# ManoScript manual

*i2PEPICO scripting language | Version 3.41 | 20 October 2026*

Use ManoScript to acquire measurements, process closed data files, calculate coincidence results, and export analysis. This manual has four parts: a quick start with small examples, the language rules (with a one-page list of rules and pitfalls), the reference of all native commands and constants, and the Import libraries shipped with the program, with complete examples. Check experimental settings before using any example with hardware.

**Using this manual with an AI assistant.** Give the assistant the whole manual, not excerpts, and ask it to follow "Rules and pitfalls" and to use only commands from the Command reference. ManoScript reports few errors: a misspelled command or variable is usually accepted silently as text. Read a generated script line by line before running it, and test it on closed data files first.

### New in 3.41

The main changes of earlier versions are listed in Version history at the end of this manual.

- Export2D and Export2DSlices can count only the ions that hit a region of the ion image: set UseIonROI := 1 and IonROI before the call (see Export2D: export for PIMP). All entry points support it, including time, window and trigger slicing and Export2DMeas. Without UseIonROI, exports are unchanged.
- An export with an ion ROI records it in an IonROI: line of the index file; PIMP ignores that line.
- The shipped Imports no longer fill result images: they save TOF spectra only, which saves memory. After an Export2D or Export2DSlices call, the result slots hold no images.
- ScriptImport prints its version and the label of its bundled Imports on every run.
- The manual now describes how trigger slicing treats a trigger that arrives before Maximum delay (see Overlapping triggers in Time slicing); the behaviour itself is unchanged.

# Quick start

The following scripts are complete Process scripts for the analysis program. Select a folder with raw .i2d files (and, for examples 2 and 4, one file) in the main window's input controls, paste a script into a new Process script and start it. Lines starting with ! are remarks.

### 1. Write a text file

```
! Writes a small table to C:\Data\Test\squares.txt
Folder := EnsureDirectory[C:\Data\Test]
Output[Folder, squares.txt, 0]
PrintLn[\N, \Square]
For[N, 1, 5]
PrintLn[N, {N*N}]
Next
```

EnsureDirectory creates the folder and returns it with a trailing backslash. Output selects the file; mode 0 replaces an existing file. Print and PrintLn separate values with tabs, and the file is written when the script ends. The file starts with its own full path on the first line. The backslashes in \N and \Square make these words literal text: without them, N would be replaced by the value of the variable N once it exists.

### 2. Integrate a TOF peak in one file

```
CleanCoincCrit
AddCoincCrit[AddEvent[Im1,0,0,ROIAll,TOFStart,Nil],AddEvent[Im2,0,20,ROIAll,TOFStop,Nil]]
Load[CurrInFolder, CurrInFile, 0, 20, 0.004]
Output[CurrInFolder, peak.txt, 0]
PrintLn[\File, \Seconds, \Counts]
PrintLn[CurrInFile, LoadedMeasTime, TOFInt[TOFRes[1], 5.2, 5.6, 0]]
```

The coincidence criterion starts the TOF clock with an electron (Im1, anywhere on the detector) and stops it with an ion (Im2) 0–20 µs later. Load reads the file into a TOF distribution from 0 to 20 µs with 0.004 µs (4 ns) bins. TOFRes[1] is the TOF distribution of the first criterion set, and TOFInt sums it between 5.2 and 5.6 µs without background subtraction. CurrInFolder already ends with a backslash, which Load needs.

### 3. One line per file in a folder

```
Output[CurrInFolder, peaks.txt, 0]
PrintLn[\File, \Energy, \Counts]
CleanCoincCrit
AddCoincCrit[AddEvent[Im1,0,0,ROIAll,TOFStart,Nil],AddEvent[Im2,0,20,ROIAll,TOFStop,Nil]]
CycleFiles[CurrInFolder, *.i2d]
Load[nil, CycFile, 0, 20, 0.004]
PrintLn[StripPath[CycFile], LoadedParam[Start, E], TOFInt[TOFRes[1], 5.2, 5.6, 0]]
Next
```

CycleFiles runs the lines up to Next once for every matching file, in sorted order. CycFile is the current file with its full path, so Load gets nil as its folder. LoadedParam[Start, E] is the photon energy recorded at the start of that measurement.

### 4. A subroutine with parameters

```
CleanCoincCrit
AddCoincCrit[AddEvent[Im1,0,0,ROIAll,TOFStart,Nil],AddEvent[Im2,0,20,ROIAll,TOFStop,Nil]]
Load[CurrInFolder, CurrInFile, 0, 20, 0.004]
Output[CurrInFolder, windows.txt, 0]
Gosub[Window, 5.2, 5.6]
Gosub[Window, 7.0, 7.4]
End

:Window W_From W_To
PrintLn[W_From, W_To, TOFInt[TOFRes[1], W_From, W_To, 0]]
Return
```

The label line :Window W_From W_To declares a subroutine with two parameters. Gosub copies its values into them, runs the subroutine and continues after the Gosub when Return is reached. End stops the main script before it runs into the subroutine. All variables are global, so give parameters distinctive names.

## 5. Use a shipped Import

```
Imports[Export2D]
UseSLS1 := 0
Ring := ROIRing[(0;0),0.03,0.06,1]
Gosub[Export2DScan, *.i2d, CurrInFolder, Concat[CurrInFolder, MyExport], ROIRingAsCircle[Ring],
Ring]
```

Imports on the first line makes the subroutines of the Export2D library available. Export2DScan exports every .i2d file of the input folder into the subfolder MyExport for PIMP. Replace the ring with your calibrated ROI. The shipped Imports are described in Shipped Imports. To count only the ions in a region of the ion image, add UseIonROI := 1 and, for example, IonROI := ROICircle[(0;0),0.4,1] before the Gosub (see Export2D: export for PIMP).

# Script syntax

Each nonempty line contains one command or one remark. A remark begins with an exclamation mark as the first character; a label begins with a colon as the first character. Only the very first character counts: an indented remark is executed as a command. Spaces before a command are allowed; do not indent with tabs, because a command preceded by a tab is not recognized. Command names, variable names, labels, and import names are case insensitive.

Assignment lines have a variable on the left, an assignment operator, :=, and an expression on the right. The variable name can be any non-reserved name. Function names and constants cannot be used. Any line that contains := is treated as an assignment, with everything before the first := as the variable name.

Procedural commands have the function name followed by the function parameters in square brackets. When more than one parameter is given, they are separated by commas. Always write the square brackets, also for commands without parameters when they are followed by nothing else, for example PrintLn[]. Curly brackets, {}, indicate a mathematical expression (see Calculations).

All spaces outside double quotation marks are removed before a line is evaluated, so PrintLn[Hello World] prints HelloWorld. Put literal text in double quotation marks to retain its spaces. The quotation marks are removed before the command is evaluated; they protect spaces only, not commas or brackets, and a quotation mark itself can never be part of a value. A leading backslash on a parameter escapes evaluation: it is removed, and the remaining parameter is returned literally. This is why Output examples use \2D_TPES_index.txt.

**Words, names and literal text.** A word that is not a variable, constant or command evaluates to itself. That is why Concat[CurrInFolder, MyExport] gives a folder name. However, a parameter that is exactly the name of a variable, constant or command is replaced by its value or executed: PrintLn[E] prints 0, because E is the constant for the energy channel, and a parameter End stops the script. Write \E to print the letter. A misspelled command name is not an error either; the line simply does nothing useful.

**Arguments.** Commas separate arguments except inside nested square brackets. Parentheses do not protect commas, which is why points are written (X;Y) with a semicolon. Empty arguments are dropped without a message: Print[A,,B] prints two values. Pass Nil where an argument is to be empty. Text after the closing square bracket is ignored.

**Persistent results.** The program is also capable of loading several files while keeping the results in memory so that TOF distributions and Image results can be manipulated across files. The results are stored in a slot indexed by the last, optional parameter of Load and can be accessed by passing an extra parameter to Results. The TOFRes, Im1Res and Im2Res shorthands can only be used to access the default, index 0 slot. The Loaded* functions may also take an extra parameter specifying which result set is to be accessed.

Save interchange scripts with the .scr extension and CRLF line endings. script.dat is the application's script-tree storage; it is not an individual interchange script.

## Values and variables

All values are text. A value is numeric when it reads as a number with a decimal point (never a decimal comma). A condition is true when it is a nonzero number; zero, any other text and an undefined name are false. All variables are global and are deleted at the start and end of every run, so a run never sees values from the previous one. FileCounter is an ordinary variable that starts at 0.

Arrays are plain variables with the index appended: A[3] is the variable A_3, and A[I, J] is A_<I>_<J> with the values of I and J. An element that was never assigned evaluates to its own name, for example A_3.

Points in (X;Y) notation are taken literally, so variables are not substituted inside them. Use Point[X, Y] or calculated values to build a point from variables.

