ECE 367 - Experiment #6
Kitchen Timer

Spring 2006 Semester

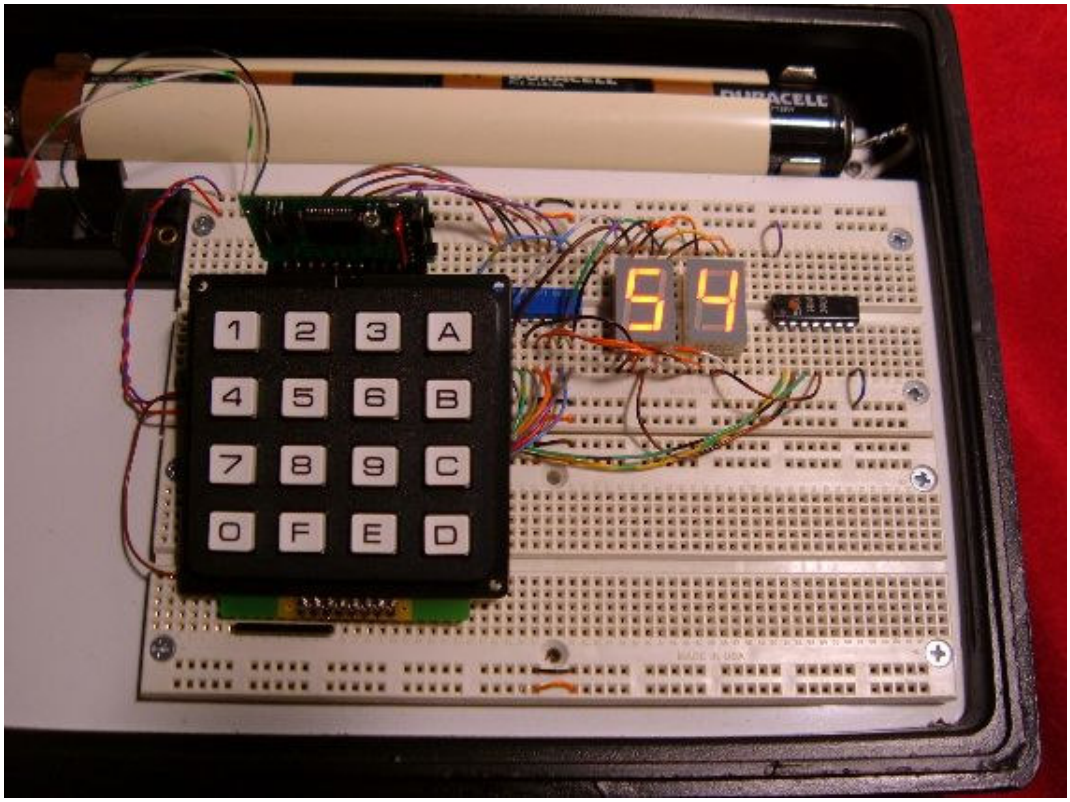
Introduction

This experiment has you construct a circuit interfacing nine I/O lines from the 68HC11 with two seven segment displays and a matrix keypad, and write assembly language code to realize a programmable countdown "kitchen timer" that allows the user to perform various functions: preset count value, start/resume and pause countdown. The purpose of this experiment is to teach advanced hardware and software techniques of interfacing microcontrollers.

