USER MANUAL

MDS-ACCT

Modular Digital Solution for ACCT

Rev. 1.0

www.bergoz.com

bergoz INSTRUMENTATION

More than 40 years of experience recognized in the world of particle accelerators

Record of updates

Version	Date	Updates performed
1.0	02/04/2026	First release

DISTRIBUTORS

U.S.A.

GMW Associates
www.gmw.com
sales@gmw.com

Japan

Hayashi-Repic Co., Ltd.
www.h-repic.co.jp
sales@h-repic.co.jp

India

GEEBEE International
www.geebeeinternational.com
info@geebeeinternational.com

China

Beijing Conveyi Limited
www.conveyi.com
sales@conveyi.com

South Korea

Seyoung Co., Ltd
www.seyoungsys.com
apark@seyoungsys.com

TABLE OF CONTENTS

INITIAL INSPECTION

It is recommended that the shipment be inspected immediately upon delivery. If it is damaged in any way, contact Bergoz Instrumentation or your local distributor. The content of the shipment should be compared to the items listed on the invoice. Any discrepancy should be notified to Bergoz Instrumentation or its local distributor immediately. Unless promptly notified, Bergoz Instrumentation will not be responsible for such discrepancies.

WARRANTY

Bergoz Instrumentation warrants its beam current monitors to operate within specifications under normal use for a period of 12 months from the date of shipment. Spares, repairs and replacement parts are warranted for 90 days. In exercising this warranty, Bergoz Instrumentation will repair, or at its option, replace any product returned to Bergoz Instrumentation or its local distributor within the warranty period, provided that the warrantor's examination discloses that the product is defective due to workmanship or materials and that the defect has not been caused by misuse, disassembly, neglect, use of faulty part, accident or abnormal conditions, repair made by the customer, or operations. Damages caused by ionizing radiations are specifically excluded from the warranty. Bergoz Instrumentation and its local distributors shall not be responsible for any consequential, incidental or special damages.

ASSISTANCE

Assistance in installation, use or calibration of Bergoz Instrumentation beam current monitors is available from Bergoz Instrumentation, 01630 Saint Genis Pouilly, France. It is recommended to send a detailed description of the problem by email to info@bergoz.com.

SERVICE PROCEDURE

Products requiring maintenance should be returned to Bergoz Instrumentation or its local distributor: The purchaser/customer must ask for a RMA (Return Material Authorization) number to Bergoz Instrumentation or its local distributor before return of goods. Bergoz Instrumentation will repair or replace any product under warranty at no charge.

For products in need of repair after the warranty period, Bergoz Instrumentation will assess the technical issue and send a quote to the purchaser/customer. The purchaser/customer must provide a purchase order before repairs can be initiated. Bergoz Instrumentation can issue fixed price quotations for most repairs.

RETURN PROCEDURE

All products returned for repair should include a detailed description of the defect or failure as well as name, phone number and email of a contact person to allow further inquiry. Contact Bergoz Instrumentation or your local distributor to determine where to return the product. Returns must be notified by email prior to shipment.

The shipment of a product under warranty or out of warranty back to the factory is paid by the user/customer, including the customs fees. The return of this repaired product under warranty back to the customer is paid by Bergoz Instrumentation.

Return of product out of warranty should be made prepaid or will be invoiced. Bergoz Instrumentation will not accept freight-collect shipments. Shipments should be made via UPS, FedEx or DHL. Within Europe, the transportation services offered by the national Post Offices can be used. The delivery charges or customs clearance charges arising from the use of other carriers will be charged to the customer.

SAFETY INSTRUCTIONS

The Toroid sensor contains materials such as cobalt and iron. Those materials may become radioactive when exposed to high energy particle beams. Follow applicable radiation-safety procedures when the Toroid sensor must be handled.

GENERAL DESCRIPTION

The MDS-ACCT is a dedicated digitizer module specifically designed to interface with up to two ACCTs from Bergoz Instrumentation.

Unlike general-purpose digitizers, the MDS-ACCT is a fully integrated solution that completely handles the ACCTs. It features resolution, bandwidth, and sampling rate perfectly matched to the characteristics of ACCT signals and can directly convert the measured voltages waveforms into beam current waveforms.

In addition to digitizing the signals delivered by the ACCTs and transmitting them over Ethernet, the module also allows switching of the measurement range on ACCT-E-3R units.

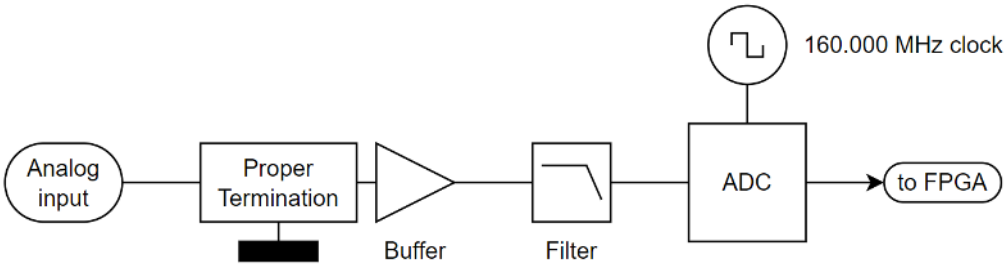
MDS-ACCT in its tabletop chassis (MDS-TTC)

The acquired waveforms are transmitted over Gigabit Ethernet using the UDP/IP protocol, enabling unicast, multicast, and broadcast distribution.

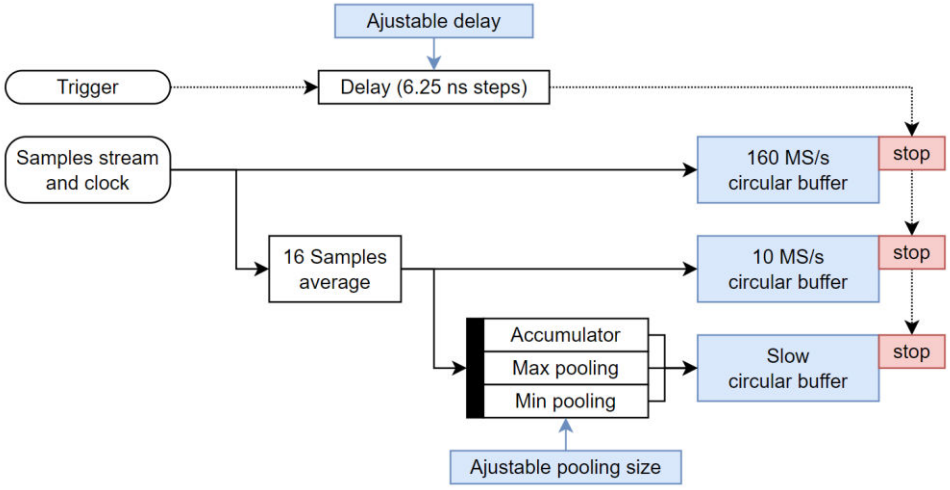
Since two ACCTs can be connected to a single MDS-ACCT, you can easily measure the current or charge of your beam at two different locations. This allows straightforward evaluation of the beam loss in the section between the two ACCTs.

SIMPLIFIED DIAGRAM AND OPERATING PRINCIPLE

Signals from the ACCT-E are first processed by analog electronics before reaching the analog-to-digital converter:



Next, the samples digitized by the ADC are transmitted to an FPGA, which is responsible for digital processing and storage in three different buffers at different sampling rates:



Finally, a CPU handles data processing and formats all acquired data points into UDP/IP packets, which are transmitted over Ethernet.

