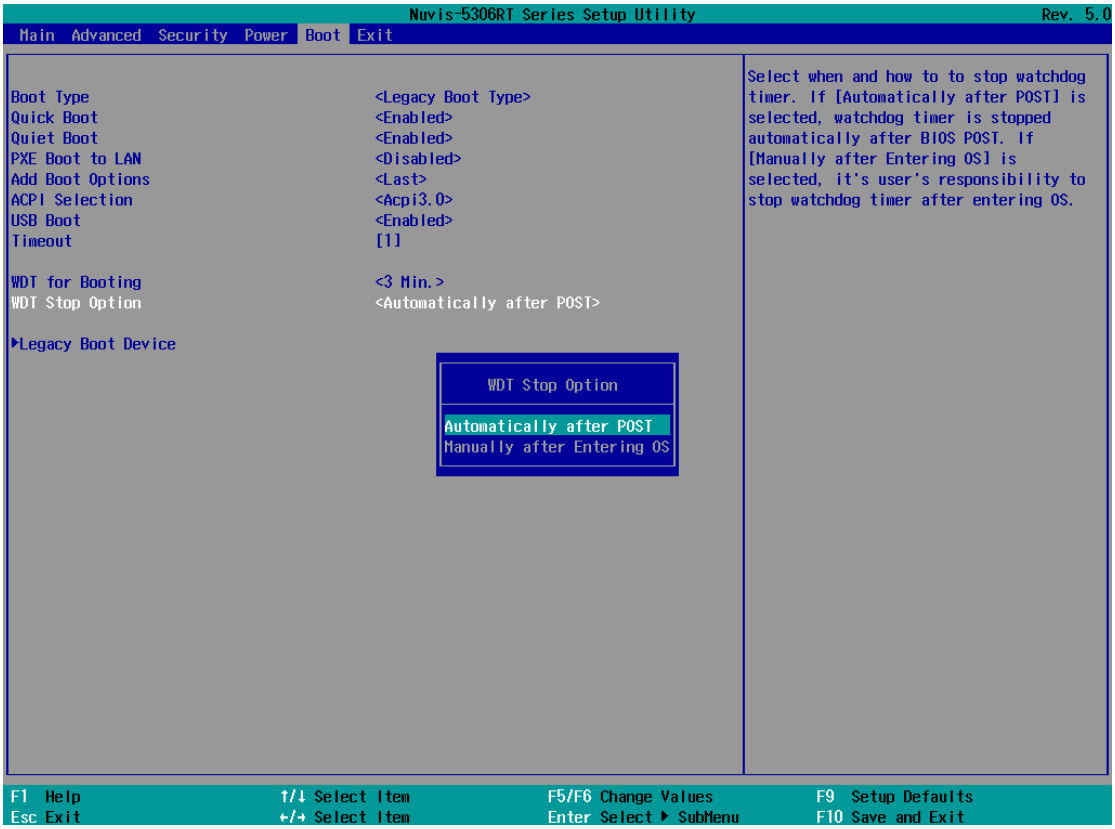
The system BIOS has a useful feature which allows users to use watchdog timer to ensure a successful boot process. You can specify the timeout value for watchdog timer. Once the watchdog timer expires, BIOS issues a reset command to initiate another boot process. You can also set the behavior of how to stop the watchdog timer. There are two options in BIOS menu, "**Automatically after POST**" and "**Manually after Entering OS**". When "**Automatically after POST**" is selected, the BIOS automatically stop the watchdog timer after POST (Power-On Self Test) OK. When "**Manually after Entering OS**" is selected, it's user's liability to stop the watchdog timer when entering OS. This guarantees the system can always boot to OS, otherwise another booting process will be initiated. For information about programming watchdog timer, please refer to Watchdog Timer & Isolated DIO.



To set the watchdog timer for boot in BIOS:

1. When system boots up, press F2 to enter BIOS setup utility.
2. Go to **[Boot]** menu.
3. Disable or select timeout value for **[WDT for Booting]** option.

4. Once you give a timeout value, the **[WDT Stop Option]** option appears. You can select *"Automatically after POST"* or *"Manually after Entering OS"*.



5. Press F10 to "Exit Saving Changes.

4.1.10 Selecting Legacy/ UEFI Boot Device

When multiple bootable devices are connected (e.g. HDD, mSATA, USB flash disk, USB DVD-drive), you may need to select one of them as the boot device. There are two ways to select the device. You can either, press F12 when system boots up to go to Boot Manager and then select one of the devices, or select the boot device in BIOS settings.

To set boot order for devices in UEFI Boot Device:

1. When system boots up, press F2 to enter BIOS setup utility
2. Go to **[Boot] > [UEFI Boot Device]**
3. Highlight the device you wish to make boot order changes to and press F5/ F6 or +/ - to change device boot order.

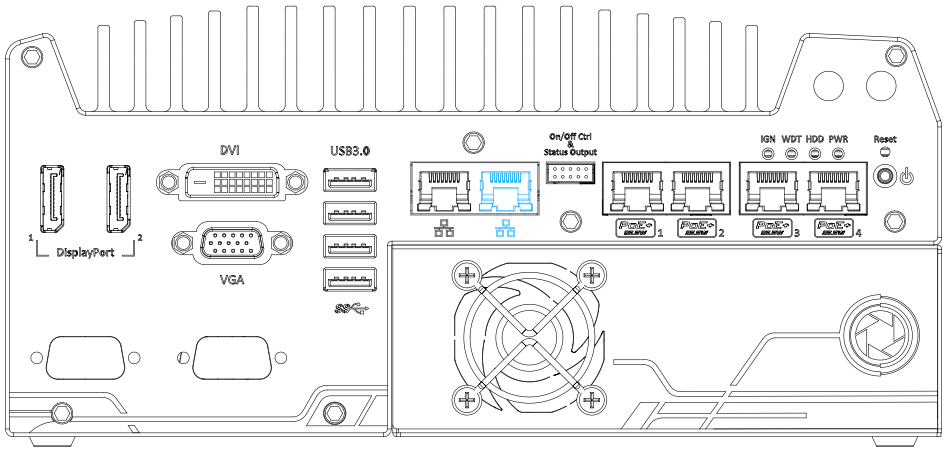
To select boot order for devices in Legacy Boot Device:

1. When system boots up, press F2 to enter BIOS setup utility
2. Go to **[Boot] > [Legacy Boot Device]**, you can choose the type of device to boot list by selecting “**By Device”** or “**By Device Type**”.
3. Highlight the device or device category you wish to make boot order changes to and press F5/ F6 or +/ - to change device boot order.

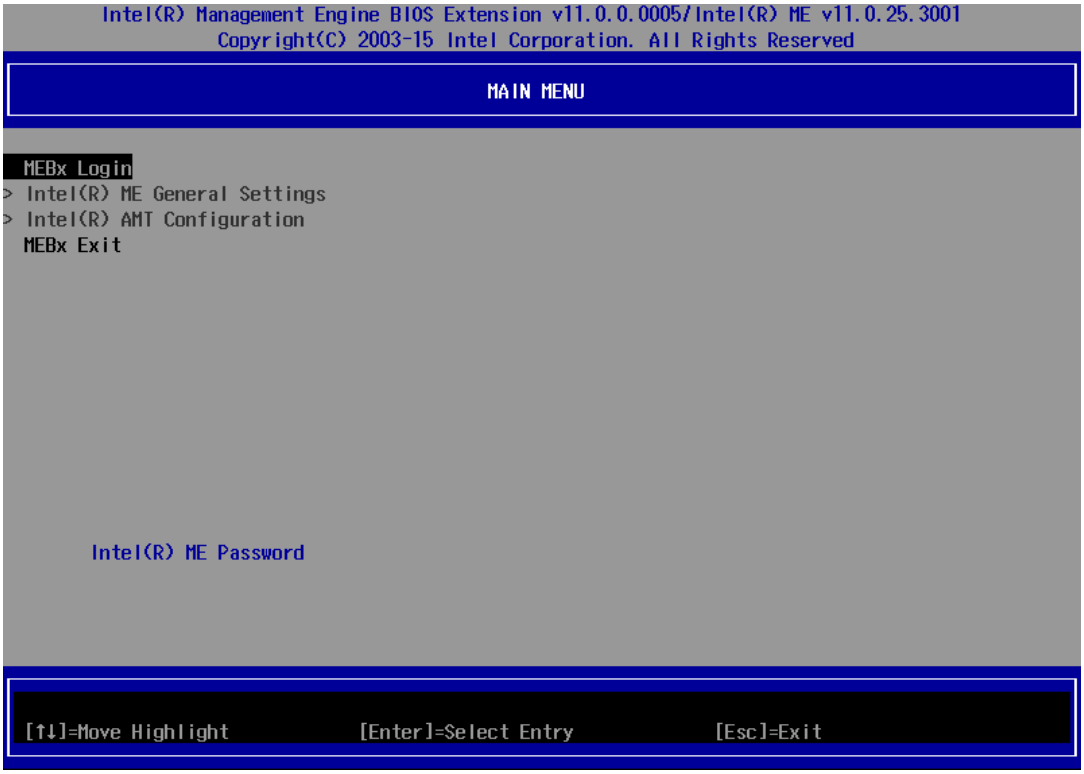
4.2 AMT Configuration

Intel® AMT (Active Management Technology) is a hardware-based technology for remotely managing target PCs via Ethernet connection. The system supports AMT function via its Ethernet port implemented with Intel I219-LM. Prior to using AMT to remotely control the system, you need to configure AMT password and network settings.

1. Make sure you have connected the proper Ethernet port (via I219-LM).



2. When the system boots up, press F10 to enter the MEBx configuration menu.



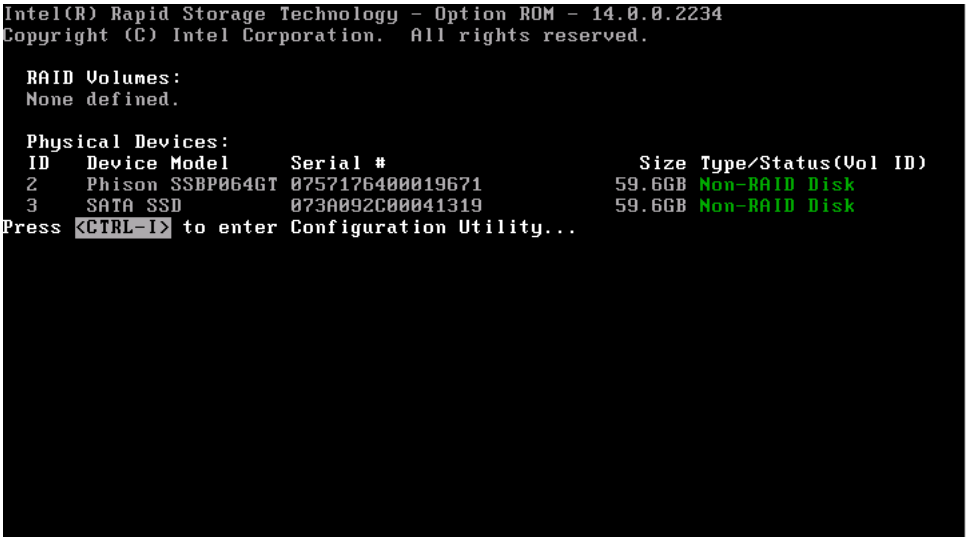
3. Highlight MEBx Login and press Enter, a prompt will appear asking for password. The default password is “admin”. For further MEBx configuration details, please refer to Intel® MEBX User Guide.

4.3 RAID Volume Configuration

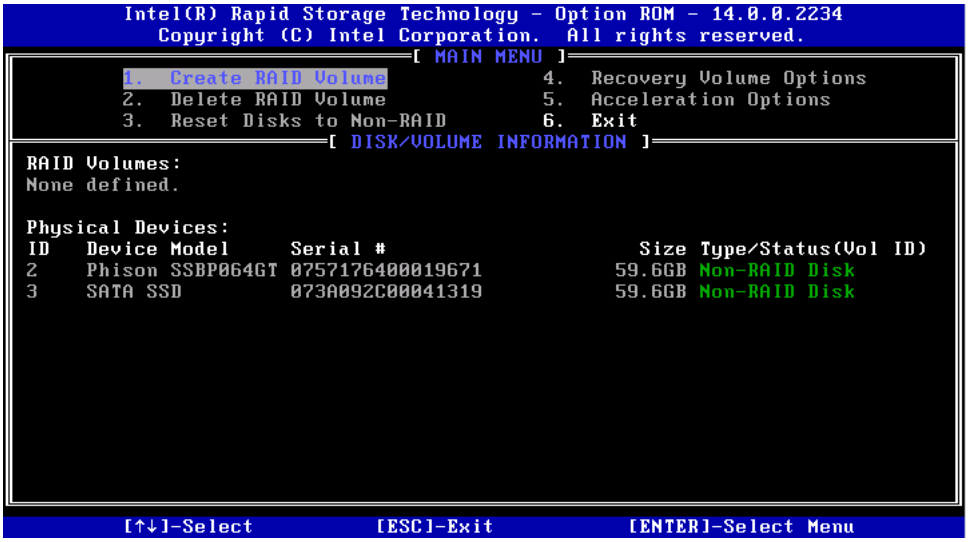
The system supports hardware RAID function for more reliable and efficient disk access. The built-in RAID supports RAID 0 (data stripping) and RAID 1 (data mirroring). You can configure RAID mode according to your needs by following steps listed below.

1. Configure the SATA controller by going to **[Advanced]** > **[SATA Configuration]** > **[SATA Mode Selection] and select [RAID]** (refer to SATA Configuration for details).

2. Reboot the system once RAID mode has been set, upon reboot, you will see a UI banner for RAID configuration.



3. Press **[Ctrl + I]** to enter the RAID configuration menu. Follow the on-screen instructions to create, delete or configure the RAID volume.



5 OS Support and Driver Installation

5.1 Operating System Compatibility

The system support most operating system developed for Intel® x86 architecture. The following list contains the operating systems which have been tested by Neousys Technology.

- Microsoft Window 7 (x86*/ x64*)
- Microsoft Window 8 (x64)
- Microsoft Window 10 (x64)
- Microsoft Embedded Standard 7
- CentOS 7
- Debian 8.7**
- Fedora 24**
- OpenSUSE 42.1**
- Ubuntu 14.04.4 LTS and 16.04 LTS**

NOTE

** Due to xHCI driver is not included natively in Windows 7, you may encounter Keyboard/ mouse issues when installing Windows 7. Neousys offers a Windows-based batch file and step-by-step installation guide.*

*** For distributions, graphics driver and RAID function may not be completely implemented in its kernel. You may encounter restrictions when using these features, such as triple independent display and RAID. For optimum operation, it is the users' responsibility to manually check for new drivers and upgrades!*

Neousys may remove or update operating system compatibility without prior notice. Please contact us if your operating system of choice is not on the list.

5.2 Install Drivers Automatically

The system comes with a “Drivers & Utilities” DVD that offers “one-click” driver installation process. It automatically detects your Windows operating system and installs all necessary drivers for you system with a single click.

To install drivers automatically, please refer to the following procedures.

1. Insert the “Drivers & Utilities” DVD into a USB DVD-drive connect to your system. A setup utility launches and the following dialog appears.

2. Click on **“Automatic Driver Installation”** and the setup utility will automatically detect your Windows operating system and install all necessary drivers. The installation process takes about 6~8 minutes depending on your Windows version. Once driver installation is done, the setup utility reboots your Windows and you may begin using your system.

5.3 Install Drivers Manually

You can also manually install each driver for the system. Please note when installing drivers manually, you need to install the drivers in the following sequence mentioned below.

5.3.1 For Windows 7 (x86)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Graphics driver (x:\Driver_Pool\Graphics_SKL_APL\Win_7_32\Setup.exe)
3. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_32\Setup.exe)
4. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_32\APPS\PROSETDX\Win32\DxSetup.exe)
5. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.3.2 For Windows 7 (x64)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Graphics driver (x:\Driver_Pool\Graphics_SKL_APL\Win_7_8_10_APL_64\Setup.exe)
3. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_64\Setup.exe)
4. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_64\APPS\PROSETDX\Winx64\DxSetup.exe)
5. TPM 2.0 driver (x:\Driver_Pool\TPM2\Win7_64\Windows6.1-KB2920188-v7-x64.msu)
6. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.3.3 For Windows 8 (x86)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_32\Setup.exe)
3. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_32\APPS\PROSETDX\Win32\DxSetup.exe)
4. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.3.4 For Windows 8 (x64)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Graphics driver (x:\Driver_Pool\Graphics_SKL_APL\Win_7_8_10_APL_64\Setup.exe)
3. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_64\Setup.exe)
4. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_64\APPS\PROSETDX\Winx64\DxSetup.exe)
5. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.3.5 For Windows 10 (x32)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_32\Setup.exe)
3. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_32\APPS\PROSETDX\Win32\DxSetup.exe)
4. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.3.6 For Windows 10 (x64)

The recommended driver installation sequence is

1. Chipset driver (x:\Driver_Pool\Chipset_10_APL\Win_ALL\SetupChipset.exe)
2. Graphics driver (x:\Driver_Pool\Graphics_SKL_APL\Win_7_8_10_APL_64\Setup.exe)
3. Audio driver (x:\Driver_Pool\Audio_ALC262\Win_ALL_64\Setup.exe)
4. LAN driver (x:\Driver_Pool\GbE_I210_I350\Win_ALL_64\APPS\PROSETDX\Win10_x64\DxSetup.exe)
5. ME driver (x:\Driver_Pool\ME_10_Series\Win_ALL_AMT\SetupME.exe)

5.4 Install WDT_DIO Driver Package

Neousys provides a driver package which contains function APIs for watchdog timer, digital I/ O, per-port PoE power on/off control and other platform-related functions. You should install the driver package (WDT_DIO_Setup.exe) in prior to use these functions.

NOTE

Please install WDT_DIO_Setup_v2.2.7.9 or later versions for The system.

5.4.1 For Windows 7 (x86)

Please execute the driver setup program in the following directory.

x:\Driver_Pool\WDT_DIO\XP_Win7_8_32\WDT_DIO_Setup_v2.2.7.9.exe

5.4.2 For Windows 7/ 8/ 10 (x64)

Please execute the driver setup program in the following directory.

x:\Driver_Pool\WDT_DIO\Win7_8_64\WDT_DIO_Setup_v2.2.7.9(x64).exe

5.4.3 For Windows 7/ 8/ 10 (WOW64)

Please execute the driver setup program in the following directory.

x:\Driver_Pool\WDT_DIO\Win7_8_WOW64\WDT_DIO_Setup_v2.2.7.9(wow64).exe

Appendix A: Using WDT & DIO

The watchdog timer (WDT) function ensures reliable system operation. The WDT is a hardware mechanism to reset the system if the watchdog timer expires. Users can start the WDT and keep resetting the timer to make sure the system or program is running. Otherwise, the system shall be reset.
In this section, we'll illustrate how to use the function library provided by Neousys to program WDT functions. Currently, WDT driver library supports Windows 7/ 8.1/ 10 32-bit and 64-bit versions. For other OS support, please contact Neousys Technology for further information.

Installing WDT_DIO Library

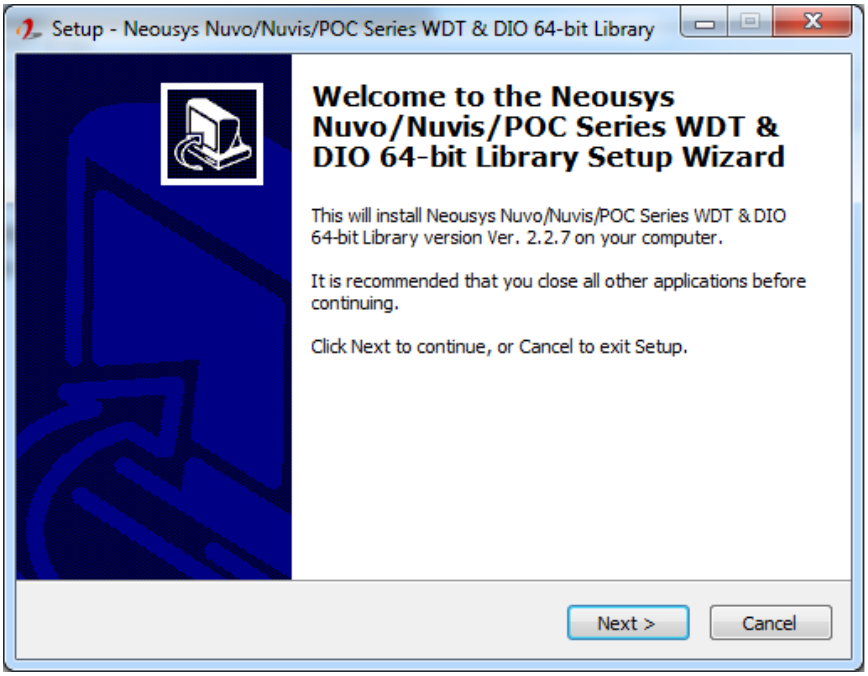
The WDT_DIO function library is delivered in the form of a setup package named **WDT_DIO_Setup.exe**. Prior to programming WDT, you should execute the setup program and install the WDT library. Please use the following WDT_DIO_Setup packages according to your operating system and application.

- For Windows 7/ 8.1/ 10 32-bit OS, please install WDT_DIO_Setup_v2.2.7.9.exe or later version.
- For Windows 7/ 8.1/ 10 64-bit OS with 64-bit application (x64 mode), please install WDT_DIO_Setup_v2.2.7.9(x64).exe or later version.
- For Windows 7/ 8.1/ 10 64-bit OS with 32-bit application (WOW64 mode), please install WDT_DIO_Setup_v2.2.7.9(wow64).exe or later version.

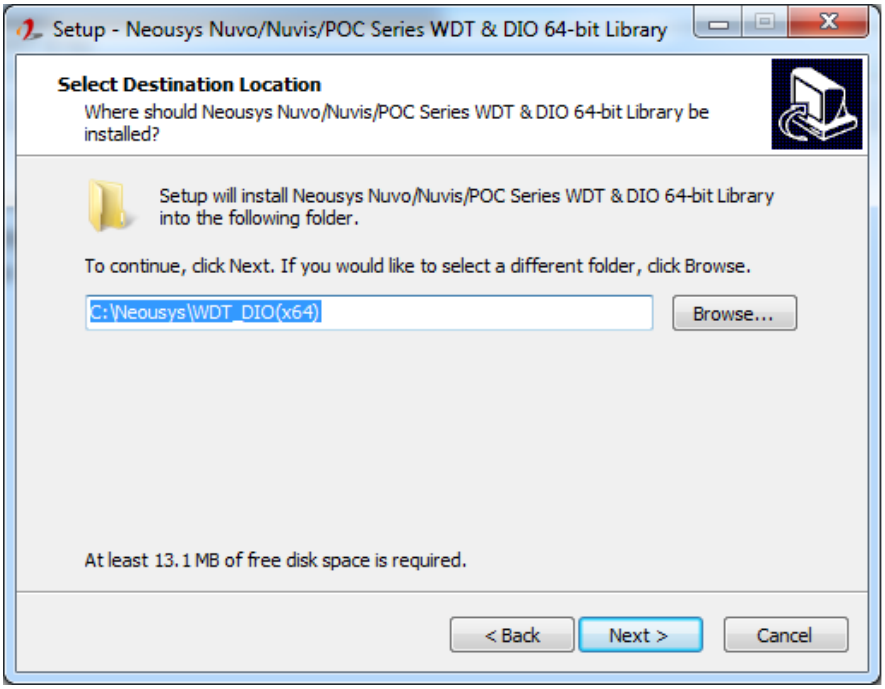
WDT and DIO Library Installation

To setup WDT & DIO Library, please follow instructions below.