## Calculations

Text in curly brackets is calculated by a formula parser and replaced by the result before the line runs. Available are + - * / ^, the comparisons < > <= >= <> =, the logical operators & | !, the constant PI and the functions SQR,

SQRT, SIN, COS, TAN, ATAN, SINH, COSH, COTAN, EXP, LN, LOG, LOGN, ABS, SIGN, TRUNC, CEIL, FLOOR, RND, RANDOM, INTPOW, POW, MIN, MAX and MOD. Commands with square brackets inside the curly brackets, for example {2*TOFInt[TOFRes[1],5.2,5.6,0]}, are evaluated first.

Variables are recognized inside curly brackets only when they are separated by a space, + - * / ( ) or a comma. In {N^2} or {N<5}, N is not replaced; write {(N)^2} and {(N)<5}. An unknown X or Y silently counts as 0, and E is the ManoScript constant 0, not Euler's number. The parser's own IF function cannot be used. Results are printed in full floating-point precision. Parser errors, such as "Expression is empty", carry no line number.

Every pair of curly brackets on a line is calculated once, before the command of the line runs. This includes both branches of If and the condition and statement of While. So While[{(N)<5}, …] tests a fixed value; use While[Smaller[N, 5], …], because bracket commands are evaluated again on every pass.

## Labels and subroutines

A label is a line whose first character is a colon. An indented line is not a label. The first word after the colon is the label name. Any further words are the names of the subroutine's parameters, separated by spaces, not commas: :Normalize X Y declares the label Normalize with parameters X and Y. A parameter name starts with a letter or underscore and contains only letters, digits and underscores. It must not be a function name or a constant, and it must not appear twice on one label. Do not put a remark on a label line; put it on the line above. A label name may contain backslashes, for example :Imports\Helpers\Normalize. This is a convenient way to keep names unique, but the program does not interpret the backslashes. Each label name may occur only once in the whole run, including all imported scripts. All labels are checked before the first line runs. The errors are "Duplicate label", "Invalid parameter name", "Duplicate parameter", "Reserved name or constant used as parameter", "Empty label name" and "Comma in label". A subroutine ends with Return. Without Return, execution continues with the following lines, including the next label and the code after it.

Return and Next are the same command. Each acts on the innermost open For loop, CycleFiles loop or subroutine call: it starts the next pass of a loop, or returns from a call. Inside a For or CycleFiles loop within a subroutine, If[C, Return, Nil] therefore continues the loop instead of leaving the subroutine; leave such a loop through its end first. Loops and calls share one stack of 100 levels ("Stack overflow"); a Next or Return with nothing open raises "Stack underflow". Goto out of a loop leaves the loop open on the stack.

## Subroutine parameters

Gosub[Label, Value1, Value2…] passes one value per parameter, in order. All values are evaluated in the calling code first, and only then copied into the parameter variables. Parameters are ordinary variables, and all variables are global. A parameter overwrites any existing variable with the same name, keeps its value after Return, and can be used to return a result. Changing a parameter inside the subroutine does not change the variable that was passed in.

Arguments may calculate values or call ordinary functions, but must not perform control flow. Gosub, Goto, For, CycleFiles, Return, Next and End in a Gosub argument raise "Control flow in Gosub argument". Gosub and Goto check their argument list strictly: commas inside curly or nested square brackets are protected, and "Missing ]", "Text after ]", "Empty argument list", "Trailing comma", "Empty argument" and "Unmatched bracket" are errors.

## Imports

Subroutines can be kept in library scripts. In the script tree, create a folder named Imports directly under the root. This folder may contain only scripts, no subfolders, and the script names in it must be unique. A script uses libraries by naming them on its very first line, for example Imports[Helpers, Math]. The names are script names from the Imports folder, not paths. They must not contain \, /, spaces, commas or quotation marks. The Imports line must be the first line: a blank line or a remark above it switches imports off, and an Imports line anywhere else is an error. Libraries may import other libraries in the same way, and they can also be run on their own. The libraries shipped with the program are described in Shipped Imports.

When the script starts, the program builds a run copy. The first line becomes the remark ! Imports[…] and is followed by the rest of the script. Each library then follows, with the libraries it imports placed before it. Each library is preceded by an End line and a remark ! Import: Name. A library is included only once, even if it is named

several times. The End lines stop the script when execution runs past the end of one script, so it does not continue into the next. Libraries share all variables, loaded results and coincidence criteria with the calling script.

The run copy is fixed at start, so editing any script during the run has no effect. Changes to a library must be saved before the next start. While the script runs, line numbers match the editor for the main script; inside a library they continue past the end of the main script. If an error is found in a library while the run copy is built, the line number refers to that library. The errors at start are "Import not found", "Ambiguous import", "Imports folder contains a subfolder", "Invalid import name" and "Import is only valid as the first line". A malformed Imports line further down the script reports its syntax error instead (for example "Missing ]" or "Empty argument list"). Apart from these and the label checks, all errors are found only when the offending line runs.

## Rules and pitfalls

- Remarks (!) and labels (:) must start in the first column. Do not indent with tabs.
- Spaces outside double quotes are removed. Quote text whose spaces matter.
- Write literal words with a leading backslash (\E, \File, \MeasStart) whenever they could be the name of a variable, constant or command.
- Misspelled commands and variables are not reported; they evaluate to their own text. Check spelling against the Command reference.
- Always write square brackets: PrintLn[], not PrintLn.
- Never leave an argument empty; use Nil.
- Use (X;Y) for constant points and Point[X, Y] for points from variables.
- In curly brackets, put variables in parentheses when they touch ^ or a comparison: {(N)^2}, {(N)<5}.
- All curly brackets of a line are calculated before the line runs, including both If branches. Calculate into a variable first when a branch must not be evaluated.
- Use Smaller, Larger and Equals, not curly brackets, for While conditions. While repeats one command; Gosub, For or Goto as that command do not work.
- End the main script with End before the first label.
- Return and Next are the same command; do not leave a loop with Return.
- Do not nest CycleFiles loops, and do not call Export2DScan or Export2DRebuild inside your own CycleFiles loop.
- Folders passed to Load must end with a backslash, or pass Nil and a full path as the file name.
- Text printed before the first Output is discarded.
- Do not use variable names starting with E2D_, E2DS_ or GF_; the shipped Imports use them.

# Command reference

Signatures describe required and optional arguments. Square brackets belong to the script syntax. An ellipsis means additional arguments of the same kind. Integration times and Sleep use seconds; TOF coordinates use microseconds. Result indices start at 0, while coincidence indices start at 1. "Mainly for Imports" marks commands that exist chiefly to support the shipped Import libraries; they work in any script.

## Commands by task

| Task | Commands |
|---|---|
| Flow control | If, For, Next, CycleFiles, CycFile, While, Gosub, Return, Goto, End, Nil, Set |
| Logic and comparison | Equals, Smaller, Larger, And, Or, XOr, Not, Exists, FileExists |
| Text and files | Concat, StripPath, EnsureDirectory, ReadArray, WriteArray, ShellExec, SaveMyself |
| Output | Output, Print, PrintLn, FlushOutput |
| Coincidence criteria | CleanCoincCrit, AddCoincCrit, AddEvent, CurrCoincCrit, CurrCoincSwapROI |
| Loading and results | Load, Results, TOFRes, Im1Res, Im2Res, Add, Multiply, SetNoiseIter |
| Loaded measurement data | LoadedParam, LoadedParamFloat, LoadedMeasTime, LoadedMeasStart, LoadedChannelCount, LoadedIdentifier, ClockReset, FillLoadedParams |
| TOF evaluation | TOFInt, TOFCoG, SaveTOF, PlotTOF |
| Images and ROIs | ROIAll, ROICircle, ROIRing, ROICross, ROIPolygon, ROIRingAsCircle, Point, ROIArea, ROIToText, CurrROI, CurrRingROIAsCircle, ImageInt, ImageIntP2, ImageCenter, ImageRadius, SaveIm, PlotImage, RequestSubtract, SetAxisLength |
| Plots | NewSeries, AddXY |
| Main window and scan settings | CurrInFolder, CurrInFile, CurrOutFile, CurrSample, CurrFullSample, DayRelativeSample, CurrIntTime, CurrSetE, CurrScanStart, CurrScanStop, CurrScanStep, CurrScanIntTime, CurrScanOut |
| Measurement and beamline | Measure, LastMeasuredFile, ChannelCount, ChannelFreq, SetE, SetSlit, EPICS, EPICSFloat, SetEPICS, SwitchEPICSLabels, PrintEPICSFixed, PrintEPICSFixedStr |
| Time and messages | Sleep, WaitTil, ScriptTime, SendMessage, AlarmMessaging |

**`Add[TOF distribution 1, TOF distribution 2]`**

Adds the second TOF distribution to the first one.

**`AddCoincCrit[CoincCrit] or AddCoincCrit[Event1, Event2...]`**

Adds a coincidence criterion set to the load routine. CurrCoincCrit returns the currently set coincidence criteria from the i2 window, which can be used as a parameter. Alternatively, the individual events can be defined using the AddEvent function. At least one AddCoincCrit command has to come before any experimental data are loaded. If several criteria sets are defined, the resulting images and time-of-flight distributions can be accessed using the corresponding indices in the Results function.

