Enabling Jumbo Packet Support for C6678

Content is no longer maintained and is being kept for reference only!

This wiki will detail the necessary steps required to enable jumbo packet support for the Texas Instruments C6678.

Contents

Software Requirement

For this document the following software versions are required

- BIOS_MCSDK_2_1_2_6
- NDK_2_21_01_38
- pdk_C6678_1_1_2_6
- Bios_6_33_06_50
- xdias_7_21_01_07
- xdctools_3_23_04_60
- Compiler c6000_7.4.4

bios-mcsdk can be downloaded from the following link

http://software-dl.ti.com/sdoemb/sdoemb_public_sw/bios_mcsdk/latest/index_FDS.html

NDK modifications

In order to have the Jumbo support for C6678 (packet sizes 1.5kb-9kb), the **_INCLUDE_JUMBOFRAME_SUPPORT** preprocessor definition has to be added in various files. The modifications below (file paths in bold) are the only changes made to the appropriate files. Please search through your file to find the relevant line. Also, note that some lines of code have been truncated for brevity, and any lead-up comments have been included to show the general vicinity of the change.

- ***'ndk_2_21_01_38\packages\ti\ndk\netctrl\package.bld'***

```
/* IPv6 netctrl_min library */
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=0 " +
" -DNETSRV_ENABLE_HTTP=0 " +
" -DNETSRV_ENABLE_NAT=0 " +
```

```
/* IPv6 netctrl "standard" library */
var libName = "netctrl";
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=1 " +
" -DNETSRV_ENABLE_HTTP=1 " +
```

```
/* IPv6 netctrl_full library */
var libName = "netctrl_full";
2
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=1 " +
" -DNETSRV_ENABLE_HTTP=1 " +
" -DNETSRV_ENABLE_NAT=1 " +
```

```
/* IPv4 netctrl_min library */
var libName = "netctrl_min_ipv4";
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=0" +
```

```
/* IPv4 netctrl "standard" library */
var libName = "netctrl_ipv4";
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=1" +
```

```
/* IPv4 netctrl_full library */
var libName = "netctrl_full_ipv4";
var libOptions = {
copts: " -D_NDK_EXTERN_CONFIG " +
" -D_INCLUDE_JUMBOFRAME_SUPPORT " +
" -DNETSRV_ENABLE_TELNET=1" +
```

- ***ndk_2_21_01_38\packages\ti\ndk\stack\package.bld***

```
/*
* Jumbo frame support
*
* To add support for jumbo frames, add the following file to this array
* and throw the following define to the compiler in 'copts' below:
*
* -D_INCLUDE_JUMBOFRAME_SUPPORT
*/
"pbm/jumbo_pbm.c",
```

```
/* stk.lib */
var libOptions = {
copts: "-D_NDK_EXTERN_CONFIG -D_INCLUDE_NIMU_CODE " + "-D_INCLUDE_JUMBOFRAME_SUPPORT",
incs: ndkPathInclude,
};
```

```
/* stk6_ppp_pppoe.lib */
var libName = "stk6_ppp_pppoe";
var libOptions = {
copts: "-D_INCLUDE_PPP_CODE -D_INCLUDE_PPPOE_CODE " +
"-D_NDK_EXTERN_CONFIG -D_INCLUDE_NIMU_CODE " +
"-D_INCLUDE_IPv6_CODE " + "-D_INCLUDE_JUMBOFRAME_SUPPORT",
incs: ndkPathInclude,
};
```

Building the NDK packages

Once the above changes have been made to the NDK package, it has to be rebuilt. This rebuilt NDK will be included in the NIMU driver, and also for the compilation of the test example that will be discussed in later section (5).

Modify the ***ndk.mak*** file as needed. The modifications for this example are below.

ndk_2_21_01_38\ndk.mak

DESTDIR ?=
XDC_INSTALL_DIR ?= C:/ti/bios_mcsdk_2126_jumbomodtest/xdctools_3_23_04_60
SYSBIOS_INSTALL_DIR ?= C:/ti/bios_mcsdk_2126_jumbomodtest/bios_6_33_06_50
ti.targets.elf.C66 ?= C:/ti/ccsv5/tools/compiler/c6000_7.4.4

For the purpose of this document, the NDK was rebuilt using gmake on a Windows 7 machine. The process is documented here.

