

CapeMCA™ Users Manual

**A Real-Time Gamma Energy Spectrum Monitor
for CapeSym's STM32-Based
USB Multi-Channel Analyzers**

™CapeMCA is a trademark of
CapeSym, Inc.
6 Huron Dr.
Natick MA 01760
USA



Table of Contents

Versions and Compatibility	3	Channel Range for Isotopes.....	33
Windows Application	4	Memorize MCA Parameters.....	34
Establishing USB Communications.....	4	Recall MCA Parameters.....	34
Graphical User Interface.....	5	Factory Settings Restore.....	34
Region of Interest.....	6	Soft Reset of MCA.....	34
Spectrum Peak Locations.....	6	Exit to Bootloader.....	35
Peak Tracking.....	7	MCA Arrays	35
Peak Labels.....	8	Meaning of Array Lights.....	36
Saving a Spectrum.....	9	MCA-to-MCA Data Packets.....	37
Auto Save.....	10	Array Calibration.....	39
Loading a Spectrum.....	12	Calibrations	40
Auto Load.....	13	Step 1: High Voltage Bias.....	41
Record N42.....	13	Step 2: Pulse Threshold.....	41
Load N42.....	14	Step 3: Divisor for Integral to Channel.....	42
Stream Pulses to File.....	16	Divisor Adjustment Procedure.....	42
MCA Firmware	16	Step 4: Temperature Stabilization.....	42
Run Mode Configuration.....	17	TS Calibration Setup Procedure.....	44
Zero Out MCA.....	18	TS Calibration Auto Save Procedure.....	45
Spectrum Data Type.....	19	TS Calibration Auto Load Procedure.....	47
Temperature Stabilization (TS).....	21	TS Calibration Entry Procedure.....	50
Energy Correction (EC).....	21	TS Auto Calibration.....	50
Pulse Pileup Rejection (PPR).....	22	TS Auto Calibration Procedure.....	50
Channel Zero Feedback.....	24	Quick Two-Point Calibration.....	52
DAC and Temperature.....	25	Quick Two-Point Calibration Procedure.....	53
CPI and Baseline.....	25	Step 5: Energy Correction.....	53
CPI and n[] Capture.....	25	EC Calibration Procedure.....	55
Spectrum Updates.....	27	Stabilization and Energy Validation.....	56
Update Interval.....	27	Quick Stability Check.....	57
Source Memory.....	27	Create Stability Record.....	57
Interval Depth.....	28	Step 6: Pulse Pileup Rejection.....	59
Request # of Channels.....	28	PPR Calibration Procedure.....	62
Stop Mode Commands.....	29	Appendix A: USB Device Interface	63
Update MCA Parameters.....	29	Appendix B: File Formats	72
Firmware Version.....	30	SPE Spectrum File:.....	72
Divisor for Integral to Channel.....	31	CHN Spectrum File:.....	73
Detector Index in Array.....	31	N42 Spectrum File:.....	73
Moving Spectrum.....	31	Appendix C: MCA Parameters	76
Communication Interval.....	32	Appendix D: Host Code Examples	79
Pulse Threshold.....	33	Appendix E: Serial UART Interface	80
DAC When TS Off.....	33		
Min. and Max. Pulse Channel.....	33		

Versions and Compatibility

CapeMCA is the real-time gamma energy spectrum monitor for CapeSym's Universal Serial Bus (USB) multichannel analyzers (MCAs). These MCAs are designed to operate continuously, processing pulses and accumulating gamma spectra, pausing only briefly at low frequency of (0.1-10Hz) to check for information requests from the host computer. The MCAs are fully self-contained, generating the bias voltage for silicon photomultipliers (SiPMs) arrays, and automatically adjusting the bias to compensate for the temperature-dependence of the SiPM breakdown voltage and response amplitude.

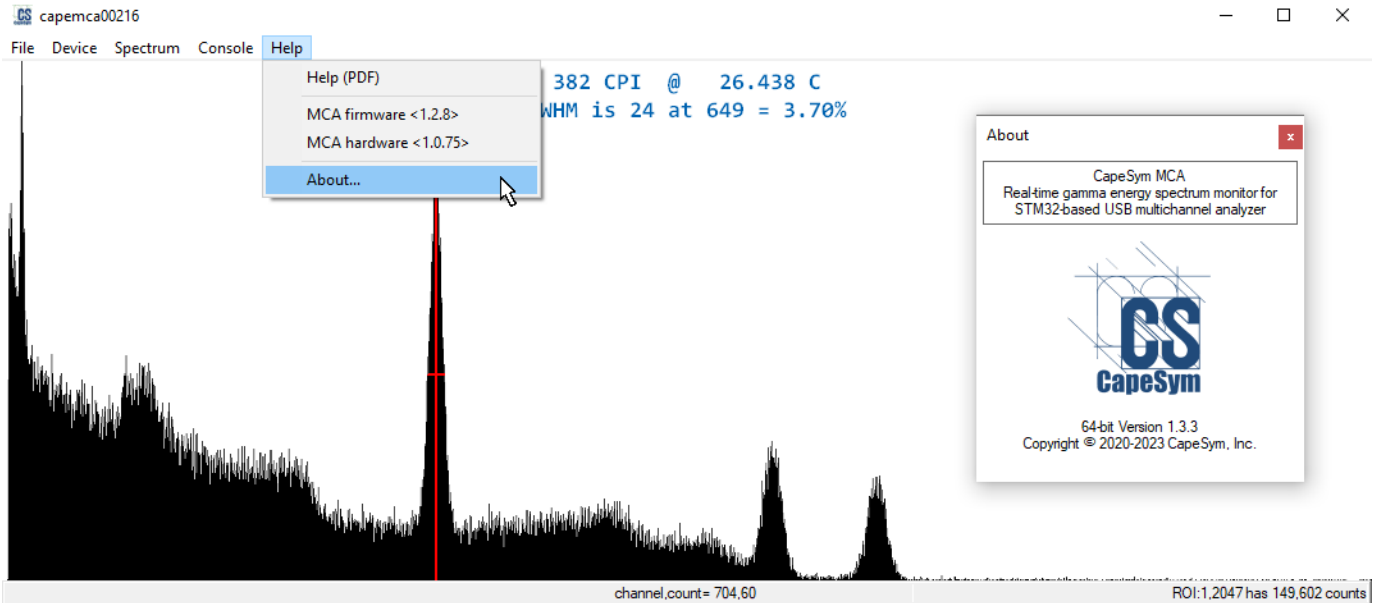


Figure 1: The version number of the Windows software is shown in the About dialog, while the version number of the firmware and hardware are shown in the Help menu after connecting to the hardware.

The CapeMCA software, which runs on the Microsoft Windows x64 desktop operating system, is used to request data from the hardware via the USB 2.0 interface. A 32-bit version is available to support Windows for ARM tablets and other platforms that use Win32 emulation. In this document, the Windows computer will often be called the *host* or *PC*, with the CapeMCA application referred to as the *host software*. The code that runs on the MCA hardware will be called *firmware*. The host software and the firmware will have different version numbers, which are also distinct from the MCA hardware versions.

Table 0: Latest firmware versions and compatible host software feature matrix

Hardware Version	Latest Firmware	Firmware Update	PC Host Version	Features								
				TS	EC	PLM	ADC	Msps	CB	DT	EI	AR
1.0.5x or less	1.0.65	Jan-17-2021	1.1.13	Scaling	PL	no	12	5.56	no	no		no
1.0.62	1.1.11	Oct-02-2023	1.3.4	Voltage	QFE	yes	12	5.56	yes	yes	UART	no
1.1.18	1.2.0	Jul-05-2023	1.3.2	Voltage	PL	no	12	5.56	no	yes		yes
1.0.75	1.3.4	Nov-02-2023	1.3.5	Voltage	QFE	yes	14	up to 10	yes	yes	2xSPI	yes
1.5.2	1.3.1	Jul-06-2023	1.3.2	Voltage	PL	no	12	up to 11	no	yes	2xSPI	yes
1.5.7	1.3.2	Jul-21-2023	1.3.2	Voltage	PL	no	12	up to 11	no	yes	2xSPI	yes

TS: temperature stabilization, EC: energy correction, PL: piecewise linear, QFE: quadratic fit to errors, PLM: pulse list mode, ADC: bit resolution, Msps: million samples/s, CB: capture box count, DT: dead time, EI: external interfaces, AR: array ready








Windows Application

Turn on the MCA device by connecting it to a Windows host computer using a USB cable with a *microB* connector. Windows 10 or higher will automatically recognize the device and setup the USB driver. See Appendix A for more information on USB drivers for older Windows versions or other platforms.

During power-on initialization, a yellow or blue LED on the MCA will illuminate. This LED will turn off in less than a second, signaling the end of the high voltage stabilization period and the onset of processing of gamma radiation. The MCA accumulates a gamma energy spectrum and then pauses to check whether the host has requested that the spectrum be sent via USB. If so, the spectrum is transferred to the CapeMCA host application while the MCA returns to pulse processing. Pulse processing continues until the next communication interval. At each pause for communication, the MCA turns on the green LED (whether or not data is transferred), and then turns it off again when pulse processing continues.

When the MCA is functioning properly at power-up, a yellow or blue LED will appear briefly followed by the green LED blinking periodically (Table 1). The blinking frequency is determined by a parameter in the MCA firmware, which is usually set to 1 Hz at the factory. The MCA is powered by the USB 5V supply, so if no LEDs are visible, check the USB cable connections. In array configurations involving multiple MCAs, LED timings and meanings may be different.

Table 1: LED Indication of MCA Status

LED Color	Duration/Freq.	MCA State
 Yellow or Blue	~0.75 second	Immediately following power-on or soft reset
 Blue or Yellow	~0.25 second	Start up from stop, during HV stabilization
 Green	0.1-10 Hz	MCA running, communication phase
 Red	continuous	MCA not running, awaiting host instruction
 Blue, Red, Yellow	0.1-10 Hz	MCA in array running but not synchronized
 Off	0.1-10 seconds	MCA running, data acquisition phase
 Off	continuous	Bootloader is running, power-on reset required

Proper operation of the MCA requires that a detector module is also connected. ***The detector module should remain connected to the MCA at all times*** while the MCA is powered. Connecting the detector module while the MCA is already powered will produce erroneous readings from the sensor. In some cases the detector module attachment to the MCA is permanent. Attempting to detach the detector from the MCA in such cases will destroy the instrument, even when USB power is not connected.

Establishing USB Communications

Start the CapeMCA application. Establish USB communications with the MCA hardware by selecting the *Connect* item from the Device menu. If multiple MCA devices are connected to the computer a dialog will appear allowing selection of one of the MCAs. Details about the success or failure of the USB device connection are then reported to the console. Multiple tries at establishing a connection may be required under some circumstances.

The spectrum will be displayed a few seconds after establishing the USB connection. This delay depends on the communication interval setting in the MCA firmware. After a brief delay, the CPI (Counts Per Interval) text at the top of the window should begin updating at every communication interval. If no updates are being reported, try selecting *Disconnect* from the Device menu, and then try *Connect* again. If updates still fail to appear in the GUI, try unplugging the microB connector from MCA hardware and plugging it back in, then retry Device > *Connect*. Sometimes the LEDs on the MCA may indicate 5V power connection even though the USB data lines are not fully engaged. This might be true if the connector is not inserted all the way, or if a USB charger or charging cable is used that does not have data lines.

Do not allow the force of plugging or unplugging the USB cable to be transmitted to the detector. Stress on the detector module could damage its seal causing the detector to degrade due to air exposure.

If the spectrum still does not appear, verify that an LED turns on when the USB cable is first plugged in. After this LED turns off, if the green LED does not start blinking, it is possible that MCA communication interval was set to such a large value that MCA will not attempt to communicate with the host for a several seconds. In this situation it will be necessary to wait for multiple communication intervals in order to stop the MCA and restore factory settings, because host commands are processed at a rate of only one command per communication interval.

When a connection is established to the MCA hardware, the *Disconnect* menu item becomes available in the Device menu. The *Disconnect* item will break the USB communication link without disrupting the behavior of the MCA hardware. As long as the hardware remains plugged in to the USB power source, it will continue to accumulate a spectrum according to the current settings. The user may then use *Connect* to re-establish the communication link and retrieve the accumulated data. The USB cable may be unplugged without first using the *Disconnect* menu item. Physical disconnection will be detected by software.

Graphical User Interface

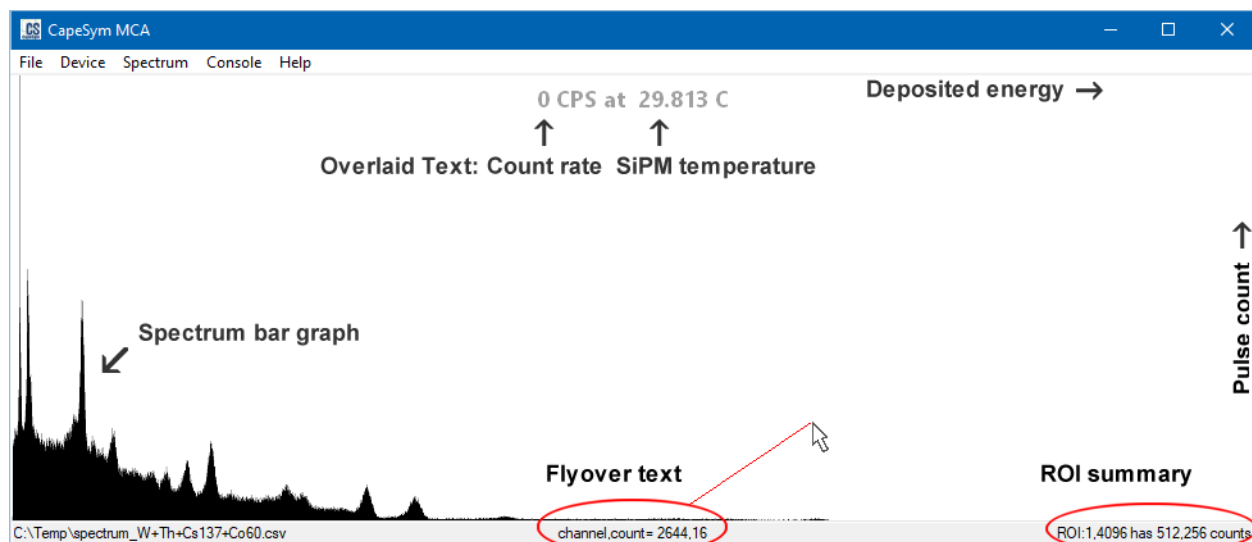


Figure 2: Key features of the graphical user interface are labeled. The horizontal axis represents deposited energy and the vertical axis represents the number of pulses in each energy channel.

As shown in Figure 2, the GUI consists of a window with a menu bar at the top, a bar graph of the spectrum in the middle, and a status bar to the bottom. The height of each vertical bar in the spectrum represents the number of pulses that have been detected at a particular energy level (referred to as a *channel*). So the x-

axis represents deposited energy, each bar width typically spanning one kiloelectron-volt (keV) of energy, and the y-axis represents the pulse count. The longer the data acquisition continues, the larger the count range represented by the y-axis, because the spectrum is rescaled to fit in the window with every update.

If there is too much noise in the signal, the counts in the lowest energy channels may completely washout the energy peaks at higher channels due to automatic rescaling of the y-axis. To recover, exclude the noisy channels from the region of interest (ROI).

During live updates, text overlaid at the top of the bar graph area reports the pulses detected over the prior interval, along with the current temperature of the SiPM array in degrees Celsius (C). When a previously saved spectrum is reloaded into the GUI, the text may display 0 CPI and 0.0 C temperature, because this information was not stored in the spectrum file. When the detector module is damaged or disconnected, the temperature will be replaced with dashes. If a negative temperature is displayed at room temperature, an internal reset has occurred in the temperature sensor inside the detector module and the MCA needs to be unplugged from USB power in order to reinitialize the sensor.

The status bar at the bottom of the main window displays three text fields. On the right side is the name of the loaded file or the most recently saved file. The middle text field displays the channel number and count in the bar under the mouse pointer. This field is continuously updated as the pointer moves in the window. The left side of the status bar displays a summary of counts in the current region of interest.

Region of Interest

The region of interest (ROI) is the range of channels currently displayed in the main window. By default, the entire spectrum is displayed from channel 1 to channel 4095. The total count within the ROI is displayed on the status bar. The ROI may be constrained to a smaller number of channels to focus in on a particular part of the spectrum. The ROI range may be set from the Spectrum menu through the *X-axis range* submenu. When the ROI is modified, the y-axis is scaled automatically to make the largest visible count fill the height of the graphing area.

The fastest way to change the ROI is to move the scroll wheel on the mouse. Rotating the wheel forward zooms the ROI by narrowing the range. Pulling the mouse wheel back zooms out by expanding the range. When zooming in with the mouse wheel, the channel under the mouse pointer is moved towards the center of the ROI. The ROI can be reset to full range by pulling back on the wheel until the scale stops changing.

By dragging the mouse with the left mouse button pressed, a smaller ROI can be selected. The ROI will be set to the channel span of the drag rectangle when the left mouse button is released. The height of the drag rectangle is irrelevant since the y-axis is automatically scaled to the highest spectral component in the ROI. To reset the ROI to the full range use Reset ROI from the Spectrum > *X-axis range* menu.

Normally, the y-axis is displayed as a linear scale, but a logarithmic scale may be chosen instead from the *Y-axis scale* submenu. The y-axis options may be accessed more quickly using the up and down arrow keys. The left and right arrow keys will shift the ROI to the left or right, respectively.

Spectrum Peak Locations

Local peaks in the spectrum of pulse counts may indicate the presence of characteristic photon energies due to photoelectric absorption or x-ray fluorescence events. Peaks may be selected and tracked in the GUI by

clicking on the spectrum with the left mouse button. When the mouse button is released, the software will find the nearest peak and mark it with a vertical line. Multiple peaks may be selected in the GUI by subsequent left mouse button clicks. Unmark peaks by using the right mouse button.

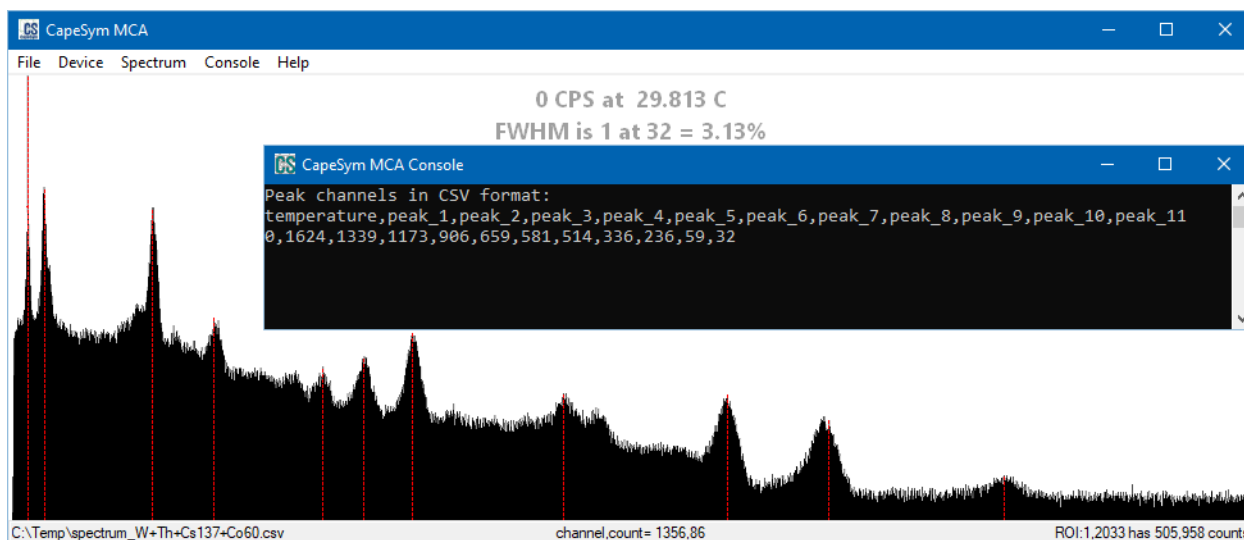


Figure 3: Tracking of energy peaks in the spectrum. Each vertical red line marks the channel location of a local maximum in the spectrum. Channel locations are written to the console.

If the selected peak is tall enough, the full-width at half maximum (FWHM) will also be marked with a horizontal line, and the width (in channels) reported at the top of the graph along with the channel location of the peak. The ratio of FWHM to the peak's channel is also reported as a percentage. When multiple peaks are selected the FWHM computation will only be reported for the last peak selected. FWHM is found by walking down from the peak to find the two channels having less counts than half of the peak counts. No curve fitting is done, so it will only work for sufficiently prominent peaks.

A separate console window serves as a blackboard for textual output, including the channel numbers of all of the selected peaks. The console window is hidden by default at program startup but may be revealed by selecting *Show console* from the Console menu. To print the peak locations in the console, select the *Peaks to console* item from the Spectrum menu. As shown in Figure 3, peaks are listed in order of selection. The first entry in the console output is always SiPM temperature, or zero as a placeholder. Any time the displayed spectrum is updated from the MCA or loaded from a file, a new line of peak location data will be written to the console, for as long as the *Peaks to console* menu item is checked.

Peak Tracking

As new spectral data arrive, the channel locations of the identified peaks will be automatically shifted to the new peak locations, so that the selections continue to track growing and shifting peaks. This is particularly important when temperature stabilization is turned off (or when there is no temperature calibration) because channel location will shift dramatically as the temperature of the SiPMs change. Real-time tracking of temperature-dependent peaks is a very useful feature of the host software, especially during the calibration process, as described later in the manual.

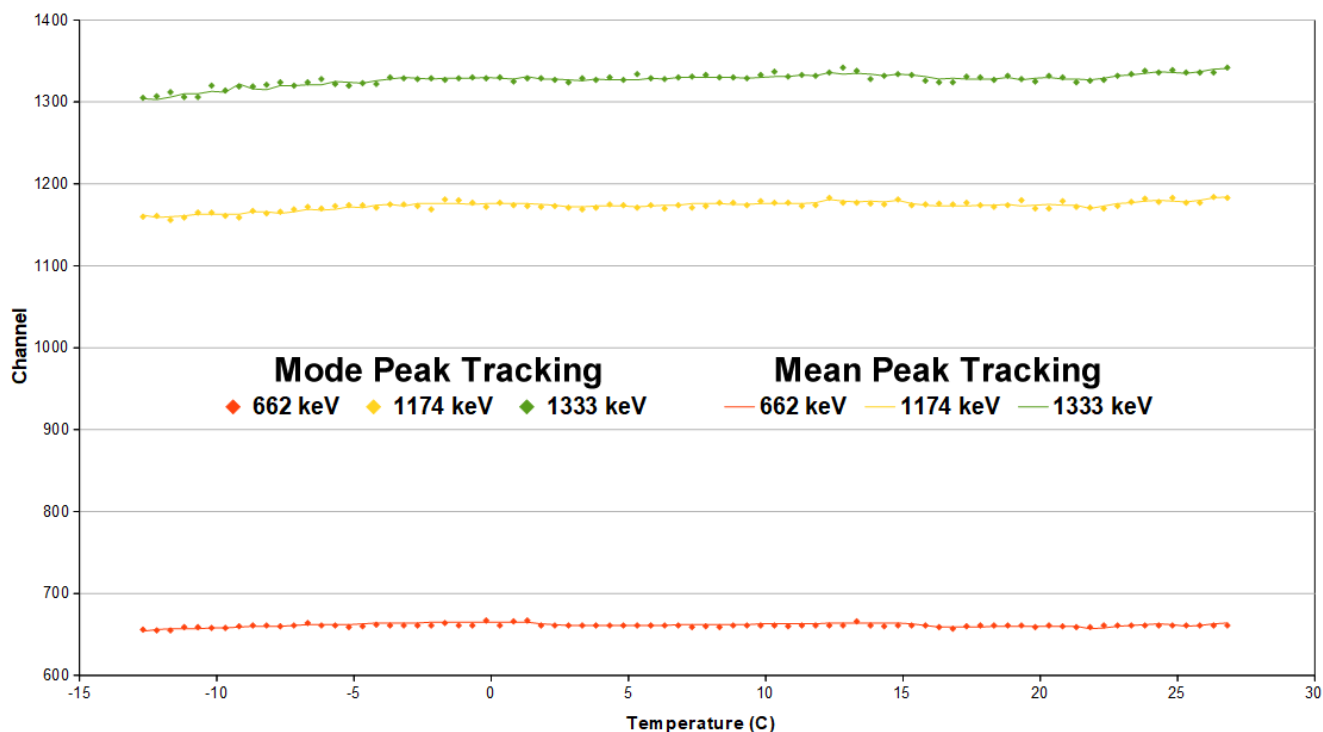


Figure 4: Tracking of Cs-137 and Co-60 photopeaks during a rise in temperature from -13°C to 27°C. The diamond symbols denote the channel location of the mode of each peak, while the lines follow the channel location of the mean of each peak.

Peaks can be tracked in two ways (Figure 4). The peak tracking method is selected by *Peak type* in the Spectrum menu. The default operation is to track the peak maximum, referred to as the *Mode* of the peak. This maximum can be found easily but it is not likely to be the center of the photoelectric absorption energy distribution (a.k.a. *photopeak*). Switching the Peak type to *Mean* causes the statistical mean to be tracked. The statistical mean is computed just for the part of the photopeak above the full-width at half-maximum height. When the underlying probability distribution is normal, the mean will tend to give a channel location closer to the Gaussian maximum. Note that the half-maximum (of the mode) must be above the background in order to track the mean. When the half-maximum level falls below the background, or when two peaks merge above the half-maximum level, mean tracking may lose the peak.

Peak Labels

Spectrum peaks can be labeled with their characteristic gamma energy, expressed as an integer in keV. These peak labels do not change with live updates, unlike the peak channel locations reported to the console. Peak labeling is enabled by selecting the *Label peak when add* from the Spectrum menu. When peak labeling is enabled, each left mouse click will be followed by an input dialog for entering the integer label (Figure 5). The label is then displayed in red text just above the peak indicator line. Peak labels will also appear in the header line in the console instead of the generic categories (peak_1, peak_2, etc.) when the *Peaks to console* mode is enabled.

Labeling may be applied to some peaks and not others. Just hit the Cancel button when the label dialog appears, as shown above. Peak labels are cleared (along with the peak lines) with the right mouse button. Disable peak labeling by unchecking *Label peak when add* in the menu.

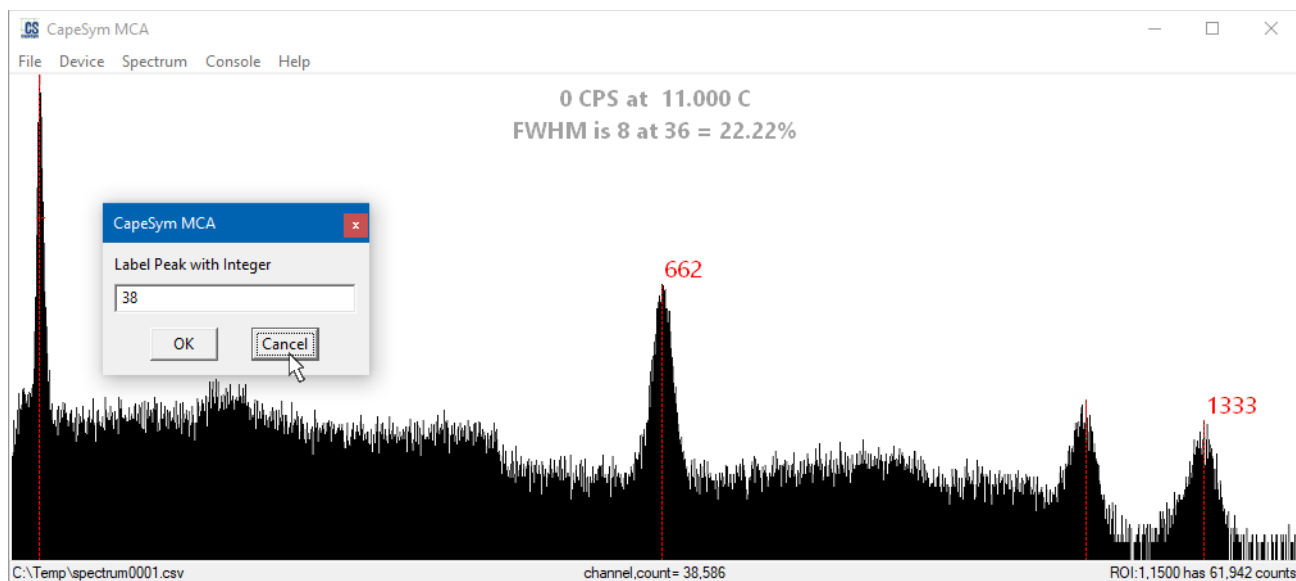


Figure 5: A left mouse button click on the far left peak in the spectrum opens a dialog for adding an integer label. The channel under the mouse pointer is offered as the label, but this can be changed. The Cancel button will skip the label while still marking the peak location for tracking.

Saving a Spectrum

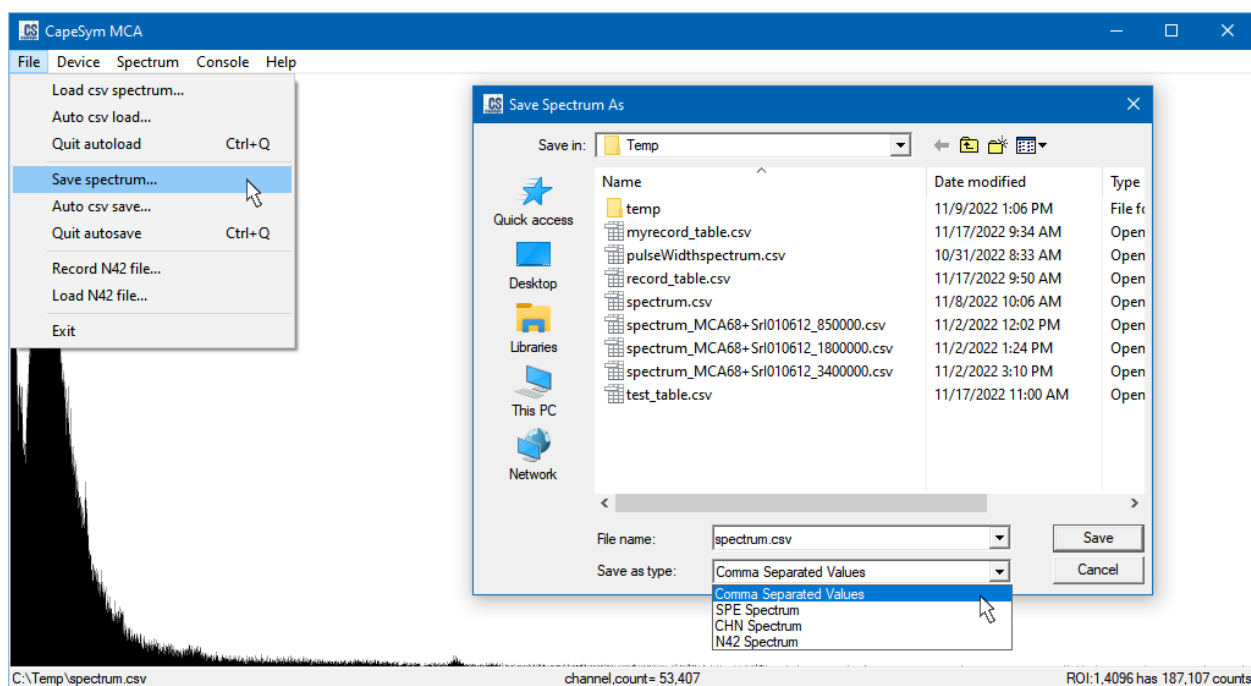


Figure 6: One way to save the spectrum to a file is to use the Save spectrum... menu item from the File menu. Other ways to save: record spectra to an N42 file or use the Auto Save feature.

Any spectrum displayed in the GUI may be saved to a file by using the File menu (Figure 6). It is not necessary to *Disconnect* the MCA to save the current spectrum, however the spectrum will continue to

update every second even while the save dialog is open. Disconnecting the MCA will stop further screen updates and allow a particular observation to be saved, while the MCA continues to count pulses.

