The best top-down designs of mice and young men.

Lecture 18 objectives

- Switch debouncing uses input capture and output compare
- Matrix keyboard scanning

Example 3: Count the number of times a switch is pressed
Touch switch causes a falling edge on PT3
Bouncing causes multiple falling edges
Assume bounce<5ms
OC4 interrupt 10 ms after first falling edge on PT3

```
// PT3 input = external signal
unsigned short Count;   // number of presses
void IC_Init(void){
asm sei             // make atomic
  TIOS &= ~0x08;  // PT3 input capture
  TSCR1 = 0x80;   // enable TCNT
  TSCR2 = 0x03;   // 24MHz/8 clock
  TCTL4 = (TCTL4&0x3F)|0x80; // falling edge
```

```
  TIE  |= 0x08;   // Arm IC3
  TFLG1 = 0x08;   // clear C3F
  TIOS |= 0x10;   // PT4 output compare
  TIE  &= ~0x10;  // disarm OC4
  Count = 0;
asm cli
}
void interrupt 11 IC3Han(void){
  TIE  &= ~0x08;      // disarm IC3
  TC4   = TCNT+30000; // 10 ms later
  TIE  |= 0x10;       // arm OC4
  TFLG1 = 0x10;       // clear C4F
}
void interrupt 12 OC4Han(void){
  TMSK1 &= ~0x10;   // disarm OC4
  TMSK1 |= 0x08;    // arm IC3
  TFLG1 = 0x08;     // clear C3F
  if((PTT&0x08)==0){
    Count++;
  }
}
```

*****************upgrade IC.C to solve bounce ***********

8.1.4. Basic Approaches to Interfacing Multiple Keys

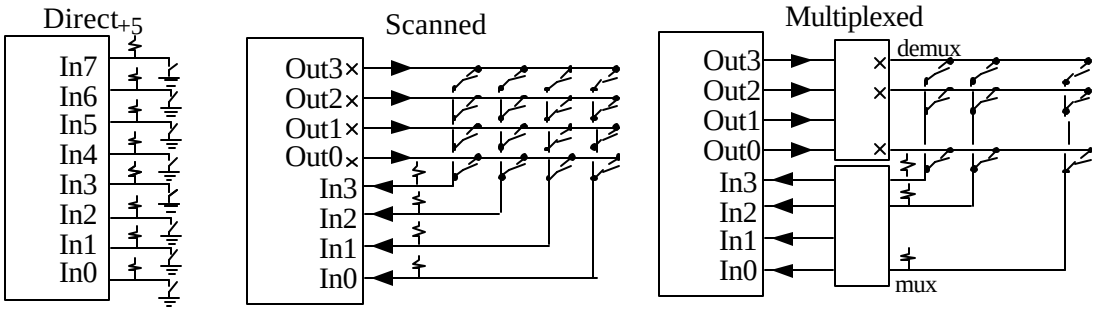


Figure 8.19. Three approaches to interfacing multiple keys.

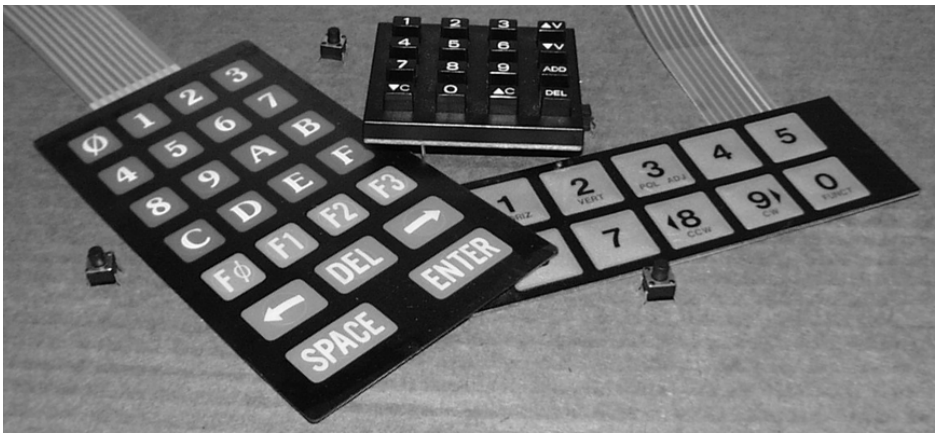


Figure 8.20. Multiple keys are implemented by placing the switches in a matrix.

Row	Out3	Out2	Out1	Out0
3	0	HiZ	HiZ	HiZ
2	HiZ	0	HiZ	HiZ
1	HiZ	HiZ	0	HiZ
0	HiZ	HiZ	HiZ	0

Table 8.1. Scanning patterns for a 4 by 4 matrix keyboard.

