Samsung

ARTIK™ Modules 7

710 Software User Guide

TABLE OF CONTENTS

List of Figures

List of Tables

VERSION HISTORY

Revision	Date	Description	Maturity
V1.0	October 19, 2016	ARTIK 710 Software User Guide	Release

INTRODUCTION

This guide will provide you with an in-depth introduction to your Samsung ARTIK™ 710 Development Board. This document will touch upon all relevant functionality that is present on the ARTIK 710 Development Board as depicted in *Figure 1*. Each section in this user guide will explain one of the interfaces, with a software centric approach.

There are at least 3 potential ways on the ARTIK 710 Development Board to access I/O devices using a software approach:

The first method to interface with the variety of peripherals is using command line tools. This is usually the initial approach to verify functionality and features of individual IO devices. Many of the examples throughout this manual use command line tools that are natively present or can be installed.

The second method we explore is the use of scripting languages such as Python. Many examples throughout this manual will cover this approach.

The third method of accessing peripheral devices is through the use of the Software API that Samsung provides as part of the development environment. Examples of this approach cannot be found in this manual but will be found in the Software API document.

Throughout this manual our focus is on User mode examples. Kernel mode examples like designing a driver are not covered in this user guide. For more information on kernel mode programming please consult the ARTK 710 Software Developers Guide.

Figure 1. ARTIK 710 Development Board

CONNECTIVITY

EXPLORING ZIGBEE

This section will describe how to test the ZigBee® functionality in the ARTIK 710 Development Board that holds the ARTIK 710 Module.

TESTING ZIGBEE CONNECTION

Once the ZigBee daemon that is running as a default process on the ARTIK 710 Module is installed, we can test the ZigBee functionality. Note that after initialization, the ZigBee daemon is reset automatically and it will take about 10 seconds, before the daemon is up and running. There are 2 basic ways to test the ZigBee functionality:

1. Use the 'zigbee-test' application. Once a testing application runs we can see what commands are available using the 'h' option on the command line.

```
$ zigbee-test ON_OFF_LIGHT
$ zigbee-test ON_OFF_SWITCH
```

   Note that in order to test the ZigBee functionality at least 2 devices are required.

2. Use the 'zigbee-cli' application as follows:

```
$ zigbee-cli ON_OFF_LIGHT
$ zigbee-cli ON_OFF_SWITCH
```

Once a CLI application runs, Ember® commands are available. Example Ember commands are 'network form 20 3 0xabcd', 'network find joinable', or 'zcl on-off on / send 0 1 1'.

When using the 'zigbee-cli' application, below commands can check the ZigBee daemon logs:

```
$ journalctl –f | grep zigbeed &
```

EXPLORING BLUETOOTH®

OVERVIEW

Bluetooth® has a variety of profiles that are created for specific use cases. This section will describe some of the supported profiles that you can use in your development efforts. The Bluetooth interface can also be used as an alternative way to wirelessly communicate with your ARTIK 710 Module.

PAN: PERSONAL AREA NETWORK PROFILE

To communicate between a Samsung phone and your ARTIK 710 Development Board you can setup a PAN profile using Bluetooth. The following 4 steps can be defined to create and verify a connection:

1. Run the following script: 'bluetoothctl' on your native ARTIK 710 Development Board. Your terminal will look something like this.

```
$ bluetoothctl
[bluetooth]# agent on
[bluetooth]# agent KeyboardDisplay
[bluetooth]# default-agent
[bluetooth]# scan on
```

```
[bluetooth]# pair “mac address of Samsung Phone”
[bluetooth]# exit
```

2. Confirm passkey on phone and enable tethering.
3. Run the following script: ‘connmanctl’ on your native ARTIK 710 Development Board. Your terminal will look something like this.

```
$ connmanctl
connmanctl> services
```

Confirm the Bluetooth service key #

```
[bluetooth]# connect “Bluetooth_#_#”
[bluetooth]#  exit
```

4. Ping the Samsung phone

```
$ ping 8.8.8.8
```

You should now see successful packet send and receive responses on your terminal.

HID: Human Interface Device profile

The HID profile can be used when you want to connect a Bluetooth keyboard to your ARTIK 710 Development Board. The following example shows how to connect a ZAGG pocket Bluetooth keyboard to your ARTIK 710 Development Board.

```
$ bluetoothctl
[bluetooth]# pair “mac address of ZAGG keyboard”
[bluetooth]# connect “mac address of ZAGG keyboard”
[bluetooth]# exit
```

Now you can start using the keyboard.

PXP: ProXimity Profile

In this section we will use the ‘test-heartrate’ and the ‘test-proximity’ files that can be found under ’/etc/bluetooth/test’. Once located we can go through the following steps to test the Proximity Profile using a HTC fetch proximity sensor:

1. Reset the HTC fetch proximity sensor by holding the button for a few seconds.
2. Run the following script: ‘bluetoothctl’ on your native ARTIK 710 Development Board. Your terminal will look something like this.

```
$ dnf install python3-dbus python3-gobject
$ bluetoothctl
[bluetooth]# pair “mac address of HTC fetch”
[bluetooth]# exit
$ cd /etc/bluetooth/test
$ ./example-gatt-pxp
```

3. Walk away with the HTC sensor and notice that the device starts to beep when out of range of your Development environment.

HRP : Heart Rate Profile

To run the Heart Rate Profile do the following:

1. Run the following script: ‘bluetoothctl’ on your native ARTIK 710 Development Board. Your terminal will look something like this.

```
$ dnf install python3-dbus python3-gobject
$ bluetoothctl
[bluetooth]# pair “mac address of Polar Heart rate sensor”
[bluetooth]# connect “mac address of Polar Heart rate sensor”
[bluetooth]# exit
$ ./example-gatt-client
```

2. Once started you will see the heart rate readings on your terminal.

OBEX FTP : OBJECT EXCHANGE FILE TRANSFER PROFILE

To transmit files between your ARTIK 710 Development Board and your Samsung phone you can use the OBEX FTP Bluetooth profile as follows:

1. Download the Astro Bluetooth app on your Samsung phone and make certain the following settings are selected: (1) BT, (2) Discoverable, (3) FTP.
2. Pair your ARTIK 710 Development Board with your Samsung phone using the ‘bluetoothctl’ script. Make certain that your ARTIK 710 Development Board is in discoverable mode for BT.

```
$ bluetoothctl
[bluetooth]# discoverable on
[bluetooth]# pair “mac address of Samsung phone”
[bluetooth]# exit
```

3. List folder on phone

```
$ obexftp –b “phone’s mac address” –l
```

4. Send file from Samsung phone to ARTIK 710 Development Board.

```
 $ obexftp –b “phone’s mac address” –g “path & filename”
```

5. Send file from ARTIK 710 Development Board to Samsung phone.

```
$ obexftp –b “phone’s mac address” –p “path & filename”
```

After step 4 and step 5 you should see both files the respective destination folders.

A2DP: ADVANCED AUDIO DISTRIBUTION PROFILE

When you need to stream audio over a Bluetooth connection you can use the A2DP profile. The following steps are needed to setup this profile on your ARTIK 710 Development Board:

1. Turn on your Samsung EO-SG900 level Box mini portable speaker set.
2. Pair your ARTIK 710 Development Board with your Samsung phone using the ‘bluetoothctl’ script. Make certain that your ARTIK 710 Development Board is in discoverable mode for BT.

```
$ bluetoothctl
[bluetooth]# pair “mac address of BT speaker”
[bluetooth]# exit
```

3. On your ARTIK 710 Development Board run pulse audio control

```
$ pactl list cards
```

4. If the Bluetooth speaker is not set then:

```
$ pactl set_card_profile 1 a2dp sink
```

5. On the ARTIK 710 Development Board use 'mplayer' to set the output to 'pulse' and play a '.wav' file available from '/usr/share/sounds/alsa/'. In addition you can always download your own mp3 files.

```
$ mplayer –ao pulse 'filename'
```

6. Now you can hear the audio via the BT speaker.

Exploring Wi-Fi

This section will describe how to configure a generic Wi-Fi connection, how to create a P2P (Peer 2 Peer) wireless connection and how to setup a Soft AP mode connection.

