



Bridgeport Instruments, LLC, 11740 Jollyville Rd., Ste 500, Austin, TX 78759,  
[www.BridgeportInstruments.com](http://www.BridgeportInstruments.com)

By **Michael Momayezi**  
Phone 512-553-9933  
Fax 512-553-9934  
email **momayezi**  
**@bridgeportinstruments.com**

# MCA-1000 Users Manual

V. 2.2, Nov 2021

## Summary

The MCA-1000 is a family of low-cost multi-channel analyzer (MCA) for measuring gamma-ray radioactivity with a scintillator.

The PMT-1000 includes an operating voltage supply and plugs directly onto a photomultiplier.

The SiPM-1000 includes the SiPM operating voltage supply and plugs directly onto an SiPM-array.

Using its embedded 32-bit ARM processor, it provides accurate gamma-ray spectroscopy and gain stabilization together with many advanced features such as automatic background subtraction and alarm computations.

This document describes how to use the MCA-1000 from the graphical user interface and from Python scripts. Since all commands are formatted as human readable JSON strings, developers will find they can program the MCA-1000 in any language they want.

## Feature summary

- Histograms: Either two 1K×32 or one 2K×32
- Up to 150kcps histogramming rate
- PMT-1000: Support for positive and negative operating voltage and different PMT pinouts.
- Automatic sample vs background histogramming with alarm function and programmable region of interest.
- Radiation Portal mode with automatic background tracking and programmable alarm
- Only 75mW (5V@15mA) power consumption, USB and serial interface.

## Table of Contents

1. Supporting Documentation
3. Energy Spectrum
4. Sample vs Background
5. Mathematics of Errors and Alarms
6. Radiation Portal Monitor
7. Two-Channel Logger
8. Arrival Time Spectrum
9. Analog
10. Gain Stabilization
11. Software Upgrades
12. Mechanical and Pinouts
13. Ordering



Fig. 1: Two of the many MCA-1000 variants, one for PMT and one for SiPM. PMT-1000 MCA with high voltage unit on the left. On the right: SiPM-1000 MCA+detector assembly with SiPM array and 38mm NaI(Tl) crystal in detachable housing.

**Compact SiPM-detector:** Note that the SiPM-1000 electronics, including the SiPM-array all fit into the top part with the silver rim. The bottom grey aluminum housing is detachable and contains the 38mm NaI(Tl) scintillator.

## 1. Supporting documentation

**Open-source software:** The software is open source and mostly written in Python. For Windows, Bridgeport Instruments provides a software installer, which by default will create a C:\BPISoftV3 directory. That folder includes Python 3.7 with three added packages: ZMQ ([www.zeromq.net](http://www.zeromq.net)), wxPython and Matplotlib v 3.2. ZMQ is used to implement the client-server behavior and it is accessible from more than 40 programming languages. Matplotlib is used for the graphics interface, and wxPython is used to create a traditional user interface with pull down menus.

The wxMCA folder that contains the MCA software can be placed anywhere on the hard disk. Inside is a folder called documentation and the best starting page is [wxMCA/documentation/english/introduction/introduction.html](http://wxMCA/documentation/english/introduction/introduction.html) which you can open in any web-browser. This set of linked documents contains a description of every control variable and every data field used by the MCA-1000.

## 2. Getting started

### 2.1 Using the USB interface

For USB communication the MCA-1000 uses

libusb0.1 (Linux) and libusb\_win32 on Windows 10. The Windows 10 libusb dll is digitally signed. When the MCA hardware is present during the installation, the installer will automatically link the MCA-1000 to libusb\_win32.

However, it is also possible to simply copy the BPISoftV3 folder onto the C: drive and install the Windows USB-driver manually. This needs to be done only once, and Windows will later recognize any other MCA-1000, by its USB vendor ID of 0x1FA4 and the USB product ID of 0x0101 (PMT-1000) or 0x201 (SiPM-1000).

Open the Windows Device Manager and connect the MCA-1000 to a USB port. Wait until it appears in the Device Manager as an unknown device of type pmtMCA or sipmMCA.

Open the C:\BPISoftV3 folder and launch zadig-2.4.exe. Use Options/List All Devices to refresh the device list and select the device with the BPI vendor ID of 0x1FA4. To the right of the green arrow, select libusb-win32 (v 1.2.6.0) and click the big 'Install Driver' button below. Note that it may take up to 20 seconds for the screen to update - be patient. Once the 'Driver Installation Successful' message appears you can close the window. In the Device Manager you will now see the MCA-1000 listed under 'libusb-win32 devices'.

### 2.2 Launching the software

**First launch the MCA Data Server.** Look inside the wxMCA folder. Under Windows, double-click on run\_mds.cmd. Under Linux, launch wxMCA/mds/mds\_server.py making sure you use a python 3.6 or higher installation that also includes ZMQ, wxPython and Matplotlib.

If the MDS reports no MCA found, the OS may have been slow in enumerating the MCA on the USB bus. Kill the MDS and launch it again.

**Then launch the User Interface.** Under Windows, double-click on run\_wxMCA.cmd. Under Linux, launch wxMCA/wxGUI/MCA\_Main.py

**The items in the menu bar are mostly self-explanatory.** You can view results and edit

instrument settings in a spread-sheet environment. To send the changed values to the instrument, you need to click on File → to MCA.

### 3. Energy Spectrum

**The MCA provides fast and accurate measurements of energy spectrum and count rates.** To acquire an energy histogram, select Display → Histogram from the menu bar. In response, a histogram panel will open. On that panel use ( **New** ) to erase the old histogram and count rate data and begin a new acquisition. The panel does not automatically refresh, so click the "Refresh" button to get an updated energy spectrum.

Use "Save" to append the energy histogram and count rate data to a default data data file or a file of your choice. The default file name is of the form xyz\_histo\_status.json where xyz is the detector serial number. By default that file is located in wxMCA/user/mca1k/data.

#### 3.1 Adjusting operating voltage

**You need to adjust the operating voltage to set the maximum measurable energy.** In its default setting the MCA has a 1024 bins of which about 900 are being used. Users need to adjust the operating voltage to change the gain of the PMT. In the table below we recommend calibrations for different applications.

