Interface Testing on iMX Developer's Kits

Embedded Artists

Embedded Artists AB
Davidshallsgatan 16
SE-211 45 Malmö
Sweden

http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Document Revision History

Revision	*Date*	*Description*
A	2015-11-27	First release
B	2016-11-16	Added section about QSPI. Updated with new COM boards.
C	2017-04-25	Updated PCIe section with new information about i.MX 7

2 Introduction

This document describes commands to do basic testing of the peripheral interfaces on Embedded Artists i.MX 6/7 based COM boards. Different CPUs have different capabilities so each section starts with a table showing what is supported for each CPU.

Additional documentation you might need is.

- The *Getting Started* document for the board you are using.
- The *Adding Displays to iMX Developer's Kits* document about displays and how to use them

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

`Constant width text` – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the
development workstation, i.e., on the workstation where you edit,
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a
document.
```

3 Test Tools

3.1 Introduction

After following the instructions in the *Getting Started* document it is possible to boot into the u-boot or Linux, but how can the different peripheral interfaces available on the iMX Developer's Kits be tested?

This document aims to give short commands to verify the presence and test the functionality of each peripheral interface.

3.2 Requirements

Each test will describe what it requires in the form of cables or other hardware.

The software on the COM Board should be the `core-image-base` image built with frame buffer support and with the needed packages. It is available as a part of the manufacturing tool package on the Embedded Artists iMX Related Resources page. Make sure to use the latest version.

To run tests the board must be booted and a terminal program must be used on a host computer to interact with the board.

3.2.1 Own build

The prepared build comes configured with all required packages. To add the packages to another build, add the following lines to the *conf/local.conf* file:

```
IMAGE_INSTALL_append = " \
    i2c-tools-misc \
    i2c-tools \
    pciutils \
    can-utils \
    iproute2 \
    evtest \
    alsa-utils \
    fbida \
"
```

4 Tests

4.1 eMMC

The COM Boards have eMMC flash that is used to persistently store everything needed to boot into Linux.

COM board	eMMC device in u-boot	eMMC device in Linux
iMX6 SoloX COM	mmc dev 1	/dev/mmcblk2
iMX6 Quad COM	mmc dev 2	/dev/mmcblk3
iMX6 DualLite COM	mmc dev 2	/dev/mmcblk3
iMX6 UltraLite COM	mmc dev 1	/dev/mmcblk1
iMX7 Dual COM	mmc dev 1	/dev/mmcblk2
iMX7 Dual uCOM	mmc dev 1	/dev/mmcblk2

No extra hardware needed for this test.

4.1.1 U-boot

One of the roles of the u-boot is to write new bootloader(s), Linux kernel and file systems to eMMC. To accomplish this there are a set of u-boot commands available:

```
=> mmc
mmc - MMC sub system

Usage:
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
mmc bootbus dev boot_bus_width reset_boot_bus_width boot_mode
 - Set the BOOT_BUS_WIDTH field of the specified device
mmc bootpart-resize <dev> <boot part size MB> <RPMB part size MB>
 - Change sizes of boot and RPMB partitions of specified device
mmc partconf dev boot_ack boot_partition partition_access
 - Change the bits of the PARTITION_CONFIG field of the specified
device
mmc rst-function dev value
 - Change the RST_n_FUNCTION field of the specified device
   WARNING: This is a write-once field and 0 / 1 / 2 are the only
valid values.
mmc setdsr - set DSR register value
```

Take care when using them as they will potentially corrupt the system!

A couple of safe commands (shown for the iMX6 UltraLite COM board):

```
=> mmc dev 1
mmc1(part 0) is current device

=> mmcinfo
Device: FSL_SDHC
Manufacturer ID: fe
OEM: 14e
Name: MMC04
Tran Speed: 52000000
Rd Block Len: 512
MMC version 4.41
High Capacity: Yes
Capacity: 3.5 GiB
Bus Width: 8-bit

=> mmc part

Partition Map for MMC device 1  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     8192            16384           00000000-01     0c
  2     24576           7364608         00000000-02     83
```

4.1.2 Linux

As Linux boots from the eMMC it is already tested when you log in.

To see available disk space:

```
# df -h
Filesystem            Size      Used Available Use% Mounted on
/dev/root             3.4G     65.2M      3.2G   2% /
devtmpfs            340.0M         0    340.0M   0% /dev
tmpfs               500.1M    216.0K    499.9M   0% /run
tmpfs               500.1M     76.0K    500.1M   0% /var/volatile
```

Create a file

```
# echo Hello World > greeting
```

Show the content of the file

```
# cat greeting
Hello World
```

To list files

```
# ls -la
drwxr-xr-x    2 root     root          1024 Sep 23 15:25 .
drwxr-xr-x    3 root     root          1024 Sep 23 14:52 ..
-rw-r--r--    1 root     root            12 Sep 23 15:25 greeting
```

4.2 Network

The COM Carrier Board has two Gigabit Ethernet connectors. Some CPUs only support one Ethernet interface as shown in the table below. The primary Ethernet connector (the only one accessible in the u-boot) is marked in the table below as "left" or "right". Left means the one closest to the HDMI connector.

COM board	Ethernet interfaces in u-boot	Ethernet interfaces in Linux
iMX6 SoloX COM	1 (left)	2 (left)
iMX6 Quad COM	1 (left)	1 (left)
iMX6 DualLite COM	1 (left)	1 (left)
iMX6 UltraLite COM	1 (right)	2 (right)
iMX7 Dual COM	1 (left)	1 (left)
iMX7 Dual uCOM	1 (left)	1 (left)

This test requires one or two network cables, a network with a DHCP server and access to Internet. The examples assume that the network is 192.168.5.0/255.255.255.255. Replace the IP addresses below to match the network that the board is connected to.

4.2.1 U-boot

The u-boot has basic network functionality but only for the first interface so to test network connectivity use the Ethernet connector as indicated in the table above.

Use the ping command to test the network. It only handles IP addresses, i.e. no host names. It also requires the `ipaddr` variable to have the current IP address.