**`AddEvent[Channel, TOFMin, TOFMax, ROI, TOF distribution role, Image role]`**

Returns a reference to a coincidence criterion definition for the given Channel (Ch1…Ch8, Im1, Im2), with the TOF window given by TOFMin and TOFMax (not used for the first event in an event set), the image region of interest given by ROI (only used for Im1 and Im2), what role the event should play in the TOF distribution (TOFStart or TOFStop or Nil), and if an image is defined, which window it should belong to (Im1, Im2). Several AddEvent functions can be used as parameters for an AddCoincCrit call. If TOFMin and TOFMax are negative, the time-of-flight window is still defined by their absolute values, but relative to the TOFStop event in the coincidence train as

opposed to the first event. Known issue: with such TOFStop-referenced windows and several stops per start, later events are counted more than once. For delays after a trigger, use trigger slicing in Load instead.

**AddXY[Series, X, Y]**

Adds a new point to the Series specified as the first parameter.

**AlarmMessaging[Enabled]**

Sets the alarm-messaging checkbox in the notification window. A nonzero numeric value enables it; zero or nonnumeric text disables it. Delivery uses the configured notification settings.

**And[Param1, Param2]**

Returns 1 when both arguments evaluate to nonzero numeric values; otherwise returns 0.

**ChannelCount[Channel]**

After measurement, ChannelCount returns the counts in channels Ch1…Ch8 as well as Im1 and Im2.

**ChannelFreq[Channel]**

After measurement, ChannelFreq returns the count rates in channels Ch1…Ch8 as well as Im1 and Im2.

**CleanCoincCrit**

Removes all coincidence criteria added using AddCoincCrit.

**ClockReset or ClockReset[Result index]**

Returns the number of master-clock resets recorded in the loaded result set. The optional result index defaults to 0.

A time-sliced slot returns the count of the whole file.

**Concat[Param1, Param2...]**

Evaluates the parameters and concatenates their value returning one string that can be used for example as filename.

**CurrCoincCrit**

Returns a reference to the current coincidence criteria as defined in the i2 window.

**CurrCoincSwapROI[Event index, New ROI]**

Returns the currently set coincidence criteria set, but changes the ROI belonging to the specified event to the one given in the New ROI parameter.

**CurrFullSample**

Measurement build: returns the absolute folder of the sample currently typed or selected in the main window, with a trailing backslash, creating it if necessary. Selecting a listed item is not required. A short sample name is resolved under the configured day folder; an absolute value is used as given and need not lie under the current day. Spaces, control characters, UNC and device paths, root-relative (\Sample) and drive-relative (C:Sample) forms, and a colon inside a folder name raise with the sample, folder and script line; no folder is created for a rejected value. Use this value for the Export2D measurement caller. The analysis build returns the configured sample folder as before.

**CurrInFile**

Returns the current input file name from the main window.

**CurrInFolder**

Returns the current input folder from the main window.

**CurrIntTime**

Returns the integration time as set in the main window.

**CurrOutFile**

Returns the current output file name as set in the main window.

**CurrRingROIAsCircle[Index]**

Returns the inner circle of a ring ROI in image window specified by Index or an error if the current ROI is not a ring.

**CurrROI[Index]**

Returns the current ROI from the image window specified by Index (1 or 2). The ROI is read when the run first uses it; later changes in the image window are not seen until the next run.

**CurrSample**

Returns the sample field from the main window. This may be a short sample name or an absolute folder path. Use CurrFullSample when a normalized complete folder path is required.

**CurrScanIntTime**

Returns the scan integration time from the scan window.

**CurrScanOut**

Returns the current output file name in the scan window.

**CurrScanStart**

Returns the current start energy in the scan window.

**CurrScanStep**

Returns the step size set in the scan window.

**CurrScanStop**

Returns the maximum scan energy set in the scan window.

**CurrSetE**

Returns the value from the monochromator control field in the main window.

**CycFile**

In a CycleFiles cycle, CycFile returns the current file name together with the path.

**CycleFiles[Folder, Pattern]**

Sets up a cycle over matching files in Folder using Pattern, for example *.i2d. CycFile returns the current filename including its path. Next repeats the cycle until all matches have been processed. Matches are sorted. If no files match, CycleFiles raises immediately with the folder, mask, and script line, before entering the body.

**DayRelativeSample[Folder]**

Mainly for Imports (Export2DMeas). Converts a raw folder under the configured day folder (Root\YYYY-MM\YYYY-MM-DD\) into the day-relative sample accepted by Measure, for example acetaldehyde\Run1. The day folder itself returns an empty sample. Folder is a folder path, not a sample name: a relative argument is expanded from the process working directory, like EnsureDirectory. Forward slashes, repeated separators, trailing separators and . or .. segments are normalized, and the day prefix is compared ordinally without case sensitivity. Before returning, the sample is checked to reconstruct exactly the same folder when Measure prefixes the day folder. A folder outside the current day, a sibling of the day folder, spaces in the folder or data root, UNC and device paths, root-relative and drive-relative forms, and a colon inside a folder name raise with the folder, day folder, reconstructed folder and script line. Never returns a parent traversal or an absolute fallback. No directory is created. The Export2DMeas wrapper uses it so that Measure receives the relative sample and the Meas field keeps showing it.

**End**

Stops the script.

**EnsureDirectory[Path]**

Mainly for Imports (Export2D, Export2DMeas), and generally useful. Creates missing parent directories synchronously and returns the normalized absolute directory path with a trailing backslash. Exactly one nonempty path is required. Relative paths use the process working directory. Existing directories and their contents are

preserved. Spaces and Unicode are supported without a shell. A file blocking the requested path or a parent, or another creation failure, raises with the path and script line. Export2DOpen uses this command before selecting its index or writing spectra.

**EPICS[ID]**

Returns the current formatted EPICS value for ID (see constants).

**EPICSFloat[ID]**

As EPICS, but always returns an unformatted number, that is, even for for example absorber position.

**Equals[Param1, Param2]**

Returns 1 when both numeric values are equal (1 equals 1.0), or when the text values match without case sensitivity. Otherwise returns 0. Windows folder comparisons ignore case, but do not normalize trailing separators or relative paths.

**Exists[What]**

Returns 1 if What, an image or TOF distribution returned by Results, Im1Res, Im2Res or TOFRes, exists in the loaded results; otherwise 0. It does not check variables or files; use FileExists for files.

**FileExists[FullPath]**

Mainly for Imports (Export2D). Returns numeric text 1 if the evaluated path is an existing file, otherwise 0. Directories, missing files and an empty path return 0. Exactly one argument is required. The native Unicode filesystem API is used without opening a persistent handle or invoking a shell. Relative paths use the process working directory. This command does not replace Exists, which still checks analysis-result objects. Export2DResume first flushes pending output, then uses FileExists to decide whether to print an initial header; matching scientific settings when resuming are the caller's responsibility.

**FillLoadedParams or FillLoadedParams[Result index]**

Fills the information window with parameters from the loaded result set. The optional result index defaults to 0.

**FlushOutput or FlushOutput[]**

Mainly for Imports (Export2D, Export2DMeas). Publishes everything printed so far, including an unfinished Print line after trimming its trailing tab. This also happens automatically before Output switches files and at script completion, including Stop, Terminate, and a guarded attempt on error. The whole output is written to the real filename plus .tmp, then moved onto the named file after closing the write handle. Every failed move is ignored: the named file stays unchanged and the complete .tmp remains for retry or manual recovery. A failed temporary write raises and attempts partial-file cleanup without changing the named file. With no active output, this command does nothing. Return does not flush. In addition, the output is published automatically at most every 10 s after a line that completed new output lines. Use an explicit flush after PrintLn when an external program needs to see a line immediately.

**For[Variable, Initial value, Final value]**

Assigns the initial integer value, executes the body, and increments the variable by one at each Next. The final value is included. The body executes at least once, even if the initial value exceeds the final value. Variable is a name and is not evaluated; the initial and final values must be integers, and the final value is fixed when For starts. Changing the variable in the body changes the loop. Use FileCounter as the variable to substitute exclamation marks in Load and Measure filenames: each ! is replaced by one digit of FileCounter, from the right, padded with leading zeros, so data!!!.i2d becomes data007.i2d for FileCounter 7. With fewer ! than digits, the leading digits are dropped (1234 gives 234).

**Gosub[Label, Value1, Value2...]**

Calls the subroutine at Label. When the subroutine executes Return, execution continues on the line after the Gosub. If the label declares parameters (see Labels and subroutines), give one value for each, in order. A label without parameters takes no values. The values are evaluated before any parameter is set, so Gosub[Normalize, A, X] into :Normalize X Y sets X to the value of A and Y to the old value of X. Label may also be a variable or expression that gives the label name. A subroutine may call other subroutines on its own lines. "Wrong parameter count in line N" is raised if the number of values does not match. No value is evaluated and no jump happens.

Gosub has no return value. When Gosub is used inside another function, such as a branch of If or the statement of While, expressions in curly brackets in its values are evaluated before the call, even when that branch is not taken. You may calculate such values into a variable first: T := {B + 1}, then If[C, Gosub[Normalize, A, T], Nil].

