Keypad Interfacing

1 Introduction

The Pt-51 board has a 4 way DIP switch to provide inputs to the microcontroller. But if more inputs are required to be provided, we connect a keypad/keyboard to it. In order to connect a keypad/keyboard to the microcontroller, we need some external circuitry. However, the AT89C5131 has an on-chip keyboard interface (highlighted in the Fig. 1) to connect as many as 8xn switches to the microcontroller.

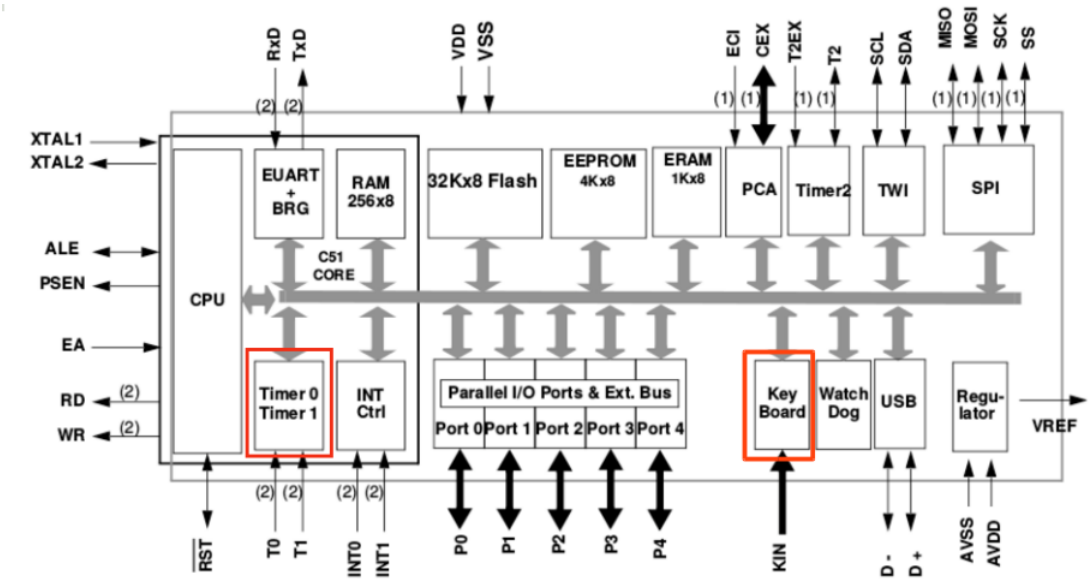


Figure 1: Block Diagram of AT89C5131

The main objective of this experiment is to

- Learn the detection of a "key press" and "identification" of the key pressed in a 4x4 matrix.
- Use "keyboard interface" to connect a hex(4x4) keypad to Port1 of the micro-controller. Port 1 can be used as standard I/O or it forms the 8 input lines of keyboard input interface of AT89C5131.
- Write an interrupt service routine (ISR) when a main program is interrupted by an external interrupt.

1.1 Background

The keypads are organized in a matrix of rows and columns. The micro-controller accesses both rows and columns through ports; therefore a hex keypad can be connected to one of the 8-bit ports of a micro-controller as shown in the Fig 2.

When a key is pressed, a row and a column make a contact; otherwise there is no connection between rows and columns. There are 2 ways of key press detection: (1) the interrupt method, and (2) the scanning method or polling method.

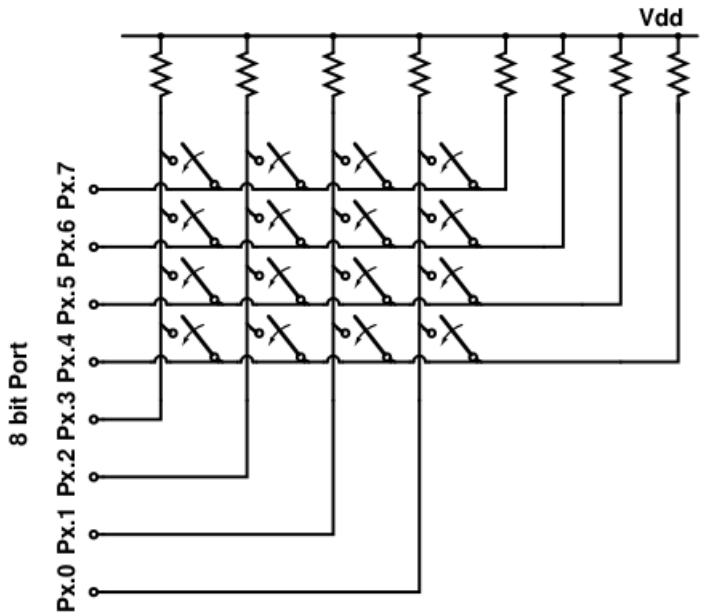


Figure 2: Keypad Interface

1.1.1 Polling Method

Since the keypad itself can not send any notification, microcontroller has to continuously keep checking if any key is pressed. To check if any key is pressed, the microcontroller grounds all columns by writing 0 to the output latches (port pins Px.0 to Px.3 in Fig. 2 connected to the columns, then it reads the rows. If the data read from the rows is 1111, no key has been pressed and the process continues until a key press is detected. After a key press is detected, the microcontroller will go through the process of identifying the key. This results in keeping the microcontroller busy doing nothing else but keep checking if a key is pressed and then identify the key pressed and hence, the method is not efficient.

1.1.2 Interrupt Method

In this method, the microcontroller can execute its main program and will be interrupted only when a key is pressed. The microcontroller will then execute the ISR. The ISR will ensure that the first key press detection was not erroneous and wait 10ms for debouncing the key. The ISR will identify the key pressed and then return to the main program. This allows the microcontroller to execute other tasks as well.
We shall use this approach for keypad interfacing.

