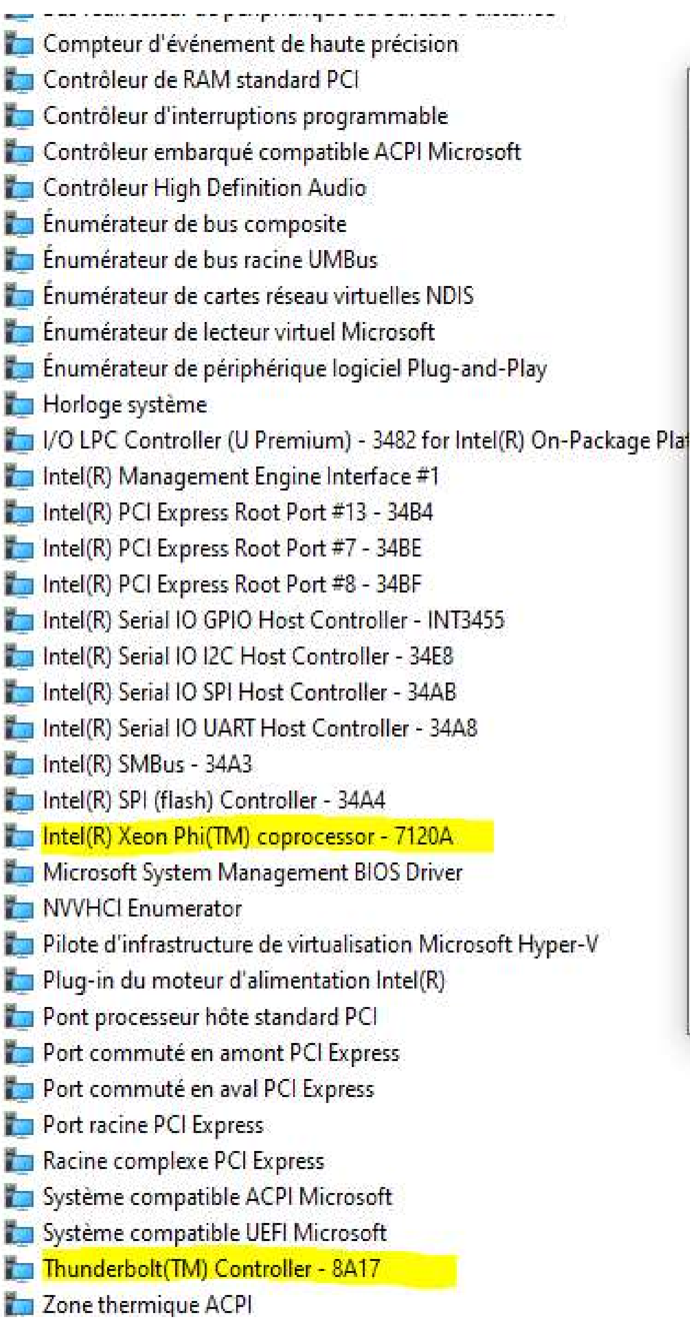
I did some testing with my akitio to connect my intel xeon phi 7210A to my chuwi box pro as I saw it had a thunderbolt port.
The result is conclusive and I can communicate with the card and test some code.
I installed mpss 3.8.6 under windows 10 home and tested the following programs:
-micinfo OK/KO because it can't read the pci bus
-micckeck OK
-micctrl OK
-micflash OK
-micsmc and micsmc-gui can't detect the card (I think you have to change the code that uses mic_pci_config)

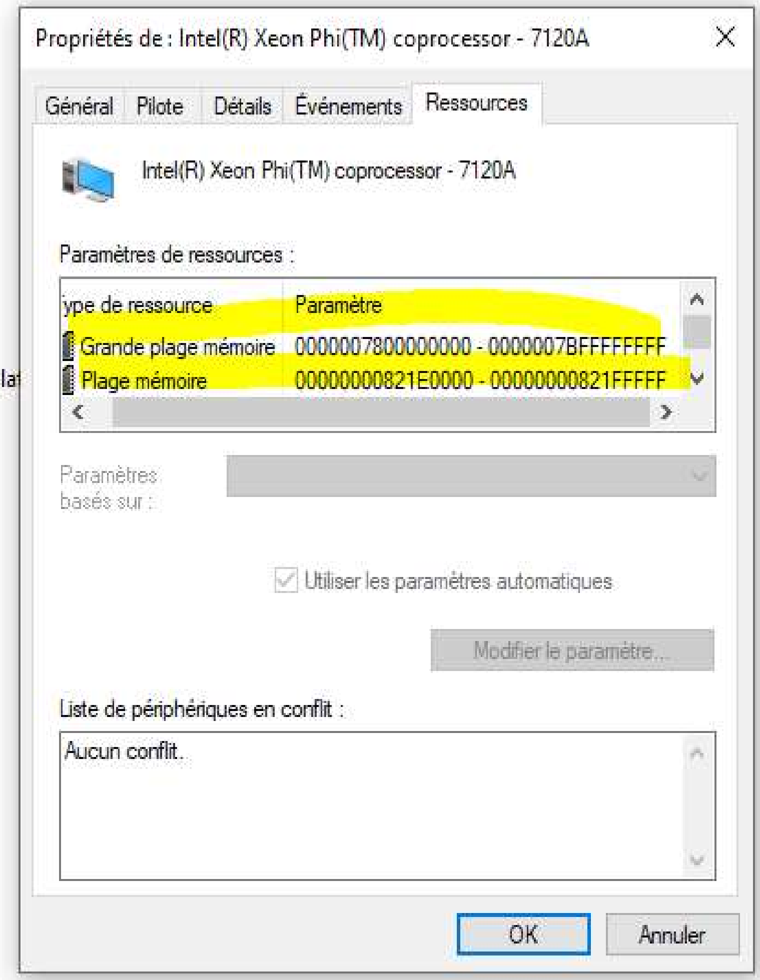
I can connect to the card via putty and run compiler programs from the host (command line or via visual studio and intel compilers)

The mini pc chuwi box pro (UEFI) does not show an option from the bios to enable "4G decoding above" mode but it should take the activation of I/O in 64 bits memory.

