EECS 3215 – Lab I: Analogue to Digital Conversion on the LPC 802

Dr. James Andrew Smith, PEng

Overview: If you think about the "real" world you will realize the signals you might measure there are analog signals. For example, when listening to music on your smartphone, you are hearing analog sounds in the earbuds even though the music is stored in the device in a digital format. So, in order to get the music (signals) stored in the device in the first place, the original sound must be converted to a digital representation by the analog-to-digital converter.[1]

Learning Objectives: In this exercise, you will see that there are several registers with a variety of bits to be initialized to use the A/D. Although there are a variety of operating modes, you will learn one basic mode to get you started. Other modules will help you learn about other details such as A/D resolution and sampling criteria.

Success Criteria: You will be able to demonstrate that you can start the analog-to-digital conversion process, wait for it to finish, and then read the digital result.

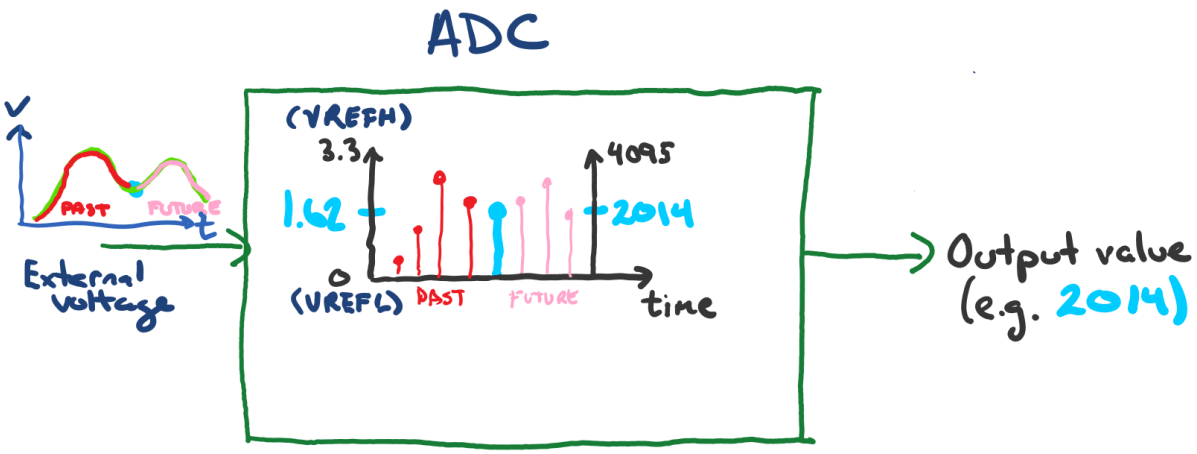


Figure 1 The Analogue to Digital Converter (ADC) measures analogue voltages and converts them to integers in a register.

The ADC is a surprisingly complicated device, on this microcontroller and on most other ones, too. (Even the Arduino's ADC is complicated – it's just that the complexities are hidden from you.) To get started on the LPC802, we will only use a small number of its features and settings. Build on this to do exciting things!

Prerequisites: To successfully complete this module and program the A/D you must have some hardware that can supply an analog voltage to one of the A/D input channels. You must know how to access registers and control bits in C and be able to write and compile the software needed to initialize and start the A/D and to retrieve the converted value. You will have had to attend/viewed the class and/or read the class notes to be prepared for this lab.

[1] This lab is based on the Freescale University Program Laboratory Short Course LABS12CINTRO26, Rev 2 by Fred Cady, Natasha Kholgade and Ken Hsu (RIT)

Marking Guide:

Item				
				Notes:
Explore Demo 1	**Grade:**	**Out of 5**	**0: no attempt; 2.5: partially complete / successful 5.0: complete**	
Explore Demo 2	**Grade:**	**Out of 5**	**0: no attempt; 2.5: partially complete / successful 5.0: complete**	

The LPC 802 ADC Information

To make the ADC work on the LPC802 you need to do the following:

