EGCP 2110-01
Microprocessors
Laboratory #11

Service Interrupt

Prepared By:
Alex Laird
Collin Barrett

on

Saturday, December 6th, 2009

Low-Level Source Code

Below is the assembly code that we ensured worked in the lab. We tested it extensively on the Z80 microprocessor and across our circuit and a serial connection a PC. Upon presentation, our code ran correctly.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Created by: Alex Laird and Collin Barrett
;Date: Nov. 18, 2009
;Class: Microprocessors
;Lab 10: Service Interrupt
;Description: Outputs a voltage received from an input port
;             of the ADC either once or twice every second.
;             Interfaces with a serial connection to the PC,
;             so the user has the ability to press a key on
;             they keyboard to alter functionality.
;             Press 1: The sample rate will change to 1
;                      sample/sec.
;             Press 2: The sample rate will change to 2
;                      sample/sec.
;             Press A: Averaging of five voltages before
;                      outputting will be enabled.
;             Press a: Averaging will be disabled and the
;                      instantaneous voltage will be
;                      displayed.
;             Press H: Enable hex display of voltages.
;             Press h: Enable decimal display of voltages.
;             Press P: Pause the voltage display until unpaused.
;             Press U: Unpause and continue voltage display at the specified sample
;                      rate.
;             Press D or d: Displays a string of characters
;                           containing the developer's names.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Declare aliases for commonly used memory addresses.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ADC   EQU   0C8H
BUFFER      EQU   0C0H
ADATA EQU   0D0H
ACMD  EQU   0D2H
BCMD  EQU   0D3H
DELAY EQU   0FFDH
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;place the program at the beginning of user memory
      ORG   1800H

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Initializes the SIO chip with appropriate values and sets
; the interrupt mode on the Z80 to IM 2.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SETUP:
;setup interrupt register
      LD    A, 1DH            ;set byte for interrupt
```

```
      LD    I, A

;setup SIO chip
      LD    A, 18H            ;reset all SIO parameters
      OUT   (ACMD), A

      LD    A, 02H            ;set pointer to register 2
      OUT   (BCMD), A
      LD    A, 0FEH           ;load register 2 with 0FEH
      OUT   (BCMD), A

      LD    A, 01H            ;set pointer to register 1
      OUT   (BCMD), A
      LD    A, 00H            ;load register 1 with 00H
      OUT   (BCMD), A

      LD    A, 04H            ;set pointer to register 4
      OUT   (ACMD), A
      LD    A, 04H            ;load register 4 with 04H
      OUT   (ACMD), A

      LD    A, 03H            ;set pointer to register 3
      OUT   (ACMD), A
      LD    A, 0C1H           ;load register 3 with 0C1H
      OUT   (ACMD), A

      LD    A, 05H            ;set pointer to register 5
      OUT   (ACMD), A
      LD    A, 068H           ;load register 5 with 068H
      OUT   (ACMD), A

      LD    A, 01H            ;set pointer to register 1
      OUT   (ACMD), A
      LD    A, 18H            ;load register 1 with 18H
      OUT   (ACMD), A

      LD    HL, 1E00H   ;pointer to the ISR
      LD    (1DFEH), HL ;load interrupt address to ROM

;enable the interrupt mode 2 on the processor
      IM    2
      EI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Initializes memory locations and registers to their default
; values.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;reset registers that aren't loaded
      LD    A, 00H
      LD    D, A
      LD    E, A
;initialize the TIME memory location with 1 sample/sec
      LD    HL, 1000D
      LD    (TIME), HL
;initialize averaging to off by default
      LD    A, 'F'
```

```
      LD    (AVG), A
;initialize to decimal output
      LD    (HEX), A
;initialize as unpaused
      LD    (PAUSED), A
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Maintains control of all program functionality, looping
; back to itself after every significant function or
; interrupt trigger.  Since interrupts are enabled, if an
; interrupt is received, the Z80 will jump to the ISR routine
; (at 1A00, near the bottom of this code) and then return to
; this function.
; This function polls the least-significant-bit of the ADC.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN:
;initialize the RESULT memory location to zero
      LD    HL, 0000H
      LD    (RESULT), HL
;take five samples if averaging is enabled, otherwise just one
      LD    C, 05H
      LD    A, (AVG)
      CP    'T'
      JP    Z, READ
      LD    C, 01H
READ:
      OUT   (ADC), A
POLL:
      IN    A, (BUFFER)
      BIT   0, A
      JP    NZ, POLL

