



삼성 Exynos-4210 듀얼코어 프로세서로
배우는 안드로이드 플랫폼 학습
www.hardkernel.com

Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 0. 시작하기에 앞서
- 1. 기초 과정: 안드로이드 플랫폼을 빌드하고 수정하는 방법
- 2. **중급 과정: 안드로이드 기반의 임베디드 시스템 구성 및 API활용**
- 3. 하드웨어 확장 및 활용
- 4. 부록 Google Open Accessory Library

www.hardkernel.com

시작하기에 앞서(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 본 과정은 ODROID-A4를 사용하여 안드로이드 시스템 개발에서부터 앱 개발까지의 과정을 포함하고 있다.
- ODROID-A4는 안드로이드 OS가 설치되어진 8GB T-Flash카드와 시스템을 디버깅할 수 있는 디버그보드, 컴퓨터와 통신할 수 있는 TTA20 to USB케이블로 구성되어 있다.



- 안드로이드 시스템을 개발하기 위해서는 USB to Serial 변환장치가 별도로 필요하다.
- 옆의 사진은 전 과정을 수행하기 위한 개발환경이다.
기초과정과 중급과정에서는 Debug Board를 바로 ODROID-A4에 연결하여 사용하면 된다.



시작하기에 앞서(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 중급 과정은 ODROID-A4의 각종 하드웨어 장치들(CPU, LCD, 터치, 오디오, 카메라, 센서, 배터리, WiFi, Bluetooth)이 회로도->앱으로 어떻게 연동되는지를 설명하고 있다.
- 이 장을 모두 마스터했다면 어디가서 안드로이드 플랫폼을 좀 할 줄 안다고 하라.
- 궁금한 내용은 게시판에 질문을 남기자. 단, 다른 사람이 한 질문 중에 내가 아는 답이 있으면 꼭 답을 달아주자. 혹시 아는가? 답을 잘 달아주는 사람은 하드커널에서 선물이라도 보내줄지.

http://com.odroid.com/sigong/nf_board/nboard.php?brd_id=odroidaf

1. 중급 과정 : 안드로이드 기반의 임베디드 시스템 구성 및 API 활용

www.hardkernel.com

Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Architecture of Exynos4210 CPU, RAM and Flash storage
- LCD/backlight driver
- Keypad 드라이버의 구조
- Touchscreen 드라이버의 구조
- Audio input/output driver
- Camera driver
- 센서 디바이스 드라이버의 구조
- Battery gauge and charger drivers
- WiFi device driver and HAL
- Bluetooth device driver and HAL

www.hardkernel.com



Architecture of Exynos4210 CPU, RAM and Flash storage

www.hardkernel.com

Agenda

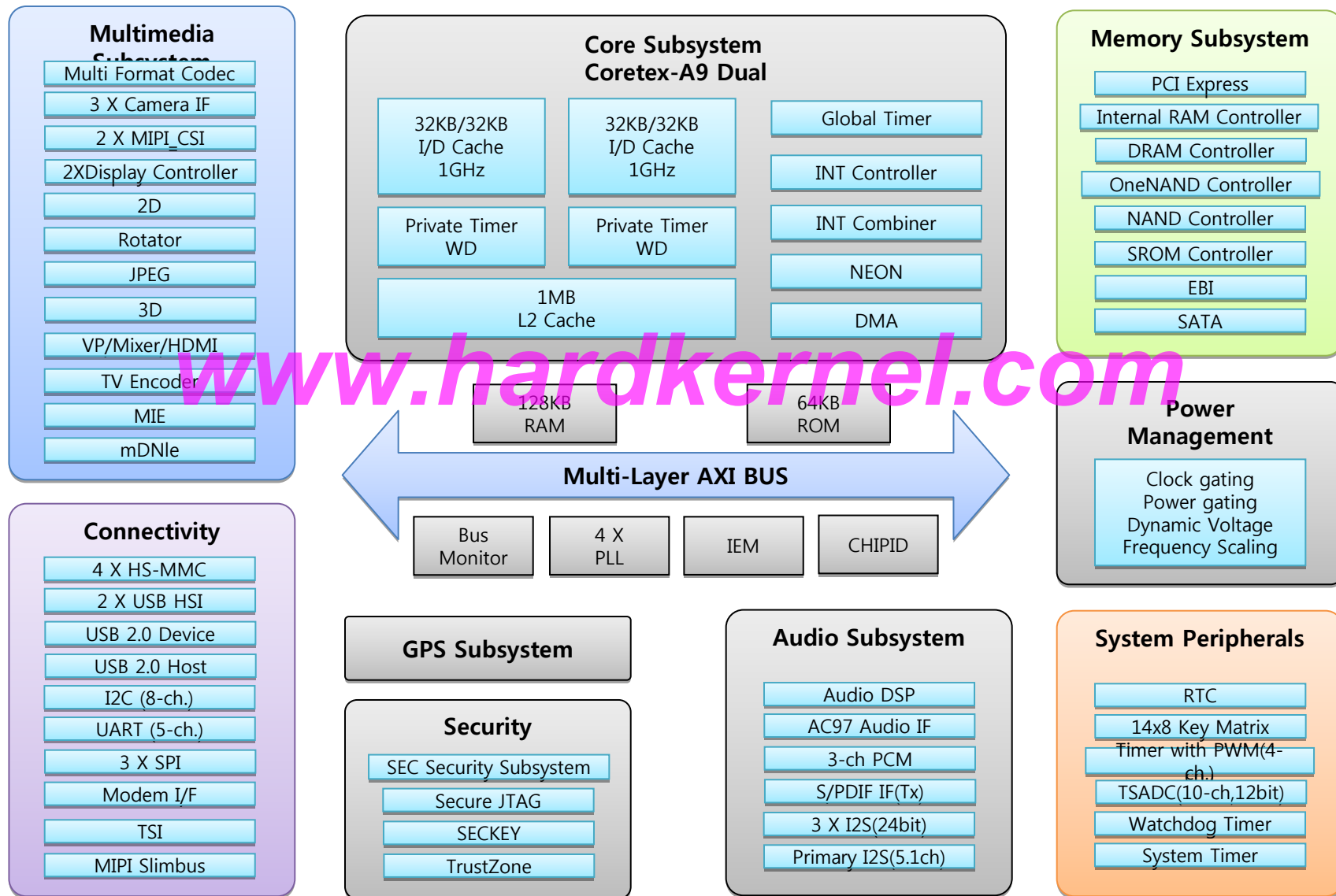
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Exynos4210 Architecture
- Exynos4210 Memory Map
- ODROID-A4 메모리 구성
- ODROID-A4 Low level init Code
- ODROID-A4 boot device(T-flash) 초기화
- DVFS, CPU-Hotplug 드라이버
- ODROID-A4 CPU Max freq. 조정
- Lab/Exam
 - T-Flash 파티션 조정(u-boot)
 - CPU0/CPU1 Clock의 임의 제어 (Kernel)
 - CPU1 On/Off

www.hardkernel.com

Exynos4210 Architecture

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



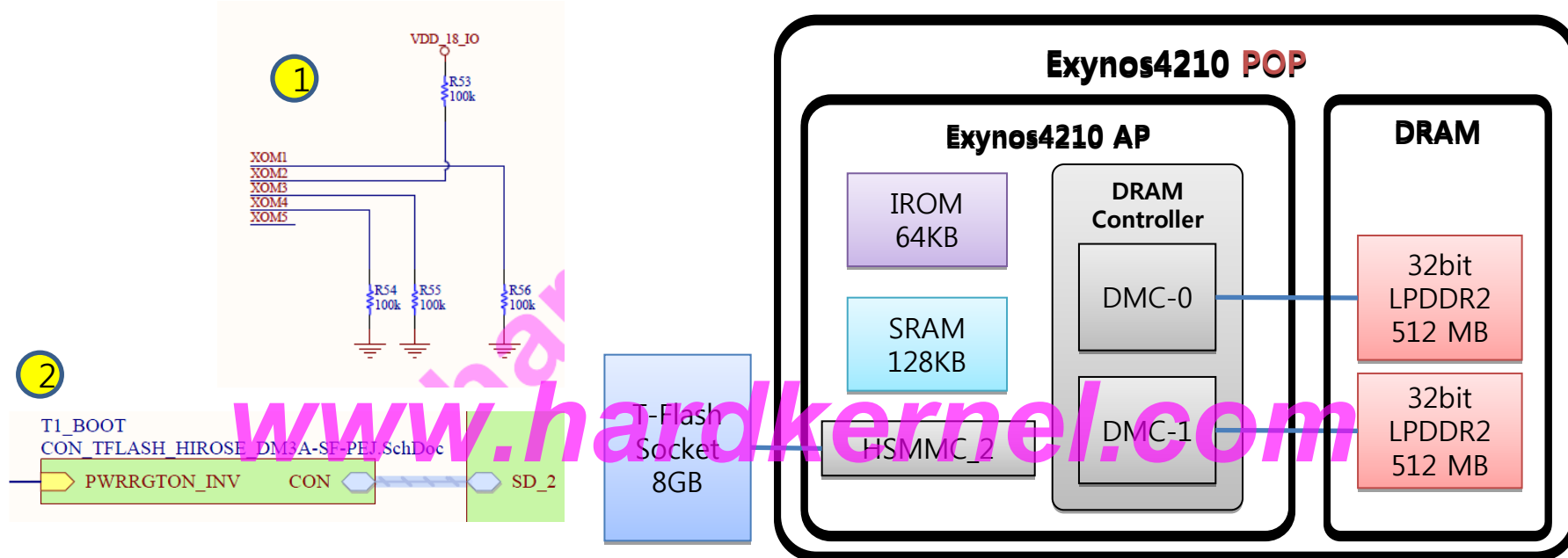
Exynos4210 Memory Map

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Base Address	Limit Address	Size	Description
0x0000_0000	0x0001_0000	64 KB	iROM
0x0200_0000	0x0201_0000	64 KB	iROM (mirror of 0x0 – 0x10000)
0x0202_0000	0x0204_0000	128 KB	iRAM
0x0300_0000	0x0303_0000	192 KB	SRP-RAM or general purpose
0x0381_0000	0x0383_0000	-	AudioSS's SFR region
0x0400_0000	0x0402_0000	128 KB	SROMC's Bank0
0x0500_0000	0x0502_0000	128 KB	SROMC's Bank1
0x0600_0000	0x0602_0000	128 KB	SROMC's Bank2
0x0700_0000	0x0702_0000	128 KB	SROMC's Bank3
0x0800_0000	0x0C00_0000	64 MB	PCI-e memory
0x0C00_0000	0x0C40_0000	-	OneNANDC's channel 0/1
0x0C60_0000	0x0CD0_0000	-	OneNANDC's SFR region
0x0CE0_0000	0x0D00_0000	-	NFCON's SFR region
0x1000_0000	0x1400_0000	-	SFR region
0x4000_0000	0xA000_0000	1.5 GB	DMC-0's memory
0xA000_0000	0x0000_0000	1.5 GB	DMC-1's memory

ODROID-A4 메모리 구성

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



- IROM : boot code가 내장된 Exynos4210 내부 ROM
- SRAM : IROM의 boot code에서 초기화하고 boot device 의 BL1이 로딩되어 실행되는 RAM
- DRAM : Exynos4210 AP에 POP(Package On Package) 형태로 연결되어 있으며 ODROID-A4에는 32bit LPDDR2 1GB(512MB+512MB)가 POP되어진 package를 사용하고 있다.
- T-Flash : ODROID-A4는 OM pin이 SD 2번 채널로 부팅하도록 설정되어 있고, SD 2번 채널의 T-Flash 소켓을 통해 8GB 가 제공된다.

1. A4_BASE_v02_Schematics.pdf 파일 page-4 boot device option 참조.

2. A4_BASE_v02_Schematics.pdf 파일 page-1 T-Flash socket

ODROID-A4 low level init Code(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ U-boot – board/samsung/smdkv310/ lowlevel_init.S : BL1 init process

```
/* PS_HOLD pin set to high */
.....
bl pmic_init
.....
bl uart_asm_init
.....
bl read_om
.....
bl system_clock_init
.....
bl mem_ctrl_asm_init
.....
b load_uboot
```

PMU의 XPSHOLD High로 Set (Power Key가 Low가 되어도 PMIC가 꺼지지 않도록 하기 위함.)

MAX8997 PMIC 초기화 함수를 호출.

Debug serial로 사용하는 UART port를 초기화 하는 함수 UART2

Boot device option을 읽는 함수로 odroid-a4는 SD/MMC boot로 설정되어 있다.

Clock source, Clock rate, Clock gating 등의 system PLL초기화 함수

DRAM interface 초기화 함수

BL2를 추가화된 DRAM 번지로 로딩하고 실행하는 함수

➤ U-boot – board/samsung/smdkv310/ lowlevel_init.S : Set PSHOLD

```
ldr      r0, =0x1002330C
ldr      r1, [r0]
orr      r1, r1, #0x5300
str      r1, [r0]
```

/* Exynos4210 PMU 레지스터 PS_HOLD_CONTROL */

/* PSHOLD Output High */

➤ U-boot – arch/arm/cpu/armv7/exynos/pmic_hkdk4210.c pmic_init() 함수

- I2C 포트를 통해 연결된 MAX8997 PMIC를 초기화하는 함수이다.
- Default On 이 아닌 Buck, LDO 등의 출력을 설정할 수 있다.
- System PLL을 설정하기 전 vdd_arm, vdd_int, vdd_mif 등의 전압을 조정하는 코드들이 있으며, 데스크탑 PC의 bios 에서 오버클럭을 할 때 전압을 조정하는 것과 유사하다고 볼 수 있다.
- Odroid-A4에서는 Reset 버튼의 Time out 관련 설정을 default 7초에서 1초로 변경하는 코드를 추가함.

ODROID-A4 low level init Code(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ U-boot – board/samsung/smdkv310/ lowlevel_init.S : uart_asm_init , read_om 함수

- 회로설계를 하고 하드웨어가 나오면 가장먼저 디버깅을 해야 할 부분이라 하겠다.
- UART 포트의 pin-function, clock source, clock rate, Baud-rate 등을 설정한다.
- read_om 함수는 boot device option을 읽는 함수이다.

➤ U-boot – board/samsung/smdkv310/lowlevel_init.S : system_clock_init 함수

- System PLL 및 boot device clock을 초기화 하는 함수.
- u-boot include/configs/smdkv310.h 파일에서 predefine된 값으로 PLL 수정 가능.
- u-boot board/samsung/smdkv310/smdkv310_val.h 파일에 설정된 Clock에 해당하는 PLL값이 define 되어 있다.
- lowlevel_init.S 파일의 system_clock_init 함수는 SFR 레지스터에 define 된 PLL값을 쓰는 함수이다.
- ODROID-A4는 APLL : 1GHz, DMC bus clock: 400Mhz 로 초기화 하고 있다.
- kernel에서 cpu-freq, bus-freq scaling 드라이버가 enable 되지 않으면 현재의 설정을 유지한다.
- board/samsung/smdkv310/lowlevel_init.S line 317 : PLL Set start

➤ U-boot – board/samsung/smdkv310/mem_setup.S : mem_ctrl_asm_init 함수

- DRAM Physical 인터페이스를 초기화 하는 함수
- Exynos4210 POP 타입이 아닌 경우, 구성된 메모리에 따라 디버깅 및 수정이 필요한 부분이다.
- ODROID-A4에는 32bit 512MB LPDDR2 메모리 2개가 POP된 타입이며 DMC0, DMC1 모두 400MHz 타이밍으로 초기화한다.
- mem_ctrl_asm_init 함수는 Physical timing 설정 함수이고, Logical address mapping은 다음 페이지 참조.
- board/samsung/smdkv310/mem_setup.S line 127 DMC0 – LPDDR2 init sequence start
- board/samsung/smdkv310/mem_setup.S line 249 DMC1 – LPDDR2 init sequence start

ODROID-A4 low level init Code(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

```
COM14@Serial - Xshell 4 (Free fo...
EXYNOS4 # bdfinfo
arch_number = 0x00000B16
boot_params = 0x40000100
DRAM bank
-> start = 0x40000000
-> size = 0x10000000
DRAM bank
-> start = 0x50000000
-> size = 0x10000000
DRAM bank
-> start = 0x60000000
-> size = 0x10000000
DRAM bank
-> start = 0x70000000
-> size = 0x10000000
ethaddr = 08:40:5c:28:0a:5b
ip_addr = 192.168.0.20
baudrate = 0 bps
TLB addr = 0x3FFF0000
relocaddr = 0xC3E00000
reloc off = 0x00000000
irq_sp = 0xC3CFBF38
sp start = 0xC3CFBF30
FB base = 0x00000000
EXYNOS4 #
```

➤ Include/configs/smdkv310.h

```
#define CONFIG_SYS_SDRAM_BASE        0x40000000

#define SDRAM_BANK_SIZE              0x10000000 /* 256 MB */
#define PHYS_SDRAM_1                 CONFIG_SYS_SDRAM_BASE
#define PHYS_SDRAM_1_SIZE            SDRAM_BANK_SIZE
.....
```

➤ Board/samsung/smdkv310/smdkv310.c

```
void dram_init_banksize(void)
{
.....
printf("EVT1 ");
if(dmc_density == 6){
printf("POP_BWn");
nr_dram_banks = 4;
gd->bd->bi_dram[0].start = PHYS_SDRAM_1;
gd->bd->bi_dram[0].size = PHYS_SDRAM_1_SIZE;
.....
}
```

BANK	Start Address	Size
BANK #0	0x40000000	256 MB
BANK #1	0x50000000	256 MB
BANK #2	0x60000000	256 MB
BANK #3	0x70000000	256 MB

- u-boot 에서 bdfinfo 명령어로 확인하면 왼쪽의 테이블과 같이 초기화 되었음을 확인 할 수 있다.
- Kernel MACHINE에서 fixup() 함수가 없으면 u-boot의 meminfo 가 boot param으로 넘겨지고 커널의 시스템메모리로 alloc 되게 된다.
- 위의 소스는 ODROID-A4의 Logical Address mapping 코드이고 ODROID-A4는 u-boot에서 초기화된 meminfo를 커널에서 사용하고 있다.

ODROID-A4 boot device(T-flash) 초기화

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

```

COM14@Serial - Xshell 4 (Free for Home/School)
CPU: S5PC210 [Samsung SOC on SMP Platform Base on ARM CortexA9]
APLL = 1000MHz, MPPLL = 800MHz
EVT1 POP_B
DRAM: 1 GiB

Checking Boot Mode ... SDMMC
MMC Device 0: 7647 MB
MMC Device 1 not found
*** Warning - using default environment

Net: smc911x: Invalid chip endian 0x00000000
No ethernet found.
Hit any key to stop autoboot: 0
EXYNOS4 # fdisk -c 0
fdisk is completed

partition # size(MB) block start # block count partition_id
1 5827 3580200 11934000 0x0C
2 515 137700 1055700 0x83
3 1030 1193400 2111400 0x83
4 134 3304800 275400 0x83
EXYNOS4 #
  
```

Partition Type	Start Sector	Partition Size	Partition Name
Normal Partition			Media (VFAT)
		128MB	CACHE (EXT4)
		1GB	DATA (EXT4)
		512MB	SYSTEM (EXT4)
Low-Level Partition	50561		Reserved
	17793	16MB	RAMDISK
	1409	8MB	KERNEL
	1507	16KB	U-BOOT Env
	33	512KB	U-BOOT
	1	16KB	U-BOOT BL1
	0	512B	MBR

- ODROID-A4 u-boot에서 fdisk 명령어로 파티션을 나누는 과정이다.
- fdisk 명령어로 나누어지는 파티션은 오른쪽 T-Flash 섹터맵의 Normal 파티션에 해당하는 영역으로 안드로이드 init.odroida4.rc 에서 /system, /data, /cache 로 마운트 되는 것을 확인할 수 있다.
- VFAT 영역은 user 영역으로 미디어 파일 및 이동식 디스크로 사용할 수 있는 영역이다. 안드로이드 vold.fstab 파일 참조
- 왼쪽 그림의 파티션 정보에서 VFAT 파티션이 1번임에도 block start # 를 보면 물리적인 섹터의 뒤쪽에 있음을 알 수 있다.
이유는 Windows XP 이하의 OS에서 첫번째 파티션이 ext4 파티션인 경우 이동식디스크 연결이 되지 않기 때문이다.
- .odt 파일은 섹터 0 ~ VFAT 파티션 start 섹터 + 약 1MB 정도의 사이즈로 Dump한 파일이다.

DVFS, CPU-Hotplug 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ DVFS?, CPU-Hotplug?

- Dynamic Voltage and Frequency Scaling : Power management 를 위해 시스템 로드 에 따라 CPU의 동작주파수를 변화시키고 동작 주파수에 따른 CPU 전압을 제어하는 기술이다.
- CPU-Hotplug : 멀티코어가 임베디드 시스템에 적용되면서 역시 Power management를 위해 시스템 로드 에 따라 메인 CPU를 제외한 나머지 CPU를 On/Off 할 수 있도록 하는 기술이다. DVFS의 ondemand governor와
- Mobile 플랫폼인 ODROID-A4는 저전력으로, desk-top 플랫폼인 ODROID-PC는 퍼포먼스로 설정 되어있다.
- Exynos4210 에서는 현재 CPU0, CPU1 독립적인 주파수 설정은 동작하지 않는다. 즉 CPU0, CPU1은 항상 같은 주파수로 동작하도록 되어있다.

➤ CPU_FREQ governor

- performance – cpu가 사용 가능한 최고 주파수를 사용한다. (ODROID-PC 설정)
- powersave – cpu가 사용가능한 최저 주파수를 사용한다.
- userspace - 사용자가 임의로 주파수를 제어할 수 있는 방식이다.
- ondemand – 평상시 최저 주파수에 있다가 CPU를 요구하는 작업이 있는 경우 최고 주파수로 올라가는 방식이다.
주파수가 내려갈 때도 최저 주파수로 바로 내려간다.
- conservative – 평상시 최저 주파수에 있다가 CPU의 부하에 따라 단계적으로 클럭이 올라가는 방식이다.
부하에 따라 주파수가 내려갈 때도 단계적으로 내려간다.
- interactive – 주파수가 올라갈 때는 ondemand, 내려갈 때는 conservative 의 방식

➤ CPU Hotplug Policy

- stand alone – 최저 주파수에서의 부하가 설정된 값보다 작은 경우 CPU1 Off
- Intergrated DVFS – DVFS 모니터링을 통해 낮은 주파수 상태를 유지하는 경우 CPU1 Off
- nr_running – run 프로세스의 개수를 모니터링하고 설정된 수보다 작은 상태를 유지하는 경우 CPU1 Off
- DVFS nr_running – DVFS 주파수 및 run 프로세스 개수를 동시에 모니터링.

ODROID-A4 CPU Max freq. 조정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Exynos4210 DVFS driver (Kernel)

- arch/arm/mach-exynos/cpufreq.c
- arch/arm/mach-exynos/cpufreq-4210.c

➤ Exynos4210 CPU Hotplug driver (Kernel)

- arch/arm/mach-exynos/hotplug.c
- arch/arm/mach-exynos/stand-hotplug.c
- arch/arm/mach-exynos/dvfs-hotplug.c
- arch/arm/mach-exynos/dynamic-nr_running-hotplug.c
- arch/arm/mach-exynos/dynamic-dvfs-nr_running-hotplug.c

➤ ODROID-A4 CPU Max Freq. 조정하기

- arch/arm/mach-exynos/cpufreq-4210.c line 323

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ T-Flash 파티션 조정(u-boot)

❖ Low Level Partition

1

include/configs/smdkv310.h

```
#define CONFIG_ENV_SIZE 0x4000 // 16KB
```

arch/arm/include/asm/arch-exynos/movi_partition.h

```
#define PART_SIZE_BL1 (16 * 1024)
```

```
#define PART_SIZE_UBOOT (512 * 1024)
```

```
#define PART_SIZE_KERNEL (8 * 1024 * 1024)
```

```
#define PART_SIZE_ROOTFS (16 * 1024 * 1024)
```

arch/arm/cpu/armv7/exynos/movi_partition.c

```
int init_raw_area_table(block_dev_desc_t * dev_desc, int location)
```

```
{
    .....
    image[0].start_blk = location;
    .....
}
```

❖ Normal Partition

2

common/cmd_mmc_fdisk.c

```
#define SYSTEM_PART_SIZE (512*1024*1024)
```

```
#define USER_DATA_PART_SIZE (1024*1024*1024)
```

```
#define CACHE_PART_SIZE (128*1024*1024)
```

```
.....
```

```
int make_mmc_partition() 함수
```

```
{
    .....
    get_SDInfo(total_block_count, &sdInfo);
    .....
}
```

- Low Level partition은 파일시스템이 없이 섹터단위로 기록하는 영역이고 1과 같이 정의되어 있다.
- BL1 사이즈는 Exynos4210 IROM 코드에 정의된 사이즈로 수정할 경우 부팅할 수 없다.
- Low Level partition 사이즈를 변경한 경우 make distclean, smdkv310_config 과정이 반드시 필요하다.
- Normal partition은 2와 같이 정의 되어 있고, VFAT 영역은 세개(system, data, cache)의 파티션을 나누고 나머지 부분을 할당한다.
- Normal partition의 경우 u-boot 프롬프트에서 fdisk 명령어로 파티션을 새로 정의할 때 적용된다.

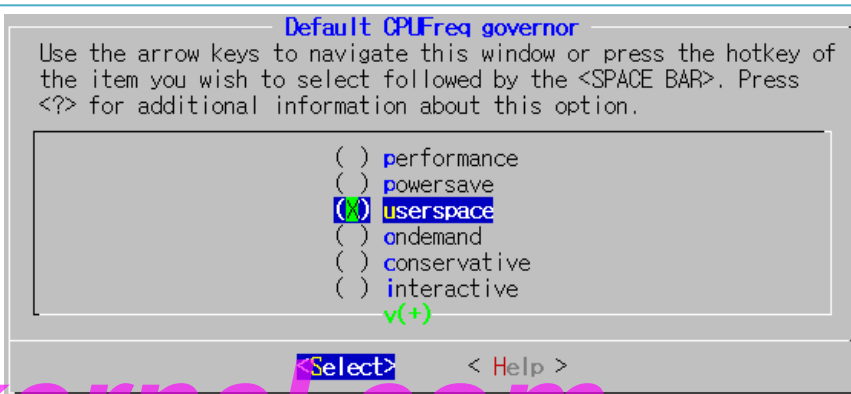
Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ CPU0/CPU1 Clock의 임의 제어 (Kernel)

- kernel root directory
- make odroid_a4_defconfig
- make menuconfig
 - > CPU Power Management --->
 - > CPU Frequency scaling --->
 - > CPU Frequency scaling --->
 - > Default CPUFreq governor (ondemand) --->
 - userspace 선택
- make zImage

1



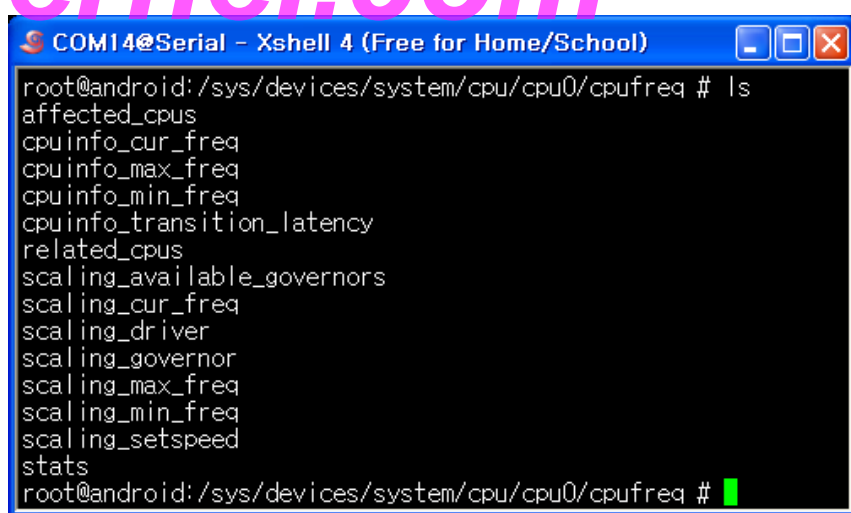
- Target board boot
- debug serial 에서 kernel 프롬프트 확인.
- cd /sys/devices/system/cpu/cpu0/cpufreq/
- ls 명령어로 확인하면 오른쪽 그림과 같다.
- 먼저 현재 CPU0의 주파수를 확인해 본다.


```
cat cpuinfo_cur_freq
```
- cpu0 주파수를 200MHz로 변경해 본다.


```
echo 200000 > scaling_setspeed
```
- cpu0의 주파수를 변경하고 cpu1의 주파수를 확인해 본다.


```
cat ../../cpu1/cpufreq/cpuinfo_cur_freq
```
- cpu1의 주파수를 변경하고 cpu0의 주파수를 확인해 본다.
- 주파수 변화에 따른 전류 소모량을 측정해 본다.

2



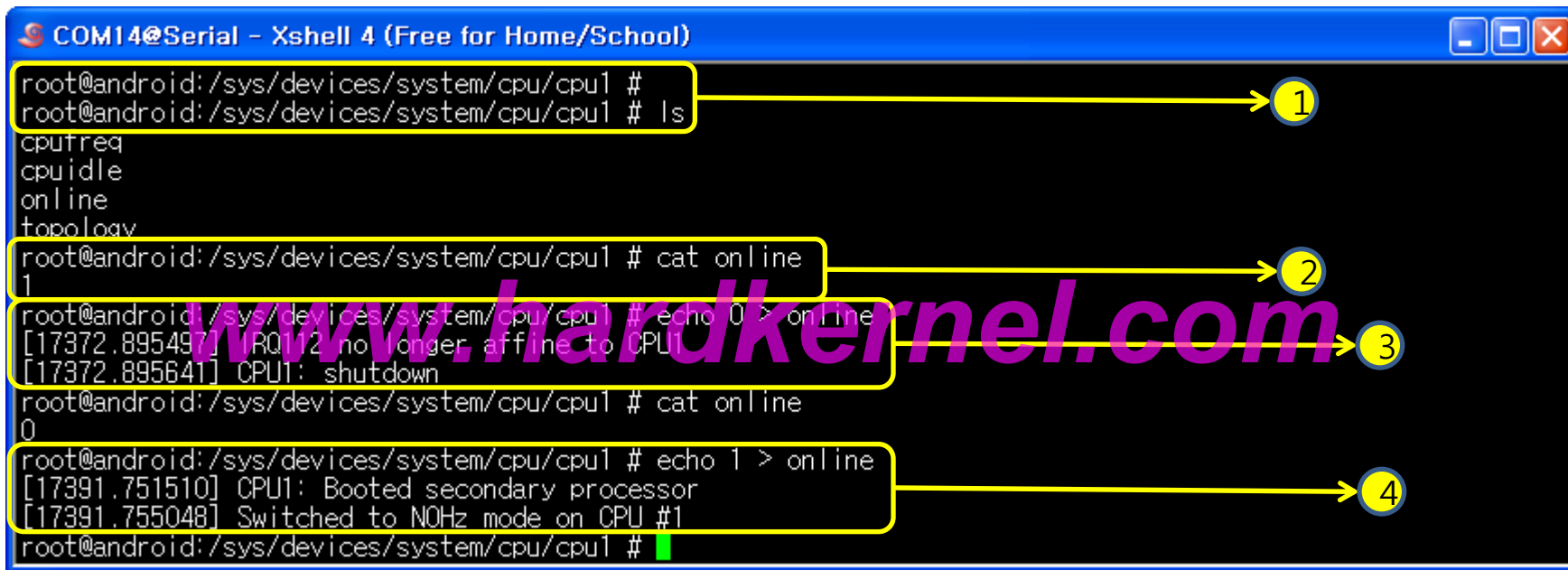
➤ 1과 같은 과정으로 커널을 컴파일하고 fastboot 를 통해 수정된 커널을 업데이트 한다. (기초과정 참조)

➤ 2의 과정으로 Target Board에서 주파수를 변경해 보고 주파수에 따른 전류소모량의 상관관계를 이해한다.

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ CPU1 On/Off



```
COM14@Serial - Xshell 4 (Free for Home/School)
root@android:/sys/devices/system/cpu/cpu1 #
root@android:/sys/devices/system/cpu/cpu1 # ls
cpufreq
cpuidle
online
topology
root@android:/sys/devices/system/cpu/cpu1 # cat online
1
root@android:/sys/devices/system/cpu/cpu1 # echo 0 > online
[17372.895497] IRQ112 no longer affine to CPU1
[17372.895641] CPU1: shutdown
root@android:/sys/devices/system/cpu/cpu1 # cat online
0
root@android:/sys/devices/system/cpu/cpu1 # echo 1 > online
[17391.751510] CPU1: Booted secondary processor
[17391.755048] Switched to N0Hz mode on CPU #1
root@android:/sys/devices/system/cpu/cpu1 #
```

1: `ls` command to list files in `/sys/devices/system/cpu/cpu1`

2: `cat online` command to check current CPU1 state

3: `echo 0 > online` command to shutdown CPU1

4: `echo 1 > online` command to boot CPU1

- DVFS를 userspace governor로 수정한 커널로 Target board boot on
- debug serial 에서 kernel 프롬프트 확인.
- `cd /sys/devices/system/cpu/cpu1`
- cpu1의 현재상태를 확인한다.
- cpu1을 shutdown 하고 정상적으로 shutdown 됐는지를 확인한다.
- cpu1을 다시 부팅한다. 이때 cpu1의 주파수는 cpu0의 주파수와 동일하게 설정된다.
- DVFS가 ondemand governor인 경우 꺼진 cpu1을 켜도 부하가 없으면 곧바로 shutdown 된다.



LCD/backlight driver

www.hardkernel.com

Agenda

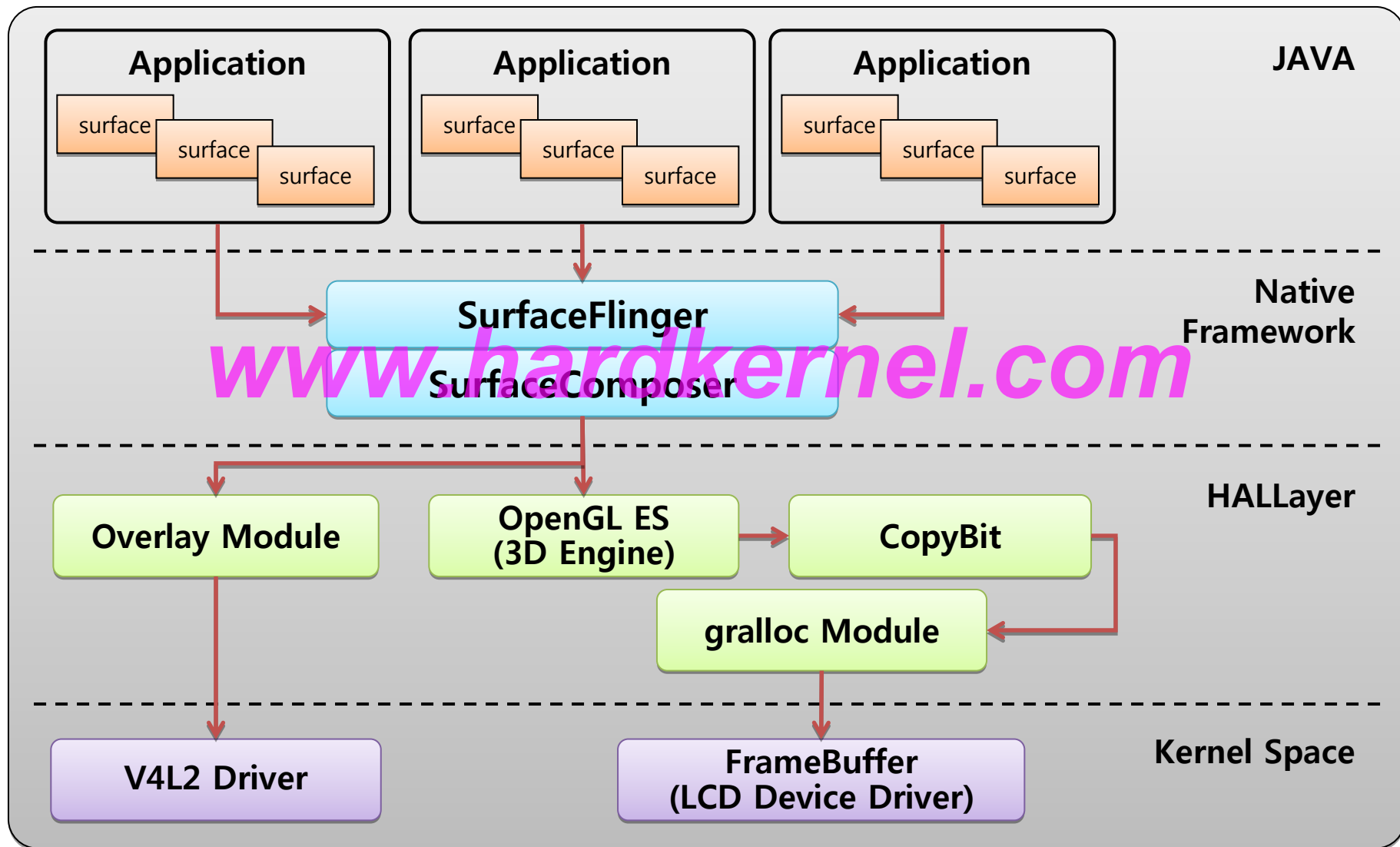
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 안드로이드 framework 개요
- Exynos4210 디스플레이 서브시스템
- ODROID-A4 디스플레이 구성
- ODROID-A4 LCD(LMS397KF04) Spec.
- LCD 드라이버
- PWM Backlight 드라이버
- Lab/Exam
 - Kernel Splash Screen 변경하기(RGB display, Linux LOGO)
 - Backlight 제어 API

www.hardkernel.com

안드로이드 framework 개요

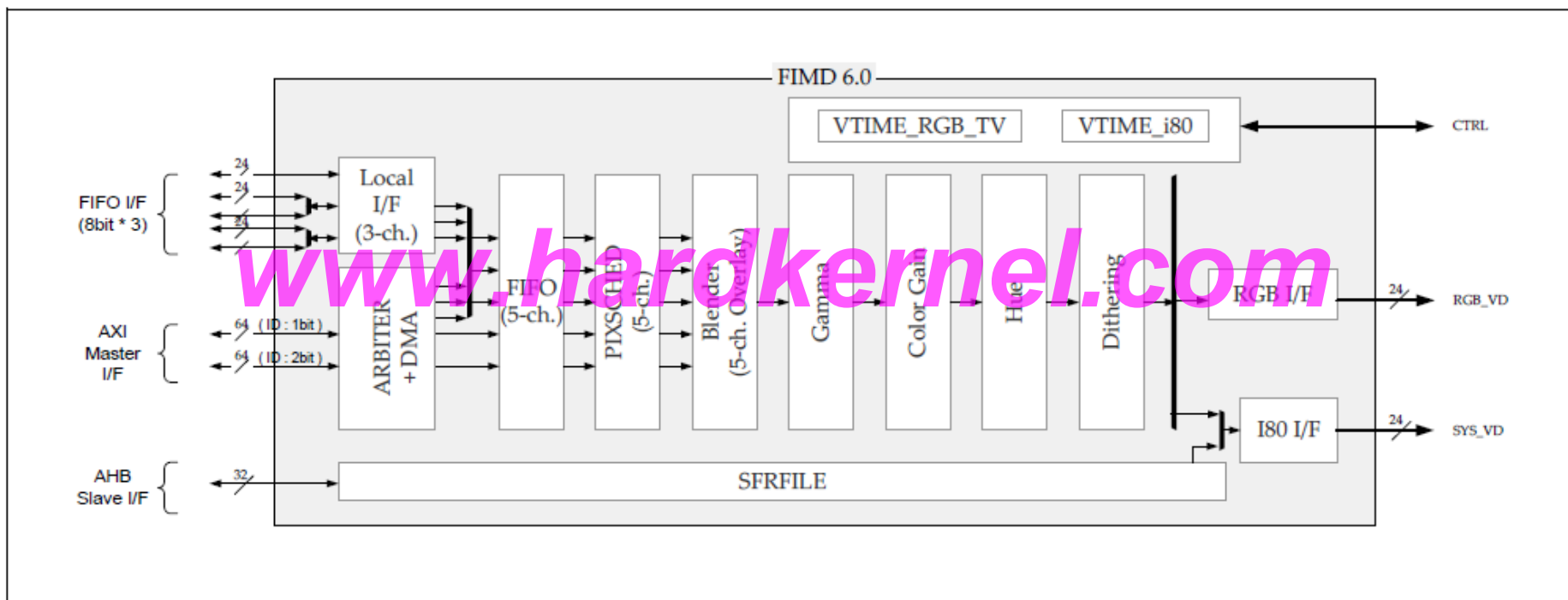
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Exynos4210 디스플레이 서브시스템

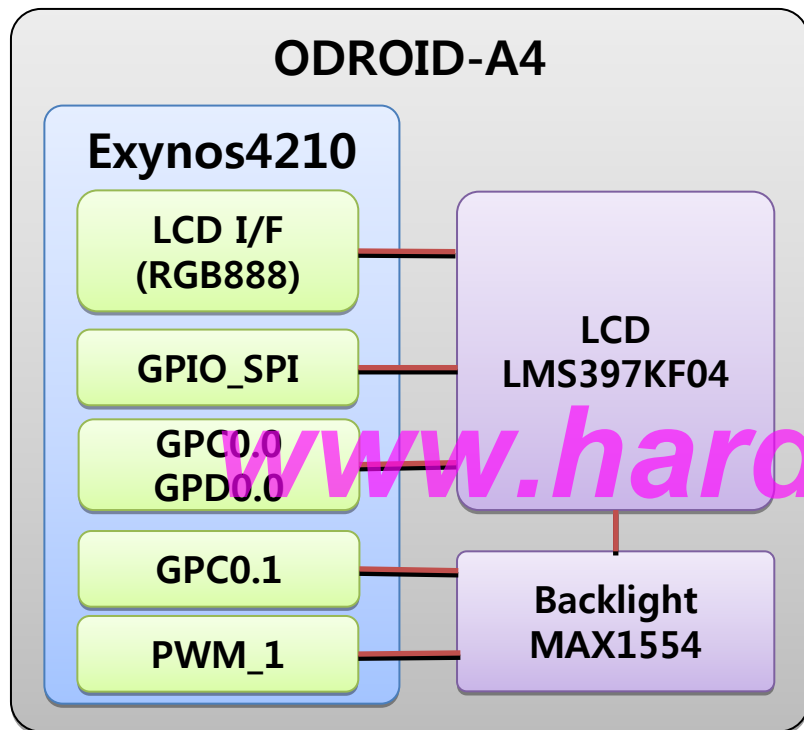
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Exynos4210 Display Controller Block Diagram

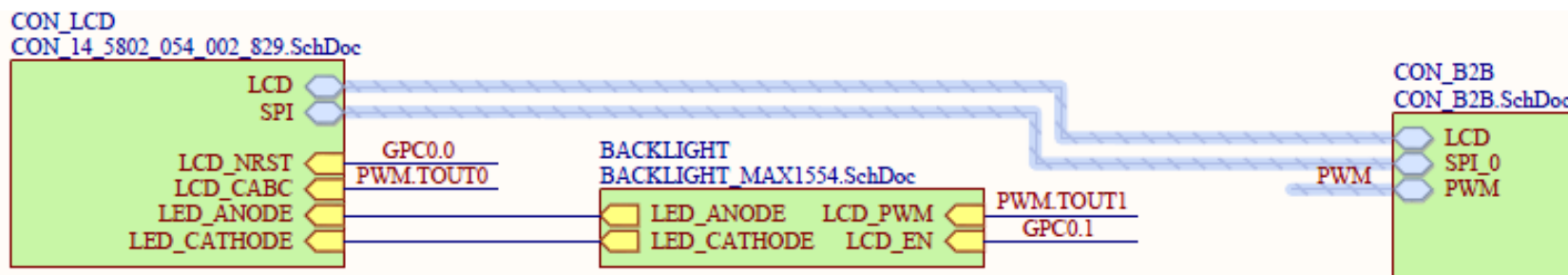


ODROID-A4 디스플레이 구성

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

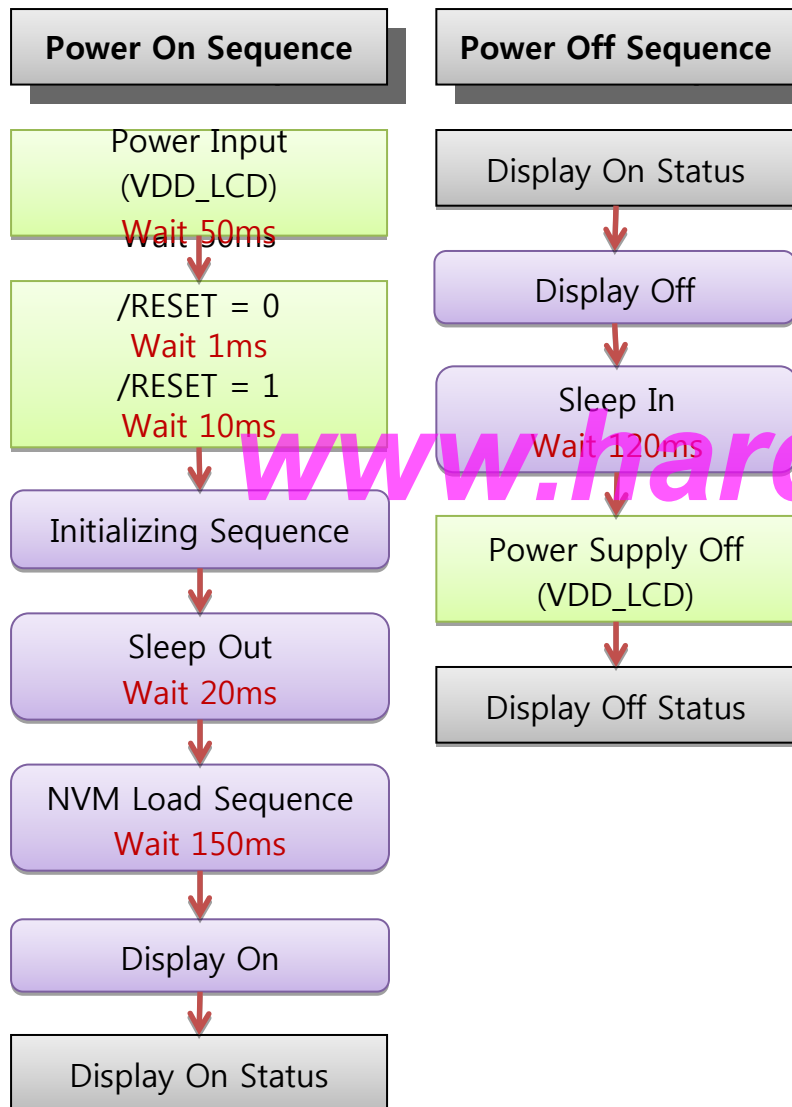


- ODROID-A4의 디스플레이 하드웨어 구성은 왼쪽의 블록다이어그램과 같다.
- Exynos4210 LCD_A 포트에 24Bit-RGB888 Interface
- LMS397KF04 LCD Init 을 위한 gpio bit-banging SPI
- PWM_1 Input으로 LCD_BLU Driving 위한 MAX1554
- 전원 관리를 위한 조도센서 BH1780GHI
- 아래 회로도의 LCD_NRST, LCD_CABC, LCD_EN 핀과 연결된 GPIO pin.
- PWM.TOUT1 핀이 밝기 조절을 위한 PWM output
- PWM.TOUT0 핀은 LCD 자체 전원관리 Unit(CABC)을 On/Off 하는 GPIO로 사용.



