System Description,

Data Format Description

&

Command Reference

for the mesytec psd+ and MDLL system

Version 1.10

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Table of contents:

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Overview

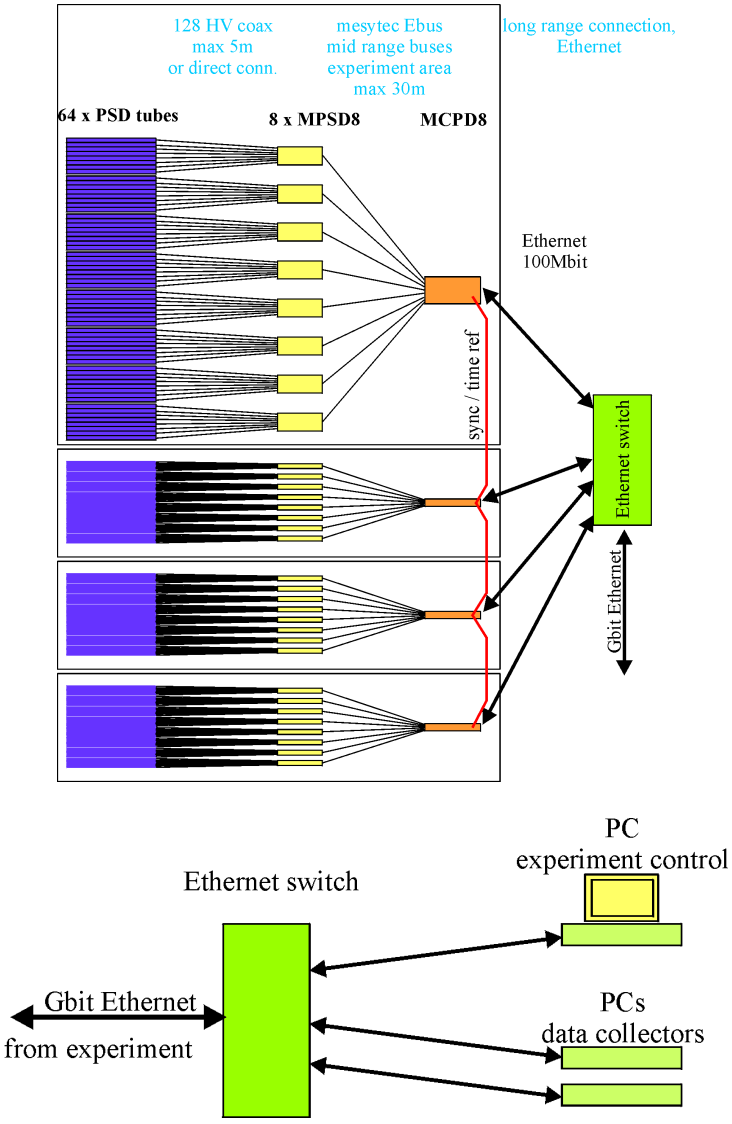
System Layout

mesytec psd+ is a readout system for experiments with medium to large numbers of position sensitive neutron detectors. It provides highly precise TOF timing and allows the integration of trigger sources like monitor counters, chopper signals, …
All events carry a 48 bit wide timestamp with 100 ns timing resolution.

One ore more central units collect all neutron data as well as auxiliary signals from monitor counters, chopper systems and environmental sensors.

The peripheral units, responsible for (pre-)amplification, position calculation of the neutron events as well as for basic testing can be controlled remotely.

The following picture shows a general setup as an example:



Up to eight position sensitive detector tubes are connected to one peripheral module

MPSD-8+. Calculated event positions are transmitted on a point to point bus connection to the central modules MCPD-8. Each MCPD-8 can serve up to 8 peripherals = 64 detectors.

Data are collected and buffered in the MCPD-8 and transmitted to one or more control and DAQ computers. Communication uses UDP/IP on a 100 Mbit/s Ethernet connection.

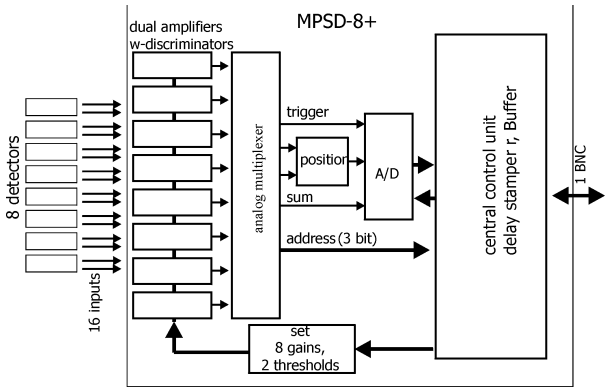
Main Components

mesytec psd+ systems consist of three principal components:

- **"Peripheral modules" like MPSD-8+**

  They interface to the detectors, do the analog signal processing, calculate energies and positions, generate digital data on positions and a differential timing. They have several parameters (gains, thresholds, test pulsers …) that can be controlled remotely.

  Data communication takes place on a dedicated high speed serial interface between peripheral and central modules.

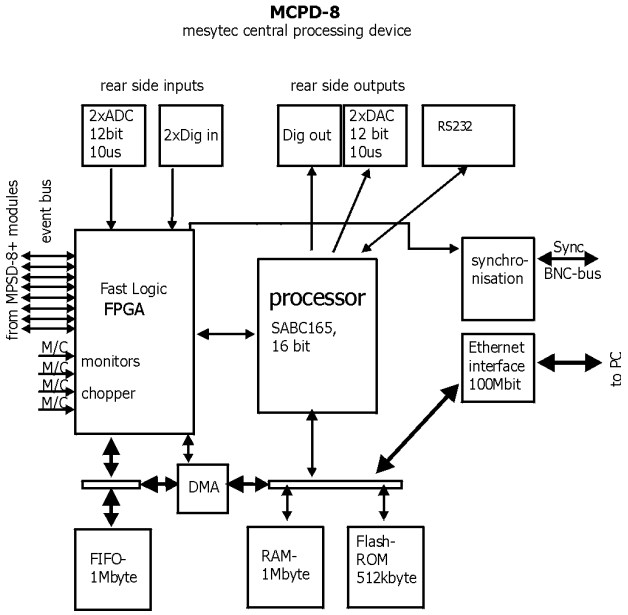


- **Central module MCPD-8**

  Are responsible for data buffering, control of and communication with peripheral modules. They are the interface to the data acquisition and control computer(s).

  Data communication takes place on the high speed serial interface towards the peripheral modules and on 100 Mbit/s Ethernet towards daq computers.

  Communication with the MCPD-8 via UDP/IP protocol is the way to control all system parameters.



- **Data acquisition and control computers**

  At least one computer is required in order to control the psd+ system and to collect the data generated by the peripheral modules.
  Communication takes place between computer and central module, allowing also to control the peripheral module parameters.

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Possible Setups

The range of possible setups stretches from the minimum system, with one peripheral module MPSD-8+, one central module MCPD-8 and one pc to large systems with multiple fully equipped MCPD-8 branches – each serving 64 detectors on 8 MPSD-8+.

One MCPD-8 can handle up to eight peripheral modules. Communication takes place on a deterministic point to point connection, allowing highest data rates. This setup is recommended for instruments with high event rates and demand for lowest possible deadtimes.

Several MCPD-8 can be combined to realize a readout system meeting the requirements regarding number of detectors, data rates and deadtime limits.

At least one data acquisition and control computer is needed to perform three main tasks:
- control peripheral settings (like gains, thresholds and test pulser)
- collect and store incoming raw data
- preprocess / display event data for online monitoring

In larger setups with high data rates, it can be a good decision to split the tasks, one computer for control and online monitoring plus separate computers for data collection and storing may be a good distribution.

Communication protocols

There are two protocols used among the mesytec psd+ system:

- A proprietary protocol on the high speed serial interfaces from peripheral to central modules. MCPD-8 serves as an interface between the control pc and the bus protocol.

- A protocol using the standardized UDP/IP protocol family on Ethernet for the communication between central modules and computers. This protocol will be called “mesytec protocol” in the following.

mesytec Protocol

Communication between daq / control computers and the central modules MCPD-8 is based on the standardized UDP / IP protocol family.

There are two principle kinds of communications in a psd+ system:

- The MCPD-8 modules receive and answer command buffers. Usually commands are emitted by a single control PC. Each command is echoed to the commanding computer, delivering a command response and a status description as well as a variable amount of information.

- During data acquisition, the MCPD-8 modules will emit data packages autonomously. The address of the recipient is set up during the initialisation process of the system. Thus it is possible to send data packages and command answers to different recipients.

  For experiments with high data rates, it might be useful to have data taking distributed on more than one pc.

Both types of communication packets are wrapped in UDP packets.

Using UDP communication allows efficient data transmission without too much protocol overhead. But it has to be remembered that there's no intrinsic protection against packet loss (like in TCP e. g.) Thus some simple control mechanisms have been implemented into the mesytec protocol in order to achieve sufficient data security.

Data Buffers

During data acquisition, the MCPD modules transmit a continuous stream of event buffers.