2 Detection of key press using external interrupt

Refer Fig 2. All port lines are pulled up to Vdd by resistors. The value of resistors is sufficiently high such that these provide just a weak pull up. If a line to which a '1' is written is shorted with a line to which a '0' is written, the resultant value will a '0' because the pull up is weak.

Associated with each row-column combination is a key, which is just an inexpensive single pole switch. The switch will short its column to its row when pressed. Assume that all column bits have been put in input mode by writing '1's to them. If a '0' is output to a row line and a switch in this row is closed, the corresponding column line will go to '0' while all other columns will remain at '1'. Thus, we can identify the column of the pressed key by writing '0's to *all* the rows, '1's to *all* the columns and reading the port. Similarly, if we write '0' to all the columns, '1's to all the rows and

read the port, we could identify the row.

In fact we can do better. We write '0000' to the row lines, and '1111' to the column lines. We now read the port and examine the nibble corresponding to the columns. If any switch is closed, the corresponding bit in its column will be '0'. We now write back *this same nibble* to the columns, and '1's to all rows. Since the column in which the key is situated would have been read as '0', it will now be driven to '0'. Because of the shorting provided by the switch, the corresponding row will also go to '0'. We can now identify the row by checking which row bit is '0'.

3 Debounce Check

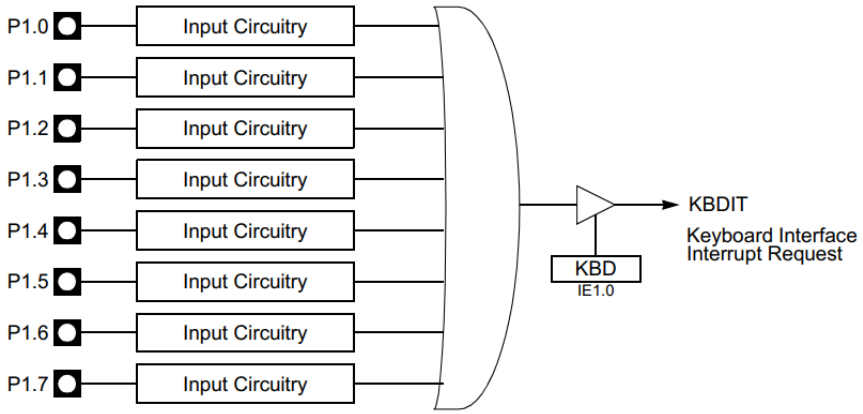
We assume that any change of state of a key which occurs faster than (say) about 20 milliseconds is due to bouncing and must be rejected. Therefore, we read the keyboard twice with an interval of 10 milliseconds. We consider a key pressed or released only if we find it in the same state on two consecutive readings.

4 Algorithm

1. Check if the previously pressed key is released.
2. Once a new key press is detected, go to the keypad interrupt ISR (from 003BH).
3. Wait for 10ms (you can use software delay here) and after that check if Key is still pressed or not. If no key press is detected, return from the ISR.
4. If a key press is detected, figure out corresponding row and column.
5. Report the key pressed (display on LCD and return from the ISR.

5 Keypad Interface of AT89C5131

8051 family micro controllers have only two external interrupts. So for interrupt driven keypad interfacing, we need to connect the four column bits to a 4 input AND gate and the output of this AND gate may be connected to $\overline{INT0}$ or $\overline{INT1}$ so that whenever any key is pressed, one of the columns goes low and hence output of the AND gate goes low making the $\overline{INT0}$ or $\overline{INT1}$ pin go low. So, instead of connecting an additional 4-input AND gate (in our case) we will use the on-chip Keyboard Interface, which fulfills the same purpose.
The Keypad Interface is based on 8 inputs with programmable interrupt capability on both high or low level. It communicates with the microprocessor through 3 special function registers: KBLS (Keyboard Level Selection), KBE (Keyboard interrupt Enable) and KBF (Keyboard Flag register).



6 Homework

1. Study how to detect a row(R), given in the book of Mazidi ('Keyboard Interfacing' subsection of 'LCD and Keyboard' section).
2. Read the section on Keyboard interface from the datasheet of AT89C5131 on page no. 88.
3. From that, figure out how to detect the column(C) of a particular Row.
4. 4*R+C will also give you the index of the key pressed.

7 Labwork

1. First write the main program. The main program will display 'A','B','C'..... one by one on the LCD, at an interval of 100ms. After displaying 16 characters, let the display repeat itself from 'A' infinitely on the same line.
2. Let this program be interrupted by pressing one of the keys. When the main program encounters an interrupt following steps should be followed:
   (a) Context saving: Push the registers used in the main program on to the stack if required to be used by the ISR.
   (b) Give a debouncing duration of 10ms to check if key is actually pressed or not.
   (c) Detect the key pressed.
   (d) When a valid key press is detected, and thereafter the pressed key is identified, display "The key pressed is XX" on the LCD for 1sec and then resume from the main program from where it was interrupted. Suppose 'ABCDEF' was on the LCD before you pressed a key (let us assume 5), it will clear the screen, display 'Key 5 is pressed' starting from the leftmost position of the 1st line and resume displaying from 'G' exactly where 'G' is supposed to be displayed. The characters before 'G' will not be displayed.
   (e) **Point to note**: While getting out of the ISR, you need to do 2 things. Firstly, ensure that the key that was being pressed has been released. Secondly, after releasing the key you have to reset the KBF register (009EH). It can be cleared by software only. However, it does not get cleared immediately when you move 00H to KBF. So you have to write a loop which will check if the value of KBF has actually been reset to 00H or not. Pop the saved registers if any and return from ISR and resume the main program.