Scientific Journal of Impact Factor (SJIF): 4.14

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3, Issue 10, October -2016

LINUX DEVICE DRIVER DEVELOPMENT ON ARM CORTEX A9 BASED EMBEDDED SYSTEM

Practical Approach

Kapil Kumar[1], *Shoukath Cherukat*[2]

[1] *Department of M.Tech. Electronics Design Technology, NIELIT, Calicut,*
[2] *Department of Embedded Systems, NIELIT, Calicut.*

Abstract — *Linux Operating System (OS) has a kernel source which provides resources to hardware and software. In order to access hardware resources, drivers for the particular hardware must be developed and get registered with kernel of the particular OS. Device driver take on a special role in Linux kernel. They are distinct "black boxes" that make a particular piece of hardware respond to a well-defined internal programming interface; they hide completely the details of how the device works. User activities are performed by means of a set of standardized calls that are independent of the specific driver; mapping those calls to device-specific operations that act on real hardware is then the role of the device driver. Each driver gets registered with the kernel using major and minor number.*

Linux Device Drivers are developed particularly for LCD, Touch screen, Camera, and NFC (PN512). All the four drivers are using different interface in order to communicate with the processor. LCD is interfaced with MIPI, Touch screen uses I2C, Camera driver communicate with the processor using CSI interface and NFC (PN512) is interfaced with UART to the processor.

Keywords- *Device Driver, Kernel, NFC, Camera, LCD, Touch screen*

I. INTRODUCTION

Device drivers provide essential system functionalities by controlling all device-specific hardware operations. Although device drivers play an important role, drivers are one of the most unreliable components in OS. Device drivers are generally known to be unreliable because developing high-quality device drivers is difficult. There are several reasons why. First, the driver developers have to understand both hardware and software. To properly operate the hardware device, the developer has to be aware of all hardware registers to trigger valid operations. At the same time, the developer has to understand software, specifically the underlying operating system, or kernel. Because kernel APIs have many variants and complex implication in its behavior, it is generally difficult to write code. This paper focuses on development methodology of device drivers particularly for LCD, Touch Screen, Camera, & NFC for i.MX 6 based applications processors platform that ARM® Cortex® architecture, including Cortex-A9.

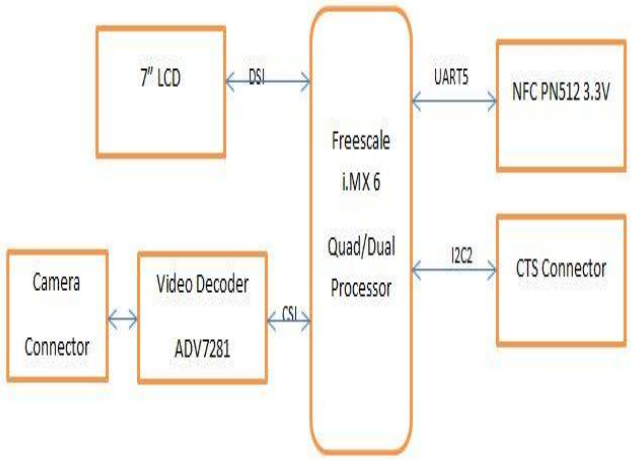

Fig. 1. Block Diagram of Hardware Connected with the Processor

II. HARDWARE SPECIFICATIONS OF DEVICE

Drivers have been developed exclusively for the Linux OS running on ARM CORTEX A9 based Board. To implement the drivers, following hardware has been used. Specifications of the hardware can be found below.

A. LCD Display

LCD Display module is MTF070ICN-PB1 which is 40-pin connector from Microtech Technology Co. Ltd. Important Features of this display module is, that, it supports MIPI interface, resolution 800 x 3(RGB) X 1280, and having 4-Data Lanes which supports bandwidth of around GHz order.

B. Touch screen

Touch Screen (Atmel maXTouch) is interfaced using I2C2 to the processor.