**Goto[Label]**

Continues execution at a label without parameters. A missing label raises "Label not found"; a label declaring parameters raises "Goto target has parameters". Use Gosub and Return for subroutines. Goto does not create a return frame.

**If[Condition, True branch, False branch]**

Conditional operation. The first parameter is evaluated, and if it returns a True value, that is, a non-zero number, the True branch is executed, if False, that is, zero, is returned, the False branch takes control. Nonnumeric text, including an undefined variable, counts as False. If returns the value of the branch it executes, so X := If[C, A, B] works. Only the chosen branch is executed, but all curly brackets on the line are calculated beforehand (see Calculations). See also Equals, Smaller, Larger, And, Or, XOr and Not for comparisons.

**Im1Res[Coincidence index]**

Shorthand for Results[Coincidence index, Im1, 0]. This accesses the default result slot only; use Results to select another persistent result slot.

**Im2Res[Coincidence index]**

Shorthand for Results[Coincidence index, Im2, 0]. This accesses the default result slot only; use Results to select another persistent result slot.

**ImageCenter[Image, ROI]**

Returns the center coordinates of Image, returned by Results, over the given region of interest. The results can be used for example to evaluate the image radius using ImageRadius.

**ImageInt[Image, ROI]**

Returns the sum of counts in the Image over the region of interest given by the ROI parameter. The Image parameter is returned by Results, there are several functions that return image regions of interest, which can be used as the second parameter.

**ImageIntP2[Image, ROI, EVector]**

Integrates the image by multiplying each pixel with the second-degree Legendre polynomial of the cosine of the angle between the axis defined by the pixel and the ROI center and the E vector corresponding to the polarization of the light. The image ROI must be a circle or a ring. The result can be used in angular anisotropy calculations. The Image parameter is returned by Results, there are several functions that return image regions of interest, which can be used as the second parameter.

**ImageRadius[Image, ROI, Center]**

Calculates the average radius for Image, as returned by Results, around Center, of the format (0.1;-0.1), taking the region of interest into account.

**Larger[Param1, Param2]**

Returns 1 only when both values are numeric and Param1 is greater than Param2; otherwise returns 0.

**LastMeasuredFile or LastMeasuredFile[]**

Mainly for Imports (Export2DMeas). Returns the absolute resolved path of the last successfully completed file-backed Measure in this script run, including substituted counter digits and any overwrite-protected rename. The value is assigned only after the destination has been destroyed and its final buffered raw-data write succeeded. It is empty before a successful measurement, at the start of each run, after a failed new attempt, and after memory/no-destination measurements. Load does not change it. The getter is a reserved zero-argument command value, not an assignable script variable. In NOMEAS/analysis builds it remains empty.

**Load[Folder, Name, TOFMin, TOFMax, TOFResolution]**

**Load[Folder, Name, TOFMin, TOFMax, TOFResolution, Result index]**

**Load[Folder, Name, TOFMin, TOFMax, TOFResolution, Result index, Slice width]**

**Load[Folder, Name, TOFMin, TOFMax, TOFResolution, Result index, Slice width, (Start;End)]**

**Load[Folder, Name, TOFMin, TOFMax, TOFResolution, Result index, Slice width, Trigger channel, Maximum delay]**

Loads an experimental data set with the predefined coincidence criteria sets. Folder and Name are joined without a separator, so Folder must end with a backslash (CurrInFolder does). When Name contains a path, Folder is ignored and may be Nil; in CycleFiles cycles, CycFile already contains the folder. Exclamation marks in Name are replaced by FileCounter digits as in Measure. Load also writes the folder and name of the loaded file into the main window, so CurrInFolder and CurrInFile then refer to it. The Name parameter can also be Memory, indicating that the previous measurement to memory should be processed. TOFMin, TOFMax and TOFResolution stand for the desired TOF distribution parameters in µs. The optional Result index selects the persistent result slot and defaults to 0. Loading another dataset into the same slot replaces that slot's previous results. Slots must be filled without gaps: a Result index beyond the next free slot raises "Loaded data slots must be filled in continuously". Without slicing, Load returns 0.

Time slicing (new in 3.4; see Time slicing). With a seventh parameter, Slice width in ms, Load slices the file by TDC time from time zero. It fills consecutive result slots starting at Result index, one per complete slice, and returns their number. A slice is complete when it ends at least the coincidence window (the larger of TOFMax and the widest criterion window) before the last recorded event. Later slices are dropped. With three further parameters, Slice width, Trigger channel (Ch1…Ch16) and Maximum delay in ms, it slices by the delay after the latest hit on the trigger channel. Each event is counted at most once, after the latest trigger only (see Overlapping triggers in Time slicing). It fills Maximum delay ÷ Slice width slots and returns that number. Events before the first trigger hit, or later than Maximum delay, are ignored. Trigger slicing has not been tested with real triggered data.

With an eighth parameter (Start;End) in ms instead, the absolute slicing is limited to a time window. The slices stay on the grid of the whole file (slice k covers k × width to (k + 1) × width after time zero), and Load keeps every full slice with Start ≤ k × width and (k + 1) × width ≤ End. Start is rounded up and End down to slice boundaries, so the window need not lie on the grid. An End beyond the end of the file is shortened to the last complete slice. Slot j of a window is exactly the whole-file slice it covers, and LoadedMeasStart remains the time after time zero of the file. Start and End must lie within 0 to 86400000 ms (24 h). Point[S, E] or ({S};{E}) build the window from variables, for example Load[nil, F, 0, 20, 0.004, 0, 1, (60000;62000)].

A coincidence belongs to the slice of the hit that satisfies its first criterion. Slots after the returned range keep their previous contents. Each sliced slot contains the file's beamline parameters, identifier, calibration and clock-reset count, and its own channel counts, measurement time and slice time.

Images are kept for every slot when an image role is set: about 1.3 MB per image and slot for 400 × 400 pixels. Use Nil image roles when slicing finely. Slice widths that are multiples of 0.02 ms give exactly printable slice midpoints. The number of slots is not limited, and every slot keeps its TOF distributions in memory. These errors stop the script before the file is read: "Slice width must be a number", "Slice width must be positive", "Slice width is below the TDC resolution", "Trigger channel must be one of Ch1...Ch16", "Maximum delay must be a number" and "Maximum delay must be at least one slice width", and, for a window, "Slice window must be written (Start;End) in ms", "Slice window must lie within 0 to 86400000 ms", "Slice window must contain at least one full slice" and "Slice window has too many slices; use a wider slice or a shorter window" (more than 2147483647 slices). More than nine arguments raise "Wrong parameter count".

**LoadedChannelCount[Channel] or LoadedChannelCount[Channel, Result index]**

Returns the channel count from a loaded result set. The optional result index defaults to 0.

For a time-sliced slot, it returns the recorded hits in that slice or delay bin. The slots need not add up to the file total. In trigger slicing, the first slot holds the number of trigger hits.

**LoadedIdentifier[Result index]**

Returns the identifier stored in the loaded measurement. The optional result index defaults to 0.

**LoadedMeasStart or LoadedMeasStart[Result index]**

Returns the date and time at the start of the loaded measurement. The optional result index defaults to 0.

For a time-sliced slot, it returns the slice midpoint in ms after TDC time zero, or the delay midpoint after the trigger, with two decimals.

**LoadedMeasTime or LoadedMeasTime[Result index]**

Returns the integration time of the loaded measurement in seconds. The optional result index defaults to 0.

For a time-sliced slot, it returns the slice width in seconds with five decimals. In trigger slicing, it returns the total time that trigger periods spent in that delay bin.

**LoadedParam[When, ID] or LoadedParam[When, ID, Result index]**

Returns the formatted beamline parameter for ID from the loaded measurement at Start or Final. The optional result index defaults to 0.

**LoadedParamFloat[When, ID] or LoadedParamFloat[When, ID, Result index]**

Returns the numeric beamline parameter for ID from the loaded measurement at Start or Final. The optional result index defaults to 0.

**Measure[Raw folder, Output name, Integration time, Min. beam current, Refresh]**

Acquires with the existing native three-, four- or five-argument forms. An absolute raw folder, or a short sample name under the configured day folder, and a filename select a file-backed measurement with counter substitution and overwrite protection. An empty output (Nil or an empty variable; an argument left empty is dropped) selects no destination; Memory selects a memory measurement. The extension is always .i2d, and exclamation marks are replaced by FileCounter digits (see For). Minimum beam current is optional and waits before acquisition until the beam is sufficient. The optional fifth Refresh argument controls image/TOF refresh and defaults to True. LastMeasuredFile is cleared before each attempt and becomes the absolute resolved path only after successful file-backed acquisition and its final buffered write. The imported Meas wrapper is called as Gosub[Meas, RawFolder, Pattern, Duration, MinBeam]; it deliberately uses four actual arguments and processes the completed path outside native Measure. Stop/Terminate retain their existing semantics; interrupted acquisition does not guarantee a scientifically complete measurement. Measure updates the sample, output file, input folder and file and integration time in the main window. A script that contains Measure[ on any line that is not a remark, including the lines of its Imports, checks the notification settings and the free disk space when it starts.