Configuring a Wi-Fi Connection

When configuring this Wi-Fi connection it will be setup in station mode, meaning it will operate as a client that connects to a wireless access point.

1. Attach a Wi-Fi antenna on your ARTIK 710 Module and define the interface that you will scan:

```
$ wpa_cli -iwlan0 scan
```

2. Now scan for the wireless access points.

```
$ wpa_cli scan_results
```

3. Configure 'wpa_supplicant.conf' to include your Wi-Fi router settings. You can use the 'wpa_passphrase' command to write your router SSID and PASSWORD into 'wpa_supplicant.conf'.

```
$ wpa_passphrase MyAP abcd1234 >> /etc/wpa_supplicant/wpa_supplicant.conf
```

Note: Do not run 'wpa_passphrase' to update 'wpa_supplicant.conf' with same ssid at every system boot. 'wpa_supplicant' will connect automatically after system reboot to the ssid you have written to 'wpa_supplicant.conf'.

4. Before restarting the service check the 'wpa_supplicant.conf' file located in '/etc/wpa_supplicant' to verify that a new network block has been included

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=wheel
network={
   ssid="MyAP
   #psk="abcd1234"
   psk=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
}
```

5. **Note:** Make certain that the spelling in the file is correct, the 'wpa_supplicant' will not start at booting time and will not connect to the AP, if there is misspelling in the configuration file.
6. Now restart the 'wpa_supplicant' service.

```
$ systemctl restart wpa_supplicant
```

7. Get an IP address from your DHCP server using 'dhclient'.

```
$ dhclient wlan0
```

8. Now 'ifconfig wlan0' should provide you with an IP address that has been assigned. If you are unable to get an IP address, try 'dhclient -1' and again check with 'dhclient wlan0'

PERMANENT WI-FI SERVICES

One way to automatically start Wi-Fi services when rebooting is by modifying the system-init script. The following script in '/etc/init.d/' with name wlan will start Wi-Fi services when booting.

```
#! /bin/bash
#chkconfig: - 99 10

start() {
/usr/sbin/dhclient wlan0
}

stop(){
kill dhclient
}

restart(){
stop
start
}


case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    restart
    ;;
  *)
    echo "Usage:$0 {start|stop|restart}"
esac

exit 0
```

Now save the script, change its mode to executable and enable it for reboot.

```
$ chmod a+x /etc/init.d/wlan

$ chkconfig --add wlan

$ chkconfig --level 12345 wlan on
```

SOFTAP WI-FI MODE

When you want to independently run the SoftAP, creating a NAT is necessary. To create a NAT follow the steps below.

1. First configure 'hostapd.conf' in '/etc/hostapd/' using your favorite editor.

```
# Customize these for your local configuration...
interface=wlan0
driver=nl80211
ssid=ARTIK_AP
auth_algs=1
hw_mode=g
channel=6
wpa=2
```

```
wpa_passphrase=artik@iot
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
```

2. Next configure 'dnsmasq' in /etc/ using your favorite editor. This will set DHCP, DNS, etc.

```
bind-interfaces
# Specify range of IP addresses for DHCP leases
dhcp-range=192.168.1.2,192.168.1.100
```

3. Stop the connection manager to prevent it from stealing connections and configure the Wi-Fi module operation mode.

```
# First Stop the Connection Manager
$ systemctl stop connman
$ ifconfig eth0 up
$ dhclient eth0

# Reset the Network Driver to change the mode.
$ modprobe -r dhd
$ modprobe dhd op_mode=2
```

4. Now set up wlan0 using the gateway IP-address.

```
# set up the ip address of wlan0
$ ifconfig wlan0 192.168.1.1 up
```

5. Start 'dnsamsq'.

```
$ dnsmasq -C /etc/dnsmasq.conf
```

6. Configure 'iptables' to make the softAP operate in IP Masquerade.

```
# set up the ip-forwarding using MASQUERADE
$ sysctl net.ipv4.ip_forward=1
$ iptables --flush
$ iptables -t nat --flush
$ iptables --delete-chain
$ iptables -t nat --delete-chain
$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ iptables -A FORWARD -i wlan0 -j ACCEPT
```

7. Start 'hostapd'

```
$ hostapd /etc/hostapd/hostapd.conf -B
```

EXPLORING ETHERNET

Ethernet Interface

This section describes how to setup a network using the available Ethernet port. The Ethernet IP address of the ARTIK 710 Module is setup automatically using DHCP.

```
[root@ia-001E06617A39 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.97  netmask 255.255.255.0  broadcast 192.168.200.255
        inet6 fe80::21e:6ff:fe61:7a39  prefixlen 64  scopeid 0x20<link>
        ether 00:1e:06:61:7a:39  txqueuelen 1000  (Ethernet)
        RX packets 5  bytes 689 (689.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 37  bytes 6537 (6.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

If you want to change the IP address dynamically, use the 'ifconfig' command. Changing the IP address temporarily allows access to different networks in the same location without changing the default settings of the board.

```
$ ifconfig eth0 <new IP>
```

```
[root@ia-001E06617A39 ~]# ifconfig eth0 192.168.200.100
[root@ia-001E06617A39 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.100  netmask 255.255.255.0  broadcast 192.168.200.255
        inet6 fe80::21e:6ff:fe61:7a39  prefixlen 64  scopeid 0x20<link>
        ether 00:1e:06:61:7a:39  txqueuelen 1000  (Ethernet)
        RX packets 168  bytes 18295 (17.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 252  bytes 33749 (32.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

You can also change the IP address statically such that it is effective after rebooting the ARTIK 710 Module. The 'connman' tool is the current network manager. To check if the 'connman' network manager is active use:

```
$ systemctl is-active connman.service
active
```

The DHCP server is the default configuration mechanism to assign IP-addresses. To use a specific IP address not provide by the DHCP server use:

```
$ connmanctl
connmanctl> services
*AR Wired            ethernet_124f5445f286_cable
*AR AP-5G            wifi_ec1f72d52113_496f5442697a2d3547_managed_psk
connmanctl> config ethernet_124f5445f286_cable --ipv4 manual 192.168.100.5 255.255.255.0 192.168.100.1
```

To return to the original settings use:

```
connmanctl> config ethernet_124f5445f286_cable --ipv4 dhcp
```

SYSTEM INTERFACES

EXPLORING GPIO

This section describes the General Purpose IO interface, that is present on the ARTIK 710 Development Board. The ARTIK 710 Module firmware (Linux®) provides the 'sysfs' interface to control and monitor GPIOs. For controlling any GPIO from user-space, write the code to create the node and assign its attributes as follows:

1. Create a GPIO node: The control of a GPIO can be exported to user-space by writing its number(N) to the GPIO control interface file '/sys/class/gpio/export'.

```
$ echo 93 > /sys/class/gpio/export
```

2. Configure/Read GPIO attributes using the 'sysfs' entries for various GPIO attributes that can be found under '/sys/class/gpio/gpioN/' (N is the corresponding number of the GPIO) which can be used to configure or read gpio attributes.
    a. 'direction' contains a string of either in or out. This value is normally just written, but can be read back if desired.

```
$ echo out > /sys/class/gpio/gpio93/direction
```

    b. 'value' contains a string of either 0 (low) or 1 (high). If pin direction was configured as:
        i. 'out' – then 'value' can be written or read. Reads return the last value written.
        ii. 'in' – then 'value' can only be read. Reads return the signal state at the GPIO pin.

```
$ echo 1 > /sys/class/gpio/gpio93/value
```

3. Remove GPIO node : The control of a GPIO can be unexported/ reverted back from user-space by writing its number(N) to the GPIO control interface file '/sys/class/gpio/unexport'.