C. Camera

Camera module is OV5642 which is used for the purpose of video recording so that it can be stream on LCD Display. This module gets connected to the Board using camera serial interface.

D. NFC PN512

NFC PN512 is connected to the processor using UART interface. The PN512 transmission module supports the Read/Write mode for ISO/IEC 14443 A/MIFARE and ISO/IEC 14443 B using various transfer speeds and modulation protocols. PN512 NFC frontend supports the following operating modes:

- Reader/Writer mode supporting ISO/IEC 14443A/MIFARE and FeliCa scheme
- Card Operation mode supporting ISO/IEC 14443A/MIFARE and FeliCa scheme
- NFCIP-1 mode The modes support different transfer speeds and modulation schemes.

III. HARDWARE SETUP OF DEVICE WITH i.MX6 PROCESSOR

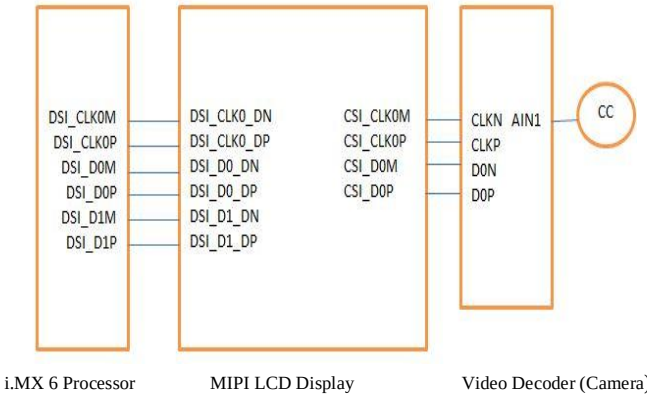

Fig. 2. Schematic Design - i.MX 6 Processor, LCD, Camera

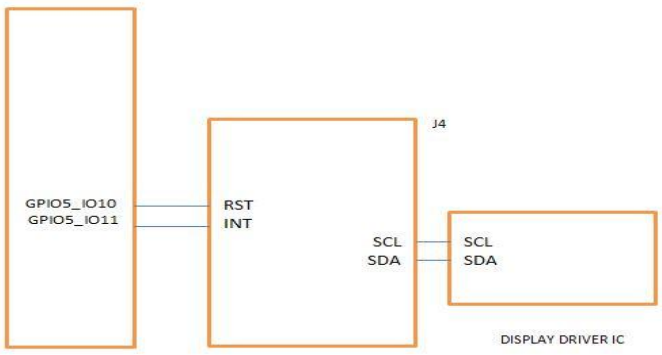

Fig. 3. Schematic Design - i.MX 6 Processor & Capacitive Touch screen

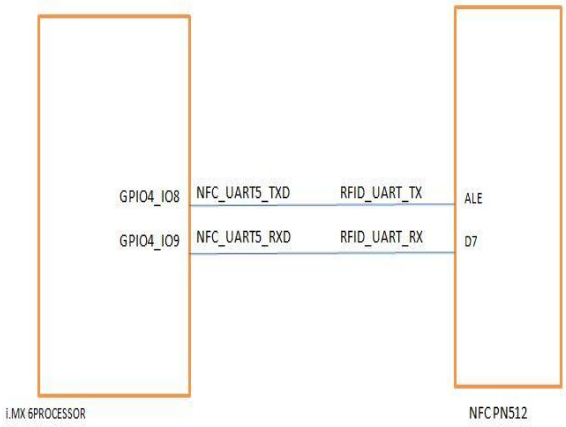

Fig.4 Schematic Design - i.MX 6 Processor & PN512

IV. DRIVER DEVELOPMENT AND IMPLEMENATION

This Section details the driver development procedures followed for each driver separately.

A. LCD Display Driver

Linux LCD Device Driver is in the middle of user program and hardware.

