```
;***************************************************************************
;*
;* Title: BCD to Hex
;* Author:  Judah Ben-Eliezer
;* Version: 1.0
;* Last updated:
;* Target:          ;ATmega4809 @3.3MHz
;*
;* DESCRIPTION
;* This program polls the flag associated with pushbutton 1. This flag is
;* connected to PE0. If the flag is set, the contents of the array bcd_entries
;* is shifted left and the BCD digit set on the least significant 4 bits of
;* PORTA_IN are stored in the least significant byte of the bcd_entries array.
;* Then the corresponding segment values for each digit in the bcd_entries
;* display are written into the led_display. Note: entry of a non-BCD value
;* is ignored.
;*
;* This program also continually multiplexes the display so that the digits
;* entered are constantly seen on the display. Before any digits are entered
;* the display displays 0000.
;*
;* This program also polls the flag associated with pushbutton 2. This flag
;* is connected to PE2. If the flag is set, the digits in the bcd_entries
;* array are read and passed to the prewritten subroutine BCD2bin16. This
;* subroutine performs a BCD to binary conversion. The binary result is
;* partitioned into hexadecimal and placed into the array hex_results. The
;* contents of the hex_results array is converted to seven segment values
;* and placed into the led_display array. The multiplexing then causes
;* the hexadecimal equivalent of the BCD value entered to be displayed in
;* hexadecimal.
;*
;* VERSION HISTORY
;* 1.0 Original version
;***************************************************************************

.nolist
.include "m4809def.inc"
.list

.dseg
bcd_entries: .byte 4
led_display: .byte 4
digit_num: .byte 1
hex_results: .byte 4

.cseg
start:
    cbi VPORTE_DIR, 0
    ldi r16, $00
    out VPORTA_DIR, r16
    com r16
    out VPORTD_DIR, r16
    out VPORTC_DIR, r16
    ldi XH, HIGH(bcd_entries)
    ldi XL, LOW(bcd_entries)
    ldi YH, HIGH(led_display)
    ldi YL, LOW(led_display)
    com r16
    st X+, r16
    inc r16
    st X+, r16
    inc r16
    st X+, r16
    inc r16
    st X, r16
    cbi VPORTE_OUT, 1
    sbi VPORTE_OUT, 1

main_loop:
    rcall multiplex_display
    rcall mux_digit_delay
    rcall poll_digit_entry
    rcall poll_bcd_hex
    rjmp main_loop


;***************************************************************************
;*
;* "poll_digit_entry" - Polls Pushbutton 1 for Conditional Digit Entry
;*
;* Description:
;* Polls the flag associated with pushbutton 1. This flag is connected to
;* PE0. If the flag is set, the contents of the array bcd_entries is shifted
;* left and the BCD digit set on the least significant 4 bits of PORTA_IN are
;* stored in the least significant byte of the bcd_entries array. Then the
;* corresponding segment values for each digit in the bcd_entries display are
;* written into the led_display. Note: entry of a non-BCD value is ignored.
;* Author:
;* Version:
;* Last updated:
;* Target:
;* Number of words:
;* Number of cycles:
;* Low registers modified:
;* High registers modified:
;*
;* Parameters:
;* Returns:
;*
;* Notes:
;*
;***************************************************************************
poll_digit_entry:
    ldi XH, HIGH(bcd_entries)
    ldi XL, LOW(bcd_entries)
    sbis VPORTE_IN, 0
    rjmp poll_digit_entry
    in r16, VPORTA_IN
    rcall reverse_bits
    rcall check_for_non_bcd
    rcall shift_bcd_entries
    rcall bcd_to_led

;***************************************************************************
;*
;* "poll_bcd_hex" - Polls Pushbutton 2 for Conditional Conversion of BCD to
;* Hex.
;*
;* Description:
;* Polls the flag associated with pushbutton 2. This flag is connected to
;* PE2. If the flag is set, the digits in the bcd_entries array are read
;* and passed to the prewritten subroutine BCD2bin16. This subroutine
;* performs a BCD to binary conversion. The binary result is partitioned
;* into hexadecimal and placed into the array hex_results. The contents of
;* the hex_results array is converted to seven segment values and placed
;* into the led_display array.
