Lanner

Network Computing

Hardware Platforms for Network Computing

NCS2-POEIG402A/802A User Manual

Version: 1.1

Date of Release: 2021-08-16

Icon Descriptions

The icons are used in the manual to serve as an indication of interest topics or important messages. Below is a description of these icons:

Note: This check mark indicates that there is a note of interest and is something that you should pay special attention to while using the product.

Warning: This exclamation point indicates that there is a caution or warning and it is something that could damage your property or product.

Online Resources

The listed websites are links to the on-line product information and technical support.

Resources	URL
Lanner	http://www.lannerinc.com
Product Resource	http://www.lannerinc.com/download-center
RMA	http://eRMA.lannerinc.com

Copyright and Trademarks

Revision History

Version	Date	Descriptions
0.1	2020/05/07	1st official release

Table of Contents

CHAPTER 1: PRODUCT OVERVIEW

NCS2-POEIG802A/402A is an expansion card with 8/4 port PoE+ support. This module can be used on Lanner standard system that support NCS2 NIC module.

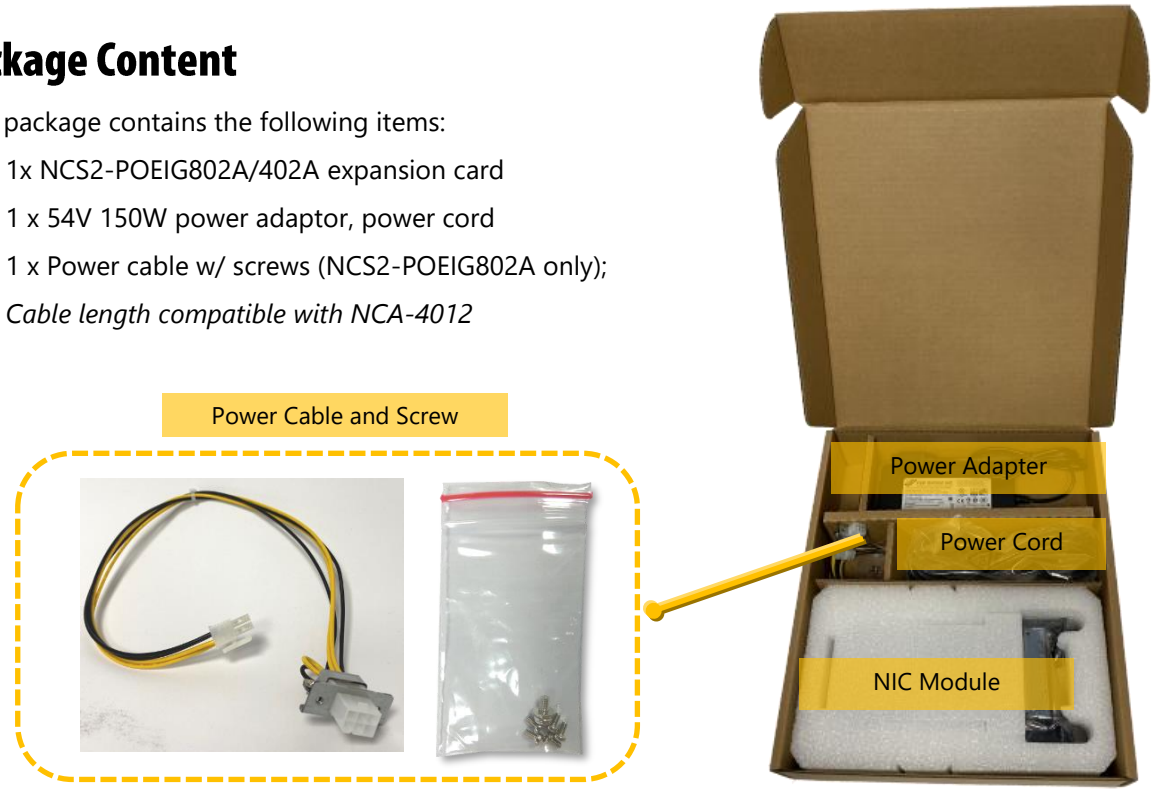
Ordering Information

Item	Description
NCS2-POEIG802A	8 x PoE+ ports w/ LED, 2 x Intel I350-AM4, 2 x PCIe*4 signal over PCIe*8 golden finger,54V / 240W for 8 ports POE+ default 25.5W per port, 54V Power input
NCS2-POEIG402A	4 x PoE+ ports w/ LED, 1 x Intel I350-AM4, 1 x PCIe*4 signal over PCIe*8 golden finger,54V / 120W for 4 ports POE+ default 25.5W per port, 54V Power input

Package Content

Your package contains the following items:

- 1x NCS2-POEIG802A/402A expansion card
- 1 x 54V 150W power adaptor, power cord
- 1 x Power cable w/ screws (NCS2-POEIG802A only);
  Cable length compatible with NCA-4012



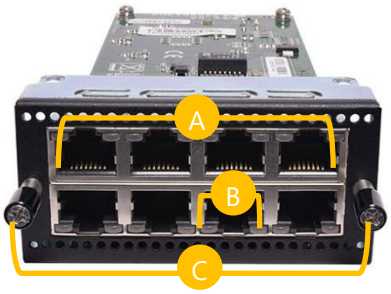
Specifications

Feature	NCS2-POEIG802A	NCS2-POEIG402A
LAN Port	8 x PoE+ ports w/ LED	4 x PoE+ ports w/ LED
Interfaces	PCI Express Base Specification Revision 2.0 (5GT/s)	
Chipset	2 x Intel I350-AM4	1 x Intel I350-AM4
Connector type	8 xRJ-45 connector All RJ-45 connector with ACT/Link, Speed LED	4 xRJ-45 connector All RJ-45 connector with ACT/Link, Speed LED
PoE Power output	54V / 240W for 8 ports POE+ default 25.5W per port	54V / 120W for 4 ports POE+ default 25.5W per port
POE power Input	54V Power input	

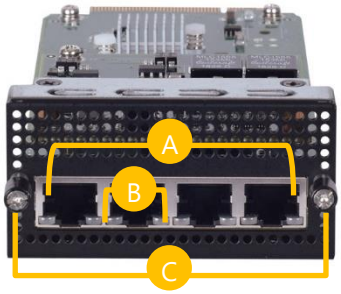
Physical Overview

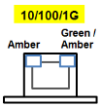
Front View

NCS2-POEIG802A



NCS2-POEIG402A



No.	Description	
A	POE Ports	8/4 PoE Ports, Total Power support up to 240/120W
B	RJ45 LAN LED	

Speed	Amber (Active/Link)	Green / Amber (Speed)
10M	Flash / steady	OFF
100M	Flash / steady	ON (Green)
1G	Flash / steady	ON (Amber)

