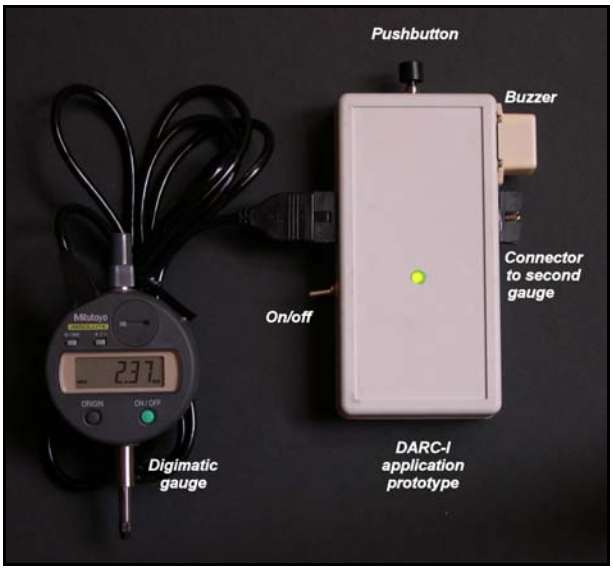
Mitutoyo sensor interface

Application Note: Interfacing ToothPIC DARC-I to Mitutoyo Digimatic sensors



Summary

The Digimatic range of sensors from Mitutoyo Ltd are distance measurers (calipers, micrometers, etc) with digital outputs. This application note describes how one customer interfaced them to the DARC-I ToothPIC data acquisition and remote control system. The topics discussed in this application note are relevant to interfacing a variety of digital-output sensors with custom data formats to any ToothPIC firmware solution.

The customer wanted to create a prototype measuring device that would measure variations in surface height at different points along a surface. On the press of a button, the DARC-I application was to make measurements on two Digimatic sensors and then transmit these to a PDA via Bluetooth for storage.

During the development of the prototype, two new features were added: the ability to monitor the battery voltage, and a buzzer to provide audible confirmation that the PDA had logged the data when the button was pressed.

Hardware design

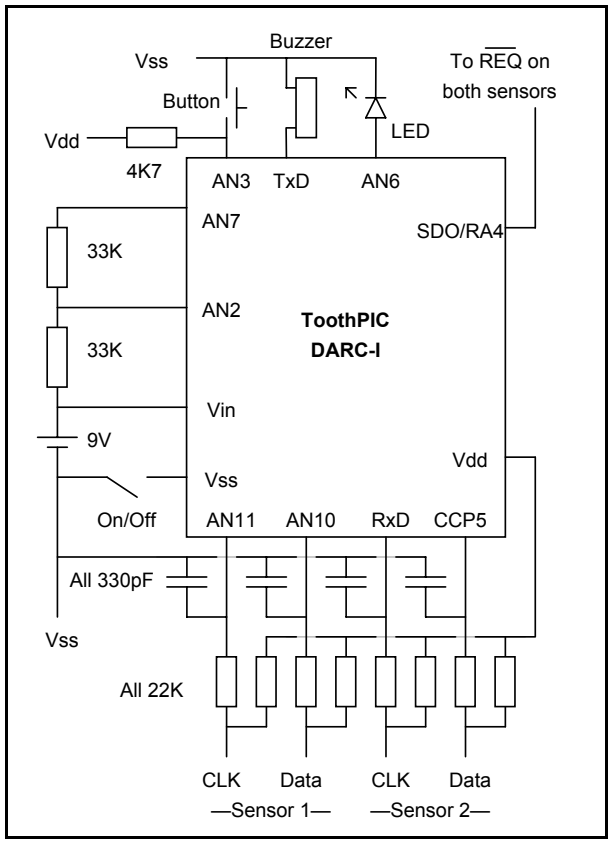
It was possible to realize the design using just the ToothPIC DARC-I and a few discrete external components. (See schematic, right.)

The sensors required an open-collector output pin to request data. There is one such pin on the ToothPIC: the SDO pin, which is multiplexed to the open-collector RA4 pin.

The Mitutoyo data was clocked in from two open-collector outputs from each sensor to the ToothPIC, via pull-up resistors and filter circuits as recommended by Mitutoyo.

