Mini-DAQ Custom Configuration Guide

Shaun Fraser

August 2023

1 Introduction

This document serves as instruction on how to configure the mini-DAQ for channel setups other than the default 8-channel Analogue + 3 axis Acceleration setup. This could be of particular use if additional digital sensors are added or if more analogue channels are required: bearing in mind that with the current ADC spec on the breadboard 8 analogue channel is the hardware maximum. All required code files and supporting documents can be found on GitHub : `https://github.com/shaun11740/EtherCAT-Mini-DAQ`.

2 Overview of Configuration

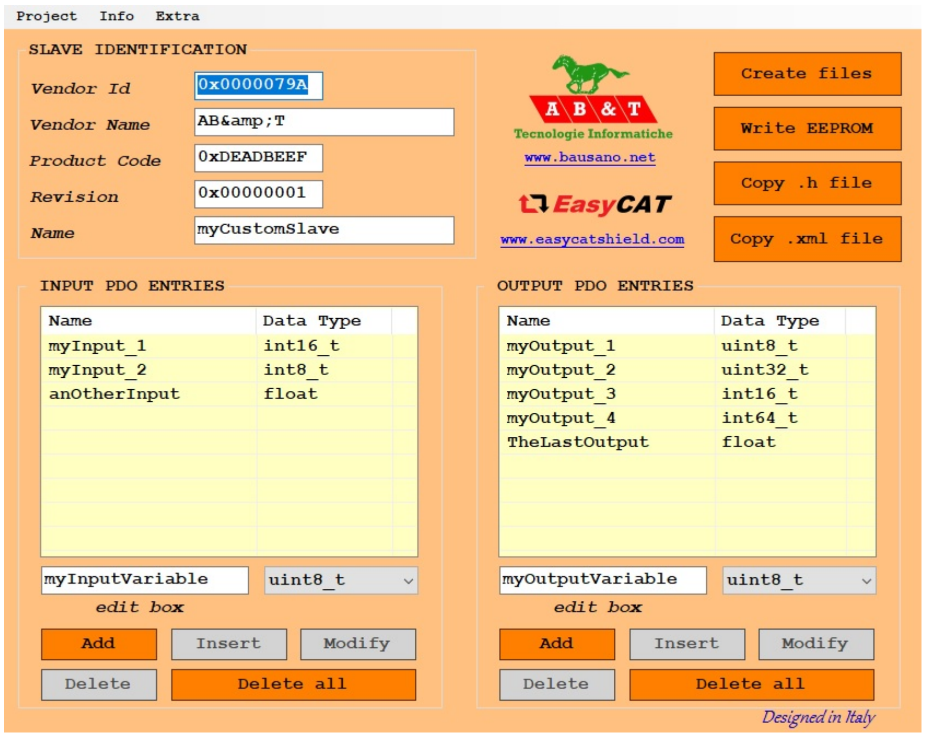
There are 3 main parts to re-configuring the Mini-DAQ:

1. Modifying Arduino code
2. Using EasyConfigurator program to create custom EtherCAT files
3. Setting up new EtherCAT network in LabVIEW

The EasyConfigurator comes with the EasyCAT board and allows the user to easily program the EEPROM of the EasyCAT board, aswell as create the .h file for the arduino program and XML file for the CompactRIO/LabVIEW project.

3 EasyConfigurator

The EasyConfigurator is the first point of call for redesigning the mini-DAQ I/O. It has a friendly GUI where inputs and outputs can be defined and configured. Figure 1 shows the EasyConfigurator. The bottom half of the interface shows the I/O configuration. You can add as many inputs and outputs as you want, setting the datatype for each (float, unsigned 16-bit integer, double etc). Remember that in this context for data acquisition we are only interested in inputs. Outputs only become relevant when we are looking at feedback and control where the CRIO is feeding information back to the slave device. Once all I/Os have been decided and added we can follow the steps to configure the EasyCAT.