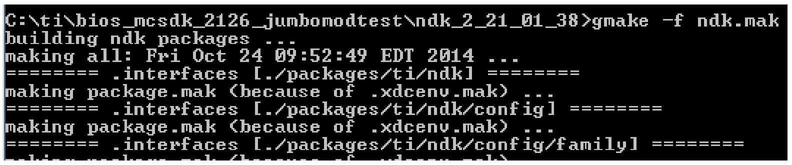
http://processors.wiki.ti.com/index.php/Rebuilding_The_NDK_Core_Using_Gmake

If the build errors related to the package.bld files are encountered, ensure that the spaces are left trailing the flag e.g + "-D_INCLUDE_JUMBOFRAME_SUPPORT " Errors can be encountered when incorrect formatting is used.

- Add the ***<xdc_install_dir>*** to your PATH environment variable so that the gmake executable can be found.

> ***gmake –f ndk.mak***

Once the gmake command is entered, it could take a while for the NDK to build.



```
C:\ti\bios_mcsdk_2126_jumbomodtest\ndk_2_21_01_38>gmake -f ndk.mak
building ndk packages ...
making all: Fri Oct 24 09:52:49 EDT 2014 ...
======== .interfaces [./packages/ti/ndk] ========
making package.mak (because of .xdcenv.mak) ...
======== .interfaces [./packages/ti/ndk/config] ========
making package.mak (because of .xdcenv.mak) ...
======== .interfaces [./packages/ti/ndk/config/family] ========
```

NIMU modifications

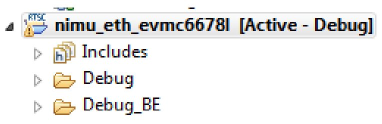
pdk_C6678_1_1_2_6\packages\ti\transport\ndk\nimu\src\nimu_eth.c

```
/* Allocate the PBM packet for the Max MTU size*/
if (NULL == (hPkt = PBM_alloc(10236))) {
/* could not get a free NDK packet, maybe the next time around we can... */
gRxDropCounter++;
```

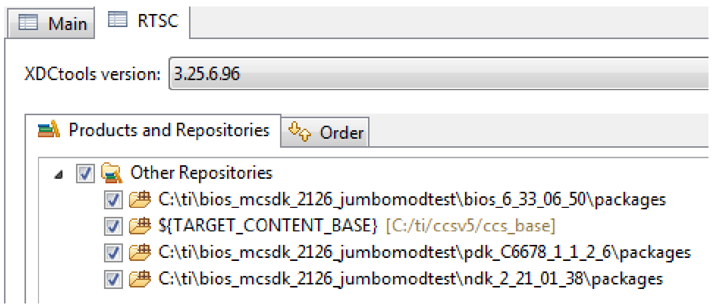
|

Rebuilding the NIMU

In order to the Jumbo support to be incorporated into the NIMU, it has to be rebuilt. Open CCS and navigate to the NIMU project located at ***pdk_C6678_1_1_2_6\packages\ti\transport\ndk\nimu***. Import this project into the CCS workspace.