SPECIFICATIONS

Two main inputs

Resolution	16 bits
Range	±1 V, or ±10 V with -HZ option
Impedance	50 Ohm, or 10 kOhm with -HZ option
Signal Noise Ratio	72 dB typ. at 160 MS/s (more at lower sampling rates)
Temperature drift	200 ppm/°C typ.
Bandwidth	DC - 3 MHz (± 0.1 dB) typ. DC - 30 MHz (-3 dB) max

Trigger

Modes	Dedicated rear SMA	IN1
Trigger level HIGH	2 V to 5 V (50 ohm)	Configurable
Trigger level LOW	0 V to 0.6 V (50 ohm)	Configurable

Trigger edge	Rising
Minimum trigger length	1 µs
Timestamp counter	64 bits, based on the sampling clock (6.25 ns resolution)

Sampling Clock

Source	Internal
Main clock frequency	160.0000 MHz
Stability	±10 ppm
Aging	±5 ppm (1 year at 25°C nom.)
Period Jitter	3 ps rms
Integrated Phase Noise	2 ps rms

Ethernet

Connector	RJ-45
Speed	Auto-negotiating up to 1 GbE
IP address	IPv4, DHCP with configurable fallback IP
UDP ports	Receive: 5005, Transmit: 61483 (configurable)

Power supply

Input voltage	100 – 240 VAC
AC frequency	50 / 60 Hz
Fuse	2x 800 mA, slow blow, 5x20 mm (1 MDS station) 2x 1.6 A, slow blow, 5x20 mm (2 MDS station) 2x 2.4 A, slow blow, 5x20 mm (3 MDS station) 2x 3.2 A, slow blow, 5x20 mm (4 MDS station)

Connectors

Rear coaxial	SMA	500 cycles
Rear power	IEC C14	50 000 cycles
Front Ethernet	RJ-45	750 cycles

Other

Debug terminal connector	Front panel USB-C
Debug terminal protocol	Virtual COM port, 115 200 baud
Operating ambient temperature	0°C to 35°C (MDS-TTC tabletop) 0°C to 45°C (MDS-RFC well-ventilated 19" rackmount)
Recommended warmup time	90 minutes

ELECTRICAL CONNECTIONS

Front Panel

Picture	Description
	USB-C debug terminal. Acts as a serial COM port (115 200 baud) and displays information about the MDS configuration and setup.
	RJ-45 Gigabit Ethernet connector.
	Reset button. Press it gently, for example, with a tooth-pic for at least 1s to hard reset the MDS.
	MDS Status LED. ○ White: Powered and booting. Yellow: Has booted and is initializing. Green: Initialized properly, waiting for a trigger. Blue Cyan: Has already acquired at least one waveform. (the color switches with each trigger) No light: Power issue. Red: Unexpected Issue. Please reset and check the debug terminal.
	Power Status LED. Must always be ○ white after power-up. Power-up sequence: Red -> Yellow -> ○ White.

Rear Panel

Label	Description
100- 240 VAC	AC power input. Protected with one fuse on line and one fuse on neutral.
IN 1 IN 2	MDS inputs. Connect ACCT analog electronics to these inputs with 50 Ohm BNC to SMA coaxial cables. Nominal voltage: -1V to +1V in 50 Ohm (-10V to +10V in high impedance with -HZ option) Absolute maximum voltage: +- 10 % of the nominal voltage.
DB9	Sub-D9 connector used to change the ranges of the ACCT-E-3R.
RESET	If the MDS crashes due to radiation event or unusual EMI, you can try to reset it by injecting a small current in this pin for at least 1 second. Required voltage level: +1 V to +5 V
TRIGGER	Trigger input. Each pulse sent in this input triggers an acquisition which is processed and sent via the Ethernet port. Nominal voltage: +3.3 V pulses in 50 Ohm Pulse length: At least 1 µs.
a b c d	General purpose input and output. Used only for custom software modification provided by Bergoz Instrumentation. If you are interested in such a modification, please contact us at info@bergoz.com.

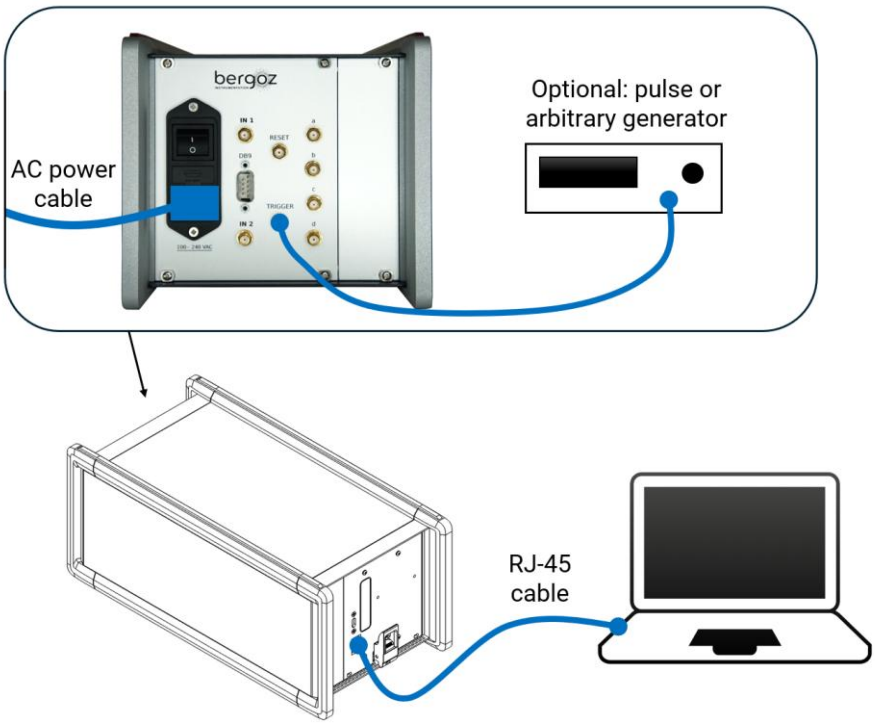
QUICK START GUIDE - PART 1

This guide will show you how to verify that the MDS is starting correctly, and how to observe raw data packets.

What you will need

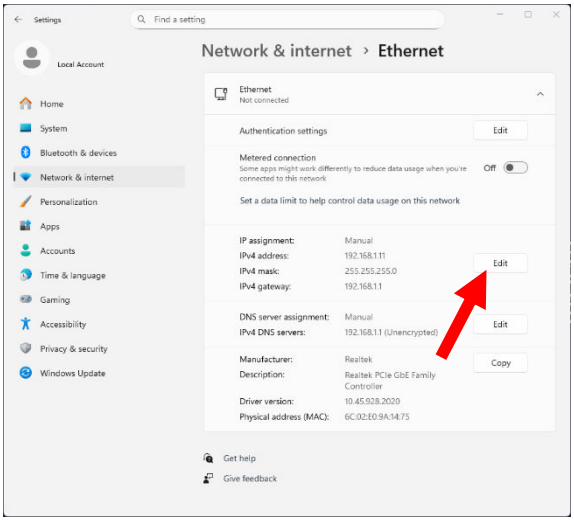
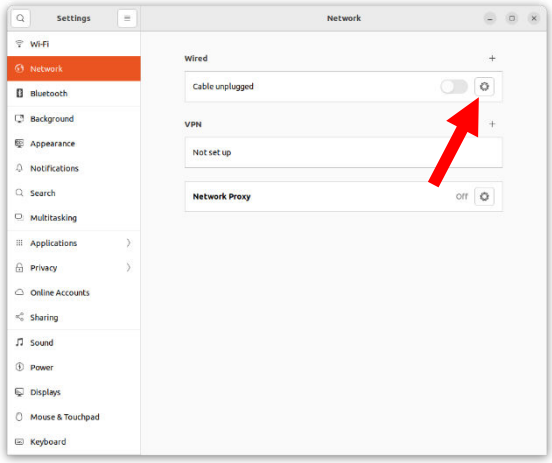
- A computer on which you can change the Ethernet settings.
- One RJ-45 Ethernet cable.
- Wireshark, which is a network protocol analyzer software.
  You can download it for free on its official website: wireshark.org.
  During installation, leave all settings at their default values and allow the installation of ncap.
- Optional but recommended: A pulse (or arbitrary) generator, with a BNC to SMA cable.

