Lab #5 Interrupts

Objectives:

- to introduce the concept of an interrupt
- to experiment with the integrated timer for generating internal interrupts
- to design and implement an interrupt-driven external interface using an ADC and DAC
- to gain further experience with assembly language programming on a microprocessor (microcontroller)

Introduction and Background:

Interrupts provide a mechanism for initiating a program (the "interrupt service routine" or "ISR") in response to an external or internal request. Interrupt-driven systems may thus be viewed as "event-driven" systems that free the CPU from continual polling to check for device readiness. Common examples of interrupts include implementing a handshake interface to peripherals, generating response to a clock or timer and counting external events.

We will be implementing three interrupt exercises in this Lab session. The first is a simple example of interrupt responding to a toggle switch. The second is a interrupts responding to an internal timer overflow (which generates an interrupt). Finally the third example is used to illustrate a handshake interface to an A/D converter circuit.

Read the background information on the interrupt system for the 8051 microcontroller and familiarize yourself with the SFRs that are required to use it. Study the sample 8051 assembly language code for all parts of this lab as well as the ADC/DAC circuit given below. The data sheets for the ADC and DAC devices used in this lab are available on the lab web site. The circuits will be pre-wired for you as shown in the diagram at the end of this lab.

Preparation:

Read sections 6.1 to 6.2.1 and 6.3 and 6.4 of the 8051 textbook. Read this entire lab and calculate the code execution times as required.

Programming exercises

1. Interrupt response to a toggle switch

Connect one of the toggle switches to the INT1 input on the expansion header of the MCM51A board. Assemble and load the test code "ouch.a51" and run the program. Stop the program and explain the values that were placed on the stack. Try with both an un-debounced and debounced switch and explain the behaviour !

2. Interrupt response to an internal timer overflow

Assemble and load the test code "tick.a51" and run the program. Connect the scope probe to P1.3 and measure the frequency of the clock that is being generated. Can you adjust the code to make exactly 1 second "ticks" ?

Measured clock frequency: ____________________

Calculate the clock frequency that you expect given that 16-bit Timer 0 is being clocked up at a frequency of 11.059 MHz.

Calculated clock frequency: ____________________

3. Interrupt interface to analog-to-digital converter

(a) Assemble, load and run the program "dac.a51" that outputs a ramp function to the DAC converter. Display the waveform on the scope. What is the frequency that is generated with this code ?

Measured Frequency: _______________ Calculated Frequency: _______________

(b) Using the function generator as a signal source apply a 1 KHz sine wave (do not exceed 0 to +5V) to the analog input of the ADC. Assemble, load and run the 8051 program "adc.a51" that continually reads a sample from the analog input and displays the output through the DAC.

(c) Now modify the code in part (b) to implement a simple filter. Collect four successive samples from the ADC, then output their *average* to the DAC. Observe the output of the filtered waveform on the scope. Observe the effects on the output as the input frequency is increased.

Note: Please do NOT modify the pre-wired ADC and DAC circuits. If you are not sure how to connect to these circuits, ask for help. Do NOT guess !!

Answer the following questions:

What is the sampling frequency ? ________________________

What is the A/D conversion time ? ________________________

At what frequency does A/D conversion break down ? ___________

Explain what happens to the filtered output as the input frequency increases.

Explain the waveform observed on the output of the FIR filter.

Code for Part 1 "ouch.a51"

```
$MOD51
;------------------------------------------------------------------------
;  Example code for MCM51A boards       D. Capson   2002
;
;  Demo program for Comp Eng 3DJ4 Lab #5 Part 1  - using interrupts
;------------------------------------------------------------------------

; addresses of some monitor routines that are used

PUTS    equ 003ah   ; send text string to serial port, DPTR shows start, Null ends
CRLF    equ 003ch   ; output CR, LF to serial port

; ... some ASCII characters

CR    equ 0dh
LF    equ 0ah
NULL  equ 0


org 8000h   ; start location
ljmp start

org 8013h   ;  External interrupt (INT1)
ljmp Int1ISR

org 8100h

message: db 'Comp Eng 3DJ4 Demo program', CR, LF,'Displays a message every time an
external interrupt comes in on INT1', CR, LF, NULL

ouch:   db ' *** OUCH !! *** ', NULL

start:  mov 81h, #30h       ;  initialize the stack pointer (SFR 81H = SP)
      lcall CRLF          ;  output a cr and linefeed to the terminal
      mov dptr, #message  ;  show the sign-on message on the terminal
      lcall PUTS

; ---------------  main loop -------------------

      orl 0A8h, #10000100b  ; interrupt enables  (SFR A8H = IE register)

      setb IT1        ; specify edge activated external interrupt for INT1
                      ;   (IT1 = bit 2 of TCON)

wait:   sjmp wait     ; program loops here waiting for interrupts


; --------------- interrupt service routine for INT1  -----------------------
Int1ISR:
      mov dptr, #ouch       ; display "** Ouch !! **"
      lcall PUTS
      reti
;-----------------------------------------------------------------------------

end
```

