Application 6: External Multiplexed Display and Keypad Decoder

Purpose: To demonstrate and emulate the functions of a keypad and two digit LED display controller.

Goals:

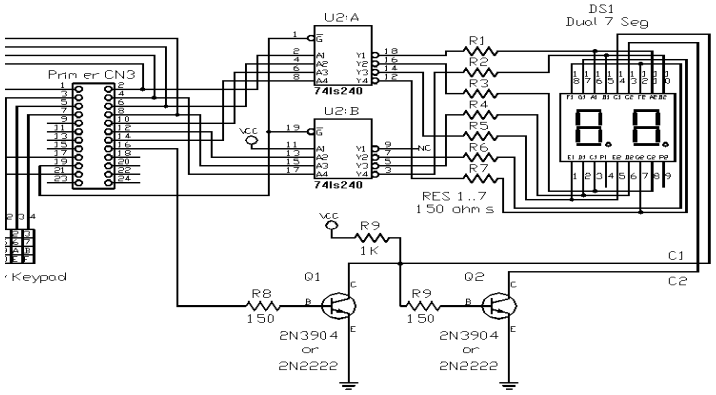
1. Build and test a keypad and numeric LED display interface.
2. Load a program that will demonstrate the numeric LED display interface.
3. Modify the program and load additional code which will demonstrate the keypad decoder.

Component Description	Digi-Key part number
2) 2N3904 or 2N2222	2N3904-ND or 2N2222-ND
1) 74ls240	DM74LS240N-ND
1) 4x4 matix keypad	GH5004-ND
1) 2 digit LED display	P355-ND
9) 150 ohm 5% 1/4 watt resister	
1) 1 Kohm 5% 1/4 watt resister	

This application will be demonstrated in two phases: with the display only, and then with the keypad and display.

Display Controller Circuit Description

To drive an external 7 segment display using the trainer, the 8 output lines (numbered 0 to 7) would be the obvious choice. This would provide control for each of the 7 elements leaving one output line free. What if we want to drive two digits?. We need 7 more outputs which we don't have. The answer to this problem is to use a multiplexed scheme of driving the digits.



We can drive the anodes of each of the elements of the pair of 7 segment displays with the same outputs (one output per matching pair of segments) and use the 8th (bit 7) to select which display will turn on by driving the cathode of the desired digit to ground. This will allow us to display data on the left digit and turn the right one off, and vice-versa. If this is done rapidly enough it will appear as if both digits are showing simultaneously, due to "persistence of vision" in the human eye.

To lessen the load on the output port, the outputs drive a 74LS240 tri-state inverting buffer and the outputs of this go to the anodes of both digits of the display. The buffer's two enable lines are tied to the Primer's digital to analog (D/A) output and they tri-state the outputs when the D/A is output is 5V. This turns off the display which will be necessary when including the keypad in the circuit. When the D/A output is 0V the buffer is enabled

and the outputs go to the opposite logic level as their respective inputs.