1. Power the ADC with PDRUNCFG
2. SYSAHBCLKCTRL to enable clock register interfaces
3. Use ADCASYNCCLKSEL and ADCASYNCDIV to control ADC clock
4. Do you deal with one of the four IRQs for the ADC?
5. Enable a particular ADC channel
6. Read the ADC…

Setup the C file.

```
/**
 * @file    ch8_lpc802_adc_pot_v1.c
 * @brief   Application entry point.
 */
#include "LPC802.h"
#include <stdint.h>
/* TODO: insert other definitions and declarations here. */
void init_ADC(void);
```

Set up the ADC (part 1 of 3)
Power-up the ADC

```
// The Potentiometer on the LPC board is connected to PIO0_15 (ADC_8)
// Set PINENABLE0 register bit 18 to 0 to make PIO0_15 to ADC_8 (Table 97 in User Manual)
// Bits 10 17 and 19 to 21 to 1 to disable the other channels (they are 1 out of reset)
void init_ADC(void)
{
        // ---------------------------------------------------
        // Step 1. Power the ADC with PDRUNCFG
        // Power Config, Bit 4: 0 is powered on, 1 is powered down.
        SYSCON->PDRUNCFG &= ~(SYSCON_PDRUNCFG_ADC_PD_MASK);
```

Set up the ADC (part 2 of 3)
Enable the ADC & SWM via the clock & reset

```
// ---------------------------------------------------
// Step 2. SYSAHBCLKCTRL to enable clock register interfaces:
// 1. ADC register interface ON
// 2. Switch Matrix register interface ON.
// See Table 63 in the User Manual
// Here, turn on ADC and SWM
SYSCON->SYSAHBCLKCTRL0 |= (    SYSCON_SYSAHBCLKCTRL0_ADC_MASK |
                               SYSCON_SYSAHBCLKCTRL0_SWM_MASK);
// Reset the ADC module.
// Table 64 in User Manual.
// Bit 24 in Peripheral Reset Control Register 0: go 0, then 1 to reset and clear.
SYSCON->PRESETCTRL0 &= ~(SYSCON_PRESETCTRL0_ADC_RST_N_MASK);// Assert reset (0)
SYSCON->PRESETCTRL0 |= (SYSCON_PRESETCTRL0_ADC_RST_N_MASK);// Remove reset(1)
```

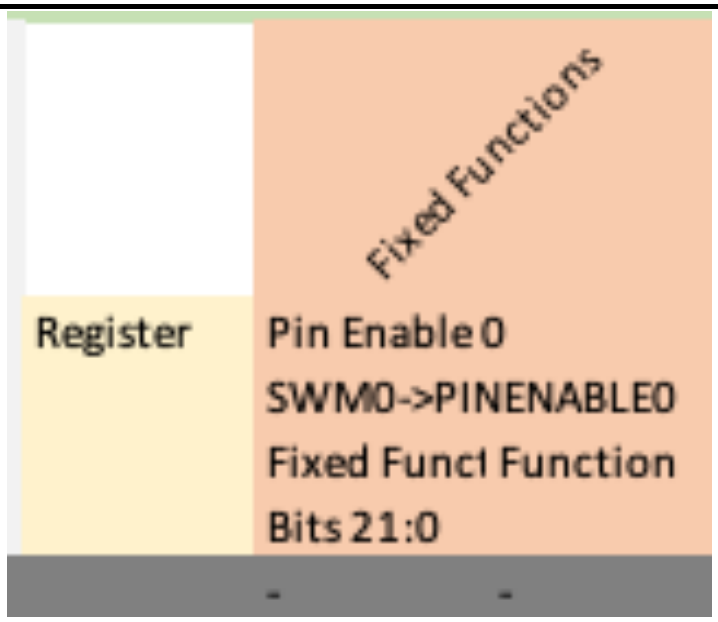
The Switch Matrix Custom Table. : The potentiometer is connected to ADC_8. Locate the "Pin Enable" setting to make that happen.