**Multiply[TOF distribution or image data, Factor]**

Multiplies the loaded TOF distribution or image data with the given factor.

**NewSeries[Name]**

Returns a reference to a new series, and changes the TOF window to 2D plot mode. The result of the NewSeries function can then be passed on to AddXY to add points to the plot. The new series will appear under the optional Name in the graph window.

**Next**

Returns the execution to the beginning of the innermost For or CycleFiles cycle, or ends it after the last pass. Next and Return are the same command (see Labels and subroutines).

**Nil**

Does nothing and returns an empty value, for example as an unused branch of If or an omitted Folder in Load.

**Not[Param1]**

Returns 1 only when the argument is numeric and zero. Returns 0 for nonzero numeric values and for nonnumeric text.

**Or[Param1, Param2]**

Returns 0 only when both arguments are numeric and zero. Otherwise returns 1, including when either argument is nonnumeric.

**Output[Folder, Name] or Output[Folder, Name, Mode]**

Selects the output location and filename. The extension is .txt unless Name ends in .out. A new file starts with its own full path as the first line. Text printed before the first Output of a run is discarded. Two arguments or Mode 1 protect an existing file by choosing a numbered alternative. Mode 0 replaces the requested canonical name. Mode 2 (mainly for Imports: Export2DResume) resumes that name: existing text is loaded once, with its detected encoding and line boundaries, then subsequent Print/PrintLn text is published through the same whole-buffer temporary-file mechanism. A missing file starts with the usual filename line. An existing empty file remains empty until text is printed. An unreadable existing file raises before the active output is flushed or changed. Reopening the same active append destination retains pending text and unpublished buffered rows instead of rereading stale disk content. Mode must evaluate to 0, 1 or 2; count and mode are checked before switching. Both .out and .txt are preserved, with .txt used for other names. Selection does not create or truncate the file. Switching destinations flushes the old output, including a pending line. All failed temporary-file moves remain ignored. Use EnsureDirectory first for missing parents; see FlushOutput for publication and error behavior.

**PlotImage[Image, ImageWindow]**

Plots the image, given by the first parameter, in the image window specified by the second parameter. Images can be accessed after Load using the Results function, ImageWindow is Im1 or Im2, depending on which window the image is to be plotted in.

**PlotTOF[TOF distribution]**

Plots the specified TOF distribution, returned by the Results function, in the TOF window.

**Point[X, Y]**

Can be used in ROI definitions for the center point, alternatively to the immutable and constant (X;Y) specification. It can also represent a polygon vertex.

**Print[Param1, Param2, ...]**

Evaluates the parameters and appends their values to the current output line. A tab follows each value. Print without square brackets appends nothing. Text accumulates until PrintLn, FlushOutput, an output switch, or script completion finishes the line.

**PrintEPICSFixed[What]**

Prints the parameter (should be a number) into the zoom line of the Beamline parameters (EPICS) window. Result is also shown in main window in the analysis program.

**PrintEPICSFixedStr[What]**

Prints the parameter as string into the zoom line of the Beamline parameters (EPICS) window. Result is also shown in main window in the analysis program.

**PrintLn[Param1, Param2, ...] or PrintLn[]**

Appends values as Print does, removes the final tab, and finishes the pending line. PrintLn[] (or PrintLn without brackets) finishes an existing pending line; it does nothing when no text is pending. It does not by itself force publication to disk.

**ReadArray[File name, Array variable]**

The file is read and each line is assigned as a new item to array variable. Each line is evaluated like an argument, so a line that is the name of a variable, constant or command is replaced or executed; the array name is not evaluated. The index of the first element is zero, that is, if ReadArray[x.txt, ab] is executed and x.txt has 10 lines, ab[0] to ab[9] will be set. See also WriteArray.

**Return**

Ends a subroutine called with Gosub; execution continues on the line after the Gosub. Return is the same command as Next and acts on the innermost open loop or call (see Labels and subroutines).

**RequestSubtract[Image]**

Returns 1 when the selected image window exposes an enabled and checked subtraction control; otherwise returns 0. Pass Im1 or Im2.

**Results[Coincidence index, What, Result index]**

Returns an image or TOF distribution from a loaded result set. What can be Im1, Im2 or TOF. Coincidence index 1 refers to the first coincidence-criteria set added using AddCoincCrit, 2 to the second, and so on. The optional Result index selects the persistent result slot and defaults to 0; it corresponds to the final optional parameter of Load. Im1Res[Index], Im2Res[Index] and TOFRes[Index] are shorthand for the default result slot only.

**ROIAll**

Returns an image region of interest for the whole image.

**ROIArea[ROI]**

Returns the area of the given image region of interest in pixel numbers.

**ROICircle[Center, Radius, Anisotropy]**

Returns a circular image region of interest, of which the center is given as, for example, (0.1;-0.1).

**ROICross[Center, Width]**

Returns a cross image region of interest, of which the center is given as, for example, (0.1;-0.1), the width is a scalar.

**ROIPolygon[Point1, Point2, Point3, ...]**

Returns a polygon image region of interest whose ordered vertices are given as (X;Y), for example: ROIPolygon[(0.12;-0.05),(-0.08;-0.05),(0;0.20)]. Point[X,Y] may be used instead of (X;Y). At least three non-collinear vertices are required. Coordinates must be between -10 and 10 and are rounded to three decimal places.

**ROIRing[Center, Radius1, Radius2, Anisotropy]**

Returns a ring image region of interest, of which the center is given as, for example, (0.1;-0.1).

**ROIRingAsCircle[ROI]**

Returns a circle ROI with the supplied ring's center, inner radius and anisotropy. The original ring is unchanged. The argument must be a ring ROI, including a stored handle or CurrROI for a selected ring. Use ROIRingAsCircle[Ring] as the Center argument of the Export2D Imports to avoid separately specifying matching circle geometry.

**ROIToText[ROI]**

Converts the given image region of interest into a readable canonical text format to be printed. Export2D uses it for the Center:, Ring: and IonROI: lines of its index. Printing a ROI handle directly gives internal text such as ~3~0. Whenever current ROIs are used in evaluation scripts, the analysis can only be repeated if the ROIs are printed in the output file. Note that this is not the Manoscript command to create a ROI, but the ROI signature.

**SaveIm[Image, ROI, SaveROI, FileName, FileExt,Precision]**

Saves the loaded Image under the given ROI (can be ROIAll if the total image is to be saved) to the file named FileName, of which the extension is changed to ".img" with FileExt included before the extension (for example to allow for different ms-VMIs belonging to the same file). SaveROI can be True or False and tells the program whether to save the ROI in the file, as well. If the ROI is not saved, the output only consists of 400 × 400 numbers. The last optional parameter, Precision, specifies the number of digits to be printed after the decimal point (default 2).

**SaveMyself[Folder]**

Saves the script as it is being run, under the script's name with the extension .scr and overwrite protection, in the specified folder. For a script with Imports, this is the run copy: the first line is a remark and all imported libraries are included, so the saved file runs without the Imports folder. The same text is recorded in the logbook when the script starts.

**SaveTOF[TOF distribution, TOFMin, TOFMax, FileName, FileExt, SaveTimes, Precision]**

Saves the loaded TOF distribution between times TOFMin and TOFMax to the file named FileName, of which the extension is changed to ".tof" with FileExt included before the extension (for example to allow for saving center and ring TOF distributions belonging to the same file): run001.i2d with FileExt _center gives run001._center.tof. SaveTimes can be True or False and tells the program whether to include times of flight or only the ion counts per channel. The last parameter, Precision, is optional, and specifies the number of digits to be printed after the decimal point. The default, 0, is only reasonable for non-normalized and non-subtracted TOF distributions with solely integer counts in each TOF bin.

**ScriptTime**

Returns the number of seconds the script has been running.

**SendMessage[Text]**

Passes a script message to the notification window. Delivery depends on the configured notification settings. The Imports use SendMessage followed by End to stop on misuse with an explanation.

**Set[Variable, Value]**

The Set function is equivalent to the assignment line Variable := Value, but can be invoked from within an If function, for instance.

**SetAxisLength[NewLength]**

Changes the axis length of the images to NewLength. The measurement and the data analysis programs have to be restarted every time the image size is changed. Starting September 2016, the image size is saved to file, and is changed automatically upon loading a file with a different image size. For earlier experimental data, the image size has to be adjusted manually prior to loading the file.

**SetE[E] or SetE[E, NoBlock]**

Sets the light energy to E and waits for the monochromator to reach the given energy. If the second parameter evaluates to 0, as the constant NoBlock does, the execution continues immediately, and the script does not wait for the monochromator. Any other second parameter waits as well.

**SetEPICS[ID, NewValue]**

Can be used to write a new value to an EPICS channel, that is, close and open valves, gas filter set voltages directly etc. Channel IDs are not documented, may change, and need to be confirmed.