If the buffer is enabled, bit 7 selects which display to turn on. If bit 7 is high, the voltage applied to the base of Q1 will bring the cathode for the left display to ground, causing it to turn on. When this happens, the base of Q2 is pulled to ground causing it to turn off, which turns off the display on the right. When bit 7 is low, this turns off Q1 which allows the base voltage of Q2 to rise and turn on the display on the right.

```
;
; External Multiplexed Display and Keypad Decoder program.
;
OPORT       EQU   11H         ;OUTPUT PORT
IPORT       EQU   12H         ;INPUT PORT
MOS         EQU   1000H       ;MOS CALL ADDRESS
DACSRV      EQU   0EH         ;D/A SERVICE

          ORG     0FF01H

LOOP:     IN      IPORT       ;READ DIP SWITCHES
          MOV     B,A
          CALL    HEXOUT      ;DISPLAY B
          JMP     LOOP
;
; Display the hex value of B on the LEDs.  This routine must be
; called repeatedly in order for the data to be shown continuously,
; since it works on the principle of persistance of vision.  The right
; digit is turned on and off first, then the left digit is turned on and off.
;
HEXOUT:   MOV     A,B         ;GET VALUE
          ANI     0FH         ;MASK OFF UPPER NIBBLE
          CALL    BIN7SG      ;CHANGE TO 7 SEG VALUE
          OUT     OPORT       ;SEND TO PORT
          CALL    FLSHDG      ;TURN ON DISPLAY MOMENTARILY

          MOV     A,B         ;GET ORIGINAL VALUE
          ANI     0F0H        ;NOW MASK OFF LOWER NIBBLE
          RRC
          RRC
          RRC
          RRC
          CALL    BIN7SG      ;CHANGE TO 7 SEG VALUE
          ORI     80H         ;SET BIT 7 SO LEFT DIGIT IS DISPLAYED
          OUT     OPORT       ;SEND TO PORT
          CALL    FLSHDG      ;TURN ON DISPLAY MOMENTARILY
          RET
;
; Change the binary number in A to its 7 seg. output pattern.
;
BIN7SG:   PUSH    H
          PUSH    D
          LXI     D,TAB7SG    ;POINT TO START OF TABLE
          MVI     H,0
          MOV     L,A         ;HL = OFFSET INTO TABLE
          DAD     D           ;ADD TABLE ADDR TO OFFSET
          MOV     A,M         ;GET OUTPUT PATTERN
          POP     D
          POP     H
          RET
;
; TRANSLATE TABLE FOR LED OUTPUT
;
TAB7SG:   DB      40H,79H,24H,30H
          DB      19H,12H,02H,78H
          DB      00H,18H,08H,03H
          DB      46H,21H,06H,0EH
;
; This flashes on and off the digit selected by bit 7 sent to OPORT.
```

```
;
FLSHDG:   PUSH    D
          PUSH    PSW
          CALL    LEDON       ;ENABLE LEDS
          LXI     D,0FFH
DELAY1:   DCX     D
          MOV     A,D
          ORA     E
          JNZ     DELAY1
          CALL    LEDOFF      ;DISABLE LEDS
          POP     PSW
          POP     D
          RET
;
; LEDON, LEDOFF, TURN ON/OFF THE LEDS THROUGH THE D/A OUTPUT
;  5V OUT TRI-STATES THE OUTPUTS OF THE 74LS240
;  0V OUT ENABLES THE OUTPUTS OF THE 74LS240
;
LEDON:    MVI     E,0         ;SEND OUT 0V
          JMP     LEDCTL
LEDOFF:   MVI     E,0FFH      ;SEND OUT 5V
LEDCTL:   MVI     C,DACSRV    ;D/A SERVICE
          CALL    MOS
          RET
```

Display Controller Software Description

The program will be described from the lowest level subroutine to the main routine.

LEDON, LEDOFF
The subroutine LEDON turns on the selected display by sending 0V from the D/A into the 74LS240 enables and LEDOFF turns them off by sending 5V.

FLSHDG
This CALLs LEDON, goes into a delay loop and then CALLs LEDOFF. This causes the display selected by bit 7 to display for the period of time of the delay.

BIN7SG
This converts the number in the accumulator (A), which is in the range of 0 to F hex, to its corresponding binary pattern which will be used by another routine to illuminate the desired display segments. Since each element of a digit is controlled by bits 0 to 6 the bit pattern sent to the output port will form specific patterns. The table TAB7SG used by this routine has these bit patterns for digits 0 to F.

HEXOUT
This displays the hex value of the B register on the displays. This routine must be called repeatedly in order for the data to appear to be shown continuously, since it works on the principle of persistence of vision. The upper 4 bits of B are masked off leaving only the lower 4 bits which are converted to the appropriate binary pattern using BIN7SG and and this pattern is sent to the output port. Since the patterns received from BIN7SG always have bit 7 cleared, this will turn on the digit on the right when FLSHDG is called. To display the left digit, the lower 4 bits are masked off of B and the upper 4 are moved to the lower 4 bit positions. This value is converted using BIN7SG, bit 7 of the result is set to 1, and it is sent to the output port. This time when FLSHDG is called, the left digit will be displayed since bit 7 is set.