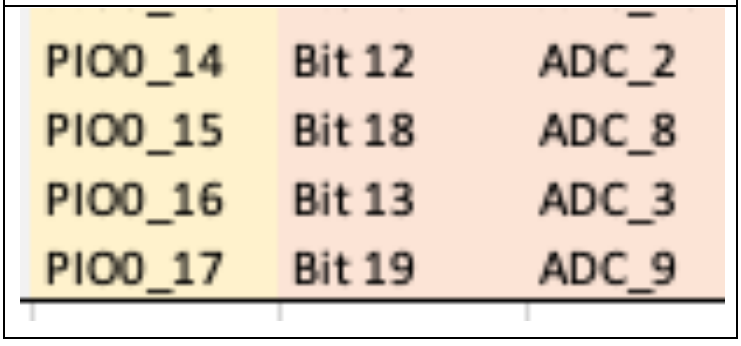
NXP LPC802 Switch Matrix Mappings

Name	Register	Fixed Functions	Extra Functions	Package Pin	Board Header
		Pin Enable 0	-	-	-
		SWM0->PINENABLE0			
		Fixed Funct Function	-	M001JDH r	-
		Bits 21:0	-	TSSOP20	-
		-	-	-	-
GPIO 0	PIO0_0	Bit 0	ACMP_I1	TDO	Pin 19
GPIO 1	PIO0_1	Bit 1	ACMP_I2	ACMP_I2 /	Pin 12
GPIO 2	PIO0_2	Bit 5	SWDIO	TMS	Pin 8
GPIO 3	PIO0_3	Bit 4	SWCLK	TCK	Pin 7
GPIO 4	PIO0_4	Bit 21	ADC_11	TRST*	Pin 6
GPIO 5	PIO0_5	Bit 6	RESETN	-	Pin 5
GPIO 6	PIO0_6	n/a	n/a	-	n/a
GPIO 7	PIO0_7	Bit 11	ADC_1	ACMPvref	Pin 17
GPIO 8	PIO0_8	Bit 15	ADC_5	-	Pin 14
GPIO 9	PIO0_9	Bit 14	ADC_4	-	Pin 13
GPIO 10	PIO0_10	Bit 17	ADC_7	-	Pin 10
GPIO 11	PIO0_11	Bit 16	ADC_6	WKTCLKIN	Pin 9
GPIO 12	PIO0_12	n/a	n/a	-	Pin 4
GPIO 13	PIO0_13	Bit 20	ADC_10	-	Pin 3
GPIO 14	PIO0_14	Bit 12	ADC_2	ACMP_I3	Pin 20
GPIO 15	PIO0_15	Bit 18	ADC_8	-	Pin 11
GPIO 16	PIO0_16	Bit 13	ADC_3	ACMP_I4	Pin 1
GPIO 17	PIO0_17	Bit 19	ADC_9	-	Pin 2

