Lab 5 MSP432 Timers, Interrupts, Analog-to-Digital Converter and SPI DAC

Introduction

Interrupts

Interrupts are important events that must be recognized and serviced by any program. The microcontroller has hardware and software features to allow these important events. Interrupts are an important asset when writing high performance applications. You may use them in embedded systems to synchronize the microcontroller with slower I/O (such as serial interface), and in some applications with a real- time operating system, interrupts are used to carry out all processing tasks. The ARM Cortex-M Series contains a Nested Vector Interrupt Controller (NVIC). The NVIC in the Cortex-M4 supports up to 240 interrupts with 256 levels of priority each. The NVIC configuration in the MSP432 only supports up to 16 priority levels and uses only 95 interrupts. The MSP432's NVIC information can be found in Chapter 2.2.2 (Page 82) in the TI MSP432 Reference Manual. Serial communication is used when data is sent in sequence, one bit at a time, over a specified channel or bus. It is used in extended communication sessions and computer networks due to its improved signal integrity and increasing transmission speeds. One could continuously poll to see if new bytes are ready to be transferred, or use interrupts whenever a byte is ready. The MSP432 has a serial port (RS-232) available to communicate using hardware interrupts.

Since we are going to be using Digital I/O pins in interrupt mode, there are some new registers to be set.

DIR	SEL0/SEL1	IE	IES	Port Mode
0	00	0	0	Input, rising edge trigger
0	00	0	1	Input, falling edge trigger
0	00	1	0	Input, rising edge trigger interrupt
0	00	1	1	Input, falling edge trigger interrupt

The IE and IES registers will need to be set depending on whether we want a rising or falling edge triggered interrupt. Since an interrupt will only occur on an input pin, the DIR bit is always set to 0. Also the SEL0/SEL1 bits will also always be zero for an I/O pin.

Part 1 - Timers

Many embedded systems require precise timing to generate events at certain times. To generate events at the correct time, one of the features included in microcontrollers are timers. Timers can be used to generate interrupts at specific intervals or even be used as pulse counters if supported by the microcontroller. Most timers are based on a free-running counter that increments every clock cycle. When a timer has been started, it will continue running until it is stopped or reaches a maximum value (which can be programmed). Interrupts can be thrown when the timer has reached this value. Counter registers range from 32-bits down to 16-bits depending on the module. Because of the limited bits and high clock speeds, timers often include pre-scalars to slow down the clock, increasing the length of time you can count at the expense of resolution.

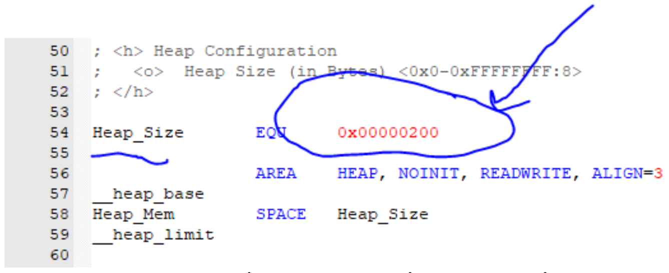
The TI MSP432 contains several timer modules, each with their own set of features. The first timer module is the ARM System Tick Timer. The ARM System Tick Timer is present in many or all ARM cores. It features a 24-bit countdown timer and is often used in Real-time Operating Systems (RTOS) because of its high priority.

The MSP432 features two Timer 32 counter modules that can generate interrupts when zero is reached, ideal for timed events and ADC conversion triggering. Both timers are configurable for 16/32 bit counting modes, clock dividing prescale units, and three running modes, (one-shot, periodic, free running).

The Timer Module (TM) is a very simple timer built upon the one used in the HCS08 (NXP's 8 bit microcontrollers). It can be used for input capture and output compare as well as generating PWM signals. This timer is ideal for motor control and timing associated with external inputs and outputs.