```
$ echo 93 > /sys/class/gpio/unexport
```

For more details, one can refer to: *https://www.kernel.org/doc/Documentation/gpio/sysfs.txt*.

GPIO numbers(N) can be found in the ARTIK 710 Common Board Pin map table as described in the next section. An alternate route to determining the GPIO pin mapping is to analyze the '/sys/kernel/debug/gpio' file on the ARTIK 710 Module using the GPIO bank name.

EXPLORING PIN MAP

In order to program GPIO, PWM, ADC I2C, SPI and UART functionality using simple APIs we are associating the silkscreen naming on the ARTIK 710 Development board with ARTIK pin macros. This pin numbering will always be used as its first argument of the system IO API.

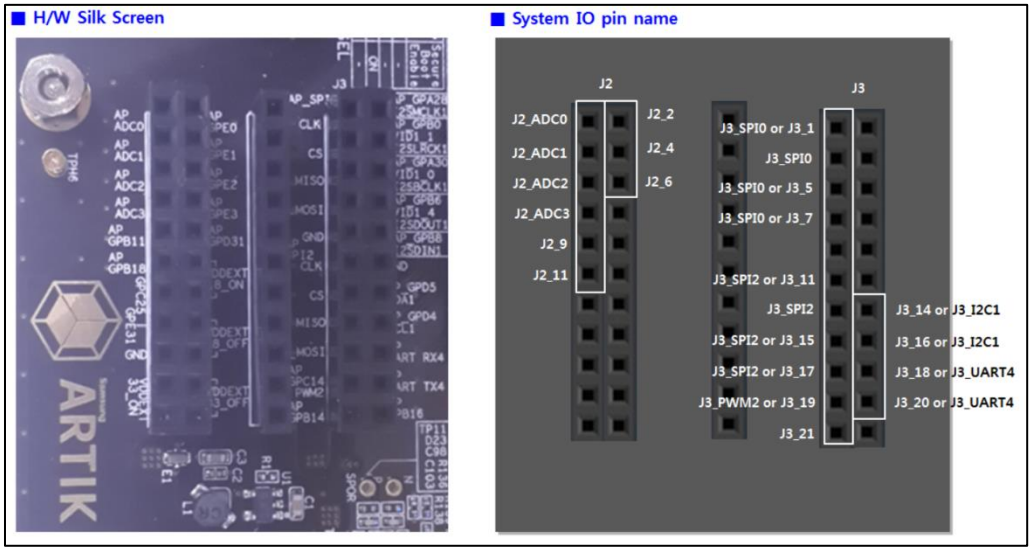


Figure 2. Development Board I/O Placement

Some physical pins have multiple programmable functions as can be seen in *Table 1*, only when rebooting the ARTIK 710 Development board you can switch from one function to the other.

Table 1. Alternate Function on same Physical Pins on ARTIK 710 Development Board

API Pin Name 1	Linux Function 1	API Pin Name 2	Linux Function 2
J3_1	GPIO(93)	J3_SPI0	SPI0.0 CLK
		J3_SPI0	SPI0.0 CS
J3_5	GPIO(95)	J3_SPI0	SPI0.0 MISO
J3_7	GPIO(96)	J3_SPI0	SPI0.0 MOSI
J3_11	GPIO(73)	J3_SPI2	SPI2.0 CLK
		J3_SPI2	SPI2.0 CS
J3_14	GPIO(101)	J3_I2C1	I2C-1 SDA
J3_15	GPIO(75)	J3_SPI2	SPI2.0 MISO
J3_16	GPIO(100)	J3_I2C1	I2C-1 SCL
J3_17	GPIO(76)	J3_SPI2	SPI2.0 MOSI
J3_18	GPIO(60)	J3_UART4	UART4 RX
J3_19	GPIO(78)	J3_PWM2	PWM chip 0 channel 2
J3_20	GPIO(61)	J3_UART4	UART4 TX
J3_21	GPIO(46)		
		J2_ADC0	ADC0
J2_2	GPIO(128)		
		J2_ADC1	ADC1
J2_4	GPIO(129)		
		J2_ADC2	ADC2
J2_6	GPIO(130)		
		J2_ADC3	ADC3
J2_9	GPIO(43)		
J2_11	GPIO(50)		

Table 2. ARTIK 710 Development Board Common board Pin Map

Silk Name	GPIO Name	Linux Function #1	Linux Function #2
ADC0			ADC 0
ADC1			ADC 1
ADC2			ADC 2
ADC3			ADC 3
ADC4			ADC 4
ADC5			ADC 5
I2C_SCL	GPD4	GPIO(100)	I2C-1 SCL
I2C_SDA	GPD5	GPIO(101)	I2C-1 SDA
SPI_CS	GPC10	GPIO(74)	SPI2.0 CS
SPI_CLK	GPC9	GPIO(73)	SPI2.0 CLK
SPI_MISO	GPC11	GPIO(75)	SPI2.0 MISO
SPI_MOSI	GPC12	GPIO(76)	SPI2.0 MOSI
GPIO0	GPE0	GPIO(128)	
GPIO1	GPE1	GPIO(129)	
GPIO2	GPE2	GPIO(130)	
GPIO3	GPB14	GPIO(46)	
GPIO4	GPA14	GPIO(14)	
GPIO5	GPB9	GPIO(41)	
GPIO6	GPA25	GPIO(25)	
GPIO7	GPA0	GPIO(0)	
GPIO8	GPA26	GPIO(26)	
GPIO9	GPA27	GPIO(27)	
AGPIO0	ALIVE1	GPIO(161)	
PWM0_OUT	GPC14	GPIO(78)	PWM chip 0 channel 2
UART0_RX	GPB28	GPIO(60)	UART4 RX
UART0_TX	GPB29	GPIO(61)	UART4 TX

For more information on how to program specific GPIO pins see also *http://developer.artik.io*.

Exploring UART

UART ports can be accessed and controlled from userspace using the '/dev' standard Linux interface. UART ports appear as 'ttyXXX' nodes under the '/dev' directory by default. There are a number of different ways to access UART ports. Few of them are described in the following 3 sections.

Shell/cmdline

One of the convenient ways for experimentation and playing with serial ports from the shell is using the 'stty' (http://linux.die.net/man/1/stty) utility which is by default present on the firmware image of the ARTIK 710 Module. One can use the command line utility 'stty' to configure and print settings of the serial port and once set, the port can be treated as a regular file for reading and writing. The port can be configured as follows:

```
$ stty -F /dev/ttySAC4
speed 9600 baud; line = 0;
-brkint -imaxbel
$ stty -F /dev/ttySAC4 115200
$ stty -F /dev/ttySAC4
speed 115200 baud; line = 0;
-brkint -imaxbel
```

```
$
```

Data can be read using:

```
$ cat /dev/ttySAC4
hello

testing uart
```

And data can be written using:

```
$ echo “hello ..” > /dev/ttySAC4
```

Python

The last proposed method for accessing the serial port is the use of a python script. One can use the ‘pyserial’ module(*https://pypi.python.org/pypi/pyserial*) after installing below packages:

```
$ dnf install python-devel
$ pip install pyserial
```

Once downloaded you can use the API(*http://pythonhosted.org/pyserial/pyserial_api.html#classes*) to communicate with your device as shown in below example:

```
import serial
uart = serial.Serial("/dev/ttySAC4", 115200, timeout=5)

uart.write("Receiving\n")

try:
     while True:
            x = uart.readline()
            print x

except KeyboardInterrupt:
        uart.close()
```

EXPLORING I²C

I²C devices can be controlled from kernel as well as user-space depending upon the complexity and functionality of the I²C devices. For writing drivers in kernel-space, refer to standard Linux documentation (*https://www.kernel.org/doc/Documentation/i2c/writing-clients*).

For accessing I²C devices from user space, the ARTIK 710 Module firmware by default contains ‘i2c-tools’ package which contains ‘i2cdetect’, ‘i2cdump’, ‘i2cget’ and ‘i2cset’ utilities.

I²C DETECT

The ‘i2cdetect’ utility is a user-space program that scans an I²C bus for devices. It generates a table with the list of detected devices on the specified bus. Specify one of the valid bus numbers with the command. The following exercise will show how to use this utility.

