AN1536

Digital Boat Speedometers

Prepared by: Bill Lucas
Industrial Technology Center

INTRODUCTION

This application note describes a Digital Boat Speedometer concept which uses a monolithic, temperature compensated silicon pressure sensor, analog signal–conditioning circuitry, microcontroller hardware/software and a liquid crystal display. This sensing system converts water head pressure to boat speed. This speedometer design using a 30 psi pressure sensor (Motorola P/N: MPX2200GP) yields a speed range of 5 mph to 45 mph. Calibration of the system is performed using data programmed into the microcontroller's internal memory.

A key advantage in all Motorola pressure sensors is the patented X–ducer™, a single piezoresistive implant that replaces the traditional Wheatstone bridge configuration used by competitors. In addition to the X–ducer, Motorola integrates on–chip all necessary temperature compensation, eliminating the need for separate substrates/hybrids. This state–of–the–art technology yields superior performance and reliability. Motorola pressure sensors are offered in several different port configurations to allow measurement of absolute, differential and gauge pressure. Motorola offers three pressure sensor types: uncompensated, temperature compensated and calibrated or fully signal conditioned.

WATER PRESSURE TO BOAT SPEED CONVERSION

A typical analog boat speedometer employs a pitot tube, a calibrated pressure gauge/speedometer and a hose to connect the two. The pitot tube, located at the boat transom, provides the pressure signal corresponding to boat speed. This pressure signal is transmitted to the gauge via the hose. Boat speed is related to the water pressure at the pitot tube as described by the following equation:

$$P \propto e * (V^2/2g)$$

where:

V = speed
P = pressure at pitot tube
e = specific weight of media
g = gravitational acceleration

X–ducer is a trademark of Motorola. Inc.

For example, to calculate P in lb/in^2 for an ocean application use:

V = speed in mph
e = 63.99 lbs/ft^3 at 60°F, seawater
(e will be smaller for fresh water)
g = 32 ft/sec^2
15 mph = 22 ft/sec
1 ft^2 = 144 in^2

$$P = (63.99[lb/ft^3] / 144[in^2/ft^2]) \ (V^2[mph]^2 \ (22/15)^2[(ft/sec)/mph]^2 / 2 \ (32.2)[ft/sec^2])$$

$$P[PSI] = \left(\frac{V}{8.208}\right)^2$$

For example, if the boat is cruising at 30 mph, the impact pressure on the pitot tube is:

$$P = (30/8.208)^2 = 13.36 \text{ psi.}$$

DIGITAL BOAT SPEEDOMETER DESCRIPTION AND OPERATION

The MPX2200GP senses the impact water pressure against the pitot tube and outputs a proportional differential voltage signal. This differential voltage signal is then fed (via an analog switch and gain circuitry) to a single slope analog–to–digital converter (A/D) which is external to the microcontroller. The A/D circuit can complete two separate conversions as well as a reference conversion simultaneously. This A/D utilizes the microcontroller's internal timers as counters and software to properly manipulate the data. The analog switch provides a way to flip the sensor outputs after an A/D conversion step, which is necessary to null out the offset effects of the op–amps. This is accomplished by performing an analog conversion, reversing the sensor's differential output signal, performing another analog conversion, summing the two readings, then dividing this sum by two. Any op–amp offset present will be the same polarity regardless of the sensor output polarity, thus the op–amp offset can be mathematically nulled out. The digital representation of any analog signal is ratiometric to the reference voltages of the A/D converter. Also, the sensor's output is ratiometric to its excitation voltage. Therefore, if both the sensor and A/D reference voltages are connected to the same unregulated supply, the variations in sensor output will be nullified, and system accuracy will be maintained (i.e., systems in which both the A/D converter's digital value — due to variations in the A/D's reference voltages — and sensor's output voltage are ratiometric to the supply voltage so that a voltage regulator is not necessary).

Figure 1 shows the pressure sensor (XDCR) connected to the analog switches of the 74HC4053 which feeds the differential signal to the first stage of op–amps. An A/D conversion is performed on the two op–amp output signals, V_{out1} and V_{out2}. The difference ($V_{out1} - V_{out2}$) is computed and stored in microcontroller memory. The analog switch commutates (op–amp connections switch from Y_0 and Z_0 to Y_1 and Z_1), reversing the sensor output signals to the two op–amps, and another conversion is performed. This value is then also stored in the microcontroller memory. To summarize, via software, the following computation takes place:

Step 1: $V_{first} = V_{out1} - V_{out2}$

Step 2: $V_{second} = V_{out2} - V_{out1}$

Step 3: $V_{result} = (V_{first} + V_{second}) / 2$

Again, because any op–amp offset will remain the same polarity regardless of sensor output polarity, this routine will effectively cancel any amplifier offset. Any offset the sensor may introduce is compensated for by software routines that are invoked when the initial system calibration is done.

