Using the ADC ports on 30pin header

There are two ADC inputs on the 30pin header provides 12-bit analog-to-digital converter.

CON10 - 2×15 pins

Pin Number	Net Name	GPIO & Export No
3	ADC_0.AIN0	XADC0AIN_0
23	ADC_0.AIN3	XADC0AIN_3

Using sysfs

Input voltage range is 0~1.8Volt. Otherwise your ODROID will be damaged permanently.

Kernel 3.10

Read raw data.(ADC Channel 0)

```
cat /sys/devices/12d10000.adc/iio:device0/in_voltage0_raw
```

Read raw data.(ADC Channel 3)

```
cat /sys/devices/12d10000.adc/iio:device0/in_voltage3_raw
```

Kernel 4.9 / 4.14

Read raw data.(ADC Channel 0)

```
cat /sys/bus/iio/devices/iio:device0/in_voltage0_raw
```

Read raw data.(ADC Channel 3)

```
cat /sys/bus/iio/devices/iio:device0/in_voltage3_raw
```

Using mmap

Example C source code to access the ADC.

mmap_adc.c

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <stdint.h>

/* EXYNOS ADC_V1 registers definitions */
#define ADC_V1_DATAX      0x0C

/* Future ADC_V2 registers definitions */
#define ADC_V2_CON1       0x00
#define ADC_V2_CON2       0x04
#define ADC_V2_STAT       0x08

/* Bit definitions for ADC_V2 */
#define ADC_V2_CON1_SOFT_RESET    (1u << 2)

#define ADC_V2_CON2_OSEL  (1u << 10)
#define ADC_V2_CON2_ESEL  (1u << 9)
#define ADC_V2_CON2_HIGHF (1u << 8)
#define ADC_V2_CON2_C_TIME(x) (((x) & 7) << 4)
#define ADC_V2_CON2_ACH_SEL(x)    (((x) & 0xF) << 0)
#define ADC_V2_CON2_ACH_MASK  0xF

/* Bit definitions common for ADC_V1 and ADC_V2 */
#define ADC_CON_EN_START        (1u << 0)
#define ADC_DATX_MASK           0xFFF

static volatile uint32_t *adc;
static unsigned char channel;
static uint32_t con1, con2;
static int val0;
static int val3;

void exynos_read_raw(unsigned char channel);

int main(int argc, char **argv)
{
    int fd;

    if ((fd = open("/dev/mem", O_RDWR | O_SYNC)) < ) {
        printf("Unable to open /dev/mem\n");
        return -1;
    }

    adc = mmap(, getpagesize(), PROT_READ | PROT_WRITE,
                    MAP_SHARED, fd, 0x12D10000);
    if (adc < ) {
        printf("mmap failed.\n");
        return -1;
    }
```

```
    // exynos_adc_hw_init
    con1 = ADC_V2_CON1_SOFT_RESET;
    *(adc + (ADC_V2_CON1 >> 2)) |= con1;
    con2 = ADC_V2_CON2_OSEL | ADC_V2_CON2_ESEL |
        ADC_V2_CON2_HIGHF | ADC_V2_CON2_C_TIME();
    *(adc + (ADC_V2_CON2 >> 2)) |= con2;

    while (1)
    {
        // End of A/D conversion
        if (*(adc + (ADC_V2_STAT >> 2)) & 4) {
            /* Read value */
            if (channel == 0x0) {
                val0 = *(adc + (ADC_V1_DATAX >> 2)) & ADC_DATX_MASK;
                printf("ADC Channel 0 : %d\n", val0);
            } else if (channel == 0x3) {
                val3 = *(adc + (ADC_V1_DATAX >> 2)) & ADC_DATX_MASK;
                printf("ADC Channel 3 : %d\n", val3);
            }

            if (channel == 0x0) {
                /* start channel 0 conversion */
                exynos_read_raw(3);
            }
             else if (channel == 0x3) {
                // start channel 3 conversion
                exynos_read_raw();
            }
        }
        channel = con2 & ADC_V2_CON2_ACH_MASK;
        sleep(1);
    }

    return ;
}

void exynos_read_raw(unsigned char channel)
{
    con2 = *(adc + (ADC_V2_CON2 >> 2));
    con2 &= ~ADC_V2_CON2_ACH_MASK;
    *(adc + (ADC_V2_CON2 >> 2)) &= con2;
    con2 |= ADC_V2_CON2_ACH_SEL(channel);
    *(adc + (ADC_V2_CON2 >> 2)) |= con2;