I show you some pictures below of my manipulations and tests of some programs that are in the mpss sdk:



Compteur d'événement de haute précision
Contrôleur de RAM standard PCI
Contrôleur d'interruptions programmable
Contrôleur embarqué compatible ACPI Microsoft
Contrôleur High Definition Audio
Énumérateur de bus composite
Énumérateur de bus racine UMBus
Énumérateur de cartes réseau virtuelles NDIS
Énumérateur de lecteur virtuel Microsoft
Énumérateur de périphérique logiciel Plug-and-Play
Horloge système
I/O LPC Controller (U Premium) - 3482 for Intel(R) On-Package Plat
Intel(R) Management Engine Interface #1
Intel(R) PCI Express Root Port #13 - 34B4
Intel(R) PCI Express Root Port #7 - 34BE
Intel(R) PCI Express Root Port #8 - 34BF
Intel(R) Serial IO GPIO Host Controller - INT3455
Intel(R) Serial IO I2C Host Controller - 34E8
Intel(R) Serial IO SPI Host Controller - 34AB
Intel(R) Serial IO UART Host Controller - 34A8
Intel(R) SMBus - 34A3
Intel(R) SPI (flash) Controller - 34A4
Intel(R) Xeon Phi(TM) coprocessor - 7120A
Microsoft System Management BIOS Driver
NVVHCI Enumerator
Pilote d'infrastructure de virtualisation Microsoft Hyper-V
Plug-in du moteur d'alimentation Intel(R)
Pont processeur hôte standard PCI
Port commuté en amont PCI Express
Port commuté en aval PCI Express
Port racine PCI Express
Racine complexe PCI Express
Système compatible ACPI Microsoft
Système compatible UEFI Microsoft
Thunderbolt(TM) Controller - 8A17
Zone thermique ACPI



Propriétés de : Intel(R) Xeon Phi(TM) coprocessor - 7120A

Intel(R) Xeon Phi(TM) coprocessor - 7120A

Paramètres de ressources :

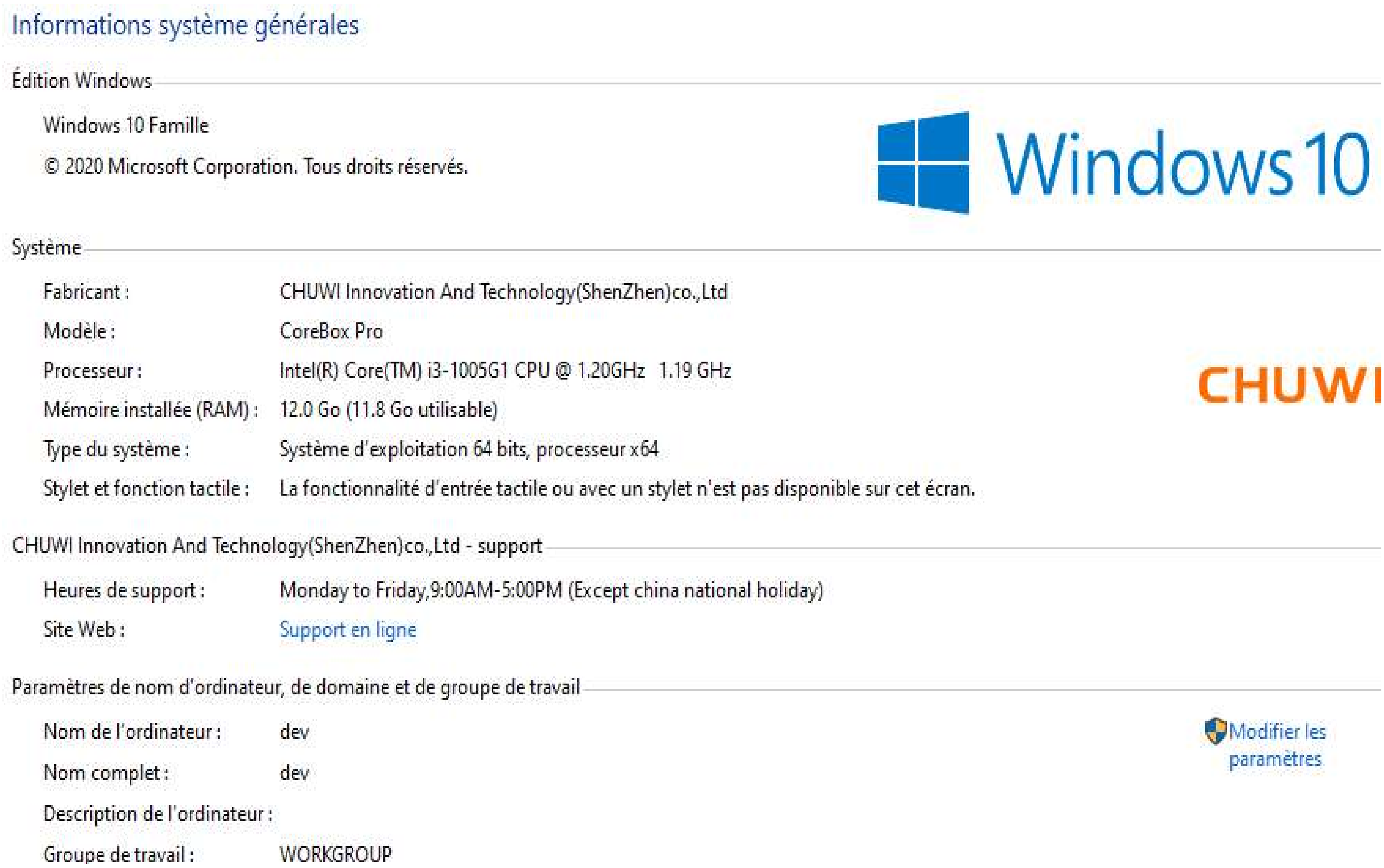
ype de ressource	Paramètre
Grande plage mémoire	0000007800000000 - 0000007BFFFFFFFF
Plage mémoire	00000000821E0000 - 00000000821FFFFF

Paramètres basés sur :

☑ Utiliser les paramètres automatiques

Liste de périphériques en conflit :

Aucun conflit.



Informations système générales

Édition Windows

Windows 10 Famille

© 2020 Microsoft Corporation. Tous droits réservés.