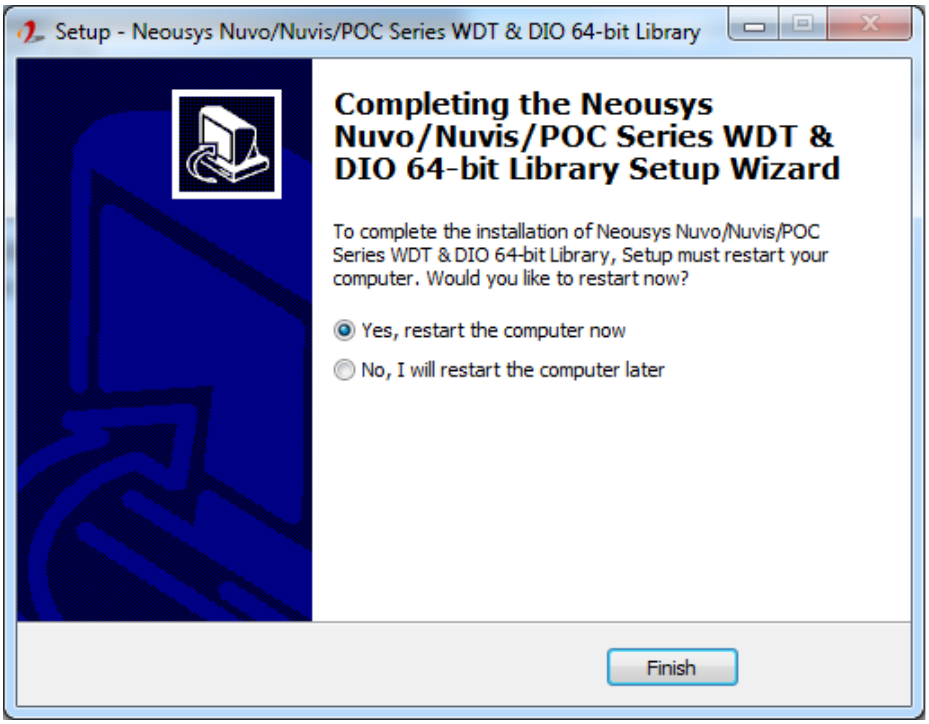
1. Execute **WDT_DIO_Setup.2.2.7.9.exe**. and the following dialog appears.



2. Click “Next >” and specify the directory of installing related files. The default directory is *C:\Neousys\WDT_DIO*.



3. Once the installation has finished, a dialog will appear to prompt you to reboot the system. The WDT & DIO library will take effect after the system has rebooted.



4. When programming your WDT or DIO program, the related files are located in

Header File:	\Include
Library File:	\Lib
Function Reference:	\Manual
Sample Code:	\Sample\WDT_Demo (Demo for Watchdog Timer) \Sample\DIO_Demo (Demo for DIO Control) \Sample\COS_Demo (Demo for change-of-state DI) \Sample\CAN_Demo (Demo for CAN bus manipulation) \Sample\IGN_Demo (Demo for ignition status manipulation) \Sample\POE_Demo (Demo for PoE per-port on/off control)

WDT Function Reference

InitWDT

Syntax	BOOL InitWDT(void);
Description:	Initialize the WDT function. You should always invoke InitWDT() before set or start watchdog timer.
Parameter	None
Return Value	**TRUE:** Successfully initialized **FALSE:** Failed to initialize
Usage	BOOL bRet = InitWDT()

SetWDT

Syntax	BOOL SetWDT(WORD tick, BYTE unit);
Description	Set timeout value and unit for watchdog timer. When InitWDT() is invoked, a default timeout value of 255 seconds is assigned.
Parameter	*tick* WORD value (1 ~ 65535) to indicate timeout ticks. *unit* BYTE value (0 or 1) to indicate unit of timeout ticks. 0 : unit is minute 1: unit is second
Return Value	If value of unit is correct (0 or 1), this function returns TRUE, otherwise FALSE.

Usage	WORD tick=255;
	BYTE unit=1; //unit is second.
	BOOL bRet = SetWDT(tick, unit); //timeout value is 255 seconds

StartWDT

Syntax	BOOL StartWDT(void);
Description	Starts WDT countdown. Once started, the WDT LED indicator will begin blinking. If ResetWDT() or StopWDT is not invoked before WDT countdowns to 0, the WDT expires and the system resets.
Parameter	None
Return Value	If the timeout value is given in correct format (WDT started), this function returns TRUE, otherwise FALSE
Usage	BOOL bRet = StartWDT()

ResetWDT

Syntax	BOOL ResetWDT(void);
Description	Reset the timeout value to the value given by SetWDT().If ResetWDT() or StopWDT is not invoked before WDT countdowns to 0, the WDT expires and the system resets.
Parameter	None
Return Value	Always returns TRUE
Usage	BOOL bRet = ResetWDT()

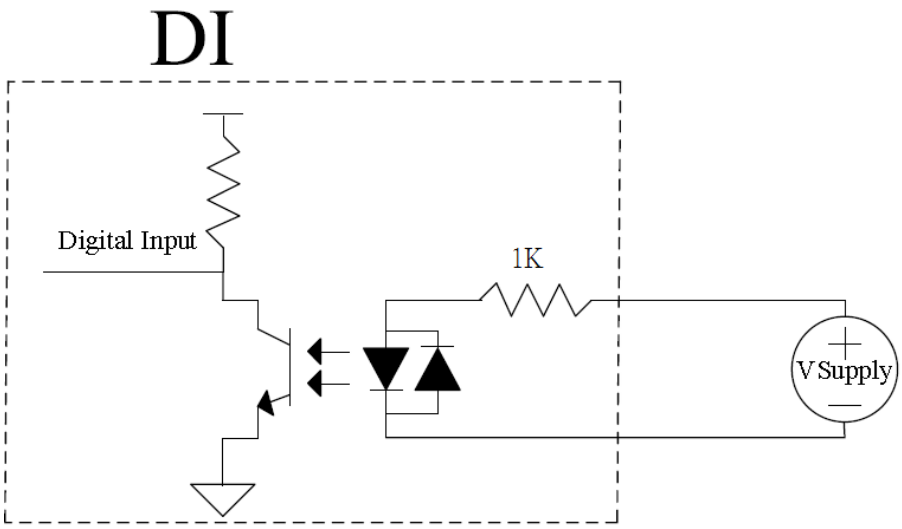
StopWDT

Syntax	BOOL StopWDT(void);
Description	Stops the countdown of WDT. When WDT has stopped, the WDT LED indicator stops blinking.
Parameter	None
Return Value	Always returns TRUE
Usage	BOOL bRet = StopWDT()

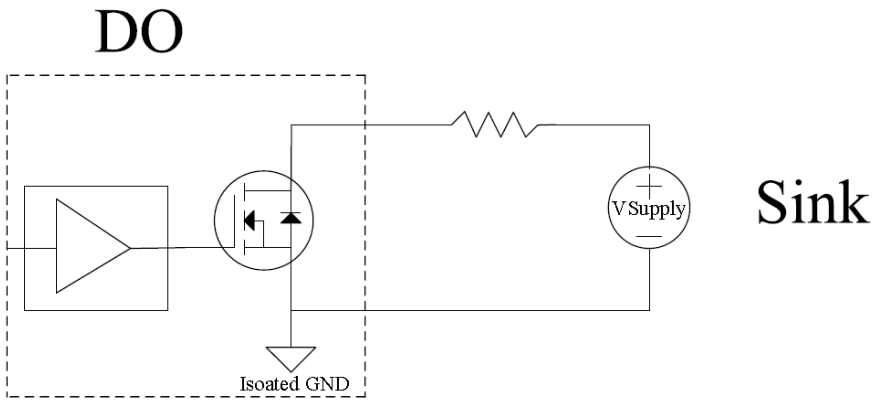
Using DIO Function

Wiring for DIO

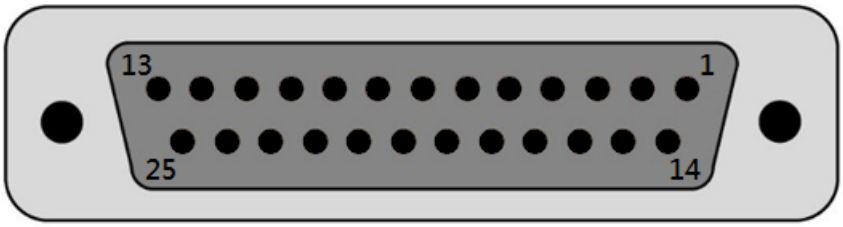
The digital input function of System series is implemented using a photo-coupler with a internally series-connected 1kΩ resistor. You need to provide a voltage to specify the logic high/low state. The input voltage for logic high is 5~24V, and the input voltage for logic low is 0~1.5V.



The digital output function of System series is implemented using Power MOSFET + Analog Device iCoupler® component. The DO channels are configured as NO (normally-open) configuration. When you turn on system, all DO channels have a deterministic state of logic 0 (circuit disconnected from GND return). When logic 1 is specified, MOSFET is activated and GND return path is established. The digital output function on System series supports sinking current connection. The following diagrams are the suggested wiring for DO:



DIO Pin Definition



Pin No.	Definition	I/O	Description
1	ISO_DI3H	I	Digital input channel 3
2	ISO_DI2H	I	Digital input channel 2
3	ISO_DI1H	I	Digital input channel 1
4	ISO_DI0H	I	Digital input channel 0
5	Reserved	-	Reserved pin. Keep unconnected
6	Reserved	-	Reserved pin. Keep unconnected
7	ISO_DO3	O	Digital output channel 3
8	ISO_DO2	O	Digital output channel 2
9	ISO_DO1	O	Digital output channel 1
10	ISO_DO0	O	Digital output channel 0
11	VDD	-	DO voltage source input for inductive load
12	ISO5V	-	Isolated 5V power supply
13	Reserved	-	Reserved pin. Keep unconnected
14	ISO_DI3L	-	Digital input channel 3 GND
15	ISO_DI2L	-	Digital input channel 2 GND
16	ISO_DI1L	-	Digital input channel 1 GND
17	ISO_DI0L	-	Digital input channel 0 GND
18	Reserved	-	Reserved pin. Keep unconnected
19	Reserved	-	Reserved pin. Keep unconnected
20	DOGND	-	Digital output GND
21	DOGND	-	Digital output GND
22	DOGND	-	Digital output GND
23	DOGND	-	Digital output GND
24	DOGND	-	Digital output GND
25	DOGND	-	Digital output GND

DIO Function Reference

InitDIO

Syntax	BOOL InitDIO(void);
Description:	Initialize the DIO function. You should always invoke InitDIO() before write/read any DIO port/channel.
Parameter	None
Return Value	**TRUE:** Successfully initialized **FALSE:** Failed to initialize
Usage	BOOL bRet = InitWDT()

DIReadLine

Syntax	BOOL DIReadLine(BYTE ch);
Description:	Read a single channel of isolated digital input.
Parameter	*ch* BYTE value specifies the DI channel to be read. *ch* should be a value of 0 ~ 3.
Return Value	The status (TRUE or FALSE) of the specified DI channel.

Usage	BYTE ch=3; //DI channel #3
	BOOL DIChValue = DIReadLine(ch); //read DI channel #3

DIReadPort

Syntax	WORD DIReadPort(void);
Description:	Read the entire isolated digital input port (4 channels).
Parameter	None
Return Value	The status (TRUE or FALSE) of the specified DI channel.
Usage	WORD DIPortValue = DIReadPort ();

DOWriteLine

Syntax	void DOWriteLine(BYTE ch, BOOL value);
Description:	Write a single channel of isolated digital output.
Parameter	*ch* BYTE value specifies the DO channel to be written. *ch* should be a value of 0 ~ 3. *value* BOOL value (TRUE or FALSE) specifies the status of DO channel.
Return Value	None

Usage	BYTE ch=3; //DI channel #3
	BOOL DOChValue=TRUE;
	DOWriteLine(ch, DOChValue); //write DO channel #3 as TRUE

DOWritePort

Syntax	void DOWritePort(WORD value);
Description:	Write the entire isolated digital output port (4 channels).
Parameter	*value* WORD value specifies the status of the DO port. *value* should be a value of 0~15.
Return Value	None
Usage	WORD DOPortValue=0x0C; //1100b DOWritePort(DOPortValue); //write DO port as 1100b

DOWriteLineChecked

Syntax	void DOWriteLineChecked(BYTE ch, BOOL value);
Description:	Write a single channel of isolated digital output and read-back the value of DO register. Note that this function is not returned until the DO register is checked and identical to the written value.
Parameter	*ch* BYTE value specifies the DO channel to be written. *ch* should be a value of 0 ~ 3. *value* BOOL value (TRUE or FALSE) specifies the status of DO channel.
Return Value	None

Usage	BYTE ch=3; //DI channel #3
	BOOL DOChValue=TRUE;
	DOWriteLineChecked(ch, DOChValue); //write DO channel #3 as TRUE

DOWritePortChecked

Syntax	void DOWritePortChecked(WORD value);
Description:	Write the entire isolated digital output port (8 channels) and check it has been done. Note that this function is not returned until the write value has been checked the same with the device registry.
Parameter	*value* WORD value specifies the status of the DO port. *value* should be a value of 0~15.
Return Value	None
Usage	WORD DOPortValue=0x0C; //1100b DOWritePortChecked(DOPortValue); //write DO port as 1100b

COS Function Reference

SetupDICOS

Syntax	BOOL SetupDICOS(COS_INT_SETUP *lpSetup, DWORD cbSetup);
Description	Setup Digital-Input(DI) Change-of-State(COS) interrupt parameters.
Parameter	***lpSetup* [in]** A pointer to a **COS_INT_SETUP** structure that contains the COS configuration information for the DI device. This data structure contains the following variables: ***portMask*** WORD value specifies the interrupt mask for corresponding channel(s). ***edgeMode*** WORD value specifies that interrupt is generated when level change (set to 0) or on rising/falling edge (set to 1) for the corresponding channel(s). ***edgeType*** WORD value specifies that interrupt is generated on rising edge (set to 0) or falling (set to 1) edge for corresponding channel(s). This value is neglected if *edgeMode* is set to 0 for the corresponding channel(s). ***cbSetup* [in]** The length of the structure, in bytes. The caller must set this member to size of (COS_INT_SETUP).
Return Value	**TRUE** if setup successes **FALSE** if setup failed

Usage	COS_INT_SETUP setup;
	memset(&setup, 0, sizeof(setup));
	setup.portMask = 0x0f; // enable ch.0~3
	setup.edgeMode = 0; // level
	setup.edgeType = 0x00; // Lo/Hi
	BOOL bRet = SetupDICOS(&setup, sizeof(setup));

RegisterCallbackDICOS

Syntax	BOOL RegisterCallbackDICOS(COS_INT_CALLBACK callback);
Description:	Registers a callback function, which is called when the DICOS interrupt occurred.
Parameter	***callback*** **[in]** Specifies the callback function. The prototype for this function is descripted as follow. void __stdcall callback_func(COS_INT_CALLBACK_ARG* arg);
Return Value	**TRUE** if setup successes, **FALSE** if setup failed.

Usage	void __stdcall callback_func(COS_INT_CALLBACK_ARG* arg)
	{
	printf("data=0x%02x, flag=0x%02x, seq=%02d\n", arg->portData, arg->intrFlag, arg->intrSeq);
	}
	BOOL bRet = RegisterCallbackDICOS(callback_func);

StartDICOS

Syntax	BOOL StartDICOS(void);
Description	Start DI Change-of-State interrupt
Parameter	None
Return Value	**TRUE** if start procedure successes **FALSE** if start procedure failed
Usage	BOOL bRet = StartDICOS();

StopDICOS

Syntax	BOOL StopDICOS(void);
Description	Stop DI Change-of-State interrupt
Parameter	None
Return Value	**TRUE** if stop procedure successes **FALSE** if stop procedure failed
Usage	BOOL bRet = StopDICOS();

DI COS Example

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "WDT_DIO.h"

//Step 0,     define a Change-of-State Interrupt callback function
void __stdcall callback_function(COS_INT_CALLBACK_ARG* arg)
{
      printf("data=0x%02x, flag=0x%02x, seq=%02d\n",
      arg->portData, arg->intrFlag, arg->intrSeq);
}

int main(int argc, char* argv[])
{
  //Step 1, initialize DIO library by invoking InitDIO()
      if ( ! InitDIO() )
      {
            printf("InitDIO --> FAILED\n");
            return -1;
      }
      printf("InitDIO --> PASSED\n");

  //Step 2, setup Change-of-State Interrupt mask and level/edge mode
      COS_INT_SETUP    setup;
```

```
    memset(&setup, 0, sizeof(setup));

    setup.portMask = 0x0f;          // 00001111b, enable ch.0~3
    setup.edgeMode = 0x00;    // generate interrupt on level change
    setup.edgeType = 0x00;        // rising/falling edge, only effective when
edgeMode = 1

    if ( ! SetupDICOS(&setup, sizeof(setup)) )
    {
        printf("SetupDICOS --> FAILED\n");
        return -2;
    }
    printf("SetupDICOS --> PASSED\n");

 //Step 3, register the callback function
    if ( ! RegisterCallbackDICOS(callback_function) )
    {
        printf("RegisterCallbackDICOS --> FAILED\n");
        return -3;
    }
    printf("RegisterCallbackDICOS --> PASSED\n");

 //Step 4, start the DI Change-of-State Interrupt
    if ( ! StartDICOS() )
    {
        printf("StartDICOS --> FAILED\n");
        return -4;
    }
    printf("StartDICOS --> PASSED\n");

    printf("\npress any key to stop...\n");
    system("pause >nul");

 //Step 5, stop the DI Change-of-State Interrupt operation
    if ( ! StopDICOS() )
    {
        printf("StopDICOS --> FAILED\n");
```

```
            return -5;
        }
        printf("StopDICOS --> PASSED\n");

        printf("\npress any key to exit...\n");
        system("pause >nul");
        return 0;