The format to use for the spectrum file may be changed through the file type selection. The default file format for a spectrum is Comma Separated Values (CSV). For a different file format, use the *Save as type:* list box in the dialog to select from the list of available file formats, such as SPE, CHN, and N42.

The CSV format is written in plain text, each line of text containing the channel number and count separated by a comma. The first channel is 1, so there is no count rate or SiPM temperature information included in the CSV file when using the *Save spectrum...* menu item. This information is encoded in channel zero during live updates. When using Auto Save mode, the zero channel may be optionally added to all CSV saves so that every spectrum will include temperature, which may be important for assessing temperature stabilization after re-calibration.

The N42 file format that is written using *Save spectrum...* contains only channel counts and temperature. For the fuller N42 spectrum format, including real- and live-time durations, use the *Record N42 file...* menu item. This menu item is discussed further later on. See Appendix B for complete file format descriptions.

Auto Save

The Auto Save feature is designed to monitor live updates from the MCA and save the spectrum to a file once certain criteria are satisfied. Auto Save will only operate when live updates are coming from the MCA hardware. Therefore the MCA device must be connected (and not stopped) in order for Auto Save to function properly. However, depending on the save criteria, it may be possible to disconnect or stop the MCA briefly during an on-going Auto Save operation, if necessary.

The dialog provides several different options for saving spectrum files, including saving a sequence of spectra acquired at different temperatures as required for evaluating temperature stabilization performance. The Auto Save function will only save files in the CSV format, because the files are intended to be reloaded via the Auto Load function to provide peak location data for the calibration. The option to save the SiPM temperature in channel 0 is also needed for the performance evaluation.

When Auto Save saves a file it appends a number to the filename, so that the sequence of files can be easily reloaded. If the filename already ends in a number, that number is removed and the next number in the sequence is applied to the filename. The very first number to use for the sequence is specified in the Auto Save dialog.

The Auto Save dialog entries may be saved with the OK button, without starting the actual Auto Save operation. The Auto Save operation is only started when the ENABLE AUTOSAVE box is checked. Hitting the Cancel button allows the dialog to be exited with keeping any changes. As long as the Cancel button is used, the Auto Save dialog may be opened during an on-going Auto Save operation. Hitting the OK button while an operation is on-going will cause the operation to restart at the beginning of the sequence, or to stop if the ENABLE AUTOSAVE box is unchecked.

The progress of the Auto Save operation is reported to the console, so it may be helpful to have the console open while auto saving. To stop an on-going Auto Save operation the ENABLE AUTOSAVE box may be unchecked in the dialog (followed by OK), or the *Quit autosave* menu item may be selected.

In addition to saving a sequence of spectra for temperature calibration, the Auto Save operation may be

used to perform single or multiple timed acquisitions, or to perform single or multiple acquisitions of fixed numbers of pulse counts. These options are explained with a few concrete examples of dialog settings.

To accumulate and save 90-seconds of data acquisition, use the Auto Save dialog settings shown in Example 1. The current spectrum will be zeroed out and a new spectrum accumulated for 90 seconds before saving it to a file called spectrum0001.csv.

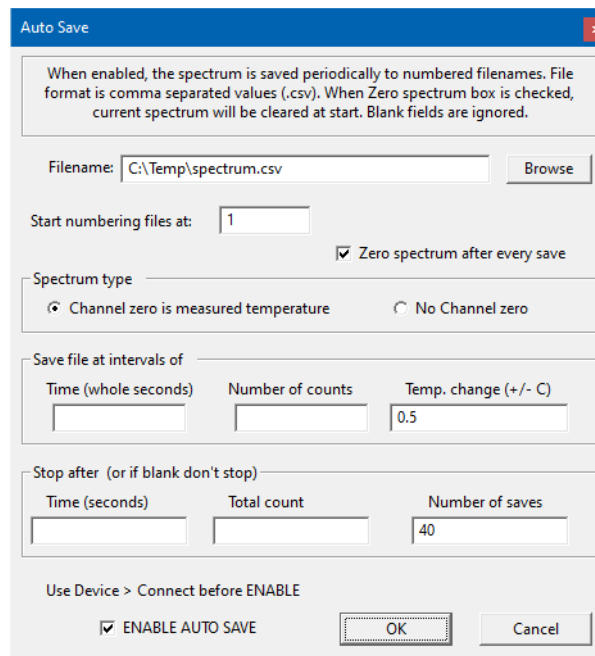
Example 1: 90-Second Spectrum Acquisition

Example 2: Save Spectrum at 100,000 Counts

Example 2 cause the current spectrum to be saved to a file once the total count reaches 100,000. The Auto Save operation is then terminated because the *Number of saves* is set to 1. If the *Number of saves* was set to a higher number, another file would be saved every time 100,000 counts were added to the spectrum. If the *Number of saves* was left blank, files would continue to be added until the user disabled the Auto Save operation. If the desire was to save a new spectrum of 100,000 counts each time 100,000 pulses were detected, then user should have checked the box next to *Zero spectrum after every save*, so that a fresh spectrum would be started for each 100,000 count accumulation.

The following dialog settings may be used for evaluating the accuracy of temperature stabilization. This Auto Save configuration will save and then zero out the spectrum after every 0.5°C increase in temperature, until a total change of 20°C has been recorded. Note that temperature change entered as a positive number causes a save to occur for only increases in temperature. For saves at temperature decreases instead, a negative value -0.5°C should be entered.

Again, to continue saving indefinitely, the *Number of saves* box could be left blank. In which case the Auto Save feature would be terminated when the user selects the *Quit autosave* item from the File menu. This command is a shortcut for opening the Auto Save dialog, unchecking the ENABLE AUTO SAVE box, and hitting the OK button. The keystroke Ctrl-Q is even quicker.



Example 3: Record Spectra vs. Temperature

Loading a Spectrum

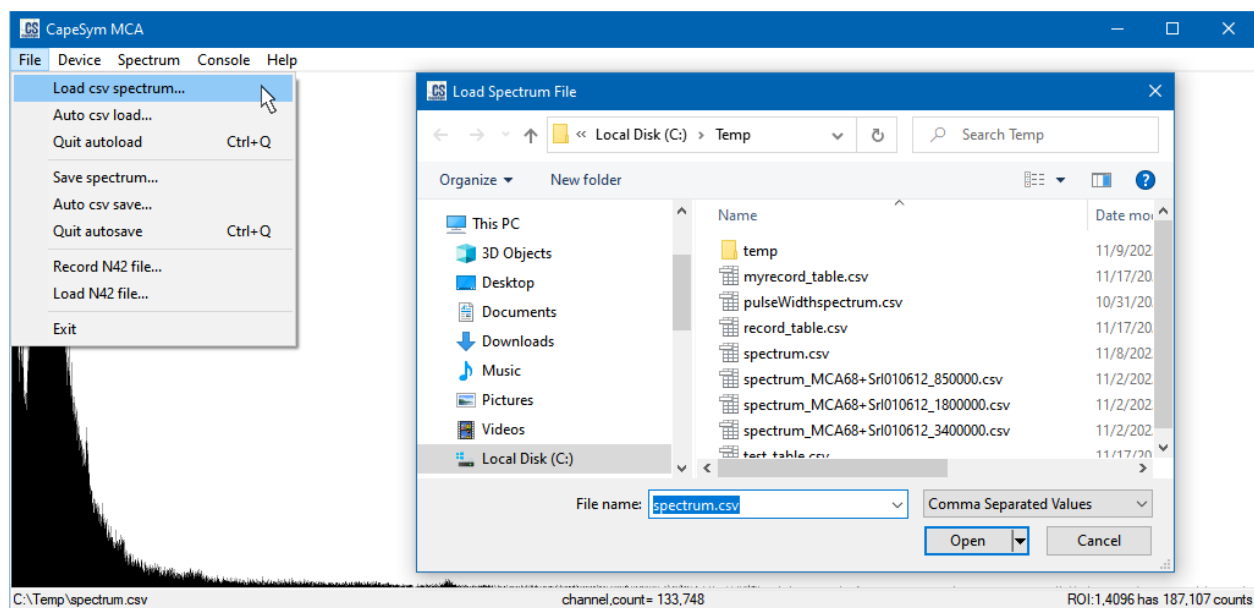


Figure 7: The Load spectrum... menu item opens a dialog that allows browsing to a .csv file.

File > Load csv spectrum... can be used to load files saved by the application in the CSV format. The MCA device should be disconnected before using the load operation, otherwise the spectrum will be immediately over-written by the next live update. To facilitate disconnection, a prompt is issued before file selection. Responding with "No" to the disconnect prompt will terminate the load operation.

Auto Load

The Auto Load dialog may be used to play back a sequence of spectra recorded using the Auto Save mechanism. Auto Load can be used to create an animated display loop, but its main purpose is to extract peak locations versus temperature for assessing the temperature calibration.

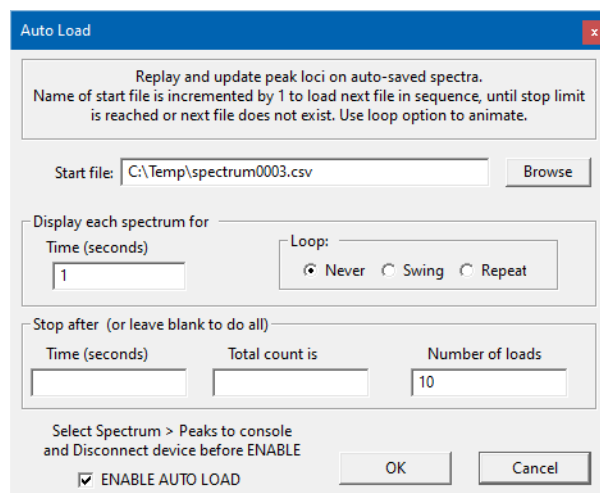


Figure 8: The Auto Load dialog for loading a sequence of spectra saved by Auto Save.

The first file to be loaded is given in the Start file edit box. In Figure 8, the start file is "spectrum0003.csv". That file will be loaded and displayed for 1 second, and then the filename will be incremented and the next file "spectrum0004.csv" loaded and displayed for 1 second. After the tenth file has been loaded and displayed, in this case "spectrum0012.csv", the Auto Load sequence will be terminated. The end of the Auto Load operation is reported in a message box.

To repeat the sequence continuously until the *Quit autoload* command is selected, use one of the other *Loop:* options and leave the *Number of loads* box empty. The *Repeat* option will restart the sequence at the start file for each loop. The *Swing* option will reverse the direction of the file loads at the end of the sequence. File number increments will become decrements until end of the sequence and then the direction will again reverse. Note that it is not possible to Swing repeatedly on a sub-sequence of the filenames. With the exception of the starting file, the animation will move from the last file to the first file in the numerical sequence and back again, forever, until manually disabled.

Record N42

The *Record N42 file...* menu item allows multiple radiation measurements to be recorded into a single .n42 file using the ANSI N42.42 (2011) standard with CapeSym-specific extensions. These measurements can then be replayed using the *Load N42 file...* operation. The N42 format is a text-based XML syntax detailed further in Appendix B.

To record to an N42 file, first select a new filepath with the Browse button in the dialog. If an existing file is selected, it will be over-written without a warning prompt. It is not possible to add measurements to an existing N42 file recorded in a prior session. All recordings must be added before exiting the dialog. The recording file should have the extension .n42 to signify the N42 XML standard file type.

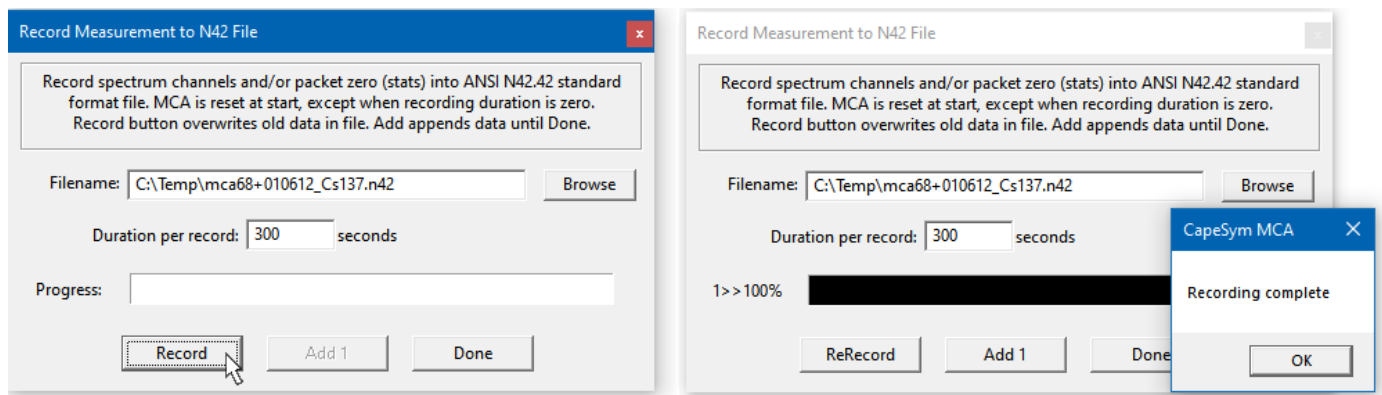


Figure 9: Recording radiation measurements in N42 file format. Start recording with the Record button (left image). After the recording is complete (right image), add another recording to the file using the Add 1 button, or close the N42 file (and complete the N42 syntax) by using the Done button. Using ReRecord erases the entire file and starts a new recording.

Select a duration in whole seconds for the recording. When the Record button is pressed the file will be created and the MCA will be zeroed out. The software monitors the passage of time in the host computer until the specified duration has passed, and then writes the MCA data to the file. The Record and Add buttons are disabled, and the rest of the menus are inaccessible for the duration. The only allowed action is the Done button, which will terminate the recording and close the N42 file.

<< The N42 file format is not valid until the Done button is pressed >>

If the duration is set to 0 seconds when the Record button is pressed, the MCA and spectrum are not zeroed. Instead, the current data are immediately written to the file. A spectrum previously saved in CSV format may be loaded and recorded to N42 by using the zero duration option, when the MCA is disconnected. Some information will be inaccurate for zero duration recording, because accurate values are only obtained from packet0 feedback during live recordings. No matter what duration is chosen, the type of data recorded depends on the type of requests currently being made to the MCA. When the software is requesting spectra of 1024 channels, for example, only 1024 channels are recorded.

After the first recording has completed, additional recordings may be added to the N42 file by using the Add1 button. There is no limit to the number of recordings that may be added to a file, and a different duration can be applied to each added measurement. The N42 recording file will contain the same type of measurements for all recordings, because the type of data request can only be changed by exiting the dialog. However, spectra with the same number of channels may be recorded at different times, and for different durations.

Load N42

To examine the contents of any .n42 file recorded by the application, use the Load N42 file... menu item. This operation does not support all types of N42 formatted files, it only accepts files created with the XML extensions provided by CapeSym. These extensions are defined in an XML schema (Appendix B).

When an N42 file is loaded, the table of measurements is displayed, as shown in Figure 10. Each row corresponds to a measurement contained in the file, as denoted by the identifier (e.g. M1), along with the date, time and duration of the recording. The other fields in the table depend on the request type specified by the Spectrum menu when the recording was done. If the measurement includes a spectrum, then the

table will display the number of channels recorded. If no spectrum is present in the measurement, the number of channels will be zero. If the request type included the statistics from packet0 (see below), then columns E through J will contain those numbers. "NA" in those columns means the packet0 feedback was not requested when the measurement was recorded.

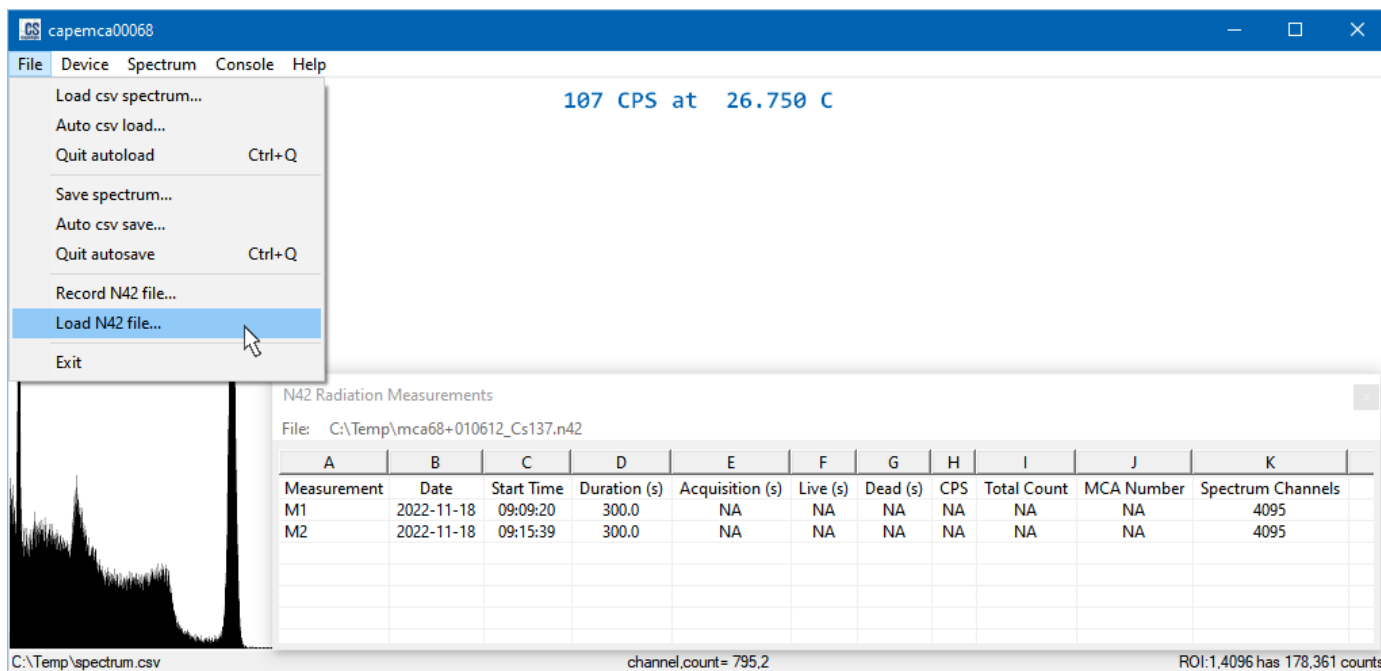


Figure 10: Table of radiation measurements from one N42 file as revealed by Load N42 file...

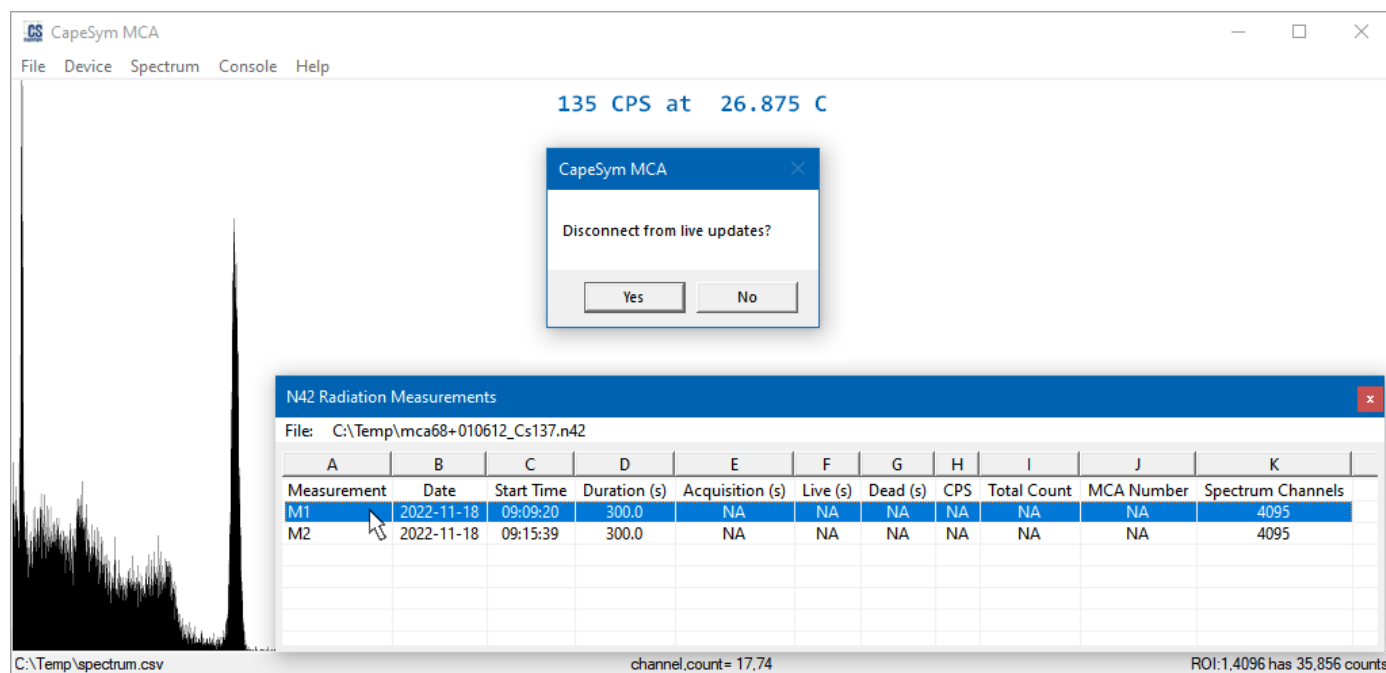


Figure 11: Clicking on one of the rows with the left mouse button will display the spectrum and other measurement information in the main window. If the MCA is connected, then the prompt to disconnect must be answered with the "Yes" button before the measurement data will be displayed.

Measurements in the table can be displayed in the main window with a mouse click, but only when the MCA is disconnected (Figure 11). The display of the measurement depends on the type of request configured in the Spectrum menu, because this determines how the main window is drawn. For example, to see the packet0 statistics, one of the request modes that includes packet0 must be checked in the *Spectrum* > *Request # of channels* submenu, otherwise these statistics will not be drawn in the window.

The File menu at the top of the N42 Radiation Measurements table may be used to open another N42 file. This is the same operation as accessing the *Load N42 file...* menu item in the main file menu. The table's File menu also allows the table itself to be saved in CSV format. This is the only way to export the table. The table cannot be copied to the clipboard or otherwise edited. Using Ctrl-C on the table may cause the Connect operation to be executed in the Device menu.

Stream Pulses to File

Another way to save data is to use the streaming dialog to append successive readings into a single file. But this is only appropriate when the data is not accumulative. Streaming is designed to work with pulse lists which only report the events that happened since the previous communication interval. If no events happen in a particular interval, then no data is appended to the file. Figure 12 shows the streaming dialog being used to accumulate 25 pulse lists.

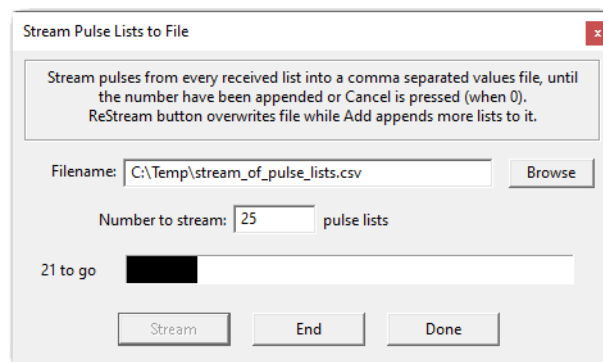


Figure 12: Streaming a sequence of pulse lists to a single file.

In contrast, the MCA is already accumulating events over time when the spectrum data type is *Pulse counts*, and this accumulated total is returned each communication interval. So streaming these spectra to a file would mostly collect the same data over and over again. It possible prevent this data overlap, by enabling moving spectra (see below) and setting the depth of moving spectra to zero. With these settings, all the spectrum data types of *Pulse counts*, *Pulse widths* and *Pulse sample* are zeroed after each communication interval, such that only events occurring since the prior interval are included. For example, setting the spectrum data type to *Pulse sample* and enabling moving spectra (=1) with depth=0 allows a specified number of pulses to be recorded to a single CSV file.

MCA Firmware

The MCA firmware contains many parameters that can be accessed and changed through the Windows application. These operating parameters are retained in non-volatile memory while the MCA is powered off. When power is applied, the MCA parameters are loaded into random-access memory (RAM) to facilitate frequent updating while the device is powered on. Connecting to the MCA from the host software

causes the RAM copy of the parameters to be transferred from the MCA to the host (Figure 13). The parameters may then be viewed and modified in the host software, then sent back to the MCA from the *Update MCA parameters* dialog. The new parameters must be explicitly memorized within the MCA to survive the next reset or power off/on cycle.

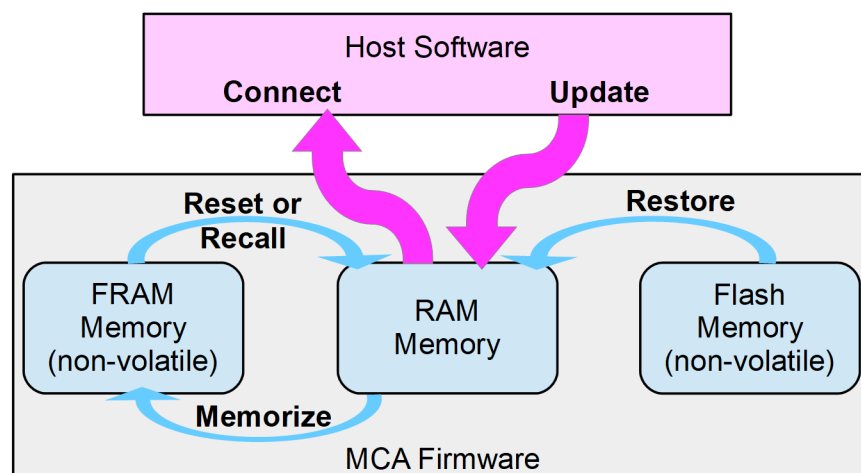


Figure 13: Operations for transferring the MCA parameters.

The user can modify some of the parameters from the *Run mode configuration* submenu of the Device menu, such as turning on and off the temperature stabilization, energy correction, and pulse pileup rejection. These changes are written to the MCA RAM memory at the next communication interval. These changes will not be retained in non-volatile memory when the device is powered off unless they are memorized.

Many of the parameters are modifiable only through the *Stop mode commands* and *Calibrations* menus. The *Calibrations* menu facilitates modification of parameters that provide the MCA with accurate behavior, including those that control pulse pileup rejection, energy correction, and temperature stabilization.

When the USB communications are disconnected, by the *Disconnect* menu item, the host software can no longer adjust the parameters inside the MCA. When the MCA is reconnected the active RAM parameters are again transmitted to the host software from the MCA. Any settings or calibration changes made in the host software while the MCA was disconnected will be over-written by the *Connect* menu item. To preserve parameter changes, the user must transfer them to the MCA before disconnecting.

When power is removed, the MCA parameters in RAM are discarded so that the device will return to the default state of operation when powered back on. To retain the MCA settings during power-off, the parameters in RAM must be transferred to the MCA's non-volatile memory before unplugging it. This memorization of the MCA parameters is done from the *Stop mode commands* menu, which is only available when the MCA is taken out of Run mode and put into Stop mode.

Run Mode Configuration

The default mode of operation for the MCA device at power-up is *Run mode*. In Run mode, the data acquisition and pulse processing loop is running almost all the time. There is a short break in pulse processing at regular time intervals to allow the accumulated pulse data to be transferred to the host

application via USB. This break in processing lasts for a couple of milliseconds, depending what other processing is enabled. For example, if a moving sum of spectra is being used, this sum must be updated before the spectrum is transmitted. The sum of spectra can require an additional 8 ms to compute, depending on the number of spectra being added together.

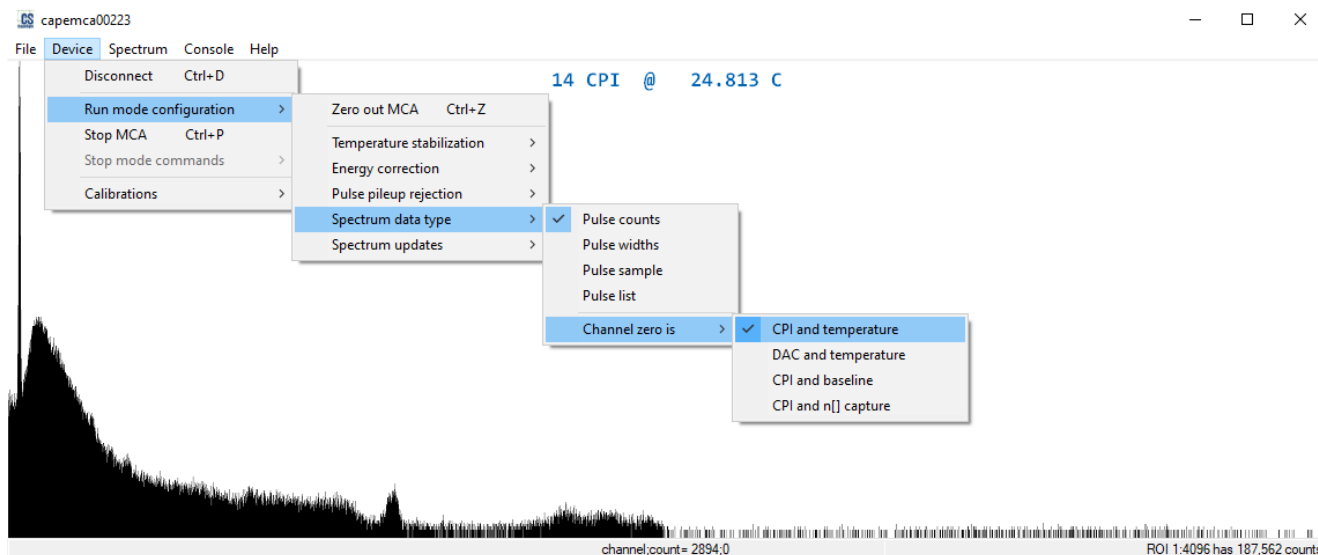


Figure 14: Run mode configuration menu allows some MCA operating parameters to be changed on the fly, including turning on and off the temperature stabilization, energy correction, and pulse pileup rejection, and changing the type of data to be returned as a spectrum and displayed in the window.

As noted above, the *Run mode configuration* menu allows several MCA operating parameters to be changed while the MCA is in Run mode. Only one parameter change can be made during each break for communications, and only if data has not been already requested. Requesting data prevents any other command from being processed until the next communication interval.

<< Only one command is processed per communication interval >>

Change requests sent from the host software are processed by the MCA firmware at the end of the communication interval. For example, with a communication interval of 1 second, the *Zero out spectrum* command would be processed within about a second, and then another second would be required to get the first spectrum of new data. Attempting to make several requests within a single communication interval will result in all but one of the requests being ignored by the MCA.

Changing the communication frequency to 10 Hz would allow 10 commands to be processed per second, while changing the communication frequency to 0.1 Hz would allow one command to be processed every 10 seconds. Starting in version 1.3.5 of the host software, the communication interval can be changed while the MCA is in Run mode, through the *Spectrum updates* submenu, as discussed further below.

Zero Out MCA

The MCA continuously accumulates a gamma energy spectrum in its default operating mode. A count is accumulated forever in each of the 4095 energy channels, as well as in the total count of packet0. Each channel counter is a 32-bit unsigned integer that can reach 4,294,967,295 ($2^{32}-1$) after which the channel rolls over to zero. The packet0 counter uses a 32-bit floating point value that avoids rolling over but sacrifices some accuracy at such large numbers. All of the counters can be zeroed out from the host

application by issuing the *Zero out MCA* command. The MCA is automatically zeroed on reset, or whenever the spectrum data type changes. The MCA may also be zeroed periodically by the Auto Save or N42 recording functions. Neither the spectrum nor packet0 are retained when the MCA is powered off.

Spectrum Data Type

The type of spectrum data collected by the MCA and transmitted over the host interface is determined by two main parameters: *Spectrum data type* and *Feedback type*. The feedback type parameter determines the contents of channel zero of the spectrum, while the spectrum type parameter determines what information is in the other 4095 channels. An additional information packet that summarizes the MCA activity, but contains no channel information, may also be transmitted, depending on the number of channels requested by the host computer. Requesting zero channels results in only the summary packet being sent, hence it is referred to as “packet zero” or *packet0* in the user interface.