```
=> setenv ipaddr 192.168.5.7
=> ping 192.168.5.22
Using FEC0 device
host 192.168.5.22 is alive
```

4.2.2 Linux

The Linux image has full support for both Ethernet ports and while booting it will initialize the first one (eth0). To see the status of the network interface(s):

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1A:F1:01:9B:E7
          inet addr:192.168.5.71  Bcast:192.168.5.255
mask:255.255.255.0
          inet6 addr: fe80::21a:f1ff:fe01:9be7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:27 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2624 (2.5 KiB)  TX bytes:5643 (5.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

If the CPU supports a second interface (`eth1`), then it can be started with:

```
# ifup eth1
fec 21b4000.ethernet eth1: Freescale FEC PHY driver [Generic PHY]
(mii_bus:phy_addr=2188000.ethernet:02, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
udhcpc (v1.22.1) started
Sending discover...
Sending discover...
libphy: 2188000.ethernet:02 - Link is Up - 1000/Full
IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
Sending discover...
Sending select for 192.168.5.72...
Lease of 192.168.5.72 obtained, lease time 691200
/etc/udhcpc.d/50default: Adding DNS 192.168.5.2
```

As can be seen above the interface is detected and DHCP is used to get an IP address.

One way to test the network is with the `ping` program. Unlike the u-boot version the Linux version handles host names as well (use `Ctrl-C` to end the program):

```
# ping www.sunet.se
PING www.sunet.se (192.36.171.231): 56 data bytes
64 bytes from 192.36.171.231: seq=0 ttl=56 time=16.412 ms
64 bytes from 192.36.171.231: seq=1 ttl=56 time=18.279 ms
64 bytes from 192.36.171.231: seq=2 ttl=56 time=19.125 ms
64 bytes from 192.36.171.231: seq=3 ttl=56 time=17.355 ms
^C
--- www.sunet.se ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 16.412/17.792/19.125 ms
```

The `ping` command uses the first interface (`eth0`) by default. To specify that it should use another interface use the `-I` option:

```
# ping -I eth1 www.sunet.se
```

When using the second interface (`eth1`) it is possible that the ping program fails. This is most likely because the routing table does not handle the interface. To fix this first look at the current routing table:

```
# route
Kernel IP routing table
Destination  Gateway     Genmask         Flags Metric Ref Use Iface
default      192.168.5.1 0.0.0.0         UG    0      0      0 eth0
192.168.5.0  *           255.255.255.0   U     0      0      0 eth0
192.168.5.0  *           255.255.255.0   U     0      0      0 eth1
```

The default route is only for `eth0`, so remove it and add a default route for `eth1` instead:

```
# route del default
# route add default gw 192.168.5.1 eth1
```

After this change eth1 will work instead.

4.3 USB Host

The COM Carrier Board has two USB type A sockets which can be used on all CPUs.

The tests require a USB Memory Stick.

4.3.1 U-boot

The u-boot has USB support for reading/writing USB memories. Connect a USB memory stick to one of the two ports and then issue the following commands:

```
=> usb start
starting USB...
USB0:   Port not available.
USB1:   USB EHCI 1.00
scanning bus 1 for devices... 4 USB Device(s) found
   scanning usb for storage devices... 2 Storage Device(s) found
   scanning usb for ethernet devices... 0 Ethernet Device(s) found

=> usb storage
  Device 0: Vendor: USB      Rev: 1100 Prod: Flash Disk
            Type: Hard Disk
            Capacity: 1912.0 MB = 1.8 GB (3915776 x 512)
  Device 1: Vendor: Kingston Rev: 1.00 Prod: DataTraveler G2
            Type: Removable Hard Disk
            Capacity: 15259.7 MB = 14.9 GB (31252024 x 512)

=> fatls usb 0
 24055271   core-image-base-imx6sxea-com.rootfs.tar.bz2
 79691776   core-image-base-imx6sxea-com.rootfs.ext3
   316520   u-boot-imx6sxea-com.img
  6084744   zimage-imx6sxea-com
    42732   imx6sxea-com-kit.dtb

5 file(s), 0 dir(s)
```

4.3.2 Linux

Linux has support for a wide range of USB devices including mouse, keyboard, memory sticks, hubs etc.

It is possible to see which USB devices are currently connected:

```
# lsusb
Bus 001 Device 004: ID 0951:1624 Kingston Technology DataTraveler G2
Bus 001 Device 003: ID 8087:07dc Intel Corp.
Bus 001 Device 002: ID 0424:2513 Standard Microsystems Corp. 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

When a new USB device is connected some status messages will be printed in the console. The following comes when inserting a USB memory stick:

```
usb 1-1.3: new high-speed USB device number 5 using ci_hdrc
usb-storage 1-1.3:1.0: USB Mass Storage device detected
scsi1 : usb-storage 1-1.3:1.0
scsi 1:0:0:0: Direct-Access     Kingston DataTraveler G2  1.00 PQ:
0 ANSI: 2
sd 1:0:0:0: [sda] 31252024 512-byte logical blocks: (16.0 GB/14.9
GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Incomplete mode parameter data
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] Incomplete mode parameter data
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 1:0:0:0: [sda] Incomplete mode parameter data
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] Attached SCSI removable disk
```

The interesting part above is the "sda: sda1" which indicates which device (`sda1`) that the USB memory stick is assigned to. Section 4.5.2 describes an alternative way to find the device name.

To be able to access the memory stick it must first be mounted:

```
# mkdir /mnt/usb
# mount /dev/sda1 /mnt/usb
```

The memory stick is now available in the /mnt/usb directory on the file system:

```
# ls /mnt/usb/
core-image-base-imx6sxea-com.rootfs.ext3
core-image-base-imx6sxea-com.rootfs.tar.bz2
imx6sxea-com-kit.dtb
u-boot-imx6sxea-com.img
zImage-imx6sxea-com
```

Before physically removing the memory stick from the COM Carrier Board, it should be unmounted to make sure that all pending write operations are committed to prevent data loss:

```
# umount /mnt/usb
```

There are many different USB devices and the level of support varies with the kind of device. Keyboards will work without any extra work – just plug in and start typing. A mouse will work but without a graphical desktop it will be difficult to use it.

4.4 MMC and uSD Cards

The COM Carrier Board has two slots for external memory cards – a slot for the uSD card on the top of the carrier board and a slot for the full size MMC cards on the bottom side.

Note that only one of the slots can be used at a time.