<b>662keV peak pos.</b>	<b>Comment</b>
662	1.0keV/bin => High gain for measuring low energies down to 20keV
331	2.0keV/bin => Measure NORM up to K-40 at 1460keV
220	3.0keV/bin => Wide range up to Tl-208 at 2615keV

Table 1: Cs-137 calibration for different tasks; NORM: Naturally occurring radioactive materials

Access the operating voltage control by editing ( **cal\_ov** ) in the File → arm\_ctrl spreadsheet. Enter a new operating voltage in the voltage field and save via File → to MCA.

Go back to the histogram panel and click "New" to start a new acquisition. If you change the operating voltage by 1%, the PMT gain will

change by about 6%.

#### The GUI software can assist with the calibration.

To do that it relies on two variables: The variable ( **cal\_energy** ) is the gamma-ray energy you want to use for the calibration. For example, when using a Cs-137 source,  $cal\_energy = 661.62keV$ . Enter just the energy in keV, without the "keV". Secondly, set the keV/bin variable ( **keV\_bin** ) as desired, cf Table1

There is a third variable, related to the type of PMT, but its exact value is not important. By default,  $pmt\_exp = 5.8$  which is appropriate for 8-dynode R6231/3 PMT and the 10-dynode CR105. For other 10-dynode PMT's  $pmt\_exp = 7.5$  is appropriate. It should be noted that the calibration process will always converge, even if the  $pmt\_exp$  is not a perfect match for the PMT in use. When in doubt, use the smaller value.

To perform the calibration, acquire a spectrum, then adjust  $fit\_min$  and  $fit\_max$  such that that they enclose the calibration peak. The algorithm assumes that the calibration peak is the tallest peak in the fit range. The algorithm also selects an appropriately narrow fit range around the peak. It only uses  $fit\_min$  and  $fit\_max$  as an indicator of the region where to find the calibration peak.

Acquire a spectrum with sufficient number of counts in the peak, eg 1000 per bin at the peak position. Then click on the ( **Calib** ) button. The algorithm will make a fit to the peak, and compare the peak position with where it should be, given the  $keV\_bin$  and  $cal\_energy$  variables. From that the algorithm computes a new operating voltage and applies it to the detector. Acquire a new spectrum and observe that now the peak has moved close to its desired position. Repeat these steps until the peak is in the right place.

#### 3.2 Count rate measurement

**Observe how the count rate accuracy improves with measurement time.** The MCA reports count rates together with their statistical errors. This gives users a useful tool. Manually, or programmatically, they can end a measurement precisely when the desired accuracy has been reached, which saves time and money.

The error (in %) is computed from the number of events as  $100 \cdot 2 / \sqrt{N}$ , where N is the number of events. This is called the statistical 2- $\sigma$  error. The true count rate lies within this error range with a 95% probability.

Note that the MCA corrects the recognized count rate using the known dead time per event. Hence the reported count rate is greater than the recognized count rate. The reported count rate is also an accurate estimate of the true number of counts per second. In fact the reported count rate should match the true input count rate with about 1% accuracy (systematic error) for input count rates up to 100kcps. The statistical count rate error is computed correctly using the number of recognized events.

### 3.3 Dead time

**The MCA has a non-extendable dead time of 6.50 $\mu$ s per recognized event.**

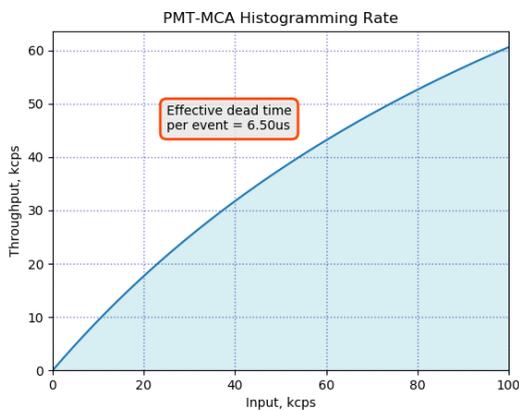


Fig. 2: Expected histogramming rate vs input count rate.

### 3.4 High count rates

**There may be a small gain shift at high count rates.** Users should be aware that at input count rates above a few 10kcps there may occur small gain shifts in the energy spectrum. Typically, at 50kcps the gain may fall by about 1%, but the exact gain shift will depend on the PMT that is used on the detector.

## 4. Sample vs Background

**The instrument provides automatic background subtraction.** To accurately measure the radioactivity of a weak source, or a weakly contaminated sample can be a complicated process. However, in all cases the process requires a precise measurement of the

background and the ability to correctly subtract that background spectrum from the sample spectrum. And the MCA-1000 greatly simplifies this task.

**The MCA-1000 stores sample and background data on the instrument and reports the difference.** From the dashboard, open the "Sample - Bck" panel. From top to bottom you will see three spectrum displays. The one at the top is the background spectrum. Initially this will be empty. The middle panel has the sample spectrum, and initially it will show the previously acquired spectrum. Finally, the bottom panel has the difference spectrum, which initially equals the middle spectrum.

To acquire a background spectrum, remove all unwanted radioactive materials from near the detector and click on "New" next to the top spectrum panel. You can increase the accuracy of the results by measuring the background spectrum 4 times longer than a typical sample spectrum. Click "Ref" next to the top spectrum to update the display.

Once you have counted the background long enough, click on "New" next to the middle spectrum panel. This starts a new sample spectrum acquisition and stops the background acquisition. The background spectrum will now have stopped and remain unchanged.

If you simply acquire a sample spectrum under the same conditions that you acquired the background spectrum, you will see a similar spectrum grow; click "Ref" next to the middle spectrum to update the display.

Every time the display is updated, the software reads three spectra from the MCA: the background, the sample and the difference spectrum. You will notice, that even in the absence of any new radioactivity the difference spectrum is not exactly a flat, empty spectrum. The reason for that is that the gamma-rays arrive randomly, and there is therefore a natural statistical variation between acquired spectra.