Code for Part 2 "tick.a51"

```
$MOD51
;--------------------------------------------------------------------------
;  Example code for MCM51A boards       D. Capson   2002
;
;  Demo program for Comp Eng 3DJ4 Lab #5  - using the timer/counter and interrupts
;--------------------------------------------------------------------------

; addresses of monitor routines that are used
PUTS    equ 003ah   ; send text string to serial port, DPTR shows start, Null ends
CRLF    equ 003ch   ; output CR, LF to serial port

; ... some ASCII characters

CR    equ 0dh
LF    equ 0ah
NULL  equ 0


org 8000h   ; start location
ljmp start

org 800bh   ;  Timer 0 overflow interrupt (TF0)
ljmp timer

org 8100h

message: db 'Comp Eng 3DJ4 Demo program - display a clock tick from internal timer
interrupt ', CR, LF, NUll

tick:   db 'Tick! ', NULL

start:  mov 81h, #30h       ;  initialize the stack pointer (SFR 81H = SP)
      lcall CRLF          ;  output a cr and linefeed to the terminal
      mov dptr, #message  ;  show the sign-on message on the terminal
      lcall PUTS

; ---------------  main loop -------------------

      mov R7, #0fh          ; a counter for keeping track of 15 interrupts

      orl TMOD, #01h        ; select timer 0 to be a 16-bit count
                          ; (notice that TMOD is not bit-adressable)

      mov 8Ch, #0fh         ;  intial value of timer to give 1/15 sec
      mov 8Ah, #19h         ;  SFR 8Ch = TH0    SFR 8Ah = TL0
                          ;  Timer counts up at freq = clock freq/12
                          ;  therefore, we get a new clock pulse
                          ;  every 1.085 microsec.  Timer overflow
                          ;  interrupts when timer hits zero.

      orl 0A8h, #10000010b  ; interrupt enables  (SFR A8H = IE register)

      setb TR0            ; start timer 0  ( TR0 = bit 5 of TCON )

wait:   sjmp wait
```

```
; --------------- interrupt service routine for timer 0  -----------------------
timer:

   ;  get timer 0 started again immediately

      mov 8Ch, #0fh        ;  intial value of timer to give 1/15 sec
      mov 8Ah, #19h        ;  SFR 8Ch = TH0    SFR 8Ah = TL0
      setb TR0

      djnz R7, exit     ; check if its the 15th time its been interrupted
      cpl P1.3          ; complement Port 1, bit 3 so we can observe with scope
      mov dptr, #tick   ; if it is, then display "Tick!"
      lcall PUTS        ;  ie. exactly 1 sec has elapsed

      mov R7, #0fh      ; reset our counter for 15 interrupts

exit:   reti

end
```

