
HOTLINE™ CRT QUICK START GUIDE

(LAYER2 ETHERNET VERSION)

1 INTRODUCTION

Hotline Command-Response Transaction (HCrt) is datagram-based protocol for sending command-requests and receiving responses over a transport layer. In the case of this document, we assume a Layer 2 Ethernet Transport. The reader of this document does *not* need to know the specifics of the HCrt protocol, although it may be helpful if the user wishes to create their own initiator or completer logic or wishes to observe and decode packets with Wireshark.

This document will walk the reader through the steps needed to have a software-based HCrt Initiator communicate with a hardware-based HCrt Completer.

2 PRE-REQUISITES

2.1 SOFTWARE INITIATOR

- Running Windows 7
- You must have WinPcap Installed, 4.1 or newer.
 - You can get WinPcap here: <http://www.winpcap.org/>
 - Optionally, installing WireShark also installs WinPcap <http://www.wireshark.org/>
- Install Python 2.7.5 from <http://www.python.org/download/releases/2.7.5/>
 - Be sure to use a 32-bit x86 version and NOT a X86-64 version
- Place the Atomic Rules provided package "hcrt.pyd" in the C:\Python27 directory
 - A copy is here: <https://dl.dropbox.com/u/5548901/hotline/hcrt.pyd>
- Place the Microsoft provided package "msvcr110.dll" in the C:\Python27 directory
 - A copy is here: <https://dl.dropbox.com/u/5548901/hotline/msvcr110.dll>

2.2 HARDWARE COMPLETER

- A KC705 board loaded with a bitstream which implements the AR Hotline Ethernet to AXI Bridge

2.3 CONNECTING INITIATOR TO COMPLETER

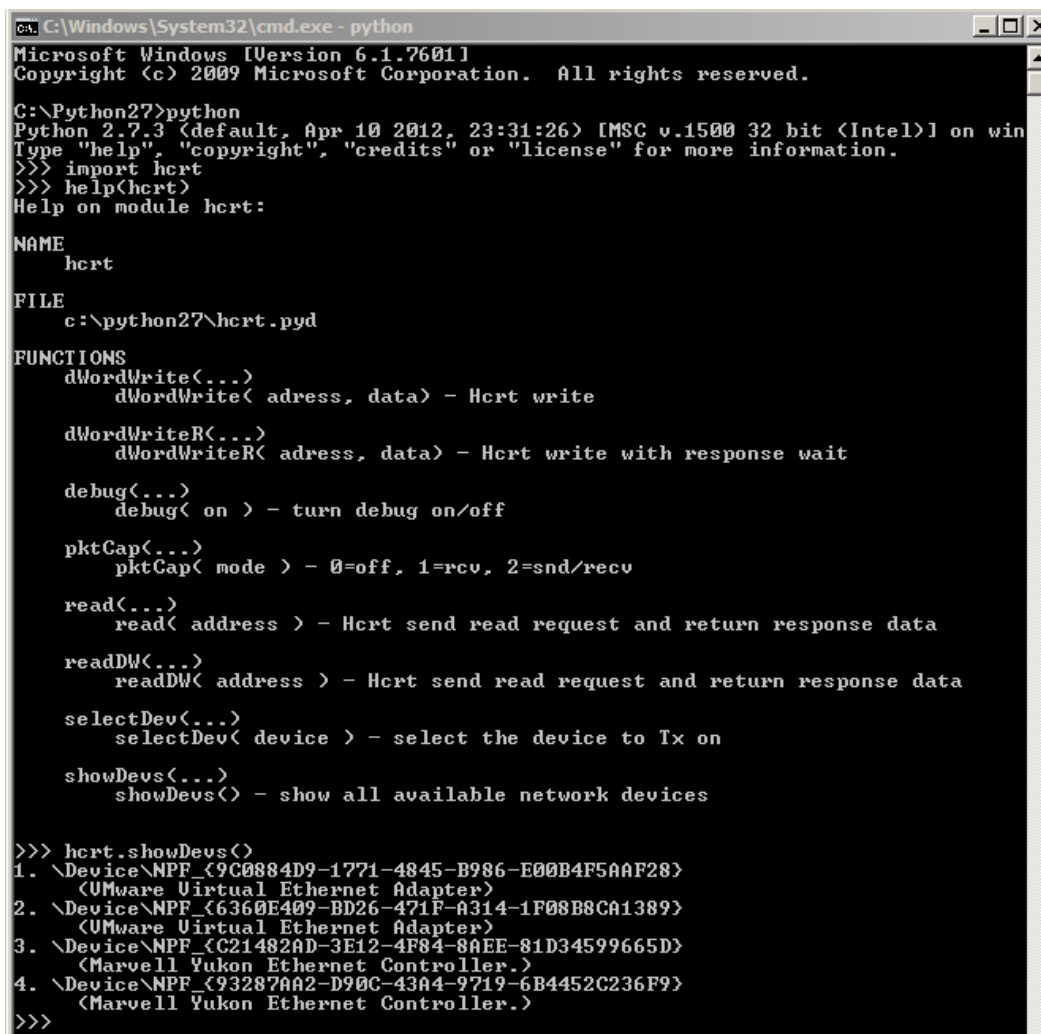
- If the PC has an unused secondary 1000Base-T port, connect it directly to the KC705
 - If the PC only has one 1000Base-T port, use a L2 switch to connect the PC to the KC705
- The choice of which Ethernet port to use is a personal preference: Using a dedicated port is nice for Wireshark as there is no other (non-HCrt) traffic. However, HCrt works fine on a generic LAN with other traffic as it has a unique EtherType