C	Lock Screws	To secure the expansion slot door

Top View

NCS2-POEIG802A

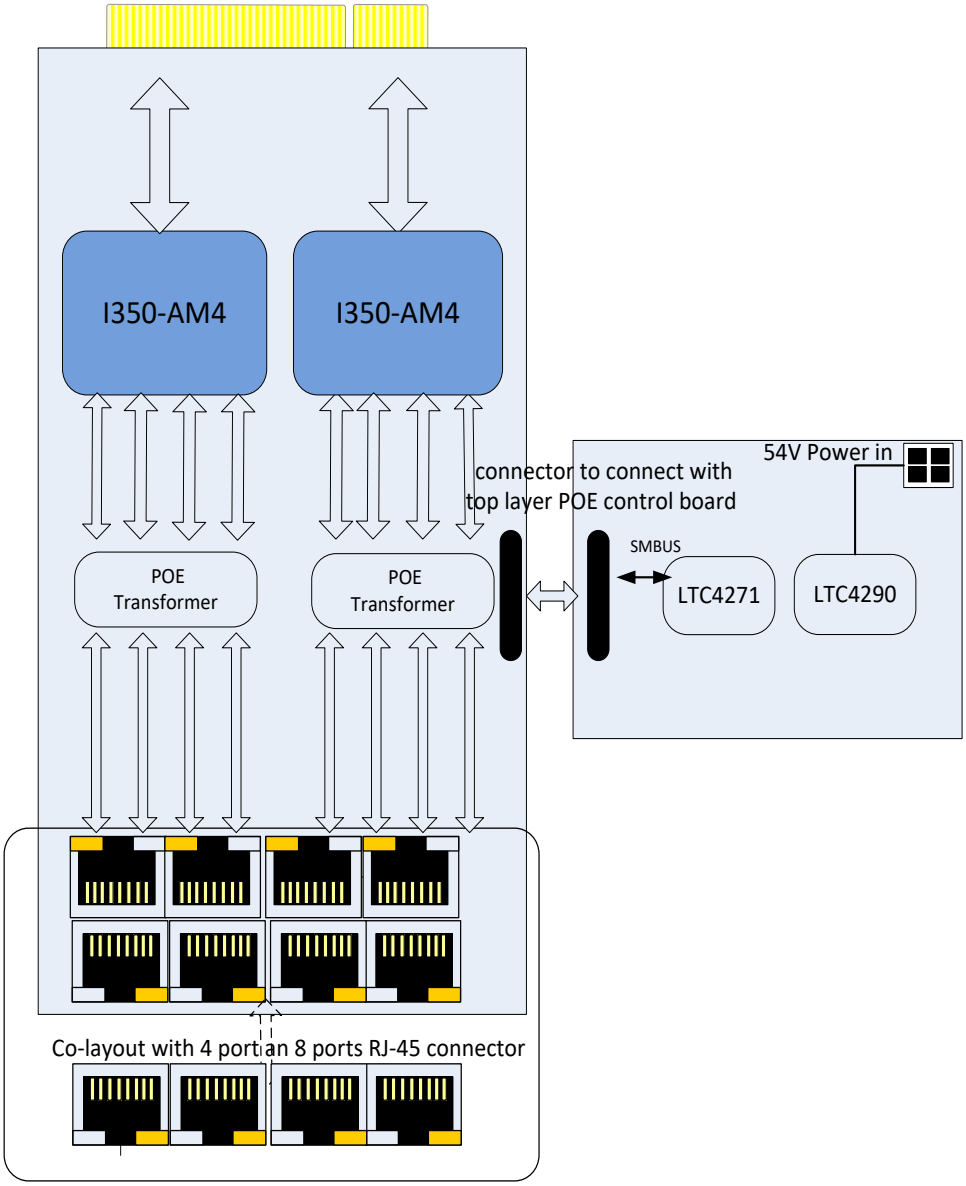
NCS2-POEIG402A

No.	Description
D	Gen2 PCIEx8 Golden Finger
E	PoE Power Cable Connector Remark: NCS2-POEIG402 PoE power cable was already fixing in front panel

CHAPTER 2: BOARD INFORMATION

Block Diagram

The block diagram indicates how data flows among components on the board. Please refer to the following figure for your motherboard's layout design.



- Note: The updated IEEE 802.3at-2009 PoE standard also known as PoE+ or PoE plus, provides up to 25.5 W of power for "Type 2" devices.
- The original IEEE 802.3at standard version of PoE supplies up to 30W of DC power (50~57 VDC and 600 mA) to each device. Only 25.5W is assured to be available at the powered device (PD) as some power is dissipated in the cable depended on RJ-45 cable (Cat3/Cat5/Cat5e..).
- Max. PSE Power (NCS2-POEIG802): 30W
- Max. PD Power: 25.5W

CHAPTER 3: HARDWARE SETUP

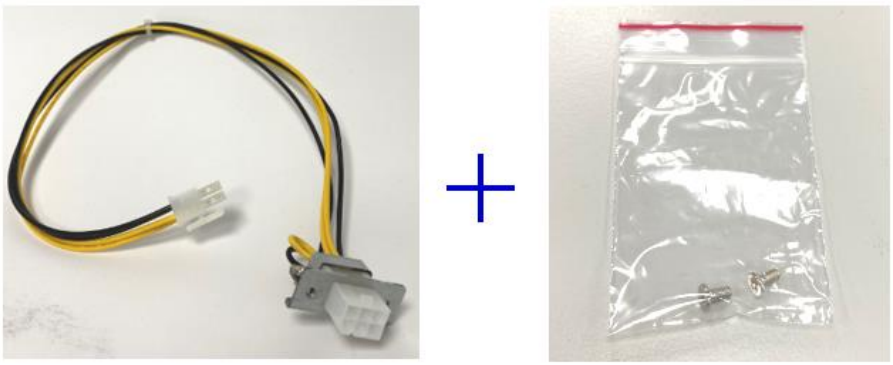
To access some components and perform certain service procedures, you must perform the following procedures first:

(1) To reduce the risk of personal injury, electric shock, or damage to the equipment, please remove all power sources

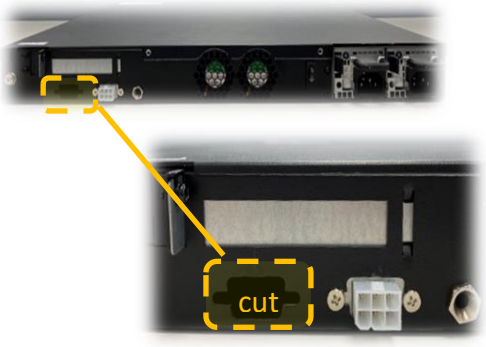
(2) Please wear ESD protected gloves before conducting the following steps.

Installing the PoE+ Adapter Cable

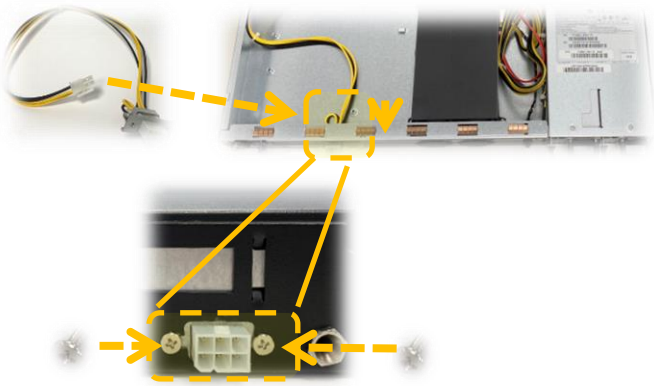
1. With the cable kit, this PoE+ NIC module can be installed onto the system chassis along with the module's power cable.