COM board	MMC/uSD device in u-boot	MMC/uSD device in Linux
iMX6 SoloX COM	mmc dev 0	/dev/mmcblk1
iMX6 Quad COM	mmc dev 0	/dev/mmcblk1
iMX6 DualLite COM	mmc dev 0	/dev/mmcblk1
iMX6 UltraLite COM	mmc dev 0	/dev/mmcblk0
iMX7 Dual COM	mmc dev 0	/dev/mmcblk0
iMX7 Dual uCOM	mmc dev 0	/dev/mmcblk0

A uSD or a full size memory card is required to run the tests.

4.4.1 U-boot

The u-boot has support for reading/writing memory cards.

To show information about the memory card it must first be selected with the "mmc dev" command.

```
=> mmc rescan
=> mmc dev 0
=> mmc info
Device: FSL_SDHC
Manufacturer ID: 9
OEM: 4150
Name: AF UD
Tran Speed: 50000000
Rd Block Len: 512
SD version 2.0
High Capacity: No
Capacity: 981.5 MiB
Bus Width: 4-bit
```

To list the content of the sdcard:

```
=> fatls mmc 0
     1491   btngraph.gif
     1396   btnlast.jpg
     1266   btnminus.gif
     1049   btnnext.jpg
     1375   btnplus.gif
```

4.4.2 Linux

A new memory card will be detected automatically when it is inserted and a message like this one will be printed in the console (example is for iMX6 UltraLite COM board so mmc number is 0):

```
mmc0: host does not support reading read-only switch. assuming
write-enable.
mmc0: new high speed SD card at address b368
mmcblk0: mmc1:b368 AF UD 981 MiB
 mmcblk0: p1
```

To use the memory card it must first be mounted:

```
# mkdir /mnt/sdcard
# mount /dev/mmcblk0p1 /mnt/sdcard
```

The card is now mounted:

```
# df -h
Filesystem           Size      Used Available Use% Mounted on
/dev/root            3.4G     65.2M      3.2G   2% /
devtmpfs           340.0M         0    340.0M   0% /dev
tmpfs              500.1M    216.0K    499.9M   0% /run
tmpfs              500.1M     76.0K    500.1M   0% /var/volatile
/dev/mmcblk1p1     981.1M      3.8M    977.4M   0% /mnt/sdcard
```

To see the content:

```
# ls /mnt/scard
DEFCXX.JS         TXTPSET.XML       digi4.gif         metaserv.js
DEFIO01.JS        TXTPSIOA.XML      error.gif         metaset.htm
DEFIO02.JS        TXTPSIOD.XML      excanvas.js       metaset.js
DEFIO03.JS        TXTPSIOS.XML      g_hor1.jpg        metasys.htm
```

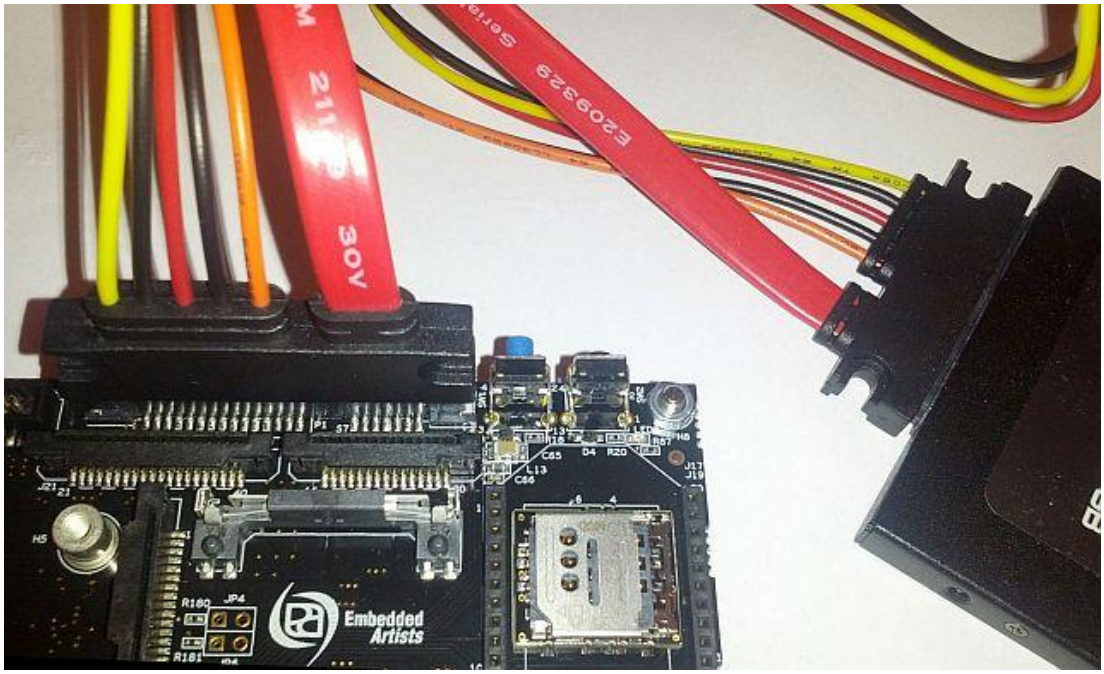
As with the USB memory stick, don't forget to unmount it before physically removing it from the COM Carrier Board:

```
# umount /mnt/scard
```

4.5 SATA

COM board	SATA in u-boot	SATA in Linux
iMX6 SoloX COM	Not Supported by CPU	Not Supported by CPU
iMX6 Quad COM	Not Enabled	Yes
iMX6 DualLite COM	Not Supported by CPU	Not Supported by CPU
iMX6 UltraLite COM	Not Supported by CPU	Not Supported by CPU
iMX7 Dual COM	Not Supported by CPU	Not Supported by CPU
iMX7 Dual uCOM	Not Supported by CPU	Not Supported by CPU

This test requires a SATA disk (can be SSD) and a cable to connect it to the Carrier Board.



4.5.1 U-boot

The u-boot can be (but is not in the default build) configured with SATA support.

4.5.2 Linux

A SATA disk may have many partitions and to see which devices have been assigned to the disk:

```
# ls -l /dev/disk/by-id/
ata-ADATA_SP550_1F3520275635 -> ../../sda
ata-ADATA_SP550_1F3520275635-part1 -> ../../sda1
ata-ADATA_SP550_1F3520275635-part2 -> ../../sda2
…
```

The interesting lines above are starting with 'ata' and shows that the SATA disk's two partitions are available as sda1 and sda2.

