Definitions:

Term	Definition
uint8	8-bit unsigned integer
float	IEEE 754-2008 binary32 (aka 'single') floating point number.
int32	32-bit signed integer
uint16	16-bit unsigned integer
int16	16-bit signed integer

Physical Serial Interface:

The serial link by default runs at 1.25 megabaud. The baud rate can also be set to 115200 or 19200, although this will result in slower streaming output rates:

Baud Rate	Streaming Sample Speed
1,250,000	7000 Hz.
115,200	500 Hz.
19,200	100 Hz.

Table 1

The serial link uses a two-wire half-duplex configuration – the sensor cannot receive bytes while it is writing. Therefore, to stop streaming, it is necessary to send a jamming sequence of at least 14 characters, which is one byte longer than a streaming sample.

The serial link uses a 9-bit format with 8 bits of data and the last bit used for 'even' parity.

Modbus Interface:

The Modbus interface follows the timing and checksum rules specified in the "Modbus Over Serial Line" specification at http://www.modbus.org/specs.php.

The Digital F/T board runs as a Modbus slave until the 'start streaming' custom Modbus command is sent, after which it starts streaming data until it receives a 14-byte jamming sequence (the jamming sequence can consist of random character values).

If you are writing your own Modbus interface to the Digital F/T, you must support the "Read Holding Registers" (section 6.3 in the "Modbus Protocol Specification" v1.1b) command and one or both of the "Write Single Register" (section 6.6) and "Write Multiple Registers" (section 6.12) commands. You must also be able to send Custom (aka "User Defined") function codes.

Each serial Modbus packet has the following structure (taken from section 2.3 of the “Modbus Serial Line Protocol and Implementation Guide” V1.02):

Address Field (always 10 for Digital F/T)	Function Code	Data (may be up to 252 bytes and contain a structure with multiple fields)	CRC High Byte	CRC Low Byte

The sensor uses a fixed Modbus slave address of 10, and only communicates using the binary “RTU” serial Modbus packet format. The CRC calculation for the binary RTU mode, including sample source code, is described in Section 6.2.2 of the “Modbus Serial Line Protocol and Implementation Guide” v1.02.

Custom Service Codes:

Each custom service code uses the same serial Modbus command and response format:

Slave Address (10)	Function Code Command: The function code to execute. Response: The function code executed (same as command).	Data	CRC High Byte	CRC Low Byte

Code	**Description**	**Data**	**Response**
106	Unlock storage command: Use this function code before writing gains and offsets for selected calibration.	1 data byte – 0xaa to unlock, 0x18 to lock.	A Modbus packet with one data byte of value 1 for successful execution, or a standard Modbus error packet if unsuccessful.
70	Start strain gauge streaming	1 data byte with value 0x55.	A Modbus packet with one data byte with value 1 for successful reception, or a standard Modbus error packet if unsuccessful. After successful reception of a command to start strain gauge streaming, the sensor will delay for 20 ms to give the client time to disable the Modbus protocol. After this 20 ms delay, the sensor will begin sending out streaming data samples that are not encapsulated in Modbus frames.

Table 2

Important Holding Register Addresses

Addresses are base 0:

Address	Description
0x00e3	Calibration 1
0x01a3	Calibration 2
0x0263	Calibration 3
0x0323	Calibration 4
0x03e3	Calibration 5
0x04a3	Calibration 6
0x0563	Calibration 7
0x0623	Calibration 8
0x06e3	Calibration 9
0x07a3	Calibration 10
0x0863	Calibration 11
0x0923	Calibration 12
0x09e3	Calibration 13
0x0aa3	Calibration 14
0x0b63	Calibration 15
0x0c23	Calibration 16
0x001D	Status Word
0x0000-0x0005	Active Gain pots
0x0006-0x000B	Active Offset DACs
0x000C	Optional "Session ID" register.
0x001F	Baud Rate Register. Values:

Register Value	Baud Rate
0	1,250,000
1	19,200
2	115,200

The value is only stored to non-volatile memory if bit 0 of the "Mode" register is a '1'. Baud rate changes take effect immediately.

Address	Description
0x001E	Mode Register. Set bit 0 of this register to a '1' in order to save changes to the baud rate. The mode bit must be set **before** the baud rate is changed in order to save the new baud rate.

Table 3

Data Structures:

Endianness:

The Digital F/T stores all numbers in big-endian (aka "Network Order") format. If your host system uses a little-endian ("Byte-swapped") format, any custom software you write will have to swap the bytes in each numeric field before performing any calculations using those values.

Streaming Sample Format:

```
{
int16  G0;
int16  G2;
int16  G4;
int16  G1;
int16  G3;
int16  G5;
uint8  check;
}
```

The fields G0-G5 make up the “gauge vector”. When calculating F/T values, this vector is multiplied with the calibration matrix using the standard matrix multiplication algorithm. Note that the strain gauges are sent in the order (G0, G2, G4, G1, G3, G5), and must be reordered to (G0, G1, G2, G3, G4, G5) before performing F/T calculations (or the calibration matrix itself can be reordered to accommodate the streaming sample ordering). The streaming sample is structured this way in order to sample the gauges which “cooperate” on the most axes as close together as possible.