LCD(LMS397KF04) Spec.

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



\$KERNEL_ROOT/drivers/video/samsung/lms397kf04.c

```

static void init_lms397kf04_ldi(void)
{
    unsigned short pos = 0;
    lcd_reset();
    while(InitData[ pos ] != INIT_END) {
        if(InitData[ pos ] == DELAY_CMD) delay_ms(InitData[ pos + 1 ]);
        else {
            if (InitData[ pos + 1 ] == GAMMA_RED) .....
            if (InitData[ pos + 1 ] == GAMMA_GREEN) .....
            if (InitData[ pos + 1 ] == GAMMA_BLUE) .....
            else {
                // Index reg, wdata pointer, wdata size
            }
        }
        pos += 2; // Buffer position move
    }

    InitHW = 1; // Initialize True;
    printk("lcd power on sequence done...\n");
}

.....

void s3cfb_set_lcd_info(struct s3cfb_global *ctrl)
{
    InitHW = 0; // Initialize True;

    // init gpio && lcd reset
    init_gpio();

    // set lcd info
    lms397kf04.init_ldi = init_lms397kf04_ldi;
    lms397kf04.deinit_ldi = deinit_lms397kf04_ldi;
    ctrl->lcd = &lms397kf04;

    printk("registerd LMS397KF04 LCD Driver.\n");
}
  
```

LCD 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

\$KERNEL_ROOT/arch/arm/mach-exynos/setup-fb-odroid.c

```
// Exynos4210 LCD Block pin-function 초기화.
// Exynos4210 LCD Block clock 초기화.
// LCD On/Off, Backlight On/Off 함수 설정.

int s3cfb_backlight_on(struct platform_device *pdev)
{
    // Backlight On/Off gpio control
    // GPC0.1 회로로 참조
    .....
}

int s3cfb_backlight_off(struct platform_device *pdev)
{
    // Backlight On/Off gpio control
    // GPC0.1 회로로 참조
    .....
}
```

\$KERNEL_ROOT/arch/arm/mach-exynos/mach-odroid-4210.c

```
// display driver platformdata 등록.
static struct s3c_platform_fb lms397kf04_data __initdata = {
    .hw_ver = 0x70,
    .clk_name = "sclk_lcd",
    .nr_wins = 5,
    .default_win = CONFIG_FB_S5P_DEFAULT_WINDOW,
    .swap = FB_SWAP_HWORD | FB_SWAP_WORD,
};

static void __init smdkv310_machine_init(void)
{
    .....
    s3cfb_set_platdata(&lms397kf04_data);
    .....
}
```

\$KERNEL_ROOT/drivers/video/samsung/lms397kf04.c

```
static struct s3cfb_lcd lms397kf04 = {
    .width = 480,           /* LMS397kf04 LCD timing data */
    .height = 800,         /* LCD resolution */
    .bpp = 32,             /* bit per pixel */
    .freq = 60,            /* refresh frequency */

    .timing = {             /* Exynos4210 LCD Block timing factor와
                             LCD Timing factor간의 용어의 차이가 있을
                             수 있으므로 Timing Diagram을 통해
                             정확한 값을 찾아 설정한다. */
        .h_fp = 20,
        .h_bp = 30,
        .h_sw = 0,
        .v_fp = 6,
        .v_fpe = 1,
        .v_bp = 7,
        .v_bpe = 1,
        .v_sw = 1,
    },

    .polarity = {          /* Clock Edge trigger LCD에 맞게 설정 */
        .rise_vclk = 1,
        .inv_hsync = 1,
        .inv_vsync = 1,
        .inv_vden = 0,
    },
};
```

\$KERNEL_ROOT/drivers/video/samsung/s3cfb_main.c

```
static int s3cfb_probe(struct platform_device *pdev)
{
    // Exynos4210 FIMG init
    // Framebuffer Alloc.
    // register framebuffer
    // display logo
    // power state register
    // platform data setup (backlight_on(), lcd_on(), init_ldi())
    //
}
```

PWM Backlight 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

\$KERNEL_ROOT/arch/arm/mach-exynos/mach-odroid-4210.c

```
// backlight pwm gpio, pin function define
static struct samsung_bl_gpio_info odroid_bl_gpio_info = {
    .no = EXYNOS4_GPD0(1),
    .func = S3C_GPIO_SFN(2),
};

static struct platform_pwm_backlight_data odroid_bl_data = {
    .pwm_id = 1,
};

// backlight driver register
static void __init smdkv310_machine_init(void)
{
    samsung_bl_set(&odroid_bl_gpio_info, &odroid_bl_data);
    .....
}
```

\$KERNEL_ROOT/arch/arm/plat-samsung/dev-backlight.c

```
// pwm frequency, max/default value 초기화
platform_pwm_backlight_data samsung_dfl_bl_data = {
    .max_brightness = 255,
    .dft_brightness = 128,
    .pwm_period_ns = 44200,
    .init = samsung_bl_init,
    .exit = samsung_bl_exit,
};

void samsung_bl_set(struct samsung_bl_gpio_info *gpio_info,
    struct platform_pwm_backlight_data
    *bl_data)
{
    // platform data 등록
    // pwm device parent 설정. pwm_id 에 해당하는 timer로 설정.
    // pwm_id 에 해당되는 timer device register
    // pwm driver register
}
```

\$KERNEL_ROOT/drivers/video/backlight/pwm_bl.c

```
static int pwm_backlight_probe(struct platform_device *pdev)
{
    struct platform_pwm_backlight_data *data = pdev->dev.platform_data;

    // Platform data를 가져와서 pwm 포트를 초기화한다.
    pb->pwm = pwm_request(data->pwm_id, "backlight");

    // backlight device register.
    bl = backlight_device_register(dev_name(&pdev->dev), &pdev->dev, pb,
        &pwm_backlight_ops, &props);

    // pwm 주파수 초기값 적용.
    backlight_update_status(bl);

    // driver data 저장.
    platform_set_drvdata(pdev, bl);
}

// brightness값을 pwm_duty 값으로 변환하여 pwm 출력을 설정하는 함수
static int pwm_backlight_update_status(struct backlight_device *bl)
{
    struct pwm_bl_data *pb = dev_get_drvdata(&bl->dev);
    int brightness = bl->props.brightness;
    int max = bl->props.max_brightness;
    .....
    if (brightness == 0) {
        pwm_config(pb->pwm, 0, pb->period);
        pwm_disable(pb->pwm);
    } else {
        brightness = pb->lth_brightness +
            (brightness * (pb->period - pb->lth_brightness) / max);
        pwm_config(pb->pwm, brightness, pb->period);
        pwm_enable(pb->pwm);
    }
    return 0;
}
```

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Kernel Splash Screen 이미지 변경하기

\$KERNEL_ROOT/drivers/video/samsung/s3cfb_ops.c

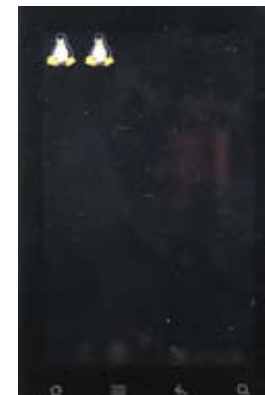
```
int s3cfb_draw_logo(struct fb_info *fb)
{
    u32 height = var->yres / 4;
    u32 line = fix->line_length;
    u32 i, j;
    .....
    for (i = height * 3; i < height * 4; i++) {
        for (j = 0; j < var->xres; j++) {
            memset(fb->screen_base + i * line + j * 4 + 0, 0x00, 1);
            memset(fb->screen_base + i * line + j * 4 + 1, 0xff, 1);
            memset(fb->screen_base + i * line + j * 4 + 2, 0xff, 1);
            memset(fb->screen_base + i * line + j * 4 + 3, 0x00, 1);
        }
    }
    return 0;
}
```

\$KERNEL_ROOT/drivers/video/samsung/s3cfb_main.c

```
int s3cfb_register_framebuffer(struct s3cfb_global *fbdev)
{
    .....
    if (j == pdata->default_win) {
        s3cfb_check_var_window(fbdev, &fbdev->fb[j]->var, fbdev-
        >fb[j]);
        s3cfb_set_par_window(fbdev, fbdev->fb[j]);
        // s3cfb_draw_logo(fbdev->fb[j]);

        if (fb_prepare_logo(fbdev->fb[j], FB_ROTATE_UR)) {
            fb_set_cmap(&fbdev->fb[j]->cmap, fbdev->fb[j]);
            fb_show_logo(fbdev->fb[j], FB_ROTATE_UR);
        }
    }
}
```

- 현재 ODROID-A4의 Kernel logo가 그려지는 부분이다.
- 가로로 3등분하여 각각 R/G/B 를 프레임버퍼에 쓰는 코드.
- s3cfb_draw_logo 함수를 왼쪽과 같이 수정.
- 수정된 코드는 LCD를 4등분하여 네번째 영역을 Yellow로 채우도록 추가된 코드이다.
- 수정된 코드로 Kernel 이미지를 만들어 적용하면 아래와 같다.
- s3cfb_register_framebuffer() 함수에서 RGB display를 주석처리
- config에 정의된 boot logo를 display할 수 있도록 코드를 추가
- 수정된 커널 이미지를 적용하면 리눅스 펌웨어 두마리를 볼 수 있다.
- drivers/video/logo/logo_dec_clut224.ppm 파일이 C파일로 변환되어 커널 이미지에 포함된다.
- netpbm 패키지를 설치하여 이미지 파일을 ppm 파일로 변환한후 커널에 적용시킴으로서 사용자 이미지로 수정하는 것도 가능하다.



Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Backlight 제어 API

```
COM14@Serial - Xshell 4 (Free for Home/School)

root@android:/ #
root@android:/ # cd /sys/class/backlight/pwm-backlight.0
root@android:/sys/class/backlight/pwm-backlight.0 # ls
actual_brightness
bl_power
brightness
device
max_brightness
power
subsystem
type
uevent
root@android:/sys/class/backlight/pwm-backlight.0 # cat brightness
146
root@android:/sys/class/backlight/pwm-backlight.0 # echo 255 > brightness
root@android:/sys/class/backlight/pwm-backlight.0 # cat brightness
255
root@android:/sys/class/backlight/pwm-backlight.0 # echo 1 > brightness
root@android:/sys/class/backlight/pwm-backlight.0 # cat brightness
1
root@android:/sys/class/backlight/pwm-backlight.0 # █
```

- backlight sysfs 폴더로 이동.
- brightness node를 확인한다.
- 현재의 밝기 값을 확인
- 최대 밝기인 255로 셋팅 하고,
ODROID-A4의 화면밝기를 확인한다.
- 최저 밝기인 1로 셋팅 하고,
화면밝기의 변화를 확인한다.
(0 은 화면이 꺼졌을 때의 값이다.)
- 설정 -> display -> brightness 를 실행.
- 밝기 조절바를 옮겨가면서 brightness node를 읽어보면 값의 변화를 확인할 수 있다. 화면밝기 조정은 brightness node를 통해서 조정된다.

\$ANDROID_ROOT/device/hardkernel/odroida4/conf/init.odroida4.rc

```
#=====
#
# Permissions for backlight
#
#=====
chown system system /sys/class/backlight/pwm-backlight.0/brightness
```

안드로이드 init.rc에서의 Permission 설정

Keypad 드라이버의 구조

www.hardkernel.com

Agenda

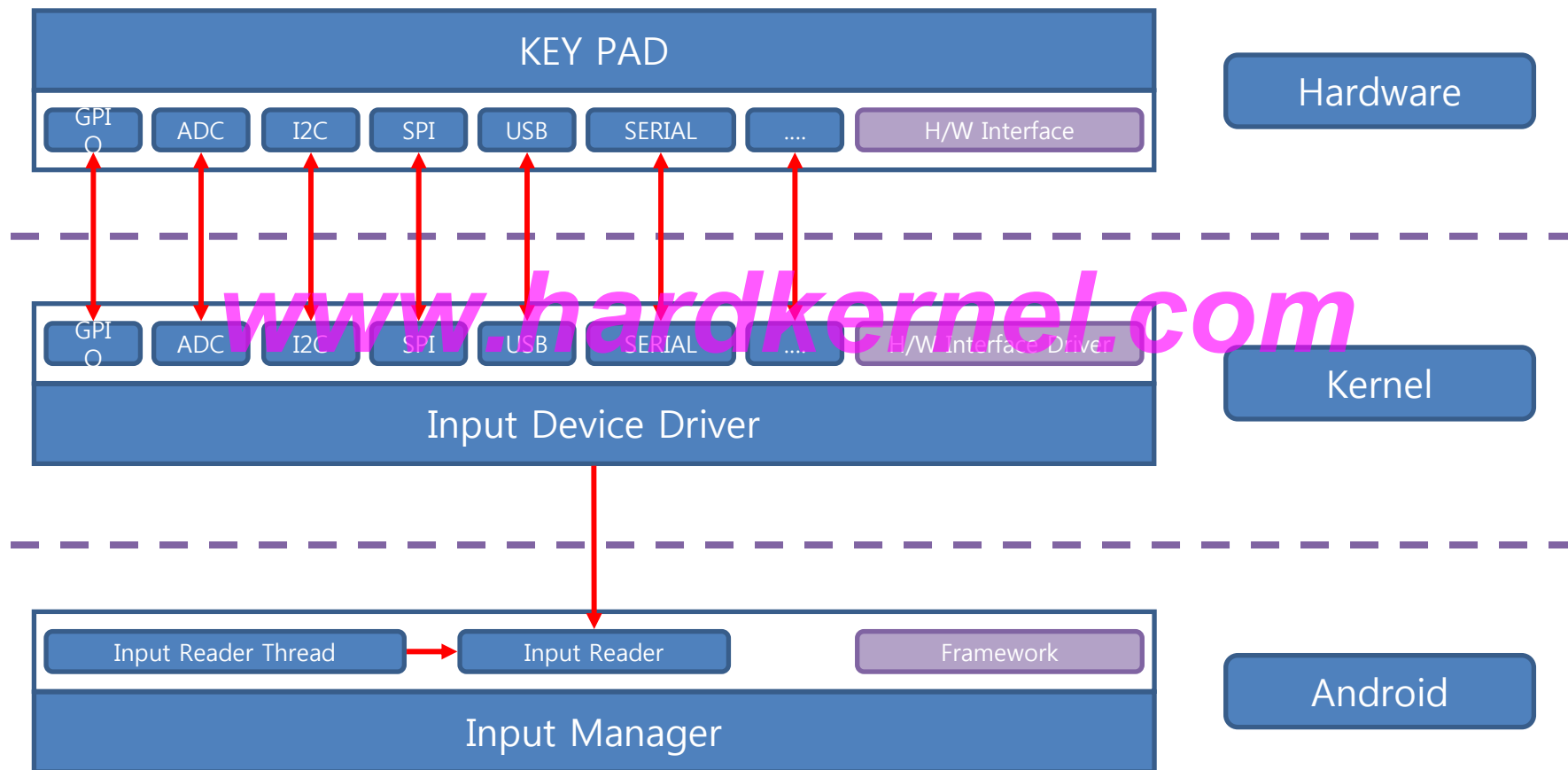
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Keypad Data Flow
- Keypad Hardware Interface
- Keypad 드라이버 구조
- Keypad driver probe
- Keypad data report
- Keypad driver remove
- Reference
- Lab/Exam
 - Keypad 드라이버를 Interrupt 방식으로 변환 하기

www.hardkernel.com

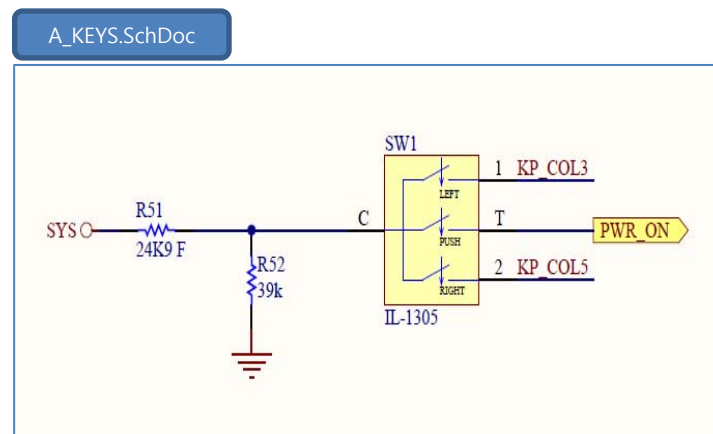
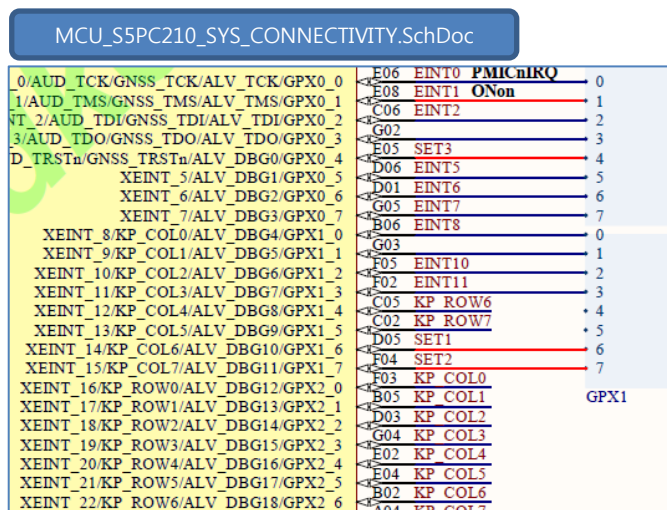
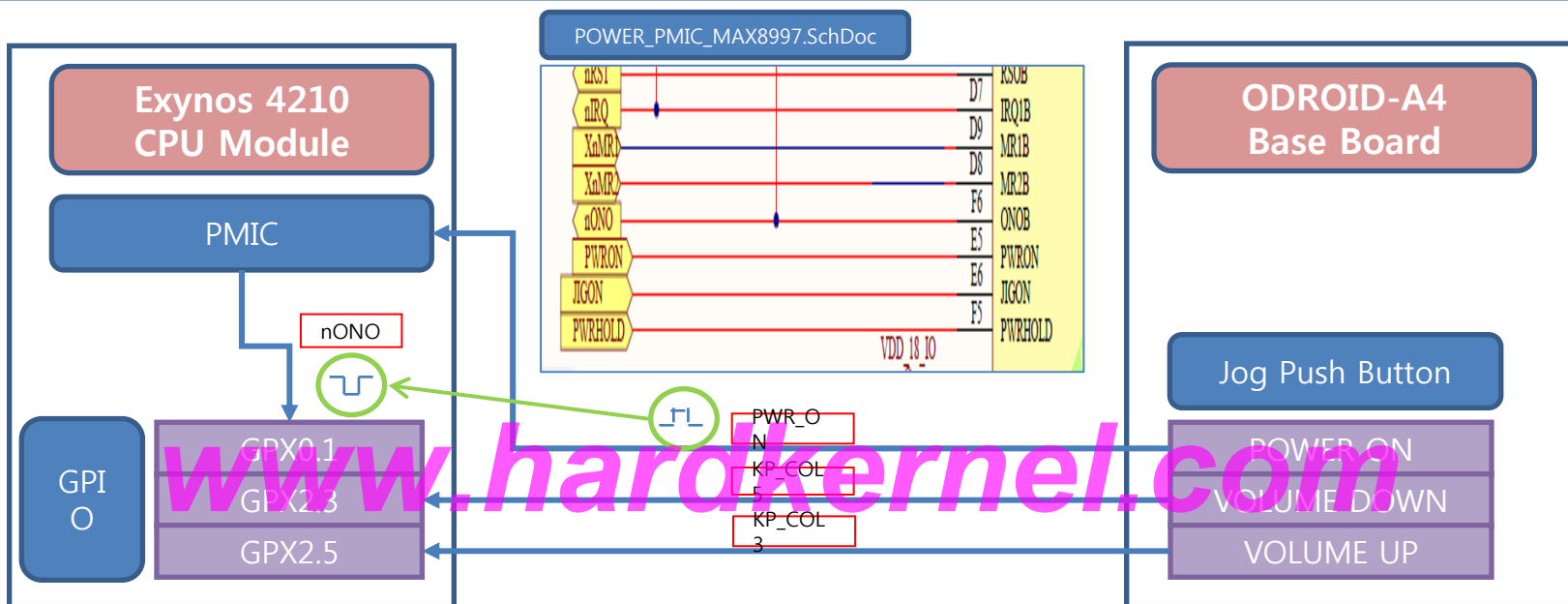
Keypad Data Flow

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



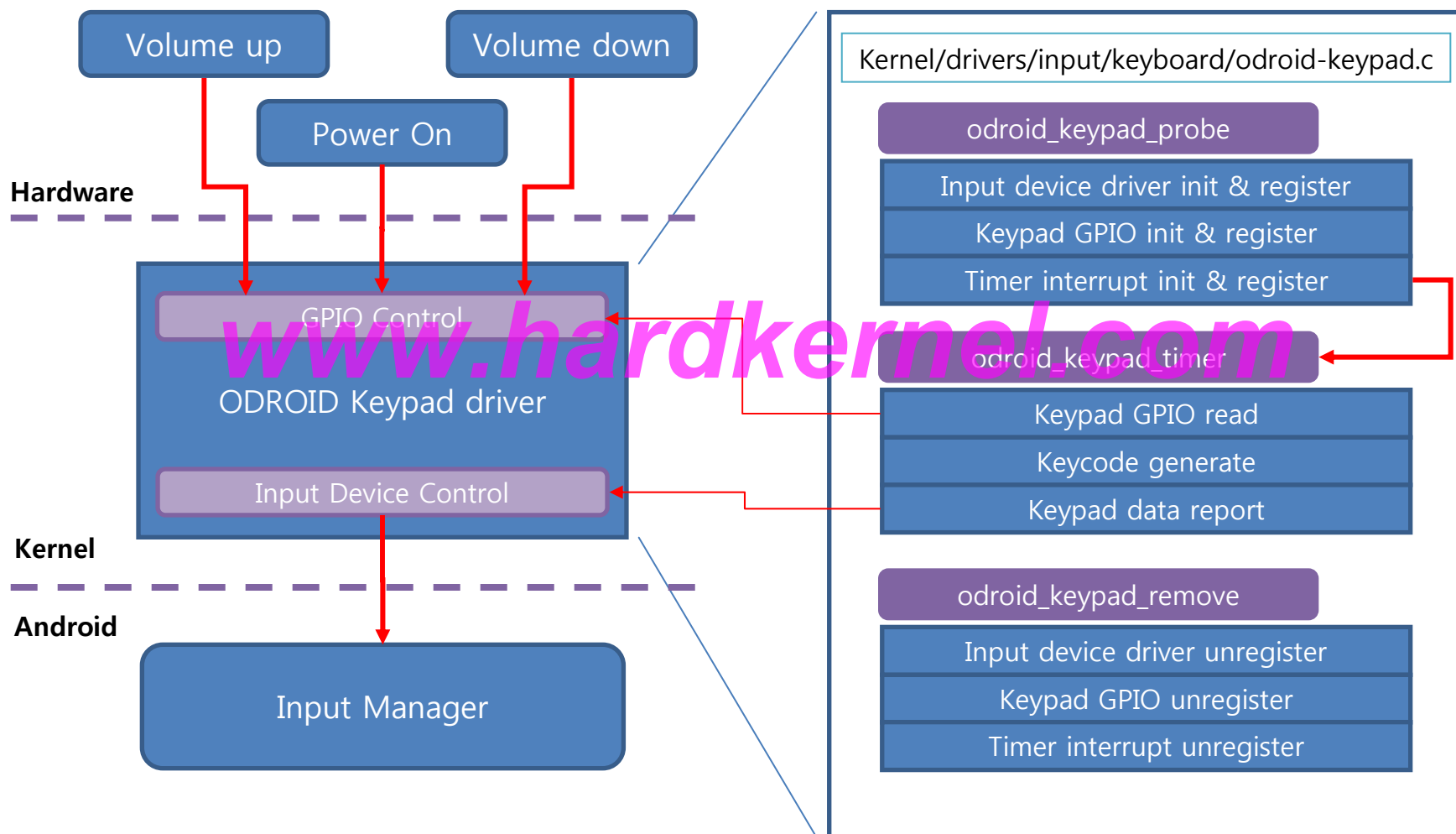
Keypad Hardware Interface

www.hardkernel.com 이 자료는 (주)하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Keypad 드라이버 구조

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Keypad driver probe(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Input device driver init & register

```
static int __devinit odroid_keypad_probe(struct platform_device *pdev)
{
    ...

    if((keypad->input = input_allocate_device()) goto err_free_input_mem;

    snprintf(keypad->phys, sizeof(keypad->phys), "%s/input0", DEVICE_NAME);

    keypad->input->name      = DEVICE_NAME;
    keypad->input->phys      = keypad->phys;
    keypad->input->id.bustype = BUS_HOST;
    keypad->input->id.vendor  = 0x16B4;
    keypad->input->id.product = 0x0701;
    keypad->input->id.version = 0x0001;
    keypad->input->keycode    = Keycode;
    keypad->input->open       = odroid_keypad_open;
    keypad->input->close      = odroid_keypad_close;
    set_bit(EV_KEY, keypad->input->evbit);

    for(key = 0; key < MAX_KEYCODE_CNT; key++){
        code = Keycode[key];
        if(code <= 0) continue;
        set_bit(code & KEY_MAX, keypad->input->keybit);
    }

    if(input_register_device(keypad->input){
        printk("-----\n");
        printk("%s input register device fail!!\n", DEVICE_NAME);
        printk("-----\n");
        goto err_free_all;
    }

    ...
}
```

Input device driver memory allocation

Input device struct initialize

Report data type 설정

```
...
#define MAX_KEYCODE_CNT 3

int Keycode[MAX_KEYCODE_CNT] = {
    KEY_POWER,
    KEY_VOLUMEUP,
    KEY_VOLUMEDOWN,
};
...
```

Kernel/include/linux/input.h

Report keycode enable

Input device driver register

Keypad driver probe(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Keypad GPIO init & register

```
// GPIO Index Define
enum
{
    KEYPAD_POWER,
    KEYPAD_VOLUME_UP,
    KEYPAD_VOLUME_DOWN,
    KEYPAD_POWER_LED,
    GPIO_INDEX_END
};
```

```
static struct {
    int    gpio_index, gpio;
    char   *name;
    bool   output;
    int    value, pud;
} sControlGpios[] = {
    { KEYPAD_POWER,      EXYNOS4_GPX0(1), "KEY POWER"          , 0, 0, S3C_GPIO_PULL_NONE },
    { KEYPAD_VOLUME_UP,  EXYNOS4_GPX2(5), "KEY VOLUME UP"      , 0, 0, S3C_GPIO_PULL_DOWN },
    { KEYPAD_VOLUME_DOWN, EXYNOS4_GPX2(3), "KEY VOLUME DOWN"    , 0, 0, S3C_GPIO_PULL_DOWN },
    { KEYPAD_POWER_LED,  EXYNOS4_GPK1(6), "POWER LED"         , 1, 1, S3C_GPIO_PULL_NONE },
};
```

```
static void    odroid_keypad_config(odroid_keypad_t *keypad)
{
    Int i;

    // Control GPIO Init
    for (i = 0; i < ARRAY_SIZE(sControlGpios); i++) {
        if(gpio_request(sControlGpios[i].gpio, sControlGpios[i].name)) {
            // request error message
        }
        else
        {
            if(sControlGpios[i].output)
                gpio_direction_output(sControlGpios[i].gpio, sControlGpios[i].value);
            else
                gpio_direction_input(sControlGpios[i].gpio);

            s3c_gpio_setpull(sControlGpios[i].gpio, sControlGpios[i].pud);
        }
    }
    ...
}
```

System에 GPIO 사용 요청 및 등록

GPIO output mode 설정 및 초기화

GPIO input mode 설정

CPU내부 Pull up/down register 설정

Keypad driver probe(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Timer interrupt init & register

Timer init

Timer callback function register

Kernel/drivers/input/keyboard/odroid-keypad.h

```
...  
#define DEVICE_NAME "odroid-keypad"  
  
#define KEY_PRESS 1  
#define KEY_RELEASE 0  
  
// ns : ktime_set(sec, nsec)  
#define KEYPAD_TIMER_PERIOD 100000000  
...
```

Timer start

```
static int __devinit odroid_keypad_probe(struct platform_device *pdev)  
{  
    ...  
    hrtimer_init(&keypad->timer, CLOCK_MONOTONIC, HRTIMER_MODE_REL);  
    keypad->timer.function = odroid_keypad_timer;  
    odroid_keypad_config(keypad);  
    ...  
}
```

현재 시간을 기준으로 100ms 후에
timer callback function 실행

```
static void odroid_keypad_config(odroid_keypad_t *keypad)  
{  
    ...  
    hrtimer_start(&keypad->timer, ktime_set(0, KEYPAD_TIMER_PERIOD),  
                  HRTIMER_MODE_REL);  
}
```

www.hardkernel.com

Keypad data report(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

```
static int      odroid_keypad_get_data(void)
{
    int    key_data = 0;

    key_data |= (gpio_get_value(sControlGpios[KEYPAD_POWER].gpio)      ? 0 : 0x01);
    key_data |= (gpio_get_value(sControlGpios[KEYPAD_VOLUME_UP].gpio)    ? 0x02 : 0);
    key_data |= (gpio_get_value(sControlGpios[KEYPAD_VOLUME_DOWN].gpio) ? 0x04 : 0);

    return    key_data;
}
```

```
static void odroid_keypad_control(odroid_keypad_t *keypad)
{
    static    unsigned short    prev_keypad_data = 0, cur_keypad_data = 0;

    cur_keypad_data = odroid_keypad_get_data();

    if(prev_keypad_data != cur_keypad_data) {
        generate_keycode(keypad, prev_keypad_data, cur_keypad_data, &Keycode[0]);
        prev_keypad_data = cur_keypad_data;
        input_sync(keypad->input);
    }
}
```

```
static enum hrtimer_restart odroid_keypad_timer(struct hrtimer *timer)
{
    odroid_keypad_t *keypad = container_of(timer, odroid_keypad_t, timer);
    odroid_keypad_control(keypad);
    hrtimer_start(&keypad->timer, ktime_set(0, KEYPAD_TIMER_PERIOD),
    HRTIMER_MODE_REL);
    return HRTIMER_NORESTART;
}
```

```
#define    MAX_KEYCODE_CNT 3
int Keycode[MAX_KEYCODE_CNT] = {
    KEY_POWER,
    KEY_VOLUMEUP,
    KEY_VOLUMEDOWN,
};
```



1 = press
0 = release

Keypad GPIO read

Keypad GPIO상태의 변경을
검사하여 Key data를 전송한다.

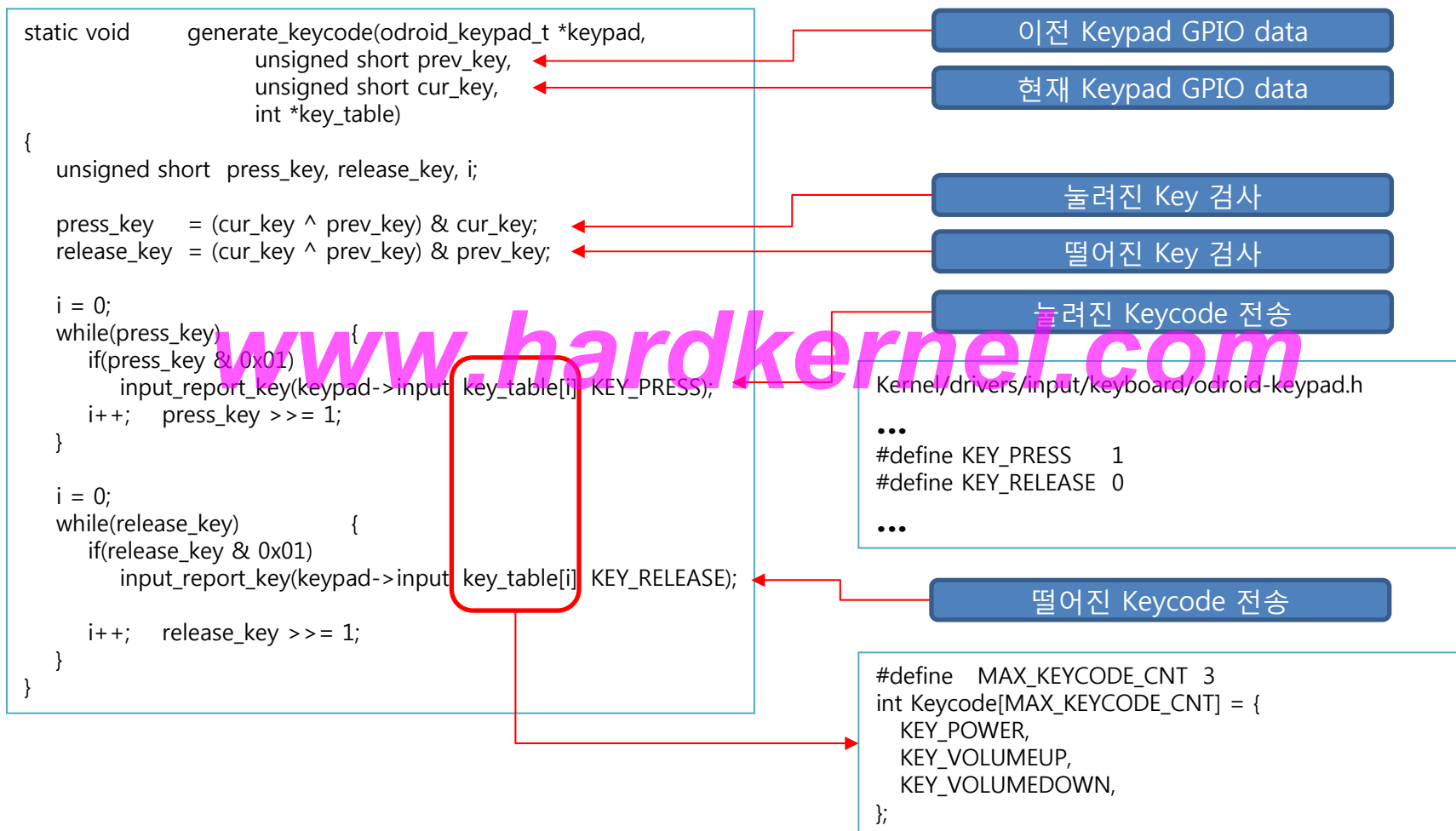
Timer callback function

Timer 재시작

www.hardkernel.com

Keypad data report(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Keypad driver remove

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

```
static int __devexit odroid_keypad_remove(struct platform_device *pdev)
{
    int i;

    odroid_keypad_t *keypad = dev_get_drvdata(&pdev->dev);

    input_unregister_device(keypad->input);

    hrtimer_cancel(&keypad->timer);

    dev_set_drvdata(&pdev->dev, NULL);

    for (i = 0; i < ARRAY_SIZE(sControlGpios); i++)
        gpio_free(sControlGpios[i].gpio);

    kfree(keypad);

    return 0;
}
```

Input device driver unregister

Timer stop

GPIO unregister

Reference(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

GPIO Control API (include/linux/gpio.h)

Function

int gpio_is_valid(int number);

Description

GPIO가 유효한지 확인.

Return

에러인 경우 0 or Negative value를 return, 유효한 경우 GPIO number를 return함.

Usage

```
ret = gpio_is_valid(EXNOS4_GPX1(0));
```

Function

int gpio_request(unsigned gpio, const char *label);

Description

해당 GPIO를 사용가능 하도록 Memory에 할당.

Return

에러인 경우 0 or Negative value를 return, 유효한 경우 GPIO number를 return 함.

Usage

```
ret = gpio_request(EXNOS4_GPX1(0), "power on key");
```

Function

void gpio_free(unsigned gpio);

Description

gpio_request function으로 할당한 GPIO memory를 해제함.

Return

없음.

Usage

```
gpio_free(EXNOS4_GPX1(0));
```

Reference(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

GPIO Control API (include/linux/gpio.h)

Function

int gpio_get_value(unsigned gpio);

Description

해당 GPIO의 상태를 읽음.

Return

해당 GPIO pin이 high인 경우 1, low인 경우 0을 return함.

Usage

```
ret = gpio_get_value(EXNOS4_GPX1(0));
```

Function

void gpio_set_value(unsigned gpio, int value);

Description

해당 GPIO의 Logic을 Control

Return

없음.

Usage

해당 GPIO에 Logic High를 출력하는 경우

```
gpio_set_value(EXNOS4_GPX1(0), 1);
```

해당 GPIO에 Logic Low를 출력하는 경우

```
gpio_set_value(EXNOS4_GPX1(0), 0);
```

Reference(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

GPIO Control API (include/linux/gpio.h)

Function

int gpio_to_irq(unsigned gpio);

Description

해당 GPIO를 IRQ number로 변환함.

Return

에러인 경우 0 or Negative value를 return, 유효한 경우 GPIO irq number를 return 함.

Usage

```
if(request_irq(gpio_to_irq(EXNOS4_GPX1(0)),
               sControlGpios[i].irq_func,
               (IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING | IRQF_DISABLED),
               sControlGpios[i].name,
               keypad));
```

- 더 많은 정보는 [kernel/Documentation/gpio.txt](#)를 참조

Reference(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

hrtimer Control API (include/linux/hrtimer.h)

Function

```
void hrtimer_init(struct hrtimer *timer, clockid_t witch_clock,  
                  enum hrtimer_mode mode);
```

Description

고분해능 Timer를 초기화 한다. 사용할 clock은 /include/linux/time.h에 정의 되어 있으며 시스템에 지원하는

다양한 clock을 나타낸다. 또한 timer mode로 상대값 또는 절대값으로 초기화 된다.

Return

없음.

Usage

```
struct hrtimer timer;
```

```
init hrtimer_init(&timer, CLOCK_MONOTONIC, HRTIMER_MODE_REL);
```

Function

```
int hrtimer_start(struct hrtimer *timer, ktime_t time,  
                  const enum hrtimer_mode mode);
```

Description

hrtimer_init() 함수로 초기화 되어진 timer를 시작한다. 이 함수는 만기시간(ktime_t) 및 시간 모드값을 포함한다.

Return

0 : 성공, 1 : timer가 이미 사용중임.