;* Author:
;* Version:
;* Last updated:
;* Target:
;* Number of words:
;* Number of cycles:
;* Low registers modified:
;* High registers modified:
;*
;* Parameters:
;* Returns:
;*
```

```
;* Notes:
;*
;***************************************************************************
poll_bcd_hex:
    sbis VPORTE_IN, 2
    ret
    ldi XH, HIGH(bcd_entries)
    ldi XL, LOW(bcd_entries)
    ld r18, X+
    ld r17, X+
    swap r17
    ld r19, X+
    or r17, r19
    ld r16, X+
    swap r16
    ld r19, X+
    or r16, r19
    rcall BCD2bin16
    ldi XH, HIGH(hex_results)
    ldi XL, LOW(hex_results)
    ldi r19, $00
    or r19, r15
    andi r19, $F0
    swap r19
    st X+, r19
    lds r20, r15
    lds r21, r14
    andi r20, $0F
    st X+, r20
    ldi r19, $00
    or r19, r21
    andi r19, $F0
    swap r19
    st X+, r19
    andi r21, $0F
    st X, r21
    ret


;***************************************************************************
;*
;* "mux_digit_delay" - title
;*
;* Description: delays 0.1 * r23
;*
;* Author:  Judah Ben-Eliezer
;* Version: 1.0
;* Last updated:
;* Target:
;* Number of words:
;* Number of cycles:
;* Low registers modified:
;* High registers modified:
;*
;* Parameters:
;* Returns:
;*
;* Notes:
;*
;***************************************************************************
mux_digit_delay:
    ldi r23, $08 ; 0.1 * r23 = delay
outer_loop:
    ldi r24, $06
inner_loop:
    dec r24
    brne inner_loop
    dec r23
    brne outer_loop
    ret


;***************************************************************************
;*
;* "reverse_bits" - Reverse Bits
;*
;* Description: Reverses the bit positions in a byte passed in. Bit 0
;* becomes bit 7, bit 6 becomes bit 1, and so on.
;*
;* Author:                  Judah Ben-Eliezer
;* Version:                 1.0
;* Last updated:            101120
;* Target:                  ATmega4809
;* Number of words:         8
;* Number of cycles:
;* Low registers modified:  r16, r17, r18
;* High registers modified: none
;*
;* Parameters: r16: byte to be reversed.
;* Returns: r16: reversed byte
;*
;* Notes:
;*
;***************************************************************************
reverse_bits:
    ldi r18, $08
loop_8:
    ror r16
    rol r17
    dec r18
    brne loop_8
    ret

check_for_non_bcd:
    cpi r17, $0A
    brsh reset
    ret

reset:
    cbi VPORTE_OUT, 1
    sbi VPORTE_OUT, 1
    ret

shift_bcd_entries:
    ldi r18, $03
shift_loop:
    ldi XH, HIGH(bcd_entries)
    ldi XL, LOW(bcd_entries)
    dec r18
    add XL, r18
    ld r19, X+
    st X, r19
    brne shift_loop
    ret

bcd_to_led:
    ldi XL, LOW(bcd_entries)
    ldi XH, HIGH(bcd_entries)
    st X, r17
    ldi r20, $04
conversion_loop:
    dec r20
    ldi XH, HIGH(bcd_entries)
    ldi XL, LOW(bcd_entries)
    ldi YH, HIGH(led_display)
    ldi YL, LOW(led_display)
    add XL, r20
    add YL, r20
    ld r18, X
    rcall hex_to_7seg
    st Y, r18
    cpi r20, $00
    brne conversion_loop
    ret

multiplex_display:
    ldi r22, $00
    sts digit_num, r22
    ldi YL, LOW(led_display)
    lds r17, digit_num
    lds r20, digit_num
    andi r17, $03
    add XL, r17
    ld r18, Y
    ldi r21, $80
    inc r20
loop:
    lsr r21
    dec r20
    brne loop
    lsl r21
    com r21
    out VPORTC_OUT, r21
    out VPORTD_OUT, r18
    cbi VPORTE_OUT, 1
    sbi VPORTE_OUT, 1
    inc r17
    sts digit_num, r17
    ret

;***************************************************************************
;*
;* "BCD2bin16" - BCD to 16-Bit Binary Conversion
;*
;* This subroutine converts a 5-digit packed BCD number represented by
;* 3 bytes (fBCD2:fBCD1:fBCD0) to a 16-bit number (tbinH:tbinL).