Système

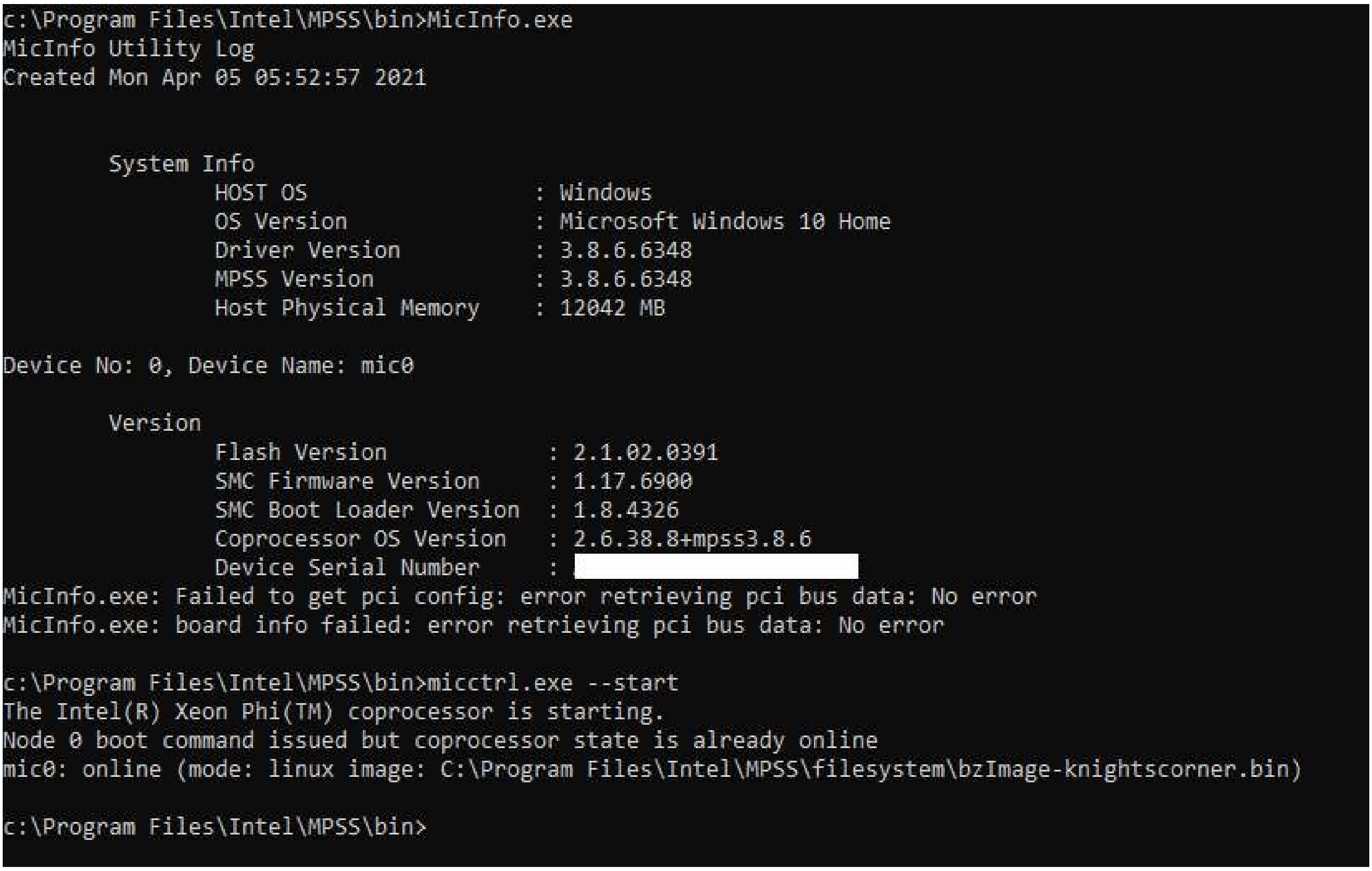
Fabricant :	CHUWI Innovation And Technology(ShenZhen)co.,Ltd
Modèle :	CoreBox Pro
Processeur :	Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz
Mémoire installée (RAM) :	12.0 Go (11.8 Go utilisable)
Type du système :	Système d'exploitation 64 bits, processeur x64
Stylet et fonction tactile :	La fonctionnalité d'entrée tactile ou avec un stylet n'est pas disponible sur cet écran.

CHUWI Innovation And Technology(ShenZhen)co.,Ltd - support

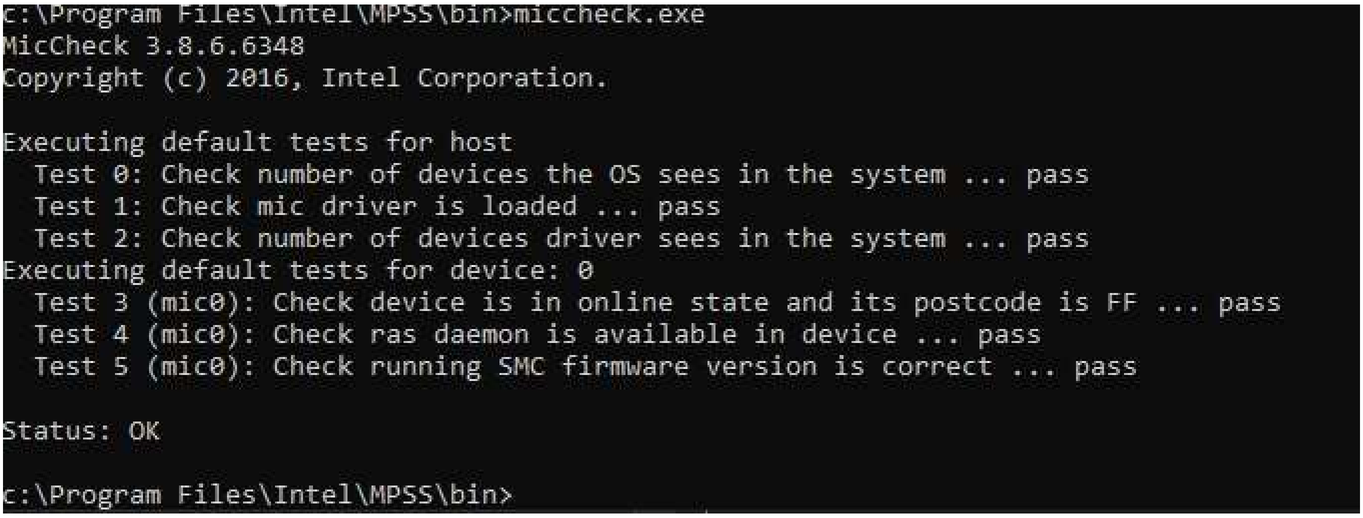
Heures de support :	Monday to Friday,9:00AM-5:00PM (Except china national holiday)
Site Web :	Support en ligne