SLAVE IDENTIFICATION

Vendor Id	0x0000079A
Vendor Name	AB&amp;T
Product Code	0xDEADBEEF
Revision	0x00000001
Name	myCustomSlave

INPUT PDO ENTRIES

Name	Data Type
myInput_1	int16_t
myInput_2	int8_t
anOtherInput	float

myInputVariable uint8_t

OUTPUT PDO ENTRIES

Name	Data Type
myOutput_1	uint8_t
myOutput_2	uint32_t
myOutput_3	int16_t
myOutput_4	int64_t
TheLastOutput	float

myOutputVariable uint8_t

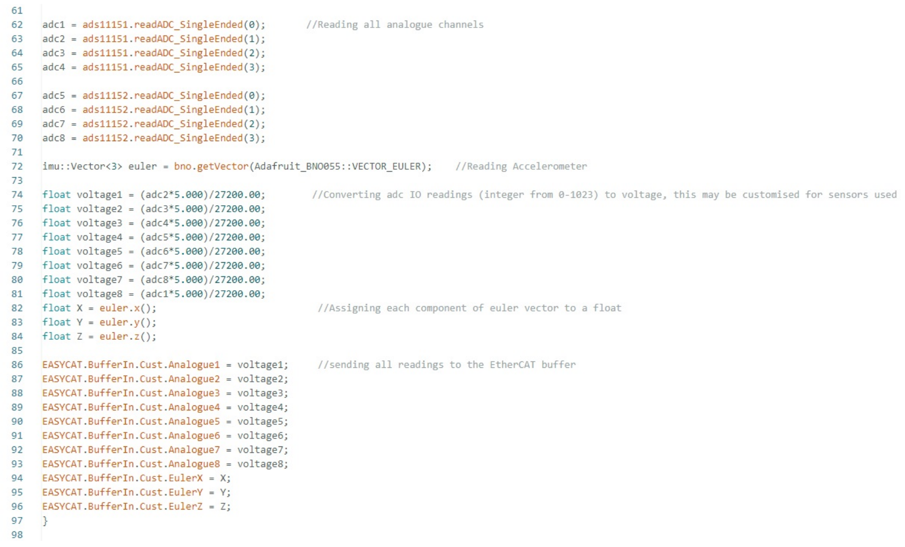
Figure 1: EasyConfigurator

1. Fill out the slave identification to the specification for EasyCAT board shown
2. Name your slave files, ideally the same as you will name the Arduino program
3. Click 'Create Files'
4. Plug EasyCAT board direcly into the network, then click Write EEPROM and select the .bin file you just created
5. Check that the .h file and .xml file are saved somewhere accessible

You are now finished with the EasyConfigurator.

4 Arduino Program

The goal with re-configuring the Arduino program is to create new variables with match the I/O variable types you made in the EasyConfigurator, then assign the sensor reading values to these variables and send them to the EtherCAT buffer. Figure 2 shows an example for the default 8-channel, 3-axis Accelerometer setup. Ensuring that the custom .h file is in the same directory as your



```
adc1 = ads11151.readADC_SingleEnded(0);       //Reading all analogue channels
adc2 = ads11151.readADC_SingleEnded(1);
adc3 = ads11151.readADC_SingleEnded(2);
adc4 = ads11151.readADC_SingleEnded(3);

adc5 = ads11152.readADC_SingleEnded(0);
adc6 = ads11152.readADC_SingleEnded(1);
adc7 = ads11152.readADC_SingleEnded(2);
adc8 = ads11152.readADC_SingleEnded(3);

imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);    //Reading Accelerometer

float voltage1 = (adc2*5.000)/27200.00;        //Converting adc IO readings (integer from 0-1023) to voltage, this may be customised for sensors used
float voltage2 = (adc3*5.000)/27200.00;
float voltage3 = (adc4*5.000)/27200.00;
float voltage4 = (adc5*5.000)/27200.00;
float voltage5 = (adc6*5.000)/27200.00;
float voltage6 = (adc7*5.000)/27200.00;
float voltage7 = (adc8*5.000)/27200.00;
float voltage8 = (adc1*5.000)/27200.00;
float X = euler.x();                            //Assigning each component of euler vector to a float
float Y = euler.y();
float Z = euler.z();

EASYCAT.BufferIn.Cust.Analogue1 = voltage1;     //sending all readings to the EtherCAT buffer
EASYCAT.BufferIn.Cust.Analogue2 = voltage2;
EASYCAT.BufferIn.Cust.Analogue3 = voltage3;
EASYCAT.BufferIn.Cust.Analogue4 = voltage4;
EASYCAT.BufferIn.Cust.Analogue5 = voltage5;
EASYCAT.BufferIn.Cust.Analogue6 = voltage6;
EASYCAT.BufferIn.Cust.Analogue7 = voltage7;
EASYCAT.BufferIn.Cust.Analogue8 = voltage8;
EASYCAT.BufferIn.Cust.EulerX = X;
EASYCAT.BufferIn.Cust.EulerY = Y;
EASYCAT.BufferIn.Cust.EulerZ = Z;
}
```