Each event buffer consists of a buffer header (20 x 16 bit = 40 bytes) and a variable number of events. Each event has a length of 48 bits = 6 bytes.

The total length of an event buffer varies between 40 bytes (header only) up to 1.500 bytes (limited by unfragmented Ethernet frame length).

Buffer Structure:
(in 16 bit words):

Buffer Length (in 16 bit words)		Word 0
Buffer Type		
Header Length (in 16 bit words)		
Buffer Number		
Run-ID		
MCPD-ID	Status	
Header Timestamp Lo		
Header Timestamp Mid		
Header Timestamp Hi		
Parameter 0 Lo		
Parameter 0 Mid		
Parameter 0 Hi		
Parameter 1 Lo		
Parameter 1 Mid		
Parameter 1 Hi		
Parameter 2 Lo		
Parameter 2 Mid		
Parameter 2 Hi		
Parameter 3 Lo		
Parameter 3 Mid		
Parameter 3 Hi		Word 20
Event 0 Lo		Word 21
Event 0 Mid		
Event 0 Hi		

.
.
.

Event n Lo		
Event n Mid		
Event n Hi		Word 20+3*n

(must be identical with buffer length – 1)

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Header data dictionary:

Buffer Type: 16 bit type descriptor.
Bits 0 … 14 carry a version information.
Bit 15 = 0: data buffer
(Bit 15 = 1: command buffer)

Header Length: Length of header information in 16 bit words

Buffer Number: Simple 16 bit counter for data buffers. Incremented automatically by MCPD-8 to allow for loss monitoring.

Buffer Length: total length in multiple of 16 bit words, stretching from "buffer type" to last data word.

Run ID: Simple 16 bit run counter, set by software, to allow for integrity Control.

MCPD-ID: 8 bit ID of sending MCPD module, to be assigned during hardware initialisation.

Status: 8 bit wide bit field for sync and start/stop status of the sending MCPD-8
currently only bits 0, 1 used:
bit 0: 1 = DAQ running, 0 = DAQ stopped
bit 1: 1 = sync o.k., 0 = sync error

Header Timestamp: 48 bits current status of the synchronized system timer (100 ns timing resolution)
Represents the value of the system timer at the moment of buffer opening.
All events in a buffer carry a positive 19 bit offset timing information relative to this header timestamp.

Parameter 0 … 3: MCPD-8 allows the transmission of selected counter / input values with each buffer. So Parameter 0 … 3 represent the values of the selected counters / inputs at the moment of buffer opening (t = header timestamp)
Also the digital input status and ADC values can be mapped here.

Event 0 … n: (n+1) x 48 bit event information
Event structure is explained in detail below.

Event structure:

Each event has a fixed 48 bit length. The contents differs according to the event id.

ID = 0: Neutron data event
ID = 1: Trigger data event

Neutron data events (ID = 0):

MSB LSB

ID (1 bit) = **0**	ModID (3 bit)	SlotID (5)	Amplitude(10)	Position (10)	Timestamp (19)

ID: ID = 0 signalling a "neutron" event , resulting from a detector event at a peripheral modules like MPSD-8.
(Monitor counter events e.g., that of course also are "neutron events" are generated at the MCPD-8, don't carry a position information and are therefore regarded as "other events" in this context.)
1 bit

ModID: Module ID of MPSD-8, determined by serial bus number (bus input at MCPD-8)
3 bit

Slot ID: channel (slot) number in the MPSD module:
[0…7] for MPSD-8 and MSTD-16
5 bit (but 2 MSBs not used, only [2...0] are valid!)

Amplitude: amplitude (energy) of the neutron event if protocol TPA is selected, otherwise = 0
MPSD-8+: 10 valid bits
MPSD-8: 8 valid bits, bits 0, 1 = 0!
MSTD-16+: 9 valid bits [8...0]

Position: position of the neutron event
10 bit

Timestamp: timing offset to the corresponding header timestamp
event time = header timestamp + event timestamp
19 bit

Address reconstruction:

The complete, two dimensional address of a neutron event consists of max. 16 + 10 bit and is composed by the following partial information:

Channel (= individual detector tube):

MCPD-ID:	MCPD- branch, if multiple	8 bit
ModID:	Bus number on identified MCPD-8	3 bit
SlotId:	Subchannel within identified MPSD-8	5 bit

Bit 15 … 8	Bit 7 … 5	Bit 4 … 0
MCPD-ID	ModID	SlotId

A system using only MPSD-8 can reduce the address length further:

- ModID has only 3 valid bits
- MCPD-ID normally doesn't use the full 8 bit range (but is due to users definition!)

So a reduced calculated address format for a system using 4 MCPD-8 and a total maximum of 4 (MCPD-8) x 8 (MPSD-8) x 8 (Detectors/MPSD-8) = 256 detectors could look like this:

Bit 7 … 6	Bit 5 … 3	Bit 2 … 0
MCPD-ID	ModID	SlotId

Position (= event position on identified tube):

Data: 10 bit position data along detector tube.
No calculation needed (but possibly calibration against physical data…)

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Trigger events (ID = 1):

Several trigger sources (counters, timers, digital inputs) can initiate a data taking event. Triggered by levels or defined overflows of the trigger sources, data are taken from the data sources and written to an event structure. Possible trigger and data sources are timers, counters, and ADC values.

MSB LSB

ID (1 bit) = **1**	TrigID (3 bit)	DataID (4)	Data (21 bit)	Timestamp (19)

ID: ID = 1 signalling a "not neutron" event
(= generated centrally in MCPD-8)
possible trigger and data sources are:
- Counters
- Timers
- Digital inputs
1 bit

TrigID: Trigger ID characterizing the event trigger source
1 … 4: Timer 1 … 4
5, 6: rear TTL input 1, 2
7: compare register

DataID: DataID characterizing the data source.
Data taking was initiated by the trigger source identified in TrigID, at the time "header timestamp + event timestamp"
0 … 3: Monitor / Chopper input 1 … 4
4, 5: rear TTL input 1, 2
6, 7: ADC 1, 2

Data: Counter, Timer or ADC value of the identified data source
21 bit (depending on source not necessarily all valid)

Timestamp: timing offset to the corresponding header timestamp
event time = header timestamp + event timestamp
19 bit

Command Buffers

Each command buffer consists of a buffer header (9 x 16 bit = 18 bytes) and a trailing data block of variable length. The contents of the data block depends on the individual commands.

The total length of a command buffer varies between 18 bytes (header only) up to 1.500 bytes (limited by unfragmented Ethernet frame length)
(Padding bytes to keep Ethernet minimum buffer sizes must be added).

Buffer Structure:
(in 16 bit words):

Buffer Length (in 16 bit words)		Word 0
Buffer Type		
Header Length (in 16 bit words)		
Buffer Number		
Cmd		
MCPD-ID	Status	
Header Timestamp Lo		
Header Timestamp Mid		
Header Timestamp Hi		Word 8
Command Checksum		Word 9
Data 0		Word 10
Data 1		
Data 2		

…

Trailing data	
with	
Variable length	Word (buffer length –1)

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Header data dictionary:

Use and meaning of header data varies depending on sender. While MCPD-8 fills in all data, some of them may be left uninitialized when sending a data block from a control pc to a MCPD-8:

Buffer Type: 16 bit type descriptor
Bits 0 … 14 carry a version information and may be left blank in buffers sent by control pc.
Bit 15 = 0: data buffer
Bit 15 = 1: command buffer

Header Length: Length of header information in 16 bit words

Buffer Number: Simple 16 bit counter to allow loss monitoring.
Separate counters for data and cmd buffers.
A control software could increment with each cmd issued.
MCPD-8 will increment its own counter with each cmd answered.

Buffer Length: In multiple of 16 bit words. Spans from "Buffer Type" to last data word. Only counts useful data words.
Padding bytes added to fulfill minimum Ethernet buffer size will not be counted.

CMD-ID: 16 bit value representing the command that is answered / issued in this buffer. Please see the following chapter for a detailed description of the individual commands.

MCPD-ID: ID of the addressed / sending MCPD module, to be assigned during hardware initialisation

Status: 8 bit wide bit field for sync and start/stop status. Can be left blank in buffers sent by control pc.

Header Timestamp: 48 bits synchronized system timer (100 ns binning) generated shortly before Ethernet transmission. It allows to have a timed log of command communication.
Can be left blank in buffers sent by control pc.

Command Checksum:
16 bit XOR checksum covering all words from "Buffer Type" (Word 0) to last data word (Word buffer length –1).
Set checksum field to 0x0000 before calculation.
Attention:MCPD-8 does not calculate a checksum in its cmd answers!

Functional descriptions:

All commands are used to control properties and behaviour of two main targets:

- The peripheral modules MPSD-8
- The central module(s) MCPD-8

While MPSD-8 has only few properties to control, MCPD-8 is far more complex. Both are controlled by sending UPD based command buffers to MCPD-8. Commands for MPSD-8 are translated in the central module MCPD-8 and transmitted over the serial connections.