}
```

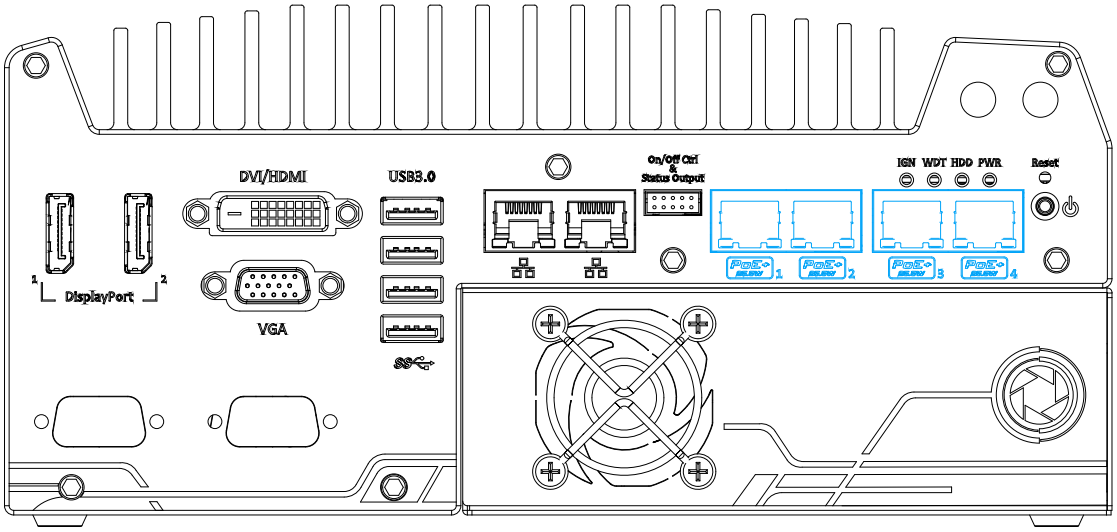
Appendix B: PoE On/ Off Control

The system offers 802.3at PoE+ ports with a unique feature to allow users manually turn on or off the power supply of each PoE port. This can be function can be useful in power device (PD) fault-recovery or power reset.

The function APIs are encapsulated in Neousys WDT_DIO driver package. Please follow the instructions in Appendix B Watchdog Timer & Isolated DIO to install the driver package prior to programming PoE on/off control function.

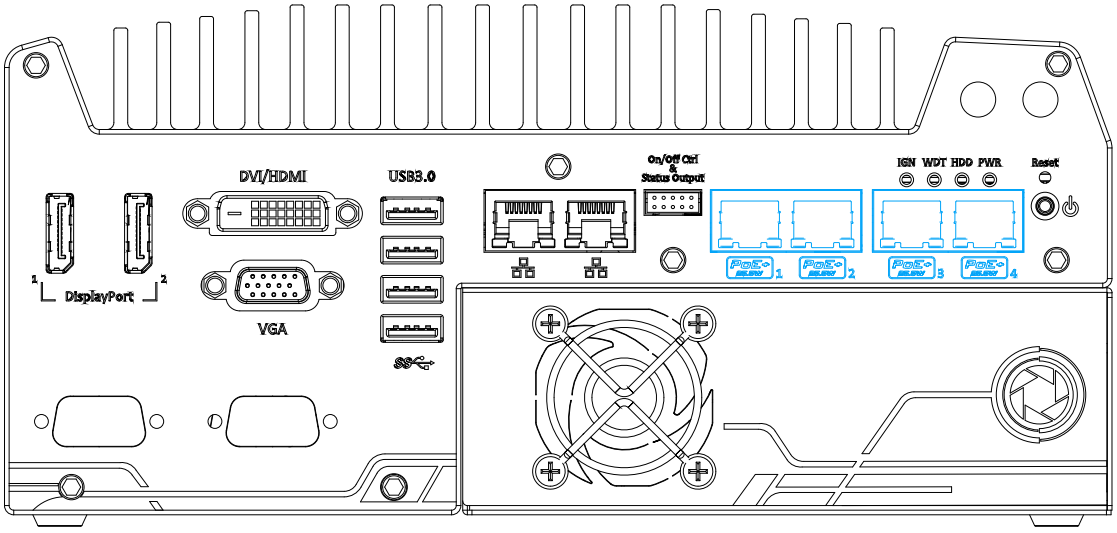
GetStatusPoEPort

Syntax	BYTE GetStatusPoEPort (Byte port);
Description	Get current on/off status of designated PoE port.
Parameter	*port* BYTE value specifies the index of PoE port. Please refer to the following illustration, *port* should be a value of 3 ~ 6.
Return Value	BYTE value indicating PoE on/off status 0 if port is disabled (off) 1 if port is enabled (on)
Usage	BYTE bEnabled = GetStatusPoEPort (3); //Get on/off status of PoE Port#3



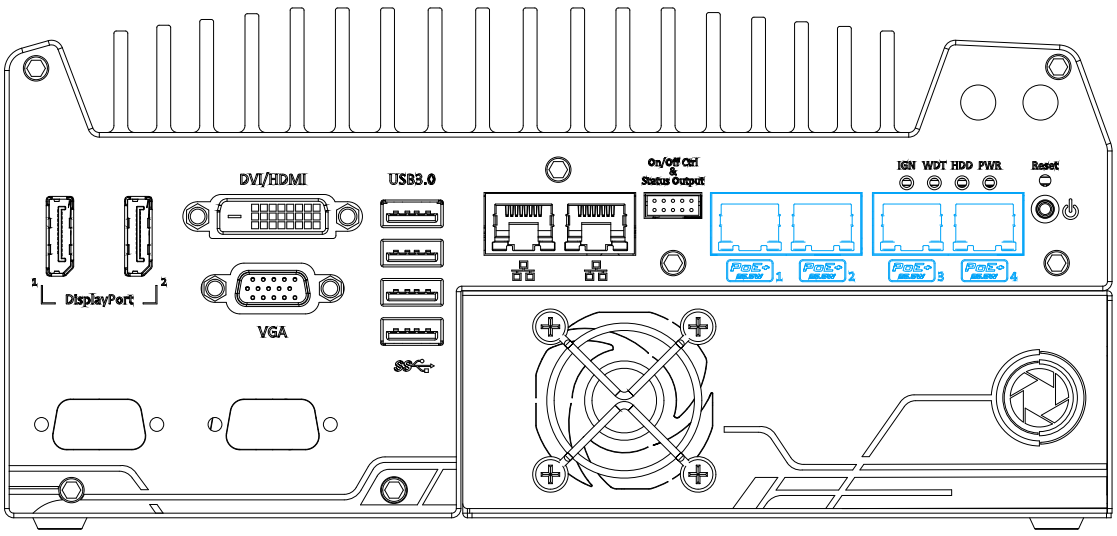
EnablePoEPort

Syntax	BOOL EnablePoEPort (BYTE port);
Description	Turn on PoE power of designated PoE port.
Parameter	*port* BYTE value specifies the index of PoE port. Please refer to the following illustration, *port* should be a value of 3 ~ 6.
Return Value	**TRUE** if enabled success **FALSE** if fail to enable.
Usage	BOOL bRet = EnablePoEPort (3); //Turn on PoE Port#3



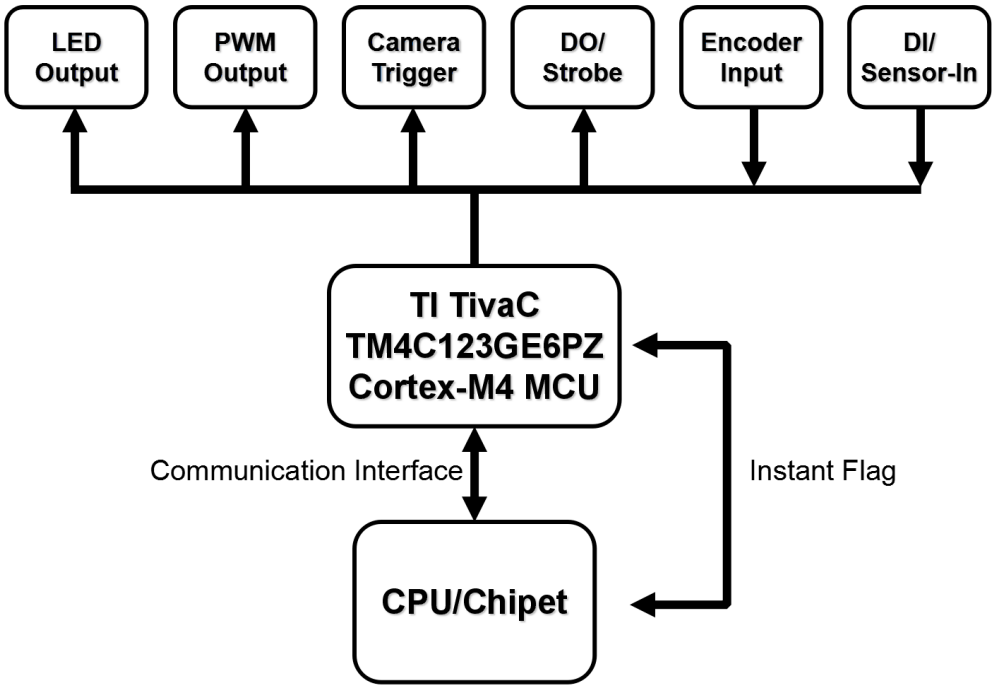
DisablePoEPort

Syntax	BOOL DisablePoEPort (BYTE port);
Description	Turn off PoE power of designated PoE port
Parameter	*port* BYTE value specifies the index of PoE port. Please refer to the following illustration, *port* should be a value of 3 ~ 6
Return Value	**TRUE** if disabled success **FALSE** if fail to disable
Usage	BOOL bRet = DisablePoEPort (3); //Turn off PoE Port#3



Appendix C: NuMCU Programming Guide

Nuvis-5306RT is a state-of-the-art vision controller featuring I/O functions designed for machine vision applications, including LED lighting control, camera trigger, QEI input, PWM output and isolated DIO. To allow comprehensive control of MCU, Nuvis-5306RT offers a new feature, NuMCU, which incorporates MCU programming environment, I/O control library and built-in JTAG debugger, so users can develop user-defined MCU algorithms to achieve more accurate and more complicated I/O control. NuMCU is delivered in the package of abstract I/O access API and a set of sample programs. This document illustrates step-by-step procedures to setup NuMCU programming environment and basic techniques to write/download/debug MCU programs.



Setting Up Your Programming Environment

Install TI Code Composer Studio Version 6.2.0

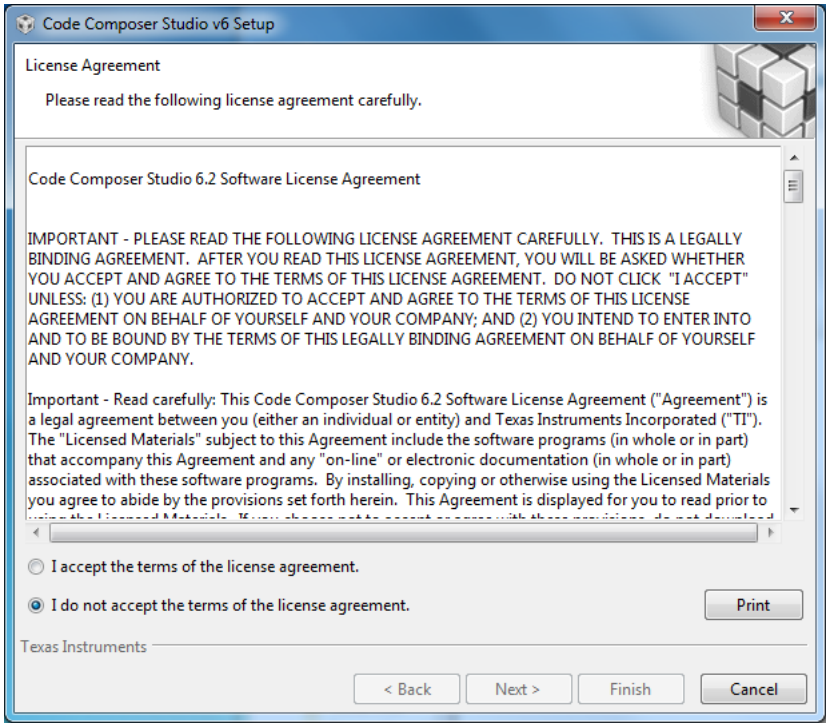
Nuvis-5306RT NuMCU programming library and sample codes are developed based on TI Code Composer Studio version 6.2.0 and TivaWare version 2.1.4.178. We suggest you to duplicate same programming environment on your Nuvis-5306RT-NuMCU to minimize the issues of compatibility and configuration.

To install TI CCS 6.2.0, please follow the steps described below.

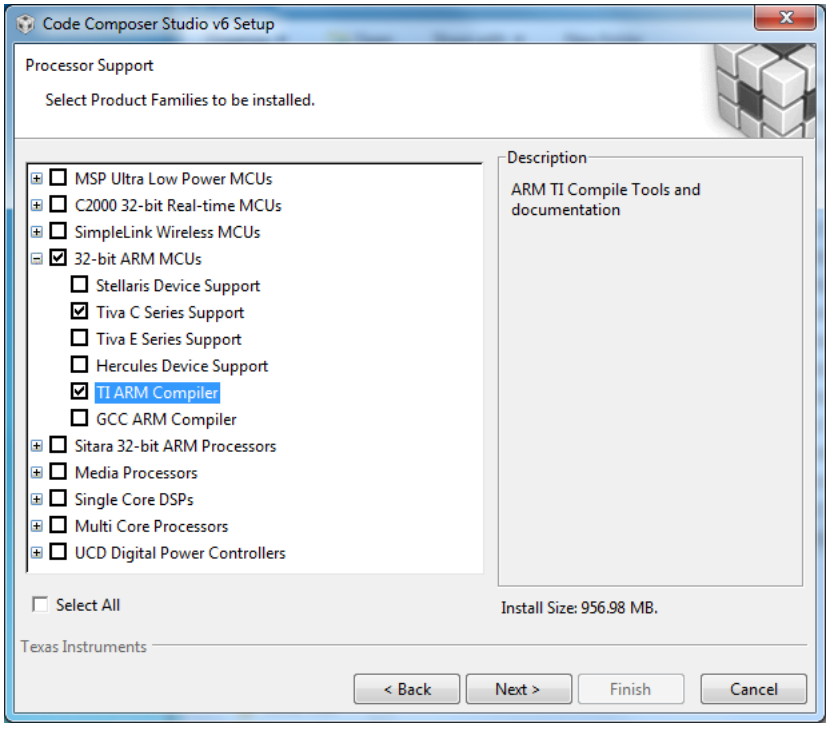
Download TI CCS 6.2.0 from the following link.

http://processors.wiki.ti.com/index.php/Download_CCS#Code_Composer_Studio_Version_6_Downloads

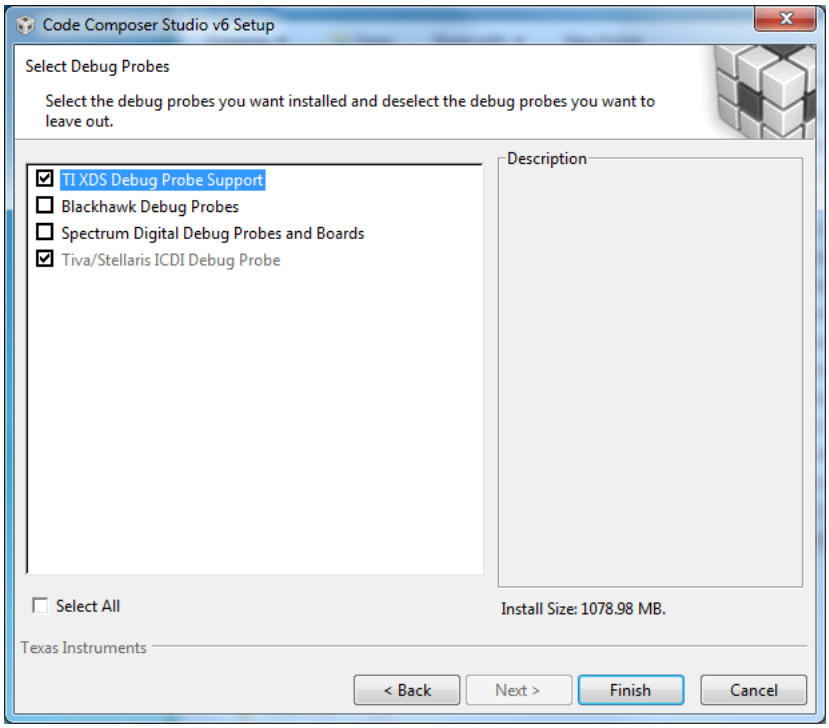
Unzip CCS6.2.0.00050_win32.zip and execute ccs_setup_6.2.0.00050.exe.



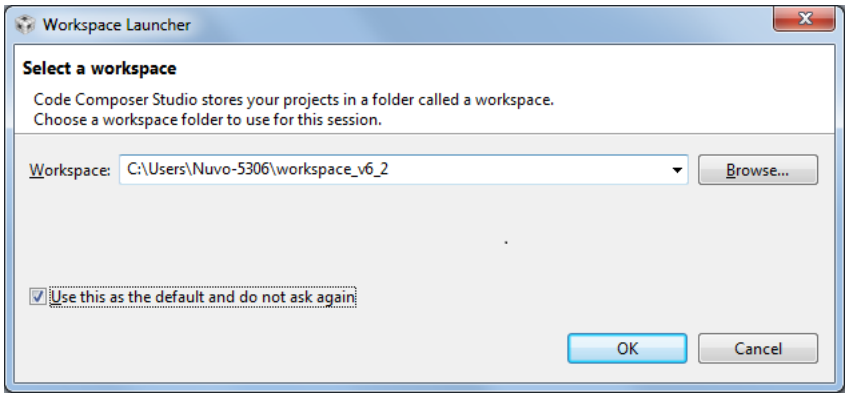
During installation, check “**Tiva C Series Support**” and “**TI ARM Compiler**” to include necessary files for Tiva C MCU.



Also check “**TI XDS Debug Probe Support**” to enable NuMCU binary download and debug support.



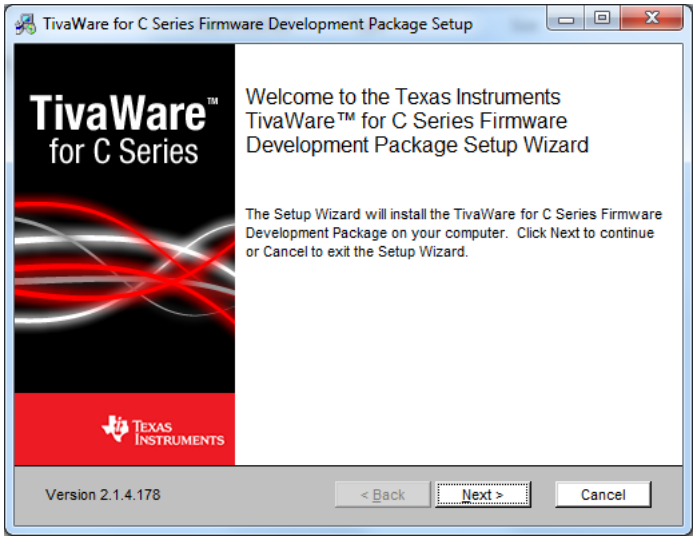
When installation is complete, CCS prompts to select a workspace for all CCS projects. Please use the default workspace (\workspace_v6_2) for all NuMCU sample programs.



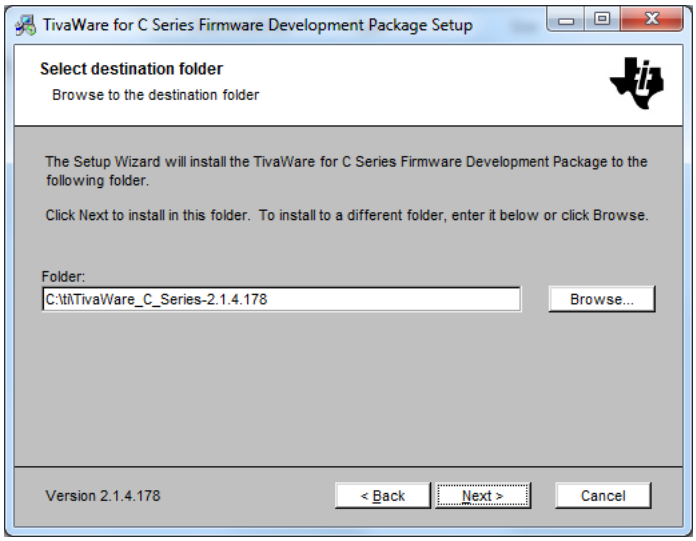
Install TI TivaWare Version 2.1.4.178

In addition to TI Code Composer Studio version 6.2.0, NuMCU programming environment requires TivaWare version 2.1.4.178. We strongly suggest you to install TivaWare of exact the same revision to avoid possible compatibility issue of programming MCU.

To install TivaWare version 2.1.4.178, please follow the steps described below.
Download TivaWare version 2.1.4.178 from the following link.
http://www.ti.com/tool/sw-tm4c
Execute SW-TM4C-2.1.4.178.exe.



Please install TivaWare to its default folder as all NuMCU sample programs are configured to use this default folder.



Install NuMCU Package

NuMCU is delivered in the package of abstract I/O access API and a set of sample programs. The abstract I/O access API (defined in NV5306.h) reduces the learning curve for programmers by encapsulating detailed register-level manipulation into easy-to-use C functions. Sample programs illustrates how to control each I/O function (LED, camera trigger, DIO, encoder and etc.), and some useful techniques of programming MCU (timer, systick, UART communication and etc.).

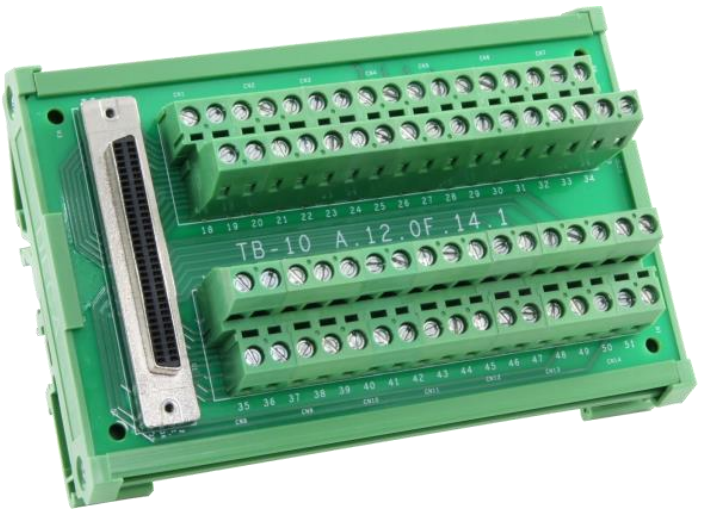
To install NuMCU package, please follow the steps described below

Unzip NuMCU_Package.zip.

Copy all folders into your CCS workspace (refer to section 1.1 step 5)

Totally 8 samples programs are included. And the folder “NuMCU_LIB” contains the header file (NV5306.h) and the static linking library (NV5306.lib).

Nuvis-5306RT Vision-Specific I/O



Pin Assignment on TB-10

Signal		ISO5V		ISO12V		ISOGND	PHA	PHB	ISOGND	DI4L	DI4H	DI5L	DI5H	DI6L	DI6H	DI7L	DI7H
Pin	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
Pin	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Signal		DOGND		DOGND		ISOGND	IDX			DI0L	DI0H	DI1L	DI1H	DI2L	DI2H	DI3L	DI3H
Signal	LED0+	LED0-	LED1+	LED1-	DOGND	DO0	DOGND	DO1	DOGND	DO2	DOGND	DO3	VDD	DOGND	TRIG0	DOGND	TRIG1
Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Pin	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Signal	LED2+	LED2-	LED3+	LED3-	DOGND	DO4	DOGND	DO5	DOGND	DO6	DOGND	DO7	ISO5V	DOGND	TRIG2	DOGND	TRIG3

Vision-Specific I/O Function Description

Signal	Function Description
LED0+/LED0- LED1+/LED1- LED2+/LED2- LED3+/LED3-	**LED driving output** LED0~LED3 are used to directly connect and power LED lights of the vision system. Each channel can be configured to output 24V constant voltage or user-defined, up to 2A constant current to drive either CV or CC LED light using NuMCU library. The LED driving output also supports digital dimming control by adjusting duty cycle from 0 to 100%. When connecting LED lights, wire LED+ to positive polarity (anode) and LED- to negative polarity (cathode). Total power budget for four LED output channels is limited to 80W. Users shall cautiously program the LED outputs and make sure all connected LED lights consume less than 80W at the same time.
DO0 DO1 DO2 DO3	**Isolated digital output (high-current)** DO0~DO3 are open-drained DO channels designed to control external actuator devices, such as relay, valve and motor. Each channel can carry up to 24VDC, 1A

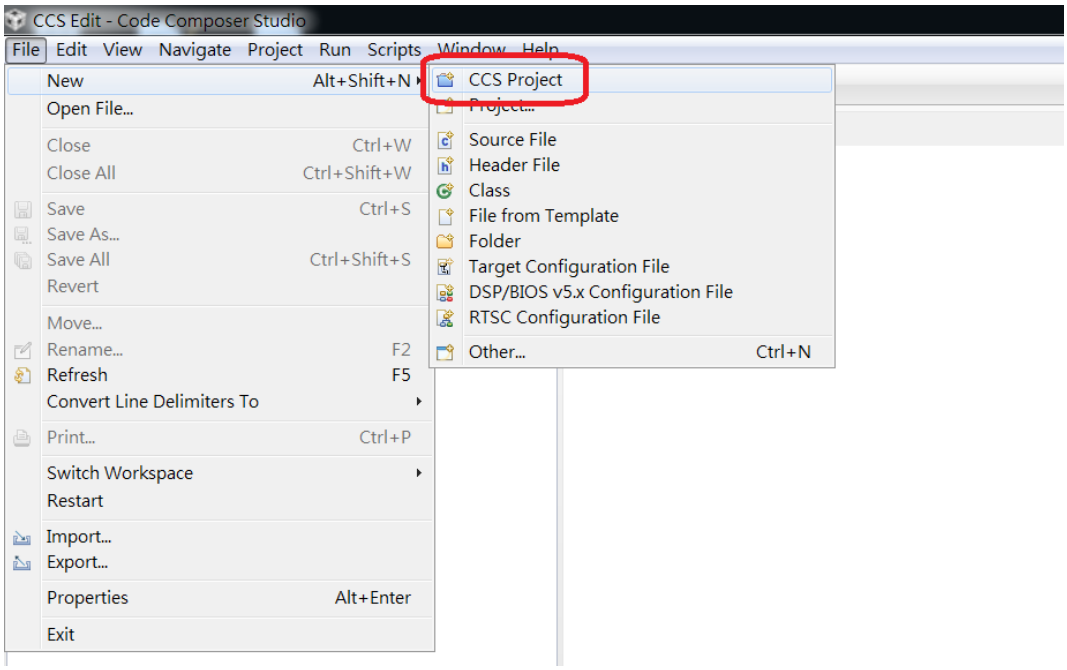
DO4/PWM0 DO5/PWM1 DO6/PWM2 DO7/PWM3	**Isolated digital output (high-speed) or PWM output**
TRIG0 TRIG1 TRIG2 TRIG3	**12V camera trigger output**
DI0H/DI0L DI1H/DI1L DI2H/DI2L DI3H/DI3L DI4H/DI4L DI5H/DI5L DI6H/DI6L DI7H/DI7L	**Isolated digital input**
PHA PHB IDX ISOGND	**Quadrature encoder input**

Program MCU with NuMCU Library

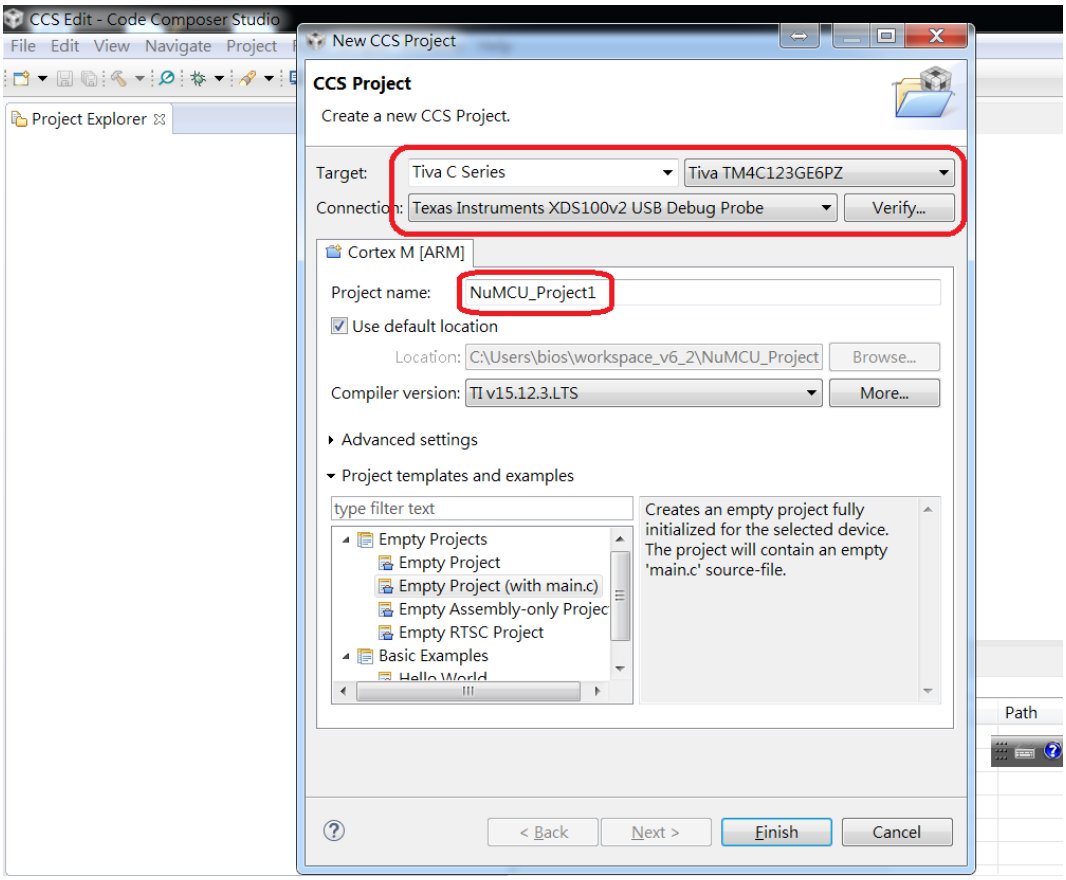
Neousys' NuMCU technology offers a comprehensive way to program real-time I/O operations by combing programming IDE (Code Composer Studio, CCS), JTAG debugger and easy-to-use MCU library. Users can write a C program in CCS, compile it and immediately download it to Nuvis-5306RT's on-board MCU. Benefited by the JTAG debugger, you can even set the breakpoints and debug your code step-by-step. In this chapter, we'll lead you to step-by-step create a program utilizing NuMCU library to control the versatile I/O functions of Nuvis-5306RT.

Create a new CCS Project

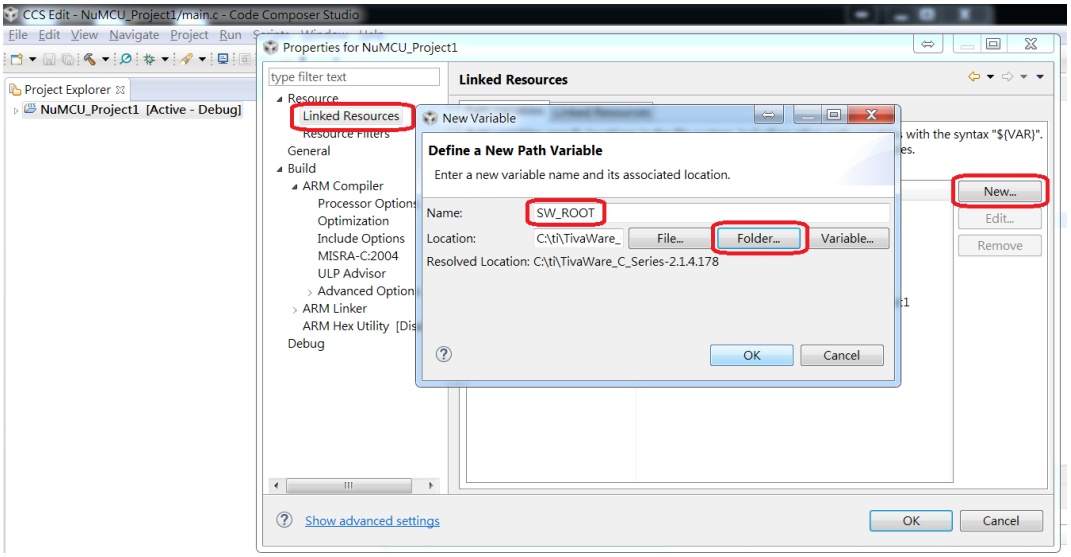
Launch CCS and select [File] → [New] → [CCS Project].



