OPTOFORCE
SENSING FLEXIBILITY

# MANUAL

for

GENERAL DAQs

with

USB, UART or CAN

interface

Version 1.5

December, 2016

## TABLE OF CONTENTS

# 1. INTRODUCTION

There are three DAQ (Data Acquisition) types:

- Single-channel for 3-axis force sensor (*code name "31")*
- Multi-channel for 3-axis force sensors (*code name "34")*
- Single-channel for 6-axis force/torque sensor (*code name "64")*

All three are covered in this document. It is recommended to identify first which one applies to You.

There are three interface types:

- USB (CDC - virtual serial port)
- UART (3.3V TTL Rx/Tx with 1M Baud)
- CAN  (Standard CAN according to the ISO 11898)

Basically, all three using the same protocol (same bytes has to be transmitted and received in order to communicate with the device).

Both USB and UART are serial port with the following communication settings:

| | |
|---|---|
| Baud rate: | 1 000 000 |
| Stop bit: | 1 |
| Parity: | none |
| Data bits: | 8 |
| Flow control: | none |

In this case, obviously we can use the bidirectional communication to read and write to the device.

The CAN interface using two CAN channels for this purpose:

- 0x100 (default) for reading (Rx) to the device
- 0x101 (default) for writing (Tx) to the device

The CAN channel IDs (11bits) can be reconfigured at any time. The process of the reconfiguration will be discussed later.

⚠ This document covers the OptoForce proprietary protocols only.
*For Ethernet or EtherCAT EDS (Electronic Data Sheet) files please contact us.*

## 2. OPERATION

Once the device is powered-on it starts to cyclically (default rate is 100Hz) transfer a frame of

- 16 bytes for a single-channel 3 axis force sensor

| DATA packet bytes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Header (170,7,8,10) | | | | Sample Counter | | Status | | Fx | | Fy | | Fz | | Checksum | |

or

- 34 bytes for a multi-channel 3 axis force sensor (4 channel is supported)

| DATA packet bytes | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | ... | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| Header (170,7,8,28) | | | | Sample Counter | | Status | | Fx 1 | | Fy 1 | | Fz 1 | | ... | Fx 4 | | Fy 4 | | Fz 4 | | Checksum | |

or

- 22 bytes for a single-channel 6 axis force/torque sensor

| DATA packet bytes | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| Header (170,7,8,16) | | | | Sample Counter | | Status | | Fx | | Fy | | Fz | | Tx | | Ty | | Tz | | Checksum | |

The Force (or Torque) values are in dimensionless COUNTS that can be converted to N (or Nm) by using the values in the sensitivity report.

For example if the reading is 532 [Counts] from Fx and the sensitivity (Counts@N.C.) is 6100 and the Nominal Capacity (N.C.) for Fx is 150N, then

$$Fx\,[N] = Fx[Counts] / (Counts@N.C.) * (N.C.) = 532 / 6100 * 150N = 13.08N$$

The *Sample counter* is an UINT16 type value incremented each time the DAQ completes an internal sampling (fixed at 1 kHz).

The *Status* UINT16 indicates the current status of the sensor. *The detailed description will be discussed later.*

The *Checksum* (UINT16 ) can be used to check the integrity of the packet. The checksum is a sum of the preceding bytes starting from the header (170 + 7+ 8 + ……).

⚠ *Please note that the force (and torque) values are all signed INT16 values.*

## 3. CONFIGURATION

The sensor can be configured by sending a 9 byte CONFIGURATION packet.

CONFIGUARTION packet bytes

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| Header (170,0,50,3) | | | | Speed | Filter | Zero | Checksum | |

*For CAN interface, a full frame (8 byte) and a partial frame (1 byte) should be sent. If only full frames can be sent, please make sure that the last byte of the Checksum Word are sent as the first byte of the second full frame.*

The *Checksum* (UINT16 ) needs to be calculated according to the value of the Speed,Filter and Zero.

$$Checksum = 170 + 0 + 50 + 3 + Speed + Filter + Zero$$

The *Speed* byte sets the update speed.

| Speed (decimal) | Update frequency |
|---|---|
| 0 | Stops the transmission |
| 1 | 1000 Hz |
| 3 | 333 Hz |
| 10 | 100 Hz (default) |
| 33 | 30 Hz |
| 100 | 10 Hz |

The internal filtering can be configured by setting the Filter byte.

| Filter (decimal) | Cut-off frequency |
|---|---|
| 0 | No filtering |
| 1 | 500 Hz |
| 2 | 150Hz |
| 3 | 50 Hz |
| 4 | 15 Hz (default) |
| 5 | 5 Hz |
| 6 | 1.5 Hz |

In order to clear the sensors offset the sensor can be zeroed by setting the ZERO byte to 255 (decimal). It can be restored to the original values by setting it to 0.

