



Bridging USB to Ethernet with Digi Embedded Linux 5.9.6.2 on Digi ConnectCard Wi-i.MX28

Document History

Date	Version	Change Description
2013/11/27	V1.0	Initial entry/outline, tested with DEL-5.9.6.2 on Wi-iMX28 V6

Table of Contents

Document History	1
Table of Contents	1
1 Problem Description	2
2 Requirements	2
3 Software Setup	2
3.1 Create a new Digi EL Kernel/Rootfs Project	3
3.2 Configure the kernel for USB device Ethernet gadget.....	4
3.3 Built the kernel/rootfs project	8
4 Hardware Setup.....	9
5 Deploying firmware to module	9
6 Testing from Linux	10
7 Testing from Windows	10
8 Test output on Wi-i.MX28.....	10
9 References	11

1 Problem Description

Some customers just want to use the Digi ConnectCard i.MX28 as an USB to Ethernet device or using it similar like an USB dongle, enabling a host to access Ethernet through USB. With Digi Embedded Linux you are able to quickly configure and build firmware with appropriate features to implement this kind of bridge. We also provide ready to run Linux kernel and root file system firmware images for demo purposes. You may skip sections project configuration and adding startup files. If you just want to run the firmware jump to section hardware setup.

2 Requirements

To try the example in this document you need:

- Digi ConnectCard (Wi-)i.MX28 module mounted on Digi development board.
- Digi Embedded Linux (DEL) 5.9.6.2 or above development environment.
- You can get everything together in a Digi Embedded Linux JumpStart Kit i.MX28
- You also need a cable USB A type to Micro-USB B connector
- Ethernet cable
- Serial cable to access the Linux console

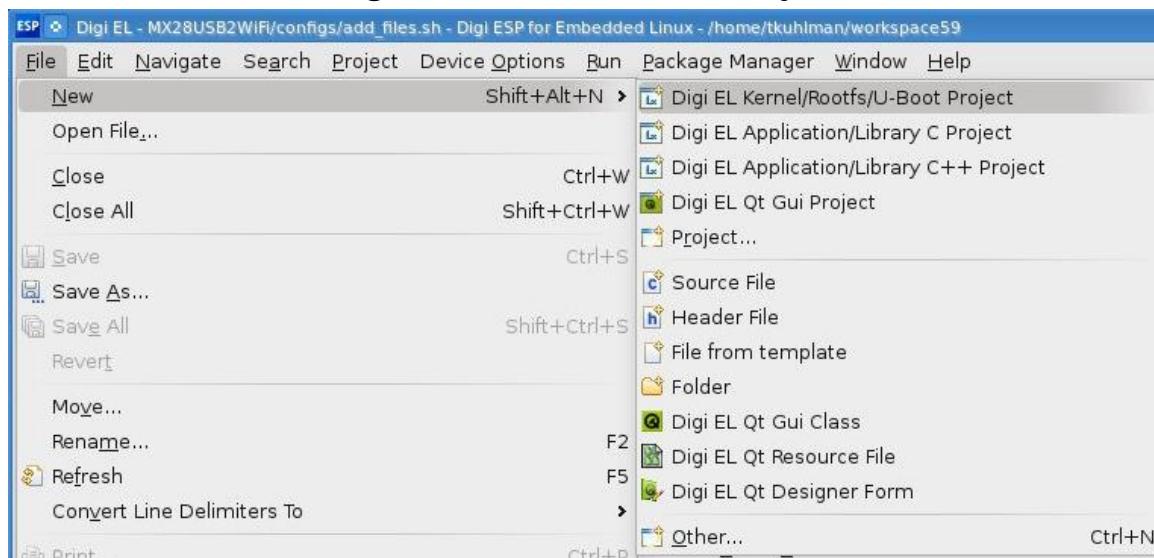
3 Software Setup

For your convenience find all files and images compiled into the archive these instructions came with, you might want to skip this section.