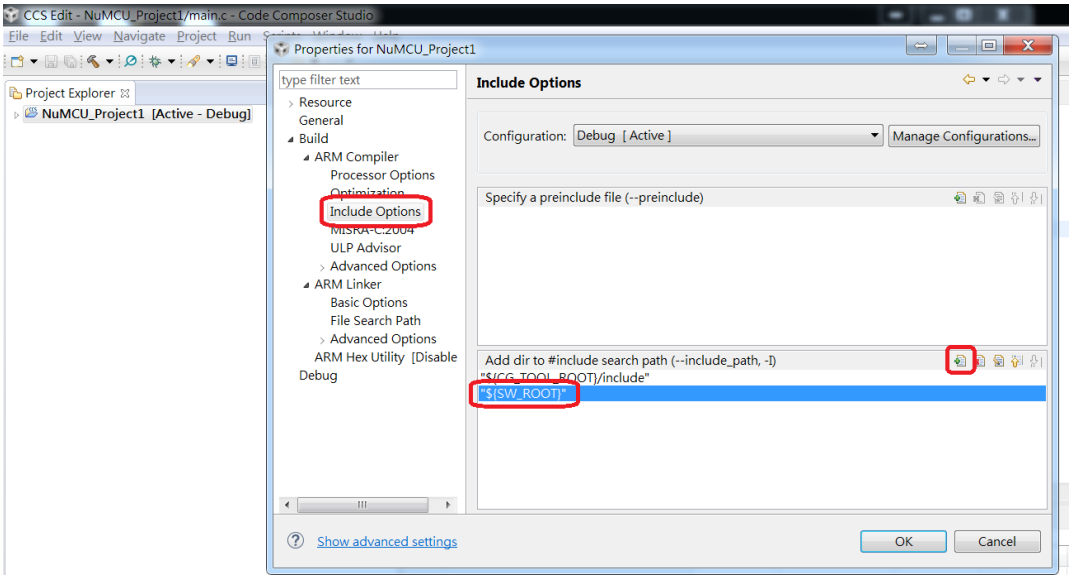
1. In [New CCS Project] dialog, select “Tiva TM4C123GE6PZ” as target device and “Texas Instruments XDS100v2 USB Debug Probe” as debug connection. We also need to input the project name and then click [Finish] to create a new CCS project.

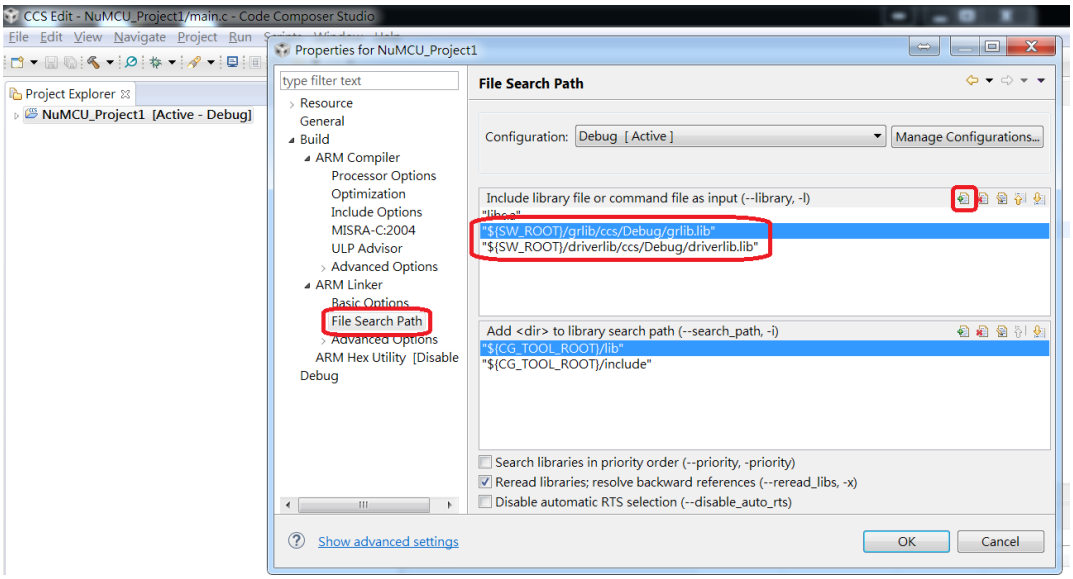


2. After project is created, we need to include the directory of TivaWare (refer to section 1.2). Right-click on project and select [Properties]. In [Properties] dialog, expand [Resource] tag and click on [Linked Resource]. Click on [New...] button and add a new path variable for the directory of TivaWare. Here we enter “SW_ROOT” as variable name and click [Folder...] to select the directory of TivaWare.



3. We also need to include SW_ROOT for ARM Compiler and ARM Linker. Click on [Include Options] under [ARM Compiler] and [File Search Path] under [ARM Linker]. Then add new path(s) for the project as following.



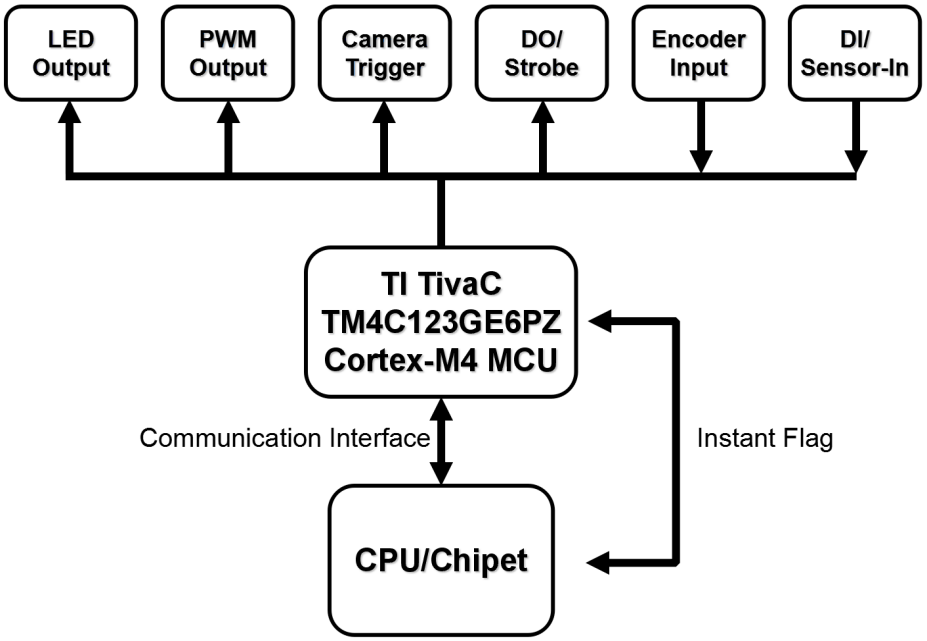


4. Click [OK] to finish setting project properties. Notice that by default the new project is active in debug mode. For release mode, you need to repeat step 3 & 4 to configuration project properties.

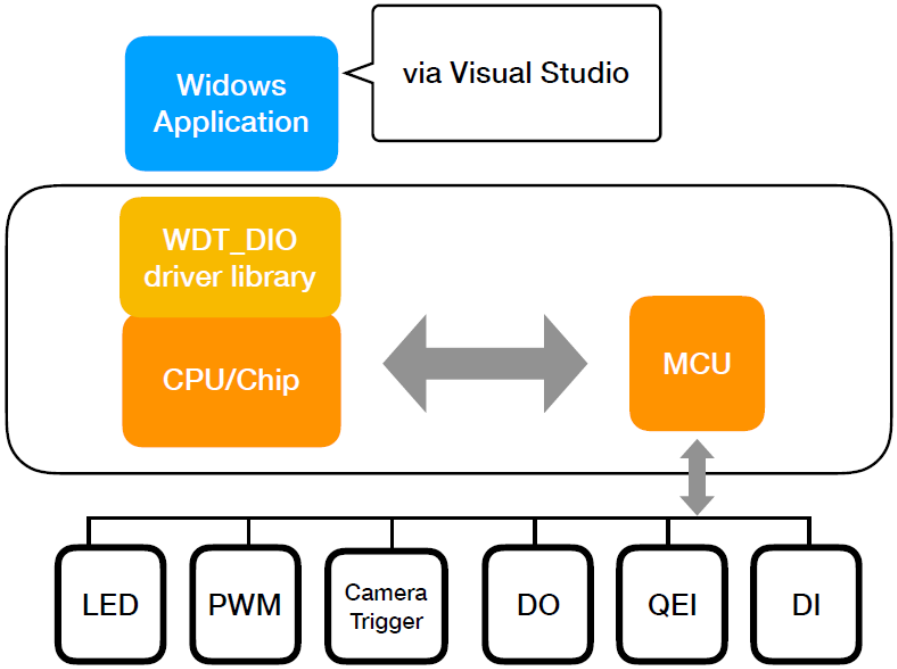
5. Write your NuMCU Programming. Launch CCS and select [File] → [New] → [CCS Project].

Appendix D: DTIOv2 Programming Guide

Nuvis-5306RT is a state-of-the-art vision controller featuring I/O functions designed for machine vision applications, including LED lighting control, camera trigger, QEI input, PWM output and isolated DIO. To allow comprehensive control of MCU, Neousys designs two versions of Nuvis-5306RT. The one is NuMCU version and the other one is DTIOv2 version. This document is prepared for DTIOv2 version.



In DTIOv2 version, users don't need to understand MCU and can develop windows application based on WDT_DIO driver library to control the machine vision system. It is easier to control vision I/O. I will introduce more in the following paragraphs.

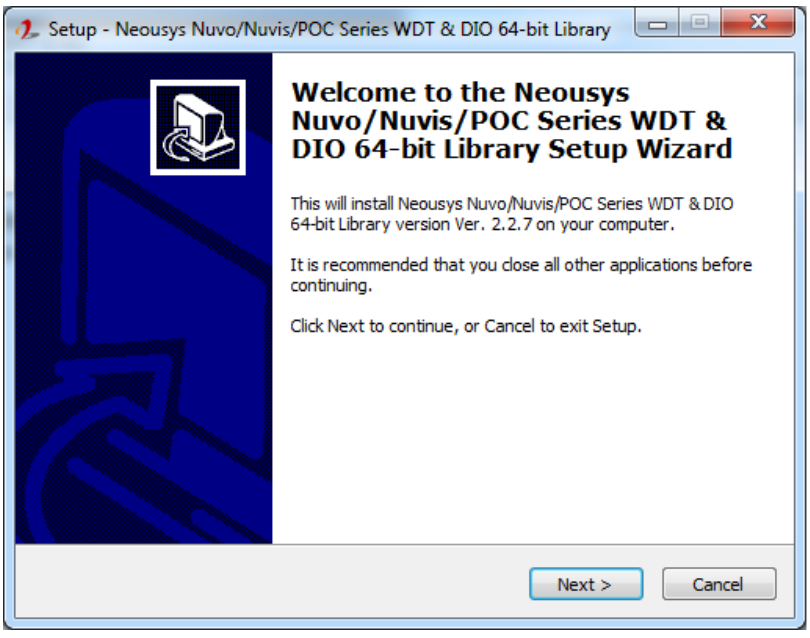


Programming Under Windows Environment

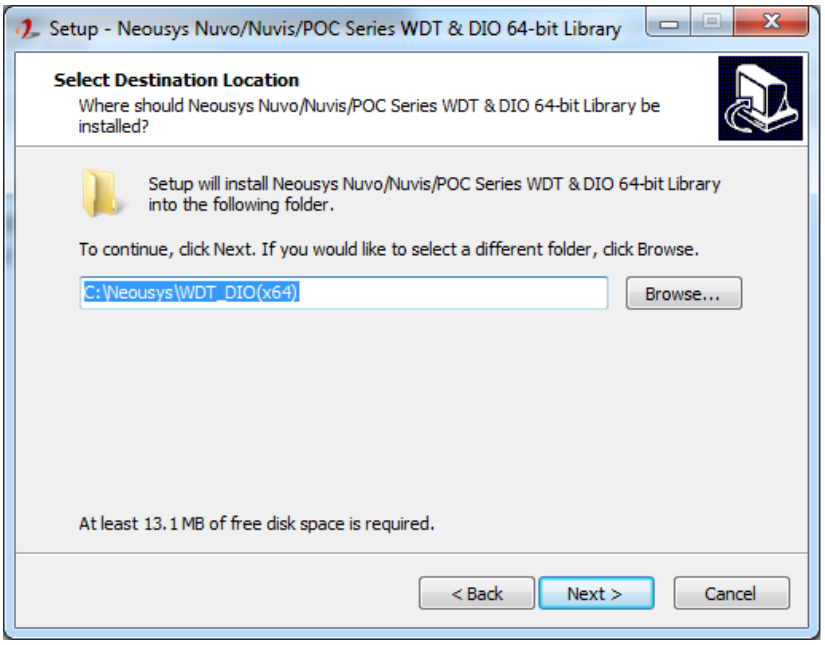
We provide DTIOv2 library so users can program under windows environment easily. Users should setup environment before starting programming.

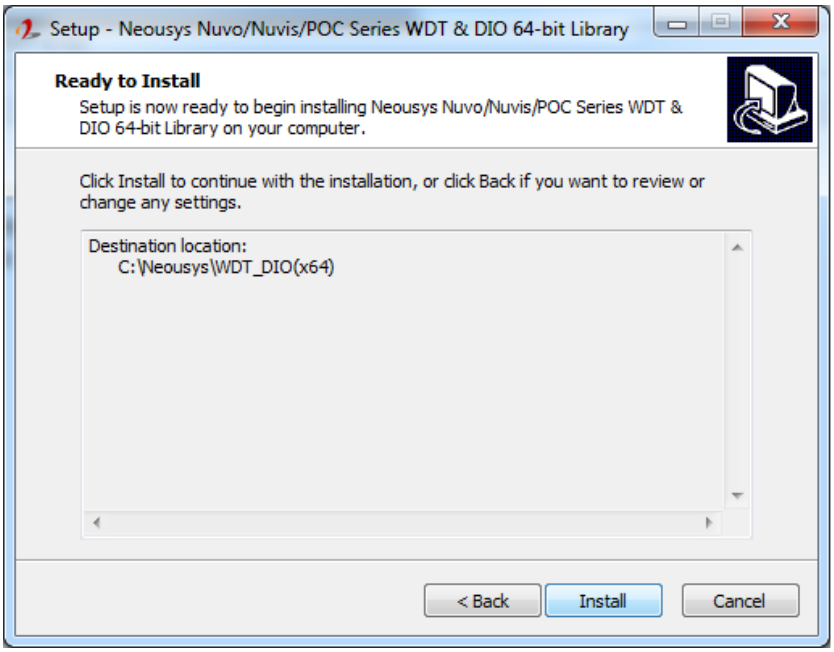
Step 1. WDT_DIO Library Installation

Please download the WDT_DIO_Setup_v2.2.7.9 or later version of WDT_DIO library from our website and install it. Please follow the steps below to complete the installation.

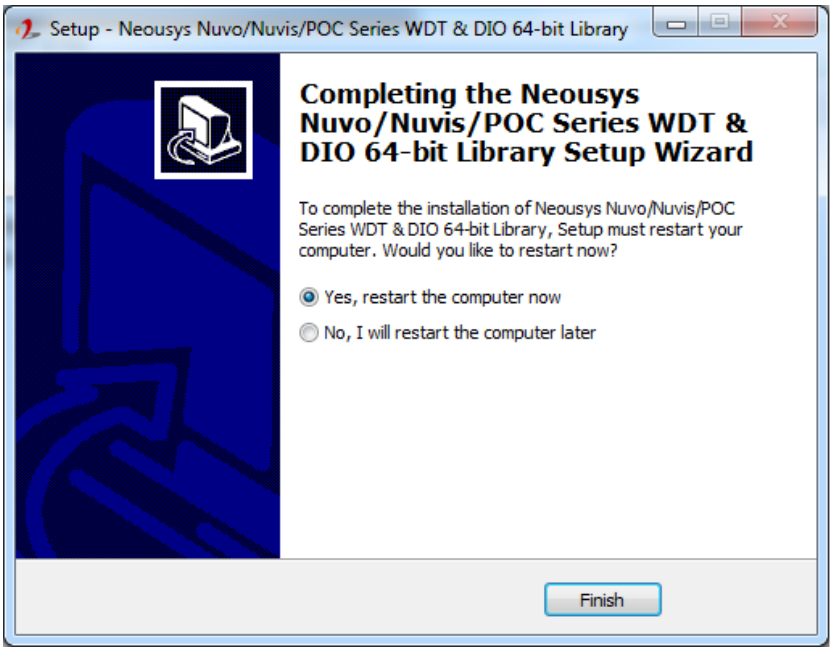


Choose the directory you wish to install WDT_DIO library to.



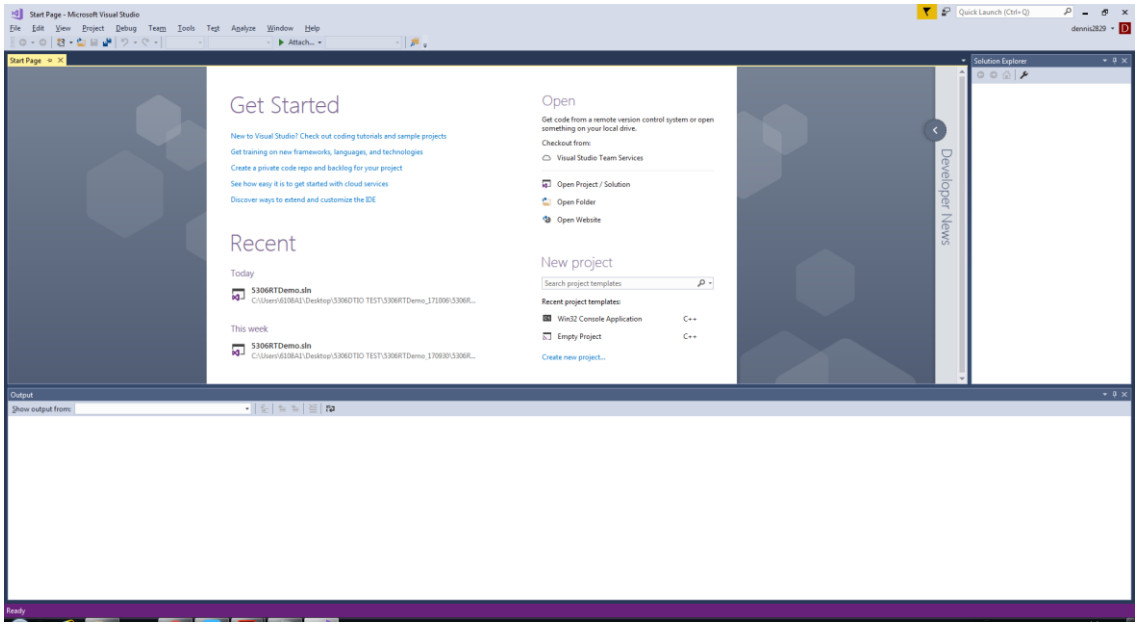


Restart the computer to complete the installation.



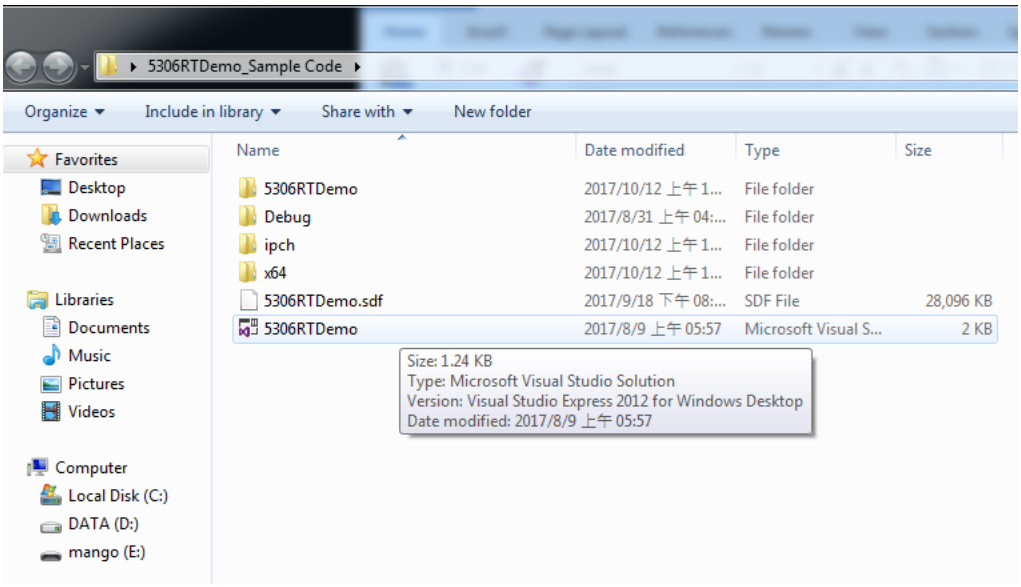
Step 2. Install Integrated Development Environment (IDE)

Choose your preferred integrated development environment (IDE) to install. For demonstration, we will be using Visual Studio 2017 to compile our sample code.



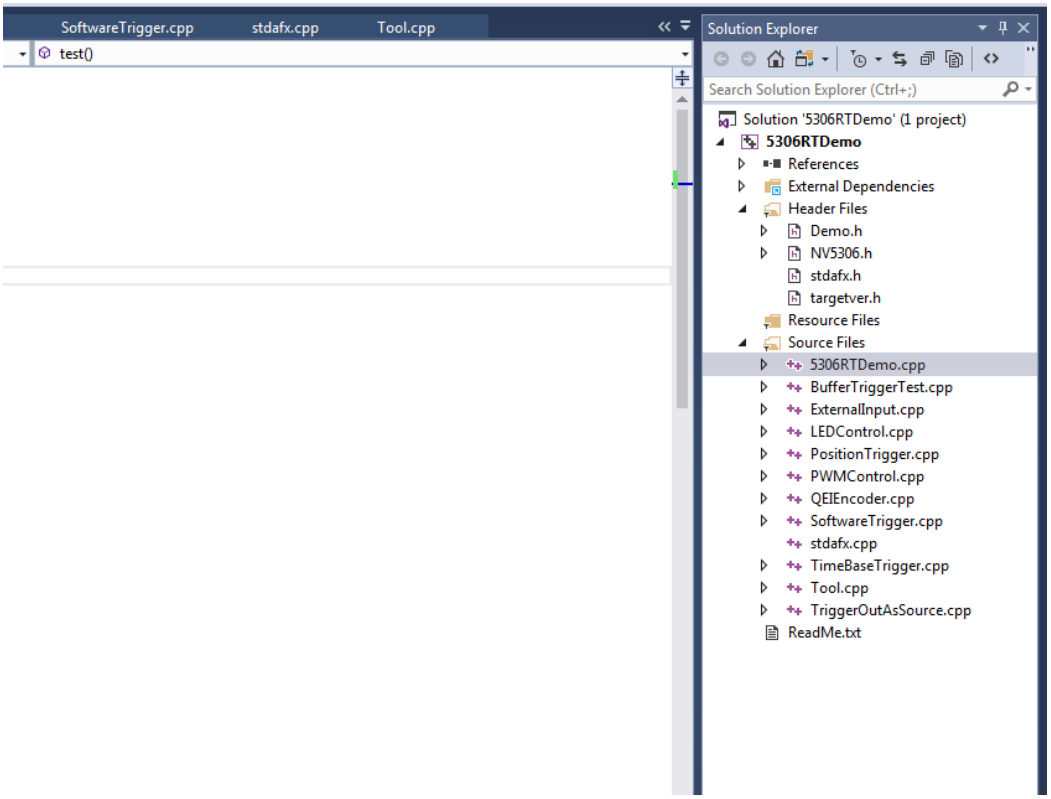
Step 3: Startup Visual Studio

Startup Visual Studio and open the file “5306RTDemo” in the supplied sample code.



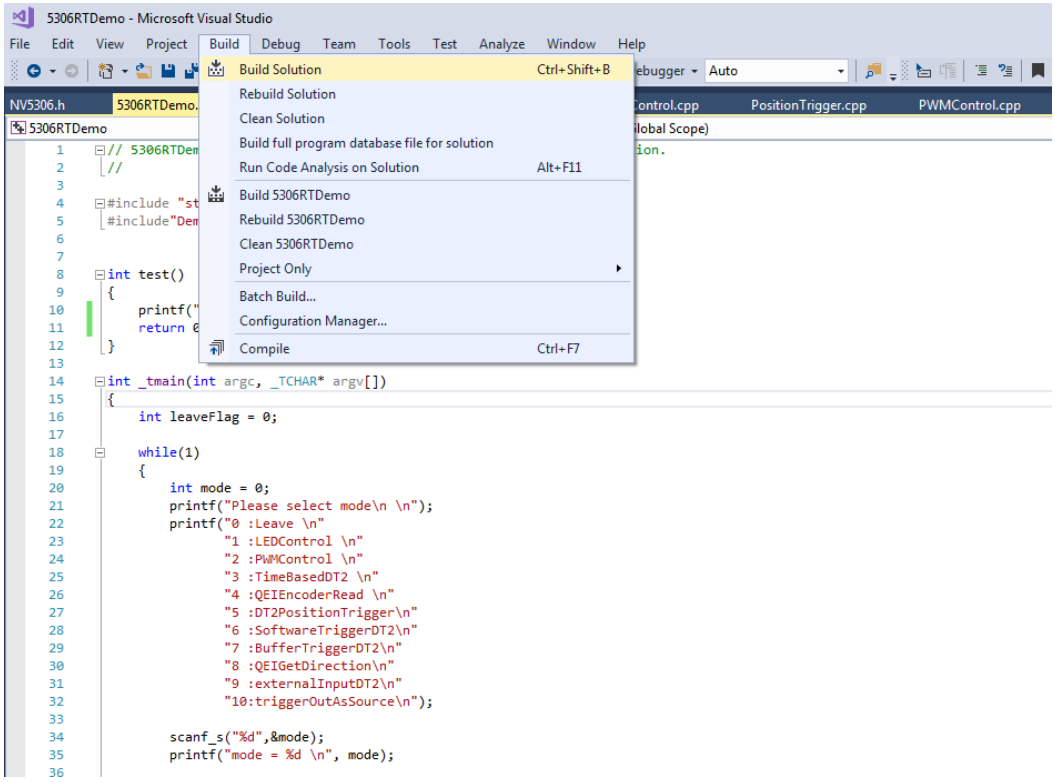
Step 4: Execute 5306RTDemo.cpp file

A “Solution Explorer” window should appear on the right and click on “5306RTDemo.cpp”.



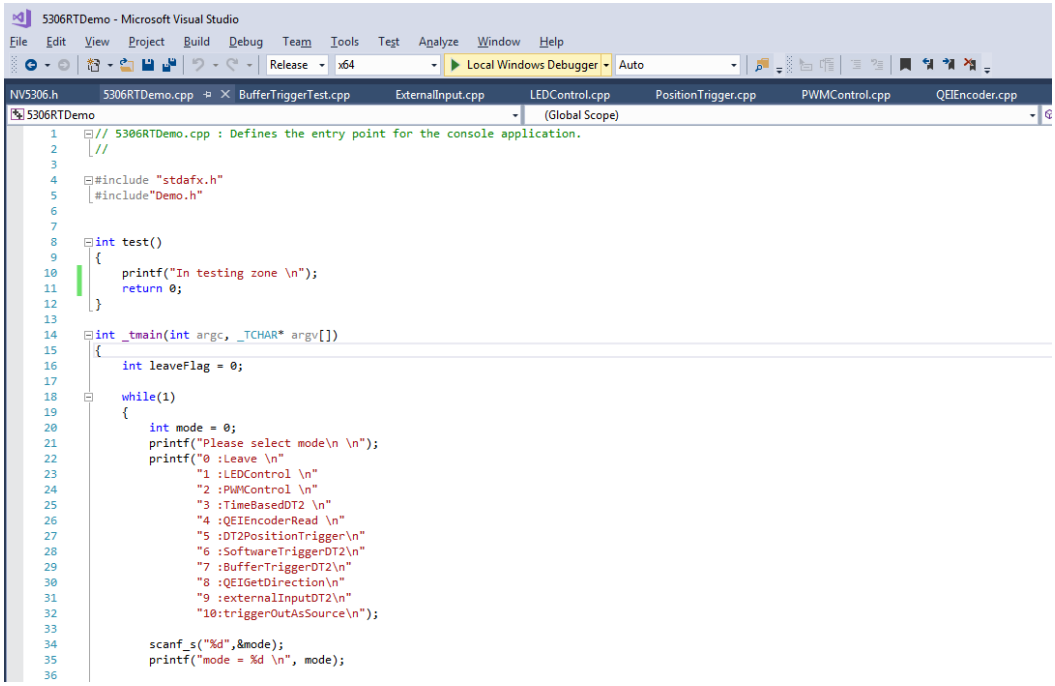
Step 5: Build Solution

Go to “Build > Build Solution”.



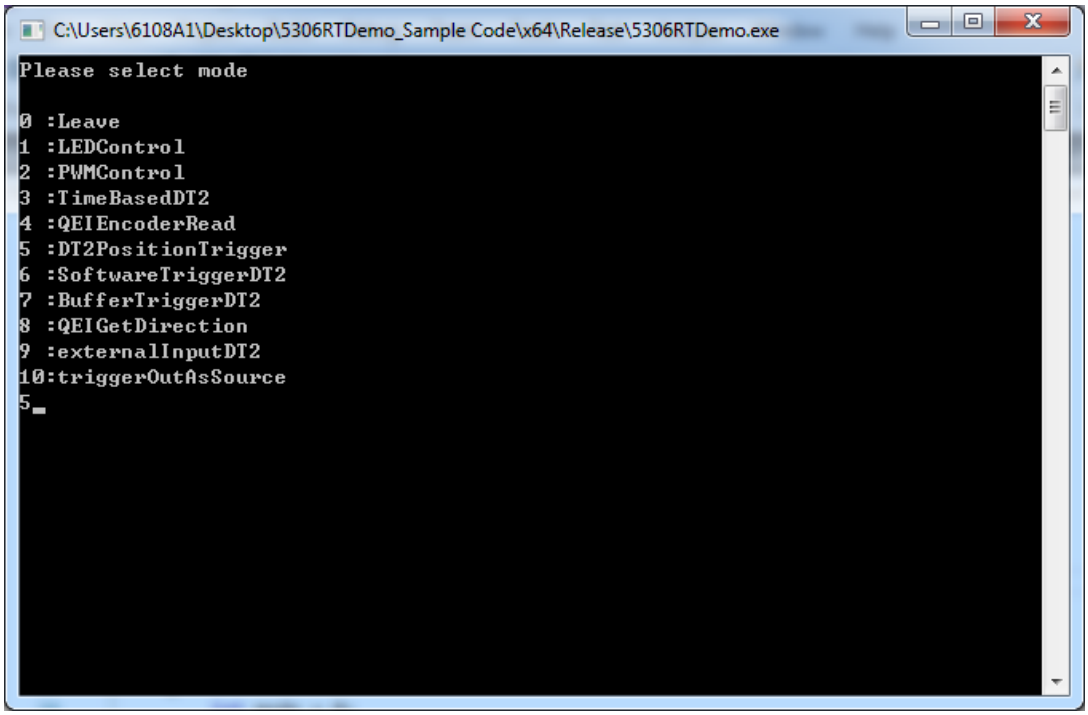
Step 6: Execute Local Windows Debugger

Click on “Local Windows Debugger”



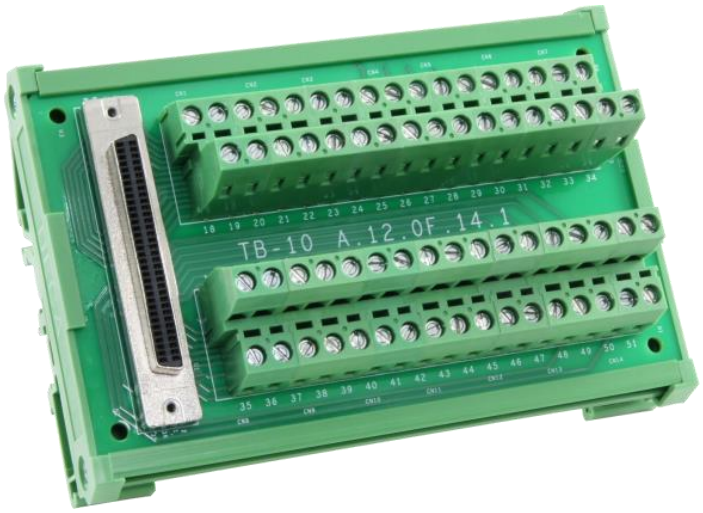
Step 7: Pre-configured Solution

There are 10 sample programs in the pre-configured solution. You can choose the solution you wish to access by entering the number designation.



Nuvis-5306RT Vision-Specific I/O: TB-10 Pin

Assignment



Signal		ISO5V				ISOGND	PHA	PHB	ISOGND	DI4L	DI4H	DI5L	DI5H	DI6L	DI6H	DI7L	DI7H
Pin	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
Pin	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Signal		DOGND				ISOGND	IDX			DI0L	DI0H	DI1L	DI1H	DI2L	DI2H	DI3L	DI3H
Signal	LED0+	LED0-	LED1+	LED1-	DOGND	DO0	DOGND	DO1	DOGND	DO2	DOGND	DO3	VDD	DOGND	TRIG0	DOGND	TRIG1
Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Pin	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Signal	LED2+	LED2-	LED3+	LED3-	DOGND	DO4	DOGND	DO5	DOGND	DO6	DOGND	DO7	ISO5V	DOGND	TRIG2	DOGND	TRIG3

Vision-Specific I/O Function Description

Signal	Function Description
LED0+/LED0- LED1+/LED1- LED2+/LED2- LED3+/LED3-	**LED driving output** LED0~LED3 are used to directly connect and power LED lights of the vision system. Each channel can be configured to output 24V constant voltage or user-defined, up to 2A constant current to drive either CV or CC LED light using NuMCU library. The LED driving output also supports digital dimming control by adjusting duty cycle from 0 to 100%. When connecting LED lights, wire LED+ to positive polarity (anode) and LED- to negative polarity (cathode). Total power budget for four LED output channels is limited to 80W. Users shall cautiously program the LED outputs and make sure all connected LED lights consume less than 80W at the same time.
DO0 DO1 DO2 DO3	**Isolated digital output (high-current)** DO0~DO3 are open-drained DO channels designed to control external actuator devices, such as relay, valve and motor. Each channel can carry up to 24VDC, 1A
DO4/PWM0 DO5/PWM1 DO6/PWM2 DO7/PWM3	**Isolated digital output (high-speed) or PWM output**
TRIG0 TRIG1 TRIG2 TRIG3	**12V camera trigger output**
DI0H/DI0L DI1H/DI1L DI2H/DI2L DI3H/DI3L DI4H/DI4L DI5H/DI5L DI6H/DI6L DI7H/DI7L	**Isolated digital input**
PHA PHB IDX ISOGND	**Quadrature encoder input**

Programming with DTIOv2 Library

Neousys' DTIO technology offers a simple way to program real-time I/O operations with WDT_DIO library. Users can write and run C++ program based on our APIs under windows environment. It will shorten the developing time compared with programming MCU. In this document, we will introduce the concept of DTIOv2 and demonstrate some sample programs to help users easier to use Nuvis-5306RT.

LED Brightness

Users need not purchase additional LED controllers to turn on/ off LEDs as LED drivers are included in our product.

Adjust the Brightness of LED

An API is provided for users to access the LED controller. There are three LED modes to indicate different statuses.

Mode	Description
Constant current (CC)	User can adjust the brightness of LED by changing the constant current intensity. The unit of this variable is mA. The max output current is 2000 mA.
Constant voltage (CV)	We provide constant 24V in this mode and adjust the brightness of LED by modifying duty cycle.
Constant current with duty cycle (CD)	User can adjust the brightness of LED by changing the constant current intensity and modifying duty cycle at the same time.

LED Function Reference

LED_SetCurrentDriving

Syntax	BOOL __cdecl LED_SetCurrentDriving(DWORD mode, DWORD data);
Description	Adjust the brightness of each LED channel
Parameter	**mode [in]** The LED mode is based on a 32 bits parameter. From 16th to 19th bit, are used to assign LED channel. If the 16th bit is set as 1, LED 0 will be configured. If the 17th bit is set as 1, LED1 will be configured, and so on. The second half of 32 bits parameter is used to configure mode. There are 4 LED brightness modes.

Mode	Value
disabled	0x00
constant current	0x01
constant voltage	0x02
constant current with duty cycle	0x03

data [in]

Specifies the value to driving LED. The meaning of data is different in different mode.

Mode	Unit
constant current	mA
constant voltage	Percentage of duty cycle
constant current with duty cycle	31~16bits: mA 15 ~0 bits: Percentage of duty cycle

Return Value	Returns TRUE if operation successful, FALSE if operation failed.

Usage	DWORD pins = (0x00010000)<< 0;
	DWORD mode = (pins) | 0x01;
	if (LED_SetCurrentDriving(mode,100) == false)
	{
	printf("Setting LED failed \n");
	return 0;
	}

LED Example -- Turn On and Off LED

Below is an example on how to turn on and off LED. In the demonstration, the program will turn on LED0 for 10 seconds and then turn off the LED0.