Wiring diagram

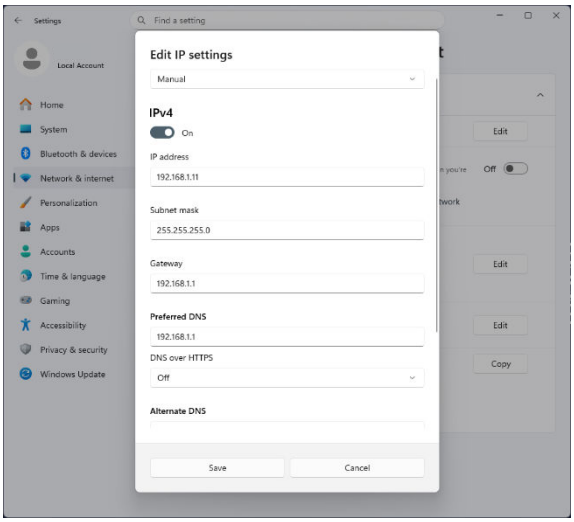
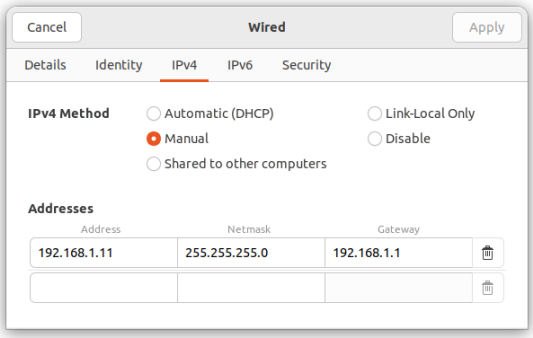


Ethernet setup

Open the computer settings, go to the "network" tab, then "Ethernet" tab, then press the following button:

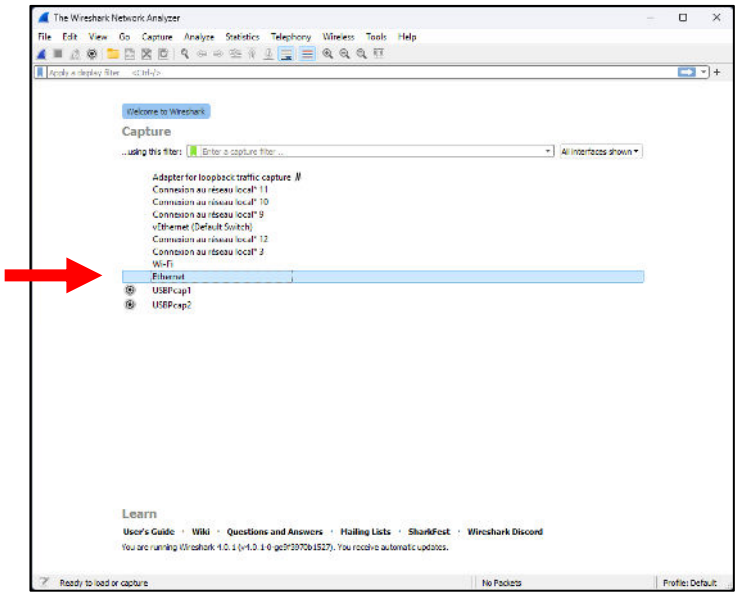
Windows	Ubuntu
	

Configure the IPv4 address in "Manual" mode, with these settings:

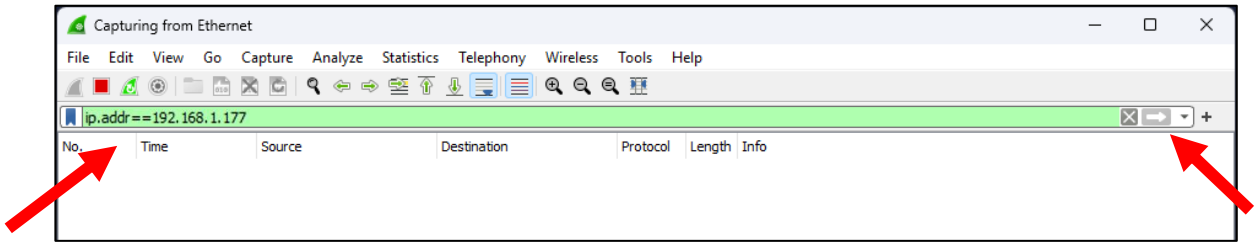
Windows	Ubuntu
	
IP address: 192.168.1.11 Subnet mask: 255.255.255.0 Gateway: 192.168.1.1 Preferred DNS: 192.168.1.1	Address: 192.168.1.11 Netmask: 255.255.255.0 Gateway: 192.168.1.1

Wireshark setup

Open Wireshark, then double-click on "Ethernet" or "Any" (either one will be visible depending on your OS. You may need to open Wireshark as an administrator to see all interfaces):



On the next screen, enter "ip.addr==192.168.1.177" in the filter bar (192.168.1.177 is the default IP address of every MDS), then click on the arrow or press Enter:



Wireshark is now ready to capture all data packets coming from the MDS.

Power-up

Set the power switch on the MDS chassis to the "I" position (rear panel).

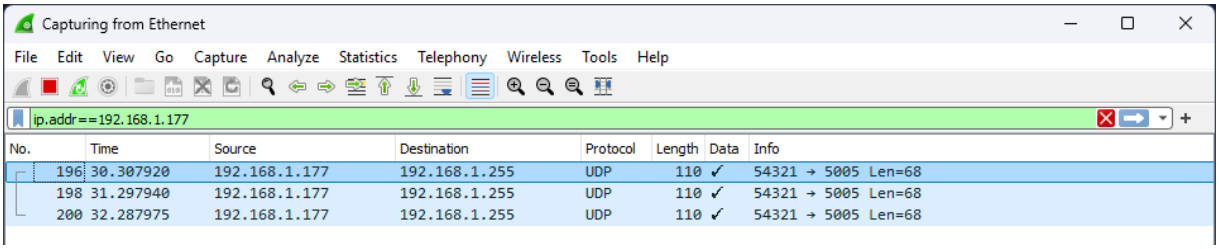
On the front panel, the "Power Status" LED follows this sequence:

	Red -> Yellow -> White

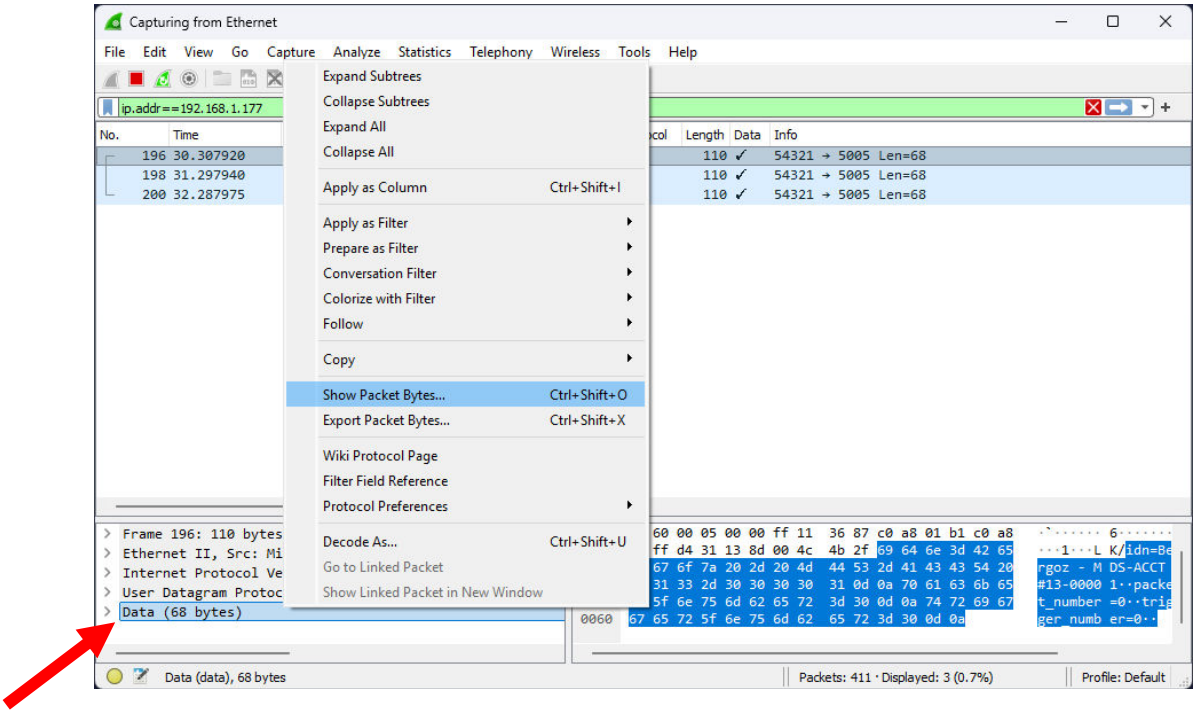
Then, the "MDS Status" LED follows this sequence:

	White -> Yellow (for up to 1 minute) -> Green

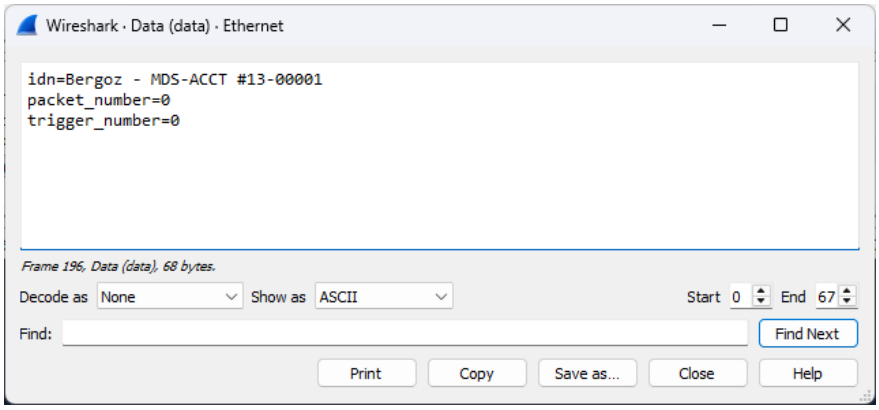
At the same time, three packets are captured by Wireshark:



Left click on a packet, then right click on "data" at the bottom left part of the window, and finally left click on "Show Packet Bytes":



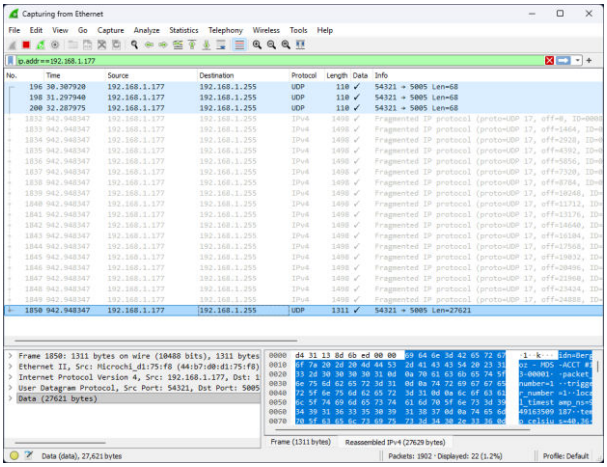
You should normally see text like this:

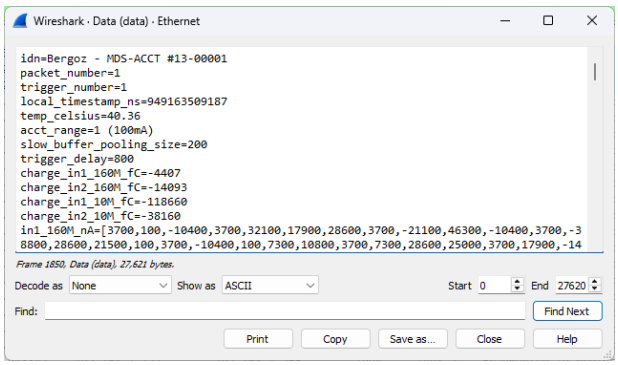


Congratulation, your MDS seems to be working!

Each time the MDS is powered up, these same three packets are sent automatically. They are used to signal the presence of the device on the network.

You can now use a generator to send a pulse to the "TRIGGER" input with these settings: [Impedance = 50 Ohms, Vlow = 0 V, Vhigh = 2V, T_{high} = 1 µs, 1 Hz].
The "MDS Status" LED will then change color, and Wireshark will capture one packet per trigger:





ETHERNET PROTOCOL

For each trigger, the MDS will send one UDP packet to the IP address defined in a configuration file on the microSD card (more info in the dedicated chapter). These packets look like this:

```
idn=MDS-ACCT #18
packet_number=226
trigger_number=226
…
in1_10M_nA=[123, 456, 789, 654, 321, …
```

By default, the MDS-ACCT tries to use DHCP to get an IP address. If an error occurs, or if you configure it to not use DHCP, the following default IP configuration is used:

- MDS IP = 192.168.1.177
- Netmask = 255.255.255.0
- Gateway = 192.168.1.1

The default client IP address (to which the acquisitions are sent) is set to 192.168.1.255 (broadcast). The default UDP/IP destination port is 61483. IP configuration and client IP and port can be configured on the microSD card.

Packets content description

General information	
idn=*Bergoz - MDS-ACCT #13-00001*	A unique identifier used to distinguish packets coming from different MDS units. String
packet_number=*42*	Number of UDP/IP packets sent. Incremented by 1 for each transmission. Unsigned int 32
trigger_number=*42*	Trigger number for the acquisition. Should normally match the packet number, unless the MDS received triggers faster than it can process the acquisitions. Unsigned int 32
local_timestamp_ns=*105...*	MDS local timestamp, in nanoseconds since power-up. (Based on a 160.0000 MHz clock, ±10 ppm). Unsigned int 64
temp_celsius=*35.24*	MDS ADC's board temperature. Float
acct_range=*1 (100mA)*	Current ACCT-E-RM-3R range readback. String
slow_buffer_pooling_size=*200*	Number of consecutive raw samples grouped into one slow buffer sample. Effective slow buffer sampling rate = 10 MS/s ÷ this value. Slow buffer processing depends on waveform type: • Current (nA): samples are averaged. • Raw ADC: min, max, and accumulation are computed over each group. This parameter is displayed only if at least one slow buffer transition is enabled. Unsigned int 16
trigger_delay=*800*	Trigger delay in 6.25 ns steps. If set to 0, the acquisition circular buffer stops at the time of the trigger. Example: If set to 800, the circular buffers will also contain 800 * 6.25 ns = 5 µs of signal after the trigger. Unsigned int 32

Computed data	
charge_in1_160M_fc=*123456* **charge_in2_160M_fc**=*-123*	Pulse charge computed using samples from the 160 MS/s buffer of input IN1 (or IN2). The value is in fC. For the value to be accurate, the pulse must be within the [10%, 100%] timing interval of the waveform. int 64
charge_in1_10M_fc=*123456* **charge_in2_10M_fc**=*-123*	Pulse charge computed using samples from the 10 MS/s buffer of input IN1 (or IN2). The value is in fC. For the value to be accurate, the pulse must be within the [5%, 95%] timing interval of the waveform. int 64

For more information on how charges are calculated, please refer to the section titled “Note on the automated charge measurement” at the end of this manual.

Beam Current Waveforms	
in1_160M_nA=*[123456,...]* **in2_160M_nA**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at 160 MS/s and converted to nA. Array of int 32
in1_10M_nA=*[123456,...]* **in2_10M_nA**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at 10 MS/s and converted to nA. Array of int 32
in1_slow_nA=*[123456,...]* **in2_slow_nA**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at low sampling rate and converted to nA. Array of int 32

ACCT Output Voltage Waveforms	
in1_160M_uV=*[123456,...]* **in2_160M_uV**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at 160 MS/s and converted to µV. Array of int 32
in1_10M_uV=*[123456,...]* **in2_10M_uV**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at 10 MS/s and converted to µV. Array of int 32
in1_slow_uV=*[123456,...]* **in2_slow_uV**=*[123456,...]*	Waveform of the IN1 (or IN2) input acquired at low sampling rate and converted to µV. Array of int 32

ADC Raw Waveforms	
in1_160M_raw=*[12345,...]* **in2_160M_raw**=*[12345,...]*	Raw ADC waveform from input IN1 (or IN2) acquired at 160 MS/s. Array of unsigned int 16
in1_10M_raw=*[12345,...]* **in2_10M_raw**=*[12345,...]*	Raw ADC waveform from input IN1 (or IN2) acquired at 10 MS/s. Array of unsigned int 16
in1_slow_raw_acc=*[123456,...]* **in2_slow_raw_acc**=*[123456,...]*	Accumulation-pooling raw ADC waveform from input IN1 (or IN2) acquired at slow sampling rate. Divide each sample by "**slow_buffer_pooling_size**" to obtain the average-pooling raw ADC waveform. Array of unsigned int 32
in1_slow_raw_min=*[12345,...]* **in2_slow_raw_min**=*[12345,...]*	Min-pooling raw ADC waveform from input IN1 (or IN2) acquired at slow sampling rate. Array of unsigned int 16
in1_slow_raw_max=*[12345,...]* **in2_slow_raw_max**=*[12345,...]*	Max-pooling raw ADC waveform from input IN1 (or IN2) acquired at slow sampling rate. Array of unsigned int 16

Supported configuration messages

Some settings on the MDS-ACCT can be modified remotely via UDP/IP communication. Other settings not documented here must be modified on the MDS's microSD card (see later in this user manual).