Paramètres de nom d'ordinateur, de domaine et de groupe de travail

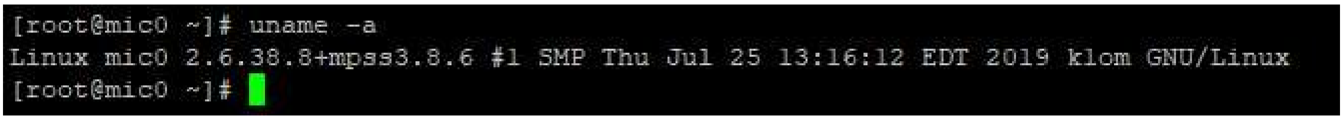
Nom de l'ordinateur :	dev
Nom complet :	dev
Description de l'ordinateur :	
Groupe de travail :	WORKGROUP



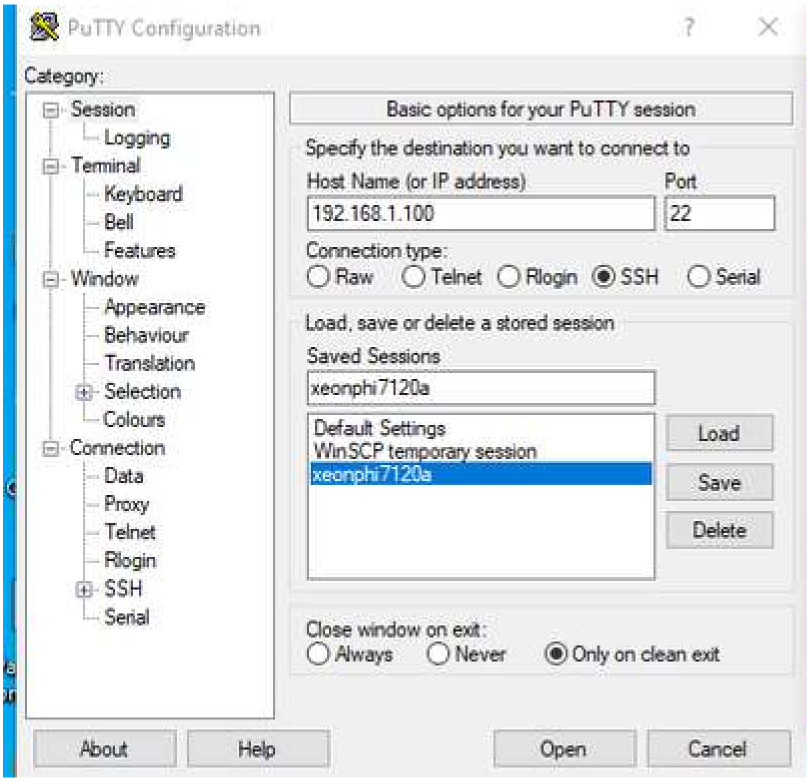
```
c:\Program Files\Intel\MPSS\bin>MicInfo.exe
MicInfo Utility Log
Created Mon Apr 05 05:52:57 2021


        System Info
                HOST OS                 : Windows
                OS Version              : Microsoft Windows 10 Home
                Driver Version          : 3.8.6.6348
                MPSS Version            : 3.8.6.6348
                Host Physical Memory    : 12042 MB

Device No: 0, Device Name: mic0

        Version
                Flash Version            : 2.1.02.0391
                SMC Firmware Version     : 1.17.6900
                SMC Boot Loader Version  : 1.8.4326
                Coprocessor OS Version   : 2.6.38.8+mpss3.8.6
                Device Serial Number     : [redacted]
MicInfo.exe: Failed to get pci config: error retrieving pci bus data: No error
MicInfo.exe: board info failed: error retrieving pci bus data: No error

c:\Program Files\Intel\MPSS\bin>micctrl.exe --start
The Intel(R) Xeon Phi(TM) coprocessor is starting.
Node 0 boot command issued but coprocessor state is already online
mic0: online (mode: linux image: C:\Program Files\Intel\MPSS\filesystem\bzImage-knightscorner.bin)

c:\Program Files\Intel\MPSS\bin>
```



```
c:\Program Files\Intel\MPSS\bin>miccheck.exe
MicCheck 3.8.6.6348
Copyright (c) 2016, Intel Corporation.

Executing default tests for host
  Test 0: Check number of devices the OS sees in the system ... pass
  Test 1: Check mic driver is loaded ... pass
  Test 2: Check number of devices driver sees in the system ... pass
Executing default tests for device: 0
  Test 3 (mic0): Check device is in online state and its postcode is FF ... pass
  Test 4 (mic0): Check ras daemon is available in device ... pass
  Test 5 (mic0): Check running SMC firmware version is correct ... pass

Status: OK

c:\Program Files\Intel\MPSS\bin>
```



```
[root@mic0 ~]# uname -a
Linux mic0 2.6.38.8+mpss3.8.6 #1 SMP Thu Jul 25 13:16:12 EDT 2019 k1om GNU/Linux
[root@mic0 ~]#
```



Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.100

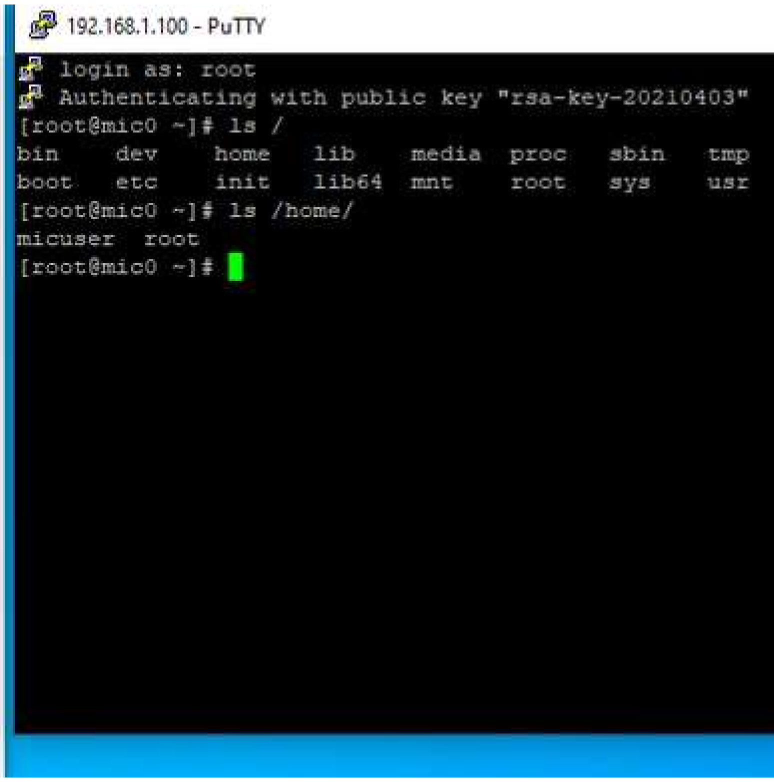
Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