**The instrument computes probabilities and alarms.** The MCA computes the probability that the count rate in the difference spectrum was caused by nothing but background. The message box next to the difference spectrum shows the result of that computation. You can

confine this computation to a region of interest (ROI) by adjusting `roi_low` and `roi_high` in the alarm panel. For instance you can increase the sensitivity to Cs-137 by setting `[roi_low, roi_high]` to `[580, 780]`. The ROI is entered in units of raw histogram bins. For instance at 2keV/bin, the previous `[roi_low, roi_high]` would be `[290, 390]`.

Finally, you can set an alarm threshold. When the probability that the measured sample radioactivity is caused by nothing but background is too low, the MCA can raise an alarm. The alarm threshold can be set in `alarm_ctrl` via the `alarm_thr` variable. For example, a value of `1.0e-3` is interpreted as a probability of 1 in 1000.

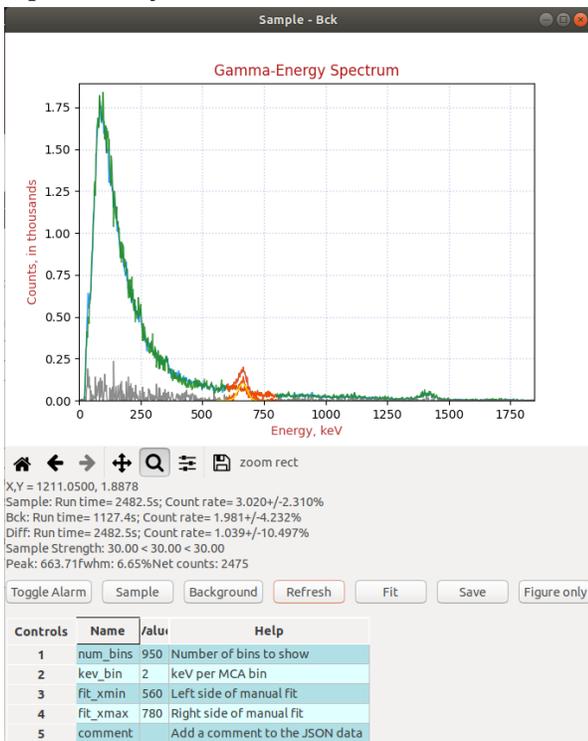


Fig. 3: Example of a background histogram and a nearly identical sample histogram (blue and green).

In figure 3 we show an example of a natural background histogram (green) and a sample histogram (blue), with a weak Cs-137 source at 2.5m distance.

The run-time corrected difference spectrum is shown in grey. The region of interest used for the alarm computation is indicated in orange.

**The instrument reports the signal strength and confidence interval.** The excess of count rate over background caused by a small amount of radioactivity is called the signal. Here the signal strength is reported as  $-\log_{10}$  of the

above-mentioned probability. The maximum signal strength that will be reported is 30.

The instrument also reports the 95% confidence interval for the measured source strength. Users can thus make an informed choice on how to set alarm levels and thresholds for taking a decision. The mathematics is described in the math section below.

## 5. Mathematics of Errors and Alarms

This section is for the reader with an interest in the mathematics that is used by the instrument.

### 5.1 Count rates and their errors

When measuring count rates, the instruments counts events and the elapsed time. Systematic errors for measuring the run time are very small and are ignored here. The dominant variation come from the fact that the number of events during a given time interval is Poisson-distributed. When the total number of events counted is greater than 100, we can approximate the resulting Poisson distribution with Gaussian normal distribution with an average of ( $\mu$  = number of events) equal to the number of events and a standard deviation of  $\sigma = \sqrt{\mu}$ . For a given measurement the instrument reports the 2- $\sigma$  relative count rate error

$$\epsilon = 2/\sqrt{\mu}$$

**Dead time does not contribute to statistical error.** The instrument knows the incurred dead time per recognized event and computes a live time as run time minus dead time. The reported count rate is `events/live_time`.

### 5.2 Background subtraction

**Background time can be different from sample time.** The instrument can subtract a background spectrum from a sample spectrum even when the two have been acquired for different amounts of time. The calculation for the difference histogram bins is

$$\text{diff} = \text{sample} - \frac{\text{sample\_time}}{\text{back\_time}} \cdot \text{back}$$

Here, `diff`, `sample`, and `back` are the histogram bin contents of the three energy

spectra.

Count rates are measured by summing all events within the alarm region of interest, and dividing by the respective live times. Count rate errors are computed from the number of events in the region of interest.

The difference count rate is computed as a direct difference,  $R_D = R_S - R_B$ , from the dead-time corrected sample and background rates.

The count rate error for the difference spectrum has to be computed from the two uncorrelated errors of the sample and the background counting:

$$\epsilon_d = \frac{\sqrt{(R_S \cdot \epsilon_S)^2 + (R_B \cdot \epsilon_B)^2}}{R_S - R_B}$$

Notice that for very small difference count rates the resulting relative error can be quite large. Further, a difference rate with a large relative error, eg  $1.0\text{cps} \pm 200\%$ , simply means that the result is compatible with a difference of 0cps.

### 5.3 Computing probabilities

For a given sample measurement time,  $T_S$ , the MCA knows how many background events to expect on average:  $N_B = T_S \cdot R_B$ . It compares that number to the number of actually measured events,  $N_S$ . Using correct Poisson statistics, not a Gaussian approximation, it calculates the probability that  $N_S$  could have been caused by the known background:  $P = P(N \geq N_S | N_B)$ . If  $N_S \gg N_B$ , this probability will be very small. If it falls below the given alarm probability (**alarm\_thr** in the alarm panel), the MCA can raise an alarm. In other words, a stronger signal causes a lower probability.

**However, a user might prefer to use a measure of the signal strength that increases with the signal strength.** Hence we define a signal strength as  $A = \log_{10}(1/P)$

For the confidence interval of the signal strength, the instrument reports a narrower measure. It computes the two opposites we get by assuming  $1\text{-}\sigma$  errors in the background count and the sample count, but pointing in opposite directions. The resulting rectangular

two-dimensional confidence interval contains about 71% of all cases.