```
int LEDControl ()
{

  //Set to config LED0
  DWORD pins = (0x00010000)<< 0;
  //Set LED Mode as constant current
  DWORD mode = (pins) | LED_MODE_CC;

  // Set constant current as 100 mA. It means that LED is turned on.
  if (LED_SetCurrentDriving(mode,100) == false)
  {
        printf("Setting LED failed \n");
        return 0;
  }

  Sleep(10000);

  // Set constant current as 0 mA. It means that LED is turned off .
  if (LED_SetCurrentDriving(LED_DISABLE,0) == false)
  {
        printf("Setting LED failed \n");
        return 0;
  }

  return 0;

}
```

Quadrature Encoder Interface (QEI)

The Concept of QEI

A Quadrature Encoder is known as a 2-channel incremental encoder. It converts linear /rotation displacement into a pulse signal. You can track the position and the direction of rotation by monitoring the number of pulses and the relative phase of two signals. In addition, the third channel, index channel is used to reset the position counter.

Signal Operation Modes

The QEI module supports two modes of the signal operation, quadrature phase mode and clock/ direction mode. In **quadrature phase mode**, the encoder produces two clock signals that are 90 degrees out of phase. The edge relationship is used to determine the direction of rotation. In **clock/ direction mode**, the encoder produces a clock signal to indicate steps and the other clock signal to indicate the direction of the rotation.

Capture Mode

If you set quadrature phase mode, there are two capture modes user can choose, capture modes A and B. Capture mode A can be set to update the position counter on every edge of phase A. Capture A and B mode can be set to update the position on every edge of phase A and B. For example, if the pulse is 1000 Hz, the first mode will count 2000 times per second and the second mode will count 4000 times per second.

The max position

A quadrature encoder keeps sending pulse signal to MCU and MCU keeps accumulating position counter. When the value of the position counter is equal to the max position, the position counter will reset to zero.

Index channel

Some 2-channel incremental encoders provide additional channel called index channel (z signal). It will send pulse when reset point is reached. If you set “QEI_CONFIG_RESET_IDX” flag, QEI will reset per revolution.

Swap

Two phase signals, phase A and B can be swapped before being interpreted by QEI module in order to change forward or backward definition. It can also be used to correct miss-wiring of the system.

QEI Function Reference

QEI_Setup

Syntax	BOOL __cdecl QEI_Setup(DWORD idx, QEI_SETUP *lpSetup, DWORD cbSetup);
Description	Setup parameters used in the specified QEI controller
Parameter	**idx [in]** Specify the index of QEI controllers. Currently there is only one QEI controller, so always configure 0. **lpSetup [in]** A pointer to a **QEI_SETUP** structure that contains the QEI configuration. This data structure contains the following variables: **config [in]** DWORD value specifies the configuration for the quadrature encoder.

Mode	Value
QEI_CONFIG_CAPTURE_A (0x00000000)	Only count on phase A
QEI_CONFIG_CAPTURE_A_B (0x00000008)	Count on phase A and phase B
QEI_CONFIG_NO_RESET (0x00000000)	Do not reset position on index pulse
QEI_CONFIG_RESET_IDX (0x00000010)	Reset position on index pulse
QEI_CONFIG_QUADRATURE (0x00000000)	Phase A and phase B are quadrature
QEI_CONFIG_CLOCK_DIR (0x00000004)	Phase A and phase B are clock and direction
QEI_CONFIG_NO_SWAP (0x00000000)	Do not swap phase A and phase B
QEI_CONFIG_SWAP (0x00000002)	Swap phase A and phase B
QEI_CONFIG_CAPTURE_A (0x00000000)	Only count on phase A

	maxPos [in] DWORD value specifies the maximum position value. **velPeriod [in]** DWORD value specifies the number of clock ticks over which to measure the velocity. Set 0 value to disable velocity function. **velPreDiv [in]** DWORD value specifies the pre-divider applied to the input quadrature signal before it is counted. ***cbSetup* [in]** The length of the structure, in bytes. The caller must set this parameter to size of (QEI_SETUP).
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	QEI_SETUP QSetup;
	memset(&QSetup, 0, sizeof(QSetup));
	QSetup.config = QEI_CONFIG_CAPTURE_A_B | QEI_CONFIG_RESET_IDX | QEI_CONFIG_QUADRATURE | QEI_CONFIG_NO_SWAP;
	QSetup.maxPos = 4000;
	QSetup.velPeriod = 0;
	QSetup.velPreDiv = QEI_VEL_DIV_1;
	
	if (!QEI_Setup(0, &QSetup, sizeof(QSetup)))
	{
	printf("QEI_Setup failed \n");
	return 0;
	}

QEI_Start

Syntax	BOOL __cdecl QEI_Start(DWORD idx);
Description	Startup QEI controller operation.
Parameter	***idx* [in]** Specifies the index of QEI controller. By default, there is only one QEI controller and therefore, always configure 0.
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	// Start QEI controller
	if (! QEI_Start(0))
	{
	printf("QEI_Start failed \n");
	return 0;
	}

QEI_Stop

Syntax	BOOL __cdecl QEI_Stop(DWORD idx);
Description	Stops QEI controller operation
Parameter	**idx [in]** Specifies the index of QEI controller. By default, there is only one QEI controller and therefore, always configure 0.
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	// Stop QEI controller
	if (! QEI_Stop(0))
	{
	printf("QEI_Stop failed \n");
	return 0;
	}

QEI_GetDirection

Syntax	BOOL __cdecl QEI_GetDirection(DWORD idx, DWORD *lpDirection);
Description	This function returns the rotation direction to the last memorized rotation direction setting. NOTE The last memorized encoder rotation direction may not be the current rotation direction.
Parameter	**idx [in]** Specifies the index of QEI controller. By default, there is only one QEI controller and therefore, always configure 0. **lpDirection [out]** Points to the DWORD value that specifies the current rotation direction. 1: If moving in the forward direction. -1: if moving in the reverse direction.
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	DWORD dir = 0;
	if (! QEI_GetDirection(0, &dir))
	{
	printf("QEI_GetDirection failed \n");
	return 0;
	}

QEI_GetPosition

Syntax	BOOL __cdecl QEI_GetPosition(DWORD idx, DWORD *lpPosition);
Description	This function acquires the current position of the encoder.
Parameter	**idx [in]** Specifies the index of QEI controller. By default, there is only one QEI controller and therefore, always configure 0. **lpDirection [out]** Points to the DWORD value that specifies current position of the encoder.
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	DWORD dwPos = 0;
	if (! QEI_GetPosition(0, &dwPos))
	{
	printf("QEI_GetPosition failed \n");
	return 0;
	}

QEI_SetPosition

Syntax	BOOL __cdecl QEI_SetPosition(DWORD idx, DWORD position);
Description	This function sets the current position of the encoder.
Parameter	**idx [in]** Specifies the index of QEI controller. By default, there is only one QEI controller and therefore, always configure 0. **lpDirection [out]** Value specifies current position of the encoder.
Return Value	Returns TRUE if setup successful, FALSE if setup failed.

Usage	if (! QEI_SetPosition(0, 0))
	{
	printf("QEI_SetPosition failed \n");
	return 0;
	}

QEI Example 1-- Read Direction and Position