The Real Time Clock (RTC) is a timer used to keep track of seconds. It is powered by an independent power supply (usually a coin cell) and can continue running while the microcontroller is unpowered. The MSP432 also includes two watchdog timers. The first is a timer for internal software such that if it overflows, the MCU will be reset. The second is an external watchdog which is designed to monitor external circuits and send an external reset if it happens to overflow.

NOTE: If you want to use the sprintf function, you will need to increase the HEAP size. To do that you must edit the "startup_msp432p401r_uvision.s" file. You should set the heap size to 0x00000200.



```
; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x00000200

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit

```

Fig. 5.1 Increasing HEAP size

Analog to Digital Converter

The Analog to Digital Converter (ADC) on the MSP432 contains many features such as differential mode, Programmable Gain Amplifier (PGA), Automatic compare function, and more. These features allow for more flexibility and helps offload software processing by implementing the features in hardware. The ADC can be configured in a number of ways to suit the application's requirements. **NOTE**: The MSP432P ADC is 14 bits.

Part 2 - ADC14 with Timer 32 module

For your applications, it's possible to utilize software triggers to start ADC conversions under situations desired by the programmer. The ADC14 module on the MSP432 features an SC flag used to determine whether or not ADC conversions are being executed or not. This flag will be important for starting ACD conversion during predetermined time intervals set by the Timer32 module configuration.

Objectives

The objectives of this exercise are to understand how to enable and use interrupts for various modules, to use a timer to generate and measure time, to successfully digitize an analog signal that is passed through a photo-electronic device, and to read analog input from a line scan camera.

In Part 1 of this exercise, you are to program both Timer32 modules and push buttons to perform two operations. The first switch (SW1) and timer will be used to toggle an LED which will blink on or off every second (½ second on, ½ second off). The second switch (SW2) and timer will measure how long between presses of the switch (SW2). After the switch is released (upon the second press), the time in milliseconds is to be displayed in a terminal.

In Part 2, you will digitize an analog signal that is passed through a photo-electronic device, display binary logic to a terminal through a UART interface, and see the effects of aliasing on the data.

In Part 3, you will combine your understanding of timers, interrupts, and analog to digital conversion to interface a line scan camera with your TI IDE car.

Files

https://kgcoe-git.rit.edu/CMPE_460/CMPE_460_Files

Relevant Technical Reference Manual Chapters

- SYSTICK Timer
- Chapter 12: Digital IO (DIO)
- Chapter 18: Timer32
- Chapter 19: Reference Module
- Chapter 20: Precision ADC
- Chapter 22: universal Serial Communication (UART)
- Chapter 23: universal Serial Communication (SPI)
- MCP4901 DAC datasheet

Materials

- MSP432 Board and USB cable
- TMP36 temperature sensor, CdS cell (both in Lab Kit)
- Jumpers
- Breadboard
- Line Scan Camera
- DAC: MCP4901

Prelab Activities

Warning: Extensive preparation is required for this exercise. Please give yourself plenty of time. Complete the following steps using registers found in the MSP432 Reference Manual: The first 3 items should be done in C code. The last two are to be researched.

- Configure SW1 to be interrupted on the falling edge (Chapter 12.2.7)
- Configure SW2 to be interrupted on falling edge
- Configure Timer32 for accurate time delay (Chapter 18)
- Investigate how to Enable the ADC, which is ADC A6
- Investigate how to Read result from ADC A6

It is highly encouraged to look at the next parts of this exercise and implement the rest of the configurations for the modules we will use

Lab Procedure

Part 1 - Timers and Interrupts

The goal of this section of the exercise is as follows:

- Upon press of SW1, an interrupt will be generated that will start a timer (Timer32-1) that will flash LED1 at a rate of 0.5s on and 0.5s off. A subsequent press of SW1 will stop the timer and stop the LED1 from flashing.