2. The colored area indicated in the picture is to be removed to accommodate the cable connector cage. Press on the four connected points with a flathead screwdriver to cut off the unneeded parts. It is recommended not to use your bare hands to tear apart the metal pieces in case of injury.

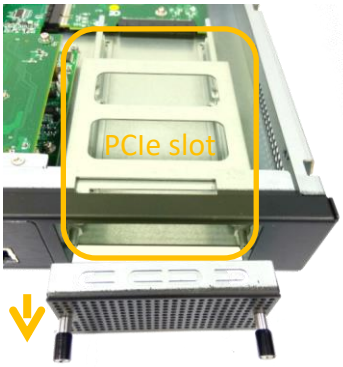


3. Align the screws holes on the cable connector cage as well as the holes on the chassis, and secure the cage onto the chassis using the screws.



4. The System comes with 1 NIC module slot for network bandwidth expansion. Please follow the steps for installation. Rotate clockwise and loosen the two lock-screws.

5. Remove the door and locate the PCIe slot for module insertion

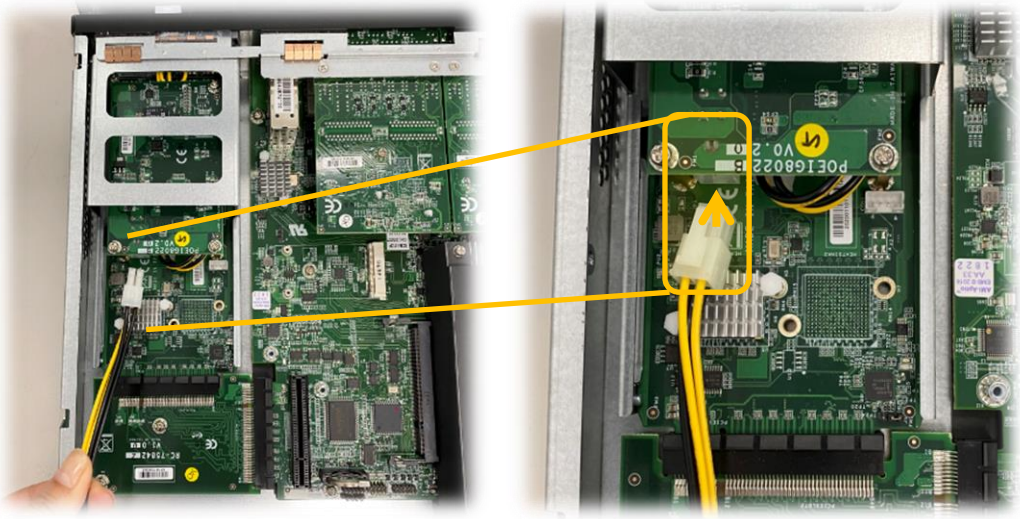


6. **Insert your PoE+ NIC module**. (The module shown in the image below is for reference only).

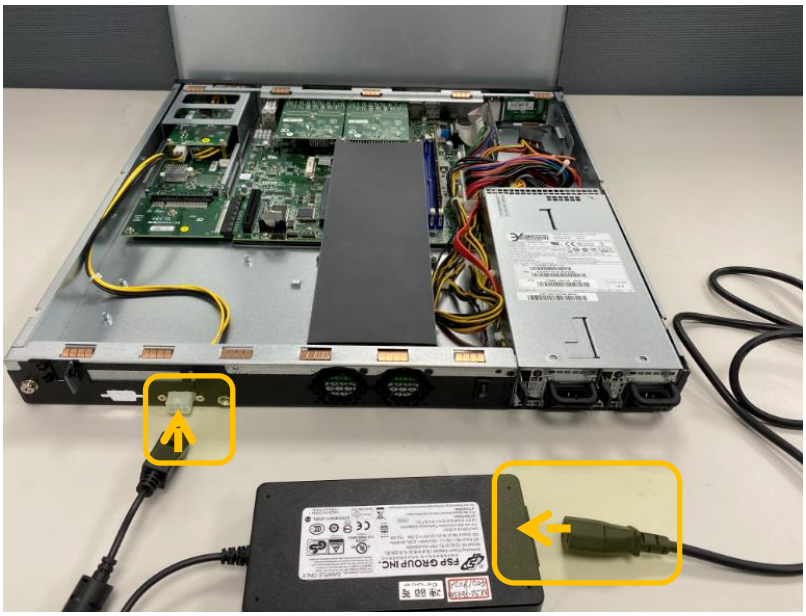
Once the module is firmly seated, rotate counter-clockwise and tighten the two lock-screws.

Align the gold fingers to the PCIe slot on the motherboard carefully while inserting this module.

7. Insert the cable into the power connector hole on the PoE+ NIC module



8. Insert the power adapter cable and then plug the power cord.



Warnings: Be sure to lock it well before power on.

CHAPTER 4: MODULE SUPPORT PACKAGE

- **Switch Channel**： Select POE module smbus path.
- **Select Slot**：Selects module slot, only support lanner AVL platform.
- **Select Port** ：Selects PoE module port.
- **Get Slot Info**：Show module firmware version.
- **Get Slot Support Port Number**：Show how many ports supported of this PoE module.
- **Get Slot Powered Port Number**：Show how many ports were powered of this PoE module.
- **Get Slot Current**：Show total output current of this PoE module.
- **Get Slot Icut**：Show total current shutdown value of this POE module.
- **Get Slot Max Power**：Show max output power value of this POE module.
- **Set Slot Max Power**：Set max output power value of this POE module.
- **Set Slot Default**：Restore to default value of this POE module.
- **Save Slot Setting**：Save current setting of this POE module.
- **Get Port Icut**：Show current shutdown value of this selected port.
- **Get Port Ilim**：Show current limitation value of this selected port.
- **Get Port Current**：Show output current of this selected port.
- **Get Port Voltage**：Show output voltage of this selected port.
- **Get Port Class**：Show PD class which was connected of this selected port.
- **Get Port Power State**：Show power status *(0 means disable, 1 means enable)* of this selected port.
- **Set Port Power State**：Set power *status (0 means disable, 1 means enable)* of this selected port.
- **Set Port Priority**：Set power priority of this selected port. *Lower priority port will be shutdown first if power is not enough.*
- **Get Port Priority**：Show power priority of this selected port.
- **Get Port Last Event**：Show last event record of this selected port.
- **Get Port Detail**：Show detail status of this selected port.
- **Get Port Fatal Event**：Show fatal event of this selected port.
- **Set Clear Fatal Event**：Clear fatal event flag of this selected port.
- **Get Slot Mode**：Show the slot mode of this selected port. *(0h -default means insert first and power first. 1h -pre-book means pre-set the provided electricity watt value.)*
- **Set Slot Mode**：Set the slot mode of this selected port. *(0h -default means insert first and power first. 1h -pre-book means pre-set the provided electricity watt value.)*
- **Get Port PreBook Class**：Show the pre-book class value of this selected port.
- **Set Port PreBook Class**：Set the pre-book class value of this selected port.