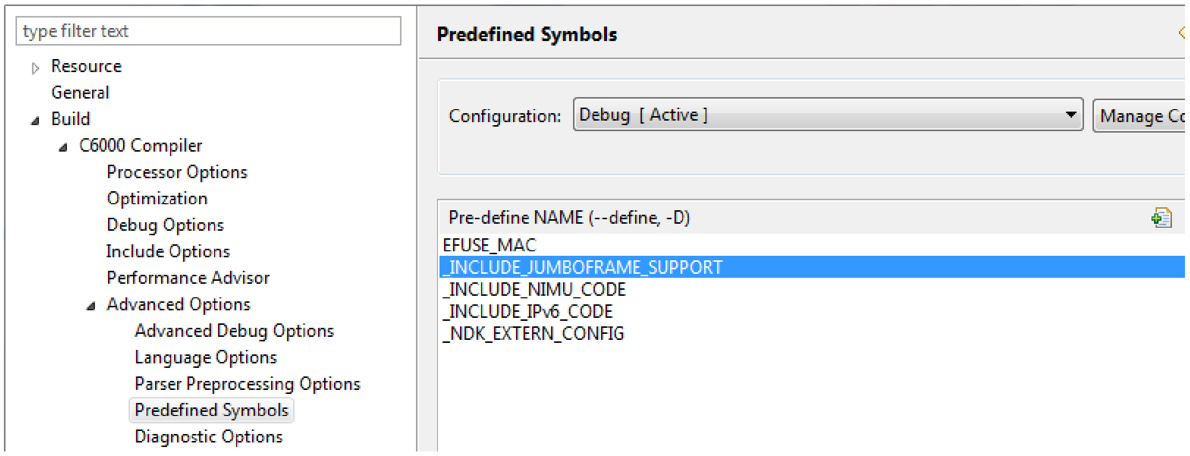

nimu_eth_evmc6678l [Active - Debug]
Includes
Debug
Debug_BE

Right click on project>properties.
In RTSC, the path to the NDK does not necessarily needs to point to the modified NDK. The main change was **PBM_alloc(10236)** in ***nimu_eth.c***.



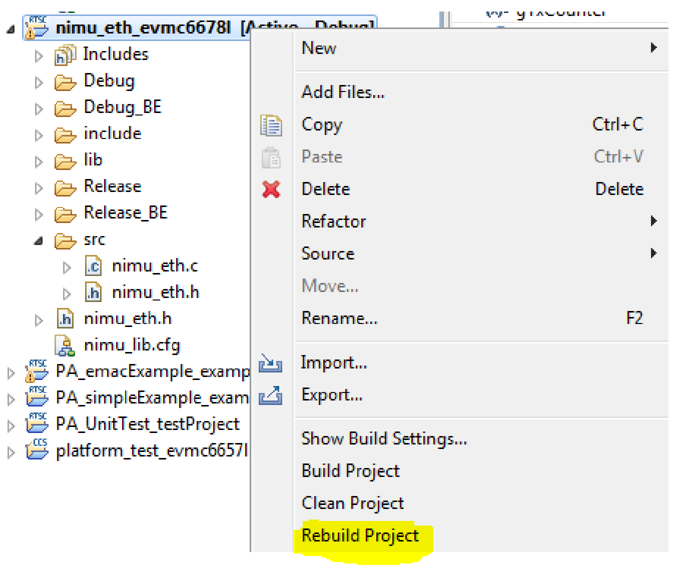
XDCtools version: 3.25.6.96
Other Repositories
C:\ti\bios_mcsdk_2126_jumbomodtest\bios_6_33_06_50\packages
${TARGET_CONTENT_BASE} [C:/ti/ccsv5/ccs_base]
C:\ti\bios_mcsdk_2126_jumbomodtest\pdk_C6678_1_1_2_6\packages
C:\ti\bios_mcsdk_2126_jumbomodtest\ndk_2_21_01_38\packages

The predefined symbol ***_INCLUDE_JUMBOFRAME_SUPPORT*** has also to be included in the build. Add this, by clicking the + symbol.



Predefined Symbols
Configuration: Debug [Active]
Pre-define NAME (--define, -D)
EFUSE_MAC
_INCLUDE_JUMBOFRAME_SUPPORT
_INCLUDE_NIMU_CODE
_INCLUDE_IPv6_CODE
_NDK_EXTERN_CONFIG

Rebuild the NIMU project.



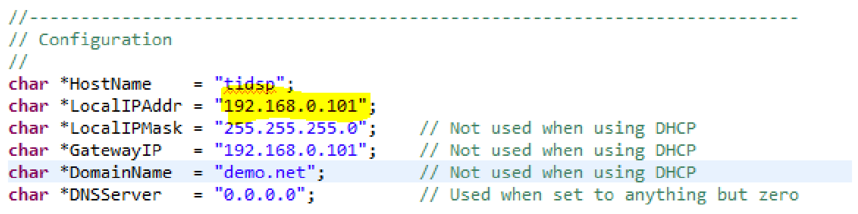
Building an example

In order to test the jumbo packet functionality, import the helloWorld example from the path: ***mcsdk_2_01_02_06\examples\ndk\helloWorld\evmc6678l***. Modify ***helloWorld.c*** to use a static IP of your choice. For this example, 192.168.0.101 is used. You can also use DHCP which can be enabled on the EVM. Ensure that you have the right mode selected on the EVM:
SW9