The single slope A/D provides 11 or more unsigned bits of resolution. This capability provides a water pressure resolution to at least 0.05 psi. This translates to a boat speed resolution of 0.1 mph over the entire speed range.

Figure 2 describes the pressure versus voltage transfer function of the first op–amp stage.

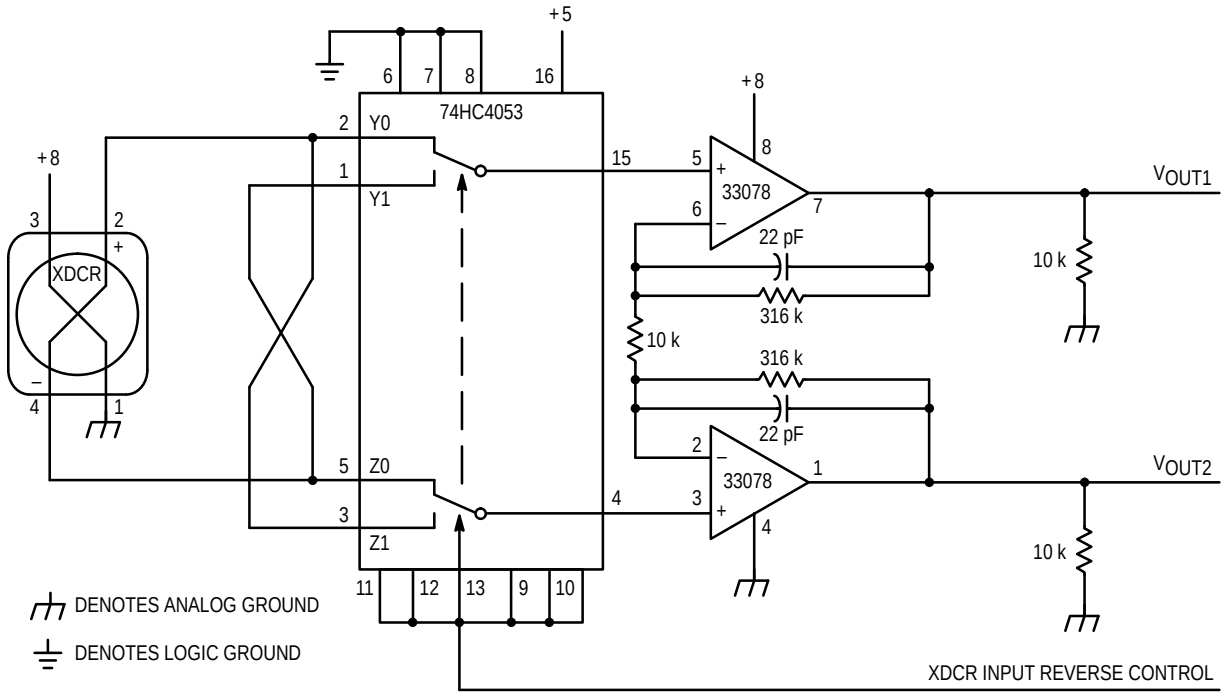


Figure 1. X–ducer, Instrument Amplifier and Analog Switch

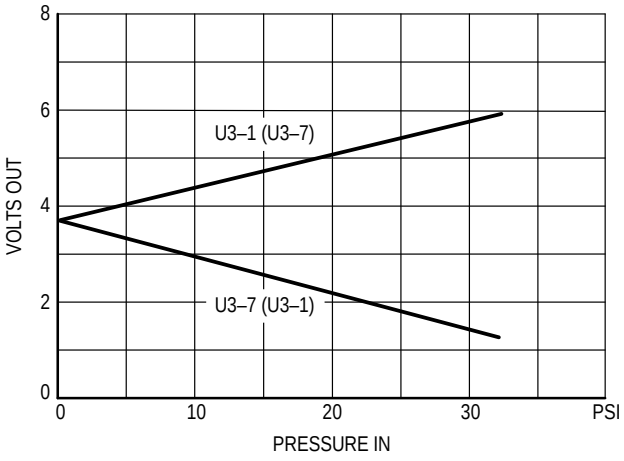


Figure 2. Instrument Amplifier Transfer Function

Figure 3 details the analog circuitry, microcontroller's timer capture registers and I/O port which comprise the single slope A/D. The microcontroller's 16–bit free running counter is also employed, but not shown in the figure.

Comparators U6A, U6B and U6D of the LM139A are used to provide the A/D function. Constant current source, U7, resistors R13 and R14 and diode D2 provide a linear voltage ramp to the inverting inputs of U6, with about 470 microamps charge current to capacitor C8, with transistor Q1 in the off state. C8 will charge to 5 volts in about 5 milliseconds at the given current. Q1 is turned on to provide a discharge path for C8 when required. The circuit is designed such that when the voltage to the inverting inputs of the comparators exceeds the voltage to the noninverting comparators, each comparator output will trip from a logic 1 to a logic 0.