MPSD-8:

Each MPSD-8 has eight dual amplification stages. The gain values are kept symmetric internally, so there's one gain value for each MPSD-8 channel.
One common lower threshold for all eight channels allows neutron / gamma discrimination.

A built in test pulser is useful for remote electronics checks without the need for neutron signals. Charge is injected at the preamp input, so the complete amplification, discrimination and position calculation process can be checked.
The pulser signal can set to each of the eight channels and can be varied in amplitude and position.

Thus, a MPSD-8 has the following parameters to be controlled remotely:

Amplification and discrimination unit:

- 8 gain values (8 bit)
- 1 common lower threshold value (8 bit)

Test pulser unit:

- 1 amplitude (8 bit)
- 1 channel within module (0 … 7)
- 1 position within channel (left / middle / right)
- on / off

MCPD-8:

The central processing module MCPD-8 offers the following groups of properties / functional modules:

Address and communication parameters:

Device ID
Each MCPD-8 in a system is given a device ID that is used for all communication. The device ID can be set individually during initialisation and is remembered after power up. It is in the responsibility of the user to keep IDs unique.

Device ip address
IP address of the MCPD-8. Also stored permanently, can be set to every valid address. A "panic button" inside the MCPD-8 allows to reset the address to a default value "192.168.168.121".

Data host ip address
MCPD-8 is able to send data buffers to a dedicated DAQ ip address to split up data taking and control tasks. This address can be preset and is remembered after power up.

UDP port numbers
UDP port number for command and data buffers can be set to a desired value. Values are saved permanently. If not set, the port number of the last cmd packet is used.

Timing parameters:

Timing master
A flag to define whether a device operates as timing master or slave.

48 bit master clock
Master timing register, can be set to any value and is incremented every 100 ns during data acquisition.

Termination of sync line
Timing sync line has to be terminated at the end, so the last MCPD-8 has to be set to sync "on".

General properties:

Run id
An arbitrary 16 bit value to identify the current run. Value is transmitted in every data buffer header. Run ID value of the master MCPD-8 will be propagated via sync bus.

Functional units:

Besides the aggregation of neutron event data, MCPD-8 can contribute own data originating from six digital (TTL) and two analog inputs.
To process these external signals, it provides functional units for triggering, counting and AD/DA conversion.

Timer, counter, capture units, ports:

An MCPD-8 offers a total of:

- four auxiliary timers (16 bit wide, 10 us time base, 655 ms max. period)
- six triggerable counter cells (21 bit wide)
- two triggerable ADC cells: 12 bit, +/-4,5 V (jumper closed) or 0...9V (open)
- four multi-purpose counters (48 bit wide).
- Two DAC ports: 12 bit, +/- 3 V (jumper closed) or 0…6 V (open)
- One RS-232 serial port (default configuration 9.600, 8N1) can be used to control / read out external devices (HV sources, sample environment, …). Strings can be sent and answers read back.

Auxiliary timers:

Besides the central TOF timing unit, the MCPD-8 has four auxiliary timer units that are intended as trigger sources for auxiliary event triggering.
Each auxiliary timer is a possible trigger source for one of the six counter cells or one of the two ADC cells.
Triggering a counter or an ADC cell leads to the generation of a trigger event. A trigger event is a timestamped entry in data stream, comparable to a neutron data event. It carries the according counter value or ADC value together with the precise 19 bit timestamp. Auxiliary event data format is described above.

The auxiliary timers are reset automatically at DAQ start or reset. They have a time base of 10 us and a width of 16 bit, leading to a maximum period of 655,36 ms.

A capture register associated with each timer defines the value at which the timer unit triggers the associated event(s).
Every time the capture register equals the timer, a trigger is generated and the timer is reset. Thus it is possible to generate a trigger with a period between 10 us and 655.36 ms.

Timers are assigned to their data sources in the counter/ADC control registers. One timer can trigger more than one data source.

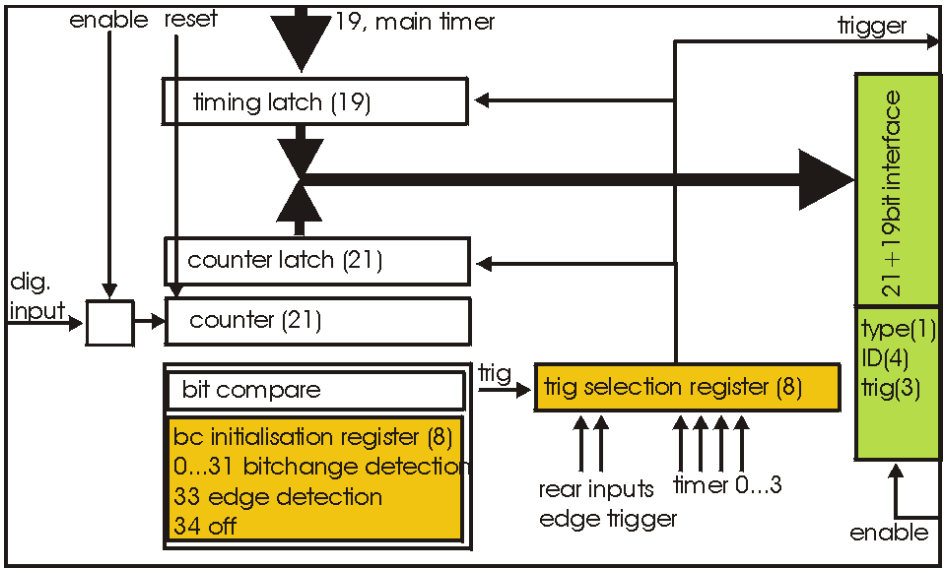
Auxiliary timers do not stop at DAQ stop. This enables e.g. a continuous transfer of ADC data when waiting for a change in sample environment.

Auxiliary timer configuration is done by command #11 (set auxiliary timer), the only parameter to submit is the capture register value.

Counter cells:

Counter cells can be used e.g. for monitor counters, chopper inputs, auxiliary timing purposes and more.
They can be used as simple counters and are also able to generate timestamped events within the data stream.

They operate like shown schematically below:



Inputs:

Each counter cell is driven by one dedicated TTL input.
The assignment of input and counter cell is defined by hardware as follows:

Cell No.	Input
0	Monitor / Chopper 1
1	Monitor / Chopper 2
2	Monitor / Chopper 3
3	Monitor / Chopper 4
4	Dig. Input 1
5	Dig. Input 2

4 x Monitor / Chopper on the frontpanel
2 x Digital Input on the backpanel

Counters:

Each cell has a 21 bit counter cell which increments on the rising edge of the TTL input signal. There are two possibilities to read out the counters:

- Counter values can be copied into each transmitted data buffer header by defining them as a source for a so called “Parameter”.
  (Please refer to the data buffer header description, as well as to the parameter command description for details.)
  Thus they are transmitted whenever a data buffer is transmitted (which is every 40 ms minimum).
  Parameters are latched at the time of buffer creation, so their values have the same timestamp as the header.
  This operating mode is intended e.g. for continuous transmission of counter values, where a precise readout frequency or single event detection is not the issue. (e. g. monitor counters)

- Counters can be configured to emit a timestamped trigger event, based on several trigger sources. When triggered, the latched values of the main timer (19 bit) and the counter (21 bit) are – together with some operational information – written into a trigger event like described above. The trigger event is then immediately buffered within the current data buffer.
  This operating mode is intended for counter sources where detection and timestamping of single events (e.g. choppers), counter overflows (stop on monitor counts, ...) or a precise readout timing is required.

Both operating modes can be combined (e.g. configuring a monitor counter as a parameter source *and* as a trigger event source.

Triggering:

Event trigger sources for a counter cell can be:

Trigger ID	Trigger source
0	No Trigger (only counting)
1	Aux Timer 1
2	Aux Timer 2
3	Aux Timer 3
4	Aux Timer 4
5	Dig Input 1 (rear panel)
6	Dig Input 2 (rear panel)
7	Compare Register (allows also self triggering)

Using one of the Aux Timers as trigger source will lead to generation of trigger events with a frequency defined by the Aux Timer.

Digital Inputs will count and trigger on the rising edge of the TTL input signal.

Triggering by Compare register has three different operating modes:

- writing a value from 0 to 20 triggers whenever the bit specified by the given value becomes “1”. For example: a compare register value of “0” will lead to triggering every second count, “2” triggers every eight counts ...
- a compare register value of 21 triggers on counter overflow
- a compare register value of 22 (as a special value) triggers on every rising edge of the input

Triggering on every rising edge for example allows to generate timestamped chopper signals.