A pushbutton was connected to AN3 which was used to register when a reading needed to be taken. It was 'debounced' in software to reduce component count. A buzzer was powered by output TxD, and an LED, with integral resistor, was used as an 'on' indicator. The use of an I/O pin rather than direct power connection for the LED has two advantages: in the case of a circuit malfunction, the LED will not light. Additionally, if sleep mode is used, the LED can be extinguished to save power.

To provide battery health monitoring, a potential divider halved the battery voltage and fed the result into AN2 to provide an indication of the unregulated battery voltage under power. The analog input would compare this to *Vdd* indicating the power remaining until the battery voltage fell to 5V. The ground rail of the potential divider was powered by AN7 so that this circuit could be powered-down during sleep mode.



Software design

ASCII commands are sent to and from the DARC-I via a Bluetooth connection. A few simple changes to the publicly-available source code were made to customize the firmware for Mitutoyo sensors.

The main firmware addition was the code to read data from the sensors, *e.g.*:

```
#define NUM_DATA 13
#define NUM_BIT  4
#define HIGH_BIT (1 << (NUM_BIT-1))
char TranslateLeftMitutoyoData( void )
{
  static rom unsigned char pLeftGauge[] = { 0x81 };
  signed char SecsAtStart, tDiff;
  unsigned char pData[NUM_DATA];
  unsigned char iData, iBit;

  // if clock is not high, error
  if ( MCLPin == 0)  return 0x00;

  // issue data request
  ReqPin = 0;
  SecsAtStart = RealTimeClock.sec;

  // get data, timeout if gauge not connected
  for (iData = 0; iData < NUM_DATA; iData++ )
  {
    // get hexadecimal digit
    pData[iData] = 0;
    for (iBit = 0; iBit < NUM_BIT; iBit++)
    {
      // wait for clock to go low
      while ( MCLPin == 1 )
      {
        // if waited more than 2 secs, error
        tDiff = RealTimeClock.sec - SecsAtStart;
        if (tDiff < 0) tDiff += 60;
        if (tDiff >= 2)  return 0x00;
      }

      // get new data bit and shift into place
      pData[iData] = pData[iData] >> 1;
      if ( MDLPin == 1 ) pData[iData] &= HIGH_BIT;

      // clock has started, so clear data request
      ReqPin = 1;

      // wait for clock to go high
      while (MCLPin == 0 )
      {
        // if waited more than 2 secs, error
        tDiff = RealTimeClock.sec - SecsAtStart;
        if (tDiff < 0) tDiff += 60;
        if (tDiff >= 2)  return 0x00;
      }
    }
  }

  SendResponseBytes( pLeftGauge , 0, 1, SRB_FIRST);
  SendResponseBytes( 0, pData, NUM_DATA, SRB_LAST);
  return 0xFF;  // success!
}
```

The entire modified source code is available for download as `DARC-I Mitotoyo.c`. It includes the following commands:

`038400`	Read sensors
`038401`	Check for sensor presence and read
`0385xx`	Buzz buzzer for `xx` units of 10ms
`0209`	Enter sleep state (cycle power to wake)

In response to a button press or a `0384yy` command, DARC-I reads the sensors and the battery power and reports with the following responses. (Sensor readings D1, D2 *etc* are defined in *Design specifications for Digimatic Data Output Interface*, available from Mitutoyo Ltd.)

`0F81aabbccddeeffgghhiijjkkllmm`

Reading of sensor 1: D1=aa, D2=bb, etc.
If mm >= F0, error occurred.

`0F82aabbccddeeffgghhiijjkkllmm`

Reading of sensor 2: D1=aa, D2=bb, etc.
If mm >= F0, error occurred.

`0F83wwww`

Battery voltage: `3FF` = 10V, `0200` = 5V or less, varies linearly in between.

Applications to other sensors

The principles described here apply widely to:

- Interfacing ToothPIC to any sensor with a custom data interface
- Low-power design for battery powered products
- Use of open collector circuits for voltage level shifting

Contact FlexiPanel Ltd for further technical support and to enquire about consulting services for customizing ToothPIC products.