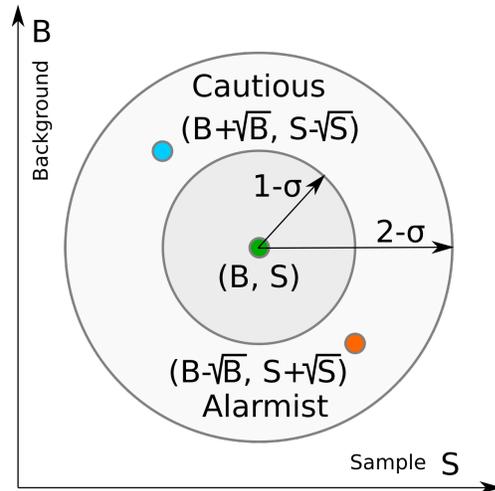


Fig. 4: Two dimensional confidence interval.

## 6. Radiation Portal Monitor

**The MCA can act as a radiation portal monitor with background tracking and a programmable alarm function.** In a Radiation Portal Monitor (RPM) the data acquisition unit must perform a number of tasks:

In a Radiation Portal Monitor (RPM) the data acquisition unit must perform a number of tasks:

- Continuous background measuring;
- Deliver an alarm within a few seconds after the time of closest approach by a source;
- Keep a few seconds of alarm history so that a polling host will not miss an alarm;
- Automatically reset the system on a continuous alarm to remain operational;
- Support a programmable alarm threshold based on a false-alarm rate;
- Support an adjustable background averaging time, eg 30s for a big gamma-ray detector vs 5 minutes for a neutron detector,
- Be able to recognize passing sources without the aid of an occupancy detector.

The MCA-1000 implements all that functionality within its embedded 32-bit ARM processor. An alarm can be sent as digital pulse

with programmable width. The output is fed by a line driver. The chip itself can for a short time source over 3A at 5V – enough to drive a light or a buzzer. Alarms can also be read via USB or the serial communications interface.

The DC output from the line driver must be limited to 30mA. Shorting the line driver output for more than 1ms may render the line driver inoperable. An internal 50Ω resistor provides a series termination to reduce reflections on a 50Ω coaxial cable. With the 50Ω resistor in place the output power is limited to 30mA@3.5V output voltage; ie to 100mW.

## 6.1 Theory of operation

In the discussion below, times are given in units of time slices. Typically a time slice is 100ms long and the processor perform one RPM computation step every time slice. In some software versions, time slices can be set in multiples of 50ms.

**At first the instrument measures the background.** On start, after power on or a reset, the ARM processor begins to measure backgrounds. Until backgrounds are know with sufficient precision, the ability to alarm is disabled. This wait time is user programmable, cf **wait** in the RPM panel. Typically the wait time is a fraction of the background averaging time, eg 20% to 50%. A shorter wait time yields an active RPM earlier at an elevated risk for a false alarm until a full background averaging time has passed.

As long as there is no alarm, the events counted during one time slice are considered background events. Using the region of interest (**roi\_low, roi\_high**) in the RPM panel allows the user to constrain the attention on a part of the energy spectrum.

The background counts per time slice are averaged using geometric averaging. Using the  $w = 1/Bck\_avg$  from the alarm panel the processor applies the formula:  $B_{n+1} = B_n + w \cdot (N - B_n)$ . The background averages are then stored in a 128-long FIFO. This way the instrument can look back 128 time slices to find an untainted background after an alarm has occurred.

We have  $0 \leq w \leq 1$  for the weight. If the

background at time  $t = 0$  changes from  $B_0$  to  $B_1$ , The background average responds to a step function with an exponential function of  $B(t) = (B_1 - B_0) \cdot (1 - \exp(-t/\tau))$  where  $\tau = 1/w$ . The standard deviation of the averaged background, ie its noisiness, improves as if  $1/w = Bck\_avg$  samples had been averaged.

### **What happens when a passing source causes an alarm?**

The instrument continuously computes a moving window sum of the last  $L$  time slices and compares that sum to the number of expected background counts during  $L$  time slices. For every time slice the instrument computes the probability that the observed counts could have been caused by the known background, cf the mathematics section above. It computes  $P = P(N \geq N_L | N_B)$ , with  $N_B = L \cdot B$ . If that probability is less than **epsilon** on the alarm panel, it will trigger an alarm.

### **An alarm is typically raised no later than $L/2$ time slices after the closest encounter.**

For a very strong signal, that alarm may be raised right at the leading edge of the  $L$  period; for a weak signal the alarm may be raised about  $L/2$  time slices after the source has passed the point of closest approach. In many applications the allowed latency  $L/2$  is about 2 seconds, which allows for a 4s summation time. At a time slice length of 100ms this means  $L = 40$ . In those applications, the time during which the radiation signal is detectable is 4s to 8s and and simulations show that 4s to 6s summation times produce the highest sensitivity; ie lowest minimum detectable activity.

**No missed alarms.** The instrument keeps a history of the alarm status with a maximum length of 128 time slices. This is controlled by the **history** parameter in the alarm panel. A polling device, via serial interface or USB, will be informed if there was an alarm present in the last **history** time slices. This way a polling host may have a a latency of 128 time slices (12.8s) and still will not miss an alarm.

As long as there is an alarm present in the alarm history FIFO, the instrument will suspend background updates and use the oldest background average in its memory as the best

estimator of the background.

**Automatic device reset ensures the instrument remains functional when the background suddenly increases.** Consider the case of a radiation detection backpack. The wearer walks into a room where radioactive material is present, and this causes an alarm. After *history* time slices, the instrument automatically resets and starts to accept the elevated radiation level as the new background. After a *wait* period (30s typically) the instrument will again be ready to alarm if the operator suddenly encounters an even higher radiation level.