  00h - Class 0 (15.4W)

  01h - Class 1 (4W)

  02h - Class 2 (7W)

  03h - Class 3 (15.4W)

  04h - Class 4 (30W)

  99h - Unknown (Default)

Power over Ethernet Module

POE_SwitchChannel

The **POE_SwitchChannel** function set smbus switch channel, help user to select POE module smbus path

- **Syntax**

  int32_t POE_SwitchChannel(uint8_t ubAddr,uint8_t ubChannel)

- **Parameters**

  ubAddr

  [out] assigns the smbus switch address

  ubChannel

  [out] assigns smbus switch channel

  Return Value

- ERR_Success: function is successful

  ERR_NotSupport: function not support

  ERR_Error: function general access error

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubAddr, ubChannel;
        ubAddr = 0xE6;
        ubChannel =0x01;
        iRet = POE_SwitchChannel(ubAddr, ubChannel);
        if ( iRet == ERR_Success ) printf("---> POE_SwitchChannel switch address=0x%X,
channel=0x%X\n", ubAddr, ubChannel);
    return 0;
}
```

POE_SelectSlot

The **POE_SelectSlot** function selects module slot without study schmatic, only support specific lanner platform.

- **Syntax**

  int32_t POE_SelectSlot(uint8_t ubSlot);

- **Parameters**

  ubSlot

  [out] assign selected module slot number.

  Return Value

- ERR_Success: function is successful

  ERR_NotSupport: function not support

  ERR_Error: function general access error

  ERR_NotExist:Library file not found or Device not exist

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubslot;
        ubslot=1;
        iRet = POE_SelectSlot(ubslot);
        if ( iRet == ERR_Success ) printf("select Module number=0x%X\n", ubslot);
    return 0;
}
```

POE_SelectPort

The **POE_SelectPort** function selects module port.

- **Syntax**

  int32_t POE_SelectPort(uint8_t ubPort);

- **Parameters**

  ubPort

  [out] assign selected module port number.

  Return Value

- ERR_Success: function is successful

  ERR_NotSupport: function not support

  ERR_Error: function general access error

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubPort;
        ubPort=2;
        iRet = POE_SelectPort(ubPort);//
        if(iRet==0)
        {
          printf("Port : %X\n",ubPort);
        }
        else
        {
          printf("Select port fail, fail number : %x", iRet);
        }
    return 0;
}
```

POE_GetSlotInfo

The **POE_GetSlotInfo** function returns module firmware version.

- **Syntax**

  int32_t POE_GetSlotInfo(uint8_t* pubMajor,uint8_t* pubMinor);

- **Parameters**

  pubMajor

  [in] return POE module firmware major version.

  pubMinor

  [in] return POE module firmware minor version.

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t major_version;
  uint8_t minor_version;
        major_version=0;
        minor_version=0;
        iRet = POE_GetSlotInfo(&major_version,&minor_version);//
        if(iRet==0)
        {
          printf("major                    version:%x,                    minor
version:%x\n",major_version,minor_version);
        }
        else
        {
          printf("Get Device infomation fail, fail number : %x", iRet);

        }
    return 0;
}
```

POE_GetSlotSupportPortNumber

The **POE_GetSlotSupportPortNumber** function returns how many ports this module support.

- **Syntax**

  int32_t POE_GetSlotSupportPortNum(uint8_t* pubNumber);

- **Parameters**

  pubNumber

  [in] return counts of POE module supports port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t Number;
            printf("Get POE Module Support Port Number : \n");
            Number=1;
            iRet = POE_GetSlotSupportPortNum(&Number);//
            if(iRet==0)
            {
              printf("number : %d\n",Number);
            }
            else
            {
              printf("Get Support Port Number fail, fail number : %x", iRet);

            }
    return 0;
}
```

POE_GetSlotPoweredPortNumber

The **POE_GetSlotPoweredPortNumber** function returns how many ports were powered by this module.

- **Syntax**

  int32_t POE_GetSlotPoweredPortNum(uint8_t* pubNumber);

- **Parameters**

  pubNumber

  [in] return counts of POE module powered ports.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t Number;
      printf("Get Powered Port Number : \n");
      Number=0;
      iRet = POE_GetSlotPoweredPortNum(&Number);//
      if(iRet==0)
      {
        printf("number : %d\n",Number);
      }
      else
      {
        printf("Get powered Port count fail, fail number : %x", iRet);

      }
    return 0;
}
```

POE_GetSlotCurrent

The **POE_GetSlotCurrent** function returns total output currents by this module.

- **Syntax**
  - int32_t POE_GetSlotCurrent(float* pfCurrent);
- **Parameters**

  pfCurrent

  [in] return POE module output current value.

  Return Value
- ERR_Success: function is successful

  ERR_Error: function general access error
- **Remarks**
- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Current;
  Current=0;
printf("Get Slot Total Current : \n");
iRet = POE_GetSlotCurrent(&Current);//
if(iRet==0)
{
  printf("current:%6.2f(mA)\n",Current);
}
else
{
  printf("Get Slot Current fail, fail number : %x", iRet);

}
    return 0;
}
```

POE_GetSlotIcut

The **POE_GetSlotIcut** function returns total Icut valueof this POE module.

- **Syntax**
  - int32_t POE_GetSlotIcut(float* pfIcut);
- **Parameters**

  pfIcut

  [in] return POE module total Icut value.

  Return Value
- ERR_Success: function is successful

  ERR_Error: function general access error
- **Remarks**
- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Current;
  Current=0;
  printf("Get Slot Total Icut : \n");
  iRet = POE_GetSlotIcut(&Current);//
  if(iRet==0)
  {
    printf("total icut:%6.2f(mA)\n",Current);
  }
  else
  {
    printf("Get Slot total icut fail, fail number : %x", iRet);
  }
    return 0;
}
```

POE_GetSlotMaxPower

The **POE_GetSlotMaxPower** function returns max output power value of this POE module.

- **Syntax**

  int32_t POE_GetSlotMaxPower(uint16_t* puwPower);

- **Parameters**

  puwPower

  [in] return POE module max total output power value.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint16_t Power;
Power=0;
printf("Get PSE Max Power Setting : \n");
iRet = POE_GetSlotMaxPower(&Power);//
if(iRet==0)
{
  printf("pse max power:%d(W)\n",Power);
}
else
{
  printf("Get PSE Max Power fail, fail number : %x", iRet);

}
  return 0;
}
```

POE_SetSlotMaxPower

The **POE_SetSlotMaxPower** function set max output power value of this POE module.

- **Syntax**

  int32_t POE_SetSlotMaxPower(uint16_t uwPower);

- **Parameters**

  uwPower

  [out] set POE module max total output power.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint16_t uwPower;
uwPower=90;
printf("Set PSE Max Power : \n");
iRet = POE_SetSlotMaxPower(uwPower);//
if(iRet==0)
{
  printf("set pse max power:%d(W)\n",uwPower);
}
else
{
  printf("Set PSE Max Power fail, fail number : %x", iRet);

}
  return 0;
}
```

POE_SetSlotDefault

The **POE_SetSlotDefault** function set POE module back to default setting.