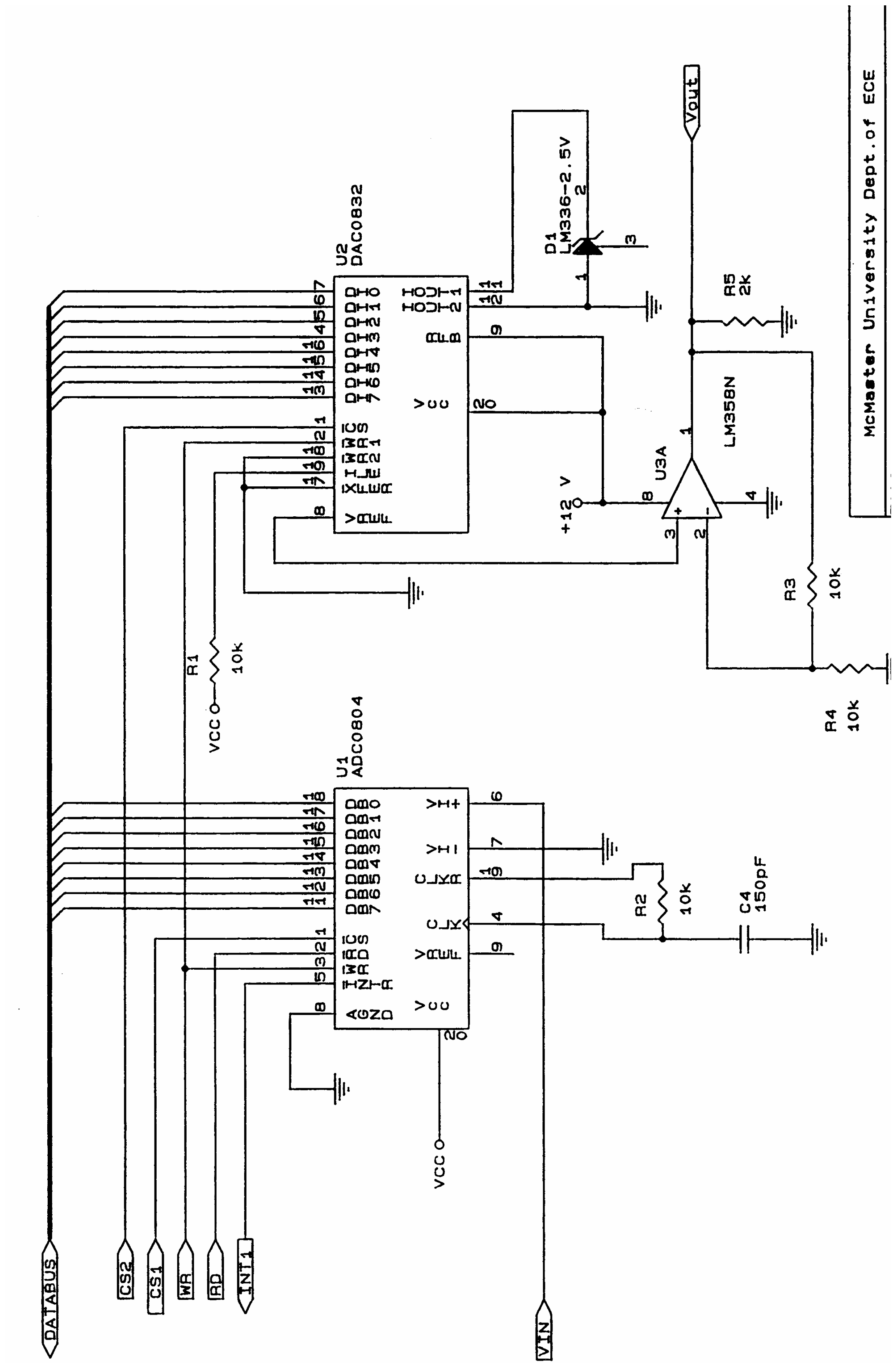
Code for Part 3(a)   "dac.a51"

```
$MOD51
;------------------------------------------------------------------------
;  Example code for MCM51A boards       D. Capson   2002
;
;  Output a ramp function to DAC Interface
;------------------------------------------------------------------------

org 8000h   ; start location
ljmp start


org 8010h

start:

      mov a, #00h

again: inc a
       mov dptr, #1800h        ;  put the sample out to the DAC
       movx @dptr, A           ;  by writing to /cs1
       sjmp again

end
```

Code for Part 3(b) "adc.a51"

```
$MOD51
;------------------------------------------------------------------------
;  Example code for MCM51 boards       D. Capson  2002
;
;  Lab #5  -  collect a sample from ADC and display via the DAC Interface
;------------------------------------------------------------------------

; addresses of monitor routines that are used

PUTS    equ 003ah   ; send text string to serial port, DPTR shows start, Null ends
CRLF    equ 003ch   ; output CR, LF to serial port

; ... some ASCII characters

CR    equ 0dh
LF    equ 0ah
NULL  equ 0


org 8000h   ; start location
ljmp start

org 8013h   ;  INT1  vector
ljmp sample

org 8100h

message: db 'Comp Eng 3DJ4 Test program - ADC/DAC Interface', CR, LF, NULL

start:  mov 81h, #30h       ;  initialize the stack pointer (SFR 81H = SP)
        lcall CRLF          ;  output a cr and linefeed to the terminal
        mov dptr, #message  ;  show the sign-on message on the terminal
        lcall PUTS

; ---------------  main loop -------------------

      orl 0A8h, #10000100b  ; interrupt enables  (SFR A8H = IE register)
;         enable INT1 ^

      orl 88h, #00000100b   ; INT1 control bit   (SFR 88H = TCON register)
;                      ^  INT1 is edge activated



;  Get things started...

      mov dptr, #1000h        ; issue start-of-convert to ADC by issuing
      movx @dptr, A           ; a /cs0 together with the /WR pulse

wait:   sjmp wait             ;  wait for interrupts to indicate a sample is
                              ;  available
; --------------- interrupt service routine for INT1  -----------------------
sample:
      mov dptr, #1000h        ; get the sample from the ADC by
      movx A, @dptr           ; reading with /cs0 ...

      mov dptr, #1800h        ; ... and put the sample out to the DAC
      movx @dptr, A           ;     by writing to /cs1

; now issue another start-of-convert to initiate the next sample

      mov dptr, #1000h        ; /cs1 responds to address 1000h
      movx @dptr, A           ; issue the /WR pulse with /cs0

      reti
;---------------------------------------------------------------------------


end
```

Acknowledgement: The circuit on the following page was designed and drawn by Mr. Ken Frost, Digital Systems Specialist, Dept of Electrical and Computer Engineering


U2
DAC0832
D1
LM336-2.5V
Vout
R5
2k
LM358N
U3A
+12 V
R3
10k
R4
10k
R1
10k
VCC
U1
ADC0804
R2
10k
C4
150pF
DATABUS
CS2
CS1
WR
RD
INT1
VIN
McMaster University Dept.of ECE