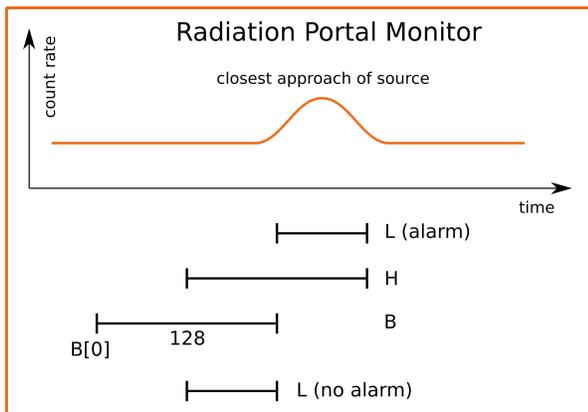


Fig. 5: Programmer's model of the Radiation Portal Monitor software.

## 7. Two-channel Logger

**A chart recorder for rapidly changing situations:** The MCA-1000 can log two parameters from the `arm_status` array in time steps as small as 50ms. Those data include status data (such as temperature or operating voltage) or radiation data such as counts per time step and alarm probabilities.

The `arm_logger` is implemented in an 8kB general purpose region of memory. For example users can log the change in operating voltage in response to a rapid change of count rate – to verify that the gain remains stable in such circumstances. Or the user may log net counts over background and the resulting alarm probability – to study the performance of the portal monitor alarm function. In case custom software is implemented on device the logger size may be reduced, or the logger may be omitted altogether.

The logger can be used to monitor rapidly

changing parameters in time intervals ranging from 50ms to 12.75s. The logger creates running logs for two parameters. The parameters are chosen by their index in the `arm_status` register. The parameter index is also indicated in the `wxMCA arm_status` table.

See the software documentation for details.

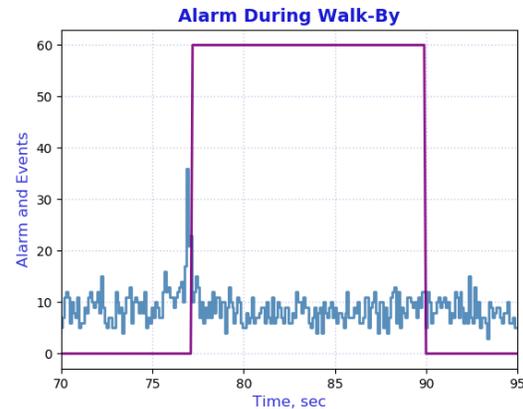


Fig. 6: Logger example: Events per 100ms time slice and result of the built-in alarm computation while a check source is moving past the detector.

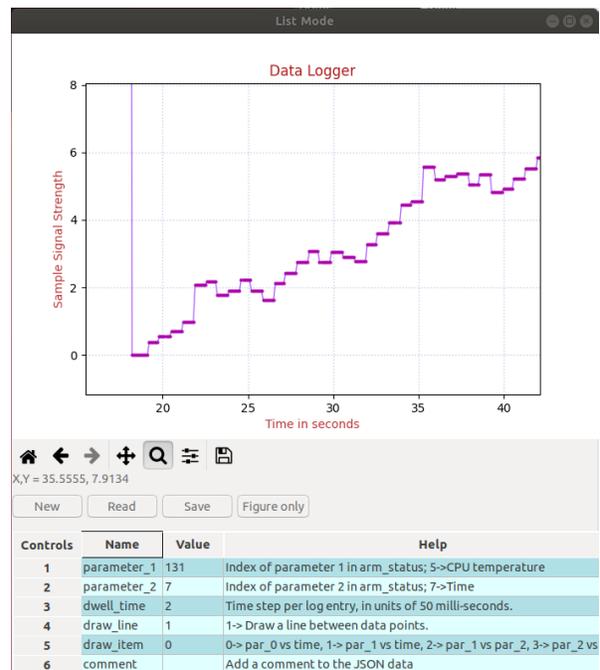


Fig. 7: Logger example: After having measured the background, a new sample run is started at  $T=13s$ . The sample is slightly more radioactive than the background and the certainty of that rises with time, cf "Sample vs Background" section.

## 8. Arrival Time Spectrum

**A scientific method to measure count rates.** The MCA-1000 provides fast and accurate measurements of count rates. A very accurate,

if advanced, method, is to collect a histogram of times since the last event. To acquire a time histogram, first select acquisition mode 1 in arm\_ctrl. Then select Display → Arrival Times from the menu bar. In response, a histogram panel will open. On that panel use ( **New** ) to erase the old histogram and count rate data and begin a new acquisition. The panel does not automatically refresh, so click the ( **Refresh** ) button to get an updated energy spectrum.

Use ( **Save** ) to append the arrival time histogram and count rate data to a data file. The file name can be automatically generated or selected from the dialog that opens.

The time bins are  $4/3=1.333\mu\text{s}$  wide. For randomly occurring events the arrival time histogram will show a falling exponential with a characteristic time of  $\tau=1/\text{count\_rate}$ .

**A straight line in the GUI:** In the wxGUI, the logarithmic display of the time spectrum shows a falling straight line.

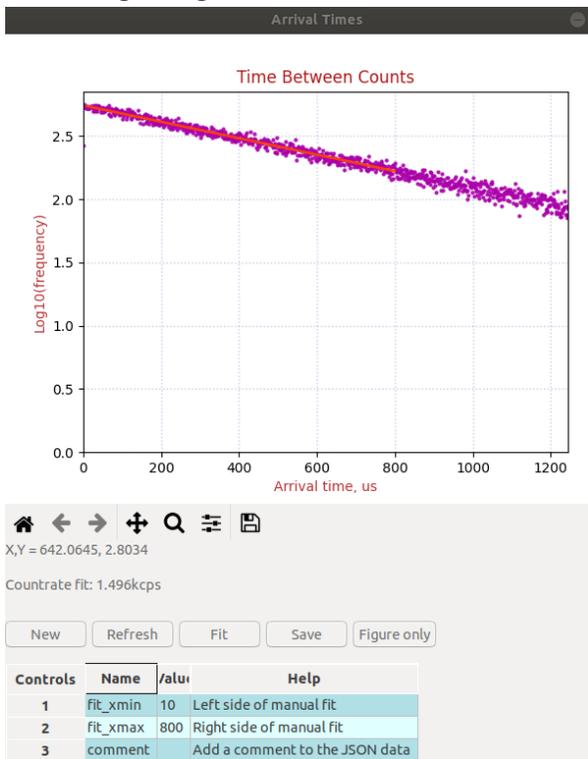


Fig. 8: Arrival time spectrum as shown in the GUI. A straight line fit reveals the true count rate.