One A/D conversion consists of the following steps: (1) setting the pressure sensor output polarity (via software and the analog switches of U4) to the amplifier inputs of the MC33078 (U3), (2) reading the value of the free running

counter, (3) turning off Q1, and (4) charging C8 and waiting for the three (U6) comparator outputs to change from 1 to 0. When the comparator outputs change state, the microcontroller free running counter value is clocked into the microcontroller's input capture register. Contained in this register then is the number of counts required to charge C8 to a value large enough to trip the comparators. Via software, the voltage signal from U3 (corresponding to the applied pressure signal) can be compared to the "reference."

The boat speed display for this design employs an MC145453 LCD driver and four–digit liquid crystal display, of which three digits and a decimal point are used. Figure 4 shows the connections between the display driver and the display. The display driver is connected to the microprocessor's serial peripheral interface (SPI). The software necessary to initialize, format and drive the LCD is included in the software listing contained in this article.

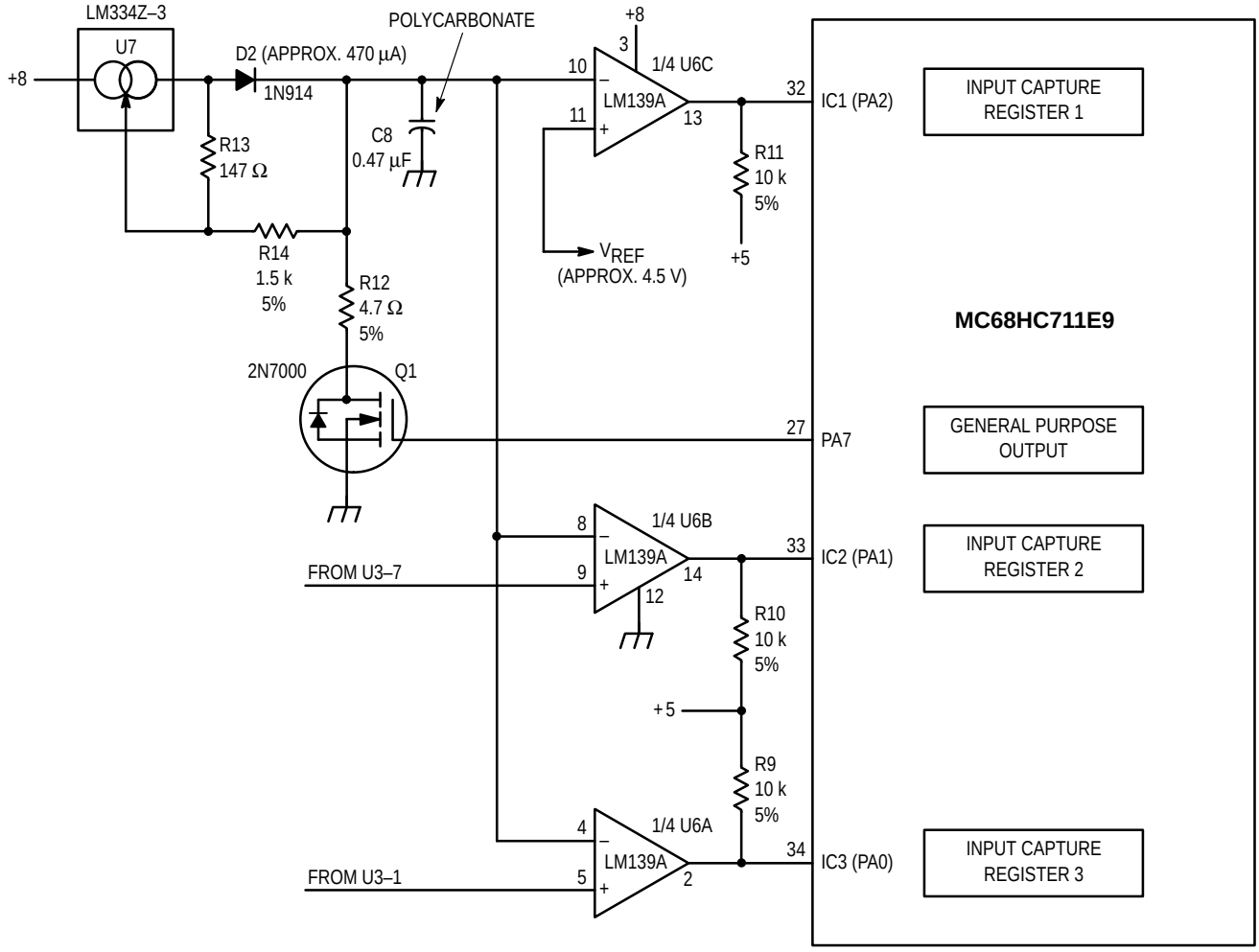


Figure 3. Analog–to–Digital Converter Front End with Microcontroller

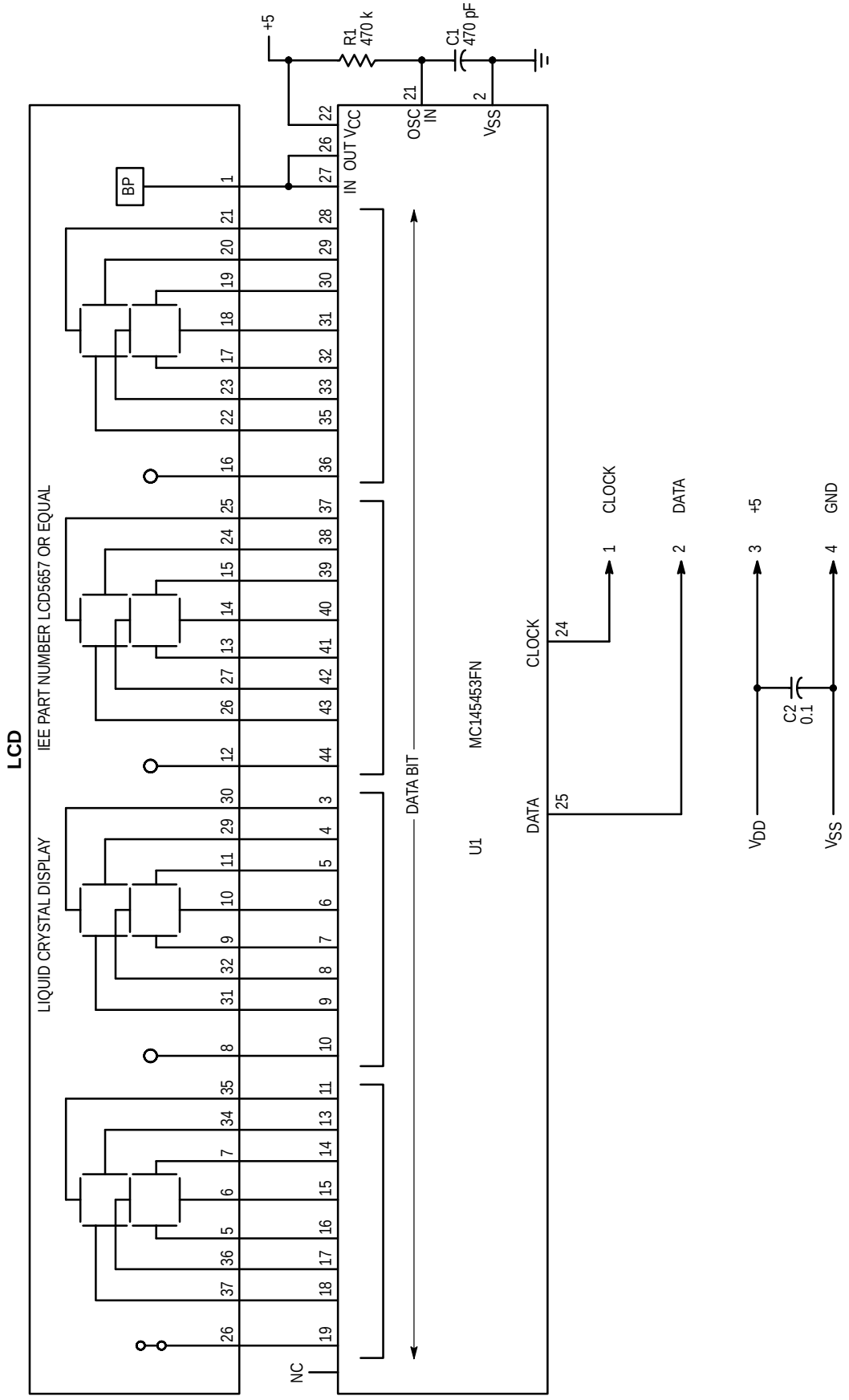
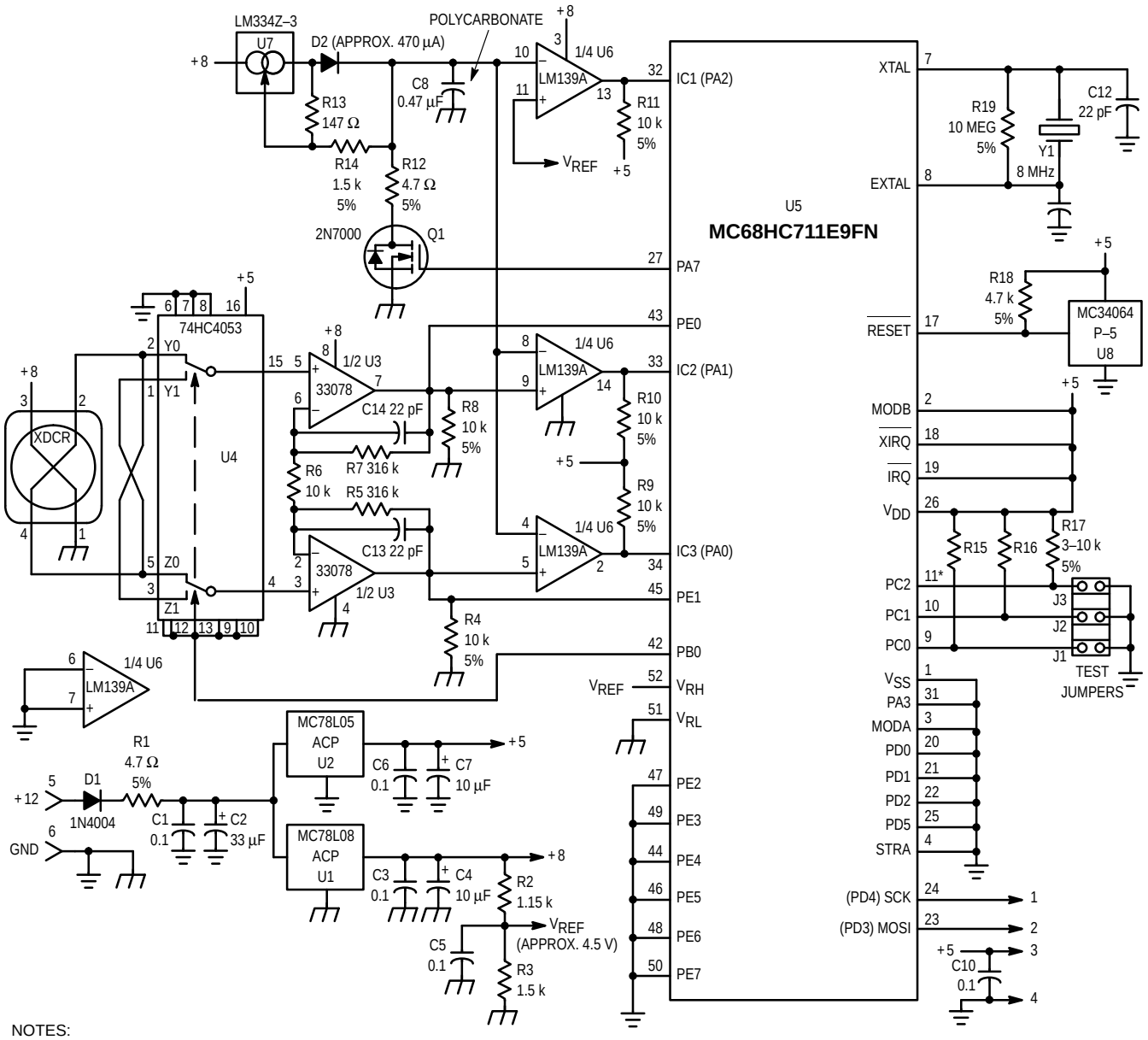