3 GETTING STARTED

3.1 DETERMINING THE CORRECT ETHERNET DEVICE ON THE HOST PC

Your host PC may have multiple Ethernet devices that can send and receive L2 packets. Some may be tied to real physical ports; others may be virtual ports associated with virtual machines. The first step is to determine the enumeration number of the Ethernet device we wish to talk through. Follow these steps to determine that number:

1. Launch a command shell in the directory C:\Python27 where you have installed Python and previously dropped a copy of the "hcrt.pyd" package and the "msvcr110.dll" DLL.
2. Type "python" to launch the interactive Python shell.
3. Type "import hcrt" to load the hcrt package
4. Type "help(hcrt)" to see the list of hcrt methods
5. Type "hcrt.showDevs()" to see the list of available network devices
6. Determine from the results what number corresponds to your network device.



```

C:\Windows\System32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Python27>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
Type "help", "copyright", "credits" or "license" for more information.
>>> import hcrt
>>> help(hcrt)
Help on module hcrt:

NAME
    hcrt

FILE
    c:\python27\hcrt.pyd

FUNCTIONS
    dWordWrite(...)
        dWordWrite( address, data) - Hcrt write

    dWordWriteR(...)
        dWordWriteR( address, data) - Hcrt write with response wait

    debug(...)
        debug( on ) - turn debug on/off

    pktCap(...)
        pktCap( mode ) - 0=off, 1=rcv, 2=snd/rcv

    read(...)
        read( address ) - Hcrt send read request and return response data

    readDW(...)
        readDW( address ) - Hcrt send read request and return response data

    selectDev(...)
        selectDev( device ) - select the device to Tx on

    showDevs(...)
        showDevs() - show all available network devices

>>> hcrt.showDevs()
1. \Device\NPF_{9C0884D9-1771-4845-B986-E00B4F5A0F28}
   <VMware Virtual Ethernet Adapter>
2. \Device\NPF_{6360E409-BD26-471F-A314-1F08B8CA1389}
   <VMware Virtual Ethernet Adapter>
3. \Device\NPF_{C21482AD-3E12-4F84-8AEE-81D34599665D}
   <Marvell Yukon Ethernet Controller.>
4. \Device\NPF_{93287AA2-D90C-43A4-9719-6B4452C236F9}
   <Marvell Yukon Ethernet Controller.>
>>>
  
```

The text below shows the sequence above on an example machine. Devs 1 and 2 are for the VMs. Dev 3 is the primary device (enet0) connected to the LAN. Dev 3 is a secondary device (enet1) also available.

50

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
```

55

```
C:\Python27>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import hcrt
```

60

```
>>> help(hcrt)
Help on module hcrt:
```

```
NAME
    hcrt
```

65

```
FILE
    c:\python27\hcrt.pyd
```

```
FUNCTIONS
```

70

```
    dWordWrite(...)
        dWordWrite( adress, data) - Hcrt write
```

```
    dWordWriteR(...)
        dWordWriteR( adress, data) - Hcrt write with response wait
```

75

```
    debug(...)
        debug( on ) - turn debug on/off
```

```
    pktCap(...)
        pktCap( mode ) - 0=off, 1=rcv, 2=snd/recv
```

80

```
    read(...)
        read( address ) - Hcrt send read request and return response data
```

85

```
    readDW(...)
        readDW( address ) - Hcrt send read request and return response data
```

```
    selectDev(...)
        selectDev( device ) - select the device to Tx on
```

90

```
    showDevs(...)
        showDevs() - show all available network devices
```

95