;continue only if output is unpaused
      LD    A, (PAUSED)
      CP    'T'
      JP    Z, READ

;get a voltage sample (loop if averaging)
      CALL  SAMPLE
      DEC   C
      JP    NZ, READ

;if averaging is disabled, multiply the sample by five
      LD    A, (AVG)
      CP    'T'
      CALL  NZ, MFIVE

;display voltage output
      LD    A, (HEX)
      CP    'T'
      CALL  Z, PRINTH
      CP    'T'
      CALL  NZ, PRINTD

;delay as long as the TIME is set to
      LD    HL, (TIME)
```

```
      CALL  DELAY

;jump back to main initialization
      JP    MAIN
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Collects a sample of the voltage across the ADC pin
; attached to the 10k pot and stores it in the RESULT memory
; location.
; This function pushes the BC and HL registers to the stack,
; so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SAMPLE:
      PUSH  BC
      PUSH  HL

;read the value from the ADC and add it to RESULT memory location
      IN    A, (ADC)
      LD    H, 00H
      LD    L, A
      LD    BC, (RESULT)
      ADD   HL, BC

;load added value back into RESULT memory location
      LD    (RESULT), HL

      POP   HL
      POP   BC
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; When averaging is disabled, the result should be multiplied
; by five to attain accuracy. Multiplies the sample stored in
; the RESULT memory location by five and stores it back in
; RESULT.
; This function pushes the BC and HL registers to the
; stack, so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MFIVE:
      PUSH  BC
      PUSH  HL

;load from the RESULT memory location
      LD    HL, (RESULT)
      LD    H, 00H
      LD    B, H
      LD    C, L

;add four times for the multiplication
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC

;load added value back into RESULT memory location
```

```
      LD    (RESULT), HL

      POP   HL
      POP   BC
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Multiplies the value stored in the RESULT memory location
; by ten. It is assumed. The result is then stored back in the
; RESULT memory location.
; This function pushes the BC and HL registers to the stack,
; so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MTEN:
      PUSH  BC
      PUSH  HL

;load from the RESULT memory location
      LD    HL, (RESULT)
      LD    H, 00H
      LD    B, H
      LD    C, L

;add nine times for the multiplication
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC
      ADD   HL, BC

;store the result back in memory
      LD    (RESULT), HL

      POP   HL
      POP   BC
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Outputs the voltage stored in the RESULT memory location as
; a hex value to the serial port and a decimal value to BCD.
; This function pushes the AF, BC, and HL registers to the
; stack, so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PRINTH:
      PUSH  AF
      PUSH  BC
      PUSH  HL

;prepare one's digit for BCD output
      LD    HL, (RESULT)
      LD    B, H
```

```
      SLA   B
      SLA   B
      SLA   B
      SLA   B

;store RESULT and multiply by ten
      PUSH  HL
      CALL  MTEN

;combine ones and tenths and output to BCD
      LD    HL, (RESULT)
      LD    A, B
      OR    H
      OUT   (BUFFER), A

;revert RESULT since hex doesn't require multiplication
      POP   HL
      LD    (RESULT), HL

;output hex prefix
      CALL  CHBUF
      LD    A, '0'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'x'
      OUT   (ADATA), A

;display first hex character
      LD    HL, (RESULT)
      LD    A, H
      AND   0FH
      CALL  HEXPREP
      CALL  CHBUF
      OUT   (ADATA), A

;output decimal point
      CALL  CHBUF
      LD    A, '.'
      OUT   (ADATA), A

;display second hex character
      LD    HL, (RESULT)
      LD    A, L
      SRL   A
      SRL   A
      SRL   A
      SRL   A
      CALL  HEXPREP
      CALL  CHBUF
      OUT   (ADATA), A

;display third hex character
      LD    A, L
      AND   0FH
      CALL  HEXPREP
      CALL  CHBUF
      OUT   (ADATA), A
```