Figure 4. Boat Speedometer Display Board

Table 1 lists the jumper wire selections needed for calibration and operational modes. The jumper wire junction block (J1, J2, J3) is connected to the microprocessor, pins PC0, PC1 and PC2, respectively as shown in Figure 5.

Table 1.

J1	J2	J3	
OUT	OUT	OUT	Display speed in mph
OUT	OUT	IN	100 psi X–ducer installed
OUT	IN	OUT	30 psi X–ducer installed
OUT	IN	IN	15 psi X–ducer installed
IN	OUT	OUT	Full scale calibrate
IN	OUT	IN	Zero calibrate
IN	IN	OUT	Display pressure in psi
IN	IN	IN	Display speed in mph



NOTES:
UNLESS OTHERWISE NOTED, ALL RESISTORS 1% METAL FILM.

* U5 PINS 11–16 (PC2–PC7) ARE CONNECTED HERE FOR TERMINATION PURPOSES.

Figure 5. Boat Speedometer Processor Board

The calibration of this system is as follows. Refer to Table 1.

CAUTION: While installing or changing the proper jumpers described by each step, power must be off. Reapply power to read the display after jumpers have been installed in their proper location for each step. In each step there is a few seconds' delay after switching the power on and before an output is displayed. Steps 1 through 3 must be performed prior to system being operational.