Table 2: Types of spectral data transmitted by the MCA over the host interface

Spectrum data type	Channel x (> 0) is	Spectrum value at channel x is
<i>Pulse counts</i>	Deposited energy	Count of all pulses with x energy
<i>Pulse widths</i>	Deposited energy	Width of most recent pulse with x energy
<i>Pulse sample</i>	Sample index	ADC level of unprocessed signal buffer
<i>Pulse list</i>	Pulse index (two channels)	Interarrival interval (in samples), integral of pulse

Two types of energy spectra may be accumulated within the MCA: the spectrum of pulse counts and the spectrum of pulse widths. The x-axis for both types of spectra is related to the total, above-threshold energy in the pulse, as measured by integrating the raw pulse signal. The y-axis is what distinguishes these two spectrum types. In one case the y-axis is the number of pulses, and in the other case the y-axis is the above-threshold width of a single pulse.

The default MCA setting will accumulate and return pulse counts for each energy channel. The type of returned data may be changed by switching the *Spectrum data type* (Figure 14). Selecting *Pulse widths* returns the width of the mostly recently recorded pulse in each channel. (If no pulse has yet been recorded in a channel, the width for that channel will be zero. An actual width of zero is impossible.) The width is expressed as the number of sampling intervals for which the pulse stayed above threshold. At a sampling rate of ~5.56 million samples per second, each sample interval is about 180 ns, so a pulse width of 200 samples corresponds to ~36 μs.

The *Pulse sample* spectrum type allows the user to observe one instance of the sample buffer. The x-axis in this case represents the sample number and the y-axis is the the raw signal level acquired by the analog-to-digital converter (ADC). This MCA output is primarily for debugging the raw pulse signal by directly reading out the sample buffer. For example, it is possible to judge the amount of dark current by the baseline signal level when no pulse is present (Figure 15).

The *Pulse sample* spectrum data type does NOT provide continuous readout of all pulses, because the sample buffer contains way too much data to transfer to the host software in real-time. With this output, the MCA digitizes the signal, and detects and processes pulses as it normally would, but when a pulse is detected within the minimum and maximum channel range parameters, the buffer of 4095 samples containing the pulse is copied to the spectrum array. There is no accumulation of spectral data. At the end of the communication interval, the processing is stopped and the data displayed is the most recent buffer

captured. The CPI reflects the number of pulses detected during the entire interval, not the number in the displayed buffer.

The minimum and maximum channel range values can be changed by *Update MCA parameters...* in Stop mode. While the spectrum is limited to 4096 channels, much larger pulses will be detected and counted by the MCA. These pulses can be observed in Pulse sample mode by setting the minimum and maximum channel range beyond 4095, for example to 5000 and 10000 respectively.

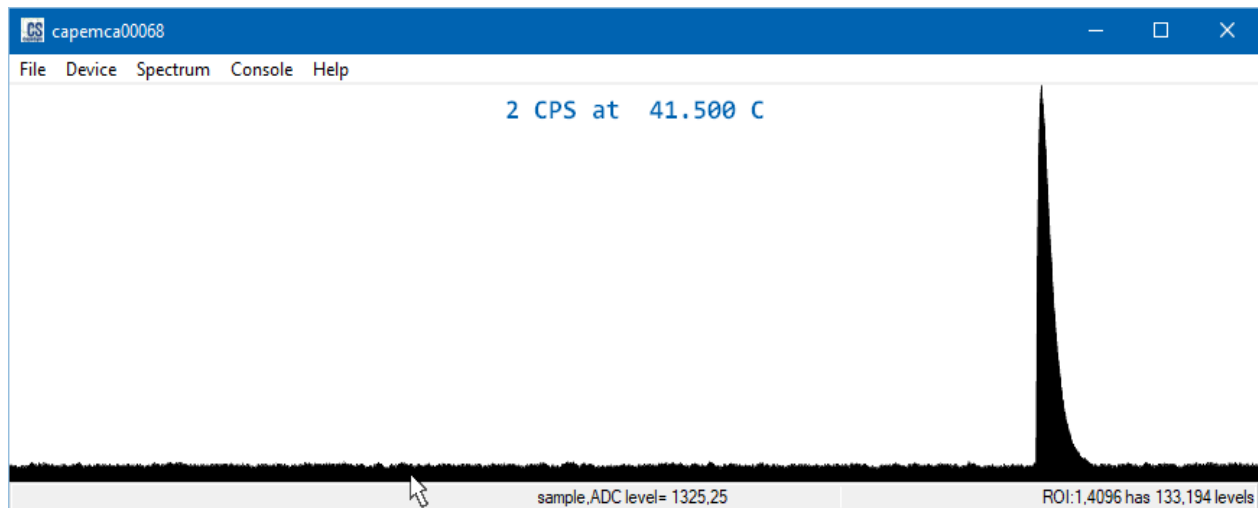


Figure 15: The baseline is the average value of the SiPM signal when no pulse is present, as shown here in Pulse sample mode. ADC level at the pointer is 25, which is a relatively high baseline of $3.3V \cdot (25/4095)$ or about 20 mV, assuming 12-bit ADC resolution.

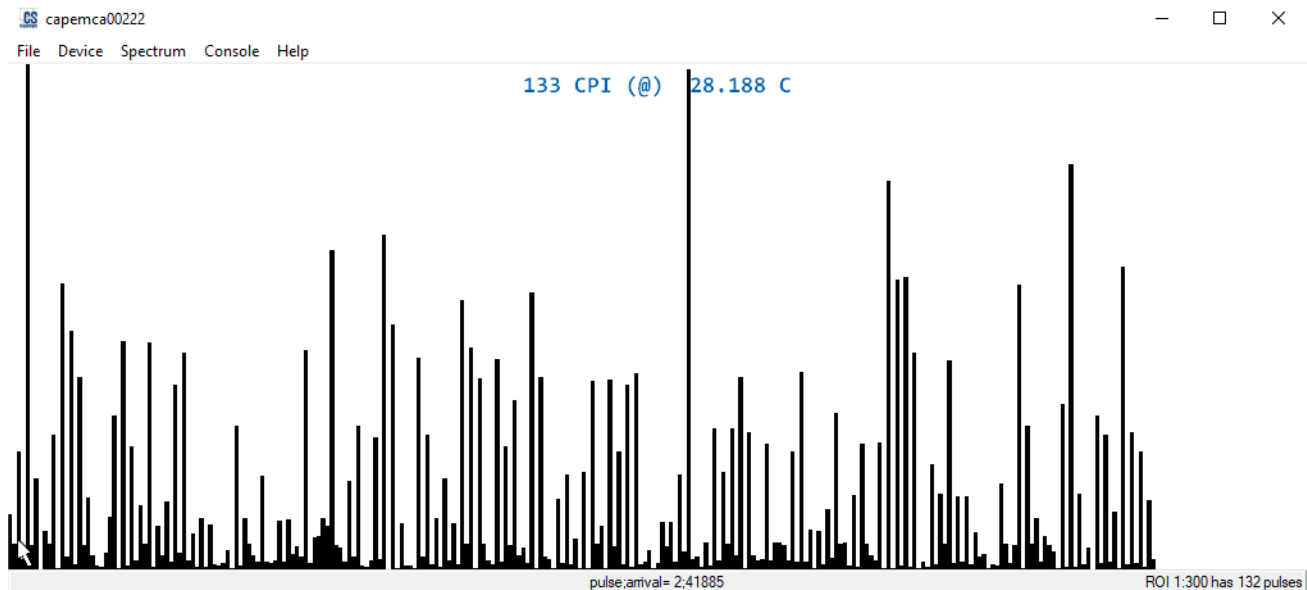


Figure 16: Pulse list showing taller interarrival interval bars followed by smaller pulse integral bars. When the cursor hovers over the interarrival interval of the second pulse, this information is displayed on the statusbar. Note that one pulse detected during the interval was not included in this list of 132 pulses because its channel was not in the specified min/max channel range of [10,4095].

The fourth spectrum data type is for *Pulse list* mode. Instead of the energy spectrum, returned data is the list of pulses detected in the interval. Each pulse is described by its interarrival interval and the integral of the raw pulse signal. During detection, the channel of the pulse is estimated in the usual way using the integral divisor and any subsequent energy correction (see below). Only pulses with channel locations that lie within the minimum and maximum channel range parameters are included in the list.

The interarrival interval is the number of samples between the onset of the pulse and the onset of the previously detected pulse. For the first pulse in the list, the interarrival interval is the number of samples since the onset of sampling for this interval. Figure 16 depicts the display of a pulse list with the cursor pointing to the second interarrival interval in the list.

Since each pulse is described by two 32-bit integers, each pulse takes up two channels of the spectrum, with the interarrival interval coming first and the pulse integral second. The first pulse is written to channels 1 and 2, such that interarrival intervals are in odd channels and pulse integrals are in even channels. The full pulse list of 4095 channels, therefore, can contain no more than 2047 pulses. Count rates of more than 2047 CPI are not supported by pulse list mode. Any pulses which are detected during an interval after the pulse list has been filled will be included in the CPI (channel zero) and CPS (packet0) values, but will NOT be included in any pulse list. However, decreasing the communication interval might allow pulse list mode to accommodate pulse rates of over 10,000 CPS (untested).

Temperature Stabilization (TS)

Temperature stabilization automatically adjusts the high voltage bias on the SiPM array to maintain a consistent relationship between the energy deposited in the detector and the size of the pulse integral, over a wide range of temperatures. Details on how this is accomplished are discussed in the Calibrations section. Temperature stabilization requires that the MCA parameters be tuned to the specific MCA electronics and detector in use. Two different detectors will not have the same calibration for a given MCA.

The Run mode configuration menu allows temperature stabilization to be turned on and off while a spectrum is being accumulated. Turning off temperature stabilization causes the high voltage to remain constant at its current setting, but it can be adjusted manually using the Calibrations menu. Turning on temperature stabilization will transfer control of the high voltage back to the MCA firmware, which will likely result in a dramatic shift in the channel locations of the peaks of the spectrum as the high voltage is changed. For consistent results, temperature stabilization should remain on after calibration.

Energy Correction (EC)

The energy correction algorithm corrects for the fact that the light emitted from the scintillator is not completely proportional to the energy deposited in the detector. The detector will scintillate more for some energies than for others. The type of radiation can also affect the amount of scintillated light, so energy correction should focus on only gamma radiation photopeaks for consistent results. Energy correction allows up to 32 photoelectric absorption peaks to be associated with known gamma energies so that the pulse integral to channel mapping can compensate for the non-proportionality. Energy correction depends on temperature stabilization; both must be properly calibrated for consistent behavior of the MCA.

In the absence of energy correction, the channel that a pulse belongs in is determined by dividing the integral of the raw pulse signal by the integral divisor parameter s .

$$x = \text{channel} = \text{integral} \div s$$

The energy correction is obtained by fitting a quadratic curve to the errors between this channel formula and the known gamma energy in keV (desired channel) for two or more photopeaks. Accuracy of the fit can be improved by including more peaks. The fitted coefficients represent the difference between energy and channel in this form:

$$\text{energy} - \text{channel} = c_1 + c_2 x + c_3 x^2$$

So energy correction uses the formula

$$\text{energy} = c_1 + (c_2 + 1)x + c_3 x^2$$

The parameters are 32-bit integers so the coefficients are translated into integers as follows.

$$p_1 = \text{floor}(10^3 c_1)$$

$$p_2 = \text{floor}(10^6 c_2)$$

$$p_3 = \text{floor}(10^9 c_3)$$

Note that when energy correction is enabled the parameter p_1 provides the channel offset. By modifying this parameter directly, it is possible to shift the entire energy range represented by the spectrum. For instance, if correction makes channel 1 represent 1 keV, adding 1000 to p_1 would shift the entire energy range for the spectrum such that channel 1 represents 2 keV instead. Adding 4,000,000 to p_1 would shift channel 1 to count pulses that deposit 4001 keV, and channel 4000 would then represent 8000 keV. However, care must be taken to ensure that the quadratic fit is still accurate for these higher energies.

Pulse Pileup Rejection (PPR)

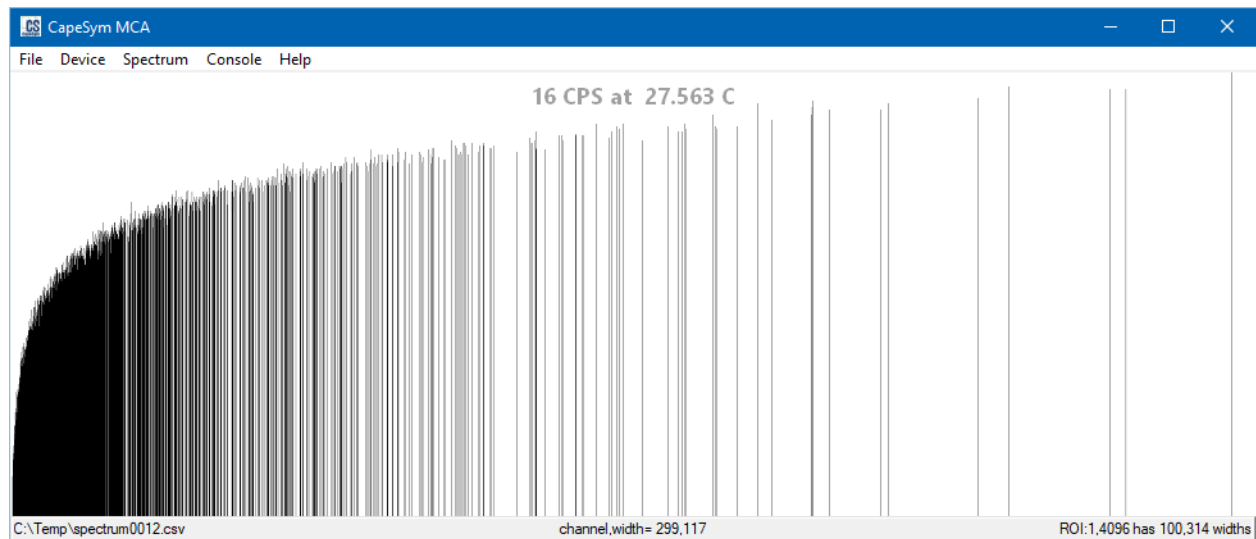


Figure 17: Pulse width spectrum accumulated from dozens of minutes background radiation.

At high count rates, pulses may overlap in time, causing inaccuracies in the mapping of pulse integrals to channel locations. Pulse Pileup Rejection uses the expected width of pulses in each channel to reject overlapping pulses and exclude them from the spectrum. These pulses are counted, however, and included in the CPI information returned with the spectrum. Excluding piled-up pulses from the spectrum helps maintain the energy resolution of peaks during high count rates. The accuracy of PPR depends on many other parameters in the MCA including those of the temperature stabilization and energy correction.

Many types of Macropixel detectors exhibit a characteristic relationship between pulse width and the gamma energy deposited in the detector, as demonstrated by the pulse width spectrum (Figure 17). Pulse width spectra are useful (at low count rates) because they show the expected relationship between pulse width and energy. When count rates get too high, pulses start to overlap making energy measurements of individual pulses uncertain. Most of the piled-up pulses will have pulse widths that are much longer than normal for the measured pulse integral, allowing them to be rejected from inclusion in the spectrum by matching their measured width against the expected width for a normal pulse.

The pulse width, expressed as the number of samples between the first and last threshold crossing, may be reasonably fit by a single curve involving four coefficients.

$$\text{pulse width} = c_1 + c_2 x + c_3 \sqrt{x} + c_4 \ln(x + 3)$$

where x is any channel number in the range [1,4095]. The fitting coefficients are calculated by the host software (during PPR calibration) and then stored as signed integers in the MCA's parameters. Each integer parameter is 10,000 times the corresponding real-valued coefficient. The curve is plotted over the pulse width spectrum using *Plot pulse width (PW) limits* from the *Calibrations > Pulse pileup rejection* menu.

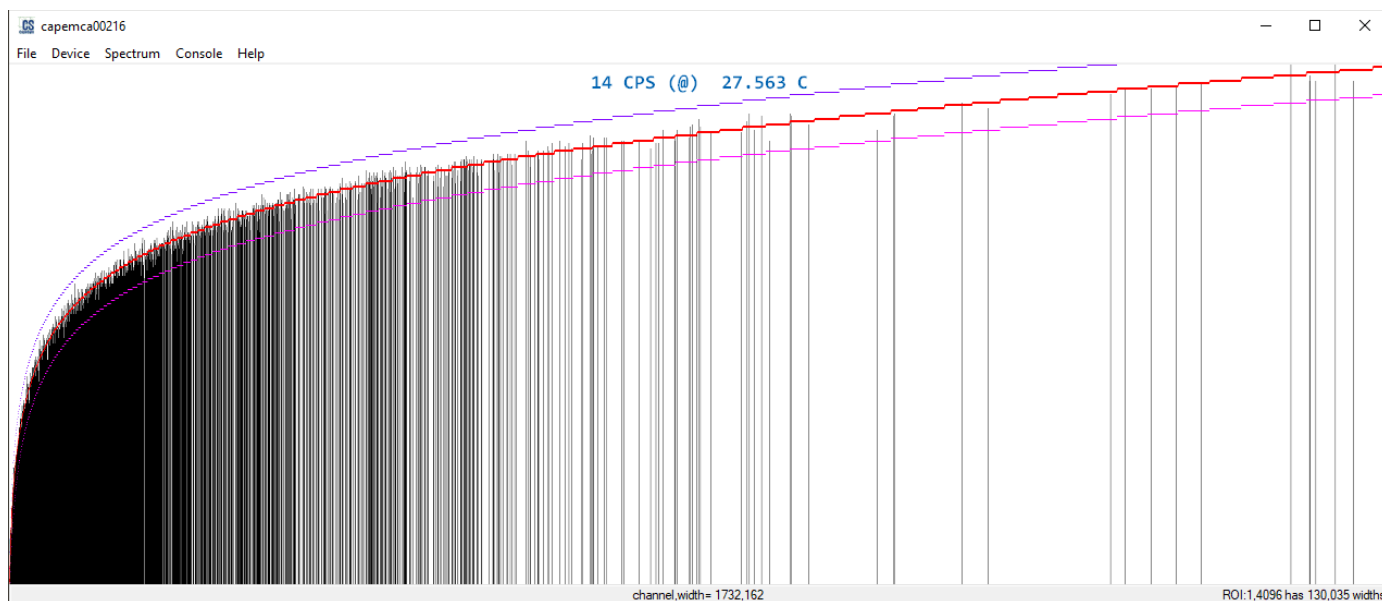
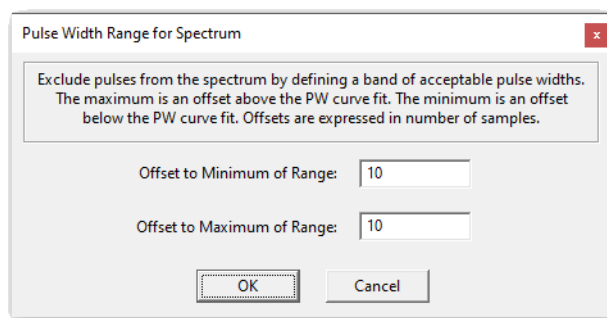


Figure 18: The fitted curve (red) to the pulse width spectrum is plotted along with the upper (purple) and lower (magenta) pulse width limit curves. Pulses with width outside these limits are rejected as pile-ups.



The range of accepted (not rejected) pulse widths is defined by a maximum and minimum at each channel, specified by two offset parameters that give the distance above and below the fitted curve. These parameters may be entered by selecting *Enter PW limit offsets...* from the PPR calibrations menu.

The default offsets of 10 set the maximum of the accepted width range at 10 samples more than the fitted value, and the minimum at 10 samples less than the fitted value (Figure 18). Neither the coefficients nor the offsets will be effective until these parameters are sent to the MCA, since this is where PPR is executed.

The PPR parameters can be changed directly in the *Update MCA Parameters* dialog when in Stop mode (as discussed further below). For example, to shift the PPR range down by 30 samples, $30 \times 10,000 = 300,000$ could be subtracted from the first coefficient parameter, as was done to produce Figure 19.

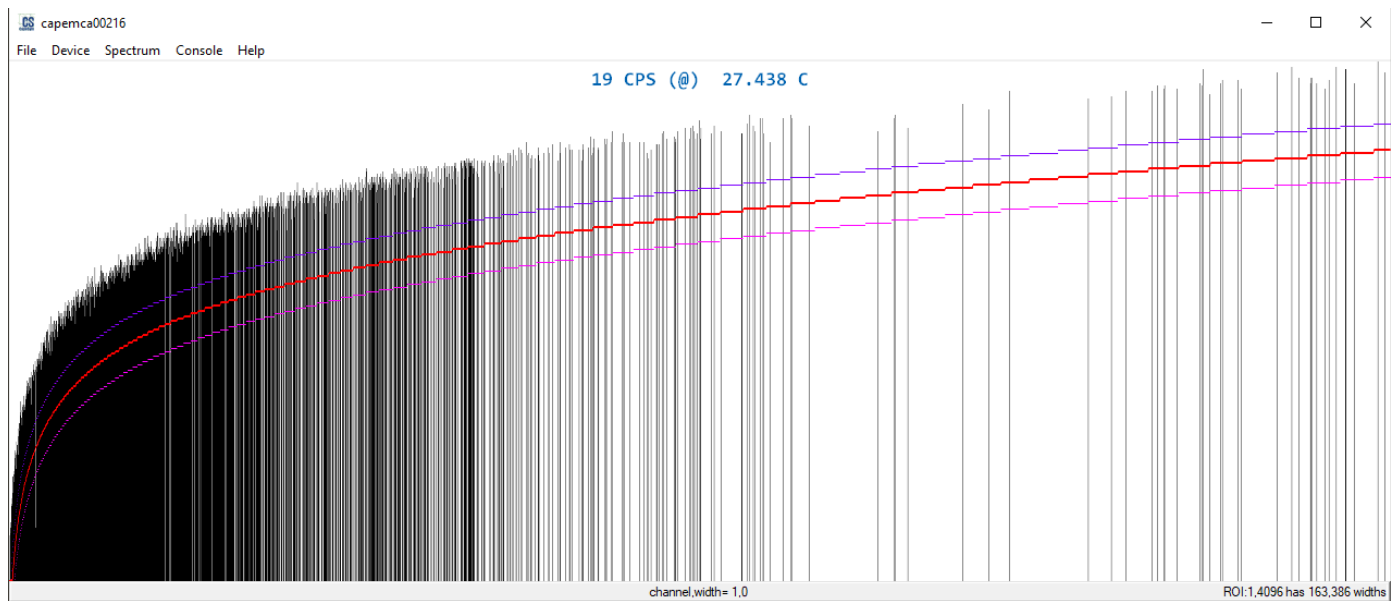


Figure 19: After the curve fit, the PPR region can be shifted by adjusting coefficient *c1* in the parameters of the MCA, but just shifting the offsets would accomplish the same result.

The curve fitted to pulse widths is also used for defining the in-box (*n[]*) capture region, as detailed below, so it may be undesirable to shift this curve away from its fitted values. The offsets can also be shifted below the pulse width curve without changing the fitting coefficients. For example, to get the same range shift the offset to maximum could be set to -20 and the offset to minimum could be set to 40. This results in the same PPR as shifting the fitted curve down by 30 samples.

Channel Zero Feedback

When channel zero feedback is set to *CPI and temperature* (see Figure 14), the pulse count is returned in channel zero of the spectrum along with the temperature of the SiPM array. These two values are received in *spectrum[0]* at every communication interval and displayed at the top of the main window. The pulse count is number of events that occurred since the last communication interval, so it is displayed as counts per interval (CPI). When the communication interval is 1 second, this number is close to the actual counts per second (CPS). More accurate CPS is available in *packet0*. Note that the displayed CPI and CPS values pertain to only a single MCA, even in situations in which multiple MCAs are connected.

Table 3. Feedback components available during different types of spectral processing.

Spectrum Type	Allowed Content in Channel Zero					Packet0
	CPI	NPI	baseline	DAC	Temperature	
<i>Pulse counts</i>	•	•	•	•	•	•
<i>Pulse widths</i>	•	•	•	•	•	•
<i>Pulse sample</i>	•		•	•	•	•
<i>Pulse list</i>	•		•	•	•	•

CPI: Counts per interval, NPI: capture box count per interval, DAC: digital-to-analog level for bias voltage

Packet0 provides the necessary floating point representation for CPS, while CPI is an integer. Hence, the inaccuracy with respect to the actual count rate. The CPI value is stored as a 16-bit integer in the upper 16 bits of spectrum[0]. The 16-bit field limits this feedback to 65,535 CPI, beyond which it will roll over to zero.

The SiPM temperature is stored in the lower 16 bits of spectrum[0]. To handle the fractional part of the temperature, the 16-bit integer stored as 16 times the actual temperature reading. For example, 25.25°C would be stored as the integer 404 because (404/16=25.25). The spectrum[0] format of two 16-bit integers is used for all of the channel zero feedback options, as detailed further in Appendix A.

DAC and Temperature

During calibration of temperature stabilization it is helpful to get feedback about the digital-to-analog converter (DAC) voltage that controls the high voltage bias on the SiPM array. When *DAC and temperature* is selected, the count rate is replaced with the DAC level (a 12-bit number in the range [0,4095]), and the DAC is displayed at the top of the window. This DAC value is the commanded value, i.e. the parameter value for the DAC level when temperature stabilization is turned off. The DAC level will also be reflected in the MCA parameters dialog. More about the relationship between DAC level and high voltage bias can be found in the temperature stabilization section.

CPI and Baseline

The third option for channel zero feedback is count rate in the most significant 16 bits and the SiPM signal baseline value in the lower 16 bits. The baseline value is between 0 and 16384, reflecting all possible readings of the analog-to-digital conversion (ADC). The maximum level is 16384 for 14-bit ADC, and 4096 for 12-bit ADC. The ADC values represent voltages from 0 V to 3.3V. At room temperature, the baseline should be no more than a few mV for a small detector module. However, higher baseline values will occur for larger arrays, or at high radiation levels or high temperatures (see Figure 17). Persistently higher baselines at room temperature may be an indication of damaged or degraded SiPMs.

CPI and n[] Capture

The fourth feedback option is CPI and n[] capture rate, the latter being the number of CPI events that were also within the capture box region. The capture box defines a region of the pulse width versus channel measurement space. The capture box can be used to count the subset of pulses that have unusual pulse widths. For instance, the capture box could be used to count pulses in a certain energy range that exhibit a pulse width deficit (Figure 20). These high-energy, shorter pulses may be typical of neutron capture events in certain types of scintillators containing lithium.

The *Pulse reject/capture* menu is used to setup and display the capture box region. However, the gamma pulse width spectrum must be fit with a curve via the *Fit curve to PW spectrum* menu item before defining the capture box, because the box is defined using offsets from this pulse width curve. Once the pulse width curve is established, setup the capture box using the *Enter box parameters...* menu item.

In this dialog, the offsets define the pulse width region relative to the fitted PW curve. The offset to the maximum PW defines the top of the box, whereas the larger offset to the minimum PW defines the bottom of the box. The channel minimum defines the left side of the box and the channel maximum defines the right side of the box. All pulses that land within the box will be included in the $n[]$ count per interval feedback.

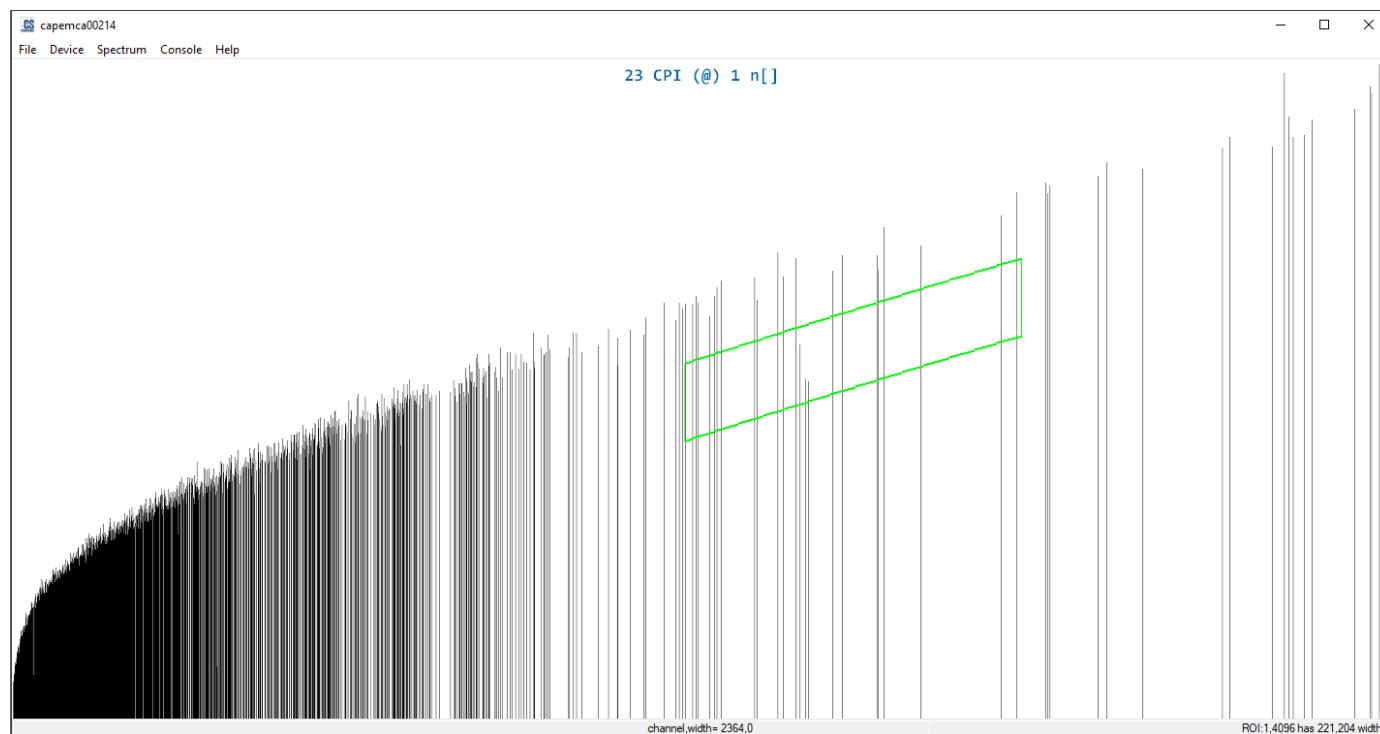
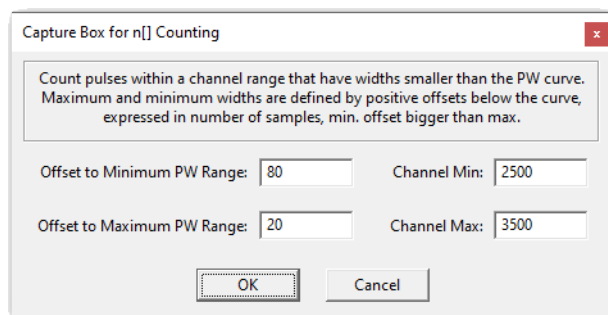


Figure 20: The capture box region (green) plotted on top of the pulse width spectrum. Counts within this region are displayed by selection of CPI and $n[]$ capture for the channel zero feedback. The count per interval within the box ($n[]$) is shown along with the total count per interval (CPI).

As with the PW offsets for PPR, the capture box can be moved above the PW curve by using negative offsets. For example, giving the offset to minimum PW the value -20 and offset to maximum PW the value -80 would create a box of the same size above the fitted PW curve instead of below it (Figure 20).



To reiterate, the CPI includes all pulses detected during the last interval (between green LED flashes) as determined by the MCA's communication interval parameter. “Detected” means the pulse exceeded the threshold for the minimum number of samples, typically 16 samples for the standard 5 to 5.56 Msps sampling rates or 32 samples for the 10 to 11 Msps sampling rates. CPI will include every count in the spectrum and capture box, plus any pulses excluded by PPR, if enabled, as well as those pulses too big or too small to lie within the spectrum's channels.

