EC336: Embedded Systems

Interfacing ADC and DAC

Arvind S Kumar (15EC106), Pavan M (15EC137),
Samarth B (15EC143), Sripathi M (15EC149)

31 August 2018

Submitted to:

Prof. Ramesh Kini M, Prof. Arulalan Rajan
Department of Electronics and Communication Engineering

NITK, Surathkal

1 Goal of the Lab

The aim of this lab was to interface the inbuilt ADC and external DAC with the microcontroller chip. The board chosen by our group for this exercise was Texas Instruments' MSP430G2553 launchpad.

2 Components Used

For this lab the components used are :

- MSP430 microcontroller
- LEDs
- 220 Ω resistors
- DAC0808 IC
- Op-amp IC νA741
- Capacitors 1 νF

3 Board Details

Some of the features of the MSP430 are - (Refer Figure 1 and Figure 2 for more information)

- 16-bit RISC architecture
- Von-Neumann architecture
- Upto 16 MHz clock frequency
- Low power consumption and five power saving modes

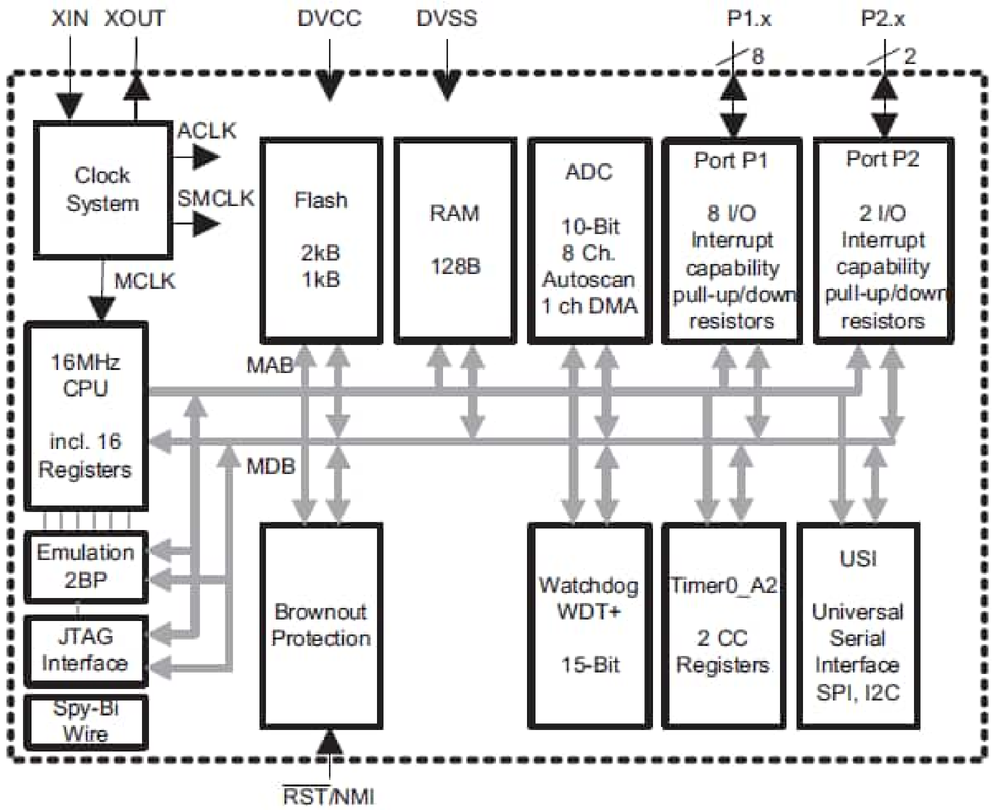
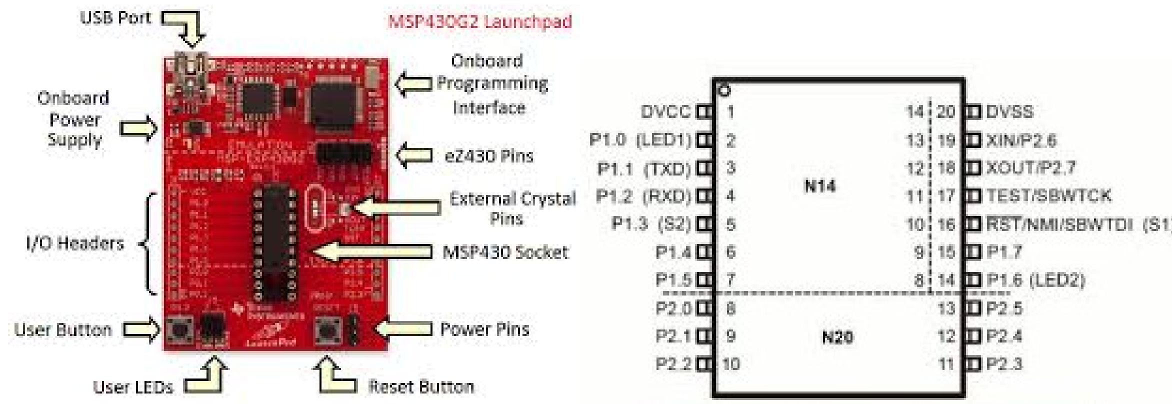