- Upon press of SW2, an interrupt will be generated that will start a timer (Timer32-2) and turn on LED2, that will time how long SW2 is held. When SW2 is pressed again, display the elapsed time between SW2 presses in PUTTY. When turning on LED2, turn on LED2 starting with RED. After each subsequent press, change the color to BLUE, then GREEN, then CYAN, then MAGENTA, then YELLOW, then WHITE. Repeat. When the LED2 is on, the elapsed timer is running.

1. Create a new project in Keil μVision5.xx.0. Make sure to set the HEAP to at least 0x200, and the board frequency to 48MHz (see Appendix A).

2. Import any Led, Button, or uart initialization functions from previous exercises that are useful to the exercise.

3. Add initialization to SW1 and SW2 to support interrupts (falling) in in your button initialization function.

4. Call the LED, switch, and uart0 initialization functions.

5. At the end of the button initialization function set SW1 to cause an interrupt when pressed, and set SW2 to cause an interrupt when pressed (see Page 676 of the reference manual for more detail)

6. Copy (or include) uart.h from Lab 2

   (a) Modify code below to add function declarations as appropriate for your uart (PUTTY) code:

7. Anytime an interrupt is triggered for the peripheral (timer, switch, etc), its IRQ Handler will be called based on its NVIC mapping (done in the initialization of each peripheral). Each IRQ handler will clear the interrupt and determine what to do. The stubbed out ISR functions (IRQ Handlers) for each timer and switch can be found as appropriate. Using the pseudo-code below, set up each ISR.

   (a) For Timer32_1():
      (i) Set timer for a 0.5 Hz frequency
      (ii) When switch1 pressed, flash LED1 at 1 Hz rate (0.5s on/0.5s off)
      (iii) When SW1 pressed again, turn off LED1 (if it is flashing)

   (b) For Timer32_2 ():
      (i) Set timer for a 1 millisecond period (1KHz frequency)
      (ii) Increment state of LED2
      (iii) Increment a millisecond counter

   (c) For PORT1: (toggle blinking LED1 upon press of SW1)
      (i) Clear the interrupt
      (ii) Check if SW1 or SW2 is pressed.
      (iii) For SW1 if pressed, process based on state of LED1 (on or off)
      (iv) For SW2 if pressed, process based on state of LED2 (R,G,B, etc)

8. Set up the Timer32 modules. You will only need to use the CONTROL and LOAD registers in for the modules. Use the search feature (CTRL-F) to find the register names in the MSP432 Reference Manual.

   (a) Determine the frequency you want the counter to increment. You need to take into consideration the counter's size, the MCU's peripheral clock speed, and the division factors that are available.
   (b) The LOAD Register determines at what point during the counting sequence the TIMER32 module will generate an interrupt. Change this value accordingly.
   (c) Configure the Control register. Configure the TIMER32 modules to generate 1s and 1ms interrupts. Make sure to enable interrupts and the timer at the end of the initialization functions.
   (d) Enable interrupts on the TIMER32 modules in the main function (Lab5-Timer.c) and set up an ISR for it.
   (e) Clearing/setting bit 7 of the timers disables/enables them to run.
   (f) The above steps should be added to Timer32_1_Init and Timer32_2_Init in Timer32.c.
   (g) There are Interrupt service routine handler stubs in the Timer32.c file.

9. Disable or enable interrupts by using the DisableInterrupts or EnableInterrupts functions in CortexM.c (supplied). Do not modify these functions.

(Sign-off Requirement) Demonstrate full functionality of both timers by loading software onto the MSP432, connecting it via the serial port, and pressing SW1 and SW2 with a lab instructor or TA. Take a screenshot of the terminal for your report.

Part 2 - Analog to Digital Converter

ADC Channels and Pins: The ADC14 peripheral has multiple channels that are connected to various MCU pins. The MSP432 dedicates six pins (A0:A23) for analog input signals on various pins. (See p.36 of Launchpad Reference manual for more detail). We will be using A6 for this lab exercise, which is on P4.7.