Usage

```
ret = hrtimer_start(&timer, ktime_set(0, 100000000), HRTIMER_MODE_REL);
```

www.hardkernel.com

Reference(5)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

hrtimer Control API (include/linux/hrtimer.h)

Function

void hrtimer_cancel(struct hrtimer *timer);

Description

사용중인 timer를 취소한다. 만약 timer가 시작되어진 경우라면 종료 시점까지 대기한다.

Return

없음.

Usage

hrtimer_cancel(&timer);

Function

void hrtimer_try_to_cancel(struct hrtimer *timer);

Description

사용중인 timer를 취소한다. 만약 timer가 시작되어진 경우라면 오류를 return한다.

Return

0 : timer stop, -1 : 타이머 사용중.

Usage

ret = **hrtimer_try_to_cancel(&timer);**

- 더 많은 정보는 Kernel/Documentation/timer/hrtimer.txt를 참조

Reference(6)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

hrtimer Control 예제

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/hrtimer.h>
#include <linux/ktime.h>

struct timer timer;

enum hrtimer_restart timer_handler(struct hrtimer *timer) // Timer callback function
{
    printk(KERN_WARNING, "timer_handler\n");
    return HRTIMER_NORESTART;
}

int init_module(void)
{
    ktime_t ktime;

    ktime = ktime_set(0, 1000000000); // ktime_set(sec, nanosec);
    hrtimer_init(&timer, CLOCK_MONOTONIC, HRTIMER_MODE_REL);
    timer.function = &timer_handler;
    hrtimer_start(&timer, ktime, HRTIMER_MODE_REL);

    return 0;
}
```

LAB/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Keypad 드라이버를 Interrupt 방식으로 변환 하기

각 Key에 해당하는 Interrupt function 작성하기

```
irqreturn_t    irq_power(int irq, void *handle)
{
    odroid_keypad_t *keypad    = (odroid_keypad_t *)handle;
    int key_status;

    key_status = gpio_get_value(sControlGpios[KEYPAD_POWER].gpio) ? false : true;
    input_report_key(keypad->input, Keycode[KEYPAD_POWER], key_status);
    input_sync(keypad->input);

    printk("%s : %d\n", __func__, key_status);
    return IRQ_HANDLED;
}

irqreturn_t    irq_volumeup(int irq, void *handle)
{
    odroid_keypad_t *keypad = (odroid_keypad_t *)handle;
    int key_status;

    key_status = gpio_get_value(sControlGpios[KEYPAD_VOLUME_UP].gpio) ? true : false;
    input_report_key(keypad->input, Keycode[KEYPAD_VOLUME_UP], key_status);
    input_sync(keypad->input);

    printk("%s : %d\n", __func__, key_status);
    return IRQ_HANDLED;
}

irqreturn_t    irq_volumedn(int irq, void *handle)
{
    odroid_keypad_t *keypad = (odroid_keypad_t *)handle;
    int key_status;

    key_status = gpio_get_value(sControlGpios[KEYPAD_VOLUME_DOWN].gpio) ? true : false;
    input_report_key(keypad->input, Keycode[KEYPAD_VOLUME_DOWN], key_status);
    input_sync(keypad->input);

    printk("%s : %d\n", __func__, key_status);
    return IRQ_HANDLED;
}
```

현재 Keypad GPIO상태를 검사한다.

해당 Keycode report

Input data sync

Debugging을 위한 Kernel Message

www.hardkernel.com

// GPIO Index Define

```
enum    {
    KEYPAD_POWER,
    KEYPAD_VOLUME_UP,
    KEYPAD_VOLUME_DOWN,
    KEYPAD_POWER_LED,
    GPIO_INDEX_END
};
```

```
#define    MAX_KEYCODE_CNT 3
int Keycode[MAX_KEYCODE_CNT] = {
    KEY_POWER,
    KEY_VOLUMEUP,
    KEY_VOLUMEDOWN,
};
```


LAB/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Interrupt function pointer 추가

```
static struct {
    int        gpio_index, gpio;
    char        *name;
    bool        output;
    int        value, pud;
    irqreturn_t (*irq_func)(int irq, void *handle);
} sControlGpios[] = {
    { KEYPAD_POWER,      EXYNOS4_GPX0(1), "KEY POWER"      , 0, 0, S3C_GPIO_PULL_NONE, irq_power},
    { KEYPAD_VOLUME_UP,  EXYNOS4_GPX2(5), "KEY VOLUME UP"   , 0, 0, S3C_GPIO_PULL_DOWN, irq_volumeup},
    { KEYPAD_VOLUME_DOWN, EXYNOS4_GPX2(3), "KEY VOLUME DOWN", 0, 0, S3C_GPIO_PULL_DOWN, irq_volumedn},
    { KEYPAD_POWER_LED,  EXYNOS4_GPK1(6), "POWER LED"      , 1, 1, S3C_GPIO_PULL_NONE, NULL},
};
```

Interrupt처리를 위한
함수 포인터 추가

각 Key에 해당하는
Interrupt function name 등록

Interrupt function 등록하기

```
static void odroid_keypad_config(odroid_keypad_t *keypad)
{
    int i;

    for (i = 0; i < ARRAY_SIZE(sControlGpios); i++) {
        if(sControlGpios[i].irq_func != NULL) {
            if(request_irq(gpio_to_irq(sControlGpios[i].gpio),
                sControlGpios[i].irq_func,
                (IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING | IRQF_DISABLED),
                sControlGpios[i].name,
                keypad))
                printk("%s : %s irq request err!\n", __FUNCTION__, sControlGpios[i].name);
            else {
                if(gpio_request(sControlGpios[i].gpio, sControlGpios[i].name))
                    printk("%s : %s gpio request err!\n", __FUNCTION__, sControlGpios[i].name);
                else {
                    if(sControlGpios[i].output)
                        gpio_direction_output(sControlGpios[i].gpio, sControlGpios[i].value);
                    else
                        gpio_direction_input(sControlGpios[i].gpio);
                }
            }
        }
        s3c_gpio_setpull(sControlGpios[i].gpio, sControlGpios[i].pud);
    }
}
```

GPIO Handling 값을
Interrupt Handling 값으로 바꾸는 과정

Keypad GPIO값이 변경되는 시점에
Interrupt 를 발생하게 설정

Interrupt function에 인자로
보내어질 데이터 설정

LAB/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Kernel Message를 통한 Keypad debugging

- Host PC (Linux)에서 수정되어진 Kernel build
- Uboot의 fastboot를 이용하여 새로 Build되어진 Kernel image(zImage) download
- Kernel booting 후 JOG Key에 대한 kernel message 확인

Odroid - Xshell 4 (Free for Home/School)

```
[ 151.366863] irq_volumeup : 1
[ 151.694059] irq_volumeup : 0
[ 151.718739] irq_volumeup : 0
[ 151.718836] irq_volumeup : 0
[ 151.718938] irq_volumeup : 1
[ 151.721779] irq_volumeup : 1
[ 151.724609] irq_volumeup : 1
[ 151.727475] irq_volumeup : 1
[ 151.730337] irq_volumeup : 0
[ 152.442165] irq_volumedn : 0
[ 152.442260] irq_volumedn : 0
[ 152.442679] irq_volumedn : 1
[ 152.445204] irq_volumedn : 1
[ 152.448036] irq_volumedn : 1
[ 152.450902] irq_volumedn : 1
[ 152.453760] irq_volumedn : 0
[ 152.873401] irq_volumedn : 1
[ 152.873519] irq_volumedn : 1
[ 152.873604] irq_volumedn : 1
[ 152.876410] irq_volumedn : 1
[ 152.879269] irq_volumedn : 0
[ 153.843253] irq_power : 1
[ 154.164186] irq_power : 0
[ 154.601120] request_suspend_state: sleep (0->3) at 154601075583 (2012-03-30 )
[ 154.605939] touch_suspend
[ 154.607177] s3cfb_early_suspend is called
[ 154.720130] lcd power off sequence done...
```

Volume UP Key Control

Volume DOWN Key Control

Power Key Control

Touchscreen 드라이버의 구조

www.hardkernel.com

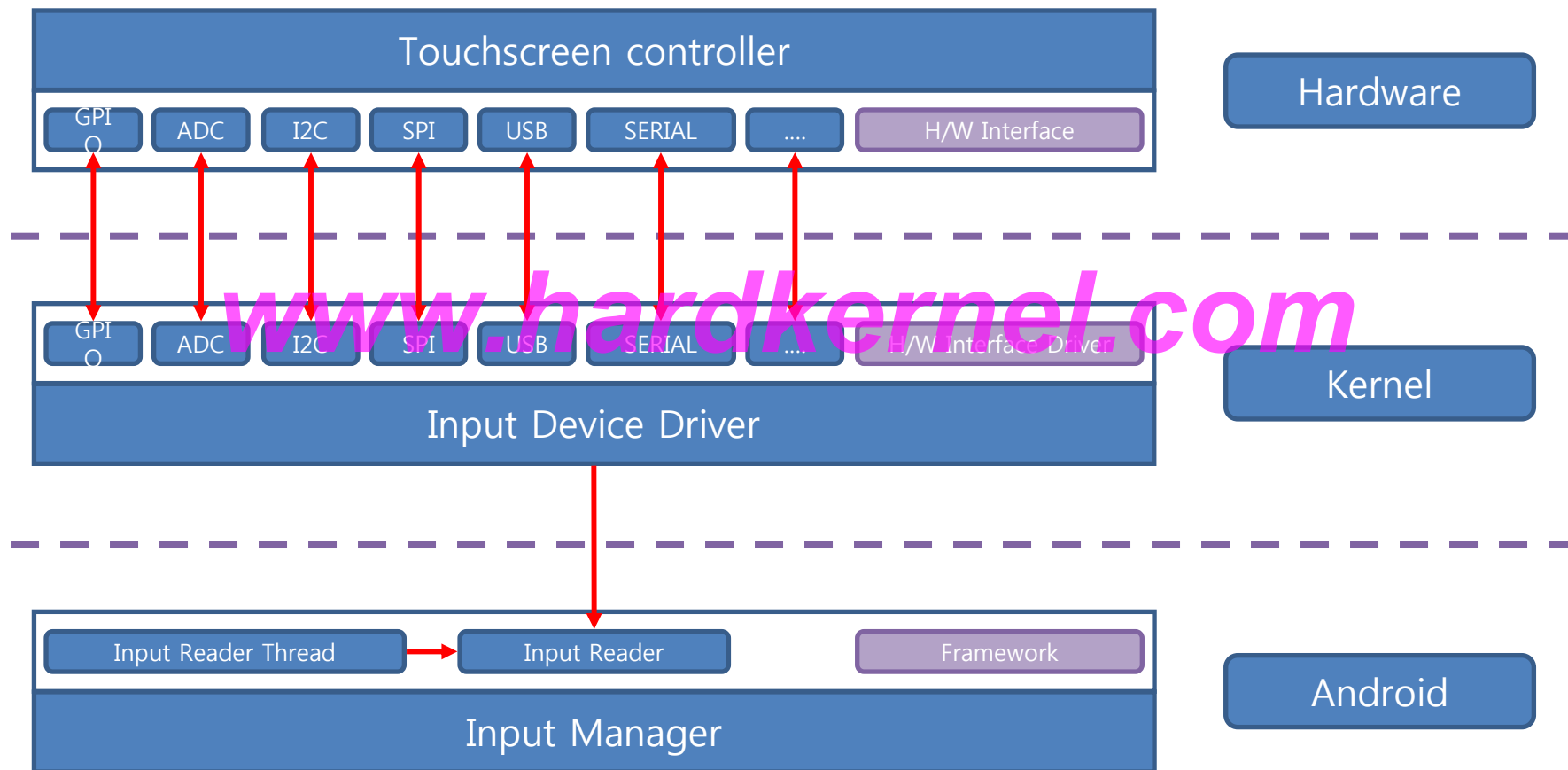
Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Touchscreen Data Flow
- Touchscreen Hardware Interface
- Touchscreen Controller I2C Specification
- Touchscreen Controller Register
- Touchscreen Controller Running Process
- Touchscreen 드라이버 구조
- Touchscreen Platform data 설정
- Touchscreen driver probe
- Touchscreen data report
- Touchscreen driver remove
- Reference
- Lab/Exam
 - Touchscreen 드라이버를 Timer Interrupt 방식으로 변환 하기

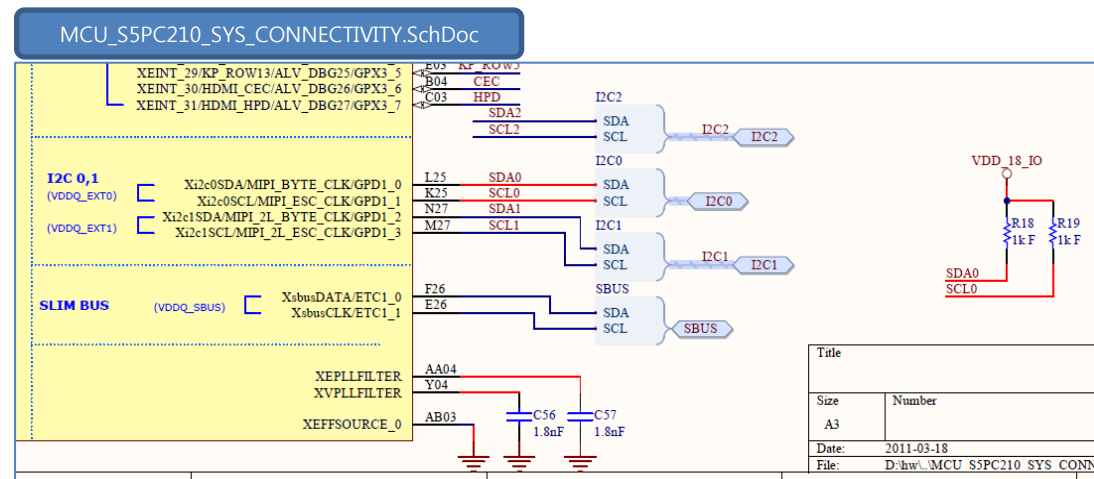
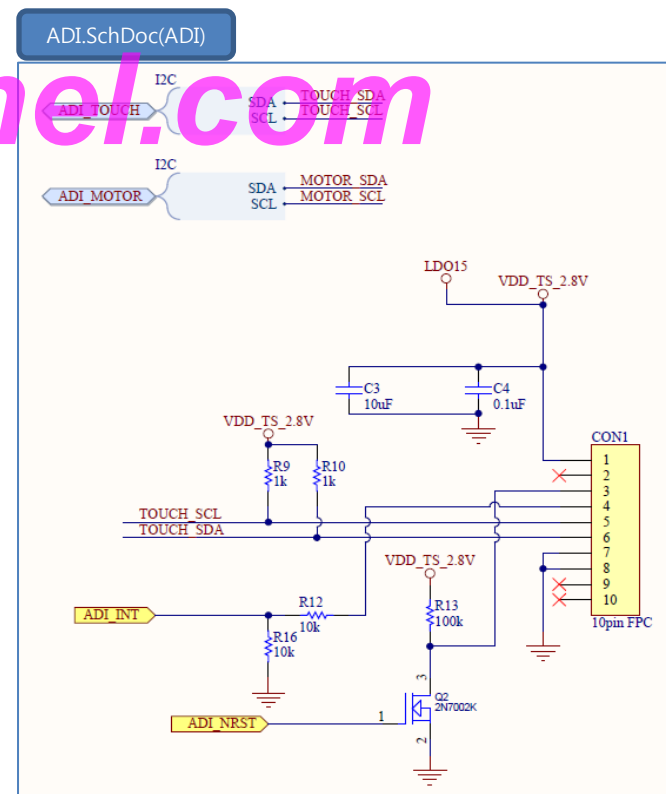
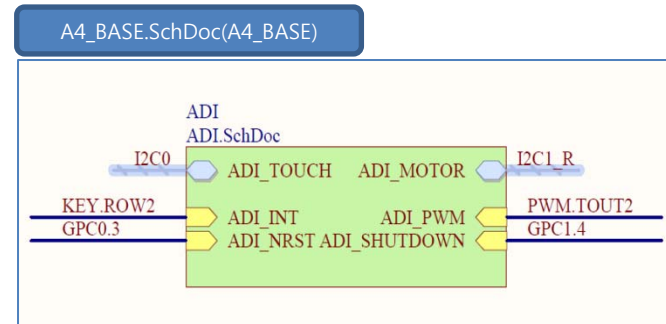
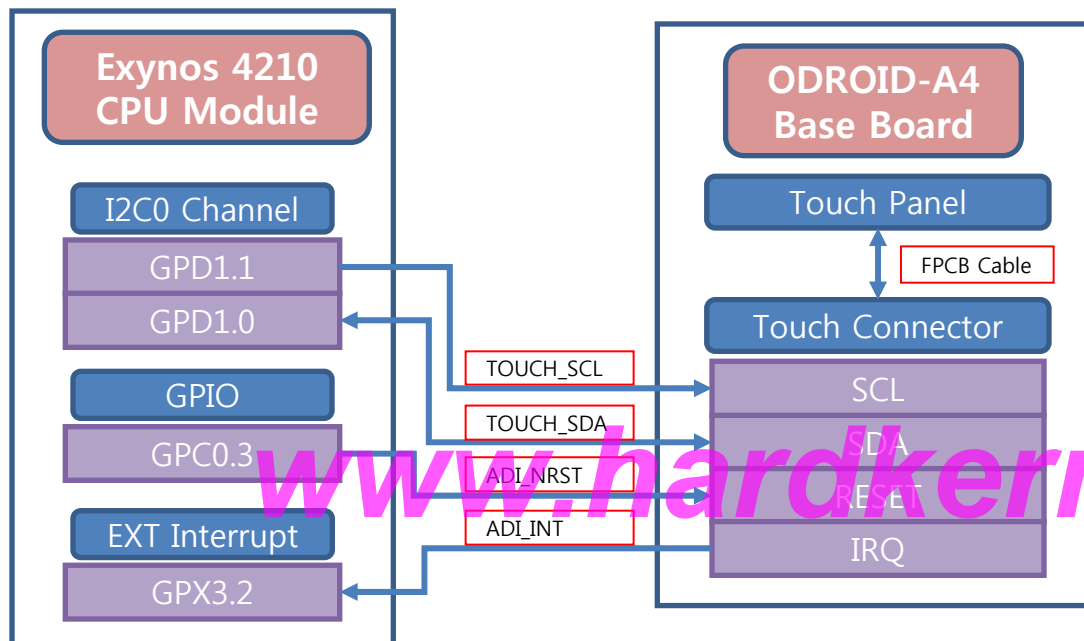
Touchscreen Data Flow

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Touchscreen Hardware Interface

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Touchscreen Controller I2C Specification

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

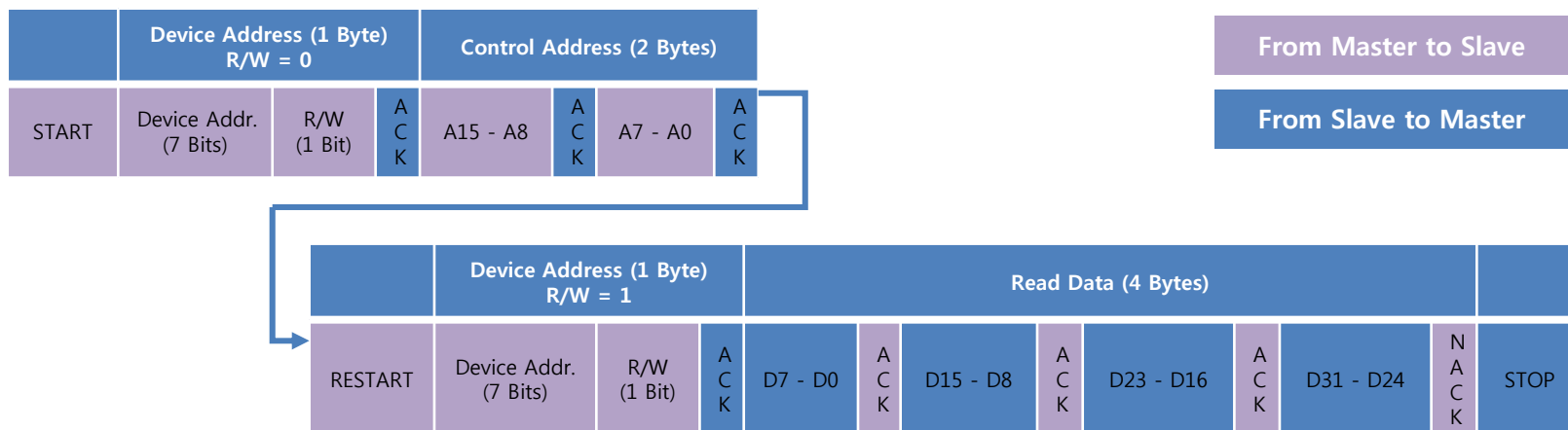
I2C Controller Specification	
I2C Slave Address	0x28
Interface	I2C 100Khz
Resolution	2048 X 2048
Report Rate	Up to 120Khz

Pin Assignments		
Name	I/O	Description
FRAME_DONE	O	Interrupt
SDA	I/O	I2C Data
SCL	I	I2C Clock

I2C Write Operation



I2C Read Operation



Touchscreen Controller Register

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Command Register

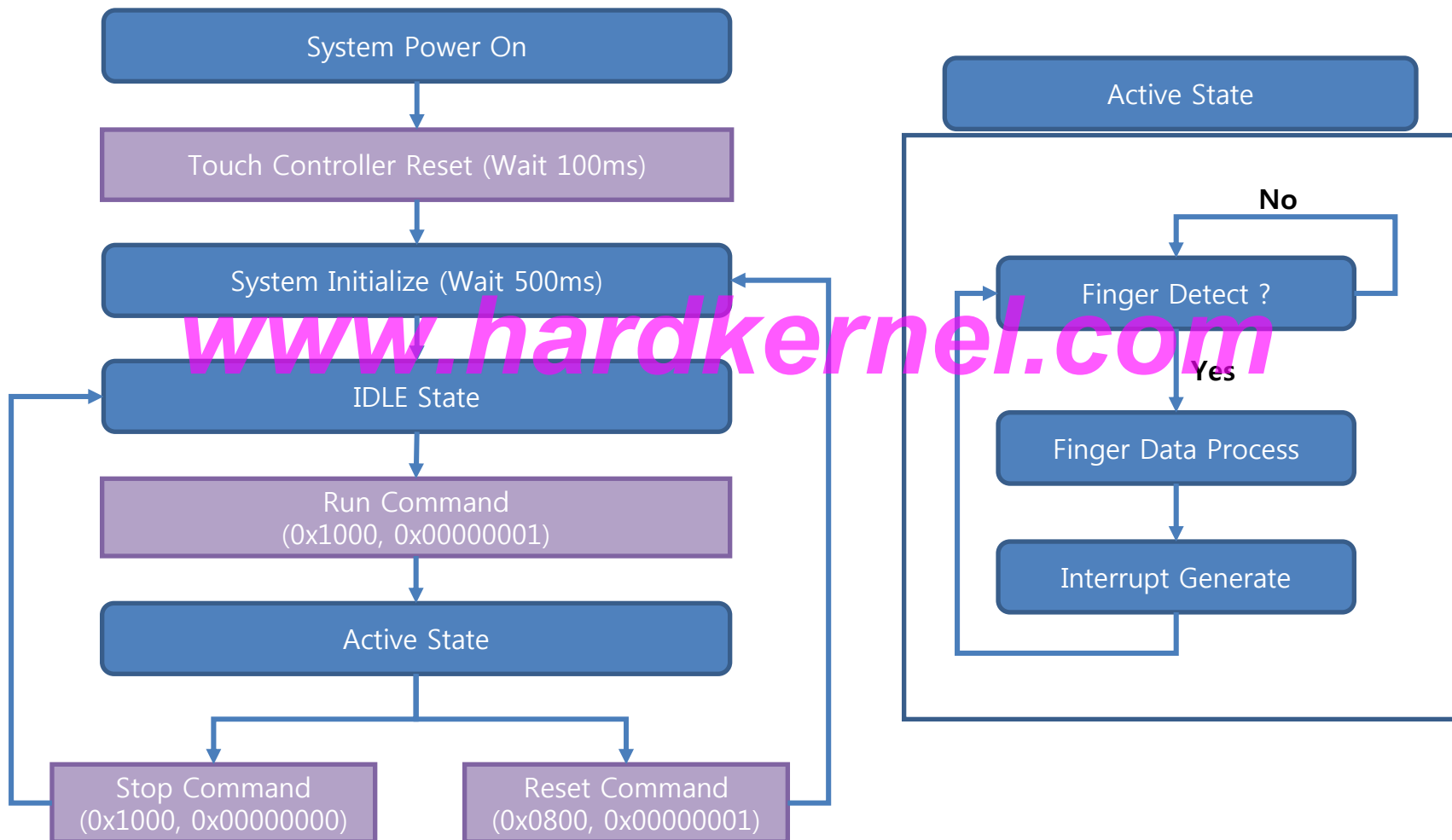
Name	Description	Address (2 Bytes)	Value (4 Bytes)
Reset	Touch Controller Reset(IDLE State)	0x0800	0x00000001
Run	Touch Operation Start(IDLE -> ACTIVE State)	0x1000	0x00000001
Stop	Touch Operation Stop(ACTIVE -> IDLE State)	0x1000	0x00000000
Charger On	Entering charger mode	0x1800	0x00000001
Charger Off	Exit charger mode	0x1800	0x00000000
Sleep	Entering low power mode	0x2000	0x00000001
Wake Up	Exit low power mode	0x2000	0x00000000
Recalibration	Touch panel recalibration	0x2800	0x00000001

Data Register

Name	Description	Address (2 Bytes)	Value (4 Bytes)
Touch Count	Touch key status, Touch point count	0xFF00	Key status[D15-D8], Touch count [D4-D0]
AXIS 1	Finger 1 status & data	0xFF04	Touch status[D28-D27], X[D21-D11], y[D10-D0]
AXIS 2	Finger 2 status & data	0xFF08	Touch status[D28-D27], X[D21-D11], y[D10-D0]
...
AXIS 10	Finger 10 status & data	0xFF28	Touch status[D28-D27], X[D21-D11], y[D10-D0]
Extension 1	Finger 1 extension data	0xFF40	Touch area[D10-D8], Touch Pressure[D7-D0]
Extension 2	Finger 2 extension data	0xFF44	Touch area[D10-D8], Touch Pressure[D7-D0]
...
Extension 10	Finger 10 extension data	0xFF64	Touch area[D10-D8], Touch Pressure[D7-D0]

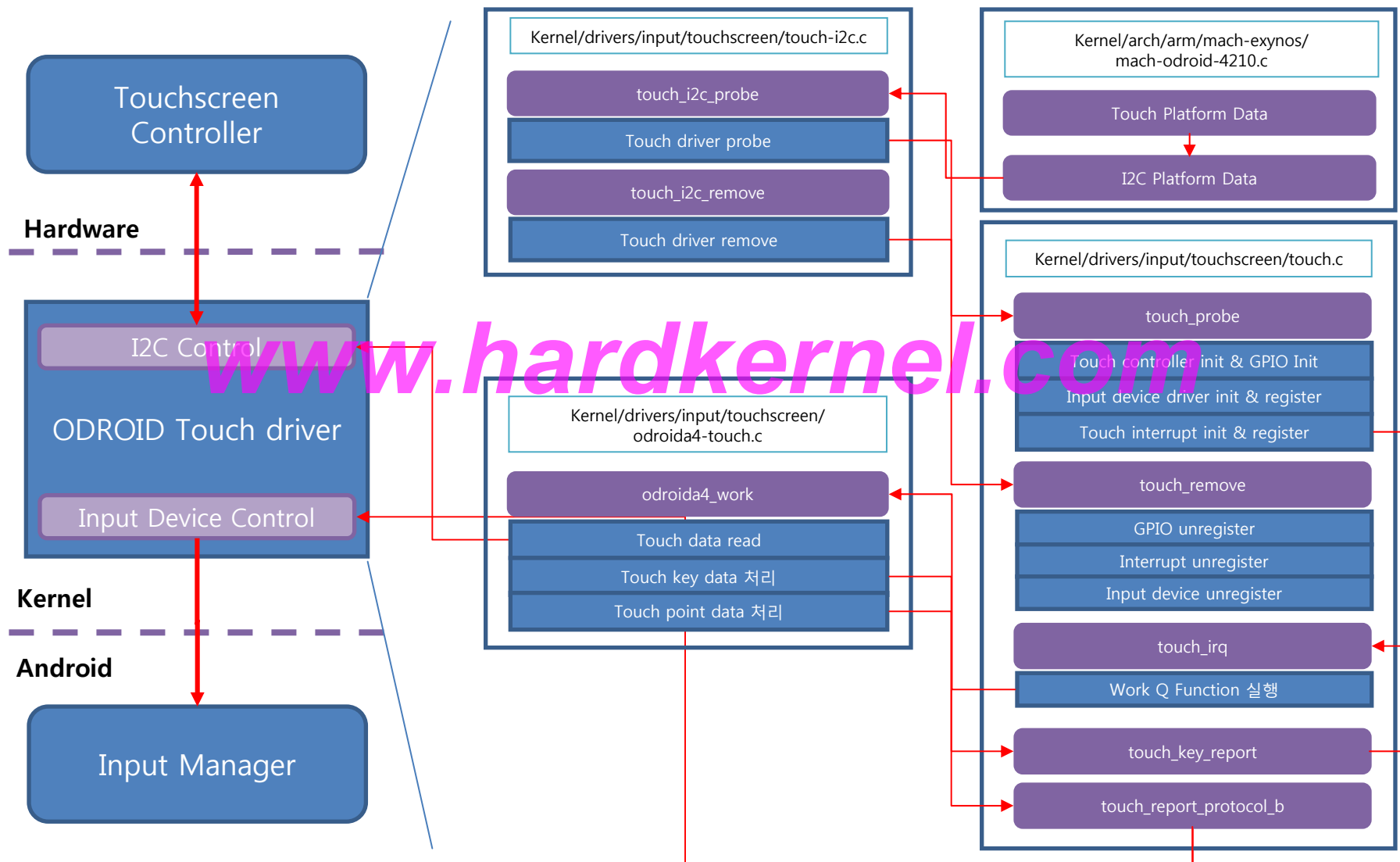
Touchscreen Controller Running Process

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Touch 드라이버 구조

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Touchscreen Platform data 설정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Odroid-A4 Touchscreen Platform data 설정

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
#include <linux/input/touch-pdata.h>
#include <linux/input/odroida4-touch.h>
```

```
static int odroida4_keycode[] = {
    KEY_HOME, KEY_MENU, KEY_BACK, KEY_SEARCH
};
```

```
static struct touch_pdata odroida4_touch_pdata = {
    .name = "odroida4-ts", /* input drv name */
    .irq_gpio = EXYNOS4_GPX3(2), /* irq gpio define */
    .reset_gpio = EXYNOS4_GPC0(3), /* reset gpio define */
    .reset_level = 1,
    .irq_mode = IRQ_MODE_NORMAL
    .irq_flags = IRQF_TRIGGER_FALLING | IRQF_DISABLED,
```

```
.abs_max_x = 480,
.abs_max_y = 800,
.area_max = 10,
.press_max = 100,
```

```
.id_max = 10 + 1,
.id_min = 1,
```

```
.vendor = 0x16B4,
.product = 0x0702,
.version = 0x0001,
```

```
.max_fingers = 10,
```

```
.keycode = odroida4_keycode,
.keycnt = 4,
```

Touch Key code table 설정

Touch H/W control Pin 설정
Interrupt mode 및 flag 설정

Touch controller
Report max data 설정

I2C 채널 0에 연결되어 있는
Device 정보 설정

```
//-----
// Control function
//-----
.touch_work = odroida4_work,
.enable = odroida4_enable,
.disable = odroida4_disable,
.early_probe = odroida4_early_probe,
.probe = odroida4_probe,
//-----
// I2C control function
//-----
.i2c_write = odroida4_i2c_write,
.i2c_read = odroida4_i2c_read,
//-----
// Calibration function
//-----
.calibration = odroida4_calibration,
//-----
// Firmware update control function
//-----
.fw_filename = "aim902_fw.bin",
.fw_filesize = (1024 * 8), // 8K bytes
.input_open = odroida4_input_open,
.flash_firmware = odroida4_flash_firmware,
};

#endif

...
static struct i2c_board_info i2c_devs0[] __initdata = {
    ...
    #if defined(CONFIG_TOUCHSCREEN_AIMS902)
        I2C_BOARD_INFO(I2C_TOUCH_NAME, (0x28)),
        .platform_data = &odroida4_touch_pdata,
    },
    #endif
    ...
};
...
```

Touch control function 설정

I2C Slave Address 등록
사용되어질 Platform data 등록

Touchscreen driver probe(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch driver probe

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
static struct i2c_board_info i2c_devs0[] __initdata = {
...
    #if defined(CONFIG_TOUCHSCREEN_AIM902)
    {
        I2C_BOARD_INFO(I2C_TOUCH_NAME, (0x28)),
        .platform_data = &odroida4_touch_pdata,
    },
    #endif
...
};
...
```

I2C 채널 0번에 연결되어 있는 Device의 정보를 기록한다.

Kernel/include/linux/input/touch-pdata.h

```
...
#ifdef CONFIG_HAS_EARLYSUSPEND
#include <linux/earllysuspend.h>
#endif

#include <linux/interrupt.h>

#define I2C_TOUCH_NAME "odroid-ts"
#define I2C_SEND_MAX_SIZE 512
...
```

Platform Driver 등록 이름과 Device Driver의 이름이 같아야 동작한다.

Kernel/drivers/input/touchscreen/touch-i2c.c

```
...
static const struct i2c_device_id touch_id[] = {
    { I2C_TOUCH_NAME, 0 },
};

MODULE_DEVICE_TABLE(i2c, touch_id);

static struct i2c_driver touch_i2c_driver = {
    .driver = {
        .name = I2C_TOUCH_NAME,
        .owner = THIS_MODULE,
    },
    .probe = touch_i2c_probe,
    .remove = __devexit_p(touch_i2c_remove),
    .suspend = touch_i2c_suspend,
    .resume = touch_i2c_resume,
    .id_table = touch_id,
};
```

I2C Driver의 ID정보를 설정한다. Platform driver register시 해당 ID를 검사한다.

```
...
static int __devinit touch_i2c_probe
(struct i2c_client *client, const struct i2c_device_id
 *id)
{
    return touch_probe(client);
}
...
```

Device Driver가 Register되면 I2c driver struct에 설정되어진 Probe함수를 호출한다.

Kernel/drivers/input/touchscreen/touch.c

```
...
int touch_probe(struct i2c_client *client)
{
    int rc = -1;
    struct device *dev = &client->dev;
    struct touch *ts;
    ...
}
```

Touchscreen driver probe(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch controller init & GPIO Init

Kernel/drivers/input/touchscreen/touch.c

```
...
void touch_hw_reset (struct touch *ts)
{
    if(ts->pdata->reset_gpio) {
        if(gpio_request(ts->pdata->reset_gpio, "touch reset")) {
            printk("%s : request port error!\n", "touch reset");
        }
        else {
            gpio_direction_output(ts->pdata->reset_gpio,
                                ts->pdata->reset_level ? 0 : 1);
            mdelay(10);
            gpio_direction_output(ts->pdata->reset_gpio,
                                ts->pdata->reset_level ? 1 : 0);
            mdelay(10);
            gpio_direction_output(ts->pdata->reset_gpio,
                                ts->pdata->reset_level ? 0 : 1);
            mdelay(10);
        }
    }
}

...
int touch_probe (struct i2c_client *client)
{
    ...
    if(ts->pdata->reset_gpio) touch_hw_reset(ts);

    if(ts->pdata->early_probe) {
        if((rc = ts->pdata->early_probe(ts)) < 0)
            goto err_free_mem;
    }
    ...
}
```

Platform data에 reset관련 Port가 정의 되어있다면 H/W Reset을 실행한다.

Platform data에 early_probe 함수가 등록되어 있다면 등록되어진 함수를 실행한다.

Kernel/drivers/input/touchscreen/odroida4-touch.c

```
...
int odroida4_early_probe (struct touch *ts)
{
    unsigned short cmd ;
    unsigned int wdata;

    INIT_DELAYED_WORK_DEFERRABLE(&ts->filter_dwork,
                                filter_dwork);

    mdelay(100);

    cmd = REG_CMD_RESET;
    wdata = 0x00000001;
    ts->pdata->i2c_write(ts->client,
                      (unsigned char *)&cmd, sizeof(cmd),
                      (unsigned char *)&wdata, sizeof(wdata));

    mdelay(500);

    // controller on command send
    cmd = REG_CMD_STOP_RUN;
    wdata = 0x00000001;
    if(ts->pdata->i2c_write(ts->client,
                      (unsigned char *)&cmd, sizeof(cmd),
                      (unsigned char *)&wdata, sizeof(wdata)) < 0)

        return -1;

    mdelay(100);
    return 0;
}

...
```

Touch Controller에 Command를 전송하여 Touch Controller의 상태를 Active상태로 전환한다.

Touchscreen driver probe(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Input device driver init & register

Kernel/drivers/input/touchscreen/touch.c

```
...
int touch_probe (struct i2c_client *client)
{
    ...
    if(!(ts->input = input_allocate_device()))
        goto err_free_mem;

    snprintf(ts->phys, sizeof(ts->phys),
             "%s/input0", ts->pdata->name);

    if(ts->pdata->input_open) ts->input->open =
        touch_input_open;
    else
        ts->input->open = ts->pdata->input_open;

    if(ts->pdata->input_close) ts->input->close = touch_input_close;
    else
        ts->input->close = ts->pdata->input_close;

    ts->input->name = ts->pdata->name;
    ts->input->phys = ts->phys;
    ts->input->dev.parent = dev;
    ts->input->id.bustype = BUS_I2C;

    ts->input->id.vendor = ts->pdata->vendor;
    ts->input->id.product = ts->pdata->product;
    ts->input->id.version = ts->pdata->version;

    set_bit(EV_SYN, ts->input->evbit);
    set_bit(EV_ABS, ts->input->evbit);
    ...
}
```

Input device driver
Memory allocation

Platform data에 설정된 값으로
Input driver 설정

Touch data report type설정

```
...
if(ts->pdata->keycode) {
    int key;
    set_bit(EV_KEY, ts->input->evbit);
    for(key = 0; key < ts->pdata->keycnt; key++) {
        if(ts->pdata->keycode[key] <= 0) continue;
        set_bit(ts->pdata->keycode[key] & KEY_MAX,
                ts->input->keybit);
    }
}
input_set_drvdata(ts->input, ts);

input_set_abs_params(ts->input, ABS_MT_POSITION_X,
                    ts->pdata->abs_min_x, ts->pdata->abs_max_x, 0, 0);
input_set_abs_params(ts->input, ABS_MT_POSITION_Y,
                    ts->pdata->abs_min_y, ts->pdata->abs_max_y, 0, 0);

if(ts->pdata->area_max)
    input_set_abs_params(ts->input, ABS_MT_TOUCH_MAJOR,
                        ts->pdata->area_min, ts->pdata->area_max, 0, 0);

if(ts->pdata->press_max)
    input_set_abs_params(ts->input, ABS_MT_PRESSURE,
                        ts->pdata->press_min, ts->pdata->press_max, 0, 0);

if(ts->pdata->id_max) {
    input_set_abs_params(ts->input, ABS_MT_TRACKING_ID,
                        ts->pdata->id_min, ts->pdata->id_max, 0, 0);
    input_mt_init_slots(ts->input, ts->pdata->max_fingers);
}

if ((rc = input_register_device(ts->input))) {
    dev_err(dev, "(%s) input register fail!\n", ts->input->name);
    goto err_free_input_mem;
}
...
}
```

Touch key data report type설정

Platform data를 참조하여
Report되어질 Keycode Enable

Touch Input driver register

Touchscreen driver probe(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch interrupt init & register

Kernel/drivers/input/touchscreen/touch.c

```
...
int touch_probe(struct i2c_client *client)
{
...
    ts->client = client;
    ts->pdata = client->dev.platform_data;

    if(ts->pdata->irq_gpio)
        ts->irq = gpio_to_irq(ts->pdata->irq_gpio);
...
    if(ts->irq) {
        switch(ts->pdata->irq_mode) {
...
        case IRQ_MODE_NORMAL:
            INIT_WORK(&ts->work, touch_work_q);
            if((ts->work_queue =
                create_singlethread_workqueue("work_queue")) == NULL)
                goto err_free_input_mem;

            if((rc = request_irq(ts->irq, ts->pdata->irq_func,
                                ts->pdata->irq_flags, ts->pdata->name, ts))) {
                printk("irq %d request fail!\n", ts->irq);
                goto err_free_input_mem;
            }
            break;
...
}
```

Touch Driver struct 초기화
I2C의 Platform Data를
Touch Platform Data로 사용

GPIO번호를 IRQ번호로 변경

www.hardkernel.com

Touch Interrupt Register

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
#if defined(CONFIG_TOUCHSCREEN_AIMS902)
#include <linux/input/touch-pdata.h>
#include <linux/input/odroida4-touch.h>

static int odroida4_keycode[] = {
    KEY_HOME, KEY_MENU, KEY_BACK, KEY_SEARCH
};

static struct touch_pdata odroida4_touch_pdata = {
    .name = "odroida4-ts", /* input drv name */
    .irq_gpio = EXYNOS4_GPX3(2), /* irq gpio define */
    .reset_gpio = EXYNOS4_GPC0(3), /* reset gpio define */
    .reset_level = 1,
    .irq_mode = IRQ_MODE_NORMAL
    .irq_flags = IRQF_TRIGGER_FALLING | IRQF_DISABLED
...
}
```

IRQ Pin의 상태가 High에서 Low로 변경되는 시점에
등록되어진 IRQ function 실행.
IRQ function 실행 중 다른 IRQ는 Disable되어짐.

Work Q 초기화 및 Work Q function 등록.
IRQ handler안에서 I2C Control function을
사용하지 못하기 때문에 Work Q를 사용한다.

Touchscreen data report(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch key data 처리

Kernel/drivers/input/touchscreen/odroida4-touch.c

```
...
void odroida4_work (struct touch *ts)
{
    status_reg_u status;
    data_reg_u data;
    ext_data_reg_u ext_data;
    button_u button;

    ...

    cmd = REG_TS_STATUS;
    ts->pdata->i2c_read(ts->client, &cmd, sizeof(cmd),
        &status, sizeof(status_reg_u));

    ...

    button.bits.bt0_press = (status.bits.button & 0x01) ? 1:0;
    button.bits.bt1_press = (status.bits.button & 0x04) ? 1:0;
    button.bits.bt2_press = (status.bits.button & 0x10) ? 1:0;
    button.bits.bt3_press = (status.bits.button & 0x40) ? 1:0;

    ts->pdata->key_report(ts, button.ubyte);
    ...
}
```

Touchscreen controller의 status register를 읽어온다.