Figure 1: MSP430 Architecture

- 16 general purpose registers
- 10-bit ADC
- Supports SPI, I2C, UART and USB interfaces
- Has 16 GPIO pins divided as two ports (8-bit each)

4 Interfacing Details

MSP430 has an inbuilt Analog to Digital Converter abbreviated as ADC. The resolution of ADC is 10 bits. Analog to Digital converters functionally



(a) Board (b) Pin Diagram

Figure 2: Board details

take in analog input and produce digital bits as outputs. Since the resolution is 10 bits, the number of levels during quantisation would be 1024. Here the task was to configure the LED outputs such that LEDs turned on based on analog input level. For demonstration purpose, the output range was divided into 2 parts and LED was turned on if the input signal exceeded that level and turned off otherwise. The challenge here was to enable a GPIO pin to sense the change in analog value and the controller would process the change and give a digital value. The result was shown using LEDs.

Digital to Analog converters abbreviated as DAC are used to convert back the digital signals after processing to analog so that this output signal analog in nature can be used for actuation. MSP430 unfortunately does not have an inbuilt DAC, hence an external DAC in form of a digital chip was employed. A digital input was fed to 6 out of the 8 pins of the IC (the other two were grounded) in form of high or low and this was converted into analog voltage at the output and this varying voltage was used to control the brightness of LED.

Electrical Characteristics

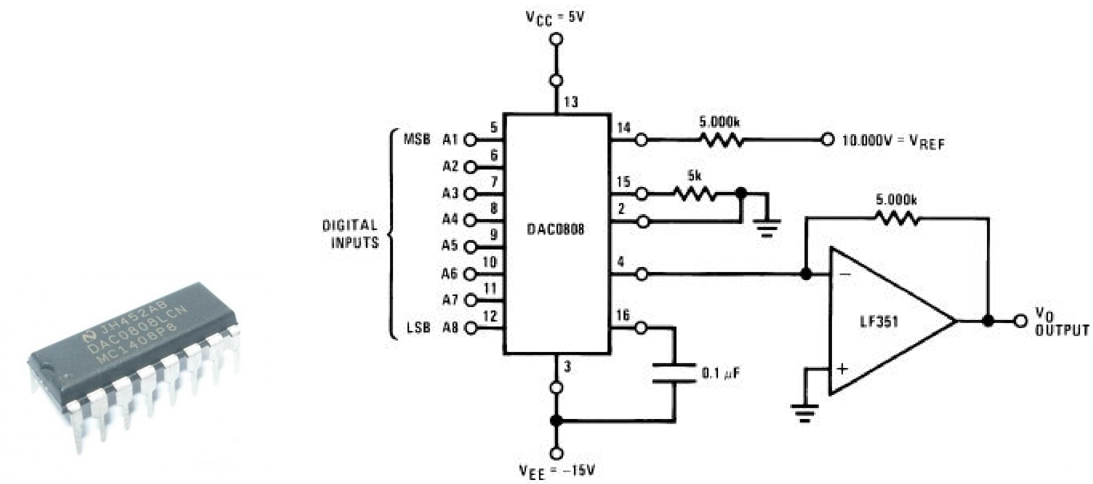
(V_{CC} = 5V, V_{EE} = −15 V_{DC}, V_{REF}/R14 = 2 mA, and all digital inputs at high logic level unless otherwise noted.)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
E_r	Relative Accuracy (Error Relative to Full Scale I_O)	(*Figure 4*)				%
	DAC0808LC (LM1408-8)				**±0.19**	%
	Settling Time to Within ½ LSB (Includes t_{PLH})	T_A=25˚C (Note 7), (*Figure 5*)		150		ns
t_{PLH}, t_{PHL}	Propagation Delay Time	T_A = 25˚C, (*Figure 5*)		30	100	ns
TCI_O	Output Full Scale Current Drift			±20		ppm/˚C
MSB	Digital Input Logic Levels	(*Figure 3*)				
V_{IH}	High Level, Logic "1"		2			V_{DC}
V_{IL}	Low Level, Logic "0"				0.8	V_{DC}
MSB	Digital Input Current	(*Figure 3*)				
	High Level	V_{IH} = 5V		0	0.040	mA
	Low Level	V_{IL} = 0.8V		−0.003	−0.8	mA
I_{15}	Reference Input Bias Current	(*Figure 3*)		−1	−3	µA
	Output Current Range	(*Figure 3*)				
		V_{EE} = −5V	0	2.0	2.1	mA
		V_{EE} = −15V, T_A = 25˚C	0	2.0	4.2	mA
I_O	Output Current	V_{REF} = 2.000V, R14 = 1000Ω, (*Figure 3*)	1.9	1.99	2.1	mA
	Output Current, All Bits Low	(*Figure 3*)		0	4	µA
	Output Voltage Compliance (Note 3)	E_r ≤ 0.19%, T_A = 25˚C				
	V_{EE}=−5V, I_{REF}=1 mA				−0.55, +0.4	V_{DC}
	V_{EE} Below −10V				−5.0, +0.4	V_{DC}
SRI_{REF}	Reference Current Slew Rate	(*Figure 6*)	4	8		mA/µs
	Output Current Power Supply Sensitivity	−5V ≤ V_{EE} ≤ −16.5V		0.05	2.7	µA/V
	Power Supply Current (All Bits Low)	(*Figure 3*)				
I_{CC}				2.3	22	mA
I_{EE}				−4.3	−13	mA
	Power Supply Voltage Range	T_A = 25˚C, (*Figure 3*)				
V_{CC}			4.5	5.0	5.5	V_{DC}
V_{EE}			−4.5	−15	−16.5	V_{DC}