In the current firmware version (beta 1), two settings can be changed remotely:

- Range of the connected ACCT-E-3R (no effect for single ranges)
- Trigger delay

To change these settings, send the following messages via UDP/IP to the MDS-ACCT's IP address on port 5005:

Change the ACCT-E-RM-3R measurement range	
Message to send	range=**X**
Details	**X** must be 1, 2 or 3
Exemple	range=2

Modify trigger delay	
Message to send	trigger_delay=**X**
Details	**X** must be an integer between 0 and 2 000 000 000. **X** corresponds to the delay between the rising edge of a trigger and the end of an acquisition, in steps of 6.25 ns. For example, to stop the acquisition 5 µs after the rising edge of the trigger, you must set **X** = 5E-6 / 6.25E-9 = 800
Exemple	trigger_delay=800

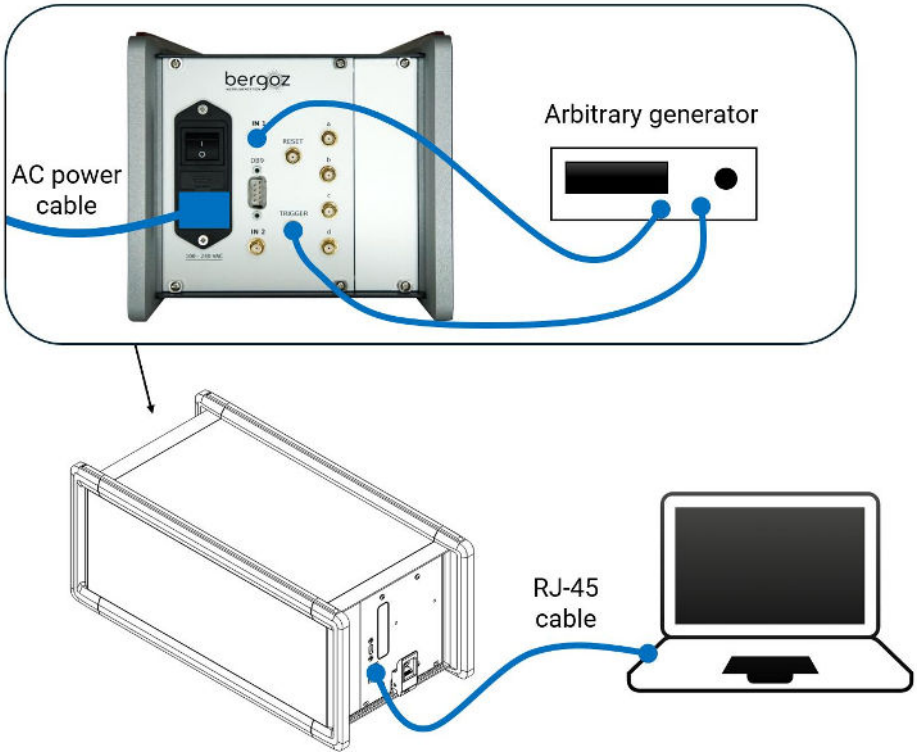
QUICK START GUIDE - PART 2

This second guide will allow you to observe real signals transmitted by the MDS using a Python program.

What you will need

- Ideally, you should be comfortable with Python and know how to install libraries using PIP. If not, we recommend asking someone who is familiar with it for help, or learning how to do so before continuing (there are many online guides available).
- Python 3.X installed, with the Numpy and Matplotlib libraries.
- Have followed the "Quick Start Guide – Part 1" and have your setup in the same state as at the end of that first part.
- Arbitrary signal generator with two synchronizable outputs, as well as two BNC-to-SMA coaxial cables.

Wiring diagram



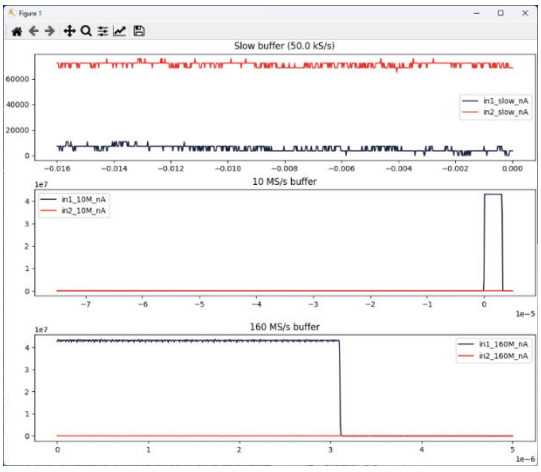
Copy and paste the program provided later in this manual into a new Python program, then run it (PROGRAMMING EXAMPLES -> Plotting a graph in Python).

Next, set up the arbitrary waveform generator as follows:

	Output 1	**Output 2**
MDS connector	IN1	TRIGGER
Impedance	50 Ohm (if no option) Hz (if -HZ option)	50 Ohm
Waveform	Pulse	Pulse
Pulse length	3 µs	1 µs
Rise & fall time	320 ns	320 ns
Voltage low	0 V	0 V
Voltage High	0.1 V (if no option) 1 V (if -HZ option)	2 V
Frequency	0.1 Hz	0.1 Hz
	Synchronized	

These settings simulate the output signal of an ACCT-E when a current pulse equal to 10% of the measurement range passes through the ACCT.

Normally, each time a trigger occurs, a window like the one below will open:



This window displays the acquired waveforms. The pulse should have an amplitude of approximately 10 % of the ACCT range (the vertical axis is scaled in nA). You must close the window to see the next acquisition.

This Python program is intentionally kept simple to make it easier to understand. Please note that it may fail if the microSD card's configuration differs significantly from the factory settings.

PROGRAMMING EXAMPLES

Plotting a graph in Python

This program waits to receive a packet from the MDS, then displays it.

```
import socket, ast
import numpy as np
import matplotlib.pyplot as plt

LSTEN_PORT = 61483

def parse_packet_to_dictionary(mds_packet:str):
    new_dictionary = {}

    for mds_packet_line in mds_packet.split("\n"):
        if mds_packet_line.count("=") == 1:
            mds_packet_line = mds_packet_line.strip() # clean the line
            parameter, value = mds_packet_line.split("=")
            try:
                value = ast.literal_eval(value)
            except:
                pass
            new_dictionary[parameter] = value

    return new_dictionary

def make_a_beautiful_graph(parsed_packet:dict):
    figure, axes = plt.subplots(ncols=1, nrows=3, figsize=(10, 8), layout="constrained")

    waveformes_length = len(parsed_packet["in1_slow_nA"])
    trigger_delay = parsed_packet["trigger_delay"] * 6.25e-9

    slow_buffer_sampling_rate = 10e6 / parsed_packet["slow_buffer_pooling_size"]
    time_scale_slow = np.linspace(-waveformes_length / slow_buffer_sampling_rate + trigger_delay, 0.0 +
trigger_delay, waveformes_length)
    axes[0].plot(time_scale_slow, parsed_packet["in1_slow_nA"], color="#19213c", label="in1_slow_nA")
    axes[0].plot(time_scale_slow, parsed_packet["in2_slow_nA"], color="#e3352d", label="in2_slow_nA")
    axes[0].set_title(f"Slow buffer ({slow_buffer_sampling_rate/1e3:.01f} kS/s)")
    axes[0].legend()

    time_scale_10M = np.linspace(-waveformes_length / 10e6 + trigger_delay, 0.0 + trigger_delay,
waveformes_length)
    axes[1].plot(time_scale_10M, parsed_packet["in1_10M_nA"], color="#19213c", label="in1_10M_nA")
    axes[1].plot(time_scale_10M, parsed_packet["in2_10M_nA"], color="#e3352d", label="in2_10M_nA")
    axes[1].set_title("10 MS/s buffer")
    axes[1].legend()

    time_scale_160M = np.linspace(-waveformes_length / 160e6 + trigger_delay, 0.0 + trigger_delay,
waveformes_length)
    axes[2].plot(time_scale_160M, parsed_packet["in1_160M_nA"], color="#19213c", label="in1_160M_nA")
    axes[2].plot(time_scale_160M, parsed_packet["in2_160M_nA"], color="#e3352d", label="in2_160M_nA")
    axes[2].set_title("160 MS/s buffer")
    axes[2].legend()

    plt.show()


sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("0.0.0.0", LSTEN_PORT))
print("Ready to listen !")

while True:
    # Wait to receive an UDP/IP packet
    raw_data, sender_addr = sock.recvfrom(65000)
    ascii_data = raw_data.decode()

    parsed_packet = parse_packet_to_dictionary(ascii_data)

    make_a_beautiful_graph(parsed_packet)
```