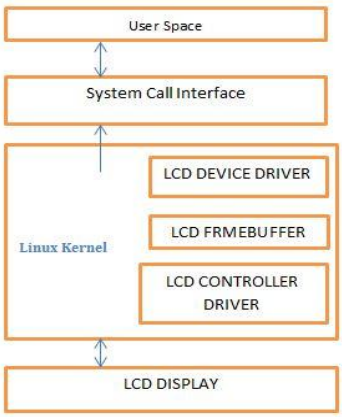

Fig. 5. Display Driver Framework

The driver is responsible for pushing decoded raw data to display interface. Frame buffer driver drive display interface as a block of memory, user program use simple memory control command to read/write data from display device.

B. Touch screen

Capacitive touch has multi touch support and less prone to dust particle. Most capacitive touch controller has capability to support multiple configuration (x & y axis). According to configuration set for the touch one has to scale the inputs received from touch to the area of LCD.

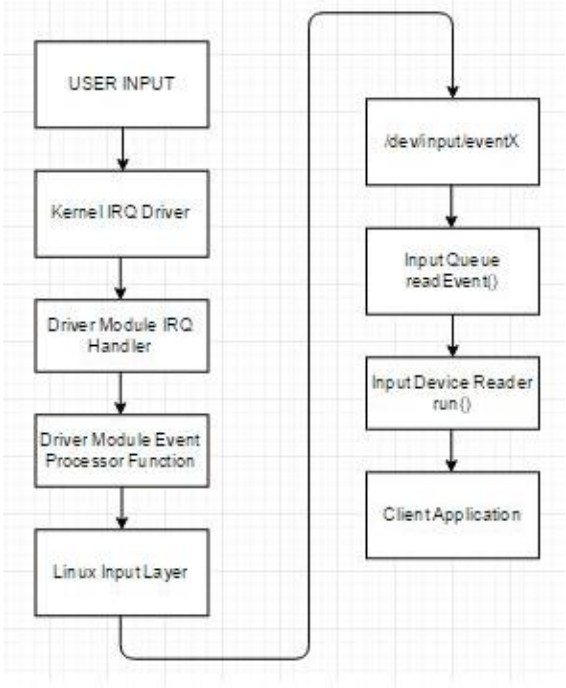

Fig. 6.Capacitive Touch Event Flowchart

C. Camera

Camera subsystem initially provided a unified API between host drivers and sensor drivers. Later camera sensor API has been replaced with the V4L2 standard subdev API.

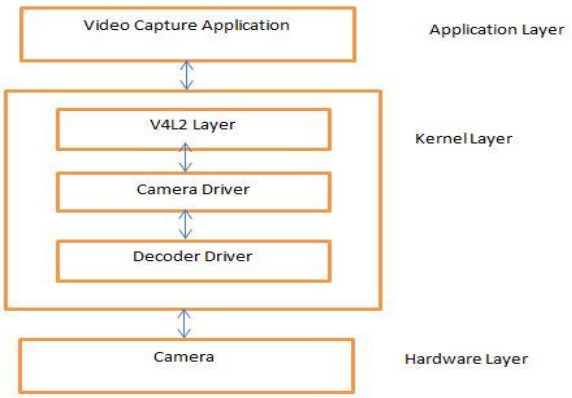

Fig. 7.Camera Driver Framework

The camera host API to the soc-camera core has been preserved. Soc-camera implements a V4L2 interface to the user, currently only the "mmap" method is supported by host drivers. However, the soc-camera core also provides support for the "read" method. The subsystem has been designed so as to support multiple camera host interfaces and multiple cameras per interface, although most applications have only one camera sensor.

D. NFC PN512

NFC subsystem is required to standardize the NFC device drivers development and to create an unified user space interface.

The NFC subsystem is responsible for:

- NFC adapter's management, polling for targets and low-level data exchange.