- **Syntax**

  int32_t POE_SetSlotDefault(void);

- **Parameters**
  - (None)

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**
- Poe module setting include:
  - PSE Total Power Setting
  - Port Control Setting
  - Port Priority Control Setting

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  iRet = POE_SetSlotDefault();
  if(iRet==0)
  {
    printf("Set POE Module back to default setting\n");
  }
  else
  {
    printf("Set POE Module back to default fail, fail number : %x", iRet);
  }
  return 0;
}
```

POE_SaveSlotSetting

The **POE_SaveSlotSetting** function save POE module current setting.

- **Syntax**

  int32_t POE_SaveSlotSetting(void);

- **Parameters**
  - (None)

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**
- Poe module setting include:
  - PSE Total Power Setting
  - Port Control Setting
  - Port Priority Control Setting

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  iRet = POE_SaveSlotSetting();
  if(iRet==0)
  {
    printf("Save setting to POE Module\n");
  }
  else
  {
    printf("Save POE Module setting fail, fail number : %x", iRet);
  }
  return 0;
}
```

POE_GetPortIcut

The **POE_GetPortIcut** function returns icut value of this port.

- **Syntax**

  int32_t POE_GetPortIcut(float* pfIcut);

- **Parameters**

  pfIcut

  [in] return Icut value of current select port.

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Current;
  Current =3;
  printf("Get Icut : \n");
  iRet = POE_GetPortIcut(&Current);//
  if(iRet==0)
  {
    printf("icut:%6.2f(mA)\n",Current);
  }
  else
  {
    printf("Get icut fail, fail number : %x", iRet);

  }
  return 0;
}
```

POE_GetPortIlim

The **POE_GetPortIlim** function returns ilim value of this port.

- **Syntax**

  int32_t POE_GetPortIlim(float* pfIlim);

- **Parameters**

  pfIlim

  [in] return Ilim value of current select port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Current;
  Current =3;
  printf("Get Ilim : \n");
  iRet = POE_GetPortIlim(&Current);//
  if(iRet==0)
  {
    printf("ilim:%6.2f(mA)\n",Current);
  }
  else
  {
    printf("Get ilim fail, fail number : %x", iRet);

  }
  return 0;
}
```

POE_GetPortCurrent

The **POE_GetPortCurrent** function returns output current of selected port.

- **Syntax**

  int32_t POE_GetPortCurrent(float* pfCurrent);

- **Parameters**

  pfCurrent

  [in] return output current of selected port.

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Current;
  Current =3;
  printf("Get Current : \n");
  iRet = POE_GetPortCurrent(&Current);//
  if(iRet==0)
  {
    printf("current:%6.2f(mA)\n",Current);
  }
  else
  {
    printf("Get Current fail, fail number : %x", iRet);

  }
  return 0;
}
```

POE_GetPortVoltage

The **POE_GetPortVoltage** function returns output voltage of selected port.

- **Syntax**

  int32_t POE_GetPortVoltage(float* pfVoltage);

- **Parameters**

  pfVoltage

  [in] return output voltage of selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  float Voltage;
  Voltage =3;
  printf("Get Voltage : \n");
  iRet = POE_GetPortVoltage(&Voltage);//
  if(iRet==0)
  {
    printf("voltage:%6.2f(mV)\n",Voltage);
  }
  else
  {
    printf("Get Voltage fail, fail number : %x", iRet);

  }
return 0;
}
```

POE_GetPortClass

The **POE_GetPortClass** function returns PD class which was connected with selected port.

- **Syntax**

  int32_t POE_GetPortClass(uint8_t* pubClass);

- **Parameters**

  pubClass

  [in] return PD class which connect to selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t Class;
  printf("Get Port Class: \n");
  Class=0;
  iRet = POE_GetPortClass(&Class);//
  if(iRet==0)
  {
    printf("Class : %x\n",Class);
  }
  else
  {
    printf("Get Port Class fail, fail number : %x", iRet);

  }
return 0;
}
```

POE_GetPortPowerState

The **POE_GetPortPowerState** function returns selected port power status, 0 represent disable, 1 represent enable.

- **Syntax**

  int32_t POE_GetPortPowerStat(uint8_t* pubState);

- **Parameters**

  pubState

  [in] return power status of selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t State;
  printf("Get Power State : \n");
  State=0;
  iRet = POE_GetPortPowerStat(&State);//
  if(iRet==0)
  {
    if(State == 1)printf("Enabled!\n");
    else if(State == 0)printf("Disable!\n");
    else printf("Wrong parameter\n");
  }
  else
  {
    printf("Get Power State fail, fail number : %x", iRet);

  }
return 0;
}
```

POE_SetPortPowerState

The **POE_SetPortPowerState** function set selected port power status, 0 represent disable, 1 represent enable.

- **Syntax**

  int32_t POE_SetPortPowerStat(uint8_t ubEnable);

- **Parameters**

  ubEnable

  [out] assign power status of selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t State;
  State=0;
  printf("Set Power State : \n");
  iRet = POE_SetPortPowerStat(State);//
  if(iRet==0)
  {
    if(State == 1)printf("Enabled!\n");
    else if(State == 0)printf("Disable!\n");
    else printf("Wrong parameter\n");
  }
  else
  {
    printf("Set Power State fail, fail number : %x", iRet);
  }
return 0;
}
```

POE_SetPortPriority

The **POE_SetPortPriority** function set selected port power priority, lower priority port will be shutdown first if power is not enough.

- **Syntax**

  int32_t POE_SetPortPriority(uint8_t ubPrior);

- **Parameters**

  ubPrior

  [out] assign power priority of selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- Priority range : 0~3

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubPrior;
  ubPrior =0;
  printf("Set Port Priority : \n");

  iRet = POE_SetPortPriority(ubPrior);//
  if(iRet==0)
  {
    printf("priority : %d\n",ubPrior);
  }
  else
  {
    printf("Set Port Priority fail, fail number : %x", iRet);

  }
return 0;
}
```

POE_GetPortPriority

The **POE_GetPortPriority** function return selected port power priority, lower priority port will be shutdown first if power is not enough.

- **Syntax**

  int32_t POE_GetPortPriority(uint8_t* pubPrior);

- **Parameters**

  pubPrior

  [in] return power priority of selected port.

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- Priority range : 0~3

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubPrior;
  ubPrior =0;
      printf("Get Port Priority : \n");
      iRet = POE_GetPortPriority(&ubPrior);//
      if(iRet==0)
      {
        printf("priority : %d\n",ubPrior);
      }
      else
      {
        printf("Get Port Priority fail, fail number : %x", iRet);
      }
return 0;
}
```

POE_GetPortLastEvent

The **POE_GetPortLastEvent** function returns selected port last event record.

- **Syntax**

  int32_t POE_GetPortLastEvent(uint8_t* pubEvent);

- **Parameters**

  pubEvent

  [in] return last event of selected port.

Return Value

- `ERR_Success: function is successful`

  `ERR_Error: function general access error`

- **Remarks**

- Event returns value description:
  - 0h – No event
  - 1h – Disconnect
  - 2h – Icut violation (Tcut)
  - 3h – Tstart violation
  - 4h – Denied, insufficient power
  - 5h – Revoked
  - Fh – Fatal Event

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t State;
      printf("Get Last Event : \n");
      State=0;
      iRet = POE_GetPortLastEvent(&State);//
      if(iRet==0)
      {
        if(State == 0)printf("No events\n");
        else if(State == 1)printf("Disconnect\n");
        else if(State == 2)printf("Icut violate\n");
        else if(State == 3)printf("Tstart violate\n");
        else if(State == 4)printf("Denied,insufficient power\n");
        else if(State == 5)printf("Revoked,insufficient power\n");
        else if(State == 6)printf("Revoked,insufficient power\n");
        else if(State == 15)printf("Fatal event happend\n");
        else printf("Wrong parameter\n");
      }
      else
      {
        printf("Get Last Event fail, fail number : %x", iRet);
      }
  return 0;
}
```

POE_GetPortDetail

The **POE_GetPortDetail** function return selected port detail status.