```
;output units
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'V'
      OUT   (ADATA), A

      CALL ENDLNE

      POP   HL
      POP   BC
      POP   AF
      RET

;prepare hex for output depending it being a letter or number
HEXPREP:
      CP    10D
      JP    NC, HLETTER
HNUMBER:
      ADD   A, '0'
      RET
HLETTER:
      SUB   0AH
      ADD   A, 'A'
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Outputs the voltage stored in the RESULT memory location as
; a decimal, good to three decimal places, to both the BCD
; and the serial port.
; This function pushes the AF, BC and HL registers to the
; stack, so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PRINTD:
      PUSH  AF
      PUSH  BC
      PUSH  HL

;prepare one's digit for BCD output
      LD    HL, (RESULT)
      LD    B, H
      SLA   B
      SLA   B
      SLA   B
      SLA   B

;output the one's digit to the serial connection
      CALL  CHBUF
      LD    A, (RESULT+1)
      ADD   A, 30H
      OUT   (ADATA), A

;output the decimal point
      CALL  CHBUF
```

```
      LD    A, '.'
      OUT   (ADATA), A

;output tenths
      CALL  MTEN
      CALL  CHBUF
      LD    A, (RESULT+1)
      ADD   A, 30H
      OUT   (ADATA), A

;combine ones and tenths and output to BCD
      LD    HL, (RESULT)
      LD    A, B
      OR    H
      OUT   (BUFFER), A

;output hundredths
      CALL  MTEN
      CALL  CHBUF
      LD    A, (RESULT+1)
      ADD   A, 30H
      OUT   (ADATA), A

;output thousandths
      CALL  MTEN
      CALL  CHBUF
      LD    A, (RESULT+1)
      ADD   A, 30H
      OUT   (ADATA), A

;output units
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'V'
      OUT   (ADATA), A

      CALL ENDLNE

      POP   HL
      POP   BC
      POP   AF
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Waits to return until the buffer is empty so data can be
; sent.
; This function pushes the AF registers to the stack, so they
; remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
CHBUF:
      PUSH  AF
CHBUF2:
      IN    A, (ACMD)
      BIT   2, A
```

```
      JP    Z, CHBUF2

      POP   AF
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Outputs the set of characters ending a line and starting a
; new one to the serial output.
; This function pushes the AF registers to the stack, so they
; remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ENDLNE:
      PUSH  AF

      CALL  CHBUF
      LD    A, 0DH
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 0AH
      OUT   (ADATA), A

      POP   AF
      RET
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Sends the characters 'Alex Laird and Collin Barrett' as a
; stream on the serial output.
; This function does no pushing or popping, and register A is
; effected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DISPNMS:
      CALL  CHBUF
      LD    A, 'A'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'x'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'L'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
```

```
CALL  CHBUF
LD    A, 'r'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'd'
OUT   (ADATA), A
CALL  CHBUF
LD    A, ' '
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'a'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'n'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'd'
OUT   (ADATA), A
CALL  CHBUF
LD    A, ' '
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'C'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'o'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'l'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'l'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'i'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'n'
OUT   (ADATA), A
CALL  CHBUF
LD    A, ' '
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'B'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'a'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'r'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'r'
OUT   (ADATA), A
CALL  CHBUF
LD    A, 'e'
OUT   (ADATA), A
```

```
      CALL  CHBUF
      LD    A, 't'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 't'
      OUT   (ADATA), A

      CALL  ENDLNE

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Sets the sample rate to 1/second. This sets the value of
; the HL register, which is used to the built-in DELAY
; subroutine, to 1000D so the program will collect once per
; second. Sends the characters '1 Sample/sec' as a stream on
; serial output.
; This function pushes the HL registers to the stack, so they
; remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ONE:
      PUSH  HL

      CALL  CHBUF
      LD    A, '1'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'm'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'p'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, '/'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
```