## 9. Analog

### 9.1 Sample and Hold

The MCA-1000 uses a sample and hold amplifier to measure pulse heights. The input integrating preamplifier has a rise time equal to

the light emission time of the scintillator. It is an RC-decay preamplifier without active reset that returns to baseline within  $4\mu\text{s}$ , which sets the minimum theoretical possible dead time between pulses.

### 9.2 Dead time

The MCA-1000 can operate in 5 distinct modes, each with a different dead time. In all cases, the dead time ( $T$ ) is non-extendable and the maximum throughput is simply computed as  $1/T$ .

Mode	Dead time	Description
0	$6.52\mu\text{s}$	Standard histogram
1	$4.10\mu\text{s}$	Counting only
2	$13.65\mu\text{s}$	Histogram with EMI suppression
3	$4.66\mu\text{s}$	Arrival times
4	-	Standard histogram with programmable dead time; (reserved)

Table : Dead times in the different operating modes

## 10. Gain Stabilization

**The MCA-1000 provides built-in gain stabilization which can be enhanced with an LED.** In all cases gain stabilization relies on lookup tables to adjust one or more parameters as a function of the measured temperature of the photo-sensor.

The most common approach for both, the PMT-1000 and the SiPM-1000 is to use a single lookup table of the operating voltage vs temperature. That lookup table depends on the photo-sensor as well as the scintillator. It also provides a correct temperature compensation only if electronics, photo-sensor and scintillator are in a thermal steady state. If temperatures are changing too fast and the scintillator and photo-sensor temperature do not have predictable relationship, the gain stabilization cannot perform optimally. In practice it is necessary to thermally insulate scintillator and photo-sensor to make sure they respond slowly and in lockstep to a temperature excursion.

For the PMT-1000 it is also possible to use an LED to assist with the gain stabilization. This is useful if the user wants to compensate for

gain drifts of the PMT due to aging and wear-out rather than just temperature. In this case, there will be look up table to predict how the response to the LED light pulse should change over temperature to keep a given gamma-ray peak at the same point in the spectrum; eg a 662keV Cs-137 peak in bin MCA 331 (keV\_bin=2.0).

The MCA Data Server documentation shows how the lookup tables are constructed and where they are stored. The included examples show how the lookup tables can be uploaded to the non-volatile memory of the ARM processor.

The MCA-1000 devices ship with a default setting for the lookup tables. All tables are made for NaI(Tl). The voltage vs temperature table is used for gain stabilization mode  $gs=1$ ; When  $gs=2$ , the ARM processor adjusts the operating voltage such that the measured LED response matches the required response at the given temperature.

**A  $gs=1$  example:** Assume that the detector was calibrated at 25°C, and that now the temperature is 30°C. The voltage vs lookup table may indicate that compared to 25°C the operating voltage at 30°C should be 1% higher. The ARM processor will compute the new voltage target as  $cal\_ov*1.01$  and increase the operating voltage accordingly. This will be reflected in the `arm_status` data in the `voltage_target` field.

**A  $gs=2$  example:** A PMT-1000 can be equipped with an optional LED pulser that shines light into the back of the photo-multiplier (PMT). Staying with the above example of calibration at 25°C, the ARM processor looks at the recorded LED value at 25°C. If the temperature is now 30deg;C it will look up by how much the LED value should have changed to maintain the 662keV peak constant. The ARM processor then proceeds to adjust the operating voltage in small and measured steps to ensure that the measured LED response matches the required LED response at 30°C. Note that in this case, the voltage vs temperature lookup table is not used.

## 10.1 PMT-1000 Gain Accuracy

**Gain control with 15-bit DAC:** In the PMT-1000 we use 15-bit of a 16-bit DAC to

control the PMT voltage up to 1500V. Discretization errors in the gain due to the DAC differential non-linearity are limited to less than 0.1%.

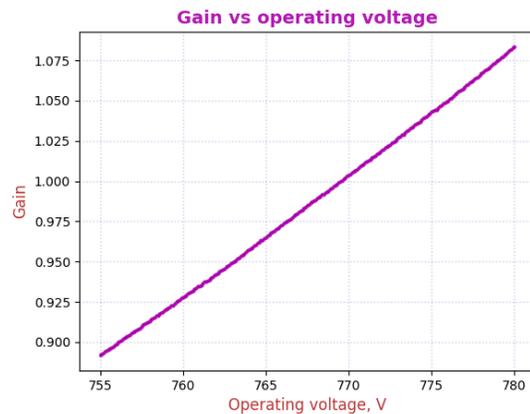


Fig. 9: Gain change of the PMT in response to a smooth change of the requested operating voltage. The voltage was increased in 50mV steps, and the peak position was measured at each step.

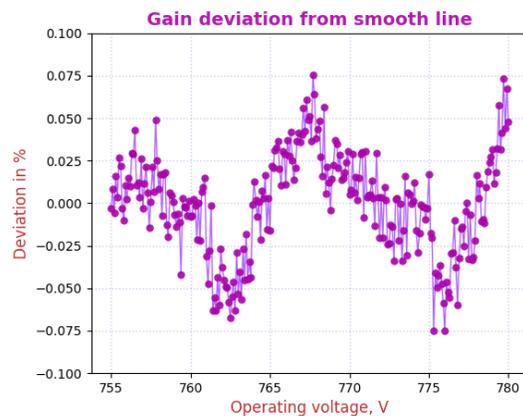


Fig. 10: Gain deviation from a smooth fit derived from the data in the previous figure. The gain error due to DAC nonlinearity is below  $\pm 0.1\%$ .