Spectrum Updates

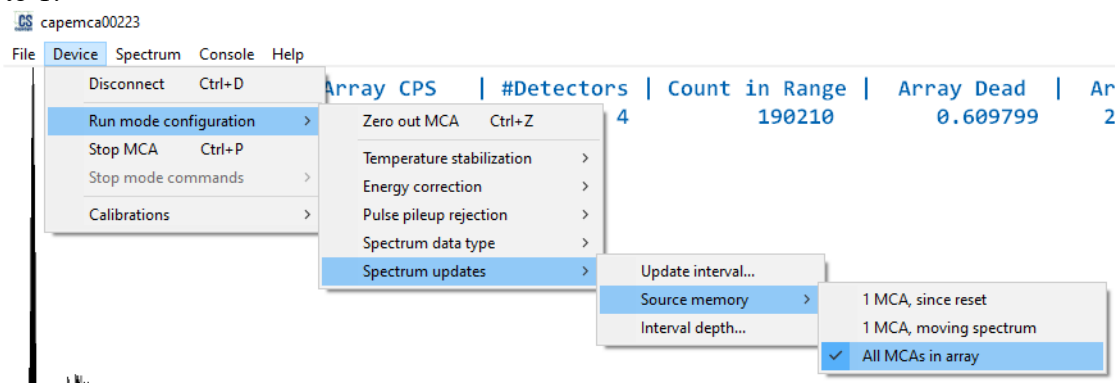
The *Spectrum updates* menu allows the frequency and source of spectral data to be modified while in run mode. These menu items change one of the operating parameters, modifying the parameter in the MCA firmware on the next communication interval. If the MCA is part of an array, the parameter change will be forwarded to the other MCAs as part of the packet0 data sharing that happens at every communication interval. The spectral data returned to the host software may come from just the one MCA connected to the host via USB, or from the whole array, depending on the selection in the *Source memory* submenu.

Update Interval

The *Update interval...* menu item displays the current communication interval parameter and allows the user to change it. The parameter specifies the interval in 100ms increments. A value of 1 is 100ms and a value of 10 is 1000ms, as discussed below. The frequency of spectrum data requests made by the host software will increase to reflect the new frequency.

Source Memory

Spectral data received by the host usually comes from an array of 4096 memory locations in the MCA. The MCA accumulates pulse counts continuously in this array until the spectrum is zeroed or the device is reset. This default source of spectrum memory will be indicated by a check mark next to the menu item: *1 MCA, since reset*. Instead of accumulating a single spectrum forever, the MCA can be configured to keep only spectral data acquired during the last few intervals, with the channel counts from the most recent interval being shifted into a larger multiple array memory while the counts from the oldest interval are shifted out and forgotten. Hence, the term “moving spectrum”, which corresponds setting the *Moving spectrum* parameter to 1.



When there are multiple MCAs configured in an array, with MCA-to-MCA communication links between them, the source spectral data can be *All MCAs in array*. The spectral data from all of the MCAs is passed to the one MCA connected to the host just prior to processing the host request. The data are combined to form a single spectrum that is transmitted to the host. The spectrum source memory in this case is the sum of the single array memory of every MCA in the array. It is not possible to use moving spectra while

combining array data in this way, because these spectra are accumulated forever until zero or reset.

For a single MCA configuration, the *1 MCA* items are the only possibilities. Choosing the third option will result in the same result as choosing as the first option. Choosing the moving spectrum option for an array will cause every MCA in the array to also switch to moving spectrum mode such that the packet0 information will track only the most recent data acquisition intervals.

Interval Depth

The depth of moving spectra memory specifies how many intervals will be retained, to be later summed into one spectrum and returned to the host. For example, a value of 8 will retain 8 spectra, each one recorded during one interval. When the communication interval is set at 1 second, this would mean that 8 seconds of spectral data is retained in each MCA. The parameter is only relevant when the moving spectrum mode is enabled. Furthermore, the maximum allowable depth depends on the size of the RAM inside the MCA, and so may change with hardware version.

Request # of Channels

The configuration changes discussed so far have involved modifying parameters that are stored inside the MCA. However, the host software can also configure the type of data requests it makes to the MCA which changes the amount of data that the host will receive in reply. These request types are selected in the *Request # of channels* section of the Spectrum menu (Figure 21). Request types differ in the number of channels of the spectrum returned, chosen from the set of lengths {0,256,512,1024,2048,4096}, and in the presence or absence of the additional 64-byte statistics data (packet0). Details of the packet0 data structure are presented in Appendix A.

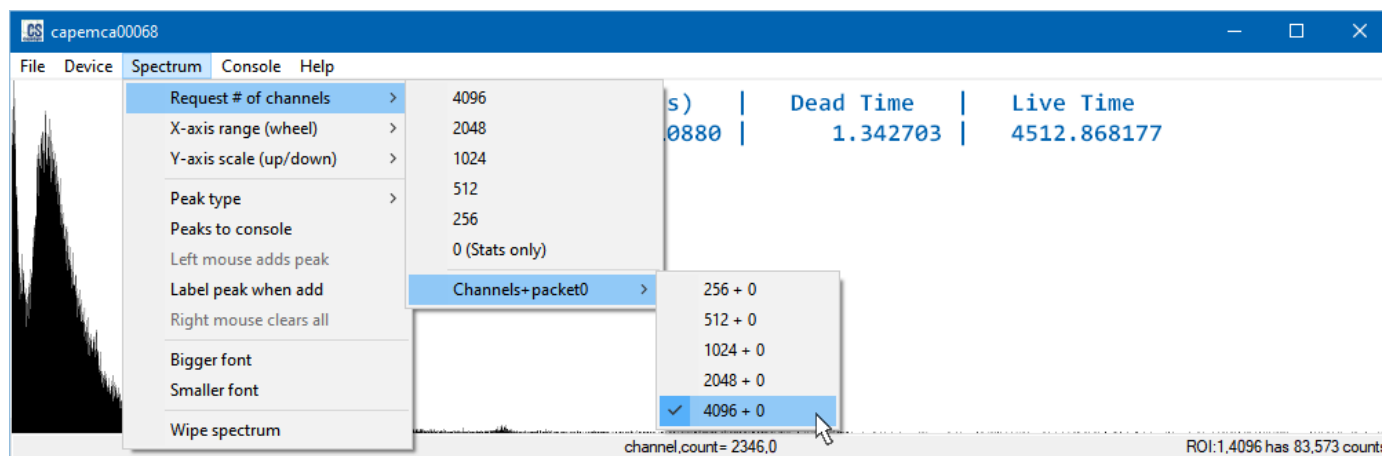


Figure 21: The menu items for configuring the data request during live updates from the MCA. Different numbers of channels in the spectrum may be requested along with the packet0 data. When packet0 is included, the displayed text at the top of the window changes to show some of the returned statistics.

Packet0 is designed to provide rapid data feedback in autonomous vehicle and other applications which do not require the details of the gamma energy distribution. In addition, packet0 provides information that can be used to determine the total acquisition time as well as the dead and live portions of it. Some of the packet0 information is displayed in the main window when *Request # of channels* is set to 0, but all of the

packet0 fields are stored when N42 file recording is used.

Some packet0 fields have been added to support multiple detector modules interconnected by cables or a *backplane*. When the host software receives a packet0 which reports that more than one detector is present, the additional array information from packet0 is automatically displayed in the main window. This information includes the sum of count rates from all detectors, the total count rate in a specified channel range, and the direction of the source of radiation in the specified channel range. The channel range used for calculating these fields is delimited by the minimum and maximum channels also used for pulse sample and pulse list modes. These channel range parameters can be updated in Stop mode.

When the detector or radiation source is moving, the *Moving spectrum* option may be combined with the source direction feedback to facilitate continuous source tracking. All of the MCAs in the array are configured the same way, with moving spectrum enabled and the same depth and communication interval. Note that source direction tracking is not possible with fewer than four detectors.

Stop Mode Commands

The *Stop MCA* menu item causes the MCA hardware to stop processing pulses from the detector module and to stop accumulating a spectrum. Entering Stop mode allows the MCA firmware to be updated beyond the on/off adjustments available in the *Run mode configuration* menu, through the *Stop mode commands* menu (Figure 22). To exit Stop mode, use the *Start up* command on the Device menu, or generate a *Soft reset* from the Stop mode commands menu. When a reset is generated the MCA parameters will be reinitialized from non-volatile memory (see Figure 13), while the Start up command allows Run mode to resume using the MCA parameters that are currently in RAM.

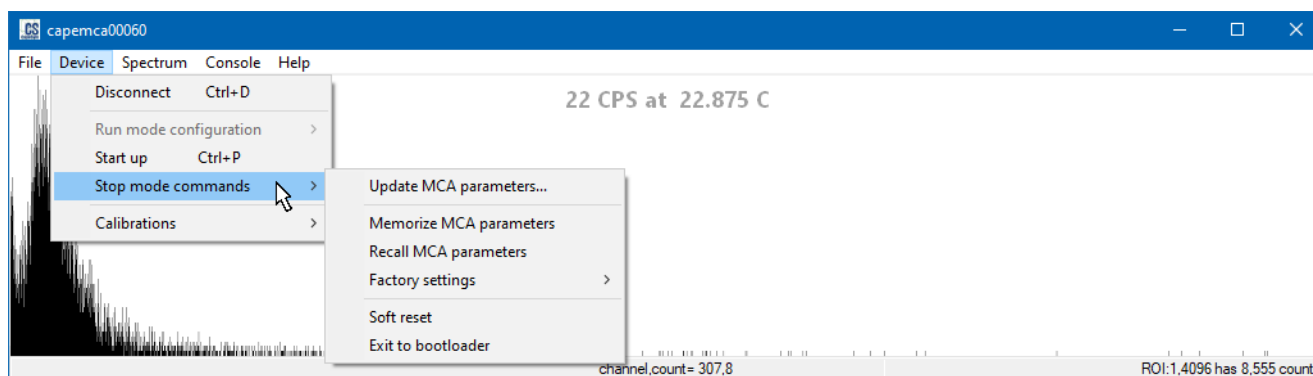


Figure 22: All of the modifiable MCA parameters can be viewed and edited using the *Update MCA parameters...* menu item, but many parameters are easier to update from the *Calibrations* menu.

Update MCA Parameters

The *Update MCA parameters...* menu item opens the MCA Parameters dialog (Figure 21). The spreadsheet-like interface allows the values in the first column to be edited, while the second column provides a description of the parameter. The purpose and modification of many of these parameters are described in the Calibrations section. Most parameters are easier to modify using the Calibrations menu rather than by entering them directly in the MCA Parameters dialog.

The MCA parameters are arranged in an array of 128 integer values (Appendix C), so that the entire set of parameters may be transferred easily from the MCA firmware to the host software, as well as from the host

software to the MCA firmware. In the MCA firmware, the array entries are indexed from 0 to 127, the first parameter in the array having the index of 0 and the last parameter having index 127.

The entire parameter array is transferred to the MCA firmware when the *Send to MCA* button is pressed in the MCA Parameters dialog (Figure 23). These parameters are written to volatile memory (RAM) so that they can be used by the MCA but not be permanent. The parameter changes will be forgotten by a soft reset or loss of power to the MCA. Note that the parameter changes might not show up in the Run mode configuration menus without reading them from the MCA. To reread the MCA parameters select the *Disconnect* menu item (without disconnecting the USB cable), followed by *Connect* to reload the parameters from the firmware. The menus will then be properly updated.

Firmware Version

The first three items in the parameters array denote the firmware version. These parameters should not be changed by the user, and so are not made available for modification in the MCA Parameters dialog. The dialog displays parameter with index 3 in the top cell. The parameter with index 3 is the *Divisor for integral to channel*. For a complete list of parameters in the array, along with indexing, see Appendix C.

Modifying the MCA firmware version numbers would make getting support and repairs difficult, since it would not be clear what build is actually installed on the hardware. Furthermore, changing the version numbers will cause a factory restore of all parameters on the next reset or power-on of the MCA. The restored parameters will also over-write the FRAM memory, such that all of the user's parameter changes will be permanently lost.

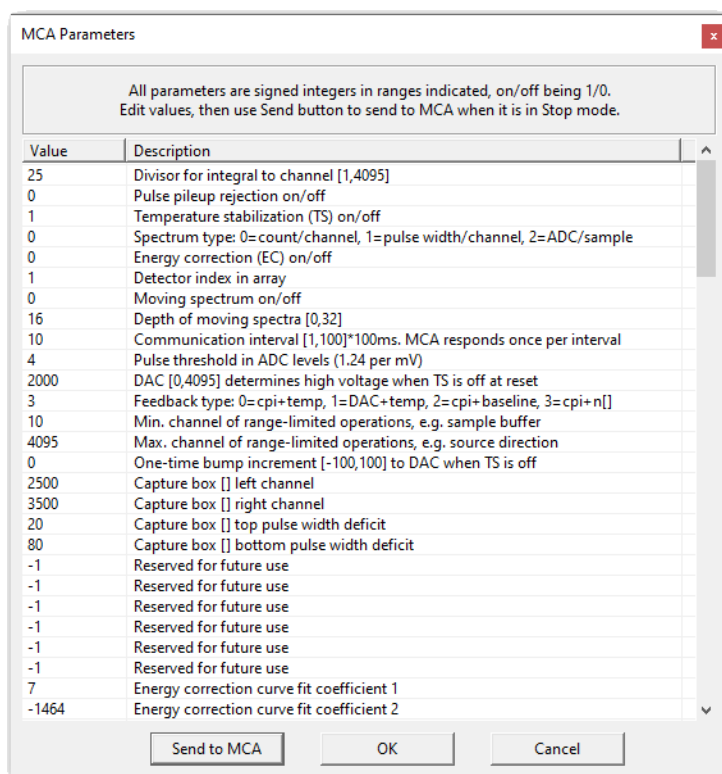


Figure 23: MCA Parameters dialog provides a spreadsheet-like interface for viewing and editing the modifiable MCA parameters. Clicking the *Send to MCA* button causes the entire parameter array to be written to the volatile memory (RAM) of the MCA, replacing the current parameter array there. Committing parameters to non-volatile memory requires another step: *Memorize MCA parameters*.

Divisor for Integral to Channel

The divisor for converting the integral of the pulse signal to its channel number provides a coarse calibration of how gamma energy will be binned for counting. The divisor cannot provide accurate channel placement over a wide range of energies and temperatures, however, due to the complexities of the scintillation material and SiPMs. The pulse threshold, bias voltage, and energy correction settings also play an important role in final channel placement. For detailed discussion, see the Calibrations section.

Because the divisor scales the pulse integral, it effectively sets the channel width in terms of how much energy is represented by one channel. For example, suppose that the divisor is fixed at 200 and, after temperature stabilization, this places the photopeak for Cs-137 (662 keV) at channel 662. Then the channel width is roughly 1 keV, assuming scintillation light yield was proportional to deposited energy. Since temperature stabilization is designed to keep the pulse integral constant, stabilization is not affected by changing the divisor. Setting the divisor to 100 would reduce the channel width to 0.5 keV and place the 662 keV photopeak at channel 1323, while temperature stabilization continues to hold it there. The other calibrations for energy correction and pulse pileup rejection would need to be redone as after changing the divisor, because these calibrations depend on the channel locations of the peaks.

<< Changing the divisor invalidates energy correction and pulse pileup rejection >>

Detector Index in Array

This parameter defines the position of the detector when multiple devices are combined into a sensor array to provide additional capabilities, such as estimating the direction of the source of gamma radiation. This parameter is not used in standalone operation, nor should it be changed by the user in most circumstances. Each detector in the array must be given a unique index to support the proper weighting to be given in the direction calculation. The index is between 1 and the maximum array size, typically 32 detectors. The maximum number is programmed into the firmware and so is not modifiable by the user.

Moving Spectrum

Normally, the MCA is continuously accruing counts in the spectrum, retaining all old information until a zero out command is received. When the detector is moving, it may be helpful to forget old pulses and retain only the most recent ones. This can be done automatically by turning on *Moving spectrum* operation (Figure 24). When moving spectra are enabled the MCA is said to be in *tracking* mode.

The pulse sample, pulse list, and pulse width types of spectra should not be used when the MCA is configured to produce a moving spectrum, because the moving spectrum is computed by summing the channel values from several spectra obtained in sequential intervals. Summation is only meaningful for pulse counts, not for pulse width or pulse sample data.

<< Only the pulse count spectrum should be used as a moving spectrum >>

Each output spectrum will be the sum of the most recent spectra acquired during sequential communication intervals. The number of spectra retained is given by the depth of moving spectra parameter. The maximum number retained is determined by the amount of RAM in the MCA, typically limited to 16 or 32 spectra. For example, with the communication interval set to 1 s, and the depth of spectra set to 32, each spectrum received from the MCA will be the sum of the last 32 seconds of radiation. During the next interval, a fresh spectrum will be acquired for 1 s, and then added to retained spectra while the oldest spectrum is forgotten.

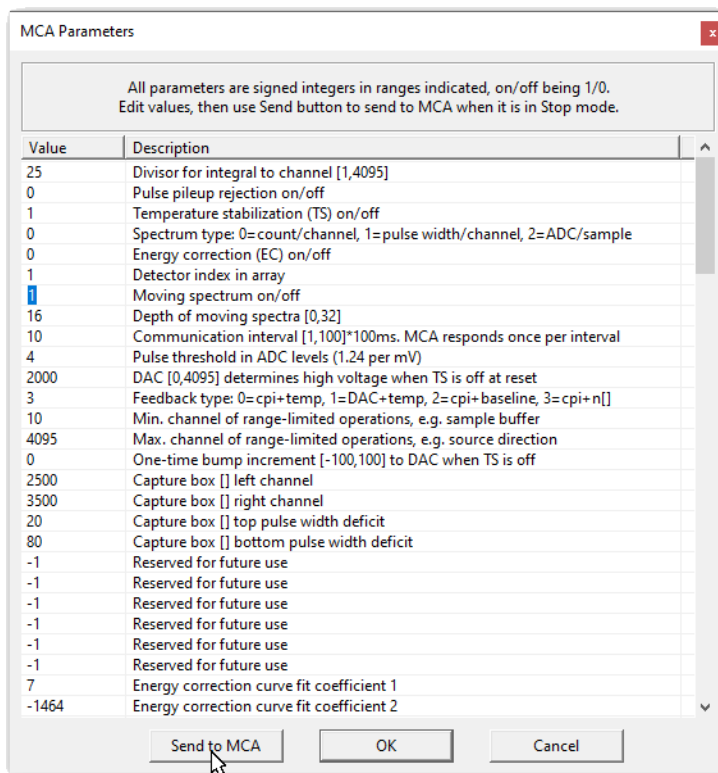


Figure 24: Turn on Moving spectra operation from the MCA Parameters dialog by setting the highlighted parameter to 1 and clicking the [Send to MCA] button.

When the depth of moving spectra is set to 0 or 1, only the data acquired since the previous communication interval is returned by the MCA. When depth is 0, nothing is stored in memory and all of the moving spectra memory is zeroed out. When depth is 1, memory is not cleared but is instead overwritten by the recently acquired data. When depth is 2, the sum of two spectra from the last two intervals is returned.

The moving spectrum calculation requires extra time for the summation of all of the channels in all of the spectra. With a depth of 32, this calculation will add an additional 8.4 ms to the dead time for communication. Dead time can be minimized by using a shorter depth with a longer communication interval. For example, setting depth to 8 and communication interval to 40 (4 s) will give 32 seconds of moving spectra but take only 2 ms to calculate. Thus, only 2 ms is added to the dead time for communication, rather than 8.4 ms when the 1 s intervals were used.

Communication Interval

The communication interval parameter determines how often the MCA communicates with the host software, and how long an individual acquisition continues without a break for communication. The parameter is an integer that multiplies 100 ms, so a value of 1 indicates a 100 ms interval, 2 is 200 ms, 10 is 1 s, 20 is 2 s, 100 is 10 s, and so on. Care should be used when selecting longer intervals, because the MCA will be unresponsive to user commands and parameter changes for a couple of intervals after connecting. Recall that only one command will be processed per interval. So a value of 100 would make the MCA device unresponsive for at least 20 s while the initial host commands are processed after connecting. The

LEDs will be mostly off during this time making it difficult to assess the state of the device.

<< Increasing the interval may make it difficult to communicate with the MCA >>

Pulse Threshold

The pulse threshold determines the signal level for which a pulse is detectable. The threshold is applied to a moving average filter, rather than the raw signal, to reduce detection noise. Pulse integration begins when the filter rises above the threshold, and continues until it drops below the threshold, however briefly. In order to be counted as a valid pulse, the filter must remain above threshold for at least 16 samples. The pulse threshold is given in ADC levels. For 12-bit resolution, one ADC level is $4096/3300\text{mV} = 1.24$ bits per mV. In other words, each bit of the digitized signal represents about 800 uV, and a threshold of 4 would be equivalent to ~ 3.2 mV. For 14-bit resolution, one ADC level is $16384/3300\text{mV} = 4.96$ bits per mV.

<< Changing the pulse threshold will invalidate existing calibrations >>

DAC When TS Off

The high voltage bias for the SiPMs is controlled by a 12-bit digital-to-analog converter (DAC) output voltage. When the temperature stabilization is on, the DAC is automatically calculated from the current temperature of the SiPMs. The DAC parameter (index 13) determines the value to use at reset or power-up, when the temperature stabilization (TS) is not turned on. The user can control the bias voltage by turning TS off, setting this parameter, sending it to the MCA, memorizing the parameters, and then resetting the MCA. Using Start up will not work in this scenario since the parameter is only applied to the DAC after a reset.

The DAC parameter must be a value between 0 and 4095, inclusive, resulting in high voltage of

$$\text{High Voltage} = (0.001773) \text{ DAC} + V_{\text{DAC}=0}$$

where the voltage for DAC=0 will vary from MCA to MCA, but should be between 25 and 25.5 V.

Min. and Max. Pulse Channel

Two parameters (index 15 and 16) specify a channel range for pulses to include when the *Spectrum data type* is set to *Pulse sample* or *Pulse list*. In addition, these minimum and maximum exclusive limits are applied to the detector array count range used for source direction calculations.

Table 4 summarizes the channel range limits applied to measurement data from the MCA. To be included in the *Pulse counts* or *Pulse widths* spectrum, the channel must be within the limits of the data structure, which contains 4096 counting bins. In contrast, all detected pulse are included in the CPI and CPS, regardless of how big they are.

Channel Range for Isotopes

This submenu provides quick adjustment of the minimum and maximum pulse channel range parameters, to center the range on the most characteristic photopeak(s) of a particular isotope. Selecting an isotope from the menu, first modifies the two channel range parameters, and then sends the entire set of parameters to the MCA. This feature, if present, is mainly used to change the directional target of an array of detectors.

Table 4: Channel ranges (exclusive) used for various measurements

Measurement Data	(0, 2 ³¹)	(0, 4096)	(minPulseChannel, maxPulseChannel)
Pulse counts		•	
Pulse widths		•	
Pulse sample			•
Pulse list			•
CPI (count per interval)	•		
NPI (in-box count per interval)		•	
CPS (packet0)	•		
CountInRangeArray (packet0)			•
(x,y,z) Direction (packet0)			•

Memorize MCA Parameters

Some of the parameters will only become active after a soft reset or power on/off of the MCA. In this case, the entire parameter array must be committed to non-volatile memory BEFORE any reset, by using *Memorize MCA parameters* from the Stop mode commands menu. Once parameters are memorized, they will be automatically loaded into RAM for use whenever there is any reset.

Recall MCA Parameters

The user may also request that the memorized parameters in the MCA be restored into RAM by issuing the *Recall MCA parameters* command in Stop mode (see Figure 13). This command may be used to erase the parameter changes made by the *Send to MCA* button, although power off would do that just as well.

Factory Settings Restore

The MCA firmware should have a backup of the original parameters stored in program flash. These can be copied into the MCA volatile RAM by using *Factory settings > Restore*. To make the restoration permanent follow up with *Memorize MCA parameters* to copy them into non-volatile memory. Note that a factory restore will wipe out any calibrations that were done after the original factory programming, so the restored settings should be thoroughly tested before committing them to memory.

The MCA parameters after factory restore will be unfamiliar to the host software, because they have not been transferred to the host software yet. This transfer can be accomplished by a Disconnect/Connect sequence. If you open the MCA Parameters dialog immediately after factory restore the old parameters will be displayed. To see the restored parameters, return to Run mode, then use the Disconnect item from the Device menu to break the USB connection, followed by Connect to reestablish the connection. The restored parameters which are active in RAM will then be transferred to the host software and will be displayed in the MCA Parameters dialog.

Soft Reset of MCA

A soft reset will reboot the MCA microcontroller while power is maintained. The microcontroller goes through its full initialization, which includes initializing all of the peripherals and reloading the memorized

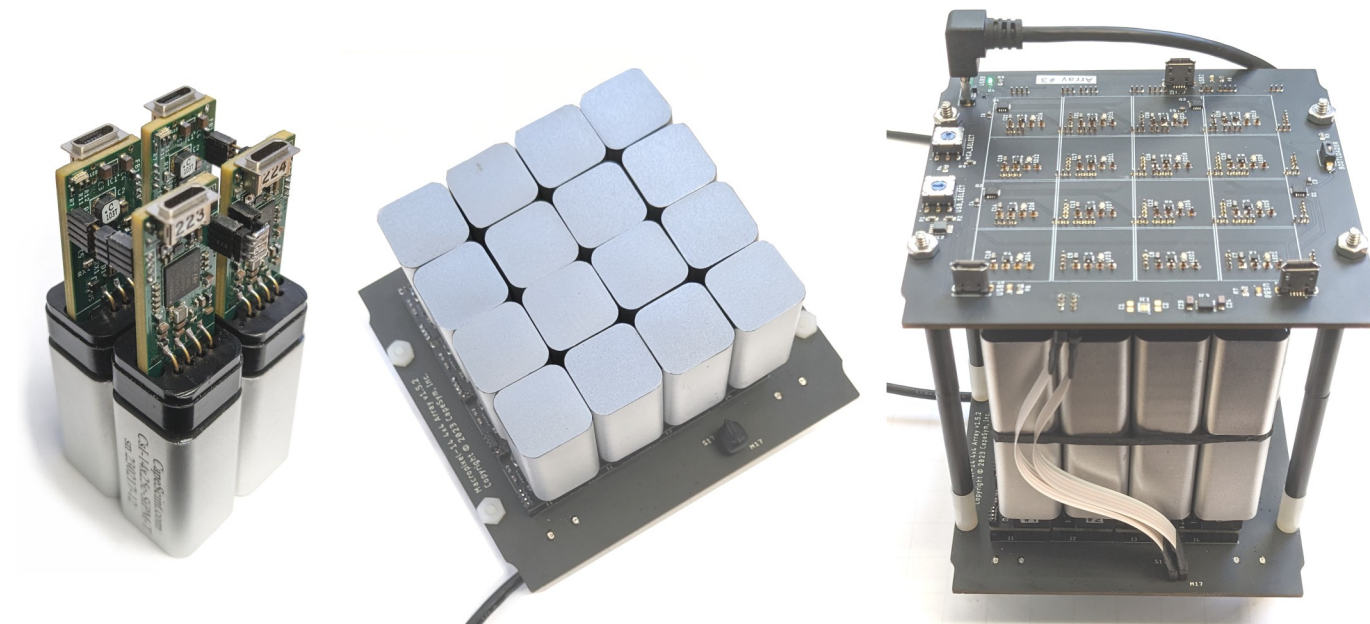
parameters. Any parameter changes in RAM will be over-written. The USB connection will be broken and the WINUSB device driver restarted in the PC. The USB connection can then be reestablished from the Device menu by selecting the *Connect* menu item.

Exit to Bootloader

While in Stop mode, the MCA hardware can be put into full programming mode through the menu item *Exit to bootloader*. This command disconnects the STM32 WINUSB device driver and causes the DFU device driver to become operative in its place. The STM32 bootloader will be running on the MCA, which allows the entire firmware of the MCA to be rewritten, as discussed further in Appendix A. To restore MCA firmware operation, the device must be completely powered off by unplugging the USB cable.

MCA Arrays

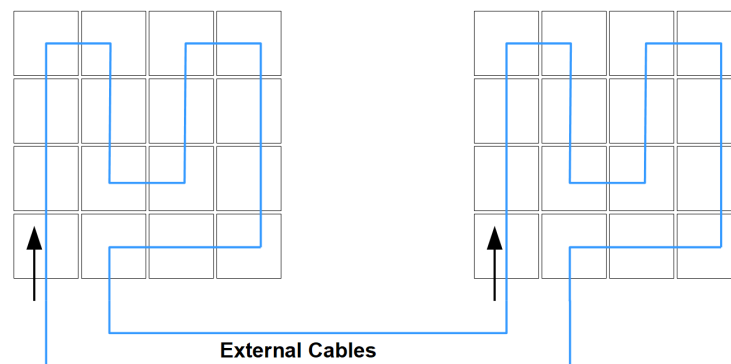
Some types of MCAs are designed to work together to improve performance. The larger volume of the combined detectors provides increased gamma sensitivity. If all MCAs are accurately calibrated, such that the same gamma energy is mapped to the same channel, then the entire array can maintain a gamma energy resolution comparable to that of each individual detector but at the much higher total pulse rate. An array is also able to estimate the direction of a localized source of radioactivity through the self-shielding effect.



Every MCA in an array must receive power, either from a power-only USB cable, or from a *backplane* designed to distribute power to the MCAs. One, and only one, MCA must also receive a full USB data connection to the host computer. The USB data lines on the power-only cables must be unconnected; they cannot be shorted together. Each MCA uses the absence or presence of USB data line connections to determine its role in MCA-to-MCA communications. The MCA that is connected to the host computer is arbitrary, since all are fully capable. Every MCA in the array is running the same firmware. To change which MCA is connected to the host via USB, turn the array off, switch the USB connection, then turn the entire array back on to restart the array initialization process.

The current firmware supports arrays of even numbers of detectors, up to 32 in two 4x4 layers. Any arrangement of detectors is possible, but to support directional sensing the MCAs must be placed a predefined, tightly-packed configuration, with the detector indexes of the MCAs in a particular order. Rearrangement of the MCAs or loose spacing of the detectors will invalidate the direction information computed by the array. To enable direction sensing in three dimensions, the array must consist of two layers with a minimum of four detectors in each layer.

MCA-to-MCA communications are enabled by two Serial Peripheral Interface (SPI) connections from each MCA, one to each of its two neighbors in a circular daisy chain. Each connection is a 4-wire link supporting the full duplex {SS,MOSI,MISO,SCLK} SPI protocol. Each MCA is master of one link and slave of the other. The daisy chain is formed by connecting the master side of one MCA to the slave side of its neighbor, all the way around the array until the head is connected to the tail. The entire array must be daisy-chained, by a backplane or cables or some combination of cables and backplanes.



Data moves around the SPI daisy chain in both directions simultaneously. This limits the number of required data transfers to half the number of MCAs in the array. For example, in an array of 32 detectors every MCA gets a copy of every one else's data packet after just 16 data packet transfers across every SPI link. Since there is always an even number of MCAs, the final data packet that an MCA receives from the clockwise direction is identical to the final packet it receives from the counterclockwise direction. This fact allows every MCA to perform a data integrity test by comparing the final two packets.

Meaning of Array Lights

To power up an array, the same power source must be applied to all MCAs at the same time. At power up each MCA initializes its clocks and then pauses for 0.5 seconds to ensure that any USB data lines get fully connected. Each MCA shows a yellow or blue indicator light during this phase, which is followed by the attempt to detect USB data lines. The MCA which detects the USB data connection displays green, while the other MCAs display red to signal that no data lines are present. The array then attempts to synchronize via the SPI communication links.

The MCA with the host connection will be the initiator of the synchronization signal. Each MCA in the daisy chain passes the signal along, until it gets back to hosted MCA. If this initial synchronization attempt times out after two seconds, e.g. due to a broken SPI daisy chain or an unpowered MCA, then all MCAs in the array switch to standalone mode. Any further attempt at MCA-to-MCA communication is abandoned until next power on sequence. All MCAs in the array will flash a yellow or blue indicator at every communication interval when running in standalone mode. The host computer will only receive data from

the single MCA to which it is connected. The blinking lights will slowly desynchronize across the array because individual MCA clocks differ slightly in frequency.

When the initial synchronization is successful, a second timed synchronization will be performed to determine how many detectors are present in the array. If this is also successful and the number of detectors is determined to be greater than one, every MCA in the array will include the MCA-to-MCA communications routines in its run loop. During the communication interval the lights on all MCAs turn green simultaneously, indicating that synchronization is being maintained via the SPI daisy chain.