Calibration

1. The pressure range of the system must be established. The present software installed in this design supports 15, 30 and 100 psi sensors. Using an MPX2200D sensor (30 psi) for example, only jumper J2 should be installed. After power is applied, the LCD should read "30." Power off the system prior to proceeding to step 2.
2. The total system offset, due to the sensor and A/D, must be established for the software routine to effectively calibrate. With power off, jumpers J1 and J3 should be installed. Reapply power, and the LCD should respond with "000." The offset value measured in this step is thus stored for use in circuit operation. Power off the system prior to proceeding to step 3.
3. In this step, the system full scale span is calibrated. With power off, install jumper J1 only. Now apply the full rated pressure (30 psi for MPX2200GP) to the sensor, power on and ensure the display reads "FFF." The full scale span measured in this step is thus stored for use in circuit operation. Power off the system prior to step 4.

Operation

4. Ensure power is off, and install jumpers J1, J2 and J3. The system is now ready for operation. Simply apply power and pressure to the sensor, and the LCD will display the proportional speed above 5 mph, up to the limits of the sensor.

REFERENCES

Burry, Michael (1989). "Calibration–Free Pressure Sensor System," Motorola Application Note AN1097.

```
NOTE. THIS  WAS COMPILED WITH A COMPILER COURTESY OF:
                INTROL CORP.
                9220 W. HOWARD AVE.
                MILWAUKEE, WI. 53228
                PHONE (414) 327-7734.
  SOME SOURCE CODE CHANGES MAY BE NECESSARY FOR COMPILATION WITH OTHER COMPILERS.
  THE HEADER FILE io6811.h HAS I/O PORT DEFINITIONS FOR THE I/O PORTS PARTICULAR TO THE MC68HC711E9. A TYPICAL ENTRY FOR
PORT A WILL FOLLOW. THE FIRST LINE ESTABLISHES A BASE ADDRESS BY WHICH ALL I/O FACILITIES AND COUNTERS ARE BIASED.
  REFER TO THE  MC68HC711E9 DATA FOR MORE INFORMATION RELATIVE TO I/O AND TIMER ADDRESSES.

  #define    IOBIAS 0x1000  /* BASE ADDRESS OF THE I/O FOR THE 68HC11 */
  #define PORTA (* (char *) (IOBIAS + 0))   /* PORT A */

  THE STARTUP ROUTINE NEED ONLY LOAD THE STACK TO THE TOP OF RAM, ZERO
  THE MICROCONTROLLER'S RAM AND PERFORM A BSR MAIN (BRANCH TO SUBROUTINE "MAIN").
  THIS SOURCE CODE, HEADER FILE, COMPILED OBJECT CODE, AND LISTING FILES ARE AVAILABLE ON:
                                      THE MOTOROLA FREEWARE LINE
                                      AUSTIN, TX.
                                      (512) 891-3733.

Bill Lucas 6/21/90
THE CODE STARTS HERE  */

#include <io6811.h>  /*  I/O port definitions */

/* define locations in the eeprom to store calibration information */
#define EEPROM (char*)0xb600 /* used by calibration functions */
#define EEBASE 0xb600 /* start address of the eeprom */
#define ADZERO (* ( long int *)( EEBASE + 0 )) /* auto zero value */
#define HIATOD (* ( long int *)( EEBASE + 4 )) /* full scale measured input */
#define XDCRMAX (* ( char *)( EEBASE + 8 )) /* full scale input of the xdcr */
union bytes {
          unsigned long int l;
          char b[4];
            };  /* ADZERO.l for long word   ADZERO.b[0]; for byte */

const char lcdtab[] = { 95,  6,  59,  47, 102, 109, 125, 7, 127, 111,  0   };
/*   lcd pattern table   0   1    2    3   4    5    6   7   8    9  blank */

const int dectable[] = { 10000, 1000, 100, 10 };

char digit[5]; /* buffer to hold results from cvt_bin_dec function */

/* ###################################################################### */

/* real time interrupt service routine  */

void real_time_interrupt (void) /* hits every 4.096 ms. */
   {
    TFLG2 = 0x40;  /* clear the interrupt flag */
   }

/* ###################################################################### */

/* ###################################################################### */

/*          write_eeprom(0xA5,EEPROM);   write A5h to first byte of EEPROM */
void write_eeprom(char data, char *address)
{
             PPROG = 0x16;             /* single-byte erase mode */
             *address = 0xff;     /* write anything */
             PPROG = 0x17;             /* turn on programming voltage */

             delay();
             PPROG = 0x0;        /* erase complete */

    /* now program the data */
    PPROG = 0x02;                 /* set eelat bit */
             *address = data;     /* write data */
             PPROG = 0x03;       /* set eelat and eepgm bits */
             delay();
             PPROG = 0;                /* read mode */
             /* programming complete */
}
/* ###################################################################### */

long int convert(char polarity)
```

```
            {

unsigned int cntr;   /* free running timer system counter */
unsigned int r0;     /* difference between cntr and input capture 1 register */
unsigned int r1;     /* difference between cntr and input capture 2 register */
unsigned int r2;     /* difference between cntr and input capture 3 register */
unsigned long difference; /* the difference between the upper and lower