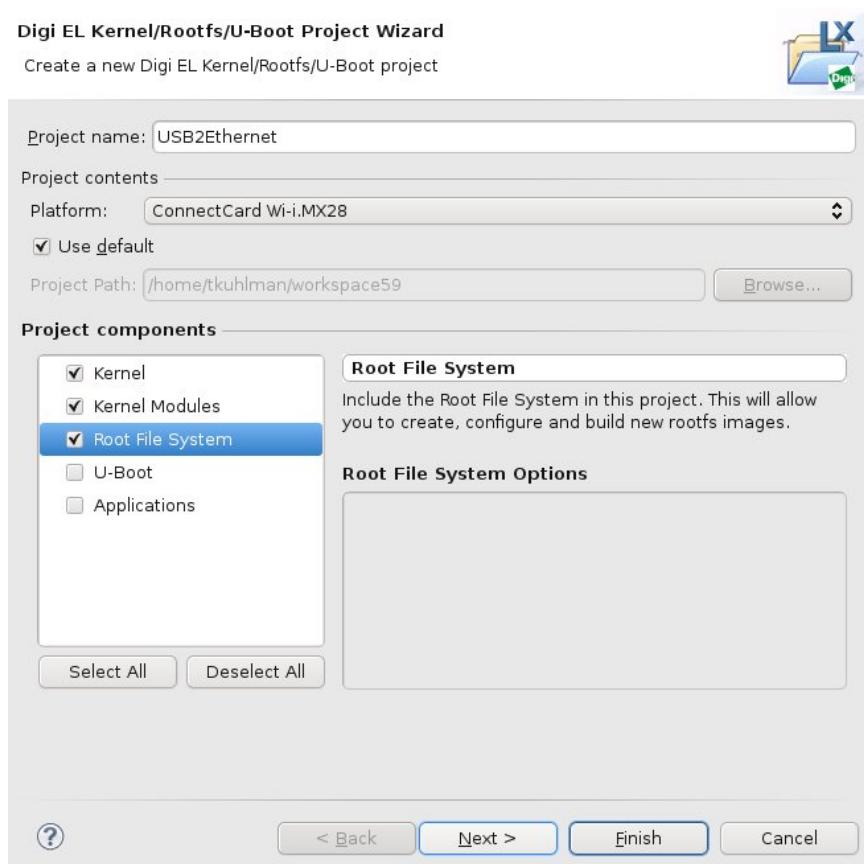
Install Digi Embedded Linux (DEL) 5.9.6.2 or higher, apply latest patches with the Package Manager.

USB to Ethernet bridging with Digi Embedded Linux

3.1 Create a new Digi EL Kernel/Rootfs Project

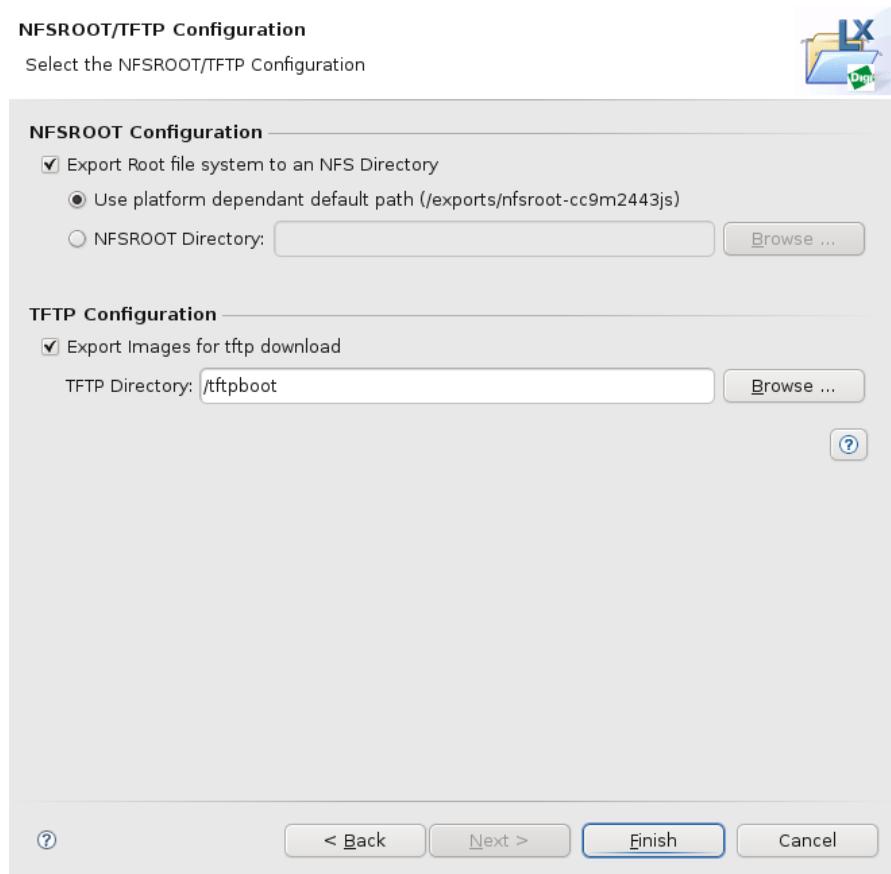


Select Kernel/Kernel Modules and Root File System as project components:



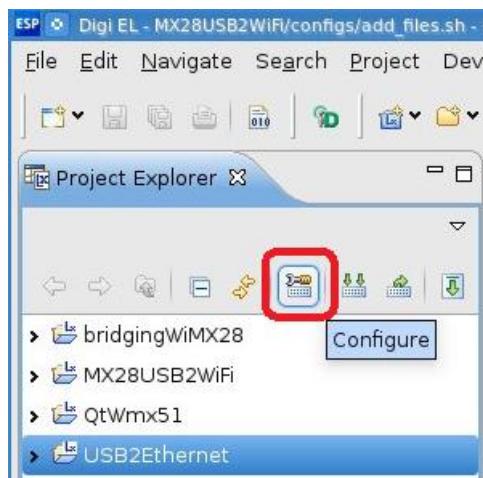
Select appropriate NFSROOT/TFTP Configuration