Two error conditions may be indicated by a different light pattern. If an MCA flashes red instead of green, then the data integrity test failed in that MCA. Data integrity is a local test performed by every MCA, and the red flash is the only indication that there may be a problem with SPI reliability in the current environment. The second error condition involves freeze up with all MCAs displaying green continuously. This condition may be caused by more than one MCA having a USB host connection or, more rarely, by dropped SPI data frames, leading the MCA-to-MCA communication link to halt prematurely between two MCAs. Recovering from the freeze up condition requires a power off reset to reestablish synchronization.

During normal operation, synchronous green blinks should stop when the host computer puts its MCA into stop mode. In stop mode the hosted MCA will display red continuously and the other MCAs display green continuously while they wait for the stopped MCA to resynchronize. From this state it is usually possible to return to run mode by sending a two-byte command from host computer, e.g. by the *Start up* menu item in the CapeMCA host software.

MCA-to-MCA Data Packets

Two types of data packets are transferred over the MCA-to-MCA communication link, one containing the packet0 information and the other containing the full spectrum. The spectrum transfer only happens when the moving spectrum parameter is set to two. It is not possible to share spectra across the array while also using the moving spectrum feature, because both of these features use the same memory bank inside the MCA. Another reason to disable the spectrum transfer (by setting the moving spectrum parameter to 0 or 1) is to eliminate the communication time overhead. Each spectrum transfer is 16384 bytes and at an SPI data rate of 8 Mbit/s it takes about 16ms to get from one MCA to the next one. For a 32-detector array the required 16 sequential transfers would therefore require more than $\frac{1}{4}$ of a second. Hence, more than $1\frac{1}{4}$ seconds of real-time would pass for every 1 second of live time, assuming a 1-second host update rate.

The packet0 related data packet is much smaller, only 48 bytes, so it is always transferred at each communication interval. This data packet contains a summary of the radioactivity observed by each MCA, including the counts per second in the last interval, total pulse count, total dead and live times, and directional information. In addition, this data packet forwards certain commands and parameters from the host computer to the rest of the array.

Table 5 lists the valid commands and parameters that can be applied to an entire array. If the host computer issues one of these two-byte commands to the hosted MCA, it will be forwarded to other MCAs for immediate action. The run mode commands are usually issued by the host computer by using the [i,v] command, where i is the index of the parameter to modify (between 3 and 32) and v is new value. Parameter indexes not listed in the table should not be changed by this command. For example, issuing the [8,1] command would overwrite detector indexes throughout the array, invalidating all directional sensing capabilities. Certain parameters must be consistent across the array for proper operation, and these

parameters {9,10,11,15,16} are automatically sent in every data packet transfer, whether there is a change by the host or not. Changing one of these parameters in the hosted MCA by any method (such as stop mode) will result in the change propagating through the array. These parameters can be memorized locally and repropagated on power up, but only if the same MCA always remains connected to the host computer.

Table 5: Valid array-wide commands and parameter changes

Cmd	Index: Parameter(s)	Local Action Taken by Each MCA
1,1	none	Zero spectrum, packet0 and moving spectrum (if enabled)
i,v	i: See below	RAM parameter[i] set to value v (v<256)
	4: Pileup rejection	Turns on/off the use of pulse width limits
	5: Temp. stabilization	Turns on/off temperature stabilization
	6: Spectrum type	0=pulse counts, 1=pulse widths, 2=pulse sample, 3=pulse list
	7: Energy correction	Turns on/off energy correction
	9: Moving spectrum	0=no sharing, 1=locally moving, 2=spectrum sharing
	10: Depth of moving	Only applicable when moving spectrum is 1
	11: Comm. interval	Communication interval as multiple of 100ms
none	9: Moving spectrum	0=no sharing, 1=locally moving, 2=spectrum sharing
none	10: Depth of moving	Only applicable when moving spectrum is 1
none	11: Comm. interval	Communication interval as multiple of 100ms
none	15: Min. channel range	Lower bound of pulse samples, lists, and directional counts
none	16: Max. channel range	Upper bound of pulse samples, lists, and directional counts

Directional sensing is based on the total count in the range between the minimum channel and maximum channel parameters. In the self-shielding effect, each detector receives a different amount of gamma radiation due to its position in the array relative to the source of radioactivity. Each MCA independently records a spectrum from its detector and extracts the total count in the range. These values are then sent through the array data packets so that every MCA gets a copy of the count in range from every other detector. The pattern of these counts across the array are then used to determine the location of the source, using a projection matrix optimized for the type of detectors and their known positions in the array. Thus, the directional calculation can only be performed when the spectrum type is pulse counts per channel.

Table 6: How shared spectra are combined based on spectrum type

Type	Spectrum Data	Method of Combining Two Spectra
0	Pulse counts/channel	Channel-by-channel summation
1	Pulse widths/channel	Keep one non-zero pulse width per channel
2	ADC signal/sample	Keep largest value per sample
3	Pulse list	Append lists until run out of space (at 2047 pulses)

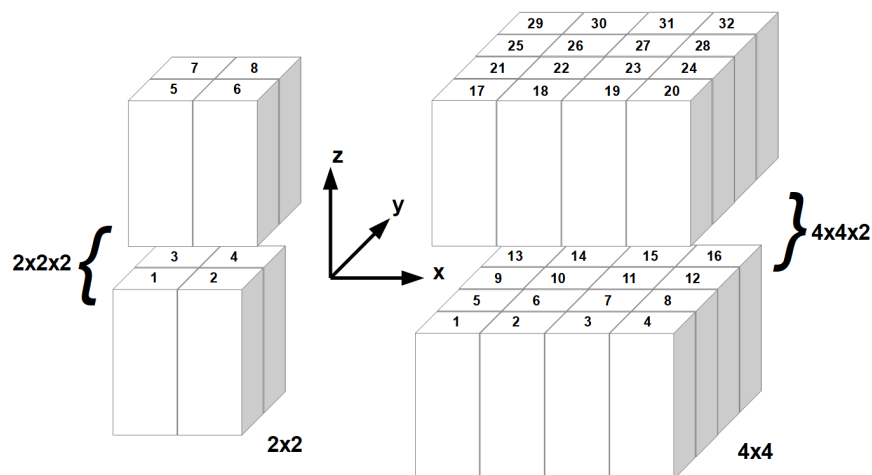
When spectrum sharing is turned on, every MCA receives a copy of the spectrum from every other MCA in the array. These spectra are combined within the MCA by the procedure listed in Table 6. For instance, when the spectrum type is pulse counts, the spectra are combined by adding the counts channel-by-channel so that the combined spectrum has counts that are the sum of the individual spectra counts. The combined spectrum is the one sent to the host in response to a data request. The host software should be configured to request both the 4096-channel spectrum and packet0, which together contain all of the array information. Note that spectrum[0] always contains just the local information from the hosted MCA. Counts and temperatures in spectrum[0] are the same as what would be received in standalone mode.

Array Calibration

Subsequent sections of this manual detail the calibrations that can be performed within an individual MCA. These calibrations must be done locally, with the MCA operating in standalone mode and connected directly to the host computer via USB. Ideally, every MCA would be calibrated prior to being assembled into an array. But if an MCA needs to be calibrated while it belongs to an array, this can be done.

To recalibrate one MCA in an array, first switch the USB host connection to that MCA. Before power on, the array operation must be disabled. This can be done in a couple of ways: 1) break the SPI daisy chain by disconnecting a cable, or 2) leaving one or more MCAs in the array powered off. At power up, either of these modifications will cause the initial synchronization to fail, and the MCAs will all enter standalone mode. Another tactic to get the hosted MCA into standalone mode is to power up the array and then enter stop mode. From stop mode execute a soft reset of the MCA. This causes the hosted MCA to reinitialize but, with the other MCAs already initialized, the timed synchronization will fail and cause the hosted MCA to operate in standalone mode.

No array-specific calibrations are required except for those pertaining to directional sensing. Directional sensing requires the detector indexes are positioned in as specific way, as shown in the following figure for 2x2x2 and 4x4x2 arrays. A smaller array capable of only two dimensional sensing is created by using just the first layer (2x2 or 4x4) of the three dimensional arrangement. The projection matrix to implement the self-shielding direction calculation for a given arrangement of detectors is typically calculated off-line using GEANT4 simulations. The matrix is then programmed into the firmware when the array is built and is not modifiable by the user. To work correctly, the detector indexes must be arranged as shown.



Calibrations

Attempting to re-calibrate the MCA hardware may destroy any existing calibration, so make sure that you fully understand the procedures described below before attempting operations in the Calibrations menu. There is no guarantee that the current calibration was stored in flash at the factory, so the factory reset command might not be able to restore the current calibration. To make sure you can return to an existing calibration, use the *Parameters to console* item in the Console menu to list the parameters. Copy the parameters from the console by highlighting the text with the mouse and using the Enter key to copy the highlighted text from the console to the clipboard. Then paste the parameters into a text editor such as Notepad and save the file for future reference.

When parameters are changed through the Calibrations menu (Figure 25), they are not immediately applied to the MCA – calibration parameter changes must be explicitly sent to the MCA to put them into effect. Calibration parameters can be sent to the MCA when the device is in Stop mode via the *Update MCA parameters...* menu item. This dialog also allows the new calibration parameters to be reviewed before modifying the MCA hardware.

<< Send to MCA is necessary to put parameter changes into effect >>

Calibration parameters are all interrelated and must be adjusted in a particular order for best results. For example, changing the pulse threshold parameter will change the pulse integral and pulse width for a given energy deposition. This change would invalidate most of the calibrations, including temperature stabilization, energy correction, and the limits used for pulse pileup rejection. The pulse width limits should be the final step of calibration, and none of the other calibration parameters should be changed once the limits are established.

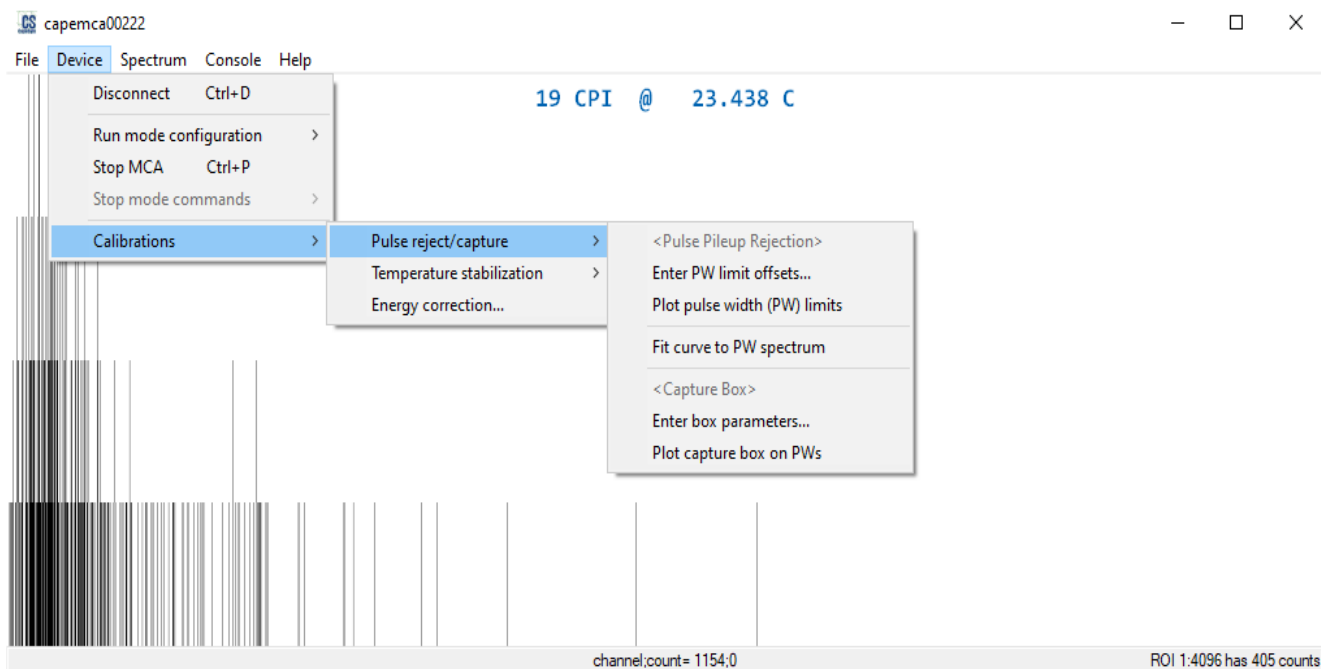


Figure 25: Calibration menu items allow the parameters used for pulse pileup rejection, capture box counting, temperature stabilization, and energy correction to be viewed and modified.

Table 7: Order of Parameter Modifications for Proper Calibration

Step	Index: Parameter(s)	Purpose
1	13: DAC value	Set default high voltage bias applied to SiPM array
2	12: Pulse threshold	Remove counts caused by electronic noise
3	3: Integral divisor	Set approximate channel number for 662 keV photopeak
4	38-56: Add TS lines	Stabilization through optimal DAC vs. temperature
5	57+: Energy correction	Correct channel locations of known energy peaks
6	32-37: PW limiting	Pulse pileup rejection by setting pulse width limits

<< Calibration parameters must be modified in the prescribed order !>>

The order of calibrations and the associated parameters that will be modified are given in Table 7. The entire calibration process is described in the following sections, and the full set of MCA parameters are listed in the MCA Parameters dialog and in Appendix C.

Step 1: High Voltage Bias

A reasonable default DAC level (parameter index 13) is set during the factory programming of the MCA, and can usually be restored by the *Factory settings* item in the Stop mode commands menu. However, if this programming is lost, restoring a reasonable value to this parameter would be the first order of business.

The default 12-bit DAC value needs to be near 2000, in the middle of its range of [0,4095], in order to get a reasonable high voltage bias. This bias must be 1V to 6V over the SiPM breakdown voltage of about 24.5V at room temperature. If the DAC level is zero, the bias output by the MCA will be only 25V to 25.5V, which will be insufficient for Geiger mode operation of the SiPM array. An over-voltage of about 4.5V is recommended for effective room temperature operation.

Prior to final assembly of the MCA with its detector, the bias voltage can be monitored while the DAC level is adjusted until 29V output is obtained. This DAC value then becomes programmed in the factory settings. The bias output pin may not be accessible after assembly, in which case the default DAC value should just be set to 2000 to proceed with the rest of the calibrations.

Step 2: Pulse Threshold

The purpose of this first step is to set the minimum pulse threshold to keep noise out of the low energy channels. When the pulse threshold is set too low, SiPM dark current fluctuations can resemble pulses causing a high count rate in the lowest channels (less than 10). Because the dark current increases with temperature, the determination of the best pulse threshold may require a warming plate or environmental chamber to increase the temperature of the detector to the range of 35-40°C. A pulse threshold of 4 is typically sufficient to suppress noise up to about 40°C. Higher environmental temperatures or excessive radiation damage may warrant a higher pulse threshold, although it may also be sufficient to just shift the ROI to excluded the low energy channels from the display when the returned count rate is not used.

Step 3: Divisor for Integral to Channel

Return the detector and MCA back to room temperature, and wait for the temperature to reach a steady state. Then expose the detector to a known radiation test source with as few energy peaks as possible. For purposes of this discussion, a Cs-137 test source will be used throughout the rest of the calibration procedure. Step 2 will adjust the divisor parameter (index 3) to place the 662 keV peak at channel 662 in room temperature (about 21°C).

Divisor Adjustment Procedure

1. Device > Connect
2. Make sure temperature stabilization, energy correction, and pulse pile rejection are *disabled*
3. Device > Run mode configuration > Zero out spectrum
4. Wait for a well-formed 662 keV photopeak to develop
5. Spectrum > Peak type = Mean
6. Click (left mouse button) on the 662 keV photopeak and note the channel location: c
7. Device > Stop MCA
8. Device > Stop mode commands > Update MCA parameters...
9. Modify the divisor parameter according to the formula: $\text{new divisor} = \text{old divisor} \times \left(\frac{c}{662}\right)$
10. Press [Send to MCA] button
11. Device > Stop mode commands > Memorize MCA parameters
12. Device > Stop mode commands > Soft Reset
13. Re-Connect to the MCA and check the channel location of the 662 keV photopeak

The above procedure may be repeated if the desired accuracy is not obtained, but a stable channel location will not be possible until the calibration of temperature stabilization is completed. If the temperature of the device is changing, making further changes to the divisor will be fruitless. To the greatest degree possible, the temperature should be constant during the above procedure.

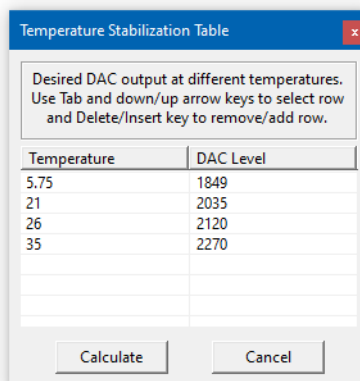
Step 4: Temperature Stabilization

Temperature stabilization is done by setting the proper digital-to-analog (DAC) output to control the high voltage (HV) bias on the SiPM array. The HV bias determines how far above the breakdown voltage the SiPMs are operating, which determines the amplitude of the electrical pulse generated in response to light inside the scintillator. SiPM properties such as breakdown voltage are strongly dependent on temperature, so the bias must be adjusted as temperature changes to maintain a constant pulse amplitude. The microcontroller can do this automatically if the optimal DAC values are known for a few key temperatures.

Thus, the goal of this step is to find the DAC levels that will place the 662 keV photopeak at channel 662 for specific temperatures. The microcontroller will interpolate/extrapolate linearly the DAC levels between/beyond these temperatures. Therefore, calibration requires at least two points, each point consisting of a temperature and corresponding DAC value, in order to stabilize the linear dependence of breakdown voltage on temperature. Additional points may be needed to compensate non-linear temperature dependencies, such as the increase in dark current above room temperature.

The (temperature,DAC) pairs of the calibration are listed in the Temperature Stabilization Table, accessed by Device > Calibrations > Temperature stabilization > Edit HV DAC table... These table entries are measured during the calibration procedure, and then entered by the user. The entries will then be converted

to a set of MCA parameters defining the line segments for temperature stabilization. If the table is empty when the MCA is connected, then no temperature stabilization calibration has been done yet.



Temperature Stabilization Table

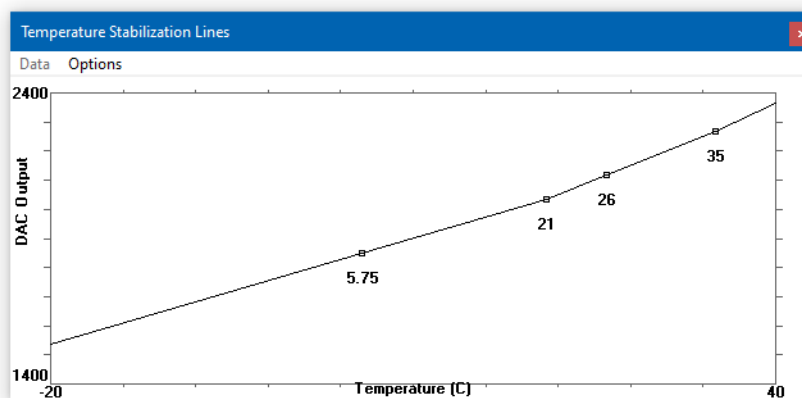
Desired DAC output at different temperatures.
Use Tab and down/up arrow keys to select row
and Delete/Insert key to remove/add row.

Temperature	DAC Level
5.75	1849
21	2035
26	2120
35	2270

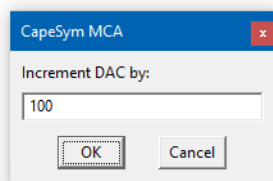
Calculate Cancel

Up to seven points may be used to define the piece-wise linear relationship between DAC and temperature. However, no more than six points will be retained. The last temperature entered in the table is used to determine the slope of the extrapolation at higher temperature. The slope between the first two points in the table determine the extrapolation to lower temperature. For the minimal calibration, two points must be entered and only one point, at the lowest temperature, will be retained after the slope is calculated. This minimal calibration defines a single line for the entire temperature range.

After clicking OK in the table, the full piece-wise linear profile can be visualized by selecting Device > Calibrations > Temperature stabilization > Plot HV DAC lines..., as shown in the following figure. An accurate temperature stabilization should provide one slope at room temperature, along with different slopes in the higher and lower temperature ranges. Increased slope is needed above room temperature, and decreased slope below room temperature.



The calibration procedure involves determining the DAC value that will set the 662 keV photopeak at channel 662 for a few key temperatures. To do this the temperature stabilization, energy correction, and pulse pileup rejection should first be turned off, so that the channel location will be determined only by the HV bias on the SiPMs. To facilitate tracking the location of the photopeak while temperature is changing, it may be helpful to turn on the Moving spectrum option (Figure 24). After clicking on the 662 keV photopeak, set the Peak type to Mean in the Spectrum menu so that the center of the peak is tracked and reported above the spectrum.



The DAC output can be changed while in Run mode by using Calibrations > Temperature stabilization > Bump DAC up/down... as shown above. Just enter an integer between -100 and 100 in this dialog and hit OK to change the DAC value. Increasing the DAC by 100 will increase the HV output by about 0.177 V. The breakdown voltage dependence is about 21.5 mV per °C at room temperature, a coefficient usually specified in the SiPM datasheet. A shift of 10°C will typically shift the 662 keV photopeak location by 60 channels. So a DAC increment of 100 would shift the 662 keV peak location about 50 channels.

One method of conducting the calibration would be to maintain the detector at a fixed temperature and adjust the DAC until the 662 keV photopeak is shifted to channel 662. Then manually record the temperature and DAC value, change the temperature, and repeat the process. To get continual feedback of the current DAC level while in Run mode, set the Rate feedback type to *DAC level for HV* from the Run mode configuration menu. Once a few points have been recorded, they can be entered in the Temperature Stabilization Table. The table will compute the slopes and y-intercepts for the temperature stabilization parameters, and these parameters may then be transferred to the MCA for validation by using the MCA Parameters dialog.

The above calibration method may be difficult to apply if the temperature of the detector cannot be reliably controlled, so an alternative method is detailed below that can be used while the detector slowly warms up from freezing cold. To calibrate at temperatures significantly above room temperature, a warming plate may be required. To slow down the temperature rise for small detectors, it may be helpful to keep a larger thermal inertia, such as a chilled block of leaded glass, in contact with the detector during warm up.

TS Calibration Setup Procedure

1. Device > Connect
2. Device > Stop MCA
3. Device > Stop mode commands > Update MCA parameters...
 - i. Pulse pileup rejection = 0;
 - ii. Temperature stabilization = 0
 - iii. Error correction = 0
 - iv. Moving spectrum = 0
 - v. Communication interval = 10 (for 1 second updates)
 - vi. DAC = 1700-2000 (when TS is off at reset)
 - vii. Rate feedback type = 1 (DAC value and temperature)
4. Press [Send to MCA] button
5. Device > Stop mode commands > Memorize MCA parameters
6. Device > Stop mode commands > Soft Reset
7. Device > Connect
8. Console > Parameters to console
9. Console > Show console
10. Verify all of the MCA parameter settings have been set as desired

After the above procedure, the MCA and detector can be unplugged from USB power and placed in a frost-

free freezer until they are well below 0°C. (The minimum operating temperature is -40°C.) Freeze the thermal inertia block to the same temperature. Keep everything frozen until ready to begin recording spectra in Step 4 of the following calibration procedure.

The goal of the following procedure is to automatically record temperature and DAC values at closely spaced temperature intervals while the DAC value is stepped through increasing values. For each DAC step, the temperature should be allowed to pass through the temperature at which the photopeak aligns with channel 662. After the photopeak moves below 662, the DAC will be incremented to the next step while recording continues. A series of (temperature,DAC) pairs can then be inferred from the recorded data to determine the values to enter into the Temperature Stabilization Table. Figure 26 illustrates the procedure.

A moving spectrum is not required during the Auto Save Procedure, because the spectrum will be zeroed out after it is recorded at each 0.5°C interval. Each spectrum will contain all of the events that occurred during each 0.5°C rise in temperature. Hence, the peak location in the spectrum will have occurred at the recorded temperature T minus half the interval width, or approximately $T - 0.25^{\circ}\text{C}$.

TS Calibration Auto Save Procedure

1. File > Auto save...
 - i. Filename: spectrum_0000.csv
 - ii. Check *Encode 16-temperature with 16-bit CPS* (which will be DAC instead of CPS)
 - iii. Save file at intervals of Temp. change = 0.5
 - iv. Leave ENABLE AUTO SAVE unchecked for now
2. Press [OK] to keep these Auto save settings
3. Spectrum > Peak type > Mean
4. Remove the components from the freezer and place in foam insulation with the thermal inertia touching the detector. Place the Cs-137 close enough to the detector for about 500-1500 CPS
5. Plug in the USB cable.
6. Device > Connect
7. File > Auto save... to check the ENABLE AUTO SAVE and hit OK
8. Click the left mouse button while the mouse pointer is on the 662 keV photopeak
9. Device > Calibrations > Temperature stabilization > Bump DAC up/down... and increment or decrement as needed to put the photopeak about 20-30 channels above channel 662
10. Keep the DAC constant until photopeak is about 10-20 channels below channel 662
11. Device > Calibrations > Temperature stabilization > Bump DAC up/down... increment by 75
12. Device > Run mode configuration > Zero out spectrum
13. Right click on spectrum with mouse to clear the peak tracking.
14. Click left mouse button while the mouse pointer is on the 662 keV photopeak
15. Keep the DAC constant until photopeak is about 10-20 channels below channel 662
16. Repeat from steps 11-15 until the detector reaches the desired maximum temperature or
17. To continue beyond room temperature, transfer detector to warming plate, and repeat steps 11-15 until the detector reaches ~40°C
18. File > Quit autosave
19. Device > Disconnect

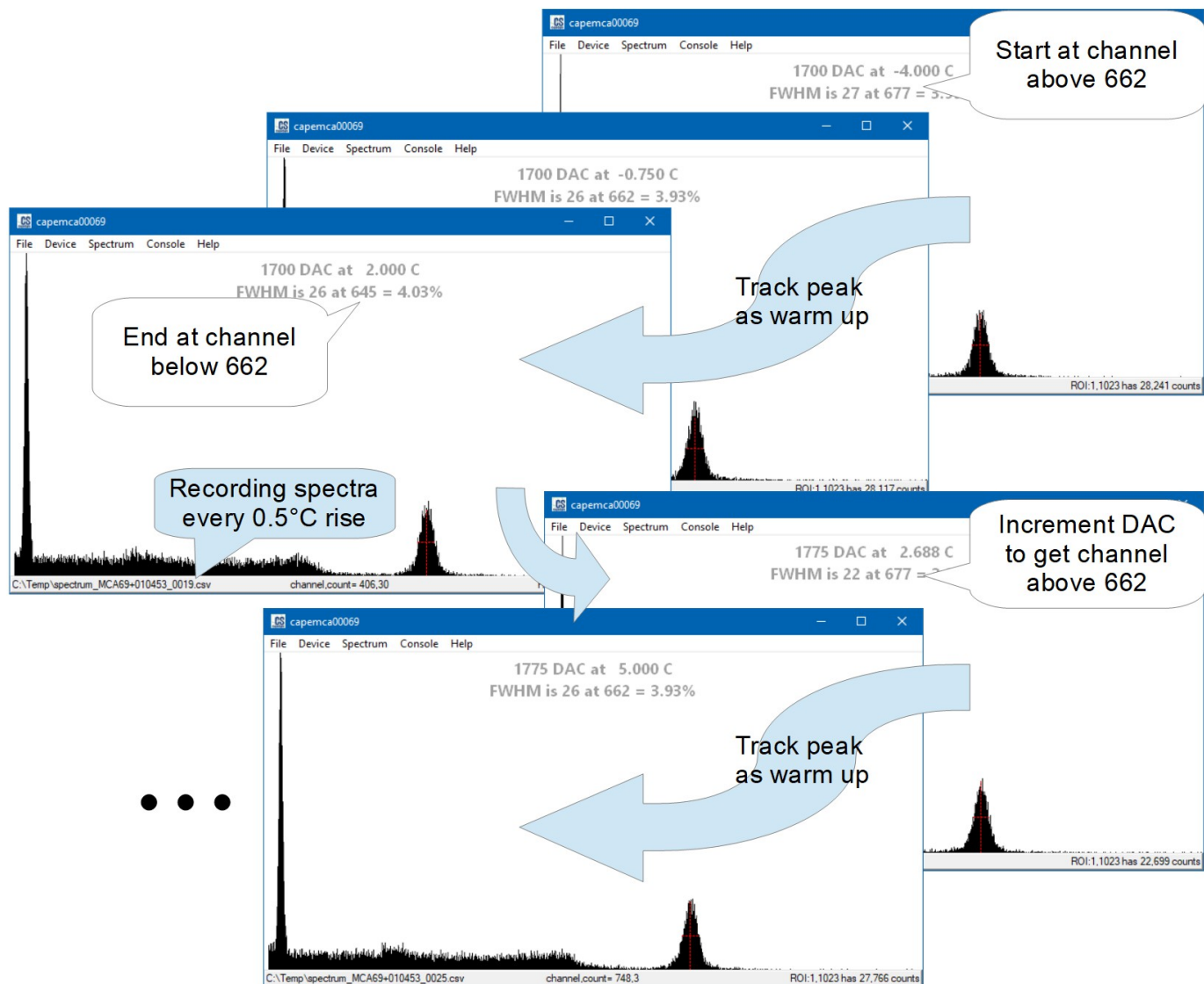


Figure 26: The TS calibration Auto Save procedure involves periodically shifting the DAC level so that the channel location of the 662 keV photopeak passes through 662 as the detector warms up. By auto-saving a spectrum at every half a degree rise in temperature, a record is made of the temperature where channel 662 is crossed for every DAC level specified.

The entries of the Temperature Stabilization Table can now be determined from the data contained in the sequence of spectra recorded by the above procedure. This information is easily extracted using the Auto Load procedure with peak tracking turned on, as follows. The photopeak will be tracked automatically, at least until the next DAC level where there is a jump in the channel location. If this jump is too wide, tracking may fail, in which case the right and left mouse clicks will be required to clear the bad track and get tracking back on the peak. Dealing with large jumps may require increasing the display time for each load to allow this manual intervention, or by limiting the number of loads to just the spectra from a single DAC level. The following procedure assumes that the jumps are small enough to allow continuous tracking over the whole data set.

TS Calibration Auto Load Procedure

1. File > Load spectrum to load the first spectrum in the data set: spectrum_0001.csv
2. Spectrum > Peak type > Mean
3. Click on the 662 keV photopeak to begin tracking its location.
4. Spectrum > Peaks to console
5. Console > Show console
6. File > Auto load...
 1. Filename: spectrum_0001.csv
 2. Time (seconds) = 1
 3. Loop: check *Never*
 4. Check ENABLE AUTO LOAD
7. Press [OK] to begin the Auto load operation.
8. Observe that temperature and channel location is being tracked correctly and reported to the console. The sequence will stop automatically when the all files have be processed.
9. Uncheck Spectrum > Peaks to console
10. In the console, use the mouse pointer to highlight the temperature and peak location data, then press Enter to copy this data to the clipboard.
11. Open a spreadsheet and paste the data into it.

The spreadsheet data may now be separated into the channel location tracks for each DAC level, as shown in Figure 27. Line fits to these tracks allow the temperatures of channel 662 crossings to be accurately inferred, as depicted by the X marks in the figure. Plotting the DAC levels versus these temperatures (Figure 28) provides a set of potential table entries for the TS calibration. A single line fits the middle of the temperature range (5°C to 32°C) pretty well for this example, so a single line segment between the second and seventh points was deemed to be sufficient. However, significant deviations from linearity can occur, especially above and below the room temperature range. The final table entries chosen for this detector are the first two and the last two data points (Figure 29).

The Temperature Stabilization Table allows the user to enter up to seven points to define six lines, describing a piece-wise linear relationship between the DAC and temperature. Fewer lines may be defined using fewer points, but a minimum of two points is required for a line to be calculated. Extrapolation to temperatures lower than the first table entry uses the slope defined by the first two points, and extrapolation to temperatures higher than the last entry uses the last two points. Only the first temperature for each line segment is stored as a parameter, so the last table entry will be forgotten when the lines are calculated. Reopening the table will show one fewer points and temperatures rounded to the nearest 1/16th of a degree. The last point, although forgotten, could be inferred from the temperature calibration lines plot, and restored if desired.