NOTE: The MSP432P input voltage should be limited to no more that 3.8V on the ADC pin. Exceeding this value will damage the pin. Please be careful.

Analog to Digital Converter (ADC) The reference manual provides initialization information starting in **Chapter 22**.

Part 2a - CDS Cell

A CDS Cell is a simple photo-electronic device that provides a variable resistance value to a circuit. This value decreases with increasing light and increases with decreasing light, exhibiting photoconductivity. The component is made of a high resistance semiconductor material that absorbs photons, giving electrons enough energy to jump bands, thereby conducting electricity and allowing more current to flow through a circuit.

TIMER 32 (T32) For this part of the lab the timer 32-1 module was configured to produce an interrupt every ½ second. These interrupts produce software triggers to read in analog signals from the MCU.

1. Create a new project in Keil μVision 5.13.0
2. Download Lab5-ADC.c, ADC.c from gitlab
3. You will need Timer32 from the previous part of the exercise
4. Take the UART code from Exercise 2 and import it to the ADC project
5. In your main function of file Lab5-ADC.c initialize the uart0 and all other necessary devices.
6. In a 'for' loop print out decimal and hexadecimal representations of analog input from the ADC. The MEM0 registers contains the result of the ADC conversion of Channel 6 for this specific case, which will be on P4.7.
7. On the breadboard create the simple circuit shown in Figure 5.2. Both power and ground can be provided from the MSP432 **in this case**.

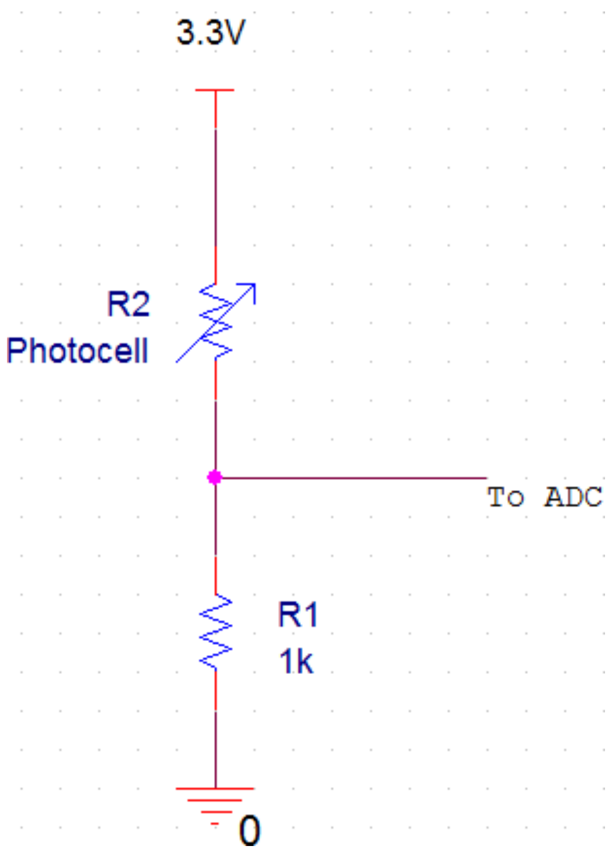


Figure 5.2: Photocell Circuit

8. Connect the middle node of the photocell circuit to the ADC A6 at pin P4.7 (look at the wiring schematic for confirmation of this). The simple voltage divider based on the variance of the CDS cell will provide a variable analog value that will be observable through the UART interface, similar to previous lab exercises. **NOTE: To test your code, connect P4.7 to ground and observe the output at approximately 0. Then connect P4.7 to 3.3V and observe the ADC readings to be approximately 16,383. DO NOT CONNECT IT TO 5V!! You will damage the MSP432.**

9. **(Sign-off Requirement)** Demonstrate your working circuit to your Lab Instructor or a TA

10. ***Repeat the procedure*** with the TMP36 temperature sensor (this device outputs voltage directly, no need to construct a circuit). Display the temperature in both **deg.** ***C and*** **deg.** ***F***. You may create a new project or modify the photocell project to now use the temperature sensor. Use **3.3V** for the power for the TMP36..