```
int QEIEncoderSample()
{
  // Config the data structure of QEI
  // Capture phase A and phase B signal, reset position when encounter index, don't
swap phase        A and phase B
  // Set max position as 4000
  QEI_SETUP QSetup;
  memset(&QSetup, 0, sizeof(QSetup));
  QSetup.config = QEI_CONFIG_CAPTURE_A_B | QEI_CONFIG_RESET_IDX |
  QEI_CONFIG_QUADRATURE | QEI_CONFIG_NO_SWAP;
  QSetup.maxPos = 4000;
  QSetup.velPeriod = 0;
  QSetup.velPreDiv = QEI_VEL_DIV_1;

  if (!QEI_Setup(0, &QSetup, sizeof(QSetup)))
  {
        printf("QEI_Setup failed \n");
        return 0;
  }

  // Start QEI controller
  if (!QEI_Start(0))
  {
        printf("QEI_Start failed \n");
        return 0;
  }


  for(int i = 0; i< 30; i++)
  {
        DWORD    dwPos = 0;

        // read the Direction of the rotation
        if (!QEI_GetDirection(0, &dwPos))
        {
```

```
            printf("GetDirection fail \n");
            return 0;
        }
        printf("direction : %d \n", dwPos);


        // read the position
        if (!QEI_GetPosition(0, &dwPos))
        {

            printf("GetPosition fail \n");
            return 0;
        }

        printf("Position : %d \n", dwPos);

        //Sleep 300 ms
        Sleep(300);

    }

    // Start QEI controller
    if (!QEI_Stop(0))
    {
        printf("QEI_Stop failed \n");
        return 0;
    }



    return 0;
}
```

Deterministic Trigger I/O version 2 (DTIOv2)

The Concept of DTIOv2

Trigger target

The trigger target is defined as the output signal that can control other equipment, such as LED, camera and motor.

Trigger source

The trigger source is defined as the input signal that can trigger an output signal. Digital and encoder inputs are the trigger sources in DTIOv2. In addition, all trigger targets can also be configured as a trigger source.

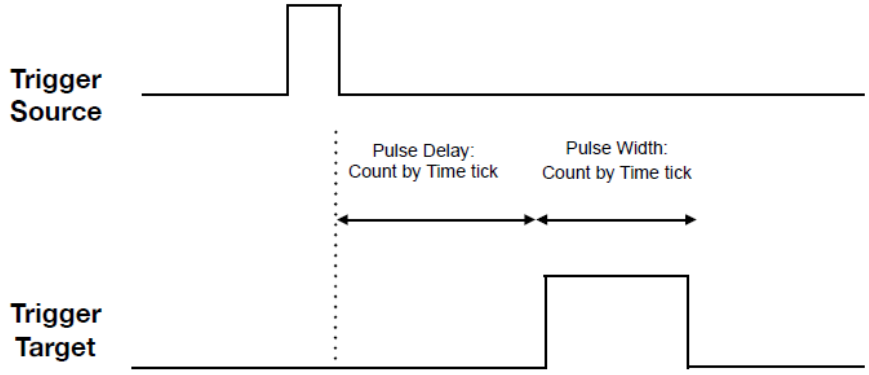
Channel

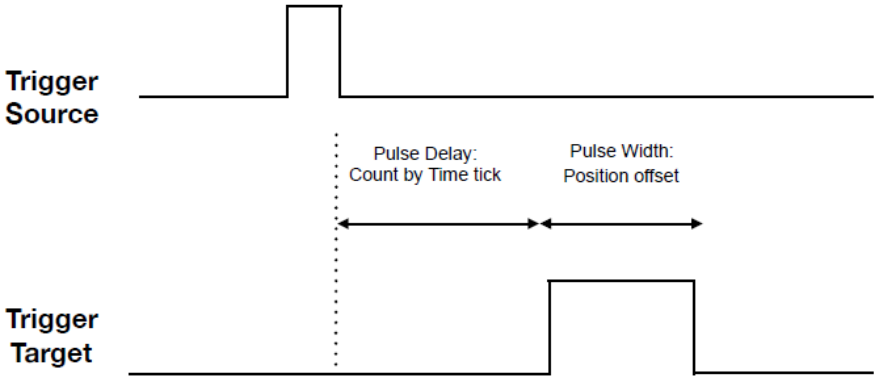
Due to comprehensive input & output mappings, we designed “channel” to simplify the relationship of the trigger source and the trigger target. It is one-to-one mapping. There is exactly one trigger source and one trigger target in each channel.

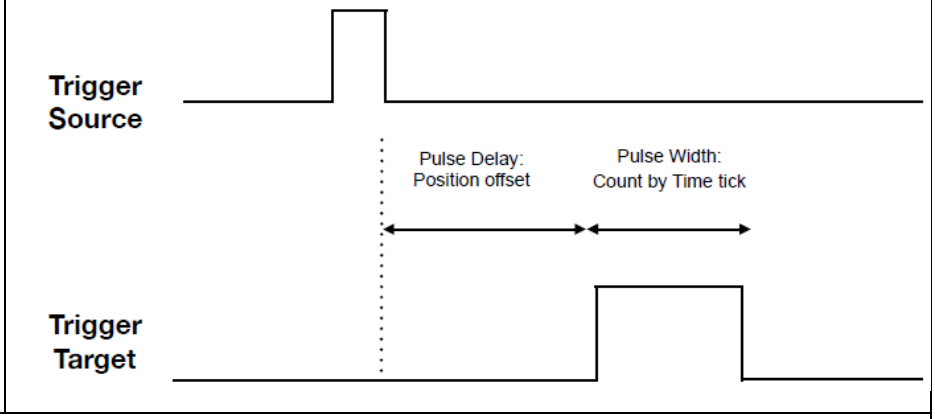
Max quantity of channels is eight in the DTIOv2 library, so users can define eight trigger sources and trigger target mappings at most. Trigger source can be configured in multiple channels but trigger target only can be configured in one channel. The trigger and target relationship is defined in the following paragraph.

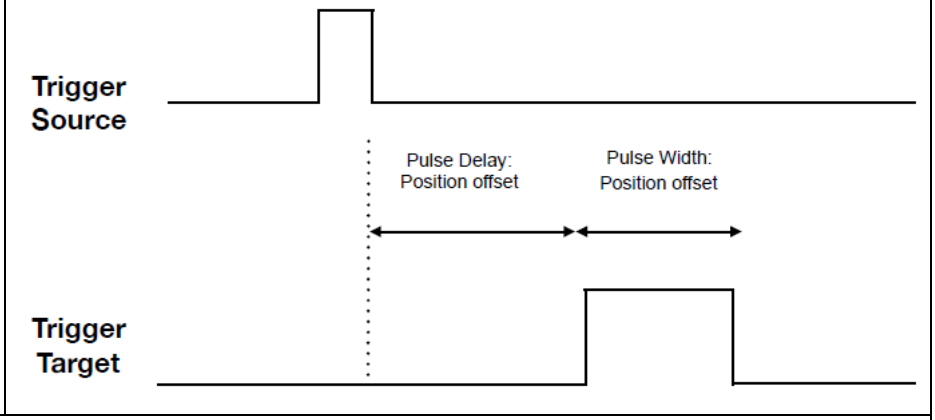
Four types of output signals

The status of output signal can be described by the pulse delay and the pulse width. In DTIO version 1, time is the only unit to tell the distance of pulse delay and pulse width. It is simple but is not accurate enough due to unstable signal conveyance speed. In order to improve the accuracy of DTIO system, we introduce QEI in DTIOv2. Now real displacement can be used to describe the distance of pulse delay and pulse width. Therefore, 2 types of unit and 2 pulse sections can create 4 modes for users to control output signals.

Mode	Illustration/ Description
PTT	
Output pulse with time delay and time width	

PTP	
Output pulse with time delay and position offset	

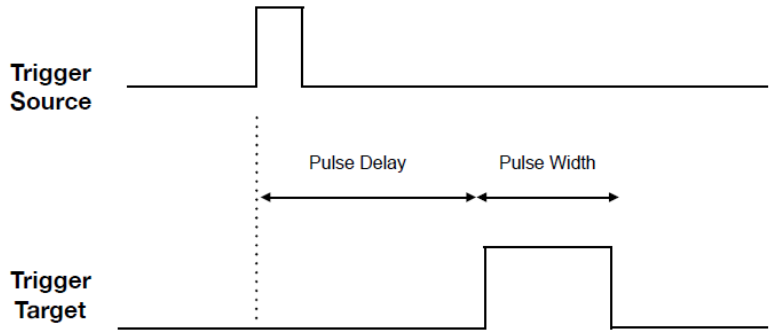
PPT	
Output pulse with position offset and time width	

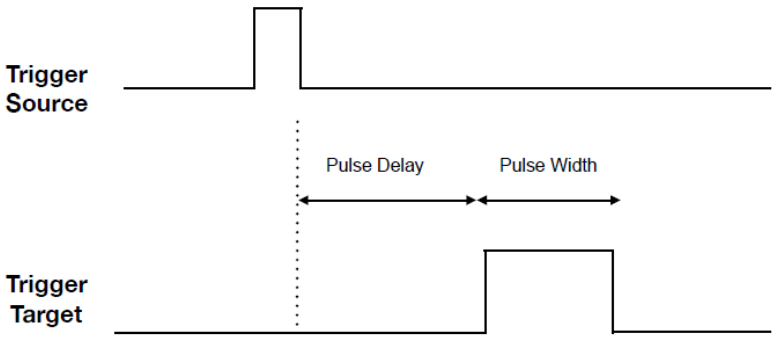
PPP	
Output pulse with position offset and position offset	

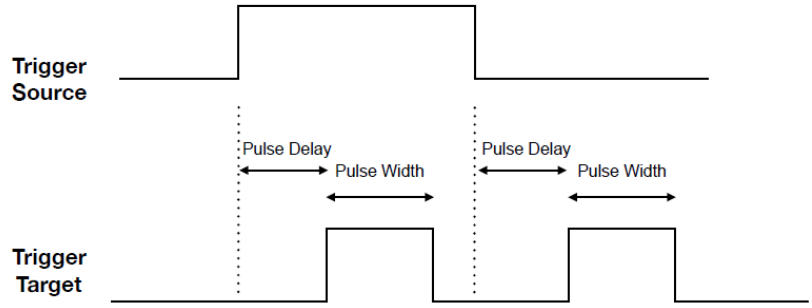
Trigger Mode

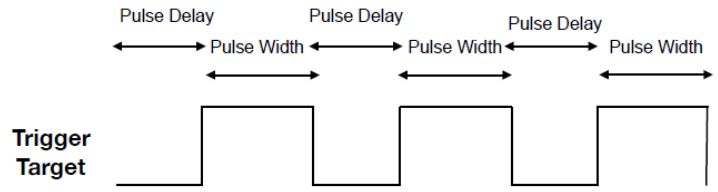
There are five trigger modes that can be set by user.

Trigger Mode	Description
Never	It is the initial value. It means turn off this channel.

Rising edge	
1. It means that the trigger target is triggered when the state of the trigger source changes from zero to one. 2. If the trigger source is QEI, the trigger target is triggered when the value of QEI is larger than the trigger data.	

Falling edge	
1. It means that the trigger target is triggered when the state of the trigger source changes from one to zero. 2. If the trigger source is QEI, the trigger target is triggered when the value of QEI is smaller than the trigger data.	

Level	
It means that The trigger target is triggered twice in the duration of one trigger source pulse.	

Always	
Infinite pulse	

Time Unit

The basic operation concept of MCU is infinite loop. MCU will process the results of each trigger target based on trigger source in every loop. Time unit is the time interval between two continuous actions that MCU generates the result to the trigger target. We will calculate the pulse delay and pulse width based on this time unit. For example, if the time unit is 25 μs and the pulse delay is 2000, it means that pulse delay is 50 ms (2000 x 25 μs).

The default value of time unit is 25 μs. However, users can adjust the time unit to fit their application needs. The value should be configured between 25 ~2500 μs and setting shorter time unit(< 25 μs) is not safe and may cause unexpected result.

Delta

Integer value specifies the fine-tuning factor for time unit. The value of +/- 1 means increasing / decreasing 0.0125 μs for the time unit. Default value of delta is 0.

Time Unit and Delta

We will use the simple formula to explain the relationship between time unit and delta. We assume that Total Time Unit (TTU) is the value of time unit that applied to MCU.

Total Time Unit (TTU)

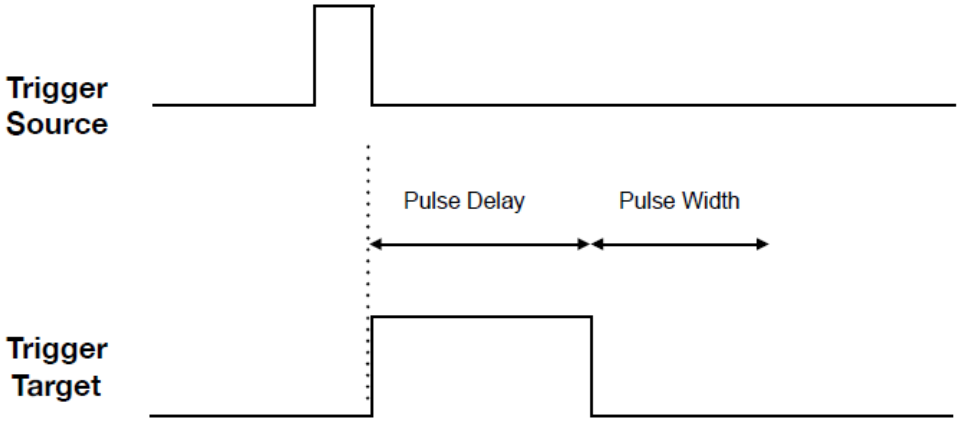
Total Time Unit (TTU) = time unit (ex: 25 μs) + delta * 0.0125

For example, time unit is equal to 25 μs, delta is equal to 10. And then We can calculate TTU and the result is 25.125 μs (TTU = 25 + 10*0.0125 = 25.125)

Please be careful about the value of TTU. If TTU is below 25 μs or even below 0 μs, that will cause unexpected result. We didn’t constrain to time unit and delta, so it is recommended calculating TTU before setting up configuration.

Active Low

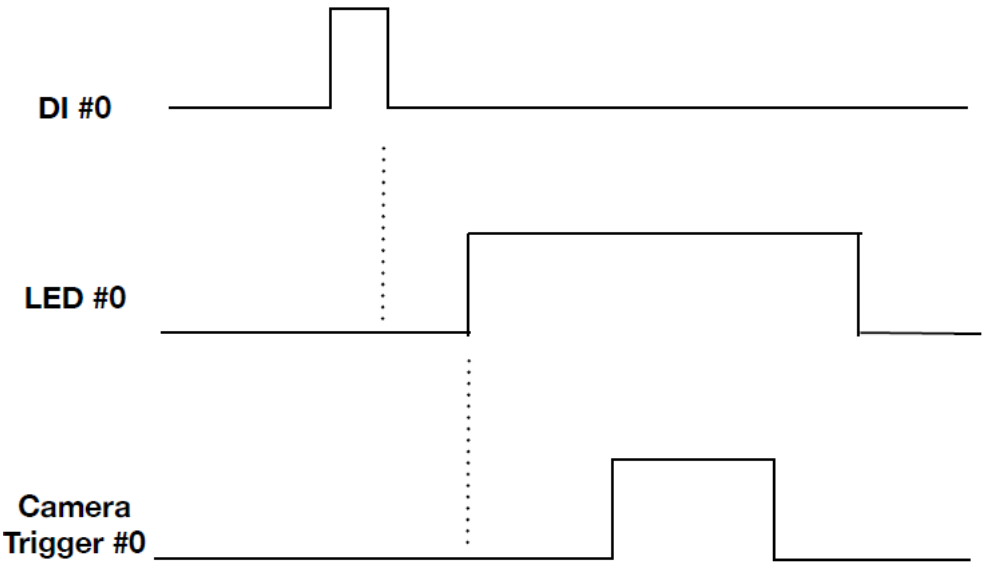
When the flag of the active low is set, the state of the output signal is high during pulse delay and the output signal is low during pulse width. We will introduce about how to enable this flag in the following paragraph.



Active low Mode

Trigger Target as Trigger Source

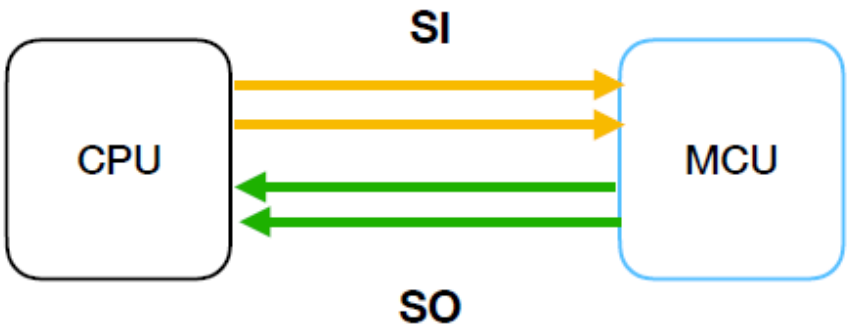
It is the new feature we introduce in DTIOv2. User can use “the trigger target” to trigger other trigger target. For example, you can configure a channel that triggers the camera after LED0 is turned on.



Trigger target as trigger source Mode

Software Trigger

We provide 2 software trigger inputs (SI in short) and 2 software trigger outputs (SO in short) in Nvis-5306RT. User can send signals from CPU to MCU directly via SI and receive signals directly from MCU via SO.



Software Trigger

DTIOv2 Function Reference

DT2_Setup

Syntax	BOOL __cdecl DT2_Setup(DT2_SETUP *lpSetup, DWORD cbSetup);
Description	Sets up parameters used in the specified DTIOv2 controller.
Parameter	**DT2_SETUP [in]** **Index [in]** Set the parameter to configure which channel to use. There are 8 channels in DTIOv2, configurable parameters are 0~7. **modeFlag [in]** There are 4 modes you can configure in this parameter. Set the following value to configure the mode you want. Please notice that only “active low” mode flag can be enabled with three other modes at the same time. More information will be provided in the next chapter.

Mode	Value
Disable	0
Active Low	DT2_INIT_HIGN (0x01)
Trigger Buffer	DT2_TRIG_BUFF (0x02)
External Input (default deactivated)	DT2_TRIG_EIDD (0x04)
External Input (default activated)	DT2_TRIG_EIDA (0x08)

modeType [in]

Mode	Value
PTT	DT2_MODE_PTT (0x00)
PTP	DT2_MODE_PTP (0x01)
PPT	DT2_MODE_PPT (0x02)
PPP	DT2_MODE_PPP (0x03)

trigMode [in]

Mode	Value
Never	DT2_TRIG_NEVER (0x00)
Rising edge	DT2_TRIG_RISING (0x01)
Falling edge	DT2_TRIG_FALLING (0x02)
Level	DT2_TRIG_LEVEL (0x03)
Always	DT2_TRIG_ALWAYS (0x04)

trigData [in]

This parameter configures the trigger position when QEI is set as the trigger source.

trigSrc [in]

This parameter configures the trigger source.

Digital Input

I/O	Value
DI 0	DT2_SRC_DI_00 (0x00010001)
DI 1	DT2_SRC_DI_01 (0x00010002)
DI 2	DT2_SRC_DI_02 (0x00010004)
DI 3	DT2_SRC_DI_03 (0x00010008)
DI 4	DT2_SRC_DI_04 (0x00010010)
DI 5	DT2_SRC_DI_05 (0x00010020)
DI 6	DT2_SRC_DI_06 (0x00010040)
DI 7	DT2_SRC_DI_07 (0x00010080)

Digital Output

I/O	Value
DO 0	DT2_SRC_DO_00 (0x00020001)
DO 1	DT2_SRC_DO_01 (0x00020002)
DO 2	DT2_SRC_DO_02 (0x00020004)
DO 3	DT2_SRC_DO_03 (0x00020008)
DO 4	DT2_SRC_DO_04 (0x00020010)
DO 5	DT2_SRC_DO_05 (0x00020020)
DO 6	DT2_SRC_DO_06 (0x00020040)
DO 7	DT2_SRC_DO_07 (0x00020080)

Camera Trigger Output

I/O	Value	
TRIG 0	DT2_SRC_TRIG_00	(0x00020100)
TRIG 1	DT2_SRC_TRIG_01	(0x00020200)
TRIG 2	DT2_SRC_TRIG_02	(0x00020400)
TRIG 3	DT2_SRC_TRIG_03	(0x00020800)

LED Controller

I/O	Value	
LED 0	DT2_SRC_LED_0	(0x00200001)
LED 1	DT2_SRC_LED_1	(0x00200002)
LED 2	DT2_SRC_LED_2	(0x00200004)
LED 3	DT2_SRC_LED_3	(0x00200008)

Software Trigger Input/ Output

I/O	Value	
SI 0	DT2_SRC_SI_00	(0x40000001)
SI 1	DT2_SRC_SI_01	(0x40000002)
SO 0	DT2_SRC_SO_00	(0x80000001)
SO1	DT2_SRC_SO_01	(0x80000002)

QEI

I/O	Value	
QEI	DT2_SRC_QEI_0	(0x00080001)

trigTgt [in]

This parameter configures the trigger target.

Digital Output

Mode	Value	
DO 0	DT2_TGT_DO_00	(0x00020001)
DO 1	DT2_TGT_DO_01	(0x00020002)
DO 2	DT2_TGT_DO_02	(0x00020004)
DO 3	DT2_TGT_DO_03	(0x00020008)
DO 4	DT2_TGT_DO_04	(0x00020010)
DO 5	DT2_TGT_DO_05	(0x00020020)
DO 6	DT2_TGT_DO_06	(0x00020040)
DO 7	DT2_TGT_DO_07	(0x00020080)

Camera Trigger Output

I/O	Value	
TRIG 0	DT2_TGT_TRIG_00	(0x00020100)
TRIG 1	DT2_TGT_TRIG_01	(0x00020200)
TRIG 2	DT2_TGT_TRIG_02	(0x00020400)
TRIG 3	DT2_TGT_TRIG_03	(0x00020800)

LED Controller

I/O	Value	
LED 0	DT2_TGT_LED_0	(0x00200001)
LED 1	DT2_TGT_LED_1	(0x00200002)
LED 2	DT2_TGT_LED_2	(0x00200004)
LED 3	DT2_TGT_LED_3	(0x00200008)

Software Trigger Input/ Output

I/O	Value	
SO 0	DT2_TGT_SO_00	(0x80000001)
SO1	DT2_TGT_SO_01	(0x80000002)

pulseDelay [in]

Mode	Description
PTT	Represent time of the pulse delay. Ex: pulseDelay = 1000 Pulse Delay = 1000 x time unit(default value is 25) ms
PTP	
PPT	Represent position offset of the pulse delay. Ex: pulseDelay = 1000 The distance from the trigger pointer to the rising edge of pulse is 1000 encoder units
PPP	

pulseWidth [in]

Mode	Description
PTT	Represent time of the pulse width. Ex: pulseWidth = 1000 Pulse Width = 1000 x time unit(default value is 25) ms
PPT	
PTP	Represent position offset of the pulse width Ex: pulseWidth = 1000 The distance from the rising edge to the falling edge is 1000 encoder units
PPP	

	pulseData1 [in] It is the reserved parameter. Set 0 in all situations. **pulseData2 [in]** It is the reserved parameter. Set 0 in all situations. **cbSetup [in]** cbSetup is the length of the structure and is calculated in byte. You can use size of(DT2_SETUP) to calculate.
Return Value	Returns TRUE if DT2_SETUP successful, FALSE if DT2_SETUP failed.

Usage	// Fill DT2_SETUP data struture
	DT2_SETUP aSetup;
	memset(&aSetup, 0, sizeof(aSetup));
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = 0;
	aSetup.trigMode = DT2_TRIG_ALWAYS;
	aSetup.index = 0;
	aSetup.modeType = DT2_MODE_PTT;
	aSetup.pulseDelay = 20000;
	aSetup.pulseWidth = 20000;
	
	// Configure parameters of the channel
	if (!DT2_Setup(&aSetup, sizeof(aSetup)))
	{
	printf("DT2_Setup failed \n");
	return 0;
	}

DT2_Start

Syntax	BOOL __cdecl DT2_Start(void);
Description	Starts DTIOv2 controller operation.
Parameter	None
Return Value	Return TRUE if DT2_Start successful, FALSE if DT2_Start procedure failed.

Usage	// Start Deterministic Trigger I/O v2 operation
	if (!DT2_Start())
	{
	printf("DT2_start failed \n");
	return 0;
	}

DT2_Stop

Syntax	BOOL __cdecl DT2_Stop(void);
Description	Stop DTIOv2 controller operation.
Parameter	None
Return Value	Return TRUE if DT2_Stop successful, FALSE if DT2_Stop procedure failed.

Usage	// Stop Deterministic Trigger I/O v2 operation
	if (!DT2_Stop())
	{
	printf("DT2_Stop failed \n");
	return 0;
	}

DT2_SetUnit

Syntax	BOOL __cdecl DT2_SetUnit(DWORD unit, int delta);
Description	Specify the time unit for DTIOv2. This function can be skipped if you want to use the default setting (25 μs).
Parameter	**Unit [in]** Specify the time unit in micro-second (recommended settings 25 to 2500). **Delta [in]** Integer value specifies the fine-tuning factor for time unit. The value of +/- 1 means increasing / decreasing 0.0125 μs for the time unit. Default value of delta is 0.
Return Value	Return TRUE if DT2_SetUnit succeeded, FALSE if DT2_SetUnit failed.

Usage	// Set time unit as 20 microseconds
	if (!DT2_SetUnit(20, 0))
	{
	printf("DT2_SetUnit failed \n");
	return 0;
	}

DT2_GetUnit

Syntax	WORD __cdecl DT2_GetUnit(void);
Description	Acquire current time unit setting (in microseconds).
Parameter	None
Return Value	WORD is the data type of the return value. It represents the value of the current time unit.
Usage	WORD unit = DT2_GetUnit();

DT2_StPush

Syntax	BOOL __cdecl DT2_StPush(DWORD mask, DWORD value);
Description	Writes data to MCU via software trigger. You can write all software trigger inputs through this function simultaneously.
Parameter	**Mask[in]** Each bit of this parameter determines whether relative software trigger input is used or not. If software trigger input 0 (SI0) is used, set bit 1 as 1. If software trigger input 1(SI1) is used, set bit 2 as 1. Other bits set 0 if those software trigger inputs are not used. **Value[in]** Each bit of this parameter represents a software trigger input. At most it can configure 32 software trigger inputs. We provide 2 software trigger inputs in DTIOv2, so we only use bit 1 and bit 2.
Return Value	Return TRUE if DT2_StPush successful, FALSE if DT2_StPush failed.

Usage	Set software trigger input 0 as 1.
	
	if (!DT2_StPush(0x00000001, 1))
	{
	printf("DT2_StPush failed \n");
	return 0;
	}

DT2_StPull

Syntax	BOOL __cdecl DT2_StPull(DWORD* lpValue);
Description	Read data from MCU via software trigger. You can read all software trigger outputs through this function simultaneously.
Parameter	**lpValue[out]** A pointer to a DWORD value that specifies the software trigger outputs. Each bit of this parameter represents a software trigger output. At most it can configure 32 software trigger outputs. We provide 2 software trigger outputs in DTIOv2, so we only use bit 1 and bit 2 here.
Return Value	Returns TRUE if DT2_StPull successful, FALSE if DT2_StPull failed.

Usage	DWORD soValue = 0;
	if (! DT2_StPull(&soValue))
	{
	printf("DT2_StPull failed \n");
	return 0;
	}

DTIOv2 Programming Tips

Always Cease All Actions Before Starting Again

DO NOT execute DT2_Start when MCU is still running, unexpected errors may happen. Always execute DT2_Stop for MCU to cease all actions. Once DT2_Stop returns successful (all actions ceased), execute DT2_Start to start up again is the safe and recommended method.

Stop When You See Error Messages

Most of functions we provide are designed to return false messages when errors occur. Therefore, we recommend stopping program.