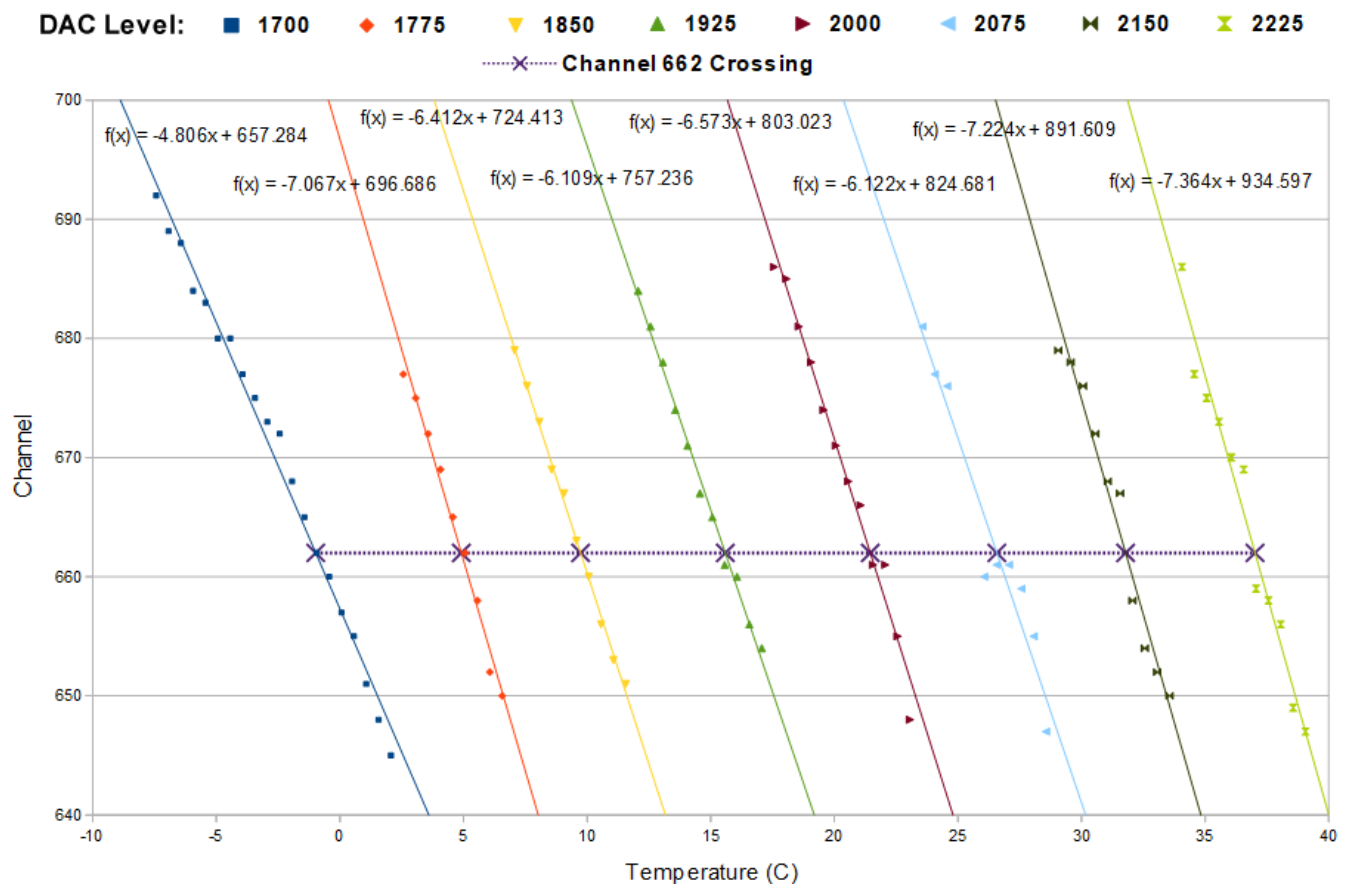


Figure 27: The spreadsheet data separated into the channel location tracks for each DAC level.

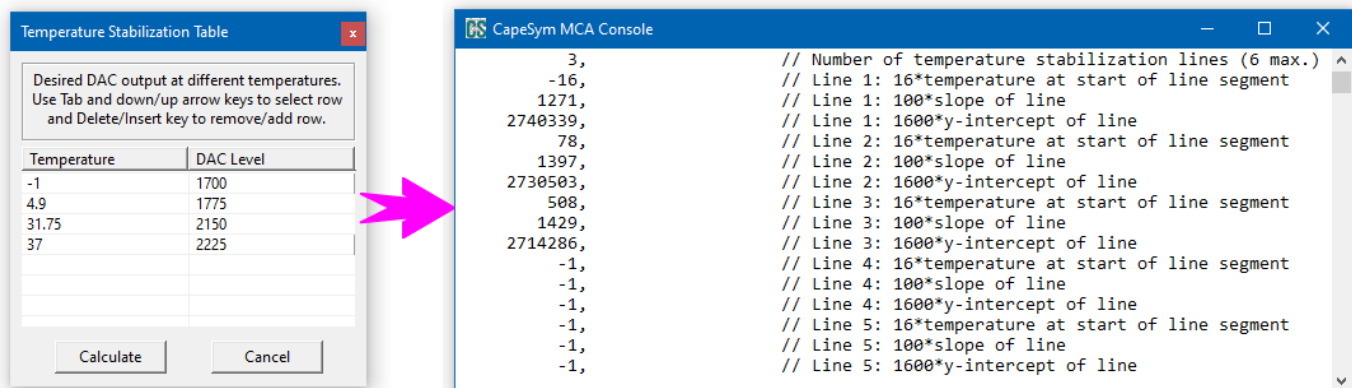


Figure 28: The TS calibration is completed by entering selected points into the table (left), and pressing the Calculate button. The actual lines of the TS calibration are then calculated and added to the MCA parameters which can be viewed in the console. As discussed above, four entries results in the three lines.

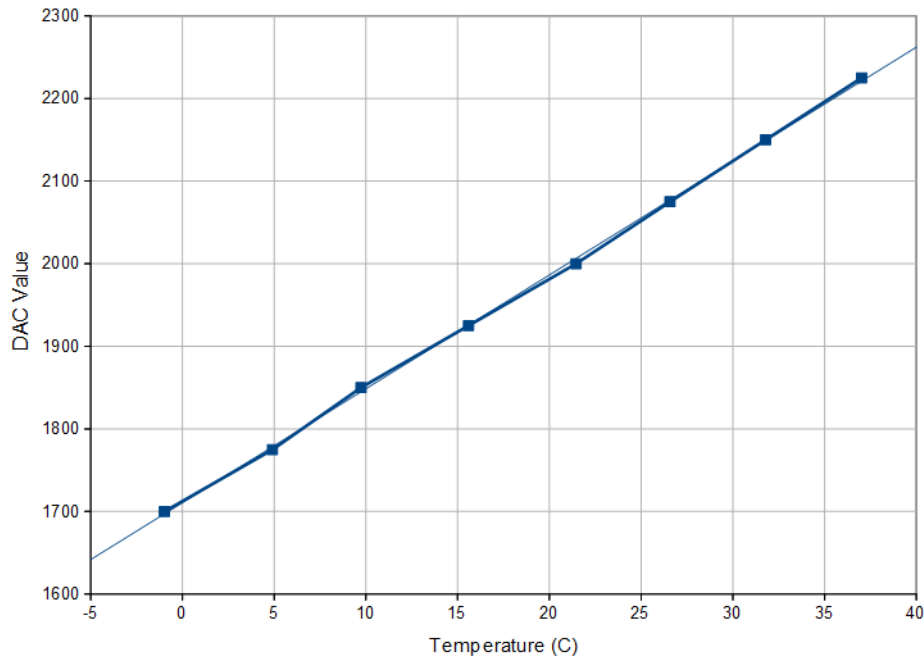


Figure 29: Plot of the 662 crossing from spreadsheet analysis of the previous figure. In this temperature range, the data is well-fit by a straight line.

The TS calibration lines can be viewed in the console, by using the Parameters to console menu item, or in the MCA parameters dialog. The lines can also be graphed by selecting Device > Calibrations > Temperature stabilization > Plot HV DAC lines... (Figure 30). For this example, only three points are displayed in the plot and in the table, even though four points were originally entered (Figure 29). Recalculating lines from the three points remaining in the table will eliminate one of the four line segments shown in the plot, and result in just two temperature points remaining. Calculating lines from fewer than two entries will completely eliminate the temperature stabilization parameters.

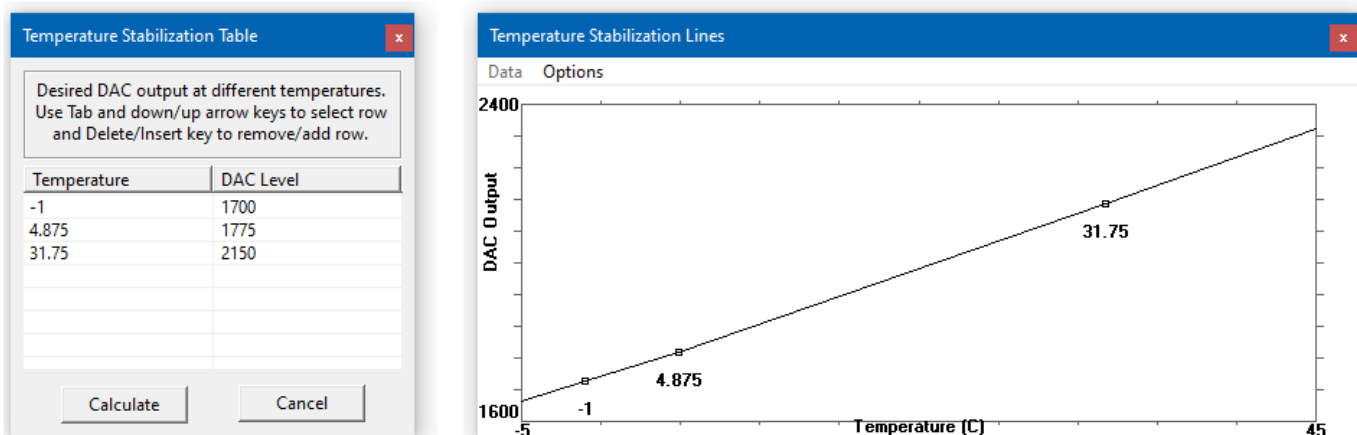


Figure 30: The TS calibration table and lines after the user's entries were processed. Each line segment is represent by a starting temperature point followed by a slope and y-intercept to define the DAC levels.

Once entered in the table, the TS calibration lines reside only in the host software memory. The updated MCA parameters must be sent to the MCA by using the MCA Parameters dialog after entry. The table entries must be entered AFTER connecting to the MCA because the Connect operation overwrites the host's copy of the MCA parameters. For best results use the following procedure.

TS Calibration Entry Procedure

1. Device > Connect
2. Device > Stop MCA
3. Device > Calibrations > Temperature stabilization > Edit HV DAC table...
4. Enter the (temperature,DAC) points as discussed in the above text.
5. Press [Calculate] button in the Temperature Stabilization Table dialog
6. Device > Stop mode commands > Update MCA parameters...
7. Press [Send to MCA] button
8. Device > Stop mode commands > Memorize MCA parameters
9. Device > Stop mode commands > Soft Reset

TS Auto Calibration

The calibration steps are consolidated in an automatic process which may be invoked by the menu item *Calibrations > Temperature stabilization > Auto calibration...* This item will open the Auto TS Calibration dialog. The dialog will remain open for the duration of the calibration, which may take more than an hour to complete. Access to other software functions are blocked while the dialog is open, so the software and MCA parameters should be properly configured prior to opening the dialog.

MCA parameters should be configured as in the ***TS Calibration Setup Procedure*** described above, with pulse pileup rejection, temperature stabilization, energy correction, and moving spectrum turned off, and with DAC and temperature feedback enabled. Peak tracking will be used in the host software, so the menu items *Peak type > Mean* and *Label peak when add* should be checked in the Spectrum menu. Place the chilled detector near a Cs-137 test source (500-1500 CPS). Then plug in the USB connector and execute the following procedure. Choose the number of lines assuming each will span no more than 5°C. A 25°C span would need at least 5 lines, but maybe up to 7 lines. The calibration may be completed early with fewer lines by using the Done button when the dialog makes it available.

TS Auto Calibration Procedure

1. Device > Connect
2. Click on the 662 keV photopeak with the left mouse button and add the label 662
3. Device > Calibrations > Temperature stabilization > Auto calibration...
4. Verify that the DAC value is displayed in the Current DAC value box. If not correct, cancel the dialog and set the Feedback type to DAC and temperature, then reopen the dialog.
5. Set the temperature resolution to 0.5°C
6. Set the number of lines to fit. (Seven can be kept, but more fitted before down selection.)
7. Press the [Start] button to start the data collection process

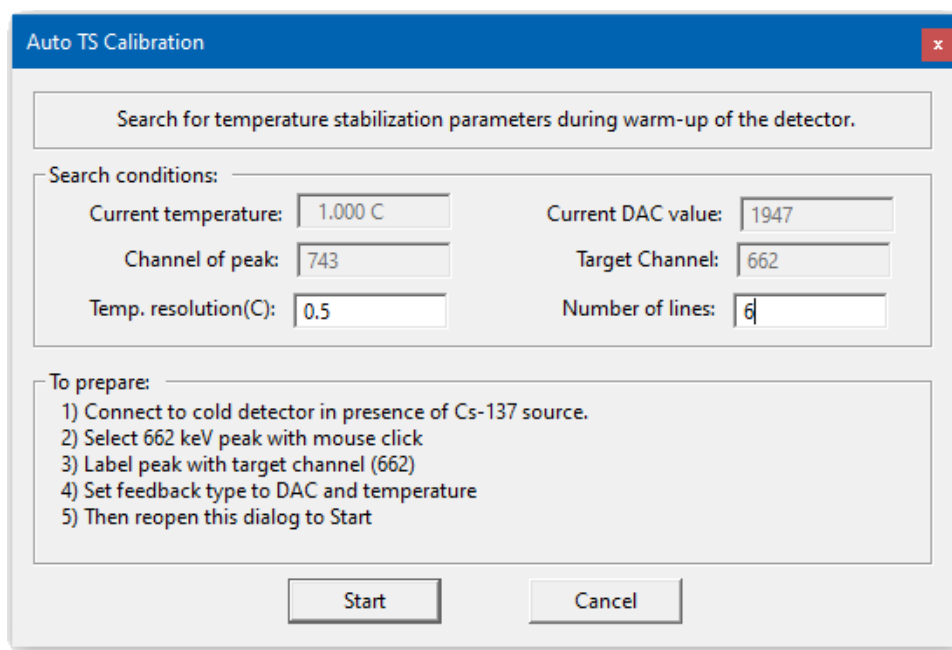


Figure 31: The appearance of the dialog for automatic TS calibration when everything is ready to Start the calibration process.

The calibration will proceed in three stages. In stage 1, the DAC will be adjusted to bring the channel location of the photopeak to just above the target channel. As the detector warms up, the shift in the SiPM breakdown voltage will move the photopeak below the target channel. Capturing this movement of the peak through the target channel is the purpose of stage 2. Stage 2 is repeated at a different DAC value for each line specified in Step 6 above. The software records the temperature and peak location for every 0.5°C rise in temperature. These values are reported to the console while the auto calibration is working. Copying and plotting these values would produce a figure similar to Figure 27, but the line fit is done automatically in stage 3 now.

When the channel location falls ~2.3% below the target channel, the DAC is incremented by 75. This is stage 3, which marks the end of data collection for one line and the start of data collection on the next one. Because of the increase in the DAC, the peak location will jump about 20 channels above the target. The calibration then returns to stage 2, with more temperature and peak location points recorded while the detector continues to warm up. Alternations of stage 2 and 3 will continue to be reported in the dialog until all of the data points have been collected for the last line (Figure 32). The dialog must remain open continuously throughout this entire process.

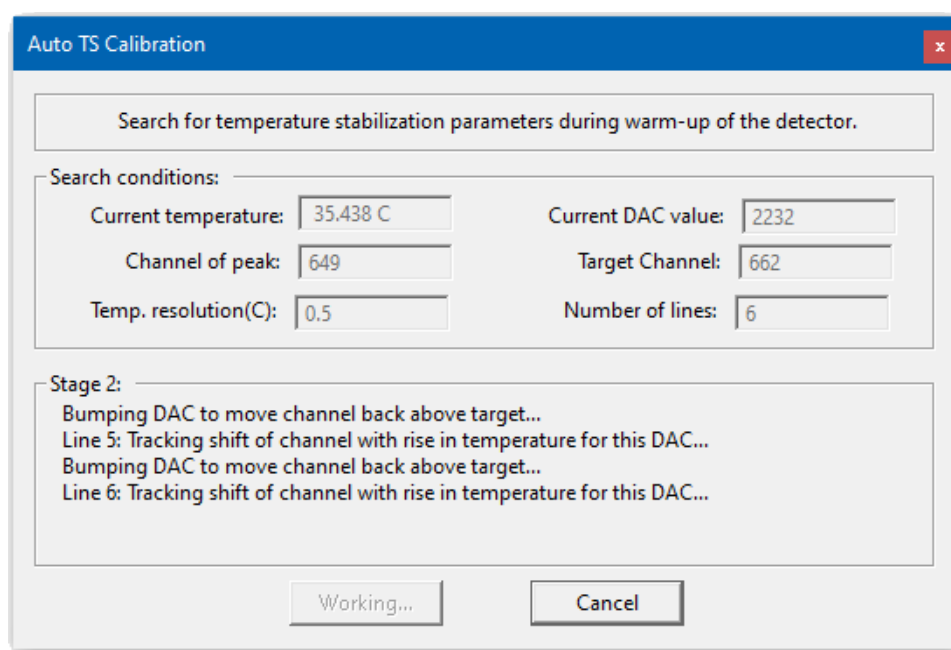


Figure 32: The appearance of the dialog for automatic TS calibration near the end of the data collection process.

When the final line is complete, the calibration dialog will close and the Temperature Stabilization Table and Lines will be displayed. The table may then be edited and used to calculate the MCA parameters as in the above TS Calibration Entry Procedure. The calibration procedure may be terminated early, provided enough 662 channel crossings were collected to move forward with a calibration. If there is not enough data yet, the Cancel button will simply terminate the process and close the dialog. At which point the entire automatic calibration procedure must be restarted with the detector in a cold state.

Quick Two-Point Calibration

If temperature stabilization is only needed at room temperature, a quick two-point stabilization will suffice for many applications. The quickness of the two-point calibration comes from the fact that the **Divisor Adjustment Procedure** (Step 3) can be done at the same time as stabilization, or it can be skipped altogether by using an initial DAC adjustment to place the target photopeak at the desired channel for the first temperature point.

The idea for this procedure is to record the DAC for when the detector is actually at room temperature, and then record it again after the MCA has heated up the detector by several degrees. These two temperature states can usually be controlled by a fan blowing room temperature air across the MCA. When the fan is on, the MCA should remain near room temperature, and when the fan is off the MCA should heat up the detector by several degrees. Aim for a temperature difference of more than 5°C for the two points. In self-heating is insufficient for a large temperature difference, try blocking convective air flow around the device or placing it against a warm surface such as a cup of coffee or a cup warmer.

MCA parameters can be configured as in the **TS Calibration Setup Procedure** described above, with pulse pileup rejection, temperature stabilization, and energy correction turned off, and with DAC and temperature

feedback enabled. However, it is not necessary to disable tracking mode. Tracking mode might be helpful during this procedure, as long as the Moving spectra provide enough counts to accurately determine the photopeak channel location while the temperature of the detector is slowly drifting.

Quick Two-Point Calibration Procedure

1. With the fan on, turn on the MCA and Connect to it. Note the first DAC and temperature point.
2. Note the channel location c of the 662 keV photopeak
3. Use *Device > Stop MCA* to enter Stop mode
4. *Device > Stop mode commands > Update MCA parameters...*
5. Modify the divisor parameter according to the formula: $\text{new divisor} = \text{old divisor} \times \left(\frac{c}{662}\right)$
6. Press [Send to MCA] button
7. *Device > Stop mode commands > Memorize MCA parameters*
8. *Device > Stop mode commands > Soft Reset*
9. Re-Connect to the MCA and check the channel location of the 662 keV photopeak. The photopeak should be close to channel 662 while the temperature is still near room temperature.
10. Turn the fan off and let heating by the MCA warm up the detector several degrees
11. Adjust the DAC to shift the photopeak location back to near channel 662
12. Note the second DAC and temperature point
13. *Device > Stop MCA*
14. *Device > Calibrations > Temperature stabilization > Edit HV DAC table...*
15. Enter the two (temperature,DAC) points noted in steps 1 and 12
16. Press [Calculate] button in the Temperature Stabilization Table dialog
17. *Device > Stop mode commands > Update MCA parameters...*
18. Verify that the temperature stabilization parameters were updated by the [Calculate] button
19. Set the temperature stabilization on/off parameter to 1
20. Press [Send to MCA] button
21. *Device > Stop mode commands > Memorize MCA parameters*
22. *Device > Stop mode commands > Soft Reset*
23. Turn the fan back on, then re-Connect and verify that the 662 keV photopeak location is stabilized near channel 662 as the detector module cools off.

Step 5: Energy Correction

The detector response is not proportional to the energy deposited over the full gamma energy range, from 10 keV to 4000 keV. This non-proportionality means that stabilization of 662 keV at channel 662 will not also place other characteristic gamma energies at their expected channel locations. For example, the Co-60 photopeak at 1174 keV will be stabilized at a much lower channel than 1174 when the Cs-137 photopeak is used for TS calibration.

<< Calibration of temperature stabilization must be done first >>

The MCA firmware allows the channel locations of a wide range of gamma energies to be remapped to appropriate channels, through entries in the Energy Correction Table (Figure 33). The procedure for updating the Energy Correction Table is given in this section.

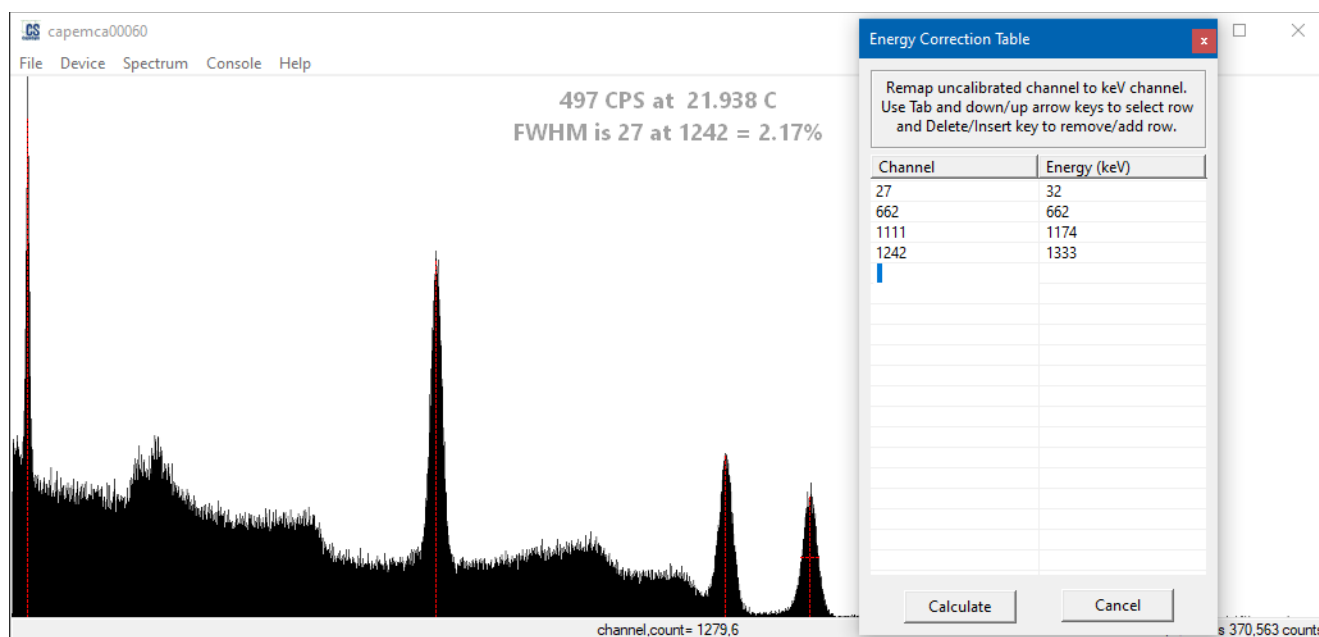


Figure 33: The Energy Correction Table specifies that peaks found at particular channels (first column) are to be remapped to the channel locations corresponding to their characteristic gamma energies in keV (second column).

The channel entries for the Energy Correction Table must be measured on one or more spectra after temperature stabilization has been calibrated and while temperature stabilization is turned on. For best results, the temperature stabilization will have also been validated, as described in the next section.

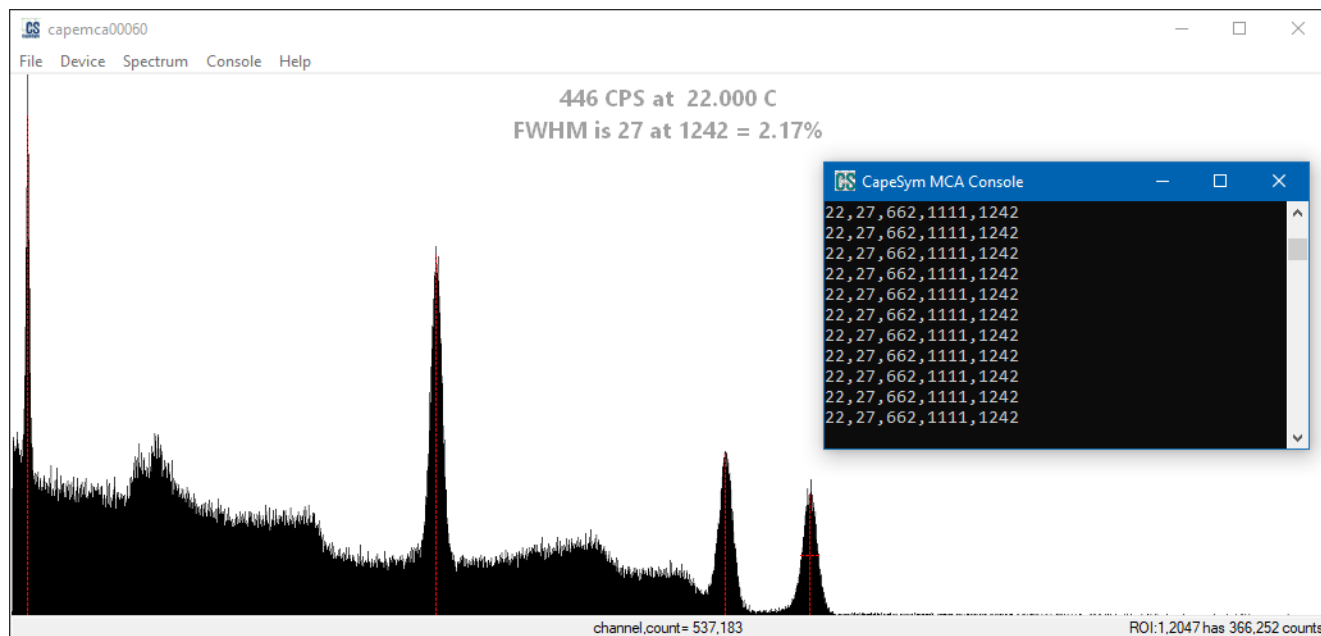


Figure 34: The process of measuring the channel locations of the four peaks of the spectrum from a combination of Co-60 and Cs-137. Peaks are marked and tracked while being continuously reported the console, until sufficient counts have been obtained to ensure accurate channel locations.

To demonstrate the procedure, a spectrum from Cs-137 and Co-60 is used, which has four peaks at energies of about 32 keV, 662 keV, 1174 keV, and 1333 keV (Figure 34). Only four peaks are used for calibration here, but many more peaks would be needed to accurately compensate for detector non-proportionality over the full energy range. The most non-linear region of non-proportionality is below 662 keV. The single peak at 32 keV is not sufficient to get good accuracy throughout the low-energy range. Additional calibration peaks will be needed in this range to guarantee an accurate calibration.

EC Calibration Procedure

1. Device > Connect
2. Make sure temperature stabilization is **enabled**, while both energy correction and pulse pile rejection are **disabled**
3. Device > Run mode configuration > Zero out spectrum
4. Use left mouse button clicks to select the peaks to track
5. Spectrum > Peak type = Mean
6. Spectrum > Peaks to console
7. Console > Show console
8. Wait for spectrum to accumulate a large number of counts for accuracy
9. Device > Stop MCA
10. Device > Calibrations > Energy correction....
11. Enter channel locations as listed in console and enter their gamma energies
12. Press [OK] to calculate the MCA parameters
13. Device > Stop mode commands > Update MCA parameters...
14. Press [Send to MCA] button to transfer the parameters
15. Device > Start up
16. Device > Run mode configuration > Energy correction > Enable EC
17. Device > Run mode configuration > Zero out spectrum
18. Use right mouse button to erase the old peak tracks
19. Use left mouse button clicks to select the peaks to track
20. Console > Show console
21. Wait for spectrum to accumulate a large number of counts to verify the correction
22. Device > Stop MCA
23. Device > Stop mode commands > Memorize MCA parameters

As shown in Figure 35, the corrected peaks are within one channel of the desired locations, for an accuracy of less than 1%.

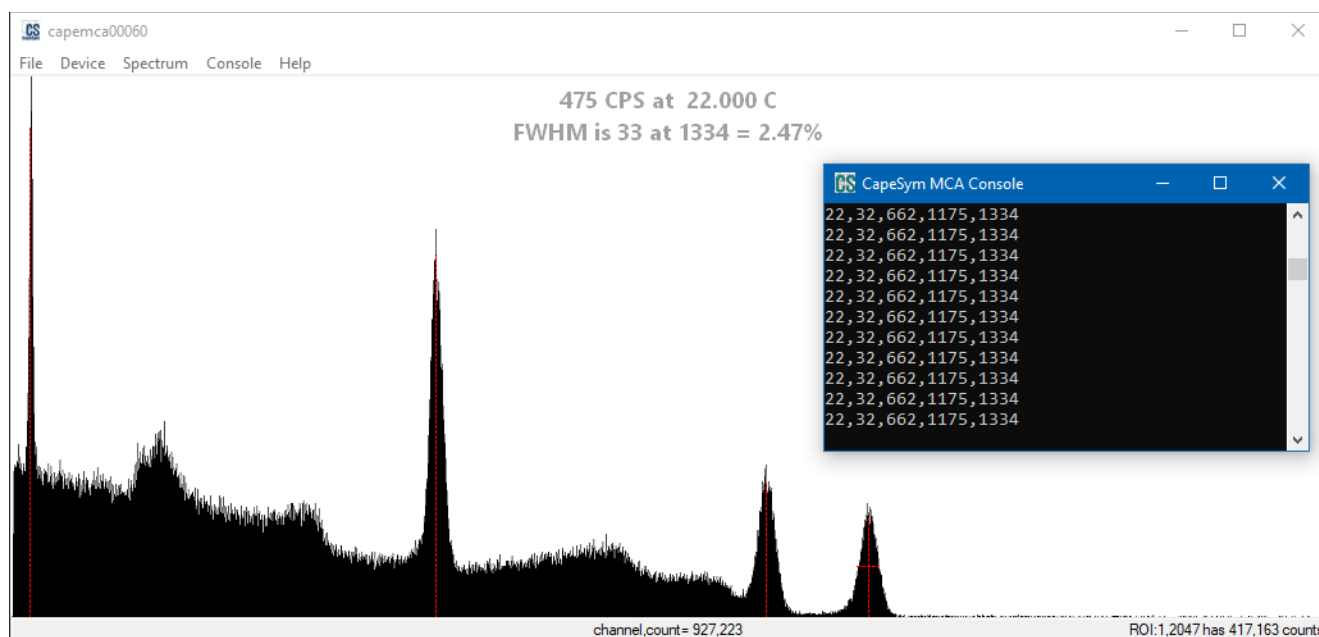


Figure 35: Validation of energy correction is Step 21 of the calibration procedure. The new peak locations reported to the console closely match the specified energies.