11. **(Sign-off Requirement)** Demonstrate your CDS and temperature sensor working with your Lab Instructor or a TA.

Part 3 - Car Camera

Line Scan Camera: The TI Car Kit comes with an independent line scan module that uses a light array and lens that returns an analog 1-D view slice of what it sees. The timing diagram for the camera can be seen in Figure 5.3. The camera has 2 inputs and 1 output:

1. CK (clock) - latches SI and clocks pixels out (low to high) continuous signal (use P5.4)
2. SI (serial input to sensor) begins a scan / exposure, pulse must go low before rising edge of next clock pulse (use P5.5)
3. AO (analog output) - Analog pixel output from the sensor (0-Vdd) or tri-stated (use P4.7 – A6)

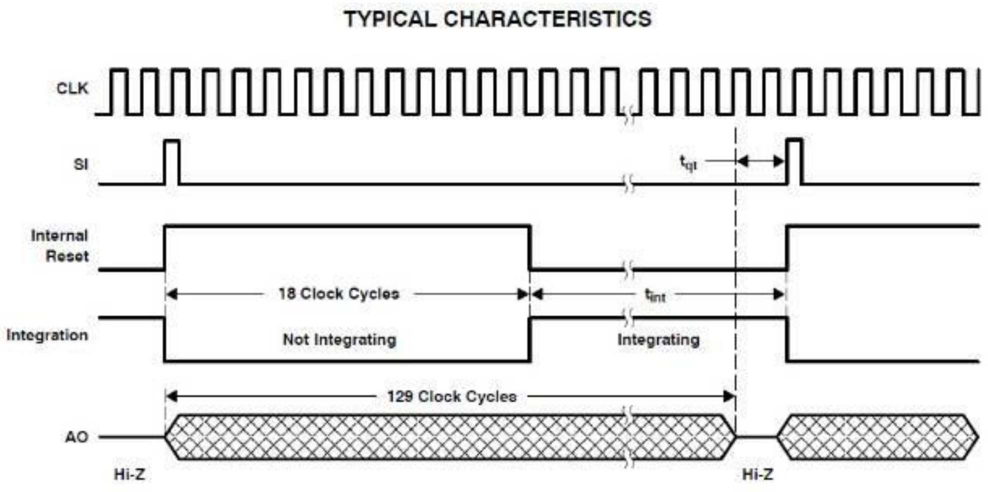
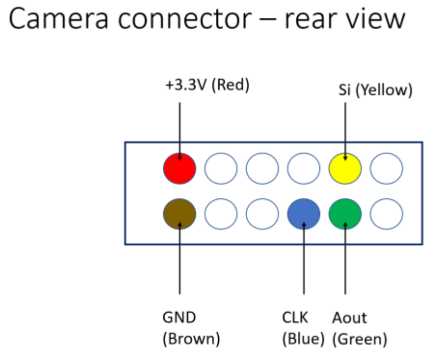


Figure 5.3: Line Scan Camera Timing Diagram

(Reproduced from Line Scan Camera datasheet)

1. Create a new project in Keil μVision 5.13.0
2. Wiring Interface (Rear View of camera):



3. Write code to interface with the Line Scan Camera using the timer modules and the ADC. Use Lab5-Camera.c as a template. We will be using the Parallax TSL 1401 Line Scan Camera. Some notes to consider:
   - Usage documents: Line Scan Camera Use.pdf and Parallax-TSL1401-DB-Manual.pdf in GitLab under General Resources\Datasheets.
   - Do not exceed over 100ms for the integration time otherwise the capacitors will saturate
   - Each set of readings will take 128 cycles, each cycle is one piece of the input
   - Use the SysTick timer to toggle your clock (CLK) signal on a Digital I/O pin connected to P5.4 at a period of 10μs *(100KHz)*
   - SI will pulse the camera as a start signal on a Digital I/O pin connected to P5.5
   - ControlPins.c contains the templates for the SI and CLK pins that control the camera

- Use the Timer32_1 timer module to determine the integration period (time delay) on the SI pin P5.5 (the time between SI pulses is the integration time). The integration time determines how much light the sensor is exposed to.