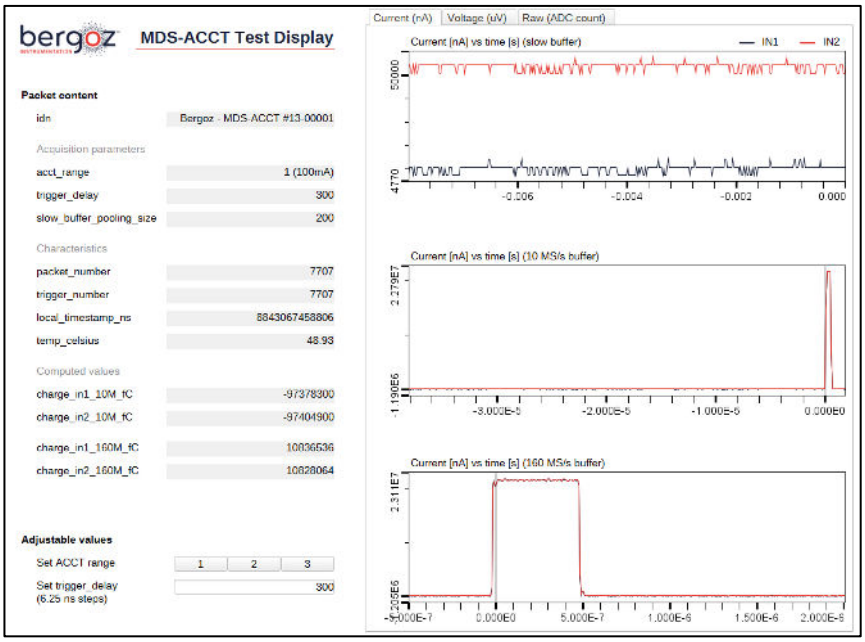
Capture and record packets in C

This program listens for packets from the MDS. When it receives one, it saves it to a new file in the “acquisitions” folder and resumes listening. To stop it, press “Ctrl+C”.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/time.h>
#include <time.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 61483
#define BUFFER_SIZE 65000

void write_to_file(const char *data, ssize_t len) {
    struct timeval tv;
    gettimeofday(&tv, NULL);

    struct tm *tm_info = localtime(&tv.tv_sec);
    char filename[128];

    // Format : YYYYMMDD_HHMMSS_uuuuuu.txt
    strftime(filename, sizeof(filename), "acquisitions/%Y%m%d_%H%M%S", tm_info);
    snprintf(filename + strlen(filename), sizeof(filename) - strlen(filename),
             "_%06ld.txt", tv.tv_usec);

    FILE *fp = fopen(filename, "wb");
    if (fp == NULL) {
        perror("fopen");
        return;
    }

    fwrite(data, 1, len, fp);
    fclose(fp);
    printf("New packet saved in this file: %s\n", filename);
}

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    char buffer[BUFFER_SIZE];
    socklen_t addr_len = sizeof(client_addr);

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(EXIT_FAILURE);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    printf("Listen on UDP port: %d...\n", PORT);

    while (1) {
        ssize_t len = recvfrom(sockfd, buffer, BUFFER_SIZE, 0,
                               (struct sockaddr *)&client_addr, &addr_len);
        if (len < 0) {
            perror("recvfrom");
            continue;
        }