```
      CALL  CHBUF
      LD    A, 'c'
      OUT   (ADATA), A

      CALL ENDLNE

;set the sample time to one second
      LD    HL, 1000D
      LD    (TIME), HL

      POP   HL
      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Sets the sample rate to 2/second. This sets the value of
; the HL register, which is used to the built-in DELAY
; subroutine, to 500D so the program will collect twice per
; second. Sends the characters '2 Sample/sec' as a stream on
; serial output.
; This function pushes the HL registers to the stack, so they
; remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TWO:
      PUSH  HL

      CALL  CHBUF
      LD    A, '2'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'm'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'p'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, '/'
      OUT   (ADATA), A
      CALL  CHBUF
```

```
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'c'
      OUT   (ADATA), A

      CALL ENDLNE

;set the sample time to one second
      LD    HL, 500D
      LD    (TIME), HL

      POP   HL
      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Enables averaging. Sends the characters 'Enabled Averaging'
; as a stream on output buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EAVG:
      CALL  CHBUF
      LD    A, 'A'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'v'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'r'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'g'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'n'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'g'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
```

```
      CALL  CHBUF
      LD    A, 'n'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'b'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A

      CALL ENDLNE

;set averaging state to enabled
      LD    A, 'T'
      LD    (AVG), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Disables averaging. Sends the characters 'Enabled Averaging'
; as a stream on output buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DAVG:
      CALL  CHBUF
      LD    A, 'A'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'v'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'r'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'g'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'n'
      OUT   (ADATA), A
```

```
      CALL  CHBUF
      LD    A, 'g'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'b'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A

      CALL ENDLNE

;set averaging state to disabled
      LD    A, 'F'
      LD    (AVG), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Enables hex output for the voltage. Sends the characters
; 'Hex display' as a stream on output buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EHEX:
      CALL  CHBUF
      LD    A, 'H'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'x'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
```

```
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'p'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'y'
      OUT   (ADATA), A

      CALL ENDLNE

;set averaging state to enabled
      LD    A, 'T'
      LD    (HEX), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Disables hex output (thus enabling decimal output). Sends
; the characters 'Decimal display' as a stream on output
; buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DHEX:
      CALL  CHBUF
      LD    A, 'D'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'c'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'm'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
```

```
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, ' '
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'i'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'p'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'l'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'y'
      OUT   (ADATA), A

      CALL ENDLNE

;set averaging state to enabled
      LD    A, 'F'
      LD    (HEX), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Pauses the output of voltages. Sends the characters
; 'Paused' as a stream on output buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PAUSE:
      CALL  CHBUF
      LD    A, 'P'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'u'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
```

```
      OUT   (ADATA), A

      CALL ENDLNE

;set paused state to true
      LD    A, 'T'
      LD    (PAUSED), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Unpauses the output of voltages at the specified sample
; rate. Sends the characters 'Unpaused' as a stream on output
; buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
UNPAUSE:
      CALL  CHBUF
      LD    A, 'U'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'n'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'p'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'a'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'u'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 's'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'e'
      OUT   (ADATA), A
      CALL  CHBUF
      LD    A, 'd'
      OUT   (ADATA), A

      CALL ENDLNE

;set paused state to false
      LD    A, 'F'
      LD    (PAUSED), A

      JP    ENDISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The Interrupt Service Routine (ISR) is activated when the
; SIO chip sends an interrupt signal to the Z80. Upon
; activation, interrupts are immediately disabled until the
; ISR has finished. This routine pushes the AF registers to
; the stack, so they remain unaffected.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ISR:
      ORG   1E00H
      DI
      PUSH  AF

;read in data that caused the interrupt
      IN    A, (ADATA)

;analyze received data
      CP    'D'
      JP    Z, DISPNMS
      CP    'd'
      JP    Z, DISPNMS
      CP    '1'
      JP    Z, ONE
      CP    '2'
      JP    Z, TWO
      CP    'A'
      JP    Z, EAVG
      CP    'a'
      JP    Z, DAVG
      CP    'H'
      JP    Z, EHEX
      CP    'h'
      JP    Z, DHEX
      CP    'P'
      JP    Z, PAUSE
      CP    'p'
      JP    Z, PAUSE
      CP    'U'
      JP    Z, UNPAUSE
      CP    'u'
      JP    Z, UNPAUSE

ENDISR:
      POP   AF
      EI
      RETI
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Declare memory storage locations, aliases, and their sizes.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
AVG:  DEFS  1
HEX:  DEFS  1
PAUSED:     DEFS  1
TIME: DEFS  2
RESULT:     DEFS  2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Program Analysis

Utilizing the circuit that we've designed over the entire semester, our code is designed to retrieve a voltage from an attach 10k pot on the trainer board and report that voltage to PC monitor through a serial connection. The voltage is read in on an Analog to Digital Convert (ADC) interrupt signal through the ADC.