```
$i2cdetect 1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1.
```

```
I will probe address range 0x03-0x77.
Continue? [Y/n] Y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- 5d -- 5f
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

I²C SET/GET

The 'i2cset' and 'i2cget' utilities are small helper programs to set/get values of registers visible on the I²C bus. Specify one of the valid bus numbers with the command. For example, the following command reads the byte value from the external sensor (HTS221) register at address 0x5F on I²C bus 1.

```
$ i2cget -f -y 1 0x5F 0x10
0x3F
$ i2cget -f -y 1 0x5F 0x11
0x00
$ i2cget -f -y 1 0x5F 0x12
0x56
$ i2cget -f -y 1 0x5F 0x13
0x32
```

I²C DUMP

The 'i2cdump' utility is a user-space program to dump registers of I²C devices which are visible through the I²C bus.

```
$ i2cdump -y 1 0x5F
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 bc    ...............?
10: 3f 00 56 32 9a be ea a1 9e b2 04 00 e8 01 80 9a    ?.V2????????.????
20: 00 00 00 00 00 00 00 03 d8 c5 e1 00 54 c6 f9 00    .......????.T??.
30: 38 85 a7 1f 00 c4 1b 00 02 03 e9 d6 00 00 34 03    8???.??.????..4?
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 bc    ...............?
90: 3f 00 56 32 9a be ea a1 9e b2 04 00 e8 01 80 9a    ?.V2????????.????
a0: 00 00 00 00 00 00 00 00 d8 c5 e1 00 54 c6 f9 00    ........???.T??.
b0: 38 85 a7 1f 00 c4 1b 00 02 03 e9 d6 00 00 34 03    8???.??.????..4?
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

PYTHON

An alternate route to access I²C interfaces is using a python script. One can use the SMBus module(*https://pypi.python.org/pypi/smbus-cffi/*) after installing below packages:

```
$ dnf install python-devel
$ dnf install libffi-devel
$ pip install smbus-cffi
```

You can look inside the '/sys/class/i2c-dev/' directory to find what number corresponds to which I^2C adaptor. The 'i2c-dev' module is by default loaded. You can use the 'SMBus' module APIs (*http://wiki.erazor-zone.de/wiki:linux:python:smbus:doc*) to communicate with your device as shown in below example:

```
import smbus
import time
bus = smbus.SMBus(7)
address = 0x1d

def status():
        status1 = bus.read_byte_data(address, 1)
        print(status1)
        status2 = bus.read_byte_data(address, 2)
        print(status2)
        return

while True:
        status()
        time.sleep(1)
```

EXPLORING PWM

When you want to control the PWM outputs, you can use 'sysfs' to control the PWM outputs, either by command line interaction or by writing appropriate code (similar to that described above for other interfaces). Here is the path information you will need to export control of these pins, using PWM2 as an example.

```
#define SYSFS_PWM_PATH "/sys/class/pwm/pwmchip0/"
```

Kernel-level control of the PWM pins is typically not necessary unless the period or duty-cycle are rapidly modulated. Using the command line the following exercise demonstrates use of a PWM pin, setting a fixed blink rate.

1. Export PWM2, a PWM2 subdirectory is created.

```
$ echo 2 > /sys/class/pwm/pwmchip0/export
```

2. Set attributes
   a. Set period (Unit: ns) to 1 sec

```
$ echo 1000000000 > /sys/class/pwm/pwmchip0/pwm2/period
```

   b. Set duty_cycle (Unit: ns) to 500 msec

```
$ echo 500000000 > /sys/class/pwm/pwmchip0/pwm2/duty_cycle
```

   c. Enable PWM0

```
$ echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```

   d. Disable PWM0

```
$ echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
```

3. Unexport PWM2

```
$ echo 2 > /sys/class/pwm/pwmchip0/unexport
```

The PWM2 subdirectory is removed.

EXPLORING ADC

When you want to read the values of the various ADC ports, you can use 'sysfs' using the command line. When you need these ADC values in your SW environment you need to write appropriate C or Python code. Here is the path you will need for the ARTIK 710 Development Board:

```
#define SYSFS_ADC_PATH "/sys/devices/platform/c0000000.soc/c0053000.adc/iio:device0/in_voltage0_raw"
```

The following command line examples demonstrate how to read the current value of each ADC pin on the ARTIK 710 Module.

To read a raw voltage from ADC0 use:

```
$ cat /sys/devices/platform/c0000000.soc/c0053000.adc/iio:device0/in_voltage0_raw
```

To read a raw voltage from ADC1 use:

```
$ cat /sys/devices/platform/c0000000.soc/c0053000.adc/iio:device0/in_voltage1_raw
```

The 'in_voltage0_raw' function returns a raw (unscaled) measurement value. To convert it to a voltage measurement, use the following equation: Voltage = 'in_voltageX_raw' *1.8/4095 mV.

Kernel-level monitoring of the ADC pins is also possible, but may not provide a significant advantage over using the standard Linux routines.

EXPLORING USB

The ARTIK 710 supports 3x USB2.0 Host interfaces and 1x USB OTG interface.

USB DEVICE MODE TESTING

Verifying USB Device functionality can be done by installing the adb tool on your host environment. If you have a Linux environment use:

```
$ sudo apt-get install android-tools-adb
```

Once the 'adb' tool is installed, we connect the USB mini-B connector to your host PC and the target USB you want to verify. Run the 'adb daemon' on your ARTIK 710 Development Board using:

```
$ systemctl start adbd.service
```

If you want to run the service permanently use:

```
$ systemctl enable adbd.service
```

You can also push a file using the 'adb push' command from your host PC.

```
$ adb push TEST_FILE /root
```

USB HOST MODE TESTING

USB Host Mode functionality can be verified using the following steps:

Please connect plug in a USB stick into USB 2.0 Host Port (White ports) and run 'dmesg | tail' on the target board to verify mounting information:

```
[ 902.463356] [c0] sda: sda1
[ 902.480563] [c0] sd 0:0:0:0: [sda] Attached SCSI disk.
```

Mount the partition using:

```
$ mount /dev/sda1 /mnt
```

Now you can write a 4GB file for testing using:

```
$ dd if=/dev/zero of=/mnt/testfile bs=1M count=4096
```

MULTI MEDIA

EXPLORING AUDIO

INTRODUCTION

To control the Audio of the ARTIK 710 Development Board you can use the ALSA mixer that is part of the latest BSP. Part of the ALSA mixer utilities is the 'amixer' package. Options of the 'amixer' package can be explored by typing 'amixer' on the command line.

Before using the 'amixer' application, initialize the audio state running the following shell script:

```
$ /usr/bin/audio_setting.sh
```

Now we can continue by playing a '.wav' file using:

```
$ aplay {wav file name}
```

Volume changes can be applied using 'DAC1' (values can vary between [0-175], 175 being the highest volume):

```
$ amixer sset "DAC1" 140
```

You can also record audio using the external or internal microphone. To do this you first select the microphone that you want to use:

```
# Select external MIC
$ amixer sset "RECMIX1L BST1" on
$ amixer sset "RECMIX1R BST1" on
$ amixer sset "RECMIX1L BST2" off
$ amixer sset "RECMIX1R BST2" off

# Select internal MIC
$ amixer sset "RECMIX1L BST1" off
$ amixer sset "RECMIX1R BST1" off
$ amixer sset "RECMIX1L BST2" on
$ amixer sset "RECMIX1R BST2" on

# Enable both MIC input
$ amixer sset "RECMIX1L BST1" on
$ amixer sset "RECMIX1R BST1" on
$ amixer sset "RECMIX1L BST2" on
$ amixer sset "RECMIX1R BST2" on
```

Once the microphone is selected set the recording volume:

```
# Global volume setting
$ amixer sset "Mono ADC" 80%
$ amixer sset "STO1 ADC" 80%

# Internal MIC volume
$ amixer sset "IN2 Boost" 50

# External MIC volume
$ amixer sset "IN1 Boost" 50
```

Once all microphone parameters are set you can run a recording test using:

```
$ arecord -f dat -d 5 test.wav
```

A test.wav file is recorded for 5 seconds using stereo recording at 48kHz sampling frequency with 16-bit little endian. You can check the default parameter settings like sample frequency and resolution using 'arecord –h'.