        write_to_file(buffer, len);
    }

    close(sockfd);
    return 0;
}
```

Integrating an MDS into an EPICS network using the provided Python softIOC

On the USB flash drive delivered with your MDS-ACCT, you will find a Python softIOC, as well as a Control System Studio (CSS) Phoebus GUI to integrate your MDS-ACCT into an EPICS network.



CSS Phoebus GUI made for the MDS-ACCT Python softIOC

The Python SoftIOC named "MDS-ACCT softIOC.py" runs a PVAccess (PVA) server, as well as an algorithm that links the MDS-ACCT's UDP/IP protocol to the associated process variables of this PVA server. The Numpy and PvaPy libraries are required. This SoftIOC must be run on a computer whose IP address matches the MDS-ACCT's target/client IP.

The related Process Variables are the following:

- MDS-ACCT:idn (String)
- MDS-ACCT:packet (Structure)
- MDS-ACCT:requested_range (Unsigned Byte)
- MDS-ACCT:requested_trigger_delay (Unsigned Long)

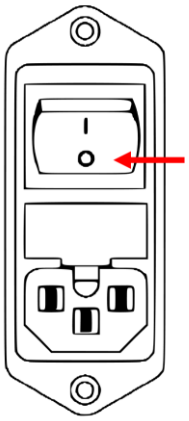
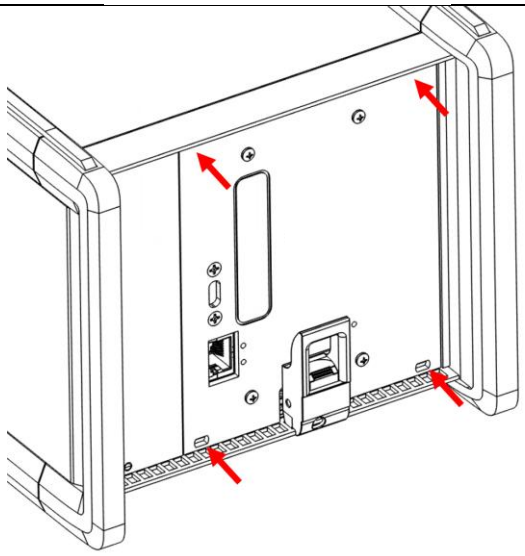
The "MDS-ACCT_display.bob" display can be opened using CSS Phoebus. It allows you to view the latest acquisition data as well as all parameters transmitted by the MDS-ACCT. This GUI also allows you to change the range of the ACCT-E-RM-3R and adjust the trigger delay.

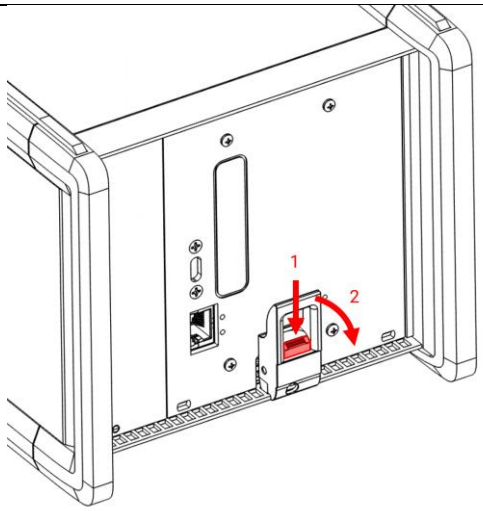
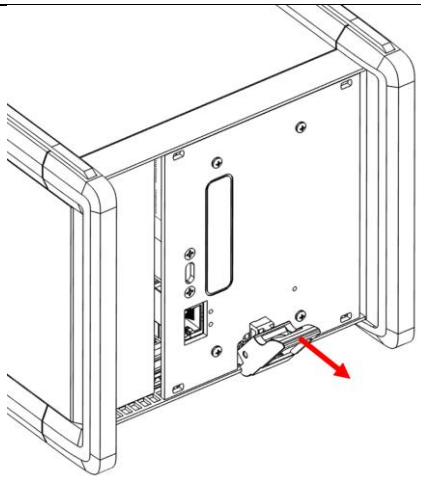
This Python SoftIOC and this CSS Phoebus display are examples that can serve as a starting point for integrating an MDS-ACCT into your control system. Feel free to inspect and modify the Python code to suit your needs.

MICROSD CARD CONFIGURATION

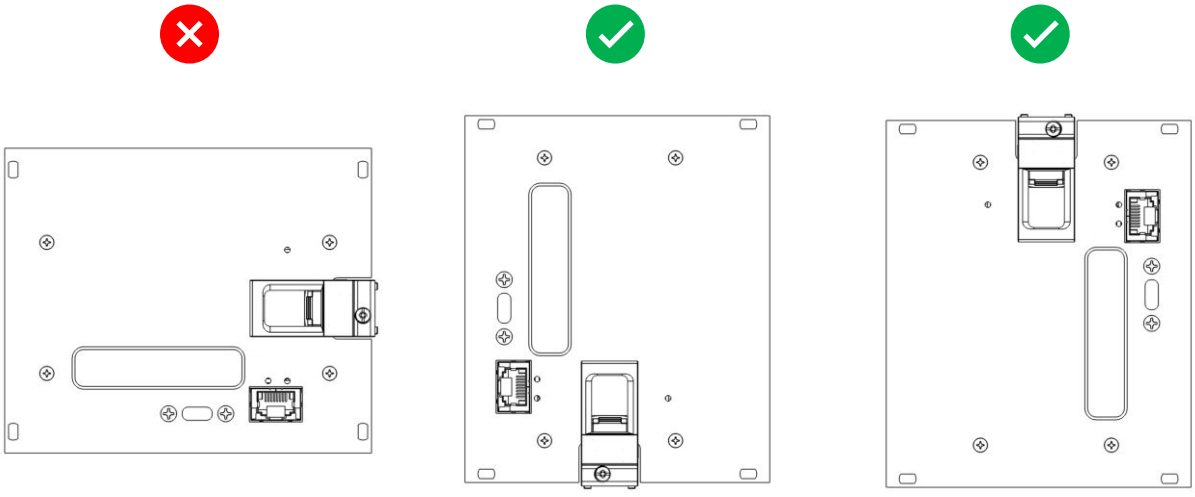
The MDS contains a microSD card that stores numerous configurable settings, calibration parameters, and the MDS's compiled firmware.

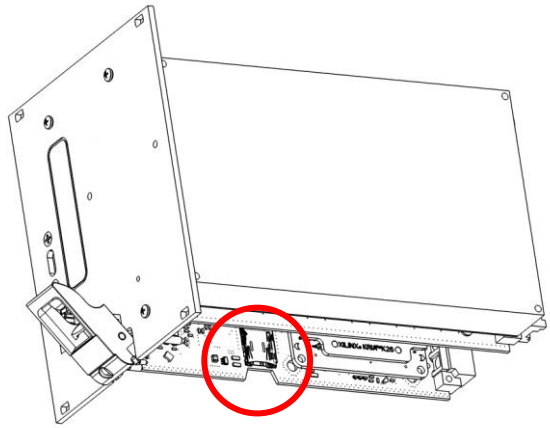
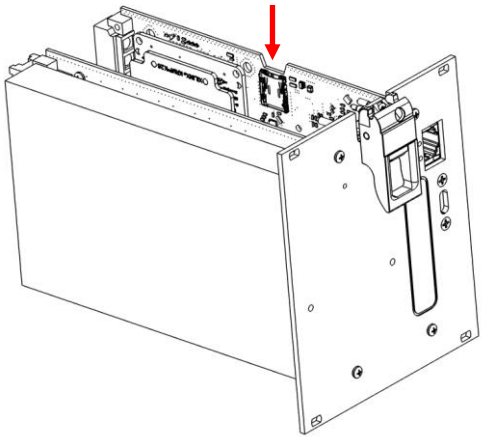
Here are the steps to access the microSD card:

	Turn off the chassis.
	Unscrew the 4 screws in the corners of the MDS module using a PZ1 screwdriver.

	Press down on the gray part of the extraction handle, then lower the handle.
	Extract the MDS module by pulling on the handle.

Warning! Never place or hold the MDS module on its side!



	The microSD card is located on the main board, on the bottom inner side.
	You can place the MDS module upside down on a table, then press the microSD card to eject it.

Files and folders in the root directory of the microSD card

- cal Folder containing calibration settings (DO NOT TOUCH)
- packet Transmitted waveforms configuration files
- settings Acquisition configuration files
- BOOT.bin MDS compiled firmware (DO NOT TOUCH)
- save.zip Backup of factory settings

"settings" directory

File name	Description
cIp.txt	IP Client – The IP (v4) address to which the data is sent.
cPort.txt	Client port – The UDP port to which the data is sent.
ethDHCP.txt	Use DHCP to obtain IP address? Must be 0 (=no) or 1 (=yes).
ethGW.txt	Default ethernet gateway (if DHCP is not used or does not work).
ethIP.txt	Default ethernet IP (if DHCP is not used or does not work).
ethMask.txt	Default ethernet mask (if DHCP is not used or does not work).
idn.txt	Unique identification name.
sPoolLen.txt	Number of 10 MS/s samples to be averaged to make one slow buffer sample. The sampling rate of the slow buffer will be equal to 10 MS/s divided by this value.
trigDly.txt	Trigger delay in 6.25 ns steps. If set to 0, the acquisition circular buffer stops at the time of the trigger. Example: If set to 800, the circular buffers will also contain 800 * 6.25 ns = 5 µs of signal after the trigger.
trigLvl.txt	Trigger level in ADC counts, when MDS is configured to trigger on a rising edge of IN1.
trigSrc.txt	Trigger source: 0 = TRIGGER SMA on rear panel. 1 = Trigger on the rising edge of IN1.
wfLength.txt	Waveform length for every buffer. Maximum value: 1000 samples.

"packet" directory

The files in this folder allow you to choose which waveforms should be transmitted. They contain only the values 0 (do not transmit) or 1 (transmit).

File name IN1	File name IN2	Description
1na160.txt	2na160.txt	Current values, in nA, acquired at 160 MS/s.
1na10.txt	2na10.txt	Current values, in nA, acquired at 10 MS/s.
1naS.txt	2naS.txt	Current values, in nA, acquired at low sampling rate.
1uv160.txt	2uv160.txt	Voltage values, in µV, acquired at 160 MS/s.
1uv10.txt	2uv10.txt	Voltage values, in µV, acquired at 10 MS/s.
1uvS.txt	2uvS.txt	Voltage values, in µV, acquired at low sampling rate.
1raw160.txt	2raw160.txt	Raw ADC values, acquired at 160 MS/s.
1raw10.txt	2raw10.txt	Raw ADC values, acquired at 10 MS/s.
1rawSacc.txt	2rawSacc.txt	Accumulation pooling of raw ADC values, acquired at low sampling rate.