USB to Ethernet bridging with Digi Embedded Linux



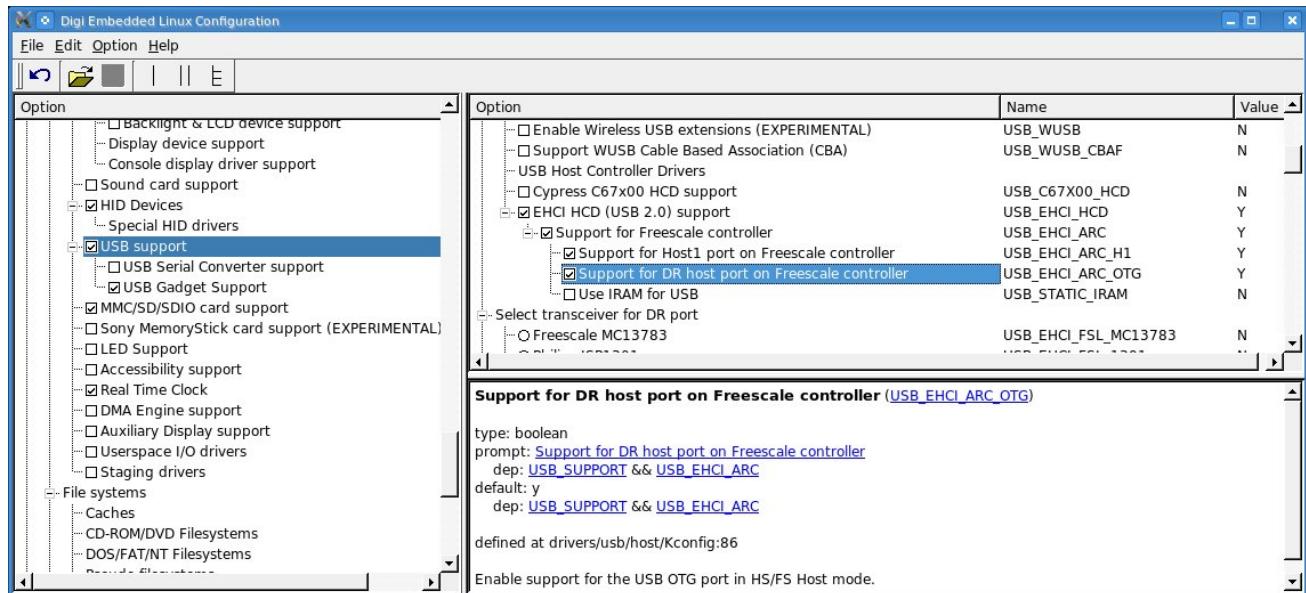
3.2 Configure the kernel for USB device Ethernet gadget

Open the Kernel config:

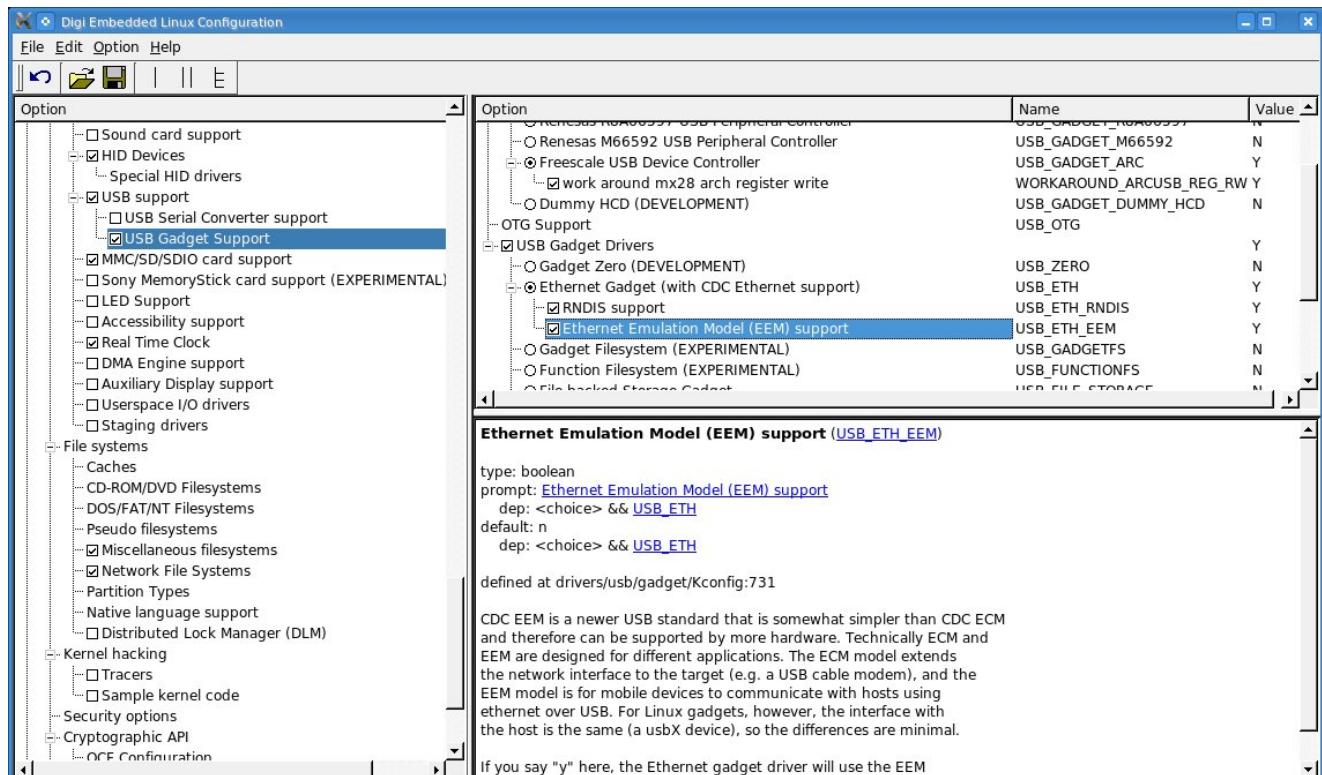


In the USB support section, select DR Host on Freescale OTG driver:

USB to Ethernet bridging with Digi Embedded Linux

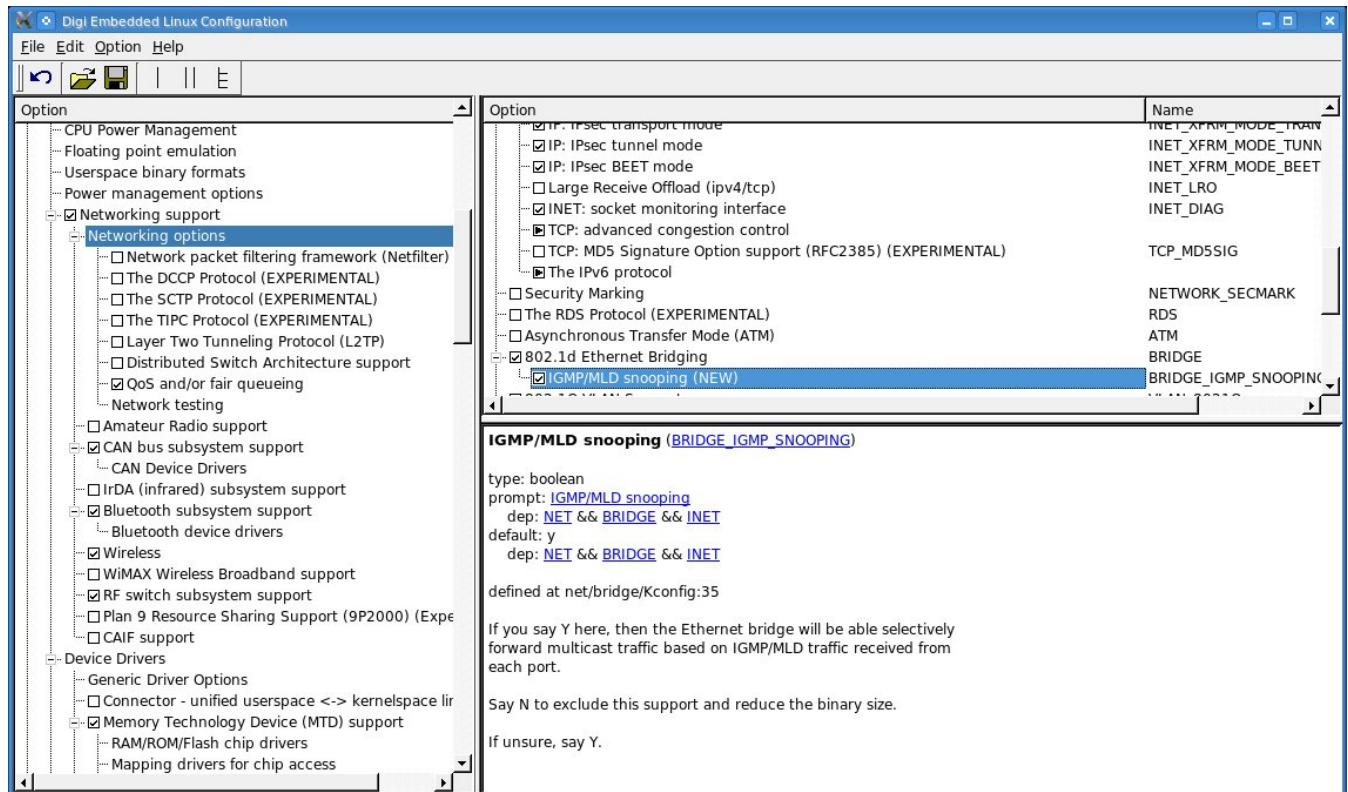


Enable USB Gadget Support and USB Ethernet Gadget with RNDIS and EEM support. If you only use this gadget, you can select it as statically linked kernel module “checkbox” (otherwise as dynamically linked module “.”):