*If the device has been set to zero and needs to be re-zeroed at a later time first send 0 as the ZERO byte wait at least 2ms and send 255 again.*

Example: In order to set 1000 Hz update rate and 500Hz cut-off frequency and cancel the offset (by zeroing) the 16 bytes to be sent should be the following:

*170  0  50  3  1  1  255   1  224*

⚠ *Please note that the configuration will reset on power reset.*

After sending any packet to the device (e.g.: the CONFIG packet) the device acknowledges with replying with the content of the error register (80). Therefore, the user can monitor whether the command was received or not.

If there were no error, you should receive **0** as the content of the error register:

170 0 80 1 **0** 0 251

*(the last two byte is the checksum of this packet)*

## 4. STATUS

The description of the STATUS word is the following:

**STATUS Word**

| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| ERROR | | OVERLOAD | | | | | | Multiple sensor selection | |
|---|---|---|---|---|---|---|---|---|---|
| **DAQ type** | **Sensor type** | **Fx** | **Fy** | **Fz** | **Tx** | **Ty** | **Tz** | **Single/Multiple** | **Sensor number** |
| 000 = No error<br>001 = DAQ error<br>010 = Communication error<br>011 = Reserved<br>100 = Reserved<br>101 = Reserved<br>110 = Reserved<br>111 = Reserved | 000 = No error<br>001 = The sensor has not been detected<br>010 = Sensor failure<br>100 = Temperature error<br>011 = Reserved<br>101 = Reserved<br>110 = Reserved<br>111 = Reserved | 0 = The axis has not been overloaded<br><br>1 = The axis has been overloaded | | | These are not used in case of force only sensors | | | 0 = Only a single sensor has error (or no error)<br><br>1 = Multiple sensors have error /This case, only the last sensor number has been indicated/ | 000 = No sensor has error<br>001 = Sensor #1<br>010 = Sensor #2<br>011 = Sensor #3<br>100 = Sensor #4<br>101 = Reserved<br>110 = Reserved<br>111 = Reserved |

Example: a decimal 514 equal to 0b0000001000000010 would imply an overload condition of axis Fx of the sensor #2 (channel 2).

## 5. CAN ID RECONFIGURATION

The CAN address can be changed anytime via the following two standard frames:

| CAN standard frames | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| IDs (11bit) | Length (bytes) | Frame number | BYTES | | | | | | | |
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Sensor_ID _ Rx | 8 | 1 | Header (170,0,60,8) | | | | Sensor_ID_Rx | | Sensor_ID_Tx | |
| | 6 | 2 | EEPROM (‘S’,’A’,’V’,’E’) | | | | Checksum | | | |

The new IDs can be given by the Sensor_ID_Rx and Sensor_ID_Tx (that can be identical if there is no available CAN address) as UINT16 variable.

In order to store the new IDs to the EEPROM of the DAQ, the EEPROM field of the packet must contain the ‘S’,’A’,’V’,’E’ ASCII characters (83,65,86,69) (otherwise the new ID will not be saved).

Once the new IDs are stored to the EEPROM a power cycling (power off the power on) needs for the change to take effect.

The *Checksum* (UINT16 ) needs to be updated according to the value of the Sensor_ID_Rx and Sensor_ID_Tx:

$$\begin{aligned} Checksum &= 170+0+60+8+HighByte(Sensor.ID.Rx)+LowByte(Sensor.ID.Rx) \\ &\quad + HighByte(Sensor.ID.Tx)+LowByte(Sensor.ID.Tx)+83+65+86+69 \end{aligned}$$

<u>Example</u>

To store a new ID of 0x103 for Tx and 0x104 for Rx the following (14)  bytes has to be sent:

170 000 060 008 001 004 001 003

083 065 086 069 002 038

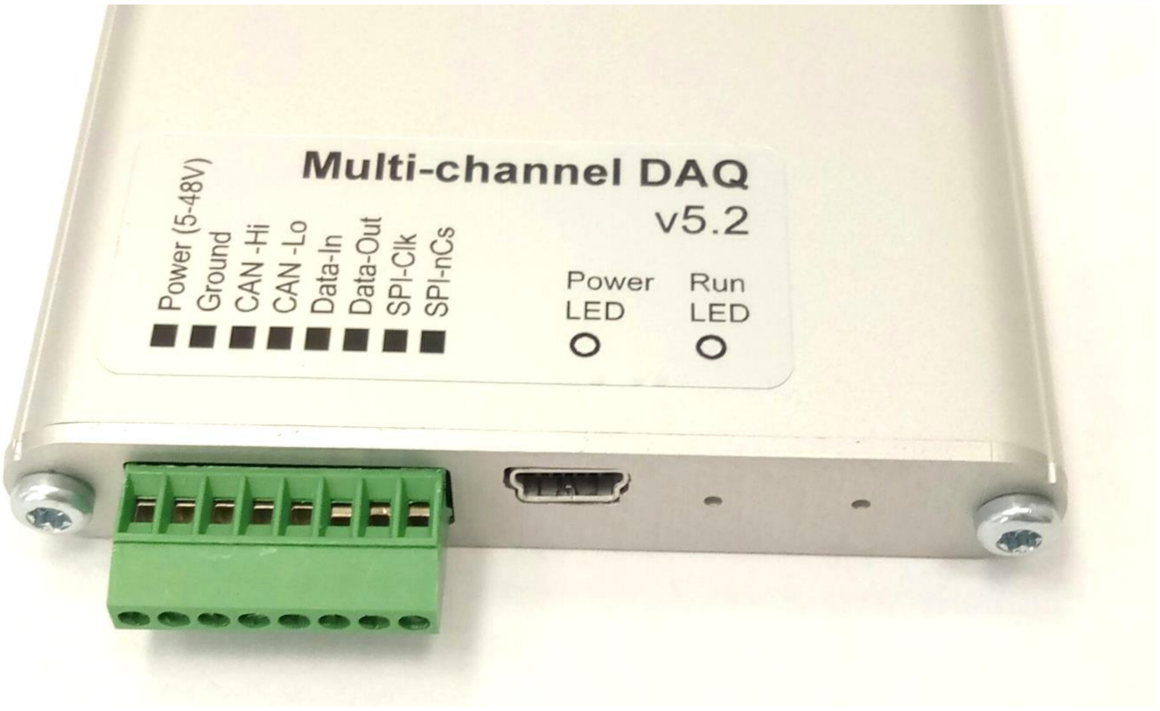
*All the numbers are decimal (the leading zeros are only shown for visualization).*