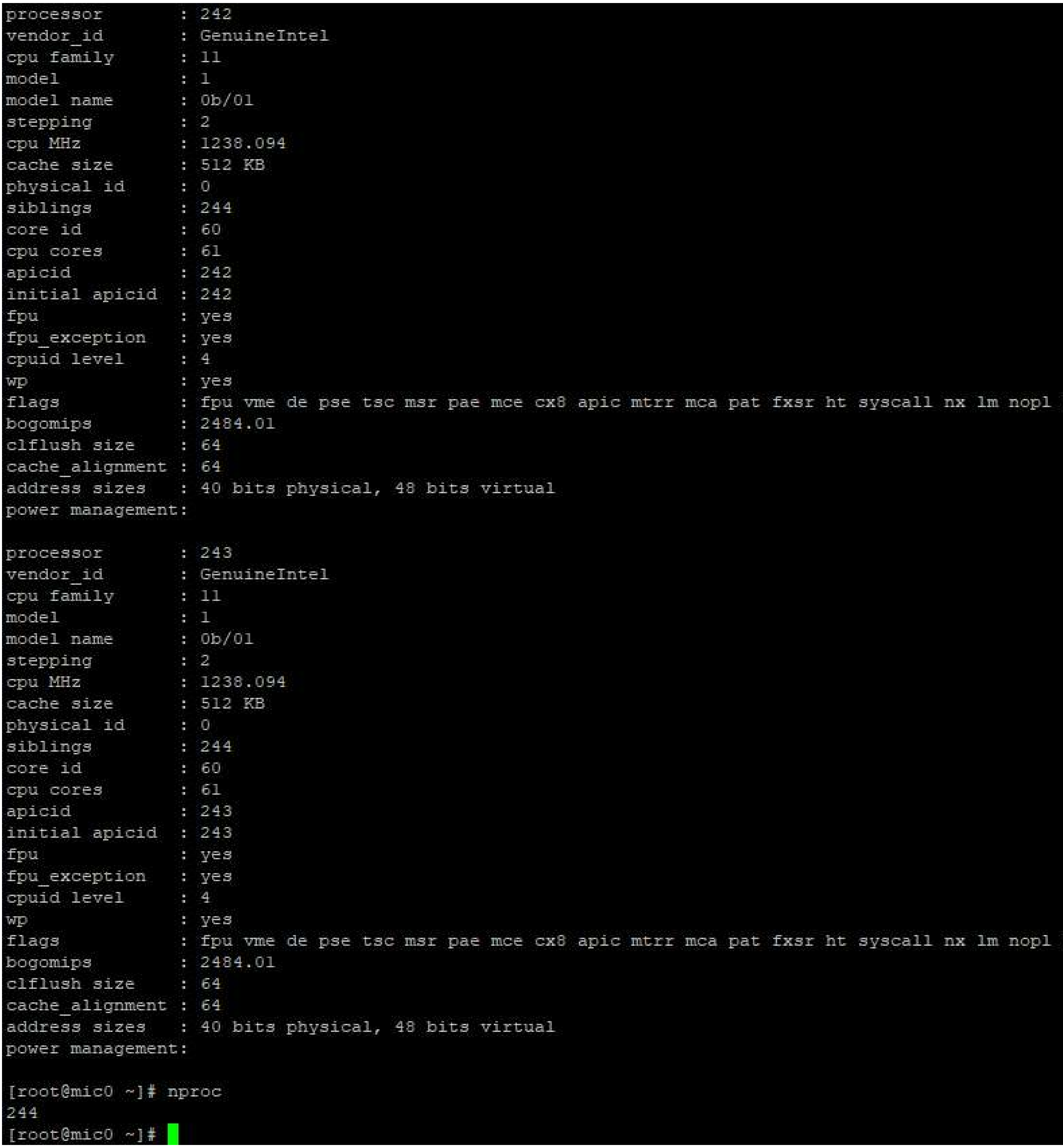
Saved Sessions: xeonphi7120a

Default Settings
WinSCP temporary session
xeonphi7120a

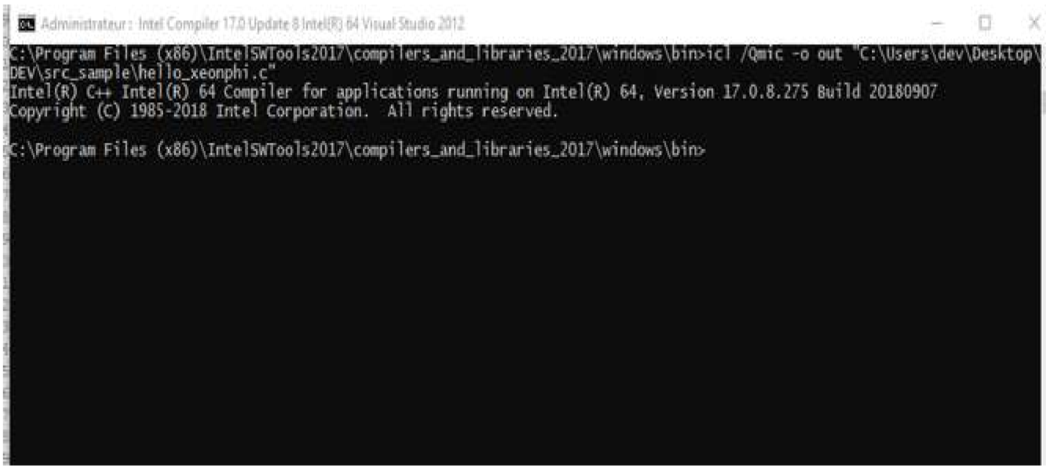
Close window on exit: ○ Always ○ Never ● Only on clean exit