## 10.2 SiPM-1000/C Gain Accuracy

**The very small SiPM-1000/C uses a 12-bit DAC to set the operating voltage, which limits gain accuracy to typically 0.25%.** Due to the small size of the SiPM-1000/C printed circuit board, that MCA uses the 12-bit DAC that is built into the ARM processor to adjust the operating voltage.

The 12-bit voltage setting DAC of the SiPM-1000 has a differential non-linearity that affects how smoothly the user can vary the operating voltage and hence, the gain. The next graph shows the Cs-137 662keV peak position with a 2keV/bin MCA setting; ie the 662keV peak should be in bin 331. Note how the peak

position (ie the SiPM gain) varies with operating voltage in a staircase manner due to DAC differential non-linearity.

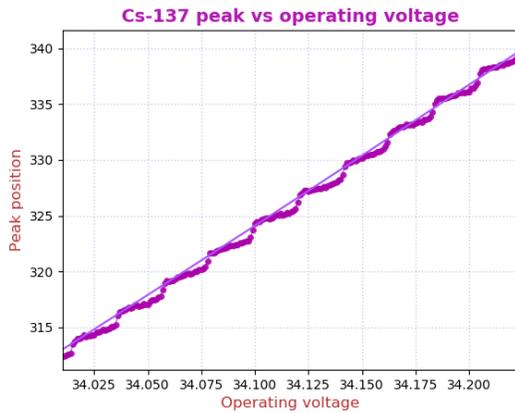


Fig. 11: Gain steps in response to a smooth change of the requested operating voltage. The voltage was increased in 1mV steps, and the peak position was measured at each step. The straight line in the graph is a fit through the entire 1V, 100 steps range.

The difference between the expected smooth gain vs voltage curve and the actual one, can be expressed as a gain error, in %, compared to the ideal case. This is shown in the figure below. Mostly, the gain error is  $\pm 0.25\%$  or less, but it could reach  $\pm 0.5\%$

Designers of analysis algorithms should be aware that any algorithm adjusting the gain, including the built-in gain stabilization, is limited by this phenomenon.

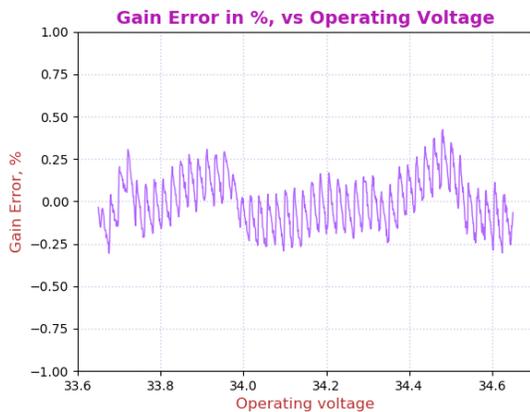


Fig. 12: Difference of observed gain compared to a straight-line fit expressed as a gain error in %. For a 662keV peak a 0.25% error amounts to 1.7keV and a 0.5% error amounts to 3.3keV.

### 10.3 SiPM-1000 Gain Stabilization

**SiPM-based system react faster to temperature changes and need less thermal insulation.** Unlike a vacuum photo-multiplier

tube (PMT) which has a thermal relaxation of 1 to 2 hours, SiPM photo-detectors react quickly. Hence it is the scintillator crystal and its packaging that set the overall thermal response time. When using a PMT, the scintillator and PMT will not be in thermal steady state until 2 to 4 hours after a temperature step. The only way to ensure consistent thermal behavior is to encapsulate the crystal and PMT in a thick insulation layer; up to 5cm of insulating foam. This perfectly OK for a stationary system but undesirable for portable or hand-held applications.

For a 38mm $\times$ 38mm NaI crystal the thermal relaxation time is about 5 minutes, meaning that it exhibits a 10% to 90% response to a thermal step within 11 minutes. In an uncooled system, the SiPM-array is tightly coupled to the NaI crystal and the two are always at the same temperature.

When mounted in the housing supplied by Bridgeport Instruments, the mechanical shock absorbers provide some extra insulation and the thermal 10% to 90% response slows down to about 30 minutes. As a result, the SiPM and the NaI crystal remain in a thermal steady state.

**Even without additional insulation, for a system with a 38mm NaI-crystal, the embedded gain stabilization is excellent.** In

the graph below we show the response of the 38mm size SiPM-1000 assembly with NaI crystal to a temperature drop of 5°C. The temperature chamber changed the temperature quickly (within 5 minutes) from 26°C down to 21°C.

The crystal plus SiPM-array combination reacts more slowly, and the embedded gain stabilization method has no problem keeping up.

As discussed in the previous section, the 12-bit voltage DAC imposes limitations. The gain stabilization error in response to the temperature step is below 0.25%. But during the temperature drop, the system encounters a domain with a high differential non-linearity of the 12-bit voltage setting DAC, and the gain jumps by 0.5%.

Of course, this jump is tiny compared to the natural width of a 662keV peak of 6% to 8% fwhm.

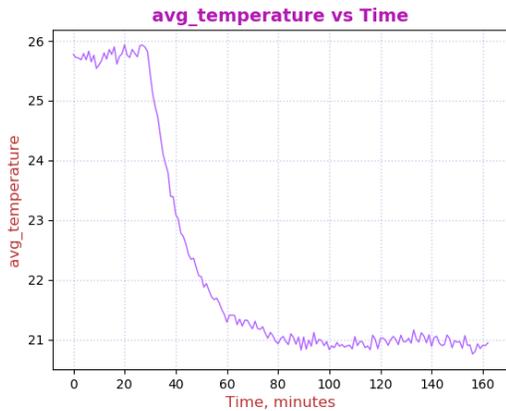


Fig. 13: Measured change in temperature of the SiPM-array in response to a much more rapid change of environmental temperature.