The ‘check’ field is a single byte containing a checksum and a status bit:

Bit Number	Meaning
0-6	Additive checksum
7	Status

The additive checksum can be calculated using the following algorithm. The ‘&’ operator in the following pseudocode indicates a bitwise ‘and’ operation, and the ‘sampleBytes’ parameter contains the 13 bytes that make up a streaming sample packet.

```
CheckSample( sampleBytes:array of bytes ) : true/false
      checkSum = 0
      for i = 0 to 11
            checkSum = checkSum + sampleBytes[i]
      next i
      if ( checkSum & 0x7f ) <> ( sampleBytes[12] & 0x7f )
            return false     /* checksum does not match. */
      else
            return true      /* checksum matches. */
      end if
```

The status bit is ‘0’ if the system is healthy, or ‘1’ if there is an error code. If the status bit indicates an error, stop streaming by sending a “jamming sequence” of 14 bytes, then use standard Modbus to read the system status holding register to get the full error code. The jamming sequence can consist of arbitrary values; any character value will stop streaming.

Since there is no contingency in Modbus for high-speed streaming data, the streaming samples are not sent in standard Modbus packets - they are sent without any encapsulation. This requires the client software to disable any Modbus processing of received packets while data streaming is active.

Status Word:

The status word is a bitmap which contains information about the errors that can occur in various subsystems of the Digital F/T sensor:

Bit Number	Meaning If Set
0	Watchdog reset – the Digital F/T was reset by the watchdog timer.
1	Excitation voltage too high.
2	Excitation voltage too low.
3	Artificial analog ground out of range.
4	Power supply too high.
5	Power supply too low.
6	Not used.
7	Error accessing stored settings in EEPROM.
8	Invalid configuration data.
9	Strain gauge bridge supply current too high.
10	Strain gauge bridge supply current too low.
11	Thermistor too high.
12	Thermistor too low.
13	DAC reading out of range.
14	Not used.
15	Any error sets this bit.

Calibration Structure:

Calibration data is read using standard Modbus holding register read commands. You cannot write data to the calibration structures.

```
 {
  uint8   CalibSerialNumber[8];
  uint8   CalibPartNumber[32];
```

```
uint8   CalibFamilyId[4];
uint8   CalibTime[20];
float   BasicMatrix[6][6];
uint8   ForceUnits;
uint8   TorqueUnits;
float   MaxRating[6];
int32   CountsPerForce;
int32   CountsPerTorque;
uint16  GageGain[6];
uint16  GageOffset[6];
uint8   Resolution[6];
uint8   Range[6];
uint16  ScaleFactor16[6];
uint8   UserField1[16];
uint8   UserField2[16];
uint8   SpareData[16];
}
```

Field	**Description**
CalibSerialNumber	The serial number of the calibration as a null-terminated ASCII string.
CalibPartNumber	The ATI part number of the calibration as a null-terminated ASCII string.
CalibFamilyID	The ATI Calibration family as a null-terminated ASCII string.
CalibTime	The date the calibration was released as a null-terminated ASCII string in a format like ‘1970-01-01 00:00:00’.
BasicMatrix	The calibration matrix. Multiply this matrix with the gauge vector to get a vector containing the force and torque values.
ForceUnits	An integer which encodes the force units of the calibration.

Code	**Force Unit**
1	Pound
2	Newton
3	Kilopound
4	Kilonewton
5	Kilogram-equivalent force
6	Gram-equivalent force

TorqueUnits	An integer which encodes the torque units of the calibration.

Code	Force Unit
1	Pound-inch
2	Pound-foot
3	Newton-meter
4	Newton-millimeter
5	Kilogram(-equivalent)-centimeter
6	KiloNewton-meter

MaxRating	The maximum rated load of each axis, in the order [Force X, Force Y, Force Z, Torque X, Torque Y, Torque Z].
CountsPerForce	The counts per unit force of the calibration.
CountsPerTorque	The counts per unit torque of the calibration.
GageGain	The gain values used for each of the six strain gauges in this calibration.
GageOffset	The offset values used for each of the six strain gauges in this calibration.
Resolution	Not used in this product.
Range	Not used in this product.
ScaleFactor	Not used in this product.
UserField1	Not used in this product.
UserField2	Not used in this product.
SpareData	Not used in this product.

Each calibration structure is 338 bytes long, so to read the settings for a particular calibration, start at the address of the desired calibration from Table 3 and read 169 holding registers, since each register is two bytes wide. It will be necessary to split the reading up into multiple Modbus commands, due to length restrictions in the Modbus protocol.

Tool Transformations:

A tool transformation is a mathematical procedure that allows you to calculate the force and torque values that are acting at a point other than the origin of the sensor itself, and also allow you to rotate the reference frame, for example to line up the sensor axes with your tooling axes. The transformation is calculated by applying rotation and translation matrices to the original calibration matrix (the "BasicMatrix" element of the calibration structure).