                     instrument amplifier outputs */
unsigned long int pfs; /* result defined as percent of full scale relative to
                          the reference voltage */

    if (polarity == 1)          /* set the hc4053 configuration */
        PORTB &= 0xfe;          /* polarity = 1 means + output of sensor */
         else  PORTB |= 0x1;    /* is connected to the upper opamp */

    delay();   /* this will allow the hc4053 to stabilize and the cap
                  to discharge from the previous conversion */
             TFLG1=0X07;                    /* clear the input capture flags */
                  cntr=TCNT;                    /* get the current count */
                   PORTA &= 0X7F;               /* turn the fet off */
                     while ((TFLG1 & 0X7) < 7); /* loop until all three input capture
                                                   flags are set */
              r0 = TIC1 – cntr;             /* reference voltage */
              r1 = TIC2 – cntr;             /* top side of the inst. amp */
              r2 = TIC3 – cntr;             /* lower side of the inst. amp */
              PORTA |= 0X80;               /* turn the fet on */
              if (polarity == 1)
                 difference = ( r1 + 1000 ) – r2;
                   else  difference = ( r2 + 1000 ) – r1;
                    pfs = (difference * 10000) / r0;
                      if (difference > 32767) /* this will cover up the case
                                                 where the a to d computes a
                                                 negative value */
                          pfs=0;
                                                  return ( pfs );
             }

atod()  /* computes the a/d value in terms of % full scale */
  {
unsigned long int x,y,z;
  x = convert(1);  /* normal */
   y = convert(0);  /* reversed */
    z = (x + y)>>1 ;  /* 2x difference / 2  */
     return(z);  /* z is percent of full scale */
  }

integrate() /* returns the a/d value in terms of % full scale and computes
               offset from calibration values */
{
unsigned long int j;
int i;
j=0;
for (i=0; i<20; ++i)
     j +=atod();
        j = (j/20) – ADZERO;  /* null out the xdcr zero input offset */
          return(j);
}

cala2d()   /* returns the average of 50 raw a/d conversions this is only
                used by the calibration functions */
{
unsigned long int j;
int i;
j=0;
for (i=0; i<50; ++i)
  { j +=atod(); }
    j=j/50;
    return(j);
}

/* ###################################################################### */

cvt_bin_dec ( unsigned int arg )
{
char i;

  for ( i=0; i < 6; ++i )
```

```
    {
      digit[i] = 0;  /* put blanks in all digit positions */
    }

       for ( i=0; i < 4; ++i )
         {
             if ( arg  >= dectable [i] )
               {
                 digit[i] = arg /dectable[i];
                 arg = arg-(digit[i] * dectable[i]);
               }
          }
digit[i] = arg;
}

/* ###################################################################### */

delay()
{
int i;
for (i=0; i<1000; ++i); /* delay about 15 ms. @ 8 mhz xtal */
}

/* ###################################################################### */

/* set-up i/o for the single slope a/d, initialize the spi port, then
   initialize the MC145453 for output */

init_io(void)
{
char i;

/* set-up i/o for the a/d */
PACTL |= 0X80;   /* make pa7 an output */
PORTA |= 0X80;   /* turn the fet on */
PORTB &= 0X7F;   /* set-up the HC4053 in the Y0/Z0 connect mode */
TCTL2 = 0X2A;    /* capture on falling edge for timer capture 0,1,2 */
TFLG1 = 0X07;    /* clear any pending capture flags */

/* set-up the i/o for the spi subsystem */
PORTD=0x2f;  /* set output low before setting the direction register */
DDRD=0x38;   /* ss = 1, sck = 1, mosi = 1 */
SPCR=0x51;   /* enable spi, make the cpu the master, E clock /4  */

/* initialize the lcd driver */
  for (i=0; i<4; ++i)   /* four bytes of zeros */
       {
         write_spi(0);
       }
  write_spi (2);  /* this creates a start bit and data bit 1
                     for the next write to the mc145453 */
}

/* ###################################################################### */

/* this is an attempt at the newton square root method */
sqrt(unsigned long b)
{

unsigned long x0,x1;

   if ( b < 4 ) { b=2; return (b); }
       else
     x0=4;
     x1=10;
        while (x0 != x1)
          {
            if( (x1-x0) ==1 ) break;
              x1=x0;
              x0=(( (b/x0) +x0 ) >> 1 );
          }
   b=x0;
   return (b);
}

/* ###################################################################### */
```

```
write()
{
char i;
  digit[1]=10;
  if (digit[2]==0)
       {digit[2]=10;}
    if (  digit[2]==10 && digit[3]==0 )
         {digit[3]=10;}
   for ( i=1; i<5; ++i )
     {
      if (i==4)
       write_spi((lcdtab[digit[i]])+0x80);
        else
         write_spi(lcdtab[digit[i]]);
     }
 write_spi (2);      /* this creates a start bit and data bit 1
                        for the next write to the mc145453   */
}