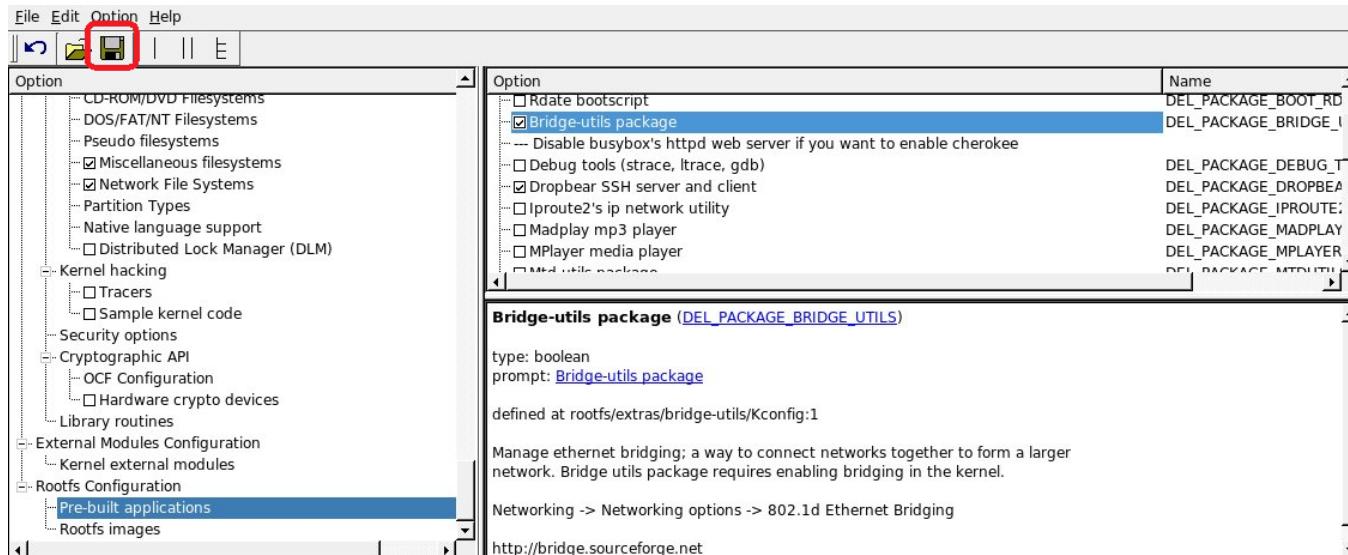


In the Networking support section, enable 802.1d Ethernet Bridging with snooping under Networking options:

USB to Ethernet bridging with Digi Embedded Linux



Configure rootfs with pre-build Bridge-utils package (brctl command):