```
>>> hcrt.showDevs()
1. \Device\NPF_{9C0884D9-1771-4845-B986-E00B4F5AAF28}
   (VMware Virtual Ethernet Adapter)
2. \Device\NPF_{6360E409-BD26-471F-A314-1F08B8CA1389}
   (VMware Virtual Ethernet Adapter)
3. \Device\NPF_{C21482AD-3E12-4F84-8AEE-81D34599665D}
   (Marvell Yukon Ethernet Controller.)
4. \Device\NPF_{93287AA2-D90C-43A4-9719-6B4452C236F9}
   (Marvell Yukon Ethernet Controller.)
>>>
```

100

105

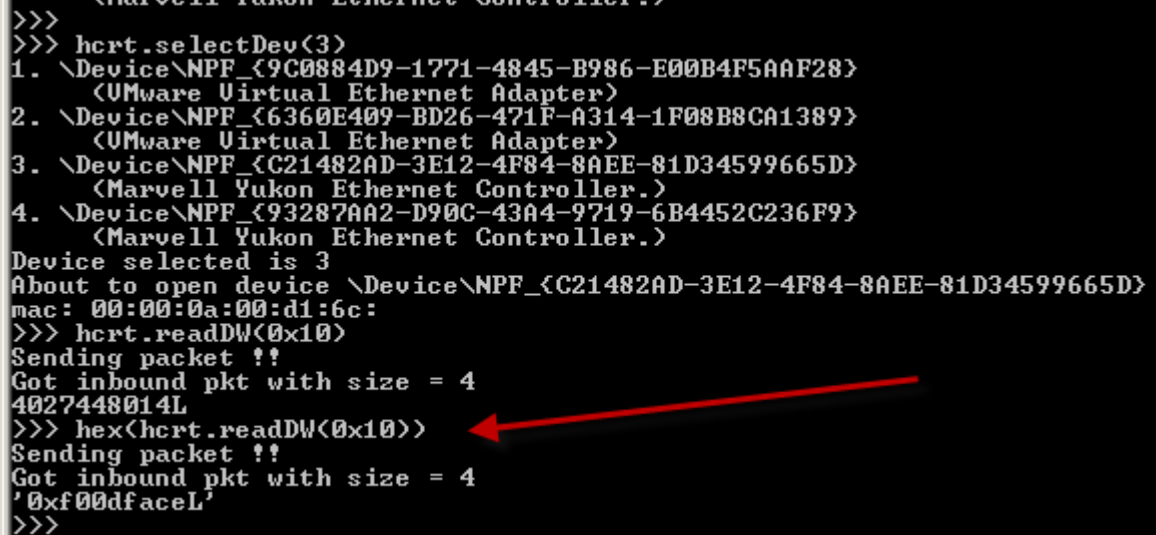
3.2 TESTING CONNECTIVITY BETWEEN THE HOST AND FPGA

Once you know the correct device, you can set it as the port to use, and then try a simple transaction with the FPGA.

1. Set the device number with the correct number for your system (here it is '3') by typing "hcrt.selectDev(3)". The program will echo back the devices, device selected, and the source MAC address.
2. Try reading the FPGA AXI interface at Offset 0x10, which often contains the RO Cookie "0xF00D_FACE". Do this by typing "hcrt.readDW(0x10)". To print it in hex say "hex(hcrt.readDW(0x10))" If you see the correct value returned, you have just read a value from the AXI fabric in the FPGA.

3.

```
>>> hcrt.selectDev(3)
1. \Device\NPF_{9C0884D9-1771-4845-B986-E00B4F5AAF28}
   (UMware Virtual Ethernet Adapter)
2. \Device\NPF_{6360E409-BD26-471F-A314-1F08B8CA1389}
   (UMware Virtual Ethernet Adapter)
3. \Device\NPF_{C21482AD-3E12-4F84-8AEE-81D34599665D}
   (Marvell Yukon Ethernet Controller.)
4. \Device\NPF_{93287AA2-D90C-43A4-9719-6B4452C236F9}
   (Marvell Yukon Ethernet Controller.)
Device selected is 3
About to open device \Device\NPF_{C21482AD-3E12-4F84-8AEE-81D34599665D}
mac: 00:00:0a:00:d1:6c:
>>> hcrt.readDW(0x10)
Sending packet !!
Got inbound pkt with size = 4
4027448014L
>>> hex(hcrt.readDW(0x10))
Sending packet !!
Got inbound pkt with size = 4
'0xf00dfacel'
>>>
```



4 DEBUGGING

Check to make sure you have physical connectivity; that the link lights are on; and, if needed, use Wireshark to observe packets departing from and returning to the host.