- **Syntax**

  int32_t POE_GetPortDetailStat(uint8_t* pubStat);

- **Parameters**

  pubStat

  [in] return port detail status of selected port.

  Return Value

- `ERR_Success: function is successful`

  `ERR_Error: function general access error`

- **Remarks**

- Port detail state description
  - 0h – Detect unknown
- 1h – Detect short
- 2h – Detect PD capacitance too high
- 3h – Detect signature resistance too low
- 4h – Detect good (but not powered on)
- 5h – Detect signature resistance too high
- 6h – Detect open
- 7h – Detect charged
- 8h – Port turning off (no new detect)
- 9h – Port turning on
- Ah – Port on
- Bh – Port on (legacy)
- Fh – Port Disabled

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t State;
      printf("Get Detail State : \n");
      State=0;
      iRet = POE_GetPortDetailStat(&State);//
      if(iRet==0)
      {
        if(State == 0x0)printf("Detect unknown\n");
        else if(State == 0x1)printf("Detect short\n");
        else if(State == 0x2)printf("Detect PD capacitance too high\n");
        else if(State == 0x3)printf("Detect signature resistance too low\n");
        else if(State == 0x4)printf("Detect good (but not powered on)\n");
        else if(State == 0x5)printf("Detect signature resistance too high\n");
        else if(State == 0x6)printf("Detect open\n");
        else if(State == 0x7)printf("Detect charged\n");
        else if(State == 0x8)printf("Port turning off (no new detect)\n");
        else if(State == 0x9)printf("Port turning on\n");
        else if(State == 0xA)printf("Port on\n");
        else if(State == 0xB)printf("Port on (legacy)\n");
        else if(State == 0xF)printf("Port Disabled\n");
        else printf("Wrong parameter\n");
      }
      else
      {
        printf("Get Detail State fail, fail number : %x", iRet);
      }
return 0;
}
```

POE_GetPortFatalEvent

The **POE_GetPortFatalEvent** function returns selected port fatal event.

- **Syntax**

  int32_t POE_GetPortFatalEvent(uint8_t* pubEvent);

- **Parameters**

  pubEvent

  [in] return last fatal event of selected port.

  Return Value

- `ERR_Success: function is successful`

  `ERR_Error: function general access error`

- **Remarks**

- Event returns value description:
  - Bit 0 – 48VTooLow
- Bit 1 – OutofVDD
- Bit 2 – MOSFETBad
- Bit 3 – OverTemp
- Bit 4 – IcutViolation
  - Bit 5 – TstartViolation
- Bit 7 – Abnormal

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t State;
      printf("Get Fatal Event : \n");
      State=0;
      iRet = POE_GetPortFatalEvent(&State);
      if(iRet==0){
        if(State != 0){
        if((State & FATAL_48VTooLow) != 0)printf("48V supply too low\n");
        if((State & FATAL_OutofVDD) != 0)printf("PSE out of VDD\n");
        if((State & FATAL_MOSFETBad) != 0)printf("FET bad\n");
        if((State & FATAL_OverTemp) != 0)printf("Over temperature\n");
        if((State & FATAL_IcutViolation) != 0)printf("Icut violation\n");
        if((State & FATAL_TstartViolation) != 0)printf("Tstart violation\n");
        if((State & FATAL_Abnormal) != 0)printf("Abnormal\n");
        }
        else{
          printf("No fatal\n");
        }
      }
      else{
        printf("Get Fatal Event fail, fail number : ", iRet);
      }
  return 0;
}
```

POE_SetClearFatalEvent

The **POE_SetClearFatalEvent** function clears port fatal event flag.

- **Syntax**

  int32_t POE_SetClearFatalEvent(void);

- **Parameters**

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  iRet = POE_SetClearFatalEvent();
  if(iRet==0)
  {
        printf("Clear POE Module event\n");
  }
  else
  {
      printf("Clear POE Module event fail, fail number : ", iRet);
  }
  return 0;
}
```

POE_GetSlotMode

The **POE_GetSlotMode** function returns selected slot mode.

- **Syntax**

  int32_t POE_GetSlotMode(uint8_t* pubMode);

- **Parameters**

  pubMode

  [in] return POE module slot mode, default or pre-book. 0: default 1: pre-book

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- Mode returns value description:
  - 0h –default
- 1h –pre-book
- **Example**

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubmode=0;
  printf("Get Slot mode: \n");
  iRet = POE_GetSlotMode(&ubmode);//
  if(iRet==0)
  {
      if(ubmode==0)printf("Slot is in default mode\n");
      else if(ubmode==1)printf("Slot is in pre-book mode\n");
      else printf("Get Slot mode: ", iRet);
  }
  else
  {
      printf("Get Slot mode fail, fail number: ", iRet);
  }
  return 0;
}
```

POE_SetSlotMode

The **POE_SetSlotMode** function set selected slot mode, POE module support two different mode: default mode judge whether supply power or not by the plug-in port priority and how much power can supply, Pre-book mode will book some power before judgement.

- **Syntax**

  int32_t POE_SetSlotMode(uint8_t ubMode);

- **Parameters**

  ubMode

  [out] assign mode of selected slot.

  Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- Mode value 0: Default mode

- Mode value 1: Pre-book mode

  Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubmode=0;
  printf("Set Slot mode: \n");
  iRet = POE_SetSlotMode(ubmode);//
  if(iRet==0)
  {
      if(ubmode==0)printf("Slot is in default mode\n");
      else if(ubmode==1)printf("Slot is in pre-book mode\n");
      else printf("Set Slot mode: ", iRet);
  }
  else
  {
    printf("Set Slot mode fail, fail number: ", iRet);
  }
return 0;
}
```

POE_GetPortPreBookClass

The **POE_GetPortPreBookClass** function return selected port prebook class value.

- **Syntax**

  int32_t POE_GetPortPreBookClass(uint8_t* pubClass);

- **Parameters**

  pubClass

  [in] return prebook class value which selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubbook=0;
        printf("Get Port Booking Class : \n");

        iRet = POE_GetPortPreBookClass(&ubbook);//
        if(iRet==0)
        {
          if(ubbook==99)printf("not booked\n");
          else printf("Class : %d\n",ubbook);
        }
        else
        {
          printf("Get Port Booking Class fail, fail number : ", iRet);
        }
return 0;
}
```

POE_SetPortPreBookClass

The **POE_SetPortPreBookClass** function set selected port prebook class value.

- **Syntax**

  int32_t POE_SetPortPreBookClass(uint8_t ubClass);

- **Parameters**

  ubClass

  [out] set prebook class value which selected port.

