# Cheap Micro Keyboard

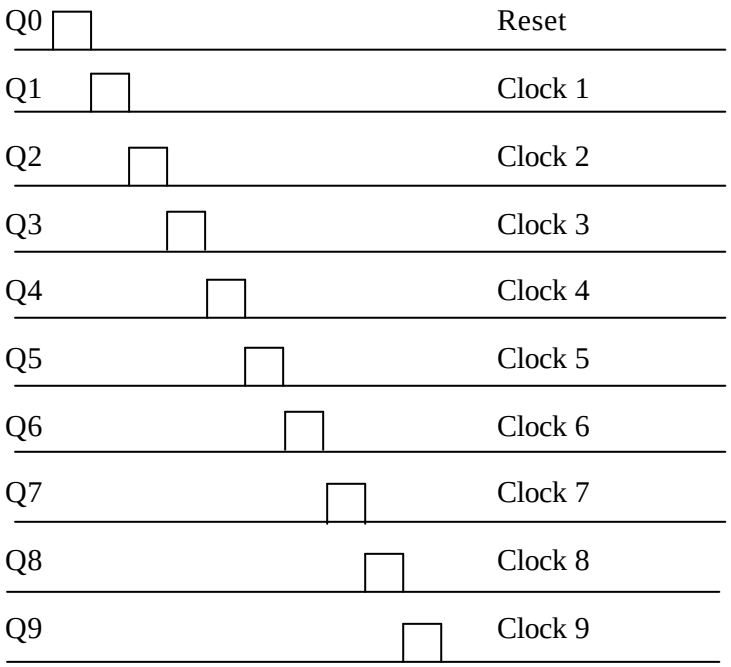
Easily adapts to any micro

© Dean Cocks
Tuesday, January 02, 2001
DeanoC@camtech.net.au

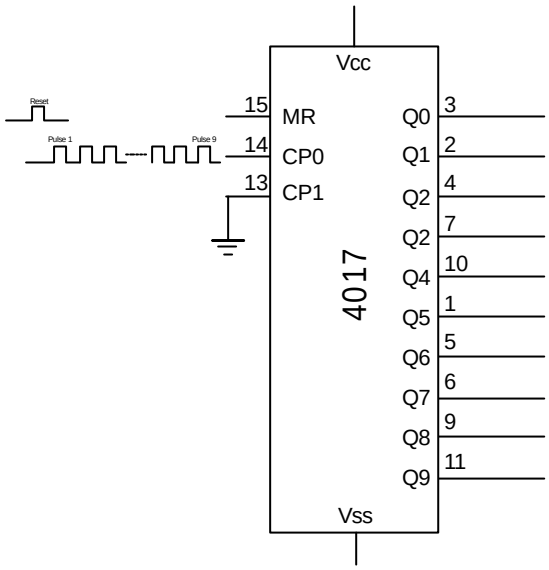
*Please read the static awareness section at the end of this document.*

The following article describes a very cheap keyboard interface which can be easily adapted to virtually any micro. It uses just on IC that costs around $1.70 AUD.
The software description given will be for a PIC 16F84 but as mentioned, can be adapted to any micro.
The software was written in Parallax mnemonics using CVAsm16 (from the net) I'm sure that parallax PASMX will work with it but I have not tried this. CVASM does the 16F877 were the PASMX did not at the time of writing the code.

**Hardware.**
As previously mentioned, the hardware consists of one IC called a 5 stage Johnson decade counter. What the IC actually contains is a counter with a reset and 2 count pins as inputs, and 11 outputs. (10 outputs and a carry pin)
When the **reset** pin (MR) is given a high pulse (5v), the outputs Q1-Q9 are made low and the output Q0 is made high.
When a **clock** pulse is given on CP0 while CP1 is being held low, the high which was on output Q0 now moves along to Q1 (making Q0 low). Continuing to pulse the clock input (CP0), advances the high along the counter until 9 pulses have been sent. On the $10^{th}$ pulse, the counter (high) output returns to Q0 and a pulse is given on the carry output.



**The 4017**




**4017 Truth Table**

| MR | CP0 | CP1 | OPERATION |
|---|---|---|---|
| H | X | X | Q0 HIGH Q1-9 LOW |
| L | H | H - L | Counter advances |
| L | L - H | L | Counter advances |
| L | L | X | No change |
| L | X | H | No change |
| L | H | L - H | No change |
| L | H - L | L | No change |

H= High (+5v)
L= low (0v)
X= Don't care

From the truth table above, you can see that if MR (reset) is taken high, regardless of all else, the chip is reset to output Q0 (high). You can also see that if CP1 is permanently held low, all that is required to clock the IC, is to pulse the CP0 line with a high pulse.

Now, knowing how the IC works, we can start to work out a circuit diagram.