Configuring a counter / ADC cell just requires setting two values for the respective cell address:

trigger source:
0: no trigger
1 … 4: trigger on aux timer 1… 4
5, 6: trigger on rising edge at rear input 1, 2
7: trigger from compare register (7 only for counter cells)

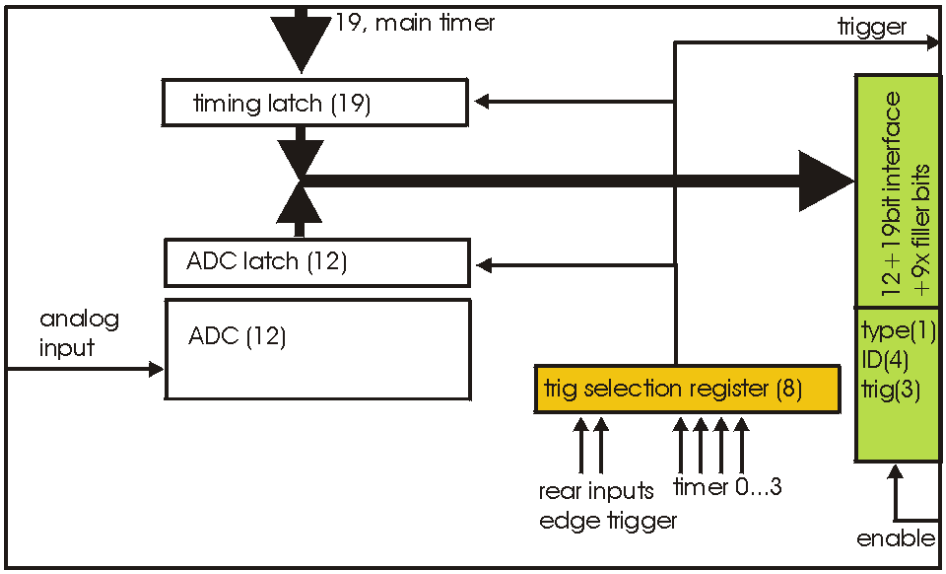
compare register (numerical value n):
0 … 20: trigger on bit n = 1
21: trigger on counter overflow
22: trigger on rising edge of input
(can be left blank for ADC cells)

This setup is done using cmd #9 (Set counter / ADC cell), described in detail below.

ADC cells:

ADC cells operate principally similar to the counter cells. Their data have 12 bit resolution (which are extended by 9 leading zeroes to keep the data format compatible). Full range is +/- 4,5 V or 0 … 9 V, depending on jumper setting inside MCPD-8.

They can be used e.g. for a continuous monitoring of ambient parameters as well as of sample environment. The ADC values are stored in registers at a sampling rate of 25 kHz they can be read out randomly at any time and rate. (although their contents will only change with 25 kHz)



Inputs are the two analog inputs on the backpanel.

The ADC cells can be triggered by:

- one of the four programmable auxilliary timers
- one of the two digital inputs on the backpanel

Multipurpose counters ("Parameters"):

MCPD-8 offers four additional 48 bit wide multipurpose counters / data fields.
Their values are automatically transmitted as parameter 0 … 3 with every data buffer.

Data buffers will be emitted whenever an event buffer is full, but with a minimum frequency of 25 Hz. So the multipurpose counters are a good means for a continuous monitoring of counting inputs.

All six TTL inputs can operate as counter source:
4 x Monitor / Chopper on the frontpanel
2 x Digital Input on the backpanel

Furthermore, a parameter field can carry a copy of one of the following information:

Event counter
Master clock
Current status of all digital inputs and ADCs

Parameter 0 and 1 will stop on a stop command and will continue / update on a continue command. Parameter 2 and 3 will run continuously.

The assignment of parameters is done by command #11 (set parameter source):
Each one of the four possible parameters can be assigned one of the following sources:

0 … 3: Monitor/Chopper inputs 1…4
4, 5: backpanel TTL inputs 1, 2
6: combination of all digital inputs, and both ADC values
7: event counter
8: master clock

Counter / Port Status Polling:

As a last means to derive counter values and port status from MCPD-8, there's a polling command that delivers all possible data sources in one cmd answer buffer:

Parameter 0 … 3
6 digital inputs
2 digital outputs
2 ADC inputs
2 DAC outputs

Application Considerations:

The various functional units of the MCPD-8 allow building a highly flexible system, delivering exactly the data needed by an instrument. Not only neutron data , but also environmental parameters can be recorded in several ways.

As it is not really self explaining how to use the many flexible possibilities the PSD system offers, here are some considerations and hints that might be helpful adapting the readout system to an instrument.

Data Sources:
The PSD system offers three different kinds of data:

- Neutron event data, generated in a MPSD-8 or MSTD-16 module:
  Position and / or Energy information, together with a detector address and a precise event timestamp.

- Counter / TTL event data, generated in MCPD-8:
  Counter values are buffered immediately on triggering together with ID information and a precise timestamp, just like a neutron event.

  Sources:
  All six TTL inputs can each be used to drive a counter.

  Triggers:
  Counter event readout can be triggered from one of four trigger sources:

  - One of four auxiliary timers:
    This allows for repeated readout at fixed intervals (e.g. to get an easy monitor rate calculation.)

  - One of two auxiliary TTL inputs:
    This allows for a readout at a status that may be defined by any external device (e.g. a “ready” signal from sample environment)

  - Increments over a defined bit position of the counter register.

An event will be triggered when this bit toggles from 0 to 1.
E.g. a “0” in compare register (means: “trigger, whenever bit 0 becomes “1”) generates a trigger every second count.
A “2” in compare register (means: “trigger, whenever bit 2 becomes “1”) will trigger every 8 counts, …
This allows reading the counter value at any binary power up to 20.

  - Edge triggering of the TTL input:
    Every rising edge of the TTL input triggers an event.
    This might be useful e.g. to generate precisely timestamped chopper signal events, start or stop events correlated with sample environment, …

- ADC event data:

  Sources:
  Two 12 bit ADCs with a selectable full range of +/-4,5 or 0...9 V are read out continuously with a sampling rate of 25 kHz automatically. The corresponding registers can be read out and transmitted as timestamped ADC events.

  Triggers:
  ADC data readout can be triggered by nearly the same trigger sources as counters described above:

  - One of four auxiliary timers:
    This allows for repeated readout at fixed intervals (e.g. to get a continuous readout of changing environmental parameters like temperature, fields, pressure, ...)

  - One of two auxiliary TTL inputs:
    This allows for a readout at a status that may be defined by any external device (e.g. a "ready" signal from sample environment)

  (There's no self-triggering with a compare register for the ADCs.)

Examples:

Chopper Signals:

Chopper signals need precise timing, so it's a good application for a self triggering TTL input:

- Connect the TTL signal from copper electronics to one of the four Monitor/Chopper inputs or to one of the backpanel TTL inputs.
- Configure the according counter cell using cmd #9 with:
  - cell#: 0-3 for frontpanel, 4,5 for backpanel
  - trigger source = 7 (trigger on compare register)
  - compare register value = 22 (trigger on rising edge of input)

Now each time the TTL input rises, a trigger event is generated, timestamped and buffered with the following data :

TrigID = 7 (compare register)
DataID = n (n=address of input)
Data = counter value (may perhaps be discarded in data processing for chopper pulses)

If a divider for chopper pulses is required, the compare register can be set to values other than 22: a „0" will lead to triggering every second rising edge, a „1" every fourth", ...

Monitor Counters:

Monitor counters normally don't need precise timing with a 100 ns timebase. So there are several possibilites for monitor counter readout – depending on instrumental needs:

Connect the monitor counting output (TTL) to one of the four Monitor/Chopper inputs or to one of the backpanel TTL inputs.

Monitor counter as header parameter:

Define the appropriate counter (0...3: frontpanel, 4,5: backpanel) as source of one of the four parameter fields that will be transmitted with every data buffer header.

Use command # 11 with:

Parameter# = 0...3
Source = 0..3 for frontpanel inputs, 4,5 for backpanel inputs

Now each time a new data buffer is generated, the defined counters will be read into their assigned parameter fields. Timestamp is the header timestamp.
Buffers will be generated and transmitted whenever a buffer is full or at least every 25 ms (40 Hz minimum buffer frequency).

Monitor counter as timer triggered event:

Using one of the auxiliary timers as trigger source, monitor counter values can be transmitted with a constant frequency: timebase for the 16 bit timers are 10 us, the maximum interval is therefore 655,36 ms.

- Configure one of the four auxiliary timers with the appropriate trigger interval, using command #10:
  - timer# = 0...3
  - capture register 0... 65.536 (e.g. 10.000 for 100 ms intervals)

- Configure the counter cell using command #9:
  - cell# = 0..3 for frontpanel inputs, 4,5 for backpanel inputs
  - trigger source = 1...4 for aux timer 0...3
  - compare register: will not be used

Now each time the auxiliary timer reaches the capture value, a trigger event will be generated, timestamped and buffered with the following data :

TrigID = 1...4 (timer#)
DataID = n (n=address of input)
Data = monitor counter value

Monitor counter as counter triggered event:

- Configure the according counter cell using cmd #9 with:
  - cell#: 0-3 for frontpanel, 4,5 for backpanel
  - trigger source = 7 (trigger on compare register)
  - compare register value = 0...21

Now each time the bit position defined in compare register value becomes „1“, a trigger event is generated, timestamped and buffered with the following data :

TrigID = 7 (compare register)
DataID = n (n=address of input)
Data = counter value (may perhaps be discarded in data processing for chopper pulses)

This allows reading out monitor counters at any power of from 2^1 up to 2^{22}

Monitor counter as edge triggered event:

Identical to chopper signals described above.