Figure 2: Arduino Configuration Example

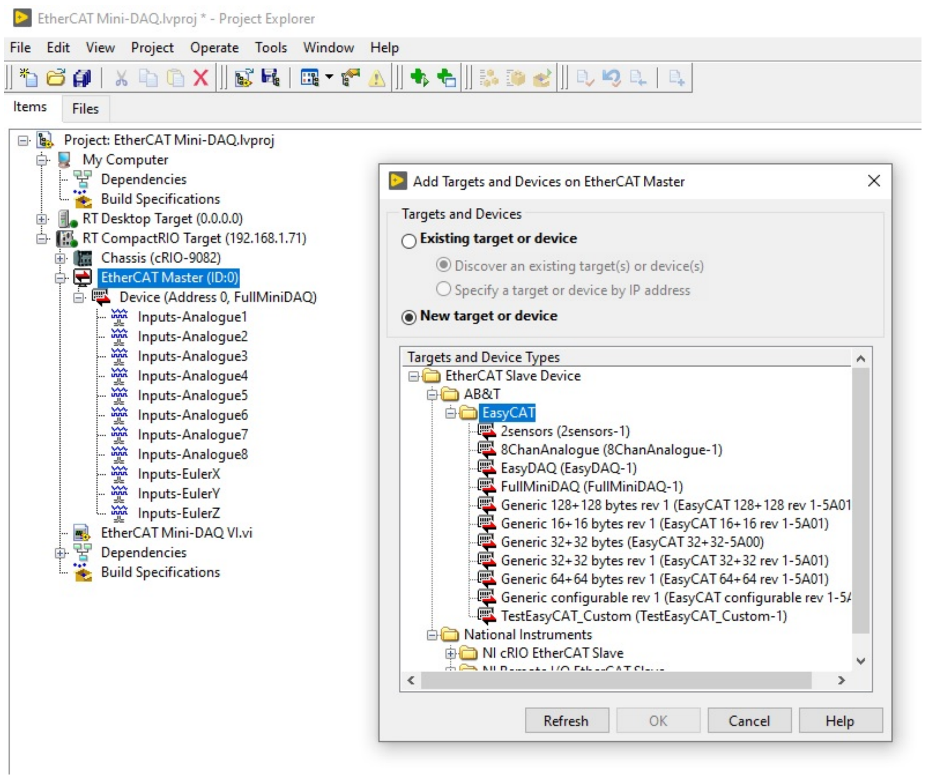
Arduino file, adding all new channel variables to the EtherCAT buffer using "$EASYCAT.BufferIn.Cust.VariableName = Variable$" should allow you to assign each variable to each input on the EtherCAT I/O. You should also make sure that the EasyCAT.h file remains in the directory.