Platform data에 설정된 key report 함수 실행

Kernel/include/linux/input/odroida4-touch.h

reserved2[31:16] button[15:8] reserved1[7:5] ts_cnt[4:0]

R3 P3 R2 P2 R1 P1 R0 P0

Key Release Status (1 = true)

Key Press Status (1 = true)

Kernel/drivers/input/touchscreen/touch.c

```
...
static void touch_key_report (struct touch *ts, unsigned char
    button_data)
{
    static button_t button_old;
    button_t button_new;

    button_new.ubyte = button_data;

    if(button_old.ubyte != button_new.ubyte) {
        if((button_old.bits.bt0_press != button_new.bits.bt0_press) &&
            (ts->pdata->keycmd > 0)) {
            if(button_new.bits.bt0_press)
                input_report_key(ts->input, ts->pdata->keycode[0], true);
            else
                input_report_key(ts->input, ts->pdata->keycode[0], false);
        }

        button_old.ubyte = button_new.ubyte;
    }
    ...
}
```

변경되었던 key code값 report

Kernel/include/linux/input/touch-pdata.h

P7 P6 P5 P4 P3 P2 P1 P0

Touchscreen data report(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch data 처리

Kernel/drivers/input/touchscreen/odroida4-touch.c

```
...
void odroida4_work (struct touch *ts)
{
    status_reg u    status;
    data_reg_u      data;
    ext_data_reg_u  ext_data;
    button_u        button;

    ...
    for(cnt = 0; cnt < status.bits.ts_cnt; cnt++) {
        cmd = REG_TS_DATA(cnt);
        ts->pdata->i2c_read(ts->client, &cmd, sizeof(cmd),
                           &data.uint, sizeof(data_reg_u));
        cmd = REG_TS_EXT_DATA(cnt);
        ts->pdata->i2c_read(ts->client, &cmd, sizeof(cmd),
                           &ext_data.uint, sizeof(ext_data_reg_u));

        ...

        if((find_slot = odroida4_id_tracking(ts, data.bits.id)) == 0xFF)
            continue;
        if((data.bits.type == TS_EVENT_MOVE) ||
           (data.bits.type == TS_EVENT_PRESS)) {
            ts->finger[find_slot].status = true;
            ts->finger[find_slot].event = TS_EVENT_MOVE;
            ts->finger[find_slot].id = data.bits.id;
            ts->finger[find_slot].x = data.bits.x;
            ts->finger[find_slot].y = data.bits.y;
            ts->finger[find_slot].area = ext_data.bits.area;
            ts->finger[find_slot].pressure = ext_data.bits.pressure;

            ...

            ts->pdata->report(ts);
        }
    }
    ...
}
```

Touchscreen control의
Data register를 읽어온다.

Platform data에 설정된
Touch report 함수 실행

Kernel/drivers/input/touchscreen/touch.c

```
...
static void touch_report_protocol_b (struct touch *ts)
{
    ...

    if((ts->finger[id].event == TS_EVENT_UNKNOWN) || (ts->finger[id].status == false))
        continue;

    input_mt_slot(ts->input, id);    ts->finger[id].status = false;

    if(ts->finger[id].event != TS_EVENT_RELEASE) {
        input_report_abs(ts->input, ABS_MT_TRACKING_ID, ts->finger[id].id);
        input_report_abs(ts->input, ABS_MT_POSITION_X, ts->finger[id].x);
        input_report_abs(ts->input, ABS_MT_POSITION_Y, ts->finger[id].y);

        if(ts->pdata->area_max)
            input_report_abs(ts->input, ABS_MT_TOUCH_MAJOR, ts->finger[id].area);
        if(ts->pdata->press_max)
            input_report_abs(ts->input, ABS_MT_PRESSURE, ts->finger[id].pressure);

        ...

        input_sync(ts->input);
    }
    ...
}
```

Touch Data를 report 한다.

Kernel/include/linux/input/odroida4-touch.h

reserved[31:29]	type[28:27]	id[26:22]	y[21:11]	x[10:0]
delta_y[31:24]	delta_x[23:16]	reserved[15:11]	area[10:8]	pressure[7:0]

Kernel/include/linux/input/touch-pdata.h

status	event	id	x	y	area	pressure
--------	-------	----	---	---	------	----------

Touchscreen driver remove

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Touch driver remove

Kernel/drivers/input/touchscreen/touch-i2c.c

```
...
static int __devexit touch_i2c_remove(struct i2c_client *client)
{
    return touch_remove(&client->dev);
}
...
```

I2C device driver unregister

Kernel/drivers/input/touchscreen/touch.c

```
...
int touch_remove (struct device *dev)
{
    struct touch *ts = dev_get_drvdata(dev);
    if(ts->irq) free_irq(ts->irq, ts);
    if(ts->pdata->reset_gpio) gpio_free(ts->pdata->reset_gpio);
    touch_sysfs_remove(dev);
    input_unregister_device(ts->input);
    dev_set_drvdata(dev, NULL);
    kfree(ts);
    return 0;
}
...
```

Interrupt unregister

GPIO unregister

Input device driver unregister

www.hardkernel.com

Reference(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

I2C Control API (include/linux/i2c.h)

Function

int i2c_transfer(struct i2c_adpater *adap, struct i2c_msg *msgs, int num);

Description

Restart를 포함한 연속적인 메시지를 전송

Return

에러인 경우 Negative value를 return, 유효한 경우 전송되어진 message count를 되돌려 준다.

Usage

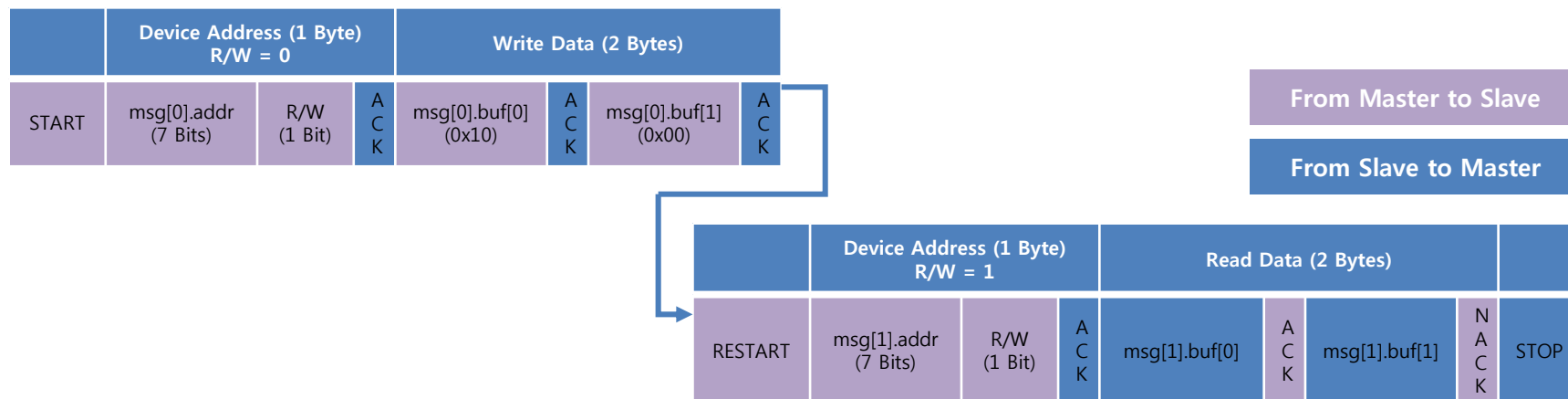
```
struct i2c_msg msg[2];
```

```
unsigned char cmd[2] = { 0x10, 0x00 }, rd[2];
```

```
msg[0].addr = client->addr; msg[0].flags = 0; msg[0].len = 2; msg[0].buf = cmd;
```

```
msg[1].addr = client->addr; msg[1].flags = I2C_M_RD; msg[1].len = 2; msg[1].buf = rd;
```

```
ret = i2c_transfer(client->adapter, msg, 2);
```



Reference(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

I2C Control API (include/linux/i2c.h)

Function

int i2c_master_send(const struct i2c_client *client, const char *buf, int count);

Description

1개의 메세지 전송. 연속적인 데이터 write.

Return

에러인 경우 Negative value를 return, 유효한 경우 전송되어진 message count를 되돌려 준다.

Usage

```
unsigned char wdata[6] = { 0x10, 0x00, 0x01, 0x02, 0x03, 0x04 };
```

```
ret = i2c_master_send(client, wdata, 6);
```

www.hardkernel.com

From Master to Slave

From Slave to Master

	Device Address (1 Bytes) R/W = 0			Write Data (6 Bytes)												
START	client->addr (7 Bits)	R/W (1 Bit)	A C K	wdata[0] (0x10)	A C K	wdata[1] (0x00)	A C K	wdata[2] (0x01)	A C K	wdata[3] (0x02)	A C K	wdata[4] (0x03)	A C K	wdata[5] (0x04)	A C K	STOP

- 더 많은 정보는 kernel/drivers/i2c/i2c-core.c 파일의 내용을 참조

Reference(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

IRQ Control API (include/linux/interrupt.h)

Function

```
int request_irq(unsigned int irq, irqreturn_t (*handler)(int, void *, struct pt_regs *),
                unsigned long irqflags, const char *devname, void *dev_id);
```

Description

irq로 지정되어진 line의 상태가 irqflags에 지정되어진 상태로 변경되면 *dev_id를 가지고 irq handler를 실행한다.

Return

에러인 경우 0이 아닌 값을 return, 유효한 경우 0을 return.

Usage

```
if (request_irq(irqnum, my_interrupt, , "my_irq", dev)) {
    printk(KERN_ERR "my_device: cannot register IRQ %d\n", irqnum);    return -EIO;
}
```

Function

```
void free_irq(unsigned int irq, void *dev_id);
```

Description

irq로 지정되어진 line의 irq handler를 제거한다.

Return

에러인 경우 0이 아닌 값을 return, 유효한 경우 0을 return.

Usage

```
free_irq(irqnum, dev);
```

- 더 많은 정보는 Kernel/include/linux/interrupt.h 파일의 내용을 참조

Reference(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

IRQ Control 예제

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/irq.h>
#include <linux/interrupt.h>
#include <mach/gpio.h>
#include <mach/regs-gpio.h>
#include <plat/gpio-cfg.h>

irqreturn_t my_irq(int irq, void *handle) // Interrupt callback function
{
    printk("%s\n", __func__);
    return IRQ_HANDLED;
}

int init_module(void)
{
    unsigned int irqnum = gpio_to_irq(EXYNOS4_GPX3(0));
    unsigned long irqflags = IRQF_TRIGGER_FALLING | IRQF_DISABLED;

    if(request_irq(irqnum, my_irq, irqflags, "my irq", NULL)) {
        printk(KERN_ERR "my_device: cannot register IRQ %d\n", irqnum); return -EIO;
    }
    return 0;
}
```

Interrupt flags의 정보는
Kernel/include/linux/interrupt.h 파일 참조

Reference(5)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Multi Touch Protocol(Type-A, Type-B)

- Touch Device가 생성하는 데이터의 종류에 따라서 Type A 또는 Type B의 Report Protocol을 가질 수 있다.
- Touch Point의 식별가능 인자를 Touch Device에서 제공하지 않는 경우 Protocol-A Type을 사용한다.
- Touch Point의 식별가능 인자를 Touch Device에서 제공하는 경우 Protocol-B Type을 사용한다.



- 더 많은 정보는 [Kernel/Documentation/input/multi-touch-protocol.txt](#)를 참조

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Touchscreen 드라이버를 Timer Interrupt 방식으로 변환 하기

Timer 변수 선언 및 초기화

Kernel/include/linux/input/touch-pdata.h

```
...
#include <linux/hrtimer.h>
#include <linux/interrupt.h>
...

struct touch {
...
    struct workqueue_struct *work_queue;
    struct work_struct work;
    struct hrtimer timer;
};
...
```

Hrtimer를 사용하기 위한 Header file을 등록한다.

사용될 timer 변수를 선언한다.

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
static struct touch_pdata odroida4_touch_pdata = {
    .name = "odroida4-ts",
    .irq_gpio = EXYNOS4_GPX3(2),
    .reset_gpio = EXYNOS4_GPC0(3),
    .reset_level = 1,
    .irq_mode = IRQ_MODE_POLLING,,
    .irq_flags = 10000000, // Timer interval(10ms)
};
...
```

Kernel/drivers/input/touchscreen/touch.c

```
int touch_probe(struct i2c_client *client)
{
...
    if(ts->irq) {
        switch(ts->pdata->irq_mode) {
...
            case IRQ_MODE_POLLING:
                INIT_WORK(&ts->work, touch_work_q);
                if((ts->work_queue =
                    create_singlethread_workqueue("work_queue")) == NULL)
                    goto err_free_input_mem;

                hrtimer_init(&ts->timer, CLOCK_MONOTONIC,
                    HRTIMER_MODE_REL);
                ts->timer.function = touch_timer;

                break;
            if(ts->pdata->irq_mode != IRQ_MODE_POLLING)
                disable_irq_nosync(ts->irq);
        }
    }
...
}
```

Timer callback function에서 사용될 Work Q를 초기화 한다.

INIT_WORK(&ts->work, touch_work_q);
if((ts->work_queue = create_singlethread_workqueue("work_queue")) == NULL)
goto err_free_input_mem;

hrtimer_init(&ts->timer, CLOCK_MONOTONIC, HRTIMER_MODE_REL);
ts->timer.function = touch_timer;

IRQ_MODE_POLLING mode에서는 Interrupt 등록을 하지 않으므로 Interrupt control 함수가 실행되지 않도록 한다.

Interrupt Mode를 POLLING mode(Timer mode)로 설정하고 irq_flags 변수값을 Timer Refresh 값으로 등록한다.

선언되어진 timer변수를 초기화하고 Timer callback function을 등록한다.

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Timer callback function 작성

Kernel/drivers/input/touchscreen/touch.c

```
...
#include <linux/hrtimer.h>

...
static enum hrtimer_restart touch_timer(struct hrtimer *timer)
{
    struct touch *ts = container_of(timer, struct touch, timer);

    queue_work(ts->work_queue, &ts->work);

    hrtimer_start(&ts->timer,
        ktime_set(0, ts->pdata->irq_flags),
        HRTIMER_MODE_REL);

    return HRTIMER_NORESTART;
}
...
```

초기화에서 등록되어진 Work Q를 실행한다.

Platform Data에 설정되어진 Timer refresh값으로 재 시작한다.

현재 동작중인 Timer 동작을 정지한다.

Interrupt control function 수정

Kernel/drivers/input/touchscreen/odroida4-touch.c

```
...
void odroida4_enable(struct touch *ts)
{
    ...
    if(ts->disabled) {
        ...
        odroida4_calibration(ts);

        if(ts->pdata->irq_mode != IRQ_MODE_POLLING) enable_irq(ts->irq);
        else
            hrtimer_start(&ts->timer,
                ktime_set(0, ts->pdata->irq_flags),
                HRTIMER_MODE_REL);

        ts->disabled = false;
    }
    ...
}

Void odroida4_disable(struct touch *ts)
{
    ...
    if(!ts->disabled) {
        if(ts->pdata->irq_mode != IRQ_MODE_POLLING) disable_irq(ts->irq);
        else
            hrtimer_cancel(&ts->timer);

        ts->disabled = true;
    }
    ...
}
```

IRQ_MODE_POLLING mode에서는 Interrupt 등록을 하지 않으므로 Interrupt control 함수가 실행되지 않도록 한다.

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Debugging을 위한 Kernel Message 추가

Kernel/drivers/input/touchscreen/touch.c

```
...
static void touch_key_report(struct touch *ts, unsigned char button_data)
{
    static button_u button_old;
    button_u button_new;

    button_new.ubyte = button_data;
    if(button_old.ubyte != button_new.ubyte) {
        if((button_old.bits.bt0_press != button_new.bits.bt0_press) &&
           (ts->pdata->keycnt > 0)) {
            printk("keycode[0] (0x%04X) %s\n",
                  ts->pdata->keycode[0],
                  button_new.bits.bt0_press ? "press":"release");
        }
        ...
    }
    if((button_old.bits.bt7_press != button_new.bits.bt7_press) &&
       (ts->pdata->keycnt > 7)) {
        printk("keycode[7] (0x%04X) %s\n",
              ts->pdata->keycode[7],
              button_new.bits.bt7_press ? "press":"release");
    }
    ...
}
button_old.ubyte = button_new.ubyte;
}
```

각각의 keycode에 대한 Kernel Debug Message를 추가한다.

Kernel/drivers/input/touchscreen/touch.c

```
...
static void touch_report_protocol_b(struct touch *ts)
{
    int id;

    for(id = 0; id < ts->pdata->max_fingers; id++) {
        ...
        input_mt_slot(ts->input, id);    ts->finger[id].status = false;

        if(ts->finger[id].event != TS_EVENT_RELEASE) {
            input_report_abs(ts->input, ABS_MT_TRACKING_ID, ts->finger[id].id);
            input_report_abs(ts->input, ABS_MT_POSITION_X, ts->finger[id].x);
            input_report_abs(ts->input, ABS_MT_POSITION_Y, ts->finger[id].y);

            printk("touch press : slot = %d, id = %d, x = %d, y = %d\n",
                  id, ts->finger[id].id, ts->finger[id].x, ts->finger[id].y);
        }
        ...
    }
    else {
        ts->finger[id].event = TS_EVENT_UNKNOWN;
        input_report_abs(ts->input, ABS_MT_TRACKING_ID, -1);

        printk("touch release : slot = %d, id = %d\n",
              id, ts->finger[id].id);
    }
    input_sync(ts->input);
}
```

새롭게 눌러지거나 이동한 Touch Point의 정보를 표시함

떨어진 Touch Point 정보를 표시함

Lab/Exam(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Kernel Message를 통하여 Touch Driver 동작 확인

- Host PC (Linux)에서 수정되어진 Kernel build
- Uboot의 fastboot를 이용하여 새로 Build되어진 Kernel image(zImage) download
- Kernel booting 후 Touch Driver에 대한 kernel message 확인

```

Odroid - Xshell 4 (Free for Home/School)
[ 2.545278] usbcore: registered new interface driver hanwang
[ 2.551091] usbcore: registered new interface driver kbtabs
[ 2.556453] kbtabs: v0.0.2:USB KB Gear JamStudio Tablet driver
[ 2.562288] usbcore: registered new interface driver wacom
[ 2.567642] wacom: v1.52:USB Wacom tablet driver
[ 3.602163] input: odroida4-ts as /devices/platform/s3c2440-i2c.0/i2c-0/0-003
[ 3.605887] -----
[ 3.611206] TOUCH SCREEN INFORMATION
[ 3.615716] -----
[ 3.622048] TOUCH INPUT Name = odroida4-ts
[ 3.626124] TOUCH IRQ Mode = IRQ_MODE_POLLING
[ 3.630635] TOUCH F/W version = 2.10
[ 3.634177] TOUCH FINGERS MAX = 10
[ 3.637574] TOUCH ABS X MAX = 480, TOUCH ABS X MIN = 0
[ 3.642691] TOUCH ABS Y MAX = 800, TOUCH ABS Y MIN = 0
[ 3.647810] TOUCH MAJOR MAX = 10, TOUCH MAJOR MIN = 0
[ 3.652841] TOUCH PRESS MAX = 100, TOUCH PRESS MIN = 0
[ 3.657958] TOUCH ID MAX = 11, TOUCH ID MIN = 1
[ 3.662467] Multi-Touch Protocol-B Used.
[ 3.666371] H/W Reset function implemented
[ 3.670447] Early-suspend function implemented
[ 3.674856] Firmware update function(udev control) implemented
[ 3.680683] Calibration function implemented
[ 3.684918] -----
[ 3.691374] i2c-core: driver [odroid-ts] using legacy suspend method
[ 3.697598] i2c-core: driver [odroid-ts] using legacy resume method
[ 3.704598] ADUX1001 VERSION - [0x02]
[ 3.708319] get_calibdata: 1435
[ 3.775724] store_calibdata: 1442
[ 3.777220] input: adux1001 as /devices/platform/s3c2440-i2c.1/i2c-1/1-0014/4

```

Driver IRQ Mode상태를 확인한다.

```

Odroid - Xshell 4 (Free for Home/School)
[ 123.044886] touch press : slot = 0, id = 1, x = 335, y = 401
[ 123.055096] touch press : slot = 0, id = 1, x = 344, y = 424
[ 123.065128] touch press : slot = 0, id = 1, x = 344, y = 424
[ 123.075180] touch release : slot = 0, id = 1
[ 126.172282] keycode[0] (0x0066) press
[ 126.263083] keycode[0] (0x0066) release
[ 128.891423] keycode[0] (0x0066) press
[ 128.982388] keycode[0] (0x0066) release
[ 132.702656] keycode[3] (0x0009) press
[ 132.742954] keycode[3] (0x0009) release
[ 132.772851] keycode[3] (0x0009) press
[ 132.792913] keycode[3] (0x0009) release
[ 134.815631] keycode[2] (0x009E) press
[ 134.906415] keycode[2] (0x009E) release
[ 135.408206] keycode[1] (0x0088) press
[ 135.508990] keycode[1] (0x0088) release
[ 136.011878] keycode[0] (0x0066) press
[ 136.052352] keycode[0] (0x0066) release
[ 141.389020] touch press : slot = 0, id = 1, x = 362, y = 447
[ 141.399156] touch press : slot = 0, id = 1, x = 362, y = 447
[ 141.409221] touch press : slot = 0, id = 1, x = 362, y = 447
[ 141.419380] touch press : slot = 0, id = 1, x = 357, y = 449
[ 141.429486] touch press : slot = 0, id = 1, x = 357, y = 449
[ 141.439576] touch press : slot = 0, id = 1, x = 348, y = 451
[ 141.448801] touch press : slot = 0, id = 1, x = 348, y = 451
[ 141.460581] touch press : slot = 0, id = 1, x = 342, y = 463
[ 141.468869] touch press : slot = 0, id = 1, x = 342, y = 463
[ 141.477094] touch release : slot = 0, id = 1

```

Touch Key를 누르는 경우
발생되는 Kernel Message

Touch Panel을 누르는 경
우
발생되는 Kernel Message



Audio input/output driver

www.hardkernel.com

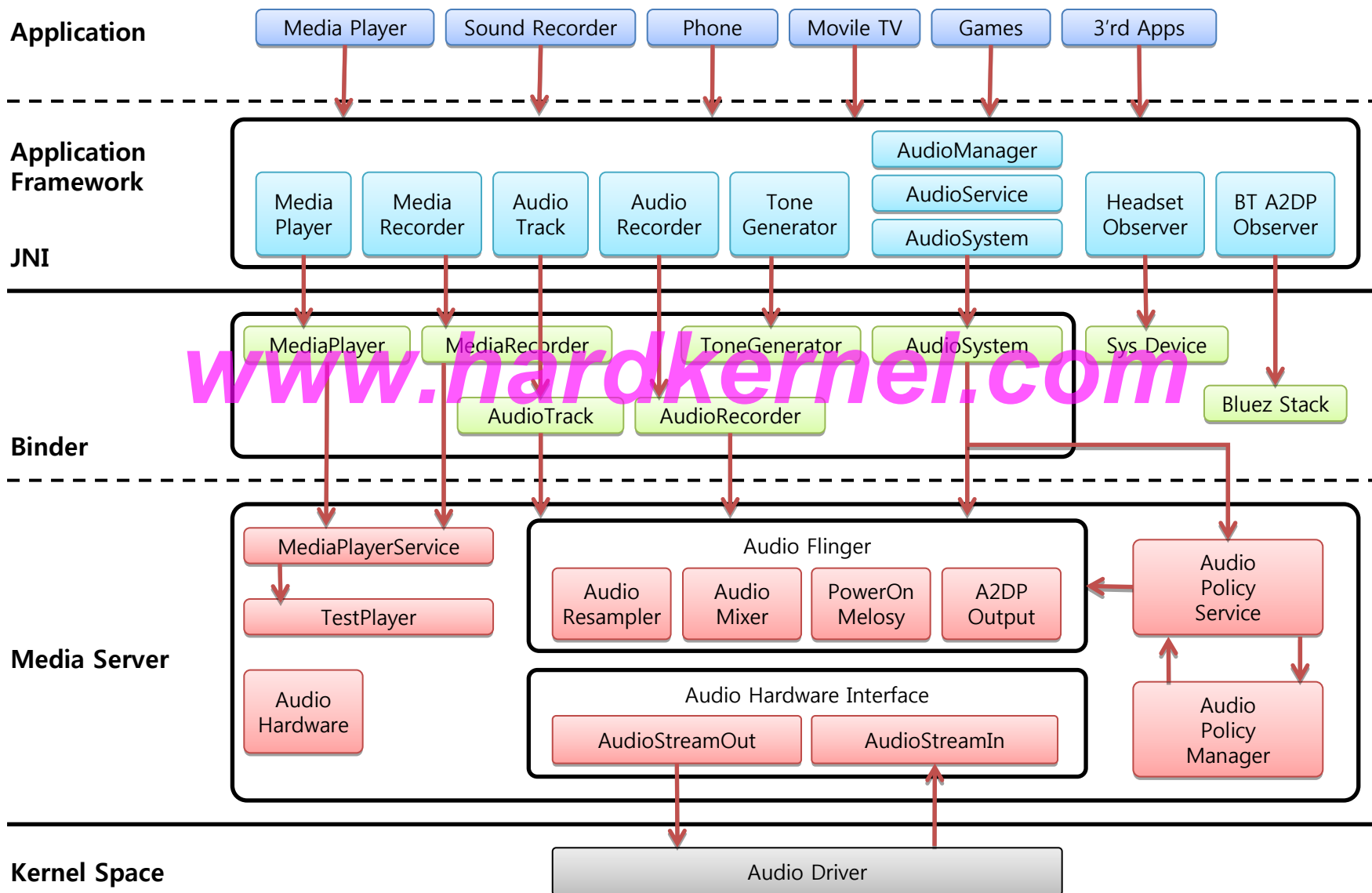
Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 안드로이드 사운드 구조
- ALSA 드라이버 구조
- ODROID-A4 오디오 입출력 구성
- Exynos4210 I2S Bus Interface
- MAX98089 오디오 Codec
- ODROID-A4 ALSA soundcard 드라이버
- Exynos4210 DMA 드라이버
- Exynos4210 I2S 드라이버
- MAX98089 Codec 드라이버
- ODROID-A4 Headset Detect 드라이버
- Android Audio In/Out summary
- Lab/Exam
 - Google voice recognition API app and Text-To-Speech

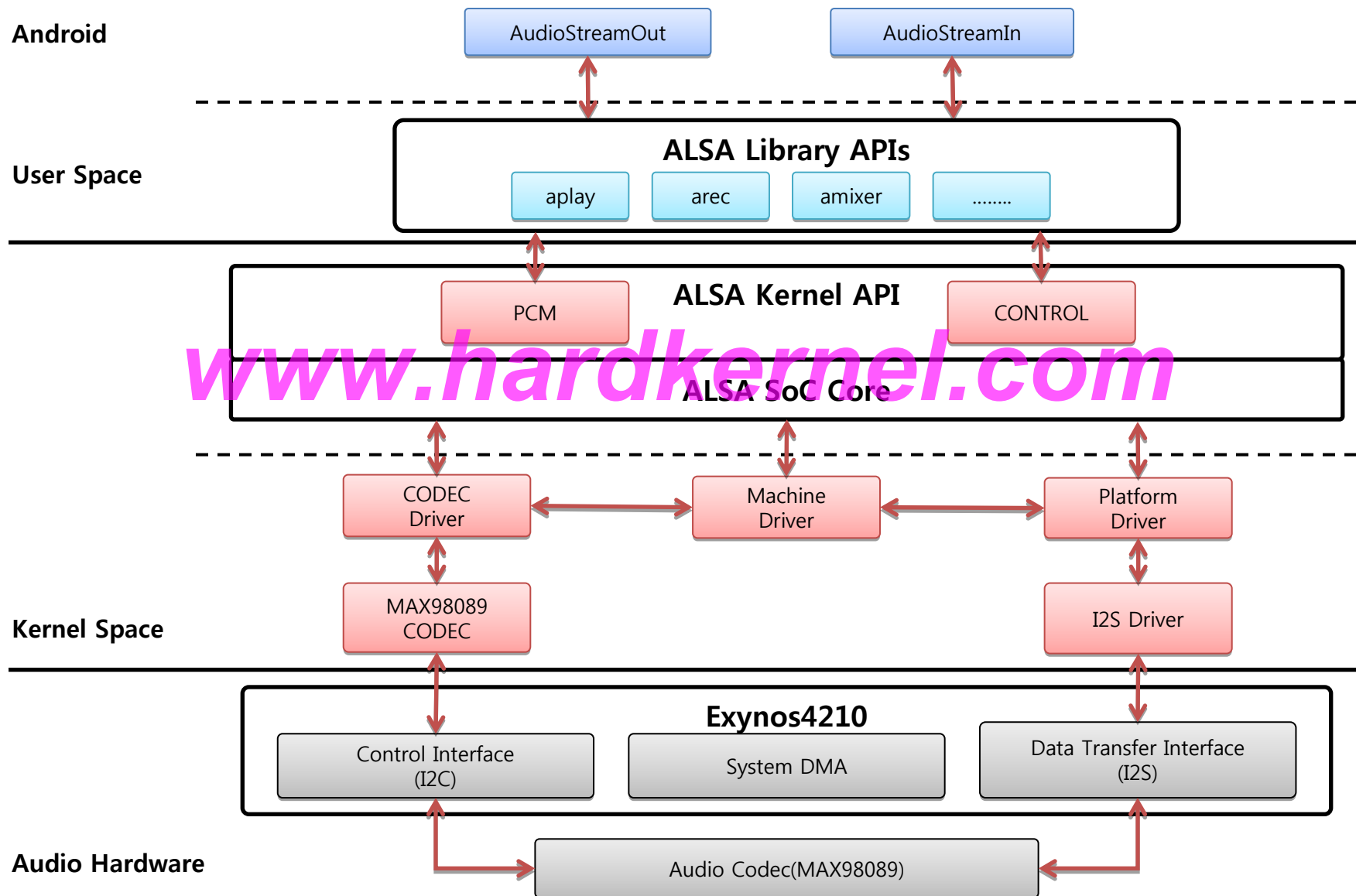
안드로이드 사운드 구조

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



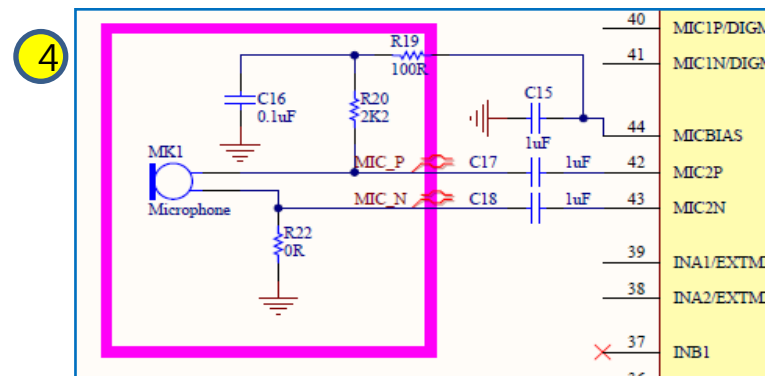
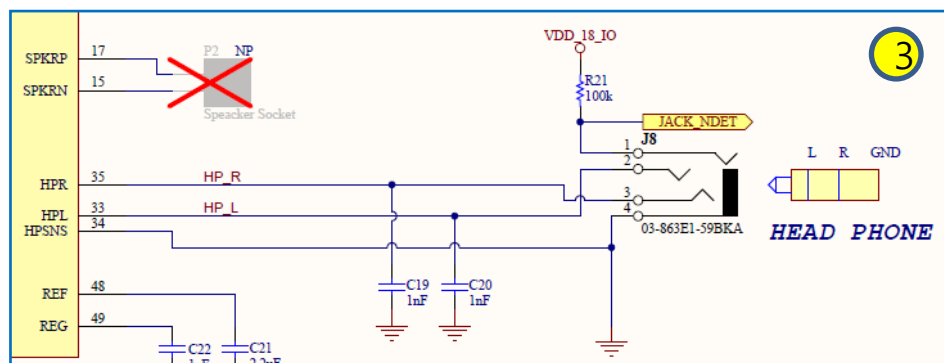
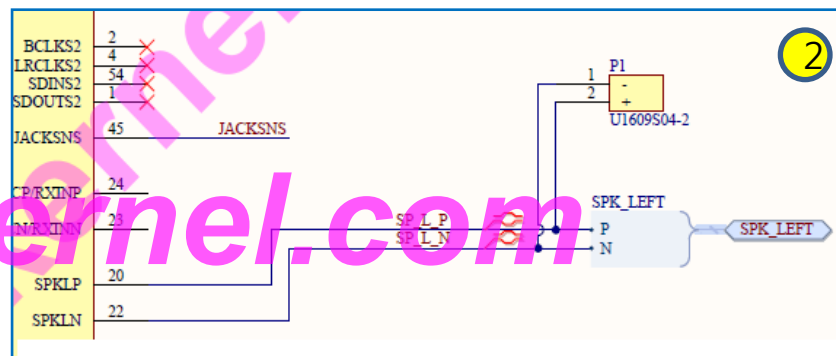
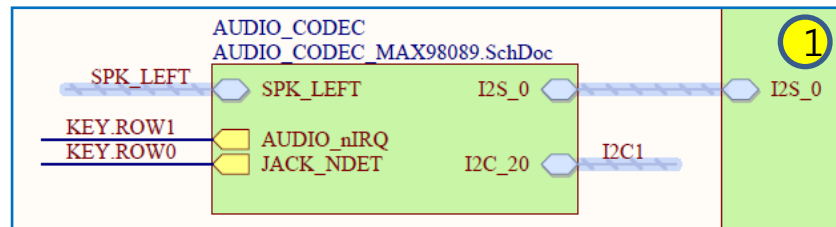
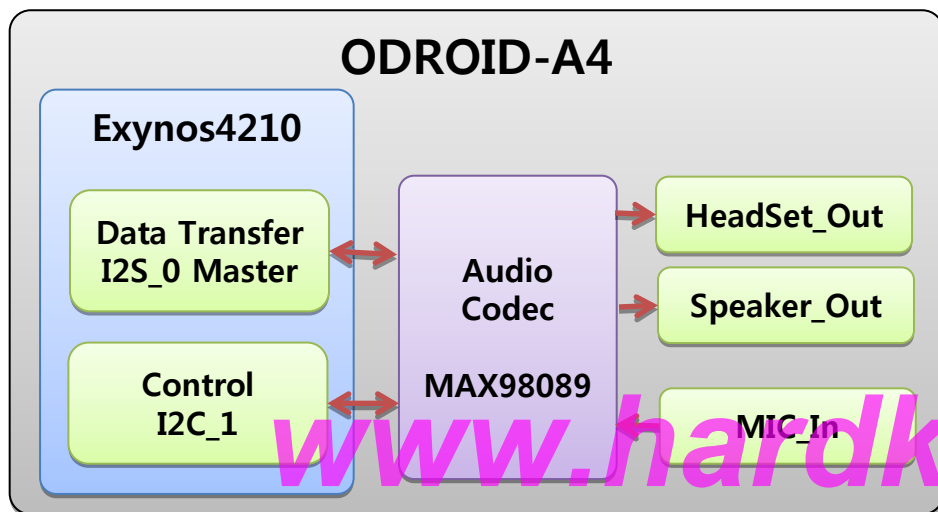
ALSA 드라이버 구조

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



ODROID-A4 오디오 입출력 구성

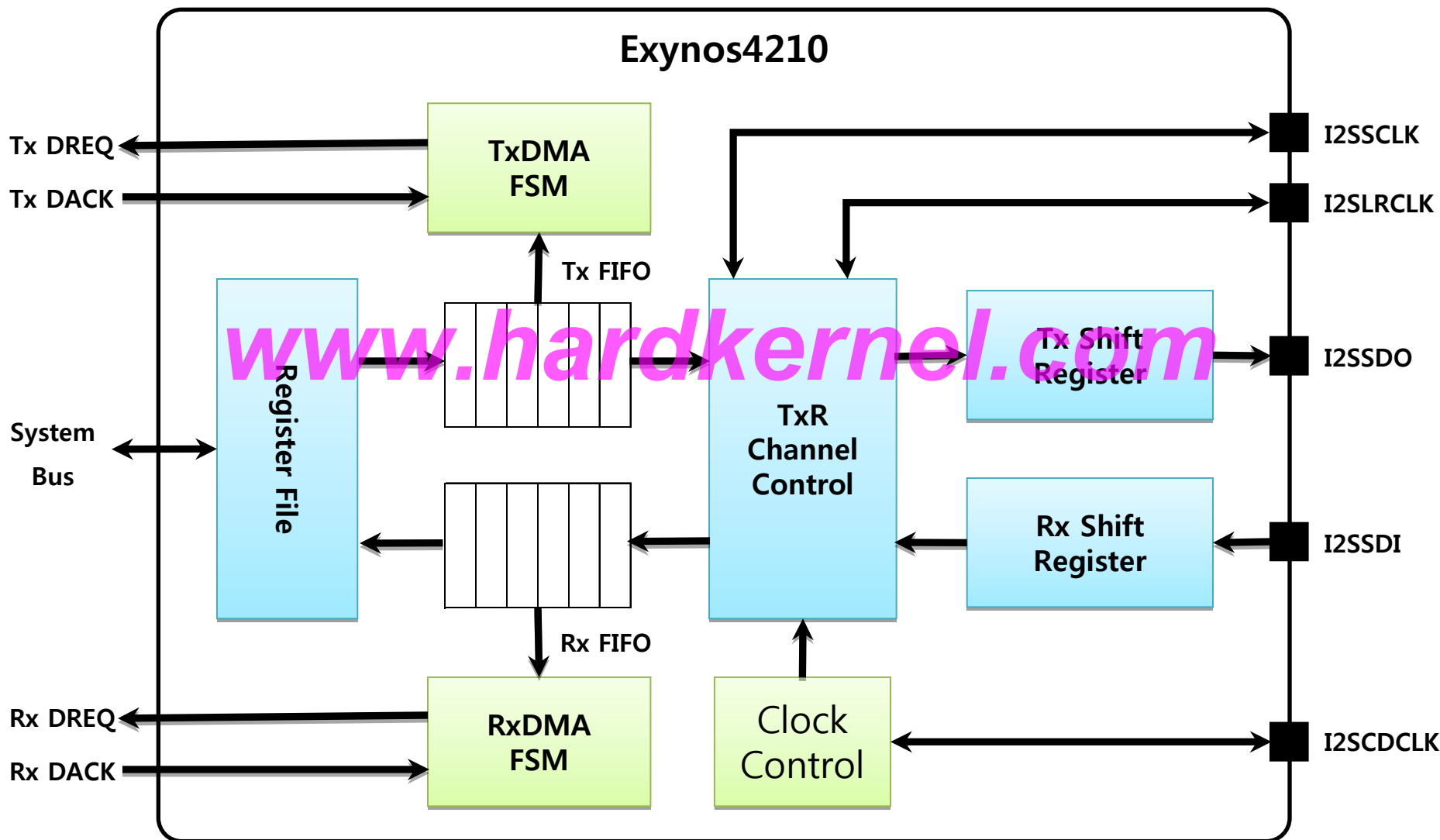
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



1. Exynos4210과 MAX98089 사이의 Data Transfer, Control Interface.
2. MAX98089 의 SPKLP/SPKLN Out 핀에 연결된 Speaker Output.
3. MAX98089 의 HPR/HPL/HPSNS 핀에 연결된 Stereo Headset Output.
4. MAX98089 의 MIC2P/MIC2N/MICBIAS 핀에 연결된 Mic Input.