External analog data, timer triggered:

Many instruments need information from other components of the setup: temperature, pressure, fields, …
Often these values can be read out by an analog voltage signal. MCPD-8 allows to digitize such signals and fill the according values into the data stream.

Just like described above for monitor counters, also the ADC readout can be triggered by auxiliary timers:

- Configure one of the four auxiliary timers with the appropriate trigger interval, using command #10:
  - timer# = 0...3
  - capture register 0... 65.536 (e.g. 10.000 for 100 ms intervals)

- Configure the ADC cell using command #9:
  - cell# = 6,7 for ADC input 1, 2
  - trigger source = 1...4 for aux timer 0...3
  - compare register: will not be used

Now each time the auxiliary timer reaches the capture value, a trigger event will be generated, timestamped and buffered with the following data :

TrigID = 1...4 (timer#)
DataID = n (n=address of ADC input)
Data = ADC value (12 bit valid data, 9 bit leading zeroes)

External analog data, TTL triggered:

ADC readout can also be triggered by an axternal TTL signal, which can be an end point switch, an external frequency source, …

- Configure the ADC cell using command #9:
  - cell# = 6,7 for ADC input 1, 2
  - trigger source = 5,6 for backpanel TTL inputs 1,2
  - compare register: will not be used

Now on each rising edge of the assigned backpanel TTL input a trigger event will be generated, timestamped and buffered with the following data :

TrigID = 5, 6 (TTL input)
DataID = n (n=address of ADC input)
Data = ADC value (12 bit valid data, 9 bit leading zeroes)

External analog data, continuously transmitted with buffer header:

If external data just have to be monitored, but precise timing is not the issue, data transmission in one of the four parameter fields of the buffer header may be a solution:

Use command #11 to set up the selected parameter field for transmission of the ADC values: Source = 6 will transmit a combination of both TTL input levels and both ADC values in the selected parameter field.

Buffers are transmitted whenever they are full with events or latest every 40 ms. Values and timestamps are generated at the moment of buffer generation.

Combination of timers, triggering, counting and header parameters:

For a sophisticated setup, most of the readout possiblities can be combined.

For example monitor counters:
It may be convenient to have edge triggered, timestamped monitor counter events in the data stream to allow a count-precise offline data processing from listmode files.
But it is time consuming to evaluate every event online to search for a stop condition for data taking. For this purpose it's mostly sufficient to have the monitor counter as one of the four header parameters to generate a stop condition.

Same for external parameters:
It might be necessary to have a temperature/field/pressure profile with a resolution of some hundred microseconds in offline data analysis. So a timer triggered event generation with the necessary frequency is a good idea.
But for online status monitoring, the ADC values as one of the header parameters, transmitted every 40 ms latest, should be more than enough in most cases.
So just configure the data source for triggering as well as for parameter transmission:

e.g. the monitor counter:

- use cmd # 9 to set up the appropriate counter cell to trigger an event on every rising edge of the input
- use cmd # 11 to define the same counter as source for one of the parameter fields

Thus the monitor counter value will be transmitted as an event on each rising edge (to be used in offline data processing if necessary) AND it will be transmitted with every buffer header (e.g. for economical generation of stop conditions).

Command Reference

General Command Format:

The structure of a command buffer is displayed here again. It's layout is identical for all commands.

Buffer Type		Word 0
Header Length (in 16 bit words)		
Buffer Number		
Buffer Length (in 16 bit words)		
Cmd		
MCPD-ID	Status	
Header Timestamp Lo		
Header Timestamp Mid		
Header Timestamp Hi		
Command Checksum		Word 9
Data 0		Word 10
Data 1		
Data 2		

…

Trailing data	
with	
Variable length	Word (buffer length –1)

In the following, only data from Word 10 on (Data 0) are displayed. The given command number has to be entered in the header field "Cmd" (Word 4)

Each command buffer has a trailing 0xFFFF as last word.

Command answer:
Every command buffer will be answered by MCPD-8. Set values instead of requested values will be inserted into the appropriate fields. If a command fails in the MCPD-8, the Cmd number will have bit 15 set.

Command Set MCPD-8:

Command List (Numerical Order):

Cmd #	Command
0	Reset
1	Start DAQ
2	Stop DAQ
3	Continue DAQ
4	Set MCPD-ID
5	Set Protocol Parameters
6	Set MCPD-8 Timing Setup
7	Set Master Clock
8	Set Run ID
9	Set Counter / ADC cell
10	Set Auxiliary Timer
11	Set Parameter Source
12	Get Parameters
13	Set MPSD-8 Gain
14	Set MPSD-8 Threshold
15	Set MPSD-8 Pulser
16	Set MPSD-8 Mode
17	Set MCPD-8 DAC
18	Send MCPD-8 Serial String
19	Read MCPD-8 Serial String
21	Set MCPD-8 TTL Outputs
22	Read MCPD-8 fast tx capabilities
23	Set MCPD-8 fast tx format
24	Read MPSD-8+ parameters
25	Set MPSD-8+ fast tx protocol
51	Retrieve MCPD-8 version information

Command List (Functional Order):

Communication Settings:

Cmd #	Command
4	Set MCPD-ID
5	Set Protocol Parameters
22	Read MCPD-8 fast bus capabilities
23	Set MCPD-8 fast bus format

General MCPD-8 Settings:

Cmd #	Command
6	Set MCPD-8 Timing Setup
7	Set Master Clock
8	Set Run ID
9	Set Counter / ADC cell
10	Set Auxiliary Timer
11	Set Parameter Source
12	Get Parameters
22	Get MCPD-8 Bus Capabilities
23	Set MCPD-8 Bus Capabilities
51	Get MCPD-8 version information

MPSD-8 Settings:

Cmd #	Command
13	Set MPSD-8 Gain
14	Set MPSD-8 Threshold
15	Set MPSD-8 Pulser
16	Set MPSD-8 Mode
24	Get MPSD-8 Params
25	Set MPSD-8 fast tx mode

DAQ Commands:

Cmd #	Command
0	Reset
1	Start DAQ
2	Stop DAQ
3	Continue DAQ

MCPD-8 Port Commands:

Cmd #	Command
17	Set MCPD-8 DAC
18	Send MCPD-8 Serial String
19	Read MCPD-8 Serial String

Command Descriptions:

Communication Settings:

Set MCPD ID#		**Cmd = 4**
Word	**Contents**	
10	ID (0 … 255)	
11	0xFFFF	

Each MCPD in a setup is given an individual ID number (8 bit). The ID is part of the header of every data / cmd packet the MCPD emits. Thus data can be assigned to a defined part of the readout system during data processing.
It is in the responsibility of the user (= frontend programmer) to keep IDs unique throughout the readout system.

Answer buffer:

Set MCPD ID# (Answer)		**Cmd = 4**
Word	**Contents**	
10	ID	
11	0xFFFF	

Set Protocol Parameters		**Cmd = 5**
Word	**Contents**	
10	MCPD ip 0	(e.g. 192)
11	MCPD ip 1	(e.g. 168)
12	MCPD ip 2	(e.g. 168)
13	MCPD ip 3	(e.g. 121)
14	Data sink ip 0	
15	Data sink ip 1	
16	Data sink ip 2	
17	Data sink ip 3	
18	Cmd UPD Port	
19	Data UPD Port	
20	Cmd pc ip 0	
21	Cmd pc ip 1	
22	Cmd pc ip 2	
23	Cmd pc ip 3	
24	0xFFFF	

MCPD ip:
The IP address of the MCPD can be remotely changed. A hardware button on the CPU module of the MCPD allows a reset to the factory address 192.168.168.121.
(This address is also mentioned as an example above to explain byte order)
MCPD ip address will not be modified if MCPD ip 0 (Word 10) is set to zero.

Data sink ip:
Also the destination ip address for data packages can be set individually. (If no address is set: the address of the cmd pc is used automatically)
Address will not be modified if Data sink ip 0 (Word 14) is set to zero.
If ip0 as well as ip1 are set to 0, the address of the pc sending this command will be used automatically from out of the ip protocol. This allows to set the address to the sending pc without knowing its address explicitly.

Cmd pc ip:
This allows to set a defined address for the pc that will send the cmds. No other pc will then be able to take control over the system unless the new address is published by the current cmd pc.
Address will not be modified if Cmd pc ip 0 (Word 20) is set to zero.
If Cmd pc ip0 as well as Cmd Pc ip1 are set to 0, the address of the pc sending this command will be used automatically from out of the ip protocol. This allows to set the address to the sending pc without knowing its address explicitly.

UDP ports
MCPD-8 is able to use individually set UPD port numbers, possibly different ones for cmd and data. No change if fields are set to zero.