Name	Register	PinAssign 0 SWM0->PINASSIGN.PINASSIGN0 USART 0 Bits 31:24	Bits 23:16	Bits 15:8	Bits 7:0	PinAssign 1 SWM0->PINASSIGN.PINASSIGN1 USART 1 Bits 31:24	Bits 23:16	Bits 15:8	USART 0 Bits 7:0	PinAssign 2 SWM0->PINASSIGN.PINASSIGN2 SPI Bits 31:24	Bits 23:16	Bits 15:8	Bits 7:0	PinAssign 3 SWM->PINASSIGN.PINASSIGN3 CTIMER Capture Bits 31:24	Bits 23:16	Bits 15:8	SPI Bits 7:0	PinAssign 4 SWM0->PINASSIGN.PINASSIGN4 CTIMER Match Bits 31:24	Bits 23:16	Bits 15:8
		U0 CTS	U0 RTS	U0 RXD	U0 TXD	U1 SCLK	U1 RXD	U1 TXD	U0 SCLK	SSEL	MISO	MOSI	SCK	CAP2	CAP1	CAP0	SSEL 1	MAT3	MAT2	MAT1	MAT0
		SWM_PINASSIGN0_U0_CTS_I_MASK	SWM_PINASSIGN0_U0_RTS_O_MASK	SWM_PINASSIGN0_U0_RXD_I_MASK	SWM_PINASSIGN0_U0_TXD_O_MASK	SWM_PINASSIGN1_U1_SCLK_IO_MASK	SWM_PINASSIGN1_U1_RXD_I_MASK	SWM_PINASSIGN1_U1_TXD_O_MASK	SWM_PINASSIGN1_U0_SCLK_IO_MAS	SWM_PINASSIGN2_SPI0_SSEL0_IO_MASK	SWM_PINASSIGN2_SPI0_MISO_IO_MASK	SWM_PINASSIGN2_SPI0_MOSI_IO_MASK	SWM_PINASSIGN2_SPI0_SCK_IO_MASK	SWM_PINASSIGN3_T0_CAP2_MASK	SWM_PINASSIGN3_T0_CAP1_MASK	SWM_PINASSIGN3_T0_CAP0_MASK	SWM_PINASSIGN3_SPI0_SSEL1_IO_MASK	SWM_PINASSIGN4_T0_MAT3_MASK	SWM_PINASSIGN4_T0_MAT2_MASK	SWM_PINASSIGN4_T0_MAT1_MASK	SWM_PINASSIGN4_T0_MAT0_MASK
GPIO 0	PIO0_0	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 1	PIO0_1	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 2	PIO0_2	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 3	PIO0_3	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 4	PIO0_4	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 5	PIO0_5	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 6	PIO0_6	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no	no
GPIO 7	PIO0_7	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 8	PIO0_8	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 9	PIO0_9	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 10	PIO0_10	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 11	PIO0_11	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 12	PIO0_12	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 13	PIO0_13	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 14	PIO0_14	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 15	PIO0_15	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 16	PIO0_16	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
GPIO 17	PIO0_17	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

References: NXP Data Sheet Table 87, 97





Set up the ADC (part 3 of 3)
Set ADC frequency & enable connection to pin

```
// ---------------------------------------------------
// Step 3. Use ADCASYNCCLKSEL and ADCASYNCDIV to control ADC clock
// Use the FRO (Free-running Oscillator) as the source for sync'ing the ADC captures.
// The ADC Clock Select is a pair of bits (SYSCON_ADCCLKSEL_SEL_MASK)
// Set to 00 to use the FRO (yes!); 01 to use an external clock (no!); 11 is no clock. (no!)
// FRO runs at 750 kHz.
SYSCON->ADCCLKSEL &= ~(SYSCON_ADCCLKSEL_SEL_MASK); // Use fro_clk as source for ADC async clock
// Divide the FRO clock into the ADC.  If 0 it shuts down the ADC clock?
SYSCON->ADCCLKDIV = 1;// divide by 1 (values: 0 to 255)
// ---------------------------------------------------
// Step 4. do you deal with one of the four IRQs for the ADC?
// i.e. ADC_SEQA_IRQ, ADC_SEQB_IRQ, ADC_THCMP_IRQ, ADC_OVR_IRQ
// For now, no.
// ---------------------------------------------------
// Step 5. Enable a particular ADC channel
// The Potentiometer on the LPC board is connected to PIO0_15
// Set PINENABLE0 register bit 18 to 0 to make PIO0_15 to ADC_8 (Table 97 in User Manual)
// Bits 10 17 and 19 to 21 to 1 to disable the other channels (they are 1 out of reset)
// Make bit 18 a 0 (active low) to turn on this ADC channel
SWM0->PINENABLE0 &= ~(SWM_PINENABLE0_ADC_8_MASK); //
}
```