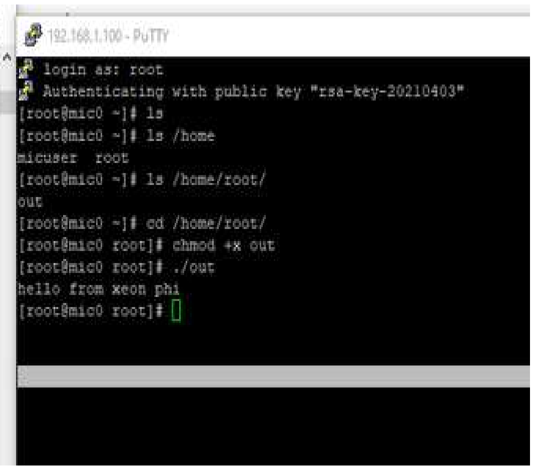
```
login as: root
Authenticating with public key "rsa-key-20210403"
[root@mic0 ~]# ls /
bin     dev     home    lib     media   proc    sbin    tmp
boot    etc     init    lib64   mnt     root    sys     usr
[root@mic0 ~]# ls /home/
micuser  root
[root@mic0 ~]#
```



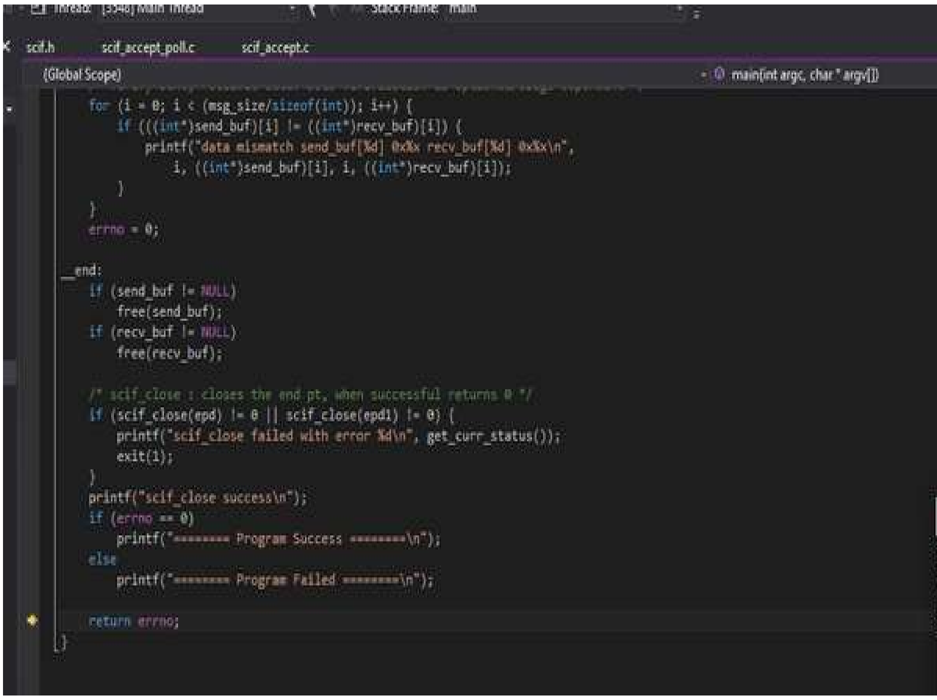
```
processor       : 242
vendor_id       : GenuineIntel
cpu family      : 11
model           : 1
model name      : 0b/01
stepping        : 2
cpu MHz         : 1238.094
cache size      : 512 KB
physical id     : 0
siblings        : 244
core id         : 60
cpu cores       : 61
apicid          : 242
initial apicid  : 242
fpu             : yes
fpu_exception   : yes
cpuid level     : 4
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr mca pat fxsr ht syscall nx lm nopl
bogomips        : 2484.01
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

processor       : 243
vendor_id       : GenuineIntel
cpu family      : 11
model           : 1
model name      : 0b/01
stepping        : 2
cpu MHz         : 1238.094
cache size      : 512 KB
physical id     : 0
siblings        : 244
core id         : 60
cpu cores       : 61
apicid          : 243
initial apicid  : 243
fpu             : yes
fpu_exception   : yes
cpuid level     : 4
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr mca pat fxsr ht syscall nx lm nopl
bogomips        : 2484.01
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

[root@mic0 ~]# nproc
244
[root@mic0 ~]#
```


Administrateur : Intel Compiler 17.0 Update 8 Intel(R) 64 Visual Studio 2012

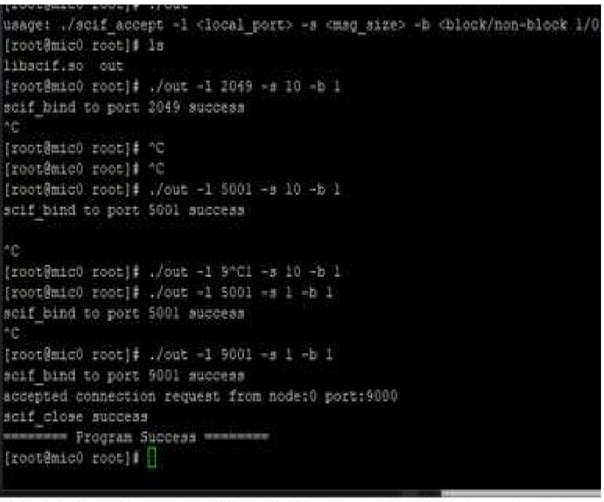
```
C:\Program Files (x86)\IntelSWTools2017\compilers_and_libraries_2017\windows\bin>icl /Qmic -o out "C:\Users\dev\Desktop\
DEV\src_sample\hello_xeonphi.c"
Intel(R) C++ Intel(R) 64 Compiler for applications running on Intel(R) 64, Version 17.0.8.275 Build 20180907
Copyright (C) 1985-2018 Intel Corporation.  All rights reserved.

C:\Program Files (x86)\IntelSWTools2017\compilers_and_libraries_2017\windows\bin>
```



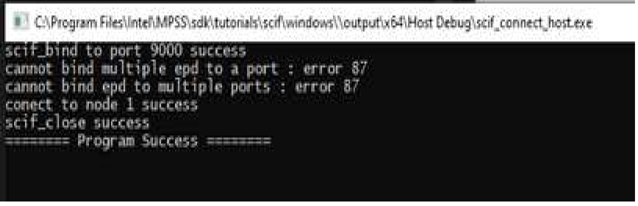
```
login as: root
Authenticating with public key "rsa-key-20210403"
[root@mic0 ~]# ls
[root@mic0 ~]# ls /home
micuser  root
[root@mic0 ~]# ls /home/root/
out
[root@mic0 ~]# cd /home/root/
[root@mic0 root]# chmod +x out
[root@mic0 root]# ./out
hello from xeon phi
[root@mic0 root]#
```