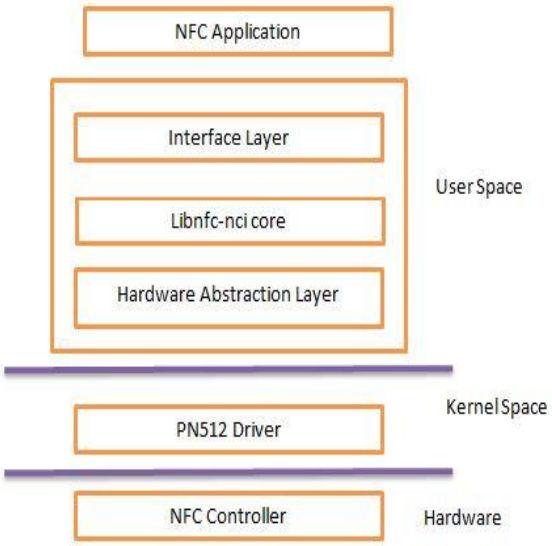

Fig. 8.NFC Driver Framework

The subsystem is divided into different parts. The 'core' is responsible for providing the device driver interface. On the other side, it is also responsible for providing an interface to control operations and low-level data exchange. The control operations are available to user space via generic netlink. The low-level data exchange interface is provided by the new socket family PF_NFC. The NFC_SOCKPROTO_RAW performs raw communication with NFC targets. When registering on the NFC subsystem, the device driver must inform the core of the set of supported NFC protocols and the set of ops callbacks.

The ops callbacks implemented are
* start_poll - setup the device to poll for targets
* stop_poll - stop on progress polling operation
* activate_target - select and initialize one of the targets found
* deactivate_target - deselect and deinitialize the selected target
* data_exchange - send data and receive the response (transceive operation)

The userspace must use PF_NFC sockets to perform any data communication with targets. All NFC sockets use AF_NFC:

To establish a connection with one target, the user must create anNFC_SOCKPROTO_RAW socket and call the 'connect' syscall with the sockaddr_nfcstruct correctly filled. All information comes from NFC_EVENT_TARGETS_FOUNDnetlink event. As a target can support more than one NFC protocol, the user must inform which protocol it wants to use.

V. TESTING AND VALIDATION

For the testing of the drivers the code developed are integrated with the kernel and a number of modifications are required in the kernel as detailed below.

A. *Kernel changes for LCD*

In /KERNEL/arch/arm/mach-mx6/board-mx6q_sabrelite.c

- Add Pin mux for LCD enable gpio, Dispay enable gpio ,lcd data lines.
- Add Driver file for LCD

Enable Driver in menu config

- Device Drivers-> Graphics support-> OMAP2+ Display Subsystem support-> OMAP2/3 Display Device Drivers -> Lcd driver->y

B. *Kernel Changes for Touch screen*

- Add pin mux for irqgpio,resetgpio in board file
- Register I2C1 touchscreen with address of chip
- Add code for touchscreeen
- Enable driver in menuconfig - Device driver ->input driver support ->Touch screen ->Driver-> y

For Touchscreen Calibration

- Add calibration points In /etc/X11/xorg.conf.d/99-calibration.conf file in files ystem .
- Section "InputClass"
- Identifier "calibration"
- MatchProduct "Touchscreen name"

C. Kernel Changes for Camera

- obj -$(CONFIG_VIDEO_OV5642) +=OV5642.O
- config VIDEO_OV5642
- tristate "OV5642 CMOS IMAGE SENSOR"

D. Kernel Changes for NFC PN512

● In /KERNEL/arch/arm/mach-mx6/board-mx6q_sabrelite.c
Add UART support
{
/* UART */
{
.modalias = "uartdev",
.max_speed_hz = 48000000, //48 Mbps
.bus_num = 2, //uart
.chip_select = 0,
.mode = UART_MODE_1,
},
● Register uart bus
● InMenuconfig
Device Drivers ---> [*] UART support ---> (put * for) User mode UART device driver support
● Added Driver code for pn512 drivers/net/nfc
● Inmenuconfig Networking Support-> NFC Subsystem support-> Near Field Communication (NFC) Devices->M pn512 driver->y