To use one of the partitions it must first be mounted:

```
# mkdir /mnt/sata
# mount /dev/sda1 /mnt/sata
```

The card is now mounted:

```
# df -h
Filesystem           Size      Used Available Use% Mounted on
/dev/root           73.5M     57.1M     12.3M  82% /
devtmpfs           340.0M         0    340.0M   0% /dev
tmpfs              500.1M    224.0K    499.9M   0% /run
tmpfs              500.1M     88.0K    500.1M   0% /var/volatile
/dev/sda1           28.8G     32.0K     28.8G   0% /mnt/sata
```

To see the content:

```
# ls /mnt/sata
test_marker4.txt
```

As with the USB memory stick, don't forget to unmount it before physically removing it from the COM Carrier Board:

```
# umount /mnt/sata
```

4.6 GPIO

All pins on the CPUs are used by the peripherals so there are no free pins to test. However, it is possible to change the configuration in the device tree file and change the pin function to be a GPIO instead of for example SPI pin.

4.6.1 U-boot

Not applicable.

4.6.2 Linux

Assuming that a pin (in this example GPIO6_IO13) is configured as GPIO in the device tree file and is not already in used then it can be examined and manipulated from the command line in Linux.

The GPIO pins are controlled with special files in sysfs.

To use a pin it must first be configured as GPIO and to do that the pin's port and pin number must be converted into a number with this formula:

Num = (Port - 1)*32 + Pin

So for GPIO6_IO13 the number is (6 - 1)*32 + 13 = 173.

To configure the pin:

```
# echo 173 > /sys/class/gpio/export
```

If that is successful then a new folder should have been created (see gpio173 below):

```
# ls /sys/class/gpio/
export       gpiochip0    gpiochip160  gpiochip32   gpiochip96
gpio173      gpiochip128  gpiochip192  gpiochip64   unexport
```

A closer inspection of the exported GPIO:

```
# ls /sys/class/gpio/gpio173/
active_low  direction   power       uevent
device      edge        subsystem   value
```

To configure the pin as an input and read the current value (0 or 1):

```
# echo in > /sys/class/gpio/gpio173/direction
# cat /sys/class/gpio/gpio173/value
0
```

To configure the pin as an output and set it high:

```
# echo out > /sys/class/gpio/gpio173/direction
# echo 1 > /sys/class/gpio/gpio173/value
```

4.7 I2C

Each CPU supports a number of I2C busses. The COM board design then either exposes all I2C busses or imposes limits. The table below shows which busses is supported on each COM board and the 7-bit addresses of the devices on each bus. Unused means that there are no devices on the bus. The table below also lists the I2C channel name (A/B/C) on the COM Carrier Board.

COM board	i2c-0 COM Carrier Board: I2C-A	i2c-1	i2c-2
iMX6 SoloX COM	08 1A 4D 55 56	Unused (I2C-B)	Unused * (I2C-C)
iMX6 Quad COM	08 1A 4D 55 56	Unused * (I2C-C)	Unused * (I2C-B)
iMX6 DualLite COM	08 1A 4D 55 56	Unused * (I2C-C)	Unused * (I2C-B)
iMX6 UltraLite COM	08 1A 4D 55 56	Unused * (I2C-B)	Not Supported
iMX7 Dual COM	08 1A 4D 55 56	Unused * (I2C-B)	Unused * (I2C-C)
iMX7 Dual uCOM	08 1A 4D 55 56	Unused * (I2C-B)	Unused (I2C-C)

*) The marked i2c busses are not enabled in the u-boot, only in Linux

Explanations to I2C addresses above:

- 0x08 – PMIC on the COM Board
- 0x1a – Audio Codec on the COM Carrier Board
- 0x4d – AR1021 Touch Controller, typically on COM Display Adapter (or on COM Carrier Board, rev A)
- 0x55 – EEPROM on the COM Board
- 0x56 – EEPROM, typically on COM Display Adapter (or on COM Carrier Board, rev A)

No extra hardware needed to run these tests.

4.7.1 U-boot

The u-boot has i2c commands:

```
=> i2c
i2c - I2C sub-system

Usage:
i2c bus [muxtype:muxaddr:muxchannel] - show I2C bus info
crc32 chip address[.0, .1, .2] count - compute CRC32 checksum
i2c dev [dev] - show or set current I2C bus
i2c loop chip address[.0, .1, .2] [# of objects] - looping read of device
i2c md chip address[.0, .1, .2] [# of objects] - read from I2C device
i2c mm chip address[.0, .1, .2] - write to I2C device (auto-incrementing)
i2c mw chip address[.0, .1, .2] value [count] - write to I2C device (fill)
i2c nm chip address[.0, .1, .2] - write to I2C device (constant address)
i2c probe [address] - test for and show device(s) on the I2C bus
i2c read chip address[.0, .1, .2] length memaddress - read to memory
i2c write memaddress chip address[.0, .1, .2] length - write memory to i2c
i2c reset - re-init the I2C Controller
i2c speed [speed] - show or set I2C bus speed
```

To see the available busses:

```
=> i2c bus
Bus 0:  mxc0
Bus 1:  mxc1
Bus 2:  mxc2
```

Bus 0 is safe to use. The other busses are not initialized and may produce errors when probed. To list all devices on bus 0:

```
=> i2c dev 0
Setting bus to 0

=> i2c probe
Valid chip addresses: 08 1A 4D 55 56
```

The scan found the following devices:

- 0x08 – PMIC on the COM board
- 0x1a – Audio Codec on the COM Carrier Board
- 0x4d – AR1021 Touch Controller, typically on COM Display Adapter (or on COM Carrier Board, rev A)
- 0x55 – EEPROM on the COM Board
- 0x56 – EEPROM, typically on COM Display Adapter (or on COM Carrier Board, rev A)

4.7.2 Linux

To see which I2C busses are available use either

```
# ls /dev/i2c*
/dev/i2c-0  /dev/i2c-1  /dev/i2c-2
```

or

```
# i2cdetect -l
i2c-0   i2c         21a0000.i2c       I2C adapter
i2c-1   i2c         21a4000.i2c       I2C adapter
i2c-2   i2c         21a8000.i2c       I2C adapter
```