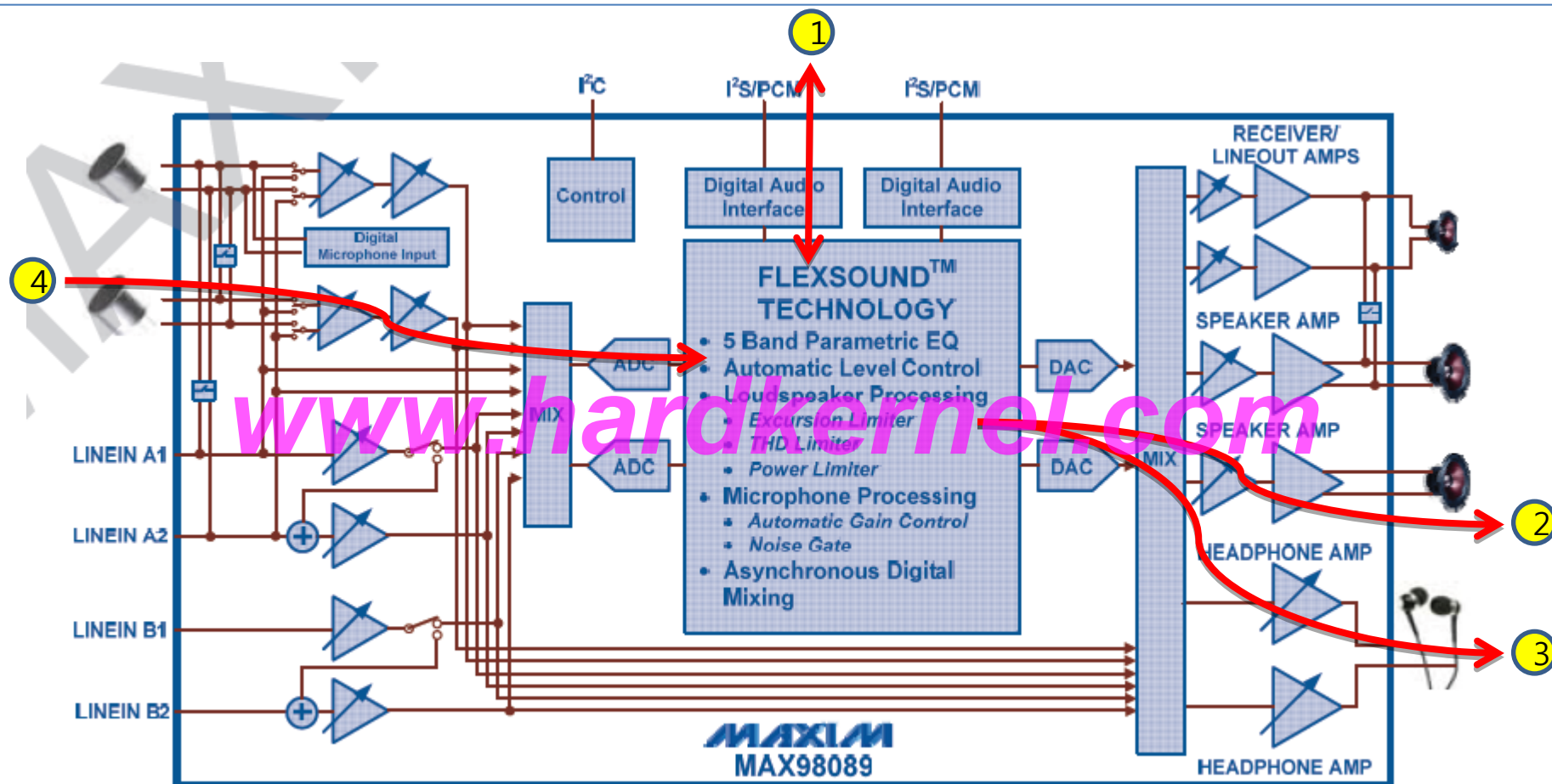
Exynos4210 I2S Bus Interface

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



MAX98089 오디오 Codec

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

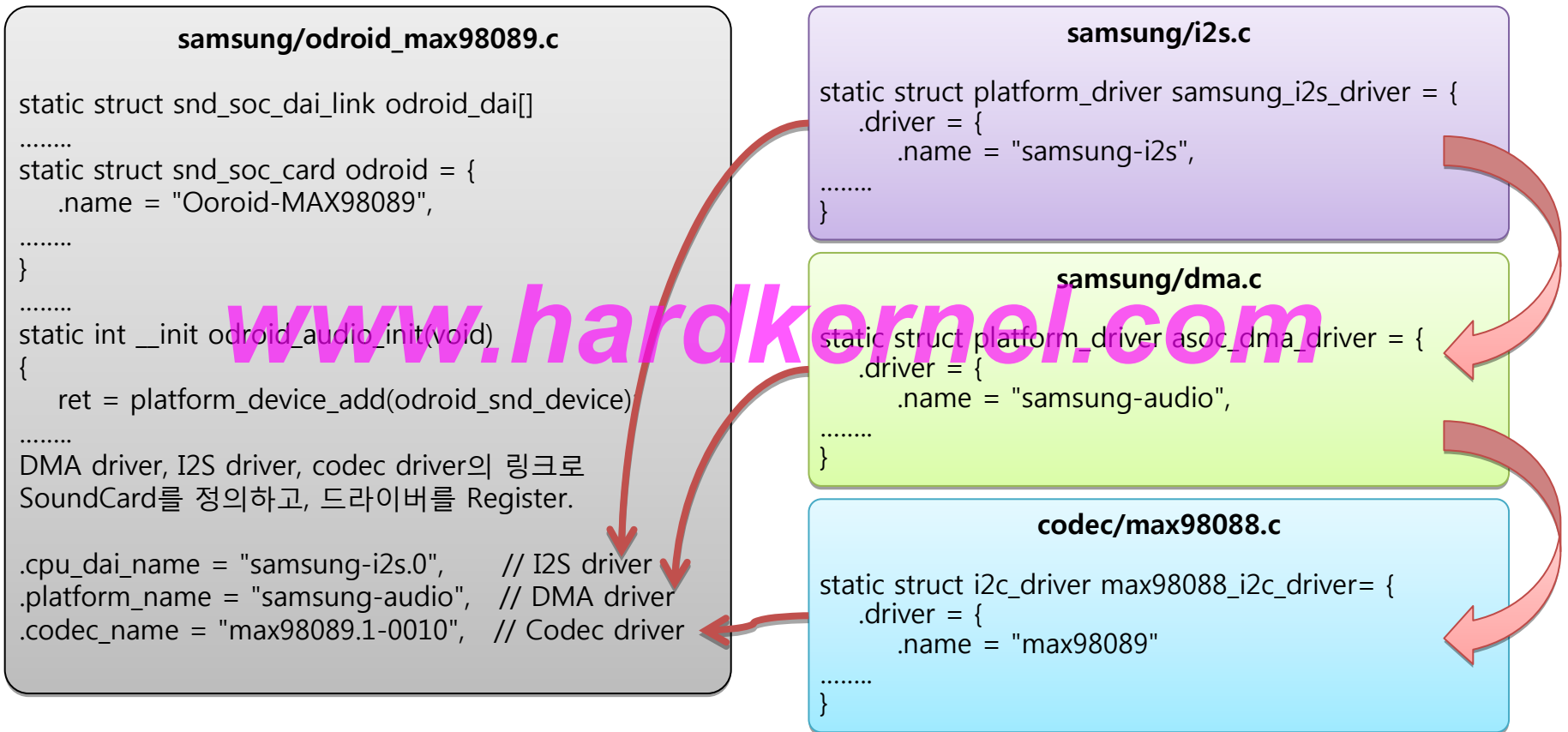


1. Exynos4210과 Data transferring 을 위한 I2S Interface.
2. Speaker Out Path의 Data 흐름. I2S -> DAC -> DAC Mixer -> Speaker AMP -> Speaker
3. Headset Out Path의 Data 흐름. I2S -> DAC -> DAC Mixer -> Headphone AMP -> Headset
4. Mic Input Path의 Data 흐름. Mic Input -> MIC AMP -> ADC Mixer -> ADC -> I2S

ODROID-A4 ALSA soundcard 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ SoundCard 드라이버의 계층 구조 (\$KERNEL_ROOT/sound/soc/)



- ODROID_A4 SoundCard 드라이버는 DMA driver, I2S driver, Codec driver 세 개의 드라이버가 링크된 플랫폼 드라이버이다.
- DMA – Data transferring, I2S – I2S Signal generate, Codec – Digital to Analog, In/Out device mixing
- 각각의 드라이버들이 레지스터 된 후 ALSA Soc Core에서 각 드라이버들간의 링크가 수행된다.
- 정상적으로 링크까지 끝나면 디버그 메시지를 통해 ALSA device list: 를 확인할 수 있다.

Exynos4210 DMA 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

1 samsung/dma.c

```
static int __devinit samsung_asoc_platform_probe(struct platform_device *pdev)
{
    return snd_soc_register_platform(&pdev->dev, &samsung_asoc_platform);
}

static int __devexit samsung_asoc_platform_remove(struct platform_device *pdev)
{
    snd_soc_unregister_platform(&pdev->dev);
    return 0;
}

static struct platform_driver asoc_dma_driver = {
    .driver = {
        .name = "samsung-audio",
        .owner = THIS_MODULE,
    },
    .probe = samsung_asoc_platform_probe,
    .remove = __devexit_p(samsung_asoc_platform_remove),
};

static int __init samsung_asoc_init(void)
{
    return platform_driver_register(&asoc_dma_driver);
}

module_init(samsung_asoc_init);
```

2 samsung/dma.c

```
static struct snd_pcm_ops dma_ops = {
    .open      = dma_open,          // pcm open()
    .close     = dma_close,        // pcm close()
    .ioctl     = snd_pcm_lib_ioctl,
    .hw_params = dma_hw_params,    // HW 설정
    .hw_free   = dma_hw_free,      // mem free
    .prepare   = dma_prepare,      // flush DMA buffer
    .trigger   = dma_trigger,      // Start, Stop trigger
    .pointer   = dma_pointer,      // DMA position move
    .mmap      = dma_mmap,         // mem alloc
};

static struct snd_soc_platform_driver samsung_asoc_platform = {
    .ops      = &dma_ops,
    .pcm_new  = dma_new,
    .pcm_free = dma_free_dma_buffers,
};
```

- Exynos4210 DMA driver register 과정이다.
- 2번에 등록된 file operation 함수들이 ALSA Soc core를 통해 호출되어진다.
- User Space 와 커널간의 Data Transferring에 사용된다.

- User Space ALSA Lib.에서 Kernel PCM API를 사용하여 PCM 데이터를 write하면 (*open)->(*hw_params)->(*prepare) 순서로 동작.
- DMA driver로 전달된 PCM data는 DMA driver에서 정의된 period size에 따라 ALSA Soc Core를 통해 I2S driver로 전달된다.

Exynos4210 I2S 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

1 samsung/i2s.c

```
struct i2s_dai *i2s_alloc_dai(struct platform_device *pdev, bool sec)
```

```
.....
i2s->i2s_dai_drv.probe = samsung_i2s_dai_probe;
i2s->i2s_dai_drv.remove = samsung_i2s_dai_remove;
i2s->i2s_dai_drv.ops = &samsung_i2s_dai_ops;
i2s->i2s_dai_drv.suspend = i2s_suspend;
i2s->i2s_dai_drv.resume = i2s_resume;
```

```
static __devinit int samsung_i2s_probe(struct platform_device *pdev)
```

```
.....
pri_dai = i2s_alloc_dai(pdev, false);
.....
pri_dai->dma_playback.dma_addr = regs_base + I2STXD;
pri_dai->dma_capture.dma_addr = regs_base + I2SRXD;
pri_dai->dma_playback.client =
    (struct s3c2410_dma_client *)&pri_dai->dma_playback;
pri_dai->dma_capture.client =
    (struct s3c2410_dma_client *)&pri_dai->dma_capture;
pri_dai->dma_playback.channel = dma_pl_chan;
pri_dai->dma_capture.channel = dma_cp_chan;
pri_dai->src_clk = i2s_cfg->src_clk;
pri_dai->dma_playback.dma_size = 4;
pri_dai->dma_capture.dma_size = 4;
pri_dai->base = regs_base;
pri_dai->quirks = quirks;
```

2 samsung/i2s.c

```
static struct snd_soc_dai_ops samsung_i2s_dai_ops = {
    .trigger = i2s_trigger,           // Start, Stop trigger
    .hw_params = i2s_hw_params,      // Hw 설정(SampleRate, Ch...)
    .set_fmt = i2s_set_fmt,          // I2S Format 설정
    .set_clkdiv = i2s_set_clkdiv,     // I2S Clock 설정
    .set_sysclk = i2s_set_sysclk,     // I2S Clock Mode 설정
    .startup = i2s_startup,           // I2S Signal Start
    .shutdown = i2s_shutdown,         // shutdown
    .delay = i2s_delay,               // I2S delay
};
```

```
static struct platform_driver samsung_i2s_driver = {
    .probe = samsung_i2s_probe,
    .driver = {
        .name = "samsung-i2s",
    },
};
```

- Exynos4210 I2S driver register 과정이다.
- 2번에 등록된 file operation 함수들이 ALSA Soc core를 통해 호출되어진다.
- DMA driver의 PCM 데이터를 I2S Format으로 Codec driver로 전송한다.
- 실제 하드웨어 연결에서 Clock 및 data 라인으로 Signal이 나가는 부분이다.
- Esynos4210 I2S Block의 Clock/SFR configuration 코드가 기술되어 있다.

MAX98089 Codec 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

codec/max98088.c

```
static struct snd_soc_dai_ops max98088_dai1_ops = {
    .set_sysclk = max98088_dai_set_sysclk,    // I2S Clock Mode
    .set_fmt = max98088_dai1_set_fmt,        // I2S Data Format
    .hw_params = max98088_dai1_hw_params,    // Sample rate, channel
    .trigger = max98088_trigger,
};

struct snd_soc_dai_driver max98088_dai[] = {
{
    .name = "max98089-aif1",
    .....
    .ops = &max98088_dai1_ops,
},
};

static int max98088_probe(struct snd_soc_codec *codec)
{
    codec->write = max98088_i2c_write;
    codec->read = max98088_i2c_read;
}

struct snd_soc_codec_driver soc_codec_dev_max98088 = {
    .probe = max98088_probe,
    .....
};

static int max98088_i2c_probe(struct i2c_client *i2c,
                             const struct i2c_device_id *id)
{
    .....
    ret = snd_soc_register_codec(&i2c->dev,
                                &soc_codec_dev_max98088, &max98088_dai[0], 1);
}
```

codec/max98088.c

```
static struct snd_soc_dai_ops max98088_dai1_ops = {
    .set_sysclk = max98088_dai_set_sysclk,
    .set_fmt = max98088_dai1_set_fmt,
    .hw_params = max98088_dai1_hw_params,
    .trigger = max98088_trigger,
};

static struct i2c_driver max98088_i2c_driver = {
    .driver = {
        .name = "max98089",
        .owner = THIS_MODULE,
    },
    .probe = max98088_i2c_probe,
    .remove = max98088_i2c_remove,
    .id_table = max98088_i2c_id,
};

static int __init max98088_init(void)
{
    return i2c_add_driver(&max98088_i2c_driver);
}
```

- MAX98089 Audio Codec driver register 과정이다.
- I2C driver를 등록하고, i2c_probe 단계에서 soc codec driver를 등록한다.
- dai operation을 통해 interface를 초기화 한다.
- I2S driver에서 PCM data를 I2S Format으로 받을 수 있도록 MAX98089 레지스터를 초기화 한다.

ODROID-A4 Headset Detect 드라이버

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

samsung/odroid_jack.c

```
arch/arm/mach-exynos/mach-odroid-4210.c
#define GPIO_HEADSET    EXYNOS4_GPX3(0)
#define EINT_HEADSET    IRQ_EINT(24)

struct switch_dev switch_jack_detection = {
    .name = "h2w",
};

static irqreturn_t detect_irq_handler(int irq, void *dev_id)
static void jack_detect_change(struct work_struct *dwork)

static int odroid_jack_probe(struct platform_device *pdev)
{
    //GPIO configuration

    ret = switch_dev_register(&switch_jack_detection);

    ret = request_irq(det_jack->eint, detect_irq_handler, IRQF_DISABLED,
        "hpjack-irq", NULL);
    irq_set_irq_type(det_jack->eint, IRQ_TYPE_EDGE_BOTH);
    INIT_DELAYED_WORK(&jack_detect_work, jack_detect_change);
}
```

codec/max98089_mixer.c

```
void max98089_set_playback_speaker
(struct snd_soc_codec *codec);

void max98089_set_playback_headset
(struct snd_soc_codec *codec);

void max98089_set_playback_speaker_headset
(struct snd_soc_codec *codec);

void max98089_disable_playback_path
(struct snd_soc_codec *codec, enum playback_path path);

void max98089_set_record_main_mic
(struct snd_soc_codec *codec);

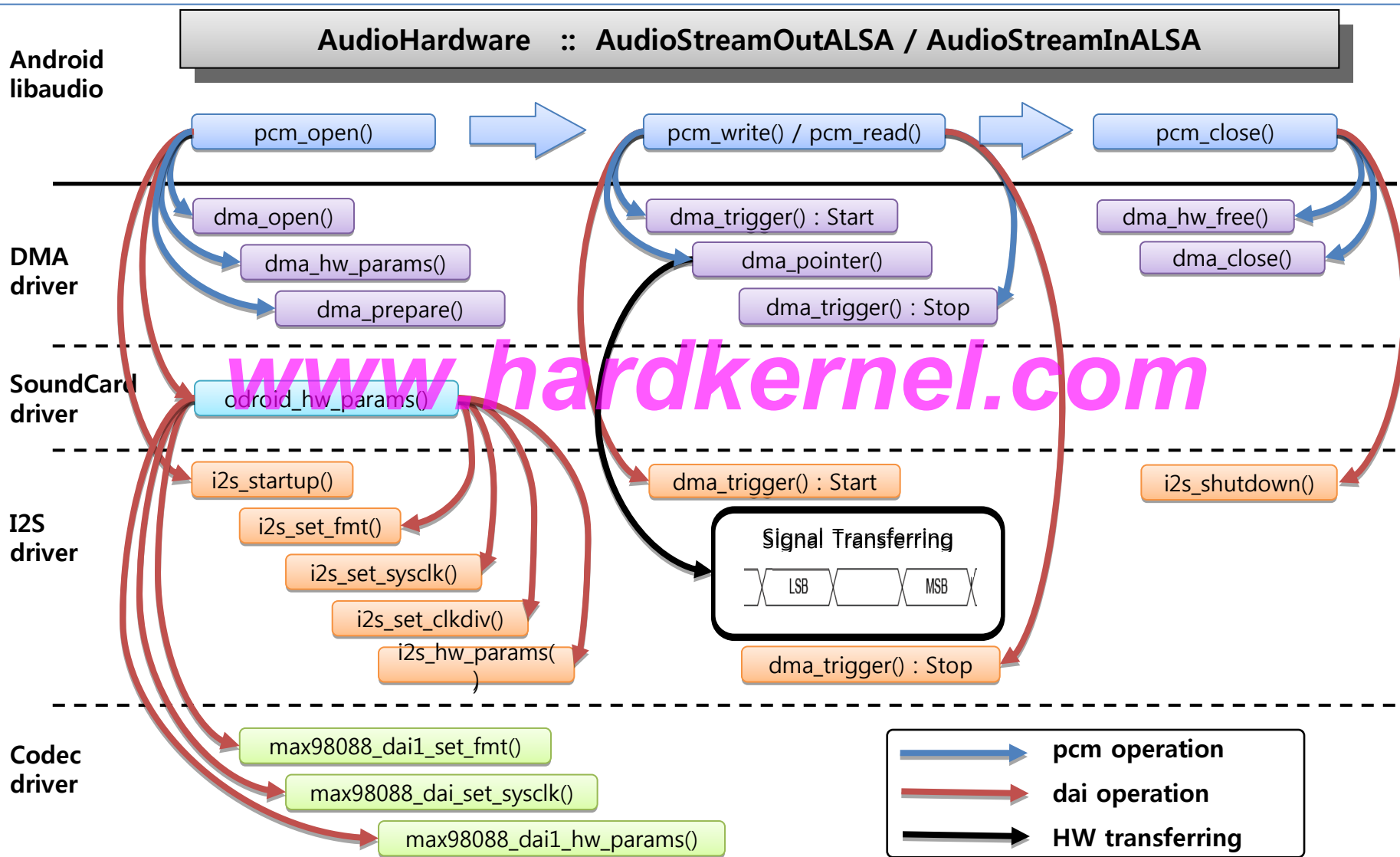
void max98089_set_record_headset_mic
(struct snd_soc_codec *codec);

void max98089_disable_record_path
(struct snd_soc_codec *codec, enum record_path rec_path);
```

- ODROID-A4 Phone-Jack 핀이 연결된 GPIO 인터럽트를 등록하고 인터럽트가 발생하면 Audio 출력을 상태에 따라 변경하는 driver.
- probe 단계에서 인터럽트를 등록하고 IRQ type을 정의한다.
- switch_device는 안드로이드에서 사용하는 것으로 switch_set_state() 함수를 통해 상태를 변경할 수 있다.
- switch_device의 상태가 변경되면 Android Headset Observer 에서 Broadcasting 하며 Audio system이 잭의 상태를 확인할 수 있다.
- ODROID-A4의 경우 오디오 출력 전환(헤드셋<->스피커)은 device driver단에서 하고 있으나 엄밀히 따지자면 switch event를 받은 Audio system 에서 ALSA mixer API 를 통해 출력전환이 이루어 질 수 있도록 해야 한다.

Android Audio In/Out summary

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Google voice recognition API app and Text-To-Speech

In this section we break down the code of an application to see how really works Google voice recognition API app & Text-To-Speech

➤ To use Text-To-Speech, we must implement in our activity, the class:

OnInitListener

➤ The first thing to do is create the attribute of the TTS. For example:

```
private TextToSpeech mTts;
```

➤ Then initialize the attribute:

```
mTts = new TextToSpeech(this,this);
```

➤ Add this code:

```
public void onInit(int status) {  
    if (status == TextToSpeech.SUCCESS){  
        mTts.setLanguage(Locale.getDefault());  
    }  
}
```

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- When completed the activity is important to turn off the TTS

@Override

```
public void onDestroy() {
    if (mTts != null) {
        mTts.stop();
        mTts.shutdown();
    }
    super.onDestroy();
}
```

- To play a text is as simple as this:

```
mTts.speak( YOUR STRING TEXT ,TextToSpeech.QUEUE_FLUSH, null);
```

- More info:

<http://developer.android.com/reference/android/speech/tts/TextToSpeech.html>

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- To use speech recognition have to launch the voice recognition:

Intent intent =

```
new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
    RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH);
startActivityForResult(intent, RECOGNIZER);
```

- For text that has been recognized has to add the following:

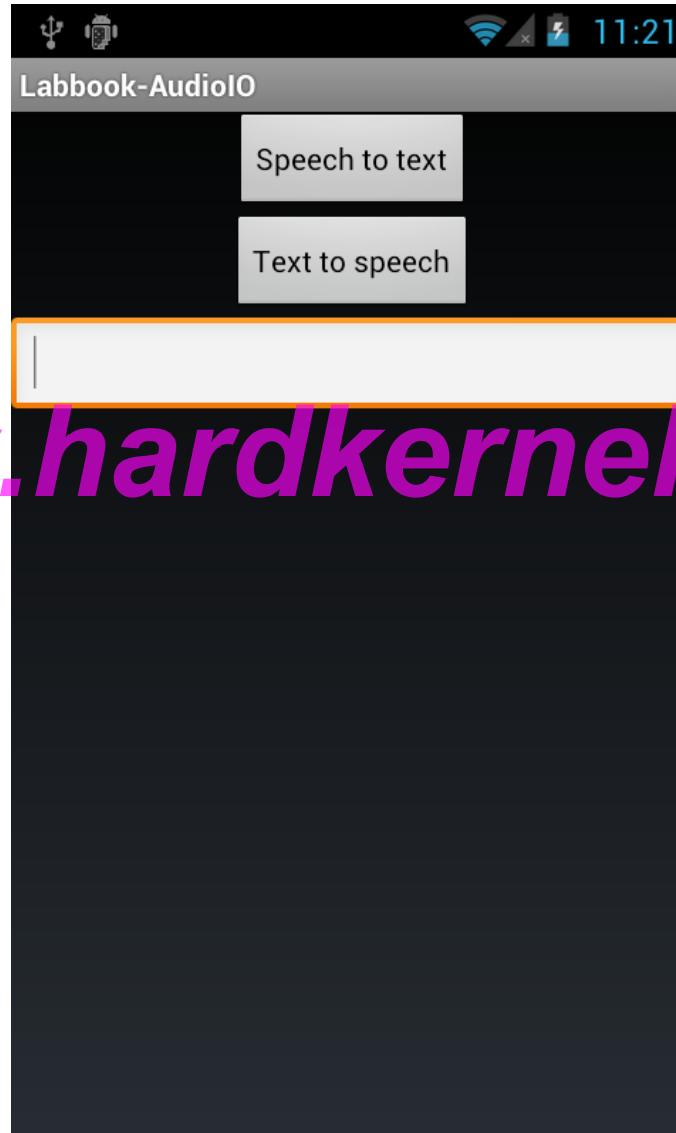
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode==RECOGNIZER && resultCode==Activity.RESULT_OK) {
        ArrayList<String> result =
            data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
    }
}
```

- More info:

<http://developer.android.com/reference/android/speech/RecognizerIntent.html>

Lab/Exam(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



www.hardkernel.com



Camera driver

www.hardkernel.com

Agenda

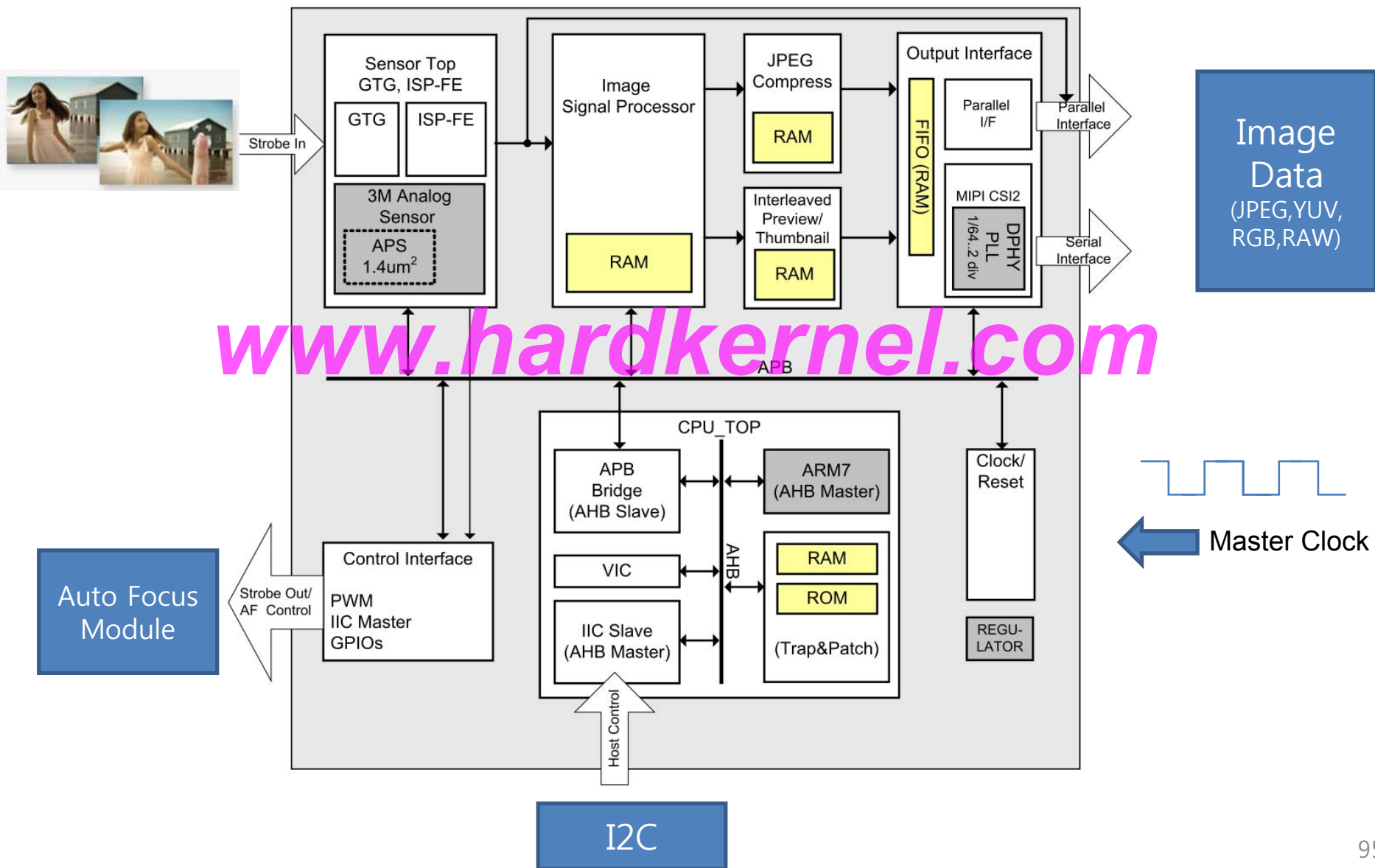
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- CMOS Image Sensor Block Diagram
- Camera Interface for Exynos4210
- Device Driver – CIS(CMOS Image Sensor)
- Device Driver – FIMC
- Device Driver – FIMC Block Diagram
- Device Driver - V4L2 Interface
- Device Driver – ODROID-A4 Platform
- Android Camera Framework
- Proprietary Video Libraries
- Lab/Exam(I) Camera preview
- Lab/Exam(II) Barcode reader 예제

www.hardkernel.com

CMOS Image Sensor Block Diagram

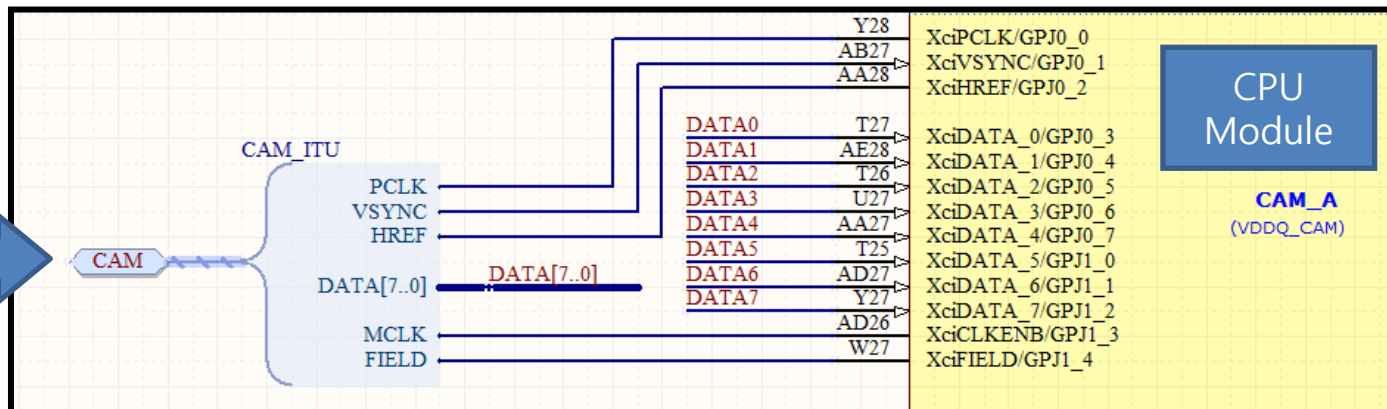
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



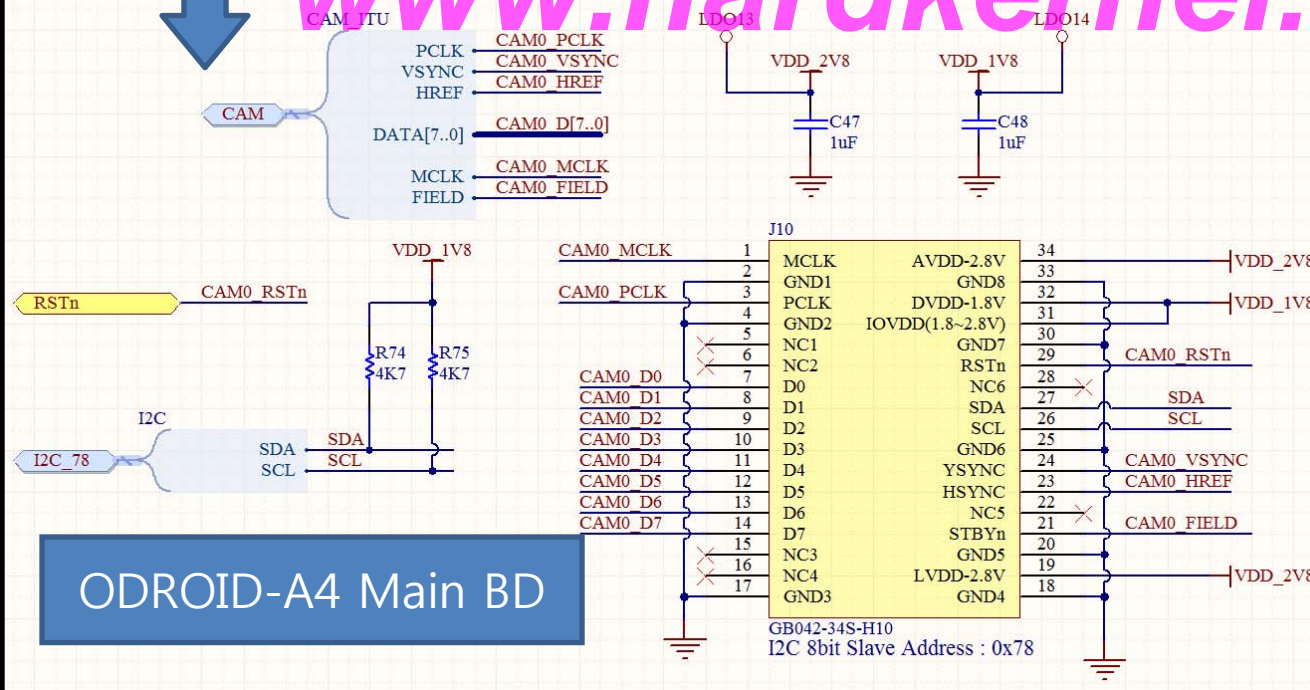
Camera Interface for Exynos4210

www.hardkernel.com 이 자료는 (주)하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

B2B Connector



www.hardkernel.com



➤ Exynos4210 CPU는

- ✓ Parallel 2개,

- ✓ MIPI 2lane 17H,

- ✓ MIPI 4lane 1개의

카메라 H/W Interface를 지원한다.

Device Driver – CIS(CMOS Image Sensor)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Kernel/driver/media/video/s5k5cagx.c, s5k5cagx.h

– Camera 초기화

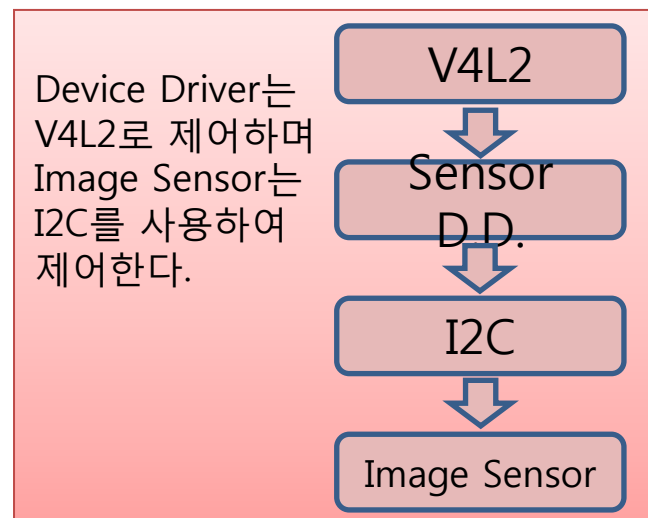
- ◆ s5k5cagx_init()
- ◆ 센서 내부 클럭 셀팅
- ◆ 카메라 센서 firmware patch

– Camera 제어 함수

- ◆ 이미지 format, frame size, 영상 효과, 모드 변경
- ◆ focus 제어,

– V4L2 interface : ioctl()

- ◆ VIDIOC_G_CTRL → s5k5cagx_g_ctr()
- ◆ VIDIOC_S_CTRL → s5k5cagx_s_ctr()
- ◆ VIDIOC_S_FMT → s5k5cagx_s_fmt()
- ◆ VIDIOC_G_PARM → s5k5cagx_g_parm()
- ◆ VIDIOC_S_PARM → s5k5cagx_s_parm()



Device Driver - FIMC

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ kernel/driver/media/video/samsung/fimc

- fimc_dev.c
 - ◆ video node 생성(/dev/video0)
 - ◆ V4L2 interface 등록
- fimc_v4l2.c
 - ◆ V4L2 interface.
 - ◆ kernel/driver/media/video/v4l2*.c
- fimc_capture.c
 - ◆ 카메라로 부터 영상 획득
 - ◆ Mclk, Camera Power On/Off
 - ◆ v4l2_subdev_call() → s5k5cagx_init()
- fimc_output.c
 - ◆ 영상 출력을 담당
 - ◆ size, rotation, format 등을 변환
- fimc_overlay.c
 - ◆ LCD 출력 담당
- fimc_regs.c
 - ◆ H/W register control 함수 모음

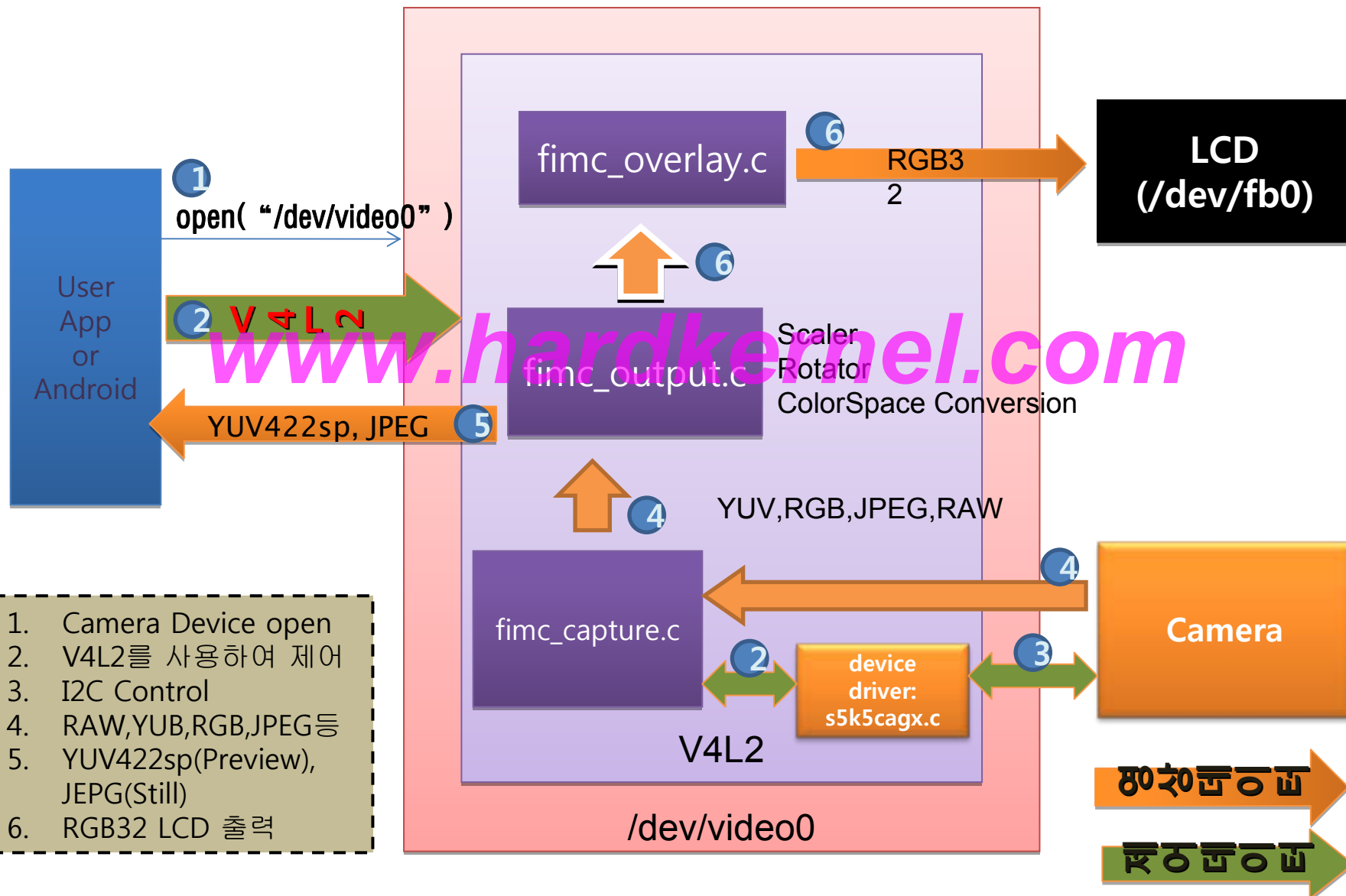
FIMC

(Fully Interactive Mobile Camera Interface)

– Post Processor, Scaler, Rotator, input/output DMA Driver등이 포함된 디바이스 드라이버 모음.

Device Driver – FIMC Block Diagram

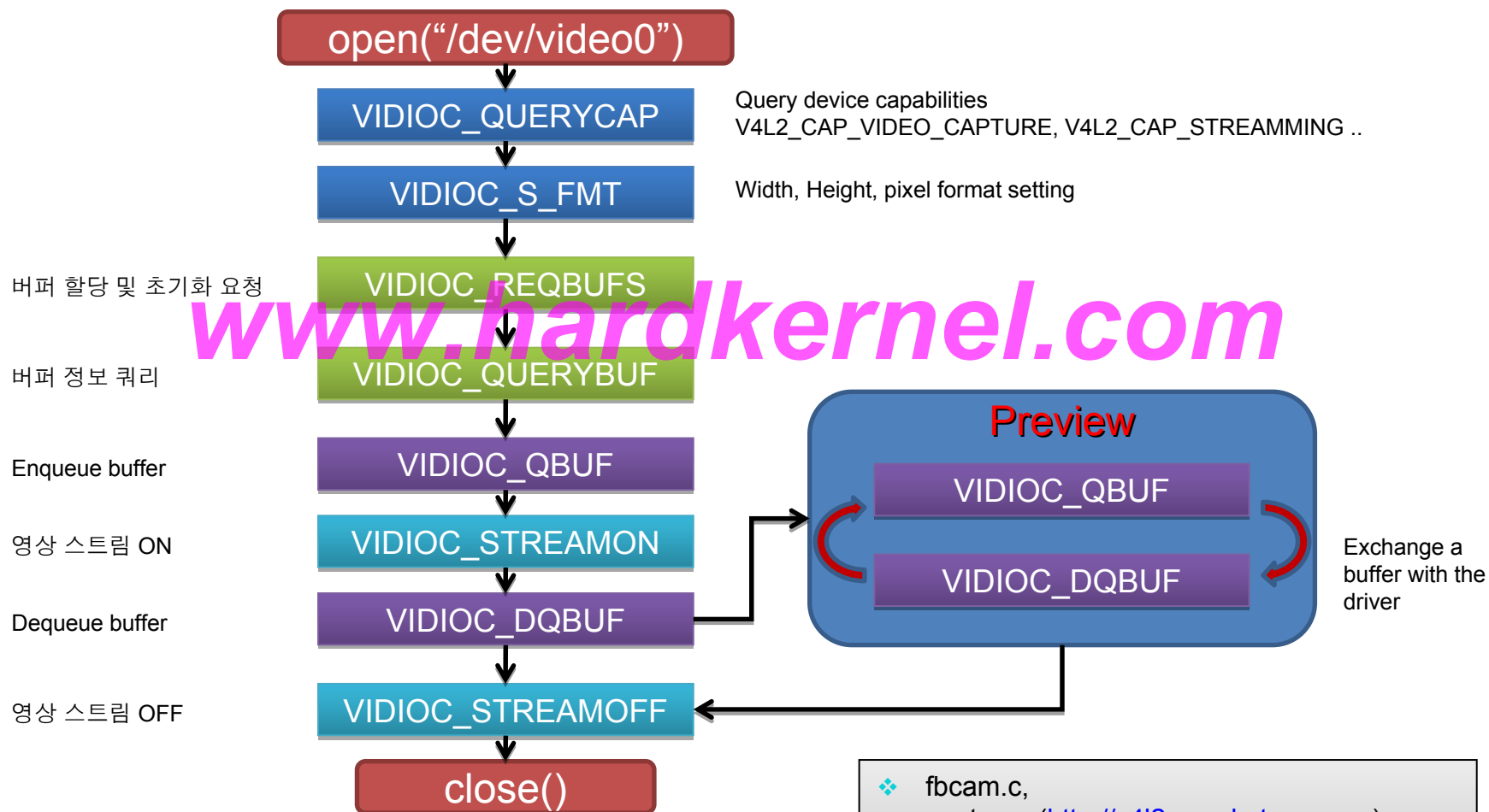
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Device Driver - V4L2 Interface

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Video for Linux 2 Interface



❖ fbcam.c,
capture.c(<http://v4l2spec.bytesex.org>)
- 간단히 카메라 영상을 LCD frame buffer에 출력 또는 파일 저장 예제 참고

Device Driver – ODROID-A4 Platform

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

```
static struct i2c_board_info s5k5cagx_i2c_info = {
    I2C_BOARD_INFO("S5K5CAGX", 0x3c),
    .platform_data = &s5k5cagx_plat,
};

static struct s3c_platform_camera s5k5cagx = {
    .id = CAMERA_PAR_A,
    .type = CAM_TYPE_ITU,
    .i2c_busnum = 1,
    .info = &s5k5cagx_i2c_info,
    .fmt = ITU_601_YCBCR422_8BIT,
    .order422 = CAM_ORDER422_8BIT_YCBYCR,
    .pixelformat = V4L2_PIX_FMT_YUYV,
    .srclk_name = "xusbxti",
    .clk_name = "sclk_cam0",
    .clk_rate = 24000000,
    .line_length = 480,
    .width = 640,
    .height = 480,
    .window = {
        .left = 0,
        .top = 0,
        .width = 640,
        .height = 480,
    },
    .inv_pclk = 0,
    .inv_vsync = 1,
    .inv_href = 0,
    .inv_hsync = 0,
    .initialized = 0,
    .cam_power = hkdck210_cam0_power_odroid_a4,
};
```

I2C Subdev로 동작

영상 포맷 설정

Reset 동작 제어