    con1 = *(adc + (ADC_V2_CON1 >> 2));
    *(adc +(ADC_V2_CON1 >> 2)) = con1 | ADC_CON_EN_START;
}
```

Register Map Summary

- Base Address : 0x12D1_0000

Name	Offset	Description	Reset Value
ADC_CON1	0x0000	ADC Control register	0x0000_0002
ADC_CON2	0x0004	ADC Configuration register	0x0000_0720
ADC_STATUS	0x0008	ADC Status register	0x0000_0000
ADC_DAT	0x000C	ADC Data register	0x0000_0000
ADC_INT_EN	0x0010	ADC Interrupt Enable register	0x0000_0000
ADC_INT_STATUS	0x0014	ADC Interrupt Status register	0x0000_0000
ADC_VERSION	0x0020	ADC Version information register	0x8000_0008

ADC_CON1

- Base Address : 0x12D1_0000
- Address = Base Address + 0x0000, Reset Value = 0x0000_0002

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	R	Reserved(read as zero, do not modify)	0x0
SOFT_RESET	[2:1]	RW	Software Reset	0x1
			0x2 = Reset	
			Other = Non-reset	
STC_EN	[0]	RW	Enables ADC start conversion:	0x0
			0x0 = Disables	
			0x1 = Enables	

ADC_CON2

- Base Address: 0x12D1_0000
- Address = Base Address + 0x0004, Reset Value = 0x0000_0720

Name	Bit	Type	Description	Reset Value
RSVD	[31:11	R	Reserved(read as zero, do not modify)	0x0
OSEL	[10]	RW	Selection of the output format	0x1
			0x0 = 2's complement	
			0x1 = Offset binary(recommended)	
ESEL	[9]	RW	Selection of the ADC output timing	0x0
			0x0 = The first output data is evaluated after 40 ADC Clock	
			0x1 = The first output data is evaluated after 20 ADC Clock (recommended).	
HIGHF	[8]	RW	Selection of the ADC conversion rate	0x0
			0x0 = 30 KSPS conversion rate	
			0x1 = 600 KSPS conversion rate (recommended)	
RSVD	[7]	R	Reserved (read as zero, do not modify)	0x0

Name	Bit	Type	Description	Reset Value
C_TIME	[6:4]	RW	Selection of the ADC conversion mode	0x2
			0x0 = 1 times conversion to get the data	
			0x1 = 2 times conversion	
			0x2 = 4 times conversion	
			0x3 = 8 times conversion	
			0x4 = 16 times conversion	
			0x5 = 32 times conversion	
			0x6 = 64 times conversion	
			NOTE: ADC_DAT register is updated on an average with a sum after 1, 2, 4, 8, 16, 32 or 64 times conversion.	
ACH_SEL	[3:0]	RW	Analog input channel selection	0x0
			0x0 = Channel 0	
			0x1 = Channel 1	
			0x2 = Channel 2	
			0x3 = Channel 3	
			0x4 = Channel 4	
			0x5 = Channel 5	
			0x6 = Channel 6	
			0x7 = Channel 7	
			0x8 = Channel 8	
			0x9 = Channel 9	
			0xA to 0xF = Reserved	

ADC_STATUS

- Base Address: 0x12D1_0000
- Address = Base Address + 0x00048, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:3]	R	Reserved (read as zero, do not modify)	0x0
FLAG	[2]	R	End of conversion flag	-
			0x0 = A/D conversion in process	
			0x1 = End of A/D conversion	
RSVD	[1:0]	R	Reserved (read as zero, do not modify)	0x0

ADC_DAT

- Base Address: 0x12D1_0000
- Address = Base Address + 0x000C, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:12]	R	Reserved (read as zero, do not modify)	0x0
ADCDAT	[11:0]	R	ADC conversion data value (0x000 to 0xFFF)	-

ADC_INT_EN

- Base Address: 0x12D1_0000

- Address = Base Address + 0x0010, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	R	Reserved (read as zero, do not modify)	0x0
INT_EN	[0]	RW	0x0 = Disables interrupt	0x0
			0x1 = Enables interrupt	

ADC_INT_STATUS

- Base Address: 0x12D1_0000
- Address = Base Address + 0x0014, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31:1]	R	Reserved (read as zero, do not modify)	0x0
INT_STATUS	[0]	RW	When Read	0x0
			0x0 = Interrupt is not generated	
			0x1 = Interrupt is generated	
			When Write	
			0x0 = No action	
			0x1 = This bit is cleared	

ADC_VERSION

- Base Address: 0x12D1_0000
- Address = Base Address + 0x0020, Reset Value = 0x8000_0008

Name	Bit	Type	Description	Reset Value
ADC_VERSION_INFO	[31:0]	R	ADC Version Information	0x8000_0008