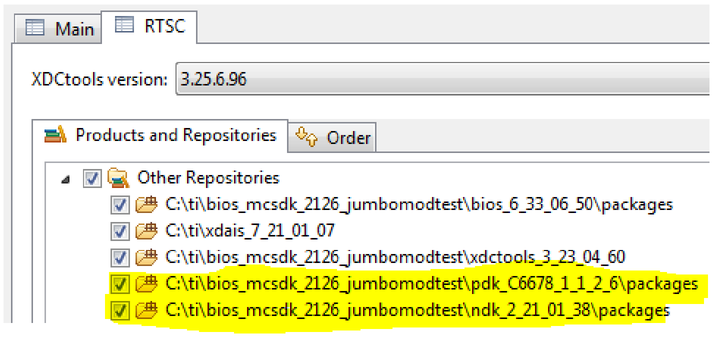
User Switch 2 ON : DHCP
User Switch 2 OFF: Static IP

This is also mentioned in the C6678 HW wiki section 5

http://processors.wiki.ti.com/index.php/TMDXEVM6678L_EVM_Hardware_Setup


```
//---------------------------------------------------------------------------
// Configuration
//
char *HostName    = "tidsp";
char *LocalIPAddr = "192.168.0.101";
char *LocalIPMask = "255.255.255.0";    // Not used when using DHCP
char *GatewayIP   = "192.168.0.101";    // Not used when using DHCP
char *DomainName  = "demo.net";         // Not used when using DHCP
char *DNSServer   = "0.0.0.0";          // Used when set to anything but zero
```

This project needs to be built with the new NDK and NIMU that was built in the previous steps.



XDCtools version: 3.25.6.96

Other Repositories
- C:\ti\bios_mcsdk_2126_jumbomodtest\bios_6_33_06_50\packages
- C:\ti\xdais_7_21_01_07
- C:\ti\bios_mcsdk_2126_jumbomodtest\xdctools_3_23_04_60
- C:\ti\bios_mcsdk_2126_jumbomodtest\pdk_C6678_1_1_2_6\packages
- C:\ti\bios_mcsdk_2126_jumbomodtest\ndk_2_21_01_38\packages

Right-click and rebuild the project.

Demo

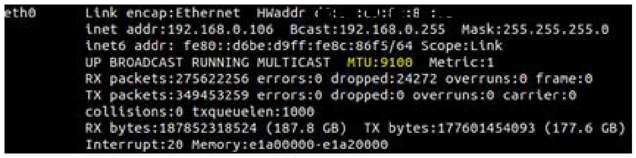
This Demo was run with the following setup:

C6678LE EVM (192.168.0.101) <-------> Ubuntu PC (192.168.0.106)

Wireshark running on the Ubuntu machine was used to observe the network traffic.

Before proceeding, ensure that the Ethernet interface on the Ubuntu machine is configured to handle a MTU size of ***9100***

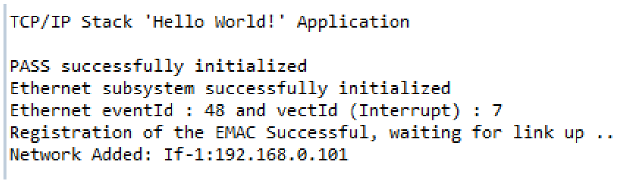
$sudo ifconfig eth0 mtu 9100



```
eth0      Link encap:Ethernet  HWaddr [redacted]
          inet addr:192.168.0.106  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::d6be:d9ff:fe8c:86f5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9100  Metric:1
          RX packets:275622256 errors:0 dropped:24272 overruns:0 frame:0
          TX packets:349453259 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:187852318524 (187.8 GB)  TX bytes:177601454093 (177.6 GB)
          Interrupt:20 Memory:e1a00000-e1a20000
```

- In CCS, launch the C6678 target in no-boot mode.
- Connect to Core0 and load the C6678 helloworld_evmc6678l.out project that was built.
- Run the program.

Observe the console for the example to come up. The IP address of the EVM will show up on the console.