**SetNoiseIter[n]**

When the false coincidence background is subtracted in for example TOFInt, the noise subtraction is done n times (1 to 50) (default 1 unless noise offset is set as 0). The first interval starts at TOFMin + NoiseOffset, the second one at TOFMin + NoiseOffset + 0.96 μs and so on.

**SetSlit[TRXR, X, TRYB, TRYT]**

Moves the front-end slits to the given positions in mm, for example SetSlit[-25, X, -20, 20]. Allowed values (inclusive): TRXR from -25 to 0, TRYB from -20 to 0, TRYT from 0 to 20. Values are never clamped. The second argument (TRXW) is ignored because that motor is disabled; use a placeholder such as X. It is evaluated like any other argument (a variable is read, a bracket call runs), but its value is discarded. All three active values are checked before any motor is written; a wrong argument count or an invalid value stops the script with an error naming the line.

After writing the three positions, SetSlit waits 3 s, then checks once per second until all three motors report done (DMOV = 1), and finally waits a further 1 s. A request to the current position completes normally. If any motor is still not done 90 s after the positions were written (the initial 3 s included), the script stops with an error listing the three DMOV values. Stop takes effect only after SetSlit has finished; Terminate interrupts the wait but does not stop the motors. The slit set and done channels are not logged; the existing slit readbacks are logged as before. In the analysis program, SetSlit does nothing.

**SetVoltageSet[Name]**

This registered legacy command is unsupported. Every call raises "SetVoltageSet support ended". It does not apply a voltage set.

**ShellExec[command]**

Executes the command (path, application name and optional parameters), waits until it exits and returns its output. No command shell is involved: use cmd /c for shell commands such as dir, and quote the whole command to keep its spaces. If the program cannot be started, the Windows error text is returned instead of an error. Terminate ends the program.

**Sleep[Time]**

Pauses the execution for Time number of seconds. Only Terminate interrupts it; Stop takes effect afterwards.

**Smaller[Param1, Param2]**

Returns 1 only when both values are numeric and Param1 is less than Param2; otherwise returns 0.

**StripPath[Name]**

Returns the filename part of a full file specification. Useful on CycFile when printing the path several times is not desirable.

**SwitchEPICSLabels[OldLabel, NewLabel]**

Switch count rate labels in the EPICS window, to be used to include low-res HPTDC channels if required, for example by SwitchEPICSLabels[Ch3, Ch9].

**TOFCoG[TOF distribution, TOFMin, TOFMax, NoiseOffset, MomentIndex, MomentCenter]**

Returns the nth moment of the given TOF distribution around MomentCenter. If MomentIndex is 1 and MomentCenter 0, this is equivalent to the peak center of gravity.

**TOFInt[TOF distribution, TOFMin, TOFMax, NoiseOffset]**

Returns the integral of the TOF distribution between TOFMin and TOFMax in µs. NoiseOffset (in µs, a multiple of 0.96 µs for high count rates) can be used to subtract the false coincidence background, or can be 0 when no subtraction is desired.

**TOFRes[Coincidence index]**

Shorthand for Results[Coincidence index, TOF, 0]. This accesses the default result slot only; use Results to select another persistent result slot.

**WaitTil[Time]**

Suspends the execution of the script until the given time passes. Time is read in the Windows date and time format of the computer, so careful testing is required before the first use on any computer. Only Terminate interrupts the wait.

**While[Condition, Statement]**

Evaluates the Condition, and executes Statement over and over again until Condition becomes False or Stop is pressed. Statement is a single command evaluated in place: Gosub, Goto and For do not work as the statement. Curly brackets in the line are calculated only once, before While starts, so write the condition with Smaller, Larger or Equals, for example While[Smaller[ScriptTime, 60], Sleep[1]].

**WriteArray[File name, Array variable, Low limit, High limit]**

The array variable is written (without overwrite protection!) to file as specified by the index range in the last two parameters. See also ReadArray.

**XOr[Param1, Param2]**

Returns 1 when exactly one argument evaluates to a nonzero numeric value; otherwise returns 0.

## Constants and special variables

The following names are reserved constants; they cannot be assigned. Each evaluates to a number, which is why a literal word that equals one of them needs a leading backslash. Further EPICS channels are identified by their numeric ID in i2PEPICO's channels.dat; verify the installed beamline configuration before using an ID.

| Name | Meaning |
|---|---|
| Ch1 through Ch16 | TDC channel references (0 to 15) |
| Im1 and Im2 | Image 1 and 2 references |
| MeasStart | As the channel of the first event of a criterion, starts the event train at TDC time zero of the file |
| TOF | TOF distribution reference |
| Start and Final | Loaded measurement parameters at the beginning or at the end of the measurement |
| E | Monochromator energy channel (0) |
| EF | Experimental chamber full range |
| SF | Source chamber full range |
| EV | Electron detector voltage (8A) |
| IV | Ion detector voltage (8B) |
| I_Ring | Beam current |
| SP | Source pressure |
| Memory | Measure to or load from memory |
| True and False | Logical values (1 and 0) |
| TOFStart and TOFStop | Event role in TOF distribution |
| NoBlock | SetE parameter (0); not to wait for mono |
| On | 1 |
| Off | 0 |

FileCounter is not reserved: it is an ordinary variable that is 0 at the start of every run. Exclamation marks in Load and Measure filenames are replaced by its digits (see For).

## Shipped Imports

The program ships six Import libraries. They are ordinary ManoScript scripts, kept in the Imports folder of script.dat (see Imports) and called with Gosub. The authoritative copies are in Resources\Imports of the source tree, named by their Import captions.

| Library | Use | Contents |
|---|---|---|
| Export2D | Analysis | Export of closed raw files for PIMP: index file plus center, ring and all TOF spectra per file. Imports Export2DSLS1 and Export2DSLS2. |
| Export2DSLS1, Export2DSLS2 | Analysis | Beamline parameter columns for historical SLS1 files and for SLS2 files. Used only by Export2D. |
| Export2DSlices | Analysis | Time-sliced export of one raw file (new in 3.4). Imports Export2D. |
| Export2DMeas | Measurement | Measure-and-export wrapper: one index row per completed measurement. Imports Export2D. |
| GFSwitch | Measurement | Gas-filter switching between the noble-gas mixture and neon. |

The analysis package's script.dat already contains Export2D, Export2DSLS1, Export2DSLS2 and Export2DSlices. ScriptImport.exe adds or updates exactly these four in any other script.dat and leaves everything else unchanged. On the measurement computer, the libraries are installed file by file with ScriptImport.exe --target <script.dat> <library>.scr. Close i2PEPICO before running ScriptImport: a running i2PEPICO overwrites script.dat when it saves or exits. ScriptImport first prints its version and the label of its bundled Imports.

**Rules for callers.** Call only the entry points listed below; the other labels in the libraries are internal and may change. All variables are global, so the libraries reserve the name prefixes E2D_ (Export2D, Export2DSLS1, Export2DSLS2, Export2DMeas), E2DS_ (Export2DSlices) and GF_ (GFSwitch): do not use these prefixes in your own scripts. UseSLS1, UseIonROI and IonROI are the input variables you set yourself; they are read when an export session opens. Imports must be the literal first line of the calling script.

### Export2D: export for PIMP

Export2D writes one export folder: an index file 2D_TPES_index.txt and, for each raw file, three TOF spectra named after the raw file, for example run001._center.tof, run001._ring.tof and run001._all.tof (0–20 µs, 4 ns bins, integer counts). The three spectra use the coincidence sets Im1 in the center ROI, Im1 in the ring ROI and Im1 anywhere, each with an Im2 hit 0–20 µs later as the stop: any Im2 hit, or only hits inside the ion ROI (see Ion ROI below). The sets have no image roles, so the export fills no result images.

UseSLS1 := 1 before the first call selects the SLS1 parameter columns. UseSLS1 := 0, or no assignment, selects SLS2.

**Ion ROI (new in 3.41).** UseIonROI := 1 (or True) before the call counts only ions inside IonROI, an ROI on the ion image (Im2): for example IonROI := ROICircle[(0.1;0.1),0.2,1], a ROIPolygon, or CurrROI[Im2] for the ROI drawn on the ion image. Image coordinates run from -0.5 to 0.5. The ion ROI applies to all three spectra. UseIonROI := 0, or no assignment, uses the whole ion image, and the export is then identical to that of version 3.4. Both variables are read when a session opens (Export2DScan, Export2DRebuild, Export2DOpen, Export2DResume, Export2DMeasureOpen and the Export2DSlices entry points); CurrROI[Im2] is different: it is resolved the first time the script uses it and then kept for the rest of the script run, also across later sessions. Draw the ion ROI before starting the script, and start the script again to use a newly drawn ROI. If UseIonROI is set but IonROI is not, the script stops with "ROI not found in line …" (a line of the Export2D library) before the index is opened.

**`Gosub[Export2DScan, Mask, RawFolder, OutputFolder, CenterROI, RingROI]`**

Exports all files in RawFolder that match Mask (for example *.i2d; not recursive) into OutputFolder, which is created when missing. The index is replaced. This is the usual offline call. Use ROIRingAsCircle[Ring] as CenterROI so that the center is exactly the hole of the ring.