To maintain the audio settings during a power cycle do the following:

```
$ reboot
```

This will assure that the audio settings are stored. To avoid a reboot use:

```
$ alsactl store
```

Exploring Camera

Overview

The ARTIK710 Development environment provides convenient hardware and software interfaces for camera hardware support. In addition a USB based camera interface is supported.

Environment

When you want to use a camera the following 2 programs need to be installed:

```
$ dnf install fswebcam                                  # Image capture program
$ dnf install ffmpeg                                    # Video encoding/decoding program
```

Preparation

Prepare your selected camera by plugging it into the ARTIK 710 Development board. Once plugged in check the video node information. The default device node of the MIPI camera connected to the ARTIK 710 Development board is 'video6'. This can be verified using the following command:

```
$ ls -al /dev/video6
crw-rw---- 1 root video 81, 3 Jun  1  2016 /dev/video6
```

Display Preview

Once the camera is plugged into the ARTIK 710 Development Board we can show a preview on the display using either 'ffmpeg' or 'gstreamer'.

Using ffmpeg

When using 'ffmpeg' first clear the screen using:

```
$ cat /dev/zero > /dev/fb0
```

Now we can display a preview using:

```
$ ffmpeg -f v4l2 -s 640x480 -r 30 -i /dev/video6 -pix_fmt bgra -f fbdev /dev/fb0
```

The used parameters have the following meaning:

Option	Description
-f v4l2	Use the V4L2 capture device
-s 640x480	The resolution is 640x480
-r 30	Frame rate is 30 fps

Option	Description
-i /dev/video6	The input device for recording is /dev/video6 (MIPI Camera)
-pix_fmt bgra	The Pixel format of LCD screen
-f fbdev /dev/fb0	The output device is a frame buffer device

Using gstreamer

An alternate route is to use 'gstreamer' that supports hardware rendering through the 'nxvideosink' plugin . When using 'gstreamer' first clear the screen using:

```
$ cat /dev/zero > /dev/fb0
```

Now we can display a preview using:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxvideosink
```

The used parameters have the following meaning:

Option	Description
camerasrc	camerasrc gstreamer plugin : input from camera sensor
camera-crop-width=1280	Width of image Unsigned Integer. Range: 0 - 1280 Default: 640
camera-crop-height=720	Height of image Unsigned Integer. Range: 0 - 720 Default: 480
nxvideosink	nxvideosink gstreamer plugin : H/W Video Renderer for LCD or HDMI

We can also show a cropped and scaled image using:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxscaler scaler-crop-
x=100 scaler-crop-y=100 scaler-crop-width=640 scaler-crop-height=320 scaler-dst-width=1920 scaler-dst-
height=1080 ! nxvideosink
```

The used parameters have the following meaning:

Option	Description
camerasrc	camerasrc gstreamer plugin : input from camera sensor
camera-crop-width=1280	Width of image Unsigned Integer. Range: 0 - 1280 Default: 640
camera-crop-height=720	Height of image Unsigned Integer. Range: 0 - 720 Default: 480
scaler-crop-x=100	Start x value of crop (optional)
scaler-crop-y=100	Start y value of crop (optional)
scaler-crop-width=640	Width of crop area (mandatory)
scaler-crop-height=320	Height of crop area (mandatory)
scaler-dst-width=1920	Target width for scaling (mandatory)
scaler-dst-height=1080	Target height for scaling (mandatory)
nxvideosink	nxvideosink gstreamer plugin: H/W Video Renderer for LCD or HDMI

In *Figure 3* the various scaling and cropping parameters used in above example are explained.

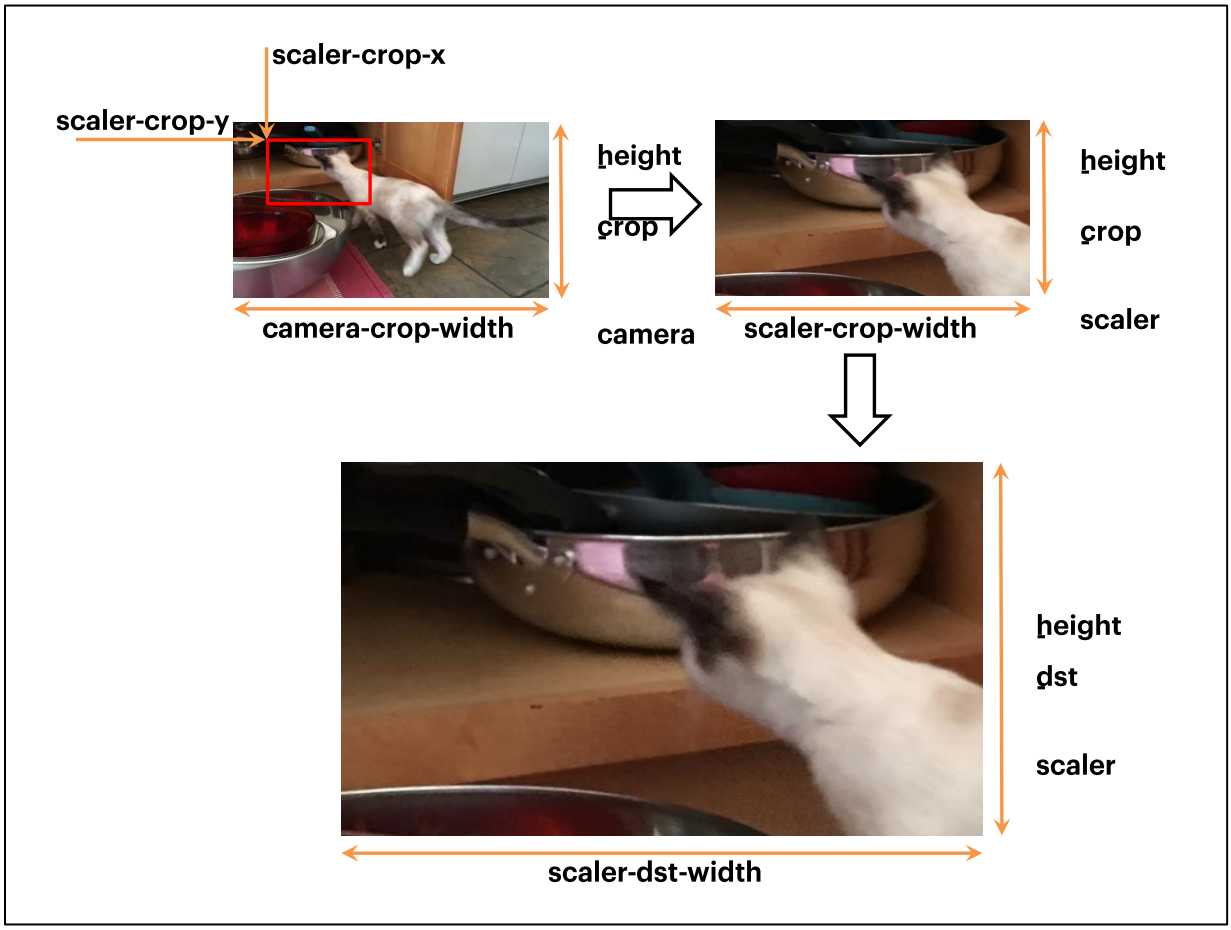


Figure 3. Cropping and Upscaling of a Camera Image using 'gstreamer'

When one wants to scale an image from HD (1280x720) to FHD (1920x1080) without cropping use:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxscaler scaler-crop-
width=1280 scaler-crop-height=720 scaler-dst-width=1920 scaler-dst-height=1080 ! nxvideosink
```

We can also show a preview that is rotated using:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 buffer-type=0 ! videoflip
method=3 ! nxvideosink
```

The used parameters have the following meaning:

Option	Description
camerasrc	camerasrc gstreamer plugin : input from camera sensor
camera-crop-width=1280	Width of image Unsigned Integer. Range: 0 - 1280 Default: 640
camera-crop-height=720	Height of image Unsigned Integer. Range: 0 - 720 Default: 480
buffer-type=0	Specify the buffer type to 'Normal' : We do not use a H/W accelerated buffer type. Since videoflip does not support H/W rotation, we need to set the buffer type to 'normal' . Note that videoflip may cause degradation of performance because the plugin does not use dedicated H/W resources.