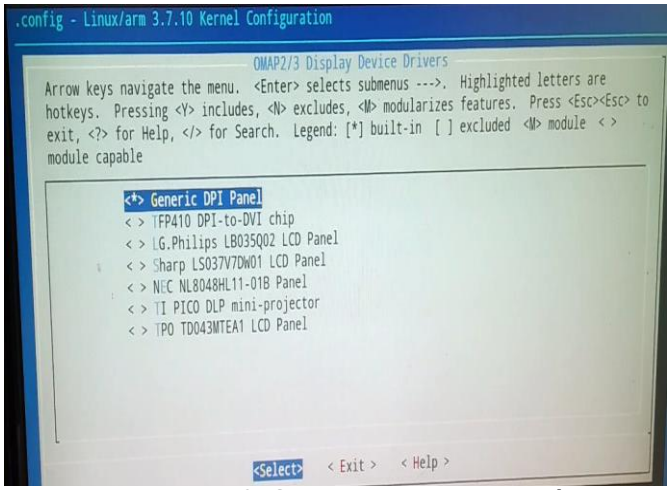

Fig. 9.LCD Driver Integration into Kernel

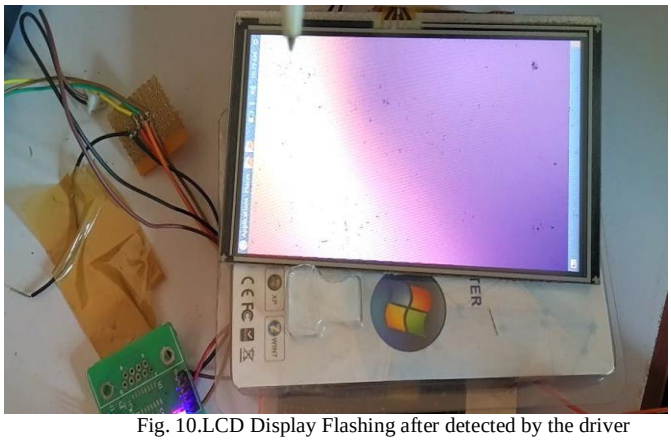

Fig. 10.LCD Display Flashing after detected by the driver

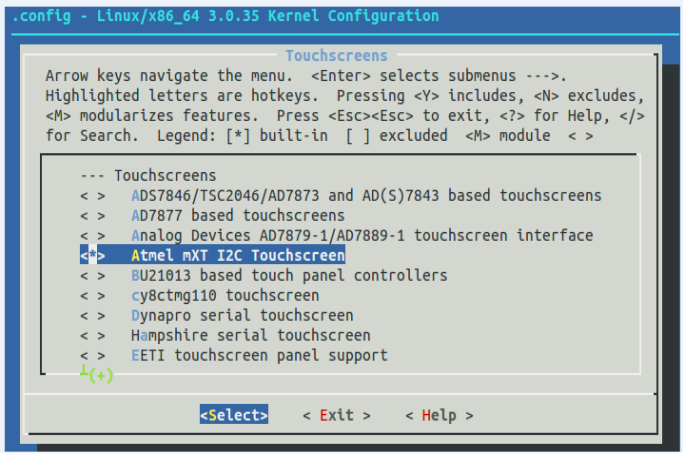

Fig. 11.Touch screen Driver Integration into Kernel

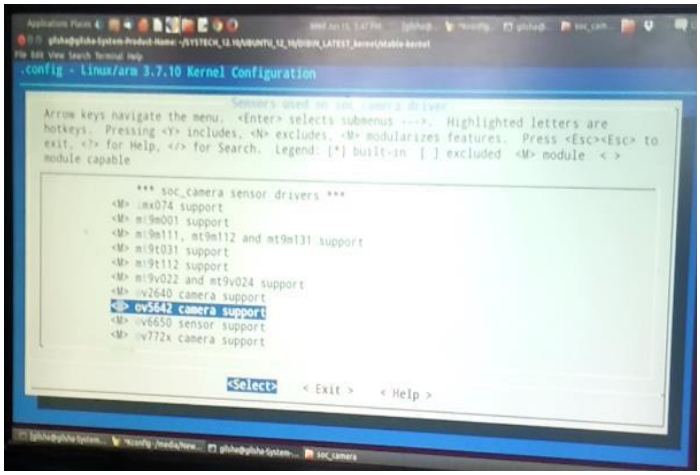

Fig. 12.Camera Driver Integration into Kernel

```
root@salher-Lenovo-G50-80:/home/salher/Desktop/nfc-drivers/libnfc/linux_libnfc-n
ci# ./nfcDemoApp
Missing argument

COMMAND:
        poll    Polling mode      e.g. <nfcDemoApp poll >
        write   Write tag         e.g. <nfcDemoApp write --type=Text -l en -r "Te
st">
        share   Tag Emulation Mode       e.g. <nfcDemoApp share -t URI -u http:/
/www.nxp.com>
        push    Push to device    e.g. <nfcDemoApp push -t URI -u http://www.nxp.
com>
                                  e.g. <nfcDemoApp push --type=mime -m "applicati
on/vnd.bluetooth.ep.oob" -d "2200AC597405AF1C0E0947616C617879204E6F74652033040D0
C024005031E110B11">

Help Options:
-h, --help                        Show help options
```

Fig. 13.Linux based Application for NFC Read/write purpose

VI. CONCLUSION & FUTURE SCOPE

Device Drivers have been developed for LCD, Touch screen, Camera and NFC specifically for ARM CORTEX A9 based embedded system running Linux OS. All the drivers have been then added as a part of kernel so that whenever such devices found, it will automatically get detected. Each of the four drivers have been first tested on development board and later tested with custom board. There is always a scope of improvement in the driver coding.