First, we know that only one output of the IC is high at any one time and all the rest are low. If we just connected the keys (push to close switch) to the outputs, and someone were to press two keys at once, there would be a short circuit between the two outputs. Now while this wouldn't matter if the outputs were low, but it does matter if it's a low output and the only high output. See fig 1 This would cause high current consumption & possible the destruction of an output (maybe the chip). To get around this problem, resistors are added in series with each output. Now, if and two switches are pressed, yes there will be a short, but the current has been reduced to a trickle buy the resistors. 2 (I=V/R I=5/2000R=2.5mA. Very safe) See Fig

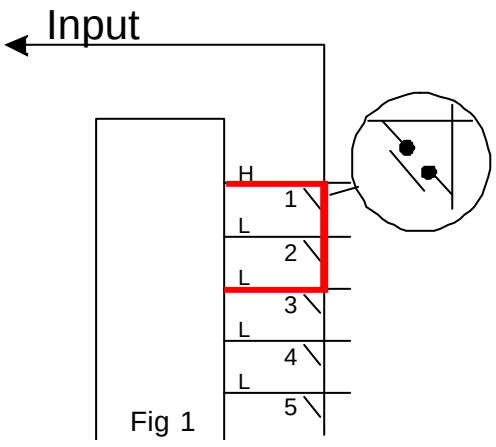


Fig 1

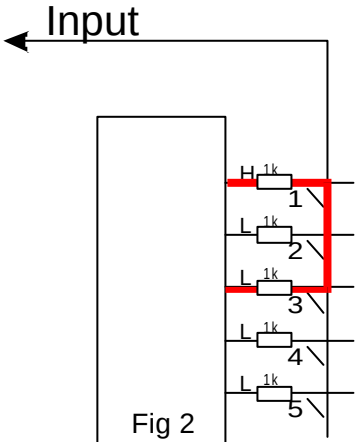
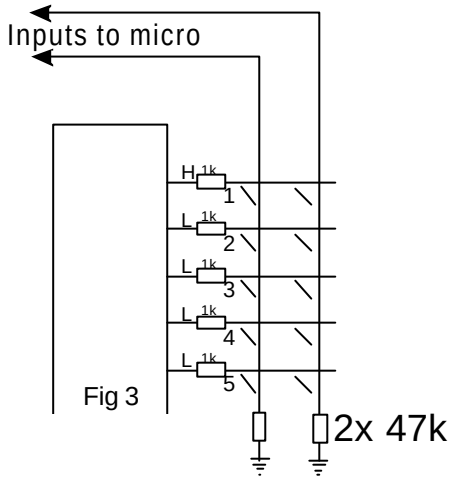


Fig 2

So now all the outputs are protected against destruction, lets look at any inputs and what state they should be in. First is the Reset line (MR). Lets assume that the micro that you are going to connect the keyboard to, sets all I/O inputs on power up (tri-state, which is normal0. This would leave the input of the 4017 IC floating. Now in the CMOS rules state that this is a no-no as the IC could oscillate and self destruct, To fix this, a resistor is placed from

the MR input to 0 volt. Now, when the micro goes into tri-state (power up), the MR line is pulled low and the IC is reset ready for the first read. On initialization, the micro should turn the input o an output and take it Low.
The same applies to the clock input (CP0), it also is pulled low but it requires that it be pulsed high to count, but more on that latter.

**Keyboard Inputs**
We now require some inputs from the 4017 to the micro as lines which are read to see if a switch has been pressed. This happens in groups of 10 switches (although not all 10 have to be used). As can be seen in fig 1 & 2, an input line is connected as a common to a row of switches ( One switch per 4017 output up to a max of 10 switches.) Now if one of the 4017's outputs is high (remember only one can) and the switch on that output is pressed, the high from the 4017 output is connected to the input line and straight into the micro. This is an input line to the micro, and as such it will float to an undetermined state which could be high. If we were to leave the input floating we would more than likely get wrong readings. So, knowing that we are going to look for high signals from the 4017 outputs, the logical state for the inputs lines would normally be in is low. It is for this reason that each input we add, we must use a pull down resistor. The value is not important as long as it's not too high or too low, so I usually use 47k. See fig 3




Now it's taking shape! We have two inputs (20 keys max in this version) and two outputs from the micro to the 4017 (MR & Clock)

So, at this point we can reset the 4017 to zero, clock the 4017 to a specified output and read a key connected to that output. So now it's