Stabilization and Energy Validation

After the temperature stabilization lines have been memorized, the sensor should be tested over a wide temperature range to verify the validity of the calibration. Further adjustments of the DAC at temperatures with significant deviations from the stable channel may then be made in the HV DAC table to update the calibration. Once the temperature stabilization (TS) is satisfactory, energy correction (EC) can also be applied and the combined effects of both adjustments validated over the desired temperature range.

In the following, two procedures are described for the validation. The Quick Stability Check makes use of the moving spectra capability to allow the channel location of a peak to be easily tracked during warming of the detector. However, there is no permanent record of the stabilization with this method. The second validation method involves Auto Saving a spectrum after every 0.5°C rise in temperature while the detector is warming. Then Auto Load with photopeak labeling may be used to track multiple peaks and create a spreadsheet of the stability performance, as demonstrated below.

For both test scenarios, the temperature of the detector and MCA will need to be varied over a wide range. An environmental temperature chamber may be helpful, but a household refrigerator might suffice. Putting the sensor in the freezer for a hour will lower the temperature well below zero. Letting it warm to room temperature would then provide a temperature range of at least -5°C to 25°C. Transferring the sensor to a warming plate after it reaches room temperature can extend the range further, but try to limit the range to about 40°C. Above 40°C noise is likely to become a problem due to increased dark current in the SiPMs.

Quick Stability Check

1. Device > Connect
2. Device > Stop MCA
3. Device > Stop mode commands > Update MCA parameters...
 - i. Temperature stabilization = 1
 - ii. Energy correction = 1
 - iii. Turn on Moving spectrum = 1 and Depth of moving spectra = 32
 - iv. Communication interval = 10 (for 1 second updates)
4. Press [Send to MCA] button
5. Device > Start up
6. Click on 662 keV peak in the main window and set Spectrum > Peak type > Mean
7. Continuously monitor the peak location as the sensor warms up

Create Stability Record

1. File > Auto save...
 1. Filename: spectrum_0000.csv
 2. Check *Encode 16-temperature with 16-bit CPS*
 3. Save file at intervals of Temp. change = 0.5
 4. Leave ENABLE AUTO SAVE unchecked for now
2. Press [OK] to keep these Auto save settings
3. Spectrum > Peak type > Mean
4. Remove the sensor from freezer and place near the Cs-137 source for 500-1500 CPS
5. Plug in the USB cable.
6. Device > Connect
7. Device > Run mode configuration > Temperature stabilization > Enable TS
8. File > Auto save... to check the ENABLE AUTO SAVE and hit OK
9. Wait for detector to warm up all the way while auto-saving spectra.
10. File > Quit autosave
11. Device > Disconnect
12. File > Load spectrum to load the first spectrum in the data set: spectrum_0001.csv
13. Left mouse click on the photopeak(s) to track.
14. Spectrum > Peaks to console
15. Console > Show console
16. File > Auto load...
 1. Filename: spectrum_0001.csv
 2. Time (seconds) = 1
 3. Loop: check *Never*
 4. Check ENABLE AUTO LOAD
17. Press [OK] to begin the Auto load operation.
18. Observe that temperature and channel location is being tracked correctly and reported to the console. The sequence will stop automatically when the all files have be processed.
19. Uncheck Spectrum > Peaks to console
20. In the console, use the mouse pointer to highlight the temperature and peak location data, then press Enter to copy this data to the clipboard.
21. Open a spreadsheet and paste the data into it.

To validate the stabilization, the above procedure was used on a sensor twice, once with TS disabled and once with TS and EC enabled (Figure 36). The detector was removed from the freezer and exposed to Cs-137 and Co-60 sources. A spectrum was recorded after every 0.5°C change in temperature using Auto Save. The peaks were then labeled and printed to the console during Auto Load. The console data was copied and pasted into a spreadsheet for plotting. With TS enabled, the peak locations were constant over a wide range of temperatures, demonstrating the accuracy of the TS calibration.

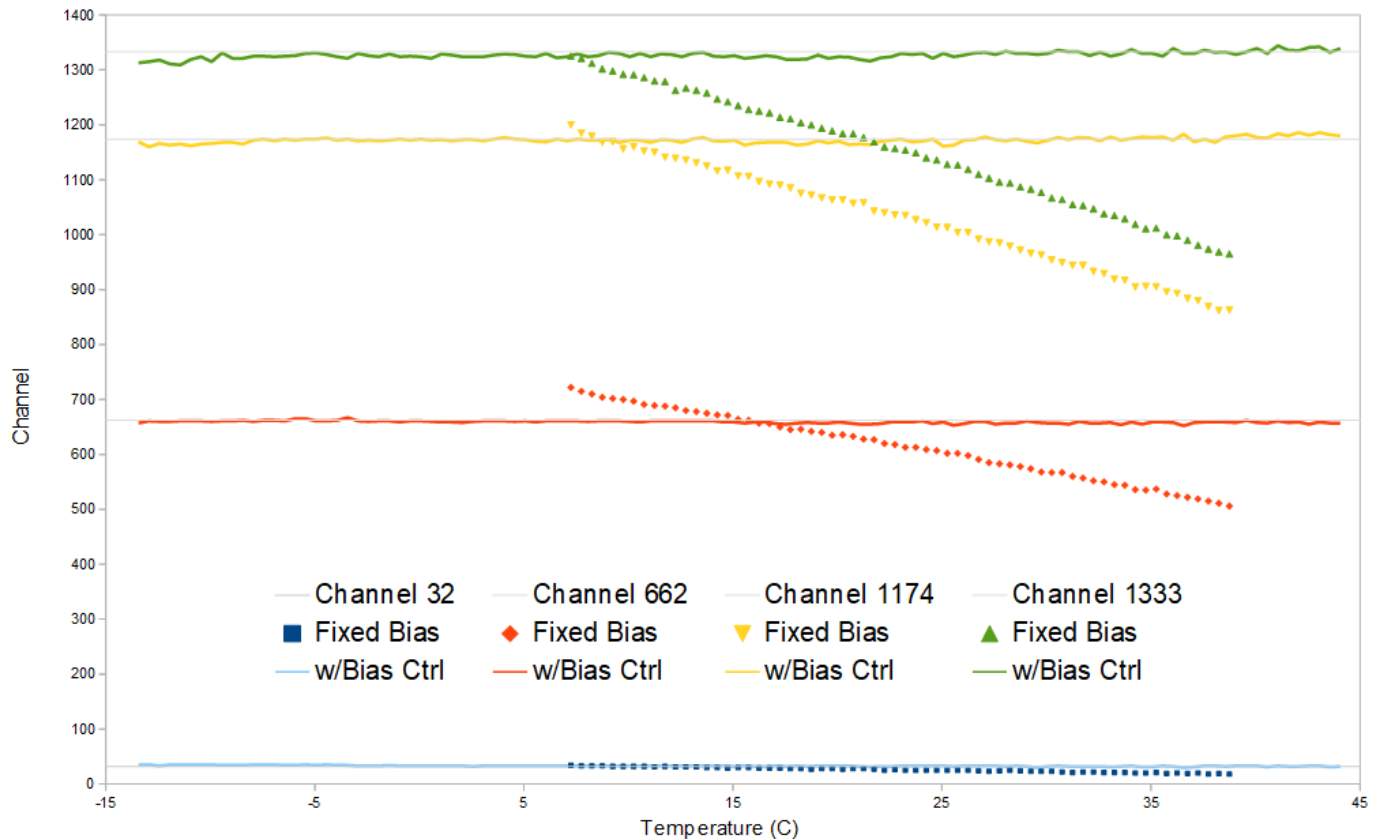


Figure 36: The gamma energy peaks of Cs-137 and Co-60 closely track the desired channels over a wide range of temperatures after calibration of the temperature stabilization and energy correction. Prior to these calibrations, when bias voltage was constant, a change in temperature caused a shift in the channel location of characteristic peaks that was proportional to the gamma energy.

Step 6: Pulse Pileup Rejection

Pulse pileup may be detected by comparing the width of pulses at high counts rates to the expected widths at low count rates. The following figure shows the pulse width spectrum at a count rate of ~2000 CPS. Compare this to the next figure obtained at 100 times lower count rate. There are many pulse widths above the expected pulse width curve because they are from two overlapping pulses.

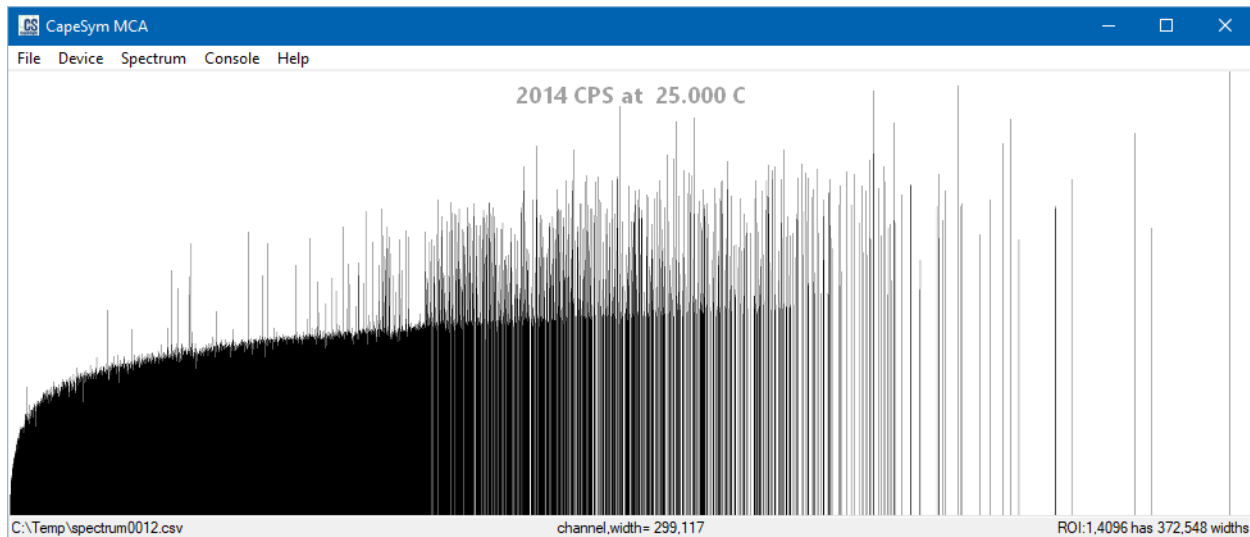


Figure 37: The pulse width spectrum at ~2000 CPS.

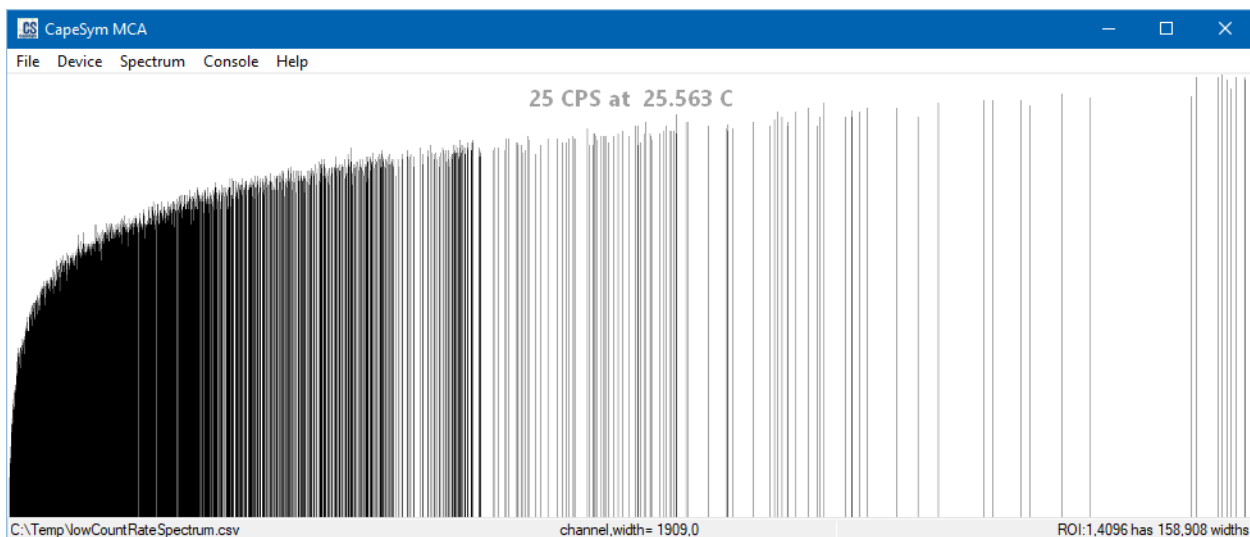


Figure 38: The pulse width spectrum at ~20 CPS.

Pulses having widths far from the expected curve may be easily rejected once the pulse width curve is described mathematically. To obtain this description, first acquire a pulse width spectrum in a low rate environment, for example in normal background. Ideally, the pulse width spectrum used for calibration would include all the energy channels. As a general rule, the denser the pulse width data, the better the

curve fit. It is especially important to acquire plenty of high-energy pulses in the upper channels. If there is an existing calibration, you may want to turn on the Pulse Pileup Rejection (PPR) during the low count rate acquisition to further ensure the ideal case of no pileups being recorded.

The pulse width curve from the current calibration may be overlaid on the acquired spectrum by using the Calibrations menu to select *Pulse pileup rejection > Plot pulse width limits*, as shown in Figure 37. Recall that the current calibration parameters from the MCA hardware were transferred to the host software when the USB connection was made. The overlaid plot allows evaluation of the goodness of fit of these the current calibration parameters. Only pulses with widths between the blue and magenta lines are added to the spectrum when pulse pileup rejection is turned on.

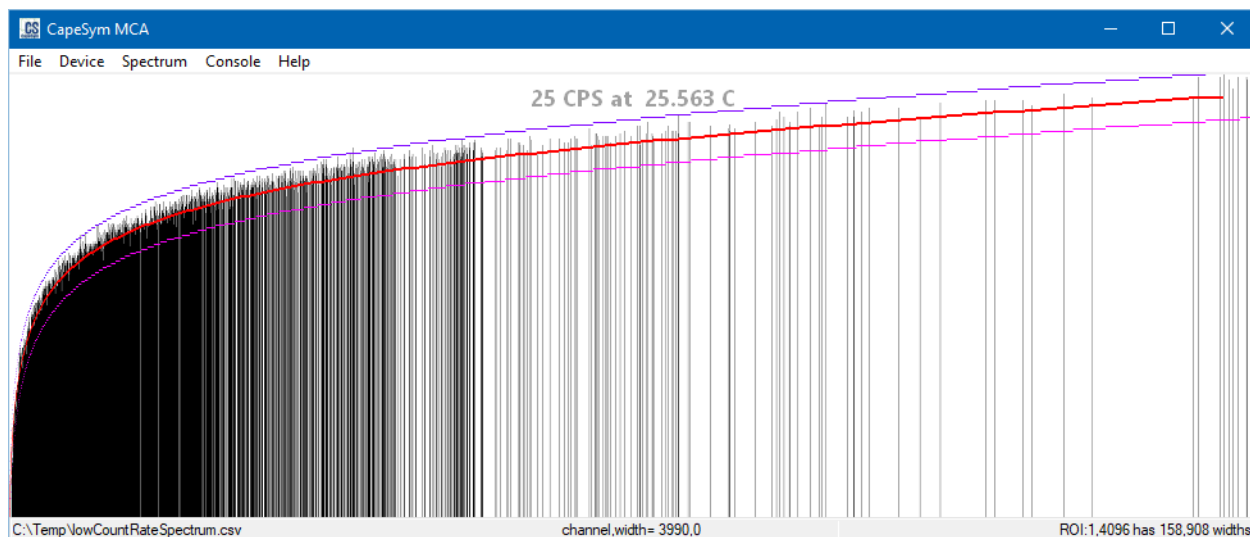


Figure 39: The plot width curve and limits from an existing calibration are overlaid on the pulse width spectrum.

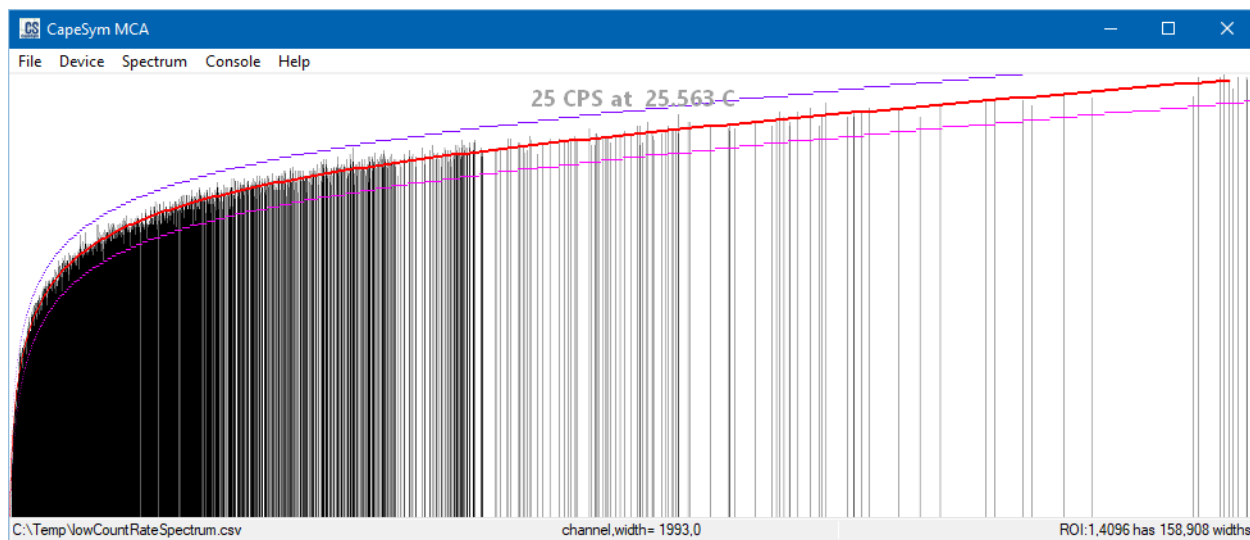


Figure 40: The pulse width curve fit to the pulse width spectrum is closer to the data.

The coefficients of the fitted curve (red line) may be recalculated in the host software by selecting *Pulse pileup rejection > Recalculate from spectrum*. The overlaid plot will be updated immediately, helping to visualize the change produced by recalculation, as depicted in the Figure 40. Be sure to check the new curve fit in the lower channels by zooming in on them with the mouse wheel (Figure 41). Old and new curve fit coefficients are written to the console during recalculation.

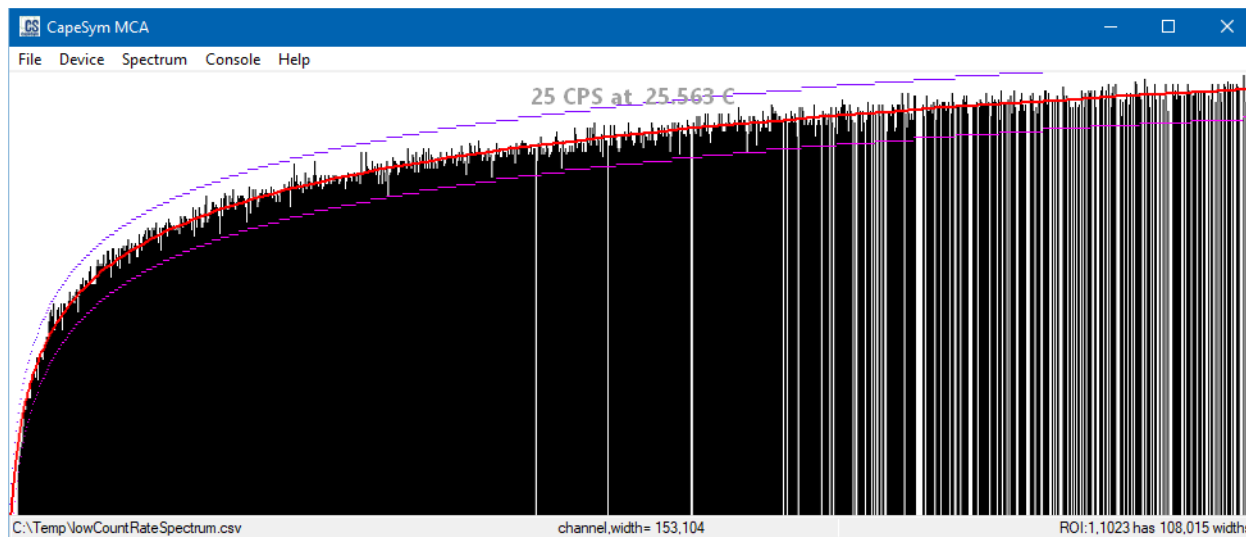


Figure 41: Zooming in on the lower energies to examine the pulse width curve fit closely.

If the change produced by recalculating the coefficients is not an obvious improvement, the pulse width limits calibration should probably be abandoned. At this point, the coefficients from the old calibration have been overwritten in the host software by the Recalculate command, but they can be recovered by exiting Stop mode, and then using a Disconnect/Connect sequence to reload them from MCA hardware.

To refine the calibration, the offsets from the fitted curve (red) to the upper (blue) and lower (magenta) pulse width limits may now be adjusted. These limits set the range of acceptable pulse width variations along the whole curve fit. The pulse width and these offsets are both expressed in terms of the number of samples at the sampling rate of the MCA. Offsets of 10 samples, about 7% of a large pulse width of 150 samples, appear to give reasonable PPR while not rejecting low energy pulses.

Once the new parameters are established they can be sent to the MCA (RAM memory) for live testing before making them permanent. To do this, select *Send to MCA* from the MCA Parameters dialog. Then Start up the MCA loop to start receiving spectra again. If PPR is enabled, now it will be using the new curve fit and limits, as shown in Figure 42.

The new PPR calibration can be made permanent by selecting *Flash parameters in MCA* from the MCA Settings menu before disconnecting the MCA. Before flashing the parameters, switch the spectrum type back to *Pulse count* so that the MCA will power up with the default spectrum type. Remember that all of the MCA settings are made permanent by the Flash command.

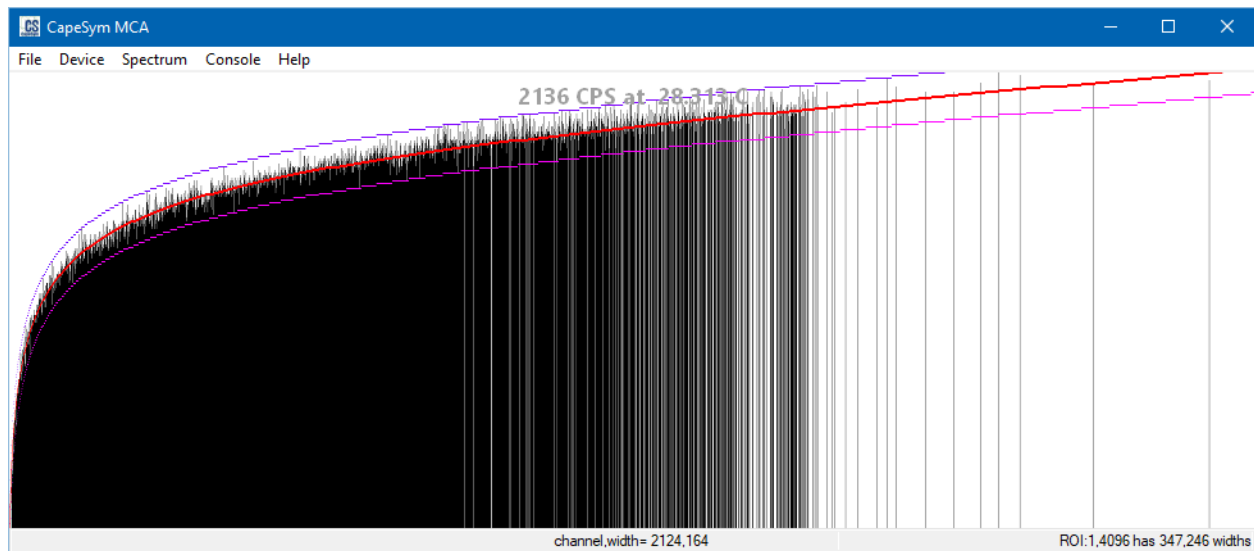


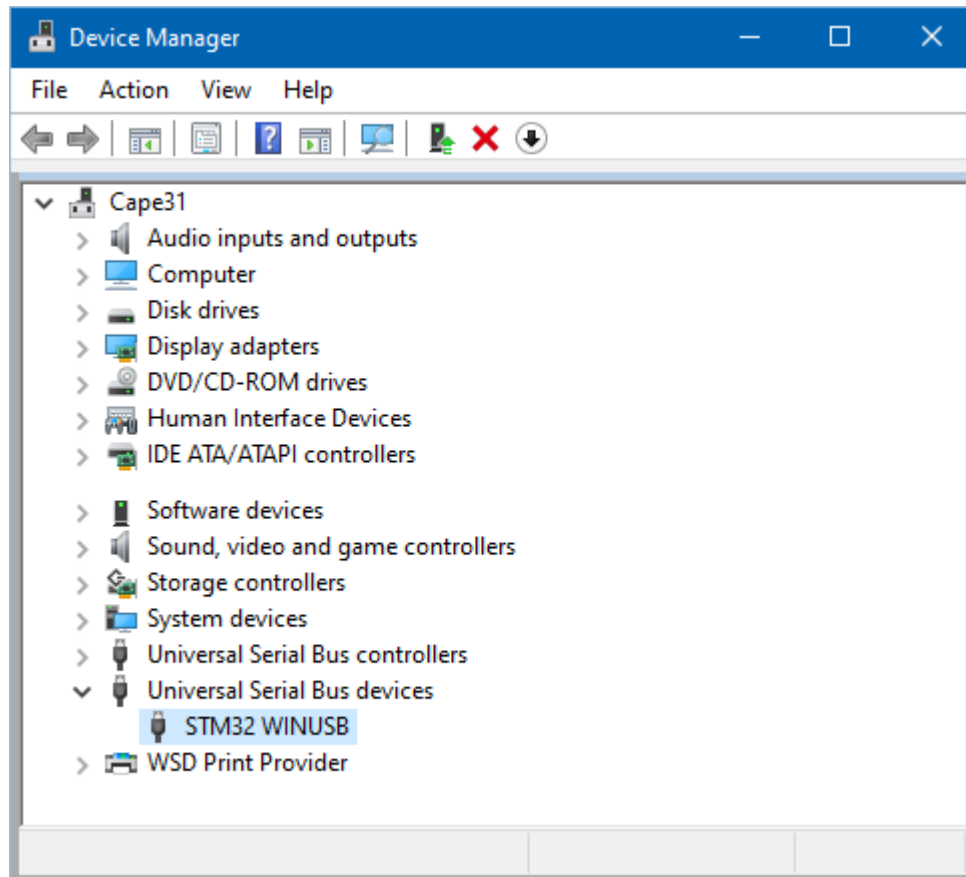
Figure 42: Testing the new pulse width curve and limits at high count rate after sending the parameter changes to the MCA.

PPR Calibration Procedure

1. Connect to the MCA and record a pulse width spectrum at low count rate
2. Stop the MCA but leave it connected
3. Check existing calibration (Calibrations > *Pulse pileup rejection* > *Plot pulse width limits*)
Only modify the existing calibration if a significant improvement is obtained in the next step
4. Recalculate curve fit (Calibrations > *Pulse pileup rejection* > *Recalculate from spectrum*)
5. Adjust offsets, if necessary (Calibrations > *Pulse pileup rejection* > *Enter limit offsets...*)
6. Send new parameters to MCA (Stop mode commands > *Update MCA parameters...*)
7. Restart the MCA without disconnecting it and test PPR
8. Switch Spectrum data type back to *Pulse count* and then *Memorize MCA parameters*

Appendix A: USB Device Interface

The Windows desktop application reads from and writes to the WINUSB device driver using the WinUsb API. The driver should automatically appear in the Device Manager when the CapeSym MCA hardware is plugged into any USB port under Windows 8 or higher. The STM32 WINUSB device interface may be identified with vendor identification number (VID) of 0x4701 in hexadecimal, and product identification number (PID) of 0x0290. To allow multiple MCAs to connect to one computer, each MCA must include a unique serial number string as part of the device descriptors, such as "capemca00032".



For Windows 7 and lower, the MCA will not be automatically identified as a USB device that uses the WinUSB driver. Instead, the WinUSB driver must be associated with the device's VID, PID and DeviceInterfaceGUID of {13EB360B-BC1E-46CB-AC8B-EF3DA47B4062} through a .inf file. A simple .inf file for making this association is listed in the following code box. Copy this code and paste it into a text file (e.g. using Notepad) and then save the file to STM_WINUSB.inf in an easily accessible folder such as C:\Temp. Open the Device Manager from the Control Panel, and find the broken STM WINUSB device. Use a right mouse click on the device to select *Update Driver Software...* and then choose *Browse my computer for drivers*. Use the Browse button to locate the folder containing STM_WINUSB.inf and then click the Next button. If a message appears saying that the driver is not signed, just select *Install this driver software anyway* to continue. The progress bar for *Installing driver software...* should display for a minute or two and then report a successful install. Now the MCA device should be properly recognized for connecting from the Device menu of the CapeMCA program.

```
;
; .inf file for associating WinUsb driver with CapeMCA USB devices
;
[Version]
Signature = "$Windows NT$"
Class      = USBDevice
ClassGUID  = {88BAE032-5A81-49f0-BC3D-A4FF138216D6}
Provider   = %ManufacturerName%
CatalogFile = WinUSBInstallation.cat
DriverVer=09/04/2012,13.54.20.543

; ===== Manufacturer/Models sections =====

[Manufacturer]
%ManufacturerName% = Standard,NTamd64

[Standard.NTamd64]
%DeviceName% =USB_Install, USB\VID_4701&PID_0290

; ===== Class definition (for Windows 8 and ealier versions)=====

[ClassInstall32]
AddReg = ClassInstall_AddReg

[ClassInstall_AddReg]
HKR,,, %ClassName%
HKR,,NoInstallClass,,1
HKR,,IconPath,%REG_MULTI_SZ%, "%systemroot%\system32\setupapi.dll,-20"
HKR,,LowerLogoVersion,,5.2

; ===== Installation =====

[USB_Install]
Include = winusb.inf
Needs   = WINUSB.NT

[USB_Install.Services]
Include =winusb.inf
Needs   = WINUSB.NT.Services

[USB_Install.HW]
AddReg=Dev_AddReg

[USB_Install.Wdf]
KmdfService=WINUSB, WinUsb_Install

[WinUsb_Install]
KmdfLibraryVersion=1.11

[Dev_AddReg]
HKR,,DeviceInterfaceGUIDs,0x10000,"{13EB360B-BC1E-46CB-AC8B-EF3DA47B4062}"

[Strings]
ManufacturerName=""
ClassName="Universal Serial Bus devices"
DeviceName="STM32 WINUSB"
REG_MULTI_SZ = 0x00010000
```


In Run mode, the MCA device will respond to only one USB command from the host computer at each communication interval, typically once per second. Every command is two bytes long, and only the first two bytes received are retained for processing. Additional two-byte commands received while the MCA is processing pulses will be discarded. For maximum compatibility, the host computer should send only one command per communication interval.