**Gosub[Export2DRebuild, RawFolder, Mask, CenterROI, RingROI]**

As Export2DScan, but the output is the fixed child folder RawFolder\PIMP, which is where the measurement wrapper writes. Use it to recreate the PIMP folder of a measurement after a crash or with corrected ROIs. Run it while no export session is active.

**Gosub[Export2DOpen, OutputFolder, CenterROI, RingROI]**

**Gosub[Export2DResume, OutputFolder, CenterROI, RingROI]**

**Gosub[Export2DFile, RawFile]**

**Gosub[Export2DClose]**

The building blocks of Export2DScan, for scripts that export files one by one. Open starts a new index and replaces an existing one (Output mode 0). Resume appends to an existing index and writes the header only if the index does not exist yet. File loads one raw file and appends its row; it does nothing without an open session. Close ends the session. A second Open or Resume while a session is active does nothing, so Close first to change folders or ROIs. The ROIs and other settings are not checked when resuming; keeping them consistent is the caller's responsibility. This includes the ion ROI: resuming an index with another UseIonROI or IonROI mixes rows silently, so start a new index for a new ion ROI.

**Index file.** 2D_TPES_index.txt is tab-separated text. Its header lines are Export2D: 1, Schema: SLS1 or SLS2, Center: and Ring: (the ROIs as ROIToText), IonROI: (only with UseIonROI; the ion ROI as ROIToText), TOFMin:, TOFMax: and TOFBin: (µs), and one Parameter: <name> | <label> line per parameter column. The column header line follows: File, MeasTime, E, MeasStart and the remaining parameter names. Each data row holds the raw file name, LoadedMeasTime (s), the photon energy, LoadedMeasStart and the parameters at the start of the measurement. PIMP does not read the IonROI: line; it records the export settings.

## Export2DMeas: measure and export

Export2DMeas is for Measure scripts on the measurement computer. Import only Export2DMeas; it brings Export2D and the schemas with it. It exports into the fixed child folder PIMP of the raw folder, so PIMP can watch the folder while the measurement runs.

**Gosub[Export2DMeasureOpen, RawFolder, CenterROI, RingROI]**

Opens or resumes the export in RawFolder\PIMP. RawFolder must lie below the current day folder and contain no spaces (checked with DayRelativeSample before anything is created). Calling it again for the same folder keeps the active session; a different folder stops the script with a message.

**Gosub[Meas, RawFolder, Pattern, IntegrationTime, MinBeamCurrent]**

Measures one file with Measure (exactly these four arguments), then loads the completed file (LastMeasuredFile), saves its three spectra, appends its index row and publishes the index with FlushOutput. Stops the script if no session is open or no file was completed. Use CurrFullSample as RawFolder and exclamation marks in Pattern for FileCounter substitution.

## Export2DSlices: time-sliced export

Export2DSlices exports one closed raw file, sliced by time, into its own folder (new in 3.4). It uses the time-sliced Load (see Load and Time slicing). Set UseSLS1, and optionally UseIonROI and IonROI, as for Export2D; the ion ROI applies to all three slicing modes.

**Gosub[Export2DSlices, RawFile, OutputFolder, CenterROI, RingROI, SliceWidth]**

Slices RawFile (full path) by TDC time from time zero. SliceWidth is in ms and is first rounded to a whole multiple of 0.01 ms; a result below 0.02 ms stops the script. Each complete slice becomes one index row and three spectra named after the raw file with a five-digit slice number counted from zero: run001.i2d_s00000.i2d, run001.i2d_s00001.i2d, …, with spectra such as run001.i2d_s00000._center.tof. MeasStart is the slice midpoint in ms (500.00, 1500.00, … for 1000 ms slices) and MeasTime the slice width in seconds (1.00000). The beamline parameters are those of the whole file. The index has two extra lines after the column header line: Source: <raw file> and Slicing: Time <width in ms>.

**Gosub[Export2DSlicesWindow, RawFile, OutputFolder, CenterROI, RingROI, SliceWidth, Start, End]**

As Export2DSlices, but only for the full slices between Start and End in ms (see Load), for example 1 ms slices from 60000 to 62000 ms. The slice files are again numbered from _s00000, while MeasStart gives the time after time zero of the file (60000.50, 60001.50, …). A third extra index line records the request: Window: requested <Start> <End>; the rows show the slices actually exported. If the window holds no complete slice, for example because it starts after the end of the file, the script stops with a message.

**Gosub[Export2DSlicesTrigger, RawFile, OutputFolder, CenterROI, RingROI, BinWidth, TriggerChannel, MaxDelay]**

Untested with real triggered data. Slices by the delay after the latest hit on TriggerChannel (Ch1…Ch16), in bins of BinWidth ms (rounded as above) up to MaxDelay ms. MeasStart is the delay-bin midpoint in ms; MeasTime is the total time, in seconds, that trigger periods spent in that bin. The extra index line reads Slicing: Trigger Ch<n> <width> <maximum delay>. Each event is counted after the latest trigger only; see Overlapping triggers in Time slicing.

Use a new, empty output folder for every raw file and do not name it PIMP. A rerun replaces the index, but spectra of slices that the new index no longer lists stay in the folder. No entry point may be called while an Export2D session is open. The spectra are kept in memory for all slices until the export finishes (about 120 KB per slice), so choose the width, or a window, so that the export gives no more than a few thousand slices. PIMP opens a sliced folder only on its own; versions released before this feature reject it (MeasStart contains no recognized date/time values).

## GFSwitch: gas filter

GFSwitch is for Measure scripts at the beamline. It switches the gas filter only when the beamline valve is open, and does nothing otherwise.

**Gosub[MixToNe]**

**Gosub[NeToMix]**

MixToNe switches from the mixture to neon if the mixture valve is open; NeToMix switches back if the neon valve is open. Each closes the beamline valve, closes the old gas, waits 18 s with everything closed, opens the new gas, waits (198 s after MixToNe, 28 s after NeToMix), reopens the beamline valve and waits 2 s more. The EPICS channel IDs are fixed in the library; check them against the installed channels.dat after a beamline change.

## Time slicing

New in 3.4. A time-sliced Load reads one raw file once and sorts its coincidences into consecutive result slots, one per time slice. It gives TOF spectra as a function of time within one file, for example to follow a process after a valve opening or during a slow change, at a resolution far below the length of a file. Export2DSlices writes such a series as a PIMP folder.

| | **Absolute slicing** | **Trigger slicing (untested with real data)** |
|---|---|---|
| Call | Load[…, Slot, Width] or Load[…, Slot, Width, (Start;End)] | Load[…, Slot, Width, Channel, MaxDelay] |
| Time axis | TDC time from time zero of the file | Delay after the latest hit on the trigger channel |
| Slots filled | One per complete slice; the last slices, which end less than the coincidence window before the last recorded event, are dropped | MaxDelay ÷ Width, always |
| LoadedMeasStart | Slice midpoint in ms (two decimals) | Delay-bin midpoint in ms |
| LoadedMeasTime | Slice width in s (five decimals) | Total time trigger periods spent in the bin, in s |
| LoadedChannelCount | Hits recorded in the slice | Hits in the bin; the first bin holds the number of trigger hits |
| Import | Export2DSlices, Export2DSlicesWindow | Export2DSlicesTrigger |

A coincidence belongs to the slice of the hit that satisfies its first criterion (usually the electron, Im1), even if its stops arrive after the slice boundary, so no coincidence is split or lost at a boundary. The slices of a file therefore never contain more counts than the unsliced spectrum. The time base is the TDC clock only; the wall-clock start of the file plays no part.

Trigger slicing is implemented, but it has not been tested with data from a triggered experiment. Check its results critically before relying on them: compare the sum of the delay bins with the unsliced spectrum, and LoadedChannelCount[Trigger, first slot] with the expected number of trigger pulses.

**Overlapping triggers.** Trigger slicing uses only the latest trigger hit, so every hit and every coincidence is counted in at most one delay bin (in none if it comes before the first trigger or later than Maximum delay), as in absolute slicing. The delay of a coincidence is that of the hit that satisfies its first criterion; its stops may arrive after the next trigger. A trigger hit ends the previous trigger period: if the next trigger arrives before Maximum delay, later events are counted from the new trigger, and the bins beyond that delay get no exposure from the shortened period. LoadedMeasTime counts only the time that trigger periods actually spent in each bin, so counts ÷ LoadedMeasTime is a correct rate in every bin, however irregular the triggers. Two consequences follow. With a periodic trigger and a Maximum delay longer than the trigger period, the bins beyond the period are never reached; they have zero exposure and PIMP shows them as missing. A response to one trigger that lasts beyond the next trigger is counted at short delays after the next one. Events are never counted once for every earlier trigger within Maximum delay (multi-start correlation). Keeping Maximum delay no longer than the shortest trigger spacing avoids the bins without exposure, but not the carry-over: with triggers 1 ms apart and a Maximum delay of 0.2 ms, a response 1.05 ms after one trigger is still counted at 0.05 ms after the next. Exposure normalization does not remove carry-over; only a trigger spacing longer than the response avoids it.