Figure 3: Specifications in Datasheet

4.1 General layout

A few specifications for the DAC IC DAC0808 from Texas Instruments are listed in the form of a figure. Refer Figure 3 above:



(a) DAC0808 IC (b) Interfacing Circuit

Figure 4: DAC0808 Interfacing Circuit

5 Program

The code for interfacing MSP430 with ADC and DAC are as follows:

Interfacing ADC using MSP430	Pseudo code
#include <msp430.h>	
int main(void){	
WDTCTL = WDTPW | WDTHOLD;	Stop watchdog timer
ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE;	ADC10ON, interrupt enabled
ADC10CTL1 = INCH_1;	input A1
ADC10AE0 |= 0x02;	PA.1 ADC option select
P1DIR |= 0x01;	Set P1.0 to output direction
for (;;)	

Continued on next page

Table 1 – continued from previous page

{	
ADC10CTL0 |= ENC + ADC10SC;	Sampling and conversion start
__bis_SR_register(CPUOFF + GIE);	LPM0, ADC10_ISR will force exit
if (ADC10MEM < 0x1FF)	
P1OUT &= ~0x01;	Clear P1.0 LED off
else	
P1OUT |= 0x01;	Set P1.0 LED on
}	
}	
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)	ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR	
__interrupt void ADC10_ISR(void)	
#elif defined(__GNUC__)	
void __attribute__ ((interrupt(ADC10_VECTOR))) ADC10_ISR (void)	
#else	
#error Compiler not supported!	
#endif	
{	
__bic_SR_register_on_exit(CPUOFF);	Clear CPUOFF bit from 0(SR)
}	

Interfacing DAC using MSP430	Pseudo code
#include <msp430.h>	
int main(void){	
WDTCTL = WDTPW | WDTHOLD;	Stop watchdog timer
P1DIR = 0xFF;	Setting 8 pins to output to give input to 8 pins of DAC
unsigned int dac_value = 0x00;	Initializing DAC INPUT
while(1)	
{	
__delay_cycles(100);	Wait for DAC Ref to settle
P1OUT = dac_value	Setting MSP430 output pins to DAC value
__delay_cycles(10);	Delay Between change in DAC value
if(dac_value == 0xFF)	If DAC value has reached Max
dac_value = 0x00	Reset DAC value
else	else
dac_value = dac_value + 1;;	Increment DAC value to increase the brightness of External LED
}	
}	

6 Knowledge gained

1. Since ADC was inbuilt, we learnt how to interface something already inbuilt in the chip by manipulating suitable registers and making use of dual functionality of the GPIO pins

2. Interfaced DAC IC with the microcontroller and controlled the brightness of the LED based on digital input fed form the controller.

3. We found that if the ISR was removed in case of the ADC only the first value is registered and the remaining values are ignored. This shows the ADC peripheral is interrupt based and not polling based.