The main loop of this first example gets its input from the DIP switches, copies the value to B, CALLs HEXOUT and loops back to read the DIP switches again.

Using the Program

Build the circuit and then check your work. Now load the following program into memory and run it. With all the DIP switches in the ON position the port will input 00 and this should be shown on the displays. The binary value input to the DIP switches will be shown in hex on the displays (refer to the section at the beginning of this manual which discusses binary to hex conversion). Set the DIP switches so one digit is different than the other.

It appears that both digits are showing at the same time. To show what is really happening, we can increase the delay in FLSHDG so we can see what is really happening. Change the byte at FF4B from 00 to FF and run the program again. The displays can now be seen alternating left to right with each change in bit 7. Note that the PRIMER's digital output LEDs reflect the data sent to the output port (output bits of 0 turn on these LEDs). Watch the binary pattern on bits 6 to 0 as the digits change.

Move the DIP switches to the off position so that "FF" is displayed (this guarantees that none of the inputs are being pulled low), stop the program and change the byte at FF4B back to 00 again.

ADDRESS	DATA	DESCRIPTION	
FF01	D3	IN	12
FF02	12		
FF03	47	MOV	B,A
FF04	CD	CALL	FF0A
FF05	0A		
FF06	FF		
FF07	C3	JMP	FF01
FF08	01		
FF09	FF		
FF0A	78	MOV	A,B
FF0B	E6	ANI	0F
FF0C	0F		
FF0D	CD	CALL	FF27
FF0E	27		
FF0F	FF		
FF10	D3	OUT	11
FF11	11		
FF12	CD	CALL	FF44
FF13	44		
FF14	FF		
FF15	78	MOV	A,B
FF16	E6	ANI	F0
FF17	F0		
FF18	0F	RRC	
FF19	0F	RRC	
FF1A	0F	RRC	
FF1B	0F	RRC	
FF1C	CD	CALL	FF27
FF1D	27		
FF1E	FF		
FF1F	F6	ORI	80
FF20	80		
FF21	D3	OUT	11
FF22	11		
FF23	CD	CALL	FF44
FF24	44		
FF25	FF		
FF26	C9	RET	
FF27	E5	PUSH	H
FF28	D5	PUSH	D
FF29	11	LXI	D,FF34
FF2A	34		
FF2B	FF		
FF2C	26	MVI	H,00
FF2D	00		
FF2E	6F	MOV	L,A
FF2F	19	DAD	D
FF30	7E	MOV	A,M
FF31	D1	POP	D
FF32	E1	POP	H
FF33	C9	RET	

```
ADDRESS   DATA     DESCRIPTION
FF34        40     (PATTERN FOR "0")
FF35        79     (PATTERN FOR "1")
FF36        24     (PATTERN FOR "2")
FF37        30     (PATTERN FOR "3")
FF38        19     (PATTERN FOR "4")
FF39        12     (PATTERN FOR "5")
FF3A        02     (PATTERN FOR "6")
FF3B        78     (PATTERN FOR "7")
FF3C        00     (PATTERN FOR "8")
FF3D        18     (PATTERN FOR "9")
FF3E        08     (PATTERN FOR "A")
FF3F        03     (PATTERN FOR "B")
FF40        46     (PATTERN FOR "C")
FF41        21     (PATTERN FOR "D")
FF42        06     (PATTERN FOR "E")
FF43        0E     (PATTERN FOR "F")
FF44        D5     PUSH   D
FF45        F5     PUSH   PSW
FF46        CD     CALL   FF58
FF47        58
FF48        FF
FF49        11     LXI    D,00FF
FF4A        FF
FF4B        00
FF4C        1B     DCX    D
FF4D        7A     MOV    A,D
FF4E        B3     ORA    E
FF4F        C2     JNZ    FF4C
FF50        4C
FF51        FF
FF52        CD     CALL   FF5D
FF53        5D
FF54        FF
FF55        F1     POP    PSW
FF56        D1     POP    D
FF57        C9     RET
FF58        1E     MVI    E,00
FF59        00
FF5A        C3     JMP    FF5F
FF5B        5F
FF5C        FF
FF5D        1E     MVI    E,FF
FF5E        FF
FF5F        0E     MVI    C,0E
FF60        0E
FF61        CD     CALL   1000
FF62        00
FF63        10
FF64        C9     RET
```