Return Value

- ERR_Success: function is successful

  ERR_Error: function general access error

- **Remarks**

- (None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "poelib.h"
int main(int argc, char* argv[])
{
  int32_t iRet;
  uint8_t ubbook=0;
        printf("Set Port Booking Class : \n");

        iRet = POE_SetPortPreBookClass(ubbook);//
        if(iRet==0)
        {
          printf("Class : %d\n",ubbook);
        }
        else
        {
          printf("Set Port Booking Class fail, fail number : ", iRet);
        }
  return 0;
}
```

CHAPTER 5: POE MCU IMPLEMENTATION

Command Summary

Command	Function Description	Access Mode
Global configuration		
0x01	Querying MCU Major Version	Read Only
0x02	Querying MCU Minor Version	Read Only
0x03	Querying and Setting PSE Power Management Mode	Read/Write
0x04	Saving Current Setting to Flash	Write Only
0x0A	Querying and Setting Max Total PSE Power	Read/Write
0x0B		
0x0E	Setting Modules Back to Factory Default	Write Only
0x0F	Module Reset Control Register	Write Only
Global Status		
0x22	Querying Number of Ports Present Status Register	Read Only
0x23	Querying Number of Ports Powered Status Register	Read Only
0x24	Querying Aggregate Icut Current Status Register	Read Only
0x25		
0x26	Querying Aggregate Measured Current Status Register	Read Only
0x27		
Port Control/Status		
0x40	Querying and Setting Port Access Select Control Register	Read/Write
0x41	Querying and Setting Port Control Register	Read/Write
0x42	Querying Port Present Status Register	Read Only
0x43		
0x44	Querying and Setting Port Priority Control Register	Read/Write
0x45	Querying Port Icut Control Register	Read Only
0x46	Querying Port Ilim Control Register	Read Only
0x48	Querying Port Measured Current Status Register	Read Only
0x49		
0x4A	Querying Port Measured Voltage Status Register	Read Only
0x4B		
0x4C	Querying Port Measured Class Status Register	Read Only
0x4D	Querying and Setting Port Pre-booking Power Control Register	Read/Write
0x4E	Querying Port Fatal Event Status Register	Read Only
0x4F	Clearing Port Fatal Event Status Register	Write Only

Command	Function Description	Access Mode
Secondary Bootloader Commands		
0xA0	Querying MCU current firmware type	Read Only
0xA1	Querying MCU Minor Version	Read Only
0xA2	MCU Reset	Write Only
0xA3	Switch to Secondary bootloader	Write Only

Global Configuration Command

mcu_major_ver (Address 01h): MCU Major Version Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
7:0	mcu_major_ver	Get MCU major version

mcu_minor_ver (Address 02h): MCU Minor Version Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
7:0	mcu_minor_ver	Get MCU minor version

pwr_mgt_mod (Address 03h): POE Power Management Mode Register. Read/Write. Saved to Flash upon request.

BIT(s)	SYMBOL	DESCRIPTION
2:0	pwr_mgt_mod	Querying and setting MCU management POE power mode. Power Management Mode is as follow: 00h – First plug first give & Priority 01h – Power-booking mode

flashcmd (Address 04h): Flash Command Register. Write Only.

BIT(s)	SYMBOL	DESCRIPTION
0	flash	When this bit is set all flashable values are saved to on-chip flash. Save current value to Flash including the following setting: **PSE Total Power Setting** (the current running value in RAM will be saved) **Power Management Mode Setting** **Port Control Setting** **Port Priority Control Setting**

max_total_pse_pwr (Address 0B-0Ah): Maximum Total PSE Power Register. Read/Write. Saved to Flash upon request.

Address 0Ah

BIT(s)	SYMBOL	DESCRIPTION
7:0	max_total_pse_pwr (lsb)	The maximum watt value currently provided by the PSE, so that the F/W can effectively control the Power On/Off of the PD. Writes to the LSB register are not accepted by F/W until a write to the MSB register has been completed. Units are 1 W/LSB.

Address 0Bh

BIT(s)	SYMBOL	DESCRIPTION
15:8	max_total_pse_pwr (msb)	Upper byte of address 0Ah

load_factory_default (Address 0Eh): Load Factory Default Register. Write Only.

BIT(s)	SYMBOL	DESCRIPTION
0	load_factory_default	When this bit is set, module will be back to the factory default values.

module_reset (Address 0Fh): Module Reset Control Register. Write Only.

BIT(s)	SYMBOL	DESCRIPTION
0	module_reset	Module will be reset when this bit is set.

Global Status Command

ports_present (Address 22h): # of Ports Present Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
6:0	ports_present	Number of ports physically present in this system. Based on address polling.

ports_powered (Address 23h): # of Ports Powered Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
6:0	ports_powered	# of ports currently powered in this system.

aggr_cut_cur (Address 25-24h): Aggregate Icut Current Status Register. Read Only.

Address 24h

BIT(s)	SYMBOL	DESCRIPTION
7:0	aggr_cut_cur (lsb)	Sum of powered ports' Icut values. After reading the LSB register, F/W updates to the MSB register are stalled unit the subsequent read to the MSB register is complete. This guarantees cohesive MSB/LSB behavior. Units are 18.75 mA/LSB.

Address 25h

BIT(s)	SYMBOL	DESCRIPTION
15:8	aggr_cut_cur (msb)	Upper byte of address 24h

aggr_measured_cur (Address 27-26h): Aggregate Measured Current Status Register. Read Only.

Address 26h

BIT(s)	SYMBOL	DESCRIPTION
7:0	aggr_measured_cur (lsb)	Sum of powered ports' measured current values. After reading the LSB register, F/W updates to the MSB register are stalled unit the subsequent read to the MSB register is complete. This quarantees cohesive MSB/LSB behavior. Units are 18.75 mA/LSB.

Address 27h

BIT(s)	SYMBOL	DESCRIPTION
15:8	aggr_measured_cur (msb)	Upper byte of address 26h

Port Control/Status Command

port_select (Address 40h): Port Access Select Control Register. Read/Write.

BIT(s)	SYMBOL	DESCRIPTION
6:0	port_select	ID number of the port whose control/status space is currently exposed in port section. Only a single port's control and status can be exposed at one time.

port_control (Address 41h): Port Control Register. Read/Write. Saved to Flash upon request.

BIT(s)	SYMBOL	DESCRIPTION
1	disable	Port disable 1: Disable 0: Non-disable (default)
0	reserved	

port_state (Address 43-42h): Port Present Status Register. Read Only.

Address 42h

BIT(s)	SYMBOL	DESCRIPTION
7:4	event	Port most recent event 0h – No event 1h – Disconnect 2h – Icut violation (Tcut) 3h – Tstart violation 4h – Denied, insufficient power Fh – Fatal Event, refer to ***port_fatevt*** register (**4Eh**)
3:0	state	Port current state 0h – Detect unknown 1h – Detect short 2h – Detect PD capacitance too high 3h – Detect signature resistance too low 4h – Detect good (but not powered on) 5h – Detect signature resistance too high 6h – Detect open 7h – Detect charged 8h – Port turning off (no new detect) 9h – Port turning on Ah – Port on Bh – Port on (legacy) Ch – Fatal event Fh – Port Disabled

Address 43h

BIT(s)	SYMBOL	DESCRIPTION
7	present	Port is physically present as determined by polling I2C interface. Port is present if associated port responds with valid devid.