```
void cam0_s5k5cagx_reset(int power_up)
{...}
```

카메라 Power 제어

```
static int hkdck210_cam0_power_odroid_a4(int onoff)
{...}
```

Master Clock
설정

출력영상
사이즈 설정

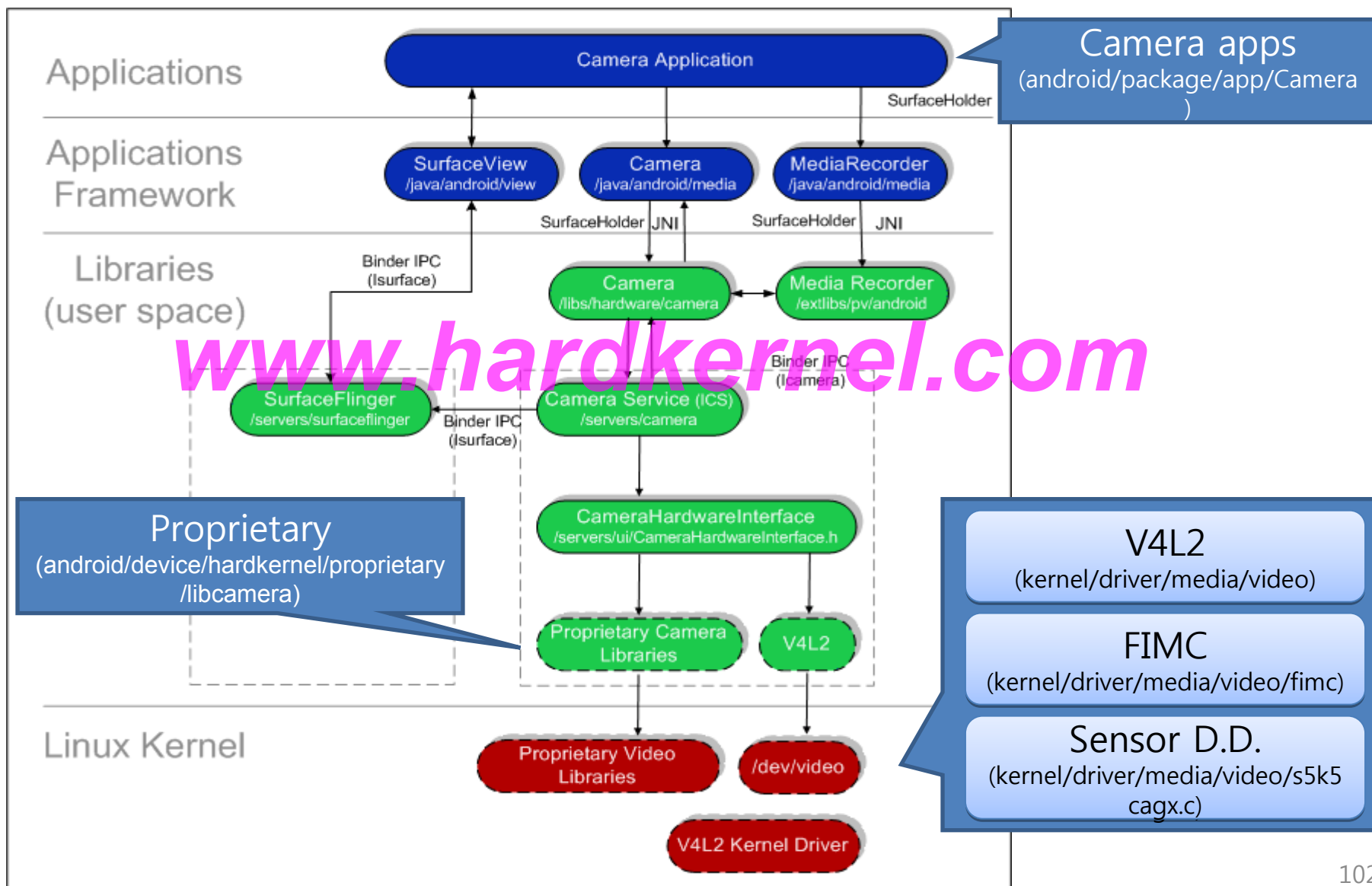
Polarity
설정

```
/* Interface setting */
static struct s3c_platform_fimc fimc_plat = {
    .default_cam = CAMERA_PAR_A,
    .camera = {
        &s5k5cagx,
    },
    .hw_ver = 0x51,
};
```

카메라를 FIMC platform에 등록
하여 FIMC를 통해 카메라를 제
어하게 된다.

Android Camera Framework

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

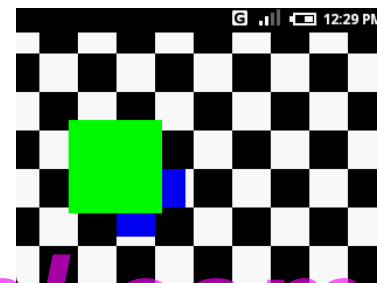


Proprietary Video Libraries

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ android/frameworks/base/services/camera/libcameraservice

- 안드로이드 카메라 레퍼런스 코드
- 에뮬레이터에서 가상 카메라로 동작 (FakeCamera)

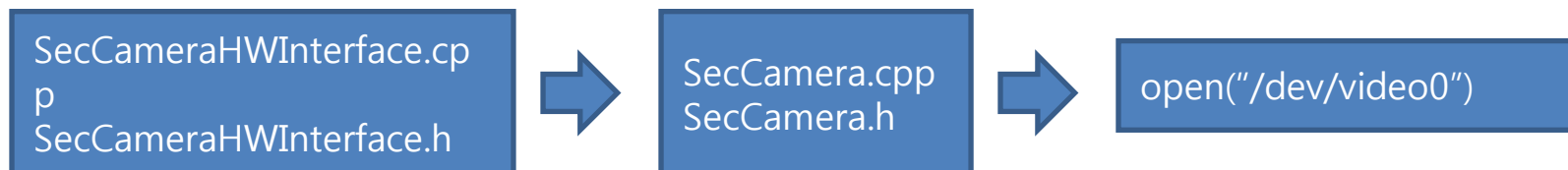


<FakeCamera>

www.hardkernel.com

➤ android/device/samsung/exynos4/libcamera

- Samsung BSP 포팅 코드
- 안드로이드에서 카메라를 최종적으로 open하게 된다.



Lab/Exam(I) – Simple Camera Preview

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Simple Camera Preview 진행 순서

- 1) 프로젝트 생성
- 2) AndroidManifest.xml에 카메라 관련 permission 추가
- 3) main.xml에서 레이아웃 수정
- 4) CameraPreview class 추가 → CameraPreview.java
- 5) CameraPreviewTestActivity.java 수정

www.hardkernel.com

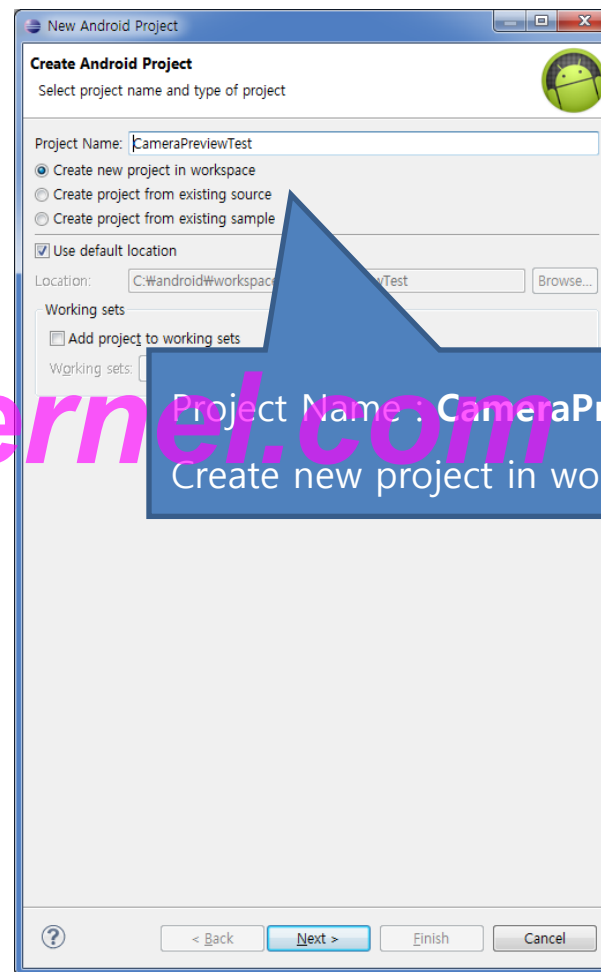
Lab/Exam(I) - 프로젝트 생성(I)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Eclipse를 실행 후 메뉴 File> New> Android Project 선택하면 다음과 같은 New Android Project 창이 표시된다.

➤ Create Android Project

- ✓ Create new project in workspace : 새로운 프로젝트 생성
- ✓ Create new project from source : 소스가 존재 할 경우 그 소스를 복사하여 프로젝트를 생성
- ✓ SDK에 포함된 Android 샘플 소스를 복사하여 프로젝트 생성



Lab/Exam(I) - 프로젝트 생성 (II)

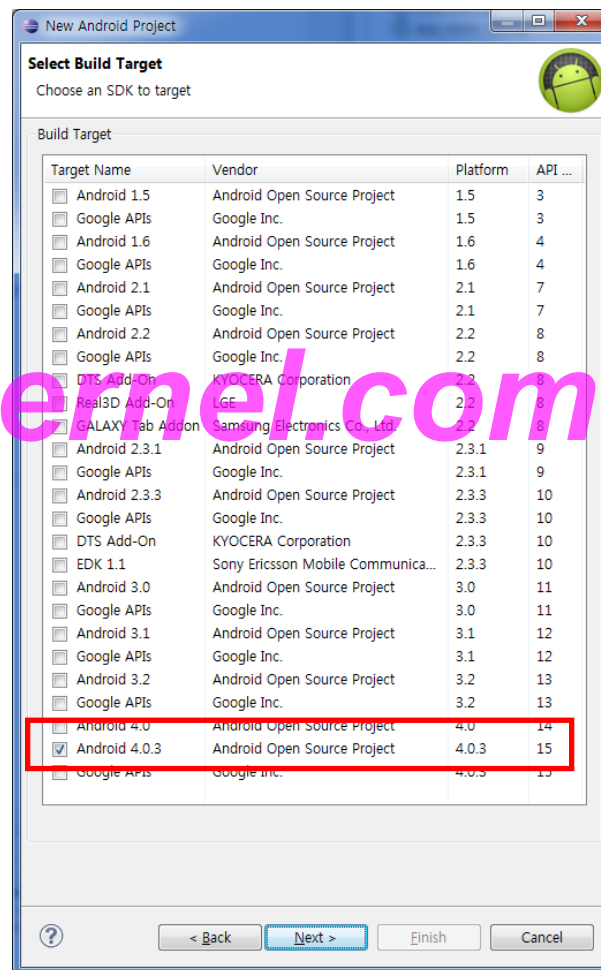
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Select Build Target

- 사용하고자 하는 디바이스의 안드로이드 버전이나 지금 만드는 어플리케이션이 지원하고자 하는 최소 안드로이드 버전을 선택한다.

www.hardkernel.com

Android 4.0.3 선택

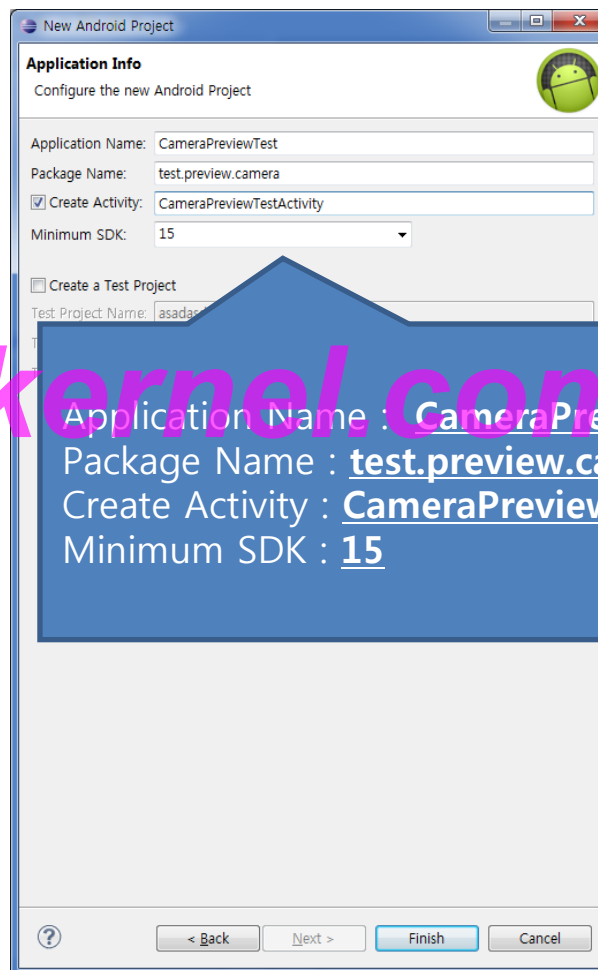


Lab/Exam(I) - 프로젝트 생성 (III)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Application Info

- Application Name : 어플리케이션의 대표 이름이고 타이틀바에 표시된다.
- Package Name : 기존 패키지와 중복 되지 않도록 넣어줌. java의 패키지 관례를 따른다. (예: com.회사이름)
- Create Activity : 선택할 경우 기본적인 Activity를 생성한다.
- Minimum SDK:최소 지원할 SDK API 버전을 입력한다.



Lab/Exam(I) – AndroidManifest.xml 수정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ AndroidManifest.xml에 다음과 같이 permission 추가

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="test.preview.camera" android:versionCode="1" android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
```

Camera Device를 사용함

```
<uses-permission android:name="android.permission.CAMERA" />
```

```
<uses-feature android:name="android.hardware.camera" />
```

Camera 기능을 사용함

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <activity
        android:name=".CameraPreviewTestActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape">
```

Activity를 Landscape로 고정

```
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>
</manifest>
```

Lab/Exam(I) – main.xml 수정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ main.xml

- 어플리케이션의 메인 layout 파일

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
```

```
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
```

```
    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1" />
```

```
</LinearLayout>
```

Preview를 담을
FrameLayout 추가

Lab/Exam(I) – CameraPreview class추가 (I)

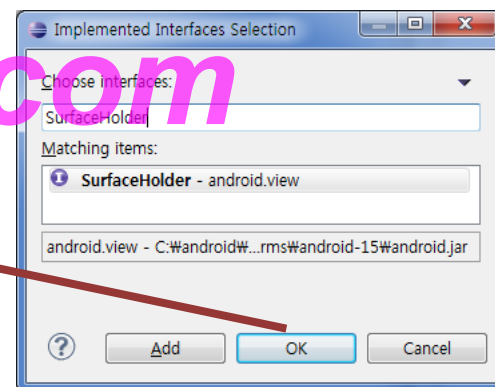
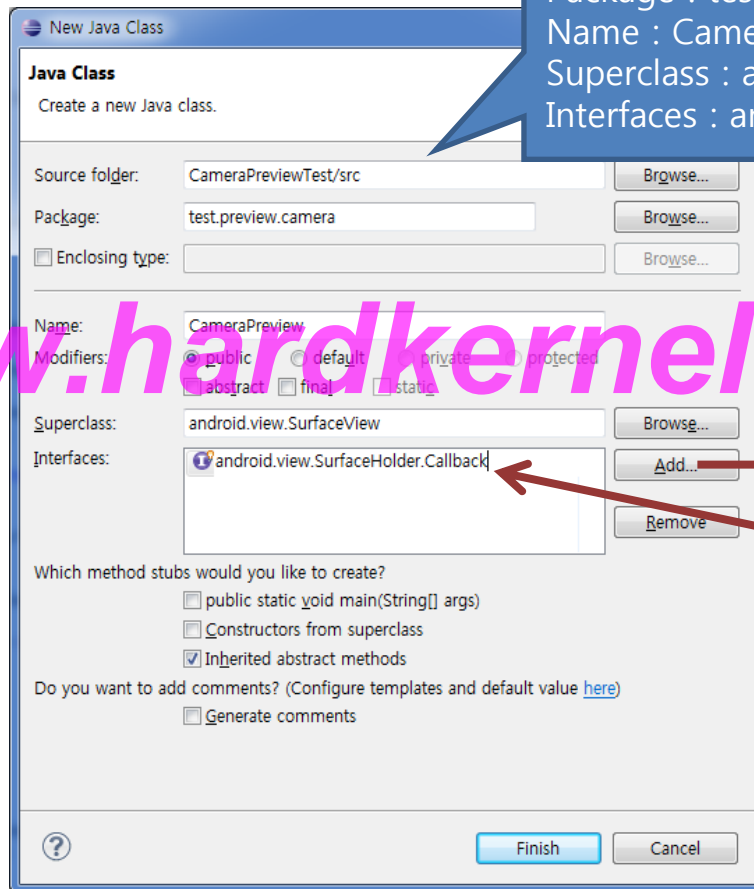
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Preview가 나오게 될 CameraPreview class를 추가한다.

➤ 메뉴 **File> New > Class**를 선택하여 New Java Class 창에서 Class를 추가한다.

➤ CameraPreview class를 추가하면 CameraPreview.java 파일이 생성되며, 다음 장의 코드를 넣어 준다.

Source folder : CameraPreviewTest/src
Package : test.preview.camera
Name : CameraPreview
Superclass : android.view.SurfaceView
Interfaces : android.view.SurfaceHolder.Callback



Lab/Exam(I) – CameraPreview class추가(II)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- CameraPreview.java에 다음과 같이 코드를 추가하여 Preview가 나오게 될 SurfaceView를 생성한다.

```
package test.preview.camera;

import java.io.IOException;
import android.content.Context;
import android.hardware.Camera;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class CameraPreview extends SurfaceView
    implements SurfaceHolder.Callback {

    private SurfaceHolder mHolder;
    private Camera mCamera;
    public String TAG = "Camera";

    public CameraPreview(Context context, Camera camera) {

        super(context);
        mCamera = camera;
        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder
            .setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }
}
```

```
public void surfaceCreated(SurfaceHolder holder) {
    try {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
    } catch (IOException e) {
        Log.d(TAG, "Error setting camera preview: "
            + e.getMessage());
    }
}
```

Preview를 표시
할 holder를 지정

```
public void surfaceDestroyed(SurfaceHolder holder) { }
```

화면이 회전하는 등의
변화가 있을 경우 호출

```
public void surfaceChanged(
    SurfaceHolder holder, int format, int w, int h) {
    if (mHolder.getSurface() == null) {
        return;
    }
    try {
        mCamera.stopPreview();
    } catch (Exception e) { }
```

Preview 중지

```
try {
    mCamera.setPreviewDisplay(mHolder);
    mCamera.startPreview();
} catch (Exception e) {
    Log.d(TAG, "Error starting camera preview: "
        + e.getMessage());
}
}
```

Preview 시작

Lab/Exam(I) – CameraPreviewTestActivity 수정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ CameraPreviewTestActivity.java

- 카메라 어플리케이션의 메인 Activity.

```
package test.preview.camera;
```

```
import android.app.Activity;
import android.content.Context;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Bundle;
import android.util.Log;
import android.widget.FrameLayout;
```

```
public class CameraPreviewTestActivity extends Activity {
    Camera mCamera;
    private CameraPreview mPreview;
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.main);
```

```
    checkCameraHardware(this);
```

```
    mCamera = getCameraInstance();
```

```
    mPreview = new CameraPreview(this, mCamera);
```

```
    FrameLayout preview
```

```
        = (FrameLayout) findViewById(R.id.camera_preview);
```

```
    preview.addView(mPreview);
```

```
}
```

Preview를 표시할
View를 생성

Preview를 표시할
View를 생성

카메라 디바이스가 실제 존재하는지 체크

```
private boolean checkCameraHardware(Context context) {
    if (context.getPackageManager().hasSystemFeature(
        PackageManager.FEATURE_CAMERA)) {
        Log.d("Camera", "checkCameraHardware true");
        return true;
    } else {
        Log.d("Camera", "checkCameraHardware false");
        return false;
    }
}
```

카메라를 액세스 하기위해 인스턴스를 얻음

```
public static Camera getCameraInstance() {
    Camera c = null;
    try {
        c = Camera.open();
    } catch (Exception e) {
    }
    return c;
}
```

카메라 OPEN

Lab/Exam(I) - Result Screenshot

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

▶ 실행결과



Lab/Exam(II) – Barcode Reader 예제

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Barcode Reader 예제 진행 순서

- 1) Zxing library overview
- 2) Zxing 라이브러리 컴파일 및 실행
- 3) Barcode reader Android app 빌드
- 4) Eclipse에서 빌드

www.hardkernel.com

Lab/Exam(II) - Zxing over view

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- ZXing (다음과 같이 발음 "zebra crossing")은 Java 작성되어 다른 언어로도 포팅 된, 멀티 포맷 1D/2D 바코드 이미지 처리 오픈 소스 라이브러리 이다. 이 라이브러리는 모바일 폰에 있는 카메라를 사용하여 서버에 연결 없이, 바코드를 스캔하고 디코드 할 수 있다.
- 현재 다음과 같은 포맷을 지원한다.
UPC-A and UPC-E, EAN-8 and EAN-13, Code 39 , Code 93, Code 128, ITF, Codabar, RSS-14 (all variants), QR Code , Data Matrix, Aztec ('beta' quality), PDF 417 ('alpha' quality)
- 홈페이지 : <http://code.google.com/p/zxing/>

Lab/Exam(II) – Zxing 라이브러리 컴파일 및 실행

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 소스 코드 다운로드: <http://code.google.com/p/zxing/> 에서 [ZXing-2.0.zip](#)
- 소스코드를 다운로드 받은 후 압축을 풀어준다.
- 압축을 푼 곳으로 들어가 다음과 같이 code와 javase를 빌드 한다.
- ant가 설치되지 않아 에러가 날경우 ant를 먼저 설치하여 준다.
(\$sudo apt-get install ant)

* Ant

Ant는 Java용 빌드 툴 이다
<http://ant.apache.org>에서 다운받거나
 우분투에서 다음 명령어로 설치할 수 있다.
 \$sudo apt-get install ant

\$ cd core

\$ ant

Core라이브러리 컴파일

\$ cd ../javase

\$ ant

Javase interface용 라이브러리 컴파일

빌드가 성공하면 다음과 같이 터미널 상에서 실행하여 볼 수 있다.

\$ cd ..

\$ java -cp javase/javase.jar:core/core.jar com.google.zxing.client.j2se.CommandLineRunner
 [URL | FILE]

❖ (ex: \$ java -cp javase/javase.jar:core/core.jar com.google.zxing.client.j2se.CommandLineRunner
 \

Lab/Exam(II) – Barcode reader Android app 빌드

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 이번에 실습할 Barcode Reader app은 Zxing library를 사용하여 Android Device에서 실행되는 Android app으로써 Zxing의 /android/ 디렉토리에 들어 Android project이다.
- 여기에 있는 /android/ 프로젝트 + core.jar(Zxing) 가 마켓에 올라가 있는 barcode scanner app이다.
- 우선 다음과 같이 터미널 상에서 빌드 하여 보자. *단, Android Device에 미리 설치된 barcode scanner 는 본인이 signed 하여 컴파일 한 app이 아닐 경우 업데이트가 되지 않으므로 먼저 지워줘야 한다.*

\$ cd core

\$ ant build export

\$ cd ../android

\$ echo "sdk.dir=**android-sdk 설치 경로**" > local.properties

\$ ant debug

\$ ant debug install

컴파일된 core.jar을 /android/libs에 복사

Android app을 빌드 하려면 android-sdk가 필요하므로 설치된 경로를 지정

Debug모드로 빌드
(Release모드인 경우는 ant release)

Android Device에 app을 설치

Unable to resolve target 'android-xx' 에러가 나올 경우 Android-SDK 특정 버전이 깔리지 않아서 이다.
Eclipse에서 Android SDK Manager를 실행하여 필요한 SDK 버전을 설치해 준다.

빌드가 끝난 후 android/bin/ 에 들어가 보면 apk파일들이 생성 된 것을 볼 수 있다.

Lab/Exam(II) – Eclipse에서 빌드

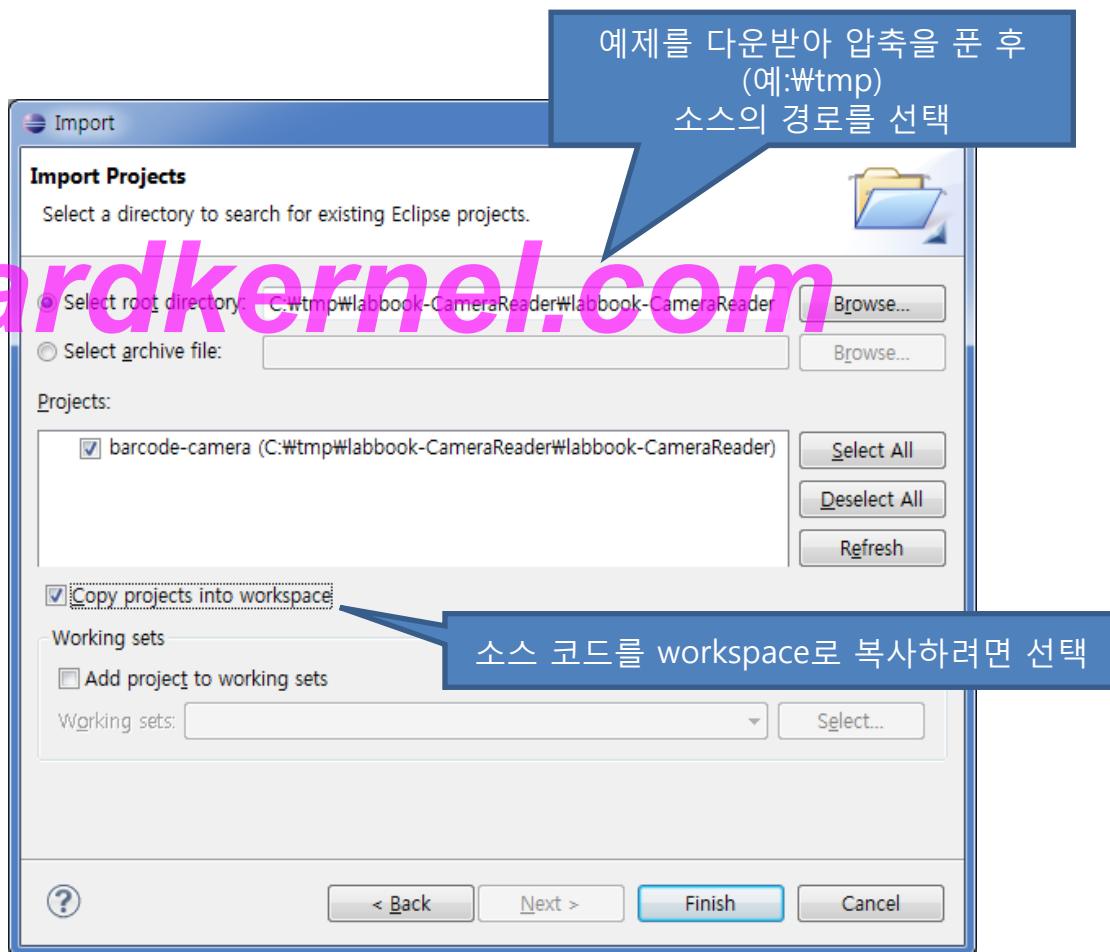
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- 이번 실습은 Eclipse 에서 빌드하여 보자 (labbook-CameraReader). Barcode Scanner와 같으며 모든 코드가 라이브러리가 아닌 java파일로 존재한다.

- Eclipse를 실행 후 메뉴 **File> Import> Existing Projects Into Workspace** 선택하면 다음과 같은 **Import Project**창이 표시된다.

- **Import Projects**
 - ✓ Select root directory : Import 하려는 source 디렉토리 선택한다.

- **Finish – Run!!**
 - ✓ 다음 그림과 같이 소스를 Import한다.
 - ✓ Eclipse에서 Run을 누르게 되면 Barcode Reader 어플리케이션이 디바이스에서 실행 하게 된다.



Lab/Exam(II) - Result Screenshot

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ 실행결과



센서 디바이스 드라이버의 구조

www.hardkernel.com

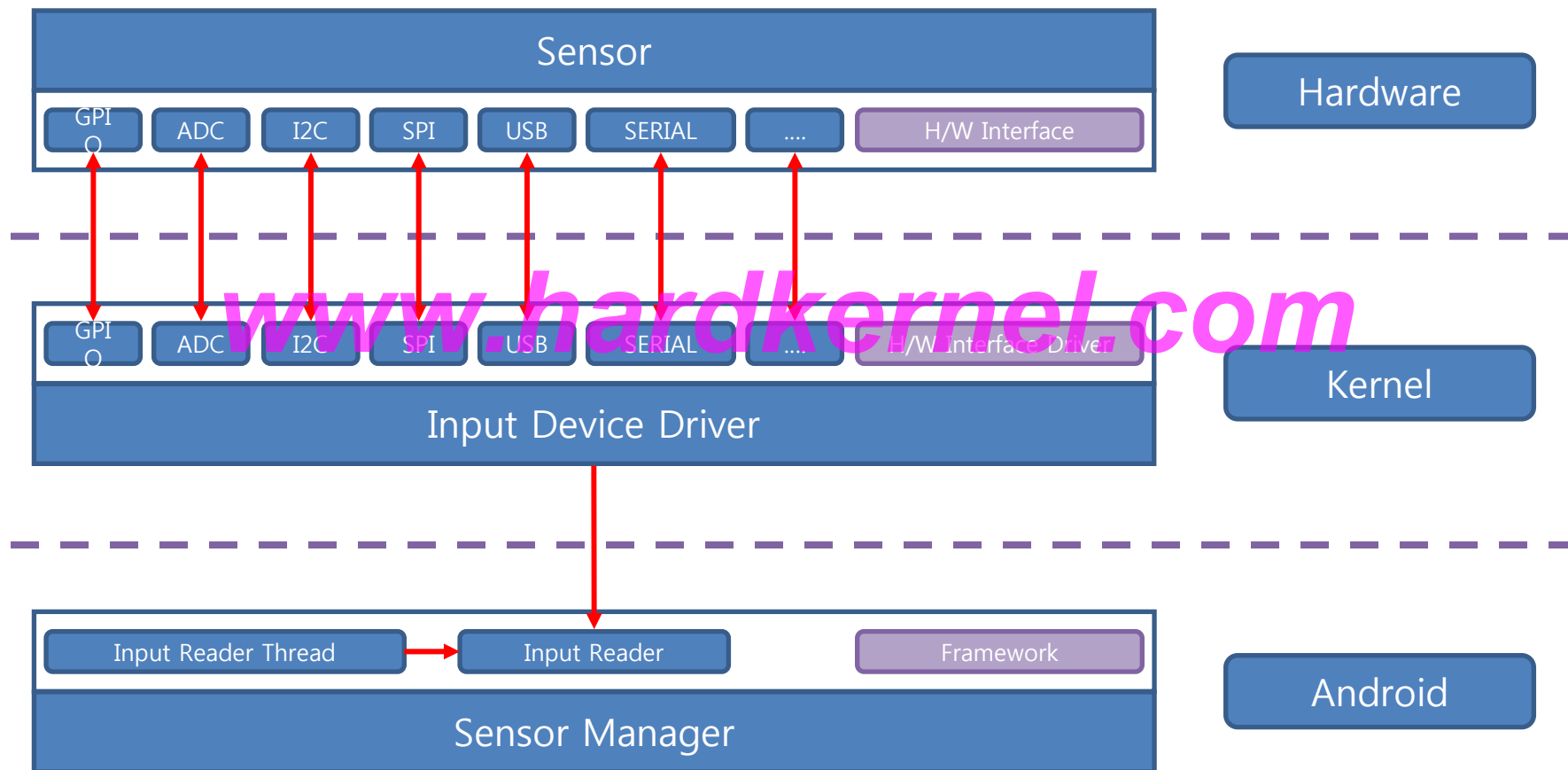
Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Sensor Data Flow
- Sensor Hardware Interface
- Sensor 드라이버 구조
- Sensor Platform data 설정
- Light Sensor(BH1780) Register
- Light Sensor driver probe
- Light Sensor driver data report
- Light Sensor driver remove
- Reference
- Lab/Exam
 - Accelerometer, Geo-magnetic field sensor, Gyroscope API and app

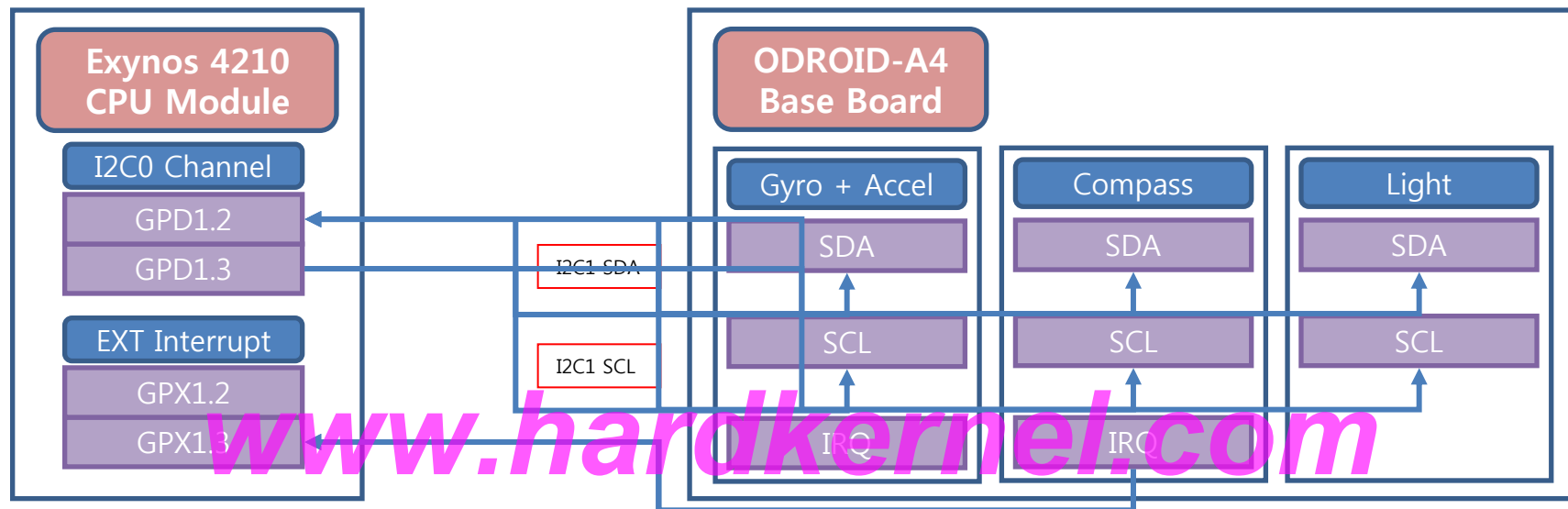
Sensor Data Flow

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

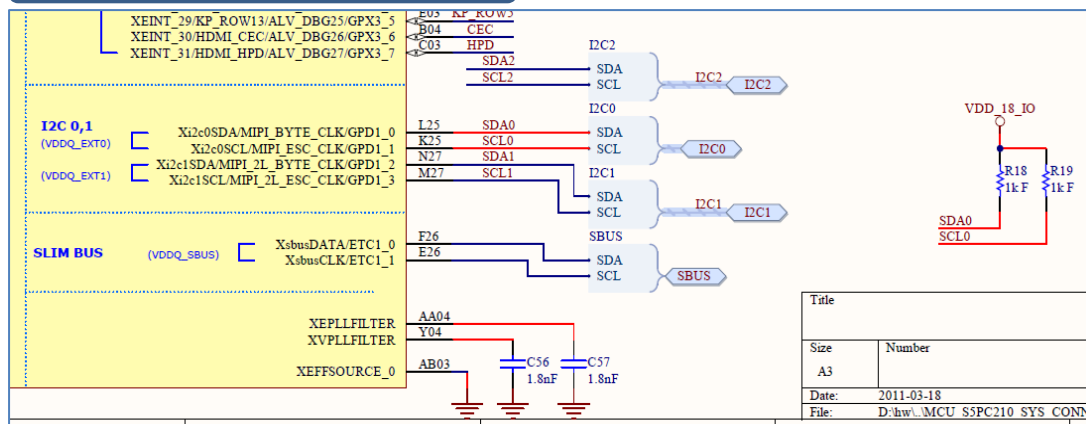


Sensor Hardware Interface

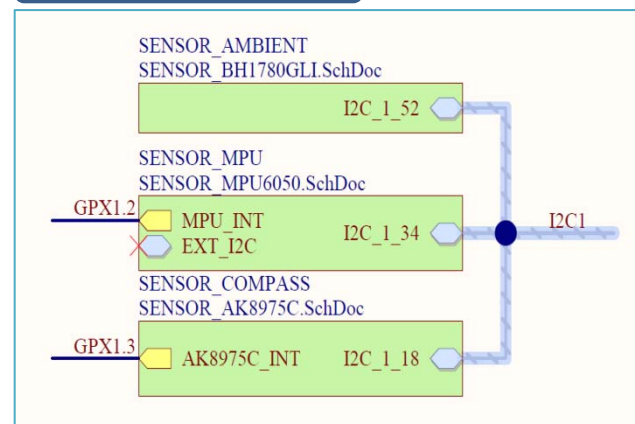
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



MCU_S5PC210_SYS_CONNECTIVITY.SchDoc

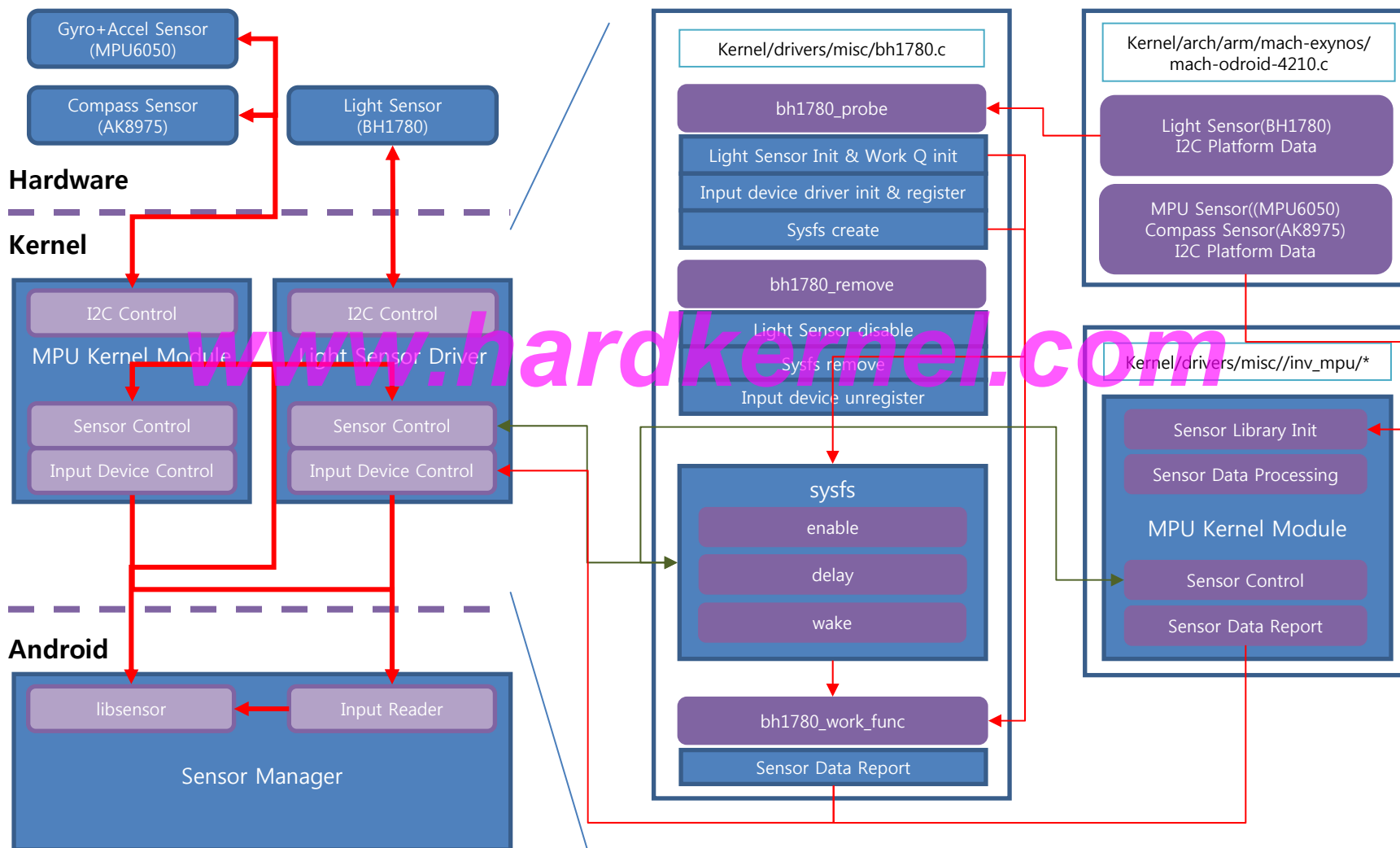


A4_BASE.SchDoc(A4_BASE)



Sensor 드라이버 구조

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Sensor Platform data 설정

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Odroid-A4 Sensor Platform data 설정

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
#ifdef CONFIG_MPU_SENSORS_MPU6050B1
#include <linux/mpu.h>

static struct mpu_platform_data mpu6050_data = {
    .int_config = 0x10,
    .level_shifter = 0,

    .orientation = {
        -1, 0, 0,
        0, +1, 0,
        0, 0, -1
    },
};

#ifdef CONFIG_MPU_SENSORS_AK8975
static struct ext_slave_platform_data mpu_compass_data = {
    .address = 0x0C,
    .adapt_num = 1,
    .bus = EXT_SLAVE_BUS_PRIMARY,

    .orientation = {
        0, 1, 0,
        1, 0, 0,
        0, 0, -1
    },
};
#endif // #if defined(CONFIG_MPU_SENSORS_AK8975)
#endif // #if defined(CONFIG_MPU_SENSORS_MPU6050B1)
...
```

MPU Kernel
Module
설정에 필요한
struct를 사용하기
위하여 등록

Gyro+Accel Sensor
H/W Mount Matrix 설정

Compass Sensor
H/W Mount Matrix 설정

```
...
static struct i2c_board_info i2c_devs1[] __initdata = {
...
#ifdef CONFIG_MPU_SENSORS_MPU6050B1
{
    I2C_BOARD_INFO("mpu6050", (0x68)),
    .irq = EXYNOS4_GPX1(2),

    .platform_data = &mpu6050_data,
},
#endif
#ifdef CONFIG_MPU_SENSORS_AK8975
{
    I2C_BOARD_INFO("ak8975", (0x0C)),
    .irq = EXYNOS4_GPX1(3),

    .platform_data = &mpu_compass_data,
},
#endif // #if defined(CONFIG_MPU_SENSORS_AK8975)
#endif // #if defined(CONFIG_MPU_SENSORS_MPU6050B1)

#ifdef CONFIG_SENSORS_BH1780
{
    I2C_BOARD_INFO("bh1780", (0x52 > 1)),
},
#endif
...
};
...
```

I2C 채널 1에 연결되어 있는
Device 정보 설정

Gyro+Accel Sensor
I2C Info 설정 및
Platform Data 설정

Compass Sensor
I2C Info 설정 및
Platform Data 설정

Light Sensor
I2C Info 설정

Light Sensor(BH1780) Register

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

BH1780 Register

Name	Description	Address (1 Bytes)	Value (1 Bytes)
CONTROL	Power OFF control	0x00	0x00
	Power ON control		0x03
PART ID	Part number ID	0x0A	0x81
Manufacture ID	Manufacture ID	0x0B	0x01
DATA LOW	Low byte of ADC	0x0C	-
DATA HIGH	High byte of ADC	0x0D	-

Command Register

7	6	5	4	3	2	1	0
1	XXX (Don't care)			Register Address			

LUX Calculation

7	6	5	4	3	2	1	0	DATA Low (0x0C)
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
7	6	5	4	3	2	1	0	DATA High (0x0D)
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	

Ex) DATA Low Byte = "1001_0000", DATA High Byte = "1000_0011"

$$\text{Lux} = (2^{15} + 2^9 + 2^8 + 2^7 + 2^4) = 33680 [\text{lx}]$$

Light Sensor driver probe(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Light Sensor driver probe

Kernel/arch/arm/mach-exynos/mach-odroid-4210.c

```
...
static struct i2c_board_info i2c_devs1[] __initdata = {
...
    #if defined(CONFIG_MPU_SENSORS_MPU6050B1)
    {
        I2C_BOARD_INFO("mpu6050", (0x68)),

        .irq = EXYNOS4_GPX1(2),

        .platform_data = &mpu6050_data,
    },

    #if defined(CONFIG_MPU_SENSORS_AK8975)
    {
        I2C_BOARD_INFO("ak8975", (0x0C)),

        .irq = EXYNOS4_GPX1(3),

        .platform_data = &mpu_compass_data,
    },
    #endif // #if defined(CONFIG_MPU_SENSORS_AK8975)
    #endif // #if defined(CONFIG_MPU_SENSORS_MPU6050B1)

    #if defined(CONFIG_SENSORS_BH1780)
    {
        I2C_BOARD_INFO("bh1780", (0x52 > 1)),
    },
    #endif
...
};
...
```

I2C 채널 1번에 연결되어 있는 Device의 정보를 기록한다.

Kernel/drivers/misc/bh1780.c

```
...
#define BH1780_NAME "bh1780"
...
static const struct i2c_device_id bh1780_id[] = {
    { BH1780_NAME, 0 },
};

MODULE_DEVICE_TABLE(i2c, bh1780_id);

static struct i2c_driver bh1780_driver = {
    .driver = {
        .name = BH1780_NAME,
        .owner = THIS_MODULE,
    },
    .probe = bh1780_probe,
    .remove = bh1780_remove,
    .suspend = bh1780_suspend,
    .resume = bh1780_resume,
    .id_table = bh1780_id,
};

...
static int bh1780_probe
(struct i2c_client *client, const struct i2c_device_id
*id)
{
...
}
```

I2C Driver의 ID정보를 설정한다. Platform driver register시 해당 ID를 검사한다.

Device Driver가 Register되면 I2c driver struct에 설정되어진 Probe함수를 호출한다.

Platform Driver 등록 이름과 Device Driver의 이름이 같아야 동작한다.

Light Sensor driver probe(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Light Sensor Init / Work Q init / Input Device Init

Kernel/drivers/misc/bh1780.c

```
...
#define BH1780_DATA_MIN 0
#define BH1780_DATA_MAX 0xFFFF
#define BH1780_DEFAULT_DELAY 150
#define BH1780_MAX_DELAY 200

static int bh1780_probe(struct i2c_client *client,
                        const struct i2c_device_id *id)
{
    ...
    dev_info(&client->dev, "%s found\n", id->name);
    dev_info(&client->dev, "part number=%d, rev=%d\n",
            ((err >> 4) & 0x0F), (err & 0x0F));

    bh1780_power_up(bh1780);
    bh1780_set_delay(&client->dev, BH1780_DEFAULT_DELAY);
    INIT_DELAYED_WORK(&bh1780->work, bh1780_work_func);
    if ((err = bh1780_input_init(bh1780)) < 0) goto error_1;

    if ((err = sysfs_create_group(&bh1780->input->dev.kobj,
                                &bh1780_attribute_group)) < 0)
        goto error_2;
    ...
}
```

Delayed Work Q의
실행 주기를 설정한
다.