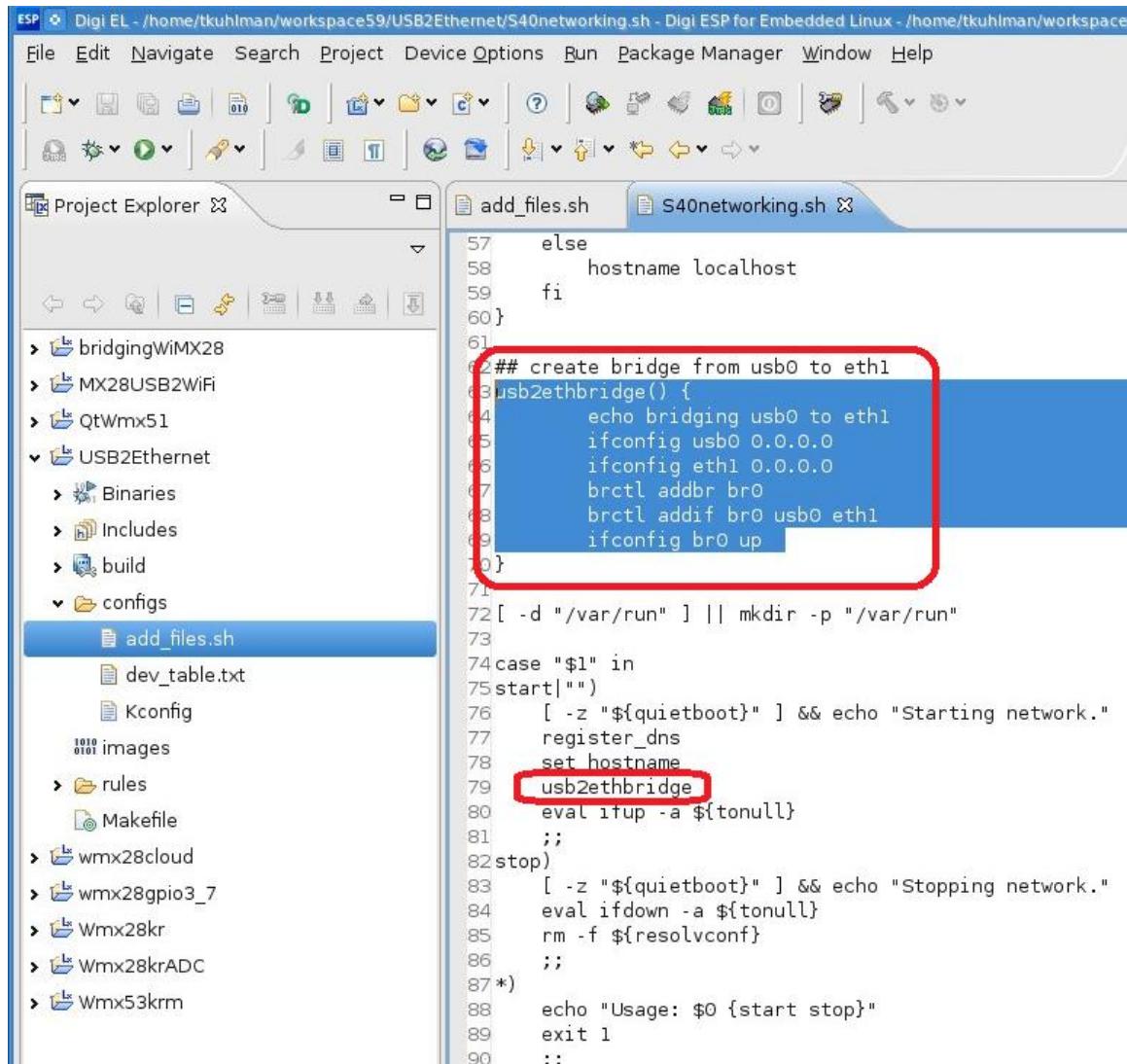


Don't forget to save your settings.

Copy S40networking.sh to your project directory and add the commands to configure and enable the bridge during runtime (if you use the Ethernet Gadget as a loadable module you have to load it before), here example to bridge usb0 to eth1 (you may select eth0 also):

USB to Ethernet bridging with Digi Embedded Linux

```
ifconfig usb0 0.0.0.0
ifconfig eth1 0.0.0.0
brctl addbr br0
brctl addif br0 usb0 eth1
ifconfig br0 up
```