```
if (!DT2_Start())
{
  printf("DT2_start failed \n");
  return 0;
}
```

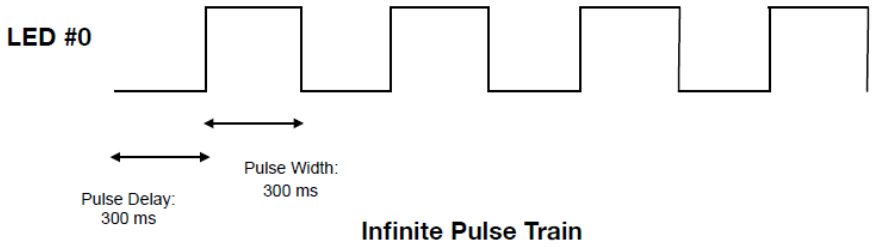
Reset All Channel When Starting DTIOv2 Program

It is recommended to reset all eight channels at the beginning of a DTIOv2 program as the system will remember the last configuration of each channel.

For example, channel 2 and channel 3 were set in previous operation. For a new operation, you only program channel 1 (without resetting previous settings), when you execute DT2_Start, channel 1, 2 and 3 will all be started. It may cause unexpected behaviors, so we suggest that a reset of all channels at the beginning of a DTIOv2 program.

```
int initialChannel()
{

 //Initialize 8 channels to avoid double definition*/
 for(int j = 0; j < 8; j++)
 {
        DT2_SETUP    zeroSetup;
        memset(&zeroSetup,0,sizeof(zeroSetup));
        zeroSetup.index = j;
        if ( ! DT2_Setup(&zeroSetup, sizeof(zeroSetup)) )
        {
              printf("Initializing DT2 channels failed \n");
           return 0;
        }

 }

 return 0;
}
```

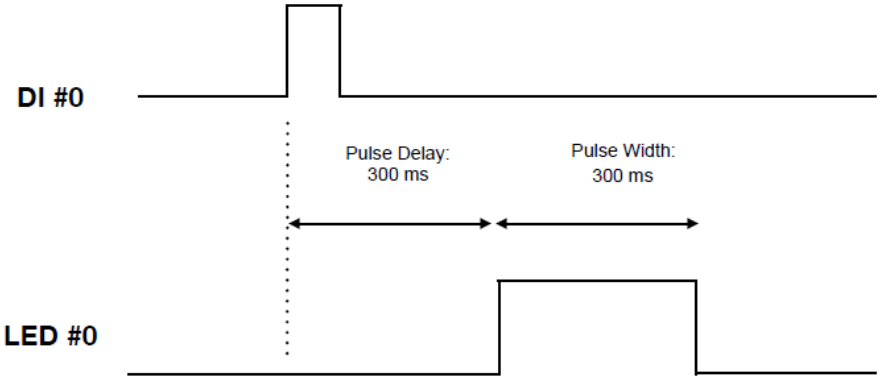
DTIOv2 Example 1 - Infinite Pulse Train

Description	This example demonstrates how to use DTIOv2 to generate an infinite pulse to trigger LED #0 in PTT mode. In this case, LED #0 will be turned off for 500 ms (20000 x 25 μs) and then turned on for 500 ms (20000 x 25 μs).The brightness of the LED is configured by constant current and set as 10 mA. DTIOv2 will be stopped after 4 seconds.
Illustration reference	 **Infinite Pulse Train**

Usage	int timeBaseDT2()
	{
	//Initialize configuration of 8 channels
	initialChannel();
	// Fill DT2_SETUP data struture
	DT2_SETUP aSetup;
	memset(&aSetup, 0, sizeof(aSetup));
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = 0;
	aSetup.trigMode = DT2_TRIG_ALWAYS;
	aSetup.index = 0;
	aSetup.modeType = DT2_MODE_PTT;
	aSetup.pulseDelay = 20000;
	aSetup.pulseWidth = 20000;
	// Configure parameters of the channel
	if (!DT2_Setup(&aSetup, sizeof(aSetup)))
	{
	printf("DT2_Setup failed \n");
	return 0;
	}

```
        // Config LED0 as CC mode and set the current as 10 mA
        DWORD pins = 0x00010000;
        DWORD mode = (pins) | LED_MODE_CC;
        if (LED_SetCurrentDriving(mode, 10) == false)
        {
                printf("Setting LED failed \n");
                return 0;
        }

        // Start Determinstic Trigger I/O v2 operation
        if (!DT2_Start())
        {
                printf("Starting DT2 failed \n");
                return 0;
        }

        // Stop DT2 after 4 senconds

        Sleep(4000);

        if (!DT2_Stop())
        {
                printf("Stopping DT2 failed \n");
                return 0;
        }

        return 0;
}
```

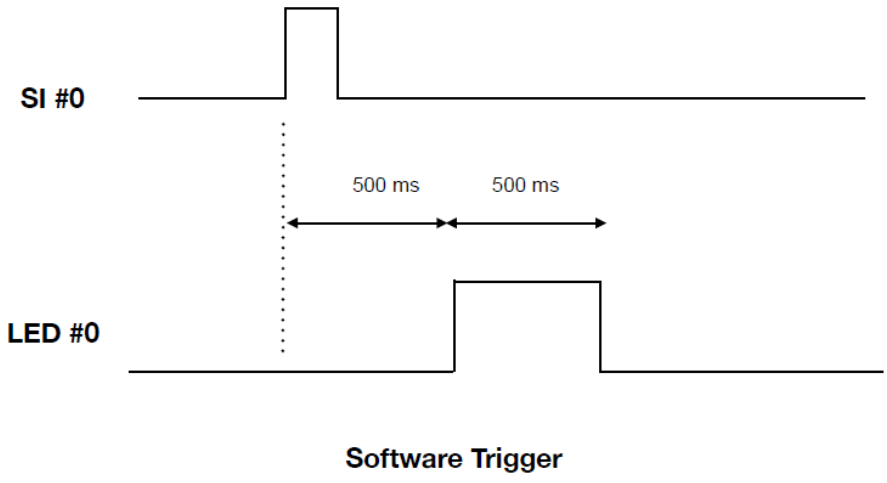
DTIOv2 Example 2 - Use DI to Trigger LED

Description	In this example, DI #0 is the trigger source and it triggers LED #0 on the rising edge. LED #0 is the trigger target. We redefine the time unit as 30 microseconds, so the pulse delay of LED is 300 ms (10000 x 30 μs) and the pulse width of LED is 300 ms (10000 x 30 μs). The brightness of the LED is configured by constant current and set as 10 mA.
Illustration reference	 **DI0 trigger LED0**

Usage	int timeBaseDIDT2()
	{
	//Initialize configuration of 8 channels
	initialChannel();
	// Fill DT2_SETUP data struture
	DT2_SETUP aSetup;
	memset(&aSetup, 0, sizeof(aSetup));
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = DT2_SRC_DI_00;
	aSetup.trigMode = DT2_TRIG_RISING;
	aSetup.index = 0;
	aSetup.modeType = DT2_MODE_PTT;
	aSetup.pulseDelay = 10000;
	aSetup.pulseWidth = 10000;
	// Set time unit as 30 microseconds
	if (!DT2_SetUnit(30, 0))
	{
	printf("DT2_SetUnit failed \n");

```
        return 0;
    }

    // Configure parameters of the channel
    if (!DT2_Setup(&aSetup, sizeof(aSetup)))
    {
        printf("DT2_Setup failed \n");
        return 0;
    }

    // Config LED0 as CC mode and set the current as 10 mA
    DWORD pins = 0x00010000;
    DWORD mode = (pins) | LED_MODE_CC;
    if (LED_SetCurrentDriving(mode, 10) == false)
    {
        printf("Setting LED failed \n");
        return 0;
    }

    // Start Determinstic Trigger I/O v2 operation
    if (!DT2_Start())
    {
        printf("Starting DT2 failed \n");
        return 0;
    }

    // Stop DT2 after 10 senconds

    Sleep(10000);

    if (!DT2_Stop())
    {
        printf("Stopping DT2 failed \n");
        return 0;
    }

    return 0;
}
```

DTIOv2 Example 3 -- Software Trigger

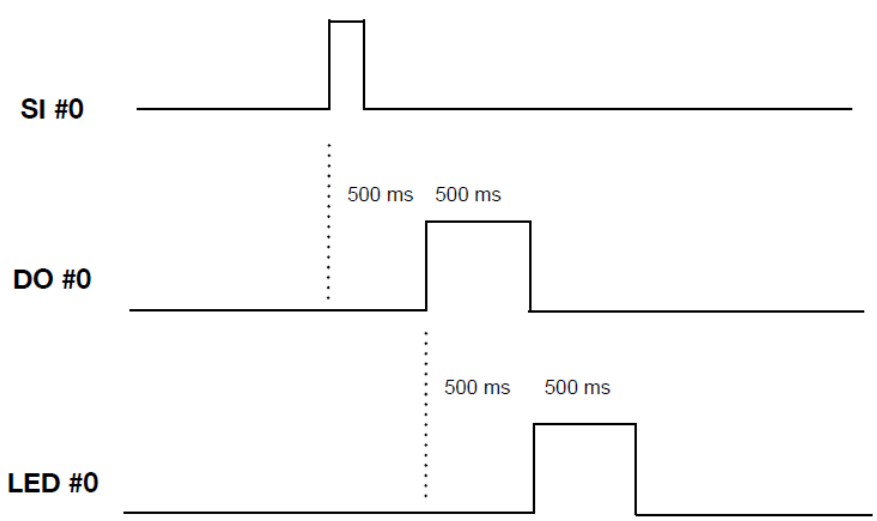
Description	This example demonstrates how to use software trigger input as the trigger source to trigger LED. Changing the status of the SI0 from 0 to 1 in short time is used to create the rising edge of the pulse signal. The LED #0 will be turned off for 500 ms (20000 x 25μs) and turned on for 500 ms(20000 x 25 μs) after each trigger. The brightness of the LED is configured by constant current and set as 10 mA. It will trigger 5 times and stop DTIOv2 in this example.
Illustration reference	 **Software Trigger**

Usage	int softwareTriggerDT2()
	{
	/*Reset configuration of each channel.*/
	initialChannel();
	
	DT2_SETUP aSetup;
	memset(&aSetup,0,sizeof(aSetup));
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = DT2_SRC_SI_00;
	aSetup.trigMode = DT2_TRIG_RISING;
	aSetup.index = 0 ;
	aSetup.modeType = DT2_MODE_PTT ;
	aSetup.pulseDelay =20000;
	aSetup.pulseWidth =20000;
	
	if (! DT2_Setup(&aSetup, sizeof(aSetup)))
	{
	printf("DT2 Setup fail \n");

```
            return 0;
    }


    // Config LED0 as CC mode and set the current as 10 mA
    DWORD pins = (0x00010000) << 0;
    DWORD mode = (pins) | LED_MODE_CC;

    if (LED_SetCurrentDriving(mode, 10) == false)
    {
        printf("Initial LED failed \n");
        return 0;
    }


    // Start Determinstic Trigger I/O v2 operation
    if (!DT2_Start())
    {
        printf("DT2_start failed \n");
        return 0;
    }

    //Change the status of SI0 from 0 -> 1 and set 1->0    to create pulse
    // We produce five times of pulse in this for loop
    for (int i = 0; i < 5; ++i)
    {
        if (!DT2_StPush(0x00000001, 0))
        {
            printf("DT2_StPush failed \n");
            return 0;
        }

        Sleep(10);

        if (!DT2_StPush(0x00000001, 1))
        {
            printf("DT2_StPush failed \n");
            return 0;
```

```
            }

            Sleep(1000);

            printf("Software trigger: %d \n", i);
        }


        // Stop Determinstic Trigger I/O v2 operation
        if (!DT2_Stop())
        {
            printf("DT2_Stop failed \n");
            return 0;
        }

        return 0;
}
```

DTIOv2 Example 4 - The Trigger Target as the Trigger Source

Description	This example is to demonstrate how to use the trigger target as the trigger source in the other channel. In the first channel, DO #0 is triggered by SI #0 and then DO #0 is used to trigger LED #0 in the second channel. Both channels are set in PTT mode and triggers on rising edge signal.
Illustration reference	 **The trigger target as the trigger source**

Usage	int triggerOutAsSource()
	{
	/*Reset configuration of each channel.*/
	initialChannel();
	
	//Trigger DO0 by SI0 based on PTT mode
	DT2_SETUP aSetup;
	memset(&aSetup, 0, sizeof(aSetup));
	aSetup.trigTgt = DT2_SRC_DO_00;
	aSetup.trigSrc = DT2_SRC_SI_00;
	aSetup.trigMode = DT2_TRIG_RISING;
	aSetup.index = 0;
	aSetup.modeType = DT2_MODE_PTT;
	aSetup.pulseDelay = 20000;
	aSetup.pulseWidth = 20000;
	
	//Trigger LED0 by DO0 based on PTT mode
	DT2_SETUP aSetup2;
	memset(&aSetup2, 0, sizeof(aSetup2));
	aSetup2.trigTgt = DT2_SRC_LED_0;

```
aSetup2.trigSrc = DT2_SRC_DO_00;
aSetup2.trigMode = DT2_TRIG_RISING;
aSetup2.index = 1;
aSetup2.modeType = DT2_MODE_PTT;
aSetup2.pulseDelay = 20000;
aSetup2.pulseWidth = 20000;


// Configure parameters of the channel
if (!DT2_Setup(&aSetup, sizeof(aSetup)))
{
        printf("DT2_Setup fail \n");
        return 0;
}

if (!DT2_Setup(&aSetup2, sizeof(aSetup2)))
{
        printf("DT2_Setup fail \n");
        return 0;
}

// Config LED0 as CC mode and set the current as 10 mA
DWORD pins = 0x00010000;
DWORD mode = (pins) | LED_MODE_CC;
if (LED_SetCurrentDriving(mode, 10) == false)
{
        printf("Setting LED failed \n");
        return 0;
}

// Start Determinstic Trigger I/O v2 operation
if (!DT2_Start())
{
        printf("DT2_start failed \n");
        return 0;
}

//Change the status of SI0 from 0 -> 1 and set 1->0    to create pulse
```

```
        // We produce five times of pulse in this for loop
        for (int i = 0; i < 5; ++i)
        {
            if (!DT2_StPush(0xFFFFFFFF, 0))
            {
                printf("DT2_StPush failed \n");
                return 0;
            }

            Sleep(10);

            if (!DT2_StPush(0xFFFFFFFF, 1))
            {
                printf("DT2_StPush failed \n");
                return 0;
            }