```
...
static int bh1780_power_up(struct bh1780_data *bh1780)
{
    i2c_smbus_write_byte_data(bh1780->client,
                              (BH1780_COMMAND_REG + BH1780_CONTROL_REG),
                              BH1780_POWER_UP);
    msleep(200); return 0;
}
```

Light Sensor(BH1780)의
Control Register를 통하
여
Light Sensor(BH1780)를
Active상태로 전환한다.

```
...
static void bh1780_work_func(struct work_struct *work)
{
    ...
    bh1780_measure(bh1780);
    input_report_abs(bh1780->input, ABS_X, bh1780->light_data);
    input_sync(bh1780->input);
    ...
}
```

Light Sensor(BH1780)의
Data를 처리하는 Work Q
함수들 등록한다.

```
...
static int bh1780_input_init(struct bh1780_data *bh1780)
{
    ...
    dev->name = "light";
    dev->id.bustype = BUS_I2C;
    input_set_capability(dev, EV_ABS, ABS_MISC);
    input_set_abs_params(dev, ABS_X,
                        BH1780_DATA_MIN, BH1780_DATA_MAX, 0, 0);
    input_set_drvdata(dev, bh1780);
    err = input_register_device(dev);

    if (err < 0) { input_free_device(dev); return err; }

    bh1780->input = dev; return 0;
}
```

Light Sensor(BH1780)의
Input Device Driver를
초기화 하고 Register 한
다.

www.hardkernel.com

Light Sensor driver probe(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Sysfs create & register

Kernel/drivers/misc/bh1780.c

```
...
#define BH1780_DATA_MIN      0
#define BH1780_DATA_MAX     0xFFFF

#define BH1780_DEFAULT_DELAY 150
#define BH1780_MAX_DELAY    200

...
static int bh1780_probe(struct i2c_client *client,
                        const struct i2c_device_id *id)
{
    ...
    dev_info(&client->dev, "%s found\n", id->name);
    dev_info(&client->dev, "part number=%d, rev=%d\n",
            ((err >> 4) & 0x0F), (err & 0x0F));

    bh1780_power_up(bh1780);

    bh1780_set_delay(&client->dev, BH1780_DEFAULT_DELAY);
    INIT_DELAYED_WORK(&bh1780->work, bh1780_work_func);

    if ((err = bh1780_input_init(bh1780)) < 0) goto error_1;

    if ((err = sysfs_create_group(&bh1780->input->dev.kobj,
                                &bh1780_attribute_group)) < 0)
        goto error_2;

    ...
}
```

사용할 각각의
Sysfs file을
선언한다.

Sensor control을 위한
Sysfs 를 생성 및 등록한다.

```
...
static ssize_t bh1780_enable_show(struct device *dev,
                                struct device_attribute *attr, char *buf)
{
}

static ssize_t bh1780_enable_store(struct device *dev,
                                struct device_attribute *attr,
                                const char *buf, size_t count)
{
}

...
static DEVICE_ATTR(enable, S_IRWXUGO,
                    bh1780_enable_show, bh1780_enable_store);
static DEVICE_ATTR(delay, S_IRWXUGO,
                    bh1780_delay_show, bh1780_delay_store);
static DEVICE_ATTR(wake, S_IRWXUGO,
                    NULL, bh1780_wake_store);
static DEVICE_ATTR(data, S_IRWXUGO,
                    bh1780_data_show, NULL);

static struct attribute *bh1780_attributes[] = {
    &dev_attr_enable.attr,
    &dev_attr_delay.attr,
    &dev_attr_wake.attr,
    &dev_attr_data.attr,
    NULL
};

static struct attribute_group bh1780_attribute_group = {
    .attrs = bh1780_attributes
};

...
```

Sysfs file을 control시
실행되어질 함수를
등록한다.

Light Sensor data report

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Light Sensor Data Report

Kernel/drivers/misc/bh1780.c

```
...
static void bh1780_work_func(struct work_struct *work)
{
    struct bh1780_data *bh1780 =
        container_of((struct delayed_work *)work,
                     struct bh1780_data, work);
    unsigned long delay =
        delay_to_jiffies(atomic_read(&bh1780->delay));

    bh1780_measure(bh1780);

    input_report_abs(bh1780->input, ABS_X, bh1780->light_data);
    input_sync(bh1780->input);

    mutex_lock(&bh1780->data_mutex);
    bh1780->light_data_last = bh1780->light_data;
    mutex_unlock(&bh1780->data_mutex);

    schedule_delayed_work(&bh1780->work, delay);
}

...
static int bh1780_probe(struct i2c_client *client,
                       const struct i2c_device_id *id)
{
    ...
    INIT_DELAYED_WORK(&bh1780->work, bh1780_work_func);
    ...
}
```

Light Sensor의
Data Report

Delay에 적용된
시간 후에
Work Q 재시작

```
...
static int bh1780_measure(struct bh1780_data *bh1780)
{
    struct i2c_client *client = bh1780->client;
    int low_data, high_data;

    if(i2c_smbus_write_byte(client,
                           (BH1780_COMMAND_REG + BH1780_DATA_LOW_REG)) < 0)
    {
        dev_err(&client->dev, "I2C write byte error: data=0x%02x\n",
                (BH1780_COMMAND_REG + BH1780_DATA_LOW_REG));
        goto err;
    }
    if((low_data = i2c_smbus_read_byte(client)) < 0)
    {
        dev_err(&client->dev, "I2C read byte error\n"); goto err;
    }

    if(i2c_smbus_write_byte(client,
                           (BH1780_COMMAND_REG + BH1780_DATA_HIGH_REG)) < 0)
    {
        dev_err(&client->dev, "I2C write byte error: data=0x%02x\n",
                (BH1780_COMMAND_REG + BH1780_DATA_HIGH_REG));
        goto err;
    }
    if((high_data = i2c_smbus_read_byte(client)) < 0) {
        dev_err(&client->dev, "I2C read byte error\n"); goto err;
    }

    bh1780->light_data = BH1780_DATA_CAL(high_data, low_data);
err:
    return 0;
}
...
```

Light Sensor의
Data Register를 읽어온다.

Light Sensor driver remove

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Light Sensor remove

Kernel/drivers/misc/bh1780.c

```
...
#define BH1780_DATA_MIN      0
#define BH1780_DATA_MAX      0xFFFF

#define BH1780_DEFAULT_DELAY 150
#define BH1780_MAX_DELAY     200

...
static int bh1780_remove(struct i2c_client *client)
{
    struct bh1780_data *bh1780 = i2c_get_clientdata(client);
    bh1780_set_enable(&client->dev, 0);
    sysfs_remove_group(&bh1780->input->dev.kobj,
                      &bh1780_attribute_group);
    bh1780_input_fini(bh1780);

    kfree(bh1780);
    return 0;
}
...
```

생성된 Sysfs remove

```
...
static void bh1780_set_enable(struct device *dev, int enable)
{
    struct i2c_client *client = to_i2c_client(dev);
    struct bh1780_data *bh1780 = i2c_get_clientdata(client);

    int delay = atomic_read(&bh1780->delay);
    mutex_lock(&bh1780->enable_mutex);

    if (enable) {
        if (!atomic_cmpxchg(&bh1780->enable, 0, 1)) {
            bh1780_power_up(bh1780);
            schedule_delayed_work(&bh1780->work,
                                delay_to_jiffies(delay) + 1);
        } else {
            if (atomic_cmpxchg(&bh1780->enable, 1, 0)) {
                cancel_delayed_work_sync(&bh1780->work);
                bh1780_power_down(bh1780);
            }
        }
        atomic_set(&bh1780->enable, enable);
        mutex_unlock(&bh1780->enable_mutex);
    }
}

...
static void bh1780_input_fini(struct bh1780_data *bh1780)
{
    struct input_dev *dev = bh1780->input;

    input_unregister_device(dev);
    input_free_device(dev);
}
...
```

실행 대기중인 Work Q를
종료하고 Light Sensor
동작을 중지한다.

Input device driver
unregister

Reference(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

I2C Control API (include/linux/i2c.h)

Function

int i2c_smbus_write_byte(const struct i2c_client *client, unsigned char value);

Description

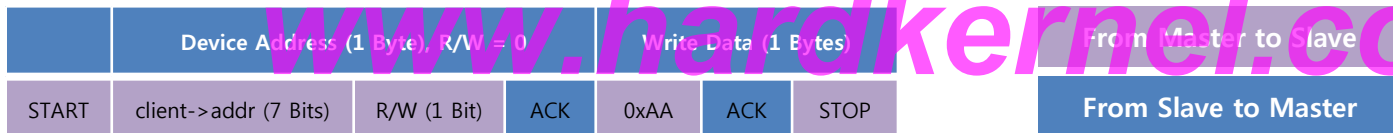
Device Address (Flag write)와 value(1 byte)를 전송

Return

에러인 경우 Negative value를 return, 유효한 경우 0을 되돌려 준다.

Usage

ret = **i2c_smbus_write_byte(client, 0xAA);**



Function

int i2c_smbus_read_byte(const struct i2c_client *client);

Description

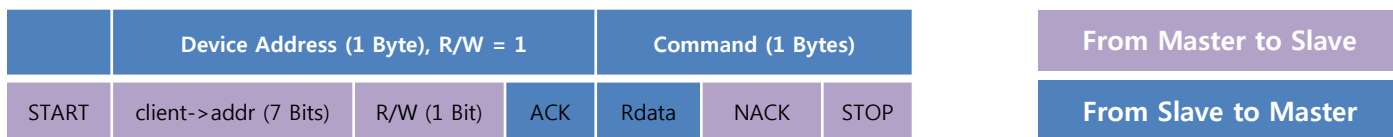
Device Address (Flag read)를 전송하고 1 byte를 읽어서 되돌려 준다.

Return

에러인 경우 Negative value를 return, 유효한 경우 읽어낸 byte 값을 되돌려 준다.

Usage

ret = **i2c_smbus_read_byte(client);**



Reference(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

I2C Control API (include/linux/i2c.h)

Function

```
int i2c_smbus_write_byte_data(const struct i2c_client *client, unsigned char command,
                             unsigned char value);
```

Description

Device Address (Flag write)를 포함한 byte command와 byte data를 전송

Return

에러인 경우 Negative value를 return, 유효한 경우 0을 되돌려 준다.

Usage

```
ret = i2c_smbus_write_byte_data(client, 0xA0, 0x06);
```



- 더 많은 정보는 `kernel/drivers/i2c/i2c-core.c` 파일의 내용을 참조

Reference(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Work Q 사용 예제

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/workqueue.h>

#define DELAY_1SEC 1000

struct delayed_work delay_work;
struct work_struct work;

irqreturn_t my_irq(int irq, void *handle) // Interrupt callback function
{
    schedule_delayed_work(&delay_work, msecs_to_jiffies(DELAY_1SEC));
    schedule_work(&work);
    return IRQ_HANDLED;
}

int init_module(void)
{
    INIT_WORK(&work, work_func);
    INIT_DELAYED_WORK(&delay_work, delay_work_func);

    unsigned int irqnum = gpio_to_irq(EXYNOS4_GPX3(0));
    unsigned long irqflags = IRQF_TRIGGER_FALLING | IRQF_DISABLED;

    if(request_irq(irqnum, my_irq, irqflags, "my irq", NULL)) {
        printk(KERN_ERR "my_device: cannot register IRQ %d\n", irqnum);
        return -EIO;
    }
    return 0;
}
```

지연시간을 있는
Work Q 변수 선언

지연시간을 없는
Work Q 변수 선언

static void work_func(struct work_struct *work)

```
{
    printk("%s\n", __func__);
}
```

지연시간을 없는
Work Q 함수

static void delay_work_func(struct work_struct *work)

```
{
    printk("%s\n", __func__);
}
```

지연시간을 있는
Work Q 함수

지연시간을 있는
Work 함수 실행
지연시간을 없는
Work 함수 실행

지연시간을 없는
Work Q 초기화

지연시간을 있는
Work Q 초기화

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Accelerometer, Geo-magnetic field sensor, Gyroscope API and app
- Sensor support in Android is fairly simple yet versatile. It is handled via the `SensorManager` which is provided from your app context via `getSystemService()`. Once you have the sensor manager, you have to register to get its notifications. To do this, we implement the `SensorListener` interface.
- Notice that we register to get sensor updates in `onResume()` and unregister in `onPause()`. This is important. Sensors data come in erratic intervals and can consume a lot of CPU and battery power. To optimize our app for performance, it is the best practice to get the notifications just while your app is in running state.
- The notifications come via `onSensorChanged()` and `onAccuracyChanged()`. Notice that each has a parameter for the sensor reporting. Android sensor API can support a large number of sensors, each one having its own unique integer ID.
- The current ones are expressed via their constants, such as:
 - `Sensor.TYPE_ACCELEROMETER`
 - `Sensor.TYPE_ACCELEROMETER.`
 - `Sensor.TYPE_GYROSCOPE.`
 - `Sensor.TYPE_ORIENTATION.`

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ More info:

<http://developer.android.com/reference/android/hardware/SensorManager.html>

<http://developer.android.com/reference/android/hardware/Sensor.html>

<http://developer.android.com/guide/topics/sensors/index.html>

➤ As an example we propose two applications:

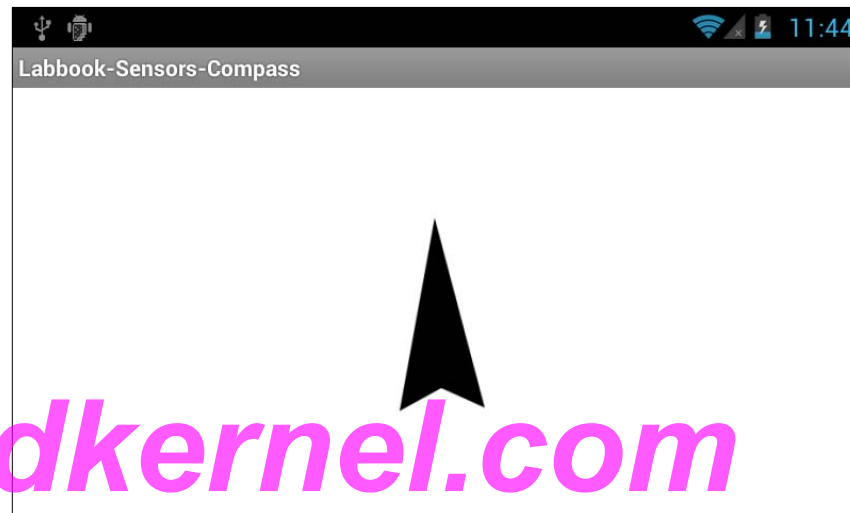
- Compass:

www.hardkernel.com

- Game:

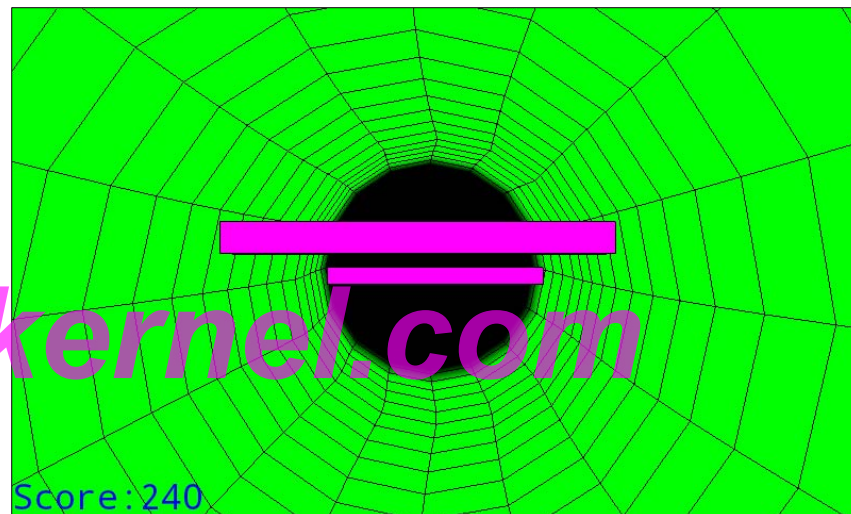
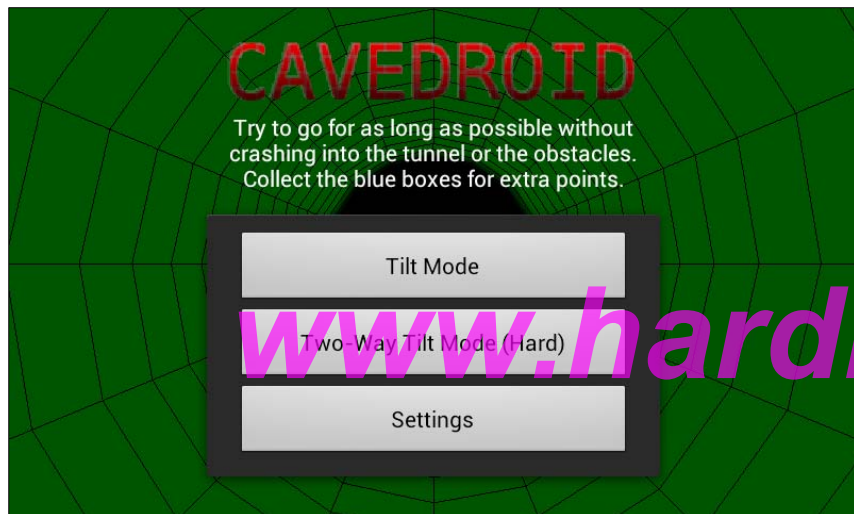
Lab/Exam(3) : Compass

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Lab/Exam(4) : Game

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0





Battery gauge and charger drivers
www.hardkernel.com

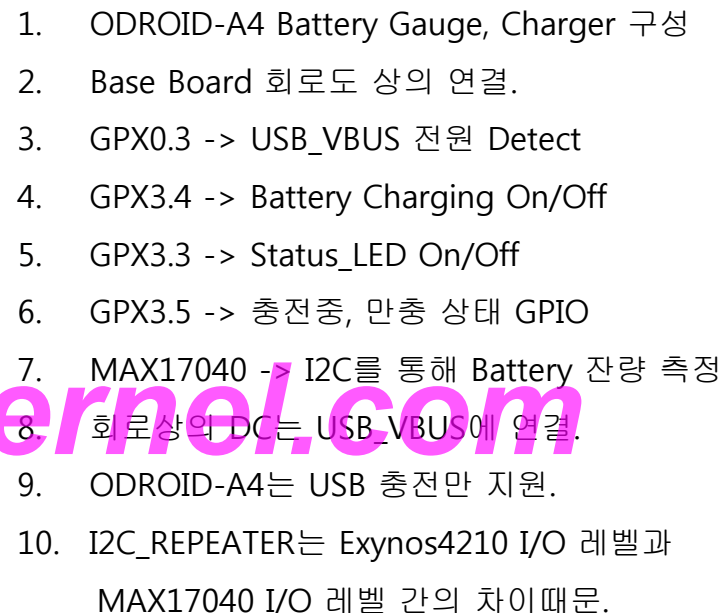
Agenda

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- ODROID-A4 Battery-Gauge, Charger 구성
- Max17040 Fuel-Gauge, MAX8677 Charger
- MAX17040_battery driver
- MAX8677a_charger driver
- Fuel-Gauge / Charger driver
- Lab/Exam.
 - Battery Monitor

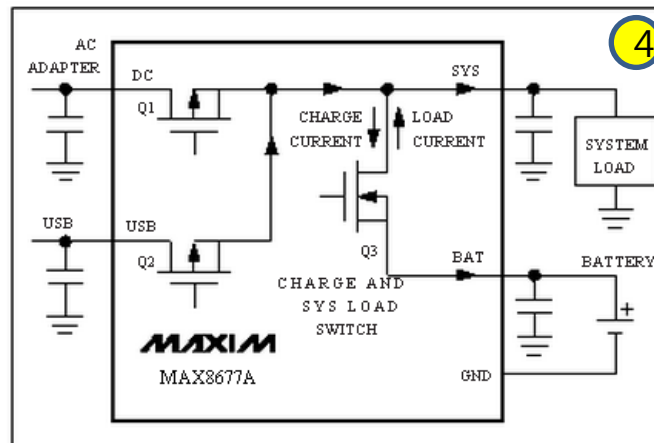
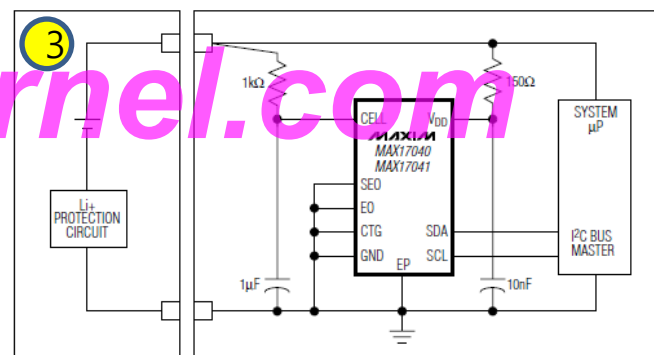
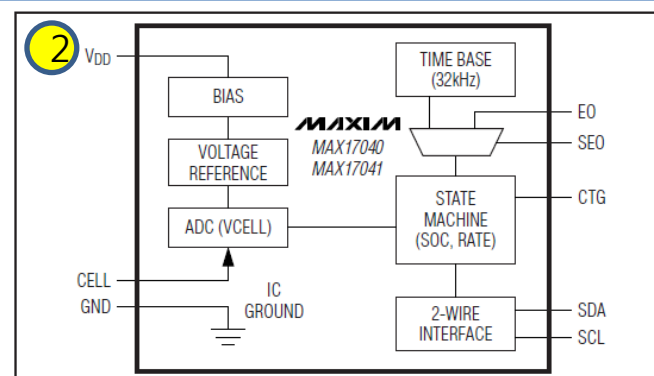
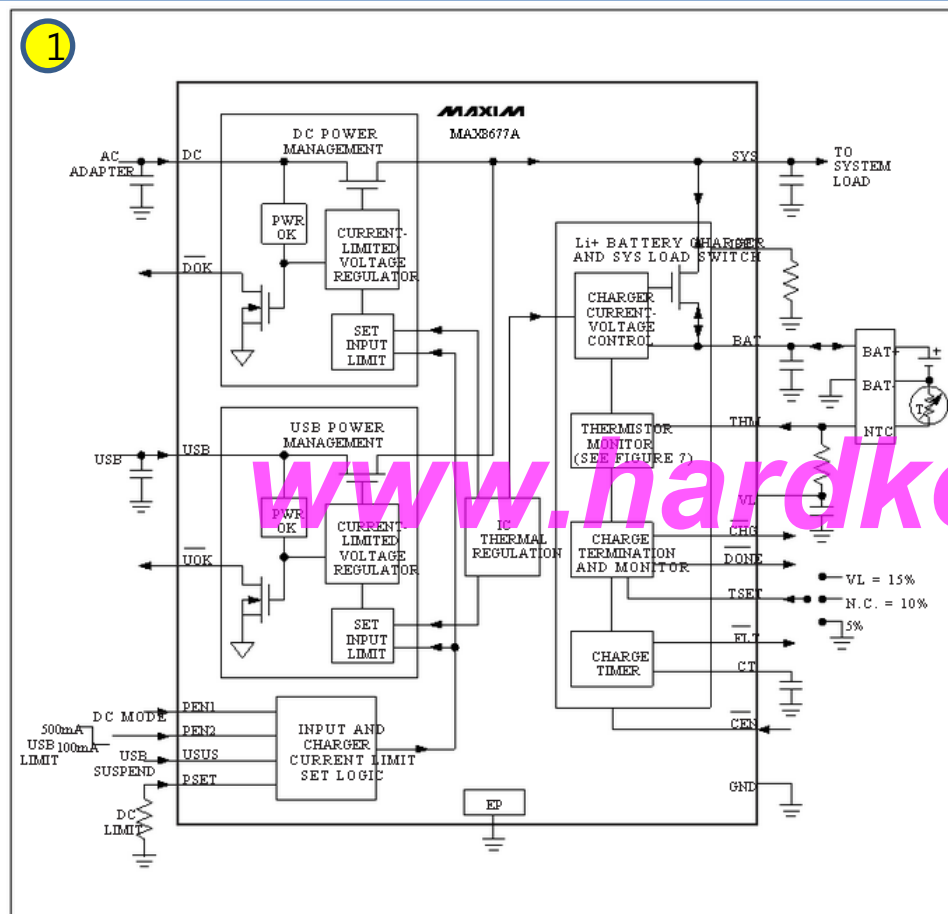
www.hardkernel.com

ODROID-A4



Max17040 Fuel-Gauge, MAX8677 Charger

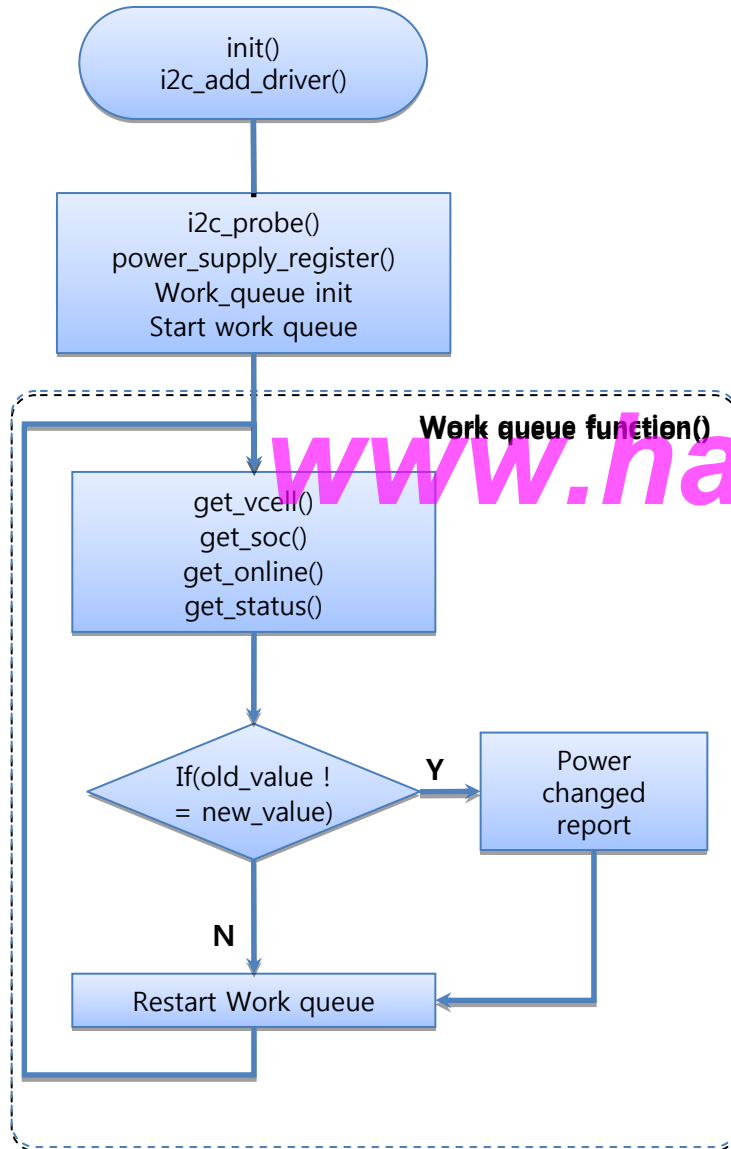
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



1. MAX8677A Charger Block diagram.
2. MAX17040 Fuel-Gauge Block diagram
3. MAX17040 Fuel-Gauge Operating Circuit.
4. MAX8677A Charger Operating Circuit.

MAX17040_battery driver

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



\$KERNEL_ROOT/drivers/power/ max17040_battery.c

```

static int __init max17040_init(void)
{
    return i2c_add_driver(&max17040_i2c_driver);
}

static int __devinit max17040_probe(struct i2c_client *client,
                                   const struct i2c_device_id *id)
{
    ret = power_supply_register(&client->dev, &chip->battery);

    INIT_DELAYED_WORK_DEFERRABLE(&chip->work, max17040_work);
    schedule_delayed_work(&chip->work, MAX17040_DELAY);
    return 0;
}

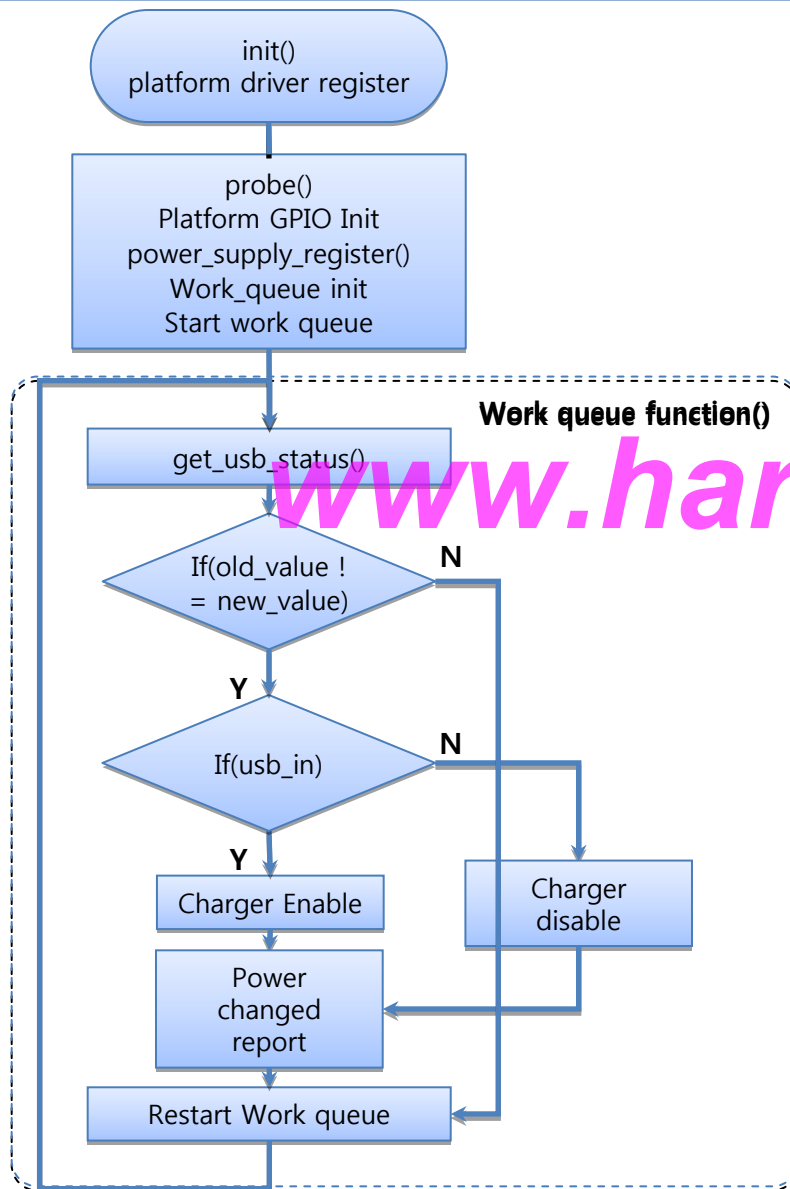
static void max17040_work(struct work_struct *work)
{
    int old_vcell = chip->vcell, old_soc = chip->soc;

    max17040_get_vcell(chip->client);
    max17040_get_soc(chip->client);
    max17040_get_online(chip->client);
    max17040_get_status(chip->client);

    if((old_vcell != chip->vcell) || (old_soc != chip->soc)){
        power_supply_changed(&chip->battery);
    }
    schedule_delayed_work(&chip->work, MAX17040_DELAY);
}
  
```

MAX8677a_charger driver

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



\$KERNEL_ROOT/drivers/power/ max17040_battery.c

```

static int __init max8677_init(void) {
    return platform_driver_register(&max8677_driver);
}

static __devinit int max8677_probe(struct platform_device *pdev)
{
    if(ret = power_supply_register(dev, &data->usb))

    INIT_DELAYED_WORK_DEFERRABLE(&data->work, max8677_work);
    schedule_delayed_work(&chip->work, MAX8677_DELAY);
    return 0;
}

static void max8677_usb_status(struct max8677_data *data)
{
    bool usb_in;
    struct max8677_pdata *pdata = data->pdata;

    usb_in = gpio_get_value(pdata->uok) ? false : true;
    if (data->usb_in == usb_in)
        return;
    data->usb_in = usb_in;
    /* Charger Enable / Disable (cen is negated) */
    if (pdata->cen)
        gpio_set_value(pdata->cen, usb_in ? 0 : 1);

    power_supply_changed(&data->usb);
}

static void max8677_work(struct work_struct *work)
{
    int old_vcell = chip->vcell, old_soc = chip->soc;

    max8677_usb_status(data);

    schedule_delayed_work(&data->work, MAX8677_DELAY);
}
  
```


Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Battery Monitor

How to get the Battery Information:

```
public class Main extends Activity {
    private TextView contentTxt;
    private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver(){
        @Override
        public void onReceive(Context arg0, Intent intent) {
            // TODO Auto-generated method stub
            int level = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            int scale = intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
            int temp = intent.getIntExtra(BatteryManager.EXTRA_TEMPERATURE, -1);
            int voltage = intent.getIntExtra(BatteryManager.EXTRA_VOLTAGE, -1);
            contentTxt.setText("level is "+level+"/"+scale+", temp is "+temp+", voltage is "+voltage);
        }
    };

    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.main);
        contentTxt = (TextView) this.findViewById(R.id.batteryTxt);
        this.registerReceiver(this.mBatInfoReceiver,
            new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
    }
}
```

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Quick Explanation

Check a Intent (ACTION_BATTERY_CHANGED) and register a receiver when the intent is fired. In ActionScript, this is DispatchEvent, we call mBatInfoReceiver and get the current level of our battery life and put that int value to our textfield or TextView.