1rawSmin.txt	2rawSmin.txt	Min pooling of raw ADC values, acquired at low sampling rate.
1rawSmax.txt	2rawSmax.txt	Max pooling of raw ADC values, acquired at low sampling rate.

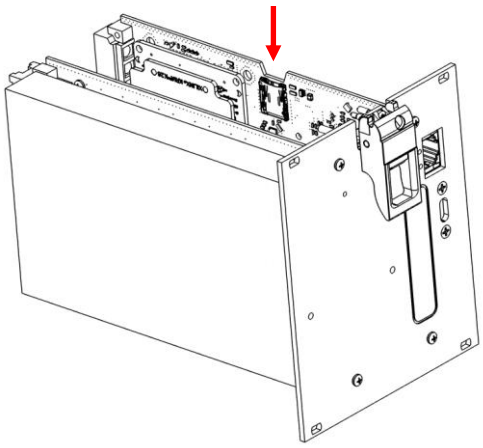
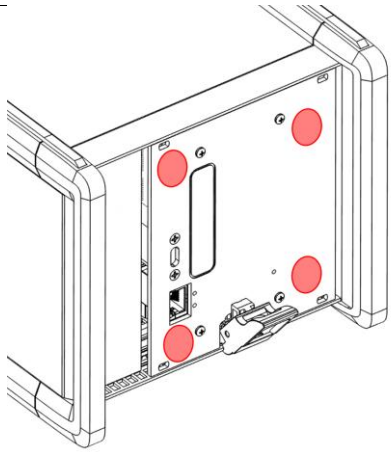
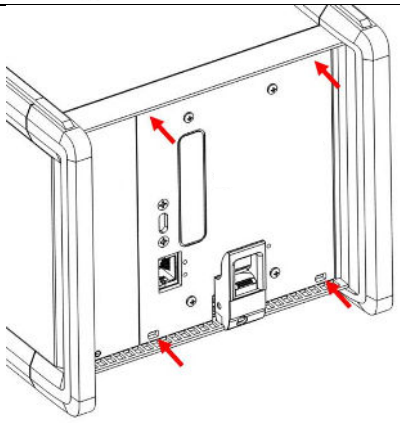
Note that the trigger frequency supported by the MDS depends on the amount of data to be transmitted. In addition, each UDP/IP packet can contain only a limited number of characters (set to 64k by the MDS).

Here are some reference values for estimating the number of characters per packet in the worst-case scenario:

Header:	~300 chars
One current waveform:	12 chars, multiplied by waveform length
One voltage waveform:	10 chars, multiplied by waveform length
One raw waveform*:	6 chars, multiplied by waveform length

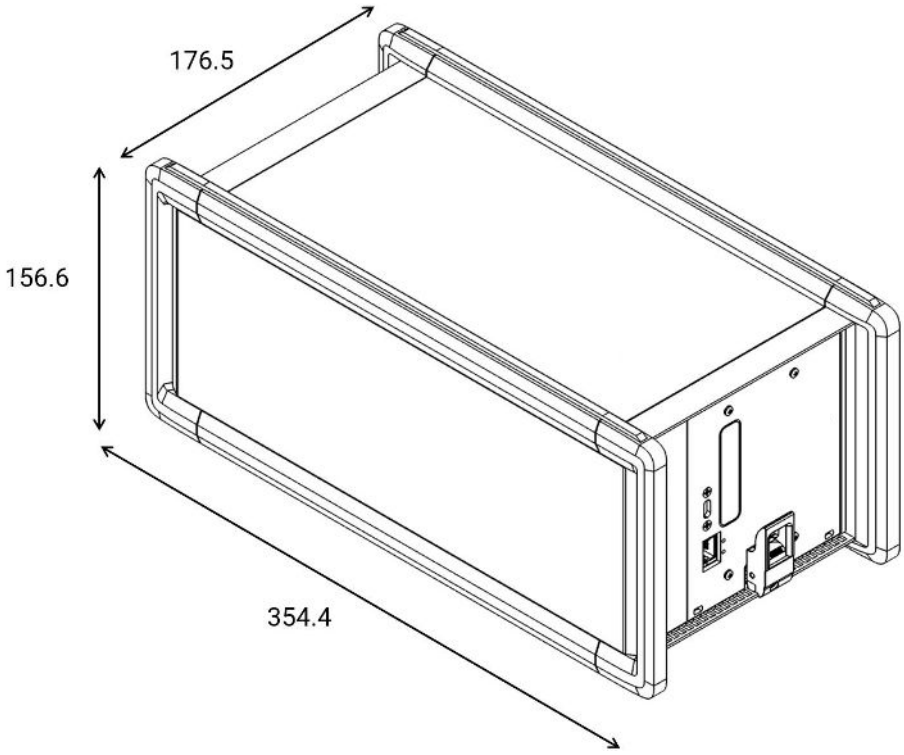
* Except the two raw slow accumulation pooling waveforms, which is 6 + log10(sPoolLen) chars (round up to the next whole number), multiplied by waveform length

To reassemble the MDS module into its chassis:

	Don't forget to reinsert the microSD card.
	Insert the MDS module with the extraction handle facing down and press on the red areas shown in the illustration until the handle is fully raised. Be careful: the screws in the corners may prevent insertion if they are not aligned properly.
	Tighten the 4 screws.

MECHANICAL DIMENSIONS AND DRAWINGS

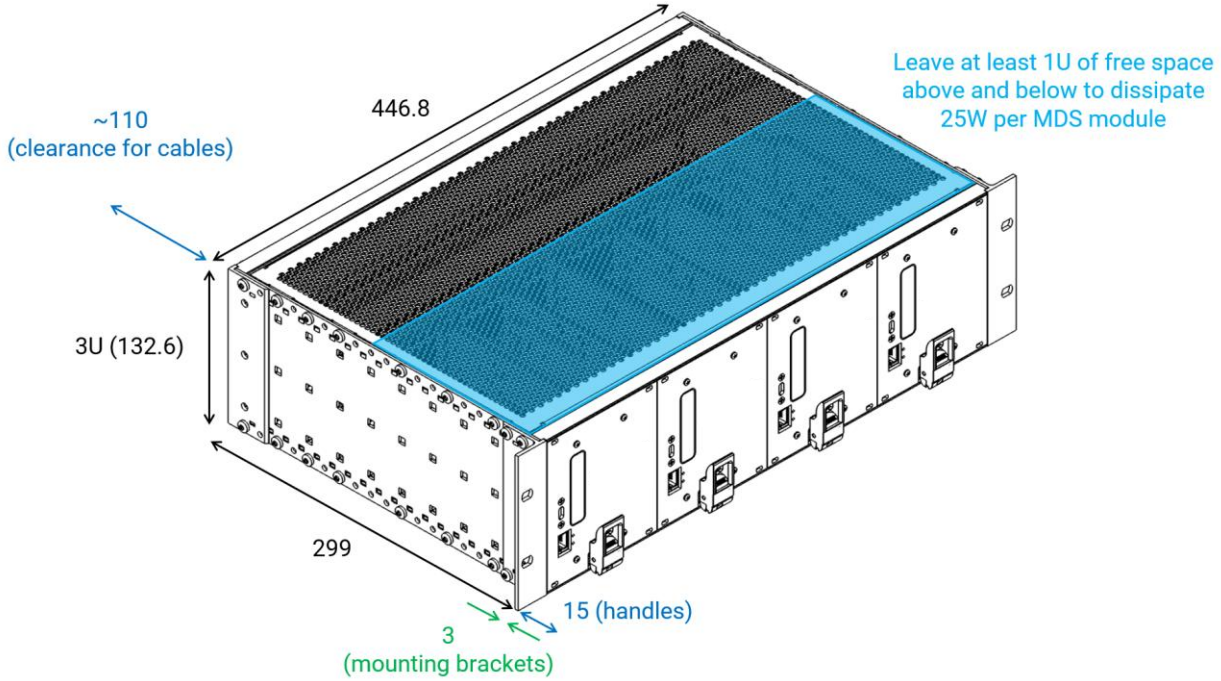
MDS-TTC



MDS-TTC outer dimensions in mm

Based on NVent Schroff reference #24582311

MDS-RFC/x



MDS-RFC/x outer dimensions in mm

MDS-RFC/x is a 3U 19" RF-shielded chassis based on NVent Schroff parts.
"x" represents the number of wired MDS stations, from 1 to 4.
The chassis is similar to the reference #24563143.

MISCELLANEOUS

Note on the -HZ option

By default, the MDS-ACCT is configured with 50 Ω, ±1 V inputs. It can optionally be assembled with high-impedance, ±10 V inputs by selecting the free of charges -HZ option.

The selection of the -HZ option depends on your ACCT-E-RM model:

- For a single range ACCT-E-RM-[...], the -HZ option for the MDS-ACCT must be selected.
- For a single range ACCT-E-RM-[...]**-50R**, the -HZ option for the MDS-ACCT **must not** be selected.
- For a 3 ranges ACCT-E-RM-3R-[...], the -HZ option is optional, as the electronics provide both a 50 Ω output and a high-impedance output.

It is recommended to operate in 50 Ω mode when working with long cables, in order to avoid signal deformations due to reflections. However, cable attenuation must be taken into account. If attenuation-independent measurements are required, high-impedance mode can be used. However, signal edges may be deformed.
As a general guideline, 50 Ω mode is recommended when the cable length between the ACCT-E-RM and the MDS-ACCT exceeds 2 meters.

Note on the automated charge measurement

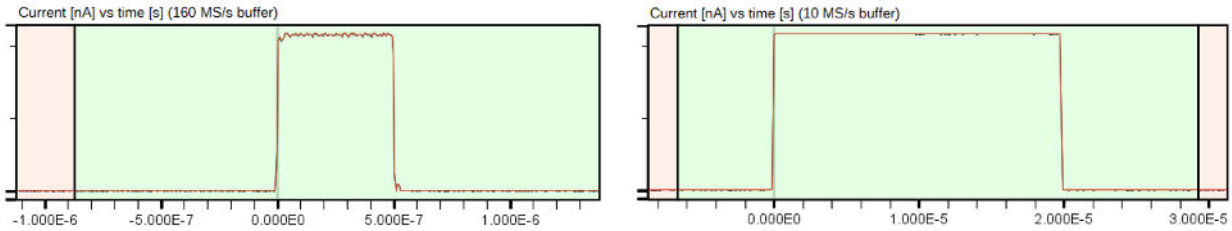
The MDS-ACCT automatically computes and transmits the pulse charge from both the 160 MS/s and 10 MS/s acquisition buffers. To ensure accurate charge estimation, a correction algorithm is applied to significantly reduce the impact of low-frequency noise that may be picked up along the acquisition chain.

This correction algorithm uses a portion of the samples to estimate baseline variations caused by low-frequency noise. Therefore, it is essential to adjust the MDS-ACCT trigger delay so that the signal of interest lies entirely outside the correction regions:

- 160 MS/s buffer:
  - First 10% of samples: offset estimation
  - Remaining 90%: signal of interest, used for charge computation

- 10 MS/s buffer:
  - First 5% and last 5%: estimation of baseline offset and slope
  - Middle 90%: signal of interest, used for charge computation

The following diagrams illustrates how to properly set the trigger delay to ensure accurate charge computation. The pulse of interest must be located within the green region, while the orange regions must remain free of any signal.



Regarding high-frequency noise, the principle of charge computation through integration inherently makes its contribution negligible. Since this noise has zero average and varies rapidly relative to the acquisition window, it largely cancels out during the integration process.

More information and latest manuals revisions can be found on our website

www.bergoz.com

If you have any questions, feel free to contact us by e-mail

info@bergoz.com

Bergoz Instrumentation – 156, rue du Mont Rond – Espace Allondon Ouest – 01630 SAINT GENIS POUILLY – France
Tél. : +33 (0) 459 42 66 42 Fax : +33 (0) 450 42 66 43 – bergoz@bergoz.com

www.bergoz.com
S.A.S. capital 152K€ - APE 2651B – TVA-VAT-IVA-USt. N° FR88414997130 – Siren 414 997 130 R.C.S. Bourg-en-Bresse