Required Hardware

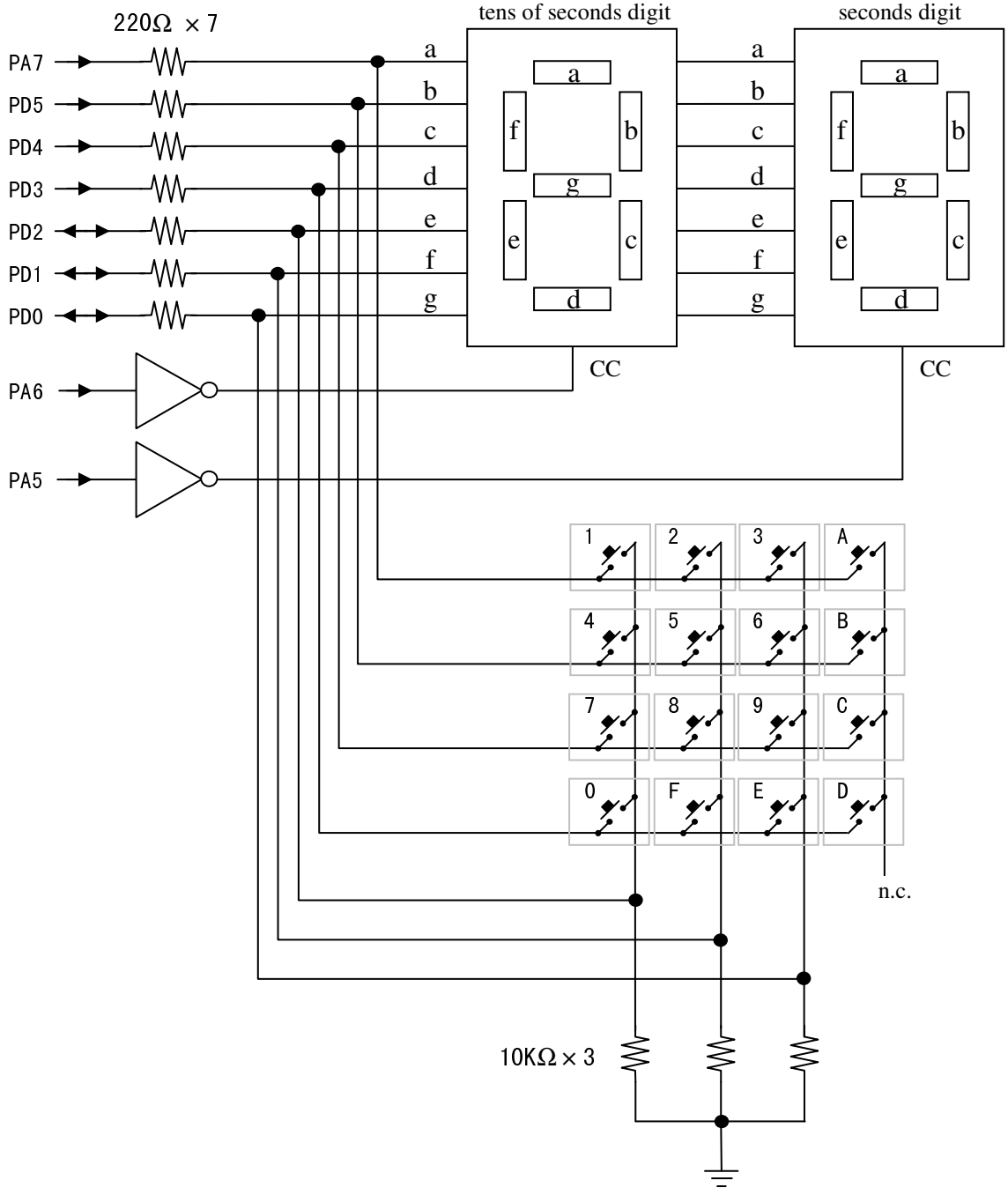
In addition to the MicroStamp11 module, this experiment requires two seven segment LED displays, a matrix keypad, two TTL hex inverters (one 74LS04 IC), the 220Ω DIP resistors and the 10KΩ SIP resistors.

Here is a photo of the completed circuit:

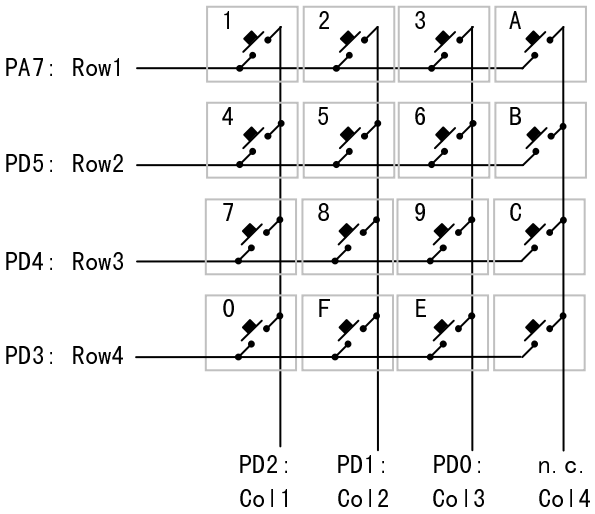


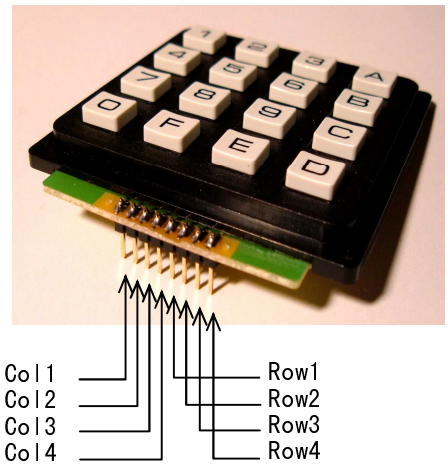
Wiring Diagram

Build the following circuit on a solderless breadboard – the circuit is very similar to that in Experiment 5. The bidirectional arrows next to {PD2, PD1, PD0} indicate that these lines will be used for both input and output during run-time.