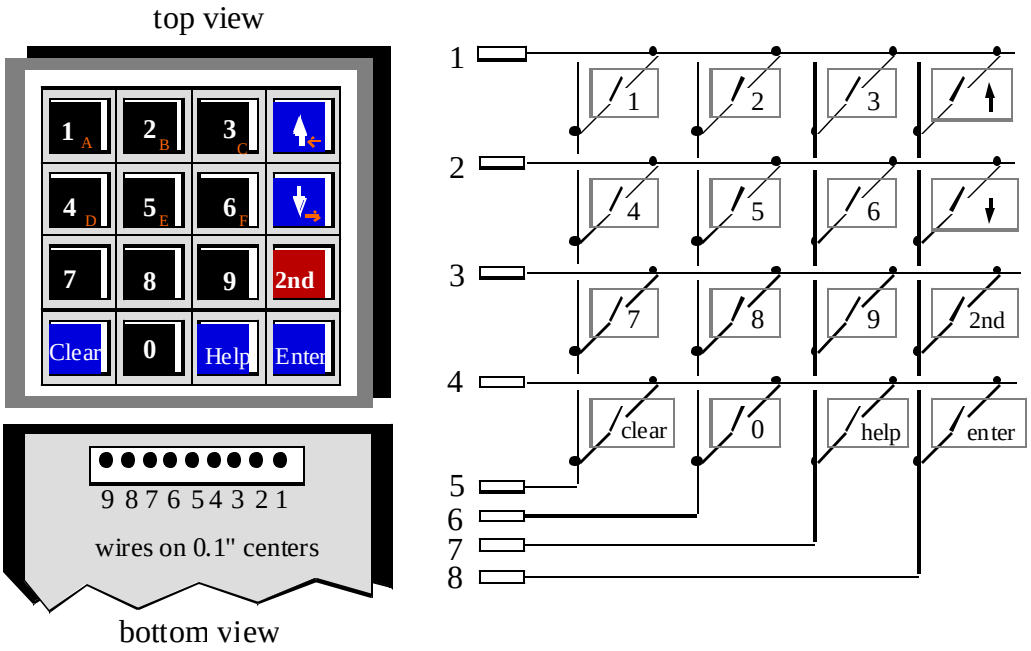
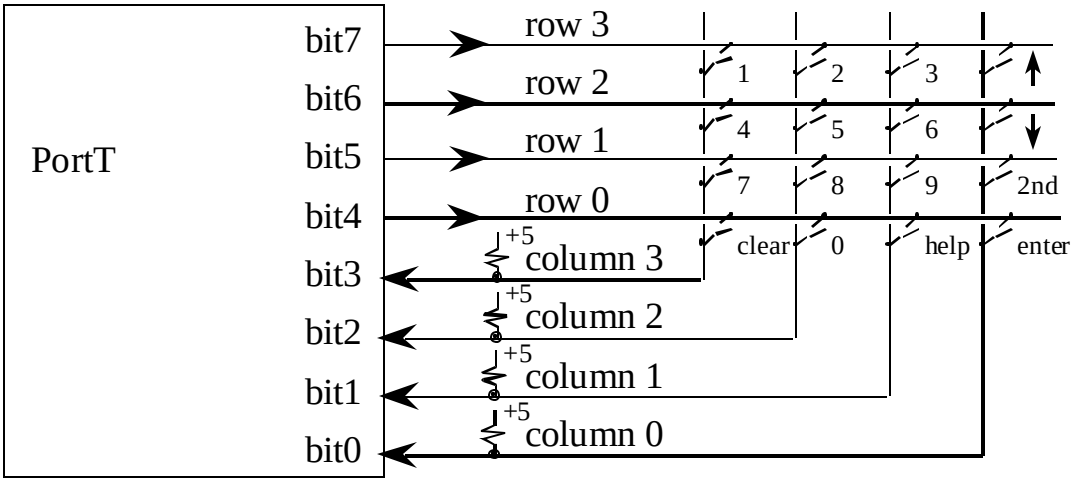
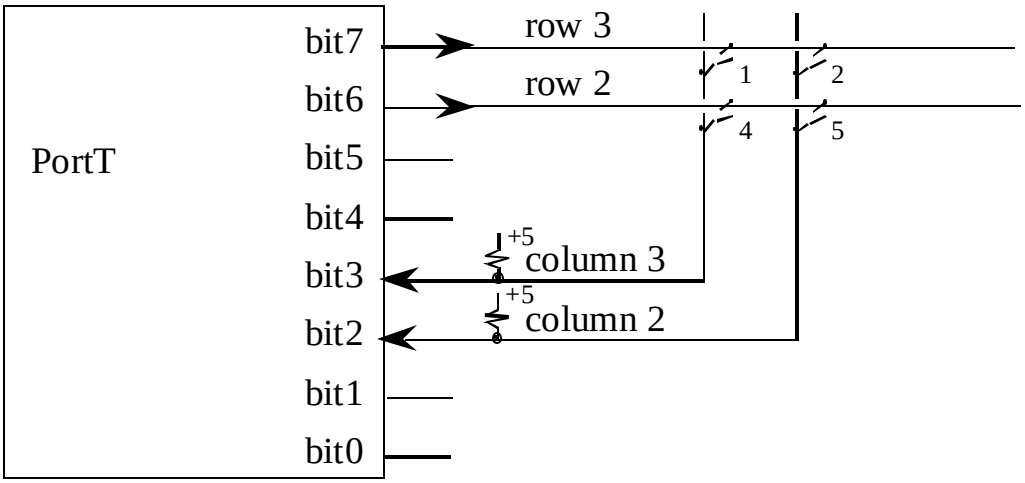


Figure 8.1. 0-9 keyboard, with up arrow, down arrow, 2nd, CLEAR, HELP, and ENTER.