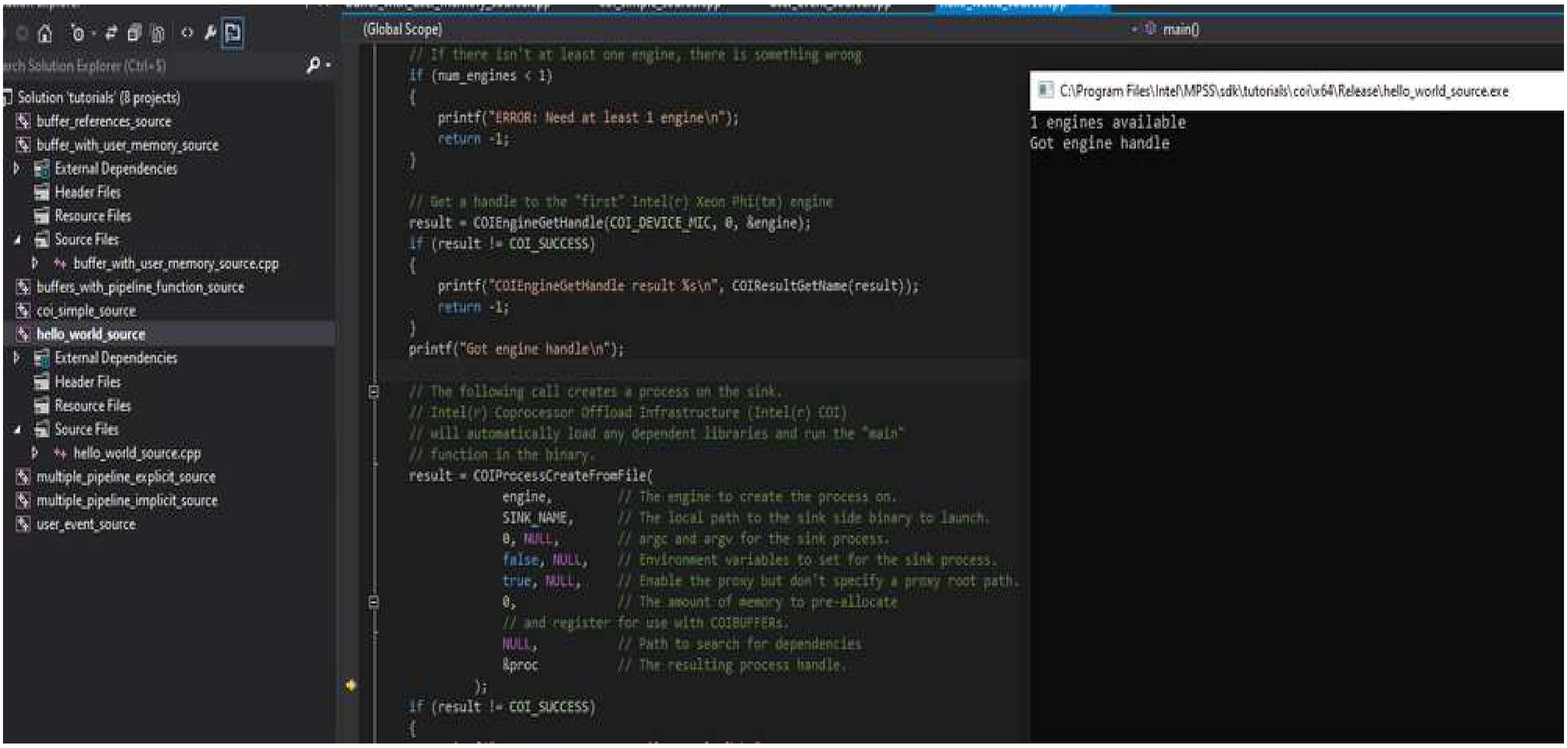
```
    for (i = 0; i < (msg_size/sizeof(int)); i++) {
        if (((int*)send_buf)[i] != ((int*)recv_buf)[i]) {
            printf("data mismatch send_buf[%d] 0x%x recv_buf[%d] 0x%x\n",
                i, ((int*)send_buf)[i], i, ((int*)recv_buf)[i]);
        }
    }
    errno = 0;

_end:
    if (send_buf != NULL)
        free(send_buf);
    if (recv_buf != NULL)
        free(recv_buf);

    /* scif_close : closes the end pt, when successful returns 0 */
    if (scif_close(epd) != 0 || scif_close(epd1) != 0) {
        printf("scif_close failed with error %d\n", get_curr_status());
        exit(1);
    }
    printf("scif_close success\n");
    if (errno == 0)
        printf("======== Program Success ========\n");
    else
        printf("======== Program Failed ========\n");

    return errno;
}
```



```
usage: ./scif_accept -l <local_port> -s <msg_size> -b <block/non-block 1/0
[root@mic0 root]# ls
libscif.so  out
[root@mic0 root]# ./out -l 2049 -s 10 -b 1
scif_bind to port 2049 success
^C
[root@mic0 root]# ^C
[root@mic0 root]# ^C
[root@mic0 root]# ./out -l 5001 -s 10 -b 1
scif_bind to port 5001 success

^C
[root@mic0 root]# ./out -l 9^C1 -s 10 -b 1
[root@mic0 root]# ./out -l 5001 -s 1 -b 1
scif_bind to port 5001 success
^C
[root@mic0 root]# ./out -l 9001 -s 1 -b 1
scif_bind to port 9001 success
accepted connection request from node:0 port:9000
scif_close success
======== Program Success ========
[root@mic0 root]#
```


C:\Program Files\Intel\MPSS\sdk\tutorials\scif\windows\\output\x64\Host Debug\scif_connect_host.exe