Scanning the Keypad

To read a 4 by 4 matrix keypad we need 4 inputs and 4 outputs. The 4 inputs will check for a key pressed in one of the 4 columns in the current row selected by the 4 outputs. Since all of the outputs are currently being used, where do we get 4 more? We will use the same ones used for the displays but we will only use them while the displays are off (this is why we needed the circuitry to turn off both displays).

The subroutine KEYSCN (shown below), which will be added to the previous program, will be CALLed while the digits are off so that the changes in the output port will not be visible. When a key is pressed, the routine will modify the B register by shifting it left 4 bits and putting the binary value of the key into the lower 4 bits.

When KEYSCN is CALLed, output bits 0 to 3 are set to 0 to select all 4 rows

at once. When the input port is read and all of the lower 4 bits are 1, this indicates no key is pressed and the routine is exited without changing B. If any of the lower 4 bits are 0 this indicates a key has been pressed. The routine then selects 1 row at a time (by setting 1 of the output bits to 0 and the others to 1) until the input port reads a 0 on any of the lower 4 bits. When this happens, the row is found, and the column is found by finding which input port bit was 0. When the row and column is found it is translated to a value from 0 to F hex. The B register is shifted 4 bits to the left and this new value is put in the lower 4 bits and the routine exits.

There is another feature in KEYSCN which keeps a key that is being held closed from modifying the B register more than 1 time. When a key is pressed, the H register is loaded with a value which defines the minimum number of times KEYSCN must be CALLed while no key is pressed before it will recognize another key press. For example, when a key is pressed, B is modified by the new key value and H is loaded with 20 hex before exiting KEYSCN. On the next entry to KEYSCN the keypad will be examined to see if a key has been pressed and if one is pressed, H is not decremented and the routine is exited without changing B. If no keys are being pressed, H is decremented and the routine is exited without changing B. If no keys are pressed for 32 (20 hex) CALLs of KEYSCN then H will be 0 and any key pressed after this time will affect the B register, and again, H will be loaded with 20 hex.

```
;
; This routine checks for a key pressed and if there is one, register B
; is shifted left one nibble and the key value is put in the low nibble.
; The subsequent CALLs after a CALL that affected B, will not affect B
; again until no key has been pressed for 20 CALLs and then a key is
; pressed again.  This prevents a single key press from being
; interpreted as more than one.
;
; On entry and exit: H=debounce counter
;
DBOUNCE   EQU     20          ;NUMBER OF CALLs FOLLOWING A KEY PRESS
KEYSCN:   XRA     A           ;A=0
          OUT     OPORT       ;SELECT ALL 4 ROWS
          IN      IPORT       ;READ ALL 4 ROWS OF KEYPAD
          ANI     0FH         ;MASK OFF UPPER 4 BITS
          CPI     0FH         ;IF 0FH THEN NO KEYS PRESSED
          JNZ     KEYSC1      ;SKIP IF KEY READY

          ; NO KEY PRESSED, SO DEC. THE DEBOUNCE (IF>0) AND EXIT
          INR     H
          DCR     H           ;IS DEBOUNCE 0?
          RZ                  ;RETURN IF YES
          DCR     H           ;DEC ONCE MORE
          RET

KEYSC1:   INR     H
          DCR     H
          RNZ                 ;IF DEBOUNCE <> 0 EXIT

          ; SCAN FOR SPECIFIC ROW
          PUSH    D
          MVI     E,01111111B ;ROW SCAN VALUE (WILL BE ROTATED)
          MVI     D,-4        ;ROW ADDER (+4=0)

KEYSC2:   MOV     A,E         ;GET ROW SCAN VALUE
          RLC                 ;ROTATE IT
          OUT     OPORT       ;SEND ROW SCAN TO OUTPUT PORT
          MOV     E,A         ;SAVE BACK NEW ROW SCAN

          MOV     A,D         ;GET ROW ADDER
          ADI     4           ;INC ROW ADDER BY 4
          MOV     D,A         ;SAVE IT
```

```
          IN      IPORT       ;SEE IF THIS ROW HAS CHAR READY
          ANI     0FH         ;MASK OFF UPPER
          CPI     0FH
          JZ      KEYSC2      ;LOOP TILL <> 0FH

          ; FIND WHAT COL. IT'S IN
          MVI     L,0FFH      ;SET SO INR WILL MAKE 0
KEYPD1:   INR     L
          RRC
          JC      KEYPD1      ;LOOP TILL NO CY
          ; NOW ADD COL. TO ROW ADDER
          MOV     A,D         ;GET ROW ADDER
          ADD     L
          MOV     L,A         ;L IS THE KEY PRESSED (0 TO F HEX)
          ; SHIFT B LEFT 1 NIBBLE AND PUT L IN LOWER NIBBLE
          MOV     A,B         ;SHIFT B
          ADD     A
          ADD     A
          ADD     A
          ADD     A           ;THIS SHIFTS LEFT PADDING 0's
          ADD     L           ;PUT L IN LOWER NIBBLE
          MOV     B,A         ;NEW B REG

          MVI     H,DBOUNCE   ;DEBOUNCE VAL. (NO KEYS ACCEPTED TILL 0)
          POP     D
          RET
```