            Sleep(1000);

            printf("Software trigger: %d \n", i);
        }

        // Stop Determinstic Trigger I/O v2 operation
        if (!DT2_Stop())
        {
            printf("DT2 stop fail \n");
            return 0;
        }

        return 0;
}
```

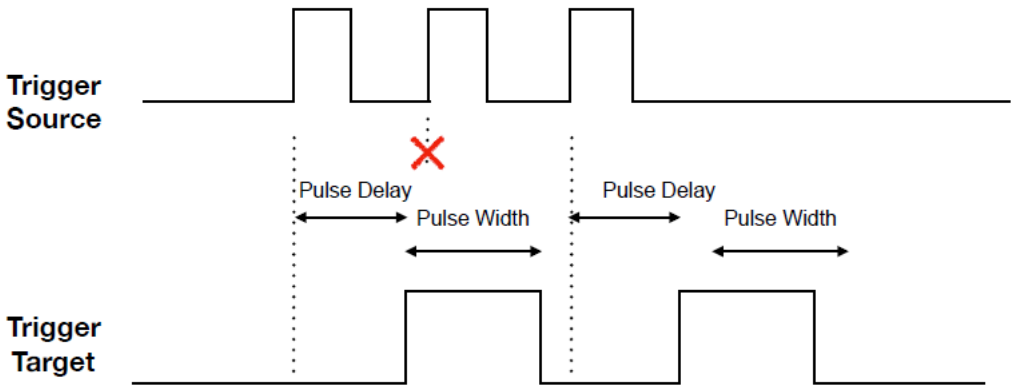
Appendix E: Advanced DTIOv2

To enhance the power of DTIOv2, we provide two new features that make DTIOv2 more applicable in various industrial environments. The two features are “Trigger Buffer” and “External Input”. They will be explained in the following sections.

The Concept of Advanced DTIOv2

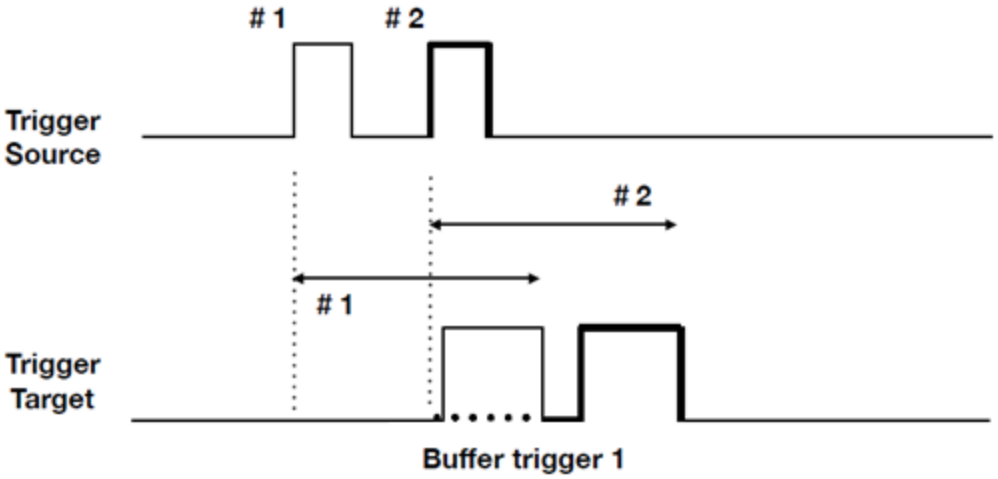
Trigger Buffer

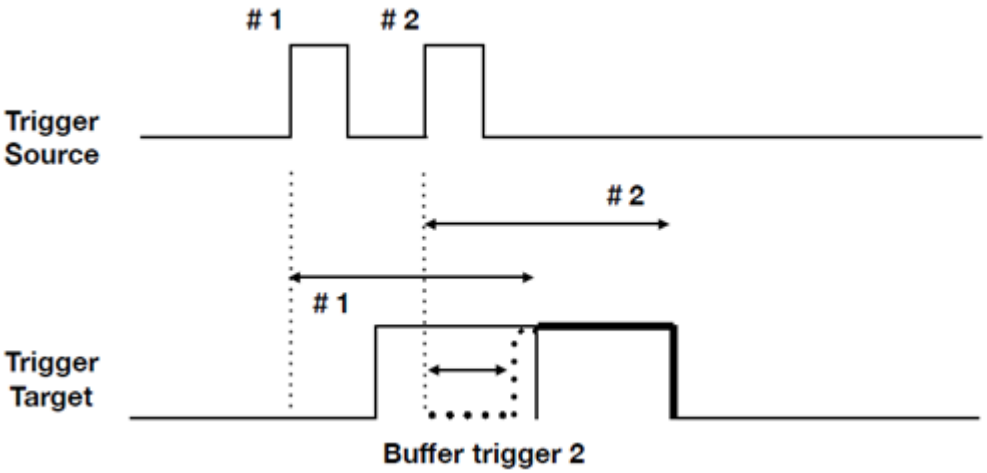
Generally, the trigger target won’t be interrupted during the active status. The active status is the time of pulse delay and pulse width. Therefore, before the active status ends, it will not receive any trigger signals from the trigger source. In other words, it ignores all trigger source inputs until the end of the pulse width. This mechanism will protect the trigger target to complete its work from unpredictable accidents.



With out Buffer trigger

In some cases, every pulse of the trigger source should result in relative action in the trigger target signal. Hence, we design a function called “trigger buffer” to process such actions. In order to protect the trigger target to finish its work, trigger signal will not be interrupted before the active status ends. However, the trigger buffer mechanism continues to calculate the pulse delay and pulse width in the background. Once the active status pulse width is over, the trigger target continues to perform signals of pulse delay and pulse width in the trigger buffer.





NOTE

1. ***Unable to trigger buffer and “always” trigger mode at the same time.*** *When the system is in “always” trigger mode, the buffer is automatically triggered and filled within a millisecond.*
2. ***Buffer size is 32.*** *If the buffer is full, the mechanism stops storing and will ignore trigger signals.*

Trigger Buffer can be Applied to 8 Channels Simultaneously

External Input

In machine vision applications, image processing is the key to determine if the product is defective or not. Image processing requires powerful processing capability to make decisions within microsecond or even milliseconds. Powerful image processing units in a computer may include CPU, GPU or dedicated MCU. What is even more efficient is when we can have two or all three working side-by-side, such as the CPU performing the initial processing and sends decoded preliminary results to the dedicated MCU. This the main idea and why we developed an “External Input” feature in DTIOv2.

Nuvis-5306RT is composed of two main processing powerhouses by default, the mother board with CPU and chipset and the dedicated MCU. What we have done is built an efficient communication channel between the two so they can work hand-in-hand.

We added an additional action during the pulse delay of the trigger target. We give an index to this pulse and send information to the computer. Then the computer will send back information containing the status of this pulse within the time of the pulse delay. Sending 1 means activate pulse and sending 0 means deactivate pulse.

If MCU does not receive the status of this pulse before the end of pulse delay, it will assign the default value to the status to this pulse. We have mentioned there are two types of External Input in mode Flag. You can choose the relative flag to decide the default value.

External Input (deactivated by default)	DT2_TRIG_EIDD (0x04)
External Input (activated by default)	DT2_TRIG_EIDA (0x08)

NOTE

1. ***Unable to trigger buffer and “always” trigger mode at the same time.*** *When the system is in “always” trigger mode, the buffer is automatically triggered and filled within a millisecond.*
2. ***External output is only valid on the last channel.*** *When configuring multiple channels, only the last channel is valid for “External Input”. Eg. If you configure channels 2, 4, 6, only channel 6 can be configured as “External Input”.*
3. ***Buffer size is 32.*** *If the buffer is full, the mechanism stops storing and will ignore trigger signals.*

Advanced DTIOv2 Function Reference

DT2_EiRegisterIndexed

Syntax	BOOL __cdecl DT2_EiRegisterIndexed(void (__stdcall *pfnHandler)(DWORD data));
Description	It registers a function that can determine whether to send an activate pulse command or a deactivate pulse command.
Parameter	**pfnHandler [in]** A point to the function that contain an argument which is DWORD.
Return Value	Return TRUE if DT2_EiRegisterIndexed successful, FALSE if DT2_EiRegisterIndexed failed.

Usage	Sending 1 means activate pulse and sending 0 means deactivate pulse.
	
	static void __stdcall __IndexedHandler(DWORD index)
	{
	/* If the index is odd number, send 1 back.
	The trigger target will change the status to pulse high.
	If the index is even number, send 0 back.
	The trigger target will stay as pulse low.
	*/
	printf("data %d \n", index);
	if (index % 2 == 1)
	{
	if (!DT2_EiSendResult(index, 1))
	{
	printf("DT2_EiSendResult failed \n");
	}
	}
	else
	{
	if (!DT2_EiSendResult(index, 0))
	{
	printf("DT2_EiSendResult failed \n");
	}
	}

```
}

if (!DT2_EiRegisterIndexed(__IndexedHandler))
{

        printf("DT2_EiRegisterIndexed failed \n");
        return 0;

}
```

DT2_EiSendResult

Syntax	BOOL __cdecl DT2_EiSendResult(DWORD idx, DWORD value)
Description	Send the decision back to MCU.
Parameter	**idx [in]** The index of the pulse. **Value [in]**
Return Value	Return TRUE if DT2_EiSendResult succeeded, FALSE if DT2_EiSendResult failed.

Usage	if (!DT2_EiSendResult(index, 1))
	{
	printf("DT2_EiSendResult failed \n");
	}

Advanced DTIOv2 Example 1 - Trigger Buffer

Description	This example demonstrates that all pulses of the trigger source won't be ignored because of the trigger buffer.

Usage	int bufferTriggerDT2()
	{
	/*Reset configuration of each channel.*/
	initialChannel();
	
	// Trigger LED0 by SI0
	// Enable buffer trigger flag to queue untriggering pulses before triggering LED0
	DT2_SETUP aSetup;
	memset(&aSetup, 0, sizeof(aSetup));
	aSetup.modeFlags = DT2_TRIG_BUFF;
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = DT2_SRC_SI_00;
	aSetup.trigMode = DT2_TRIG_RISING;
	aSetup.index = 0;
	aSetup.modeType = DT2_MODE_PTT;
	aSetup.pulseDelay = 100000;
	aSetup.pulseWidth = 20000;
	
	// Configure parameters of the channel
	if (!DT2_Setup(&aSetup, sizeof(aSetup)))
	{
	printf("DT2 Setup fail \n");
	return 0;
	}
	
	// Config LED0 as CC mode and set the current as 10 mA
	DWORD pins = (0x00010000) << 0;
	DWORD mode = (pins) | LED_MODE_CC;
	
	if (LED_SetCurrentDriving(mode, 10) == false)
	{
	printf("Initial LED failed \n");
	return 0;

```
}

// Start Deterministic Trigger I/O v2 operation
if (!DT2_Start())
{
	printf("DT2_start failed \n");
	return 0;
}

//Change the status of SI0 from 0 -> 1 and set 1->0    to create pulse
// We produce five times of pulse in this for loop
for (int i = 0; i < 5; ++i)
{
	if (!DT2_StPush(0xFFFFFFFF, 0))
	{
		printf("DT2_StPush failed \n");
		return 0;
	}

	Sleep(10);

	if (!DT2_StPush(0xFFFFFFFF, 1))
	{
		printf("DT2_StPush failed \n");
		return 0;
	}

	Sleep(1000);

	printf("Software trigger: %d \n", i);
}


Sleep(10000);

// Stop Deterministic Trigger I/O v2 operation
if (!DT2_Stop())
{
```

```
            printf("DT2_Stop failed \n");
            return 0;
        }


        return 0;

}
```

Advanced DTIOv2 Example 2 - External Input

Description

This example demonstrates the feature of External Input. LED0 is triggered by SI0 and “External Input” function is used to control the behavior of LED0. If the index is odd, send activate pulse command. If the index is even, send deactivate pulse command.

Usage

```
static void __stdcall __IndexedHandler(DWORD index)
{
	/*	If the index is odd number, send 1 back.
	The trigger target will change the status to pulse high.
	If the index is even number, send 0 back.
	The trigger target will stay as pulse low.
	*/
	printf("data %d \n", index);
	if (index % 2 == 1)
	{
		if (!DT2_EiSendResult(index, 1))
		{
			printf("DT2_EiSendResult failed \n");
		}
	}
	else
	{
		if (!DT2_EiSendResult(index, 0))
		{
			printf("DT2_EiSendResult failed \n");
		}
	}


}

int externalInputDT2()
{
	//Reset configuration of each channel.
	initialChannel();

	// Trigger DO0 by SI0 based on PTT mode
```

```
        // Enable external input
        // User can determine the trigger source status by the message
from computer
        DT2_SETUP    aSetup;
        memset(&aSetup, 0, sizeof(aSetup));
        aSetup.modeFlags = DT2_TRIG_EIDD;
        aSetup.trigTgt = DT2_TGT_LED_0;
        aSetup.trigSrc = DT2_SRC_SI_00;
        aSetup.trigMode = DT2_TRIG_RISING;
        aSetup.index = 0;
        aSetup.modeType = DT2_MODE_PTT;
        aSetup.pulseDelay = 50000;
        aSetup.pulseWidth = 20000;



        if (!DT2_Setup(&aSetup, sizeof(aSetup)))
        {
                printf("DT2 Setup fail \n");
                return 0;
        }



        // Config LED0 as CC mode and set the current as 10 mA
        DWORD pins = 0x00010000;
        DWORD mode = (pins) | LED_MODE_CC;
        if (LED_SetCurrentDriving(mode, 10) == false)
        {
                printf("Setting LED failed \n");
                return 0;
        }

        // Register callback function which can determine the status of the
trigger target
        // Our mechanism will give each untriggered pulse a index
        if (!DT2_EiRegisterIndexed(__IndexedHandler))
        {


                printf("DT2_EiRegisterIndexed failed \n");
```

```
        return 0;

}

// Start Deterministic Trigger I/O v2 operation
if (!DT2_Start())
{
        printf("DT2_Start failed \n");
        return 0;
}



//Change the status of SI0 from 0 -> 1 and set 1->0    to create pulse
// We produce five times of pulse in this for loop
for (int i = 0; i < 5; ++i)
{
        if (!DT2_StPush(0xFFFFFFFF, 0))
        {
                printf("DT2_StPush failed \n");
                return 0;
        }

        Sleep(10);

        if (!DT2_StPush(0xFFFFFFFF, 1))
        {
                printf("DT2_StPush failed \n");
                return 0;
        }

        Sleep(1000);

        printf("Software trigger: %d \n", i);
}

Sleep(3000);
```

```
        // Stop Deterministic Trigger I/O v2 operation
        if (!DT2_Stop())
        {
                printf("DT2_Stop failed \n");
                return 0;
        }

        return 0;

}
```

Pulse Width Modulator (PWM)

The Concept of Pulse Width Modulator (PWM)

Pulse Width Modulator can encode analog signal into digital signal by modulating the duty cycle of the square wave. It generates a high resolution square wave to control motor or switch power supply.

We provide Pulse Width Modulator (PWM) API for programming and operating the PWM controller. DTIOv2 contains two PWM generator blocks. Each generator block produces two PWM output signals. Two PWM output signals in the same generator block share the same timer and frequency.

Generator Block	PWM output signals
PWM_GEN_0	PWM_PIN_0
	PWM_PIN_1
PWM_GEN_1	PWM_PIN_2
	PWM_PIN_3

Configuring the Isolated Digital Output as PWM Output

Do4 ~ Do7 can be configured as PWM output (pin 0 ~ 3). We provide flexible methods to switch between Do output and PWM output.
If the pin bits are set by PWM_Start, those outputs are configured as PWM.
If the pin bits are stopped by PWM_Stop, those outputs are configured as Do.

PWM Function Reference

PWM_GenSetup

Syntax	BOOL __cdecl PWM_GenSetup(DWORD genBits, PWM_GEN_SETUP *lpSetup, DWORD cbSetup);
Description	PWM clock settings configuration.
Parameter	**genBits [in]** Specifies the generator block bits of PWM controllers. **lpSetup [in]** A pointer to a **PWM_GEN_SETUP** structure that contains the PWM generator configuration. This data structure contains the following variables: **genMode [in]** Value specifies the generator mode of the specified PWM controller.

Mode	Value
Down count Mode	PWM_GEN_MODE_DOWN (0x00000000)
Up/ down count Mode	PWM_GEN_MODE_UP_DOWN (0x00000002)
synchronous updates	PWM_GEN_MODE_SYNC (0x00000038)
Immediate updates	PWM_GEN_MODE_NO_SYNC (0x00000000)

intrTriggers [in]
The variable is not available to users at this stage. Please set 0 to disable this variable.

deadBandRise [in]
The variable is not available to users at this stage. Please set 0 to disable this variable.

deadBandFall [in]
The variable is not available to users at this stage. Please set 0 to disable this variable.

	cbSetup [in] This variable is to set the length of the structure in bytes. The caller must set this member to size of (PWM_GEN_SETUP).
Return Value	Returns TRUE if procedure successful, FALSE if procedure failed.

Usage	PWM_GEN_SETUP setup;
	memset(&setup, 0, sizeof(setup));
	setup.genMode = PWM_GEN_MODE_UP_DOWN |
	PWM_GEN_MODE_NO_SYNC;
	BOOL returnValue = PWM_GenSetup(PWM_GEN_0, &setup,
	sizeof(setup));

PWM_GenPeriod

Syntax	BOOL __cdecl PWM_GenPeriod(DWORD genBits, DWORD period);
Description	Set the period of a PWM generator.
Parameter	**genBits [in]** Specifies the generator block bits of PWM controllers. **period [in]** Specifies the period of PWM generator output, measured in clock ticks.
Return Value	Returns TRUE if procedure successful, FALSE if procedure failed.
Usage	BOOL returnValue = PWM_GenPeriod(PWM_GEN_0 | PWM_GEN_1, 800); // 100 KHz

PWM_PulseWidth

Syntax	BOOL __cdecl PWM_PulseWidth(DWORD pinBits, DWORD width);
Description	Set the pulse width for the specified PWM output.
Parameter	**pinBits [in]** Specifies the output bits of PWM controllers. **width [in]** Specifies the width of the pulse.
Return Value	Returns TRUE if procedure successful, FALSE if procedure failed.
Usage	BOOL returnValue = PWM_PulseWidth(PWM_PIN_0 | PWM_PIN_3, 400);

PWM_PulseInvert

Syntax	BOOL __cdecl PWM_PulseInvert(DWORD pinBits);
Description	Sets inversion mode for PWM outputs.
Parameter	**pinBits [in]** Specify the output bits of PWM to be inverted.
Return Value	Returns TRUE if procedure successful, FALSE if procedure failed.
Usage	BOOL returnValue = PWM_PulseInvert(PWM_PIN_0);

PWM_Start

Syntax	BOOL __cdecl PWM_Start(DWORD pinBits);
Description	Start PWM controller operation.
Parameter	**pinBits [in]** Specifies the output bits of PWM controllers.
Return Value	Returns TRUE if procedure successful, FALSE if procedure failed.
Usage	BOOL returnValue = PWM_Start(PWM_PIN_0 | PWM_PIN_1);

PWM_Stop

Syntax	BOOL __cdecl PWM_Stop(DWORD pinBits);
Description	Stop PWM controller operation.
Parameter	**pinBits [in]** Specify the output bits of PWM controllers.
Return Value	Returns TRUE if stop procedure successful, FALSE if stop procedure failed.
Usage	BOOL returnValue = PWM_Stop(PWM_PIN_0 | PWM_PIN_1);

PWM Example

Description	In this PWM example, we utilize PWM functions to provide motor control. To begin, we setup the environment and start up PWM. Next, we accelerate the speed of motor by PWM_GenPeriod and PWM_PulseWidth function. Lastly, we slow down the motor linearly and to an eventual full stop.

Usage	int PWMStart()
	{
	//Configure PWM
	PWM_GEN_SETUP setup;
	memset(&setup, 0, sizeof(setup));
	setup.genMode |= PWM_GEN_MODE_DOWN;
	setup.genMode |= PWM_GEN_MODE_NO_SYNC;
	if(!PWM_GenSetup(PWM_GEN_0, &setup, sizeof(setup)))
	{
	printf("PWM SETUP fail \n");
	return 0;
	}
	// set the speed of PWM
	if (! PWM_GenPeriod(PWM_GEN_0, 65535)) // 100 KHz
	{
	printf("PWM GenPeriod fail \n");
	return 0;
	}
	
	//Start PWM
	if (! PWM_Start(PWM_PIN_0))
	{
	printf("PWM start fail \n");
	return 0;
	}
	
	return 0;
	
	}

```
int PWMTriangleMode()
{

    for(int i = 0; i < 1 ; ++i)
    {
        // Accelerate the motor by setting PWM
        for(int count=0;count<460; count++)
        {
            if ( ! PWM_GenPeriod(PWM_GEN_0,
AccProfile[count]) )    // 100 KHz
            {
                printf("PWM GenPeriod fail \n");
                return 0;
            }

            if ( !
PWM_PulseWidth(PWM_PIN_0,AccProfile[count]>>1 ) )
            {
                printf("PWM Pulsewidth fail \n");
                return 0;
            }
            Sleep(5);



        }

        Sleep(10000);

        // Slow down the motor by setting PWM
        for(int count=459;count>0; count--)
        {
            if ( ! PWM_GenPeriod(PWM_GEN_0,
AccProfile[count]) )    // 100 KHz
            {
                printf("PWM GenPeriod fail \n");
                return 0;
            }
```

```
                    if ( !
PWM_PulseWidth(PWM_PIN_0,AccProfile[count]>>1 ) )
                    {
                        printf("PWM Pulsewidth fail \n");
                        return 0;
                    }
                    Sleep(5);


                }
        }


        // Must stop PWM
        if ( ! PWM_Stop(PWM_PIN_0) )
       {
            printf("PWM stop fail \n");
              return 0;
       }

           return 0;
}

int PWMControl()
{

        PWMStart();
        for(int i = 0; i < 1 ; ++i)
              PWMTriangleMode();

        return 0;
}
```

Advanced Examples

Combination PWM and DTIOv2

Description	In this example, we will introduce how to leverage DTIOv2 and PWM control simultaneously.

Usage	int PositionTriggerWithPWM()
	{
	// Start PWM
	//Must PWM start before QEI initial
	PWMStart();
	
	//Reset configuration of each channel.
	initialChannel();
	
	/*Fill the structure of DT2_SETUP
	1. Define ppp mode to trigger LED by QEI source
	2. Define ppp mode to trigger Camera by QEI source
	*/
	
	// Trigger LED when the position is 1000 and the LED will turn on from position 1000 to position 3000.
	DT2_SETUP aSetup;
	memset(&aSetup,0,sizeof(aSetup));
	aSetup.trigTgt = DT2_TGT_LED_0;
	aSetup.trigSrc = DT2_SRC_QEI_0 ;
	aSetup.trigMode = DT2_TRIG_RISING;
	aSetup.index = 0 ;
	aSetup.modeType = DT2_MODE_PPP;
	aSetup.trigData = 1000;
	aSetup.pulseDelay =0;
	aSetup.pulseWidth =2000;
	
	//Trigger camera when the position is 1500.

```
 DT2_SETUP    aSetup2;
 memset(&aSetup2,0,sizeof(aSetup2));
 aSetup2.trigTgt = DT2_TGT_TRIG_00;
 aSetup2.trigSrc = DT2_SRC_QEI_0 ;
 aSetup2.trigMode = DT2_TRIG_RISING;
 aSetup2.index = 1 ;
 aSetup2.modeType = DT2_MODE_PPP;
 aSetup2.trigData = 1500;
 aSetup2.pulseDelay =0;
 aSetup2.pulseWidth =500;



 // Configure parameters of the channel
 if ( ! DT2_Setup(&aSetup, sizeof(aSetup)) )
{
        printf("DT2 Setup fail \n");
          return 0;
}

 if ( ! DT2_Setup(&aSetup2, sizeof(aSetup2)) )
{
        printf("DT2 Setup fail \n");
          return 0;
}

 //Set configuration of LED0
 DWORD pins = (0x00010000) << 0;
 //Set LED Mode as constant current
 DWORD mode = (pins) | LED_MODE_CC;

 // Set constant current as 100 mA. It means that LED is turned on.
 if (LED_SetCurrentDriving(mode, 100) == false)
 {
        printf("Setting LED failed \n");
        return 0;
 }
```

```
        // Initial    QEI configuration
        QEIEncoderInit();

        // Start Determinstic Trigger I/O v2 operation
        if(!DT2_Start())
        {
                printf("Starting DT2 failed \n");
                return 0;
        }

        // Start the PWM slowly and then acceralate to the hightest velocity
we set. Keep the speed for          a while, and then slow down.
        PWMTriangleMode();

        // Stop DTIOv2
        // Must stop before next start. If not, it will cause some errors.
          if ( ! DT2_Stop() )
       {
                printf("Stopping DT2 failed \n");
                return 0;
       }


}
```