The following table gives an overview of the possible settings:

Field(s)	Value	Meaning
MCPD ip0	0	Do not change MCPD ip address
	> 0	Set MCPD ip address to values in word 10 ... 13
Data sink ip0	0	Do not change Data sink ip address
	> 0	Set data sink ip address to values in word 14 ... 17
Data sink ip0 *and* Data sink ip1	0	Set data sink ip address to address of cmd sending pc
Cmd pc ip0	0	Do not change Data sink ip address
	> 0	Set cmd pc ip address to values in word 20 ... 23
Cmd pc ip0 *and* Cmd pc ip1	0	Set cmd pc ip address to address of cmd sending pc
Udp port	0	Do not modify
	> 0	Set to given value

Get MCPD-8 tx capabilites	Cmd = 22
Word	**Contents**
10	0xFFFF

Reads possible eventbus data transmission formats of MCPD-8:

Answer buffer:

Get MCPD-8 tx cap. (Answer)	Cmd = 22
Word	**Contents**
10	Capabilites bitmap
11	Current setting
12	0xFFFF

With bitmap:

7	6	5	4	3	2	1	0
x	x	x	x	x	TOF+POS+AMP	TOF + Pos/Amp	Pos/Amp

Current setting = one of the possibilites selected:
1 = Pos/Amp
2 = TOF + Pos/Amp
4 = TOF + Pos + Amp

Set MCPD-8 tx capabilites	Cmd = 23
Word	**Contents**
10	Fast TX format
11	0xFFFF

Sets eventbus fast TX format for all MCPD-8 buses to:
1 = Position / Amplitude
2 = TOF + Position / Amplitude
4 = TOF + Position + Amplitude

Answer buffer:

Set MCPD-8 tx cap. (Answer)	Cmd = 23
Word	**Contents**
10	Current setting
11	0xFFFF

General settings:

Set MCPD-8 timing setup	**Cmd = 6**
Word	**Contents**
10	Timing / Sync master (0: MCPD is slave, 1: MCPD is master)
11	Sync bus termination (0 = on, 1 = off)
12	0xFFFF

Sets timing properties:
- Please make sure that only one MCPD-8 is set as sync master!
- Sync bus has to be terminated at both ends – master is terminated automatically, last slave on bus has to be terminated.

Set MCPD-8 master clock	**Cmd = 7**
Word	**Contents**
10	Master clock, bits 0 … 15
11	Master clock, bits 16 … 31
12	Master clock, bits 32 …47
13	0xFFFF

Master clock can be set to any value if desired. Normally, a reset is initiated before a new run and all counters are set to zero during this reset automatically.
Only if another run start time than zero is desired, this registers must be set.

Set Run Id	**Cmd = 8**
Word	**Contents**
10	Run Id
11	0xFFFF

Set value for the header field “Run ID”
Can be set to any desired value.
The master MCPD-8 distributes its Run ID over the sync bus. Thus it’s only necessary to set the Run Id at the master module.

Counter, ADC, Timer and Parameter settings:

Set Counter / ADC Cell	**Cmd = 9**
Word	**Contents**
10	Cell #: 0 … 3: monitor / chopper inputs 1…4 4, 5: dig. backpanel inputs 1, 2 6, 7: ADC 1, 2
11	Trigger source: 0 = no trigger 1 … 4: trigger on aux timer 1… 4 5, 6: trigger on rising edge at rear input 1, 2 7: trigger from compare register (7 only for counter cells)
12	Compare register, numerical value n: 0 … 20: trigger on bit n = 1 21: trigger on counter overflow 22: trigger on rising edge of input (can be left blank for ADC cells)
13	0xFFFF

This command configures the given counter cell:

One of six possible cells is addressed. The value of the according 21 bit counter is transmitted as a trigger event when triggered.

Trigger source can be one of the digital inputs, one of the four auxiliary timers or a special compare register.
Please note that the compare register does not do a full compare, but checks for a '1' at the given bit position, allowing for triggers at multiples of 2.

Counter cells are intended to generate repeated trigger events. They can be used e.g. for a continuous monitoring of counter values and ADC inputs.

Choosing the rising signal edge as trigger source enables to generate a (fully timestamped) event e.g. for each chopper signal and allows precise chopper timing calculation.

Set Auxiliary Timer	Cmd = 10
Word	**Contents**
10	Timer #: (0 … 3)
11	Capture register: (0 … 65.536) Time base is 10 us, allowing for intervals from 10 us to 655,36 ms
12	0xFFFF

Auxiliary timer compare register is set to the given value.
An identical compare generates a trigger signal (that might be used in one of the counter / ADC cells) and resets the counter to zero. Thus four independent triggers with periods between 10 us and 655,36 ms are possible.

Set Parameter Source	Cmd = 11
Word	**Contents**
10	Parameter: (0 … 3)
11	Source: 0 … 3: Monitor/Chopper inputs 1…4 4, 5: backpanel TTL inputs 1, 2 6: combination of all digital inputs, and both ADC values 7: event counter 8: master clock
12	0xFFFF

Defines the counter source for the given parameter.
While 0 … 5 are real counters, 6 delivers a combination of the current status of all defined inputs and 7, 8 get copies of the current value of event counter or master clock.

All four Parameter values are transmitted with every data buffer, delivering a continuous monitoring information.

Get All Parameters	Cmd = 12
Word	**Contents**
10	0xFFFF

Requests all available parameter information.

Answer buffer:

Get all Parameters (Answer)	Cmd = 12
Word	**Contents**
10	ADC 1 (12 valid bits)
11	ADC 2 (12 valid bits)
12	DAC 1 (12 bits)
13	DAC 2 (12 bits)
14	TTL outputs (2 bits)
15	TTL inputs (6 bits)
16	Event counter Lo
17	Event counter Mid
18	Event counter Hi
19	Parameter 0 Lo
20	Parameter 0 Mid
21	Parameter 0 Hi
22	Parameter 1 Lo
23	Parameter 1 Mid
24	Parameter 1 Hi
23	Parameter 2 Lo
26	Parameter 2 Mid
28	Parameter 2 Hi
29	Parameter 3 Lo
30	Parameter 3 Mid
31	Parameter 3 Hi
32	0xFFFF

Gathers the given information.

MCPD-8 port commands

Set MCPD-8 DAC	**Cmd = 17**
Word	**Contents**
10	DAC 0 (12 valid bits)
11	DAC 1 (12 valid bits)
12	0xFFFF

MCPD-8 offers two DAC ports that can be set in a 12 bit range.
Full range output voltage is +/- 3V or 0…6 V, according to jumper setting in MCPD-8.

Send MCPD-8 serial string	**Cmd = 18**
Word	**Contents**
10	Total length of string to send (including carriage return, linefeed, …)
11	First character to send
10+len	Last character to send
10 + len + 1	0xFFFF

MCPD-8 offers a serial RS-232 interface that can be used to control other devices.
Port settings are 9600 Bd, 8N1 by default.
The given string is sent to the serial interface. Answers are collected in a buffer that can be retrieved with the following command.

Read MCPD-8 serial string	**Cmd = 19**
Word	**Contents**
10	0xFFFF

Requests a readout of the serial buffer.
Answer buffers look like follows:

Read MCPD-8 serial string (Answer)	**Cmd = 19**
Word	**Contents**
10	Total length of string to send (including carriage return, linefeed, …)
11	First character in serial buffer
10+len	Last character in serial buffer
10 + len + 1	0xFFFF

Read MCPD-8 software versions	**Cmd = 51**
Word	**Contents**
10	0xFFFF

Returns version information of MCPD-8 microcontroller and FPGA firmware.

Answer buffers look like follows:

Read MCPD-8 version status (Answer)		**Cmd = 51**
Word	**Contents**	
10	Major CPU software version	
11	Minor CPU software version	
12	Maj. FPGA ver.	Min. FPGA ver.
13	0xFFFF	

MPSD-8 commands

Set MPSD Gain	**Cmd = 13**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Channel within MPSD (0 … 7, 8 = all)
12	Gain value (0 … 255)
13	0xFFFF

Each channel gain can be set individually. To facilitate a quick setup, using channel number 8 will write the same gain value to all channels of the addressed MPSD-8 module.

Set MPSD Threshold	**Cmd = 14**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Threshold value (0 … 255)
12	0xFFFF

Each peripheral module MPSD-8 has one common lower threshold for its window discriminator. An 8 bit value is used to set the lower discriminator threshold.

Set MPSD Pulser	**Cmd = 15**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Channel within MPSD (0 … 7)
12	Position within channel (0 = left, 1 = right, 2 = middle)
13	Pulser amplitude (0 … 255)
14	Pulser on/off (0 = off, 1 = on)
15	0xFFFF

A builtin test pulser is useful to check electronics performance without the need of “real” neutron events.

The pulser can be set to 3 positions (left, middle, right) in a psd channel. Furthermore, the pulser amplitude can be controlled and pulser function can be switched on/off.
Be sure to switch all pulsers off before starting neutron recording!

Set MPSD-8 Mode	**Cmd = 16**
Word	**Contents**
10	MPSD device (=bus) number (0 … 7, 8 = all)
11	Mode (0 = position, 1 = amplitude)
12	0xFFFF

MPSD-8 can be run in two modes:
Position mode transmits a 10 bit position information.
Amplitude (Energy) mode transmits a 10 bit signal amplitude information.