Table 8: USB Commands in Run Mode

Byte 1	Byte 2	Command	Reply Bytes
0	0	return packet0 data structure	64
0	1	return 256 x 32-bit integer spectrum	1024
0	2	return 512 x 32-bit integer spectrum	2048
0	4	return 1024 x 32-bit integer spectrum	4096
0	8	return 2048 x 32-bit integer spectrum	8192
0	16	return 4096 x 32-bit integer spectrum (default)	16384
0	32	reserved	64
0	33	return 256 x 32-bit integer spectrum + packet0	1088
0	34	return 512 x 32-bit integer spectrum + packet0	2112
0	36	return 1024 x 32-bit integer spectrum + packet0	4160
0	40	return 2048 x 32-bit integer spectrum + packet0	8256
0	48	return 4096 x 32-bit integer spectrum + packet0	16448
1	1	zero out the spectrum inside the MCA	2
2	2	stop MCA loop, go to Stop mode command processing	2
3	3	return current MCA parameters array	512
4-32	v	set parameter with byte1 index to byte2 value v	2
17	d	increment DAC by d = [-100,100]	2

When the host software sends the two-byte command [0,16] to the MCA, the MCA replies with an energy spectrum of 4096 32-bit integers, equivalent to 16,384 bytes. C code for this exchange looks like this:

```
BYTE spccmd[2] = { 0, 16 };           // cmd to request 4096x32-bit spectrum
BYTE spectrumBytes[16384];
ULONG cbWritten, cbRead;
UINT32 spectrum[4096];

if ( WinUsb_WritePipe(handle, pipout, spccmd, 2, &cbWritten, 0) )
    if ( WinUsb_ReadPipe(handle, pipin, spectrumBytes, 16384, &cbRead, 0) )
        memcpy(spectrum, spectrumBytes, 16384);
```

The returned spectrum will contain the following information:

```
spectrum[0]: upper 16-bit word is number of detection events in prior interval
              lower word is 16 times the temperature of SiPMs in degrees C
spectrum[1]: accumulated count of events in channel 1
spectrum[2]: accumulated count of events in channel 2
:
spectrum[4095]: accumulated count of events in the last channel
```

The 32-bit content of spectrum[0] is determined by the feedback type parameter, stored in parameter[14] as shown in Appendix C. For the default feedback type of 0, the counts during the prior acquisition interval (CPI) is stored in the most significant 16 bits and temperature in the least significant 16 bits. For feedback type 1, the CPI in the upper 16 bits will be replaced by the current DAC level. The DAC feedback is only used during the calibration of temperature stabilization.

Table 9: Content of spectrum[0] based on feedback type.

Type	Description of Feedback Type	0000000000000000 0000000000000000
0	Count prior interval and temperature of SiPMs	CPI $\pm^{\circ}\text{C} \times 16$
1	High voltage bias control DAC and temperature of SiPMs	DAC $\pm^{\circ}\text{C} \times 16$
2	Count prior interval and baseline of SiPMs	CPI baseline ADC
3	Count prior interval and count in capture box	CPI n[]

A code fragment for parsing spectrum[0] for the different feedback types is given here:

```
switch ( feedbackType )      // parameter[14]
{
  case 0:                    // CPI and temperature
    cpi = (spectrum[0]>>16)&0x0000ffff;      // cpi in upper word of spectrum 0
    temp16 = (__int16)(spectrum[0]&0x0000ffff); // 16x temperature in lower word
    t = (float)t/16.0f;                      // convert to Celsius
    break;
  case 1:                    // DAC and temperature
    dac = (spectrum[0]>>16)&0x0000ffff;      // dac feedback in upper word
    temp16 = (__int16)(spectrum[0]&0x0000ffff); // 16x temperature in lower word
    t = (float)t/16.0f;                      // convert to Celsius
    break;
  case 2:                    // CPI and baseline
    cpi = (spectrum[0]>>16)&0x0000ffff;      // cpi in upper word of spectrum 0
    base = (spectrum[0]&0x0000ffff);          // ADC level in lower word
    mV = (float)base*3300.0f/4095.0f;        // convert to millivolts
    break;
  case 3:                    // CPI and n[]
    cpi = (spectrum[0]>>16)&0x0000ffff;      // cpi in upper word of spectrum 0
    npi = (spectrum[0]&0x0000ffff);          // count in box in lower word
    break;
}
```

Packet0 is a shortest data packet, containing overall counting statistics without any energy spectrum. As shown in the next code fragment, packet0 is a data structure with just sixteen 32-bit fields. The code block is the packet0 structure as defined in the firmware. The first six fields pertain to the hosted MCA which is communicating with the host software via USB. The next six fields pertain to all of the MCAs in the array, which includes the hosted MCA and all of the other detectors that are communicating with the hosted MCA via a communication path other than USB. The last four fields are reserved for future use.

The cps field is a 32-bit floating point number that gives the counts per second rate for the most recent communication interval. The totalCount reports the total number of radiation pulses counted since the last reset or zeroing of the MCA. Both cps and totalCount reflect all pulses detected by an above threshold transition from baseline that is at least 16 samples in length. Many of these counted pulses are not included in the energy spectrum, either because the integral of the pulse was outside the valid channel range of [1,4095] or because they were excluded by pulse pileup rejection when it is enabled.

The next three fields provide information on the absolute amount of live and dead time during counting. The usPerInterval field is the elapsed time between onset and offset of sampling during the most recent communication interval, while the totalIntervals field tallies the number of communication intervals since the last reset or zeroing of the MCA. If the communication interval is not changed, then the product of these two fields gives the total acquisition time.

$$\text{acquisition time in seconds} = (\text{usPerInterval} \times \text{totalIntervals}) / 1000000$$

The totalPulseTime field reports the total amount of time in seconds that that sampled signal was above threshold in a counted pulse since the last reset or zeroing of the MCA. This is the only dead time during the acquisition time when additional pulses will not be counted. The totalPulseTime is estimated by dividing the number of above-threshold samples by the specified sampling rate of the MCA. The live time since the last reset or zeroing of the MCA is calculated as

$$\text{live time in seconds} = \text{acquisition time in seconds} - \text{totalPulseTime}$$

The capemcaId field reports the MCA serial number as reported in the USB interface. The detectors field is the number of MCAs that are actively communicating in the array. When configured as an array of detectors, the hosted MCA will report the count per communication interval summed over all detectors in the cpiArray field, and report the total count from all spectra in the array for the [minChannel,maxChannel] range in the countInRangeArray field. The latter measurement includes only pulses that are included in spectra, the former includes all pulses detected. The source direction may also be provided in packet0 when the hosted MCA is able to calculate that information for the array.

```
typedef struct      // to be returned in response to cmd[0,0]
{
    float cps;      // 32-bit count rate of most recent acquisition interval
    float totalCount; // sum of all above threshold pulses since reset
    float totalPulseTime; // total time in seconds inside pulses
    uint32_t usPerInterval; // duration of most recent interval in microseconds
    uint32_t totalIntervals; // total number of intervals used to acquire data
    uint32_t capemcaId; // the USB id number defined in usbd_desc.c
    uint32_t detectors; // number of detectors in array
    uint32_t cpiArray; // count in all detectors for most recent interval
    uint32_t countInRangeArray; // count in channel range across all detectors
    float xDirection; // source direction based on counts in range vector
    float yDirection;
    float zDirection;
    uint32_t reserved[4];
} PACKET0_TYPE; // sizeof(PACKET0_TYPE) = 64 bytes
```

The packet0 structure is copied byte-by-byte into the USB communication buffer, and should be deciphered the same way on the host side, as shown in the following code fragment. When the host request is for both spectrum and packet0, the bytes are strung together in one contiguous buffer with the spectrum first and packet0 coming last. Use the expected byte-length of the spectrum to calculate where in the buffer the packet0 starts when moving the 64 bytes to the local structure representation.

```
BYTE cmd[2] = { 0, 48 }; // cmd to read 4096 spectrum + packet0
UINT32 spectrum[4096];
PACKET0_TYPE packet0;
BYTE allBytes[4096*4+64];
ULONG cbWritten, cbRead;
int bytesInSpectrum = 4096*4, bytesInPacket = 64;
int bytesToRead = bytesInSpectrum + bytesInPacket;

if (WinUsb_WritePipe(winusbHandle, pipout, cmd, 2, &cbWritten, 0) )
    if ( WinUsb_ReadPipe(winusbHandle, pipin, allBytes, bytesToRead, &cbRead, 0) )
    {
        if ( bytesInSpectrum ) // move bytes to local spectrum array
            memcpy(spectrum, allBytes, bytesInSpectrum);

        if ( bytesInPacket ) // move packet0 bytes to local struct
            memcpy(&packet0, allBytes+bytesInSpectrum, bytesInPacket);
    }
```

When the [1,1] command is received the MCA sets the count in every channel to zero and then resumes counting pulses. The [1,1] command has a two-byte reply [1,1] which simply echoes the command. Commands with two-byte replies allow the reply to be compared to the command to validate transmission and reception. C code for commands with two-byte replies is shown in the following code box.

```

BYTE cmd[2] = { 1, 1 };           // cmd to zero out the spectrum
BYTE rply[2];
ULONG cbWritten, cbRead;

if ( WinUsb_WritePipe(handle, pipout, cmd, 2, &cbWritten, 0)
    if ( WinUsb_ReadPipe(handle, pipin, rply, 2, &cbRead, 0) )
        if ( (cbRead != 2) || (cmd[0] != rply[0]) || (cmd[1] != rply[1]) )
            printf("WARNING: USB message was garbled in Tx or Rx\n");

```

When the host software sends the two-byte command [3,3] to the MCA, the MCA replies with the array of 32-bit integers containing the 128 parameters, for 4*128=512 bytes in total. The returned parameter array will contain the information described in Appendix C. The host code for this exchange looks like this:

```

BYTE cmd[2] = { 3, 3 };           // cmd to request MCA parameters
BYTE paramBytes[512];
ULONG cbWritten, cbRead;
__int32 parameters[128];

if ( WinUsb_WritePipe(handle, pipout, cmd, 2, &cbWritten, 0) )
    if ( WinUsb_ReadPipe(handle, pipin, paramBytes, 512, &cbRead, 0) )
        memcpy(parameters, paramBytes, 512);

```

Two-byte command [17,d] increments the DAC level in the MCA by:

$$DAC = DAC + d;$$

where d is a signed integer between -100 and 100. The reply to this command is [17,d].

Two-byte command [i,v], where i is less than 32 and not equal to 0, 1, 2, 3, or 17, executes the following statement in the MCA.

$$parameters[i] = v;$$

As with all two-byte replies, the command is echoed in response.

When the [2,2] command is received, the MCA loop stops running and the device listens for parameter updates of exactly 512 bytes. The parameter vector is an array of 128 32-bit signed integers which define the operating mode and calibrations of the device. The parameter vector is retained in non-volatile memory while the MCA is powered off. Parameters are loaded into RAM at power up. The RAM values may be updated by the 2-byte host commands (listed above) or by writing the entire parameter vector from the host while the MCA loop is stopped. Additional actions are possible when the first integer (4 bytes) of the parameters message contains one of the following sequences.

Note that the fourth byte is the most significant byte of a 32-bit integer when the host transmission is from an Intel-based Windows PC. The MCA (ARM Cortex M7 processor) receives the individual bytes of the 32-bit integer in little endian order with the least significant byte first.

Table 10: Special First Parameter in Stop Mode

Byte 1	Byte 2	Byte 3	Byte 4	Action
175	2	2	2	Read parameters from non-volatile memory
175	4	4	4	Store RAM parameters into non-volatile memory
175	7	7	7	Jump to system bootloader
175	10	10	10	Soft reset without power down
175	21	21	21	Restore factory settings from flash memory

<< All commands in Stop mode must be 512 bytes in length >>

Only parameters arrays which are exactly 512 bytes in length are processed by the MCA in Stop mode, even when the first parameter contains a special command. The special command will not be processed until all 512 bytes have been received. All 512-byte commands which do not contain a special first parameter are considered to be valid parameter arrays and are copied to the MCA parameters in RAM. The host software code fragment for writing to the MCA in Stop mode is simply this:

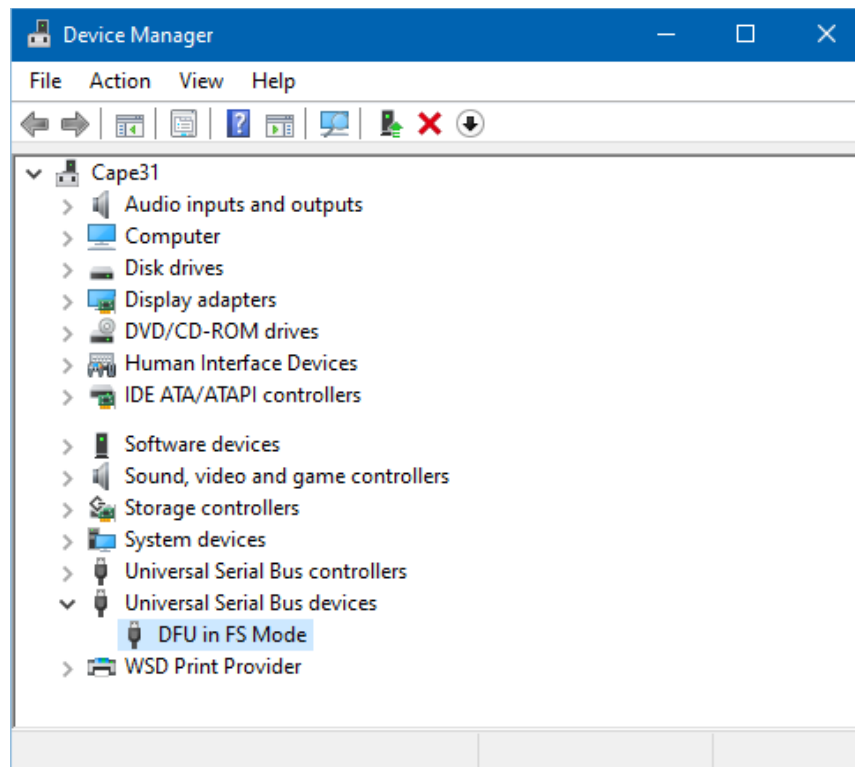
```
int parameters[128];
ULONG cbWritten;

WinUsb_WritePipe(handle, pipout, (PUCHAR)parameters, 512, &cbWritten, 0);
```

<< There is no reply to host input in Stop mode >>

When any two-byte command is received in Stop mode, the MCA run loop is immediately restarted so the command can be processed at the next communication interval. The MCA remains in Run mode accumulating a spectrum and processing two-byte commands from the host, until the host issues a two-byte command to return to Stop mode.

In Stop mode, a soft reset of the MCA can be generated by setting the first 32-bit parameter to the four-byte sequence [175,10,10,10]. This will return the MCA to its initialization state without requiring that the device be disconnected from power. However, if the USB connection has become corrupted, a power-on reset may be the only way to recover normal operation. When the bootloader is invoked, the MCA resets to Device Firmware Update (DFU) programming mode, in which an external programmer (such as the STM32 Cube Programmer) may be used to erase and rewrite the MCA firmware.



In DFU mode, the WinUSB driver is unloaded from the Device Manager and a new driver appears, therefore it is no longer possible to restart the MCA loop without a power-on reset. Power-on reset requires that the USB cable be disconnected.

Note: If the RDP level was elevated after factory programming, the entire MCA flash memory will be erased by any attempt to connect to it with the STM32 Cube Programmer. At which point, the MCA will no longer work without completely reprogramming it, if that is even possible.

Appendix B: File Formats

As noted above, the default format expected when using Auto Save and Auto Load is the comma separated values format. These files may or may not contain a channel zero, but if present channel zero will have a floating point value representing the temperature in Celsius. When saving a spectrum directly using the *Save Spectrum As* dialog, there will be no channel zero. An example is displayed in the next box.

```
1,0
2,0
3,4
4,8
5,5
6,3
7,7
8,3
9,5
10,2
11,7
:
4090,0
4091,0
4092,0
4093,0
4094,0
4095,0
```

SPE Spectrum File:

```
$SPEC_ID:
C:\Temp\spectrum.spe
$SPEC_REM:
$DATA:
1 4095
    0
    0
    4
    8
    5
    3
    7
    3
    5
    2
    7
    :
    0
    0
    0
    0
    0
    0
    0
```

SPE format files are also ASCII text files with fields before the spectrum data delimited by fixed keywords beginning with \$ in column 1.

CHN Spectrum File:

CHN format files normally contain a sequence of binary integers intermixed with ASCII characters. This is simplified here to a header sequence of 16 two-byte binary integers followed by the spectrum stored as a sequence of binary 32-bit integers. There are no line terminations, but the example here uses separate lines for clarity of the individual values.

```
-1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
4095
0
0
4
8
5
3
7
3
5
2
7
:
0
0
0
0
0
0
0
```

N42 Spectrum File:

The N42 file format is a minimal XML document adhering to the ANSI N42.42 schema, currently available online at "<https://www.nist.gov/document/n42xsd>". The schema is too big to reproduce here but a local copy is provided by the capeMCA installer. The file format for representing spectra AND packet0 data is defined by extending the N42.42 standard. These extensions are defined in a separate schema capemca.xsd, listed below. In order to validate .n42 files against the extended standard, both .xsd files have to be stored in the same folder as the .n42 file.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="capemca"
  xmlns:n42="http://physics.nist.gov/N42/2011/N42"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:import namespace="http://physics.nist.gov/N42/2011/N42"
schemaLocation="n42.xsd"/>

  <xsd:element name="DACValue" type="xsd:nonNegativeInteger"
substitutionGroup="n42:SpectrumExtension"></xsd:element>
  <xsd:element name="TemperatureCelsius" type="xsd:double"
substitutionGroup="n42:SpectrumExtension"></xsd:element>

  <xsd:element name="countsPerSecond" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalCount" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalPulseTime" type="xsd:duration"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="usPerInterval" type="xsd:nonNegativeInteger"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalIntervals" type="xsd:nonNegativeInteger"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="capemcaId" type="xsd:integer"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="detectors" type="xsd:positiveInteger"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="cpiArray" type="xsd:nonNegativeInteger"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="countInRangeArray" type="xsd:nonNegativeInteger"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="xDirection" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="yDirection" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="zDirection" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="cpsArray" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalCountArray" type="xsd:double"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalPulseTimeArray" type="xsd:duration"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
  <xsd:element name="totalLiveTimeArray" type="xsd:duration"
substitutionGroup="n42:RadMeasurementExtension"></xsd:element>
</xsd:schema>

```

A sample .n42 file recorded by the CapeMCA application is listed below. Note that when the actual real time duration or live time is not known (because this information was not provided by the user), values of zero will appear in these elements. The elements that begin with "<capemca:" are extensions provided by the above schema. They are necessary for temperature and packet0 fields because the conventional standard has no equivalent definitions. The packet0 extensions will only appear when the *Request # of channels* includes the packet0 data.

```

<?xml version="1.0"?>
<RadInstrumentData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://physics.nist.gov/N42/2011/N42" xmlns:capemca="capemca"
  xsi:schemaLocation="http://physics.nist.gov/N42/2011/N42 n42.xsd capemca
  capemca.xsd"
>
  <RadInstrumentInformation id="MCA1">
    <RadInstrumentManufacturerName>CapeSym Inc</RadInstrumentManufacturerName>
    <RadInstrumentModelName>CapeSym MCA</RadInstrumentModelName>
    <RadInstrumentClassCode>Other</RadInstrumentClassCode>
    <RadInstrumentVersion>
      <RadInstrumentComponentName>Firmware</RadInstrumentComponentName>
      <RadInstrumentComponentVersion>1.3.2</RadInstrumentComponentVersion>
    </RadInstrumentVersion>
  </RadInstrumentInformation>
  <RadDetectorInformation id="Detector">
    <RadDetectorCategoryCode>Gamma</RadDetectorCategoryCode>
    <RadDetectorKindCode>Other</RadDetectorKindCode>
  </RadDetectorInformation>
  <EnergyCalibration id="EC0">
    <CoefficientValues>0.0 0.005 0.0</CoefficientValues>
    <Remark>y = quadratic(x) where x is the integral of pulse</Remark>
  </EnergyCalibration>

  <RadMeasurement id="M1">
    <MeasurementClassCode>Foreground</MeasurementClassCode>
    <StartDateTime>2023-10-30T15:09:46</StartDateTime>
    <RealTimeDuration>PT90.0S</RealTimeDuration>
    <Spectrum id="S1" energyCalibrationReference="EC0">
      <LiveTimeDuration>PT88.956996S</LiveTimeDuration>
      <ChannelData compressionCode="None">
        0
        :
        0
      </ChannelData>
      <capemca:TemperatureCelsius>25.875</capemca:TemperatureCelsius>
    </Spectrum>
    <capemca:countsPerSecond>21.0035</capemca:countsPerSecond>
    <capemca:totalCount>2143</capemca:totalCount>
    <capemca:totalPulseTime>PT0.0267985S</capemca:totalPulseTime>
    <capemca:usPerInterval>999835</capemca:usPerInterval>
    <capemca:totalIntervals>89</capemca:totalIntervals>
    <capemca:capemcaId>219</capemca:capemcaId>
    <capemca:detectors>1</capemca:detectors>
    <capemca:countInRangeArray>0</capemca:countInRangeArray>
    <capemca:xDirection>0</capemca:xDirection>
    <capemca:yDirection>0</capemca:yDirection>
    <capemca:zDirection>0</capemca:zDirection>
    <cpsArray>0</cpsArray>
    <totalCountArray>0</totalCountArray>
    <totalPulseTimeArray>0</totalPulseTimeArray>
    <totalLiveTimeArray>0</totalLiveTimeArray>
  </RadMeasurement>
</RadInstrumentData>

```

Appendix C: MCA Parameters

The MCA contains 128 parameters stored as 32-bit signed integers. Optimal parameters are different for each unit (MCA electronics and detector). The parameter table for particular sensor unit is available in the MCA Parameters dialog, or from the console menu. Older firmware versions may use a slightly different set of parameters than shown here, because of changes to the feedback options or the energy correction algorithm.

Index	Value	Description
0	1	Major version
1	1	Minor version
2	11	Release version
3	12	Divisor for integral to channel [1,4095]
4	0	Pulse pileup rejection on/off
5	1	Temperature stabilization (TS) on/off
6	0	Spectrum type: 0=count/chan, 1=width/chan, 2=ADC/sample, 3=pulse list
7	1	Energy correction (EC) on/off
8	1	Detector index in array
9	0	Moving spectrum off/moving/shared
10	32	Depth of moving spectra [0,32] or [0,20] or [0,16]
11	10	Communication interval [1,100]*100ms
12	4	Pulse threshold in ADC levels (1.24 per mV)
13	2035	DAC [0,4095] determines high voltage when TS is off at reset
14	0	Feedback type: 0=cpi+temp, 1=DAC+temp, 2=cpi+baseline, 3=cpi+n[]
15	10	Min. channel of range-limited operations, e.g. sample buffer
16	4095	Max. channel of range-limited operations, e.g. source direction
17	0	One-time bump increment [-100,100] to DAC output
18	2000	Capture box [] left channel
19	3500	Capture box [] right channel
20	20	Capture box [] top pulse width deficit
21	80	Capture box [] bottom pulse width deficit
22	0	Reserved for future use
23	0	Reserved for future use
24	0	Reserved for future use
25	0	Reserved for future use
26	0	Reserved for future use
27	0	Reserved for future use
28	0	Energy correction curve fit coefficient 1
29	0	Energy correction curve fit coefficient 2
30	0	Energy correction curve fit coefficient 3
31	0	Energy correction curve fit coefficient 4
32	-402895	Pulse width curve fit coefficient 1
33	61	Pulse width curve fit coefficient 2
34	-6586	Pulse width curve fit coefficient 3
35	281313	Pulse width curve fit coefficient 4
36	10	Offset from curve to pulse width lower limit

37	10	Offset from curve to pulse width upper limit
38	4	Number of temperature stabilization lines (6 max.)
39	92	Line 1: 16*temperature at start of line segment
40	1220	Line 1: 100*slope of line
41	2846190	Line 1: 1600*y-intercept of line
42	336	Line 2: 16*temperature at start of line segment
43	1700	Line 2: 100*slope of line
44	2684800	Line 2: 1600*y-intercept of line
45	416	Line 3: 16*temperature at start of line segment
46	1667	Line 3: 100*slope of line
47	2698667	Line 3: 1600*y-intercept of line
48	560	Line 4: 16*temperature at start of line segment
49	1889	Line 4: 100*slope of line
50	2574222	Line 4: 1600*y-intercept of line
51	-1	Line 5: 16*temperature at start of line segment
52	-1	Line 5: 100*slope of line
53	-1	Line 5: 1600*y-intercept of line
54	-1	Line 6: 16*temperature at start of line segment
55	-1	Line 6: 100*slope of line
56	-1	Line 6: 1600*y-intercept of line
57	0	Number of calibrated energy integrals (35 max.)
58	-1	Energy 1
59	-1	Integral or channel
60	-1	Energy 2
61	-1	Integral or channel
62	-1	Energy 3
63	-1	Integral or channel
64	-1	Energy 4
65	-1	Integral or channel
66	-1	Energy 5
67	-1	Integral or channel
68	-1	Energy 6
69	-1	Integral or channel
70	-1	Energy 7
71	-1	Integral or channel
72	-1	Energy 8
73	-1	Integral or channel
74	-1	Energy 9
75	-1	Integral or channel
76	-1	Energy 10
77	-1	Integral or channel
78	-1	Energy 11
79	-1	Integral or channel
80	-1	Energy 12
81	-1	Integral or channel
82	-1	Energy 13
83	-1	Integral or channel
84	-1	Energy 14

85	-1	Integral or channel
86	-1	Energy 15
87	-1	Integral or channel
88	-1	Energy 16
89	-1	Integral or channel
90	-1	Energy 17
91	-1	Integral or channel
92	-1	Energy 18
93	-1	Integral or channel
94	-1	Energy 19
95	-1	Integral or channel
96	-1	Energy 20
97	-1	Integral or channel
98	-1	Energy 21
99	-1	Integral or channel
100	-1	Energy 22
101	-1	Integral or channel
102	-1	Energy 23
103	-1	Integral or channel
104	-1	Energy 24
105	-1	Integral or channel
106	-1	Energy 25
107	-1	Integral or channel
108	-1	Energy 26
109	-1	Integral or channel
110	-1	Energy 27
111	-1	Integral or channel
112	-1	Energy 28
113	-1	Integral or channel
114	-1	Energy 29
115	-1	Integral or channel
116	-1	Energy 30
117	-1	Integral or channel
118	-1	Energy 31
119	-1	Integral or channel
120	-1	Energy 32
121	-1	Integral or channel
122	-1	Reserved for future use
123	-1	Reserved for future use
124	115200	Serial UART interface baud rate in Hz
125	5555556	Hardware sample rate in Hz
126	16793088	Hardware version number: e.g. 0x01.00.4B.00
127	0	Energy correction table type (0 => quadratic fit)

Appendix D: Host Code Examples

Source code for a command line program to read from the MCA using the WinUSB API in Visual C++ is provided by the capeMCA Windows installer. The code includes a C++ wrapper for the WinUSB device interface and a build file for the Visual C++ nmake utility. The application can be built from the Windows command prompt using

```
> nmake /f capeMCAcli.mak
```

and run by

```
> capeMCAcli
```

```
Enumerating MCAs... found 1 MCAs
```

```
Connecting to MCAs
```

```
1: capemca00060 : connected
```

```
Requesting spectra from MCAs
```

```
Reading spectra from MCAs
```

```
Spectrum 1:
```

```
channel,count
```

```
1,0
```

```
2,0
```

```
3,0
```

```
4,92
```

```
5,915
```

```
6,1625
```

```
:
```

```
4093,0
```

```
4094,0
```

```
4095,0
```

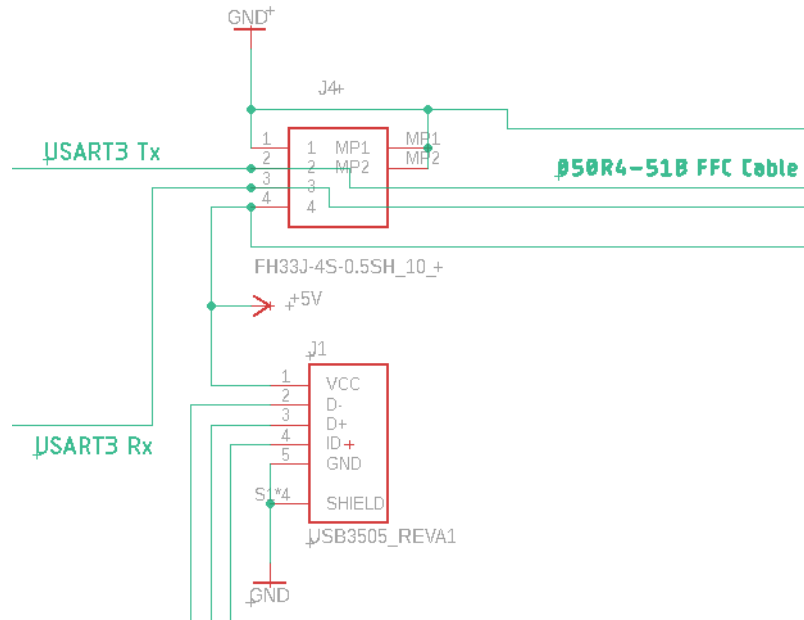
```
Done.
```

The program will search for and connect to any MCAs that have been plugged into the computer's USB ports. Then the program will read the spectrum from each device, sum the spectra together, and output the total count for each channel, as show in the above output.

The USB interface to the MCA can alternatively be accessed by using the libusb library. This would be a convenient way to communicate with the MCA from a Linux computer, and example code is also provided to demonstrate how to access the MCA from the Linux operating system.

Appendix E: Serial UART Interface

Version 1.0.62 of the MCA hardware includes an FFC connector to support a serial interface in addition to the USB bulk transfer device interface. The USB interface with microB connector allows a host computer to configure the MCA operating parameters. When no USB data lines are detected, the alternative serial interface will become active, over which spectral data may be requested and sent at every communication interval. The serial interface will transmit the same data packet that is normally sent in response to the two-byte request {0,n} via the USB interface.



The pinout of the FFC connector (0.5 mm pitch) is diagrammed in the above figure. The serial interface is provided by the USART3 communication peripheral of the STM32H743 microcontroller operating in the default full-duplex asynchronous mode at 3.3V with 1 stop bit, 1 start bit, 8 data bit, and no parity. Baud rates of up to 2.88Mbit per second are selectable through the baud rate parameter (index=124). The baud rate parameter may be saved to non-volatile memory from a host computer using the USB interface.

Power to the MCA can be provided by either the USB microB connector or on the 4-pin FFC connector (part: FH33J-4S-0.5SH(10) Hirose Electric). The v1.0.62 MCA with UART interface expects 5V input, but it has its own 3.3V regulator (MCP1755S) so any input voltage between 3.6 and 5V will work.

The serial UART interface listens for data requests from the host computer on the Rx line, and processes 1 command per communication interval. The two-byte command and variable length response format is the same as employed by the USB interface, as detailed in the following table. Note that Stop mode is not accessible via the serial UART interface, so any calibrations and parameter settings will need to be done through the USB interface first.

Byte 1	Byte 2	Command	Reply Bytes
0	0	return packet0 data structure	64
0	1	return 256 x 32-bit integer spectrum	1024
0	2	return 512 x 32-bit integer spectrum	2048
0	4	return 1024 x 32-bit integer spectrum	4096
0	8	return 2048 x 32-bit integer spectrum	8192
0	16	return 4096 x 32-bit integer spectrum (default)	16384
0	32	reserved	64
0	33	return 256 x 32-bit integer spectrum + packet0	1088
0	34	return 512 x 32-bit integer spectrum + packet0	2112
0	36	return 1024 x 32-bit integer spectrum + packet0	4160
0	40	return 2048 x 32-bit integer spectrum + packet0	8256
0	48	return 4096 x 32-bit integer spectrum + packet0	16448
1	1	zero out the spectrum inside the MCA	2
4-32	v	set parameter with byte1 index to byte2 value v	2
17	d	increment DAC by d = [-100,100]	2

Source code for an example command line program to read from the MCA using the Win32 serial port API in Visual C++ is provided by the capeMCA Windows installer. The code includes a build file for the Visual C++ nmake utility. The application can be built from the Windows command prompt using

```
> nmake /f capeMCAuart.mak
```

and run by

```
>capeMCAuart -?
CapeMCA Uart Interface
```

```
Usage: CapeMCAuart [flags]
```

Flags:

```
-b=115200 : use baud rate 115200 bit/s (default)
-p=COM1 : use COM1 for serial port (default)
-q=8 : request type {0,1,2,4,8,16,32+1,32+2,32+4,32+8,32+16}
-h : display this help message
-v : print version info
-z : zero spectrum before request
```

Read energy spectrum from 1 macropixels via COM port.
Spectral output is streamed to the console.