A Keypad Controller for Bi-directional Key Matrix

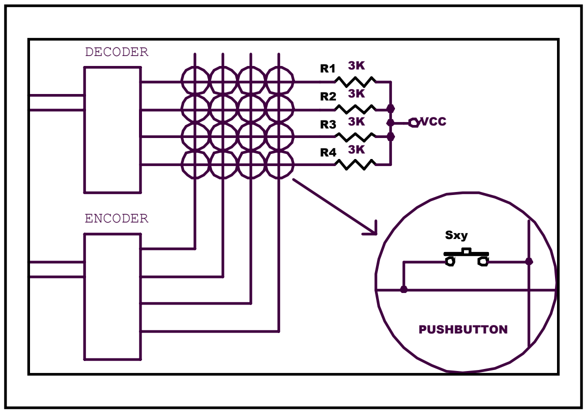
Author: Vladimir Velchev
AVEX - Vladimir Velchev
Sofia, Bulgaria
email:avex@iname.com

APPLICATION OPERATION:

The PIC microcontroller can replace the traditional decoders and encoders that are used for old-fashion keyboard controllers. But it can replace even more—the traditional keypad controllers, based on many types of new chips. It's possible due to this new idea that uses a special type of key matrix. I named it a "bi-directional key matrix." For better understanding of how it works and to see its advantages, let's take a look of the evolution of keyboard controllers.

Figure 1 shows the classic key matrix, which uses one decoder for output lines and one encoder for input lines. These components have strictly determined pins for inputs and outputs.

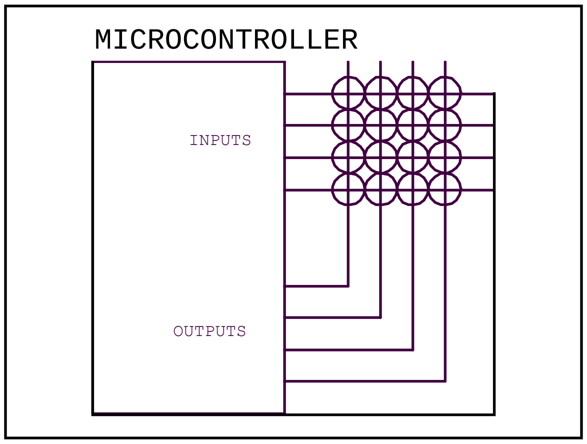
FIGURE 1: CLASSIC KEY MATRIX



NEXT GENERATION

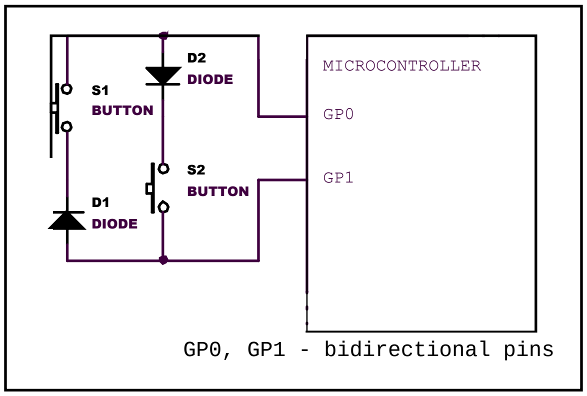
Figure 2 shows how to connect the classic key matrix to any microcontroller. It must be programmed to have some input and output pins. In this case, the encoder and the decoder are simulated by the software. But in fact, during the scanning of the matrix, all pins still are strictly determined as inputs or outputs and have one direction.

FIGURE 2: CONNECTING CLASSIC KEY MATRIX TO ANY MICROCONTROLLER



The new idea for a different key matrix circuit is shown in Figure 3:

FIGURE 3: A DIFFERENT KEY MATRIX CIRCUIT



This is an illustration of 2-button keypad. To scan this simple key matrix, the microcontroller must perform two basic steps:

1. Set GP0 pin as output and write to it logic ‘0’.Set GP1 pin as input and check its state. If GP1=0 then button S1 has been pressed. The state of button S2 does not affect the input GP1, because of diode D2.
2. Set GP1 pin as output and write to it logic ‘0’. Set GP0 pin as input and check its state. If GP0=0 then button S2 has been pressed. The state of button S1 does not affect the input GP0, because of diode D1.