New versions of MPSD-8+ are capable transmitting position and amplitude date simultaneously.

Get MPSD Parameters	**Cmd = 24**
Word	**Contents**
10	MPSD device number (0 … 7)
11	0xFFFF

Retrieves contents of MPSD-8 parameter registers.
Answer buffer:

Get MPSD Parameters	**Cmd = 24**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Eventbus transmit capabilities
12	Current eventbus fast tx format setting
13	Firmware revision
14	0xFFF

DAQ commands

Start DAQ	**Cmd = 1**
Word	**Contents**
10	0xFFFF

Start DAQ starts the data acquisition system.
All timers (master timer + auxiliary timers) start / continue running.
Neutron and trigger events will be filled into data buffers.
Start signal is propagated over the sync line. Thus it is not necessary to send a start signal to each individual MCPD-8.
MCPD-8 not set as master will refuse command.

Stop DAQ	**Cmd = 2**
Word	**Contents**
10	0xFFFF

Stop DAQ stops the data acquisition system.
All timers (master timer + auxiliary timers) stop running.
Stop signal is propagated over the sync line. Thus it is not necessary to send a stop signal to each individual MCPD-8.
MCPD-8 not set as master will refuse command.

Continue DAQ	**Cmd = 3**
Word	**Contents**
10	0xFFFF

Continue DAQ restarts the data acquisition system.
All timers (master timer + auxiliary timers) will continue running.
Stop signal is propagated over the sync line. Thus it is not necessary to send a stop signal to each individual MCPD-8.
MCPD-8 not set as master will refuse command.

Reset	**Cmd = 0**
Word	**Contents**
10	0xFFFF

Running DAQ will be stopped
All counters and timers will be reset to 0.

Reset signal is propagated over the sync line. Thus it is not necessary to send a reset signal to each individual MCPD-8.
MCPD-8 not set as master will refuse command.

MDLL

Readout module for area detectors with delayline readout

Principle of operation:

MDLL is an integrated readout module for twodimensional detectors with delayline readout. It is a compact combination of analog readout electronics and an MCPD-8 like network interface.

Input Signals:
Five analog detector signals are processed:

- X0, X1
- Y0, Y1
- Anode

Signal processing:
CFDs on X and Y signals as well as on the Anode signal generate start/stop signals for 4 TACs (Time to Analog Converters). These TACs are started by the Anode signal and stop on X / Y signals. Positions are calculated unsing these TAC times.

CFD thresholds for X, Y and Anode are remotely controllable to adjust for optimum noise reduction.

TAC ranges are remotely controllable between 100 and 500 ns – leading to a scaling of positions within this range.

A remotely controllable offset can be applied to TAC data, allowing a shift in position to adjust for geometrical/electronical variations of the detector.

A builtin test pulser (also remotely controllable) allows functional testing without using a detector.

Data processing:
Before generating a valid neutron event, a (remotely controllable) software window discriminator is applied on the timing value for each dimension (X, Y) as well as on energy. This allows filtering of erroneous events caused by multiple hits as well as by unwanted particles.

Each valid event generates a data set comprising either:
X- Position, Y-Position, Energy and Timestamp
(which is the “Standard” operating mode)
or:
X-Timing, Y-Timing, Energy and Timestamp
(which is useful for setup and control)

Data are output as UDP packages on an ethernet interface, data format is comparable to data format of MCPD-8 / MPSD-8. Please see below for a detailed description.

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Overview Main Properties:

MDLL, "detector part":

- direct inputs for 2 x X-cathode, 2 x Y-cathode, 1 x Anode
- X and Y position calculation from delay time
- Energy determination from anode signal
- Adaptable to delays of 100 ns … 500 ns
-

MDLL, "MCPD-8 part":

- same network interface like MCPD-8
- similar digital interface like MCPD-8, but:
  - only 3 digital (TTL-) Inputs for Chopper/Counter/Triggering
- no analog In-/Outputs
- Listmode data transmission, two data modes: [X, Y, E] or [tX, tY, E]
- Data format identical to MCPD-8 / MPSD-8: 8 bit Energy, 2 x 10 bit position

MDLL "detector part": Logical Properties:

- ID: 35
- 3 thresholds: X, Y, Anode
- 2 shift factors: X, Y
- 2 scaling factors: X, Y
- 3 window settings: timing sum X, timing sum Y, Anode Amplitude
- Test pulser settings: on/off, amplitude (4 steps), position (3 fixed positions)
- Data selection: X, Y, E (standard mode), timing sum X, timing sum Y, E (for setup purposes)

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

MDLL Data

MDLL has a slightly different data structure due to the fact that it has a fixed number of position channels (960 x 960).
To signalize this different structure, field „Buffer Type“ in buffer header has a value of 0x0002.

Buffer Structure:
(in 16 bit words):

Buffer Length (in 16 bit words)		Word 0
Buffer Type		
Header Length (in 16 bit words)		
Buffer Number = 0x0002 for MDLL		
Run-ID		
MCPD-ID	Status	
Header Timestamp Lo		
Header Timestamp Mid		
Header Timestamp Hi		
Parameter 0 Lo		
Parameter 0 Mid		
Parameter 0 Hi		
Parameter 1 Lo		
Parameter 1 Mid		
Parameter 1 Hi		
Parameter 2 Lo		
Parameter 2 Mid		
Parameter 2 Hi		
Parameter 3 Lo		
Parameter 3 Mid		
Parameter 3 Hi		Word 20
Event 0 Lo		Word 21
Event 0 Mid		
Event 0 Hi		

•
•
•

Event n Lo	
Event n Mid	
Event n Hi	Word 20+3*n

(must be identical with buffer length – 1)

Event structure MDLL:

Each event has a fixed 48 bit length. The contents differs according to the event id.
As MDLL has – compared to MPSD systems – a fixed number of 960 x 960 channels, transmitted data are slight ly different to MPSD data format:

ID = 0: Neutron data event
ID = 1: Trigger data event

Neutron data events (ID = 0):

MSB LSB

ID (1 bit) = **0**	Amplitude (8)	Y Position (10)	X Position (10)	Timestamp (19)

ID: ID = 0 signalling a "neutron" event , resulting from a detector event at a peripheral modules like MPSD-8.
(Monitor counter events e.g., that of course also are "neutron events" are generated at the MCPD-8, don't carry a position information and are therefore regarded as "other events" in this context.)
1 bit

Amplitude: amplitude (energy) of the neutron event
8 bit

Y Position: y position of the neutron event
10 bit

X Position: x position of the neutron event
10 bit

Timestamp: timing offset to the corresponding header timestamp
event time = header timestamp + event timestamp
19 bit

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Command Set MDLL:

Command List (Numerical Order):

Cmd #	Command
0	Reset
1	Start DAQ
2	Stop DAQ
3	Continue DAQ
4	Set MCPD-ID
5	Set Protocol Parameters
6	Set MCPD-8 Timing Setup
7	Set Master Clock
8	Set Run ID
9	Set Counter / ADC cell
10	Set Auxiliary Timer
11	Set Parameter Source
12	Get Parameters
18	Send MCPD-8 Serial String
19	Read MCPD-8 Serial String
51	Retrieve MCPD-8 version information
60	Set MDLL thresholds
61	Set MDLL spectrum
65	Set MDLL pulser
66	Set MDLL dataset
67	Set MDLL timing window
68	Set MDLL energy window

(new commands, MDLL exclusive)

UDP command set for MDLL
(extended command set for MDLL "detector properties")

Set MDLL Thresholds	**Cmd = 60**
Word	**Contents**
10	Threshold X (0 … 255)
11	Threshold Y (0 … 255)
12	Threshold Anode (0 … 255)
13	0xFFFF

Sets thresholds for the constant fraction discriminators. CFD signals start (Anode) or stop the TAC measurement from which position calculation is derived.

Set MDLL Spectrum	**Cmd = 61**
Word	**Contents**
10	ShiftX (0 … 255)
11	ShiftY (0 … 255)
12	ScaleX (0 … 255)
13	ScaleY (0 … 255)
14	0xFFFF

Sets offset values for the calculated position (ShiftX, ShiftY), shifting the position spectrum in given direction.
Sets scaling values (ScaleX, Scale Y) by defining the TAC max range between 100 ns and 500 ns.

---

mesytec GmbH & Co. KG
wernher-von-braun-str. 1, 85640 putzbrunn, germany
fon: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Set MDLL TX data set	Cmd = 66
Word	**Contents**
10	0 = E, X, Y 1 = E, tsumX, tsumY
11	0xFFFF

Defines which data set will be transmitted:
Standard is X, Y, and E (Anode amplitude).

For setup purposes timingX, timingY and E can be chosen as data set.
(allowing to define the borders for the software discrimination windows with Cmd 67 and Cmd 68 subsequently).