Specific details in the display can be changed through keys pressed on the keyboard. For instance, pressing the number "2" changes the sample time from once to twice per second. Additional specifications that our program allows are described in full at the beginning of our code. Key presses are recognized through a Serial Input/Output (SIO) controller interrupt that signals the Interrupt Service Routine (ISR) routine in our code.

The program initially sets up the SIO chip for communication with the Z80 microprocessor and the code we placed on it. It then falls into the main loop, where it waits for a voltage reading from the ADC. Upon receiving a signal indicating a voltage is ready to be read, that raw voltage is sampled and converted into a displayable value either in hex or decimal (depending on the user specifications). After the voltage display, the program returns to the main loop to and waits for the ADC interrupt again.

Upon key press, the serial connection sends an interrupt signal to the SIO chip which is in turn sent to the Z80s interrupt. Our Z80 Interrupt Service Routine (ISR) is placed at 1E00H in memory, and when the SIO chip is enabled at the beginning of our code, the Z80 interrupt is also enabled with this memory location. The key press causes this interrupt which immediately causes the Z80 to jump to our ISR, read the key press, perform whatever operation is necessary for that key press (if any), and return to the location in memory the Z80 was interrupted from.

Either once or twice per second (depending on the sample time), the voltage is read and output to both the monitor and the Binary Coded Decimal (BCD) display on the trainer board. On the monitor, the voltage can be output in either decimal format (PRINTD function) or hex format (PRINTH function).

The PRINTD first outputs the ones place to the monitor since, when it is read, it is already ready for output. For each following decimal position, all that needs to happen is to shift our sampled data to the left. This is achieved by multiplying our voltage by ten each time and grabbing the left most value for display. For the BCD display, this is only done once (the ones value is immediately output, and the BCD can only display one decimal position), but for the monitor this is done three times (so our accuracy is good to three decimal places). Of course, when the values our output to the monitor, they must be converted to ASCII numbers, so 30H is added to each value as it is output.

The PRINTH function is far simpler. The BCD is still output in the same way it is for the PRINTD function, but our values are already in hex format when they are read in, so all that needs to be done is to prepare numerical values (since hex may still have numbers) for ASCII output, shift as

necessary to put the values in the proper positions in the registers, and output to the values to the monitor.

The difference between averaging and non-averaging modes was trivial. In fact, for the most part, it was unnoticeable. You could see an instantaneous change in the displayed voltage when averaging mode was turned on or off, but in general, average a second, the values would smooth out and be identical. Our monitor and BCD values were always the same, and our gathered values compared to that of the voltmeter were never more than 0.070 V off (with a maximum error of 1.62%), as shown in the table below.

Voltages	PC Output	Multimeter Reading
0.0 V	0.000 V	0.000 V
0.5 V	0.488 V	0.5028 V
1.0 V	1.015 V	1.0133 V
1.5 V	1.523 V	1.5000 V
2.0 V	2.031 V	2.0124 V
2.5 V	2.539 V	2.5097 V
3.0 V	3.066 V	3.0196 V
3.5 V	3.554 V	3.496 V
4.0 V	4.082 V	4.016 V
4.5 V	4.550 V	4.497 V
5.0 V	4.980 V	4.947 V

Table 1 Recorded Voltages

Conclusions

The initial code for this lab was written in a matter of two hours, but the debugging process took upwards of six hours. Though our circuit was wired and performing properly, there were a few hiccups in the code that were simply difficult to catch and fix. In general, however, the debugging process was smooth; it just took a substantial amount of time. Overall, the lab went very well and was very beneficial to an understanding of circuitry, its workings with assembly coding and microprocessors, and interrupts.