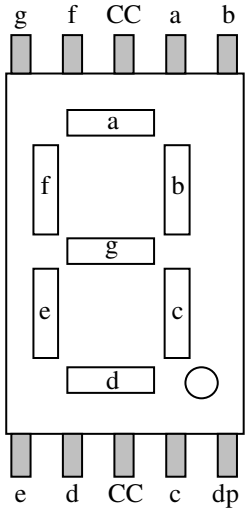


Once again, for your reference, here are pinout diagrams of the matrix keypad and seven segment LED display in your lab parts kit:

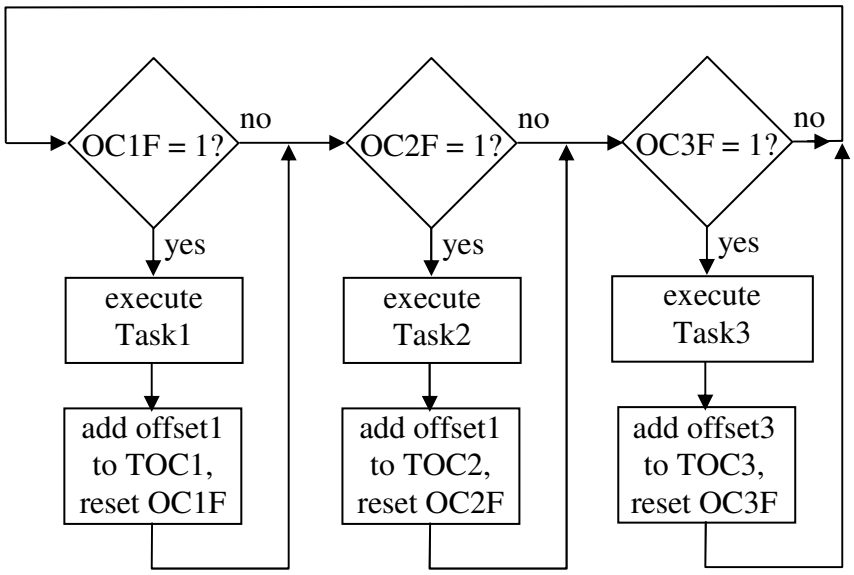




Segment:	Output Line:
a	PA7
b	PD5
c	PD4
d	PD3
e	PD2
f	PD1
g	PD0
CC (left digit)	PA6
CC (right digit)	PA5



Software Design



In this experiment you will be using TCNT and three Output Compare registers to execute three tasks at different frequencies. Here is a high-level description of the three tasks:

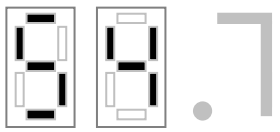
Task1 keeps count of time using three decimal digits (as in 54.7 sec). This count is decremented every 0.1 sec when the timer is running. Task1 executes every 1/10 sec.

Task2 multiplexes and updates the 7-segment displays; it executes every 1/200 sec.

Task3 polls the keypad to detect key presses and responds accordingly; it executes every 1/20 sec.

Task1 Details

The memory byte “Mode” is used to indicate whether or not the counter is running or is paused: Mode = 1 indicates that the timer is counting down, and Mode = 0 indicates that it is frozen at the current count.



Digit3 Digit2 Digit1

(The hardware displays only Digit3 and Digit2)

Task1 pseudocode:
(executed every 0.1 sec)

```
if (Mode = 1 and Key_Pressed = 0)
        Digit1 ← Digit1 − 1                 ; decrement 0.1 sec digit
        if (Digit1 = −1)
                Digit1 ← 9
                Digit2 ← Digit2 − 1         ; decrement 1.0 sec digit
                if (Digit2 = −1)
                        Digit2 ← 9
                        Digit3 ← Digit3 − 1 ; decrement 10 sec digit
                        if (Digit3 = −1)
                                Digit1 ← 0  ; end of countdown is reached
                                Digit2 ← 0
                                Digit3 ← 0
                                Mode  ← 0   ; stop counter
                        end if
                end if
        end if
end if
```

(initially Digit1 = 0, Digit2 = 0, Digit3 = 0, Mode = 0)

See Task3 description for explanation of “Key_Pressed”.

