

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 Mitutoyo.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.

```
0F83wwwww
```

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.