To scan for all devices on i2c-1:

```
# i2cdetect -y 0
root@imx6sxea-com:~# i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- UU -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- UU -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- UU -- --
50: -- -- -- -- -- UU 56 -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The scan above uses UU to indicate that a device was not probed as it was marked as being in use by a driver. The address to device mapping is described in the u-boot section above.

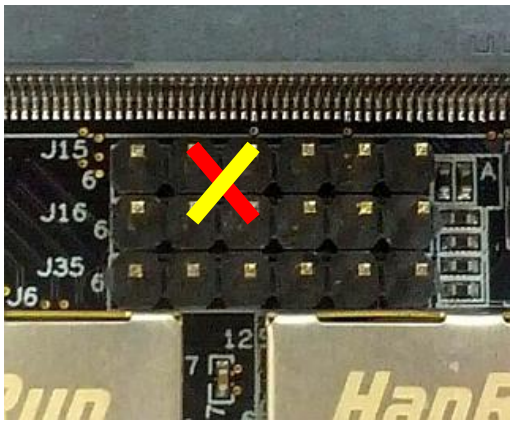
4.8 UART

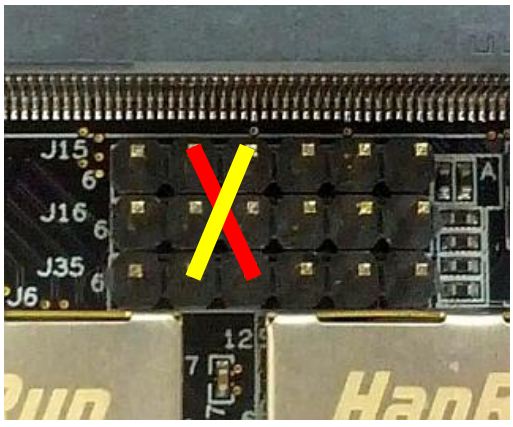
The COM Carrier Board has three UARTs:

COM board	NXP Name	Linux	On COM Carrier Board
iMX6 SoloX COM	UART1	/dev/ttymxc0	Pin list J35 (UART-A)
	UART2	/dev/ttymxc1	Pin list J15 (UART-B)
	UART5	/dev/ttymxc4	Pin list J16 (UART-C)
iMX6 Quad COM	UART1	/dev/ttymxc0	Pin list J16 (UART-C)
	UART4	/dev/ttymxc3	Pin list J15 (UART-B)
	UART5	/dev/ttymxc4	Pin list J35 (UART-A)
iMX6 DualLite COM	UART1	/dev/ttymxc0	Pin list J16 (UART-C)
	UART4	/dev/ttymxc3	Pin list J15 (UART-B)
	UART5	/dev/ttymxc4	Pin list J35 (UART-A)
iMX6 UltraLite COM	UART1	/dev/ttymxc0	Pin list J35 (UART-A)
	UART2	/dev/ttymxc1	Pin list J15 (UART-B)
	UART3	/dev/ttymxc2	Pin list J16 (UART-C)
iMX7 Dual COM	UART1	/dev/ttymxc0	Pin list J35 (UART-A)
	UART2	/dev/ttymxc1	Pin list J15 (UART-B)
	UART3	/dev/ttymxc2	Pin list J16 (UART-C)
iMX7 Dual uCOM	UART1	/dev/ttymxc0	Pin list J35 (UART-A)
	UART2	/dev/ttymxc1	Pin list J15 (UART-B)
	UART3	/dev/ttymxc2	Pin list J16 (UART-C)

UART1 is used by the console so it is tested by connecting a terminal program to the port and then boot into the u-boot.

Two jumper cables are needed to run these tests. For the iMX6 Quad and DualLite COM boards connect as in the right image below. For all other COM boards connect as shown in the left image below.





4.8.1 U-boot

No special tests to run in the u-boot. UART1 is used by the console so it gets tested automatically.

4.8.2 Linux

The examples below are for the iMX6 UltraLite COM board, so replace the devices according to the CPU you are testing.

To see which UARTs are available:

```
# ls /dev/ttymxc*
/dev/ttymxc0  /dev/ttymxc1  /dev/ttymxc2
```

The console uses /dev/ttymxc0, so it does not need further testing.

To test /dev/ttymxc1 and /dev/ttymxc2 cross-connect the RX and TX lines on the COM Carrier Board as shown above.

After connecting, set them up with a baud rate of 115200, raw mode and no echoing:

```
# stty -F /dev/ttymxc1 115200 raw -echo
# stty -F /dev/ttymxc2 115200 raw -echo
```

To listen on /dev/ttymxc1 and send on /dev/ttymxc2:

```
# cat /dev/ttymxc1 &
# echo Hello World > /dev/ttymxc2
Hello World
```

The & in the first command above means that it will be executed in the background. Don't forget to stop the background process when you're done with it by bringing it to the foreground and then pressing Ctrl+C:

```
# fg
cat /dev/ttymxc1
^C
```

4.9 PCI

COM board	PCI in u-boot	PCI in Linux
iMX6 SoloX COM	No	Yes
iMX6 Quad COM	No	Yes
iMX6 DualLite COM	No	Yes
iMX6 UltraLite COM	Not Supported by CPU	Not Supported by CPU
iMX7 Dual COM	No	Yes
iMX7 Dual uCOM	No	Yes

This test assumes that an Intel™ Dual Band Wireless-AC 7260 Plus Bluetooth (http://www.intel.com/content/www/us/en/wireless-products/dual-band-wireless-ac-7260-bluetooth.html) PCIe half mini card is inserted in the COM Carrier Board before booting into Linux

4.9.1 U-boot

Not applicable.

4.9.2 Linux

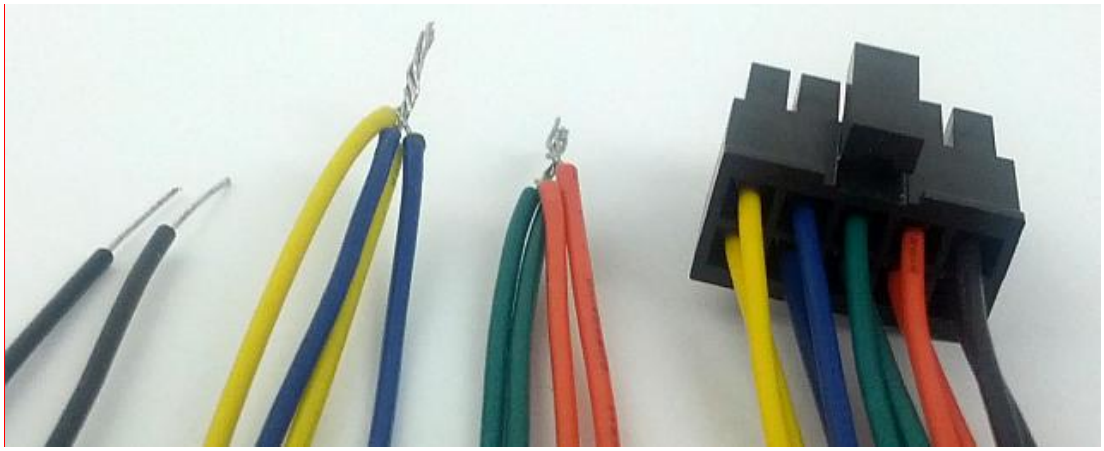
To see if the board is detected, use the `lspci` command:

```
# lspci
00:00.0 PCI bridge: Device 16c3:abcd (rev 01)
01:00.0 Network controller: Intel Corporation Wireless 7260 (rev bb)
```

4.10 CAN

The CAN bus is available on all CPUs but only in Linux. There is no CAN support in the u-boot.

To run these tests the two can busses must be connected together. Use the cable that comes with the Carrier Board. Twist together the yellow and blue wires (CANH). Twist together the orange and green wires (CANL) as shown in the image below.



4.10.1 U-boot

Not applicable.

4.10.2 Linux

Make sure that both `can0` and `can1` are detected:

```
# ip link show
...
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT
    group default qlen 10
    link/can
3: can1: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT
    group default qlen 10
    link/can
...
```

If the text <NOARP,ECHO> appears then the interface is down and must be brought up before it can be used:

```
# ip link set can0 up type can bitrate 125000
flexcan 2090000.can can0: writing ctrl=0x0e312005

# ip link set can1 up type can bitrate 125000
flexcan 2094000.can can1: writing ctrl=0x0e312005
```

As the interfaces are now up, the status will have changed:

```
# ip link show
...
2: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state
    UNKNOWN mode DEFAULT group default qlen 10
    link/can
3: can1: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state
    UNKNOWN mode DEFAULT group default qlen 10
    link/can
...
```

To test the CAN bus the two interfaces must be connected together as shown above.

The following example listens on `can1` and sends on `can0`. Start by listening on the `can1` interface in the background:

```
# candump can1 &
```

Now send a message on `can0` and see what arrives on `can1`:

```
# cansend can0 5A1#11.2233.44556677.88
  can1  5A1   [8]  11 22 33 44 55 66 77 88
```

The second line above is from the `candump` process and it shows that it has detected an 8 byte message with id 5A1 being received by the `can1` interface.

After completing the tests, terminate the background process with the `fg` command and `Ctrl+C`

```
# fg
candump can1
^C
```

4.11 Audio

There is a headphone jack on the COM Carrier Board. Audio out is supported in Linux on all CPUs, but not in the u-boot.

This test requires either headphones or speakers with a 3.5mm audio jack.

4.11.1 U-boot

Not applicable.

4.11.2 Linux

Use headphones in the jacket on the COM Carrier Board.

The `alsa-utils` package comes with a set of sample sound files available on the file system:

```
# ls /usr/share/sounds/alsa/
Front_Center.wav  Noise.wav         Rear_Right.wav
Front_Left.wav    Rear_Center.wav   Side_Left.wav
Front_Right.wav   Rear_Left.wav     Side_Right.wav
```

The sound files can be played using the `aplay` command:

```
# aplay /usr/share/sounds/alsa/Front_Left.wav
```

Note that even if the file is mono, it will be played as stereo.

Another way to test the audio is with the `speaker-test` application:

```
# speaker-test -c2 -l2 -twav
```

The speaker-test application has a lot of options to play with, but in our example the options are `-c2` for stereo, `-l2` to play each sound twice and `-twav` for wav-file testing.

If there is no sound at all, it is most likely because the output is off or muted. To get a long list of all mixer controls:

```
# amixer
Simple mixer control 'Master',0
  Capabilities: pvolume
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 127
  Mono:
  Front Left: Playback 101 [80%] [-20.00dB]
  Front Right: Playback 101 [80%] [-20.00dB]
...
Simple mixer control 'Output Mixer HiFi',0
  Capabilities: pswitch pswitch-joined
  Playback channels: Mono
  Mono: Playback [off]
...
```

Or a shorter list of the names:

```
# amixer controls
numid=2,iface=MIXER,name='Master Playback ZC Switch'
numid=1,iface=MIXER,name='Master Playback Volume'
numid=4,iface=MIXER,name='Line Capture Switch'
numid=5,iface=MIXER,name='Mic Boost Volume'
numid=6,iface=MIXER,name='Mic Capture Switch'
numid=8,iface=MIXER,name='ADC High Pass Filter Switch'
numid=3,iface=MIXER,name='Capture Volume'
numid=10,iface=MIXER,name='Playback Deemphasis Switch'
numid=11,iface=MIXER,name='Input Mux'
numid=14,iface=MIXER,name='Output Mixer HiFi Playback Switch'
numid=12,iface=MIXER,name='Output Mixer Line Bypass Switch'
numid=13,iface=MIXER,name='Output Mixer Mic Sidetone Switch'
numid=7,iface=MIXER,name='Sidetone Playback Volume'
numid=9,iface=MIXER,name='Store DC Offset Switch'
```

On the core-image-base build that you should be running, the only mixer setting that must be changed is the "Output Mixer HiFi Playback Switch" which is turned off by default. To enable it look for the "numid=xx" in the list above to find that it is "numid=14". Enable with the following commands:

```
# amixer -q cset numid=14 on
```

If for some reason there is still no sound, repeat the `amixer` command for all channels that have the word "Playback" in the name.

4.12 Display Output

Each CPU supports a different set of display options. The table below shows what is supported and which display is the default.

COM board	Parallel RGB *	LVDS0	LVDS1	HDMI	MIPI DSI **
iMX6 SoloX COM	Yes	Yes (default)	Not supported	Not supported	Not supported
iMX6 Quad COM	Yes	Yes	Yes (default)	Yes	Yes
iMX6 DualLite COM	Yes	Yes	Yes (default)	Yes	Yes
iMX6 UltraLite COM	Yes (default)	Not supported	Not supported	Not supported	Not supported
iMX7 Dual COM	Yes (default)	Not supported	Not supported	Not supported	Yes
iMX7 Dual uCOM	Yes (default)	Not supported	Not supported	Not supported	Yes

*) The Parallel RGB interface is available through the use of a COM Display Adapter board

**) The MIPI DSI interface is not controlled by the `eadisp` command described below

The Display Solutions for COM Boards page describes the different interfaces, how to physically connect a display and it has a list of some of the supported displays. There is also the *Adding Displays to iMX Developer's Kit* document which has more in-depth descriptions of the commands below.

4.12.1 U-boot

The u-boot has support for the `eadisp` command to control which display interfaces should be enabled and how they should be configured.

The u-boot is setup to show the DENX™ logo on the default display when booting but to see it the display has to be configured correctly.

Run the `eadisp` command to see available options (output is for the iMX6 Quad COM board):

```
=> eadisp

Available display configurations:
    0) lvds0  hannstar:18:64998375,1024,768,220,40,21,7,60,...
    1) lvds1  hannstar:18:64998375,1024,768,220,40,21,7,60,...
    2)   rgb  Innolux-AT070TN:24:33336667,800,480,89,164,75,75,...
    3)   rgb  nhd-4.3-480272ef:24:9009009,480,272,2,2,2,2,41,...
    4)   rgb  nhd-5.0-800480tf:24:29232073,800,480,40,40,29,13,...
    5)   rgb  nhd-7.0-800480ef:24:29232073,800,480,40,40,29,13,...
    6)   rgb  umsh-8864:24:9061007,480,272,20,20,20,20,...
    7)   rgb  umsh-8596-30t:24:33264586,800,480,128,120,20,20,...
    8)   rgb  umsh-8596-33t:24:32917475,800,480,200,200,45,45,...
    9)   rgb  rogin-rx050a:24:32917475,800,480,200,200,45,45,...
   10)  hdmi  1280x720M@60:m24:74161969,1280,720,220,110,20,5,...
   11)  hdmi  1920x1080M@60:m24:148500148,1920,1080,148,88,36,...
   12)  hdmi  640x480M@60:m24:25200342,640,480,48,16,33,10,...
   13)  hdmi  720x480M@60:m24:27027027,720,480,60,16,30,9,62,...

Current Selection:
                enabled prefer  configuration
        rgb:      no      no    Innolux-AT070TN:24:33336667,...
        lvds0:    no      no    hannstar:18:64998375,...
```

```
        lvds1:    no      no    hannstar:18:64998375,...
        hdmi:     no      no    1280x720M@60:m24:74161969,...
```

The command is described in detail in the *Adding Displays to iMX Developer's Kit* document. An example enabling the Parallel RGB interface to use the UMSH-8864 display:

```
=> eadisp enable rgb
=> eadisp conf rgb 6

selecting rgb=umsh-8864

Current Selection:
                enabled prefer  configuration
        rgb:      yes     no    umsh-8864:24:9061007,...
        lvds0:    no      no    hannstar:18:64998375,...
        lvds1:    no      no    hannstar:18:64998375,...
        hdmi:     no      no    1280x720M@60:m24:74161969,...
```

To make the change, save the environment variables and reset the board:

```
=> saveenv
=> reset
```

The display should show the DENX™ logo.

4.12.2 Linux

Each display has its own framebuffer and to see the available framebuffers:

```
# ls /dev/fb*
/dev/fb0  /dev/fb1  /dev/fb2  /dev/fb3  /dev/fb4  /dev/fb5
```

The number of framebuffers depends on the CPU and what was enabled by the eadisp command in the u-boot.

The i.MX 6Quad and 6DualLite SoCs have support for virtual displays (called overlays) which will have their own framebuffers so an iMX6 Quad COM board with LVDS0 and RGB enabled will have four frame buffers:

```
# ls /dev/fb*
/dev/fb0  /dev/fb1  /dev/fb2  /dev/fb3
```

The overlays are /dev/fb1 and /dev/fb3.

To see the resolution, bit depth etc for a framebuffer:

```
# fbset -fb /dev/fb0
mode "800x480-49"
        # D: 33.501 MHz, H: 31.515 kHz, V: 49.243 Hz
        geometry 800 480 800 480 32
        timings 29850 89 164 75 75 10 10
        accel false
        rgba 8/16,8/8,8/0,0/0
endmode
```

The information above tells us the following interesting information:

- The resolution is 800x480 pixels
- Each pixel uses 32 bits, with 8 bits each for red, green and blue. No alpha information.

For a display with 16 bits per pixel and 1024x768 it will look like this instead:

```
# fbset -fb /dev/fb1
mode "1024x768-60"
        # D: 65.003 MHz, H: 48.365 kHz, V: 60.006 Hz
        geometry 1024 768 1024 768 16
        timings 15384 220 40 21 7 60 10
        accel false
        rgba 5/11,6/5,5/0,0/0
endmode
```

Another way to see display information is to look at the files on sysfs:

```
# ls /sys/class/graphics/fb4
bits_per_pixel         fsl_disp_property      state
blank                  mode                   stride
console                modes                  subsystem
cursor                 name                   uevent
dev                    pan                    virtual_size
device                 power
fsl_disp_dev_property  rotate
```

The files can be investigated further.

```
# cat /sys/class/graphics/fb4/fsl_disp_dev_property
lcd

# cat /sys/class/graphics/fb4/fsl_disp_property
1-layer-fb
```

The displays may have power saving options that turns them off after a while. To turn the display back on, write a 0 to the blank control:

```
# echo 0 > /sys/class/graphics/fb0/blank
```

To turn it off, write a 1 instead:

```
# echo 1 > /sys/class/graphics/fb0/blank
```

A quick test is to send data directly to the framebuffer. Assuming the fbset command revealed a 800x480 display with 32 bit addressing, the following command fills the display with random data:

```
# dd if=/dev/urandom of=/dev/fb0 bs=3200 count=480
```

The display can be cleared again by writing zeroes to it:

```
# dd if=/dev/zero of=/dev/fb0 bs=3200 count=480
```

A more practical test is to display an image. Use a USB memory stick and mount it as described in section 4.3.2 . To display an image use the following command:

```
# fbi -T 2 -d /dev/fb0 /mnt/usbstick/*.png
```

4.13 Touch

The interface between the COM Display Adapter board and the COM Carrier Board have an I2C channel for touch controllers either on the COM Display Adapter board (i.e. the AR1021) or on the attached display itself. The LVDS interface on the COM Carrier Board includes an I2C channel as well. HDMI and MIPI-DSI displays can have touch controllers but they will have to be connected using an additional interface (e.g. USB) and that is not handled in this document as it is display specific.

The Display Solutions for COM Boards page describes the different interfaces, how to physically connect a display and it has a list of some of the supported displays. There is also the *Adding Displays to iMX Developer's Kit* document which has more in-depth descriptions of the commands below.

4.13.1 U-boot

The u-boot does not support touch events by itself but it is used to configure the touch interface(s) that Linux will use. The `eatouch` command to control which display interfaces should be enabled and how they should be configured.

Run the `eatouch` command to see available options (output is for the iMX6 Quad COM board):

```
# eatouch

Available Touch Controllers:
    1) ar1021
    2) ilitek
    3) sitronix
    4) egalax
    5) ft5x06

Current Setup:
                       rgb conn.       lvds0 conn.     lvds1 conn.
            ar1021     Disabled        Disabled        Disabled
            ilitek     Disabled        Disabled        Disabled
          sitronix     Disabled        Disabled        Disabled
            egalax     Disabled        Disabled        Disabled
            ft5x06     Disabled        Disabled        Disabled
```

The `eatouch` command is described in detail in the *Adding Displays to iMX Developer's Kit* document. To enable the AR1021 touch controller (used by a display with resistive touch panel connected via the parallel RGB interface):

```
# eatouch enable rgb 1

Current Setup:
                       rgb conn.       lvds0 conn.     lvds1 conn.
            ar1021   Enabled 0x4d      Disabled        Disabled
            ilitek     Disabled        Disabled        Disabled
          sitronix     Disabled        Disabled        Disabled
            egalax     Disabled        Disabled        Disabled
            ft5x06     Disabled        Disabled        Disabled
```

To make the change, save the environment variables and reset the board:

```
=> saveenv
=> reset
```

4.13.2 Linux

To see which devices are available:

```
# evtest

No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      20cc000.snvs-pwrkey
/dev/input/event1:      ar1021 I2C Touchscreen
/dev/input/event2:      EETI eGalax Touch Screen
```

In this case there are two touch controllers enabled: AR1021 and eGalax.

Use the `evtest` program again to test if a touch controller works:

```
# evtest /dev/input/event1
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "ar1021 I2C Touchscreen"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value   2494
      Min        0
      Max     4095
    Event code 1 (ABS_Y)
      Value    499
      Min        0
      Max     4095
Properties:
Testing ... (interrupt to exit)
Event: time 1446835189.051908, type 3 (EV_ABS), code 0 (ABS_X), value 2791
Event: time 1446835189.051908, type 3 (EV_ABS), code 1 (ABS_Y), value 1918
Event: time 1446835189.051908, -------------- SYN_REPORT ------------
Event: time 1446835189.063241, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1446835189.063241, -------------- SYN_REPORT ------------
Event: time 1446835189.069177, type 3 (EV_ABS), code 1 (ABS_Y), value 1919
Event: time 1446835189.069177, -------------- SYN_REPORT ------------
Event: time 1446835189.087452, type 3 (EV_ABS), code 0 (ABS_X), value 2792
Event: time 1446835189.087452, type 3 (EV_ABS), code 1 (ABS_Y), value 1917
…
```

The program will continue to listen for and display touch events until stopped with `Ctrl+C`.

4.14 QSPI

COM boards	QSPI in u-boot	QSPI in Linux
iMX6 SoloX COM	Yes	Yes
iMX6 Quad COM	Not Supported by CPU	Not Supported by CPU
iMX6 DualLite COM	Not Supported by CPU	Not Supported by CPU
iMX6 UltraLite COM	Not Supported by CPU	Not Supported by CPU
iMX7 Dual COM	Yes	Yes
iMX7 Dual uCOM	Yes	Yes

The iMX6 SoloX COM board has two 32Mbyte QSPI Flash memories.

The iMX7 Dual COM board has one 32MByte QSPI Flash memory.

The iMX7 Dual uCOM board has one 32MByte QSPI Flash memory on the uCOM Adapter Board.

4.14.1 U-boot

The u-boot has the `sf` command to handle SPI flash:

```
=> sf
sf - SPI flash sub-system

Usage:
sf probe [[bus:]cs] [hz] [mode] - init flash device on given SPI
                                  bus and chip select
sf read addr offset len         - read `len' bytes starting at
                                  `offset' to memory at `addr'
sf write addr offset len        - write `len' bytes from memory
                                  at `addr' to flash at `offset'
sf erase offset [+]len          - erase `len' bytes from `offset'
                                  `+len' round up `len' to block
                                  size
sf update addr offset len       - erase and write `len' bytes from
                                  memory at `addr' to flash at
                                 `offset'
```

The two memories can be seen with the `probe` command:

```
=> sf probe 0:0
SF: Detected N25Q256 with page size 256 Bytes, erase size 4 KiB,
total 32 MiB

=> sf probe 1:0
SF: Detected N25Q256 with page size 256 Bytes, erase size 4 KiB,
total 32 MiB
```

4.14.2 Linux

To see the QSPI flash (output from iMX6 SoloX COM board):

```
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 02000000 00010000 "21e4000.qspi"
mtd1: 02000000 00010000 "21e4000.qspi"
```

The table shows mtd0 and mtd1 but the corresponding block devices are mtdblock0 and mtdblock1. To test the flash start by creating a test file with 16Kbyte random data:

```
# dd if=/dev/urandom of=write.dat bs=1024 count=16
```

Write the random data to the block device:

```
# time dd if=write.dat of=/dev/mtdblock0
32+0 records in
32+0 records out

real    0m0.074s
user    0m0.000s
sys     0m0.030s
```

Read back the data:

```
# time dd if=/dev/mtdblock0 of=read.dat bs=1024 count=16
16+0 records in
16+0 records out

real    0m0.006s
user    0m0.000s
sys     0m0.000s
```

Compare the two files to make sure that nothing was lost:

```
# diff read.dat write.dat
```

If the files are identical the diff command will not output anything. If there is a difference then it will look like this:

```
# diff read.dat write.dat
Files read.dat and write.dat differ
```