Option	Description
videoflip method=3	video flip method (0): Identity (no rotation) (1): Rotate clockwise 90 degrees (2): Rotate 180 degrees (3): Rotate counter-clockwise 90 degrees (4): Flip horizontally (5): Flip vertically (6): Flip across upper left/lower right diagonal (7): Flip across upper right/lower left diagonal
nxvideosink	nxvideosink gstreamer plugin: H/W Video Renderer for LCD or HDMI

MOVIE RECORDING

A utility like 'ffmpeg' is widely used to encode and decode video streams from a device or file. It also supports grabbing input streams from V4L2 (Video4Linux2) devices. In addition the 'gstreamer' utility can be used to record a movie.

Using ffmpeg without hardware acceleration

Since the 'ffmpeg' utility does not support hardware acceleration, the codec is limited in its performance. The following provides a list of encoding limitations:

- VGA (640x480): 30fps, 24fps, 15fps, 10fps and 5fps.
- HD (1280x720): 15fps, 10fps and 5fps.

You can record a video without sound with 'ffmpeg' using:

```
$ ffmpeg -y -f v4l2 -s 640x480 -r 30 -i /dev/video6 -b:v 2048k -vcodec mpeg4 test.avi
```

You can also record a video with sound recording using:

```
$ ffmpeg -y -itsoffset 0.15 -thread_queue_size 2048 -f alsa -i hw:0 -f v4l2 -s 640x480 -r 30 -i
/dev/video6 -b:v 2048k -vcodec mpeg4 -t 20 test.avi
```

The various parameters have the following meaning:

Option	Details
-y	Overwrite output file
-itsoffset 0.15	Specifying a positive offset means that the corresponding streams are delayed by the time duration specified in offset. We can adjust synchronization between audio and video by specifying the audio offset. For instance when audio is faster than video by 0.15s, set: '-itsoffset 0.15' * The recommended itsoffset value for the ARTIK 710 Module using '.avi' video is 0.15s.
-thread_queue_size 2048	Maximum number of queued packets. Note: Raising this value can avoid audio packet loss on low latency and high rate live streams.
-f alsa	Use alsa for audio
-i hw:0	Input audio card is 0
-f v4l2	Use V4L2 capture device
-s 640x480	The resolution is 640x480
-r 30	The frame rate is 30 fps
-i /dev/video6	The input device for recording is /dev/video6
-b:v 2048k	Target bitrate of recorded video : 2Mbps
-vcodec mpeg4	Use MPEG-4 as the video codec
-t 20	The duration of recording is 20 sec
test.avi	The output file name

Using gstreamer with hardware acceleration

You can record an AVI or MP4 movie with no sound using:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxvideoenc
bitrate=12000000 ! avimux ! filesink location=result.avi
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxvideoenc
bitrate=12000000 ! mp4mux ! filesink location=result.mp4
```

The various parameters have the following meaning:

Option	Details
camerasrc	camerasrc gstreamer plugin : input from camera sensor
camera-id=0	Maximum number of queued packets. Raising this value can avoid audio packet loss on low latency and high rate live streams.
camera-crop-width=1280	Width of image Unsigned Integer. Range: 0 - 1280 Default: 640
camera-crop-height=720	Height of image Unsigned Integer. Range: 0 - 720 Default: 480
nxvideoenc	Video encoder gstreamer plugin
bitrate=12000000	bitrate of stream is 12Mbps. Range: 1000000 - 20000000
avimux	AVI video muxer : Multiplex audio and video into a AVI file
mp4mux	MP4 video muxer : Multiplex audio and video into a MP4 file
filesink	filesink gstreamer plugin : Write stream to a file
location=result.avi	location of output file
location=result.mp4	location of output file

You can also record an AVI or MP4 movie using the v4l2 source plugin with no sound:

```
$ gst-launch-1.0 -e v4l2src device=/dev/video6 ! video/x-
raw,format=I420,framerate=30/1,width=1280,height=720 ! nxvideoenc bitrate=12000000 ! avimux ! filesink
location=result.avi
$ gst-launch-1.0 -e v4l2src device=/dev/video6 ! video/x-
raw,format=I420,framerate=30/1,width=1280,height=720 ! nxvideoenc bitrate=12000000 ! mp4mux ! filesink
location=result.mp4
```

The various parameters have the following meaning:

Option	Details
v4l2src	v4l2 source gstreamer plugin : Reads frames from a Video4Linux2 device including camera sensor
device=/dev/video6	device node of MIPI camera 0
video/x-raw,format=I420	Format of video : I420
framerate=30/1	Frame rate : 30 fps
width=1280	width of image : 1280
height=720	height of image : 720
nxvideoenc	Video encoder gstreamer plugin
bitrate=12000000	bitrate of stream is 12Mbps. Range: 1000000 - 20000000
avimux	AVI video muxer : Multiplex audio and video into a AVI file
mp4mux	MP4 video muxer : Multiplex audio and video into a MP4 file
filesink	filesink gstreamer plugin : Write stream to a file
location=result.avi	location of output file
location=result.mp4	location of output file

When one want to simultaneously display video content on an LCD or HDMI screen while recording an AVI or MP4 movie, one can use multiple video pipelines and redirect them to various displays using ‘gstreamer’ in combination with the ‘tee’ plugin. The ‘tee’ plugin provides 1-to-N pipe fitting.

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! tee name=t \
t. ! queue ! nxvideosink \
t. ! queue ! nxvideoenc bitrate=12000000 ! avimux ! filesink location=result.avi

$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! tee name=t \
t. ! queue ! nxvideosink \
t. ! queue ! nxvideoenc bitrate=12000000 ! mp4mux ! filesink location=result.mp4
```

You can also record an FHD movie using the scaler. Although, the maximum resolution of the camera source is limited to HD (1280x720), you can still record a FHD (1920x1080) movie using the HW scaler as follows:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxscaler \
scaler-crop-width=1280 scaler-crop-height=720 scaler-dst-width=1920 scaler-dst-height=1080 ! \
tee name=t t. ! queue ! nxvideosink t. ! queue ! nxvideoenc bitrate=17000000 ! avimux ! filesink \
location=result_fhd.avi

$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! nxscaler \
scaler-crop-width=1280 scaler-crop-height=720 scaler-dst-width=1920 scaler-dst-height=1080 ! \
tee name=t t. ! queue ! nxvideosink t. ! queue ! nxvideoenc bitrate=17000000 ! mp4mux ! filesink \
location=result_fhd.mp4
```

You can also record a MP4 movie using camera source plugin with sound and simultaneous display on HDMI and LCD. For this you have to first install an additional 'gstreamer' plugin to support sound encoding using:

```
$ dnf install gstreamer1-plugins-ugly
```

Now we can start the 'gstreamer' using:

```
$ gst-launch-1.0 -e camerasrc camera-crop-width=1280 camera-crop-height=720 ! tee name=t \
t. ! queue ! nxvideoenc bitrate=12000000 ! queue ! mux. \
autoaudiosrc ! queue max-size-buffers=1000 ! liveadder start-time-selection=2 start-time=600000000 !
lamemp3enc ! queue ! mux. \
t. ! nxvideosink \
mp4mux name=mux ! filesink location=result_a.mp4
```

The various parameters have the following meaning:

Option	Details
camerasrc	camerasrc gstreamer plugin : input from camera sensor
camera-crop-width=1280	Width of image Unsigned Integer. Range: 0 - 1280 Default: 640
camera-crop-height=720	Height of image Unsigned Integer. Range: 0 - 720 Default: 480
nxvideosink	nxvideosink gstreamer plugin: H/W Video Renderer for LCD or HDMI
nxvideoenc	Video encoder gstreamer plugin
bitrate=12000000	bitrate of stream is 12Mbps. Range: 1000000 - 20000000
autoaudiosrc	Wrapper audio source for automatically detected audio source
queue max-size-buffers=1000	Queue between audio source and liveadder. This queue prevents audio frame drops when the system is under heavy load. The 'max-size-buffers' property determines how many buffers can be used for queuing audio samples.
liveadder start-time-selection=2 start-time=600000000	Mixes multiple audio streams start-time-selection=2 start-time=600000000 With this option, we can adjust starting time of audio stream in order to synchronize audio and video on recorded video file. In this case, we can drop audio stream during 600ms.
lamemp3enc	High-quality free MP3 encoder
mp4mux	MP4 video muxer : Multiplex audio and video into a MP4 file
filesink	filesink gstreamer plugin : Write stream to a file
location=result_a.mp4	location of output file

Exploring Display

This section describes the various Display interfaces available on the ARTIK 710 Development Board. The ARTIK 710 Module and consequently the associated Development environment provide the following Display options:

- HDMI v1.4a (1920x1080p)
- Standard LVDS using (GST7D0038:1024x600)

HDMI Interface

To test the HDMI Interface please connect the HDMI cable. After the cable is connected you can boot up the system and should see the eight tuxs (Linux penguin logo) during boot-up. In addition a console that is usually visible on a development PC can now be seen on the HDMI monitor.

LVDS Interface

To test the LVDS Interface the LVDS panel needs to be connected to the ARTIK 710 Development Board. To do this, follow the 2 steps depicted below:

- Due to the increased current draw when using the LVDS interface, power needs to move from Battery operated to DC Jack operated mode see below:

- Make certain that the cable direction between the LVDS panel and the ARTIK 710 Interposer board is correct:

After the display is connected you can boot up the system similar to what was done with the MIPI DSI interface and you should see the eight tuxs (Linux penguin logo) during boot-up.

Display Control

There are a series of functions to control the display. A few of them are given below:

To turn of the Display use:

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

To turn the Display back on use:

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

To disable cursor blink use:

```
$ echo 0 > /sys/class/graphics/fbcon/cursor_blink
```

To enable cursor blink use:

```
$ echo 1 > /sys/class/graphics/fbcon/cursor_blink
```

EXPLORING VIDEO CODEC

The ARTIK 710 Module that is part of the ARTIK 710 Development Board has a multi-media framework based on 'gstreamer'. The ARTIK 710 Module will use the 'gst-launch' toll for encode/decode/transcode videos.

The video material that will be produced when encoding/decoding or transcoding video is stored on the MicroSD card, so make certain that sufficient free space is available. The supported video formats that will be explored are provided in *Table 3*.

Table 3. Supported Video Formats

Function	Standard	Profile	Level
Encoder	H264	Baseline	4.0
	MPEG-4	Standard Profile	5/6
	H263	Profile 3	70
Decoder	H264	Base Profile/Medium Profile/High Profile	4.2
	MPEG-4	Advanced Simple Profile	
	H263	Profile 3	

ENCODING

To encode a video stream, with no sound, from your connected camera use:

```
$ gst-launch-1.0 -e v4l2src device=/dev/video6 ! video/x-raw,format=I420
,framerate=30/1,width=1280,height=720 ! nxvideoenc bitrate=12000000 ! qtmux ! filesink
location=result.mp4
```

The above line describes the following settings:

Option	Description
v4l2src	v4l2 source gstreamer plugin : Reads frames from a Video4Linux2 device including camera sensor
device=/dev/video6	Device node of MIPI camera 0
video/x-raw,format=I420	Format of video : I420
framerate=30/1	Frame rate : 30 fps
width=1280	Width of image : 1280
height=720	Height of image : 720
nxvideoenc	Video encoder gstreamer plugin
bitrate=12000000	Bitrate of stream is 12Mbps. Range: 1000000 – 20000000. The bitrate is an indicator for video quality, in general, the higher the bitrate the better the video quality. Higher bitrates also results in bigger file size.
qtmux	Video muxer : Multiplex audio and video into a MP4 file We recommend that you use qtmux that covers all the codecs we support
filesink	File sink gstreamer plugin : Write stream to a file
location=result.mp4	Location of output file

If we use 'gstreamer' the following resolutions and frame rates are supported:

Size	Frame Rate
VGA (640x480)	30fps
HD (1280x720)	30fps

Using 'gstreamer' the following codecs/muxers are supported:

Codec	Supported Muxer/Demuxer
H264	avimux/avidemux, mp4mux/qtdemux, qtmux/qtdemux
MPEG4	mp4mux/qtdemux, qtmux/qtdemux

TRANSCODING

To transcode an H264 stream into an MPEG-4 stream, first use the 'avdec_h264' software-decode and then re-encode the resulting stream using the 'nxvideoenc' hardware encode into the destination stream.

```
$ gst-launch-1.0 -e filesrc location=input.mp4 ! qtdemux ! h264parse ! avdec_h264 ! nxvideoenc
codec=video/mpeg ! qtmux ! filesink
location=output.mp4
```

The following table explains the various command line parameters used:

Option	Description
v4l2src	The name of camera source plugin
filesrc	The name of input source file
avdec_h264	Software h264 codec, avdec_h263, avdec_mpeg4 are also available
device=/dev/video6	Video node number
format	Format of the input stream
framerate	Specify the framerate of the input stream
width, height	Specify the width and height of the input stream
nxvideoenc* (H/W accelerator)	Specify codec type and bitrate (default: 1024000bps)
filesink location=	Specify output filename "xxx.mp4" and the directory where it is stored by "/xxx/xxx.mp4"

When trying different codec types make certain to specify the right type using:

Codec Type	Codec Name
H264	nxvideoenc video/x-h264
H263	nxvideoenc video/x-h263
MPEG-4	nxvideoenc video/mpeg

DECODING

To decode a stream use:

```
$ gst-launch-1.0 filesrc location="/mnt/result.avi" ! avidemux ! h264parse ! nxvideodec ! nxvideosink
```

The following table explains the various command line parameters used:

Option	Description
firesrc location=	Specify the name and location of the input file
qtdemux*	Specify the name of the demuxer
h264parse**	Specify the name of the parser
nxvideodec	H/W accelerator
videoconvert	Color space conversion
nxvideosink	H/W accelerated display

When trying different demuxers make certain to specify the right type using:

Container Type	Demux name
mp4, qt, 3gpp	qtdemux
avi	avidemux

When trying different codec types make certain to specify the right type using:

Codec Type	Parser name
H264	h264parse
H263	h263parse
MPEG4	mpeg4videoparse
MPEG2	mpegvideoparse

You can also decode with the 'videoflip' plugin to display a rotated video image usually referred to as landscape mode. In this case the 'buffer-type' must be set to 0, to ensure that hardware based 'videoflip' can be used in combination with 'nxvideodec'.