write_spi( char a )  /* write a character to the spi port */

  {
    SPDR=a;
       while ( ! ( SPSR & 0x80 ) ) {}  /* loop until the spif = 1 */
  }

/* ###################################################################### */

/*  This function is called at power-up and will determine the operation
    of the system. The user must complete the system configuration prior
    to setting the jumper in the first or last two configurations in the
    table or erroneous operation is guaranteed!
          test/operation jumper configuration:

           J3  J2  J1     1 = jumper removed

            1   1   1   display speed in mph
            1   1   0   reserved
            1   0   1   30 psi xdcr installed
            1   0   0   15 psi xdcr installed
            0   1   1   full scale calibrate
            0   1   0   zero calibrate
            0   0   1   display pressure in psi
            0   0   0   display speed in mph     */

setconfig()
{
char i;
for ( i=0; i<125; ++i )
    delay();   /* to let the charge pump come to life wll */
i = PORTC & 0x07;  /* and off the unused bits */
  if ( i == 7 )
    display_speed();
               if ( i == 6 )
                 setup_error();  /* non-valid pattern output -SE- on display*/
                 if ( i == 5 )
                        {write_eeprom(30,&XDCRMAX); /* xdcr is 30 psi */
                              display(30);
                             }
                           if ( i == 4 )
                              {write_eeprom(15,&XDCRMAX); /* xdcr is 15 psi */
                                display(15);
                               }
                                 if ( i == 3 )
                                   fullscale_calibrate();
                                                              if ( i == 2 )
                                                                zero_calibrate();
                                                                        if ( i == 1 )
                                       display_pressure();
                                         else
                                                                                  display_speed();
}

/* ###################################################################### */

display(char d)
```

```
{
if (d==30)
{
write_spi(0);    /* blank the upper digit */
write_spi(0);    /* blank the next to upper digit */
write_spi(47);   /* 3 */
write_spi(95);   /* 0 */
}
if (d==15)
{
write_spi(0);    /* blank the upper digit */
write_spi(0);    /* blank the next to upper digit */
write_spi(6);    /* 1 */
write_spi(109);  /* 5 */
}

write_spi(2);
while(1);
}

/* ###################################################################### */

fullscale_calibrate()
{
int i;
long int temp;
union bytes average;
temp=0;
    average.l = cala2d();  /* get the average of 50 a/d conversions */
             for ( i=0; i<4; ++i)
                write_eeprom(average.b[i],EEPROM+i+4);


write_spi(0);    /* blank the upper digit */
write_spi(113);  /* F */
write_spi(113);  /* F */
write_spi(113);  /* F */
write_spi(2);
while(1);

}

/* ###################################################################### */

zero_calibrate()
{
int i;
long int temp;
union bytes average;
temp=0;

    average.l = cala2d();  /* get the average of 50 a/d conversions */
             for ( i=0; i<4; ++i)
              write_eeprom(average.b[i],EEPROM+i);

write_spi(0);   /* blank the upper digit */
write_spi(95);  /* 0 */
write_spi(95);  /* 0 */
write_spi(95);  /* 0 */
write_spi(2);
while(1);
}

/* ###################################################################### */

/* speed=8.208(square root(%full scale*transducer full scale)) */
display_speed()
{
long atod_result;
unsigned int j;
             while(1)
             {
                         atod_result = integrate(); /* read the a/d */
                           atod_result=( (atod_result*10000) / (HIATOD-ADZERO) ) * XDCRMAX;
           atod_result=sqrt(atod_result);
          atod_result=(atod_result*8208)/10000;
         j=atod_result;
```

```
         if (j<50)
           { j=0; }
  cvt_bin_dec ( j );
  write();

            }
}

/* ###################################################################### */

/* pressure=%full scale*transducer max pressure */
display_pressure()
{
long atod_result;
int j;
             while(1)
             {
              atod_result = integrate(); /* read the a/d */
                atod_result=( (atod_result*1000) / (HIATOD-ADZERO) ) * XDCRMAX;
           j=atod_result/100;
  cvt_bin_dec ( j );
  write();

             }
}

/* ###################################################################### */

setup_error()   /* write "SE" on the display */
{
write_spi(0);
write_spi(109);  /* S */
write_spi(121);  /* E */
write_spi(0);
write_spi(2);
while(1);
}

/* ###################################################################### */


main()
{
init_io();
setconfig();  /* determine how to function */
while(1);  /* should never return here except after calibration */
}
```


Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.