Inputs GP0 and GP1 must be configured with internal pull-up resistors.

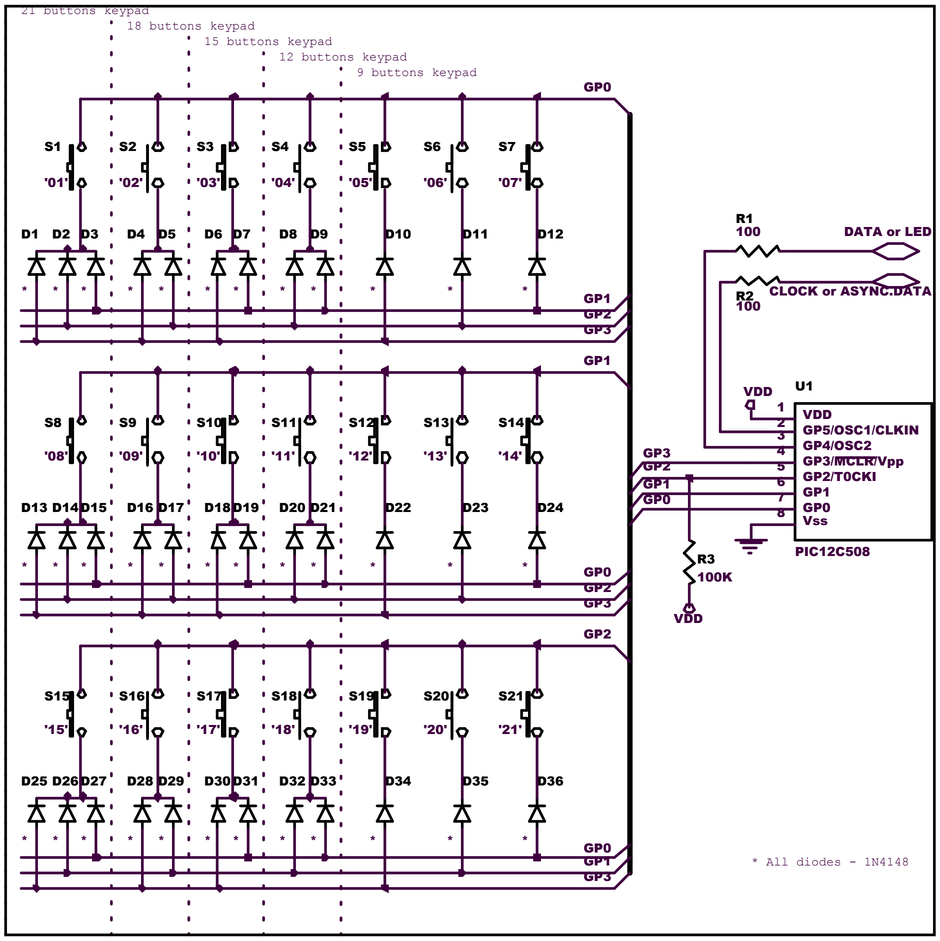
As we can see, the pins have a bi-directional working cycle.

Figure 4 shows the complete design of keypad controller with bi-directional key matrix. It uses 4 pins (GP0-GP3) for bi-directional key matrix and GP4,5 as communication inputs/outputs. GP3 is always input, so the scanning cycle will have 3 basic steps:

1. Set GP0 pin as output and write to it logic ‘0’. Set other pins as inputs and read their states. For inputs GP1-GP3, it’s possible to have 7 combinations of codes (keys). - 000_111. The combination 111 means that no key has been pressed.
2. Set GP1 pin as output and write to it logic ‘0’. Set other pins as inputs and read the code of the pressed button if any.
3. Set GP2 pin as output and write to it logic ‘0’. Set other pins as inputs and read the code of the pressed button if any.