➤ More info:

<http://developer.android.com/reference/android/os/BatteryManager.html>

www.hardkernel.com

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0





WiFi device driver and HAL
www.hardkernel.com

Agenda

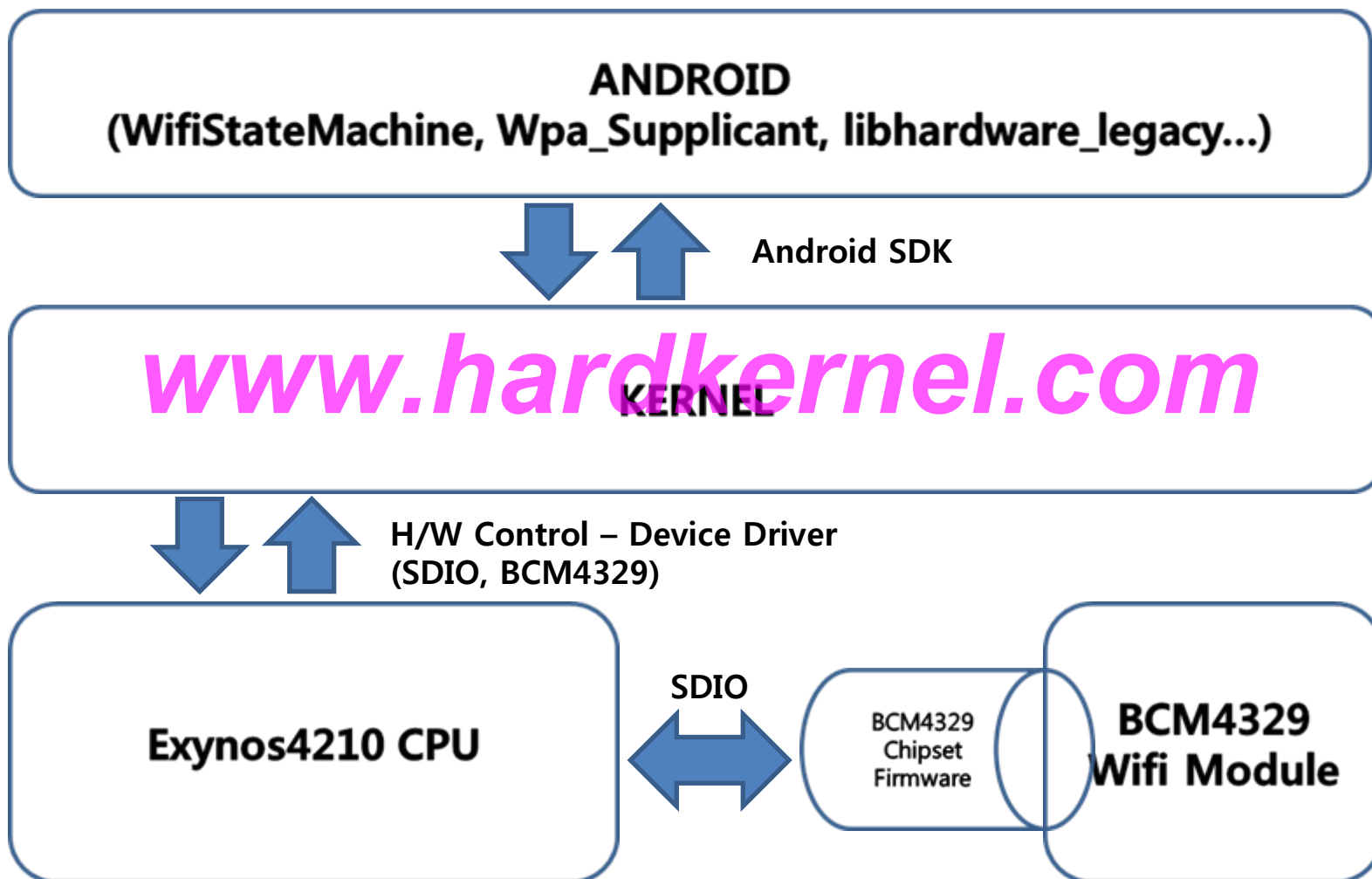
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Wifi BCM4329 Block Diagram
- Wifi Turn-on sequence
 - BCM4329 Power on
 - Insmo BCM4329 Module
- Wifi Connection Process
 - Wifi Connection Block Diagram
 - Communication between Android and Kernel
- Lab/Exam :
 - WiFi AP Scan
 - WiFi API

www.hardkernel.com

WiFi BCM4329 Block Diagram

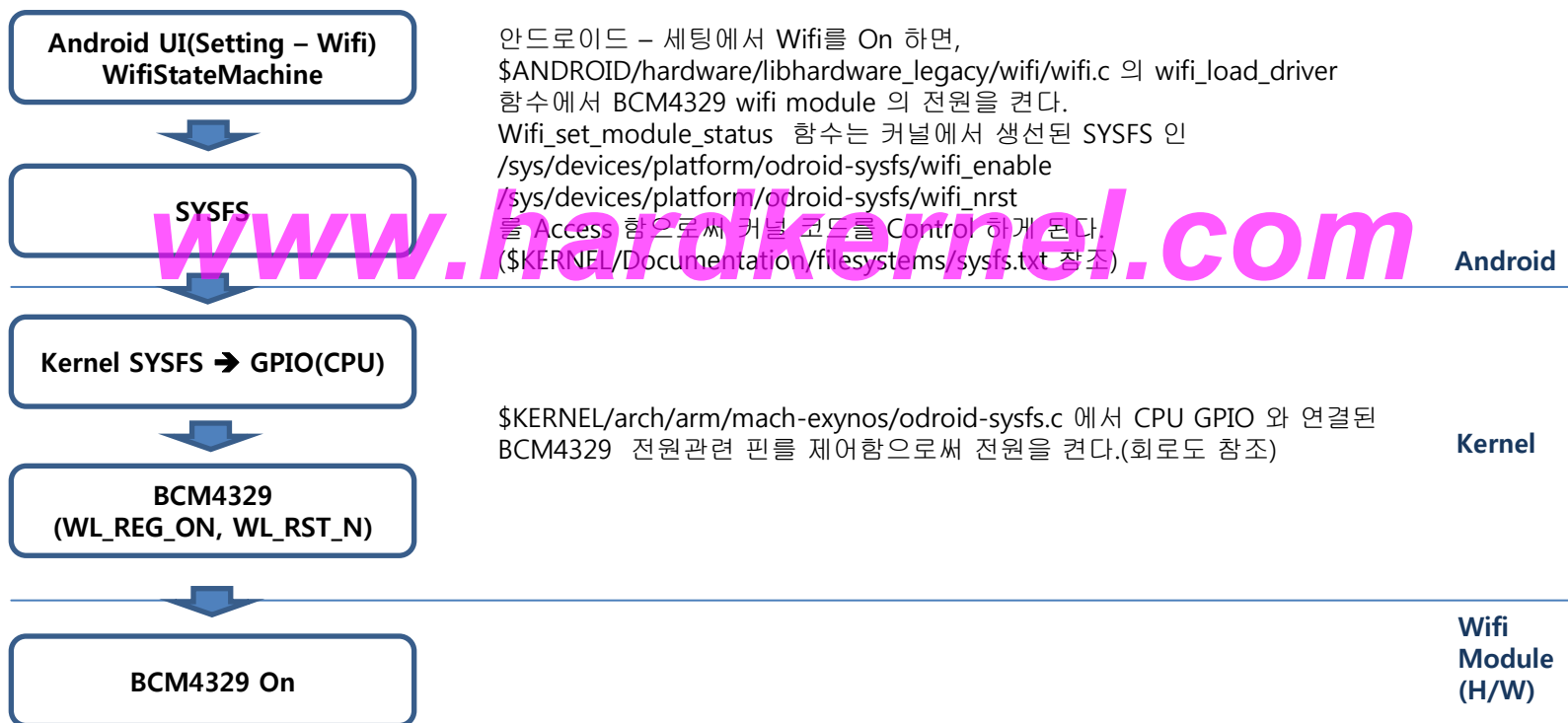
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



WiFi Turn-on sequence(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

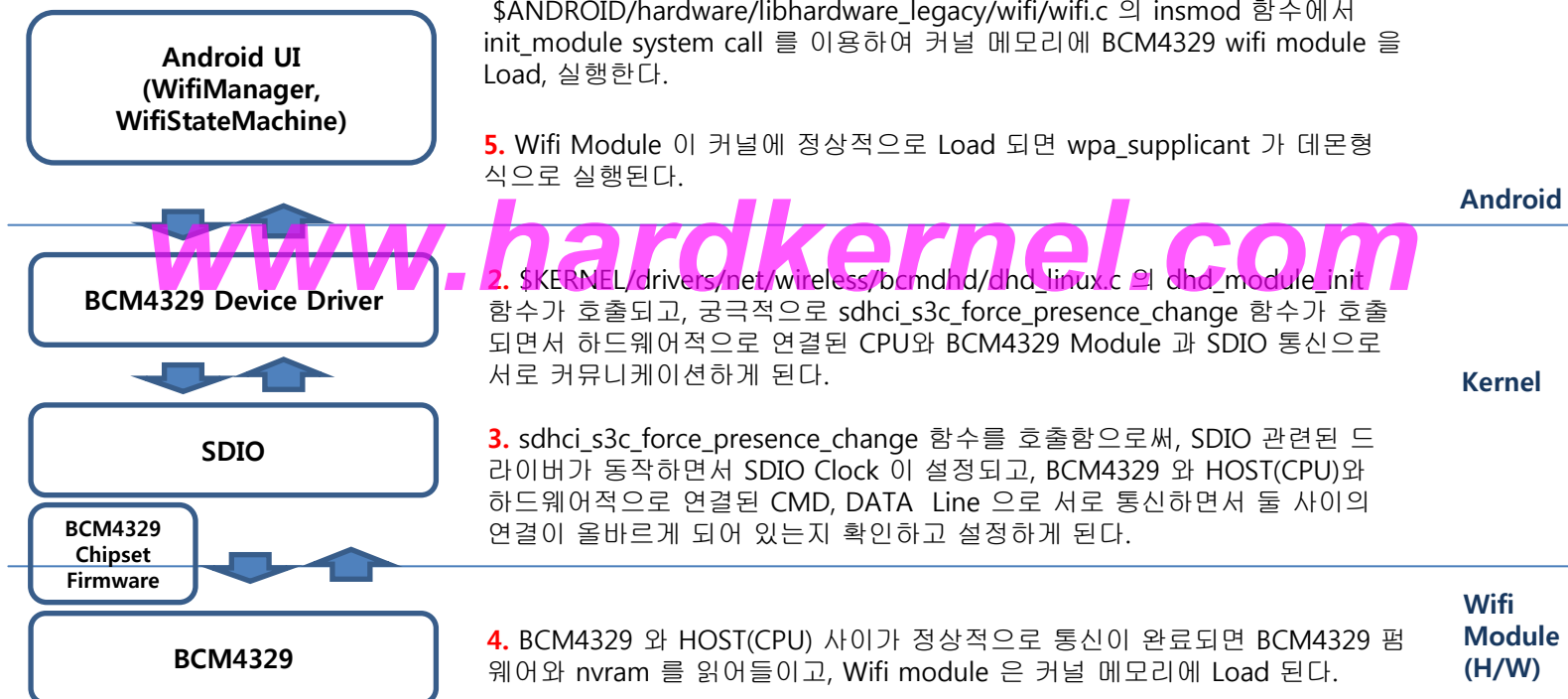
- 안드로이드 - 설정에서 WiFi On 을 했을 때 벌어지는 일들을 순차적으로 위(안드로이드 UI) 부터 아래(Hardware Control S/W) 까지의 과정은 다음과 같다.
- Power on BCM4329



WiFi Turn-on sequence(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

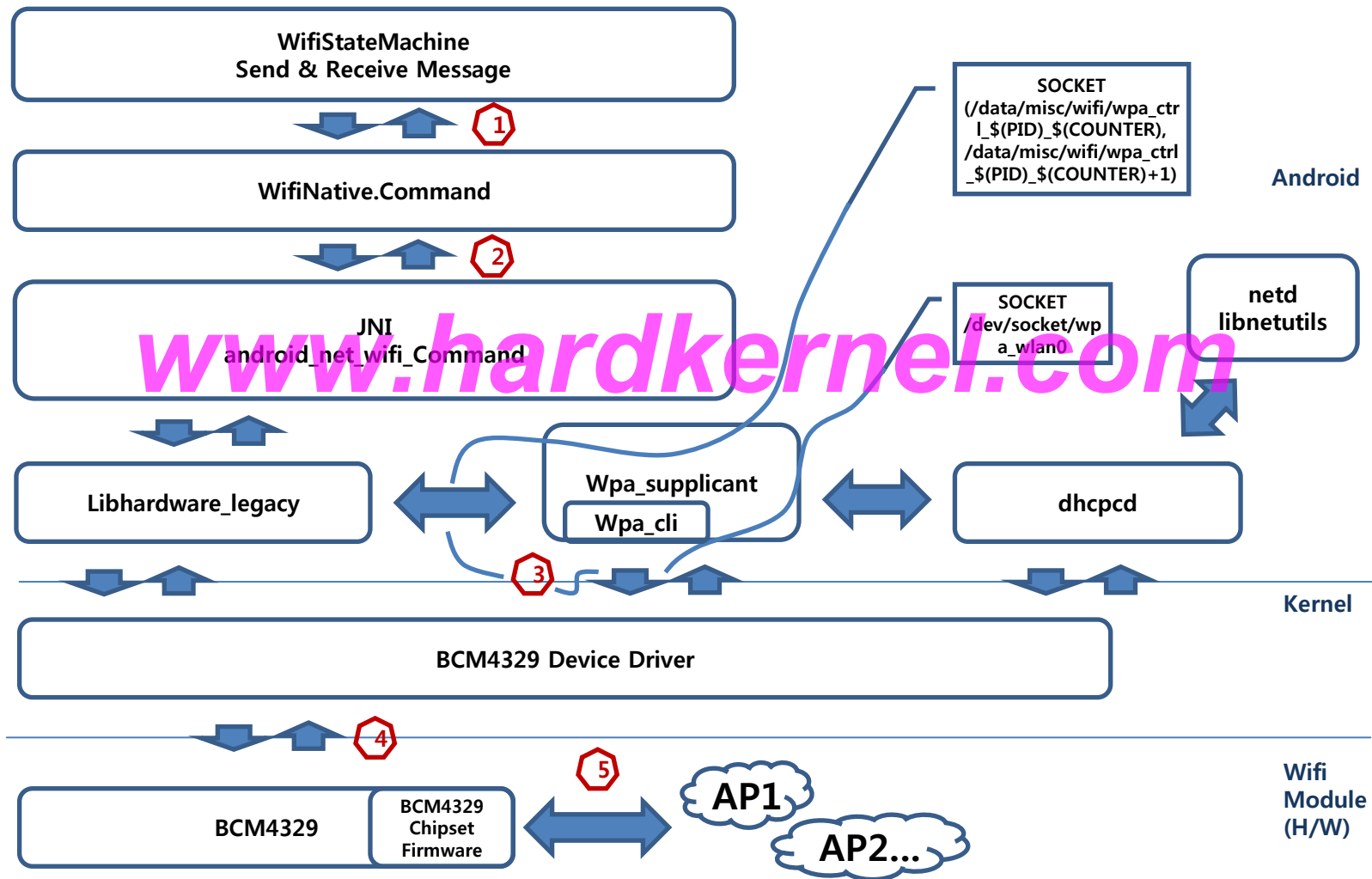
➤ Insmod BCM4329 Module



WiFi Connection Process(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Wifi Connection Block Diagram



WiFi Connection Process(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Communication between Android and Kernel

1. 안드로이드 WifiStateMachine 은 쓰레드 형식으로 실행되어 Wifi Driver Loading, WPA_SUPPLICANT, Android UI 등의 상태를 메시지 형태의 CMD 로 받게 된다. Case CMD: 에 해당하는 상태를 갱신한 후 Wifi Driver 에서 처리되어야 하는 CMD 를 아랫단으로 전달하게 된다.

(\$ANDROID/frameworks/base/wifi/java/android/net/wifi/*)

2. 안드로이드 WifiNative 는 전달받은 CMD 를 native 함수로 연결해주고,

(\$ANDROID/ frameworks/base/core/jni /android_net_wifi_Wifi.cpp)

3. Libhardware_legacy 안드로이드 라이브러리는 socket 으로 연결되어 있는 wpa_supplicant 로 CMD를 보내고, wpa_supplicant 는 socketcall system call 를 이용하여 Kernel 로 CMD를 보내게 된다.

3-1. wpa_supplicant 의 기능은 AP 와 Wifi Module 간 접속 방식 및 암호를 확인하는 절차(4WAY_HANDSHAKE) 등을 제공함으로써, 둘 사이 연결에 신뢰성을 제공한다.

3-2. WPA_SUPPLICANT 에 의해 신뢰성이 확인되면, DHCPD client 를 실행함으로써, DHCP Server 에서 IP ADDRESS 를 할당 받게 되고, netd 에 의해 DNS 관련 세팅이 이뤄지게 된다.

(\$ANDROID/hardware/libhardware_legacy/wifi/wifi.c)

(\$ANDROID/external/wpa_supplicant_8, \$ANDROID/external/dhcpd)

(\$ANDROID/frameworks/base/core/java/android/net/*)

(\$ANDROID/system/core/libnetutils, \$ANDROID/system/netd)

WiFi Connection Process(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

4. 커널 BCM4329 Device Driver 는 CMD 를 SDIO 통신을 이용하여 BCM4329(H/W) 에게 전달한다.
 (\$KERNEL/drivers/net/wireless/bcmdhd/*)
 (\$KERNEL/drivers/mmc/*)
5. BCM4329(H/W) 의 firmware code 는 주변의 AP 와 통신한다.
 (\$ANDROID/device/hardkernel/proprietary/bin/bcm4329.hcd : BCM4329 firmware)
 (\$ANDROID/device/hardkernel/proprietary/bin/bcm4329.cal : nvram – BCM4329 configuration)

www.hardkernel.com

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- WiFi AP Scan
- Create the attribute of the WifiManager.
`private WifiManager mWifiManager;`
- Then initialize the attribute:
`mWifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);`
- You should know how to check the status of WiFi and how to change:
`boolean state = mWifiManager.isWifiEnabled();`
`mWifiManager.setWifiEnabled(BOOLEAN STATE);`
- To start scanning WiFi networks:
`mWifiManager.startScan();`
- To obtain the list of networks to create a list and save the results:
`List<ScanResult> wifiList = mWifiManager.getScanResults();`

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ WiFi control API

➤ More info:

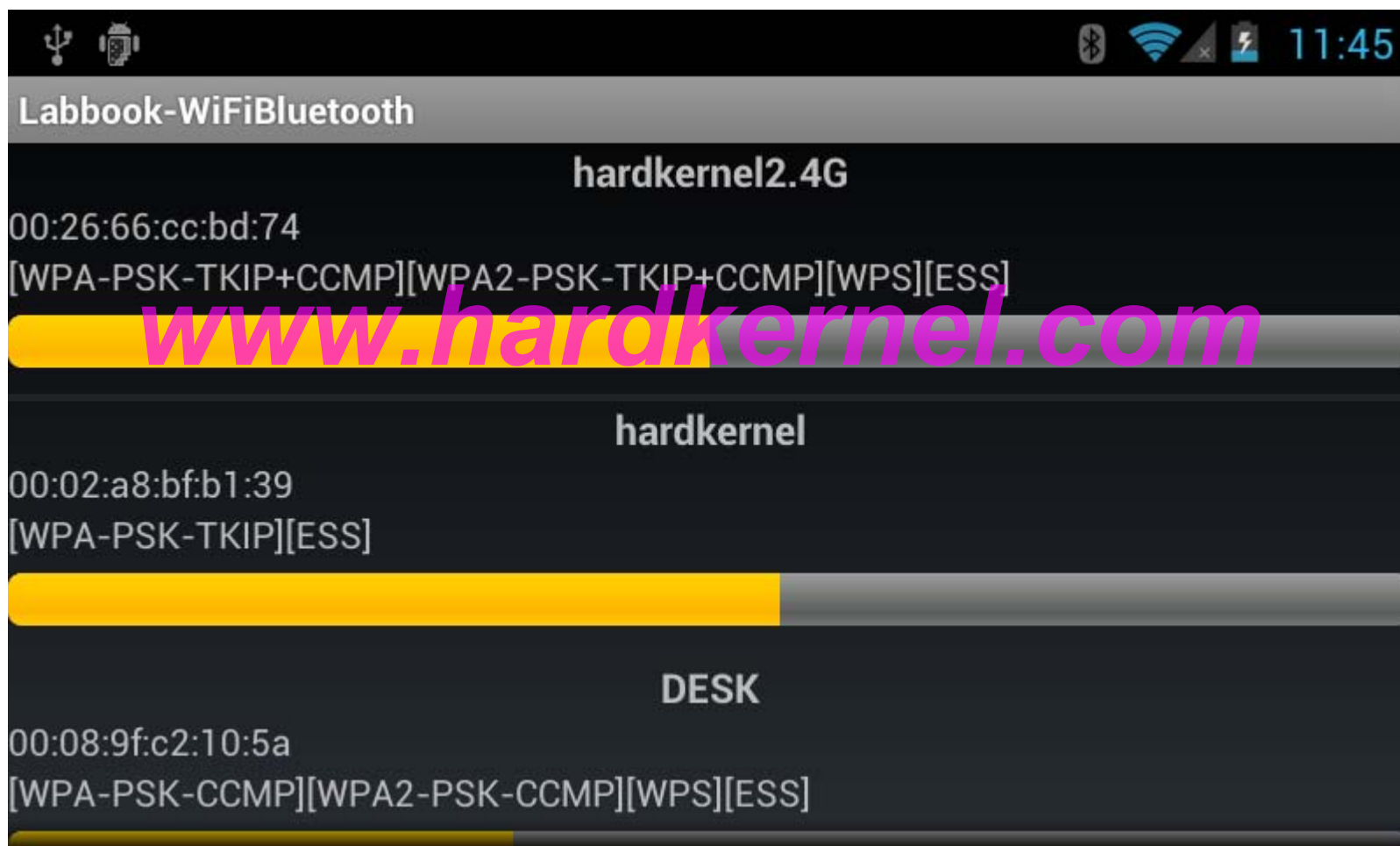
<http://developer.android.com/reference/android/net/wifi/WifiManager.html>

➤ If you want to use WiFi for exchange of information between two devices, following information is recommended:

<http://developer.android.com/guide/topics/wireless/wifi2p.html>

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0





Bluetooth device driver and HAL

www.hardkernel.com

Agenda

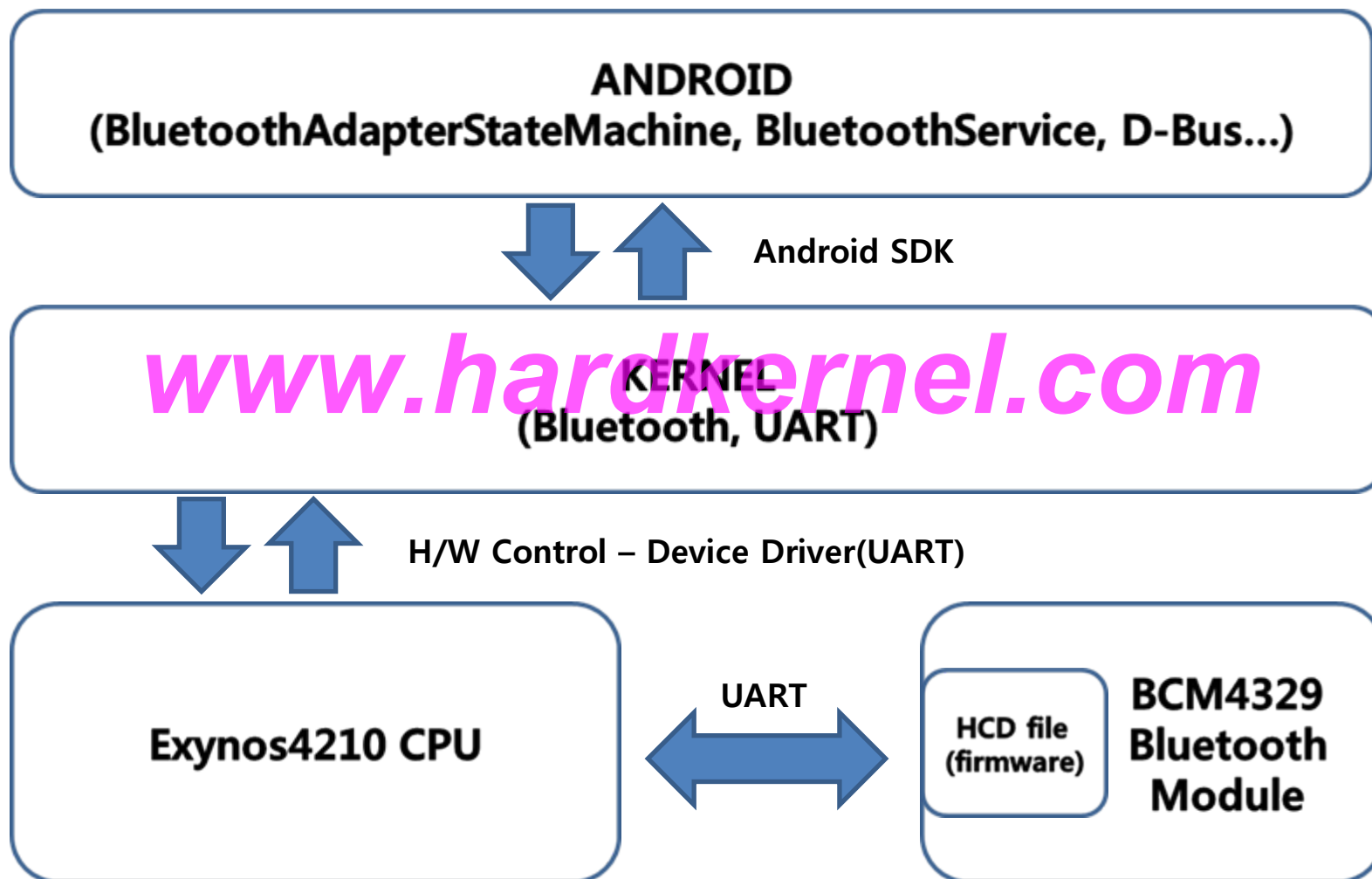
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- BCM4329 Bluetooth Block Diagram
- BCM4329 Bluetooth Turn-on sequence
 - 2-1. Power on
 - 2-2. Bluetooth Preparation
- Bluetooth Connection Process
 - 3-1. BCM4329 Bluetooth Connection Block Diagram
 - 3-2. Communication between Android and Kernel
- Lab/Exam
 - Bluetooth Control API

www.hardkernel.com

BCM4329 Bluetooth Block Diagram

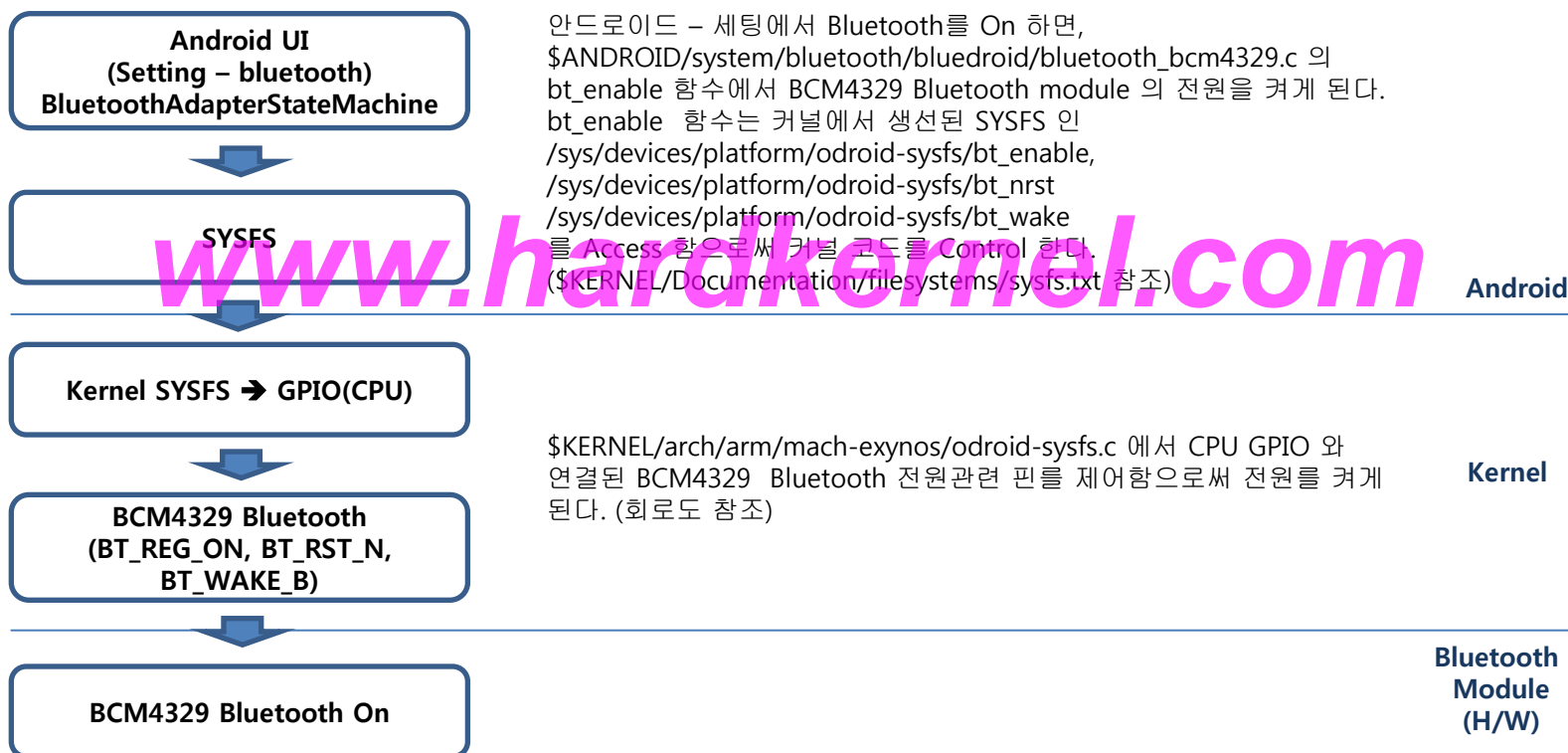
www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0



Bluetooth Turn-on squence(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

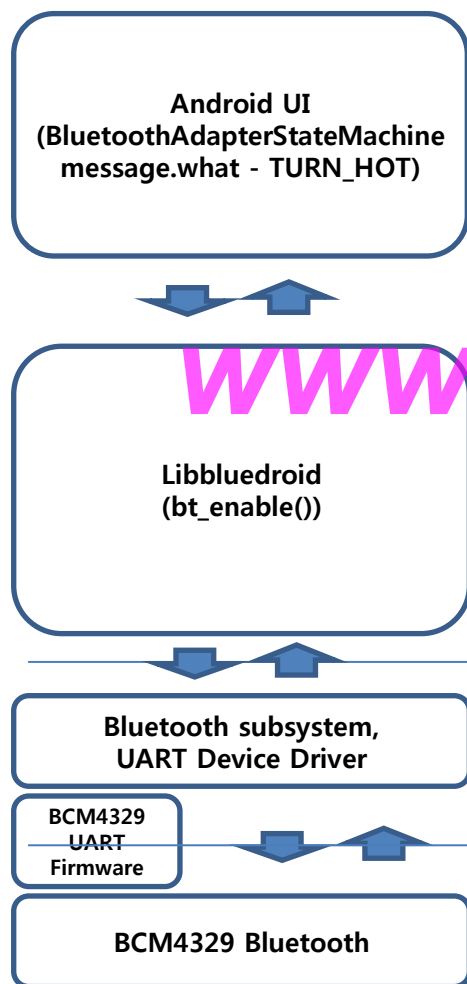
- 안드로이드 - 설정에서 Bluetooth On 을 했을 때 일어나는 일들을 순차적으로 위(안드로이드 UI) 부터 아래 (Hardware Control S/W) 까지 기술하면 다음과 같다.
- Power on BCM4329 Bluetooth



Bluetooth Turn-on squence(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Bluetooth Preparation



1. Android system_server 는 안드로이드 부팅시 Bluetooth Service 를 시작한다.
안드로이드 Framework 의 Server 중 BluetoothService 는 BluetoothAdapterStateMachine class 에 TURN_HOT MSG 를 던지며 호출한다.
BluetoothAdapterStateMachine 의 생성자는 PowerOff class 에 TURN_HOT MSG 를 전달하면서 Preparation 를 시작하게 된다.
이 과정의 의미를 BluetoothAdapterStateMachine.java 의 주석을 인용한다.

```
// Turn on Bluetooth Module, Load firmware, and do all the preparation
// needed to get the Bluetooth Module ready but keep it not discoverable
// and not connectable. This way the Bluetooth Module can be quickly
// switched on if needed
```

2. BluetoothAdapterStateMachine 는 Native Code jni/android_server BluetoothService.cpp 를 거쳐 bt_enable 함수를 실행한다.

bt_enable 함수에서 Power on BCM4329 Bluetooth 에서 설명한 대로 Power ON 이 이뤄지고, hciattach 프로세스가 데몬형식으로 실행되며, socket 를 생성하여 그 socket 으로 IO control 를 수행하여 Kernel 의 \$KERNEL/net/Bluetooth code 와 통신이 이루어진다.

Kernel 과 원활하게 통신이 이루어지면, bluetoothd 프로세스가 데몬형식으로 실행된다.

Android

3. 안드로이드의 IO control 를 통해서 받은 MSG는 Kernel 의 Bluetooth subsystem(\$KERNEL/net/bluetooth) 에서 커널부팅시 initialized 된 HCI(Host Communication Interface) socket(hdev->name:hci0) 를 open 한다.

Kernel

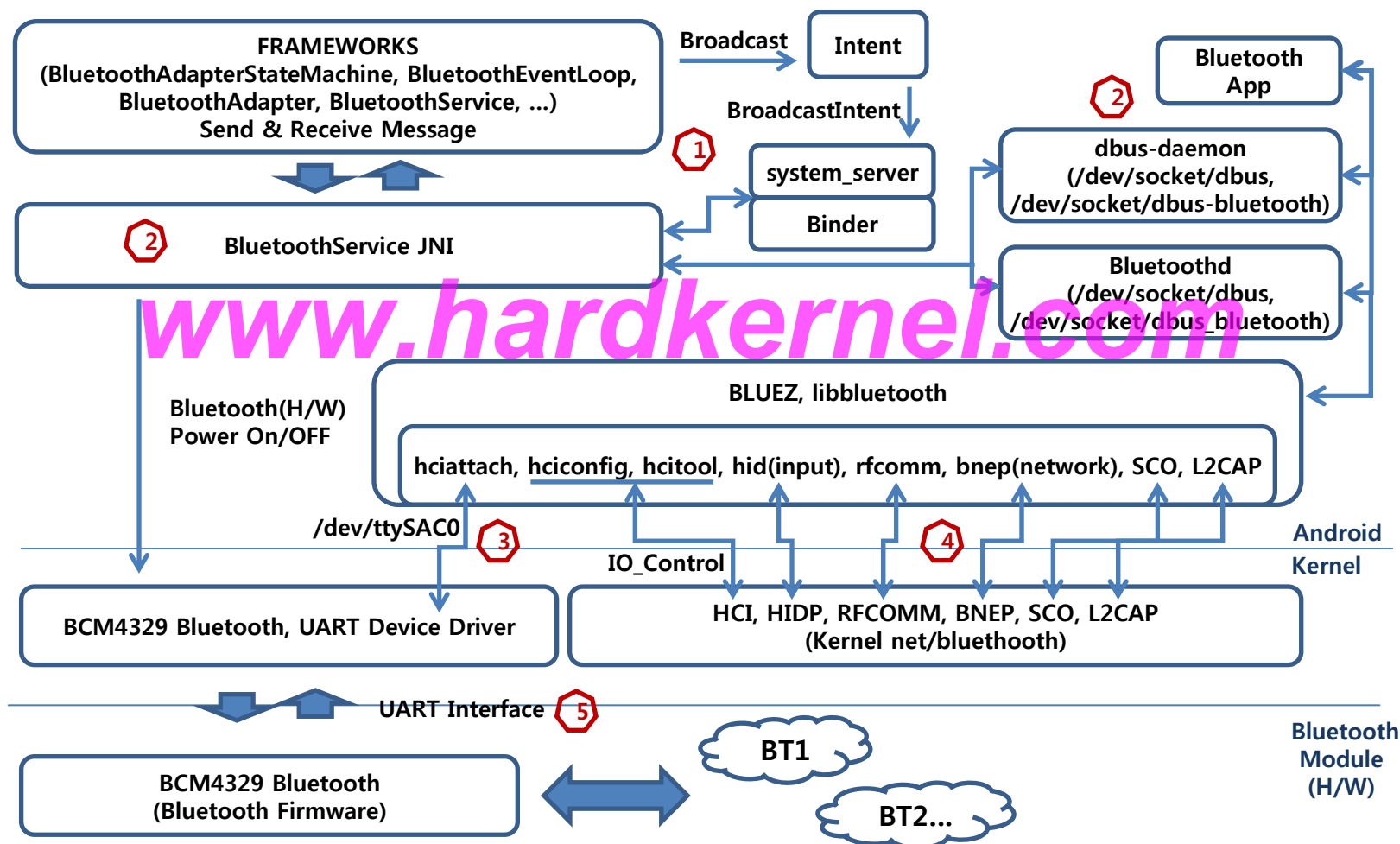
한편, Host(CPU) 와 BCM4329 Bluetooth Module 의 회로도를 보면 UART 로 연결되어 있고, Kernel 의 UART Device Driver 와 BCM4329 Bluetooth Firmware Code 는 UART Protocol 를 통해 서로 소통하게 된다.

Bluetooth Module
(H/W)

Bluetooth Connection Process(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

Bluetooth Connection Block Diagram



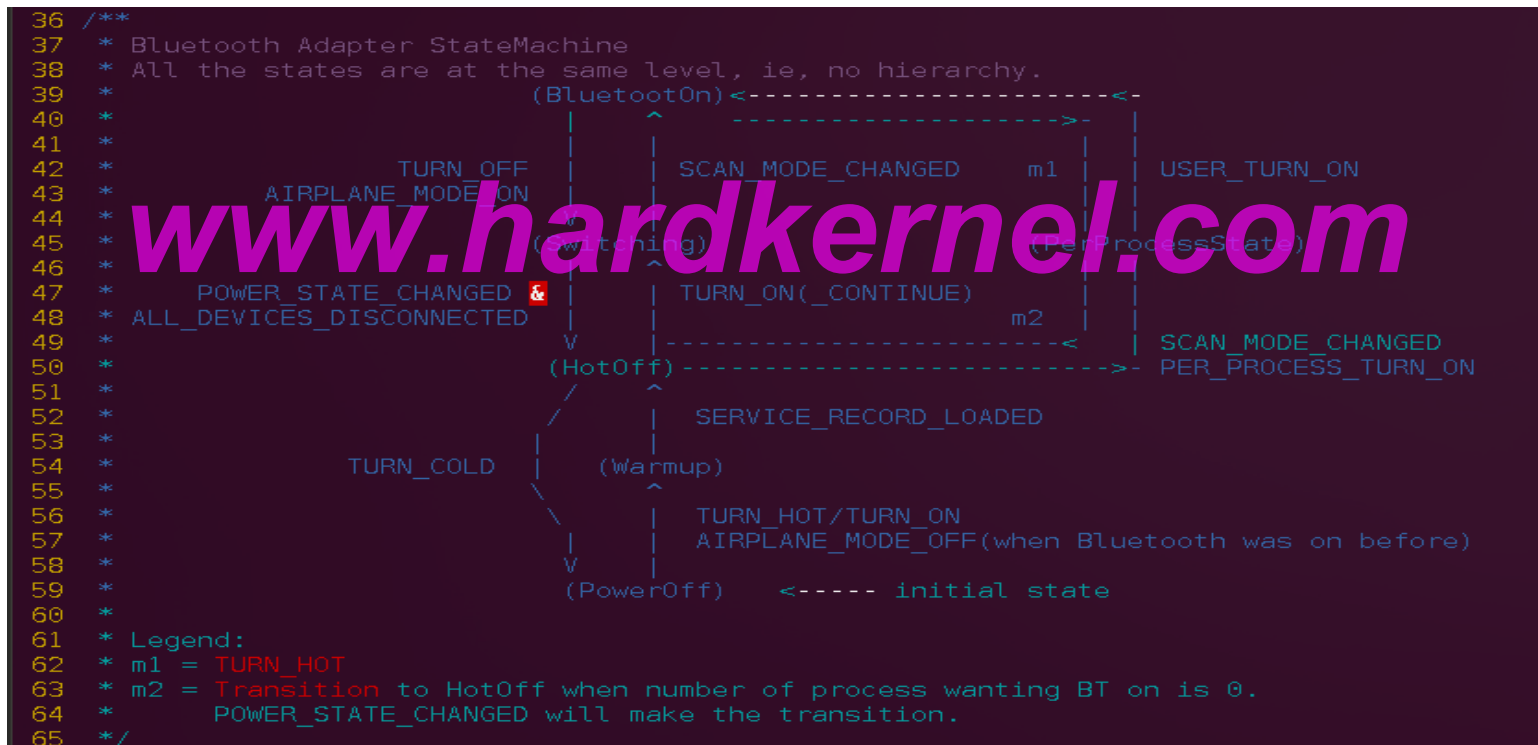
Bluetooth Connection Process(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Communication between Android and Kernel

1. 안드로이드 FRAMEWORKS 에서는 MSG에 따라 Bluetooth 상태를 처리한다.

아래 그림1은 \$ANDROID/frameworks/base/core/java/android/server/BluetoothAdapterStateMachine.java 에서 발췌한 내용이다.



[그림1] Bluetooth Adapter StateMachine

Bluetooth Connection Process(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Communication between Android and Kernel

한 예로 Android Settings 에서 사용자가 Bluetooth 를 On 으로 하면 BluetoothAdapterStateMachine 에 SCAN_MODE_CHANGED라는 MSG 를 받고, Intent 를 생성하여 Intent 와 Permission (여기서는 BLUETOOTH_PERM)를 인자로 Broadcast 하게 된다.

그럼, Context API(Common implementation of Context API) 와 ActivityManager 는 현재 실행되고 있는 Process 중 Intent 를 처리할 수 있는 권한을 가진 Process PID, UID 그리고, origId 와 함께 해당 Process 에게 넘겨지고, 결국은 Daemon 으로 실행되고 있는 system_server 가 binder(IPC:Inter Process Communication) 를 통해 다른 프로세스와 소통하게 된다.

\$ANDROID/frameworks/base/core/java/android/bluetooth/*

\$ANDROID/system/core/includeprivate/android_filesystem_config.h : 권한

Bluetooth Connection Process(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Communication between Android and Kernel

2. 안드로이드 Native Code(JNI) 에서는 주로 IPC(Socket, Dbus) 에 관한 처리(Connection, bind...)가 이루어 지고, Type(RF_COMM, SCO, L2CAP : Protocol) 에 따라 Socket 를 생성, Bind 한다.

한편, JNI 에서는 DBUS 를 이용하여 DBUS 를 이용한 Bluetooth App 과 통신을 지원한다.

\$ANDROID/frameworks/base/core/jni/android_\$BLUETOOTH

\$ANDROID/system/bluetooth

\$ANDROID/external/dbus

www.hardkernel.com

3. hciattach 는 안드로이드 부팅시 Daemon 형식으로 실행(init.rc) 되어, Bluetooth firmware file 를 메모리에 Load 하고, /dev/ttySAC0 Device node 를 통해 커널과 통신하고, 커널은 다시 Bluetooth Module(H/W) 과 통신하게 된다.

\$ANDROID/external/bluetooth

\$KERNEL/drivers/bluetooth, \$KERNEL/net/bluetooth, \$KERNEL/drivers/tty/serial/samsung.c

4. hciconfig, hcitool 는 Host communication Interface tool 로 HOST(CPU) 와 소통하도록 만든 tool 이고, HIDP, RFCOMM, BNEP, SCO, L2CAP 는 Bluetooth Device 간에 소통할 수 있게 만든 Protocol 이다. Bluez 는 IO Control 를 이용하여 Kernel 과 소통한다.

\$ANDROID/external/bluetooth

Bluetooth Connection Process(5)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

➤ Communication between Android and Kernel

5. 회로도를 보면 HOST(CPU) 와 BCM4329 Bluetooth Module 은 UART 로 연결되어 있다.

BCM4329 Bluetooth module 의

UART_TX는 HOST(CPU) 와 UART_0_RXD,

UART_RX는 HOST(CPU) 와 UART_0_TXD 에 연결되어 있다.

한편, init.odroida4.rc 에서 hciattach 실행 부분을 보면

```
service hciattach /system/bin/hciattach -n -f /system/etc/firmware/bcmdhd.hcd /dev/ttySAC0 bcm4329 921600 flow
```

flow를 on 하는 것을 볼 수 있다. hciattach 는 flow 를 옵션을 주지 않아도 default 가 on 이다.

위 옵션은 Kernel UART driver 에도 영향을 미쳐 termios.c_cflag 를 설정하게 된다.

HOST와 Bluetooth Module 간 UART 통신을 할 때, CTS, RTS 로 flow control 한다는 의미이다.

그러므로, H/W 설계를 할 때 UART 의 CTS, RTS 가 연결이 되어 있어야한다.

연결이 안되어 있으면 둘 사이의 의사소통이 안되어 Bluetooth 가 동작을 하지 않는다.

Lab/Exam(1)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Bluetooth control API
- The first thing to do is create the attribute of the BluetoothAdapter.

```
private BluetoothAdapter mBluetoothAdapter;
```
- Then initialize the attribute:

```
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```
- The first thing you should know is how to check the status of WiFi and how to change:

```
boolean state = mBluetoothAdapter.isEnabled(),  
mBluetoothAdapter.enable(),  
mBluetoothAdapter.disable();
```
- To request that Bluetooth be enabled, call startActivityForResult() with the ACTION_REQUEST_ENABLE action Intent. This will issue a request to enable Bluetooth through the system settings (without stopping your application).

```
if (!mBluetoothAdapter.isEnabled()) {  
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);  
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);  
}
```

Lab/Exam(2)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Bluetooth control API
- Discovering devices

// Create a BroadcastReceiver for ACTION_FOUND

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // Add the name and address to an array adapter to show in a ListView
            mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }
};
```

// Register the BroadcastReceiver

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter); // Don't forget to unregister during onDestroy
```

Lab/Exam(3)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

- Bluetooth control API
- Enabling discoverability
- Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, TIME);
startActivity(discoverableIntent);

➤ More info: **www.hardkernel.com**
<http://developer.android.com/reference/android/bluetooth/BluetoothAdapter.html>
<http://developer.android.com/guide/topics/wireless/bluetooth.html>

Lab/Exam(4)

www.hardkernel.com 이 자료는 ㈜하드커널에 모든 권리가 있습니다. 어떠한 상업적인 사용도 허용되지 않습니다. Rev1.0