Set MDLL Timing window	Cmd = 67
Word	**Contents**
10	Not used
11	Not used
12	Tsum Limit X low (0 … 1024)
13	Tsum Limit X high (0 … 1024)
14	Tsum Limit Y low (0 … 1024)
15	Tsum Limit Y high (0 … 1024)
16	0xFFFF

Defines a timing window for data acquisition.
Only events matching these software window restrictions are copied into data buffer for transmission.
This allows suppression of multiple hits leading to erroneous delay line timing.

Set MDLL Energy Window	Cmd = 68
Word	**Contents**
10	Lower threshold (0 … 255)
11	Upper threshold (0 … 255)
12	Not used
13	Not used
14	0xFFFF

Defines an energy window for data acquisition.
Only events matching this software window restriction are copied into data buffer for transmission.
This allows suppression of events caused by unwanted particles with deviating energy deposition.

Set MDLL Test pulser	Cmd = 65
Word	**Contents**
10	Pulser On / Off (0 / 1)
11	Pulser Amplitude (0, 1, 2, 3)
12	Pulser Position (0, 1, 2)
13	0xFFFF

Sets MDLL test pulser to one of three possible positions (below left, middle, upper right) and selects one of three possible amplitudes.
On / off switches pulser hardware on / off.

Listmode Data Format

General File Format

Incoming data fom MCPD-8 modules will usually be written to disk directly in listmode format. This is the native data format arriving from ethernet. It allows an offline replay / reconstruction / processing of data once they are taken.

The listmode files may vary from setup to setup, but a general structure can be given:

File Header ASCII format variable length length given in header carries administrational data
Header Separator binary format fixed length: 4 x 16 bit words fixed signature: 0x0000, 0x5555, 0xAAAA, 0xFFFF may be used for integrity check, file reconstruction
First Data Block binary format image of data buffer header plus trailing data like described above (p 8 ff.) variable length length given in data header
Data Block Separator binary format fixed length: 4 x 16 bit words fixed signature: 0x0000, 0xFFFF, 0x5555, 0xAAAA may be used for integrity check, file reconstruction
Trailing Data Blocks
Trailing Data Separators
...
Closing signature binary format fixed length: 4 x 16 bit words fixed signature: 0xFFFF, 0xAAAA, 0x5555, 0x0000 may be used for integrity check

During data taking, only incoming event data will be written to file. Command answers will be processed but not written to file.

File Header Format

Details of the File header will vary from system to system, depending on number and contents of additional information that is to be written to the file.

In any case, the header will begin with ASCII line:

```
mesytec psd listmode data
```

followed by a line:

```
header length: nnnnn lines
```

This allows any data processing software to skip the variable ASCII header part of the listmode file without knowing about the detailed structure.

Beginning with line nnnnn+1, the binary part of the listmode file starts with the header separator.

Data Header Format

Data Header is in binary format, its structure is defined on p. 8ff.

Data Format

Each data block contains a variable number of 48 bit wide events in binary format.
The event structure is described on page 10 ff.

Bus protocol / Data transmission

Central modules (MCPD-8) and peripheral devices (MPSD-8, MSTD-16, MPSD-8+) are connected via mesytecs highspeed serial interface. A general description is given in chapter “System Layout”.

In MCPD and MCPD-2, the interface uses a bus topology while MCPD-8 offers a point-to-point connection.

Communication on the mesytec serial bus is split into two domains:
Control and Data transmission.

Control communication
(transmission of control commands and answers in dialog mode)

Control communication is used to set up gains and thresholds and to read out device data. Command/answer transmission between MCPD and MPSD/MSTD is done in a proprietary protocol which is common among all flavours of central and peripheral devices. Thus all MCPD versions are principally able to talk to all MPSD/MSTD versions so far.

Issuing a correct UDP command like described above leads to a control communication between central and peripheral device via mesytec bus.

Data transmission
(fast transmission of measurement data from peripherals to central module)

Throughout the generations of the mesytec neutron detector readout system, data transmission on the mesytec bus was modified and extended to allow for increasing communication capabilities.

There are mainly 3 versions of data transmission format:

POS (P):
Only Position or Amplitude information is transmitted, P/A has to be selected by mode command.
This is the “legacy” transmission mode used by the first generation of MCPD and MPSD / MSTD.
(In MSTD-16: always transmission of the amplitude information.)

TOF-POS (TP):
Position or Amplitude are transmitted together with an offset timestamp. P/A has to be selected by mode command.
This is the transmission mode for the intermediate MPSD-8+ modules, which are already capable generating the 100 ns precision timestamp.

TOF-POS-AMP (TPA):
Position and Amplitude are transmitted simultaneously together with the offset timestamp.
This is the transmission mode of the most recent MPSD-8+ modules.

All three transmission modes only refer to the fast data transmission on the mesytec bus!
Ethernet data format (and of course command communication) is not influenced by the type of bus communication!

Capabilities of different peripheral modules:

All peripheral modules with ID numbers < 105 are only capable of transmitting in POS mode.
Modules with ID 105 and higher may be able to transmit also in higher modes (TP, TAP) – they present a hardware register showing their transmit capabilities and a register to set the desired transmit mode.

Capabilities are coded/set in an eight bit value like follows:
(Register 0)

7	6	5	5	3	2	1	0
x	x	x	x	x	TAP	TP	P

Devices with ID 105 and higher are at least capable transmitting in TP mode (corresponding with a register value of 3), devices with simultaneus calculation of position and amplitude present a register value of 7.

The choice of the transmission mode influences the maximum continuous bus rate:
In TP mode, a max. continuous bus rate of 800 kHz is expected, in TAP mode – due to the additional amplitude data – the max. continuous bus rate is 500 kHz.

Capabilities of different MCPD-8 versions:

As there are two versions of MCPD-8 with different communication capabilities. There are also registers in MCPD-8 indicating / selecting the possible / desired communication modes. The bitpattern is like described above for the MPSD-8+:

7	6	5	5	3	2	1	0
x	x	x	x	x	TAP	TP	P

Bus communication mode setup

By default / after power up, all devices are in the maximum possible transfer mode.
After power up / reset, MCPD-8 scans all connected peripherals for their tx capabilities and sets the according tx mode.
These defaults can be overwritten by software commands:
The registers indicating and controlling the bus protocol can be read and set via MCPD-8 using the following commands

MPSD-8:
Read peripheral register (Cmd ID = 52)
Write peripheral register (Cmd ID = 53)

Read TX capabilities of connected MPSD-8:

Read Peripheral Register	**Cmd = 52**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Register number (0 for tx capabilities)
12	0xFFFF

The answer packet will look like follows:

Read Peripheral Register	**Cmd = 52**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Register number (= 0 for tx capabilities)
12	Capability bitmap
13	0xFFFF

Set desired TX mode:

Write Peripheral Register	**Cmd = 53**
Word	**Contents**
10	MPSD device number (0 … 7)
11	Register number (= 1 for tx capabilities)
12	TX mode (1 = P, 2 = TP, 4 = TAP)
13	0xFFFF

MCPD-8:
Read Register (Cmd ID = 32)
Write Register (Cmd ID = 31)

The read and write register commands for MCPD-8 are slightly different to those of MPSD-8.
In MCPD-8 it is possible to read / write a block of registers. Thus the read/write commands are extended by a length parameter.

Read Register	**Cmd = 32**
Word	**Contents**
10	Number of Registers to read (=1)
11	Register address (102 for tx capabilities)
12	Capability bitmap
13	0xFFFF

Write Register	**Cmd = 31**
Word	**Contents**
10	Number of Registers to write (=1)
11	Register address (103 for tx mode)
12	TX mode (1 = P, 2 = TP, 4 = TAP)
13	0xFFFF

Examples for bus setups

- MCPD-8 and MPSD-8 („SPODI Type“):
  MPSD-ID is 1.
  Only possible protocol is „P“ (Position / Amplitude only).
  MPSD-8 („SPODI-Type“) is in „P“ mode by default.
  MCPD-8 has to be set to mode „P“ (Value 1 in register 103)

- MCPD-8 and MSTD-16:
  MSTD-ID is 104.
  Only possible protocol is „P“ (Position / Amplitude only).
  MSTD-16 is in „P“ mode by default.
  MCPD-8 has to be set to mode „P“ (Value 1 in register 103)

- MCPD-8 and MPSD-8+ („DNS Type“):
  MPSD-ID is 103.
  Possible protocols are „P“ (Position / Amplitude only) and „TP“ (Position / Amplitude plus TOF timing).
  MPSD-8 („DNS-Type“) is in „TP“ mode by default.
  MCPD-8 has to be set to mode „TP“ (Value 2 in register 103)

- MCPD-8 and MPSD-8+ („SANS Type“):
  MPSD-ID is 105.
  Possible protocols are „P“ (Position / Amplitude only), „TP“ (Position / Amplitude plus TOF timing) and „TAP“ (Postion plus Amplitude plus TOF timing)
  MPSD-8 („DNS-Type“) is in „TAP“ mode by default.
  MCPD-8 can be left unchanged.

  Both devices have to be set to mode „TP“ (Value 2 in register 103) if for reasons of highest data rate capabilities only one of the parameters Amplitude / Position shall be transmitted.