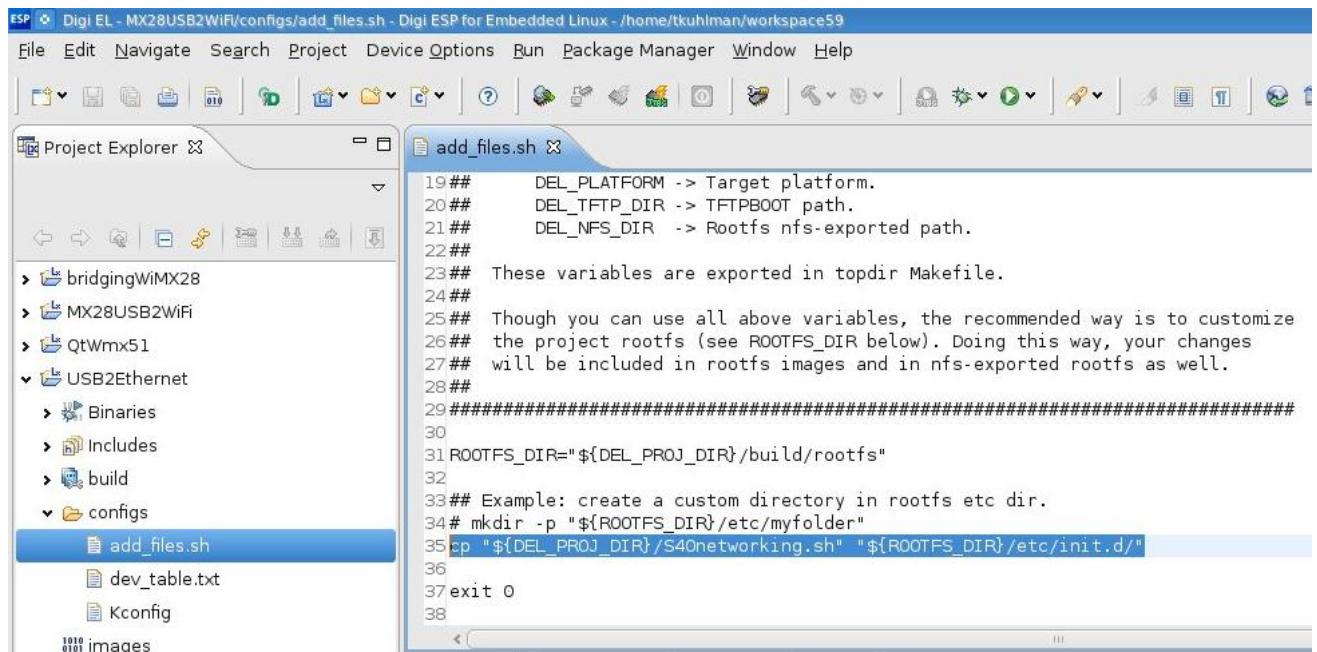


The screenshot shows the Digi ESP IDE interface. The Project Explorer on the left lists several projects and configurations, including 'bridgingWiMX28', 'MX28USB2WiFi', 'QtWmx51', 'USB2Ethernet' (which is expanded to show 'Binaries', 'Includes', 'build', and 'configs'), and 'wmx28cloud'. Inside the 'configs' folder, 'add_files.sh' is selected. The main window displays the contents of 'S40networking.sh'. A red box highlights a section of code in the 'add_files.sh' file:

```
else
    hostname localhost
fi
## create bridge from usb0 to eth1
usb2ethbridge() {
    echo bridging usb0 to eth1
    ifconfig usb0 0.0.0.0
    ifconfig eth1 0.0.0.0
    brctl addbr br0
    brctl addif br0 usb0 eth1
    ifconfig br0 up
}
[ -d "/var/run" ] || mkdir -p "/var/run"
case "$1" in
start|*)
    [ -z "${quietboot}" ] && echo "Starting network."
    register_dns
    set_hostname
    usb2ethbridge
    eval ifup -a ${tonull}
    ;;
stop)
    [ -z "${quietboot}" ] && echo "Stopping network."
    eval ifdown -a ${tonull}
    rm -f ${resolvconf}
    ;;
*)
    echo "Usage: $0 {start stop}"
    exit 1
    ;;
```

Unfold your project select configs and open add_files.sh to add a command to copy this new S40networking.sh to your rootfs during build time:

USB to Ethernet bridging with Digi Embedded Linux



The screenshot shows the Digi ESP for Embedded Linux IDE interface. The Project Explorer on the left lists several projects: bridgingWiMX28, MX28USB2WiFi, QtWmx51, and USB2Ethernet. Under USB2Ethernet, there are sub-folders for Binaries, Includes, build, and configs. The 'configs' folder contains files like add_files.sh, dev_table.txt, Kconfig, and images. The right-hand panel displays the content of the 'add_files.sh' script:

```
19##      DEL_PLATFORM -> Target platform.
20##      DEL_TFTP_DIR -> TFTPBOOT path.
21##      DEL_NFS_DIR -> Rootfs nfs-exported path.
22##
23## These variables are exported in topdir Makefile.
24##
25## Though you can use all above variables, the recommended way is to customize
26## the project rootfs (see ROOTFS_DIR below). Doing this way, your changes
27## will be included in rootfs images and in nfs-exported rootfs as well.
28##
29#####
30
31ROOTFS_DIR="${DEL_PROJ_DIR}/build/rootfs"
32
33## Example: create a custom directory in rootfs etc dir.
34# mkdir -p "${ROOTFS_DIR}/etc/myfolder"
35cp "${DEL_PROJ_DIR}/S40networking.sh" "${ROOTFS_DIR}/etc/init.d/"
36
37exit 0
38
```