Task2 Details

Task2 performs time-division multiplexing by alternating the digit being displayed every 1/200 sec. This results in an overall 100 Hz refresh rate for both digits. Memory byte "Digit_Select" is an indicator of what digit (left or right) is currently being displayed.

Task2 pseudocode:
(executed every 0.005 sec)

```
if (Digit_Select = 0)
    Digit_Select ← 1          ; activate left 7-seg. display
    PA6 ← 1
    PA5 ← 0
    Digit ← Digit3            ; display tens of sec value
else
    Digit_Select ← 0          ; activate right 7-seg. display
    PA6 ← 0
    PA5 ← 1
    Digit ← Digit2            ; display sec value
end if
```

Then, output 7-segment data corresponding to memory value Digit just as it was done in Experiments 4 and 5.

(initially Digit_Select = 0, Digit = 0, PA6 = 0, PA5 = 1)

Task3 Details

Task3 executes every 1/20 sec; it checks if any key is pressed and responds accordingly. Memory byte "Key_Pressed" keeps track of the keypad status: Key_Pressed = 1 when a keypress is detected, Key_Pressed = 0 when no keys are pressed.

To freeze the count, one must press any key in the first three columns of the keypad (A, B, C, D keys are inactive). To start the countdown sequence one must press and release either E or F when the count is frozen. To enter a new starting time value one must press numeric keys when the count is frozen.

Task3 pseudocode:
(executed every 0.05 sec)

```
if (any key is pressed)
        if (Key_Pressed = 0)
                Key_Pressed ← 1
                if (Mode = 1)                           ; if now counting then
                        Mode ← 0                        ;  stop counting
                else
                        if (E or F is pressed)
                                Mode ← 1                ; start/resume counting
                        else
                                Digit3 ← Digit2, Digit1 ← 0
                                ; initialize count by shifting in digits from the right
                                if (0 is pressed)
                                        Digit2 ← 0
                                elseif (1 is pressed)
                                        Digit2 ← 1
                                elseif (2 is pressed)
                                        Digit2 ← 2
                                   ⋮       ⋮       ⋮
                                elseif (9 is pressed)
                                        Digit2 ← 9
                                end if
                        end if
                end if
        end if
else
        Key_Pressed ← 0
end if
```

(initially Key_Pressed = 0)

Because we are sharing I/O lines between keypad and display devices, some of the bidirectional lines of PortD will periodically be configured for input (to read matrix keypad data) and then configured for output (to output data to the displays). The subroutine doing this on-the-fly reconfiguring is "Task3."