So, as a part of future scope, driver developer can be modified the code as and when required because of modification in the hardware input and output signals. And suppose, there is change in the kernel and a set of system calls/API have been added to ease the operation of Linux OS, then also drivers can be optimized.

VII. REFERENCES

[1] Alessandro Rubini& Jonathan Corbet, Linux Device Drivers, 2nd edition , O'Reilly , pp 1-715-18, 22-23

[2] Richard Petersen , The complete Reference Linux, 6th edition , Mc-Graw Hill Osborne , chapter 1-2

[3] Karim Yaghmour, Jon Masters, et.al, Building Embedded Linux Systems , 2nd edition,O'Reilly, chapter 1-6

[4] Jia-Ju Bai, Hu-Qiu Liu, Yu-Ping Wang, Shi-Min Hu, "Complete Runtime Tracing for Device Drivers Based on LLVM" , 39th IEEE Annual International Computers, Software & Application Conference 2015

[5] SeehwanYoo, Young-pil Kim, "Test-driven Development of Consumer Electronics Device Drivers: A User-level Device Driver Approach", IEEE International Conference on Consumer Electronics ,2015

[6] Chao Ma, Peng Zhao, Shi-min Hu, "Serial Driver: Improving the Reliability of Device Drivers through Serialization", IEEE Transactions on Consumer Electronics, Vol. 58, No. 3, August 2012

[7] Yu-Jung Huang, Chih-Feng Liu,, Shao-Pin Chang, et.al "Design of LCD Driver IP for SOC Applications", 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits(AP-ASIC2004)/Aug. 4-5, 2004

[8] Min Zhang, Jin-guang Sun, Shi Wang , "Research and Implementation of the CMOS Camera Device Driver Based on S3C244O" , 2010 International Conference on Intelligent Computation Technology and Automation