Figure 4 shows a circuit of a fully combined bi-directional key matrix with three bi-directional and one input pin (max. 21 buttons). Many applications need 12 – 16 buttons and for these cases it’s suitable to remove the buttons connected to 2 and 3 diodes. This will reduce the number of used diodes. A cost efficient keypad using the PIC12C508 can be built with 9 buttons or 12 buttons (if GP3 pin is changed with GP4).

FIGURE 4: KEYPAD CONTROLLER



APPLICATIONS

- Small keypads for standard IBM PC based appliances;
- Keypads for telephone industry;
- Access control keypads (GP4 can be connected to two LEDs: red for "access denied" state and green for "access granted" state;

AUTHOR'S NOTE:

Table 1, shown below, makes a comparison between the bi-directional and the classic key matrix for a different number of pins used. This table refers to a short bi-directional key matrix with only one diode per button and all the pins are bi-directional.

\

TABLE 1: BI-DIRECTIONAL AND CLASSIC KEY MATRIX COMPARISON CHART

Number of Pins Used	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Max. Number of Buttons /Classic XY Matrix/	2	3	4	6	9	12	16	20	25	30	36	42	49	56	64
Max. Number of Buttons /Short Bi-directional Matrix/	2	6	12	20	30	42	56	72	90	110	132	156	182	210	240

MICROCHIP DEVELOPMENT TOOLS USED

Assembler/Compiler version:

MPLAB 3.22, MPASM 1.5

FLOW CHART

START
Initial setup
CALL READ_KEY
USER:
Transmitting the code of the pressed key
or do something else
READ_KEY
Set GP0 as output, GP1,2,3 as inputs. GP0=0.
Read GP1,2,3 inputs.
Call table of key codes.
Key
1 - 7
pressed?
Set GP1 as output, GP0,2,3 as inputs. GP1=0.
Read GP0,2,3 inputs.
Call table of key codes.
Key
8 - 14
pressed?
Set GP2 as output, GP0,1,3 as inputs. GP2=0.
Read GP0,1,3 inputs.
Call table of key codes.
Return key code
Key
15 - 21
pressed?
Return 0
Y
Y
Y
N
N
N