**Software time**
Lets say we just powered up the micro, did the Init which set the appropriate putouts & inputs to suit the 4017 connections and gave the MR line a high pulse (can be very short, in the order of 50nS Clock is 85nS ) to set the outputs to Q0 (high) and no we got the micro to look at the input line 1. If a key were pressed on Q0 output, the input to the micro would read as a one. But if the key were on a different output, lets say Q3, then we would pulse the clock line 3 times with a read after each pulse before we found the key Q3. So now we know that the line read, was number 1 and the output was number 3. There are lots of ways that you can use this information like making the two numbers become a key code such a Line 1, Output 3, Move the 1 from the low nibble to the high nibble and 'OR' the clock (count) to the previous line number to make a key code 13 (line 1, key 3).
I like the use the line 1 input as numbers, as you can forget the line number and use the clock pulse count as the key pressed number. So 0 is a zero output 1 is a 1, output 2 is a 2 etc. right through to 9. The second input line I 'OR' "F0" with the clock count, giving me Function key codes of F0 – F9 (this is shown in the sample code.)

When the routine scans the keyboard, it looks for the first occurrence of a key and exits the routine with the keycode in the variable Got_Key. You could look for two keys by reading both inputs at once and looking for a

specific number. But that means that the two keys pressed would have to be on the same ‘Q’ output of the 4017 and on different line inputs to the micro. See fig 4.

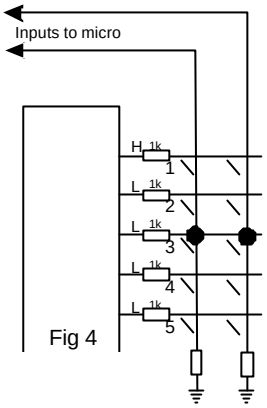


Fig 4

```
; Target Controller -      PIC16F84
;                                  _____________
;                  ------RA2 |1            18| RA1---------
;                   ------RA3 |2            17| RA0---------
;                  ------RA4 |3            16| OSC1--------XTAL
;          +5V- ------MCLR |4            15| OSC2--------XTAL
;        Ground----------Vss |5            14| VDD---------+5 V
;        Keys1-----------RB0 |6            13| RB7---------
;        Keys2-----------RB1 |7            12| RB6---------
;       Key_Clk--- -----RB2 |8            11| RB5---------
;       Key_Clr--- -----RB3 |9            10| RB4---------Serial LCD
;                            -----------------
;
; Assembler - CVASM16
; Author - El cheapo Keyboard by Dean Cocks
;
; Modification History
; Initial Version          30/9/99          keyboard scan routine

;*****************************************************************************
;Assign names to IO pins

KEYBOARD_LINE1          equ     Portb.0         ;
KEYBOARD_LINE2          equ     PortB.1         ;
KEYBOARD_CLOCK          equ     PortB.2         ;
KEYBOARD_CLEAR          equ     PortB.3         ;

Counter                 ds      1               ;used in Keyboard input
Got_Key                 ds      1
```

```
;*************************************************************************
;           Keyboard input routine
;Clears the 4017 to start at Q0 key and checks straight away
;Clocks the 4017 to advance to next 'Q' key,
;checks the inputs again
;repeats this for each output of the 4017
; if a key was pressed, it determines which line, and if the line was
;line 2 then it must have been a function key so an "F" is added  for "Function"
;If it was line 1 then just the counter is used as the count number.
; If no key was pressed then "FF" is returned.
;*************************************************************************
Getkey          Setb    Keyboard_Clear          ;Clear The Counter
                Clrb    Keyboard_Clear
                Clr     Counter
Do_Next_Key     Jb      Keyboard_Line1,Exitkey
                Jb      Keyboard_Line2,Got_Line2
                Inc     Counter
                Setb    Keyboard_Clock          ;Clock To The Next Output
                Clrb    Keyboard_Clock
                Cjb     Counter,#9,Do_Next_Key
                Mov     Counter,#0ffh           ;No Key Press Found
                Jmp     Exitkey
Got_Line2       Or      Counter,#0F0h           ;Add F0h For Function F0,1,2,3,4,5,6,7,8,9
Exitkey         Mov     Got_Key,Counter         ;Put Key Into Var
                Return
```

**Well that's it, I have it working on a digital synthesizer, the code above is directly from the source, and it hasn't missed a beat.**

**Static Awareness**

For those persons not familiar with CMOS technology, the 4000 series of IC's are very static sensitive, so take a few small precautions so as not to kill your IC. The following could save hours of trying to debug a dead IC. These precautions should be adhered to for any electronics project.

1) Do not wear nylon type clothing when handling the IC.
2) Do not assemble your project in a carpeted room unless the carpet is anti static.
3) Try not to touch the pins until connected in to the circuit.
4) If possible connect an earth wire from mains ground to your project GND (0V) via a 1meg resistor.
   and place a part of your bare skin on the project 0V to reduce the potential difference (hopefully to zero)
5) If soldering the IC in to the PCB, solder the power pins first.

## PIN CONNECTIONS

**4017B**