Main function (Part 1 of 3)
Init ADC and choose Ch. 8

```
int main(void) {
int i = 0;
uint32_t volatile adc_result = 0;
// Enable ADC and, in particular ADC Ch. 8 to the potentiometer.
```

```
init_ADC();
// Step 1. Select ADC channel 8 via CHANNELS bits in SEQA_CTL.
// ADC0->SEQ_CTRL[0] is SEQA
// ADC0->SEQ_CTRL[1] is SEQB.
// The channel bits are the bottom 12 bits: i.e.  bits 11 to 0.
// A 1 in any bit in the bottom 12 bits will turn THAT channel on:
// ADC_0 is bit 0 of ADC0->SEQ_CTRL[0] (SEQA)
// ADC_8 is bit 8 of ADC0->SEQ_CTRL[0] (SEQA)
ADC0->SEQ_CTRL[0] |= (1UL<<8);// turn on ADC channel 8.
```

Main function (Part 2 of 3)
Control sequencing of the ADC

```
// Step 2. Set TRIGPOL to 1 and SEQ_ENA to 1 in SEQA_CTRL register
ADC0->SEQ_CTRL[0] |= (1UL<<ADC_SEQ_CTRL_TRIGPOL_SHIFT); // trig pol set to 1.
ADC0->SEQ_CTRL[0] |= (1UL<<ADC_SEQ_CTRL_SEQ_ENA_SHIFT); // Sequence A turned ON.
// Step 3. set START bit to 1 in SEQA_CTRL register
// This bit can only be set momentarily.  It immediately goes back to 0
// and so always gets read back as 0.
ADC0->SEQ_CTRL[0] |= (1UL<<ADC_SEQ_CTRL_START_SHIFT);  // start bit to 1
```

Main function (Part 3 of 3)
Read Ch. 8 in a loop. Extract value

```
/* Step 4. Enter an infinite loop, continuously sample Ch 8 on the ADC.*/
  while(1) {
  // Step 6. read result bits in DAT8 (data for channel 8) for conversion result.
  ADC0->SEQ_CTRL[0] &= ~(1UL<<ADC_SEQ_CTRL_START_SHIFT);  // start bit to 0
  ADC0->SEQ_CTRL[0] |= (1UL<<ADC_SEQ_CTRL_START_SHIFT);  // start bit to 1
  // Read the captured value on ADC ch 8. Assign it to a variable.
  adc_result = ((ADC0->DAT[8])&(ADC_DAT_RESULT_MASK));// isolate bits 15:4 (data)
  adc_result = (adc_result>>ADC_DAT_RESULT_SHIFT);// shift right;get true numeric value.
  }
```

```
    return 0 ;
}
```

Explore 1 (upload screenshots)

The LPC 802 board has a built-in potentiometer. (As does the LPC804 board). The '802 board's potentiometer has a connection to the ADC Channel 8 (when you configure it properly). Set it up as described above.

Turn the potentiometer completely clockwise. Halt the debugger and visualize the contents of ADC's DAT8 register. What is it?

Now turn the potentiometer completely counter-clockwise. What is the DAT8 register value now?

Take a screenshot of the debugger window in both cases. Submit as the first part of your report to the TA.

Explore 2 (upload photos)

Now, connect the shield from the previous lab. That shield should have a 7-segment display on it. Use the 7-segment display to show a 9 when the potentiometer is completely turned clockwise. Then turn the potentiometer completely counter-clockwise and show a zero.

Take a photo with your phone (or other digital camera) of your board in both cases (CW & 9; CCW & 0). Show your board in front of your computer screen so that we can see an image of the program solution you wrote to display the analogue data as 7-segment numbers.

Students with the LPC804 boards may have a little trouble reaching the potentiometer with the screwdriver. If that is the case, consider powering down your board, removing the shield, turning the pot, reconnecting the shield and re-powering the board. Not convenient, but should work.

Report

Send a **one page PDF to Moodle** with your name and the name of one lab partner (if you worked with another person). Name and student ID should be at the top of the page. Each student needs to submit a separate PDF as we have not set up partners on Moodle.

The report is to be submitted before the end of your scheduled lab time. The time is documented during your submission by the Moodle server.