APPENDIX A: SOURCE CODE

Note: This program is not tested with real PIC12C508. All the experiments were done with PIC16C84 (PICSTART-16B1 programmer).

```
;************************************************************************
; Using PIC12CXXX as keyboard controller for bi-directional key matrix
; Written by Vladimir Velchev 08.1997.
; (C) AVEX - Vladimir Velchev
; Version 1.00
;************************************************************************

; Osc.: F=4MHz (internal)
; GP0 - input/output 0 for key matrix
; GP1 - input/output 1 for key matrix
; GP2 - input/output 2 for key matrix
; GP3 - input 3 for key matrix
; GP4 - input/output (may be DATA or LED)
; GP5 - input/output (may be CLOCK or async.output)

             LIST    P=12C508

#include <p12C508.inc>

;*** Equates
GP0_Pin      equu     0                              ;input/output 0 for key matrix
GP1_Pin      equ      1                              ;input/output 1 for key matrix
GP2_Pin      equ      2                              ;input/output 2 for key matrix
GP3_Pin      equ      3                              ;input 3 for key matrix
GP0_MASK     equ      B'00000001'    ;bit mask for GP0
GP1_MASK     equ      B'00000010'    ;bit mask for GP1
GP2_MASK     equ      B'00000100'    ;bit mask for GP2
GP3_MASK     equ      B'00001000'    ;bit mask for GP3
IOSET                 equ            B'00111111'      ;initial I/O port settings - all inputs

;*** RAM locations
KEY                   equ            H'07'            ;code of pressed key or 0 if no pressed

;*** Vectors
                      org            0                ;RESET vector
                      goto           MAIN

;*** Table of key codes (3x7=21 posible codes)
KEY_TABLE    addwf    PCL,1                           ;W- offset of table
                      retlw          D'1'             ;Codes of keys (can be 1...255)
                      retlw          D'2'
                      retlw          D'3'
                      retlw          D'4'
                      retlw          D'5'
                      retlw          D'6'
                      retlw          D'7'
                      retlw          0                ;0= no key pressed
                      retlw          D'8'
                      retlw          D'9'
                      retlw          D'10'
                      retlw          D'11'
                      retlw          D'12'
                      retlw          D'13'
                      retlw          D'14'
                      retlw          0                ;0= no key pressed
                      retlw          D'15'
                      retlw          D'16'
                      retlw          D'17'
                      retlw          D'18'
                      retlw          D'19'
                      retlw          D'20'
                      retlw          D'21'
                      retlw          0                ;0= no key pressed

;*** Code Starting Point
MAIN:
; Initial setup
                      movlw          IOSET            ;init GPIO
                      tris           GPIO
                      clrf           GPIO             ;write 0 to output latches
                      movlw          H'80'            ;init option register
                      option                          ;enable pull-ups (GP0,1,3)
Main_Loop:
                      clrwdt                          ;clear watchdog timer
                      call           READ_KEY         ;call subroutine

;Space for User code
;                     ...
;                     ...

                      goto           Main_Loop        ;go to beginning

;*** Subroutine - READ_KEY
; Input :
; Output: KEY- code of pressed key (KEY=0, ZF=ZY - if no key was pressed)
READ_KEY:
;Read keys when GP0 is set as output, GP1,2,3- inputs
                      movlw          IOSET&(~GP0_MASK) ;set GP0 as output
                      tris           GPIO
                      rrf            GPIO,W           ;read port & remove GP0 bit
                      andlw          H'07'            ;keep low 3 bits
                      call           KEY_TABLE        ;read code of key
                      iorlw          0                ;check code
                      btfss          STATUS,Z         ;skip if no key pressed
                      goto           READ_KEY_END
;Read keys when GP1 is set as output, GP0,2,3- inputs
                      movlw          (IOSET|GP0_MASK)&(~GP1_MASK) ;reset GP0 as input
                      tris           GPIO             ;set GP1 as output
                      rrf            GPIO,W           ;read port & move GP0 bit to C flag
                      andlw          H'06'            ;keep GP2,1 & clear GP0
                      btfsc          STATUS,C         ;skip if CF=0 (GP0 was 0)
                      iorlw          H'01'            ;else- set GP0=1
                      iorlw          H'08'            ;add offset of second part of table (+8)
                      call           KEY_TABLE        ;read code of key
                      iorlw          0                ;check code
                      btfss          STATUS,Z         ;skip if no key pressed
                      goto           READ_KEY_END
;Read keys when GP2 is set as output, GP0,1,3- inputs
                      movlw          (IOSET|GP1_MASK)&(~GP2_MASK) ;reset GP1 as input
                      tris           GPIO             ;set GP2 as output
                      movlw          H'0B'            ;W- mask for GP3,1,0
                      andwf          GPIO,W           ;read port and remove bits GP5,4,2
                      btfsc          GPIO,GP3_Pin     ;skip if bit GP3 is 0
                      iorlw          H'04'            ;copy bit GP3 to GP2 if GP3=1
                      andlw          H'07'            ;keep GP2-GP0 bits
                      iorlw          H'10'            ;add offset of third part of table (+16)
                      call           KEY_TABLE        ;read code of key
                      iorlw          0                ;set flags (ZF)
READ_KEY_END:
                      movwf          KEY              ;save the code of key
                      movlw          IOSET            ;reset GP0...GP3 (as inputs)
                      tris           GPIO
                      return

                      end                             ;end of program
```