```
$ gst-launch-1.0 filesrc location="/mnt/result.avi" ! avidemux ! h264parse ! nxvideodec buffer-type=0 !
videoflip method=3 ! nxvideosink
```

The following table explains the various command line parameters used:

Option	Description
buffer-type=0	Specify the buffer type to 'Normal' : Do not use H/W accelerated buffer type. Since videoflip does not support H/W rotation, we need to set buffer type to 'Normal'. Note that videoflip may cause degradation of performance because the plugin does not use H/W acceleration.
videoflip method=3	Video flip method (0): No rotation (1): Rotate clockwise 90 degrees (2): Rotate 180 degrees (3): Rotate counter-clockwise 90 degrees (4): Flip horizontally (5): Flip vertically (6): Flip across upper left/lower right diagonal (7): Flip across upper right/lower left diagonal
nxvideosink	nxvideosink gstreamer plugin : H/W Video Renderer for LCD or HDMI

BOOTING ARTIK IMAGE FROM SD-CARD

To boot from a microSD card using your own image you have to set the SW4 switch located on the Interposer board as described in *Table 4*.

Table 4. Booting Options

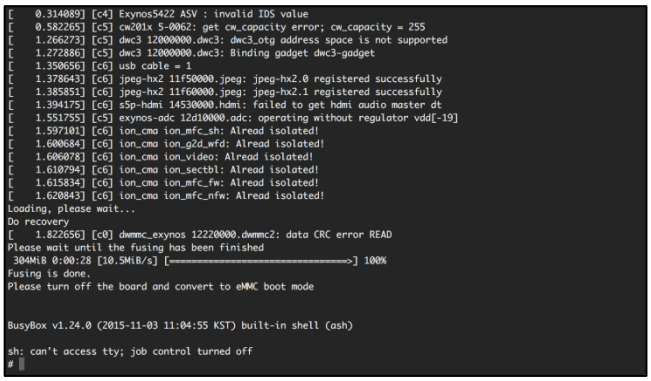
Build Version	eMMC Booting Switch 4	SD Card Booting Switch 4
All build versions	1: off, 2: off, 3: off, 4: off	1: off, 2: off, 3: off, 4: on

*Booting from eMMC is the default option

In addition the following steps should be followed:

1. Download the new ARTIK 710 image that you want to use as the booting image.
2. Download an image SD writing program from (*https://sourceforge.net/projects/win32diskimager/*).
3. Connect the SD card reader to your PC and insert your MicroSD card.
4. Write the image to the MicroSD card using the previously downloaded Windows® writing program
    a. Select the drive in which the MicroSD card is inserted and 'click A'.
    b. Choose the ARTIK 710 image and 'click B'.
    c. Write the image to the MicroSD card and 'click C'.

5. Once the image is written insert the MicroSD card into your ARTIK 710 Development Board while powered off and power on the board. Now press the reset button for about 1 second, and image updating should start. When finished the following such will show:



6. Once the update is completed a reboot from eMMC is required to use the new image. Do not power off the ARTIK 710 Development Board during this first reboot because the OS generates system configuration files during first reboot after a fusing cycle.

EXPLORING THE REAL TIME CLOCK

Most computers have one or more hardware clocks which record the current 'wall clock' time, these devices are called 'Real Time Clock' devices. In our environment there is a battery backup system to maintain time during power cycles. The RTC on the ARTIK 710 Development Board is automatically charged when wall power is used, when power is off it will maintain the RTC values for 2-3 days without using an external power source. Once the onboard power has been depleted fully the time of the RTC will be set to '2016-4-28 20:00:00 (UTC)'.

SET SYSTEM DATE

The System date and time of the ARTIK 710 Development Board can be set using:

```
$ date -s '2016-05-25 10:26:00'
Wed May 25 10:26:00 UTC 2016
```

To set the time to the hardware's clock use:

```
$ hwclock –w
```

To read the hardware time use:

```
$ hwclock –r
```

EXPLORING POWER MANAGEMENT

The ARTIK 710 Module that is part of the ARTIK 710 Development Board has a BSP that supports basic power management functionality that is based on Linux.

SYSTEM SUSPEND/WAKEUP

The ARTIK 710 Module can be set into a 'suspend state' using:

```
$ echo mem > /sys/power/state
```

The wake up from the 'suspend state', press the 'Power' key.

SYSTEM POWER OFF/REBOOT

The ARTIK 710 Module can be powered of using:

```
$ poweroff
```

To reboot the ARTIK 710 Development Board use:

```
$ reboot
```

HOT PLUGGING CPUS

To check the number of CPU's that are online use:

```
$ cat /sys/devices/system/cpu/online
0-7
```

This shows that CPU0 and CPU1 are currently online.

To turn on all CPU's except CPU0 use:

```
$ echo 1 > /sys/devices/system/cpu/cpu1/online
```

To turn off all CPU's except CPU0 use:

```
$ echo 0 > /sys/devices/system/cpu/cpu1/online
```

To check the frequency of the CPUs use:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
400000
```

To check the available CPU frequencies use:

```
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
1400000 1300000 1200000 1100000 1000000 900000 800000 700000 600000 500000 400000
```

To limit granularity of the CPU scaling frequency use:

```
$ echo 800000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq
```

To check the count before entering CPU idle status use:

```
$ cat /sys/devices/system/cpu/cpu0/cpuidle/state0/usage
```

Device Power Management

To limit maximum scaling frequency of device use:

```
$ echo 400000 > /sys/class/devfreq/nx-devfreq/max_freq
```

Thermal Management

To check the current temperature of the CPU use (32000 means 32˚C):

```
 $ cat /sys/class/thermal/thermal_zone0/temp
32000
```

LEGAL INFORMATION

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH THE SAMSUNG ARTIK™ DEVELOPMENT ENVIRONMENT AND ALL RELATED PRODUCTS, UPDATES, AND DOCUMENTATION (HEREINAFTER "SAMSUNG PRODUCTS"). NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. THE LICENSE AND OTHER TERMS AND CONDITIONS RELATED TO YOUR USE OF THE SAMSUNG PRODUCTS ARE GOVERNED EXCLUSIVELY BY THE SAMSUNG ARTIK™ DEVELOPER LICENSE AGREEMENT THAT YOU AGREED TO WHEN YOU REGISTERED AS A DEVELOPER TO RECEIVE THE SAMSUNG PRODUCTS. EXCEPT AS PROVIDED IN THE SAMSUNG ARTIK™ DEVELOPER LICENSE AGREEMENT, SAMSUNG ELECTRONICS CO., LTD. AND ITS AFFILIATES (COLLECTIVELY, "SAMSUNG") ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES, AND SAMSUNG DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, ARISING OUT OF OR RELATED TO YOUR SALE, APPLICATION AND/OR USE OF SAMSUNG PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATED TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

SAMSUNG RESERVES THE RIGHT TO CHANGE PRODUCTS, INFORMATION, DOCUMENTATION AND SPECIFICATIONS WITHOUT NOTICE. THIS INCLUDES MAKING CHANGES TO THIS DOCUMENTATION AT ANY TIME WITHOUT PRIOR NOTICE. THIS DOCUMENTATION IS PROVIDED FOR REFERENCE PURPOSES ONLY, AND ALL INFORMATION DISCUSSED HEREIN IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OF ANY KIND. SAMSUNG ASSUMES NO RESPONSIBILITY FOR POSSIBLE ERRORS OR OMISSIONS, OR FOR ANY CONSEQUENCES FROM THE USE OF THE DOCUMENTATION CONTAINED HEREIN.

Samsung Products are not intended for use in medical, life support, critical care, safety equipment, or similar applications where product failure could result in loss of life or personal or physical harm, or any military or defense application, or any governmental procurement to which special terms or provisions may apply.

This document and all information discussed herein remain the sole and exclusive property of Samsung. All brand names, trademarks and registered trademarks belong to their respective owners. For updates or additional information about Samsung ARTIK™, contact the Samsung ARTIK™ team via the Samsung ARTIK™ website at www.artik.io.