Verification: To verify that proper inputs are being received, use the provided MATLAB template (plot cameras_serial_blank.m) to read in values from the camera using the COM port on the computer and UART module on the MSP432. (Note: You will need to add start and stop values to your incoming data). Graph these values using the MATLAB plot function. Output the values coming from the camera to an oscilloscope to verify the correctness of the MATLAB script and A2D conversion on the MSP432. Along with the **raw data plot**, plot both a **low pass filter** on the data as well as an **edge detection** plot (binary). Use an oscilloscope to focus your camera.

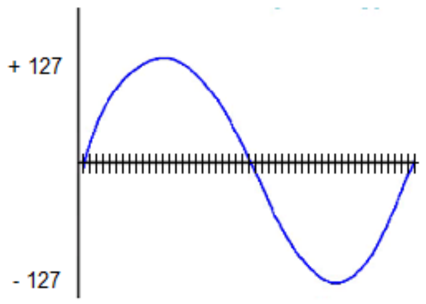
(Sign-off Requirement) Demonstrate your MATLAB and Oscilloscope Output to your Lab Instructor or a TA.

Part 4 – DAC-SPI- Interface

Sine Wave Generation

Generate a data of 100 entries, each entry represents the magnitude of the sine wave

Vout = Vmax Sine (ω t)



Write the required code to interface MCP 4901 DAC to generate 1 Hz sine wave.

Lab Report

Follow the general Lab Report Guidelines on MyCourses. In addition to handing in the printed copy of your report during the next lab period, please add any source code that changed in this exercise, along with your lab report, to the Dropbox on MyCourses.

1. Title Page
2. Short Abstract
3. Design Methodology
   (a) Describe what was done, why it was done, and how it was done. It is expected that all initialization sequences and procedures are explained in depth in this section.
   (b) All applicable information to reproduce experiment
   (c) Describe any problems encountered and the fixes performed.
4. Results
   (a) A terminal screenshot of the UART output for Part 1.

(b) A terminal screenshot of the UART output for Part 2.

(c) A screenshot of the MATLAB output for Part 3.

(d) A screenshot of the output for Part 4 (DAC SPI Interface).

5. Analysis

   (a) What is the smallest amount of time you could measure? Longest amount of time?

6. Questions

   (a) Why do the IRQ flags need to be cleared in the interrupt service routine functions?

   (b) When an interrupt occurs, how do you know which pin caused it?

   (c) What is the startup sequence of SI and CLK to initiate data transfer from the Camera?

7. Conclusion

Exercise 5: MSP432 Timers, Interrupts, and Analog-to-Digital Converter

Student's Name:________________________ Section:________________________

PreLab		Point Value	Points Earned	Comments
PreLab	Prelab C Code	10		

Demo		Point Value	Points Earned	Date
Demo	Functionality with SW1	5		
	Functionality with SW2	5		
	ADC Functionality with CdS	5		
	ADC Functionality with Temp Sensor	5		
	Linescan Camera MATLAB and Oscilloscope	5		
	DAC SPI Interface Oscilloscope	5		

To receive any grading credit students must earn points for both the demonstration and the report.

Exercise 5: MSP432 Timers, Interrupts, and Analog-to-Digital Converter

Report		Point Value	Points Earned	Comments
Abstract		10		
Design Methodology	Discussion	10		
Results and Analysis	Part 1 Screenshot	5		
	Part 2 Screenshot	5		
	Camera Plot	5		
	Timing Explanations	5		
Conclusion		5		
Questions	Question a	5		
	Question b	5		
	Question c	5		
Total for prelab, demo, and report		100		