In general, the transform takes the following form:

$$[R]_{6x6}[D]_{6x6}[C]_{6x6}[G]_{6x1} = [F]_{6x1}$$

Equation 1

Where:

R = Rotation matrix
D = Displacement (translation) matrix
C = Calibration matrix
G = Raw gage values
F = Resolved force/torque values

The rotation matrix is defined as:

$$R =$$

$$\begin{bmatrix} cy \cdot cz & sx \cdot sy \cdot cz + cx \cdot sz & sx \cdot sz - cx \cdot sy \cdot cz & 0 & 0 & 0 \\ -cy \cdot sz & -sx \cdot sy \cdot sz + cx \cdot cz & sx \cdot cz + cx \cdot sy \cdot sz & 0 & 0 & 0 \\ sy & -sx \cdot cy & cx \cdot cy & 0 & 0 & 0 \\ 0 & 0 & 0 & cy \cdot cz & sx \cdot sy \cdot cz + cx \cdot sz & sx \cdot sz - cx \cdot sy \cdot cz \\ 0 & 0 & 0 & -cy \cdot sz & -sx \cdot sy \cdot sz + cx \cdot cz & sx \cdot cz + cx \cdot sy \cdot sz \\ 0 & 0 & 0 & sy & -sx \cdot cy & cx \cdot cy \end{bmatrix}$$

Where "s" and "c" indicate the sin and cosine operations respectively, and "x", "y" and "z" are the rotations about the x, y, and z axes respectively.

The displacement matrix is defined as:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & dz & -dy & 1 & 0 & 0 \\ -dz & 0 & dx & 0 & 1 & 0 \\ dy & -dx & 0 & 0 & 0 & 1 \end{bmatrix}$$

Where "dx", "dy", and "dz", are the displacements along the x, y, and z axes respectively. The units of the displacements must match the distance components of the torque units used in your calibration, e.g. if your sensor reports torque in Newton-meters, the displacements must be measured in meters.

Because matrix math is associative, you can pre-calculate the $[R]_{6x6}[D]_{6x6}[C]_{6x6}$ factor from Equation 1 and use the resulting matrix in all subsequent force/torque calculations, until you wish to create a new tool transformation.

It is important to note that the tool transformation procedure as described here applies the displacements and rotations in the following order:

1) Displacements. The order of axis displacements does not matter, as long as they are all done together with no rotations between displacements.
2) Rotation about X.
3) Rotation about Y.
4) Rotation about Z.

If you wish to apply the elements of the transform in a different order, you can either calculate the rotation and displacement matrices using the order appropriate to your application, or you can apply the algorithm as described above multiple times, and only apply one element of the transform (e.g. one rotation or one displacement) at each step, as in the following pseudocode:

```
RegularTransform( x:matrix, y:transform ) : matrix
/* Procedure described above. */

RemoveFirstTransform( y:queue of transforms ) : transform
/* Removes first element from y and returns that element, or returns a
null value if y is empty. */

OrderedTransform( x:matrix, y:queue of transforms ) : matrix
     z = RemoveFirstTransform( y )
     if ( null = z )
          return x
     else
          return OrderedTransform( RegularTransform( x, z ), y )
```

To use the above algorithm to apply transform elements in a particular order, you would construct a queue containing the transform elements in the order you wish to apply them, and then call the `OrderedTransform` method with the original calibration matrix and your transform list as arguments.

To read calibrated F/T data from the sensor:

1) Set the sensor up to use the correct RS-485 baud rate for your application. By default, the sensor runs at 1,250,000 baud. The sensor always uses 8 data bits plus an “even” parity bit.
2) Optionally set the session ID register to protect against unexpected reboots.
3) Read the calibration information for the calibration you wish to activate, and apply any tool transformations you wish to apply to the calibration matrix to create the “active calibration matrix”.
4) Send the “Unlock storage command” to unlock the active gain and offset hardware settings.

5) Write the active gain and offset hardware settings using the “GageGain” and “GageOffset” values from the calibration structure. You must do this every time the sensor is reset, because these values are not stored to nonvolatile memory.
6) Send the “unlock storage command” again to lock the gain and offset against accidental writing.
7) Send the “Start Streaming” command
8) When each streaming sample is received, check the status bit for system health and the checksum for sample validity. If the status bit is ‘1’, indicating a system error, go to step 11 to stop streaming, then read the system status word using standard Modbus.
9) To bias (tare) the sensor, take a reading when the sensor is in its default or “unloaded” state. Save these gauge values as the “bias vector”. For each subsequent sample, subtract the bias vector from the sampled strain gauge vector to remove the bias from the calculated F/T data.
10) Multiply the strain gauge vector (with the bias vector subtracted) by the active calibration matrix to calculate the F/T values. The resulting vector will contain the force and torque values, scaled by the counts per force and counts per torque reported in the calibration structure.
11) When you are ready to stop streaming, send a ‘jamming sequence’ of 14 bytes (one byte longer than the streaming sample size) to ensure the sensor receives at least one character. Wait before sending any more Modbus commands to allow the sensor’s internal Modbus stack to timeout and throw away the characters from the jamming sequence.