2 by 2 scanned-keyboard.

- keys are divided into rows and columns
- rows are open collector output (zero or off)
- columns are input with pull up resistors
- read the columns (zero means pushed)
- interrupts via input capture



A 2 by 2 matrix keyboard interfaced to the microcomputer.

There are two steps to scan a particular row:

1) Select that row by driving it low (make output 0),
while the other rows are left not driven (make output hiZ).
set the direction register, so that only one row is output

2) Read the columns to see if any keys are pressed in that row,
0 means the key is pressed
1 means the key is not pressed

row 3	row 2	column 3	column 2
0	hiZ	1	2
hiZ	0	4	5

Patterns for a 2 by 2 matrix keyboard.

When all rows are 0, the input capture will fall when any of the 4 keys is pressed. The scanned keyboard operates properly if

1) No key is pressed
2) Exactly one key is pressed
3) Exactly two keys are pressed.

This method can not be used to detect three or more keys simultaneously pressed. If three keys are pressed in a "L" shape, then the fourth key that completes the rectangle will appear to be pressed. Therefore special keys like the shift, control, option, and alt are not placed in the scanned matrix, but rather are interfaced directly, each to a separate input port, like the piano keyboard example.

```
unsigned char LastKey;
unsigned char Count;
// assumes all of PORTT used for this interface
void Key_Init(void){
asm sei             // make atomic
  DDRT = 0xC3;      // PT7-PT6 are oc outputs to rows
                    // PT3-PT2 are inputs from columns
  PTT &= ~0xC0;     // PT7,PT6 will be zero when output
  TIOS &= ~0x0C;    // PT3,PT2 input capture
  TSCR1 = 0x80;     // enable TCNT
  TSCR2 = 0x03;     // 24MHz/8 clock
  TCTL4 = 0xA0;     // falling edge, IC2,IC3
  TIE  |= 0x0C;     // Arm IC3,IC2
  TFLG1 = 0x0C;     // clear C3F,C2F
  TIOS |= 0x10;     // PT4 output compare
  TIE  &= ~0x10;    // disarm OC4
  LastKey = 0;      // should be a fifo
  Count = 0;
asm cli
```

```
}
/* Returns ASCII code for key pressed,
   return zero if no key pressed
   returns 0xFF if more than one is pressed */
unsigned char KeyScan(void){ unsigned char numKey;
unsigned char key,column;
  numKey = 0;           // no key pressed yet
  key = 0;              // default value for no key
//*********** Row 3 ********************************
  DDRT = 0x83;          // select row 3
  column = PTT;         // read columns
  if((column&0x08)==0){
    key = '1';          // row 3, column 3
    numKey++;
  }
  if((column&0x04)==0){
    key = '2';          // row 3, column 2
    numKey++;
  }
//*********** Row 2 ********************************
  DDRT = 0x43;          // select row 2
  column = PTT;         // read columns
  if((column&0x08)==0){
    key = '4';          // row 2, column 3
    numKey++;
  }
  if((column&0x04)==0){
    key = '5';          // row 2, column 2
    numKey++;
  }
  if(numKey>1){
    key = 0xFF;         // multiple key error
  }
  return key;
}

void interrupt 10 IC2Han(void){
  TIE  &= ~0x0C;      // disarm IC3,IC2
  TC4   = TCNT+30000; // 10 ms later
  TIE  |= 0x10;       // arm OC4
  TFLG1 = 0x10;       // clear C4F
```

```
}
void interrupt 11 IC3Han(void){
  TIE  &= ~0x0C;       // disarm IC3,IC2
  TC4   = TCNT+30000; // 10 ms later
  TIE  |= 0x10;        // arm OC4
  TFLG1 = 0x10;        // clear C4F
}
void interrupt 12 OC4Han(void){ unsigned char mykey;
  mykey = KeyScan();
  if(mykey==0xFF){     // more than one
    TFLG1 = 0x10;      // acknowledge OC4
    TC4 = TC4+30000;   // wait for there to be one
  } else{
    TIE  &= ~0x10;      // disarm OC4
    TIE  |= 0x0C;       // arm IC3,IC2
    TFLG1 = 0x0C;       // clear C3F,C2F
    if(mykey){          // exactly one
      LastKey = mykey; // should be fifo Put
      Count++;
    }
  }
  DDRT = 0xC3;          // all rows zero
}
```

run IC4 without, then with bounce
demonstrate rollover