## 6. DIGITALIZER (DAQ)

**OptoForce** DAQ has a standard USB Mini-B plug that can be used to operate the DAQ in USB mode. The External Data Interface (8 pin) can be used to communicate via CAN or UART.

*The SPI interface (shares pins with the UART) is OBSOLOTE.*

The External Data Interface is the green 8 pin terminal block:



The description of the pin names is the following:

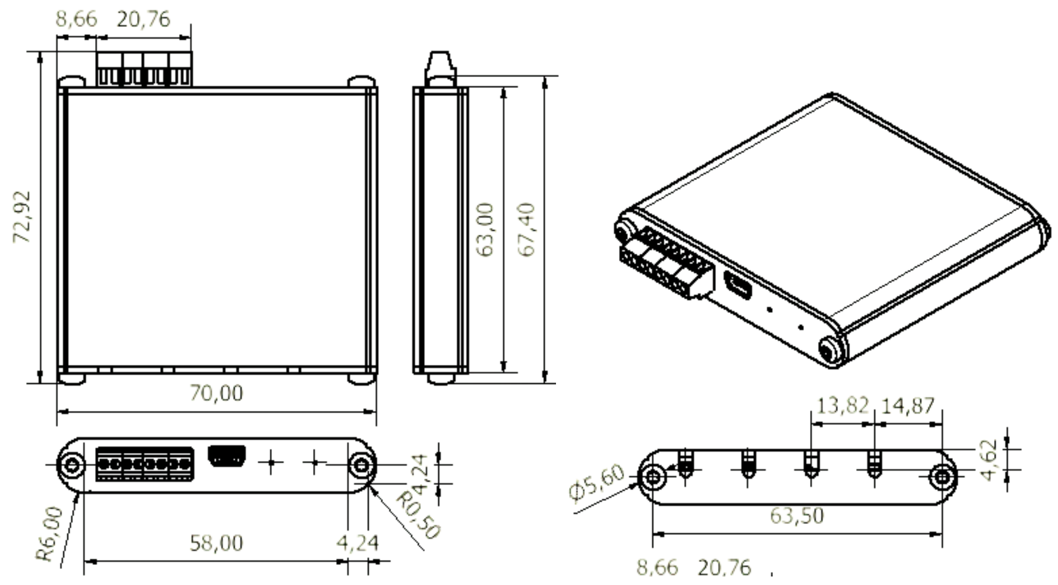
| Name | Description |
|---|---|
| POWER | This is the power supply pin for the DAQ if it is not powered via USB. The power supply voltage can be 5V-48V. |
| GROUND | The power and data signal ground pin. |
| CAN-HI | CAN bus High pin |
| CAN-LO | CAN bus Low pin |

| | |
|---|---|
| DATA-IN | In UART mode it is RX pin.<br>In SPI mode it is the Data input pin. |
| DATA-OUT | In UART mode it is TX pin.<br>In SPI mode it is the Data output pin. |
| SPI-CLK | In SPI mode it is the clock pin. |
| SPI-nCS | In SPI mode it is the Chip Select (active-low) |

Please note that the Rx and Tx signals are TTL 3.3V and can not tolerate any voltage above 3.6V or under -0.6V. *ANY VOLTAGE ABOVE 3.6V OR BELOW -0.6V CAN CAUSE DAMAGE TO THOSE PINS!!!* (Compared to the RS-232 that can have +/-12V signal voltage level.)

**The UART needs the following configuration:**

- **Baud rate: 1 000 000**
- **Stop bit: 1**
- **Parity: none**
- **Data bits: 8**
- **Flow control: none**



**In case of any question please contact me** [jozsef.veres@optoforce.com](mailto:jozsef.veres@optoforce.com) **or +36-70-502-9406**