To save you some programming time here is the code listing for subroutine Task3 (based on the pseudocode shown previously):

```
;---------------------------------------------------------------------------

; Task 3 (executed every 1/20 sec) - Poll the matrix keypad:

Task3:
        BCLR    PortA,X,$60       ; PA6,PA5 <-- 0 (turn off both displays)
        BCLR    DDRD,X,$07        ; make PD2...PD0 inputs, to read keypad cols

        ; drive all keypad row lines high:
        BSET    PortA,X,$80       ; PA7 <-- 1
        BSET    PortD,X,$38       ; PD5...PD3 <-- 1

        ; read keypad column lines to detect if any key is pressed:
        BRCLR   PortD,X,$07,C0    ; (checking if PD2...PD0 are all zero)
        JMP     C1
C0:     CLR     Key_Pressed       ; no key is now pressed
        JMP     Quit_Task3

C1:     LDAA    #0                ; one of the keys is now pressed
        CMPA    Key_Pressed
        BEQ     C2
        JMP     Quit_Task3        ; a key was pressed last time, so do
                                  ;  nothing and wait for its release

C2:     LDAA    #1                ; a new keypress is detected
        STAA    Key_Pressed
        LDAA    #0
        CMPA    Mode
        BEQ     C3
        CLR     Mode              ; stop the countdown if running
        JMP     Quit_Task3

C3:     ; a new keypress is detected in paused mode

        ; check for key press in Row4 of the matrix keypad:
        BCLR    PortA,X,$80       ; PA7 <-- 0 (Row1)
        BCLR    PortD,X,$20       ; PD5 <-- 0 (Row2)
        BCLR    PortD,X,$10       ; PD4 <-- 0 (Row3)
        BSET    PortD,X,$08       ; PD3 <-- 1 (Row4)

        BRSET   PortD,X,$01,C4    ; jump to C4 if E is pressed
        BRSET   PortD,X,$02,C4    ; jump to C4 if F is pressed
        JMP     C5

C4:     ; E or F key is pressed, change mode to resume countdown:
        LDAA    #1
        STAA    Mode
        JMP     Quit_Task3
```

```
C5:     ; one of the numeric keys is pressed in paused mode; clear the
        ;  tenths-of-sec digit and shift in the numeric key value from
        ;  the right: (tens sec digit) <-- (sec digit) <-- (key value)

        CLR     Digit1
        LDAA    Digit2
        STAA    Digit3

        ; detect key0 press (check in Row4 still in effect from above)
        BRCLR   PortD,X,$04,C6
        LDAA    #0
        STAA    Digit2
        JMP     Quit_Task3

C6:     ; check for key press in Row1 of the matrix keypad:
        BSET    PortA,X,$80     ; PA7 <-- 1 (Row1)
        BCLR    PortD,X,$20     ; PD5 <-- 0 (Row2)
        BCLR    PortD,X,$10     ; PD4 <-- 0 (Row3)
        BCLR    PortD,X,$08     ; PD3 <-- 0 (Row4)

        ; detect key1 press:
        BRCLR   PortD,X,$04,C7
        LDAA    #1
        STAA    Digit2
        JMP     Quit_Task3

C7:     ; detect key2 press:
        BRCLR   PortD,X,$02,C8
        LDAA    #2
        STAA    Digit2
        JMP     Quit_Task3

C8:     ; detect key3 press:
        BRCLR   PortD,X,$01,C9
        LDAA    #3
        STAA    Digit2
        JMP     Quit_Task3

C9:     ; check for key press in Row2 of the matrix keypad:
        BCLR    PortA,X,$80     ; PA7 <-- 0 (Row1)
        BSET    PortD,X,$20     ; PD5 <-- 1 (Row2)
        BCLR    PortD,X,$10     ; PD4 <-- 0 (Row3)
        BCLR    PortD,X,$08     ; PD3 <-- 0 (Row4)

        ; detect key4 press:
        BRCLR   PortD,X,$04,C10
        LDAA    #4
        STAA    Digit2
        JMP     Quit_Task3

C10:    ; detect key5 press:
        BRCLR   PortD,X,$02,C11
        LDAA    #5
        STAA    Digit2
        JMP     Quit_Task3

C11:    ; detect key6 press:
        BRCLR   PortD,X,$01,C12
        LDAA    #6
        STAA    Digit2
        JMP     Quit_Task3
```

```
C12:     ; check for key press in Row3 of the matrix keypad:
         BCLR     PortA,X,$80       ; PA7 <-- 0 (Row1)
         BCLR     PortD,X,$20       ; PD5 <-- 0 (Row2)
         BSET     PortD,X,$10       ; PD4 <-- 1 (Row3)
         BCLR     PortD,X,$08       ; PD3 <-- 0 (Row4)

         ; detect key7 press:
         BRCLR    PortD,X,$04,C13
         LDAA     #7
         STAA     Digit2
         JMP      Quit_Task3

C13:     ; detect key8 press:
         BRCLR    PortD,X,$02,C14
         LDAA     #8
         STAA     Digit2
         JMP      Quit_Task3

C14:     ; detect key9 press:
         BRCLR    PortD,X,$01,Quit_Task3
         LDAA     #9
         STAA     Digit2


Quit_Task3:

         BSET     DDRD,X,$07        ; return PD2...PD0 to output mode

         ;Note: we had turned off both displays by clearing PA5 and PA6,
         ; but Task2 will refresh them in at most 1/200 sec so we need
         ; not do that here.

         ;increment TOC3 by 1/20 sec from its last value:

         LDD      TOC3,X            ; D <-- TOC3
         ADDD     #Incr3            ; D <-- D + Incr3
         STD      TOC3,X            ; TOC3 <-- D
         LDAA     #$20
         STAA     TFLG1,X           ; Clear the TCNT Output Compare 3 flag

         RTS

;---------------------------------------------------------------------------
```

You may copy and paste from an on-line listing of this subroutine that is found at: http://www.ece.uic.edu/~goncharo/ece367_exp6_Task3.txt

Your job is to write the rest of the code needed to implement this kitchen timer, build the circuit, and demonstrate its operation to your T.A.

There will be some flicker in the display. Can you explain why? How would you suggest to eliminate it?

Why were two output lines dedicated to the common cathode terminals of the seven segment displays, as compared to only one line in Experiment 5?