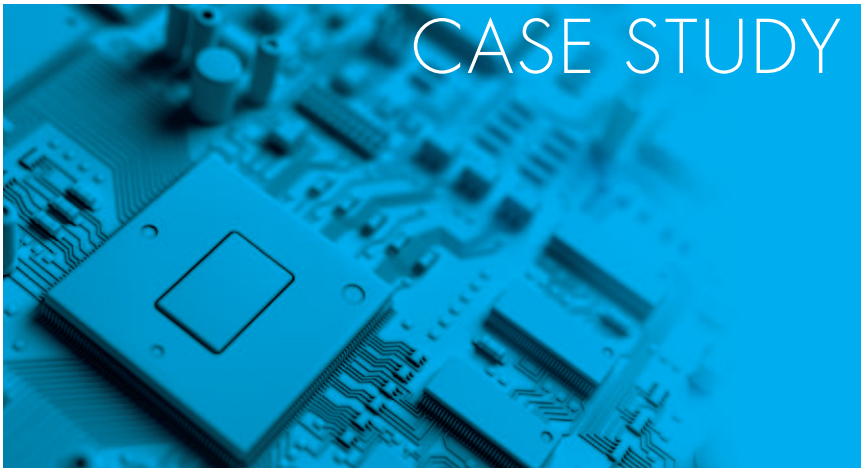
External USB Protocol Decoders for the Total Phase Data Center™

CASE STUDY



Yogender Saroha
MCCI Corporation

MCCI

CHALLENGE

Develop decoders and strategies to debug and analyze protocols that are early in development or are proprietary.

SOLUTION

Total Phase Beagle™ USB 480 protocol analyzer and Data Center™ support scripting and analysis automation to decode and filter bus trace data as it occurs.

BENEFIT

These decoders and a network packet generator can save a lot of effort during network driver debug and verification. The solution is extensible to a wide variety of other protocols.

Developers have long sought a solution that offers the real-time insight into USB bus activities required for effective debugging, coupled with custom class decoding to enable visibility into early development or proprietary protocol classes.

In this case study, MCCI Corporation demonstrates how they used the Total Phase USB Beagle 480 Protocol Analyzer and Data Center application coupled with their own custom application to decode their protocol of interest (RNDIS) to meet their advanced debugging needs.

MCCI went one step further, writing a small program to write the actual network data (ignoring the control-plane traffic) in .pcap format which can be opened in *Wireshark*, a free and open-source packet analyzer.

I have been working in embedded systems for a while now. As a USB systems engineer at MCCI, I've had the opportunity to work on both ends of the wire, doing both device-side protocol implementations and host-side drivers for various OSes.

A big area of work for me has been around USB networking devices, supporting various protocols: ECM, ENCM, NCM, RNDIS, MBIM, and proprietary protocols.

During debugging, we need to see what is actually going over wire. There are various bus analyzers available that can capture bus traces.

As a market-leader company, MCCI usually has to develop solutions before a given protocol becomes hot in market, so the decoders for the protocols I have been working on are typically not available in the analyzer at that moment.

As an engineer, I had this conviction that there should be some way to decode the bus trace data so the data on bus starts making sense. This becomes even more tempting when you have your protocol data-structures ready, either for device implementation or host device driver.

Normally I would copy the raw data to a file and copy the structure into the same file and see what value each member of that structure holds. I know this is quite a cumbersome, time consuming and error prone process. What is the other option…?

There is a better way: I started working on a framework which would help in writing decoders. The first thing I developed was a program which takes raw data as input and calls protocol specific control and data plane function to decode the protocol. This way, I separated the work of parsing the input files from the work of decoding the USB traffic into higher level functions.

Then, based on the various protocols, I (or anyone using the program) can write specific protocol decoding functions.

Then I had to find the USB bus analyzer that would be most suitable for this setup. I was looking for an analyzer which had the most features, including support for scripting. Support for scripting was very important for me as I wanted to automate the process of analysis—I wanted to push a single button or enter a single command, and have my data analyzed. I want to concentrate on getting my answers; and during a long debugging session, I don't want to accidentally analyze the wrong data due to a slip-up.

For this, I found that Total Phase Beagle USB 480 Protocol Analyzer was an excellent fit. In addition to support for scripts and a powerful filter mechanism, its live capture feature is also very useful. It captures long traces into relatively small files, and it manipulates large traces very efficiently. This is very important for tracking down subtle networking bugs. Last but not least it is great value for the money.