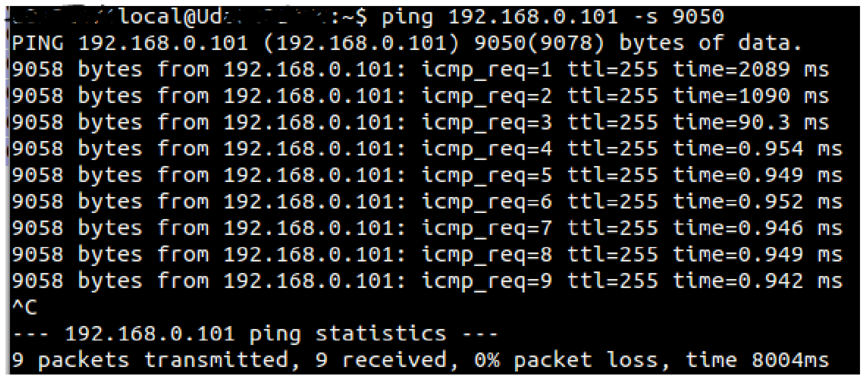


```
TCP/IP Stack 'Hello World!' Application

PASS successfully initialized
Ethernet subsystem successfully initialized
Ethernet eventId : 48 and vectId (Interrupt) : 7
Registration of the EMAC Successful, waiting for link up ..
Network Added: If-1:192.168.0.101
```

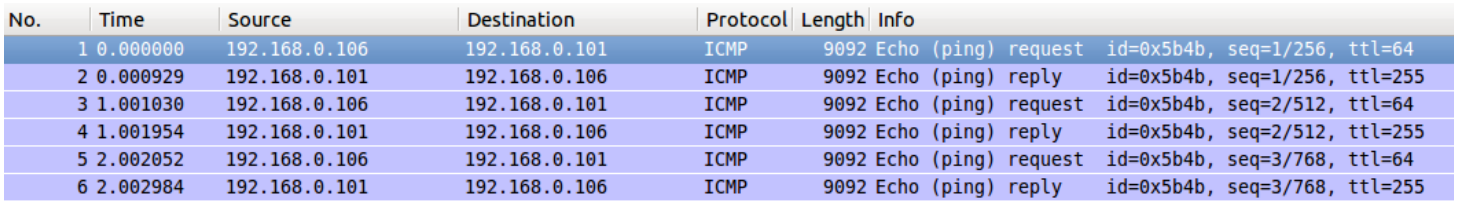
At this point, you can ping from the Linux machine with an instance of Wireshark running. You can vary the packet sizes by using the ***-s*** flag for ping.

$ ping 192.168.0.101 -s 9050

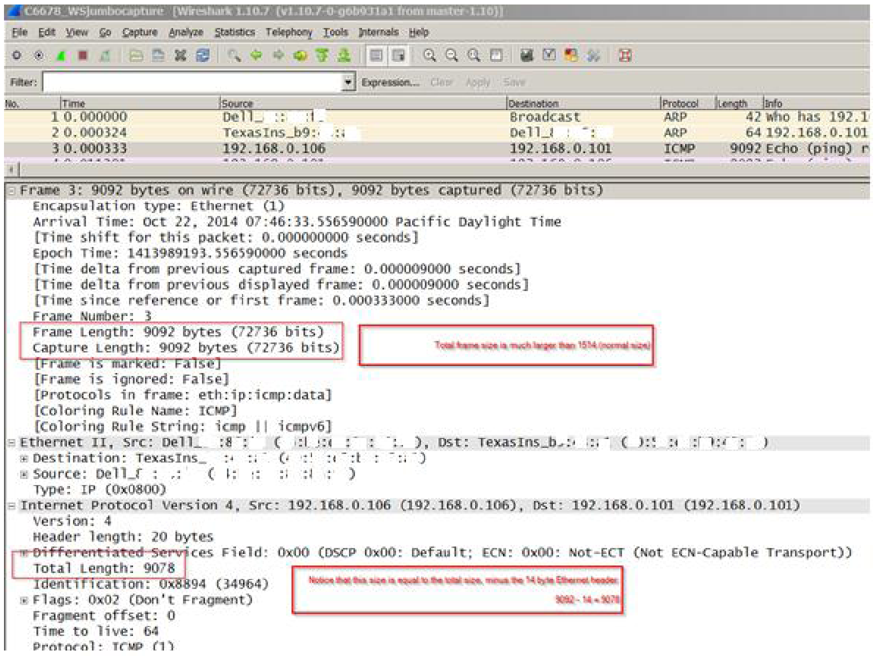


```
[redacted]local@Ud[redacted]:~$ ping 192.168.0.101 -s 9050
PING 192.168.0.101 (192.168.0.101) 9050(9078) bytes of data.
9058 bytes from 192.168.0.101: icmp_req=1 ttl=255 time=2089 ms
9058 bytes from 192.168.0.101: icmp_req=2 ttl=255 time=1090 ms
9058 bytes from 192.168.0.101: icmp_req=3 ttl=255 time=90.3 ms
9058 bytes from 192.168.0.101: icmp_req=4 ttl=255 time=0.954 ms
9058 bytes from 192.168.0.101: icmp_req=5 ttl=255 time=0.949 ms
9058 bytes from 192.168.0.101: icmp_req=6 ttl=255 time=0.952 ms
9058 bytes from 192.168.0.101: icmp_req=7 ttl=255 time=0.946 ms
9058 bytes from 192.168.0.101: icmp_req=8 ttl=255 time=0.949 ms
9058 bytes from 192.168.0.101: icmp_req=9 ttl=255 time=0.942 ms
^C
--- 192.168.0.101 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8004ms
```

On Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.106	192.168.0.101	ICMP	9092	Echo (ping) request id=0x5b4b, seq=1/256, ttl=64
2	0.000929	192.168.0.101	192.168.0.106	ICMP	9092	Echo (ping) reply id=0x5b4b, seq=1/256, ttl=255
3	1.001030	192.168.0.106	192.168.0.101	ICMP	9092	Echo (ping) request id=0x5b4b, seq=2/512, ttl=64
4	1.001954	192.168.0.101	192.168.0.106	ICMP	9092	Echo (ping) reply id=0x5b4b, seq=2/512, ttl=255
5	2.002052	192.168.0.106	192.168.0.101	ICMP	9092	Echo (ping) request id=0x5b4b, seq=3/768, ttl=64
6	2.002984	192.168.0.101	192.168.0.106	ICMP	9092	Echo (ping) reply id=0x5b4b, seq=3/768, ttl=255


C6678_WSJumbocapture [Wireshark 1.10.7 (v1.10.7-0-g6b931a1 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time        Source              Destination       Protocol Length Info
    1 0.000000    Dell_               Broadcast         ARP          42 Who has 192.1
    2 0.000324    TexasIns_b9:        Dell_             ARP          64 192.168.0.101
    3 0.000333    192.168.0.106       192.168.0.101     ICMP       9092 Echo (ping) r

⊟ Frame 3: 9092 bytes on wire (72736 bits), 9092 bytes captured (72736 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 22, 2014 07:46:33.556590000 Pacific Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1413989193.556590000 seconds
    [Time delta from previous captured frame: 0.000009000 seconds]
    [Time delta from previous displayed frame: 0.000009000 seconds]
    [Time since reference or first frame: 0.000333000 seconds]
    Frame Number: 3
    Frame Length: 9092 bytes (72736 bits)
    Capture Length: 9092 bytes (72736 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp || icmpv6]
⊟ Ethernet II, Src: Dell_ :8 ( ), Dst: TexasIns_b ( )
  ⊞ Destination: TexasIns_ ( )
  ⊞ Source: Dell_ ( )
    Type: IP (0x0800)
⊟ Internet Protocol Version 4, Src: 192.168.0.106 (192.168.0.106), Dst: 192.168.0.101 (192.168.0.101)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 9078
    Identification: 0x8894 (34964)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
```

Total frame size is much larger than 1514 (normal size)
Notice that this size is equal to the total size, minus the 14 byte Ethernet header.
9092 - 14 = 9078

{{ 1. switchcategory:MultiCore= - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **Enabling Jumbo Packet Support for C6678** here.	Keystone= - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum Please post only comments related to the article **Enabling Jumbo Packet Support for C6678** here.	C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **Enabling Jumbo Packet Support for C6678** here.*	*F p c F F c a J S F* }

Links

Amplifiers & Linear	DLP & MEMS	Processors	Switches & Multiplexers
Audio	High-Reliability	- ARM Processors	Temperature Sensors & Control ICs
Broadband RF/IF & Digital Radio	Interface	- Digital Signal Processors (DSP)	Wireless Connectivity
Clocks & Timers	Logic	- Microcontrollers (MCU)	
Data Converters	Power Management	- OMAP Applications Processors	

Retrieved from "https://processors.wiki.ti.com/index.php?title=Enabling_Jumbo_Packet_Support_for_C6678&oldid=235285"

This page was last edited on 13 August 2018, at 21:33.