;* MSD of the 5-digit number must be placed in the lowermost nibble of fBCD2.
;*
;* Let "abcde" denote the 5-digit number. The conversion is done by
;* computing the formula: 10(10(10(10a+b)+c)+d)+e.
;* The subroutine "mul10a"/"mul10b" does the multiply-and-add operation
;* which is repeated four times during the computation.
;*
;* Number of words  :30
;* Number of cycles :108
;* Low registers used   :4 (copyL,copyH,mp10L/tbinL,mp10H/tbinH)
;* High registers used  :4 (fBCD0,fBCD1,fBCD2,adder)
;*
;***************************************************************************

;***** "mul10a"/"mul10b" Subroutine Register Variables

.def    copyL   =r12        ;temporary register
.def    copyH   =r13        ;temporary register
.def    mp10L   =r14        ;Low byte of number to be multiplied by 10
.def    mp10H   =r15        ;High byte of number to be multiplied by 10
.def    adder   =r19        ;value to add after multiplication

;***** Code

mul10a: ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" high nibble
    swap    adder
mul10b: ;***** multiplies "mp10H:mp10L" with 10 and adds "adder" low nibble
    mov copyL,mp10L ;make copy
    mov copyH,mp10H
    lsl mp10L       ;multiply original by 2
    rol mp10H
    lsl copyL       ;multiply copy by 2
    rol copyH
    lsl copyL       ;multiply copy by 2 (4)
    rol copyH
    lsl copyL       ;multiply copy by 2 (8)
    rol copyH
    add mp10L,copyL ;add copy to original
    adc mp10H,copyH
    andi    adder,0x0f  ;mask away upper nibble of adder
    add mp10L,adder ;add lower nibble of adder
    brcc    m10_1       ;if carry not cleared
    inc mp10H       ;   inc high byte
m10_1:  ret

;***** Main Routine Register Variables

.def    tbinL   =r14        ;Low byte of binary result (same as mp10L)
.def    tbinH   =r15        ;High byte of binary result (same as mp10H)
.def    fBCD0   =r16        ;BCD value digits 1 and 0
.def    fBCD1   =r17        ;BCD value digits 2 and 3
.def    fBCD2   =r18        ;BCD value digit 5

;***** Code

BCD2bin16:
    andi    fBCD2,0x0f  ;mask away upper nibble of fBCD2
    clr mp10H
    mov mp10L,fBCD2 ;mp10H:mp10L = a
    mov adder,fBCD1
    rcall   mul10a      ;mp10H:mp10L = 10a+b
    mov adder,fBCD1
    rcall   mul10b      ;mp10H:mp10L = 10(10a+b)+c
    mov adder,fBCD0
    rcall   mul10a      ;mp10H:mp10L = 10(10(10a+b)+c)+d
    mov adder,fBCD0
    rcall   mul10b      ;mp10H:mp10L = 10(10(10(10a+b)+c)+d)+e
    ret



;***************************************************************************
;*
;* "hex_to_7seg" - Hexadecimal to Seven Segment Conversion
;*
;* Description: Converts a right justified hexadecimal digit to the seven
;* segment pattern required to display it. Pattern is right justified a
;* through g. Pattern uses 0s to turn segments on ON.
;*
;* Author:                      Ken Short
;* Version:                     1.0
;* Last updated:                101620
;* Target:                      ATmega4809
;* Number of words:             8
;* Number of cycles:            13
;* Low registers modified:      none
;* High registers modified:     r16, r18, ZL, ZH
;*
;* Parameters: r18: right justified hex digit, high nibble 0
;* Returns: r18: segment values a through g right justified
;*
;* Notes:
;*
;***************************************************************************

hex_to_7seg:
    andi r18, 0x0F              ;clear ms nibble
    ldi ZH, HIGH(hextable * 2)  ;set Z to point to start of table
    ldi ZL, LOW(hextable * 2)
    ldi r16, $00                ;add offset to Z pointer
    add ZL, r18
    adc ZH, r16
    lpm r18, Z                  ;load byte from table pointed to by Z
    ret

    ;Table of segment values to display digits 0 - F
    ;!!! seven values must be added - verify all values
hextable: .db $01, $4F, $12, $06, $4C, $24, $20, $0F, $00, $04, $08, $60, $31, $42,
$30, $38
```