Using the Program

The previous program will be modified slightly (assuming it is still in memory) by putting CALL KEYSCN in the program in place of IN IPORT, MOV B,A and a new subroutine will be added at the end. (Pay close attention to the addresses when entering the following program, since there is a skip in sequence of the addresses after the first three.) When you run the program you should see the key you press on the right display and the digit that was there before, moved to the left display. As you have just seen demonstrated in this application, multiplexing allows you to greatly extend the capabilities of an output port.

ADDRESS	DATA	DESCRIPTION
FF01	CD	CALL FF65
FF02	65	
FF03	FF	
:	:	
:	:	
FF65	AF	XRA A
FF66	D3	OUT 11
FF67	11	
FF68	DB	IN 12
FF69	12	
FF6A	E6	ANI 0F
FF6B	0F	
FF6C	FE	CPI 0F
FF6D	0F	
FF6E	C2	JNZ FF76
FF6F	76	
FF70	FF	
FF71	24	INR H
FF72	25	DCR H
FF73	C8	RZ
FF74	25	DCR H
FF75	C9	RET
FF76	24	INR H
FF77	25	DCR H
FF78	C0	RNZ
FF79	D5	PUSH D
FF7A	1E	MVI E,7F
FF7B	7F	
FF7C	16	MVI D,FC
FF7D	FC	
FF7E	7B	MOV A,E
FF7F	07	RLC
FF80	D3	OUT 11
FF81	11	
FF82	5F	MOV E,A
FF83	7A	MOV A,D

ADDRESS	DATA	DESCRIPTION
FF84	C6	ADI 04
FF85	04	
FF86	57	MOV D,A
FF87	DB	IN 12
FF88	12	
FF89	E6	ANI 0F
FF8A	0F	
FF8B	FE	CPI 0F
FF8C	0F	
FF8D	CA	JZ FF7E
FF8E	7E	
FF8F	FF	
FF90	2E	MVI L,FF
FF91	FF	
FF92	2C	INR L
FF93	0F	RRC
FF94	DA	JC FF92
FF95	92	

```
FF96        FF
FF97        7A    MOV     A,D
FF98        85    ADD     L
FF99        6F    MOV     L,A
FF9A        78    MOV     A,B
FF9B        87    ADD     A
FF9C        87    ADD     A
FF9D        87    ADD     A
FF9E        87    ADD     A
FF9F        85    ADD     L
FFA0        47    MOV     B,A
FFA1        26    MVI     H,14
FFA2        14
FFA3        D1    POP     D
FFA4        C9    RET
```