**Choosing the width.** Use a whole multiple of 0.01 ms; Export2DSlices rounds to this grid and PIMP requires it, and even multiples of 0.01 ms give exactly printed midpoints. Each slot holds its TOF spectra in memory until the end of the script (about 120 KB per slice for the three Export2D spectra, plus about 1.3 MB per image if image roles are set), and the number of slots is not limited by the program. A 1 h file sliced at 1 s gives 3600 slots, about 430 MB; sliced at 1 ms it would need 3.6 million slots and fail. Start coarse. PIMP is tested with up to 1000 slices and accepts at most 24 h of slice time.

**Zooming in.** To look at part of a file at a fine width, slice the whole file coarsely first, find the interesting time, then slice only a window around it: for example (60000;62000) at 1 ms gives 2000 slots, about 240 MB, instead of 3.6 million for the whole hour. The window's slices are bit for bit the slices of the whole file at the same width, so a window and a whole-file export share one time axis. The whole file is still read, so a window takes as long to load as the whole file.

**Known limitations.** A file with a gap of more than 2 s without any hit, or whose first hit comes more than 2 s after time zero, is compressed at the gap by the reader, which shifts all later slice times, and window bounds, without a warning. Slots after the returned range keep the contents of earlier Loads; use the returned number. FillLoadedParams shows the midpoint of a sliced slot as a date in 1899.

## Export examples

These examples are complete main scripts for the shipped Imports. Paste the measurement example into a Measure script and the others into Process scripts, then use the ordinary start control. Imports must be the literal first line. The examples use SLS2; for historical SLS1 files, set UseSLS1 := 1 immediately after the Imports line. The same scripts, with more remarks, are in the Examples folder of the program package.

Check the ring, filenames, energy ranges, integration time and minimum beam current before running. ROIRingAsCircle derives the matching center from the ring. CurrFullSample gives the full configured raw sample folder; CurrInFolder selects the input of the Process examples. Native filename protection and counter substitution still apply.

### Measure two energy ranges

```
Imports[Export2DMeas]
Ring := ROIRing[(-0.109;-0.134),0.03,0.06,1]
Gosub[Export2DMeasureOpen, CurrFullSample, ROIRingAsCircle[Ring], Ring]
For[FileCounter,0,9]
SetE[{10.525+FileCounter*0.010}]
Gosub[Meas, CurrFullSample, acetone_d6.DD2!!!.i2d, 3000, 380]
Next
For[FileCounter,0,12]
SetE[{10.5475+FileCounter*0.005}]
Gosub[Meas, CurrFullSample, acetone_d6.DD3!!!.i2d, 3000, 380]
Next
```

Meas takes exactly four arguments: full raw folder, filename pattern, integration time and minimum beam current. It acquires with the native Refresh default, processes LastMeasuredFile after its final raw write, saves Center/Ring/All TOFs and publishes one index row. Native Measure contains no automatic Export2D hook. The raw folder must lie under the current day folder without spaces: Export2DMeasureOpen and Meas reject other folders with DayRelativeSample before creating raw or PIMP folders, and Meas passes the checked day-relative sample to native Measure, so the sample field stays relative.

### Rebuild closed measurements

```
Imports[Export2D]
Ring := ROIRing[(-0.109;-0.134),0.03,0.06,1]
Gosub[Export2DRebuild, CurrInFolder, *.i2d, ROIRingAsCircle[Ring], Ring]
```

Press Stop in PIMP and run this while acquisition/export is idle. Rebuild replaces the canonical index and selected derived TOFs, keeping raw measurements and unrelated files. A mask with no matches raises before replacement output opens. Press Play in PIMP after completion to reread all spectra. Resume flushes pending rebuild output before checking whether its header exists.

No caller entry label, tail End, FlushOutput, Close or EnsureDirectory is required. The interpreter inserts End boundaries before imported libraries and flushes at completion. Use Export2DClose only when switching sessions within one run. Repeated MeasureOpen for the same active folder keeps its ROIs and settings; Close/Open is required to change them. Consistent scientific settings on resume remain the caller's responsibility. An interrupted run may publish partial output; failed temporary-file moves retain their existing behavior.

### Time-sliced export

```
Imports[Export2DSlices]
! PROCESS: time-sliced export of ONE closed raw i2d file for PIMP.
! Select the raw folder and the file in the main input controls.
UseSLS1 := 0
Ring := ROIRing[(0;0),0.03,0.06,1]
RawFile := Concat[CurrInFolder, CurrInFile]
OutputFolder := Concat[CurrInFolder, Slices_example]
SliceWidth := 1000
! Slice width in ms, rounded to a whole multiple of 0.01 ms (at least 0.02 ms).
! Use one empty output folder per raw file, not named PIMP.
Gosub[Export2DSlices, RawFile, OutputFolder, ROIRingAsCircle[Ring], Ring, SliceWidth]
```

This writes Slices_example\2D_TPES_index.txt with one row per complete 1 s slice, and the spectra run001.i2d_s00000._center.tof, … for a raw file run001.i2d. Delete or rename the output folder before exporting again with another width. Open the folder in PIMP on its own.

### Zoom into a time window

```
Imports[Export2DSlices]
UseSLS1 := 0
Ring := ROIRing[(0;0),0.03,0.06,1]
RawFile := Concat[CurrInFolder, CurrInFile]
OutputFolder := Concat[CurrInFolder, Window_example]
! 1 ms slices from 60 s to 62 s after time zero of the file.
Gosub[Export2DSlicesWindow, RawFile, OutputFolder, ROIRingAsCircle[Ring], Ring, 1, 60000, 62000]
```

This writes 2000 rows, run001.i2d_s00000.i2d to run001.i2d_s01999.i2d, with MeasStart 60000.50 to 61999.50. In PIMP the time axis of the folder runs from 60000 to 62000 ms.

### Trigger-sliced export (untested)

```
Imports[Export2DSlices]
UseSLS1 := 0
Ring := ROIRing[(0;0),0.03,0.06,1]
RawFile := Concat[CurrInFolder, CurrInFile]
OutputFolder := Concat[CurrInFolder, Trigger_example]
! Delay bins of 0.02 ms up to 10 ms after the latest hit on Ch3.
Gosub[Export2DSlicesTrigger, RawFile, OutputFolder, ROIRingAsCircle[Ring], Ring, 0.02, Ch3, 10]
```

Trigger slicing has not been tested with data from a triggered experiment. Check the trigger channel against the cabling of the experiment, and check the result as described in Time slicing before using it.

For PIMP controls and analysis workflows, use the PIMP user guide (pim.pdf). This manual is the reference for ManoScript itself. Other sample scripts are stored in the application's script.dat.

# Version history

Main changes per version. The program and this manual share the version number.

| Version | Date | Main changes |
|---|---|---|
| 3.41 | 20 October 2026 | Optional ion ROI for the shipped exports: UseIonROI and IonROI restrict the Export2D and Export2DSlices spectra (all entry points, including time, window and trigger slicing, and Export2DMeas) to ions in a region of the ion image, recorded in an IonROI: index line; without UseIonROI, exports are unchanged. The shipped Imports no longer fill result images. ScriptImport prints its version and bundle label on every run. The executables carry version 3.4.1.0. The manual describes how trigger slicing treats overlapping triggers. |
| 3.4 | 17–18 October 2026 | Time slicing: Load fills one result slot per TDC time slice, optionally only within a time window (Start;End), or per delay bin after a trigger channel (untested with real triggered data). New Import Export2DSlices (Export2DSlices, Export2DSlicesWindow, Export2DSlicesTrigger) writes slice series as PIMP folders numbered _s00000, _s00001, … Bare Print and PrintLn no longer crash, and commands written without required square brackets report "Wrong parameter count". ShowBusy removed. The analysis package ships settings for an electron (Im1) → ion (Im2, 0–20 µs) coincidence, so Load shows both images and the TOF spectrum at once. This manual was rewritten with a quick start, rules and pitfalls, and the shipped Imports. |
| 3.3 | 3–16 October 2026 | Imports (library scripts named on the first line) and subroutines with parameters. Export to PIMP: the Imports Export2D, Export2DSLS1, Export2DSLS2 and Export2DMeas write export folders (index file and center, ring and all TOF spectra), also during a measurement (pseudo-live export). Output publication and repeatable output (Output modes, FlushOutput), and EnsureDirectory, FileExists, LastMeasuredFile, DayRelativeSample, CurrFullSample and ROIRingAsCircle. ScriptImport installs the analysis Imports into any script.dat. SetSlit. The analysis program runs without channels.dat. |
| 3.2 | 31 August 2026 | SLS 2.0 support: the SLS2 EPICS channel table, i2d file version 4, which stores SLS2 beamline snapshots, and loaded beamline parameters looked up by channel ID; the SLS1 channel table is kept for historical files. Polygon and cross ROIs (ROIPolygon, ROICross) and Not. Count rates in the beamline parameter (EPICS) window. |
| 3.1 and earlier | up to July 2026 | Not described here. |