6:0	reserved	reserved

port_priority (Address 44h): Port Priority Control Register. Read/Write.

BIT(s)	SYMBOL	DESCRIPTION
1:0	priority	Port priority level A port can be assigned any priority between 0 and 3. A value of 0 (default) represents the lowest priority port (first to be turned off).

port_icut (Address 45h): Port Icut Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
5:0	icut	Port Icut value is posted after class events which result in successful power up. Units are 18.75 mA/LSB.

port_ilim (Address 46h): Port Ilim Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
4:0	ilim	Port Ilim value is posted after class events which result in successful power up. Values are enumerated as follows: 04h – 425mA 08h – 850mA Others – reserved

port_im (Address 49-48h): Port Measured Current Status Register. Read Only.

Address 48h

BIT(s)	SYMBOL	DESCRIPTION
7:0	port_im (lsb)	Port's measured current. Units are 122.07 uA/LSB.

Address 49h

BIT(s)	SYMBOL	DESCRIPTION
15:8	port_im (msb)	Upper byte of address 48h

port_vm (Address 4B-4Ah): Port Measured Voltage Status Register. Read Only.

Address 4Ah

BIT(s)	SYMBOL	DESCRIPTION
7:0	port_vm (lsb)	Port's measured voltage. Units are 5.835 mV/LSB.

Address 4Bh

BIT(s)	SYMBOL	DESCRIPTION

15:8	port_im (msb)	Upper byte of address 4Ah

port_cm (Address 4Ch): Port Measured Class Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
5:0	port_cm	Port's measured class. 00h – Unknown (Default) 01h – Class 1 (4W) 02h – Class 2 (7W) 03h – Class 3 (15.4W) 04h – Class 4 (30W) 05h – Reserved 06h – Class 0 (15.4W) 07h – Overcurrent

port_prebook_pwr (Address 4Dh): Port Pre-booking Power Control Register (only for **Pre-booking Power** mode). Read/Write.

BIT(s)	SYMBOL	DESCRIPTION
5:0	port_prebook_pwr	Port pre-booking state 00h – Unknown (Default) 01h – Class 1 (4W) 02h – Class 2 (7W) 03h – Class 3 (15.4W) 04h – Class 4 (30W) 05h – Reserved 06h – Class 0 (15.4W)

port_fatevt (Address 4Eh): Port Fatal Event Status Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
7:0	port_fatevt	Port most recent fatal event Bit 0 – UVLO48. 48V supply is too low Bit 1 – UVLO3. PSE comes out of VDD Bit 2 – FETBAD. MOSFET failed Bit 3 – OVERTEMP. Over temperature Bit 4 – Icut violation (Tcut). Bit 5 – Tstart violation. Bit 6 – Reserved Bit 7 – Abnormal, such as Power Adapter Loss, I2C Bus Fail

Secondary Bootloader Commands

mcu_firmware _type (Address A0h): MCU firmware type Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
7:0	mcu_firmware_type	Get MCU firmware type 1:bootloader 2:bypass 3:POE

mcu_minor_ver (Address A1h): MCU Minor Version Register. Read Only.

BIT(s)	SYMBOL	DESCRIPTION
7:0	mcu_minor_ver	Get MCU minor version

mcu_reset (Address A2h): MCU Reset Register. Write only.

BIT(s)	SYMBOL	DESCRIPTION
0	MCU reset	MCU will be reset while this bit is set

switch_to_sbl (Address 04h): SBL switch Command Register. Write Only.

BIT(s)	SYMBOL	DESCRIPTION
0	Switch to SBL	MCU will switch to secondary bootloader when this bit is set.

APPENDIX A: FAQ

Q: How many ports was PoE device (suppose max power consumption is 12.95W per PD) supported by NCS2-POEIG802A with 150W power adapter?

Ans: All of 8ports can support POE port.

Formula: 15.4Watt x 8pcs PD = 123.2Watt <150Watt power adapter

Q: How many ports was PoE+ device (suppose max power consumption is 25.5W per PD) supported by NCS2-POEIG802A with 150W power adapter?

Ans: Any 5 of 8 port can support POE+ .

Formula: 30Watt x 5pcs PD = 150Watt =150Watt power adapter

Q: Corresponding to the previous question, if I wants to plug full 8pcs PD(30W) with all 8ports, how can I do?

Ans: There is an 270W power adapter for optional to supply enough power on each port *(see optional parts of ordering information) , but this is project base and MOQ 300 is required* . Formula: 30Watt x 8pcs PD = 240Watt < 270Watt power adapter

Q: If I want to plug full 8pcs PD, but set limitation max 15W average per port to ensure every PD can work at the same time, what should I do?

Ans: You can download the MSP(Module Support Package) from website and command set prebook class value which selected port. (See more detail in user's Manual.)

Syntax : int32_t POE_SetPortPreBookClass(uint8_t ubClass);

APPENDIX B: TERMS AND CONDITIONS

Warranty Policy

1. All products are under warranty against defects in materials and workmanship for a period of one year from the date of purchase.
2. The buyer will bear the return freight charges for goods returned for repair within the warranty period; whereas the manufacturer will bear the after service freight charges for goods returned to the user.
3. The buyer will pay for the repair (for replaced components plus service time) and transportation charges (both ways) for items after the expiration of the warranty period.
4. If the RMA Service Request Form does not meet the stated requirement as listed on "RMA Service," RMA goods will be returned at customer's expense.
5. The following conditions are excluded from this warranty:
   - Improper or inadequate maintenance by the customer
   - Unauthorized modification, misuse, or reversed engineering of the product
   - Operation outside of the environmental specifications for the product.

RMA Service

Requesting an RMA#

1. To obtain an RMA number, simply fill out and fax the "RMA Request Form " to your supplier.
2. The customer is required to fill out the problem code as listed. If your problem is not among the codes listed, please write the symptom description in the remarks box.
3. Ship the defective unit(s) on freight prepaid terms. Use the original packing materials when possible.
4. Mark the RMA# clearly on the box.

Note: Customer is responsible for shipping damage(s) resulting from inadequate/loose packing of the defective unit(s). All RMA# are valid for 30 days only; RMA goods received after the effective RMA# period will be rejected.

RMA Service Request Form

When requesting RMA service, please fill out the following form. Without this form enclosed, your RMA cannot be processed.

RMA No:	Reasons to Return: ▫ Repair(Please include failure details) ▫ Testing Purpose
Company:	Contact Person:
Phone No.	Purchased Date:
Fax No.:	Applied Date:
Return Shipping Address: _____	
Shipping by: ▫ Air Freight ▫ Sea ▫ Express_____ ▫ Others:_____	

Item	Model Name	Serial Number	Configuration

Item	Problem Code	Failure Status

*Problem Code:

01:D.O.A.	07: BIOS Problem	13: SCSI	19: DIO
02: Second Time R.M.A.	08: Keyboard Controller Fail	14: LPT Port	20: Buzzer
	09: Cache RMA Problem	15: PS2	21: Shut Down
03: CMOS Data Lost	10: Memory Socket Bad	16: LAN	22: Panel Fail
04: FDC Fail	11: Hang Up Software	17: COM Port	23: CRT Fail
05: HDC Fail	12: Out Look Damage	18: Watchdog Timer	24: Others (Pls specify)
06: Bad Slot			

Request Party ***Confirmed By Supplier***

Authorized Signature / Date **Authorized Signature / Date**