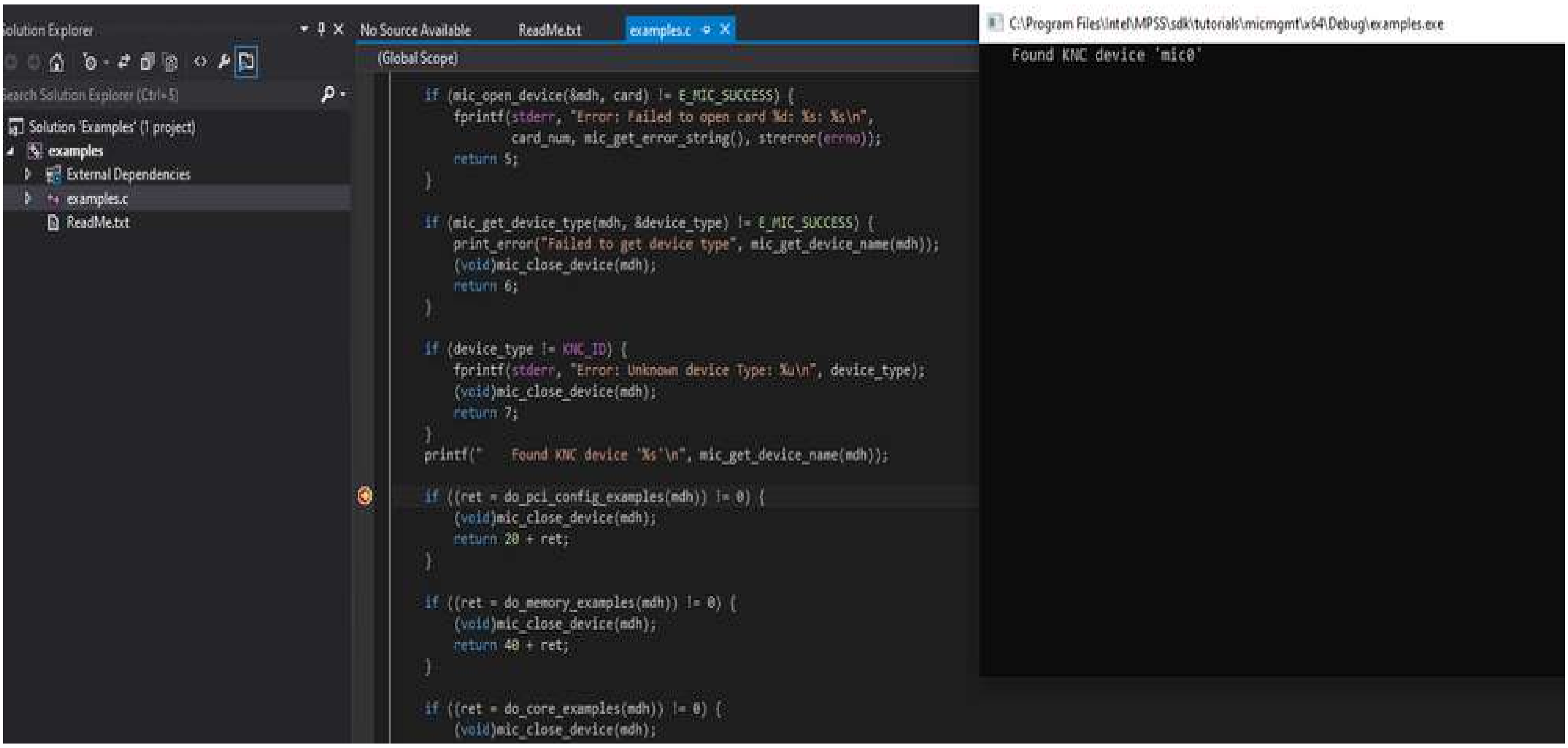
```
scif_bind to port 9000 success
cannot bind multiple epd to a port : error 87
cannot bind epd to multiple ports : error 87
conect to node 1 success
scif_close success
======== Program Success ========
```



```
    // If there isn't at least one engine, there is something wrong
    if (num_engines < 1)
    {
        printf("ERROR: Need at least 1 engine\n");
        return -1;
    }

    // Get a handle to the "first" Intel(r) Xeon Phi(tm) engine
    result = COIEngineGetHandle(COI_DEVICE_MIC, 0, &engine);
    if (result != COI_SUCCESS)
    {
        printf("COIEngineGetHandle result %s\n", COIResultGetName(result));
        return -1;
    }
    printf("Got engine handle\n");

    // The following call creates a process on the sink.
    // Intel(r) Coprocessor Offload Infrastructure (Intel(r) COI)
    // will automatically load any dependent libraries and run the "main"
    // function in the binary.
    result = COIProcessCreateFromFile(
                engine,         // The engine to create the process on.
                SINK_NAME,      // The local path to the sink side binary to launch.
                0, NULL,        // argc and argv for the sink process.
                false, NULL,    // Environment variables to set for the sink process.
                true, NULL,     // Enable the proxy but don't specify a proxy root path.
                0,              // The amount of memory to pre-allocate
                // and register for use with COIBUFFERs.
                NULL,           // Path to search for dependencies
                &proc           // The resulting process handle.
            );
    if (result != COI_SUCCESS)
    {
```

C:\Program Files\Intel\MPSS\sdk\tutorials\coi\x64\Release\hello_world_source.exe

```
1 engines available
Got engine handle
```



Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Examples' (1 project)
examples
External Dependencies
examples.c
ReadMe.txt
No Source Available
ReadMe.txt
examples.c
(Global Scope)
C:\Program Files\Intel\MPSS\sdk\tutorials\micmgmt\x64\Debug\examples.exe

```
if (mic_open_device(&mdh, card) != E_MIC_SUCCESS) {
    fprintf(stderr, "Error: Failed to open card %d: %s: %s\n",
            card_num, mic_get_error_string(), strerror(errno));
    return 5;
}

if (mic_get_device_type(mdh, &device_type) != E_MIC_SUCCESS) {
    print_error("Failed to get device type", mic_get_device_name(mdh));
    (void)mic_close_device(mdh);
    return 6;
}

if (device_type != KNC_ID) {
    fprintf(stderr, "Error: Unknown device Type: %u\n", device_type);
    (void)mic_close_device(mdh);
    return 7;
}
printf("    Found KNC device '%s'\n", mic_get_device_name(mdh));

if ((ret = do_pci_config_examples(mdh)) != 0) {
    (void)mic_close_device(mdh);
    return 20 + ret;
}

if ((ret = do_memory_examples(mdh)) != 0) {
    (void)mic_close_device(mdh);
    return 40 + ret;
}

if ((ret = do_core_examples(mdh)) != 0) {
    (void)mic_close_device(mdh);
```

```
Found KNC device 'mic0'
```

I will do a test with the xeon phi 7220A and mpss 4.4.1 afterwards.

I had also tested more than a year ago the same thing on a macbook air + thunderbolt port under ubuntu and I had to adapt the programs mpss,libscif etc... and the card was recognized and I was able to see the information of the card. Can you see my work from git (for information I am not a 100% c/c++ dev)