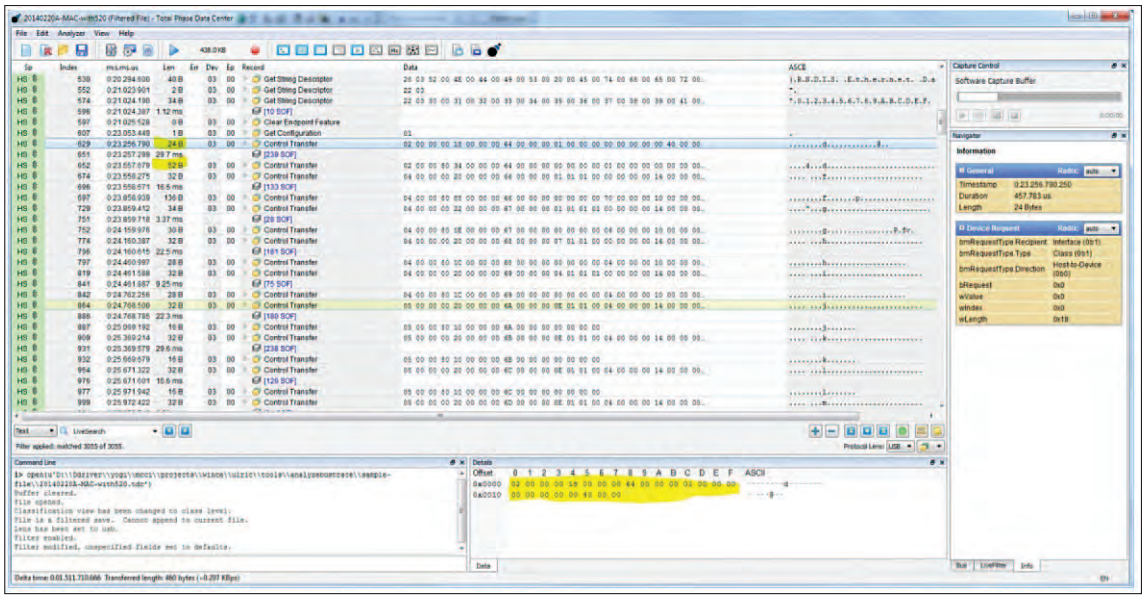
To show how this is done, we've posted the code for the framework and a sample decoder. In this case, I've included the logic for decoding the Microsoft Remote NDIS protocol [download zip file]. This protocol transports Ethernet frames over USB. Although it was originally done for Windows operating systems, RNDIS devices are now commonly used throughout the industry. The throughput of RNDIS is not as high, normally, as the throughput for newer protocols like NCM, but drivers are ubiquitous.

To use the decoder on Windows, download the distribution package and follow the instructions for installing it.

Here's an example of how I use it.

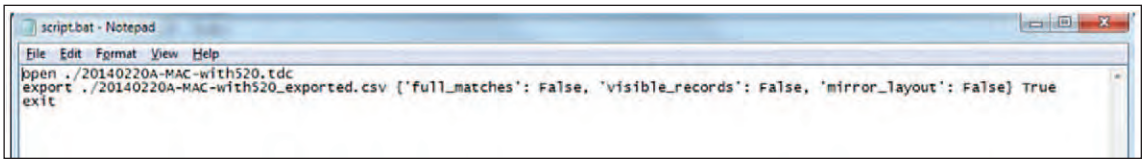
1. I captured data as normal using the Beagle analyzer

Beagle trace highlighting Control Command decoded



2. I then wrote a simple script for TotalPhase Data Center, which exports the data in a csv file

Sample script file for Total Phase Data Center



3. I then wrote a simple shell script which invokes Data Center with a script, processes the csv file, and calls my decoder program

Script file invoking Data Center script and processing exported data

AutomateAnalyzeBustrace.sh - Notepad

File Edit Format View Help

```
#!/bin/ksh
# File for Automating the processing of csv file and decoding output

# 1. Export csv file from tdc.
#       Note usage of 8.3 file name
"$DATA_CENTER_PATH/DATACE~1.EXE" -m -b script.bat

# 2. extract txn from csv file
grep txn 20140220A-MAC-with520_exported.csv > __temp

# 3. Cut column number 6, 9, 10, 11
cut -d, -f6,9-11 __temp > __temp1

# 4. Filter out NAK
grep -v "NAK" __temp1 > __temp

# 5. Filter out 0 Bytes data
grep -v "^0 B," __temp > 20140220A-MAC-with520_final.csv

# 6. Del Intermediate files
[ -f ./__temp ] && rm ./__temp
[ -f ./__temp1 ] && rm ./__temp1
[ -f ./20140220A-MAC-with520_exported.csv ] && rm ./20140220A-MAC-with520_exported.csv

# 7. Analyze the csv file
Analyzebustrace 20140220A-MAC-with520_final.csv > 20140220A-MAC-with520_decoded.txt
```