5 LabVIEW Project

Configuring the LabVIEW project is the next step in setting up a new EtherCAT configuration. If you wish to start the LabVIEW setup from scratch or if you require more detailed explanation you can follow the instructions given by NI in this very helpful article `https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YHbbCAG&l=en-GB`. Otherwise it is recommended to simply edit the current LabVIEW project by adding a new XML task. The steps for achieving this are

1. Right click EtherCAT master device
2. Click new
3. Click targets and devices
4. Select custom XML file and add to the project

This should then allow you to select this task as the task running on the EtherCAT master with ID:0. I recommend reading the NI article for this step as it can take some time to understand. Figure 3 shows the project dialogue box you are looking for. You should now have a custom Mini-DAQ LabVIEW project.


EtherCAT Mini-DAQ.lvproj * - Project Explorer
File Edit View Project Operate Tools Window Help
Items Files
Add Targets and Devices on EtherCAT Master
Refresh OK Cancel Help

Project: EtherCAT Mini-DAQ.lvproj
- My Computer
  - Dependencies
  - Build Specifications
- RT Desktop Target (0.0.0.0)
- RT CompactRIO Target (192.168.1.71)
  - Chassis (cRIO-9082)
  - EtherCAT Master (ID:0)
    - Device (Address 0, FullMiniDAQ)
      - Inputs-Analogue1
      - Inputs-Analogue2
      - Inputs-Analogue3
      - Inputs-Analogue4
      - Inputs-Analogue5
      - Inputs-Analogue6
      - Inputs-Analogue7
      - Inputs-Analogue8
      - Inputs-EulerX
      - Inputs-EulerY
      - Inputs-EulerZ
  - EtherCAT Mini-DAQ VI.vi
  - Dependencies
  - Build Specifications

Targets and Devices
- Existing target or device
  - [✓ Discover an existing target(s) or device(s)]
  - Specify a target or device by IP address
- [✓ New target or device]

Targets and Device Types
- EtherCAT Slave Device
  - AB&T
    - EasyCAT
      - 2sensors (2sensors-1)
      - 8ChanAnalogue (8ChanAnalogue-1)
      - EasyDAQ (EasyDAQ-1)
      - FullMiniDAQ (FullMiniDAQ-1)
      - Generic 128+128 bytes rev 1 (EasyCAT 128+128 rev 1-5A01
      - Generic 16+16 bytes rev 1 (EasyCAT 16+16 rev 1-5A01)
      - Generic 32+32 bytes (EasyCAT 32+32-5A00)
      - Generic 32+32 bytes rev 1 (EasyCAT 32+32 rev 1-5A01)
      - Generic 64+64 bytes rev 1 (EasyCAT 64+64 rev 1-5A01)
      - Generic configurable rev 1 (EasyCAT configurable rev 1-5/
      - TestEasyCAT_Custom (TestEasyCAT_Custom-1)
  - National Instruments
    - NI cRIO EtherCAT Slave

Figure 3: LabVIEW Project Customisation

6 LabVIEW Virtual Instrument

Finally to modify the DAQ Virtual Instrument you must do the follwoing:

1. Open VI back panel
2. Add shared variable blocks for each input channel (see figure 4)
3. Right click each shared variable and select variable for correct input (see figure 5)
4. Wire each channel to both a waveform chart and to a combined signal which is input to the Write to Measurement File Express VI (see figure 6)

Now the custom mini-DAQ should be fully configured and ready to collect data. If any problems arise, consult the other two documents on mini-DAQ: Overview and How-to-Use.