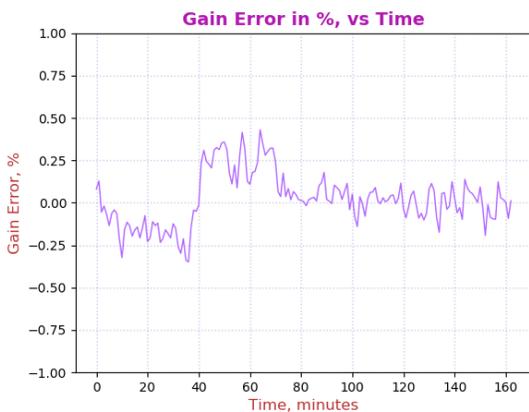


Fig. 14: Measured gain error of the SiPM-array in response to a rapid change of environmental temperature.

## 11. Software upgrades

**Upgrading and updating software:** The MCA-1000 ARM code can be upgraded or updated in the field via its USB connection. This feature allows to deploy code with bug fixes (updates) as well as the delivery of completely new custom software with application specific features and capabilities. All files are fully encrypted. They can be delivered to the developer, and even to the enduser customer without compromising software security. Once loaded into the MCA, the software can not be read back.

### 11.1 ARM code upgrade

**Using the bootloader:** When the user receives a new ARM executable, as an encrypted file, they can load that file using the boot loader. This is a two-step process.

**Step 1 Invalidate App:** Using the examples in the examples/software\_update folder, first invalidate the running application by executing the invalidate\_app.py file. This sets a bit in non-volatile memory. Then power cycle the device or force a CPU restart (arm\_restart.py)

**Step 2 Uploading code:** After the ARM CPU reboots, the MCA will be recognized by the computer's operating system as a USB device with Bridgeport's vendor ID, but now with a new product ID, namely PID=0x1000. Close the MCA Data Server.

**On Windows:** If you are performing the procedure for the first time on a Windows machine and are using libusb0.1, you must run wxMCA/zadig.exe and install libusb\_win32 as the device driver. From then on Windows will remember the driver for this new device. On Linux no action is necessary.

**Relaunch MDS:** Launch the MCA Data Server again. It will now report a PID of 0x1000, and it will report the same serial number that the MCA already had when shipped. The serial number does not change in the process.

**Uploading code:** Copy the encrypted arm executable to the software\_update/data folder. Edit the file name in the update\_arm.py file and execute that file. A typical code is uploaded in 10 seconds, but the maximum time for the biggest possible code size is 80 seconds.

**Relaunch MDS:** After the code has been uploaded, the computer will register a disconnect and reconnect series of events on the USB bus. This happens because the updated application relaunches the USB interface. Exit the MDS and launch the MCA Data Server again. It will now report a PID of the MCA, and the same serial number as before.

**Step 3 Validate App:** Test the new application to make sure it works, then validate the application. Execute the validate\_app.py file. It issues a command to clear the invalid/valid bit. With that done, the internal bootloader will automatically boot into the application after the next power cycle.

**Risks:** There is no risk to permanently disable the device. When BPI distributes an update, BPI will also distribute a safe fall back code. If

for some reason the processor fails to boot into the application after the code update, simply power cycle the device. Since its program memory has been declared invalid, the boot loader will simply wait for the next attempt to upload code via USB.

## 12. Mechanical

### 12.1 8-Pin Connector

The PMT-1000 uses a Bulgin PX0447 mini-B USB connector for power and USB communication. Mating cables are the Bulgin PX0441 (USB-A) and PX0442 (USB mini-A) series cables. They come in lengths from 2m to 4.5m.

The PMT-1000 uses a Switchcraft EN3P8MPX 8-pin connector for GPIO and serial UART communication. The mating connector is part of the EN3C8 series.

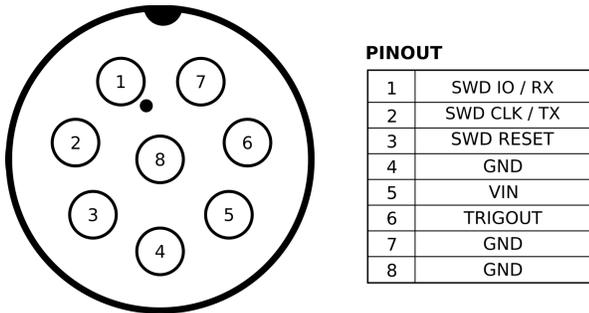


Fig. 15: Pinout of the EN3P8 connector.

## 13. Product and Part Numbers

### 13.1 Product numbers

The PMT-1000 is part of a product series that all contain an ARM M0+ 32-bit processor for communication and slow control, such as gain stabilization. In the -1000 series the MCA is implemented by software in the ARM processor. In the -3000 series the MCA is implemented independently within an FPGA for very high-speed operation. The -3000 series also uses a waveform-digitizing ADC and offers detailed pulse capture and real time pulse shape discrimination.

Both types of devices are available for vacuum photomultiplier tubes (PMT) and Si-

photomultipliers (SiPM). In both cases, the device generates the operating voltage for the photo-sensor from the incoming 5V.

<i>P/N</i>	<i>Sensor</i>	<i>FPGA</i>
PMT-1000	PMT	No
SiPM-1000	SiPM	No

Table : Part numbers.

### 13.2 USB-ID

On the USB bus devices are recognized by their Vendor ID (VID), Product ID (PID) and Serial Number (SN). The vendor ID for Bridgeport Instruments is 0x1FA4. The Product ID's are shown in the table below. Within a product the serial number is fixed, unless BPI makes a custom device that requires a non-standard driver. Note that simple extensions, such as adding a variable to the controls, does not require a new driver.

The BPI software recognizes individual devices by the unique serial number burnt into each ARM processor. The device reports that when the host reads [arm\\_version](#). The serial number communicated in response to USB setup commands is fixed for each part, to avoid that the host keeps adding every new device to an ever longer list of devices requiring a designated USB driver.

<i>P/N</i>	<i>PID</i>	<i>SN</i>
PMT-1000	0x0101	pmtMCA0001
SiPM-1000	0x0201	sipmMCA0001

Table : Product ID and USB bus serial numbers.

### 13.3 Device serial numbers

Each ARM processor has an immutable 128-bit unique serial number, which can be printed as a 32-character hexadecimal string. The MCA Data Server always uses the complete 32-character string to identify the device. Because of space constraints, the serial number printed onto the device is shortened to 8 hexadecimal characters.