4. The decoder program generates an output file which I view on screen

Decoded data

20140220A-MAC-with520_decoded.txt - Notepad

File Edit Format View Help

```
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0301, wLength 0x14, Bytes Read(20)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0303, wLength 0x2, Bytes Read(2)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0303, wLength 0x22, Bytes Read(34)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0000,wValue 0x0200, wLength 0x4, Bytes Read(4)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0000,wValue 0x0200, wLength 0xDB, Bytes Read(219)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0000,wValue 0x0201, wLength 0x4, Bytes Read(4)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0000,wValue 0x0201, wLength 0x47, Bytes Read(71)
Standard Request: bmRequestType 0x00, bRequest 0x09, wIndex 0x0000,wValue 0x0001, wLength 0x0, Bytes Read(0)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0304, wLength 0x2, Bytes Read(2)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0304, wLength 0x3A, Bytes Read(58)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0305, wLength 0x2, Bytes Read(2)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0305, wLength 0x28, Bytes Read(40)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0303, wLength 0x2, Bytes Read(2)
Standard Request: bmRequestType 0x80, bRequest 0x06, wIndex 0x0409,wValue 0x0303, wLength 0x22, Bytes Read(34)
Standard Request: bmRequestType 0x02, bRequest 0x01, wIndex 0x0082,wValue 0x0000, wLength 0x0, Bytes Read(0)
Standard Request: bmRequestType 0x80, bRequest 0x08, wIndex 0x0000,wValue 0x0000, wLength 0x1, Bytes Read(1)
RNDIS Packet:
        Message type(0x2) - Initialize
        Message Len(0x18):24
        RequestId(0x64):100
        MajorVersion(0x1):1
        MinorVersion(0x0):0
        MaxTransferSize(0x4000):16384
RNDIS Packet:
        Message type(0x80000002) - Initialize Complete
        Message Len(0x34):52
        RequestId(0x64):100
        Statuc(0x0):0
        MajorVersion(0x1):1
        MinorVersion(0x0):0
        DeviceFlags(0x1):1
        Medium(0x0):0
        MaxPacketsPerMessage(0x1):1
        MaxTransferSize(0x62c):1580
        PacketAlignmentFactor(0x0):0
        AFListOffset(0x0):0
        AFListSize(0x0):0
RNDIS Packet:
        Message type(0x4) - Query
        Message Len(0x20):32
RNDIS Packet:
        Message type(0x80000004) - Query Complete
        Message Len(0x88):136
RNDIS Packet:
        Message type(0x4) - Query
        Message Len(0x22):34
RNDIS Packet:
        Message type(0x80000004) - Query Complete
        Message Len(0x1e):30
RNDIS Packet:
        Message type(0x4) - Query
        Message Len(0x20):32
RNDIS Packet:
        Message type(0x80000004) - Query Complete
        Message Len(0x1c):28
RNDIS Packet:
        Message type(0x4) - Query
        Message Len(0x20):32
RNDIS Packet:
        Message type(0x80000004) - Query Complete
```

There is one more thing that an engineer may want to write. All the protocols in discussion here are network protocols. It's easy to write a small program which writes the actual network data (ignoring the control-plane traffic) in .pcap format which can be opened in *Wireshark*, a free and open-source packet analyzer. With this you can actually debug quite easily by comparing network packet generated by your program from the USB bus trace against the packets captured by WinPcap (or similar) driver at the network layer.

These decoders and a network packet generator can save a lot of effort during network driver debug and verification.

It takes some time to write a decoder, but the time it saves is all worth it…

Download sample files and executables here.

ABOUT TOTAL PHASE

Total Phase is a leading provider of embedded systems solutions for engineers all over the world. Our mission is to provide intelligent visibility into embedded systems by creating affordable, high-quality, and powerful solutions for anyone working in the embedded systems environment.

ABOUT MCCI CORPORATION

MCCI develops and licenses USB system software that is embedded into our customers' consumer-electronics products. MCCI also provides testing services (USB and Microsoft WHQL), and manufactures specialized test equipment for developing and testing products that use USB and MIPI technology.

+1 (408) 850-6501 | sales@totalphase.com | www.totalphase.com

©2014 Total Phase, Inc. Beagle and Data Center are trademarks of Total Phase, Inc.