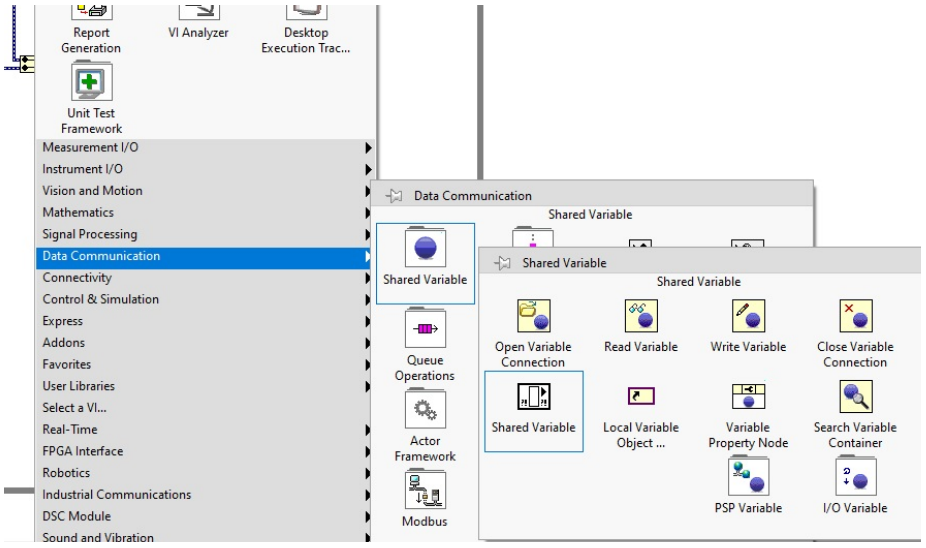


Figure 4: Adding shared variable

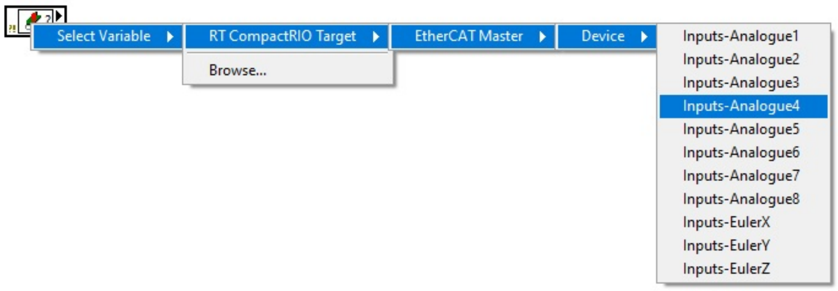


Figure 5: Choosing variable channel

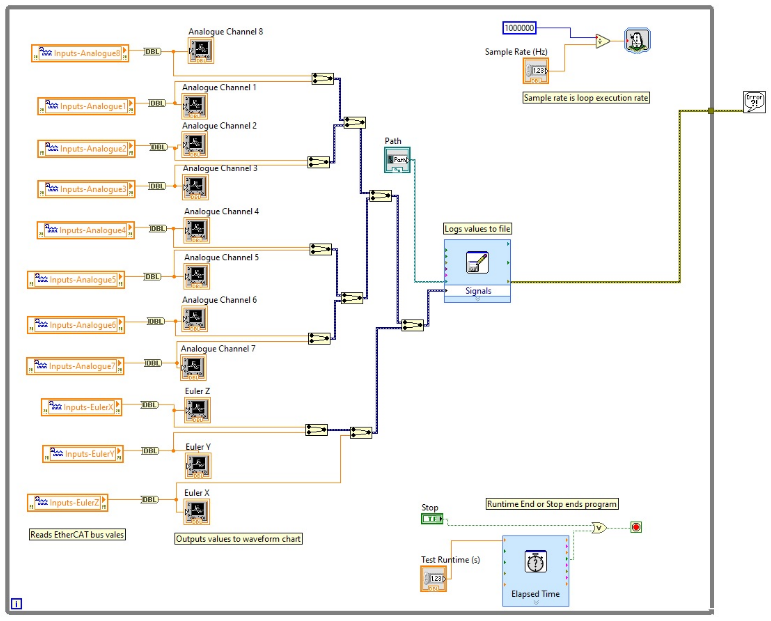


Figure 6: Finished Back Panel