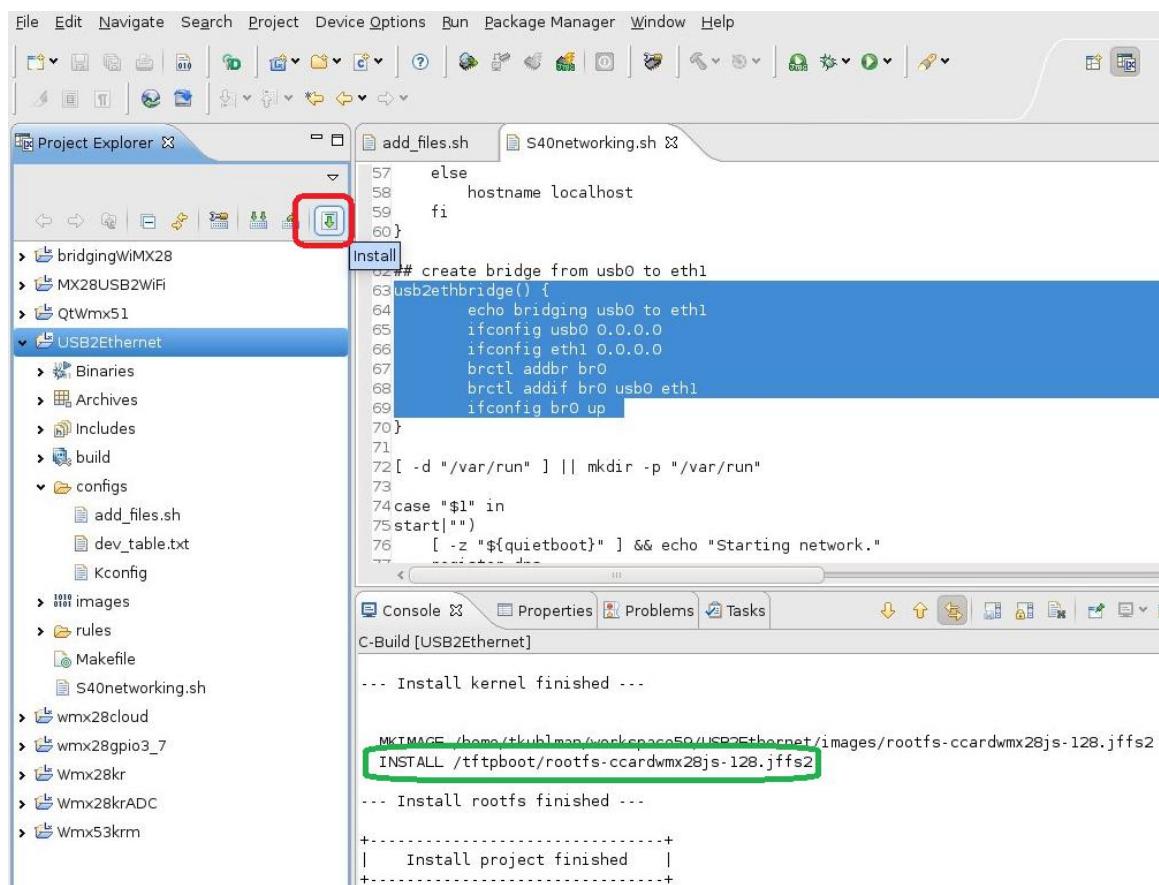
3.3 Built the kernel/rootfs project

Build the project from the project explorer (right mouse button):



Install the project into /tftpboot/ :

USB to Ethernet bridging with Digi Embedded Linux



4 Hardware Setup

- Turn off the development board.
- Connect the power cable
- Ethernet cable ETH0 to development host and ETH1 to router/switch
- USB OTG connector via Micro USB B to USB A cable to your host PC
- Connect a serial null modem cable (pins 2 and 3 crossed) to your host computer (e.g. COMA is the CONSOLE). Plug the cable into Serial Port (DUART CONSOLE) of the Digi development board.
- Open a terminal program on serial port 115200 8N1
- Power on the dev board

5 Deploying firmware to module

In the serial terminal program stop the U-Boot auto boot process.

USB to Ethernet bridging with Digi Embedded Linux

Check network settings to match eth0 from your PC to your module, and program firmware via TFTP to flash of module:

```
# flpart "r"eset "l"inux "q"uit & save  
# update linux tftp  
# update rootfs tftp  
# setenv bootcmd dboot linux flash; saveenv  
# boot (or reset or power cycle)
```

6 Testing from Linux

On your Linux PC plugged via USB to Wi-i.MX28 dev board, load the usbnet driver, e.g.:

```
sudo modprobe usbnet
```

and configure IP (or use tools to setup dhcp client for it):

```
sudo ifconfig usb0 192.168.45.1
```

Now you can use usb0 like an Ethernet network device.

7 Testing from Windows

On the Windows PC plugged via USB OTG to Wi-i.MX28 dev board, when you boot the module or plug in the USB, the USB device should be auto detected in Windows. If not use the linux.inf driver settings. A new network device will appear in Windows and you can either configure it with fixed IP settings or “Obtain automatically” if you have a DHCP server connected to eth1 on the Wi-i.MX28 e.g. a router/switch.

8 Test output on Wi-i.MX28

While you boot the Wi-i.MX28 or connect the USB cable, you should see on the serial console messages like below (or run dmesg) which show that the USB gadget is loaded and the bridge is established (e.g. for trouble shooting):

```
usbcore: registered new interface driver ums-usbat  
ARC USBOTG Device Controller driver (1 August 2005)  
g_ether gadget: using random self ethernet address  
g_ether gadget: using random host ethernet address  
usb0: MAC b6:54:97:20:c5:59  
usb0: HOST MAC ae:4e:26:7a:bc:fd  
g_ether gadget: Ethernet Gadget, version: Memorial Day 2008  
g_ether gadget: g_ether ready  
fsl-usb2-udc: bind to driver g_ether  
..  
g_ether gadget: high speed config #2: RNDIS  
PHY: 0:00 - Link is Up - 100/Full  
Looking up port of RPC 100005/1 on 192.168.42.1  
..  
Initializing random number generator... done  
Starting network.
```

USB to Ethernet bridging with Digi Embedded Linux

```
Starting bluetooth services.  
bridging usb0 to eth1  
ath6kl: temporary war to avoid sdio crc error  
. .  
# dmesg  
g_ether gadget: high speed config #2: RNDIS  
USB Gadget resume begins  
PHY: 0:00 - Link is Up - 100/Full  
ehci_fsl_bus_suspend, Host 1  
. .  
eth1: Freescale FEC PHY driver [Micrel KSZ8031RNL] (mii_bus:phy_addr=0:03, irq=-1)  
device usb0 entered promiscuous mode  
device eth1 entered promiscuous mode  
PHY: 0:03 - Link is Up - 100/Full  
br0: port 2(eth1) entering learning state  
br0: port 2(eth1) entering learning state  
br0: port 1(usb0) entering learning state  
br0: port 1(usb0) entering learning state  
. .  
br0: port 2(eth1) entering forwarding state  
br0: port 1(usb0) entering forwarding state
```

9 References

In case of problems or errors, check the Digi Embedded Linux (DEL) 5.9 Help section 9.29 USB device interface -> USB Ethernet Gadget. And APPLICATION NOTE about Network bridging in the ESP Help.

